Риддер қ. Құмаш Құрғалиев атындағы колледж мекемесінің филиалы
Филиал учреждения «Колледж имени Кумаша Нургалиева» г. Риддер

«Бекітемін»
Колледж филиалының директоры
_____ Глотова Т.С.
«___» _____20___ж.

«Утверждаю»
Директор филиала колледжа
_____Глотова Т.С.
«____»_____20___г.

«Алгоритмдеу және бағдарламалаудың негіздері» тәртібінен Әдістемелік нұсқаулар Мамандық 1304000 «Есептеу техникасы және автоматтандырылған жүйелерді бағдарламалық қамтамасыздандыру»

Практикум по дисциплине «Основы алгоритмизации и программирования» специальность 1304000 «Вычислительная техника и программное обеспечение (по видам)»

Арнайы тәртіптердің пәндік – кезеңдік комиссиясының отырысында қарастырылған және бекітуге ұсынылған

ПКК- ның төрағасы_____
№_____ хаттама
«____»_____ 20___ж

Рассмотрено на заседании предметно – цикловой комиссии спецдисциплин и рекомендовано к утверждению

Председатель ПЦК_____
от «____»_____20___г.
Протокол №_____

Риддер қ 2011 жыл
г. Риддер, 2011 г.

РЕКОМЕНДАЦИИ ПО РАБОТЕ С УЧЕБНИКОМ

В учебнике изложен весь необходимый материал для освоения программы предмета «Основы алгоритмизации и программирования» по специальности 1304000 «Вычислительная техника и программное обеспечение (по видам)», предусмотренный Государственными стандартами за первый год обучения данной дисциплине. Он содержит базовый объем знаний по изучению программирования.

При самостоятельном изучении данного курса советую сначала прочитать, понять и запомнить теоретический материал первой темы, после чего найти в конце раздела практические задания и выполнить из них все, относящиеся к данной теме. Таким же образом изучайте каждую тему. Для получения отличной оценки достаточно выполнить все задания, помеченные символами «*», «**» и «***» и пройти тестирование, приведенное в конце каждого раздела.

Если у вас возникнут затруднения в выполнении практических заданий, прочитайте еще раз внимательно теоретический материал, просмотрите внимательно материал приложений в конце учебника. Для решения сложных заданий приведены ссылки на примеры, которые помогут вам в их выполнении. Если и после этого у вас не получится решить какую-то задачу, обратитесь за помощью к автору учебника.

После изучения раздела для того, чтобы проверить свои теоретические знания, пройдите тест, вопросы к которому приведены в конце каждого раздела. В тесты включены вопросы, взятые из государственного аттестационного материала на подтверждение сертификата. Тест вы можете пройти с помощью программы тестирования в кабинете информатики или, написав ответы на вопросы на листочке. Попросите преподавателя проверить правильность прохождения теста.

Если вы без труда можете выполнить практически любую задачу, помеченную символами «*», «**» или «***», справились с тестовыми заданиями, можете считать, что раздел вами усвоен на отлично.

Для углубления знаний, формирования практических навыков программирования и реализации всех своих способностей можете решить еще и творческие задачи, помеченные четырьмя символами «****». Это уже высшая степень овладения программированием.

УСЛОВНЫЕ ОБОЗНАЧЕНИЯ

Символом «*» обозначаются задачи простого уровня.

Двумя символами «**» обозначаются задачи средней сложности.

Тремя символами «***» обозначены сложные задачи.

Четыре символа «****» обозначают, что задача носит творческий характер.

ВВЕДЕНИЕ

В настоящее время язык Паскаль является одним из самых популярных и удобных языков программирования высокого уровня. Он был создан профессором Виртом, директором Института информатики Швейцарской высшей политехнической школы, и предназначался в первую очередь для обучения учащихся практике современного программирования.

Как и любой другой язык программирования, язык Паскаль содержит набор используемых символов, служебных слов, операций и правил написания программ.

Система программирования Турбо Паскаль, разработанная американской корпорацией Borland, остается одной из самых популярных систем программирования в мире. Сегодня язык Паскаль превратился в мощную современную профессиональную систему программирования, которой по плечу любые задачи – от создания простых программ, предназначенных для решения несложных вычислительных задач, до разработки сложнейших реляционных систем управления базами данных. Появление Windows и инструментальных средств Borland Pascal with Objects и Delphi для разработки программ в среде Windows лишний раз показало, какие поистине неисчерпаемые возможности таит он в себе: и Borland Pascal, и используемый в Delphi язык Object Pascal основываются на Турбо Паскале и развивают его идеи.

Данный учебник предназначен для учащихся специальности 1304000 «Вычислительная техника и программное обеспечение (по видам)» по предмету «Основы алгоритмизации и программирования» и базируется на знаниях, умениях и навыках учащихся по предметам «Математика» и «Информатика».

Целью написания учебника по алгоритмизации является формирование у обучающихся твердых теоретических знаний и практических навыков по технологии программирования вычислительных процессов и процессов обработки различных

типов данных, созданию программ на языках программирования высокого уровня для решения различных задач.

Основные задачи, необходимые для осуществления этих целей:

- ☑ изучение методов, средств и приемов создания алгоритмов;
- ☑ изучение Турбо среды программирования, объединяющий редактор, компилятор, отладчик и другие средства, облегчающие работу программистов;
- ☑ изучение базового Паскаля, включающего простые и сложные операторы и процедуры, работу с циклами, записями, файлами.

В учебнике излагаются основные разделы рабочей программы курса алгоритмизации, предусмотренные государственными стандартами среднего профессионального образования по данной специальности. Наряду с теоретическими вопросами в нем рассматриваются многочисленные примеры выполнения практических заданий и множество задач для самостоятельного выполнения. Изложение ведется достаточно строго и доступно для понимания.

Автором учебника является преподаватель специальных дисциплин Риддерского филиала учреждения «Колледж имени К. Нургалиева» **Дранкович Алия Кадылбековна**.

1. ОСНОВЫ АЛГОРИТМИЗАЦИИ

1.1. Базовые понятия предмета алгоритмизации и программирования. Этапы подготовки и решения задач.

Любой человек ежедневно встречается с множеством работ, для выполнения которых существуют определенные правила (инструкции, предписания), объясняющие как эту работу осуществить.

Пример 1. Запишем в виде команд инструкцию по выключению компьютера в Win98.

Решение:

1. закрыть все открытые окна приложений;
2. щелкнуть по кнопке ПУСК;
3. в появившемся меню выбрать «Завершение работы»;
4. в диалоговом окне выбрать пункт «Выключить компьютер»;
5. щелкнуть по кнопке «Да».

Другими примерами предписаний могут быть план написания сочинения на уроке литературы, последовательность действий при решении квадратного уравнения на уроке математики, схема регулирования воды в водопроводе, правила проверки орфографии русского текста и т.д.

Исполнителями подобного рода предписаний могут быть как люди, так и технические устройства – автоматы, роботы, компьютеры.

Определение: Алгоритм – точное предписание последовательности действий, приводящих к получению результата. Исполнителем является человек.

Определение: Отдельные предписания – приказ алгоритма назовем командой. Алгоритм является последовательностью команд.

Определение: Процесс составления алгоритмических предписаний называется алгоритмизацией.

Слово «алгоритм» происходит от algoritmi – латинской формы написания имени великого Среднеазиатского математика

IX века аль-Хорезми, который сформулировал правила выполнения арифметических действий. Он является автором самого древнего алгоритма.

Определение: Исполнитель – это тот объект или субъект, для управления которым составлен алгоритм.

Определение: Программа – это алгоритм, записанный на языке программирования. Исполнитель – компьютер.

Определение: Программирование – различные формы организации работы с использованием программных средств.

Определение: Языком программирования называется язык описания алгоритма для исполнителя ЭВМ.

Пример 2. Написать алгоритм решения следующей задачи. Имеется два кувшина емкостью 3 и 8 литров. Как набрать из реки 7 литров воды.

Решение: Задача имеет несколько решений. Приведем одно из них:

1. наполняем маленький кувшин;
2. выливаем из маленького кувшина в большой 3 литра воды;
3. наполняем маленький кувшин;
4. выливаем из маленького кувшина в большой 3 литра воды;
5. наполняем маленький кувшин;
6. выливаем из маленького кувшина в большой 3 литра воды (теперь в маленьком – 1 литр воды, а большой наполнен);
7. освобождаем большой кувшин;
8. выливаем из маленького кувшина в большой 1 литр;
9. заполняем маленький кувшин и выливаем его содержимое в большой;
10. заполняем маленький кувшин и выливаем его содержимое в большой.

Этапы подготовки и решения задач.

Какие логические приемы мы используем для того, чтобы решить конкретную задачу? Например, сходить в магазин за продуктами, покрасить забор, решить математическую задачу, пойти в колледж.

Этапы решения задач:

1. разбираем условие;
2. обдумываем план действий, т.е. разбиваем на элементарные шаги;
3. исполняем по шагам;

Этапы решения задач на компьютере:

1. ввод текста программы в память машины;
2. запуск интерпретатора языка программирования для исполнения программы.

Построение алгоритма для решения задач из какой-либо области требует от человека тщательного анализа поставленной задачи, глубоких знаний, сложных рассуждений. На поиск алгоритмов решения некоторых научных и инженерных задач иногда уходят многие годы. Но если алгоритм создан, решение задачи по данному алгоритму не представляет больших сложностей, а требует лишь выполнения отдельных команд алгоритма в той последовательности, в какой они приведены. Это очень важная особенность алгоритма, которая позволяет исполнителю действовать *формально,* механически исполняя команды. Важным качеством алгоритма является то, что от исполнителя не требуется понимания метода решения, почему нужно выполнять предписываемые действия. Исполнитель выполняет алгоритм механически, точно следуя инструкциям-командам.

Пример 3. Рассмотрим формальность исполнения алгоритма. Возьмите циркуль и линейку и выполните предписания:

1. начертите произвольный отрезок АВ;
2. поставьте ножку циркуля в точку А;
3. установите раствор циркуля равным длине отрезка АВ;
4. проведите окружность;

5. поставьте ножку циркуля в точку B;
6. проведите окружность;
7. проведите прямую через точки пересечения окружностей;
8. отметьте точку пересечения окружностей.

Если вспомнить курс школьной геометрии и выполнить данный алгоритм, можно убедиться, что, следуя приведенным инструкциям, мы разбили отрезок AB пополам.

Отсюда следует вывод, что при выполнении алгоритма нам не понадобились ни знания геометрии, ни знания о том, что этот алгоритм делает и почему. Единственное, что потребовалось – уметь выполнить каждую команду.

Первый и второй этапы подготовки и решения задач являются творческими и могут выполняться человеком, хорошо знакомым с поставленной задачей и приемами построения алгоритма. Третий этап – исполнение – требует лишь умения формально, механически выполнять команды алгоритма.

Именно формальность исполнения алгоритма позволяет применять в качестве исполнителей машины, технические устройства.

Что касается профессии программиста, то программист пишет программы – последовательность действий, понятных компьютеру, для решения поставленной задачи, а компьютер выполняет все предписанные ему команды. Таким образом, программист должен не только знать язык программирования, но и хорошо разбираться в той области, к которой относится данная задача, а это может быть математическая, экономическая, технологическая, инженерная и др. задача, а также уметь ее решить.

1.2. Способы представления алгоритмов

Существует много способов описания, представления алгоритмов:

- ☑ в виде формулы;
- ☑ в виде таблицы;
- ☑ словесный;
- ☑ графический;
- ☑ на языке программирования.

Пример 4. В предыдущих примерах (1-3.) использовалось словесное описание. Такую задачу, как нахождение периметра прямоугольника удобно представить в виде формулы, а наблюдение за температурой каждый месяц в течение года – в виде таблицы.

Графическое описание алгоритма

Определение: Графическое описание алгоритма называется блок-схемой алгоритма. Существуют государственные стандарты описания алгоритмов блок-схемой.

Определение: Геометрические фигуры, используемые в блок-схеме, называются символами-блоками, связи между ними называются линиями потока.

Линии потока используются для указания путей перехода от фигуры к фигуре, последовательности обработки данных. Направление линий потока на блок-схеме горизонтальное и вертикальное, причем положительными считаются направления вправо и вниз, а отрицательными – влево и вверх от блока. Отрицательные направления имеют стрелки.

Различают основные и вспомогательные блоки. Основные – действия по вводу и выводу и обработке информации, вспомогательные – для пояснения блок-схемы. В приложении А приведены стандарты ЕСПД (единой системы программной документации) по построению схем алгоритмов с пояснениями.

Требования, предъявляемые к алгоритму

1. Массовость – способность решить любую конкретную задачу того типа, для решения которого он предназначен.

Пример 5. Алгоритм решения квадратного уравнения следует строить таким образом, чтобы с помощью него

решалось любое квадратное уравнение с любыми коэффициентами.

0. вводим коэффициенты;
1. находим дискриминант;
2. если дискриминант меньше нуля, то выводим сообщение, что корней нет;
3. если дискриминант равен нулю, то выводим сообщение, что корень один и вычисляем его по формуле;
4. если дискриминант больше нуля, то выводим сообщение, что уравнение имеет два корня и вычисляем эти корни.

2. Компактность – краткость, свойство минимальности инструкций.

Пример 6. Алгоритм покраски забора можно записать двумя способами. Первый вариант:

0. красим дощечку;
1. перемещаемся к следующей дощечке;
2. красим дощечку;
3. перемещаемся к следующей дощечке;
4. красим дощечку;
5. перемещаемся к следующей дощечке;
6. красим дощечку;
7. перемещаемся к следующей дощечке;

……..;

n. красим последнюю дощечку;

Второй вариант:

0. красим дощечку;
1. перемещаемся к следующей дощечке;
2. шаги 0 и 1 повторяем до тех пор, пока не закончится забор.

Очевидно, что второй вариант записи данного

алгоритма компактен, рационален, массов и состоит всего лишь из трех шагов в не зависимости от длины забора, а в первом алгоритме в случае, если дощечек в заборе 1000 штук, то шагов получится 1999.

3. Детерминированность (дискретность) – строгая определенность – однозначность предписываемых действий в каждой инструкции алгоритма. Алгоритм должен быть разбит на шаги, представляющие собой четкие, законченные действия. Переход исполнителя к следующему шагу возможен лишь после завершения предыдущего;
4. Результативность – свойство обеспечения получения результата решения задачи.

Пример 7. Алгоритм кипячения молока:

1. налить в чашку молоко;
2. поставить чашку на плиту;
3. включить плиту;
4. ждать, пока молоко закипит;
5. выключить плиту.

Некий злоумышленник изменил последовательность действий алгоритма и выдал следующую систему команд за алгоритм кипячения молока:

1. налить в чашку молоко;
2. поставить чашку на плиту;
3. ждать, пока молоко закипит;
4. включить плиту;
5. выключить плиту.

Данное предписание не может считаться алгоритмом, т.к. не приводит к ожидаемому результату.

Искусство составления эффективных алгоритмов является важнейшим показателем высокой квалификации программиста.

Схема составления алгоритма

1. Изучить постановку задачи;
2. Определить тип и формы представления исходных данных, для которых алгоритм должен быть массов. Ввести обозначения для исходных данных;

3. Определить типы и формы представления результатов, ввести для них обозначения;
4. Разработать метод решения задачи или использовать известный метод. Описать алгоритм реализации метода решения, обеспечивая свойства детерминированности, результативности, ясности инструкций и компактности описания;
5. Проверить правильность составленного алгоритма – результативность. Исправить описание алгоритма при обнаружении ошибок;
6. Провести тестирование алгоритма.

Типы алгоритмов.

1. Линейным называется алгоритм, при выполнении которого исполнитель выполняет одну команду за другой в порядке их следования.

   Пример 8. Примерами линейных алгоритмов могут быть такие как сварить суп, решить задачу по математике.

2. Разветвляющийся – алгоритм, при выполнении которого действия исполнителя определяется результатами проверки некоторых условий.

   Пример 9. а) Учащийся колледжа, встав утром с постели, рассуждает: «Если сегодня хорошая погода, то я пойду в колледж, а если нет – то не пойду». б) Девушка, собираясь на дискотеку, размышляет: "Если там будет Саша, то приглашу его на танец, если Саша пригласит Машу, то Маша об этом пожалеет, а если меня пригласит Коля, то я ему откажу".

3. Циклический – алгоритм, при исполнении которого отдельные команды или группы команд повторяются многократно.

Доказано, что любую задачу можно решить, используя

три перечисленных типа алгоритмов.

Пример 10. Посадка картофеля, покраска забора.

Итак, мы теперь знаем способы представления алгоритмов, требования к алгоритму, этапы построения алгоритма, типы алгоритмов. Рассмотрим на примерах запись различных типов алгоритмов в словесной форме и изображение их графически. При этом будем следовать этапам построения алгоритма и требованиям к алгоритму.

Пример 11. Вычислить $x = c(a - b) - ab$

Решение: Для начала необходимо внимательно прочитать условие задачи, понять ее, выделить входные и выходные данные, мысленно составить план решения.

Аргументы: a, b, c

Результат: x

Рабочие переменные: R, T

0. начало;
1. ввод a, b, c;
2. r:=a*b;
3. T:=a-b;
4. T:=T*c;
5. x:=T-R;
6. вывод x;
7. конец.

Данный алгоритм является линейным. Аргументы – это *входные* переменные, значения которых нам известны или значения которых мы сами задаем. Результаты – это *выходные* переменные, в которые будет записан результат. Рабочие переменные – это промежуточные переменные, которые нам пригодятся при решении задачи.

В начале решения задачи раздел *рабочие переменные* можно оставлять пустым и заполнять его в ходе решения задачи, когда промежуточные переменные появляются.

На первом шаге мы задаем переменным какие-либо значения. При этом мы следуем требованию массовости алгоритма. Следует обратить внимание на знак «:=», который

обозначает операцию присваивания, а не знак равенства. С помощью нее в значение переменной записывается какое-либо число.

На первом и втором шаге мы ввели новые переменные для того, чтобы разбить данный пример на простые шаги и не повторять уже выполненные действия.

На третьем шаге мы снова используем переменную T, но записываем в нее уже новое значение. При этом следует запомнить правило:

Ничто не исчезает так бесследно, как старое значение переменной!!!

Т.е. с помощью операции T:=T*c мы переменной T присвоили старое значение переменной T (оно равно (a-b)), умноженное на c. По логике, если рассматривать операцию T:=T*c как математическое выражение T=T*c, оно в корне не верно. Однако, как было ранее замечено, знак «:=» обозначает операцию присваивания, а не знак равенства. Представьте, что когда мы определяем как аргумент переменную T, в памяти компьютера выделяется пустая ячейка с именем T.

В дальнейшем представляйте переменные как ячейки памяти, которые выделяются компьютером, когда мы описываем их как аргументы, рабочие переменные или результаты, а впоследствии им могут быть присвоены другие значения, причем старые значения безвозвратно исчезают!!!

А когда мы с помощью операции (в языке программирования команда или операция – это *оператор*) присваивания задаем ей значение, в пустую ячейку записывается это значение. Если далее мы снова задаем переменной T другое значение, то старое значение бесследно исчезает, и в ячейку памяти записывается новое значение переменной.

Предпоследним шагом мы выводим x, т.е. полученное

значение переменной x мы увидим на экране компьютера.

Данный алгоритм является массовым, так как работает для любых значений входных переменных; компактным и детерминированным, так как разбит на простые действия, понятные исполнителю; результативным, так как обязательно приведет к результату. Он разработан в соответствие со схемой составления алгоритма и представлен словесно и графически в виде блок-схемы. Блок-схема алгоритма изображена на рисунке 1.1.

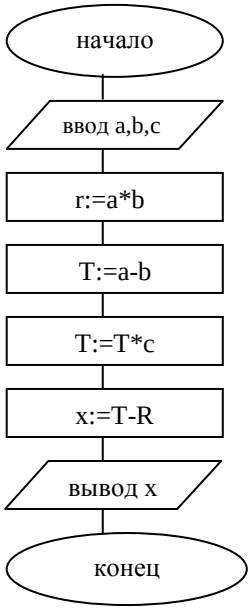


Рисунок 1.1. Блок-схема алгоритма
примера 1.11.

Пример 12. Отсортировать два числа по возрастанию.

Решение: Подумайте, как бы вы разбили на этапы решение данной задачи. Алгоритм рациональнее представить следующим образом:
Аргументы: a, b
Результаты: a, b
Рабочие переменные: R

0. начало
1. ввод a, b;
2. если a>=b, то
3. начало
4. R:=a;
5. a:=b;
6. b:=R;
7. конец;
8. вывод a, b;
9. конец.

Разберем алгоритм по шагам.

В разделы *аргументы* и *результаты* записаны одни и те же переменные, так как в начале решения задачи мы задаем им значения, решая задачу, если они не упорядочены, меняем их местами, и в конце выводим эти же переменные.

Нулевой и первый шаги понятны.

На втором шаге необходимо проверить условие a>=b и если оно верно, то поменять местами значения переменных a и b – это происходит на 3-7 шагах.

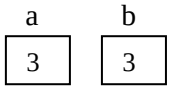
Подробнее следует объяснить замену местами значений переменных. Сначала значения переменных a и b были равны введенным числам (например 5 и 3).

a	b
5	3

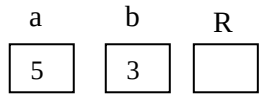
Если бы мы сделали это с помощью двух шагов: a:=b и b:=a, то получилось бы, что в ячейке памяти компьютера a старое значение удалилось и записалось новое значение, равное значению переменной b.

a	b
3	3

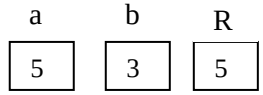
В ячейку же b записалось значение переменной a, которое теперь равно b, т.е. в ячейках a и b оказались бы одинаковые значения, равные значению переменной b.



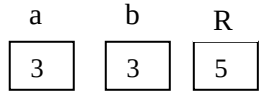
Чтобы избежать данной ошибки, вводится новая промежуточная переменная R, которая вначале пуста:



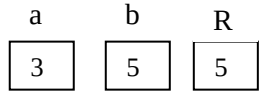
После шага 4 алгоритма значения переменных станут равными:



После шага 5:



После шага 6:



Последним шагом выводятся переменные a и b, которые уже упорядочены.

Алгоритм замены местами значений переменных необходимо запомнить наизусть!!!

Поясню, зачем в нашем алгоритме по два раза встречаются слова «начало» и «конец». На втором шаге проверяется условие и, если оно верно, должны выполниться три команды. Вот эти три команды мы и отделили словами «начало» и «конец» для того, чтобы знать, что именно эти три действия необходимо выполнить в случае выполнения условия. Шаг 8 выполняется в любом случае.

Данный алгоритм является разветвляющимся. Он соответствует требованиям массовости, компактности, детерминированности, результативности, разработан в соответствие со схемой составления алгоритма и представлен словесно и графически. Блок-схема алгоритма изображена на рисунке 7.2. В ней показано, как изображается разветвляющийся алгоритм. Видно, что если условие выполняется, то мы идем по веточке «да», где выполняются три команды, а если условие не выполняется, то уходим в конец алгоритма, где выводятся результаты и заканчивается алгоритм.

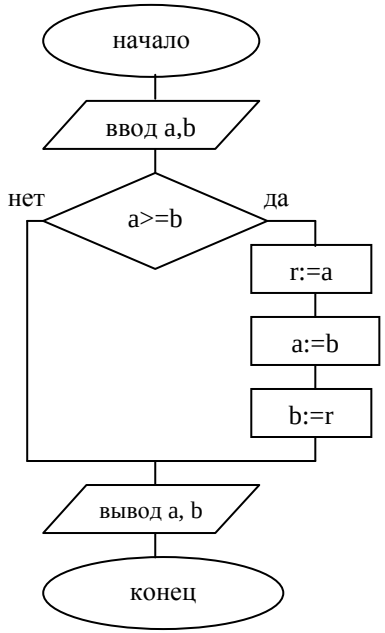


Рисунок 1.2. Блок-схема алгоритма примера 1.12

Пример 13. Даны числа. Найти их сумму.

Решение: В задании не сказано, сколько конкретно чисел необходимо сложить, однако этот параметр нам должен быть известен. Поэтому пусть количество чисел и их значения вводит пользователь (n – количество чисел, x1, x2, x3, …., xn – сами числа).

Аргументы: x1, x2, x3, …., xn, n

Результат: S

Рабочие переменные: i

Алгоритм можно представить следующим образом.

Первый вариант:

0. начало
1. ввод n;
2. ввод x1, x2, x3, …., xn;
3. S:=x1;
4. S:=S+x2
5. S:=S+x3

……..

n+1. S:=S+xn

в+2. Вывод S.

Однако в случае большого количества чисел алгоритм окажется очень длинным, что нерационально и не отвечает требованию компактности. К тому же мы не знаем заранее сколько чисел захочет ввести пользователь. Поэтому сократим его следующим образом.

Второй вариант:

0. начало
1. ввод n;
2. S:=0
3. для i:=1 до n выполнять
4. начало
5. ввод xi;
6. S:=S+xi
7. конец;
8. вывод S;
9. конец

Разберем каждый шаг алгоритма.

Сначала вводим количество чисел. Далее обнуляем сумму, т.е. в ячейку памяти S записываем ноль. Третьим шагом открываем цикл, дословно: «для i от 1 до n выполнять». Это значит, что команды, которые будут идти после слова «выполнять» (т.е. команды цикла) будут выполняться n раз, а значение переменной i будет меняться. На первом шаге цикла значение i будет равно единице, на втором – двойке, на третьем – тройке и т.д. до n.

Какие же действия будут повторяться? Их два (шаги 5, 6) – ввод следующего числа и суммирование его с предыдущей суммой.

Для наглядности вернемся к первому варианту представленного алгоритма. Шаги с 3-го по (n+1)-й очень похожи. Отличается лишь первый, в котором отсутствует старое значение переменной S. Чтобы и третий шаг внести в общую формулу S:=S+xi, сумму сначала обнуляют, и третий шаг теперь можно записать со старой суммой.

Проследим по порядку шаги цикла:

1-й шаг: i=1, вводим x1, новая сумма равна старой сумме, т.е. 0 плюс x1;

2-й шаг: i=2, вводим x2, новая сумма равна старой сумме, т.е. x1 плюс x2;

3-й шаг: i=3, вводим x3, новая сумма равна старой сумме, т.е. x1 + x2 плюс x3;

4-й шаг: i=4, вводим x4, новая сумма равна старой сумме, т.е. x1 + x2 + x3 плюс x4;

и т.д. до шага n.

В итоге сумма будет равна x1 + x2 + x3 + x4 + xn.

Ясно, что шаги цикла 5 и 6 расположены между словами «начало» и «конец» для того, чтобы знать, что именно их необходимо повторять.

И, наконец, значение переменной S, т.е. сумма чисел выводится на экран.

На рисунке 1.3 изображена блок-схема данного алгоритма. Из нее видно, что начало цикла располагается в шестиугольнике, действия, которые необходимо повторять в цикле расположены ниже и после их выполнения стрелкой показано, что цикл возвращается в начало и действия повторяются до тех пор, пока i не станет равным n. После выполнения всех шагов цикла результат выводится на экран.

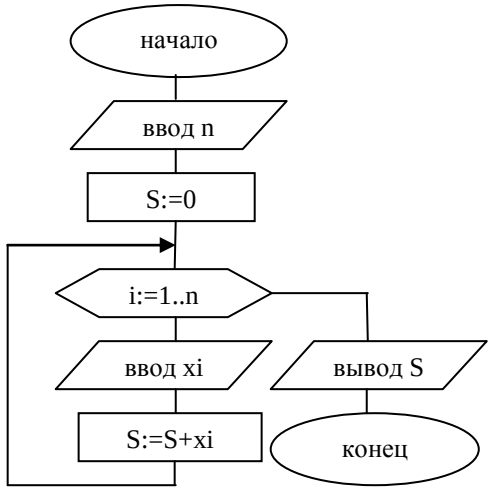


Рисунок 1.3. Блок-схема алгоритма
примера 7.13.

Данный алгоритм является циклическим. Он соответствует всем требованиям и представлен словесно и графически.

Алгоритм вычисления суммы, а именно то, что сумму необходимо обнулить до цикла, а в цикле повторить команду S:=S+xi, нужно запомнить наизусть!!!

Графическое представление различных типов алгоритмов

1. Алгоритмы условия (рисунок 1.4)

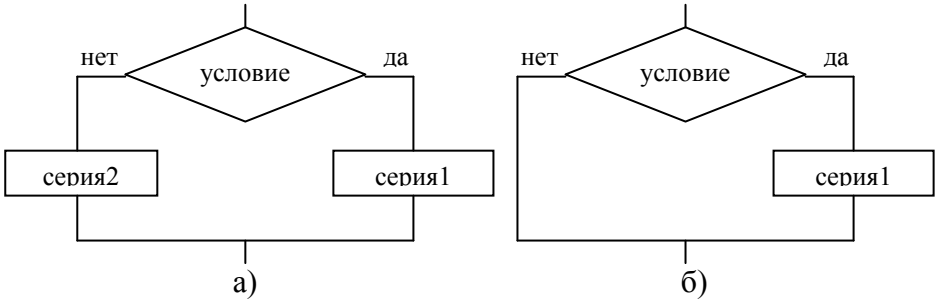


Рисунок 1.4. Блок-схемы разветвляющихся алгоритмов

Пример 14. Примером алгоритма, соответствующего схеме рисунка 1.4. (а), может быть определение по числу дней в феврале високосный год или не високосный (если в феврале 29 дней, то год високосный, а иначе не високосный).

Примером алгоритма, соответствующего схеме рисунка 1.4. (б), может быть следующий: если на улице дождь, то берем зонтик.

2. Циклические алгоритмы (рисунок 1.5)

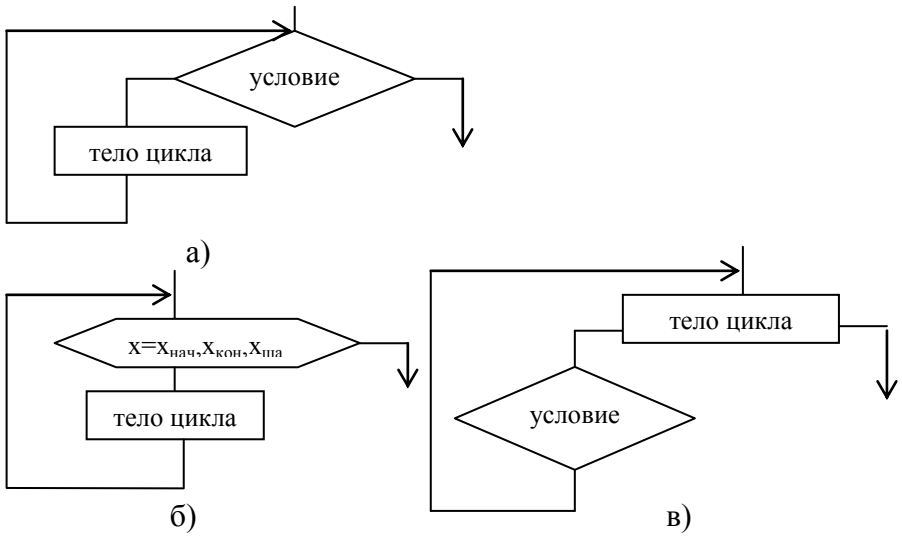


Рисунок 1.5. Блок-схемы циклических алгоритмов

Пример 15. Примером алгоритма, соответствующего схемам рисунка 1.5., может быть выкапывание картофеля: а) пока не закончится посадочная полоса, будем выкапывать лунку и перемещаться к следующей; б) от первой до последней лунки посадочной полосы будем выкапывать каждую и перемещаться к следующей; в) выкапываем лунку и перемещаемся к следующей до тех пор, пока не закончится посадочная полоса.

ПРАКТИЧЕСКИЕ ЗАДАНИЯ

1. * Приведите примеры задач (из жизни, математические и др.) и составьте для их решения алгоритмы.
2. * Приведите примеры использования формального исполнения алгоритмов на уроках математики, физики.
3. ** Имеется 9 монет, одна из которых фальшивая (более легкая). За два взвешивания на чашечных весах без гирь определить, какая монета фальшивая. Написать алгоритм решения.
4. ** Необходимо переправить на другой берег волка, козу и капусту, таким образом, чтобы все остались целы, если лодка выдерживает двух пассажиров. Написать алгоритм решения.
5. *** Придумайте еще несколько алгоритмов решения задачи из примера 2.
6. ** Составить алгоритм и блок-схему для задач:
   a) Для детского сада купили 172 альбома и 186 наборов цветных карандашей. Сколько стоит 1 альбом, если 1 набор карандашей – 60 тенге, а за всю покупку заплатили 24060 тенге.
   b) Вычислить: $y = (2 \cdot x + 3) \cdot (7 \cdot x - 5)$.
   c) Вычислить произведение нескольких чисел.
7. * Составить алгоритмы:
   a) поиска слова в словаре;
   b) заваривания чая.
8. ** По приведенному алгоритму восстановить формулу для вычисления значения Y:
   a)
      1. умножить X на X, обозначит результат R1;
      2. умножить R1 на A, обозначить результат R2;
      3. сложить R2 с B, обозначить результат R3;
      4. разделить R3 на C, считать результат значением Y.
   b)
      1. сложить X с 2, обозначить результат A1;
      2. разделить 1 на A1, обозначить результат A2;

3. сложить A2 с 5, обозначить результат A3;
4. вычесть из A2 5, обозначит результат A4;
5. разделить A4 на A3, обозначить результат A5;
6. вычесть из A5 3, считать результат значением Y.

9. * Привести примеры алгоритмов разных типов из жизни.
10. ** Выполнить словесное и графическое описание алгоритма решения задачи:
    a) Сколько мешков, вмещающих по 4 ведра картофеля, необходимо взять на уборку урожая, если всего посажено 80 лунок, в каждой в среднем 9 картофелин, а в ведро входит 30 картофелин?
    b) Программист в месяц получает 60 тысяч тенге. Из них 5 тысяч он тратит на личные расходы, жене отдает 33 тысячи, а остальные делит поровну на 4-х детей. Сколько получает в месяц на мелкие расходы ребенок программиста?
11. ** Составить алгоритм и представить его словесно и графически:
    a) вычислить: $y = \frac{7c - 4a}{2b}$;
    b) вычислить: $y = \frac{5 - (x-3)^2}{(x-3)^2 + 4}$
    c) найти модуль числа.
    d) Джерри убегает от Тома, мотая круги вокруг дерева.
12. ** От прямоугольного листа бумаги со сторонами A см и B см (рисунок 1.6) отрезали по углам квадраты со стороной X см. Затем свернули коробочку. Найти объем получившейся коробочки.

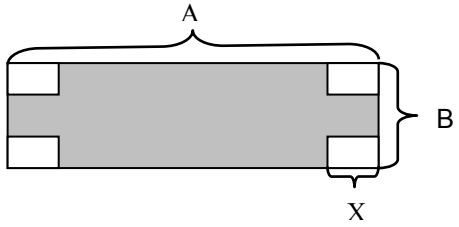


Рисунок 1.6. Иллюстрация к задаче 12

Представить алгоритм решения формулой, словесно и графически.

13. *** Сколько рулонов обоев следует купить, чтобы оклеить ими комнату длиной A метров, шириной B метров, высотой C метров. В комнате имеется два окна высотой X сантиметров и шириной Y см, а также дверь шириной M сантиметров и высотой P сантиметров. В рулоне 10 метров обоев, ширина рулона 0,5 метра. Составить алгоритм решения задачи.

Смотрите пример 12

14. ** Придумать на каждую блок-схему алгоритмов из рисунков 7.4. и 7.5. свой пример.
15. * К какому типу относятся алгоритмы:
    a) начальник дает раскомандировку подчиненному в начале рабочего дня;
    b) бомбастер разрешено продавать только лицам, младше 12 лет;
    c) если ваш рост более 175 см., у вас есть шанс стать моделью;
    d) сюжет из кинофильма «Приходите завтра»;
    e) муха пытается выбраться на улицу через стекло закрытого окна;
    f) парень на свидании с девушкой в кафе размышляет: «Если я заплачу за двоих, у меня не хватит денег на диск, а если я заплачу только за себя, не хватит на сигареты. Надо поскорее сматываться».

16. * К данным задачам нарисовать блок-схему.
    a) Иван-царевич у развилки 2-х дорог. На камне написано: "Налево пойдешь - невесту найдешь, направо пойдешь - деньги найдешь".
    b) Сварить макароны.
    c) Почистить ведро картофеля.
17. * Ниже записаны операторы присваивания, следующие в алгоритме строго друг за другом. Определить результат их выполнения.
    a) M:=0; M:=M+2; M:=3*M; M:=M-2; M:=M*M.
    b) N:=1; N:=N+1; N:=N*N; N:=N^2+1.
18. * Чему будут равны значения переменных X, Y, A, B, R после выполнения команд алгоритма:
    a) X:=3; У:=6; X:=У; У:=X.
    b) A:=2; B:=7; R:=3; R:=A; A:=B; B:=R; R:=A.
19. *** Решить квадратное уравнение. Нарисовать блок-схему.

Смотрите пример 12

20. *** Написать алгоритм и построить блок-схему для решения задачи. Вычислить значение функции:

$$y = \begin{cases} x-2, & \text{если} \quad x < 3 \\ 3x^2, & \text{если} \quad -3 \le x < 3 \\ 7-2x, & \text{если} \quad x \ge 3 \end{cases}.$$

Смотрите пример 12

21. *** Написать алгоритм и построить блок-схему для решения задачи. Вычислить:

$$S = \frac{1}{\sqrt{2}} + \frac{1}{\sqrt{20}} + \frac{1}{\sqrt{200}} + \frac{1}{2000}$$

Смотрите примеры 12 и 13

22. *** Составить алгоритм и представить его словесно и графически:
    a) Вычислить:

$$S = \frac{2}{3} + \frac{3}{4} + \frac{4}{5} + \frac{5}{6} + \frac{6}{7};$$

    b) Найти модуль числа;

c) Даны числа. Подсчитать количество положительных.

Смотрите примеры 12 и 13

ТЕСТОВЫЕ ЗАДАНИЯ ДЛЯ ПРОВЕРКИ ТЕОРЕТИЧЕСКИХ ЗНАНИЙ

1. **Алгоритм – это**
   A) четкая последовательность действий;
   B) точное предписание последовательности действий, записанное на языке, понятном человеку;
   C) последовательность действий, приводящая к результату;
   D) точное предписание последовательности действий, приводящее к результату (исполнителем является человек); *
   E) точное предписание последовательности действий, приводящее к результату (исполнителем является компьютер).
2. **Требование массовости к алгоритму заключается в том, что**
   A) масса людей может в последствии им пользоваться;
   B) у алгоритма должна быть масса решений;
   C) масса задач одного типа может быть решена с его помощью; *
   D) алгоритм применим к любому типу компьютера;
   E) алгоритм применим к любой задаче.
3. **К какому типу относятся алгоритмы 1, 2, 3 (выбрать ответы в соответственном порядке)?**
   1) Божья коровка ползет по кольцу;
   2) Девушка, собираясь на дискотеку, размышляет:"Если там будет Саша, то приглашу его на танец, если Саша пригласит Машу, то Маша об этом пожалеет, а если меня пригласит Коля, то я ему откажу.";
   3) Составление режима дня.
   A) линейный, разветвляющийся, циклический;
   B) разветвляющийся, линейный, циклический;
   C) циклический, линейный, разветвляющийся;
   D) разветвляющийся, циклический, линейный;
   E) циклический, разветвляющийся, линейный. *
4. **Линейным называется алгоритм, при котором**

A) Последовательность действий совершается одна за другой в порядке их следования; *
B) Используется оператор безусловного перехода;
C) Используется оператор GOTO;
D) Используется оператор цикла;
E) Используется оператор For.

5. Алгоритмический процесс – это

A) Процесс окончания алгоритма;
B) Способ представления алгоритма;
C) Свойства алгоритма;
D) Процесс выполнения алгоритма; *
E) Процесс перехода алгоритма.

6. Свойство алгоритма «дискретность» означает:

A) Пригодность алгоритма для решения однотипных задач;
B) Алгоритм состоит из отдельных шагов. Каждый шаг алгоритма – это некоторое законченное действие;
C) Способность алгоритма давать правильные результаты решения задач;
D) Предлагаемые действия должны быть понятными и единственно возможными; *
E) Непрерывность алгоритмического процесса.

7. Для представления алгоритма в графическом виде используют…

A) геометрические фигуры; *
B) все ответы верны;
C) формулы;
D) линии;
E) графики и функции.

8. Аргументами называются величины

A) являющиеся заголовками для алгоритма;
B) не являющиеся исходными данными для алгоритма;
C) используемые для обозначения;
D) являющиеся результатами для алгоритма;

E) являющиеся исходными данными для алгоритма. *

9. Блок-схема – это

A) Программа;
B) Команда алгоритма;
C) Вид алгоритма;
D) Способ представления алгоритма; *
E) Решение задачи.

10. С фамилией какого из древних ученых связано происхождение слова «алгоритм»?

A) Аль-Хорезми; *
B) Пифагор;
C) Евклид;
D) Аль-Коши;
E) Аль-Хайсама.

11. Переменная величина – это величина, значение которой

A) Не меняется в ходе исполнения алгоритма;
B) Постоянно меняется в ходе исполнения алгоритма;
C) Менятеся в ходе исполнения алгоритма; *
D) Является результатом для алгоритма;
E) Являются слова или тексты.

12. Алгоритм – это

A) Отдельные указания исполителю выполнить некоторые законченные действия;
B) Набор оперделений и правил для исполнителя;
C) Совокупность требований к программе;
D) Совокупность понятных и точных указаний о том, какие действия и в какой последовательности выполнять для решения любой задачи из заданного класса за конечное число шагов; *
E) Последовательность команд для ЭВМ.

13. Какой из документов является алгоритмом?

A) инструкция по получению денег в банкомате; *
B) список класса;
C) классный журнал;
D) расписание звонков;
E) правила техники безопасности.

14. Автором самого дравнего алгоритма считается…

A) Декарт;
B) Пифагор;
C) Евклид;
D) Аль-Хорезми; *
E) Герон.

2. ДАННЫЕ И ОПЕРАТОРЫ ЯЗЫКА ПРОГРАММИРОВАНИЯ

2.1. Концепция данных. Элементы языка Турбо Паскаль.

О языке программирования Паскаль

Составление программ для ЭВМ и их эксплуатация – весьма сложное и трудоемкое занятие. Оно требует больших затрат умственного труда и времени. Поэтому разработчики новых алгоритмических языков стремятся к тому, чтобы программирование было как можно более простым и доступным широкому кругу людей, работающих в различных отраслях народного хозяйства.

Именно такую задачу поставил перед собой профессор Высшего технического училища в г. Цюрихе (Швейцария) Никлаус Вирт. Предложенный им алгоритмический язык он назвал именем великого французского ученого Блеза Паскаля (1623-1662), который создал первую в мире вычислительную машину.

Язык Паскаль, утвержденный в качестве стандартного в 1979г., является наиболее совершенным по сравнению с такими универсальными языками программирования, как Алгол, Фортран, Бейсик и др.

Благодаря своей эффективности, простоте и логичности он получил широкое распространение во всем мире. Тексты программ легко проверяются на правильность, так как смысл их прост и очевиден. И наконец, язык Паскаль подходит для обучения начинающих программистов хорошему стилю программирования.

Создатель языка программирования Никлаус Вирт писал: «Опыт использования языка Паскаль убедительно продемонстрировал его достоинства: легкость программирования на нем, его пригодность с точки зрения формальных методов отладки программ, возможность его

эффективной реализации и практическую мобильность как самого транслятора с языка Паскаль, так и написанных на нем программ». Н. Вирт разработал язык Паскаль, во-первых, с целью внедрения системного подхода к программированию, созданию очень «прозрачных» программ и применению методов автоматической проверки их целостности и, во-вторых, как средство обучения новой компьютерной культуре.

Будучи педагогом, Н. Вирт сознательно попытался поднять саму дисциплину «программирование» от уровня простого ремесла до ранга сложной инженерной деятельности. Применяемый для составления программ язык, по его мнению, должен определенным образом формировать мышление программиста и помочь ему почувствовать законы программирования, красоту этого творчества. Рассмотрим возможности языка Паскаль:

1. Есть языки программирования, ориентированные на определенную область применения. Практика показала, что Паскаль в широком смысле универсальный язык.
2. Язык программирования должен, в частности, облегчать создание больших программ, разрабатываемых, возможно, несколькими программистами. Это также обеспечивает Паскаль.
3. Паскаль реализует возможность декомпозиции задач на независимые (почти независимые) подзадачи и позволяет поддерживать надежный интерфейс между ними. Возможность компоновки программ из отдельных компонентов значительно ускоряет процесс создания больших программных комплексов, повышает уровень взаимозаменяемости программных фрагментов, упрощает корректировки и повышает надежность работы комплекса.
4. Паскаль обеспечивает возможность гибкой структуризации данных за счет использования массивов, записей, множеств, файлов, динамической памяти.
5. Паскаль обеспечивает механизм надежной передачи

параметров в подпрограммы. Полезным свойством языка может выступать рекурсивность обращения к подпрограммам.

6. Поскольку Паскаль сильно типизированный язык программирования, то соблюдается строгая дисциплина использования типов.
7. В языке программирования Паскаль имеется возможность создания новых типов данных в широком диапазоне - от создания синонимов для уже существующих типов данных до возможности определения внутреннего представления объектов нового типа данных и набора операций, допустимых над объектами, имеющими этот тип. Так достигается гибкость языков программирования.
8. В Паскале также существует иерархия исполнения операций.
9. Изучая Паскаль, дорогой читатель, вы подойдете к современному уровню понимания средств общения с компьютером. Вам отчетливее будет видна перспектива развития языков программирования, средств и самой технологии создания программ. На основе развития концепции языка Паскаль уже появился язык Delphy. А сколько возможностей у языка Паскаль в совокупности с Objects 7.0.

Типы данных

Паскаль – типизированный язык программирования. Введение типов позволяет избежать, во-первых, двусмысленности, и, во-вторых, помогает при распознавании бессмысленных конструкций. Введение типов данных и переменных вносит в программу структурированность и в принципе приближает ее структуру к решаемой задаче.

1. Каждая единица языка программирования Паскаль (константа, переменная, функция или выражение) имеет только один тип. Другими словами, тип переменной определяется множеством значений, которые может принимать данная переменная.
2. Явное введение типов позволяет четко установить тип

3. каждой программной единицы, не выполняя самой программы.
4. Между каждым типом и некоторым ограниченным подмножеством операторов существует однозначная связь, т.е. для каждого типа определен свой ограниченный набор операций. Конечно, используя такие конструкции, как процедуры и функции, набор операций можно расширить.
5. Несомненно, использование типов вносит некоторую избыточность в сам язык, но это существенно помогает в обнаружении ошибок в структуре алгоритма решения той или иной задачи, реализованного в программе.

В Турбо Паскале – расширении языка Паскаль введены следующие типы, представленные на рисунке 2.1.

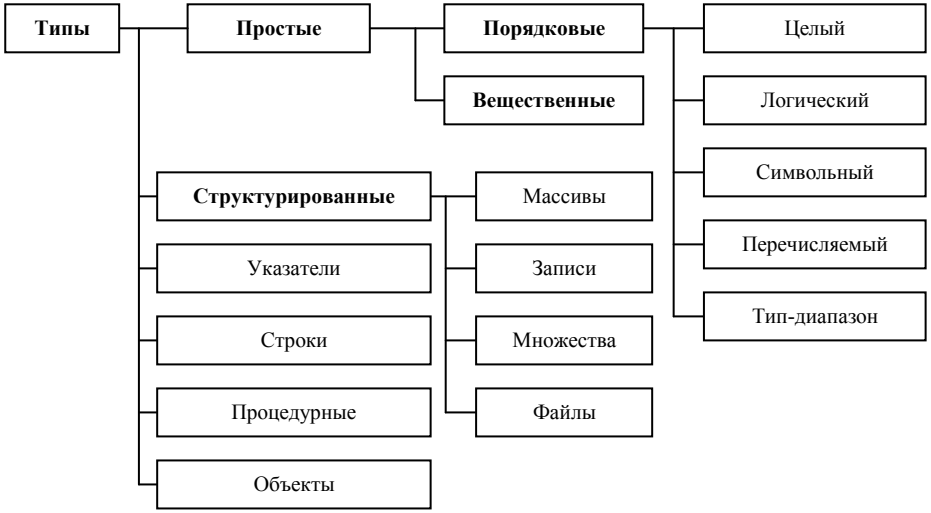


Рисунок 2.1. Типы языка Турбо Паскаль

Элементы языка

Любой естественный язык состоит из нескольких основных элементов: символов, слов, словосочетаний и предложений. Описание языка должно содержать описание

указанных структурных элементов, правил их образования и использования.

1. Алфавит языка

Символы языка – это элементарные знаки, используемые для составления любых текстов. Набор таких символов называют алфавитом языка.

Алфавит языка Turbo Pascal включает буквы, цифры, шестнадцатеричные цифры, специальные символы, пробелы и зарезервированные слова.

Буквы - это буквы латинского алфавита от a до z и от A до Z , а также знак подчеркивания “_”. В языке нет различия между заглавными и строчными буквами алфавита, если только они не входят в символьные и строковые выражения.

Цифры - арабские цифры от 0 до 9.

Каждая шестнадцатеричная цифра имеет значение от 0 до 15. Первые 10 значений обозначаются арабскими цифрами 0... 9, остальные шесть - латинскими буквами a ... f или a... f.

Специальные символы Turbo Pascal - это символы

+ - * / = , ' . : ; < > [ ] ( ) { } " @ $ #

К специальным символам относятся также следующие пары символов:

< > , < = , > = , : = , (* , *) , (. , .) , / / .

В программе эти пары символов нельзя разделять пробелами, если они используются как знаки операций отношения или ограничители комментария. Символы (. и .) могут употребляться соответственно вместо [и].

Особое место в алфавите языка занимают пробелы, к которым относятся любые символы в диапазоне кодов от 0 до 32. Эти символы рассматриваются как ограничители идентификаторов, констант, чисел, зарезервированных слов. Несколько следующих друг за другом пробелов считаются одним пробелом (последнее не относится к строковым константам).

В Turbo Pascal имеются следующие *зарезервированные (служебные) слова:*

and	**end**	**nil**	**shr**
array	**file**	**not**	**string**
asm	**for**	**object**	**then**
begin	**function**	**of**	**to**
case	**goto**	**or**	**type**
const	**if**	**packed**	**unit**
constructor	**implementation**	**procedure**	**until**
destructor	**in**	**program**	**uses**
div	**inline**	**record**	**var**
do	**interface**	**repeat**	**while**
downto	**label**	**set**	**with**
else	**mod**	**sh1**	**xor**

Зарезервированные слова не могут использоваться в качестве идентификаторов.

2. Идентификаторы языка

Идентификаторы в Turbo Pascal - это имена констант, переменных, меток, типов, объектов, классов, свойств, процедур, функций, модулей, программ и полей в записях. Идентификаторы могут иметь произвольную длину.

Идентификатор всегда начинается буквой, за которой могут следовать буквы и цифры. Напомню, что буквой считается также символ подчеркивания, поэтому идентификатор может начинаться этим символом и даже состоять только из одного или нескольких символов подчеркивания. Пробелы и специальные символы алфавита не могут входить в идентификатор.

Пример 16. Правильные идентификаторы:

a
MyProgramIsBestProgram
external
ALPHA
date_27_sep_39
_beta

Неправильные идентификаторы:

1 Program // начинается цифрой

block#l // содержит специальный символ

My Prog // содержит пробел

mod // зарезервированное слово

Вася // содержит буквы русского алфавита

3. Константы

В качестве констант в Turbo Pascal могут использоваться целые, вещественные и шестнадцатеричные числа, логические константы, символы, строки символов, конструкторы множеств и признак неопределенного указателя NIL.

Целые числа записываются со знаком или без него по обычным правилам и могут иметь значение в диапазоне от -2^{63} до $+2^{63}$ -1. Следует учесть, что если целочисленная константа выходит за указанные границы, компилятор дает сообщение об ошибке. Такие константы должны записываться с десятичной точкой, т. е. определяться как вещественные числа.

Вещественные числа записываются со знаком или без него с использованием десятичной точки и/или экспоненциальной части.

Экспоненциальная часть начинается символом е или е, за которым могут следовать знаки “+” или “-” и десятичный порядок. Символ е означает десятичный порядок и имеет смысл “умножить на 10 в степени”.

Пример 17.

3.14E5 - 3,14 умножить на 10 в степени 5;

-17e-2 – минус 17 умножить на 10 в степени минус 2.

Если в записи вещественного числа присутствует десятичная точка, перед точкой и за ней должно быть хотя бы по одной цифре. Если используется символ экспоненциальной части е, за ним должна следовать хотя бы одна цифра десятичного порядка.

Шестнадцатеричное число состоит из шестнадцатеричных цифр, которым предшествует знак доллара

$ (код символа 36) (Здесь речь идет об американском стандартном коде – кодировочной таблице, с помощью которой кодируется информация в компьютере. Данная таблица приведена в приложении В). Диапазон шестнадцатеричных чисел - от $ffffffffffffffff до $7FFFFFFFFFFFFFFF.

Логическая константа - это либо слово false (ложь), либо слово true (истина).

Символьная константа - это любой символ ПК, заключенный в апострофы:

'z' - символ “z”;

'Ф' - символ “Ф”.

Если необходимо записать собственно символ апострофа, он удваивается:

'' - символ “'” (апостроф).

Допускается использование записи символа путем указания его внутреннего кода, которому предшествует символ # (код 35).

Пример 18. (см. приложение В):

#97 - символ “a”;
#90 - символ “Z”;
#39 - символ “'”;
#13 - символ “CR”.

Строковая константа - любая последовательность символов (кроме символа CR - возврат каретки), заключенная в апострофы. Если в строке нужно указать сам символ апострофа, он удваивается.

Пример 19.

'Это - строка символов';
'That''s string'.

Строка символов может быть пустой, т. е. не иметь никаких символов в обрамляющих ее апострофах. Строку можно составлять из кодов нужных символов с предшествующими каждому коду символами #.

Пример 20. строка #83#121#109#98#111#108 эквивалентна строке 'Symbol'.

Наконец, в строке можно чередовать части, записанные в обрамляющих апострофах, с частями, записанными кодами. Таким способом можно вставлять в строки любые управляющие символы, в том числе и символ CR (код 13).

Пример 21.

#7'Ошибка !'#13'Нажмите любую клавишу ...'#7 .

Конструктор множества - список элементов множества, обрамленный квадратными скобками.

Пример 22.

[1,2,4..7,12]
[blue, red]
[ ]
[true]

В Turbo Pascal разрешается в объявлении констант использовать произвольные выражения, операндами которых могут быть ранее объявленные нетипизированные константы, имена типов и объектов, а также следующие функции от них:

abs	lo	ptr	swap
chr	odd	round	trunc
hi	ord	sizeof	
length	pred	succ	

Пример2.8:

```
const
MaxReal = Maxint div SizeOf(real) ;
NumChars = ord('Z') - ord('a') + 1;
Ln10 = 2.302585092994;
Ln10R = 1 / Ln10;
```

4. Выражения

Основными элементами, из которых конструируется исполняемая часть программы, являются константы,

переменные и обращения к функциям. Каждый из этих элементов характеризуется своим значением и принадлежит к какому-либо типу данных. С помощью знаков операций и скобок из них можно составлять выражения, которые фактически представляют собой правила получения новых значений.

Частным случаем выражения может быть просто одиночный элемент, т. е. константа, переменная или обращение к функции. Значение такого выражения имеет, естественно, тот же тип, что и сам элемент. В более общем случае выражение состоит из нескольких элементов (операндов) и знаков операций, а тип его значения определяется типом операндов и видом примененных к ним операций.

Пример 23. Примеры выражений

```
y
21 * (a + b) * c
sin(t)
a > 2
not Flag and (a = b)
NIL
[1, 3..7] * setl
```

5. Операции

В Turbo Pascal определены следующие операции:

унарные	**not, @**;
мультипликативные	***, /, div, mod, and, shi, shr;***
аддитивные	+, -, **or, xor**;
отношения	=, <>, <, >, <=, >=, **in**.

Приоритет операций определяется расставленными скобками (как на математике) и убывает в указанном порядке, т. е. наивысшим приоритетом обладают унарные операции, низшим - операции отношения. Порядок выполнения нескольких операций равного приоритета устанавливается компилятором из условия оптимизации кода программы и не обязательно слева

направо. При исчислении логических выражений операции равного приоритета всегда вычисляются слева направо, причем будут вычисляться все или только достаточные операции в зависимости от установленного в среде Turbo Pascal переключателя Options | Compiler | Complete Boolean eval: при установленном переключателе вычисляются все операции отношения, при неустановленном - только те, что необходимы для однозначного определения результата исчисления. Правила использования операций приводятся в таблице 2.1.

Таблица 2.1. Правила использования операций

Опера-ция	**Действие**	**Тип операндов**	**Тип результата**
not	Отрицание	Логический	Логический
not	То же	Любой целый	Тип операнда
@	Адрес	Любой	Указатель
*	Умножение	Любой целый	Наименьший целый
*	Тоже	Любой вещественный	Extended
*	Пересечение множеств	Множественный	Множественный
/	Деление	Любой вещественный	Extended
div	Целочисленное деление	Любой целый	Наименьший целый
mod	Остаток от деления	То же	То же
and	Логическое И	Логический	Логический
and	То же	Любой целый	Наименьший целый
shl	Левый сдвиг	То же	То же
shr	Правый сдвиг	То же	То же
+	Сложение	То же	Тоже
+	Тоже	Любой вещественный	Extended
+	Объединение множеств	Множественный	Множественный
+	Сцепление строк	Строковый	Строковый
-	Вычитание	Любой целый	Наименьший целый
-	То же	Любой вещественный	Extended
or	Логическое или	Логический	Логический
or	То же	Любой целый	Наименьший целый
=	Равно	Любой простой или строковый	Логический
<>	Не равно	То же	То же
<	Меньше	Логический	Логический
<=	Меньше или равно	То же	То же
>	Больше	То же	То же
>=	Больше или равно	То же	То же

В Turbo Pascal определены следующие логические операции:

not - логическое НЕ;

and - логическое И;

or - логическое ИЛИ;

xor - исключительное ИЛИ.

Логические операции применимы к операндам целого и логического типов. Если операнды - целые числа, то результат логической операции есть тоже целое число, биты которого (двоичные разряды) формируются из битов операндов по правилам, указанным в таблице 2.2.

Таблица 2.2. Логические операции над данными целого типа

Логические операции над данными целого типа (поразрядно)					
Операнд 1	**Операнд 2**	**not**	**and**	**or**	**xor**
1	-	0	-	-	-
0	-	1	-	-	-
0	0	-	0	0	0
0	1	-	0	1	1
1	0	-	0	1	1
1	1	-	1	1	0

К логическим же в Turbo Pascal обычно относятся и две сдвиговые операции над целыми числами:

i shl j - сдвиг содержимого / на j разрядов влево; освободившиеся младшие разряды заполняются нулями;

i shr j - сдвиг содержимого i на j разрядов вправо; освободившиеся старшие разряды заполняются нулями.

В этих операциях i и y - выражения любого целого типа.

Логические операции над логическими данными дают результат логического типа по правилам, указанным в таблице 2.3.

Таблица 2.3 Логические операции над данными логического типа

Логические операции над данными логического типа (поразрядно)					
Операнд 1	**Операнд 2**	**not**	**and**	**or**	**xor**
True	-	False	-	-	-
False	-	True	-	-	-
False	False	-	False	False	False
False	True	-	False	True	True
True	False	-	False	True	True
True	True	-	True	True	False

Операция отношения in применяется к двум операндам. Первым (левым) операндом должно быть выражение любого порядкового типа, вторым - множество, состоящее из элементов того же типа, или идентификатор множественного типа. Результат операции будет True, если левый операнд принадлежит множеству.

Структура программы

Структура любой программы должна быть такой:

program <имя>;
    {Раздел описаний}
begin
    {Раздел операторов}
end.

Заголовок содержит служебное слово program, имя программы, задаваемое пользователем-программистом.

Раздел описаний предназначен для объявления всех встречающихся в программе данных и их характеристик.

Этот раздел, в свою очередь, содержит следующие разделы:

program <имя>;
    type {Раздел типов}
    label {Раздел меток}
    const {Раздел констант}
    var {Раздел переменных}

```
function      {Раздел функций}
procedure     {Раздел процедур}
begin
        <оператор 1>;
        <оператор 2>;
        <оператор 3>;
        ….
        <оператор n-1>;
        <оператор n>
end.
```

Обратите внимание на то, что в языке Турбо Паскаль операторы разделяются точкой с запятой. Желательно каждый оператор записывать с новой строки строго под предыдущим оператором. Зарезервированные слова **begin** и **end** являются операторными скобками и все, что находится между ними является одним составным оператором, включающим в себя простые операторы.

Пример 24. Вычисление площади и средней линии трапеции.

Решение: В соответствии с этапами подготовки и решения задач, прочитаем внимательно условие задачи. Вспомним формулы, по которым вычисляется площадь и средняя линия трапеции (рисунок 2.2).

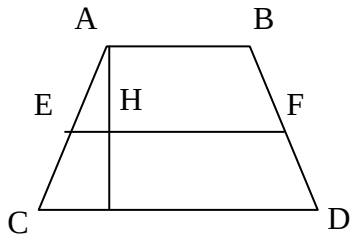


Рисунок 2.2. Иллюстрация к примеру 24

$$EF = \frac{AB + CD}{2} \qquad S = \frac{AB + CD}{2} \cdot H = EF \cdot H$$

Выделим аргументы, результаты и рабочие переменные. Ясно, что для вычислений нам необходимо знать длины верхнего и нижнего оснований (AB и CD) и высоту (H) – это и будут входные данные или аргументы. Результатами будут являться длина средней линии (EF) и площадь. Рабочие переменные пока не записываем – возможно они появятся в ходе решения задачи. Все переменные (аргументы, результаты, рабочие переменные) в Турбо Паскале записываются все вместе в разделе описаний.

```
program Sq;
var
    AB: real; {длина верхнего основания}
    CD: real; {длина нижнего основания}
    EF: real; {средняя линия}
    H: real; {высота трапеции}
    S: real; {площадь трапеции}
begin
    {1}writeln (‘Добрый день’);  {На экране появится «Добрый день»}
    {2}writeln (‘Введите значения AB, CD, H: ’); {Надпись появится на экране}
    {3}read (AB, CD, H); {Пользователь вводит в строку 3 числа. Их значения записываются, соответственно, в переменные AB, CD, H}
    {4}EF:=(AB+CD)/2;
    {5}S:=EF*H; {Расчет S}
    {6}writeln; {Пропуск строки}
    {7} write (‘Средняя линия трапеции = ‘, EF:7:2); {На экране появляется надпись и значение переменной EF}
    {8} write (‘Площадь трапеции = ‘, S:7:2); {На экране появляется надпись и значение переменной S}
    {9}readln; {Задержка экрана}
```

end.

В данном примере программа названа Sq (*двух одинаковых идентификаторов не должно быть в программе*).

Раздел описаний состоит только из раздела переменных. Всем переменным присвоен тип real – вещественный. Здесь следует заметить, что можно использовать более короткую запись. Если несколько переменных имеют один тип, их можно разделить запятой, а далее после двоеточия указать тип:

```
program Sq;
var
    AB, CD, EF, H, S: real;
```

Как мы уже договорились, представляем все переменные как пустые ячейки памяти с именами и, чтобы производить над их значениями какие-либо действия, нужно записать в ячейки эти значения. Т.е. для начала нужно задать входным данным (аргументам) значения. Пусть их вводит пользователь на свое усмотрение (таким образом, выполняется требование массовости к алгоритму).

Между словами begin и end располагаются операторы. При исполнении программы операторы как и команды алгоритма следуют строго друг за другом. Обратите внимание, что фигурными скобками в Турбо Паскале отделяются комментарии, которые компилятор игнорирует и не исполняет.

Оператор write (writeln) предназначен для вывода информации на экран в строку. При этом все, что расположено в следующих за ним скобках, выведется на экран. Курсор останется в конце текущей строки, если это оператор write, и перейдет на новую строку, если это writeln. Строковые данные (текст, комментарии, пояснения и т.п.), которые необходимо вывести, записываются в апострофах, значения переменных – без апострофов. Если необходимо вывести несколько данных (переменных, переменные и текст, несколько строковых данных и т.п.), разделителем между ними служит запятая. Например, в операторе {7} в строку выведется текст «Средняя линия

трапеции =», а затем уже рассчитанное значение средней линии. Курсор останется на текущей строке. Оператор writeln без следующих за ним скобок оставляет на экране пустую строку.

Запись EF:7:2 означает, что значение средней линии выведется в следующем формате: семь знаков (цифр) отводится на запись всего числа, из них 2 знака на дробную часть (попробуйте вывести значение средней линии без формата, т.е. просто EF, определите различие).

Оператор read (readln) служит для ввода данных. При этом read ждет от пользователя ввода значений перечисленных в скобках, идущих за данным оператором, переменных в строку (разделителем является пробел, кроме ввода символьных данных, которые вводятся подряд без пробела, так как пробел считается символом) и оставляет курсор на текущей строке. При этом число значений должно быть равным числу переменных в скобках и первое введенное значение записывается в первую переменную, второе – во вторую и т.д. Оператор Readln производит те же самые действия, за исключением того, что каждое значение переменной вводится с новой строки, т.е. разделителем значений служит Enter, и курсор переходит на другую строку.

Оператор ввода, аналогично оператору присваивания, записывает в ячейку памяти значение. Только в операторе ввода это значение вводит пользователь, а в операторе присваивания данное значение присваивается прямо в программе или рассчитывается. Если в ячейке уже находилось какое-либо значение – оно стирается и на его месте появляется новое.

Таким образом, третий оператор программы ждет, пока пользователь не введет три значения переменных AB, CD, H в строку через пробел. Эти значения запишутся в данные переменные. Теперь эти значения можно использовать в расчетах.

Операторы {3} и {4} запишут в переменные, обозначающие среднюю линию и площадь вычисленные по формулам значения.

Запустите Турбо Паскаль, наберите текст программы из примера так же как указано в примере. Чтобы исполнить программу, нажмите CTRL+F9.

2.2. Знакомство со средой программирования Турбо Паскаль

Чтобы установить Турбо Паскаль, создайте в папке Program Files папку TP7. Когда программа установки запросит, в какой каталог нужно установить TP7, выберите в дереве каталогов созданную папку.

Чтобы запустить среду программирования запустите файл C:\Program Files\TP7\Bin\Turbo.exe{\Tpx.exe}. При этом Turbo.exe работает в режиме MSDOS, а Tpx.exe – в режиме Windows, что значительно удобнее.

Чтобы выйти из TP7, выполните команду меню File-exit или нажмите клавиши Alt+x

Открыть свою программу –

Сохранять свои программы лучше в папке со своим именем на диске C. Если вы не указали, в какую папку необходимо сохранить программу, она сохраняется в каталог C:\Program Files\TP7\Bin. Чтобы сохранить свою программу, выполните команду File-save – выберите каталог, напишите имя программы на латинском языке – ok.

Рассмотрим окно среды программирования Турбо Паскаль (рисунок 2.3.) Строка меню – это кодовые слова, предназначенные для диалогового выбора продолжения работы. Окно редактора программы служит для обмена информацией между программистом и средой (текст программы набирается в окне редактора, окно программы – для результата прогона программы, справочное окно – для справочных сообщений, диалоговое окно – по мере надобности).

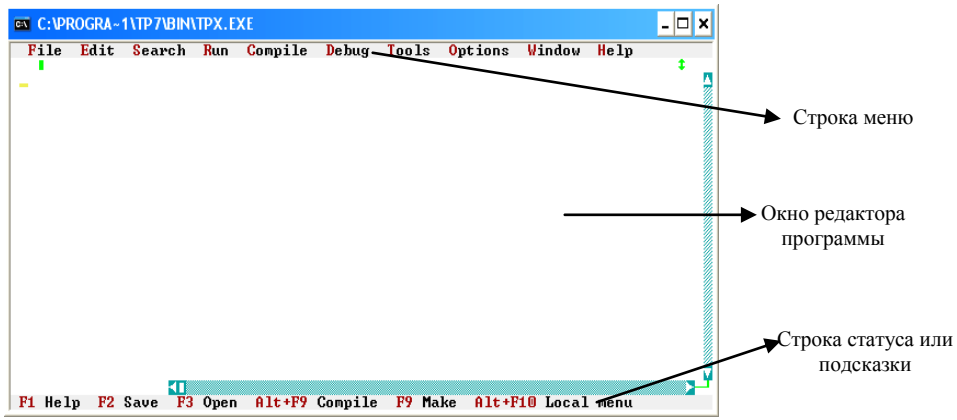


Рисунок 2.3. Окно среды программирования Турбо Паскаль

В среде одновременно могут быть открыты 9 окон. F6 – переключение между окнами.

[▄] – удаление окна с экрана, [↑] – для распахивания окна на весь экран, [⇳] – для возврата в прежние размеры. F5- распахивает окно на весь экран, Alt+F3 – закрывает окно, удаляет его с экрана (перед закрытием спрашивает: «Сохранить»?).

Ctrl+F5 – положение и размеры окна (изменяется цвет и линии рамки). Далее нажимаем клавишами ↑↓→←, с помощью которых окно перемещается по экрану, Shift+↑↓→← - изменяем размеры, затем Enter. Те же самые действия можно выполнить с мышью. Для этого подводим к левому верхнему углу и жмем ЛКМ (левую кнопку мыши) и, не отпуская, тянем. Для изменения размеров – правый угол + ЛКМ.

◆►▲▼ - вертикальная и горизонтальная полосы прокрутки, если текст не входит в размеры окна.

Вход в главное меню – F10, возврат – Esc.

В кодовом слове меню выделена буква – это значит, что при нажатии клавиш Alt+буква мы можем выполнить данную команду меню с клавиатуры.

Если рядом с кодовым словом меню имеется знак **…** - значит при нажатии на него откроется диалоговое окно, а если знак ‣ - откроется подменю.

Перемещение по опциям диалогового окна – Tab и ↑↓→←. Закрыть диалоговое окно – Esc, F1 – вызов справки.

Функциональные клавиши

Для управления средой TP7 служат функциональные клавиши F1, F2, …, F12, клавиши Alt (Alternative – дополнительный), ctrl – (control – управляющий) и shift – (shift – сдвиговый).

F2 – записать редактируемый текст в дисковый файл. Сохранить.

F3 – прочитать текст из файла в окно редактора. Открыть.

F4 – используется в отладочном режиме: начать или продолжить исполнение программы и остановиться перед исполнением той ее строки, на которой стоит курсор

F5 – распахнуть активное окно на весь экран

F6 – сделать активным следующее окно

F7 – используется в отладочном режиме: выполнить следующую строку программы: если в строке есть обращение к процедуре (функции), войти в эту процедуру и остановиться перед исполнением первого ее оператора

F8 – используется в отладочном режиме: выполнить следующую строку программы; если в строке есть обращение к процедуре (функции), исполнить ее и не прослеживать ее работу

F9 – компилировать программу, но не выполнять ее

Ctrl+F9 – выполнить прогон программы; компилировать программу, находящуюся в редакторе, загрузить ее в оперативную память и выполнить, после чего вернуться в среду TP7

Alt+F5 – сменить окно редактора на окно вывода результатов работы (прогона) программы.

Alt+Enter – свернуть окно или развернуть

Очищать окно перед исполнением программы:

```
program la;
uses crt;
….
begin
  clrscr;
```

Часто используемые команды

Page Up – на страницу вверх;
Page Down – на страницу вниз;
Home – в начало текущей строки;
End – в конец текущей строки;
Ctrl-Page Up – в начало текста;
Ctrl-Page Down – в конец текста;
Backspace – стирает символ слева;
Delete – стирает символ над курсором;
Enter – встав новую строку, разрезая стар;
Shift +↑↓→← выделение по буквам;
Ctl +↑↓→← - перемещение по словам;
Shift+Delete – вырезать;
Ctrl+Insert – скопировать;
Shift+Insert – вставить;
Ctrl+Delete – удалить;
Ctrl +Y – стирает строку с курсором.

Система меню

File (файл) – действия с файлами и выход
Edit (редактирование) - редактирование
Search (искать) – поиск текста, процедуры
Run (работа) – прогон программы
Compile (компилировать) – компиляция
Debug (отладка) – отладка программы
Tools (инструменты) – вспомогательные программы
Options (варианты) – установка параметров.
Window (окно) – работа с окнами
Help (помощь) – справка

В меню **Edit: cut** – вырезать, **copy** – вырезать, **paste** – вставить, **clear** – удалить.

2.3. Простые типы данных и их обработка

Простые типы

К простым типам относятся порядковые и вещественные типы (рисунок 2.1).

Порядковые типы отличаются тем, что каждый из них имеет конечное число возможных значений. Эти значения можно определенным образом упорядочить (отсюда и название типов) и, следовательно, с каждым из них можно сопоставить некоторое целое число - порядковый номер значения.

Вещественные типы, строго говоря; тоже имеют конечное число значений, которое определяется форматом внутреннего представления вещественного числа. Однако количество возможных значений вещественных типов настолько велико, что сопоставить с каждым из них целое число (его номер) не представляется возможным.

Порядковые типы

К порядковым типам относятся целые, логический, символьный, перечисляемый и тип-диапазон. К любому из них применима функция ORD(X), которая возвращает порядковый номер значения выражения X. Для целых типов функция ORD(X) возвращает само значение X, т.е. ORD(X)=X для X, принадлежащего любому целому типу. Применение ORD(X) к логическому, символьному и перечисляемому типам дает положительное целое число в диапазоне от 0 до 1 (логический тип), от 0 до 255 (символьный), от 0 до 65535 (перечисляемый). Тип-диапазон сохраняет все свойства базового порядкового типа, поэтому результат применения к нему функции ORD(X) зависит от свойств этого типа.

К порядковым типам можно также применять функции:

PRED (X) - возвращает предыдущее значение порядкового типа (значение, которое соответствует порядковому номеру ORD(X)- 1), т.е.

ORD(PRED(X))=ORD(X) - 1;

SUCC(X) - возвращает следующее значение порядкового типа, которое соответствует порядковому номеру ORD(X) + 1, т.е.

ORD(SUCC(X))=ORD(X) + 1.

Пример 25. Если в программе определена переменная

```
var
    c:Char;
begin
    c:='5';
end.
```

то функция PRED(C) вернет значение '4', а функция SUCC(C) - значение '6'.

Если представить себе любой порядковый тип как упорядоченное множество значений, возрастающих слева направо и занимающих на числовой оси некоторый отрезок, то функция PRED(X) не определена для левого, а SUCC(X) - для правого конца этого отрезка.

Целые типы

Диапазон возможных значений целых типов зависит от внутреннего представления, которое может занимать один, два или четыре байта. В таблице 2.4. приводится название целых типов, длина их внутреннего представления в байтах и диапазон возможных значений.

Таблица 2.4. Целые типы

Название	Длина, байт	Диапазон значений
1	**2**	**3**
Byte	1	0 ... 255
ShortInt	1	-128 ... +127

1	2	3
Word	2	0 ... 65535
Integer	2	-32768 ... +32767
LongInt	4	-2 147 483 648 ... +2 147 483 647

При использовании процедур и функций с целочисленными параметрами следует руководствоваться «вложенностью» типов, т.е. везде, где может использоваться WORD, допускается использование BYTE (но не наоборот), в LONGINT «входит» INTEGER, который, в свою очередь, включает в себя SHORTINT.

Перечень процедур и функций, применимых к целочисленным типам, приведен в таблице 2.5. Буквами b, s, w, i, l обозначены выражения соответственно типа BYTE, SHORTINT, WORD, INTEGER, LONGINT, x – выражение любого из этих типов. Буквы vb, vs, vw, vi, vl обозначают переменные соответствующих типов. В квадратных скобках указывается необязательный параметр.

Таблица 2.5. Стандартные процедуры и функции, применимые к целым типам

Обращение	Тип результата	Действие
1	2	3
abs(x)	x	Возвращает модуль x
chr(b)	Char	Возвращает символ по его коду
dec (vx[, i])	–	Уменьшает значение vx на i, а при отсутствии i - на 1
inc (vx[, i])	–	Увеличивает значение vx на i, а при отсутствии i - на 1
odd(l)	Boolean	Возвращает True, если аргумент - нечетное число
Random(w)	Как у параметра	Возвращает псевдослучайное число, равномерно распределенное в диапазоне 0..(w-1)
sqr(x)	x	Возвращает квадрат аргумента
sqrt(x)	Действительный	Возвращает корень квадратный

1	2	3
div	Целый	Целочисленное деление. Только для целых.
mod	Целый	Остаток от деления. Только для целых.

При действиях с целыми числами тип результата будет соответствовать типу операндов, а если операнды относятся к различным целым типам, - типу того операнда, который имеет максимальную мощность (максимальный диапазон значений). Возможное переполнение результата никак не контролируется, что может привести к недоразумениям.

Пример 26. Рассмотрим на примерах операторов Турбо Паскаль использование перечисленных в таблице 2.5. процедур и функций:

{1}x:=abs(-5) {x – переменная любого целого типа, ее значение после выполнения оператора {1} – число 5};

{2}s:=chr(118) {s – переменная типа char (символьный тип), ее значение после выполнения оператора {2} – символ v (см. приложение Б)};

{3}dec (x, 3){x – переменная из первого оператора, ее значение после выполнения оператора {3} – число 2, так как в первом операторе ее значение было 5};

{4}inc (x) {x – переменная из первого оператора, ее значение после выполнения оператора {4} – число 3, так как в третьем операторе ее значение стало равным 2};

{5}d:=odd(x) {x – переменная из первого оператора, d-переменная логического типа, ее значение после выполнения оператора {5} - true}

{6}y:=x div 2 {x – переменная из первого оператора, y – переменная целого типа, ее значение после выполнения оператора {6} – число 1}

{7}y:=(x+2) mod 2 {x – переменная из первого оператора, y – переменная из шестого оператора, ее значение после выполнения оператора {7} – число 1}

Рассмотрим подробнее работу функции random на примерах.

Пример 27. Пусть x – переменная целого типа, y – переменная вещественного типа, n – целое число.

1. Оператор random без параметров дает случайное
2. вещественное число в диапазоне от 0 до 1, не включая единицу, т.е. после выполнения оператора

y:=random

значение вещественной переменной y будет равно какому-либо вещественному числу, сгенерированному компьютером, из отрезка [0, 1);

3. После выполнения следующего оператора

x:=random(1)

значение переменной x будет равно какому-либо целому числу, сгенерированному компьютером, из отрезка [0, 1), а так как на отрезке [0, 1) имеется только одно целое число 0, то x в любом случае будет равен нулю;

4. После выполнения следующего оператора

y:=n*random

значение вещественной переменной y будет равно какому-либо вещественному числу, сгенерированному компьютером, из отрезка [0, n);

5. После выполнения следующего оператора

x:=random(n)

значение переменной x будет равно какому-либо целому числу, сгенерированному компьютером, из отрезка [0, n) т.е. [0, n-1];

6. После выполнения оператора

y:=5*random

значение переменной y будет равно какому-либо вещественному числу, сгенерированному компьютером, из отрезка [0, 5);

7. После выполнения следующего оператора

x:=random(5)

значение переменной x будет равно какому-либо целому числу, сгенерированному компьютером, из отрезка [0, 4];

8. Чтобы сгенерировать вещественное число в диапазоне от 2 до 7 используем оператор:

y:=5*random +2

9. Чтобы сгенерировать целое число в диапазоне от 2 до 7 используем оператор:

x:=random(6)+2

10. Чтобы сгенерировать вещественное число в диапазоне от -2 до 7 используем оператор:

y:=9*random -2

11. Чтобы сгенерировать целое число в диапазоне от -2 до 7 используем оператор:

x:=random(10)-2

Пример 28. Пусть переменная x имеет значение 13, а переменная y – значение 5, тогда:

x mod y = 3 – это остаток от деления 13 на 5;
x div y = 2 – это целая часть частного 13 и 5.

Логические типы

Логический тип (boolean) определяет те данные, которые могут принимать логические значения true (истина) и false (ложь). Для них справедливы правила:

ord (false)=0;
ord (true)=1;
false<true;
succ (false)=true;
pred (true)=false.

К булевским операндам применимы логические операции not, and, or, xor

В Турбо Паскале введены еще разновидности логического типа – это bytebool – 1B, wordbool – 2B, longbool – 4B. В таблице 2.6. приведены логические операции над данными логического типа.

Таблица 2.6. Логические операции над данными типа Boolean

Операнд1	Операнд2	Not	And	Or	Xor
True	-	False			
False	-	True			
False	False	-	False	False	False
False	True	-	False	True	True
True	False	-	False	True	True
True	True	-	True	True	false

Символьный тип

Значением переменной символьного типа является множество всех символов ПК, т.е. символьный тип (char) определяет упорядоченную совокупность символов, допустимых в данной ЭВМ. Значение символьной переменной или константы – это один символ из допустимого набора. Каждому символу приписывается целое число в диапазоне 0..255. Это число служит кодом внутреннего представления символа, его возвращает функция ORD.

Для кодировки используется код ASCII (American Standard Code for Information Interchange – американский стандартный код для обмена информацией). Это семибитный код, т.е. с его помощью можно закодировать лишь 128 символов в диапазоне от 0 до 127. В то же время в 8-битном байте, отведенном для хранения символов в Турбо Паскале, можно закодировать в два раза больше символов в диапазоне от 0 до 255. Первая половина символов ПК с кодами 0..127 соответствует стандарту ASCII. Вторая половина символов с кодами 128..255 не ограничена жесткими рамками стандарта и может меняться на ПК разных типов. На современных компьютерах, купленных в СНГ вторая половина символов включает буквы русского алфавита и некоторые специальные символы (см. приложение Б).

Символы с кодами 0..31 относятся к служебным кодам. Если эти коды используются в символьном тексте программы, они считаются пробелами. При использовании их в операциях

ввода-вывода они могут иметь самостоятельные значения, приведенные в таблице 2.7.

Таблица 2.7. Некоторые служебные коды

Символ	Код	Значение
BEL	7	Звонок; вывод на экран этого символа сопровождается звуковым сигналом
HT	9	Горизонтальная табуляция; при выводе его на экран смещает курсор в позицию, кратную 8, плюс 1 (9, 17, 25 и т.д.)
LF	10	Перевод строки; при выводе его на экран все последующие символы будут выводиться, начиная с той же позиции, но на следующей строке
VT	11	Вертикальная табуляция; при выводе на экран заменяется специальным знаком
FF	12	Прогон страницы; при выводе на принтер формирует страницу, при выводе на экран заменяется специальным знаком
CR	13	Возврат каретки; вводится нажатием на клавишу Enter (при вводе с помощью READ или READLN означает команду «Ввод» и в буфер ввода не помещается; при выводе означает команду «Продолжить вывод с начала текущей строки»)
SUB	26	Конец файла; вводится с клавиатуры нажатием Ctrl-Z; при выводе заменяется специальным знаком
ESC	27	Конец работы; вводится с клавиатуры нажатием на клавишу ESC; при выводе заменяется специальным знаком

Символьная константа может записываться в тексте программы тремя способами:

- ☑ как один символ, заключенный в апострофы, например:

  ‘A’, ‘a’, ‘ю’;
- ☑ с помощью конструкции вида #k, где k – код соответствующего символа, при этом значение k должно находиться в пределах 0..255;
- ☑ с помощью конструкции вида ^C, где C – код соответствующего управляющего символа, при этом значение C должно быть на 64 больше кода управляющего

☑ символа.

Для символьных переменных применимы две функции преобразования:

ORD(C) и CHR(K)

Первая функция определяет порядковый номер символа С в наборе символов, вторая определяет по порядковому номеру К символ, стоящий на К-м месте в наборе символов.

Предыдущий и последующий символ:

PRED(C), SUCC(C)

Пример 29.

PRED(‘F’)=’E’, SUCC(‘Y’)=’Z’

Для литер из интервала ‘a’..’z’ применима функция UPCASE(C), которая переводит эти литеры в верхний регистр ‘A’..’Z’. Для остальных символов данная функция возвращает этот символ без изменения.

Вещественные типы

В отличие от порядковых типов, значения которых всегда сопоставляются с рядом целых чисел и, следовательно, представляются в ПК абсолютно точно, значения вещественных типов определяют произвольное число лишь с некоторой конечной точностью, зависящей от внутреннего формата вещественного числа (таблица 2.8.).

Таблица 2.8. Вещественные типы

Длина, байт	Название	Количество значащих цифр	Диапазон десятичного порядка
6	Real	11…12	-39…+38
8	Double	15...16	-324…+308
10	Extended	19...20	-4951...+4932
8	Comp	19...20	$-2*10^{63}+1...+2*10^{63}-1$

Как видно из таблицы 2.8., вещественное число в Турбо Паскале занимает от 4 до 10 смежных байт и имеет следующую структуру в памяти ПК:

s	e	m

Здесь s – знаковый разряд числа; e – экспоненциальная часть; содержит двоичный порядок; m – мантисса числа (см. раздел 2.1. – константы).

Мантисса m имеет длину от 23 (для SINGLE) до 63 (для EXTENDED) двоичных разрядов, что обеспечивает точность 7…8 для SINGLE и 19…20 для EXTENDED десятичных цифр. Десятичная точка (запятая) подразумевается перед левым (старшим) разрядом мантиссы, но при действиях с числом ее положение сдвигается влево или вправо в соответствии с двоичным порядком числа, хранящимся в экспоненциальной части, поэтому действия над вещественными числами называют арифметикой с плавающей точкой (запятой).

Рассмотрим понятие числа с плавающей точкой.

Примеры чисел с фиксированной точкой: 27.3, 5.0, -16.003.

Для изображения очень больших чисел в математике и физике используется запись:

$680000000=68*10^{7}$ $0,00000005=5*10^{-8}$

В языке Паскаль также можно изображать числа с десятичным порядком в сокращенной форме. Они имеют вид mEp, где m – мантисса, E – признак записи числа с десятичным порядком, p – порядок числа.

В качестве m могут быть целые и действительные числа с фиксированной точкой. В качестве p – только целые числа. Как мантисса, так и порядок могут содержать «+» или «-».

Числами с плавающей точкой являются числа, представленные с десятичным порядком.

Пример 30.

$4*10^{5}$=4E-5

$0,62*10^{4}$=0.62E+4

$-10,88*10^{12}$=-10.88E12

Одно и то же число может быть представлено по-разному, например: $0,62*10^{4}=6,2*10^{3}=62*10^{2}$ и т.д. Поэтому числа, представленные с десятичным порядком, назвали

числами с плавающей точкой.

Как видим, Турбо Паскаль характеризуется богатой гаммой вещественных типов, однако доступ к типам SINGLE, DOUBLE и EXTENDED возможен только при особых режимах компиляции. Дело в том, что эти типы рассчитаны на аппаратную поддержку арифметики с плавающей точкой и для их эффективного использования в состав ПК должен входить арифметический сопроцессор. Компилятор Турбо Паскаля позволяет создавать программы, работающие на любых ПК (с

сопроцессором или без него) и использующие любые вещественные типы. В процессе запуска Турбо Паскаль проверяет состав аппаратных средств и выявляет наличие или отсутствие сопроцессора.

Следует учесть, что тип REAL оптимизирован для работы без сопроцессора. Если ваш ПК оснащен сопроцессором, использование типа REAL приведет к дополнительным затратам времени на преобразование REAL к EXTENDED. Поэтому никогда не используйте REAL на ПК с оспроцессором, т.к. дополнительные затраты времени на преобразование типов могут свести на нет все преимущества сопроцессора.

Особое положение в Турбо Паскале занимает тип COMP, который трактуется как вещественное число без экспоненциальной и дробной частей. Фактически, COMP – это «большое» целое число со знаком, сохраняющее 19…20 значащих десятичных цифр. В то же время в выражениях COMP полностью совместим с любыми другими вещественными типами: над ним определены все вещественные операции, он может использоваться как аргумент математических функций и т.д. Наиболее подходящей областью применения типа COMP являются бухгалтерские расчеты: денежные суммы выражаются в копейках или центах и действия над ними сводятся к операциям с достаточно длинными целыми числами.

Над действительными переменными можно выполнять следующие операции:

+, -, *, /.

К переменным вещественных типов применимы функции, указанные в таблице 2.9.

Таблица 2.9. Стандартные математические функции Турбо Паскаля

Обращение	**Тип параметра**	**Тип результата**	**Примечание**
1	**2**	**3**	**4**
abs(x)	real, integer	тип аргумента	модуль аргумента
ArcTan(x)	real	real	арктангенс (значение в радианах)
cos(x)	то же	то же	косинус (угол в радианах)
exp(x)	то же	то же	экспонента (число e в степени аргумента)
frac(x)	то же	то же	дробная часть числа
int(x)	то же	то же	целая часть числа
ln(x)	то же	то же	логарифм натуральный
pi	-	то же	$\pi = 3{,}141592653...$
Random	-	то же	псевдослучайное число, равномерно распределенное в диапазоне 0..(1)
Random(x)	integer	integer	псевдослучайное целое число, равномерно распределенное в диапазоне 0..[x-1]
Randomize	-	-	инициация генератора случайных чисел
sin(x)	real	real	синус (угол в радианах)
sqr(x)	то же	то же	квадрат аргумента
sqrt(x)	то же	то же	корень квадратный
trunc(x)	то же	целый	выделяет целую часть действительного числа путем отсечения дробной части
round(x)	то же	целый	округляет аргумент до ближайшего целого

Следует заметить, что в Турбо Паскале нет операции возведения в степень (кроме квадрата числа). Поэтому ее заменяют экспонентой, возведенной в степень, которую выражают через натуральный логарифм, т.е.

$$a^x = \exp(x*\ln(a))$$

Однако, таким приемом нельзя воспользоваться в случае отрицательного основания.

Перечисляемый тип

Перечисляемый тип задается в Турбо Паскале перечислением тех значений, которые он может получать. Номера элементов начинаются с нуля.

Пример 31. Приведем пример программы с использованием перечисляемого типа:

```
Program  prim;
type
   typemonth = (jan, feb, mar, apr, may, jun, jul, aug, sep, oct, nov,
dec);
var
   month: typemonth;
   p1, p2: typemonth;
   n1, n2: integer;
begin
   month:=aug;
   p1:=may;
   p2:=succ(jul);
   n1:=ord(p1)+1;
   n2:=ord(p2)+1;
   writeln('Порядковый номер месяца may = ',n1:2);
   writeln('Порядковый номер месяца после июля = ',n2:2);
   readln
end.
```

После запуска программы на исполнение на экране появится:

```
порядковый номер месяца may = 5
порядковый номер месяца после июля = 8
```

В данном примере январь имеет порядковый номер – 0, февраль – 1 и т.д. После слова **type** идет перечисление типов, использованных в программе. В данном случае один тип-диапазон, названный программистом typemonth.

Переменные любого перечисляемого типа можно объявлять без предварительного описания этого типа.

Пример 32.

```
var
    col: (black, white, green);
```

Максимальная мощность перечисляемого типа 65536 значений.

Пример 33. Объявление перечисляемых типов.

```
Type Colors = (Red,Green,Blue);
     Numbers = (Zero,One,Two,Three,Four,Five);
var  c:Colors;     n:Numbers;
begin
  c:=Red;          write(Ord(c));        {0}
  n:=Four;         write(Ord(n));        {4}
  c:=Succ(c);      {c=Green}
  for n:=One to Five do    write(Ord(n));        {12345}
end.
```

Следует отметить, что стандартные типы **byte, word, char** и **boolean** также можно считать вариантами перечислимого типа.

Тип-диапазон

Тип-диапазон – есть подмножество своего базового типа, в качестве которого может выступать любой порядковый тип, кроме типа-диапазона. Он задается границами своих значений внутри базового типа: <мин.знач>..<макс.знач>

Пример 34.

```
type
    digit = '0'..'9';
    dig2 = 48..57;
var
    d: digit;
    d2: dig2;
```

Можно без описания типа, *например:*

```
var
        date: 1..31;
        month: 1..12;
        echr: 'A'..'Z';
```

Пример 35.

```
program week;
uses crt;
type
  days = (mo, tu, we, th, fr, sa, su);
  weekend = sa..su;
var
  w,w2:weekend;
  r:days;
  n:integer;
begin
  clrscr;
  w:=we;  {ошибка}
  r:=tu;
  r:=succ(r);
  writeln(ord(r));  {выведет 2}
  w:=sa;
  n:=ord(w);
  writeln(n);   {выведет 5}
  w2:=pred(w);
  writeln(ord(w2));   {выведет 4}
  readln
end.
```

High(x) – максимальное значение типа-диапазона; Low(x) – минимальное значение типа-диапазона.

Пример 36.

```
var k: integer;
begin
        writeln(low(k),'..',high(k))
```

```
end.
```

После исполнения программы на экран выведется строка

```
-32768..32767
```

Тип-диапазон наследует все свойства своего базового типа, но с ограничениями, связанными с его меньшей мощностью.

Пример 37. Объявление типа-диапазон.

```
type   Numbers = (Zero,One,Two,Three,Four,Five);
       Num = Two .. Four; {диапазон на базе типа Numbers}
       Abc = 'A' .. 'z'; {все английские буквы : диапазон на базе
   типа Char}
       Digits = 0 .. 9; {цифры}
var    n:Num;          c,d:Abc;        x:integer;
begin
   n:=Four;    writeln(Ord(n));       {4  как в базовом типе}
   n:=Succ(n); { ОШИБКА (следующее значение вне диапазона)}
   read(c,d);
   if c=d then write('одинаковые буквы');
   writeln(Low(c),' .. ',High(c)); { A .. z }
   writeln(Low(x),' .. ',High(x)); { -32768 .. 32767 }
end.
```

2.4. Оператор присваивания. Ввод-вывод данных. Составной оператор.

Оператор присваивания

Общая форма оператора присваивания **V:=A**, где V – *имя переменной*, A – *выражение*. То есть левая часть – ячейка памяти с заданным именем, а правая – то, что в эту ячейку «положили». При этом, если в ячейке ранее уже было какое-то значение, оно бесследно исчезает. Подробнее на примерах действие оператора присваивания рассказано и показано в примерах 1.11, 1.12, 2.10.

В операторе присваивания левая и правая части должны иметь совместимые между собой типы. Здесь следует заострить внимание на совместимости типов в Турбо Паскале.

Как уже неоднократно отмечалось, Турбо Паскаль - это типизированный язык. Он построен на основе строгого соблюдения концепции типов, в соответствии с которой все применяемые в языке операции определены только над операндами совместимых типов. При обсуждении операций над вещественными данными мы уже затрагивали проблему совместимости вещественных и целых типов. Аналогичные проблемы возникают при операциях над строками разной длины, строками и символами и т.д. Ниже приводится более полное определение совместимости типов.

Два типа считаются совместимыми, если:

- ☑ оба они есть один и тот же тип;
- ☑ оба вещественные;
- ☑ оба целые;
- ☑ один тип есть тип-диапазон второго типа;
- ☑ оба являются типами-диапазонами одного и того же базового типа;
- ☑ оба являются множествами, составленными из элементов одного и того же базового типа;
- ☑ оба являются упакованными строками (определены с предшествующим словом PACKED) одинаковой максимальной длины;
- ☑ один тип есть тип-строка, а другой - тип-строка, упакованная строка или символ;
- ☑ один тип есть любой указатель, а другой - нетипизированный указатель;
- ☑ один тип есть указатель на объект, а другой - указатель на родственный ему объект;
- ☑ оба есть процедурные типы с одинаковыми типом результата (для типа-функции), количеством параметров и типом взаимно соответствующих параметров.

Совместимость типов приобретает особое значение в операторах присваивания. Пусть T1 - тип переменной, а T2 - тип выражения, т.е. выполняется присваивание T1:=T2. Это присваивание возможно в следующих случаях:

- ☑ T1 и T2 есть один и тот же тип и этот тип не относится к файлам или массивам файлов, или записям, содержащим поля-файлы, или массивам таких записей;
- ☑ T1 и T2 являются совместимыми порядковыми типами и значение T2 лежит в диапазоне возможных значений T1;
- ☑ T1 и T2 являются вещественными типами и значение T2 лежит в диапазоне возможных значений T1;
- ☑ T1 - вещественный тип и T2 - целый тип;
- ☑ T1 - строка и T2 - символ;
- ☑ T1 - строка и T2 - упакованная строка;
- ☑ T1 и T2 - совместимые упакованные строки;
- ☑ T1 и T2 - совместимые множества и все члены T2 принадлежат множеству возможных значений T1;
- ☑ T1 и T2 - совместимые указатели;
- ☑ T1 и T2 - совместимые процедурные типы;
- ☑ T1 - объект и T2 - его потомок.

В программе данные одного типа могут преобразовываться в данные другого типа. Например, когда используются вызовы специальных функций преобразования, аргументы которых принадлежат одному типу, а значение - другому. Таковыми являются уже рассмотренные функции ORD, TRUNC, ROUND, CHR.

Пример 38. Допустим ли в Турбо Паскале следующий оператор присваивания:

a) d/2+5:=6;
b) L:=M and N?

Решение: a) не допустим, так как в левой части стоит не переменная, а выражение; b) допустим, если переменная L имеет логический тип, так как выражение в правой части логического типа.

Составной оператор

Составной оператор – объединение нескольких операторов в одну группу. Форма записи данного оператора:

```
BEGIN
        BEGIN
                Оператор 1;
                Оператор 2;
                Оператор 3;
                ……;
                Оператор k
        END;
        Оператор k+1;
        ……;
        Оператор m;
        BEGIN
                Оператор m+1;
                ……;
                Оператор n
        END
END;
```

Как уже было сказано ранее, слова **begin** и **end** являются операторными скобками. Старайтесь все операторы между ними располагать немного правее и строго друг под другом, и сами слова **begin** и **end** также друг под другом – это правила хорошего стиля программирования, так программа будет понятной, структурированной, в ней легче потом будет найти ошибку. Обратите внимание, что после оператора, предшествующего слову **end** точка с запятой может быть опущена. Чтобы было понятно, можно представить аналогию обычных скобок в русском языке и перечислений между ними, например, следующая запись в русском языке считается грамотной:

(A, B, C, D).

Здесь, разделителем является запятая, которая не ставится перед закрывающейся скобкой. Точно также работают

операторные скобки **begin** и **end** в Турбо Паскале, только разделителем операторов является точка с запятой.

Оператор ввода

Оператор ввода – READ. Или используется READLN. Если после оператора READLN не стоит ничего, кроме точки с запятой, то компьютер будет ждать от пользователя ввода клавиши enter. В противном случае после оператора ввода записываются в скобках через запятую имена переменных, значения которых необходимо ввести пользователю (пример 2.10.). Как уже было рассмотрено в примере 2.10., отличаются две записи READ и READLN тем, что с помощью первой значения переменных вводятся в строку через пробел, а с помощью второй – с новой строки через enter. С помощью оператора ввода пользователь задает (вводит значения переменных) на этапе исполнения программы вводимым переменным нужные ему значения. В этом случае программа следует требованию массовости, т.е. значения переменных задаются не в программе программистом, а пользователем при исполнении программы, при этом пользователь может ввести произвольные значения, какие посчитает нужным.

Действие оператора ввода в каком-то смысле аналогично действию оператора присваивания. Т.е. в ячейку памяти компьютера, которую представляет вводимая переменная вводится значение, при этом, если в ней было другое значение, оно стирается и заменяется новым вводимым. Отличие от оператора присваивания лишь в том, что пользователь сам на этапе исполнения определяет и вводит значения переменным и записать в операторе ввода выражение – грубая ошибка. Т.е. в скобках можно записывать только имена описанных в разделе описания переменных, но никак не выражения, константы и т.п.

Как уже было сказано, операторы в программе при компиляции или исполнении выполняются строго друг за другом в той последовательности, в которой они были написаны программистом. Когда компилятор (компилятор- это программа,

которая выполняет последовательно две задачи: 1. проверяет текст исходной программы на отсутствие синтаксических ошибок; 2. создает или генерирует исполняемую программу — машинный код, т.е. переводит текст программы на язык, понятный процессору компьютера) доходит до оператора ввода при исполнении программы, он ждет от пользователя ввода значений переменных в той последовательности, в которой они были указаны в операторе ввода.

Пример 39. a) После исполнения следующего фрагмента программы:

```
program prim1;
var
        AB, CD, H: real;
begin
        writeln (‘Введите значения AB, CD, H: ’);
        read (AB, CD, H);
        write (AB, CD, H);
        readln
end.
```

на экран выведется надпись: «Введите значения AB, CD, H:» и на экране появится курсор, т.е. компилятор остановится в ожидании, когда пользователь введет три значения переменных AB, CD, H, а именно три действительных числа в строку через пробел.

b) После исполнения следующего фрагмента программы:

```
program prim2;
var
        AB, CD, H: real;
begin
        writeln (‘Введите значения AB, CD, H: ’);
        readln (AB, CD, H);
        write (AB, CD, H);
        readln
```

```
end.
```

компилятор остановится в ожидании, когда пользователь введет три действительных числа, каждое с новой строки через enter.

Следует обратить внимание на ввод символьных данных, так как пробел и enter считаются символами.

Пример 40.

```
var a,b,c:char;
begin
        read (a,b,c);
        writeln (a,b,c);
        readln
end.
```

Попробуйте исполнить данную программу. Если при исполнении мы введем FGH, то переменная a будет иметь значение «F», переменная b – значение «G», а переменная c – значение «H», а если введем F G H, то переменная a будет иметь значение «F», переменная b – значение « », т.е. пробел, а переменная c – значение «G». Остальные символы игнорируются.

Enter считается пробелом, поэтому перед оператором ввода символьных данных нужно ставить READLN и каждую переменную записывать в новом операторе ввода. Поэкспериментируйте с предложенной в примере 2.22. программой, используя вместо read – readln.

Если перед **end**. не записать readln, то компьютер очень быстро исполнит программу и мы не успеем ничего увидеть на экране. Поэтому для того, чтобы задержать экран, пишем перед **end**. оператор readln. В таком случае компьютер, дойдя до данного оператора, остановит исполнение программы до тех пор, пока пользователь не нажмет enter.

Оператор вывода

Оператор вывода – WRITE. Или используется WRITELN. Если после данного оператора не стоит ничего, кроме точки с запятой, то на экран выведется пустая строка, при этом WRITE оставляет курсор на текущей строке, а WRITELN переводит курсор на новую строку. В противном случае после оператора вывода записываются в скобках через запятую имена переменных, выражения, константы (не забывайте, что строковые и символьные константы записываются в апострофах), значения которых необходимо вывести пользователю (пример 2.10.). С помощью оператора вывода компьютер выводит значения переменных (выражений, констант), указанных в скобках в той последовательности, в которой они были записаны программистом.

Пример 41. Выполните следующую программу:

```
program pr;
uses crt;
var
    A, B: real;
    C: integer;
    D, E: char;
const
    F=’МИР’;
    G=2.5;
begin
    clrscr;
    {1}  write (‘Выводим значения переменных A, B, C: ’);
    {2}  writeln (A:5:2, B:5:2, C:3);
    {3}  write (‘Введите значения A, B, C: ’);
    {4}  readln (A, B, C);
    {5}  write (‘Введите значения D, E: ’);
    {6}  read (D, E);
    {7}  readln;
    {8}  write (‘Выводим значения переменных A, B, C: ’);
```

```
{9}   writeln (A:5:2, B:5:2, C:3);
{10} writeln (‘Выводим значения переменных D, E и константы G: ’);
{11} writeln (‘D=’, D:3,’   ‘, ‘E=’, E:3, ‘G=’, G:5);
{12} writeln (‘Значение константы в квадрате равно ’, sqr(G));
{13} writeln (‘Значение выражения (A+B)*C равно’, (A+B)*C);
{14} writeln (‘Выводим число 333’, 333);
readln;
end.
```

Разберем выполнение операторов программы. В первом операторе вывода записана строковая константа, которая и выйдет на экран, при этом курсор останется на той же строке.

После выполнения второго оператора вывода на ту же строку выведутся значения переменных A, B, C в строку и курсор перейдет на следующую строку. При этом для действительных переменных выделяется пять знаков, из которых два знака отведены для дробной части. Для переменной целого типа отведено три знака. Это сделано для того, чтобы оставить между значениями переменных некоторое расстояние. Ясно, что пока значения переменных A, B, C имеют нулевые значения.

После исполнения третьего оператора на экран выводится строковая константа, значение которой «Выводим значения переменных A, B, C: » и курсор остается на той же строке.

После четвертого оператора компьютер ждет от пользователя ввода значений переменных A, B, C. Введите данные значения с новой строки каждое или через пробел на той же строке и нажмите enter. Не ошибитесь в типах.

Пятый оператор выводит строковую константу «Введите значения D, E: » и оставляет курсор на той же строке.

Шестой оператор ждет ввода символьных переменных. Вводим два символа на той же строке без пробела.

Седьмой оператор необходим, чтобы перейти на новую строку.

Восьмой и девятый операторы аналогичны первому и второму, только теперь на экран выйдут значения числовых переменных, которые ввел пользователь и курсор переходит на новую строку.

В десятом операторе выводится строковая константа и курсор переходит на новую строку.

В одиннадцатом операторе выводятся три строковые константы, значения которых равны: «D=», «E=» и «G=», значения двух переменных символьного типа и значение константы, описанной в разделе описаний. Все данные значения в операторе вывода разделяются запятой, значения строковых констант записываются в апострофах.

В двенадцатом операторе вывода выводится значение строковой константы и значение выражения sqr(G). Данные значения разделяются запятой.

Тринадцатый оператор аналогичен двенадцатому.

Четырнадцатый оператор выводит значения строковой константы и числовой константы целого типа, которые разделяются запятой.

2.5. Условный оператор, оператор выбора и безусловный переход

Условный оператор

Оператор условия применяется в разветвляющихся алгоритмах.

Пример 42. Примеры разветвляющихся алгоритмов:

☑ Найти значение функции:

$$y = \begin{cases} x+3, & \text{если} \quad x < 0 \\ 2x, & \text{если} \quad x \geq 0 \end{cases};$$

- ☑ Результат такого диалога: «Введите 0, если хотите увидеть собачку и любую другую цифру, если ничего не хотите»;
- ☑ Результат диалога: «Хотите выйти из программы? (y/n)»;
- ☑ Девушка собирается на дискотеку и рассуждает: «Если там будет Саша, то я приглашу его на танец, а если Саша будет с Машей, то Маша об этом пожалеет».

Запись условного оператора в Турбо Паскале:

if <логическое выражение> **then** <оператор1> **else** <оператор2>

Допускается использование пустого оператора 1 или 2.

Эта конструкция является одним целостным оператором, поэтому никаких знаков препинания в середине ее не ставится. Дословно данную запись можно перевести: «Если логическое выражение является истинным, то выполнить первый оператор, иначе выполнить второй оператор».

В качестве операторов 1 или 2 могут быть также операторы условия. Условный оператор относится к сложным.

Условие не обязательно должно иметь форму операции отношения. Оно может принимать вид любого выражения, в частности логической переменной. Т.е. после слова **if** допускается использовать только логическое выражение, результат которого true или false (смотрите в таблицах 2.1. и 2.3. операции, тип результата которых логический) или переменная логического типа.

Пример 43. Примеры логических выражений:

1. h>0;
2. g<=3;
3. c **and** j {если c и j логического типа};
4. v=-2 {нельзя использовать оператор присваивания, т.к. это не знак сравнения};
5. f {если f логического типа};
6. (n>15) **and** (n<=25) {в Турбо Паскале нельзя использовать запись 15<n<=25, такое сложное условие разбивается на простые, обратите внимание на скобки};
7. (c=’январь’) **or** (c=’март’) **or** (c=’май’) **or** (c=’июль’) **or**

8. (с=’август’) **or** (с=’октябрь’) **or** (с=’декабрь’) {в данном случае сложное условие разбито на простые, переменная с строкового типа}.

В любом случае, чтобы проверить себя, проговорите логическое выражение, которое вы записали, подумайте, будет ли логическим его результат, т.е. истинным или ложным.

Пример 44. Примеры фрагментов программ с использованием оператора условия. Исполните данные программы, дописав разделы заголовка и описаний и, где необходимо, операторы ввода-вывода.

1. Найти значение функции:

$$y = \begin{cases} x+3, & \textit{если} \quad x < 0 \\ 2x, & \textit{если} \quad x \geq 0 \end{cases}.$$

```
writeln (‘Введите x’);
readln (x);
if x<0 then y:=x+3 else y:=2*x
```

Данную задачу можно было бы записать с помощью двух операторов условия:

```
if x<0 then y:=x+3;
if x>=0 then y:=2*x
```

Или так:

```
if x<0 then y:=x+3
else
    if x>=0 then y:=2*x {здесь одно условие вложено в другое}
```

Такие записи являются не рациональными (так как в этом случае процессору приходится два раза проверять условие, что в два раза дольше), хотя верными. Так как условия $x<0$ и $x \geq 0$ являются взаимоисключающими (т.е. если x не меньше нуля, то он обязательно больше или равен нулю), следовательно лучше в данном случае использовать слово else (иначе).

2. Написать программу, которая выводит сообщение, брать зонт или не брать в зависимости от того, идет дождь или нет.

```
var s: char;
begin
        writeln (‘Идет дождь? y/n’);
        readln (s);
        if s=’y’ then writeln (‘Берите зонт!’)
        else writeln (‘Не берите зонт!’);
        readln
end.
```

Здесь переменная s символьного типа может принимать значения либо «y» либо «n». В программе мы предложили пользователю ввести значение переменной s, т.е. y или n, далее записали, что если s=’y’, то выводим сообщение: «Берите зонт!». В любом другом случае выводим сообщение: «Не берите зонт!»

3. В данном примере переменная B логического типа и имеет одно из значений true или false. Если B – истинно, то вычисляем y:=x+1, а если ложно, то вычисляем y:=2*x:

```
B:=x<0;
 if B then y:=x+1
 else y:=2*x
```

4. В данном примере логическое выражение является сложным:

```
if (n>15) and (n<=25) then a:=n+40 else b:=m+1
```

5. Найти значение функции $x = \begin{cases} 1, \textit{если} & a = b \ и \ c < d \\ 2, \textit{если} & a = b \ и \ c \geq d \\ 3, \textit{если} & a \neq b \end{cases}$

```
if a=b then
        if c<d then x:=1
        else x:=2
else x:=3
```

Эта запись является самой рациональной и логически выдержанной. Обратите внимание, что внутри первого оператора условия содержится второй. Записаны они с отступами **else** строго под **if**. Дословно эту запись можно прочитать так: «Если a=b, то возможны два

взаимоисключающих варианта: либо c<d либо c>=d, в результате истинности которых переменной x присваиваются разные значения. Иначе, т.е. если a<>b, переменной x присваивается значение 3».

То же самое можно было записать иначе:

```
a) if (a=b) and (c<d) then x:=1;
   if (a=b) and (c>=d) then x:=2;
   if (a<>b) then x:=3
b) if (a=b) and (c<d) then x:=1;
   if (a=b) and (c>=d) then x:=2
   else if (a<>b) then x:=3
c) if (a=b) then
        if (c<d) then x:=1
        else x:=2
   else if (a<>b) then x:=3
```

Есть еще несколько вариантов. Однако такие записи не рациональны.

В Турбо Паскале можно использовать условный оператор с вложенными составными операторами. Запись такого оператора:

```
if <логическое выражение> then
        begin
                <операторы>
                …..
        end
else
        begin
                <операторы>
                ……
        end;
```

Здесь слова **begin** и **end** выступают в качестве операторных скобок, между которыми можно записать любое количество (через точку с запятой) любых операторов. Обратите внимание, что слова **if, then, else** – это один целостный оператор

и нигде знаки препинания в нем не ставятся. Точка с запятой ставится только между операторами, расположенными между **begin** и **end**.

Используется также краткая форма оператора условия:

if <логическое выражение> **then** <оператор>

Дословно данная запись переводится: «Если логическое выражение является истинным, то выполнять оператор». Т.е. в любом другом случае ничего не выполнять. Компьютер проверит условие и если оно ложно, перейдет к следующему оператору.

Пример 45. Написать программу, в результате которой два числа сортируются по возрастанию (смотрите пример 1.12).

```
program sort;
uses crt;
var a, b, R: real;
begin
        clrscr;
        writeln (‘Введите два числа’);
        readln (a, b);
        if a>=b then
                begin
                        R:=a;
                        a:=b;
                        b:=R;
                end
        writeln (‘Отсортированные числа:’);
        writeln (a, ‘      ‘, b);
        readln
end.
```

Обратите внимание на запись: отступы, выравнивание. Слова begin и end как операторные скобки располагаются строго друг под другом и все, что между ними – немного правее.

Операторные скобки, если их несколько, могут быть только вложенными.

Так программа становится более удобочитаемой, структурированной. Вспомните и повторите три оператора замены значений двух переменных. Просмотрите внимательно от начала до конца пример 1.12. Рассмотрите блок-схему.

Пример 46. Найти модуль числа. Вспомним из математики, что такое модуль числа. Модуль – это расстояние между точками. Оно не может быть отрицательным. Поэтому, если число отрицательное, мы заменяем его на противоположное.

```
writeln (‘Введите число’);
readln (a);
if a<0 then a:=-a;
writeln (‘Модуль данного числа равен ’, a);
```

Пример 47. Определить, попадет ли точка a с координатами xa и ya внутрь круга с радиусом R. Центр круга совпадает с началом координат (рисунок 2.4.).

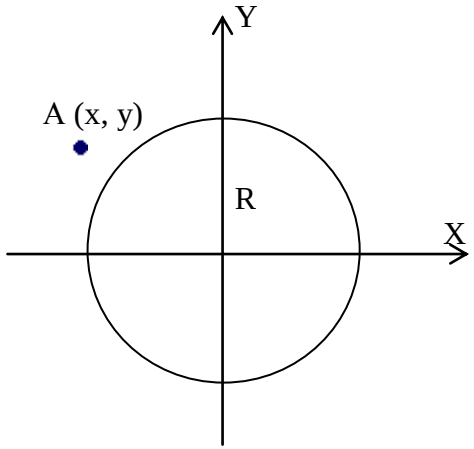


Рисунок 2.4. Иллюстрация к примеру 47.

Вспомним математику. Расстояние от начала координат до окружности равно $\sqrt{x_a^2 + y_a^2}$. Если оно меньше R, то точка попадает в круг. Блок-схема изображена на рисунке 2.5.

```
program circle;
uses crt;
var
                xa, ya, R, L:real;
begin
  clrscr;
  write ('Введите значения xa, ya, R: ');      read(xa, ya, R);
  L:=sqr(xa)+sqr(ya);
  if L<sqr(R) then
  writeln ('Точка находится внутри круга')
  else writeln ('Точка находится вне круга');
  readln;
  readln
end.
```

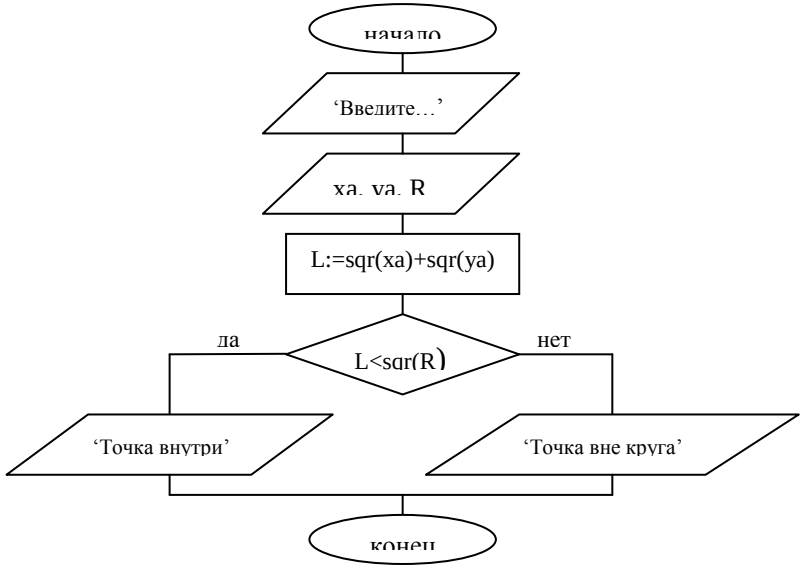


Рисунок 2.5. Блок-схема к программе примера 2.29

Пример 48. Разберем некоторые часто встречающиеся ошибки при использовании оператора условия. Выполните данные программы самостоятельно, дополнив их необходимыми разделами и операторами.

1. **if** a>0 **then**
   y:=3;
   y:=2;
   writeln (y);

В данном случае при истинном значении логического выражения a>0 выполнится один оператор y:=3. Оператор y:=2 выполнится в любом случае. Поэтому в результате, каким бы не было значение a, на экран выйдет 2.

2. **if** a>0 **then**
   y:=3
else y:=2;
writeln (y);

В данном случае при истинном значении логического выражения a>0 выполнится оператор y:=3, а иначе оператор y:=2.

3. **if** a>0 **then**
   begin
      y:=3;
      y:=2;
   end;
   writeln (y);

Здесь операторы y:=3 и y:=2 выполнятся лишь в том случае, если условие истинно, и на экран выйдет число 2, а иначе – 0, так как после слова **then** идут два оператора.

4. **if** a>0 **then**
   begin
      y:=3;
      y:=2;
   end
   else

```
begin
        y:=2;
        y:=3;
  end;
writeln (y);
```

Если условие истинно, то на экран выйдет 2, а если ложно, то на экран выйдет 3.

Оператор выбора.

Если в команде ветвления необходимо указать более двух вариантов действий, то ее запись становится неудобной. Для реализации этих ситуаций в Турбо Паскале есть оператор выбора.

Оператор выбора позволяет выбрать одно из нескольких возможных продолжений программы. Параметром, по которому осуществляется выбор, служит ключ выбора – выражение любого порядкового типа. Структура оператора:

case <ключ выбора> **of** <список выбора> [**else** <операторы>] **end**

Здесь **case, of, else, end –** зарезервированные слова (случай, из, иначе, конец);

<ключ выбора> - ключ выбора;

<список выбора> - одна или более конструкций вида:

<константа выбора> : <оператор>;

<константа выбора> - константа того же типа, что и выражение <ключ выбора>;

<операторы> - произвольные операторы Турбо Паскаля.

Оператор выбора работает следующим образом. Вначале вычисляется значение выражения <ключ выбора>, а затем в последовательности операторов <список выбора> отыскивается такой, которому предшествует константа, равная вычисленному значению. Найденный оператор выполняется, после чего оператор выбора завершает свою работу. Если в списке выбора не будет найдена константа, соответствующая вычисленному значению ключа выбора, управление передается операторам, стоящим за словом **else**. Часть **else** <оператор> можно опускать.

Тогда при отсутствии в списке выбора нужной константы ничего не произойдет и оператор выбора просто завершит свою работу.

Пример 49. Приведем примеры задач, которые удобно решать с помощью оператора выбора.

1. Понедельник – день тяжелый;
   Вторник – уже не понедельник, но уже и не пятница;
   Среда – ух, скорее бы пятница;
   Четверг – ура!, завтра пятница;
   Пятница – Это надо отметить!
2. Направо пойдешь – смерть найдешь, налево пойдешь – коня потеряешь, прямо пойдешь – богатство найдешь.

Пример 50. Определить по номеру соответствующий день недели на русском и английском языках.

```
program E32;
var
        n:integer;
begin
        writeln (‘Введите номер дня недели’);
        readln (n);
        case n of
                1: writeln (‘Понедельник - Monday’);
                2: writeln (‘Вторник - Tuesday’);
                3: writeln (‘Среда - Wednesday’);
                4: writeln (‘Четверг - Thursday’);
                5: writeln (‘Пятница - Friday’);
                6: writeln (‘Суббота - Saturday’);
                7: writeln (‘Воскресенье - Sunday’)
                else writeln (‘Такого дня недели нет’)
        end
end.
```

Блок-схема к программе изображена на рисунке 2.6.

Пример 51. Определить по номеру дня недели будний это день или выходной. Считаем выходными субботу и воскресенье.

```
program E33;
var
    n:integer;
begin
    writeln (‘Введите номер дня недели’);
    readln (n);
    case n of
        1..5: writeln (‘Будний день’);
        6, 7: writeln (‘Выходной’);
    end    else writeln (‘Такого дня недели нет’);
end.
```

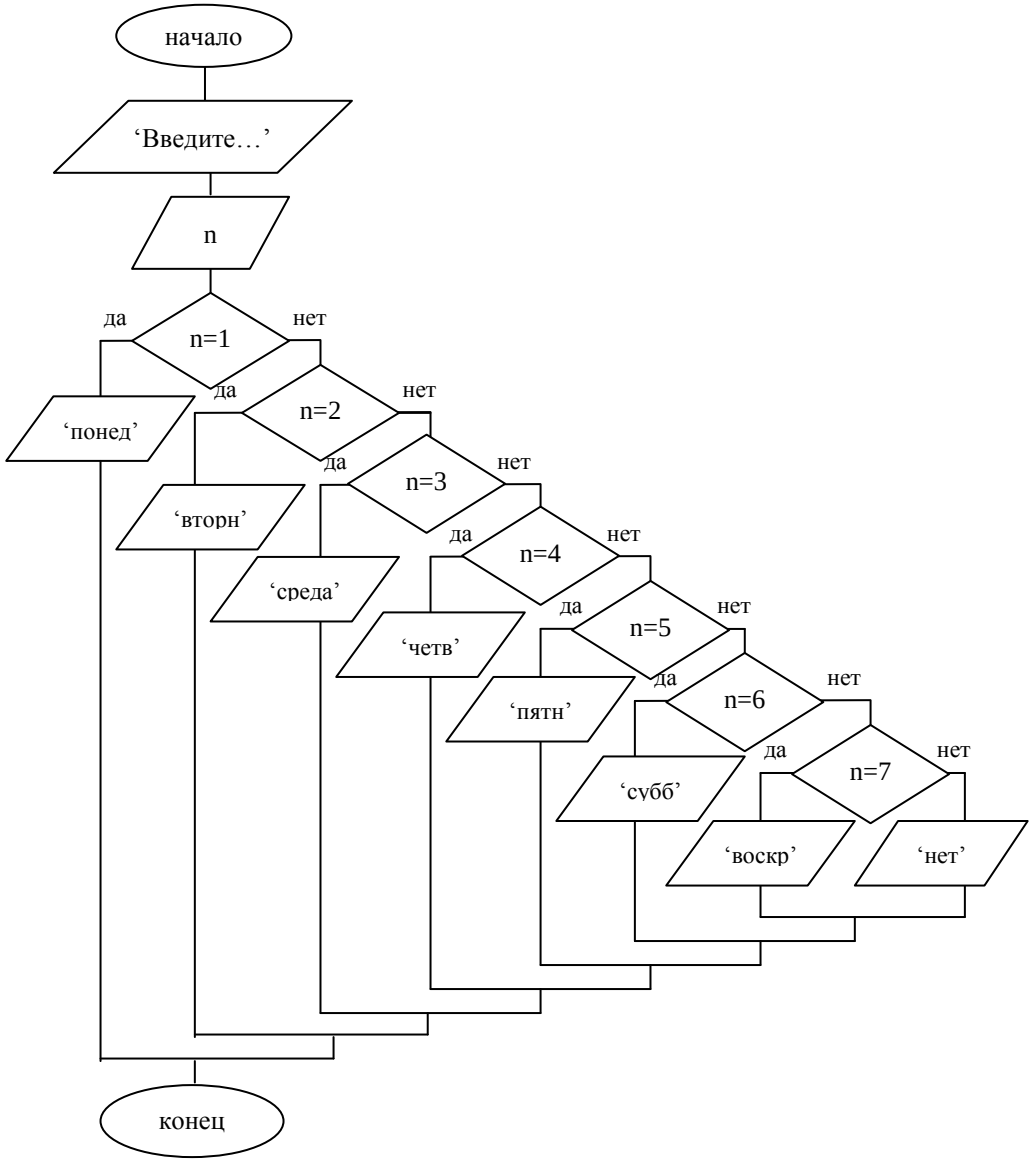


Рисунок 2.6. Блок-схема к программе примера 50

Запись 1..5 означает, что переменная принимает значения от 1 до 5. Так в Турбо Паскале записывается тип множество, о котором речь пойдет далее. Однако для переменной действительного типа такая запись будет ошибочной, так как этот тип не является порядковым, а для символьного типа переменная может принимать значения – символы, идущие строго по порядку в котором они записаны в кодировочной таблице.

Если переменная принимает несколько, не следующих друг за другом значений, то в операторе case эти значения перечисляются через запятую.

Оператор безусловного перехода.

Современная технология структурного программирования основана на принципе «программировать без оператора безусловного перехода – **goto»**: считается, что злоупотребление операторами перехода затрудняет понимание программы, делает ее запутанной и сложной в отладке. Тем не менее, в некоторых случаях использование операторов перехода может упростить программу в тех случаях, когда, например, необходимо обойти участок программы и вернуться к нему позже. Оператор перехода имеет вид:

goto <метка>;

Здесь **goto –** зарезервированное слово (перейти на метку)**;**

<метка> - метка.

Метка в Турбо Паскале – это произвольный идентификатор, позволяющий именовать некоторый оператор программы и таким образом ссылаться на него, допускается использовать целые числа без знака.

Метка располагается перед помечаемым оператором и отделяется от него двоеточием. Оператор можно помечать несколькими метками, которые в этом случае отделяются друг от друга двоеточием. Перед тем, как появиться в программе,

метка должна быть описана. Описание меток осуществляется в разделе описания label.

Пример 52. Пример описания и использования меток.

```
program lab;
label
        lb1, lb2;
begin
        ……..
        goto lb1; {после этого оператора компилятор перейдет сразу к оператору, помеченному меткой lb1}
        ……...
        lb2: writeln (‘*’);
        lb1: writeln (‘***’);
        …….
        goto lb2; {после этого оператора компилятор перейдет сразу к оператору, помеченному меткой lb2, т.е. вернется на несколько шагов назад}
        …….
end.
```

Оператор **goto** пересылает действие программы к оператору, помеченному меткой. В примере 2.34. после второго оператора безусловного перехода программа окажется зацикленной. Именно из-за возникновения таких ситуаций квалифицированный программист не использует **goto.**

Правила по использованию меток: 1. Метка, на которую ссылается оператор **goto**, должна быть описана в разделе описаний и она обязательно должна встретиться где-нибудь в теле программы; 2. Метки, описанные в процедуре (функции), локализуются в ней, поэтому передача управления извне процедуры (функции) на метку внутри нее невозможна.

2.6. Операторы цикла итерационного типа

При решении задач часто возникает необходимость многократного повторения однотипных действий при различных

значениях параметров, определяющих эти действия. Как уже говорилось, алгоритмы, реализующие такие действия, называются *циклическими*, а многократно повторяемая последовательность действий (тело цикла) – *циклами* или *итерациями*. Использование циклов позволяет выполнять большие объемы вычислений при помощи компактных программ.

В языке Турбо Паскаль имеются три различных оператора, с помощью которых можно запрограммировать повторяющиеся фрагменты программы. Здесь мы рассмотрим циклы с предусловием и с постусловием.

Цикл с предусловием

Иногда заранее неизвестно, сколько раз надо повторить цикл, но известно, что он должен выполняться, пока справедливо некоторое условие.

Оператор цикла **while** с предпроверкой условия: (цикл с предусловием или цикл с неизвестным числом повторений):

while <условие> **do** <оператор>;

Здесь **while**, **do** – зарезервированные слова (пока выполняется условие, делать);

<условие> - выражение логического типа;

<оператор> - произвольный оператор Турбо Паскаля.

Действия операторов, следующих за словом **do**, т.е. входящих в цикл, будут повторяться до тех пор, пока истинно условие. Условием, также как и в операторе условия, может быть логическое выражение, т.е. выражение, результатом которого будет true или false.

Оператор цикла с предусловием, также как и любой другой оператор, может быть и составным, т.е. после слова do может выполняться несколько операторов, в том числе и операторы условия, выбора, цикла и др., тогда эта группа отделяется операторными скобками.

Указанная после слова do серия команд (тело цикла) выполняется столько раз, сколько нужно для того, чтобы

заданное условие перестало соблюдаться. Если условие не соблюдается с самого начала, то серия не выполняется ни разу.

Пример 53. Приведем примеры задач, которые решаются с помощью оператора цикла с предусловием:

- ☑ Пока не закончится посадочная полоса, полем куст картофеля и перемещаемся к следующему.
- ☑ Пока тетя Полли не устанет, она будет повторять: «Том, подойди ко мне!»
- ☑ x:=0. Пока x не станет равным десяти, будем увеличивать его на единицу.

Пример 54. Найти так называемое «машинное эпсилон» - такое минимальное, не равное нулю вещественное число, которое после прибавления его к 1.0 еще дает результат, отличный от 1.0.

```
program epsilondetect;
uses crt;
var
        epsilon: real;
begin
        clrscr;
        epsilon:=1;
        while epsilon/2+1>1 do
                epsilon:= epsilon/2;
        writeln('Машинное эпсилон = ', epsilon); readln
end.
```

Обратите внимание на форматирование текста программы. Операторы тела цикла (в данной задаче он один) располагаются немного правее слова **while**.

В данной программе переменной epsilon присваивается перед оператором цикла начальное значение 1. Далее в цикле каждый раз данная переменная уменьшается в два раза, т.е после первой итерации epsilon=1/2, после второй итерации epsilon=1/4 и т.д., до тех пор, пока выполняется условие epsilon/2+1>1.

У читателя, привыкшего к непрерывной вещественной арифметике, может вызвать недоумение утверждение о том, что в дискретной машинной арифметике всегда существуют такие числа 0<X<eps, что 1.0+X=1.0. Дело в том, что внутреннее представление типа REAL может дать «лишь» приблизительно 1014 возможных комбинаций значащих разрядов в отведенных для него 6 байтах. Конечно же, это очень большое число, но оно несопоставимо с бесконечным множеством вещественных чисел. Аппроксимация бесконечного непрерывного множества вещественных чисел конечным (пусть даже и очень большим) множеством их внутреннего машинного представления и приводит к появлению «машинного эпсилон».

Пример 55. Вычислить значения функции на заданном интервале с заданным шагом $y=\sin x+x^2$.

```
program function1;
uses crt;
var
        x: integer;
            a,h,b: integer;
            y:real;
begin
            clrscr;
        writeln ('Введите границы интервала');
        readln (a, b);
        writeln ('Введите шаг');
        readln (h);
        x:=a;
        while x<=b do
                begin
                        y:=sin(x)+x*x;
                        writeln ('y=',y:5:2);
                        x:=x+h
                end;
        readln          end.
```

Сначала вводятся значения начала и конца интервала и шага. Затем переменной x присваивается начальное значение – левая граница интервала. Представляем отрезок, левая граница которого равна a, а правая – b. Он разбит точками на части, равные h (рисунок 2.7).

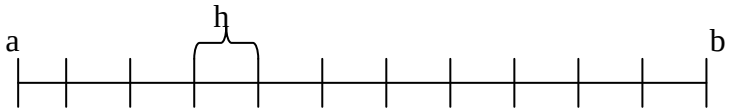


Рисунок 2.7. Интервал, разбитый на равные отрезки

Затем в цикле переменная x увеличивается каждую итерацию на h до тех пор, пока x не будет меньше или равен b. В теле цикла также высчитывается значение функции в каждой из данных точек разбиения отрезка и ее значение выводится на экран. Обратите внимание на то, что наращивание переменной x x:=x+h и вывод значений функции writeln (‘y=’,y) на экран производится внутри цикла. Таким образом, x наращивается столько раз, сколько нужно, пока не будет ложным условие, а значения функции выводятся все, а не только последнее.

Блок-схема к программе представлена на рисунке 2.8.

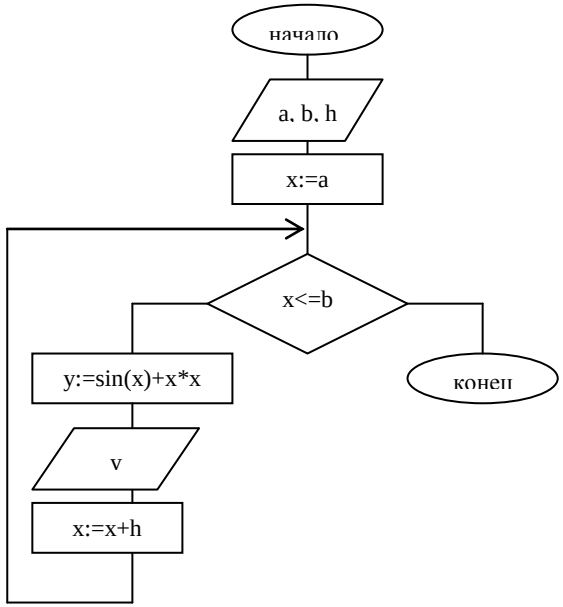


Рисунок 2.8. Блок-схема к программе примера 55

Анализируя примеры 2.36. и 2.37., можно сделать общие выводы по использованию оператора цикла с предусловием. *Запомните их, пожалуйста*.

Для правильной организации цикла необходимо:

1. задать начальное значение изменяемого параметра перед началом цикла, иначе компилятор будет работать с начальным нулевым значением параметра;
2. задать и проверить условие окончания цикла, так как обратное может привести к зацикливанию программы;
3. изменить параметр перед новым повторением цикла, иначе возникнет зацикливание, так как условие не выполнится никогда.

Цикл с постусловием

Оператор цикла **repeat**…**until** с постпроверкой условия (цикл с постусловием или цикл с неизвестным числом повторений) :

repeat <тело цикла> **until** <условие>;

Здесь **repeat, until** – зарезервированные слова (повторять до тех пор, пока не будет выполнено условие);

<тело цикла> - произвольная последовательность операторов Турбо Паскаля;

<условие> - выражение логического типа.

Принцип действия оператора цикла с постусловием аналогично действию цикла с предусловием. Операторы тела цикла выполняются *хотя бы раз* пока условие ложно, а условие проверяется в конце цикла. Этим цикл с постусловием отличается от цикла с предусловием. Значение зарезервированных слов цикла с постусловием можно представить как: «Выполнять указанные действия до тех пор, пока ложно условие».

Пример 56. Приведем примеры задач, которые решаются с помощью оператора цикла с постусловием:

- ☑ Полем куст картофеля и перемещаемся к следующему, пока не закончится посадочная полоса.
- ☑ Тетя Полли будет повторять: «Том, подойди ко мне!» до тех пор, пока не устанет.
- ☑ x:=0. Будем увеличивать x на единицу пока он не станет равным десяти.

Пример 57. В результате исполнения программы вводятся подряд символы и печатаются эти символы с кодами до тех пор, пока не будет нажата клавиша enter.

```
program codes_of_chars;
uses crt;
var     ch: char;
const   cr=13;
```

```
begin
        clrscr;
        repeat
                readln (ch);
                writeln (ch,' = ', ord(ch))
        until ord (ch) = cr;
        readln
end.
```

Блок-схема программы изображена на рисунке 2.9.

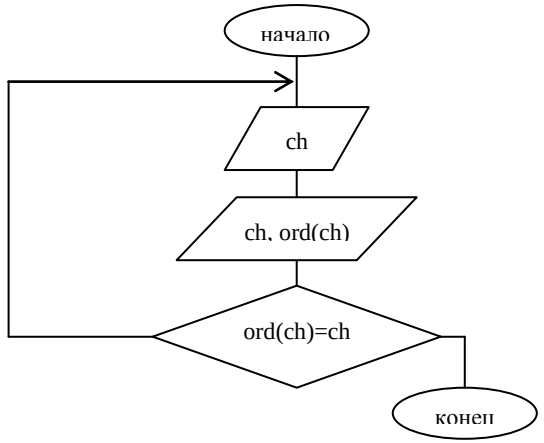


Рисунок 2.9. Блок-схема программы примера 57

Зарезервированные слова **repeat** и **until** являются операторными скобками, поэтому между ними может быть помещено сколько угодно операторов, являющихся телом цикла, без использования слов **begin** и **end**.

Примеры задач с использованием итерационных циклов.

Пример 58. Пары неотрицательных вещественных чисел вводятся с клавиатуры. Посчитать произведение для каждой пары и сумму всех чисел.

```
program cycle_while;
var     x,y,sum:real;   otv:char;
begin
```

```
    sum:=0;
    otv='Д';
    while (otv='Д') or (otv='д') do
    begin
        write('Введите числа x,y > 0 ');
        readln(x,y);
        writeln('Их произведение = ',x*y:8:3);
        sum:=sum+x+y;
        write('Завершить программу (Д/Н)? ');
        readln(otv);
    end;
    writeln('Общая сумма = ',sum:8:3);
    readln
end.
```

Пример 59. В той же задаче можно использовать другой цикл с условием:

```
program cycle_repeat;
var     x,y,sum:real;   otv:char;
begin
    sum:=0;
    repeat
        write('Введите числа x,y > 0 ');
        readln(x,y);
        writeln('Их произведение = ',x*y:8:3);
        sum:=sum+x+y;
        write('Завершить программу (Д/Н)? ');
        readln(otv);
    until (otv='Д') or (otv='д');
    writeln('Общая сумма = ',sum:8:3);
    readln
end.
```

Пример 60. Нахождение наибольшего общего делителя двух целых чисел с помощью Алгоритма Эвклида.

```
program Evklid;
```

```
var     a,b,c:integer;
begin
   write('введите два целых числа : ');
   readln(a,b);
   while b<>0 do
   begin
      c:=a mod b;
      a:=b;
      b:=c;
   end;
   writeln('наибольший общий делитель = ',a);
   readln
end.
```

2.7. Цикл с параметром. Вложенные циклы.

Цикл с параметром

Счетный оператор цикла **for** (цикл с параметром или цикл с известным числом повторений) имеет такую структуру:

for <параметр цикла> := <начальное значение> **to** <конечное значение> **do** <оператор>;

Здесь **for, to, do –** зарезервированные слова (для, до, выполнить);

<параметр цикла> - параметр цикла – переменная любого порядкового типа;

<начальное значение> - начальное значение – выражение того же типа;

<конечное значение> - конечное значение – выражение того же типа;

<оператор> - произвольный оператор Турбо Паскаля.

При выполнении оператора **for** вначале вычисляется выражение <начальное значение> и осуществляется присваивание <параметр цикла> := <начальное значение>. После этого циклически повторяется:

☑ проверка условия <параметр цикла> <= <конечное

☑ значение> и если условие не выполнено, оператор **for** завершает свою работу;
☑ выполнение оператора <оператор>;
☑ наращивание переменной <параметр цикла> на единицу.

Рассмотрите и повторите действие оператора цикла с параметром на примере 1.13.

Пример 61. С клавиатуры вводится произвольное число n. Вычислить сумму всех целых чисел от 1 до n.

```
program sum_of_integer;
uses crt;
var
        n, i, s :integer;
begin
  clrscr;
  writeln ('Введите число');
  readln (n);
  s:=0;
    for i:=1 to n do
        s:=s+i;
        writeln('Сумма равна ', s);
        readln
  end.
```

Блок-схема к программе изображена на рисунке 2.10.

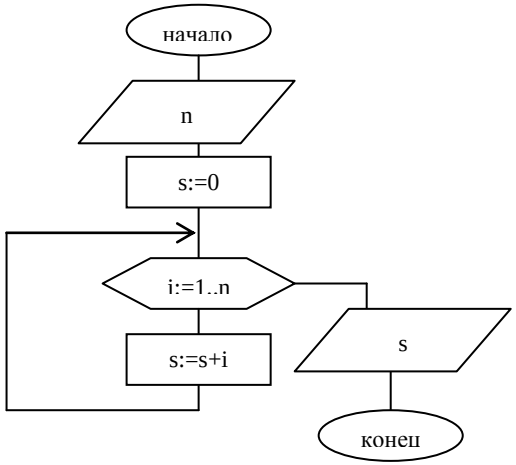


Рисунок 2.10. Блок-схема программы примера 61

Существует другая форма оператора цикла с параметром:

for <параметр цикла> := <начальное значение> **downto** <конечное значение> **do** <оператор>;

Это означает, что шаг наращивания параметра цикла равен (-1), а отсчет идет от большего к меньшему.

Пример 62. Рассмотрим пример 2.40., в котором n может быть и отрицательным числом:

```
s:=0;
if n>=0 then
for i:=1 to n do
        s:=s+i
else
for i:=-1 downto n do
        s:=s+i;
writeln('Сумма равна ', s);
……..
```

Пример 63. Дано натуральное n. Вывести сумму и значение каждой из дробей.

$$S = \frac{1}{3} - \frac{1}{3 \cdot 2} + \frac{1}{3 \cdot 3} - \frac{1}{3 \cdot 4} + \frac{1}{3 \cdot 5} - \ldots + \frac{(-1)^{n+1}}{3 \cdot n}$$

```
Program  pr;
uses crt;
var
        a, b, s: real;
        i, n: integer;
begin
        clrscr;
        writeln (‘Введите натуральное n’);
        readln (n);
        s:=0;
        for i:=1 to n do
        begin
                if odd(i) then b:=1
                else b:=-1;
                a:=b/(3*i);
                s:=s+a;
                writeln (a)
        end;
        writeln (‘s=’, s);
        readln
end.
```

Блок-схема программы изображена на рисунке 2.11.

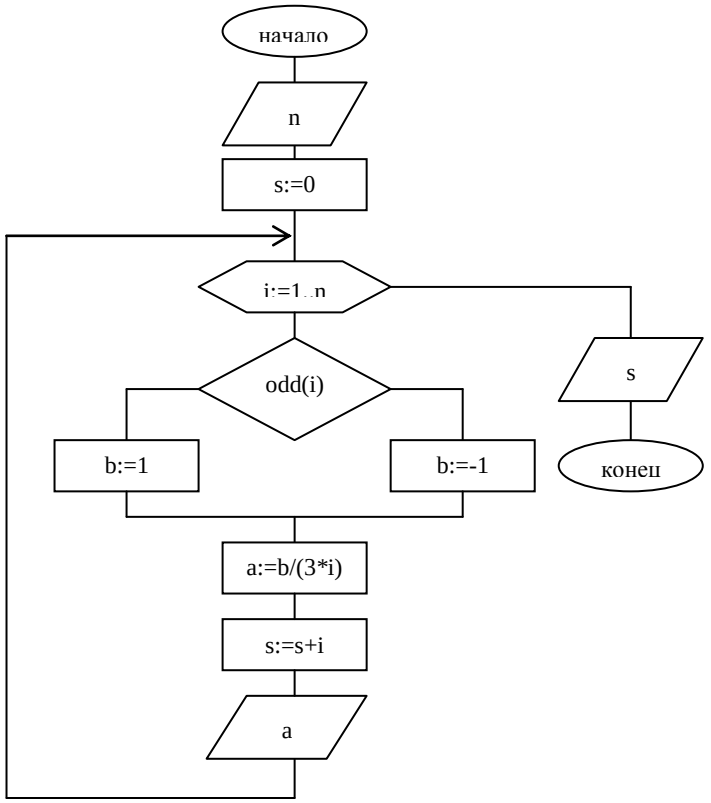


Рисунок 2.11. Блок-схема к программе примера 63

Еще несколько примеров на использование оператора цикла с параметром.

Пример 64. Вводятся 10 чисел, посчитать среди них количество положительных.

```
program cycle_for1;
var     i, kn:byte;       x:real;
begin
   kn:=0;
   for i:=1 to 10 do
   begin
      writeln('Введите ',i,' число: ');
```

```
        readln(x);
        if x>0 then kn:=kn+1 {увеличиваем количество на 1}
    end;
    writeln('Вы ввели ',kn,' положительных чисел.');
    readln
end.
```

Пример 65. Напечатать буквы от 'Z' до 'A'.

```
program cycle_for2;
var     c:char;
begin
    for c:='Z' downto 'A' do write(c);
    readln
end
```

Пример 66. Вычислить N-е число Фиббоначчи. Числа Фиббоначчи строятся следующим образом: F(0)=F(1)=1; F(i+1)=F(i)+F(i-1); для i>=1. Это пример вычислений по рекуррентным формулам.

```
program Fib;
var     a,b,c:word;     i,n:byte;
begin
    write('введите номер числа Фиббоначчи ');
    readln(N);
    a:=1; {a=F(0), a соответствует F(i-2)}
    b:=1; {b=F(1), b соответствует F(i-1)}
    for i:=2 to N do
    begin
        c:=a+b; {c соответствует F(i)}
        a:=b; b:=c; {в качестве a и b берется следующая пара чисел}
    end;
    writeln(N,'-е число Фиббоначчи =',b); {для N>=2 b=c}
    readln
end.
```

Вложенные циклы.

Циклы могут быть вложенными один в другой. Иначе их называют сложными циклами. При использовании вложенных циклов необходимо составлять программу таким образом, чтобы внутренний цикл полностью укладывался в циклическую часть внешнего цикла. Внутренний цикл может также в свою очередь содержать другой внутренний цикл.

При организации вложенных циклов необходимо учитывать следующее:

- ☑ имена параметров для циклов, вложенных один в другой, должны быть разными;
- ☑ внутренний цикл должен полностью входить во внешний.

Пример 67. Вычислить значение переменной y=2k+n при всех значениях переменных n=1, 2, 3 и k=2, 4, 6, 8. Обратим внимание на то, что если перебирать все значения n и k, получим 12 значений y.

```
program d2;
uses crt;
var n, k, y: integer;
begin
        clrscr;
        for n:=1 to 3 do
                begin
                        k:=2;
                        while k<=8 do
                                begin
                                        y:=2*k+n;
                                        writeln (n:4, k:4, y:4);
                                        k:=k+2
                                end
                end; end.
```

Обратите внимание, что оператор k:=2 находится внутри первого цикла. Подумайте, что произойдет, если поместить его перед первым циклом. Исполните программу.

Блок-схема к программе изображена на рисунке 2.12.

Следует обратить внимание на принцип действия вложенных циклов. Сначала проверяется условие первого цикла и один раз выполняются входящие в него операторы, затем полностью работает второй цикл столько раз, пока выполняется его условие, затем один раз выполняются операторы первого цикла, следующие после второго цикла. После этого второй раз проверяется условие первого цикла и выполняются один раз его операторы. Второй цикл снова выполняется столько раз, сколько необходимо, пока работает его условие и т.д. Т.е. внутри первого цикла вложенный цикл проходит все итерации, пока работает условие, а затем первый цикл повторяет свою работу.

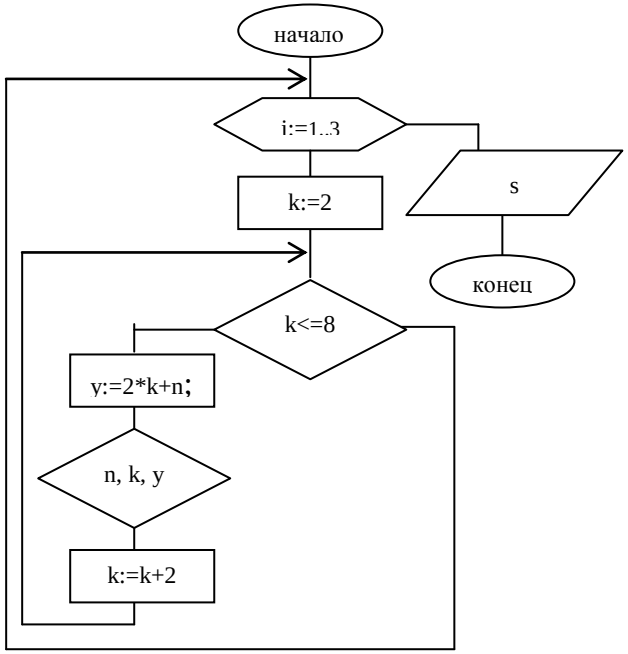


Рисунок 2.12. Блок-схема программы примера 67

ПРАКТИЧЕСКИЕ ЗАДАНИЯ

1. * Какие из следующих последовательностей символов являются идентификаторами языка Паскаль?
   a) x;
   b) z1;
   c) else;
   d) 1S;
   e) huy#;
   f) s-d/2;
   g) r_1;
   h) x1x2;
   i) 2a;
   j) abr;
   k) sin;
   l) cd 45;
   m) -t-8;
   n) Настя;
   o) div;
   p) max37.
2. * При исполнении программы на экран вывелись значения переменных A, D, G:
   a) A=-12691.5700225E-05;
   b) D=0.1230001258E+12;
   c) G=1.6658974511E+00.

   Что обозначает данная запись? Какого типа переменные A, D, G?
3. * Какого типа константы описаны в разделе констант следующей программы? Какие из данных записей являются ошибочными и почему?

```
program constant;
const
    a=2.8;
    b=25;
    c:=45.03;
    d=fgh;
    e=12E-1;
    f=$15;
    g=’Маша’;
    h=#125;
    i=true;
    j=12E-1-5;
    k:=false;
    l=true and false;
    m=#true;
```

n=#45#58#102;
o= ord('Z') - ord('a');
p=Max mod Size.

Исполните данную программу, добавив раздел операторов, в котором выведите все константы на экран.

4. * Пользуясь кодировочной таблицей из приложения Б, составьте из кодов нужных символов с предшествующими каждому коду символами # следующие строки:
   a) computer;
   b) компьютер;
   c) программа;
   d) Turbo Pascal;
   e) Visual Basic;
   f) Integer.
5. * Какие из следующих последовательностей символов являются выражениями, записанными по правилам Паскаля?
   a) k;
   b) ab;
   c) 8*|h|/2;
   d) -2*xy;
   e) sin(x)+abs(cos(1+x));
   f) a*(-4);
   g) f mod t div 4;
   h) $\frac{r}{2*f}$;
   i) a>2;
   j) not true and (3=5);
   k) x^8-y^7.
6. ** Записать по правилам Паскаля выражение:
   a) $\frac{a+b-1.7}{c+\frac{d}{e+f+0.5}}$;

b) $\sqrt{\sqrt{y}+|x|}$;

c) $\dfrac{a+\dfrac{b}{f}*\dfrac{1{,}5}{e}}{c+\dfrac{d}{2-\dfrac{5}{g}}}$;

d) $\sqrt{\sqrt{y+|x|*a}}$;

e) $\dfrac{(x+1)^3}{2x^2-1}$;

f) $2{,}136+\dfrac{2}{3}y$;

g) $3-\dfrac{2}{3y^5}$;

h) $\sin x^3 \sin^2 y+\sqrt{tg\,y}$.

7. ** Пусть значения переменных x и y равны, соответственно, 0,2 и -0,1. Какие значения будут иметь эти переменные после выполнения операторов присваивания:
   a) x:=x-2*y; y:=y/5;
   b) y:=-y; x:=x-y; y:=y-1;
   c) x:=5;
   d) x:=4*y-x; y:=y/5;
   e) x:=-y; y:=-x+y; y:=-y+1;
   f) y:=2;
   g) x:=2*x-4*y;
   h) y:=-x+y ?
8. * Какие из следующих последовательностей символов являются операторами присваивания?
   a) a:=-b;
   b) a=-c-1;
   c) a:b-sqrt(4);

d) z:=-z+5;
e) –y:=2*y;
f) a*6+8:=5.
g) l/2*x-8:=0;
h) –k:=l+1;
i) h:(-sqrt(5));
j) k:=k+1;
k) a:=h/2-5;
l) h=f*7.

9. * Какие из следующих последовательностей символов являются операторами вывода?
   a) writeln(x,y);
   b) write(x,x+1,x+2);
   c) read(a);
   d) write(100);
   e) read(h, b);
   f) write(1,23);
   g) writeln(a,b);
   h) writeln(x-5,2*x,x);
   i) print(a,k);
   j) print(y,2*z);
   k) writeln(23);
   l) write(sqr(x*3));
10. * Какие из следующих последовательностей символов являются операторами ввода?
   a) read(x,y,z);
   b) read x,y,z;
   c) readln(x);
   d) x:=read(x);
   e) readln(A,b);
   f) read(a, b+c).
   g) read a,b+2,c;
   h) read(a,sqr(b),c);
   i) readln(|a|);

j) a:=readln(b);
k) read(x, x+2*y);
l) read(x):=b.

11. ** Написать программу по следующему заданию. Вычислить площадь треугольника по формуле Герона $S = \sqrt{p(p-a)(p-b)(p-c)}$, где $p = \frac{a+b+c}{2}$, а также радиус описанной и радиус вписанной окружности по формулам $Rв = \frac{S}{p}$, $Ro = \frac{a*b*c}{4S}$. Нарисовать блок-схему к программе.
12. ** Написать программу по следующему заданию. Поменять местами две переменные целого типа. Нарисовать блок-схему к программе.
13. ** Написать программу по следующему заданию. Определить по коду символа сам символ. Нарисовать блок-схему к программе.
14. *** Написать программу по следующему заданию. Найти сумму цифр трехзначного целого числа. Нарисовать блок-схему к программе.

Указания: Представьте число как сумму разрядов, например, 231=2*100+3*10+1, воспользуйтесь функциями mod и div.

15. *** Написать программу по следующему заданию. Найти сумму цифр четырехзначного целого числа. Нарисовать блок-схему к программе.

Указания: Представьте число как сумму разрядов, например, 231=2*100+3*10+1, воспользуйтесь функциями mod и div.

16. ** Написать программу по следующему заданию. Использовать переменные, которые должен ввести пользователь, вместо конкретных данных для обеспечения требования массовости алгоритма. Для детского сада купили 172 альбома и 186 наборов цветных карандашей. Сколько стоит 1 альбом, если 1 набор карандашей – 60 тенге, а за всю покупку заплатили 24060 тенге. Нарисовать блок-схему к программе.

17. * Написать программу по следующему заданию. Использовать переменные, которые должен ввести пользователь, вместо конкретных данных для обеспечения
18. требования массовости алгоритма. Вычислите стоимость 5 коробок конфет, если в каждой коробке помещается 10 кг конфет, а стоимость одного килограмма 320 тенге. Нарисовать блок-схему к программе.
19. * Написать программу по следующему заданию. Даны два числа. Найти их сумму, разность, произведение и среднее арифметическое. Нарисовать блок-схему к программе.

К следующим заданиям (19-38) написать программу. Не использовать операторы условий и циклов! Обеспечить требование массовости алгоритма.

20. ** Компьютер генерирует целое число от 0 до 255 и выдает символ, соответствующий данному коду в кодировочной таблице.
21. ** Компьютер генерирует целое число из интервала [-100, 100]. Выдать, четное данное число или нечетное. Выдать остаток от деления данного числа на 7. Найти целую часть от деления данного числа на 9. Нарисовать блок-схему к программе.
22. ** Компьютер генерирует действительное число из интервала [-55, -27). Найти модуль данного числа. Выдать целую часть числа. Выдать дробную часть числа. Округлить данное число до сотых.
23. ** Компьютер генерирует целое число в диапазоне [-24, 36]. Выдать число, следующее за ним и предшествующее ему. Возвести данное число в целую положительную степень. Учесть случай, когда число отрицательное.
24. *** Цех выпускает 5 наименований продукции. Число единиц каждой продукции от 70 до 100 стоимостью от 200 до

2500 тенге каждое наименование. Вероятность того, что данная единица продукции реализована – 80%. Найти общую сумму выручки.

Указания: Для задания данных из диапазона используйте генерацию случайных чисел. Вероятность используйте как расчет процента от числа.

25. *** Сколько необходимо затратить времени, чтобы собрать 1 килограмм икры. Для того чтобы поймать одну рыбу необходимо затратить от 5 до 15 минут. Вероятность того, что рыба с икрой – 30%. В одной рыбе может быть от 30 до 70 грамм икры.

Указания: Для задания данных из диапазона используйте генерацию случайных чисел. Вероятность используйте как расчет процента от числа.

26. *** Бригаде рабочих необходимо выполнить объем работ, составляющий A единиц продукции. Число рабочих в бригаде N человек. Производительность каждого рабочего 10-20 единиц в день. Сколько дней необходимо на выполнение всего объема работ. Величины A и N вводятся в режиме диалога.

Указания: Для задания данных из диапазона используйте генерацию случайных чисел.

27. ** Сколько мешков, вмещающих по 4 ведра картофеля, необходимо взять на уборку урожая, всего посажено 80 лунок, в каждой в среднем 9 картофелин, а в ведро входит 30 картофелин? Нарисовать блок-схему к программе.
28. ** Программист в месяц получает 40 тысяч тенге. Из них 5 тысяч он тратит на личные расходы, жене отдает 33 тысячи, а остальные делит поровну на 4-х детей. Сколько получает в месяц на мелкие расходы ребенок программиста? Нарисовать блок-схему к программе.
29. ** Расстояние до ближайшей к Земле звезды Альфа Центавра 4,3 световых года. Скорость света принять 300 000 км/с. Скорость земного звездолета 100 км/с. За сколько лет звездолет долетит до звезды?
30. ** На заводе 35% всех рабочих – женщины, а остальные –

мужчины, которых на заводе на 252 человека больше чем женщин. Определить общее число рабочих.

31. ** При продаже товара за 1386 руб. получено 10% прибыли. Определить себестоимость товара. Нарисовать блок-схему к программе.
32. ** Производственная артель, продав продукции на 3348 руб., понесла 4% убытку. Какова стоимость этой продукции?
33. ** Если на 225 кг руды получается 34,2 кг меди, то каково процентное содержание меди в руде?
34. ** Для приготовления пшенной каши необходимы следующие продукты: масло, пшено и молоко в
35. соотношении 1:3:9. Каши получилось 2 кг. Сколько масла, пшена и молока ушло на ее приготовление?
36. ** Огород в 8 га засеян на 12% картофелем, 1/4 составляет морковь, 1/7 засеяна луком. Сколько га земли осталось не засеянной? Нарисовать блок-схему к программе.

$$x = \frac{a+2b}{5a+4}\sqrt{5(a+2b)}\frac{1}{a^2+2\frac{1}{a}}$$

37. ** Вычислить:

$$x = \sqrt{7\left(a+5b^2\right)}\frac{2a-b}{4a+b}+2a-b$$

38. ** Вычислить:

$$x = \frac{2(a+2b)}{a+\dfrac{a+2b}{a+\dfrac{a+2b}{\sqrt{a^2+b^2}}}}$$

39. ** Вычислить:

$$x = \frac{(a+b)^2}{a+\dfrac{\sqrt{a+b}}{a^2+b^2}}+3(a+b)$$

40. ** Вычислить:

$$x = \sqrt{2(a+b)^2 + a} + \frac{a}{a + \frac{a}{a+b}}$$

41. ** Вычислить:
42. *** Дано целое четырехзначное число. Программа выдает цифру, стоящую в записи числа на втором месте.

Указания: Представьте число как сумму разрядов, например, 231=2*100+3*10+1, воспользуйтесь функциями mod и div.

43. *** Дано целое трехзначное число. Программа выдает цифру, стоящую в записи числа на втором месте. Нарисовать блок-схему к программе.

Указания: Представьте число как сумму разрядов, например, 231=2*100+3*10+1, воспользуйтесь функциями mod и div.

44. *** Поменять местами
    a) третью и четвертую цифры восьмизначного числа.
    b) вторую и третью цифры восьмизначного целого числа
    c) третью и пятую цифры семизначного целого числа.

Указания: Представьте число как сумму разрядов, например, 231=2*100+3*10+1, воспользуйтесь функциями mod и div.

45. ** Вычислить количество прожитых составителем программы дней. Учесть, что в високосном году 366 дней.
46. **** Известна теория биоритмов. С момента рождения жизнь человека подчиняется трем синусоидальным биоритмам. Физический цикл — 23 дня, эмоциональный — 28 дней и интеллектуальный — 33 дня. Первая половина каждого цикла — положительная, вторая — отрицательная. При переходе от положительной к отрицательной фазе в каждом цикле есть так называемый критический день. В физическом цикле — 12-й день, в эмоциональном — 15-й день, в интеллектуальном — 17-й день. Когда два и более цикла находятся в отрицательной фазе, или в критических днях, то в такие дни повышена вероятность физических недомоганий и травм, эмоциональных срывов и ссор, замедленности интеллектуальных процессов и реакции. В Японии, например, в крупных фирмах для каждого работника составлен график и в "плохие" дни их не допускают до работы, дают отдыхать, потому что оплата

больничного или брак в работе обойдутся фирме дороже. Теперь к делу. Опираясь на результат предыдущего задания и используя операцию нахождения целочисленного остатка, рассчитайте, каков для вас сегодняшний день по всем трем циклам.

47. ** Запросите у пользователя валютный курс на сегодняшний день, затем имеющуюся у него рублевую сумму и рассчитайте, сколько долларов он может купить.80. Дискета 3,5" вмешает 1,44 Мбайт. Рукопись содержит 450 страниц текста. На каждой странице 60 строк по 80 символов в каждой. Поместится ли рукопись на дискету? Если нет, то сколько таких дискет потребуется?
48. ** Документ содержит текст из 32 строк по 60 символов в каждой и точечную черно-белую фотографию 10x15 см. Каждый квадратный сантиметр содержит 300 точек, любая точка описывается 4-мя битами. Каков общий информационный объем документа в Кбайтах?
49. *** Запросите у пользователя длину ребра куба. Найти площадь грани, площадь полной поверхности и объем этого куба. Нарисовать блок-схему к программе.

Указания: Воспользуйтесь формулами из учебника геометрии.

50. **** Есть притча о шахматах, где выигравший запросил у могущественного правителя, чтобы ему был выплачен выигрыш зерном пшеницы по следующим правилам: на первую клетку шахматной доски положить одно зерно, на вторую — два зерна, на третью — четыре, на четвертую — восемь и т. д., иными словами, на каждую последующую в два раза больше зерен, чем на предыдущую. Сколько же зерен должен был бы получить выигравший? (Замечание: на каком-то этапе выполнения этого задания возможно переполнение памяти. Подумайте, как обойти это препятствие.)

51. **** Кстати, для оценки предыдущего результата еще одно задание. В России ежегодно собирают около 90 млн тонн зерновых. Масса одного зерна около 5 грамм. Сколько зерен в таком урожае, и сколько лет пришлось бы расплачиваться России по условиям предыдущего задания?
52. **** Продав квартиру, вы получили $ 22 000 и положили их в банк. Банк начисляет 1% в первый месяц, а каждый следующий — тоже 1%, но уже с получившейся суммы. Сколько денег будет в банке на вашем счету через год? (Нестабильностью нашей экономической системы пренебрегаем)
53. ** Допустим, вы получили наследство $ 1 000 000 и хотите красиво пожить. После долгих раздумий вы решаете, что будете жить на $ 800 в месяц. На сколько лет вам хватит наследства?
54. *** Пушка стреляет под углом 30° к линии горизонта. Масса снаряда 30 кг, начальная скорость 500 м/с. Какова будет дальность полета снаряда? (Формулу вспомните из курса физики.)

Указания: Воспользуйтесь формулами из учебника физики.

55. * Использовать в программе различные функции для работы с переменными целого, символьного, порядкового, логического, вещественного типов. Организовать ввод всех переменных, кроме логических, с пояснением.

    Пример:

    writeln(‘Введите целое число’);
    readln(x);
    Результаты вывести с пояснениями.

    Пример:

    k:=sqrt(m);
    writeln(‘Значение квадратного корня из числа ’,m,’ равно ‘,k);
56. * Использовать следующие функции: abs, chr, odd, div, mod, ORD, PRED, SUCC, Abs, frac, int, trunc, round, Not, And, OR.

57. * Использовать в программе оператор генерации случайных чисел с различными параметрами. Исполнить данную программу несколько раз и посмотреть результат.

Пример:

```
program f;
var x: integer;
        y: real;
begin
        randomize;
        x:=random(10)-5;
        y:=random*10-5;
        writeln('Случайное целое число из диапазона от -5
до 5 равно ', x);
        writeln('Случайное действительное число из
диапазона от -5 до 5 равно', y);
        readln
end.
```

58. * Выполните следующую программу:

Пример:

```
program dialog;
uses crt;
var
   a:string;
   b:real;
begin
        clrscr;
        writeln('Здравствуйте, как Вас зовут?');
        readln(a);
        writeln('Очень приятно,',' ',a,' сколько Вам лет?');
        readln(b);
        writeln('А я Вас старше в два раза , мне ',b*2, '
лет');
        readln
end.
```

Продолжить диалог программы с пользователем. Придумайте вопросы и ответы самостоятельно.

59. * Выдать 7 символов, которых нет на клавиатуре. Использовать функцию chr. Смотрите кодировочную таблицу в приложении «Таблица кодов ASCII».
60. * Автобус за смену делает от 5 до 15 рейсов. За рейс перевозит от 30 до 230 пассажиров. Стоимость проезда одного пассажира 25 тенге. Определить доход, полученный за N смен. Число смен и стоимость билета вводится в режиме диалога. Нарисовать блок-схему к программе.
61. ** Найти общий расход топлива N автоколонн за неделю. Известно, что каждая автоколонна состоит из 20 до 100 автомобилей. Каждый автомобиль за один день проезжает путь длиной от 5 до 150 километров. Расход топлива одного автомобиля от 20 до 28 литров на 100 километров.
62. ** а) По данным трем числам от 1 до 255 вывести соответствующие этим номерам символы. При этом данные числа описать как тип-диапазон. б) Описать перечисляемый тип. Вывести на экран номер предшествующего элемента данному и последующего какому-либо элементу. Примечание: Обратите внимание на то, что данные перечисляемого типа нельзя использовать в операторах ввода-вывода, а можно только их номера. в) Оформите красочно Вашу программу, используя всевозможные цвета для текста. Для этого в Паскале существует оператор TEXTCOLOR(n), где n – номер цвета от 0 (черный) до 15 (белый). Данный оператор пишется перед выводимой строкой и выполняется для всего текста, расположенного ниже. Если вместо n Вы укажете номер цвета, затем знак «+», а затем «128», то текстовая строка будет еще и мерцать. Таким же образом можно установить цвет фона – оператор TEXTBACKGROUND(n). Все перечисленные операторы и их использование смотрите в приложении Г.
63. ** Укажите ошибки в программах:

    а) **program** m5;

var

```
        f,g:real;
        d:integer;
begin
        d:=f mod d;
        d:=f div d
end.
```

b)
```
program m1;
var
        a:integer;
        b:real;
begin
        b:=read(a)
end.
```

c)
```
program m;
const d=Маша;
var
        a:char;
begin
        a:=d
end.
```

d)
```
program m123;
var
        a, b:char;
begin
        b:=ord(a)
end;
```

e)
```
program m12;
var
        a:real;
        b:integer;
begin
        a:=odd(b)
```

```
end.
```

f)
```
program ttt;
var
    a,c:char;
    b:Boolean;
begin
    readln(a, c);
    b:=a=c;
    writeln(b)
end.
```

g)
```
program ttt;
var
    a,c:char;
    b:Boolean;
begin
    readln(b);
    b:=a=c;
    writeln(b)
end.
```

64. * Найти значения выражений:
    a) $\dfrac{a}{b*\dfrac{c}{d*\dfrac{e}{f*h}}}$.
    b) $\dfrac{1-\sqrt{1+\left|\sin|x|\right|}}{2}$
65. ** Дано x. Получить значение $2x^4-3x^3+4x^2-5x+6$. Позаботиться об экономии операций.
66. ** Дано a. Не используя никаких функций и никаких операций кроме умножения, получить:
    a) a^8 за три операции;
    b) a^{10} за четыре операции;
    c) a^{15} за пять операций.
67. ** Даны два числа. Вывести их на экран, а затем поменять местами и вывести заново на экран.

68. *** Написать программу, в результате которой человек задумывает число, производит над ним предложенные компьютером действия, вводит в ПК результат, а компьютер угадывает задуманное число и выводит его на экран. Диалог с пользователем выводится на экран в начале исполнения программы следующим образом:
    a) Здравствуйте! Как Вас зовут?
    b) Очень приятно, {имя}!
    c) Задумайте число.
    d) Умножьте его на 3.
    e) Вычтите результат из 100.
    f) Прибавьте задуманное число.
    g) Умножьте результат на 2.
    h) Напишите ответ.
    i) Ваше задуманное число {выводится число на экран}.

    Нарисовать блок-схему к программе.

    Указания: В данном примере задуманное число выступает как переменная x. Прочитайте внимательно задания и составьте уравнение с использованием неизвестного x, в правой части которого будет переменная y, обозначающая ответ пользователя. Выразите переменную x через y.
69. *** Решить предыдущую задачу. Диалог с пользователем осуществить следующим образом:
    a) Здравствуйте! Как Вас зовут?
    b) Очень приятно, {имя}!
    c) Задумайте число.
    d) Прибавьте к нему 5.
    e) Умножьте результат на 3.
    f) Вычтите ответ из 80.
    g) К полученному результату прибавьте 2 задуманных числа.
    h) Умножьте результат на 2.

i) Напишите ответ.
j) Ваше задуманное число {выводится число на экран}.

70. * В каком диапазоне и какого типа сгенерируются и запишутся в переменную x числа после выполнения оператора:
    a) x:=Random(8);
    b) x:=Random*5;
    c) x:=Random*7+2;
    d) x:=Random(11)-10;
    e) x:=Random(2)-5;
    f) x:=Random*4-6.
    g) x:=Random*2;
    h) x:=Random(9);
    i) x:=Random*5-10;
    j) x:=Random*8+2;
    k) x:=Random(7)-3;
    l) x:=Random(6)-8.
71. * Записать с помощью вызова функции random генерацию:
    a) Действительного числа в диапазоне от -1 до 5;
    b) Целого числа в диапазоне от 5 до 8;
    c) Действительного числа в диапазоне от -5 до 1;
    d) Действительного числа в диапазоне от -4 до 8;
    e) Целого числа в диапазоне от 2 до 7;
    f) Действительного числа в диапазоне от -9 до 3;
    g) Целого числа в диапазоне от 2 до 10.
72. * Компьютер задумывает целое число в диапазоне от 0 до 255. Выдать 3 символа: первый – символ, номер которого в кодировочной таблице равен задуманному компьютером числу, второй – предшествующий ему символ, третий – следующий за ним символ.
73. ** Использовать в программе все изученные функции и процедуры для работы с символьными, логическими данными. Организовать при этом ввод и вывод переменных и констант данных типов.
74. ** Использовать в программе все изученные функции и процедуры для работы с данными перечисляемого типа и

типа-диапазона. Организовать при этом ввод и вывод переменных и констант данных типов.

75. ** Написать программу, которая двумя способами вычисляет идеальный вес человека.
    a) Один из самых простых способов – подсчет по формуле Брока. Нужно отнять от роста в сантиметрах 110 единиц (если вам от 18 до 45). Полученное число и будет идеальным весом. Например, если ваш рост 170 см, то идеальный вес – 60 кг. Вполне допустимы колебания вокруг этой цифры в пределах 10% (в приведенном примере можно иметь от 54 до 66 кг). Если результат превышает идеальный более чем на 15% - у человека наблюдается избыточный вес.
    b) Существует более сложная, но и более точная формула подсчета идеального веса – индекс массы тела (ИМТ). Рассчитывается он так: вес тела в килограммах нужно разделить на рост в метрах, возведенный в квадрат. Например, ваш вес равен 60 кг, рост 170 см (или 1,7м). ИМТ = 60:(1,7*1,7) = 20,8. Если ваш ИМТ меньше 19 – то у вас сниженный вес, от 19 до 24,9 – соответствует нормальному весу, от 25 до 29,9 – избыточный вес, от 30 до 39,9 – свидетельствует о болезненном ожирении, свыше 40 – наблюдается сверхожирение.
76. * Является ли оператором условия следующая последовательность символов?
    a) if x<y then x:=0 else y:=0;
    b) if x>y then x:=0 else read(y);
    c) if x>=y then x:=0; y:=0 else write(z);
    d) if a<b then 100 else z:=5;
    e) if a<b<c then z:=z+1;
    f) if sqrt(z)<=3.17 then z:=z+1;
    g) if a<>b then z:=z+1;a:=a+b.

h) if x>=y then x:=5; y:=10 else write(x);
i) if x>=y then x:=0 else read(z);
j) if a<b<=c then z:=a+1;
k) if a<=b then 12 else z:=15;
l) if x<y then x:=5 else y:=8;
m) if sqr(y)<=3.7 thenyy y:=y+1;
n) if y<>x then x:=x+1;y:=y+x.

77. * Можно ли между if и then разместить условие:
a) (x<y) and (y<z);
b) x<y;
c) x<z or z>y;
d) not(x>1) or (x<=y) and (x<z);
e) not(x);
f) x=y;
g) a;
h) c:=4;
i) c<>4;
j) 15<n<=25.

78. ** Для каких значений переменной x типа real соблюдаются условия:
a) (abs(x)>2) and (abs(x)<5);
b) (abs(x)>2) or (abs(x)<5);
c) (abs(x)>=2) or (abs(x)<=5).

79. ** Чем отличаются операторы? Что выйдет на экран при n=5, n=12?

a)
```
if (n<8) and (n>1) then
    writeln('*');
    writeln('***');
```

b)
```
if (n<8) and (n>1) then
begin
    writeln('*');
    writeln('***');
end
else
    writeln('**');
writeln('****');
```

c) **if** (n<8) **and** (n>1) **then**

```
begin
    writeln('*');
    writeln('***');
end
else
begin
    writeln('**');
    writeln('****');
end.
```

80. ** По номеру месяца определить, сколько в нем дней. В случае неправильного ввода номера месяца, сообщаем пользователю об ошибке и просим ввести номер месяца заново. При этом вам пригодятся следующие данные:
    a) в январе, мае, июле, августе, октябре, декабре 31 день;
    b) в апреле, июне, сентябре, ноябре 30 дней;
    c) спросите пользователя, високосный нынче год или нет. Если год високосный, то в феврале 29 дней, а если нет – 28. Нарисовать блок-схему к программе.

Указания: Для задания високосного года используйте оператор условия, вложенный в оператор выбора.

Смотрите примеры 44-51

81. ** Реализовать программу «Электронная гадалка», используя оператор выбора. В начале программы выводим на экран: «Сосредоточьтесь и загадайте желание», задерживаем экран, а затем выводим случайным образом одно из пяти сообщений, например: «Вас ждет большая удача!», «Сегодня вы найдете кучу денег!», «Вы встретите мужчину (женщину) своей мечты!!!» и т.п. Цвет сообщений также меняется случайным образом. Нарисовать блок-схему к программе.

Указания: Для вывода текста случайным цветом перед оператором вывода используйте процедуру:

textcolor(n) – процедура, которая задает цвет выводимых на экран символов, здесь n-целое число от 0 до 15, 0 – черный чвет, 15 – белый цвет. Пусть n будет случайное число.

82. ** Найти значение функции:

a) $$y = \begin{cases} 1, & \textit{если } x = 3 \textit{ и } a \prec 6; \\ 2, & \textit{если } x = 3 \textit{ и } a \geq 6; \\ 3, & \textit{если } x \prec\succ 3. \end{cases}$$;

b) $$y = \begin{cases} 1, & \textit{если } x = 7; \\ 2, & \textit{если } x \prec\succ 7 \textit{ и } a \succ 3; \\ 3, & \textit{если } x \prec\succ 7 \textit{ и } a \leq 3. \end{cases}$$

Нарисовать блок-схему к программе.

Смотрите примеры 44.

83. ** Решить квадратное уравнение. Учесть случаи, когда получается один корень, два корня и когда нет корней. Нарисовать блок-схему к программе.
84. ** Найти корень квадратный из произвольного числа. Учесть случай, когда вводится отрицательное число. Т.е. если x<0, то пишем: «Корень из отрицательного числа вычислять отказываюсь!» Нарисовать блок-схему к программе.
85. ** Программа запрашивает имя у пользователя, затем здоровается, спрашивает возраст и предлагает: «Не хотите ли закурить? (Y/N)». Если ответ – N, то выходим из программы. Если ответ утвердительный и возраст пользователя <18, то пишем: «Вам, {имя}, рановато курить!». Если возраст>=18, то пишем: «Минздрав предупреждает, курение опасно для Вашего здоровья!». Нарисовать блок-схему к программе.
86. ** В Преображенский полк гренадеров набирали только из тех новобранцев, рост которых был не менее 180 см, а вес – не менее 80 кг. Определить, годится ли пользователь в гренадеры. (Спросить сначала имя). Если человек годится, то пишем: «Вы, {имя}, - достойный кандидат в гренадеры!». Если не годится: «{имя}, к сожалению, Вы в гренадеры не годитесь!». Нарисовать блок-схему к программе.

87. ** Даны два числа. Расположить их в порядке возрастания. Нарисовать блок-схему к программе.
88. ** Программа спрашивает пользователя о сегодняшней погоде, предлагая варианты:
    a) Идет дождь.
    b) Идет снег.
    c) Солнечно, без осадков.

Если пользователь выбирает первый вариант, пишем: «Берите зонт и надевайте плащ». Если второй вариант, то пишем: «Надевайте валенки и шубу». Если третий: «Берите панаму и ничего не надевайте!». В любом случае в конце написать: «Улыбайтесь при любой погоде!» Нарисовать блок-схему к программе.

89. * Написать программу, в которой пользователю предлагается выбор, в какую сторону пойти: направо, налево или прямо. В зависимости от выбора на экране появляется одна из надписей: «Направо пойдешь – здоровье найдешь, налево пойдешь – богатство найдешь, прямо пойдешь – счастье найдешь». В случае неправильно веденного варианта выдать сообщение: «Если пойдешь по этой дороге – не поздоровится, выбери другой путь!» и программа возвращается к выбору одного из прежних трех вариантов. Нарисовать блок-схему к программе.
90. *** Написать программу «Электронная гадалка». Сначала на экран выводится сообщение: «Здравствуйте, хотите узнать, что Вас ждет в ближайшее будущее? (Y/N)» Если ответ положительный, выводится следующий текст: «Сосредоточьтесь и загадайте желание!». А затем случайным образом выдается одно из пяти сообщений, например: «Ваше желание исполнится, но для этого Вам необходимо сегодня признаться в любви девушке (парню), который сейчас находится ближе всех к Вам», «Ваше желание не исполнится,

но вас ждет удача в любви». и т.п. Если ответ отрицательный, то выходим из программы. Нарисовать блок-схему к программе.

91. **** В восточных календарях принят 60-летний цикл, состоящий, в свою очередь, из пяти 12-летних подциклов. Подциклы обозначались цветом: зеленый, красный, желтый, белый и черный. Внутри каждого подцикла годы носили названия животных: крысы, быка, тигра, кролика (зайца или кота), дракона, змеи, лошади, овцы (барана или козы), обезьяны, петуха, собаки и свиньи. Создайте программу с использованием операторов выбора, запрашивающую номер года нашей эры и печатающую его название по восточному календарю. Для проверки: 1966 г. — год красной лошади, 1984 г. — зеленой крысы.
92. **** Усложненный вариант предыдущего задания. Теперь требуется определить не только название введенного с клавиатуры года, но и год наступления "светлого будущего", до которого вы, безусловно, доживете! Делаем это поэтапно. Вот примерный словесный алгоритм. После введения исходного четырехзначного года с клавиатуры вам необходимо выделить составляющие его цифры и записать их в переменные A1, A2, аз и A4. (Если в номере года есть нули, то в соответствующие переменные записать 4.) Затем необходимо взять от них синусы по модулю. Из полученных значений выделить по две цифры после запятой и сложить их. Если сумма больше либо равна 10, то сложить еще раз. Должно получиться 4 цифры, из которых надо составить четырехзначное число. Это и будет номер года наступления светлого будущего. Осталось определить его название по восточному календарю и вывести результаты на экран. Для проверки: при исходном годе 1986 годом светлого будущего будет 3589.
93. * Напишите программу, определяющую четность или нечетность введенного с клавиатуры целого числа. Нарисовать блок-схему к программе.

94. * Напишите программу, находящую меньшее из двух, введенных с клавиатуры чисел. Нарисовать блок-схему к программе.
95. ** По четырехзначному номеру года, запрошенному с клавиатуры определите номер столетия (например, для 1492 г. — ответ XV век, для 1812 г. — XIX век). Учесть, что началом века считается первый, а не нулевой, год. (То есть, 2000-й год, из астрономии — последний год XX века).
96. *** Запишите в виде одного условного оператора указанные действия: Известно, что из трех чисел a1, a2 и a3 одно отлично от других, равных между собой. Присвоить номер этого числа переменной n. *Смотрите примеры 44-48*
97. **** По введенным с клавиатуры коэффициентам квадратного уравнения A, B и C найдите его корни. Рассмотрите шесть возможных вариантов:
    a) a=в=c=0, корней бесчисленное множество (X — любое);
    b) a=в=0, c<>0, уравнение не имеет корней;
    c) a=0, в<>0, c<>0, вырожденное квадратное уравнение, имеется один корень (формулу вычисления корня найдите сами);
    d) d<0, где d — дискриминант, который предварительно надо вычислить; уравнение не имеет вещественных корней;
    e) d=0, уравнение имеет два одинаковых корня (вывести их значения);
    f) d>0, уравнение имеет два различных вещественных корня (вычислить и вывести их значения).

    Для проверки правильности работы программы предлагается шесть тестовых вариантов исходных данных:
    a) a=в=c=0;
    b) a=в=o, c=i;
    c) a=o, в=з, c=б (должно получиться x=-2);
    d) A=5, B=3, C=2;

e) a=i, в=2, c=1 (должно получиться xi=x2=-i);
f) A=2, B=5, C=2 (должно получиться X1=-2, X2=-8).

Смотрите примеры 44-48.

98. ** Осуществите запрос трех целых различных чисел с клавиатуры. Выведите на экран наибольшее и наименьшее.
99. **** Осуществите запрос с клавиатуры у тренера олимпийской сборной Казахстана по марафону результаты в часах, минутах и секундах трех победителей чемпионата Казахстана. Если какие-то два результата различаются менее чем на 5 сек, выведите сообщение "Вот так шла борьба за _____ медаль". В сообщении должно быть указано достоинство медали (золотая, серебряная), за которую шла борьба. В противном случае, вычислить среднюю скорость победителя в км/час, если принять длину марафонской дистанции 42 км 195 м.

Смотрите примеры 50-51.

100. **** А сейчас мы попробуем сделать пока не очень красивый, но очень простой вариант телевизионной игры "О, счастливчик!". Придумайте пять любых вопросов, и к каждому из них четыре варианта ответов. Теперь я попробую словесно описать алгоритм, а вы — перевести его на Бейсик. Итак, запрашиваем у игрока имя, и узнаем, желает ли он играть. Если не желает, прощаемся, если желает — приветствуем и предлагаем первый вопрос с вариантами ответов. Запрашиваем у игрока с клавиатуры, какой вариант он выбирает. В случае правильного ответа начисляем ему сто очков и переходим ко второму вопросу. Если ответ неверен, то отправляемся на самое начало — снова регистрация и т. д. Первый вопрос — 100 очков, второй — 200, третий — 300, четвертый — 500, пятый — 1000. Если игрок правильно отвечает на все пять вопросов, то поздравляем его и заканчиваем программу.
101. **** Каждую пятницу члены Клуба толстяков выстраиваются в определенном порядке и взвешиваются. Напишите программу, которая хранит данные взвешивания 10-ти членов Клуба за прошлую неделю. Затем программа

запрашивает новые данные взвешивания и для каждого члена Клуба либо выводит поздравление в случае похудения, либо величину прибавки веса с сожалением.

102. *** Напишите программу, запрашивающую у пользователя три разных целых положительных числа и находящую сумму двух наименьших из них. Нарисовать блок-схему к программе.

Смотрите примеры 44-48.

103. ** Записать результат исполнения следующих операторов:

```
x:=1;
l:write(x);
x:=x+2;
if x<=13 then goto l;
```

Выполните данную программу на компьютере, дополнив ее разделом описания

104. ** Записать результат исполнения следующих операторов:

```
n:=1;
l:if n<=80 then
begin
        textcolor(random(15)+1);
        write('*');
        delay(1000);
        delline;
        n:=n+1;
        gotoxy(n,1);
        goto l;
end;
```

Выполните данную программу на компьютере, дополнив ее разделом описания.

105. ** Записать результат исполнения следующих операторов:

```
y:=3;
l:if y<12 then
begin
        writeln((y-2)*5);
        y:=y+3;
        goto l;
end;
```

Выполните данную программу на компьютере, дополнив ее разделом описания

106. ** Записать результат исполнения следующих операторов:

```
n:=1; p:=1;
 l:p:=p*n;
 n:=n+1;
if n<=5 then goto l;
writeln(p);
```

Выполните данную программу на компьютере, дополнив ее разделом описания

107. * Может ли программа в качестве описания меток содержать:

a) label a; b; c;
b) label a, b;
c) label 2, -3;
d) label x;
e) label 0, 1.2, 3.

108. * Определить результат исполнения следующих операторов:

```
y:=2;
case y of
    1:write(*);
    2:write(***);
    3:write(**)
end;
```

Выполнить данную программу на компьютере, дополнив ее разделом описания.

109. ** Используя пример 47, определить, попадет ли точка а с координатами ха и уа внутрь кольца, изображенного на рисунке (рисунок 2.13). Радиусы окружностей задает пользователь. Нарисовать блок-схему к программе.

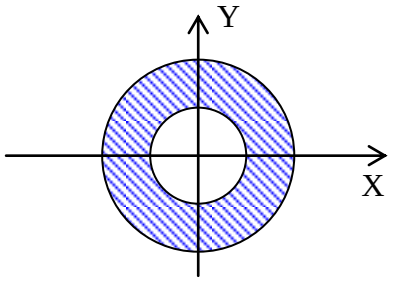


Рисунок 2.13. Иллюстрация к задаче 66

Продумайте неправильный ввод пользователя. Например, в данной программе пользователь может ввести значение большего радиуса меньше значения меньшего, что является ошибкой. Такую ошибку можно предотвратить, например, оператором цикла с постусловием:

```
Writeln('Введите 2 радиуса:');
Repeat
    If r1>r2 then writeln('Первый радиус должен быть меньше второго');
    Readln(r1, r2);
Until r1>r2;
```

110. *** Написать программу, имитирующую работу калькулятора. Запрашиваем два числа и просим ввести одно из четырех действий: «+», «-», «*», «:», а затем выполняем это действие над данными числами. В случае неправильно набранного пользователем действия возвращаемся в начало программы и просим заново ввести числа. Нарисовать блок-схему к программе.

Указания: Здесь параметром выбора будет переменная символьного типа, в которую будут записаны знаки действий. Использовать операторы выбора, условия и безусловного перехода или цикла.

111. *** Программа запрашивает ввести день недели, а затем выводит одно из семи сообщений, например: «Понедельник – день тяжелый», или «Ура! Пятница! Это надо отметить!» и т.п. В случае неправильно набранного дня недели, выдается сообщение: «Нет такого дня недели!», после чего просим пользователя правильно ввести данные. Нарисовать блок-схему к программе.

Указания: Используйте операторы выбора, оператор безусловного перехода или оператор цикла.

2. *** Напишите программу, которая выводит на экран меню, содержащее список четырех поэтов под номерами, затем запрашивает у пользователя номер поэта и выводит на экран какую-либо строчку из его стихотворения. Предусмотреть ошибку выбора пользователя, т.е. если пользователь нажимает не ту цифру, программа просит ввести номер поэта заново. Нарисовать блок-схему к программе.

Указания: Используйте операторы выбора, оператор безусловного перехода или оператор цикла.

112. *** Программа запрашивает уровень риска (число от 1 до 5). Если риск равен пяти, то выводим сообщение: «Максимальный риск. Шансов на возвращение практически нет. Пишите завещание». Если уровень риска от двух до четырех, то пишем: «Высокий риск. Шансов на возвращение немного. Проверьте снаряжение, оружие и припасы». В остальных случаях выводим: «Риск отсутствует, возврат гарантирован. В вас отсутствует дух авантюризма. Это слишком скучно». Предусмотреть ошибку ввода пользователем уровня риска. Т.е., если вводится буква или цифра, не входящая в заданный диапазон, то выводим сообщение: «Такого уровня риска нет» и просим заново ввести цифру от 1 до 5. Нарисовать блок-схему к программе.

Указания: Используйте операторы выбора, оператор безусловного перехода или оператор цикла.

113. *** В память компьютера вводится целое число. Найти произведение всех целых чисел от единицы до введенного. Нарисовать блок-схему к программе.

Указания: Можно использовать оператор безусловного перехода или оператор цикла.

a) Запрашиваем у пользователя число;
b) Произведению присваиваем единицу;
c) Включаем счетчик (присваиваем новой переменной значение 1, далее оно будет возрастать на единицу, пока не дойдем до введенного числа);
d) Если счетчик больше введенного числа, уходим в конец программы;
e) Наращиваем произведение;
f) Наращиваем счетчик;
g) Переходим к пункту 4.

114. *** Не используя операторов цикла, написать программу, которая вычисляет значение выражения:

$$y = \frac{\sqrt{x-5}}{x^2+2}$$

При этом предусмотреть, что при отрицательном значении подкоренного выражения программа исполняться не будет. Т.е., если подкоренное выражение отрицательно, сообщаем пользователю: «Корень из отрицательного числа вычислять отказываюсь!», после чего просим заново ввести данные, и программа исполняется заново. Используйте оператор безусловного перехода. Нарисовать блок-схему к программе.

Смотрите примеры из задания 103-106.

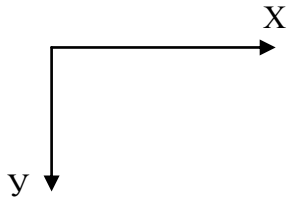
115. *** Не используя операторы цикла, вычислить сумму чисел 2, 5, 8, 11, 14, используя оператор goto. Вывести на

экран данные числа. После выполнения данного задания решите задачу, используя оператор цикла.

Смотрите примеры из задания 103-106.

116. ** Написать программу, в результате которой звездочка случайного цвета двигается по горизонтали. Измените программу, чтобы звездочка двигалась по вертикали.

Указания: Координатная плоскость экрана среды Паскаль в текстовом режиме имеет следующий вид:



При этом координаты оси У – это целые числа, отсчет которых начинается сверху с нуля и заканчивается числом 25, а координаты X следуют слева направо и имеют диапазон значений 0..80. При обращении к точке экрана сначала указывается координата X, а затем У.

Используйте принцип движения:

1. Звездочка в виде символа появляется на экране в нужной точке;
2. Осуществляем задержку экрана;
3. Далее звездочка стирается или рисуется цветом фона;
4. Перемещаем указатель в следующую позицию;
5. Повторяем выше указанные действия до тех пор, пока не дойдем до конца строки (столбца).

Используйте операторы:

textcolor(n) – процедура, которая задает цвет выводимых на экран символов, здесь n-целое число от 0 до 15, 0 – черный чвет, 15 – белый цвет.

gotoxy(x,y)-процедура переводит курсор в заданные координаты экрана (размер экрана 80 символов по горизонтали (x) и 25 по вертикали (y));

delline-процедура уничтожает строку с курсором;

delay(n) – процедура обеспечивает задержку работы программы на заданный интервал времени n – целое число.

Пример программы:

```
uses crt;
var n:integer;
label l;
begin
clrscr;
n:=1;                    {параметр для перемещения в нужную координату экрана}
l:if n<=80 then          {если координата меньше или равна 80}
begin
    textcolor(random(15)+1);{задаем случайный цвет символа}
    write('*');          {выводим звездочку и оставляем указатель на той же строке}
    delay(10000);        {задержка экрана}
    delline;             {стирание строки, на которой расположен указатель}
    n:=n+1;              {наращиваем координату x}
    gotoxy(n, 1);        {переходим в новую координату}
    goto l;              {повторяем действия, начиная с метки}
end;
end.
```

В данной программе необходимо обязательно подключить модуль CRT (после заголовка программы написать **uses crt;**), так как функции и процедуры очистки экрана, задания цвета символа, стирание строки, задержка экрана относятся к этому модулю и без его подключения работать не будут.

117. **** Написать программу, реализующую игру «Угадайка!». Суть игры в том, что случайным образом генерируется целое число от 1 до 100, а пользователь должен

угадать это число. Сначала здороваемся с пользователем и спрашиваем его имя, затем, вежливо обращаясь к игроку по имени, спрашиваем, хочет ли он сыграть в интересную игру «Угадайка!». Если ответ положительный, просим ввести число от 1 до 100. Если введенное число равно сгенерированному, то хвалим игрока за находчивость, поздравляем с победой и выходим из программы. Если введенное число больше, сообщаем об этом и возвращаемся к предложению «Введите число от 1 до 100». Точно также поступаем, если введенное число меньше. Таким образом, игра заканчивается только тогда, когда пользователь угадывает число. Используйте оператор безусловного перехода. Дополните программу заставкой, которая появится после того, как игрок угадал число. По центру экрана выводится название игры «Угадайка!» (мерцает), затем на следующей строке появляются звездочки по одной друг за другом, каждая случайным новым цветом, пробегая по строчкам экрана. При этом включите звук. Таким же образом можно организовать, чтобы слово «Угадайка!» бежало по экрану.

sound(f) – процедура заставляет встроенный динамик звучать с нужной частотой f – целое число;

nosound – процедура выключает динамик.

Смотрите пример из задания 116

118. ** Не используя операторов цикла, написать программу, которая вычисляет значение выражения:

$$y = \frac{x \quad -5x+6}{x-3}$$

При этом предусмотреть, что при нулевом значении знаменателя программа исполняться не будет. Т.е., если знаменатель равен нулю, сообщаем пользователю: «При введенном значении переменной знаменатель обращается в ноль, а на ноль делить нельзя!», после чего просим заново ввести данные, и программа исполняется заново. Используйте оператор безусловного перехода. Нарисовать блок-схему к программе.

Смотрите примеры из заданий 103-106.

119. **** Не используя операторов цикла, решить задачу. Напишите программу, которая в центре экрана помещает

подобие циферблата электронных часов, отсчитывающих время в таком формате: 12.31.22. (минуты. секунды. миллисекунды). Выведите бегущую строку из разноцветных звездочек выше циферблата, наглядно демонстрирующую отсчет секунд.

120. *** Не используя операторов цикла, решить задачу. В память компьютера вводится целое число. Найти сумму всех целых чисел от единицы до введенного.

Алгоритм исполнения:

a) Обнуляем сумму;
b) Включаем счетчик (присваиваем новой переменной значение 1, далее оно будет возрастать на единицу, пока не дойдем до введенного числа);
c) Если счетчик больше введенного числа, уходим в конец программы;
d) Наращиваем сумму;
e) Наращиваем счетчик;
f) Переходим к пункту 3.

Смотрите примеры из задания 103-106.

121. ** Пользователю предлагается ввести вещественное число в диапазоне [-5, 9). Если вводится число, не принадлежащее данному диапазону, сообщить об этом пользователю и попросить ввести заново и так продолжать до тех пор, пока число не введется правильно. Если число дробное, выдать целую и дробную часть числа, а если целое – выдать целую часть. Нарисовать блок-схему к программе.

122. *** Пользователю предлагается ввести символ латинского алфавита в диапазоне от a до f. Если вводится символ, не принадлежащий данному диапазону, сообщить об этом пользователю и попросить ввести заново и так продолжать до тех пор, пока символ не введется правильно. Если пользователь ввел один из символов a, c, e – то выдать

его порядковый номер, в других случаях – выдать заглавную букву, соответствующую введенному символу. Нарисовать блок-схему к программе.

Указания: Используйте оператор безусловного перехода.

123. *** Пользователю предлагается ввести два символа латинского алфавита в алфавитном порядке. Если символы вводятся не в алфавитном порядке, сообщить об этом пользователю и попросить ввести заново и так продолжать до тех пор, пока символы не будут введены правильно. Далее компьютер задумывает целое число из диапазона, соответствующего диапазону порядковых номеров символов, которые расположены между символами, введенными пользователем. Если задуманное число нечетное, то вывести символ, соответствующий данному номеру, иначе вывести следующий за ним символ.

Указания: Используйте оператор безусловного перехода

124. * Определить результат исполнения следующих операторов:

```
n:=0;
x:=a;
while x<=b do begin
    if x mod 5 <>0 then
        begin
                n:=n+1;
                writeln(x);
        end;
    x:=x+h;
end;
writeln(n);
```

Выполнить данную программу на компьютере, дополнив ее разделом описания.

125. * Определить результат исполнения следующих операторов:

```
p:=0;
for  i:=1 to 5 do
    p:=p+i;
```

```
writeln(p);
```
Выполнить данную программу на компьютере, дополнив ее разделом описания.

126. * Определить результат исполнения следующих операторов:

```
y:=2;
while  y<10 do
    begin
        writeln((y+1)*5);
         y:=y+3;
     end;
```

Выполнить данную программу на компьютере, дополнив ее разделом описания.

127. * Определить результат исполнения следующих операторов:

```
uses crt;
var a,n,x:integer;
begin
clrscr;
a:=0;
n:=0;
writeln('Введите числа');
while not eoln do
begin
        read(x);
        if x mod 3 =0 then
        begin
                n:=n+1;
                a:=a+x;
        end;
  end;
writeln;
```

```
writeln(a,'   ',n);
readln;
readln;
end.
```

Выполнить данную программу на компьютере.

128. * Определить результат исполнения следующих операторов:

```
x:=3;
repeat
    write(x);
    x:=x+2;
until x>13
```

Выполнить данную программу на компьютере, дополнив ее разделом описания.

129. * Определить результат исполнения следующих операторов:

```
p:=1;
for  i:=1 to n do
     p:=p*i;
writeln(p);
```

Выполнить данную программу на компьютере, дополнив ее разделом описания.

130. * Определить результат исполнения следующих операторов:

```
n:=1;
l:if n<=80 then
begin
    write('*');
    delay(1000);
    delline;
    n:=n+1;
    gotoxy(n,1);
    goto l;
end;
```

Выполнить данную программу на компьютере, дополнив ее разделом описания.

131. * Какого типа числа и в каком интервале сгенерируются в результате действия оператора random(7)+2?

132. * Какого типа числа и в каком интервале сгенерируются в результате действия оператора 2*random-5?

133. * Что выйдет на экран при n=10 в результате исполнения операторов

```
case n of
        3,4,5,6:begin
                        writeln('*');
                        writeln('***');
                end;
        7,8,9,10:writeln('**')
end;
writeln('****');?
```

Выполнить данную программу на компьютере, дополнив ее разделом описания.

134. *** Написать программу, которая печатает разноцветные звездочки в одной точке экрана до тех пор, пока не будет нажата клавиша Enter. Нарисовать блок-схему к программе.

Смотрите пример из задания 116

135. ** Напишите программу, рисующую на экране горизонтальную линию, состоящую из точек, расстояние между которыми 2.

136. *** Заполните экран горизонтальными линиями (через 2), а затем, с помощью еще одного оператора цикла, вертикальными линиями другого цвета (тоже через 2). Должна получиться решетка (рисунок 2.14).

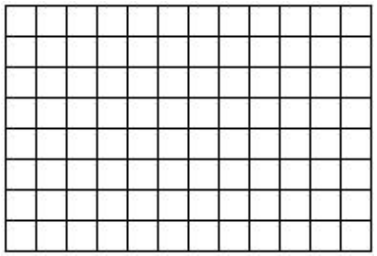


Рисунок 2.14. Иллюстрация к задаче 137.

Указания: Используйте два цикла с параметром

137. ** Вывести на экран четные числа от 2 до 20.

138. ** Выведите на экран в строку все числа первой сотни, оканчивающиеся на пять. Нарисовать блок-схему к программе.

139. *** Реализовать движение звездочки, которая случайно меняет свой цвет, по вертикали вниз, по горизонтали вправо.

Смотрите пример из задания 116.

140. *** Вычислить сумму чисел 7, 11, 15, 19, используя оператор повторения. Вывести на экран данные числа. Нарисовать блок-схему к программе.

Смотрите примеры из заданий 125-129.

141. *** Выписать нечетные числа из заданного диапазона. Нарисовать блок-схему к программе.

Смотрите примеры из заданий 125-129.

142. ** С клавиатуры вводятся действительные числа. Вычислить их среднее арифметическое. Количество чисел определяется в ходе исполнения программы. Используйте оператор цикла. Нарисовать блок-схему к программе.

Смотрите примеры из заданий 125-129.

143. ** Напишите программу., выводящую на экран степени числа 2 от 2 до 10 включительно.

144. ** Распечатайте в табличном виде (с аргументами) значение функции квадратного корня на интервале [2; 4] с шагом 0,1. Нарисовать блок-схему к программе.

145. ** Напишите программу, запрашивающую возраст пользователя, а затем печатающую текст "Да ты крут!" по числу прожитых лет. Обратите внимание, что здесь в теле

цикла не будет использоваться параметр. Такое тоже возможно. Нарисовать блок-схему к программе.

146. *** С клавиатуры вводится текстовая строка в виде символов. Подсчитать общее количество введенных символов и число букв «т» в тексте. Условием окончания цикла является проверка конца строки.

Указания: Эта проверка осуществляется с помощью функции eoln (while not eoln – пока не нажата клавиша enter). Точно также можно использовать цикл с постусловием.

147. * Необходимо при каждом запуске программы случайным образом получать либо число 0 ("решка"), либо 1 ("орел"). Используйте генерацию чисел.

148. * Смоделируйте эффект бросания игрального кубика.

149. *** С клавиатуры вводятся числа. Подсчитать количество чисел, кратных трем и тех, которые делятся на три с остатком.

Смотрите примеры из заданий 125-129.

150. *** Даны случайные числа из какого-либо диапазона. Найти их произведение, количество, вывести все числа на экран, подсчитать среднее арифметическое и количество чисел, кратных пяти.

Смотрите примеры из заданий 125-129.

151. *** Реализовать движение звездочки, которая случайно меняет свой цвет, по вертикали вниз.

Смотрите примеры из задания 130.

152. *** При помощи оператора while... вычислите с— наибольший общий делитель введенных с клавиатуры натуральных чисел x и y.

Смотрите пример 60.

153. *** Смоделировать движение звездочки по экрану под углом 45°. Использовать операторы циклов.

Указания:

а) Присвоить переменным, обозначающим координаты

b) экрана начальные значения X:=80; Y:=25;
c) Пусть DX и DY – приращения координат X и Y. Присвоить им единичные значения;
d) Так как мы их сделали положительными, координаты X и Y будут увеличиваться, а, следовательно, звездочка будет двигаться под углом 45°;
e) Переходим в точку с координатами X и Y. Выводим звездочку на экран;
f) Осуществляем задержку;
g) Стираем строку;
h) Меняем знак приращения координаты X в случае, если координата X достигает конечных значений (0 или 80);
i) Меняем знак приращения координаты Y в случае, если координата Y достигает конечных значений (0 или 25);
j) Наращиваем координаты: X:=X+DX; Y:=Y+DY;
k) Переходим к пункту 4.

154. *** Смоделировать движение звездочки по экрану под управлением клавиатуры.

Указания:

a) Для начала определитесь, какая клавиша будет отвечать за движение вправо, какая – за движение влево, вверх и вниз.
b) Присвоить переменным, обозначающим координаты экрана начальные значения X:=1; Y:=1;
c) Пусть DX и DY – приращения координат X и Y. Присвоить им единичные значения;
d) Так как мы их сделали положительными, координаты X и Y будут увеличиваться, а, следовательно, звездочка будет двигаться то по горизонтали, то по вертикали;
e) Начало тела цикла. Переходим в точку с координатами X и Y. Выводим звездочку на экран;
f) Осуществляем задержку;
g) Стираем строку;
h) В случае нажатия клавиши, отвечающей за движение вправо, увеличиваем координату X звездочки: X:=X+DX;
i) В случае нажатия клавиши, отвечающей за движение

вниз, увеличиваем координату Y звездочки: Y:=Y+DY;

j) Дальше догадывайтесь сами;

k) Используйте оператор цикла с постусловием. Пусть звездочка движется до тех пор, пока не нажали клавишу ESC. Ее код =27.

Указания: Функция **readkey** приводит программу в режим ожидания нажатия клавиши, возвращая значение типа char. Если вы хотите, чтобы звездочка двигалась по экрану при нажатии клавиши, после слова repeat присвойте какой-либо переменной типа char значение данной функции. Таким образом, можно осуществить управление движением каких-либо предметов с помощью клавиатуры.

155. ** Нарисовать звездное небо. Звездочки должны быть разбросаны по экрану случайным образом и разного цвета. Сделайте, чтобы на небе не было черных звезд, так как они не будут видны.

156. *** Смоделировать движение звездочки по экрану под углом 45°, которая отскакивает от сторон экрана под тем же углом.

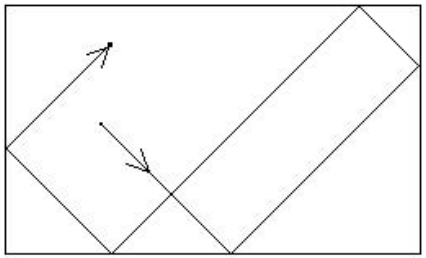


Рисунок 2.15. Иллюстрация к задаче 166

Указания:

a) Присвоить переменным, обозначающим координаты экрана начальные значения X:=80; Y:=25;

b) Пусть DX и DY – приращения координат X и Y.

c) Присвоить им единичные значения;
d) Так как мы их сделали положительными, координаты X и Y будут увеличиваться, а, следовательно, звездочка будет двигаться под углом 45°;
e) Начало тела цикла. Переходим в точку с координатами X и Y. Выводим звездочку на экран;
f) Осуществляем задержку;
g) Стираем строку;
h) Меняем знак приращения координаты X в случае, если координата X достигает конечных значений (0 или 80);
i) Меняем знак приращения координаты Y в случае, если координата Y достигает конечных значений (0 или 25);
j) Наращиваем координаты: X:=X+DX; Y:=Y+DY;
k) Используйте оператор цикла с постусловием. Пусть звездочка движется до тех пор, пока не нажали клавишу ESC. Ее код =27.

Указания: Функция readkey приводит программу в режим ожидания нажатия клавиши, возвращая значение типа char. Если вы хотите, чтобы звездочка двигалась по экрану при нажатии клавиши, после слова repeat присвойте какой-либо переменной типа char значение данной функции. Таким образом, можно осуществить управление движением каких-либо предметов с помощью клавиатуры.

157. **** Усложним задачу. На экране появляется прямоугольник, от которого точка тоже должна отражаться. (рисунок 2.16)

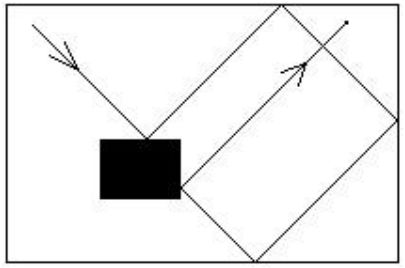


Рисунок 2.16. Иллюстрация к задаче 167

Указания:

Допустим, координаты углов прямоугольника следующие:

- левый верхний 100 20, 180 5;
- правый верхний 300 50, 180 5;
- левый нижний 100 20, 230 20;
- правый нижний 300 50, 230 20.

Тогда, по сравнению с предыдущим примером, к условиям отражения от сторон экрана добавятся еще четыре условия отражения от сторон прямоугольника. Мы приведем два — для верхней и левой сторон, а вы напишите еще два — для нижней и правой.

Для верхней:

IF (Y=5) AND (20<X) AND (X<50) THEN DY=-DY

То есть если точка достигла горизонтали 5, а по X попала в интервал (20; 50), то изменяем направление движения по Y на противоположное.

Для левой:

IF (X=20) AND (5<Y) AND (Y<20) THEN DY=-DY

Рассуждения аналогичны.

158. *** Смоделировать движение двух звездочек по экрану навстречу друг другу на некотором расстоянии.

Указания:

a) Присвоить переменной, обозначающей координату X экрана начальное значение X:=1;
b) Пусть DX – приращение координаты X.. Присвоить ей единичное значение;
c) Так как мы его сделали положительным, координата X будет увеличиваться, а, следовательно, звездочка будет двигаться по горизонтали;
d) Начало тела цикла. Переходим в точку с координатами X и Y. При этом координаты Y для звездочек будут разными и постоянными, а X – меняться. Выводим звездочку на экран;
e) Осуществляем задержку;

f) Стираем строку;
g) Меняем знак приращения координаты X в случае, если
h) координата X достигает конечных значений (0 или 80);
i) Наращиваем координату: X:=X+DX;
j) Точно также поступаем со второй звездочкой.
k) Используйте оператор цикла с постусловием. Пусть звездочка движется до тех пор, пока не нажали клавишу ESC. Ее код =27.

Указания: Функция readkey приводит программу в режим ожидания нажатия клавиши, возвращая значение типа char. Если вы хотите, чтобы звездочка двигалась по экрану при нажатии клавиши, после слова repeat присвойте какой-либо переменной типа char значение данной функции. Таким образом, можно осуществить управление движением каких-либо предметов с помощью клавиатуры.

159.**** Теперь вы готовы к более объемному и сложному заданию. Называться оно будет "Муха в графине". Сначала на экране вы рисуете прямоугольный графин с горлышком. Затем располагаете там точку и заставляете ее двигаться внутри графина, отражаясь от его стенок (будет интересней, если вы не станете ее стирать, тогда мы сможем видеть траекторию ее полета). Меняя начальное расположение точки, можно добиться, что через какое-то время она вылетит из графина. Пусть в этот момент на экране появится надпись "Ура! Я на свободе!", а муха продолжит полет, отражаясь от сторон экрана. Графин можно сделать закрашенным (рисунок 2.17).

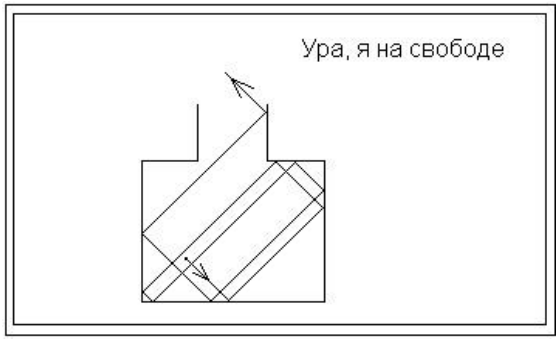


Рисунок 2.17. Муха в графине

Если вы обратили внимание, "муха" при полете слегка пробивает стенки графина. Что нужно изменить в условиях, чтобы этого избежать? Сделайте из "мухи" шарик, и снова добейтесь того, чтобы он не портил стенки графина.

160.**** Давайте сделаем простейший графический редактор. У некоторых в детстве была, наверное, игрушка "Волшебный экран", где можно было крутить колесики и рисовать отрезками прямых линий. Почему бы то же самое не сделать на компьютере? Для этого нам понадобится использование оператора readkey, о котором было написано в предыдущем примере. Сделайте так, чтобы на экране воспроизводился символ той клавиши, которую вы нажмете Хорошо бы предусмотреть рисование цветом фона для перемещения точки без следа. А чтобы потом определить, где находится точка, достаточно изменить ее цвет.

161.** Выписать значения функции в заданном диапазоне с заданным шагом. Учесть область определения.

a) $y = \frac{x^2 + 7x - 10}{x - 5}$

b) $y = 10x\sqrt{x + 4}$

c) $y = \frac{5x^2}{x+4}$

d) $y = \sqrt{2x-5}$

e) $y = \frac{7x}{x-3}$

f) $y = \sqrt{3x+2} - x^2$

g) $y = \frac{x^2+3}{x-7}$

h) $y = \sqrt{x+2} - x^2$

162. ** Найти сумму квадратов натуральных чисел из заданного диапазона.

163. ** Смоделировать движение звездочки по вертикали, а затем по горизонтали.

164. ** Вывести таблицу синусов натуральных чисел в заданном диапазоне.

165. ** Произведение натуральных чисел 1*2*3*4*…*n называется n-факториалом. Пусть задано натуральное n. Найти n-факториал.

166. * Вычислить значение переменной y=4k-5n при всех значениях переменных n=2, 4, 6 и k=3, 6, 9, 12, 15.

167. * Написать программу, выдающую таблицу умножения натуральных чисел от 1 до 9.

168. ** Найти количество целых чисел из заданного диапазона действительных чисел с заданным шагом. Распечатать эти числа в строчку. Вычислить их сумму. Использовать оператор цикла с предусловием.

169. ** Заполнить экран разноцветными звездочками. Использовать вложенные циклы с параметром.

Смотрите пример 67

170. *** Найти количество чисел, кратных 7 из заданного диапазона целых чисел с заданным шагом. Распечатать эти числа в строчку. Вычислить их сумму. Использовать оператор цикла с предусловием.

171. *** Заполнить прямоугольник с заданными координатами разноцветными точками. Использовать вложенные циклы с параметром.

Смотрите пример 67

172. ** Найти количество нечетных чисел из заданного диапазона целых чисел с заданным шагом. Распечатать эти числа в строчку. Вычислить их сумму. Использовать оператор цикла с предусловием.

173. *** Вывести строчку из разноцветных звездочек по диагонали экрана. Использовать вложенные циклы с параметром.

Смотрите пример 67

174. *** Дано натуральное n. Вычислить сумму, и вывести значение каждой из дробей, являющейся слагаемым суммы.

a) $S = \frac{1}{3^2} + \frac{3}{5^3} + \frac{5}{7^4} + \ldots$

b) $S = \frac{1}{1 \cdot 2} - \frac{3}{2 \cdot 3} + \frac{5}{3 \cdot 4} - \ldots$

c) $S = \frac{1}{3^1} - \frac{3}{5^2} + \frac{5}{7^3} - \ldots$

d) $S = \frac{1}{3^2} + \frac{3}{5^3} + \frac{5}{7^4} + \ldots$

e) $S = \frac{1}{3^1} - \frac{3}{5^2} + \frac{5}{7^3} - \ldots$

Смотрите пример 63

175. * Написать программу, которая выдает таблицу умножения для числа 5. Нарисовать блок-схему к программе.

176. * Написать программу, которая выдает таблицу квадратов чисел от 11 до 24.

177. * Написать программу, которая выдает таблицу кубов чисел от 5 до 11. Нарисовать блок-схему к программе.

178. ** Вычислить сумму:

a) $S = \frac{1}{2} + \frac{1}{3} + \frac{1}{4} + \frac{1}{5} + \frac{1}{6}$

b) $s = 2^3 + 2^4 + 2^5 + 2^6$

c) $S = 3^2 + 3^3 + 3^4 + 3^5$

d) $S = \frac{1}{6} + \frac{1}{7} + \frac{1}{8} + \frac{1}{9} + \frac{1}{10}$

e) $S = \frac{1}{2} + \frac{2}{3} + \frac{3}{4} + \frac{4}{5} + \frac{5}{6}$

f) $S = 4^2 + 4^3 + 4^4 + 4^5$

g) $S = \frac{3}{4} + \frac{4}{5} + \frac{5}{6} + \frac{6}{7}$

Смотрите пример 67

Нарисовать блок-схему к программе.

179. *** Напишите программу, выводящую на экран таблицу умножения от 2 до 10 в следующем виде (рисунок 2.18)

	2	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
2	4	6	8	10	12	14	16	18	20
3	6	9	12	15	18	21	24	27	30
4	8	12	16	20	24	28	32	36	40
5	10	15	20	25	30	35	40	45	50
6	12	18	24	30	36	42	48	54	60
7	14	21	28	35	42	49	56	63	70
8	16	24	32	40	48	56	64	72	80
9	18	27	36	45	54	63	72	81	90
10	20	30	40	50	60	70	80	90	100

Рисунок 2.18. Иллюстрация к задаче 189

Смотрите пример 67

180. ** Напишите программу вычисления произведения cos5°cos10°cosl5°...cos85°

181. *** Итак, используя полученные знания про вложенные циклы и оператор delay, напишите программу, которая в центре чистого экрана будет выводить показания хронометра — часы, минуты и секунды, разделенные двоеточием, начиная с 0 час. 0 мин. 0 с. Отрегулируйте свой хронометр, чтобы он шел правильно.

182. **** Усложним предыдущее задание. Нарисуем будильник, сделаем в нем два поля. В одно выведем текущую дату, а во второе — наш уже получившийся хронометр. Если очень захочется, то можно сделать из него будильник, добавив в нужный момент оператор sound. Рисунок 2.19.

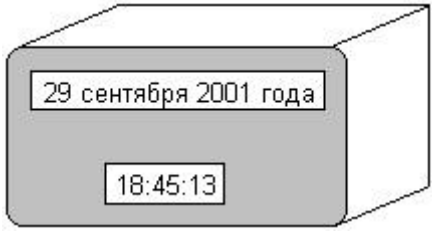


Рисунок 2.19. Иллюстрация к задаче 192

183. **** Это задание носит прикладной характер и позволяет опытным путем вычислить число п. Да, безусловно, практически все из вас знают это число с точностью по крайней мере до двух знаков. Но предлагаемый метод очень хорош. Называется он методом Монте-Карло. Монте-Карло — европейская столица игорного бизнеса, а значит там владычествует Его Величество Случай. Вот мы и попробуем поставить его себе на службу. Сначала забудьте, чему равно п и послушайте теорию вопроса. Представьте себе окружность радиусом R, вписанную в квадрат. Из этого следует, что сторона квадрата будет 2R , а его площадь SK = $(2R)^2$ =$4R^2$. Площадь круга SO = пR^2 . Далее берем и равномерно посыпаем квадрат песком. Затем нанимаем

бригаду рабочих, которые считают, сколько песчинок всего (N1) и сколько из них попало в круг (N2). Потом составляется простая пропорция — площадь квадрата так относится к площади круга, как общее количество песчинок к количеству песчинок попавших в круг. Отсюда сенсационный вывод — радиус окружности не имеет никакого значения, она должна быть лишь вписана в квадрат. Но где ж мы найдем песок, а главное тех, кто все это будет считать? Поэтому поставим компьютерный эксперимент. Нарисуем квадрат и впишем в него окружность. Координаты опорных точек (если сами рисовали) знаем. Уравнение окружности X2 +Y2 = R2 тоже знаем. Задаем цикл до 1000, в котором случайным образом определяем координаты "песчинок" так, чтобы они лежали внутри квадрата. Тут же проверяем условие, а не попала ли "песчинка" в круг (используя уравнение окружности), и если попала, подсчитываем их количество. Кроме того, рисуем их на экране разными цветами (попавшие и не попавшие). По окончании цикла подсчитываем и выводим на экран число п. Понятно, что чем больше количество "песчинок", тем более точным будет результат. Для того чтобы знать, когда закончится эксперимент, рекомендуется выводить на экран счетчик "песчинок" (как мы делали с хронометром). Но, все-таки, экспериментировать с миллионом "песчинок" не надо — замучаетесь ждать сами, да и компьютер, хотя и железный, но все же живой. Рисунок 2.20.

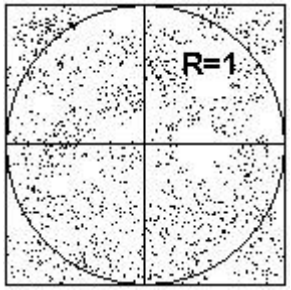


Рисунок 2.20. Иллюстрация к задаче 193

184. *** Написать программу, реализующую игру, суть которой в следующем:

На экране случайным образом появляются 10 разноцветных звездочек (в виде точек). Игроку нужно внимательно следить за их появлением, чтобы угадать, какая из них появилась последней. Затем, после нажатия клавиши enter, пользователь начинает «ловить» последнюю звездочку, шагая с помощью клавиш управления курсором по экрану вправо, влево, вверх и вниз. Путь из точек должен быть виден на экране. Шагаем до тех пор, пока не попадем в нужную точку. Игра останавливается, и после нажатия enter выдается сообщение, сколько времени пользователь «ловил» звездочку и какое место заработал.

Указания: Необходимо организовать два цикла: первый, с параметром, – выводит на экран звездочки случайным образом, т.е. их координаты – случайные числа, второй, с постусловием, пока не попадем в последнюю звездочку, - рисует пройденный путь.

Для реализации игры вам пригодятся коды клавиш:

- ✓ **Курсор вверх – 72;**
- ✓ **Курсор вниз – 80;**
- ✓ **Курсор влево – 75;**
- ✓ **Курсор вправо – 77;**
- ✓ **Enter – 13;**
- ✓ **Esc – 27.**

ТЕСТОВЫЕ ЗАДАНИЯ ДЛЯ ПРОВЕРКИ ТЕОРЕТИЧЕСКИХ ЗНАНИЙ

1. **Какая из следующих последовательностей символов является выражением, записанным по правилам Паскаля?**
   A) sin(x)+abs(abs(y-1.7)); *
   B) x.8+y2;
   C) |h|/2;
   D) 8*a&b;
   E) –5a.
   F)

2. **Указать ошибки во фрагменте программы**

```
program m5;
var
        f,g:real;
        d:integer;
begin
        d:=f mod g;
        d:=f div g
end.
```

   A) неверные имена программы и переменных;
   B) неверный оператор присваивания;
   C) переменная f должна быть типа integer, чтобы не возникло ошибки несовместимости типов;
   D) функции mod и div работают только с переменными целого типа, результат тоже является целым; *
   E) переменная d должна быть типа real, чтобы не возникло ошибки несовместимости типов.

3. **Указать ошибки во фрагменте программы**

```
program m1;
var
        a,b:real;
begin
        b:=read(a)
end;
```

   A) неверный оператор присваивания;

B) переменная b должна быть типа integer, чтобы не возникло ошибки несовместимости типов;
C) инструкцию read нельзя использовать в операторе присваивания; *
D) инструкция read работает только с переменными целого типа;
E) переменная a должна быть типа integer, чтобы не возникло ошибки несовместимости типов.

4. Указать ошибки во фрагменте программы

```
program m;
const d=Маша;
var
        a:char;
begin
        a:=d
end;
```

A) несовместимость типов в операторе присваивания;
B) несовместимость типов в операторе присваивания и неверно записано значение константы; *
C) неверно записано значение константы;
D) неверный оператор присваивания;
E) неверный порядок расположения разделов программы.

5. Указать ошибки во фрагменте программы

```
program m123;
var  a, b:char;
begin
        b:=ord(a)
end;
```

A) неверный оператор присваивания;
B) функция ord применяется для целых значений => переменная a должна быть целого типа; *

C) функция ord дает результат целого типа => переменная b должна быть целого типа;
D) нет ошибок;
E) переменная a должна быть типа real, чтобы не возникло ошибки несовместимости типов.

6. Указать ошибки во фрагменте программы

```
program m12;
var
        a:real;
        b:integer;
begin
        a:=odd(b)
end;
```

A) неверный оператор присваивания;
B) функция odd дает результат логического типа => переменная a должна быть логического типа; *
C) функция odd дает результат целого типа => переменная a должна быть целого типа;
D) нет ошибок;
E) переменная a должна быть типа real, чтобы не возникло ошибки несовместимости типов.

7. Указать ошибки во фрагменте программы

```
program ttt;
var
        a,c:char;
        b:Boolean;
begin
        read(a, c)
        b:=a=c;
        writeln(b)
end.
```

A) неверно записан оператор присваивания;
B) переменную логического типа нельзя использовать в операторе вывода;
C) нельзя сравнивать переменные типа char;

D) в программе сначала должен следовать оператор вывода, а затем оператор ввода;
E) нет ошибок; *

8. Какая из следующих последовательностей символов является именем переменной языка Паскаль?

A) r1; *
B) cd 45;
C) $z1;
D) сумма;
E) k-6.

9. Какая из следующих последовательностей символов является оператором ввода?

A) read a, b, c;
B) read(a, b, c); *
C) a:=read(b);
D) read(x, x+y);
E) read(x; y).

10. Выбрать функцию, выдающую номер данного значения переменной. Для каких типов она определена?

A) chr(x), для целых x;
B) ord(x), для действительных x;
C) odd(x), для логических x;
D) ord(x), для порядковых x; *
E) odd(x), для порядковых x.

11. Записать функцию проверки числа на нечетность. Какого типа будет результат?

A) chr(x), результат целый;
B) ord(x), результат действительный;
C) odd(x), результат логический; *
D) ord(x), результат порядковый;
E) odd(x), результат порядковый.

12. Какого типа будет результат операции деления?

A) действительный; *

B) целый;
C) логический;
D) порядковый;
E) действительный или целый.

13. Какая из последовательностей символов является оператором условия?

A) if x>y or y<0 then x:=0 else read(y);
B) if x>=y then x:=0; y:=0 else write(z);
C) if a<b then 100 else z:=5;
D) if a<b<c then z:=z+1;
E) if sqrt(z)<=3.17 then z:=z+1; *

14. Пусть значения переменных x и y равны, соответственно, 0,3 и -0,2. Какие значения будут иметь эти переменные после выполнения операторов присваивания: y:=-y; x:=x+y; y:=y+1?

A) 0.1, -0.8;
B) –0.2, 0.5, 1.2;
C) –0.2, 0.1, -0.8;
D) 0.5, 1.2; *
E) 0.1, 1.2.

15. Результат исполнения следующих операторов:

```
y:=2;
case y of
    1:write(*);
    2:write(***);
    3:write(**)
end;
```

A) **;
B) *;
C) ***; *
D) *
   *;
E) *
   *
   *.

16. Результат исполнения следующих операторов:

```
readln(a, b, h);
x:=a; n:=0;
while x<=b do
begin
    writeln(x);
    n:=n+1;
    x:=x+h;
end;
writeln(n);
```

A) кол-во и сумма чисел, введенных с клавиатуры;
B) кол-во и сумма чисел из заданного диапазона;
C) вывод чисел и их кол-во из заданного диапазона; *
D) вывод случайных чисел и их кол-во;
E) вывод случайных чисел в диапазоне от a до b.

17. Результат исполнения следующих операторов:

```
p:=0;
for i:=1 to 5 do
    p:=p+i;
writeln(p);
```

A) 15; *
B) числа 1, 5, 10, 15, 20 в столбик;
C) числа от 1 до 5 в столбик;
D) сумма и сами целые числа от 1 до 5;
E) 10.

18. Функция, выдающая код символа.

A) Int(c);
B) frac(c);
C) odd(c);
D) ord(c); *
E) mod(c).

19. Результат исполнения следующих операторов:

```
y:=2;
```

```
while  y<10 do
    begin
        writeln((y+1)*5);
         y:=y+3;
     end;
```

A) числа 2, 5, 8 по вертикали;
B) числа 2, 5, 8 по горизонтали;
C) числа 15, 30, 45 по горизонтали;
D) числа 15, 30, 45 по вертикали; *
E) числа 2, 15, 5, 30, 8, 45 по вертикали.

20. Оператором генерации случайных целых чисел от –4 до 7 является

A) 12*Random-4;
B) Random(12)-4; *
C) Random(-4)+7;
D) Random(7)-4;
E) Random(7)+4.

21. Результат исполнения следующих операторов:

```
a:=0;
n:=0;
writeln('Введите числа');
while not eoln do
    begin
        read(x);
        if x mod 3 =0 then
                begin
                        n:=n+1;
                        a:=a+x;
                end;
    end;
writeln(a,n)
```

A) кол-во и сумма чисел, кратных трем, из чисел, введенных с клавиатуры; *
B) кол-во и сумма чисел, кратных трем из заданного диапазона;
C) вывод чисел, кратных трем и их сумма;

D) вывод чисел, кратных трем и подсчет их кол-ва;
E) кол-во и сумма чисел, не делящихся на три из диапазона чисел.

22. Результат исполнения следующих операторов:

```
x:=3;
repeat
    write(x);
    x:=x+2;
until x>13
```

A) числа 3, 5, 7, 9, 11, 13 по горизонтали;
B) числа 3, 5, 7, 9, 11 по вертикали;
C) числа 2, 4, 6, 8, 10, 12 по горизонтали;
D) числа 3, 5, 7, 9, 11 по горизонтали; *
E) числа 2, 4, 6, 8, 10, 12 по вертикали.

23. Результат исполнения следующих операторов:

```
p:=1;
for i:=1 to n do
     p:=p*i;
writeln(p);
```

A) числа от 1 до n в столбик;
B) n-факториал; *
C) числа от 1 до n в строчку;
D) сумма целых чисел от 1 до n;
E) число n.

24. Результат исполнения следующих операторов:

```
n:=1;
l:if n<=80 then
begin
    write('*');
    delay(1000);
    delline;
```

```
    n:=n+1;
    gotoxy(n, 1);
    goto l;
end;
```

A) движение звездочки по вертикали в левом столбце экрана;
B) верхняя строка экрана заполняется звездочками;
C) левый столбец экрана заполняется звездочками;
D) главная диагональ экрана заполняется звездочками;
E) движение звездочки по горизонтали в верхней строке экрана. *

25. Какого типа числа и в каком интервале сгенерируются в результате действия оператора random(7)+2**?**

A) целые от -2 до 7;
B) целые от 2 до 7;
C) целые от 2 до 8; *
D) действительные от 7 до 9;
E) целые от 7 до 9.

26. Какого типа числа и в каком интервале сгенерируются в результате действия оператора 2*random-5?

A) целые от -5 до 2;
B) действительные от -5 до -3); *
C) целые от -5 до -4);
D) дейтсвительные от 2 до 5;
E) действительные от -5 до -4.

27. Какое из условий нельзя разместить между if **и** then**?**

A) (x<y) and (y<z);
B) x<y;
C) x<z or z>y; *
D) not(x>1) or (x<=y) and (x<z);
E) not(x).

28. Что выйдет на экран при n=3 в результате исполнения операторов

```
if (n<7) and (n>2) then
    begin
        writeln('*');
```

writeln(‘***’);
end
else writeln(‘**’);
writeln(‘****’);?

A) ***
****;

B) **
****;

C) *
***
****; *

D) *
***;

E) *.

29. Что выйдет на экран при n=10 в результате исполнения операторов

if (n<7) and (n>2) then
begin
writeln(‘*’);
writeln(‘***’);
end
else
begin
writeln(‘**’);
writeln(‘****’);
end.

A) ***
****;

B) **
****; *

C) *
***

****;

D) *
***;
*.

30. Что выйдет на экран при n=10 в результате исполнения операторов

```
Case n of
        3,4,5,6:begin
                        writeln('*');
                        writeln('***');
                end;
        7,8,9,10:writeln('**')
end;
writeln('****');?
```

A) ***
****;

B) **
****; *

C) *
***
****;

D) *
***;

E) *.

31. Как можно изменить цвет выводимых на экран символов?

A) Textcolorscr(color:byte);
B) Textcolor(color:byte); *
C) Textbackground(color:byte);
D) Color(color:byte);
E) Colortext(color:byte).

32. Определить результат выполнения программы:

```
x:=sqr(-5);
if x<=0 then
y:=x-10 else
y:=x-15;
```

writeln(y);

A) -20;
B) -10;
C) 20;
D) 15;
E) 10. *

33. Выберите стандартную функцию, которая используется для возведения числа x в квадрат

A) Sqr(x); *
B) Succ(x);
C) Sgr(x);
D) Sqrt(x);
E) Dec(x).

34. Укажите диапазон значений переменной, имеющей тип integer в Turbo Pascal

A) -32768..32767; *
B) -128..127;
C) 0..65535;
D) 0..255;
E) -2147483648..2147483647.

35. Выберите правильную запись числа 2,34 в экспоненциальной форме

A) 2.34;
B) 234E0;
C) 0.234E1; *
D) 2.34E-1;
E) 0.234E-1.

36. Определить, что будет напечатно программой:

```
b:=4;
for j:=1 to b do
    x:=j;
writeln(x-b+7);
```

A) 10;
B) -7;
C) 4;
D) 13;
E) 7. *

37. Укажите, какое служебное слово используется для обозначения цикла с предусловием?

A) Until;
B) Repeat;
C) For;
D) Case;
E) While. *

38. Укажите в каком варианте ответа перечислены только беззнаковые целые типы данных

A) Integer, longint;
B) Shortint, longint;
C) Word, shortint;
D) Integer, byte;
E) Word, byte. *

39. Слова, смысл и способ употребления которых задан раз и навсегда называют

A) Командами;
B) Составными словами;
C) Серией;
D) Служебными словами; *
E) Простыми словами.

40. Укажите размер в байтах, занимаемый переменной типа real

A) 6; *
B) 4;
C) 8;
D) 10;
E) 2.

41. В операторе цикла

while b do a;

Выберите верное утверждение для случая, когда b=false

A) Оператор A выполнится один раз;
B) Цикл записан неверно, произойдет ошибка;
C) Цикл выполняется пока B не равно true;
D) Произойдет зацикливание программы;
E) Оператор A не выполнится ни разу. *

42. С помощью какой процедуры можно перевести курсор на любую точку экрана?

A) Insgoto(x,y:byte);
B) Clreol(x,y:byte);
C) Movegoto(x,y:byte);
D) Gotoxy(x,y:byte); *
E) Goto(x,y:byte).

43. Определите результат выполнения фрагмента программы:

```
b:=-4;
x:=6+b;
case x of
    1: begin        writeln(x-1); halt; end;
    2: writeln(x+8);
end;
```

A) 1;
B) 10; *
C) x-1;
D) x+8;
E) 18.

44. Что будет напечатано программой:

```
program chis;
var a, b,c,sum:integer;
begin
    a:=8; b:=2; c:=5;
    write(a:3); write(b:3); write(c:3);
writeln;
```

sum:=a+b+c;
writeln(‘Сумма чисел равна ’,sum);
end.

A) 8
   2
   5 Сумма чисел равна 15;
B) 8 2 5
   Сумма чисел равна 15; *
C) 8 2 5 Сумма чисел равна 15;
D) 8 Сумма чисел равна 8
   2 Сумма чисел равна 10
   5 сумма чисел равна 15;
E) 8 2 5
   Сумма равна 15.

45. Выберите стандартную функцию, которая используется для вычисления корня квадратного из числа x

A) Square(x);
B) Sqrt(x); *
C) Sqr(x);
D) Succ(x);
E) Dec(x).

46. Вычислите значение выражения:

(5*6<7,2) or (8,7>-4)

A) 1;
B) False;
C) True; *
D) Неверный синтаксис выражения;
E) 0.

47. Определите значение переменной S после выполнения следующих операторов

s:=3;
for i:=1 to 5 do;
s:=s+i;
writeln(s);

A) 1;
B) 15;

C) 18;
D) 3; *
E) 5.

48. Укажите тип данных, который не относится к вещественным типам:

A) Real;
B) Single;
C) Shortint; *
D) Extended;
E) Double.

49. Выберите стандартную функцию, которая используется для вычисления экспоненты числа x

A) Exponenta(x);
B) Exp(x); *
C) In(x);
D) Trunk(x);
E) Sqr(x).

50. Какое утверждение неверно

A) Программа начинается со служебного слова begin;
B) Оператор присвоения обозначается знаком равенства (=); *
C) Операторы должны располагаться справа от знака присвоения;
D) Точка с запятой (;) используется для завершения оператора
E) Точка с запятой (;) называется пустым оператором

51. Укажите, каким образом в цикле for можно установить значение шага приращения параметра цикла, отличного от 1 или -1.

A) Можно просто изменять значение параметра цикла внутри цикла так, как это необходимо;

B) Можно добавить служебное слово step и указать величину шага:
C) Шаг приращения задается параметром цикла;
D) Можно включить в тело цикла выражение вида i:=i+шаг, где i – параметр цикла;
E) Шаг приращения параметра цикла может быть только 1 или -1. *

52. Выберите верное утверждение

A) Оператор проверки условия – это while .. do;
B) Ключ выбора может иметь тип string;
C) Оператор выбора – это if .. then;
D) В операторе выбора допускается не более 10 вариантов выбора;
E) Оператор выбора – это case .. of . *

53. Определите результат выполнения фрагмента программы

```
x:=5;
if x<=0 then
begin
    x:=-10;
    writeln(x+20)
end else writeln(x+35);
```

A) x+35;
B) 40; *
C) x+20;
D) 25;
E) 10.

54. Укажите стандартную функцию, которая возвращает результат целого типа

A) Sin(x);
B) In(x);
C) Odd(x);
D) Ord(x); *
E) Exp(x).

55. Процессор выполняет команды алгоритма, записанные

A) На алгоритмическом языке;

B) На командном языке;
C) В виде блок-схемы;
D) На естественном языке;
E) На машинном языке (в двоичном коде) . *

56. Определите значение переменной S после выполнения следующих операторов:

```
s:=0;
n:=1;
for  i:=2 to n do
     s:=s+1/i;
```

A) 0.5;
B) 0; *
C) 2.5;
D) 3;
E) 1.

57. Укажите диапазон значений переменной, имеющей тип byte Turbo Pascal

A) -32768..32767;
B) 0..225; *
C) 0..65535;
D) -128..127;
E) -2147483648..2147483647.

58. Укажите диапазон значений переменной, имеющей тип integer в Turbo Pascal

A) -32768..32767; *
B) 0..225;
C) 0..65535;
D) -128..127;
E) -2147483648..2147483647.

59. Как создать окно в текстовом режиме?

A) Window(x1:x2, y1:y2);
B) Window(x1, y1: byte);

C) CurrentWindow(x1, y1, x2, y2);
D) NewWindow(x1, y1, x2, y2: byte);
E) Window(x1, y1, x2, y2: byte). *

60. Какое утверждение неверно

A) Константа может входить в состав арифметического выражения;
B) Выражение может стоять в виде отдельного оператора в программе; *
C) Переменные и константы должны быть описаны перед первым использованием;
D) Выражение может стоять справа от знака присваивания;
E) Переменная может входить в состав логического выражения.

61. Укажите сколько символов в именах идентификаторов в Turbo Pascal являются значащими.

A) Не ограничено;
B) 127 символов;
C) 63 символа;
D) 255 символов; *
E) 31 символ.

62. Выберите выражение, в результате которого получено значение c=3, если a=14 и b=4.

A) C:=a/b;
B) C:=a mod b;
C) C:=b mod a;
D) C:= b div a;
E) C:= a div b. *

63. Укажите, какой комментарий записан верно с точки зрения синтаксиса языка Turbo Pascal

A) // это комментарий;
B) <!--это комментарий-->;
C) /* это комментарий */;
D) [*это комментарий *];
E) (*это комментарий *). *

64. Укажите, какой тип из перечисленных не может иметь управляющая переменная цикла for

A) Boolean;
B) Char;
C) String; *
D) Integer;
E) 0..10.

65. Как можно вставить новую строку в текст?

A) Outtext;
B) OuttextXY;
C) DelLine;
D) InsLine; *
E) LineTo.

66. Выберите правильный вариант расположения чисел, напечатанных следующим фрагментом программы:

write(1); write(2, 3); writeln(4); write(5, 6); writeln; write(7, 8);

A) 123
456
78;
B) 1234
56
78; *
C) 123456
78;
D) 1234
5678;
E) 1
23
45678.

67. Укажите вариант записи выражения, истинного при выполнении указанного условия и ложного в противном случае:

x принадлежит интервалу (2, 10) или (-2, 2)

A) (x>2) and (x<=10) or |x|<2;

B) (x>2) or (x<=10) and |x|<2;
C) (x<2) and (x<=10) or abs(x)>2;
D) (x>2) or (x<=10) and abs(x)<2;
E) (x>2) and (x<10) or abs(x)<2. *

68. Выберите верное утверждение:

A) Служебное слово elseif используется для начала вложенного оператора проверки условия;
B) После слова then в операторе if может стоять только один оператор;
C) В операторе выбора не может быть использован раздел else;
D) После слова then в операторе if может стоять несколько операторов; *
E) В качестве ключа выбора в операторе case может использоваться переменная вещественного типа.

69. Укажите верно записанное логическое выражение, если x=true, y=true

A) x and y =false;
B) x or y =true; *
C) not x =true;
D) not y = true;
E) x or y = false.

70. Определить результат выполнения программы

```
x:=4;
s:=0;
while x>0 do begin
s:=s+x;
x:=x div 2;
end;
writeln(s);
```

A) 3;
B) 2;
C) 0;
D) 1;
E) 7. *

71. Выберите вариант, который правильно описывает значение параметра цикла после завершения цикла

A) Не существует;
B) Равно нулю;
C) Равно конечному значению параметра цикла; *
D) Равно начальному значению параметра цикла;
E) Равно единице.

72. Укажите, какое служебное слово используется для обозначения цикла с постусловием

A) While;
B) If;
C) For;
D) Case;
E) Repeat. *

73. Определить результат выполнения программы

```
z:=0; x:=1; y:=1;
if x>0 then
if y>0 then z:=1 else z:=2;
write(z);
```

A) 1; 1
B) 3;
C) 2;
D) 0;
E) 4.

74. Определить результат выполнения программы

```
z:=0; x:=-1; y:=1;
if x>0 then
if y>0 then z:=1 else z:=2;
write(z);
```

A) 1;
B) 3;
C) 2;
D) 0; *

E) 4.

75. Определить результат выполнения программы

```
z:=0; x:=1; y:=-1;
if x>0 then
if y>0 then z:=1 else z:=2;
write(z);
```

A) 1;
B) 3;
C) 2; *
D) 0;
E) 4.

76. Укажите неверно записанный оператор среди перечисленных вариантов

A) If (a>b) or c then c:=false;
B) If k<>m then k:=m;
C) If x and y then s:=s+1; else s:=s-1; *
D) If b then a:=d else a:=c;
E) If a<b then a:=a*a else b:=b*b.

77. Укажите диапазон значений переменной, имеющей тип word в Turbo Pascal

A) 0..255;
B) -128..127;
C) 0..65535; *
D) -2147483648..2147483647;
E) -32768..32767.

78. Укажите, какой из перечисленных в вариантах ответов операторов имеет наинизший приоритет

A) +;
B) =; *
C) Not;
D) Div;
E) And.

79. Укажите, какой тип имеет результат логической операции

A) Диапазон значений;
B) Логический; *

C) Строковый;
D) Символьный;
E) Целочисленный.

80. Что определяет переменная i в цикле?

```
for  i:=a1 to a2 do s;
```

A) начальное значение;
B) конечное значение;
C) порядковый номер; *
D) шаг;
E) результат.

81. Определите результат выполнения фрагмента программы:

```
j:=0; s:=0;
repeat
        j:=j+1;
        s:=s+2*j;
until j=4;
writeln (s);
```

A) 8;
B) 12;
C) 20; *
D) 0;
E) 10.

82. Необходимо переменной u присвоить большее из значений x и y. Какое из приведенных решений является неверным?

A) u:=x; if x<y then u:=y;
B) if x<y then u:=y else u:=x;
C) if x>y then u:=x else u:=y;
D) if x>y then u:=x; u:=y; *
E) u:=y; if x>y then u:=x;

83. Выберите верное утверждение для цикла:

for i:=a to b do s;

если значение B меньше, чем значение A;

A) Произойдет ошибка выполнения программы;
B) Произойдет ошибка компиляции программы;
C) Значение счетчика цикла будет меняться от большего значения к меньшему;
D) Оператор s не выполнится ни разу; *
E) Оператор s выполнится один раз.

84. Имеются следующие описания:

```
type digit='0'..'9';
var d:digit;
k:0..9;
```

Укажите недопустимое присваивание

A) K:=3;
B) D:=5; *
C) D:='0';
D) D:='7';
E) K:=ord(d).

85. Определите значение переменной S после выполнения следующих операторов

```
s:=0;
i:=0;
while i<5 do
        i:=i+1;
s:=s+i;
```

A) 1;
B) 4;
C) 5; *
D) 2;
E) 0.

86. Выберите верный результат выполнения программы:

```
program aba;
var a, b:integer;
begin
        read(a, b, a);
        writeln(a, b, a);
```

end.

Если для ввода заданы числа 1, 2, 3

A) 121;
B) 123;
C) 323; *
D) 321;
E) 232.

87. Выберите идентификатор, который используется для обозначения символьного типа данных

A) Symbol;
B) Char; *
C) String;
D) Varchar;
E) Comp.

88. Выберите оператор, реализующий конструктцию неполного ветвления

A) if условие then оператор1 elseif условие then оператор2;
B) if условие then оператор; *
C) if условие then оператор1 else оператор2; if условие then оператор3;
D) if условие then оператор1 else if условие then Оператор2;
E) if условие then оператор1 else опреатор2.

89. Укажите, какой из перечисленных в вариантах ответов операторов имеет наивысший приоритет

A) *;
B) /;
C) +;
D) And;
E) Not. *

90. Имеется описание:

var s: (book, pen, pencil);

Укажите значение, которое вернет функция ord(s), если переменная s имеет значение равное pen.

A) 3;
B) 2;
C) 0;
D) Book;
E) 1. *

91. Укажите правильную запись вычисления функции в виде одного условного оператора:

Y= {cos(2x), если 0<x<2
{1-sin(3x) иначе

A) If (0<x) and (x<2) then y:=cos(2*x) else y:=1-sin(3*x); *
B) If 0<x<2 then y:=cos(x) else y:=1-sin(x);
C) If (0<x) and (x<2) then y:=cos(2*x) else y:=1-3*sin(x);
D) If (0<x) and (x<2) then y:=cos(2x) else y:=1-sin(3x);
E) If (0<x) or (x<2) then y:=cos(x) else y:=1-sin(x).

92. Выберите выражение, которое истинно для точки с координатами (x, y) лежащей внутри круга радиусом 2 и центром в начале координат.

A) x*x+y*y>=2
B) x+y>2
C) x*2+y*2<=4
D) sqr(x)+sqr(y)<=4 *
E) sqr(x*y)<=4

93. Выберите правильное описание переменной перечисляемого типа:

A) Var color=[‘black, ‘red, ‘white];
B) Var color=[‘black’, ‘red’, ‘white’];
C) Var color=set of (black, red, white);
D) Var color=(‘black’, ‘red’, ‘white’);
E) Var color: (black, red, white). *

94. Определите значение переменной S после выполнения следующих операторов:

```
s:=0; i:=1;
repeat
        s:=s+1/i;
```

```
        i:=i-1;
    until  i<=1;
```

A) 2;
B) 0;
C) 4;
D) 5;
E) 1. *

95. Выберите выражение, в результате которого получено значение c=2, если a=14 и b=4

A) C:=a/b;
B) C:=b div a;
C) C:=a mod b; *
D) C:=b mod a;
E) C:=a div b.

96. Укажите вариант ответа, в котором указано, с каких символов может начинаться имя идентификатора в Turbo Pascal

A) A..Я, a..я (буквы русского алфавита);
B) - (знак минус);
C) _ (символ подчеркивания) ; *
D) *(звездочка);
E) 0..9 (цифры).

97. Имеется фрагмент программы:

```
    case num of
        …
    end;
```

Укажите, какой тип не может иметь переменная num

A) String; *
B) Char;
C) Integer;
D) Boolean;
E) 1..10.

98. При выполнении оператора:

readln(a, b, c);

пользователь должен ввести с клавиатуры следующую последовательность символов:

A) 123;
B) 1.2. 3.;
C) 1, 2, 3;
D) 1 2 3; *
E) 1; 2; 3;.

99. Укажите тип данных, который не относится к целым типам данных

A) Extended; *
B) Word;
C) Byte;
D) Longint;
E) Integer.

3. ПРОЦЕДУРЫ И ФУНКЦИИ, КАК СРЕДСТВА СТРУКТУРНОГО ПРОГРАММИРОВАНИЯ

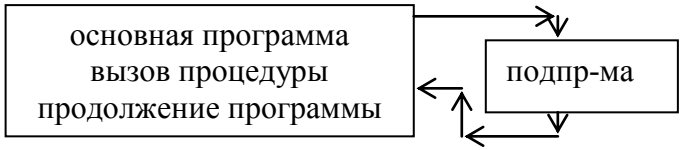
3.1. Процедуры

В практике программирования часто приходится выполнять одни и те же действия, но при различных исходных данных. Чтобы исключить повторения и сделать программу проще и понятнее можно выделить эти повторяющиеся вычисления в самостоятельную часть программы, которая может быть использована много кратно по мере необходимости. Такая автономная часть программы, реализующая определенный алгоритм и допускающая обращение к ней из различных частей общей программы называется подпрограммой.

Положительные стороны использования подпрограмм:

- ☑ Применения подпрограмм позволяет разрабатывать одну программу сообща группой программистов. Чтобы облегчить эту задачу, имена переменных в основной программе и в подпрограммах сделаны независимыми друг от друга.
- ☑ Подпрограммы оформляются в виде замкнутых участков программы, имеющих четко обозначенные вход и выход, поэтому программа становится легко читаемой и понятной.
- ☑ К положительным сторонам подпрограмм относится и экономия памяти, которая получается из-за того, что память для хранения переменных, используемых в подпрограммах, выделяется только на время работы подпрограммы.
- ☑ Использование подпрограмм позволяет реализовать один из самых прогрессивных методов программирования – структурное программирование.
- ☑ В структурированных программах легко прослеживается основной алгоритм, их нетрудно понять любому читателю, они проще в отладке и менее чувствительны к ошибкам. Языки, в которых предусмотрены такие механизмы, называются структурированными.

☑ Примеры, которые мы выполняли, небольшие, следовательно, их можно написать без подпрограмм. Иное дело – крупные программы. Писать их как единое целое без расчленения на самостоятельные фрагменты невозможно.



В языке Паскаль имеется два вида подпрограмм – процедуры (procedure) и функции (function).

Процедурой в языке Паскаль называется особым образом оформленный фрагмент программы, имеющий собственное имя. Упоминание этого имени в тексте программы приводит к активизации процедуры и называется ее вызовом. Для обмена информацией между основной программой и процедурой используется несколько параметров вызова или нет ни одного.

procedure имя (формальные параметры);
раздел описаний
begin
раздел операторов
end;

Вызов процедуры:
имя процедуры (фактические параметры);

Пример оформления процедуры:

```
procedure sum (a, b: integer; var y, z: real);
    const n=10;
    var i:integer;
begin
    for i:=1 to n do
            begin
                    y:=a+b*i;
                    z:=4*y-i
            end;
```

```
end;
begin
    sum (3.5, 7.6, y1, z);  {вызов процедуры}
end.
```

Формальные параметры представляет собой список переменных с указанием типа. Для выделения выходных параметров перед ними ставится слово **var.** В примере с процедурой входные параметры – это a, b, а выходные – y, z.

3.2. Функции

Функция – вид подпрограммы отличающейся от процедуры тем, что результат ее работы возвращается в виде значения функции, следовательно, вызов функции может использоваться в выражениях.

Отличительные особенности функции – только один результат выполнения (хотя функция может иметь несколько входных параметров), результат обозначается именем функции и передается в основную программу.

```
function имя (формальные переменные): тип;
раздел описаний
begin
    раздел операторов
end;
```

Вызов функции:

```
Имя функции (фактические параметры);
```

Вызывается функция по имени с указанием фактических параметров. Вызов функции можно делать внутри выражения.

Пример 68. Пример оформления функции: Вычислить разность факториалов F=m!-k!.

```
function fact (n: integer): real;
var i:intger;
    p:real;
begin
    p:=1;
```

```
    for i:=1 to n do
                p:=p*I;
        fact:=p
    end;
    begin
        F:=fact(m)-fact(k) {вызов функции}
    end.
```

Формальные параметры могут быть либо параметрами-значениями либо параметрами-переменными, либо параметрами-константами.

Например: **procedure** myproc (a, b:integer; var c, d: real; const e: string);

a, b – параметры-значения – входные данные после слова **var** определяются параметры-переменные – c, d – выходные данные, после слова **const** определяются параметры-константы – входные данные.

У входных и выходных разная организация компьютерной памяти. Для того, чтобы понять, в каких случаях использовать тот или иной тип параметра, рассмотрим как осуществляется замена формальных параметров на фактические в момент обращения к подпрограмме.

Если параметр определен как параметр-значение, то перед вызовом подпрограммы это значение вычисляется, полученный результат копируется во временную память и передается подпрограмме. Важно учесть, что даже если в качестве фактического параметра указано простейшее выражение в виде переменной (константы), все равно **подпрограмме будет передана лишь копия переменной (константы)**. Любые возможные изменения в подпрограмме параметра-значения никак не воспринимаются вызывающей программой, так как в этом случае изменяется копия фактического параметра.

Если параметр определен как параметр-переменная, то при вызове подпрограммы передается сама переменная, а не ее

копия (фактически в этом случае подпрограмме передается адрес переменной). Изменение параметра-переменной приводит к изменению самого фактического параметра в вызывающей программе.

В случае параметра-константы в подпрограмму также передается адрес области памяти, в которой располагается переменная или вычисленное значение. Однако компилятор блокирует любые присваивания параметру-константе нового значения в теле подпрограммы.

Представленный ниже пример поясняет изложенное. В программе задаются два целых числа 5 и 7, эти числа передаются процедуре, в которой они удваиваются. Один из параметров передается как параметр-переменная, другой – как параметр-значение. Значения параметров до и после вызова процедуры, а также результат их удвоения выводятся на экран.

Пример 69.

```
Const
    A:integer=5;
    B:integer=7;
Procedure inc2 (var c:integer; b:integer);
Begin
    C:=c+c;
`   b:=b+b;
    Writeln(‘удвоенные: ’, c:5, b:5);
End;
Begin
    Writeln(‘исходные: ’, a:5, b:5);
    Inc2(a,b);
    Writeln(‘результат: ’, a:5, b:5);
End.
```

В результате будет выведено:

Исходные:	5	7
Удвоенные:	10	14
Результат:	10	7

Имена, объявленные в разделе описаний основной программы, действуют в разделе операторов основной программы и в любой подпрограмме. Эти имена называются *глобальными.* Имена, объявленные в подпрограмме, действуют только в этой подпрограмме и в любой объявленной в ней процедуре и функции. Такие имена называются *локальными***.** Они недоступны для операторов основной программы.

Описание процедур и функций следует сразу после описания переменных.

```
program p;
    раздел описаний глобальных переменных
function f(x): real;
    раздел описаний локальных переменных
begin
    ….
end;
procedure p1;
    раздел описаний локальных переменных
begin
    ….
end;
procedure p2;
    раздел описаний локальных переменных
begin
    ……
end;
begin
    основная программа
end.
```

Пример 70.

```
program stepen;
var
    x, y:real;
function power (a, b: real): real;
```

```
begin
    if a>0 then
            power:=exp(b*ln(a))
    else if a<0 then
            power:=exp(b*ln(abs(a)))
            else if b=0 then
                    power:=1
                    else
                    power:=0
end;
begin
    repeat
            readln(x, y);
            writeln(power(x, y):12:10, power(x, -y):15:10)
    until eof
end.
```

Для выхода из программы необходимо нажать **ctrl+enter**

3.3. Рекурсивные функции

Рекурсия – это такой способ организации вычислительного процесса, при котором подпрограмма в ходе выполнения составляющих ее операторов обращается сама к себе.

Классический пример – вычисление факториала. Программа вводит с клавиатуры целое число n и выводит на экран значение n!. Для выхода из программы необходимо нажать **ctrl+enter**.

Пример 71.

```
program factor;
var
    n: integer;
function fac (n: integer): real;
begin
    if n<0 then
            writeln('Ошибка, n<0')
    else
            if n=0 then
```

```
                fac:=1
            else
                fac:=n*fac(n-1)
    end;
    begin
        repeat
            readln(n);
            writeln('n!=',fac(n))
        until eof
    end.
```

Пример 72. Вычислить N-е число Фиббоначчи. (Смотри тему Циклы)

```
program Fib;
var n:byte;
function F(k:byte):word;
begin
   if k<2 then F:=1 else F:=F(k-1)+F(k-2); {рекурсивный вызов}
end;
begin
   write('введите номер числа Фиббоначчи ');
   readln(N);
   writeln(N,'-е число Фиббоначчи =',F(N));
   readln
end.
```

ПРАКТИЧЕСКИЕ ЗАДАНИЯ

1. ** Написать программу, вычисляющую значение функции в заданном диапазоне с заданным шагом. Предусмотреть область определения и неправильный ввод, когда левая граница диапазона больше, чем правая. Использовать подпрограммы. Построить блок-схему к программе.

   a) $$y = \begin{cases} x^2, \cdot при \cdot x \le -3 \\ \frac{5}{x}, \cdot при \cdot -3 < x \le 2 \\ 2x, \cdot при \cdot x > 2 \end{cases}$$

   b) $$y = \begin{cases} x^2, \cdot при \cdot x \le -2 \\ 5x, \cdot при \cdot -2 < x \le 5 \\ 2\sqrt{x}, \cdot при \cdot x > 5 \end{cases}$$

2. * Вычислить значение заданной функции на заданном интервале с заданным шагом. Использовать подпрограммы.

3. ** Используя функцию вычислить значение F: $F = \frac{S(10)}{S(20)}$, если $S = \frac{1}{2 \cdot 3} + \frac{2}{3 \cdot 4} + \frac{3}{4 \cdot 5} + \ldots + \frac{n}{(n+1) \cdot (n+2)}$

4. *** Используя функцию вычислить значение C. Если число n больше 0, вычисляет n!, в противном случае – на экран выдается соответствующее сообщение. Значения n и m формируются случайным образом. $C = \frac{n!}{m!(n-m)!}$

Смотрите примеры 68-70

5. * Написать программу вычисления корней квадратного уравнения вида $ax^2+bx+c = 0$. Значения a, b и c вводить в режиме диалога. Предусмотреть проверку существования корней уравнения и выдать соответствующие сообщения. Для решения задачи использовать процедуру.

6. ** Используя функцию вычислить: $F = \frac{S(10)}{S(20)}$, если $S = \frac{1}{2\cdot 3} + \frac{1}{4\cdot 5} + \frac{1}{6\cdot 7} + \ldots + \frac{1}{2n\cdot(2n+1)} + \ldots$
7. ** Используя функцию вычислить значение F: $F = \frac{S(10)}{S(20)}$, если $S = \frac{1}{2} + \frac{1}{3} + \frac{1}{4} + \ldots + \frac{1}{(n+1)}$.
8. *** Используя функцию вычислить: $S = \frac{\sum_{i=1}^{10} m! - \sum_{i=1}^{20} n!}{(m+n)!}$ где n, m – случайные целые числа в диапазоне от 5 до 15

Смотрите примеры 68-70

9. *** В режиме диалога вводится значение **y**. Вычислить Z

$$Z = \frac{1.7 * f(0.25) + 2 * f(1+y)}{6 - f(y^2 - 1)} \qquad f(x) = \frac{\frac{x}{1!} + \frac{x^3}{3!} + \frac{x^5}{5!}}{\frac{x^2}{2!} + \frac{x^4}{4!} + \frac{x^6}{6!}}$$

Смотрите примеры 68-70

10. *** В режиме диалога вводится значение **y**. Вычислить Z

$$Z = \frac{f(0.34) + 1.75 * f(y+5)}{f(y^4 - 1)} \qquad f(x) = \frac{\sum_{k=0}^{10} \frac{x^{2k+1}}{(2k+1)!}}{\sum_{k=0}^{20} \frac{x^{2k}}{2k!}}$$

Смотрите примеры 68-70

11. ** Автобус за смену делает от 5 до 15 рейсов. За рейс перевозит от 30 до 230 пассажиров. Стоимость проезда одного пассажира 25 тенге. Вероятность того, что у пассажира удостоверение, проездной или он «заяц» - 30%. Определить доход, полученный за N смен. Число смен вводится в режиме диалога.
12. ** Цеху необходимо выполнить объем работ не менее 20 тыс. единиц продукции. Каждый день на работу выходит от

13. 10 до 20 рабочих. Производительность каждого рабочего от 5 до 15 единиц в день. Сколько дней необходимо затратить на выполнение всего объема работ.
14. *** Нарисуйте на экране три одинаковых домика из звездочек.

Смотрите примеры предыдущего раздела и приложения

15. *** Нарисуйте на экране три одинаковых человечка из различных символов.

Смотрите примеры предыдущего раздела и приложения

16. ** На экономическом факультете первого курса 15 групп, каждая из которых во время сессии должна сдать от 3 до 5 экзаменов. В каждой группе от 15 до 30 студентов. Вероятность того, что студент успешно сдал экзамен – 80%. Найти число успевающих и число неуспевающих студентов на факультете.
17. ** Необходимо связать кофту. На вязание одного ряда требуется от 50 до 70 грамм шерсти. Всего нужно связать 1200 рядов. В наличии имеется 2,5 килограмма шерсти. Хватит ли имеющейся шерсти для того, чтобы связать кофту. Если не хватит, то сколько рядов останется довязать и сколько потребуется еще шерсти.
18. ** Найти общий расход топлива N автоколонн за неделю. Известно, что каждая автоколонна состоит из 20 до 100 автомобилей. Каждый автомобиль за один день проезжает путь длиной от 5 до 150 километров. Расход топлива одного автомобиля от 20 до 28 литров на 100 километров. Вероятность того, что автомобиль неисправен – 10%.
19. ** В КСК «Солнечный» числится 15 домов. В каждом доме от 40 до 80 квартир. Вероятность того, что жильцы оплатили услуги «Водоканала» за холодную воду – 25%. В каждой квартире прописано от 1 до 6 человек. Тариф за холодную воду – 155 тенге в месяц за человека. Найти общую сумму долга по КСК «Солнечный».

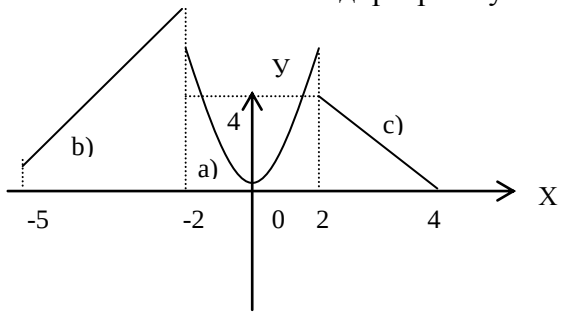
20. * Определить, является ли введенное пользователем число четным. Предусмотреть ошибку ввода, т.е. если число не является целым, выдать сообщение, что дробное число не может быть четным и попросить пользователя ввести число заново. Использовать операторы условия и безусловного перехода.
21. * Определить по номеру месяца соответствующее время года. Предусмотреть ошибку ввода пользователем, т.е. если пользователь вводит число, отличное от допустимого номера месяца, выдать сообщение об ошибке и попросить ввести число заново. Использовать операторы выбора и безусловного перехода.
22. ** Составить программу вычисления процента выполнения нормы выработки каждым членом бригады и всей бригадой. Число рабочих в бригаде вводится в режиме диалога. Каждый рабочий за смену может изготовить от 20 до 60 изделий. Средняя норма выработки одним рабочим составляет 40 изделий.
23. ** Цех выпускает 5 наименований продукции. Число единиц каждой продукции от 70 до 100 стоимостью от 200 до 2500 тенге каждое наименование. Вероятность того, что данная единица продукции реализована – 80%. Найти общую сумму выручки. Использовать подпрограмму.
24. ** Сколько необходимо затратить времени, чтобы собрать 1 килограмм икры. Для того чтобы поймать одну рыбу необходимо затратить от 5 до 15 минут. Вероятность того, что рыба с икрой – 30%. В одной рыбе может быть от 30 до 70 грамм икры. Использовать подпрограмму.
25. ** Бригаде рабочих необходимо выполнить объем работ, составляющий A единиц продукции. Число рабочих в бригаде N человек. Производительность каждого рабочего 10-20 единиц в день. Сколько дней необходимо на выполнение всего объема работ. Величины A и N вводятся в режиме диалога. Использовать подпрограмму.

26. * Написать программу для вычисления суммы квадратов натуральных чисел из заданного в режиме диалога с пользователем диапазона. Использовать цикл с параметром. Использовать подпрограмму.
27. ** Реализовать игру, суть которой в следующем: компьютер задумывает число от -5 до 4 и просит пользователя угадать это число, предоставив пользователю три попытки. Если пользователь угадывает, компьютер поздравляет его с победой, а если нет – выдает сообщение о проигрыше. Использовать подпрограмму.
28. * Вычислить сумму: $S = \frac{1}{2} + \frac{1}{4} + \frac{1}{6} + ... + \frac{1}{100}$. Использовать цикл с предусловием. Использовать подпрограмму.
29. ** Найти значение функции $Y = \frac{1}{x^2 + \frac{5}{\sin x}}$ при вводимом пользователем значении независимой переменной. Использовать подпрограмму (функцию).
30. * С клавиатуры вводятся числа. Выписать из них все числа, кратные 7-и, рассчитать сумму отрицательных чисел, количество четных чисел. Использовать подпрограмму.
31. * Изменить предыдущую задачу так, чтобы числа вводились не с клавиатуры, а генерировались компьютером случайно в диапазоне от -12 до 26. Использовать подпрограмму.
32. * Изменить предыдущую задачу так, чтобы числа принадлежали заданному диапазону с заданным шагом.
33. * С клавиатуры в строку вводится предложение из символов. Найти количество всех символов (включая пробелы), количество слов и количество букв ‘а’. Использовать подпрограмму.
34. *** Используя рекурсию, вычислить значение C. Если число n больше 0, вычисляет n!, в противном случае – на экран

35. выдается соответствующее сообщение. Значения n и m формируются случайным образом. $C = \frac{n!}{m!(n-m)!}$

Смотрите примеры 71, 72

36. ** Определить, попадет ли точка с заданными координатами внутрь квадрата, сторону и координаты верхнего левого угла которого задает пользователь. Использовать подпрограмму.
37. ** Найти значение функции, изображенной на рисунке 3.1, при заданном пользователем значении независимой переменной. Использовать подпрограмму.



a) $f(x) = x^2$
b) $f(x) = 2x + 6$
c) $f(x) = -3x + 2$

Рисунок 3.1. Иллюстрация к задаче 33

38. *** Используя рекурсию, вычислить: $S = \frac{\sum_{i=1}^{10} m! - \sum_{i=1}^{20} n!}{(m+n)!}$ где n, m – случайные целые числа в диапазоне от 5 до 15

Смотрите примеры 71, 72

39. * Целое число в пределах от -6 до 11 генерируется случайно. Если это число меньше нуля, то возвести его в пятую степень, если число кратно трем и положительно, выдать результат деления его на три, а в остальных случаях выдать

40. сумму данного числа с его квадратным корнем. Использовать подпрограмму.
41. * Школьная отметка обозначается цифрой и имеет название. Написать программу, которая определяет по значению отметки, записанной цифрой ее название. Учесть ошибку ввода пользователя, т.е. если число не соответствует школьной отметке, выдать результат об ошибке и попросить пользователя ввести отметку правильно. Использовать операторы выбора и безусловного перехода. Использовать подпрограмму.
42. * Используя оператор цикла с постусловием, вывести таблицу значений градусов температуры по Цельсию и
43. Фаренгейту. Формула перевода °Ф=°С*1,8+32. Диапазон значений с шагом задает пользователь. Использовать подпрограмму.
44. * По номеру месяца определить количество дней в нем, использовать следующие данные:
    a) январь, март, май, июль, август, октябрь, декабрь – 31 день;
    b) апрель, июль, сентябрь, ноябрь – 30 дней;
    c) в феврале, если год високосный – 29 дней, если нет – 28 дней.

    Использовать подпрограмму.
45. * Вывести на экран две горизонтальные строчки, внизу и вверху экрана из разноцветных звездочек. Использовать подпрограмму.
46. * Подсчитать значение функции в заданном интервале с заданным шагом и найти сумму этих значений. Функцию организовать в виде подпрограммы. y=sinx+2cos5x.
47. *** Напишите программу, предлагающую пользователю меню из десяти функций, и выдающую таблицу значений выбранной пользователем функции.

Смотрите примеры 68-70

48. **** Напишите программу для младших школьников, проверяющую знание ими таблицы умножения от 2 до 12. Учащемуся задаются 5 примеров перемножения случайных чисел в заданном интервале. Оценкой является количество правильных ответов. Используйте подпрограмму для печати замечаний в ответ на каждый результат, вводимый пользователем. За правильный ответ— замечание должно быть поощрительным, за неправильный — сожалеющим. Чтобы сделать опрос более интересным, необходимо заготовить по десять замечаний для правильных и неправильных ответов и выбирать их случайным образом, обращаясь при этом к пользователю по имени, запрошенному в начале программы. Сделайте красочную заставку, легкое музыкальное сопровождение тоже не будет лишним.
49. *** Напишите программы, находящие минимальное и максимальное значения из трех чисел X, Y и Z, введенных с клавиатуры. Используя их в качестве подпрограмм, напишите программу, вычисляющую значение следующей функции:

$$S = \frac{\max(x,y,z) - 2^x \min(x,y,z)}{\sin 2 + \frac{\max(x,y,z)}{\min(x,y,z)}}$$

Смотрите примеры 71, 72

ТЕСТОВЫЕ ЗАДАНИЯ ДЛЯ ПРОВЕРКИ ТЕОРЕТИЧЕСКИХ ЗНАНИЙ

1. **Имя функции используется только для …**
   A) Вызова функции;
   B) Передачи параметров;
   C) Вызова функции и возврата результата; *
   D) Выполнения некоторой последовательности действий в теле программы;
   E) Получения результата.
2. **Укажите принципиальное отличие функции от процедуры:**
   A) Функция возвращает значение, а ее вызов может использоваться в программе как переменная в выражении; *
   B) Описание функций должно предшествовать ее вызову;
   C) Функция может быть многократно вызвана из разных частей главной программы;
   D) В функциях описываются собственные метки, константы, типы, собственные переменные;
   E) В заголовке функции указывается список формальных параметров.
3. **Записан текст программы. Локальные параметры описываются в строчке:**
   1) var a,b:integer;
   2) procedure zb1;
   3) var a:integer;
   4) begin a:=sqr(5); b:=15; end;
   5) begin a:=18; b:=37;
   6) zb1;
   7) writeln(‘a3=’,a,’b3=’,b);
   A) 5;
   B) 1;
   C) 2;
   D) 3; *

E) 4.

4. Признаком конца процедуры является:

A) Procedure;
B) End; *
C) Var;
D) End.
E) ;.

5. Укажите, в каком разделе программы переменная не может быть описана

A) В заголовке подпрограммы;
B) В разделе объявления переменных подпрограммы;
C) В разделе объявления переменных программы;
D) В разделе объявления переменных модуля;
E) В заголовке модуля. *

6. В результате работы программы на экран монитора будет выведено:

```
var x:real;
procedure tr58;
begin      writeln(x:6:2); end;
begin      x:=pi;  tr58; end.
```

A) x:=3.14;
B) 3.14; *
C) 0;
D) Pi;
E) Ничего не выведет.

7. Алгоритмы, целиком используемые в составе других алгоритмов, называются…

A) Разветвляющимися;
B) Определенными;
C) Циклическими;
D) Вспомогательными; *
E) Линейными.

8. Параметр-переменная представлена в строчке:

1) var a,b:integer;

2) procedure tr54 (var x,y:integer);
3) begin
4) a:=7; b:=-8;
5) tr54 (a,b);
6) writeln(‘a=’,a,’b=’,b);
7) end.

A) В строчке 2; *
B) В строчке 7;
C) В строчке 1;
D) В строчке 9;
E) В строчке 8.

9. function (a, b, c:integer): real;
begin
a:=sqrt(b*b-4*a*c);
end;

Представленная функция:

A) Вычисляет значение функции в трех точках a, b, c;
B) Вычисляет максимальное значение;
C) Зацикливает программу;
D) Находит значение дискриминанта и вычисляет его корень; *
E) Ничего не делает.

10. var a, b:integer;
procedure tr56 (x:integer; var y:integer);
begin
if x<0 then y:=-x+3 else y:=x*2
end;
begin
a:=-2;
tr56 (a+3, b);
writeln (‘a=’,a,’b=’,b);
end.

Результатом выполнения программы будет

A) a=-2 b=3;

B) a=-1 b=3;
C) a=-2 b=2; *
D) a=1 b=2;
E) a=-2 b=1.

11. Параметры-значения в программах применяются для

A) передачи копии параметра из подпрограммы в программу; *
B) передачи данных из программы в подпрограмму;
C) передачи данных как из программы в подпрограмму, так и из подпрограммы в программу;
D) передачи данных из подпрограммы в программу;
E) обратной передачи данных.

12. Глобальные переменные описываются

A) После заголовка подпрограммы;
B) В заголовке подпрограммы после раздела объявления переменных;
C) В вызове подпрограммы;
D) Внутри подпрограммы;
E) В главной программе. *

13. Параметры-переменные в подпрограммах применяются для передачи:

A) Данных из подпрограммы в программу;
B) Копии параметра в подпрограмму;
C) Копии параметра в главную программу;
D) Данных из программы в подпрограмму;
E) Данных как из программы в подпрограмму ,так и из подпрограммы в программу. *

14. Формальные переменные подпрограммы делятся на

A) Параметры-переменные, параметры значения и параметры-константы; *
B) Локальные параметры;
C) Глобальные и локальные параметры;
D) Фактические и локальные параметры;
E) Фактические параметры.

15. Установите результат выполнения программы

```
var x:real;
```

```
procedure tr57;
var x:real;
begin
        writeln(x);
end;
begin
        x:=5;
        tr57;
end.
```

A) x=5;
B) Ничего не выведет;
C) x=0;
D) 0;
E) 5. *

16. Из подпрограммы в программу предается копия

A) Параметра константы;
B) Параметра значения; *
C) Фактического параметра;
D) Параметра переменной;
E) Глобального параметра.

17. Возможна обратная передача данных из подпрограммы в главную программу:

A) Параметра значения;
B) Фактического параметра;
C) Параметра переменной; *
D) Локальной переменной;
E) Параметра константы.

18.
```
function max(x:integer):integer;
begin
    if x<0 then max:=-x else max:=x;
end;
```

Найти соответствие фрагмента программы и стандартной функции

A) Pred(x);
B) Sin(x);
C) Sqr(x);
D) Abs(x); *
E) Trunk(x).

19. Укажите предложение, которое не является положительной стороной использования подпрограмм:

A) использование подпрограмм позволяет записать в память компьютера входные и выходные данные; *
B) применения подпрограмм позволяет разрабатывать одну программу сообща группой программистов;
C) подпрограммы оформляются в виде замкнутых участков программы, поэтому программа становится легко читаемой, понятной, менее чувствительной к ошибкам;
D) при использовании подпрограмм экономится память;
E) писать крупные программы как единое целое без расчленения на самостоятельные фрагменты невозможно.

20. В чем различие процедур и функций?

A) в функции нет глобальных параметров;
B) функция не является подпрограммой;
C) результат работы функции возвращается в виде значения функции, вызов функции может использоваться в выражениях; *
D) отличительная особенность процедуры - только один результат выполнения, который обозначается ее именем и передается в основную программу;
E) процедурой в языке Паскаль называется особым образом оформленный фрагмент программы, имеющий собственное имя, а функция – часть программы, заданная формулой.

21. В следующем описании процедуры выходными данными являются:

procedure myproc (a, b:integer; var c, d: real; const e: string);

A) a, b;
B) c, d; *
C) e;
D) a, b, c, d, e;
E) нет таких.

22. Выберите допустимый оператор вызова процедуры

procedure myproc (a, b:integer; var c, d: real);

A) myproc;
B) myproc(a, b:integer; c, d:real);
C) c:=myproc(a, b, c, d);
D) myproc(s, f:integer; var k, l:real);
E) myproc(s, f, k, l). *

23. Как называется параметр, указываемый в скобках при вызове процедуры?

A) локальный параметр;
B) фактический параметр; *
C) глобальный параметр;
D) формальный параметр;
E) уникальный параметр.

24. В следующем описании функции переменная n – это:

function fact (n: integer): real;

A) локальный параметр;
B) фактический параметр;
C) глобальный параметр;
D) формальный параметр; *
E) уникальный параметр.

25. Для данного фрагмента программы выберите допустимый оператор вызова функции:

var m,f,k:integer;
function fact (n: integer): real;
var i:intger;

```
      p:real;
begin
     p:=1;
     for i:=1 to n do
              p:=p*I;
     fact:=p
end;
```

A) fact;
B) f:=fact(m)-fact(k); *
C) fact(k);
D) f:=fact;
E) fact(m)-fact(k).

26. Рекурсия – это

A) способ организации вычислительного процесса, при котором подпрограмма обращается сама к себе; *
B) способ организации вычислительного процесса, при котором подпрограмма вызывается в цикле;
C) способ организации вычислительного процесса, при котором подпрограмма вызывается несколько раз;
D) использование в программе подпрограмм;
E) использование в программе подпрограмм, которые вызываются в другой подпрограмме.

27. Для данного фрагмента программы укажите локальные переменные

```
var m,f,k:integer;
function fact (n: integer): real;
var i:intger;
      p:real;
begin
     p:=1;
     for i:=1 to n do
              p:=p*I;
     fact:=p
end;
```

A) m, f, k;
B) n, fact;

C) i, p; *
D) n, fact, i, p;
E) m, f, k, n, i, p.

28. Для данного фрагмента программы укажите допустимый в основной программе оператор:

```
var m,f,k:integer;
function fact (n: integer): real;
var i:intger;
     p:real;
begin
    p:=1;
    for i:=1 to n do
            p:=p*I;
    fact:=p
end;
```

A) m:=p(n);
B) m:=p(i);
C) f:=m-p;
D) f:=m; *
E) f:=fact(m)-i.

29. Какую из последовательностей символов допустимо использовать вместо * в операторе вывода?**

```
var
    x, y:real;
function power (a, b: real): real;
begin
    if a>0 then
            power:=exp(b*ln(a))
    else if a<0 then
            power:=exp(b*ln(abs(a)))
            else if b=0 then
                    power:=1
```

```
                    else
                    power:=0
end;
begin
    repeat
            readln(x, y);
            writeln(***)
    until eof
end.
```

A) power(x, y):12:10, power(x,-y):15:10; *
B) power(x, y):12:10, power(x-y):15:10;
C) power(x):12:10, power(y):15:10;
D) power(a, b:real):15:10;
E) power(a, b):12:10, power(a, x, b):15:10.

4. СЛОЖНЫЕ ТИПЫ ДАННЫХ

4.1. Регулярные типы

Массивы

Обратитесь к схеме типов Паскаля (стр. 15). Рассмотрите, на каком месте расположены массивы.

Массив – это упорядоченная по номерам совокупность значений, объединенных общим именем и типом.

Пример 73. *Например:* Список учеников класса; численные значения среднесуточной температуры за месяц; буквы русского алфавита.

Пример 74. Пример массива с именем А из пяти целых чисел.

№элемента массива А	**1**	**2**	**3**	**4**	**5**
Сам элемент массива А	-9	7	1	-2	0

массив А

Элементами массива может быть структурированный тип, *например*, массив массивов:

1 – *й элемент – массив* 5 2 3
2 – *й элемент – массив* –1 2 4
3 – *й элемент – массив* 4 2 –3
4 – *й элемент – массив* –1 1 –2

или массив из записей:

№	наименование товара	единица измерения	стоимость за единицу	кол-во	общая стоимость
1	Печенье	кг	520тг	15263	7936760
2	Пряники	кг	650тг	26589	17282850
3	Булочки	кг	15тг	25982	389730

Если мы знаем, что предстоит работать с большим объемом данных, то объявляем массив. Его элементы можно легко упорядочить, найти среди них максимальный, обеспечить доступ к любому из элементов простым указанием его порядкового номера.

При описании массива указывается сначала имя массива, затем тип array, затем в квадратных скобках диапазон нумерации элементов, а затем после зарезервированного слова of тип элементов массива.

Пример 75. Примеры описания массивов:

```
var
  a: array [1..10] of real;        {10 элементов}
```

Данная запись означает, что в памяти компьютера будет образована область памяти, названная «а» из десяти пустых ячеек, в которые мы можем программно или с помощью оператора ввода с клавиатуры записать любые вещественные числа. При этом их может быть не больше десяти. Но меньшее число может быть, т.е. можно заполнить не все ячейки, а какую-то их часть.

1	2	3	4	5	6	7	8	9	10

Нумерация элементов уже задана (в квадратных скобках) и постоянна (от 1 до 10), а значения самих элементов (то, что в пустых ячейках) могут изменяться.

```
var
  b: array [49..100] of char;     {51 элемент-символ}
  c: array [-3..4] of Boolean;    {8 элементов логического типа}
```

тип-диапазон кроме ***longint***

Массив b

49	50	51	52	53	54	55	568	…	57

В пустые ячейки можно записывать и изменять символы.

Массив c

-3	-2	-1	0	1	2	3	4

В пустые ячейки можно записывать и изменять данные логического типа.

```
var
  mass: array [byte] of single;
```

С помощью данной записи будет выделена область памяти из 256 ячеек (от 0 до 255, так же как тип byte). В пустые ячейки можно записывать и изменять данные вещественного типа.

```
var
  mass: array [‘A’..’F’] of real;
```

С помощью данной записи будет выделена область памяти из 6 ячеек. В пустые ячейки можно записывать и изменять данные вещественного типа.

A	B	C	D	E	F

Следует отметить, что при работе с массивами количество и нумерация ячеек остается неизменяемой. Поэтому заполнять можно не все ячейки, но не больше ячеек, чем задано в описании.

Задать массив можно по-другому с помощью описания типа.

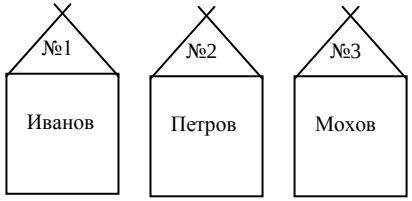
Пример 76. В этом же примере показано, как нужно обращаться к элементу массива.

```
type   {задаем свой собственный тип-массив, имя которому придумываем сами - digit, т.е. в Паскале такого типа нет}
  digit=array [1..9] of char;   {тип массив из девяти символов}
var
  m: digit;   {используем придуманный нами тип в качестве описания переменной m}
begin
  m[7]:=’j’;   {Обращаемся к элементу с указанием его порядкового номера (индекс обязателен), записываем в пустую область напротив номера 7 символ «j»}
end.
```

1	2	3	4	5	6	7	8	9
						j		

Массив можно сравнить с улицей одноэтажных домов на одного хозяина, так как:

- ✓ у массива есть имя;
- ✓ у массива есть тип;
- ✓ у массива есть размер;
- ✓ у массива есть сквозная последовательная индексация элементов;
- ✓ у каждого элемента есть значение.



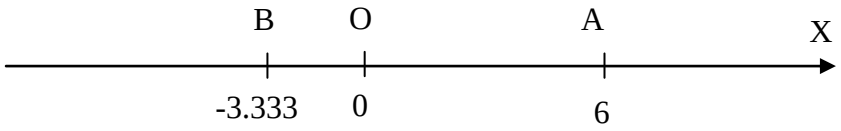
Улица Речная

Продолжая аналогию с улицей одноэтажных домов, что надо сделать, чтобы обратиться к какому-либо конкретному жильцу? Знать его адрес! Предположим, мы хотим потревожить господина Петрова — указываем его адрес — Речная[2], т. е. название улицы и дом. Зачастую начинающие программисты путают индекс элемента массива (его номер) и значение элемента массива, т. е. "кто-кто в тереме живет". Итак, еще раз: индекс или номер элемента массива — величина постоянная, а значение (как и жильцы в доме) с легкостью может меняться. Вначале мы рассмотрим одномерные массивы — такие, в которых адрес элемента массива определяется только одним индексом (номером "дома").

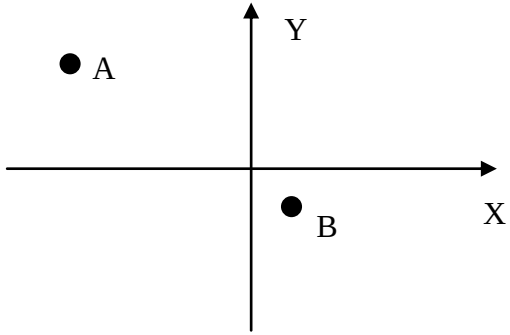
Массивы также могут быть двумерными, трехмерными и т.д. Что такое двумерный массив? Это такой набор однотипных данных, местоположение каждого элемента которого определяется не одним индексом, а двумя. Например, для тех, кто с детства играл в "морской бой", не будет открытием, что каждая клеточка игрового поля обозначается двумя символами — буквой и цифрой, например, А5 — "мимо", И10— "попал", Ж7 — "убит". Только в Паскале в качестве индексов vj;yj использовать любой тип-диапазон. Жизненный пример применения двумерных массивов — билеты в кино или театр,

имеющие для каждого зрителя две координаты — ряд и место. Описываются подобные массивы в Паскале тем же зарезервированным словом, при этом в скобках указываются две размерности массива — количество строк и количество столбцов.

Можно провести аналогию организации массивов разной размерности с математическим одномерным, двумерным, трехмерным пространством. Так одномерный массив можно сравнить с числовой прямой, на которой располагаются точки с одной координатой: A(6), B(-3.333).



Двумерный массив – с координатной плоскостью, расположение точек на которой задается двумя числами-координатами: A(-4, 3), B(2; -1.4), а трехмерный – с координатным пространством, где каждая точка имеет три координаты: A(2, 1, 0), B(5, -4, 3).



Пример 77. Например, двумерный массив 5x3 объявим так:

var a:**array** [1..5] **of array** [1..3]of integer;

Или короче:

var a:**array**[1..5,1..3] **of** integer;

Если записать каждый элемент данного массива в таблицу, получится следующее:

a[1,1]	a[1,2]	a[1,3]
a[2,1]	a[2,2]	a[2,3]
a[3,1]	a[3,2]	a[3,3]
a[4,1]	a[4,2]	a[4,3]
a[5,1]	a[5,2]	a[5,3]

Пример 78. Трехмерный массив можно объявить следующим образом:

```
type
  mat=array [0..5] of array [-2..2] of array [char] of byte;
var a: mat;
```

Или более компактно:

```
type
  mat=array [0..5, -2..2, char] of byte;
var a: mat;
```

Данная запись означает, что у каждого элемента трехмерного массива целых чисел a будет по три индекса: 1-й индекс – целое число от 0 до 5, 2-й индекс – целое число от -2 до 2, 3-й индекс – символы с кодами от 0 до 256.

Например, обратиться к элементу можно будет так:

```
a[0, -1, 'g']:=random(5)-2;
```

Суммарная длина внутреннего представления любого массива не может быть > 65520 байт. В памяти ПК элементы массива следуют друг за другом так, что при переходе от младших адресов к старшим наиболее быстро меняется самый правый индекс массива.

Пример 79.

```
var a: array [1..2, 1..2] of byte; {На первом месте строки, на втором - столбцы}
begin
  a [1, 1]:=1;
  a [2, 1]:=2;
  a [1, 1]:=3;
  a [2, 2]:=4;
```

end.

В памяти последовательно друг за другом будут расположены байты со значениями 1, 3, 2, 4.

$$a_{11} \quad a_{12}$$
$$a_{21} \quad a_{22}$$

Способы организации массива

Пример 80. Заполнение с клавиатуры:

```
var a: array [1..100] of real;
    i, n :integer;
procedure vvod (n:integer);
begin
    for i:=1 to n do
            begin
                writeln('Введите элемент массива:');
                readln(a[i]);
            end;
end;
procedure print;
begin
    for i:=1 to n do
            write(a[i]:6:2,',');
end;
begin
    writeln('Введите количество элементов массива');
    readln(n);
    vvod(n);
    print;
    readln;
end.
```

Пример 81. Заполнение случайными числами:

```
var a: array [1..200] of integer;
```

```
    n: integer;
procedure mass (n:integer);
var i: integer;
begin
    for i:=1 to n do
            begin
                {массив заполняется случайными целыми
                числами, значения которых находятся в
                диапазоне от -15 до 5}
                a[i]:=random(21)-15;
                writeln(a[i]);
            end;
end;
begin
    randomize;
    writeln('Введите количество элементов');
    readln(n);
    mass(n);
    readln;
end.
```

Пример 82. Заполнение значениями функции из какого-либо диапазона с каким-либо шагом:

```
program ddd;
uses crt;
var
  s, t, r: real;
  i, n: integer;
function f (t:real): real;
begin
  f:=2*sin(t)+3;
end;
procedure mass (a, b, h: real; var i: integer);
var
  m:array [1..100] of real;
  x:real;
begin
```

```
  x:=a;
  i:=1;
  while x<=b do
        begin
              m[i]:=f(x);
              write (m[i]:7:3);
              i:=i+1;
              x:=x+h;
        end;
  writeln;
  n:=n-1;
  writeln('Количество элементов массива равно ', n);
end;
begin
  clrscr;
  writeln('Введите интервал [a, b] и шаг h');
  readln(s, t, r);
  mass(s, t, r, i);
  readln;
end.
```

Поиск максимального элемента массива.

Одна из главных задач работы с элементами массива – это поиск минимального или максимального его элемента.

Алгоритм

- ✓ Запоминаем первый элемент в переменную max;
- ✓ Начиная с первого элемента, сравниваем каждый элемент со значением переменной max;
- ✓ Если он больше, то переменной max присваиваем значение данного элемента.

Пример 83. *Программа*

```
program m;
uses crt;
```

```
var a:array [1..100] of real;
i, n: integer;
max: real;
{процедура создания массива}
procedure sozd;
begin
  randomize;
  writeln('Введите количество элементов массива');
  readln(n);
  for i:=1 to n do
        a[i]:=random*5-2;
end;
{процедура вывода массива на экран}
procedure print;
begin
        for i:=1 to n do
              write(a[i]:8:2);
        writeln;
end;
{процедура поиска максимального элемента}
procedure maxim;
begin
        max:=a[1];
        for i:=1 to n do
              if a[i]>=max then max:=a[i];
        writeln('Максимальный элемент=', max:8:2);
end;
{основная программа}
begin
        clrscr;
        sozd;
        print;
        maxim;
        readln;
end.
```

Алгоритмы удаления элемента массива

Так как количество элементов массива и их нумерация задается при описании переменной типа массив и менять нумерацию и количество элементов нельзя, следовательно, удаления элемента массива, как такового, в Паскале не существует. Т.е. ячейка для удаленного элемента все равно в памяти останется и ее можно будет заполнить каким-либо значением. Просто данную ячейку после так называемого удаления можно будет не учитывать или присвоить ей нулевое значение.

Первый алгоритм удаления элемента из массива:

- ✓ Запрашиваем у пользователя номер элемента - k.
- ✓ Начиная с элемента с номером k, присваиваем каждому элементу значение следующего.
- ✓ Количество элементов массива уменьшаем на 1.

Пример 84. *Программа*

```
program m1;
uses crt;
var a:array [1..100] of real;
i, n, k: integer;
{процедура создания массива}
procedure sozd;
begin
    writeln('Введите количество элементов массива');
    readln(n);
    for i:=1 to n do
    begin
        writeln('Введите ',i,'-й элемент массива');
        readln(a[i]);
    end;
end;
{процедура вывода массива на экран}
procedure print;
```

```
begin
    for i:=1 to n do
            write(a[i]:8:2);
    writeln;
end;
{процедура удаления элемента массива}
procedure ydal;
begin
    writeln('Введите номер удаляемого элемента');
    readln(k);
    for i:=k to n-1 do
            a[i]:=a[i+1];
    n:=n-1;
    print;
end;
{основная программа}
begin
    clrscr;
    sozd;
    print;
    ydal;
    readln;
end.
```

Второй алгоритм удаления элемента из массива

- ✓ Запрашиваем у пользователя номер элемента – k.
- ✓ Меняем местами элемент с номером k с последним элементом.
- ✓ Количество элементов уменьшаем на единицу.

Пример 85. *Программа*

В данном случае процедура удаления будет выглядеть так:

```
procedure ydal;
begin
    writeln('Введите номер удаляемого элемента');
    readln(k);
    r:=a[k];
```

```
        a[k]:=a[n];
        a[n]:=r;
        n:=n-1;
        print;
    end;
```

В раздел описания необходимо дополнительно внести промежуточную переменную r типа real.

Замена местами элементов, стоящих на четных и нечетных местах

Алгоритм

Просматриваем в цикле каждый элемент, и, если он четный, то меняем его местами с предыдущим. Однако нужно учесть, что если в массиве нечетное число элементов, то последний элемент нужно оставить на своем месте.

Пример 86. *Программа*

```
program m2;
uses crt;
var m:array [1..100] of real;
i, n: integer;
a, b, h, r: real;
{процедура создания массива}
function f (x:real): real;
begin
    f:=sin(2*x)-1;
end;
procedure sozd;
begin
    writeln('Введите границы интервала');
    readln(a, b);
    writeln('Введите шаг');
    readln(h);
    i:=1;
```

```
    while a<=b do
    begin
            m[i]:=f(a);
            a:=a+h;
            i:=i+1;
    end;
    n:=i-1;
end;
{процедура вывода массива на экран}
procedure print;
begin
    for i:=1 to n do
            write(m[i]:8:2);
    writeln;
end;
{процедура замены четных и нечетных элементов}
procedure zamena;
begin
    for i:=1 to n-1 do
    if  i mod 2 <> 0 then
    begin
            r:=m[i];
            m[i]:=m[i+1];
            m[i+1]:=r;
    end;
    print;
end;
{основная программа}
begin
    clrscr;
    sozd;
    print;
    zamena;
    readln;
end.
```

Создание главного меню программы

Для того, чтобы не вызывать процедуру печати внутри другой процедуры, создадим главное меню программы, с помощью которого пользователь может сам выбрать нужную процедуру. При этом главная программа будет выглядеть так.

Пример 87.

```
begin
    w:=0;
    while w<>4 do
    begin
        writeln('Вы можете');
        writeln('1-создать массив');
        writeln('2-распечатать массив');
        writeln('3-обработать массив');
        writeln('4-выйти из программы');
        writeln('Ваши действия?');
        readln(w);
        case w of
                1: sozd;
                2: print;
                3: obr;
        end;
    end;
end.
```

Не забудьте описать переменную w типа integer.

Сортировки массива

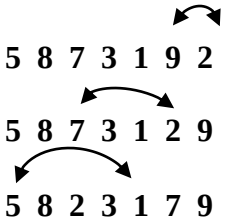
Одной из основных операций, производимых над массивами, являются операции сортировки или упорядочивания элементов массива по какому-либо признаку: чаще по возрастанию или убыванию — для чисел, и по алфавиту — для символов и строк. Сортировок придумано множество, и, говорят, тому, кто придумает новый эффективный метод сортировки,

сразу будет вручена Нобелевская премия. С древних времен, однако, до нас дошли два самых простых (но, конечно, не самых эффективных) способа сортировки, которые мы здесь и рассмотрим.

Сортировка массива методом выбора

Алгоритм

- ✓ Находим максимальный элемент и запоминаем его индекс, например в переменную с.
- ✓ Меняем местами последний элемент и элемент с номером с.
- ✓ Находим из оставшихся n-1 элементов максимальный и запоминаем его индекс в переменную с.
- ✓ Меняем его местами с предпоследним.
- ✓ Повторяем алгоритм n-1 раз.



Пример 88. *Программа*

```
program m3;
uses crt;
var a:array [1..100] of real;
i, n, k, w, c, j: integer;
{процедура создания массива}
procedure sozd;
begin
    writeln('Введите количество элементов массива');
    readln(n);
    for i:=1 to n do
    begin
        writeln('Введите ',i,'-й элемент массива');
        readln(a[i]);
```

```
        end;
    end;
    {процедура вывода массива на экран}
    procedure print;
    begin
        for i:=1 to n do
                write(a[i]:8:2);
        writeln;
    end;
    {процедура сортировки}
    procedure sort;
    begin
    for j:=n downto 1 do
        begin
                max:=a[1];
                c:=1;
                for i:=1 to j do
                        if a[i]>max then
                                begin
                                        max:=a[i];
                                        c:=i;
                                end;
                r:=a[c];
                a[c]:=a[j];
                a[j]:=r;
        end;
end;
{основная программа}
begin
    clrscr;
    w:=0;
    while w<>4 do
```

```
    begin
        writeln('Вы можете');
        writeln('1-создать массив');
        writeln('2-распечатать массив');
        writeln('3-отсортировать массив');
        writeln('4-выйти из программы');
        writeln('Ваши действия?');
        readln(w);
        case w of
                1: sozd;
                2: print;
                3: sort;
        end;
    end;
end.
```

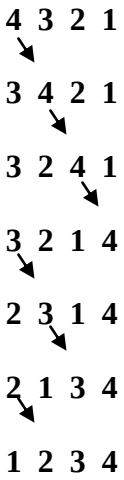
Сортировка массива методом обмена или «пузырька»

Название данного метода часто вызывает нездоровый смех у молодежи, хотя никакого тайного смысла у этого метода нет. Просто он выполняется таким образом, что максимальное число после каждого шага сортировки как бы всплывает в конец массива, на свое заслуженное место. Заключается метод в следующем. Программа, начиная с первых элементов массивов, сравнивает эти элементы попарно, и, в случае, если они расположены не по возрастанию, меняет их местами. В результате (n-1)*2 перестановок (в случае самого плохого расположения элементов массива — все элементы по убыванию) массив окажется упорядочен по возрастанию.

Алгоритм

- ✓ Начиная с первого до предпоследнего элемента, сравниваем соседние элементы.
- ✓ Если они не упорядочены, меняем их местами.
- ✓ После этих двух шагов на последнем месте окажется максимальный (минимальный) элемент. Следующим шагом просматриваем элементы, начиная с первого до пред предпоследнего и повторяем шаги. После этого предпоследний элемент окажется на своем месте.

✓ Повторяем n-1 раз.



Пример 89. *Программа*

```
program m4;
uses crt;
var a:array [1..100] of real;
i, n, j, w: integer;
r: real;
{процедура создания массива}
procedure sozd;
begin
  randomize;
  writeln('Введите количество элементов массива');
  readln(n);
  for i:=1 to n do
        a[i]:=random*5-2;
end;
{процедура вывода массива на экран}
procedure print;
```

```
begin
        for i:=1 to n do
                write(a[i]:8:2);
        writeln;
end;
procedure sort;
begin
    for j:=1 to n-1 do
        for i:=1 to n-j do
                if a[i]>a[i+1] then
                        begin
                                r:=a[i];
                                a[i]:=a[i+1];
                                a[i+1]:=r;
                        end;
end;
{основная программа}
begin
    clrscr;
    w:=0;
    while w<>4 do
    begin
        writeln('Вы можете');
        writeln('1-создать массив');
        writeln('2-распечатать массив');
        writeln('3-отсортировать массив');
        writeln('4-выйти из программы');
        writeln('Ваши действия?');
        readln(w);
        case w of
                1: sozd;
                2: print;
                3: sort;
        end;
    end;
end.
```

Двумерные массивы

Заполнение двумерного массива и вывод его на экран в виде таблицы.

Принцип заполнения двумерного массива и вывода его на экран в следующем. Работают два вложенных цикла. Во внешнем цикле просматриваются строки, с первой до последней. Во внутреннем цикле – столбцы. Вспомните организацию вложенных массивов. Сначала переменной i присваивается единица, а переменная j проходит все свои значения с первого до последнего, после чего вся первая строка (т.е. каждый ее столбец) оказывается заполненной значениями. После заполнения каждого столбца первой строки переменной i присваивается 2 и начинается заполнение всех столбцов второй строки (т.е. переменная j опять проходит все свои значения) и т.д.

$$\begin{vmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{vmatrix}$$

В процедуре печати для того, чтобы элементы массива вышли на экран в виде таблицы мы во внутреннем цикле используем оператор вывода write(a[i]), а во внешнем цикле (т.е. при переходе на новую строку) переносим курсор на новую строку с помощью пустого оператора вывода writeln.

Пример 90.

```
program m5;
  uses crt;
  var
      a: array [1..3, 1..5] of integer;
      i, j, w: integer;
  {процедура создания}
```

```
    procedure sozd;
    begin
    randomize;
    for i:=1 to 3 do
        for j:=1 to 5 do
                a[i, j]:=random(10)-2;
    end;
    {процедура печати}
    procedure print;
    begin
    for i:=1 to 3 do
        begin
                for j:=1 to 5 do
                        write (a[i,j]:6);
                writeln;
        end;
    end;
    {основная программа}
begin
    clrscr;
    w:=0;
    while w<>3 do
    begin
        writeln('Вы можете');
        writeln('1-создать массив');
        writeln('2-распечатать массив');
        writeln('3-выйти из программы');
        writeln('Ваши действия?');
        readln(w);
        case w of
                1: sozd;
                2: print;
        end;
    end;
end.
```

Нахождение количества элементов массива, равных 5 в каждой строке массива

Пример 91.

```
program m6;
uses crt;
var
        a: array [1..5, 1..5] of integer;
        i, j, w,k: integer;
    {процедура создания}
procedure sozd;
begin
    for i:=1 to 5 do
        for j:=1 to 5 do
        begin
                writeln('Введите [', i ,',', j , ']-й элемент массива');
                readln(a[i, j]);
        end;
end;
{процедура печати}
procedure print;
begin
    for i:=1 to 5 do
        begin
                for j:=1 to 5 do
                        write (a[i,j]:6);
                writeln;
        end;
end;
{процедура поиска}
procedure poisc;
begin
    for i:=1 to 5 do
        begin
```

```
                k:=0;
                for j:=1 to 5 do
                        if a[i, j]=5 then k:=k+1;
                writeln (‘В ‘,i ,’ – й строке ‘,k,’ элементов’);
        end;
end;
{основная программа}
begin
    clrscr;
    w:=0;
    while w<>4 do
    begin
        writeln(‘Вы можете’);
        writeln(‘1-создать массив’);
        writeln(‘2-распечатать массив’);
        writeln(‘3-найти кол-во пятерок в каждой строке’);
        writeln(‘4-выйти из программы’);
        writeln(‘Ваши действия?’);
        readln(w);
        case w of
                1: sozd;
                2: print;
                3:poisc;
        end;
    end;
end.
```

В двумерном массиве 5x5 поставить на первое место столбец с наибольшим количеством элементов, равных пяти.

Добавьте в программу m6 следующую процедуру. Принцип ее работы следующий. Находим количество пятерок в каждом столбце массива и во внешнем цикле сразу определяем среди них наибольшее значение и запоминаем номер столбца, в котором имеется больше всего пятерок. Затем в другом цикле меняем местами найденный столбец с первым. Обратите

внимание, что когда мы ищем что-то по строкам, то первый цикл идет по переменной i, т.е. по строкам и наоборот.

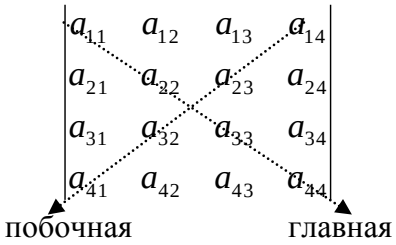
Пример 92.

```
procedure zamena;
begin
    max:=0;
    for j:=1 to 5 do
        begin
                k:=0;
                for i:=1 to 5 do
                        if a[i, j]=5 then k:=k+1;
                if k>max then
                begin
                        max:=k;
                        n:=j
                end;
                writeln (‘В ‘,j ,’ – м столбце ‘,k,’ элементов’);
        end;
    writeln (‘Столбец с максимальным числом пятерок ‘, n ,’. В
    нем ‘, max ,’ пятерок’);
    for i:=1 to 5 do
        begin
                r:=a[i, n];
                a[i, n]:=a[i, 1];
                a[i, 1]:=r;
        end;
    end;
```

Не забудьте описать переменные max, r, n типа integer, добавить в главное меню вызов процедуры замены.

Работа с диагоналями двумерного массива

В квадратном двумерном массиве (т.е. в массиве с равным количеством строк и столбцов) имеются главная и побочная диагональ.



Иногда требуется провести какие-либо действия с элементами диагоналей, элементами, расположенными ниже или выше их.

Для доступа к элементам диагоналей в двух вложенных циклах просмотра элементов массива необходимо написать условие принадлежности элемента какой-либо диагонали. Если внимательно посмотреть на индексы элементов, легко можно увидеть закономерность или общий признак, объединяющий данные элементы.

- ✓ Элементы принадлежат главной диагонали, если номер строки равен номеру столбца: i=j;
- ✓ Элементы принадлежат побочной диагонали, если сумма их индексов на единицу больше размерности массива: i+j=n+1;
- ✓ Элементы, расположенные ниже главной диагонали, имеют номера, находящиеся в отношении: i>j;
- ✓ Элементы, расположенные выше главной диагонали, имеют номера, находящиеся в отношении: i<j;
- ✓ Сумма номеров элементов, расположенных выше побочной диагонали, меньше или равна размерности массива: j+i<=n;
- ✓ Сумма номеров элементов, расположенных ниже побочной диагонали, больше или равна размерности массива: j+i>=n.

Пример 93. Приведем пример программы, в которой необходимо найти сумму элементов, расположенных выше побочной диагонали.

```
program m7;
uses crt;
```

```
var
        a: array [1..5, 1..5] of integer;
        i, j, w, s: integer;
    {процедура создания}
procedure sozd;
begin
    randomize;
    for i:=1 to 5 do
        for j:=1 to 5 do
                a[i, j]:=random(10)-5;
end;
{процедура печати}
procedure print;
begin
    for i:=1 to 5 do
        begin
                for j:=1 to 5 do
                        write (a[i,j]:6);
                writeln;
        end;
end;
{процедура нахождения суммы}
procedure sum;
begin
    s:=0;
    for i:=1 to 5 do
        for j:=1 to 5 do
                if i+j<=5 then s:=s+a[i,j];
        writeln (‘Сумма элементов выше побочной диагонали
равна ’, s);
end;
{основная программа}
```

```
begin
    clrscr;
    w:=0;
    while w<>4 do
    begin
        writeln('Вы можете');
        writeln('1-создать массив');
        writeln('2-распечатать массив');
        writeln('3-найти сумму эл. выше побочной диагонали');
        writeln('4-выйти из программы');
        writeln('Ваши действия?');
        readln(w);
        case w of
                1: sozd;
                2: print;
                3:sum
        end;
    end;
end.
```

Однако для элементов, расположенных на главной или побочной диагонали можно обойтись меньшим числом операторов и всего лишь одним циклом.

Пример 94. Например, сумму элементов главной диагонали найдем с помощью следующей процедуры.

```
procedure sum;
begin
    s:=0;
    for i:=1 to 5 do
        s:=s+a[i, i];
writeln ('Сумма элементов главной диагонали равна ', s);
end;
```

Найдите аналогично сумму элементов побочной диагонали.

Строки

Тип string – строка в Турбо Паскале используется для обработки текстов. Он во многом похож на одномерный массив символов array [0..n] **of** char, однако, в отличие от последнего, количество в строке переменной может меняться от 0 до n, где n – максимальное количество символов в строке. Значение n определяется объявлением типа string[n] и может быть любой константой порядкового типа, но не более 255. Если не указывать n, то длина строки принимается максимально возможной, а именно n=255.

К любому символу в строке можно обратиться так же, как к элементам одномерного массива.

Пример 95. Например:

```
var st: string [20];
        begin
                if st [5]='A' then …
        end;
```

Пример 96.

```
program stroki;
uses crt;
var s: string[15];
i, k:integer;
procedure vvod;
begin
        writeln ('Введите строку');
        readln (s);
        k:=0;
        for i:=1 to 15 do
                if s[i]='f' then k:=k+1;
end;
procedure vivod;
begin
        writeln (s);
```

```
        writeln (‘Количество букв f в строке =’, k);
end;
begin
        clrscr;
        vvod;
        vivod;
        readln
end.
```

Самый первый байт в строке имеет индекс 0 и содержит текущую длину строки. Первый значащий символ строки занимает 2-й байт и имеет индекс 1.

Пример 97. Например, в строке s=’компьютер’ нулевой символ будет иметь номер 9. Ищем в кодировочной таблице (приложение Б) код=9 и смотрим соответствующий ему символ. Это символ ‘○**’.** Это и есть нулевой символ строки s=’компьютер’. Выполните следующую программу.

```
uses crt;
var
        st:string[100];
        i:byte;
begin
            clrscr;
            writeln('Введите строку');
            readln(st);
            writeln(st[0]);       {выводим нулевой символ
 строки}
            readln;
end.
```

Пример 98. Удалить из строки все ведомые пробелы (т.е. пробелы, стоящие последними). Чтобы проверить правильность работы программы, напишите в условии цикла вместо (st [i]=’ ’) => (st [i]=’a’), тогда программа будет удалять все последние буквы ‘a’ в строке.

```
uses crt;
var
```

```
        st:string[100];
        i:byte;
      begin
        clrscr;
        writeln('Введите строку');
        readln(st);
        i:=ord (st [0]);          {i – текущая длина строки}
        while (i<>0) and (st [i]=' ') do
              begin
                    i:=i-1;
                    st[0]:=chr (i);          {нулевой символ
строки переписывается, т.е. уменьшается его код}
              end;
        writeln(st);
        readln;
      end.
```

Значение ord (st [0]), т.е. текущую длину строки, можно получить с помощью функции length (st) типа integer, например:

Пример 99.

```
program stroki;
uses crt;
var s: string;
i, k, n:integer;
procedure vvod;
begin
        writeln ('Введите строку');
        readln (s);
        n:=length(s);
        k:=0;
        for i:=1 to n do
              if s[i]<>' ' then k:=k+1;
end;
```

```
procedure vivod;
begin
        writeln (s);
        writeln (‘Количество символов в строке =’, n);
        writeln (‘Количество букв в строке =’, k);
end;
begin
        clrscr;
        vvod;
        vivod;
        readln
end.
```

К строкам можно применять операцию **«+»** - сцепление, например: **st :=’a’+’b’; st:=st+’c’; {st** содержит **«abc»}**

Если длина сцепленной строки превысит максимально допустимую длину n, то «лишние» символы отбрасываются.

Пример 100.

```
var st: string [2];
begin
        st:=’1’+’23’;
        writeln (st);    {на экран выйдет ‘12’}
end.
```

Процедуры и функции для работы со строками.

Здесь обратите, пожалуйста, внимание на то, процедура это или функция, так как использование их в программе различаются. Процедуры вызываются по имени с указанием фактических параметров или их значений, а функции используются в выражениях в операторе присваивания или в операторе вывода, так как имеют один-единственный результат.

- ☑ **concat (s1 [s2, …, sn])** – функция типа **string**; возвращает строку, представляющую собой сцепление строк-параметров **s1, s2, …., sn.**

Пример 101. Использование **concat**

```
a:=’abcde’;
b:=’fg’;
c:=concat(a, b)
```

{В результате работы данных операторов в переменную с запишется строка ‘abcdefg’. Переменные a, b, c имеют тип string}

- ☑ **copy (st, index, count)** – функция типа **string;** копирует из строки **st count** символов, начиная с символа с номером **index**.

Пример 102. Использование **copy**

```
a:=’abcde’;
b:=copy(a, 3, 2); {функция вызывается в операторе присваивания, типы выражений справа и слева должны быть совместимы}
```

{В результате работы данных операторов в переменную b запишется строка ‘cd’. Переменные a, b имеют тип string}

- ☑ **delete (st, index, count)** – процедура; удаляет **count** символов из строки **st count** символов, начиная с символа с номером **index.**

Пример 103. Использование **delete**

```
a:=’abcde’;
delete(a, 2, 2);        {процедура вызывается просто по имени}
```

{В результате работы данных операторов в переменной a останутся символы ‘ade’}

- ☑ **insert (subst, st, index)** – процедура; вставляет подстроку **subst** в строку **st**, начиная с символа с номером **index.**

Пример 104. Использование **insert**

```
a:=’abcde’;
b:=’fg’
insert(b, a, 2);        {процедура вызывается просто по имени}
```

{В результате работы данных операторов в переменную a добавятся символы ‘fg’, начиная со второго, т.е. a будет равна ‘afgbcde’}

```
insert(‘123’, b, 3);
```

{В результате последнего оператора переменная b будет равна ‘fg123’}

- ☑ **length (st)** – функция типа **integer**; возвращает длину строки **st.**

Пример 105. Использование **length**

```
a:=’abcde’;
n:=length(a);        {функция вызывается в операторе присваивания, типы выражений справа и слева должны быть совместимы}
```

{В результате работы данных операторов в переменную n целого типа запишется число 5}

- ☑ **pos (subst, st)** – функция типа **integer;** отыскивает в строке **st** первое вхождение подстроки и возвращает номер позиции, с которой она начинается, если подстрока не найдена, возвращает ноль.

Пример 106. Использование **pos**

```
a:=’abcdecd’;
b:=’cd’
n:=pos(b, a);        {функция вызывается в операторе присваивания, типы выражений справа и слева должны быть совместимы}
{В результате работы данных операторов в переменную n типа integer запишется число 3, так как первое вхождение подстроки b в строку a начинается с третьего символа}
n:=pos(‘ec’, a);
```

{В результате последнего оператора переменная n будет равна 5}

```
n:=pos(‘ac’, a);
```

{В результате последнего оператора переменная n будет равна 0}

- ☑ **str (x [:width [:decimals]], st)** – процедура; преобразует число **x** любого вещественного или целого типов в строку символов **st** так, как **width** и **decimals** – форматы вывода;

Пример 107. Использование **str.** Необходимо преобразовать вещественное число в строку.

```
program strok;
uses crt;
var s: string[50];
p:real;
begin
        clrscr;
        writeln('vvedi chislo');
        readln(p);
        str(p:7:3,s);
        writeln(s);
        readln
  end.
```

☑ **val (st, x, code)** – процедура; преобразует строку символов st во внутреннее представление целой или вещественной переменной x, которое определяется типом этой переменной; параметр code содержит ноль, если преобразование прошло успешно, и тогда в x помещается результат преобразования, в противном случае он содержит номер позиции в строке st, где обнаружен ошибочный символ, и в этом случае содержимое x не меняется; в строке st могут быть ведущие пробелы, однако ведомые пробелы недопустимы; например, обращение

val (‘ 123’, k, i) пройдет успешно: k получит значение 123, в i будет записан 0, в то время как обращение

val (‘123 ’, k, i) будет ошибочным: значение k не изменится, а i будет содержать 4.

Пример 108. Выполните следующую программу с различными исходными данными. Например, s=’12.36’, s=’157.hjh’, s=’gh’, s=’587’

```
program stroki;
uses crt;
var s,s1: string[50];
```

```
i :integer;
p:real;
begin
        clrscr;
        writeln ('Введите строку');
        readln (s);
        val(s, p, i);
        writeln('Полученное число= ', p);
        writeln('Код равен ', i);
        realdn
end.
```

- ☑ **upcase (ch)** – функция типа **char**; возвращает для символьного выражения **ch**, которое должно представлять собой строчную латинскую букву, соответствующую заглавную букву; если значением **ch** является любой другой символ (в том числе строчная буква русского алфавита), функция возвращает его без преобразования.

Пример 109. Исполните следующую программу с различными исходными данными.

```
program stroki;
uses crt;
var s,s1:char;
begin
        clrscr;
        writeln ('Введите символ');
        readln (s);
        s1:=upcase(s);
        writeln('Преобразованный символ  ', s1);
        realdn
end.
```

- ☑ **Операции отношения** =, <, >, <>, >=, <= выполняются над двумя строками посимвольно слева направо с учетом внутренней кодировки символов. Если одна строка меньше другой по длине, недостающие символы короткой строки заменяются значением chr (0).

‘A’>’1’

‘Turbo’<’Turbo Pascal’
‘Паскаль’>’Turbo Pascal’

Пример 110. Исполните следующую программу с различными исходными данными. Проверяйте работу программы, используя таблицу кодов (приложение Б).

```
program stroki;
uses crt;
var s,s1:string[15];
k: boolean;
begin
        clrscr;
        writeln ('Введите первую строку');
        readln (s);
        writeln ('Введите вторую строку');
        readln (s1);
        k:=s<s1;
        if k=true then writeln(‘Первая строка меньше’) else
        writeln(‘Первая строка больше’);
        k:=s=s1;
        if k=true then writeln(‘Строки равны’) else writeln(‘Строки
        не равны’);
        realdn
end.
```

Пример 111. Работа со строками.

```
var s,x,y,z:string;
begin
   x:='turbo';
   y:='pascal';
   z:=x+' '+y; { z='turbo pascal' }
   s:=''; { пустая строка }
   for c:='a' to 'z' do s:=s+c; { s='abcd..xyz' }
```

```
  writeln(s);
end.
```

Подумайте, как можно «перевернуть» строку так, чтобы она читалась справа налево.

4.2. Множественные типы

Множества.

Множества – это наборы однотипных логически связанных друг с другом объектов.

Количество элементов, входящих в множество, может меняться в пределах от 0 до 256 (множество, не содержащее элементов, называется пустым). Именно непостоянством количества своих элементов множества отличаются от массивов и записей. В множестве нет повторяющихся элементов.

Описание типа множества имеет вид:

<имя типа> = **SET OF** <баз.тип>

Здесь <имя типа> - правильный идентификатор;

SET, OF - зарезервированные слова (множество, из);

<баз.тип> - базовый тип элементов множества, в качестве которого может

использоваться любой порядковый тип, кроме WORD, INTEGER, LONGINT.

Множество в Турбо Паскале описывается, например, следующим образом (см. примеры 110-111)

Пример 112. С помощью созданного типа:

```
type
  tSet = set of <перечисление>;      {любой порядковый тип, кроме word, integer, longint}
var      s1, s2: tSet;
```

Пример 113. С помощью описания переменной:

```
var
  setD: set of '0'..'9';
  setS: set of 'A'..'Z';
  setA: set of char;
  setN: set of byte;
```

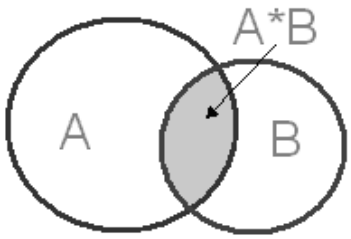
```
    setN2: set of 1..100;
  begin
    setD:=[‘0’..’9’];        {создали множество из 10-и цифр}
    setEmpty:=[];            {пустое множество}
```

Квадратные скобки с перечислением называются *конструктором множества*.

Над множествами определены следующие операции:

☑ **«*» - произведение или пересечение**.

Произведение двух множеств – это все их одинаковые элементы. Схематично произведение множеств можно изобразить так:



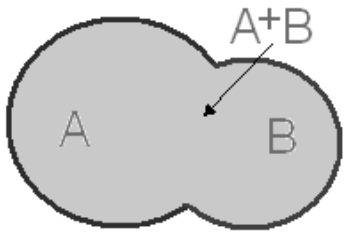
Пример 114.

```
a:=[1..5, 7, 9];
b:=[5, 8, 6, 9];
```

Произведение a*b будет равно [5, 9]

☑ **«+» - сложение или объединение.**

Сложение двух множеств – это все элементы первого и второго множества вместе взятые. Схематично объединение множеств можно изобразить так:



Пример 115.

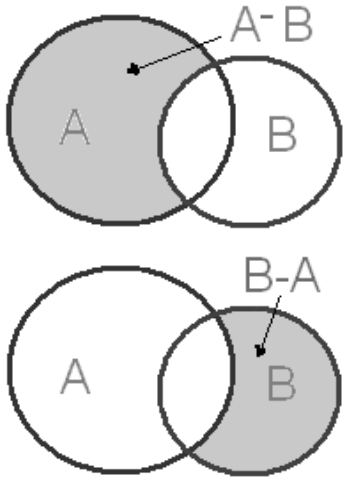
a:=[1..5, 7, 9];

b:=[5, 8, 6, 9];

Сложение a+b будет равно[1..5, 6, 7, 8, 9]

☑ **«-» - вычитание**

Вычесть из множества A множество B значит удалить из множества A все элементы, которые есть в множестве B. Смотрите схемы ниже.



Пример 116.

a:=[1..5, 7, 9];

b:=[5, 8, 6, 9];

Вычитание a-b будет равно [1, 2, 3, 4, 7]

Вычитание в-а будет равно [6, 8]

☑ **«=» - равенство множеств (эквивалентность).**

Два множества эквивалентны тогда и только тогда когда все их элементы одинаковы, причем порядок следования элементов множества безразличен.

Пример 117.

a:=[1..5, 7, 9];

b:=[5, 8, 6, 9];

a=b – выражение логического типа, будет иметь значение false

☑ **«<>» - неравенство множеств.**

Неравенство множеств противоположно операции эквивалентности.

Пример 118.

a:=[1..5, 7, 9];

b:=[5, 8, 6, 9];

a<>b – выражение логического типа, будет иметь значение true

☑ **«<=» - проверка вхождения первого множества во второе.**

Пример 119.

a:=[1..5, 7, 9];

c:=[1, 2, 3];

a<=c – выражение логического типа, имеющее значение false

☑ **«>=» - если второе множество входит в первое.**

Пример 120.

a:=[1..5, 7, 9];

c:=[1, 2, 3];

a>=c – выражение логического типа, имеющее значение true.

- ☑ **операция принадлежности элемента к множеству in,** возвращает значение true, если элемент принадлежит множеству.

  Запись в Паскале: <элемент> in <множество>

Пример 121.

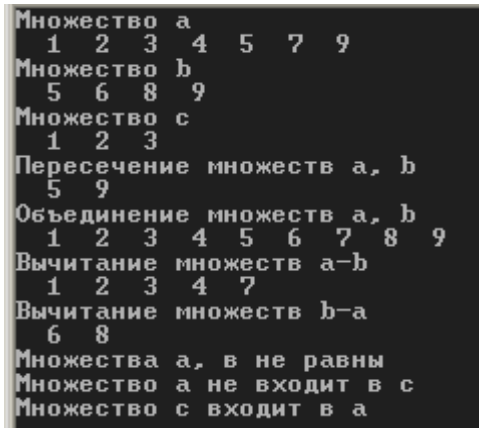
```
c:=[1, 2, 3];
3 in c        логическое выражение, возвращает true
2*2 in c      логическое выражение, возвращает false
```

Пример 122.

```
program mn;
uses crt;
var a:set of 1..10;
    b:set of 1..10;
    c:set of 1..10;
    x:boolean;
    i:integer;
begin
    clrscr;
    a:=[1..5,7,9];
    b:=[5,8,6,9];
    c:=[1,2,3];
    writeln('Множество a');
    for i:=1 to 10 do
      if i in a then write(i:3);
    writeln;
    writeln('Множество b');
    for i:=1 to 10 do
      if i in b then write(i:3);
    writeln;
    writeln('Множество c');
    for i:=1 to 10 do
      if i in c then write(i:3);
```

```
    writeln;
    writeln('Пересечение множеств a, b');
    for i:=1 to 10 do
      if i in a*b then write(i:3);
    writeln;
    writeln('Объединение множеств a, b');
    for i:=1 to 10 do
      if i in a+b then write(i:3);
    writeln;
    writeln('Вычитание множеств a-b');
    for i:=1 to 10 do
      if i in a-b then write(i:3);
    writeln;
    writeln('Вычитание множеств b-a');
    for i:=1 to 10 do
      if i in b-a then write(i:3);
    writeln;
    x:=a=b;
    if x=true then writeln('Множества a, в равны')
    else writeln ('Множества a, в не равны');
    x:=a<=c;
    if x=true then writeln('Множество a входит в c')
    else writeln('Множество a не входит в c');
    x:=c<=a;
    if x=true then writeln('Множество c входит в a')
    else writeln('Множество c не входит в a');
    readln
end.
```

Результат работы программы:



Над множествами определены две процедуры:

include (s, i) – включает новый элемент **i TSetBase** (т.е. такого же типа, что и элементы множества) в множество **S**, состоящее из элементов базового типа **TSetBase**.

exclude (s, i) – исключает элемент **i** из множества **S**, состоящее из элементов базового типа **TSetBase**.

Пример 123. Выделение из первой сотни натуральных чисел всех простых чисел. Применяем алгоритм Решето Эратосфена.

Сначала формируется множество beginset, состоящее из всех целых чисел в диапазоне от 2 до n. В множество primerset (оно будет содержать искомые простые числа) помещается единица. Затем циклически повторяются следующие действия:

1. взять из beginset первое входящее в него число next и поместить его в primerset;
2. удалить из beginset число next и все другие числа, кратные ему, т.е. 2*next, 3*next и т.д.

```
program setprim;
uses crt;
const
  n=255;
type
  setofnumber = set of 1..n;
var
```

```
    n1, next, i: word;      {вспомогательная переменная}
    beginset, primerset: setofnumber;
begin
    clrscr;
    beginset:=[2..n];       {создаем исходное множество}
    primerset:=[1];         {первое простое число}
    next:=2;
    while beginset<>[] do
        begin
            n1:=next;       {n1 – число, кратное очередному простому (next)}
            while n1<=n do
                begin
                    exclude (beginset, n1);
                    n1:=n1+next
                end;
            include (primerset, next);
            repeat
                inc (next)
            until (next in beginset) or (next>n)
        end;
    for i:=1 to n do
        if i in primerset then write (i:8);
    writeln;
    readln
end.
```

Пример 124. Сформировать массив из десяти неповторяющихся символов. Очередной символ вводится с клавиатуры.

```
uses crt;
var
    setsymbol: set of char;
```

```
        a: array [1..10] of char;
        i: integer;
        c: char;
begin
        clrscr;
        setsymbol:=[];
        i:=1;
            while i<=10 do
            begin
                read(c);
                if not (c in setsymbol) then
                begin
                        a[i]:=c;
                        setsymbol:=setsymbol+[c];
                        i:=i+1;
                End;
            End;
        for i:=1 to 10 do
                write(a[i]:3);
        writeln;
        readln;
        readln
end.
```

4.3. . Комбинированные типы

Записи

Запись – структура данных, состоящая из фиксированного числа компонентов, называемых полями записи. В отличие от массива, компоненты записи (поля) могут быть различного типа. Чтобы можно было ссылаться на тот или иной компонент записи, поля именуются. (*Например*, ведомости, документы, каталоги, списки). Записи используются когда появляется необходимость объединять данные различного типа в одну группу.

Пример 125. Пример записи

№ п/п	ФИО	Оценки

1	Иванов	5 5 5 4
целое	строковое	массив

В данной таблице запись состоит из трех полей целого, строкового типа и типа массив.

Запомните, что поля – это названия столбцов таблицы, а записи – это строки. Например в следующей таблице

№ п/п	**ФИО**	**Оценки**
1	Иванов	5 5 5 4
2	Петров	3 5 4 3
3	Сидоров	3 3 4 3
4	Васильев	5 4 4 4
целое	строковое	массив

четыре записи и три поля.

Описание записи

1-й способ с помощью описания типа:

type
<имя_типа_запись>=**record**
<поле1>:<тип_поля>;
<поле2>:<тип_поля>;
……
<полеn>:<тип_поля>
end;
{придуманный нами тип обязательно должен быть использован в разделе переменных}
var <переменная>:< имя_типа_запись >

2-й способ с помощью описания переменных:

var
<переменная>: **record**
<поле1>:<тип_поля>;
<поле2>:<тип_поля>;
……

<полеn>:<тип_поля>
end;

Пример 126. Описание записи с использованием типа.

```
type
        vedom=record
                n: integer;
                FIO: string[20];
                ocenka: array [1..3] of integer
end;
var b: vedom;
```

Чтобы обратиться к полю записи используется запись:
<название_записи>.<название_поля>

Пример 127. Описание записи с помощью переменных. Заполнение полей записи значениями.

```
var b: record
                n: integer;
                FIO: string[20];
                ocenka: array [1..3] of integer
        end;
        s: integer;
begin
        {ввести значение поля записи};
        readln(b.n);
        {или с помощью оператора присваивания}
        b.n:=2;
        b.FIO:='Иванов';
        {найти сумму 3-х оценок}
        s:=b.ocenka[1] + b.ocenka[2] + b.ocenka[3];
        writeln('№':3, 'ФИО  ':15,'Ср. балл':10);
        {вывод полей записи на экран}
        writeln(b.n:3, b.FIO:15, s/3:10:2);
        readln
        end;
```

Обращение к записи в целом, а не только к ее элементам, допускается лишь в операторе присваивания. При этом

переменные, стоящие слева и справа от оператора присваивания должны быть одного типа.

Пример 128.

```
type
        BirthDay=record
                day, month: byte;
                year: word
        end;
var
        a,b: BirthDay;
begin
        {можно использовать оператор};
        a:=b;
        {доступ к компонентам};
        a.day:=27;
        b.year:=1936;
        …….
end.
```

Чтобы упростить доступ к полям записи, используется оператор присоединения with:

```
with <имя_записи> do
        begin
                <операторы, содержащие имена полей записи>
        end;
```

Пример 129.

```
var b: record
                n: integer;
                FIO: string[20];
                ocenka: array [1..3] of integer
        end;
        s, i: integer;
begin
```

```
with b do
        begin
                writeln('Введите номер учащегося');
                readln(n);
                writeln('Введите ФИО учащегося');
                readln(fio);
                writeln('Введите три оценки');
                for i:=1 to 3 do
                        readln(ocenka[i]);
                s:=ocenka[1] + ocenka[2] + ocenka[3];
                writeln('№':3, 'ФИО ':15,'Ср. балл':10);
                {вывод полей записи на экран}
                writeln(n:3, FIO:15, s/3:10:2);
        end;
        readln
end.
```

Пример 130. Дан многочлен 4A+7B-3A+8A-2K+P-5R-2B. Найти подобные члены для переменной A и вычислить суммарный коэффициент. Один элемент многочлена можно считать записью, так как он состоит из данных различного типа – коэффициента-числа и буквы.

```
program ex4;
uses crt;
type
        elem=record
                coef: integer;
                bukva: char
        end;
var
        m1, m2: elem;
        sum: integer;
begin
        clrscr;
        sum:=0;
        writeln('Задайте многочлен');
        while not eoln do
```

```
            begin
                read(m1.coef, m1.bukva);
                if m1.bukva=’A’ then sum:=sum+m1.coef
            end;
        m2.coef:=sum;
        m2.bukva:=’a’;
        writeln(‘Суммарный коэффициент при a =’, m2.coef:3,
m2.bukva);
        readln
end.
```

Обратите внимание на то, что после заполнения всех полей создастся только одна запись. Для того, чтобы сформировать список из нескольких записей, количество которых неизвестно заранее, необходимо дополнительно создать массив такого же типа как запись, запросить у пользователя количество элементов массива и в цикле for заполнить значения каждого поля и присвоить элементу массива всю созданную запись. Или же в цикле заполнять каждую запись, сразу в этом же цикле производить над данной записью необходимые действия. Однако в этом случае нельзя будет вывести на экран весь список полностью, так как старые значения полей сотрутся и останется только последнее значение.

Пример 131. Создать список учащихся группы с полями: номер, ФИО, оценки по трем предметам. Вывести список на экран, а также средний балл для каждого учащегося. Найти учащегося с наивысшим средним баллом и его средний балл. Найти количество учащихся, фамилии которых начинаются на «А».

```
program spisok_gr;
uses crt;
type
    spisok=record
```

```
        nom:integer;
        fio:string[20];
        ocenki:array [1..3] of integer;
      end;
var
    a:spisok;
    i,k,n,s,j:integer;
    c:string[20];
    mass:array[1..20] of real;
    max:real;
    sp:array [1..20] of spisok;

procedure sozd;
begin
      writeln('Сколько учащихся в группе');
      readln(n);
      writeln('Введите список');
      max:=mass[1];
      k:=0;
      for i:=1 to n do
       begin
            writeln('Введите номер');
            readln(a.nom);
            writeln('Введите ФИО');
            readln(a.fio);
            if a.fio[1]='A' then k:=k+1;
            s:=0;
            writeln('Введите оценки');
            for j:=1 to 3 do
                begin
                    read(a.ocenki[j]);
                    s:=s+a.ocenki[j];
                end;
                readln;
                mass[i]:=s/3;
                if mass[i]>max then
                 begin
```

```
                    max:=mass[i];
                    c:=a.fio;
                end;
              sp[i]:=a;
        end;
end;

procedure vivod;
begin
      writeln('Номер','ФИО':15,'                    оценки':10, ‘ср.
      балл’);
      writeln('_____
      ___');
      for i:=1 to n do
        writeln(sp[i].nom:5,' |',sp[i].fio:15,' |',sp[i].ocenki[1]:5,
        sp[i].ocenki[2]:2,sp[i].ocenki[3]:2,mass[i]:8);
        writeln('Количество студентов с фамилиями на "А" равно
        ',k);
        writeln('Студент с наивысшим средним баллом ',c,', его
        средний балл ',max:5:2);
end;

begin
      clrscr;
      sozd;
      vivod;
      readln;
end.
```

ПРАКТИЧЕСКИЕ ЗАДАНИЯ

1. * Создать одномерный массив. Заполнить его с клавиатуры. Найти сумму элементов массива. Вывести элементы, кратные трем.
2. ** Создать 2 массива, заполненных случайными числами с равным количеством элементов. Каждый элемент третьего массива равен сумме элементов первого и второго массивов соответственно. Вывести на экран все три массива так, чтобы соответственные элементы располагались друг под другом и подпишите: 1-й массив, 2-й массив, 3-й массив.
3. ** Элементами одномерного массива являются n случайных целых чисел, значения которых заданы в диапазоне от -20 до 20 (n задает пользователь). Вывести массив на экран. Найти сумму положительных и количество нечетных элементов. Вывести номера отрицательных элементов.
4. *** Задан одномерный массив, элементами которого являются символы, вводимые с клавиатуры. Количество элементов задается пользователем. Вывести на экран массив в прямом и обратном порядке в два столбика. Выписать элементы, стоящие на местах, кратных трем в прямом массиве.

   Указания: Выводить элементы в двух напрвлениях следует в цикле в одном операторе вывода: wrtiteln(a[i]:6,a[…]:6). Вместо многоточия напишите выражение (зависящее от i), которое бы вычисляло номер элемента массива так, чтобы он нумеровал элементы в обратном порядке. Для того, чтобы выписать элементы, стоящие на местах, кратных трем, нужно в новом цикле задать условие: if i mod 3 =0 then…, так как номер элемента – это переменная i.
5. *** Написать программу, которая выводит на экран первые n чисел Фибоначчи (n задает пользователь). Эти числа определяются так: A1=1, A2=1, а каждое следующее число равно сумме двух предыдущих.

   Указания: Первые два элемента массива задавайте без цикла. В цикле нужно использовать оператор присваивания: a[i]:=a[i-1]+a[i-2]. Цикл начните не с единицы, а с тройки.

6. ** Элементы одномерного массива и их количество вводятся пользователем. Вывести массив на экран. Найти количество целых и сумму положительных элементов. Вывести номера элементов, целая часть которых равна нулю.
7. *** Вывести на экран список учащихся группы в два столбика в прямом и обратном порядке, организовав одномерный массив, количество элементов которого задает пользователь. Выписать фамилии учащихся, соящих на четных местах в прямом массиве.
   Смотрите указания к задаче 4.
8. ** Элементами одномерного массива являются значения функции y=3x2-5 в заданном диапазоне с заданным шагом. Найти произведение целых и количество элементов, целая часть которых равна нулю. Вывести номера отрицательных элементов.
9. *** Задан одномерный массив из n случайных целых чисел, значения которых заданы в диапазоне от -5 до 20. Количество элементов задает пользователь. Выписать элементы массива в два столбика в прямом и обратном порядке. Вывести на экран элементы, стоящие на нечетных местах в прямом массиве.
   Смотрите указания к задаче 4.
10. *** Элементами одномерного массива являются n случайных целых чисел, значения которых заданы в диапазоне от -40 до 50 (n задает пользователь). Вывести массив на экран. Найти сумму положительных и количество нечетных элементов. Вывести номера отрицательных элементов. Предусмотреть ошибку ввода пользователя. Например, пользователь может ввести по ошибке отрицательное количество элементов. Выведите массив в два столбца: в первом столбце – элементы

с нечетные номерами (1-й, 3-й, 5-й и т.д.), во втором – с четными номерами. Выведите массив в три столбика.
Смотрите указания к задаче 4. Смотрите указания к задаче 109 раздела 2.

11. *** Элементы одномерного массива и их количество вводятся пользователем. Вывести массив на экран. Найти количество целых и сумму положительных элементов. Вывести номера элементов, целая часть которых равна нулю. Предусмотреть ошибку ввода пользователя. Например, пользователь может ввести по ошибке отрицательное количество элементов. Выведите массив в два столбца: в первом столбце – элементы с нечетные номерами (1-й, 3-й, 5-й и т.д.), во втором – с четными номерами.
Смотрите указания к задаче 4. Смотрите указания к задаче 109 раздела 2.
12. *** Элементами одномерного массива являются значения функции y=3x2-5 в заданном диапазоне с заданным шагом. Найти произведение целых и количество элементов, целая часть которых равна нулю. Вывести номера отрицательных элементов. Предусмотреть ошибку ввода пользователя. Например, пользователь может ввести по ошибке неправильный диапазон. Выведите массив в три столбика.
Смотрите указания к задаче 4. Смотрите указания к задаче 109 раздела 2
13. * Написать программу, включающую главное меню и следующие процедуры:
    a) Создание массива;
    b) Печать массива;
    c) Поиск минимального элемента;
    d) Удаление элемента из массива 2-мя способами;
    e) Поменять местами четные и нечетные элементы массива.
14. *** Создать и распечатать массив дробных чисел, вводимых с клавиатуры. Создать главное меню и следующие процедуры:
    a) Найти сумму целых элементов;
    b) Найти максимальный из всех отрицательных элементов;

c) Выписать элементы, меньшие среднего арифметического всех элементов;
d) Поменять местами наибольший и наименьший элемент массива;
e) Удалить минимальный элемент массива;
f) Выделить красным цветом максимальный элемент массива, а зеленым – минимальный элемент.

Указания: Условие для нахождения целых элементов можно задать так: if int(a[i])=a[i] then …

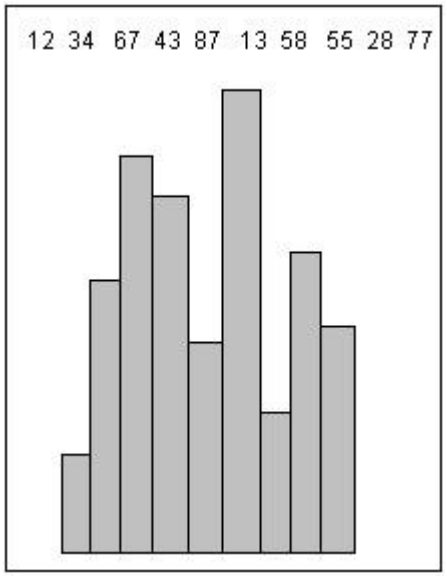
Смотрите примеры 83, 84, 85, 86, 87.

15. ** Создать и распечатать массив случайных чисел в диапазоне от -15 до 20. Создать главное меню и следующие процедуры:
a) Выписать номера элементов, больших среднего арифметического всех элементов;
b) Поменять местами первый и последний элемент массива;
c) Выписать элементы, стоящие на местах, кратных трем;
d) Найти количество элементов, являющихся двузначными числами;
e) Удалить максимальный элемент массива;
f) Найти сумму элементов, кратных 5;
g) Выписать номера отрицательных элементов.
16. * Разработать программу, в которой предусматривается создание массива, печать массива, поиск минимального элемента, удаление элемента из массива двумя способами, сортировка массива методом пузырька и методом выбора. Разработать главное меню программы.
17. ** Напишите программу вычисления выражения:

$$\frac{(x_1 - S)^2 + (x_2 - S)^2 + \ldots + (x_N - S)^2}{N}$$

где x, — элементы массива случайных чисел; N— их количество; S — среднее арифметическое элементов массива.

18. ** Найти сумму 1-го, 4-го, 9-го, 16-го и 81-го элементов массива, состоящего из 100 целых случайных чисел, каждое из которых лежит в пределах от 2 до 22.
19. * Замените в массиве из 10 случайных целых чисел, каждое из которых лежит в пределах от 1 до 10, все четные элементы нулями и выведите полученный массив на экран.
20. ** Массив состоит из 60 случайных двузначных целых чисел. Выведите их на экран в обратном порядке по 6 чисел в строке.
21. ** В массиве содержатся 10 букв — С, Ф, О, И, К, Л, О, И, Л, Н. Выведите на экран слово, образованное буквами с четными индексами, и слово, образованное буквами с нечетными индексами.
22. ** Массив состоит из 20 целых положительных и отрицательных чисел, модуль каждого из которых в пределах от 2 до 12. Выведите на экран сначала отрицательные, а затем положительные числа. Определите, модуль суммы каких чисел больше — положительных или отрицательных.
23. *** Найдите максимальный и минимальный элементы массива из 10 случайных целых двузначных чисел и разность между ними. Представьте графическую столбиковую интерпретацию этого массива (в виде столбиков из звездочек или прямых линий), выделив максимальный элемент красным, а минимальный — зеленым цветом. Остальные прямоугольники должны быть желтого цвета.
    Указания: Прямоугольники могут представлять из себя просто линии из символов «|» или «*».
    Смотрите приложение функции и процедуры модуля CRT.



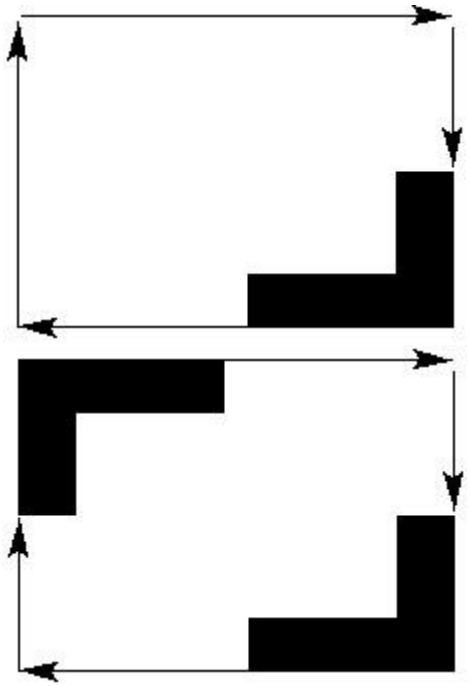
24. ** Найдите соотношение sx/sy, где sx и sy — средние арифметические значения массивов X и Y. соответственно. (Массивы из 10 элементов содержат случайные двузначные целые числа.)
25. ** Определите объем каждого из 10 цилиндров, для которых заданы радиусы оснований Rj (случайные целые числа от 5 до 25 см) и высоты Hj (случайные целые числа от 10 до 30 см).
26. *** Заданы 10 пар координат Xj, Yj одних точек на плоскости и 10 пар координат Aj, Bj других точек на плоскости. Вычислите попарно расстояния между точками по формуле: $S_i = \sqrt{(X_i - A_i)^2 + (Y_i - B_i)^2}$. Занесите эти расстояния в массив S. Проиллюстрируйте задачу графически (соответствующими отрезками на экране). Выделите разными цветами наибольшую и наименьшую длину.
    Смотрите приложение функции и процедуры модуля CRT

27. * Дан одномерный массив W из 10 случайных целых чисел, каждое из которых лежит в пределах от 1 до 100. Получите новый массив R, где каждый элемент создается из массива W делением соответствующего элемента на его индекс.
28. ** В массиве содержатся данные о среднесуточной температуре в течение февраля 2000 г. по Санкт-Петербургу. Вычислите среднюю температуру февраля, наибольшую и наименьшую температуры.
29. **** Школьники неохотно носят одежду, отличающуюся по цвету от одежды одноклассников. Напишите программу, выбирающую 2 цвета (для мальчиков и для девочек) из 12 возможных цветов, которые сегодня будут носить все. Выбор производится случайным образом и сопровождается выводом на экран прямоугольников соответствующих цветов, внутри одного из которых написано "Сегодня этот цвет для мальчиков", а внутри другого "Сегодня этот цвет для девочек".
30. **** В фирме "Green Beavis Ltd" работают семь сборщиков компьютеров. Для того чтобы повысить производительность их труда, в конце недели обрабатывают сведения о количестве компьютеров, собранных каждым из них ежедневно. Напишите программу, которая выдаст на экран следующие данные: наибольшее количество компьютеров, собранных одним служащим за неделю; среднее количество собранных за день компьютеров; лучший результат за один день; номер служащего, показавшего этот результат и день, в который он был достигнут.
31. * Дан массив X, состоящий из 100 целых случайных чисел, каждое из которых лежит в пределах от 3 до 13. С клавиатуры вводится целое число N, также лежащее в этих пределах. Определите количество элементов массива, равных числу N.
32. * Даны два числовых массива X и Y с количеством элементов 10 и 20, соответственно. Получите массив Z из 30 элементов, составленный добавлением массива X в коней массива Y.

33. ** Дан массив из 20 случайных целых чисел, каждое из которых лежит в пределах от 10 до 50. Найдите максимальное число и его номер, а если таких чисел несколько, то подсчитайте, сколько их.
34. *** Дан массив из 20 случайных целых чисел, каждое из которых лежит в пределах от 10 до 50. Определите среднее арифметическое элементов массива, а также значение элемента, ближайшего к среднему, и его номер.
    Указания: После нахождения среднего арифметического необходимо просмотреть снова каждый элемент и найти модуль разности этого элемента и среднего арифметического, а затем можно в этом же цикле найти самую меньшую из этих модулей и сразу же запомнить номер данного элемента.
35. ** Дан массив среднемесячных температур за год. Определите, в каком месяце была самая высокая температура, а в каком самая низкая, а также среднесезонные температуры.
36. ** Имеется копилка, в которой хранится 100 советских монеток достоинством в 1, 2, 3, 5, 10, 15, 20 и 50 копеек. Задайте массив M(100) случайным образом из этого набора, а затем подсчитайте, сколько в копилке было пятачков и полтинников, и какова общая сумма накопленного.
37. *** Дан массив из десяти целых двузначных случайных чисел. Найдите сумму трех максимальных из них.
    Указания: Примените алгоритм сортировки и выдайте только первые три элемента отсортированного массива.
38. *** Заполните массив R(25) случайными целыми двузначными числами так, чтобы числа не повторялись.
    Смотрите пример 124 или другой вариант решения данной задачи:

Указания: Используйте вложенные циклы, в одном из которых проверяйте каждый элемент от начала до конца, не содержит ли он числа, равного данному (тому, которое добавляется в массив.

39. **** Напишите программу "Сторож", которая бы заставила змейку оббегать стороны экрана по часовой стрелке (это можно написать и без массива). Усложните программу, взяв на службу еще одну "змейку". Теперь они ползают друг за другом. Еще сложнее — программа для движения змейки, управляемой стрелками или буквенными клавишами и совершающей повороты под прямым углом. А если и это все по плечу, то заставьте змейку поворачивать еще и под углом 45°.



40. **. Дан массив из 13 чисел. Расположите числа по возрастанию. Введите с клавиатуры число М так, чтобы оно вошло в массив и получившийся массив также был бы упорядочен по возрастанию.

41. * Дан массив букв, составляющих английский алфавит, но размешенных не по порядку. Напишите программу, преобразующую этот массив в алфавит английского языка.
42. * Дан массив из 13 четырехбуквенных русских слов (существительных и нарицательных), в единственном числе, в именительном падеже. Упорядочить их по алфавиту.
43. ** Элементами двумерного массива 5x7 являются случайные целые числа, значения которых расположены в диапазоне от -10 до 10. Распечатать массив в виде двумерной матрицы. Найти максимальный и минимальный элементы и выделить красным цветом максимальный, а зеленым – минимальный элементы.
44. *** В двумерном массиве 4x5, элементами которого являются действительные числа, введенные с клавиатуры, найти сумму элементов каждой строки и поменять местами первую и последнюю строчки.
    Смотрите примеры 91-92.
45. *** Элементами двумерного массива 3x5 являются символы, вводимые с клавиатуры. Распечатать массив в виде двумерной матрицы. Поменять местами первый и последний столбцы. Найти количество букв «а» в каждой строчке массива.
    Смотрите примеры 91-92.
46. *** Создать массив, элементы которого вводятся с клавиатуры. Подписывать при вводе данных: «Введите элемент a [1,1], a[1,2], a[1,3] …» и т.д.
    a) Закрасить красным цветом элементы, расположенные ниже побочной диагонали.
    b) Найти минимальный элемент в каждом столбце и поставить на первое место столбец с самым большим минимальным элементом.

c) Вывести номера строк, в которых больше двух четных элементов.
d) Вывести номер строки массива с максимальной суммой положительных элементов.
e) Вывести индексы четных элементов, расположенных в побочной диагонали.
f) Поменять местами строки, номера которых вводит пользователь. Найти количество отрицательных элементов в каждом столбце массива. Удалить столбец массива по запросу пользователя.

Смотрите примеры 91-92. Смотрите приложение функции модуля CRT.

47. *** Создание массива, элементы которого вводятся с клавиатуры. Подписывать при вводе данных: «Введите элемент a [1,1], a[1,2], a[1,3] …» и т.д.
a) Закрасить синим цветом элементы, расположенные выше главной диагонали
b) Найти среднее арифметическое элементов каждой строки и поставить на последнее место столбец с самым маленьким средним арифметическим.
c) Вывести номера строк массива, в которых нет элементов, кратных трем.
d) Вывести номер столбца массива с минимальным количеством нечетных элементов.
e) Вывести индексы положительных элементов, расположенных в главной диагонали.
f) Поменять местами столбцы, номера которых вводит пользователь.
g) Найти количество элементов, кратных 7 в каждой строке массива.
h) Удалить строку массива по запросу пользователя.

Смотрите примеры 91-92. Смотрите приложение функции модуля CRT.

48. ** Создать двумерный массив;
a) Вывести его на печать в виде таблицы;
b) Найти максимальный элемент массива;

c) Найти количество четных элементов;
d) Выписать элементы, расположенные в главной диагонали;
e) Найти количество положительных элементов в побочной диагонали;
f) Найти сумму четных элементов, расположенных ниже побочной диагонали;
g) Найти среднее арифметическое элементов, расположенных выше главной диагонали.

49. *** Написать программу, включающую следующие процедуры:
a) Создать матрицу A(5x5) случайных чисел в диапазоне от –30 до +30.
b) Напечатать матрицу.
c) Обнулить главную диагональ матрицы, если в ней найдется хотя бы один отрицательный элемент
d) Процедура печати должна позволять просматривать как исходную, так и получившуюся матрицу.
Указания: Перед циклом присвойте логической переменной l значение false. Просмотрите в цикле каждый элемент главной диагонали и если этот элемент отрицательный, присвойте логической переменной l значение TRUE. После всего просмотра (после цикла) напишите условие: if l=true then …После слова then нужно обнулить главную диагональ.

50. *** Написать программу, включающую следующие процедуры:
a) Создать матрицу B(5x5) случайных чисел в диапазоне от –20 до +20.
b) Напечатать матрицу.
c) Найти строку с наибольшей суммой элементов (Напечатать найденную сумму и номер строки)
Смотрите примеры 91-92.

51. *** Написать программу, включающую следующие процедуры:
    a) Создать матрицу C(8x8) случайных чисел в диапазоне от –50 до +50.
    b) Напечатать матрицу.
    c) Поменять местами элементы главной и побочной диагоналей.
    d) Процедура печати должна позволять просматривать как исходную, так и получившуюся матрицу.

    Указания: Менять местами элементы главной и побочной диагоналей можно, используя один цикл и выразив в формуле через параметр цикла элемент главной и побочной диагонали.

    Смотрите указания к задаче 4.
52. ** Написать программу, включающую следующие процедуры:
    a) Создать матрицу A(6x6) случайных чисел в диапазоне от –25 до +25.
    b) Напечатать матрицу.
    c) Найти количество отрицательных элементов, расположенных ниже главной диагонали.
53. *** Написать программу, включающую следующие процедуры:
    a) Создать матрицу B(7x7) случайных чисел в диапазоне от –40 до +40.
    b) Напечатать матрицу.
    c) Поменять местами столбец, в котором находится максимальный элемент, с первым столбцом.
    d) Процедура печати должна позволять просматривать как исходную, так и получившуюся матрицу.

    Смотрите примеры 91-92
54. *** Написать программу, включающую следующие процедуры:
    a) Создать матрицу C(8x8) случайных чисел в диапазоне от –30 до +30.
    b) Напечатать матрицу.

c) Найти суммы элементов матрицы как элементов черных и белых полей шахматной доски.

Указания: Зарисуйте двумерный массив с указанием нумерации элементов. Просмотрите внимательно закономерность в нумерации элементов, расположенных как черные и белые клетки шахматной доски.

55. *** Написать программу, включающую следующие процедуры:
   a) Создать матрицу A(7x7) случайных чисел в диапазоне от –35 до +35.
   b) Напечатать матрицу.
   c) Найти столбец с наибольшим количеством положительных элементов (Напечатать номер найденного столбца)

Смотрите примеры 91-92

56. *** Написать программу, включающую следующие процедуры:
   a) Создать матрицу B(6x6) случайных чисел в диапазоне от –50 до +50.
   b) Напечатать матрицу.
   c) Удалить строку, в которой находится минимальный элемент матрицы
   d) Процедура печати должна позволять просматривать как исходную, так и получившуюся матрицу.

Смотрите примеры 84-85, 91-92

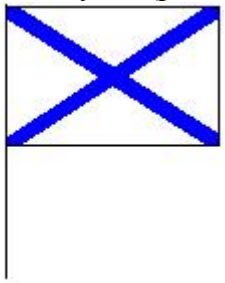
57. *** Написать программу, включающую следующие процедуры:
   a) Создать матрицу C(9x9) случайных чисел в диапазоне от –20 до +20.
   b) Напечатать матрицу.
   c) Расставить строки таким образом, чтобы элементы в первом столбце были упорядочены по убыванию.

d) Процедура печати должна позволять просматривать как исходную, так и получившуюся матрицу.
Указания: Примените алгоритм сортировки и вставьте еще один вложенный цикл.

58. *** Написать программу, включающую следующие процедуры:
a) Создать матрицу A(8x8) случайных чисел в диапазоне от –30 до +30.
b) Напечатать матрицу.
c) Упорядочить элементы нечетных строк матрицы
d) Процедура печати должна позволять просматривать как исходную, так и получившуюся матрицу.
Указания: Примените алгоритм сортировки и вставьте еще один вложенный цикл.
59. *** Дан массив A(2, 10). В первом столбце содержатся координаты X точек плоскости экрана, а во втором столбце — координаты Y тех же точек. Определите количество точек, попадающих в нижнюю правую четверть экрана, выведите их на экран, а искомые точки выделите другим цветом.
Смотрите приложение процедуры и функции модуля CRT.
60. *** Дан массив 5x4, в котором каждая строка состоит из четырех символов, составляющих английское слово. Отсортируйте массив таким образом, чтобы слова были расположены по алфавиту.
Указания: Примените алгоритм сортировки и вставьте еще один вложенный цикл.
61. *** В массиве R (5x5) упорядочьте строки по возрастанию элементов главной диагонали.
Указания: Примените алгоритм сортировки и вставьте еще один вложенный цикл.
62. **** Определите, является ли заданный массив 3x3 магическим квадратом, т. е. таким, суммы элементов которого в строках, столбцах и главных диагоналях равны между собой.
63. ** Выведите на экран номера строк массива 5x5, сумма элементов которых четна.

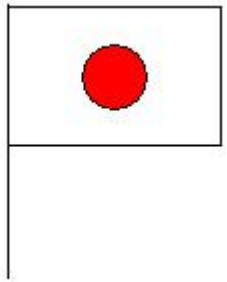
64. **** Выведите на экран изображение Андреевского флага, если у данного массива 5x5 суммы элементов диагоналей равны, и флаг Японии — в обратном случае (рис. 1.72, 1.73).

4	6	8	3	2
6	9	5	3	4
6	0	1	8	7
5	8	2	3	5
7	6	0	3	4



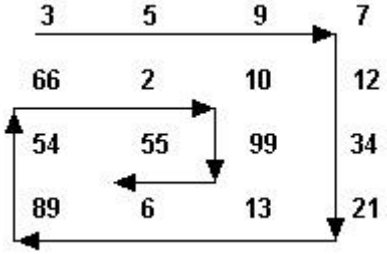
Андреевский флаг (суммы элементов диагоналей равны)

4	6	8	3	2
6	8	5	3	4
6	0	1	8	7
5	8	2	3	5
7	6	0	3	4



Флаг Японии (суммы элементов диагоналей не равны)

65. **** Одномерный массив из N элементов свернуть по спирали в квадратную матрицу размерностью корень квадратный из N по следующему образцу.



Т.е. исходный массив Sl(16) состоит из следующих элементов: 3, 5, 9, 7, 12, 34, 21, 13. 6, 89, 54, 66, 2, 10, 99, 55. Создайте массив S2(4, 4), вид которого представлен на рисунке.

66. **** Даны два одномерных массива из 20 элементов каждый. Элементом является случайное целое двузначное число. Напишите программу с использованием подпрограммы, которая изменяет исходный массив путем деления четных чи сел на их индексы. Используя эту подпрограмму, определите, в каком из массивов было произведено больше замен.
67. ** Ниже приведен фрагмент программы, операторы которой строго следуют друг за другом. Определить результат исполнения операторов 1-16

```
s1:='газон';
s2:='о';
s3:='косильщик';
s4:=concat(s1,s2,s3);
1) writeln(s4);
2) writeln(copy(s4,3,3)+copy(s4,2,1));
   delete(s1,4,2);
3) writeln(s1);
4) writeln(length(s1));
5) writeln(copy(s4,length(s1)+2,2)+copy(s3,length(s1),7));
6) writeln(pos('к',s3));
   s1:=s1+'сварщик';
   insert('о',s1,4);
7) writeln(s1);
8) writeln(length(s1));
9) writeln(upcase(chr(108)));
10)    writeln(upcase(s1[9]));
11)    writeln(ord('g')+ord('G'));
   t:='*H'<'-M';
12)    writeln(t);
13)    {Какого типа переменная t?}
14)    writeln(s1[0]);
15)    writeln(chr(ord(s1[0])+100));
```

```
    s5:=' ';
    for i:=1 to length(s4) do
    begin
       s5:=s4[i]+s5;
    end;
16)      writeln(s5);
```

68. ** Ниже приведен фрагмент программы, операторы которой строго следуют друг за другом. Определить результат исполнения операторов 1-16

```
     c1:='электр';
     c2:='о';
     c3:='емкость';
     c4:=concat(c1,c2,c3);
1)   writeln(c4);
2)   writeln(copy(c4,5,3)+copy(c4,12,3));
     delete(c3,1,2);
3)   writeln(c3);
4)   writeln(length(c3));
5)   writeln(copy(c1,length(c3)-3,2)+copy(c3,length(c3)-2,3));
6)   writeln(pos('т',c4));
     c3:='жи'+c3;
     insert('д',c3,3);
7)   writeln(c3);
8)   writeln(length(c3));
9)   writeln(upcase(chr(114)));
10)  writeln(upcase(c3[3]));
11)  writeln(ord('Q')+ord('q'));
     m:='#C'>'$D';
12)  writeln(m);
13)  {Какого типа переменная m?}
14)  writeln(c3[0]);
15)  writeln(chr(90+ord(c3[0])));
```

```
c5:=' ';
for i:=1 to length(c4) do
  begin
       c5:=c4[i]+c5;
  end;
16)  writeln(c5);
```

69. ** Даны две строки. Получить третью, равную сумме первых двух. Подсчитать в третьей строке число символов, букв, число слов, число слогов.
70. *** Подсчитать, сколько раз входит данное слово в строку и с каких позиций.
    Указания: Необходимо просматривать в цикле каждый символ строки и использовать функцию сору.
71. * Написать программу для замены в тексте слога «ма» на слог «па».
72. * Написать программу, подсчитывающую, сколько раз в данном слове встречается сочетание «со» (например, в слове «согласование» оно встречается 2 раза).
73. * Составить программу, удаляющую из данного слова все буквы «м».
74. ** Написать программу, проверяющую, является ли частью данного слова слово «том». Ответ должен быть «да» или «нет».
75. ** Определить, какая из букв «е» или «о» встречается в предложении чаще.
76. ** Написать программу, определяющую, на какую букву начинается второе слово в предложении.
77. ** Найти сумму кодов ASCII ,букв своего имени
78. ** Получить из слова «компьютер» три новых слова.
79. ** Подсчитать количество букв «а» в предложении.
80. *** Написать программу, с помощью которой вводимое пользователем слово записывается наоборот.
    Смотрите последнее задание задач 67, 68.
81. ** Удалить из предложения все буквы «о».
82. *** Написать программу, проверяющую, является ли введенное слово или фраза палиндромом, т.е. читающимся

слева направо и справа налево одинаково (например, «шалаш», «казак», «А роза упала на лапу Азора»). Программа сообщает: «Да, это палиндром» или «Нет, это не палиндром» и выводит на экран введенный текст в варианте слева направо и справа налево.

Смотрите последнее задание задач 67, 68.

83. *** Напишите программу простейшей шифровки информации, когда определяется код каждого символа строки и добавляется к нему единица, затем выводится вместо старого символа строки символ с новым получившимся кодом. Слово для тестовой проверки «CAT». После шифровки должно получиться «DBU». Программа должна включать процедуры шифрации и дешифрации.

    Указания: Необходимо просматривать в цикле каждый символ и заменять его зашифрованным. Используйте функцию ord.

84. **** Анаграмма – это слово, в котором перевернуты буквы, например, «ШАДОЛЬ» - «ЛОШАДЬ», а «ТИВОНКР» - это «ВТОРНИК». Программа рассчитана на двух игроков, соревнующихся друг с другом. В качестве слов используются только имена существительные, нарицательные, в единственном числе, в именительном падеже. Первый игрок вводит с клавиатуры слово длиной не менее пяти, но не более восьми букв (постарайтесь сделать так, чтобы во время ввода на экране вместо букв отображались символы «*»). Затем компьютер определяет длину введенного слова, разбивает его на отдельные символы и заносит их в массив, откуда случайным образом выводит на экран. Соперник в течение 2-х минут (время определяет компьютер) должен определить это слово. В случае правильного ответа (который он вводит с клавиатуры) он получает 1 очко, а сам загадывает слово первому игроку. Игра идет до пяти очков.

85. За неправильный ответ очки не начисляются. Безусловно, приветствуется дружественный интерфейс? Запрос и обращение к игрокам по именам, вывод на экран правил игры, графическое и звуковое оформление.
86. * Напишите программу, выводящую на экран символы, скрывающиеся за кодами 129—255. Распечатайте или выпишите коды строчных и прописных букв кириллицы.
87. **** Напишите программу, которая выдаст на экран пять слов максимальной длины из слова "ЭЛЕКТРИЧЕСТВО". Побеждает тот, у кого сумма букв во всех словах наибольшая.
88. **** Используя пример с подсчетом слов в телеграмме, напишите программу, имитирующую отделение связи с очень хорошим обслуживанием. Программа должна выяснять имя клиента и в дальнейшем обращаться к нему только по имени. Запрашивается также регион, куда посылается телеграмма. Их три — Россия (коэффициент 1), страны СНГ (стоимость одного слова умножается на 2) и дальнее зарубежье (стоимость одного слова умножается на 5). По России стоимость одного слова составляет 1 руб. 50 коп. (причем неважно, какой длины слово). Затем у клиента запрашивается текст телеграммы и денежная сумма, определяется количество слов, стоимость телеграммы. Если денег ровно столько, сколько надо, его благодарят и прощаются. Если больше, чем надо, то ему предлагают сдачу и прощаются. Если — меньше, то просят добавить необходимую сумму, а затем, после расчета, с клиентом прощаются. А для пущей красоты я обычно прошу нарисовать окошко телеграфа, в прорезях которого и происходит диалог компьютера с пользователем.

89. *** С клавиатуры вводятся 10 слов. Напишите программу, которая:
    a) напечатает все слова из списка, отличные от слова "SUN";
    b) напечатает слово, ближайшее к началу алфавита в списке (считаем, что все слова различны; выполнять аналогично поиску минимального из ряда чисел);
    c) слово, составленное из последних символов всех слов списка;
    d) все слова из списка, содержащие три буквы.

    Указания: Необходимо просматривать в цикле каждый символ строки
90. ** В тексте, содержащем между словами от 1 до 3 пробелов, оставить только по одному.
91. *** Подсчитать, сколько раз входит каждый символ в данную строку.
92. *** Написать программу шифратор и дешифратор, ставящую в соответствие русским символам соответствующие латинские и наоборот (аналог так называемого конвертора).

    Указания: Необходимо просматривать в цикле каждый символ строки. Используйте таблицу кодов (приложение)
93. * Имеются три множества символьного типа, которые заданы своими конструкторами:

```
y1=['A','B','D','R','H'];
```

```
y2=['R','A','H','D'];
y3=['A','R'].
```

Сформировать новое множество $x = (y1 \cup y2) \cup (y1 \setminus y2)$

$\cup$ - объединение множеств;

\ - разность множеств.

Вывести на печать полученное множество. Далее проверить, включено ли множество y3 в множество x.

Указания: Для формирования нового множества x воспользуйтесь оператором присваивания. Для вывода на экран элементов нового множества примените оператор цикла for. Параметром цикла возьмите символьную переменную, принимающую значение каждого символа латинского алфавита от A до R включительно.

94. ** Из множества целых чисел `1..20` выделить:
    a) Множество чисел, делящихся на 6 без остатка;
    b) Множество чисел, делящихся без остатка или на 2 или на 3.

    Указания: Сначала создайте множество чисел, делящихся на 2, затем множество чисел, делящихся на три. Объединение данных множеств – это числа, делящиеся на 2 или на три, а пересечение – это чила, делящиеся на 6.

95. ** Ниже приведен фрагмент программы, операторы которой строго следуют друг за другом. Определить результат исполнения операторов 1-8.

```
n1:=[1..5,7..9];
n2:=[];
n3:=[8,9,10];
n4:=n1*n3;
n5:=n1+n3;
n6:=n5-n4;
for i:=1 to 10 do
    if i in n6 then
1)  write(i,' ');
writeln;
k:=[1..6,8]>=n1;
2)  {Какого типа переменная k?}
```

```
3)  writeln(k);
    for i:=1 to 10 do
        if (i mod 2=0) and (i in n1) then n2:=n2+[i]; n7:=n3*n2;
    for i:=1 to 10 do
        if i in n2 then
4)  write(i,' ');
    writeln;
    for i:= 1 to 10 do
        if i in n7 then
5)  write(i,' ');
    writeln;
    exclude(n6,10);
    include(n6,8);
    include(n6,9);
    c:=n6=n1;
6)  {Какого типа переменная c?}
7)  writeln(c);
    for i:=1 to 10 do
        if i in (n2+n3)-n1
            then
8)  write(i,' ');
```

96. ** Ниже приведен фрагмент программы, операторы которой строго следуют друг за другом. Определить результат исполнения операторов 1-8.

```
    n1:=['A'..'L','N','O','R'];
    n2:=['B'..'D'];
    n3:=[];
    n4:=n1*n2;
    n5:=n1+n2;
    n6:=n5-n4;
    for s:='A' to 'R' do
        if s in n6 then
1)  write(s,' ');
```

```
writeln;
k:=['A','C','E'..'L']<=n1;
{Какого типа переменная k?}
writeln(k);
for s:='A' to 'R' do
   if (ord(s) mod 2=0) and (s in n1) then n3:=n3+[s]; n7:=n3*n2;
for s:='A' to 'R' do
   if s in n3 then
write(s,' ');
writeln;
for s:='A' to 'R' do
   if s in n7 then
write(s,' ');
writeln;
exclude(n2,'C');
include(n2,'M');
include(n4,'A');
c:=n2=n4;
{Какого типа переменная c?}
writeln(c);
for s:='A' to 'R' do
   if s in n3*n1-n5+n4
      then
write(s,' ');
```

97. ** Придумайте самостоятельно задачи на использование множеств, процедур и функций работы с ними.
   a) Используйте операции сложения, вычитания, умножения над множествами. Выведите результаты на экран;
   b) Используйте операции сравнения. Выведите результаты на экран;
   c) Используйте процедуры Include, Exclude. Выведите результаты на экран.
98. *** Разработать программу по одному из 4-х предложенных вариантов с процедурами, задания к которым перечислены ниже:
   a) Создать список, в котором имеются поля строкового, числового, логического типа, создать вычисляемое поле;

b) Вывести список на экран;
c) Найти запись с максимальным или минимальным значением одного из полей и выдать о нем все сведения;
d) Упорядочить список по алфавиту или в порядке возрастания значений какого-либо поля;
e) Найти количество записей со значением какого-либо поля, введенным пользователем;
f) В основной программе предусмотреть главное меню.

1) Список книг библиотеки с полями: номер, название, дата сдачи, сдана / не сдана. Вычисляемое поле – задолженность по книге (если книга не сдана и дата сдачи меньше сегодняшней даты, то книга числится в задолжниках)

2) Список междугородних маршрутов автобусов: номер маршрута, пункт отправления, пункт назначения, стоимость проезда, количество мест, форма собственности (частный или государственный). Вычисляемое поле – вырученные деньги за проезд.

3) Список служащих фирмы: номер, ФИО, количество отработанных часов, ставка за час, должность (руководитель или служащий). Вычисляемое поле – начисленные средства.

4) Список отделений в больнице: номер, название, количество палат, количество коек палате, тип отделения (стационарное / амбулаторное). Вычисляемое поле – на сколько больных рассчитано каждое отделение.

Смотрите пример 131

ТЕСТОВЫЕ ЗАДАНИЯ ДЛЯ ПРОВЕРКИ ТЕОРЕТИЧЕСКИХ ЗНАНИЙ

1. С помощью следующих операторов осуществляется

```
c:=a[1];
for i:=1 to 10 do
    if a[i]>c then c:=a[i];
```

A) нахождение суммы элементов одномерного массива;
B) поиск минимального элемента в одномерном массиве;
C) нахождение количества элементов одномерного массива;
D) поиск максимального элемента одномерного массива; *
E) сортировка одномерного массива.

2. С помощью следующих операторов осуществляется:

```
for j:=10 downto 1 do
begin
        max:=a[1];
        c:=1;
        for i:=1 to j do if a[i]<max then
                begin
                        max:=a[i];
                         c:=i;
                end;
        r:=a[c];
        a[c]:=a[j];
        a[j]:=r;
end;
```

A) поиск минимального элемента двумерного массива;
B) сортировка выбором по возрастанию одномерного массива;
C) сортировка «пузырьком» по возрастанию одномерного массива;
D) сортировка выбором по убыванию одномерного массива;
E) сортировка «пузырьком» по убыванию одномерного массива

3. Результат исполнения следующих операторов:

```
for i:=1 to 5 do
    for j:=1 to 5 do
```

```
begin
        if j>i then a[i,j]:=0;
end;
```

A) замена элементов главной диагонали массива [5,5] нулями;
B) замена элементов ниже главной диагонали массива [5,5] нулями;
C) замена элементов выше главной диагонали массива [5,5] нулями; *
D) замена элементов выше побочной диагонали массива [5,5] нулями;
E) замена элементов ниже побочной диагонали массива [5,5] нулями.

4. Результат исполнения следующих операторов:

```
for i:=1 to 5 do
    begin
        c:=a[i,1];
        for j:=1 to 5 do
                begin
                        if a[i,j]>c then c:=a[i,j];
                 end;
```

A) поиск максимального элемента каждой строки массива [5,5]; *
B) поиск минимального элемента каждого столбца массива [5,5];
C) поиск максимального элемента каждого столбца массива [5,5];
D) поиск максимального элемента главной диагонали массива [5,5];
E) поиск максимального элемента массива [5,5].

5. С помощью следующих операторов осуществляется:

```
c:=0;
```

```
for i:=1 to 10 do
    if a[i]>0 then c:=c+a[i];
```

A) нахождение суммы положительных элементов массива;*
B) поиск минимального элемента в массиве;
C) нахождение количества положительных элементов массива;
D) поиск максимального элемента массива;
E) сортировка массива.

6. С помощью следующих операторов осуществляется:

```
for j:=1 to n-1 do
    for i:=1 to n-j do
        if a[i]>a[i+1] then
                begin
                        r:=a[i];
                        a[i]:=a[i+1];
                        a[i+1]:=r;
                end;
```

A) замена элементов двумерного массива, стоящих на четных и нечетных местах;
B) сортировка выбором по возрастанию элементов одномерного массива;
C) сортировка «пузырьком» по возрастанию элементов одномерного массива; *
D) сортировка выбором по убыванию элементов одномерного массива;
E) сортировка «пузырьком» по убыванию элементов одномерного массива.

7. Результат исполнения следующих операторов:

```
for i:=1 to 5 do
    for j:=7-i downto 5 do
        a[i,j]:=0;
```

A) замена элементов главной диагонали массива [5,5] нулями;
B) замена элементов ниже главной диагонали массива [5,5] нулями;

C) замена элементов выше главной диагонали массива [5,5] нулями;
D) замена элементов выше побочной диагонали массива [5,5] нулями;
E) замена элементов ниже побочной диагонали массива [5,5] нулями. *

8. Результат исполнения следующих операторов:

```
for j:=1 to 5 do
    begin
        s:=0;
        for i:=1 to 5 do
                begin
                        s:=s+a[i, j];
                end;
    end;
```

A) сумма элементов каждого столбца массива [5,5]; *
B) сумма элементов каждой строки массива [5,5];
C) сумма элементов главной диагонали массива [5,5];
D) сумма элементов каждой строки массива [5,5] и максимальная из них;
E) сумма элементов каждого столбца массива [5,5] и максимальная из них.

9. С помощью следующих операторов осуществляется:

```
c:=0;
for i:=1 to 10 do
    if a[i]>0 then c:=c+1;
```

A) нахождение суммы положительных элементов массива;
B) поиск минимального элемента в массиве;
C) нахождение количества положительных элементов массива; *
D) поиск максимального элемента массива;
E) сортировка массива.

10. Результат исполнения следующих операторов:

```
for i:=1 to 5 do
    for j:=1 to 5-i do
        begin
                a[i, j]:=0;
        end;
```

A) замена элементов главной диагонали массива [5,5] нулями;
B) замена элементов выше главной диагонали массива [5,5] нулями; *
C) замена элементов ниже главной диагонали массива [5,5] нулями;
D) замена элементов выше побочной диагонали массива [5,5] нулями;
E) замена элементов ниже побочной диагонали массива [5,5] нулями.

11. Результат исполнения следующих операторов:

```
m:=-999999999;
for i:=1 to 5 do
    begin
        s:=0;
        for j:=1 to 5 do
                s:=s+a[i, j];
        if s>m then m:=s ;
    end;
```

A) сумма элементов каждого столбца массива [5,5];
B) сумма элементов каждой строки массива [5,5];
C) сумма элементов каждой строки массива [5,5] и максимальная из них; *
D) сумма элементов главной диагонали массива [5,5];
E) сумма элементов каждого столбца массива [5,5] и максимальная из них.

12. С помощью следующих операторов осуществляется:

```
c:=0;
for i:=1 to 10 do
    if a[i] mod 2<>0 then c:=c+a[i];
```

A) нахождение суммы нечетных элементов массива; *
B) поиск минимального элемента в массиве;
C) нахождение количества нечетных элементов массива;
D) поиск максимального элемента массива;
E) нахождение суммы четных элементов массива.

13. Результат исполнения следующих операторов:

```
for i:=1 to 5 do
    for j:=1 to 5 do
        begin
                r:=a[1,j];
                a[1,j]:=a[5,j];
                a[5,j]:=r;
        end;
```

A) замена местами четных и нечетных столбцов массива [5,5];
B) замена местами четных и нечетных строк массива [5,5];
C) замена местами первого и последнего столбцов массива [5,5];
D) замена местами первой и последней строк массива [5,5];
E) замена местами максимального и минимального элементов массива [5,5].

14. Результат исполнения следующих операторов:

```
for i:=1 to 5 do
    for j:=1 to 5 do
        for n:=1 to 6-j do
             if a[i,n]>a[i,n+1] then
            begin
                    r:=a[i,n];
                    a[i,n]:=a[i,n+1];
                    a[i,n+1]:=r;
            end;
```

A) сортировка по возрастанию элементов каждой строки массива [5,5]; *
B) сортировка по возрастанию элементов каждого столбца массива [5,5];
C) сортировка по убыванию элементов каждой строки массива [5,5];
D) сортировка по убыванию элементов каждого столбца массива [5,5];
E) сортировка массива [5,5] по возрастанию.

15. Результат исполнения следующих операторов:

```
s1:='газон';
s2:='о';
s3:='косильщик';
s4:=concat(s1,s2,s3);
writeln(copy(s4,3,3)+copy(s4,2,1));
```

A) газонокосильщик;
B) зона; *
C) газосварщик;
D) коза;
E) носильщик.

16. Результат исполнения следующих операторов:

```
s1:='газон';
s2:='о';
s3:='косильщик';
s4:=concat(s1,s2,s3);
writeln(copy(s4,length(s1)+2,2)+copy(s3,length(s1)-2,7));
```

A) косильщик; *
B) зона;
C) заноза;
D) коза;
E) носильщик.

17. Результат исполнения следующих операторов:

```
s4:=’телефон’
s5:=' ';
for i:=1 to length(s4) do
    begin
```

```
        s5:=s4[i]+s5;
    end;
writeln(s5);
```
A) нолетеф;
B) телефон;
C) 7;
D) нофелет; *
E) 1234567

18. Результат исполнения следующих операторов:

```
n1:=[1..5,7..9];
n2:=[8,9,10];
n3:=n1*n2;
for i:=1 to 10 do
    if i in n3 then write(i,' ');
```
A) 1 2 3 4 5 7 8 9 10;
B) 1 2 3 4 5 6 7 8 9 10;
C) 8 9; *
D) 8 9 10;
E) 1 2 3 4 5 7 10

19. Результат исполнения следующих операторов:

```
n1:=[1..5,7..9];
n2:=[];
n3:=[1,2,4,6,8,9,10];
for i:=1 to 10 do
    if (i mod 2=0) and (i in n1) then n2:=n2+[i];
n4:=n3-n2;
for i:=1 to 10 do
    if i in n4 then write(i,' ');
```
A) 8 9;
B) 1 6 9 10; *

C) 1 2 3 4 5 6 7 8 9 10;
D) 2 4 8;
E) 2 4 6 8.

20. В нижеприведенном разделе описания программы идентификаторы spisok, FIO, a обозначают соответственно:

```
program spisok_gr;
type
    spisok=record
        nom:imnteger;
        FIO:string[20];
        ocenka:integer
    end;
var
    a:spisok;
```

A) переменная типа запись, тип запись, поле записи
B) тип запись, переменная типа запись, поле записи;
C) поле записи, тип запись, переменная типа запись;
D) поле записи, переменная типа запись, тип запись;
E) тип запись, поле записи, переменная типа запись. *

21. Что обозначает последовательность символов a.fio[1] для описанной записи?

```
program spisok_gr;
type
    spisok=record
        nom:imnteger;
        FIO:string[20];
        ocenka:integer
    end;
var
    a:spisok;
```

A) обращение к первому учащемуся группы;
B) обращение к первой фамилии учащегося, начинающейся на «А»;
C) обращение к первому учащемуся с наивысшей оценкой;
D) обращение к первой букве фамилии учащегося; *

E) неверный синтаксис выражения.

22. С помощью следующей последовательности операторов осуществляется (выберите наиболее точное определение):

```
program spisok_gr;
type
    spisok=record
        nom:imnteger;
        FIO:string[20];
        ocenka:integer
    end;
var
    a:spisok;
    sp:array [1..20] of spisok;
    n, i:integer;
begin
    readln(n);
    for i:1 to n do
        begin
                readln(a.nom);
                readln(a.FIO);
                readln(a.ocenka);
                sp[i]:=a;
        end;
end;
```

A) создание списка, элементами которого являются сведения об учащихся: номер учащегося, его ФИО и оценка;

B) создание списка, элементами которого являются сведения об учащихся: номер учащегося, его ФИО и оценка, запись его в массив и в файл;

C) создание списка, элементами которого являются сведения об учащихся: номер учащегося, его ФИО и оценка и запись его в массив; *
D) создание массива типа запись, элементами которого являются сведения об учащихся: номер учащегося, его ФИО и оценка;
E) заполнение массива полями записи.

23. Количество элементов множества – это

A) Объект множества;
B) Размер множества;
C) Набор множества;
D) Мощность множества; *
E) Диапазон множества.

24. Объекты, называемые записями относятся к

A) Комбинированному типу; *
B) Регулярному типу;
C) Перечисляемому типу;
D) Скалярному типу;
E) Пользовательскому типу.

25. Оператор With используется при работе с типом данных Записи для

A) Обращения непосредственно к имени поля записи; *
B) Для быстрого поиска записи;
C) Обращения к имени переменной, отделяемой от записи;
D) Обращения к имени записи;
E) Для выбора поля записи.

26. Переменные f, st определены следующим образом в разделе переменных

```
var st: string[4];
    f:integer;
```

Определите процедуру, которая переводит текстовое представление переменной st в ее числовое представление в переменную f.

A) Val(st, f, code); *
B) Pos(‘5’, st, k);
C) Copy(st, f,2,4);

D) Length(st);
E) Str(a, st).

27. Дан фрагмент программы

```
var st, st1:string;
begin
    st:=’abcdef’;
    st1:=copy(st,3,2);
end;
```

Значение пременной st1 равно

A) «aef»;
B) «bcd»;
C) «bd»;
D) «abcd»;
E) «cd». *

28. Составное имя поля записи состоит из

A) Имени записи и имени поля, разделенных точкой; *
B) Имени поля и соответствующего расширения;
C) Имени поля и порядкового номера поля в записи;
D) Идентификатора переменной и идентификатора поля;
E) Имени поля и имени записи, разделенных точкой.

29. Операция in проверяет

A) Отрицание вхождения одного множества в другое;
B) Принадлежность какого-нибудь элемента множества указанному множеству; *
C) Пересечение множеств;
D) Вхождение одного множества в другое;
E) Отрицание принадлежности элемента указанному множеству.

30. Для вывода на экран элементов побочной диагонали матрицы A размерностью nxn необходимо описать следующий фрагмент

A) for i:=1 to n do

write(‘ ’, A[i, i]);

B) for i:=2 to n do
write(‘ ‘, A[i, n-1]);

C) for i:=1 to n do
write(‘ ’, A[i, i+1];

D) for i:=1 to n do
write(‘ ’, A[i, n-1];

E) for i:=1 to n do
write(‘ ’, A[i, n+1-i]; *

31. Описание типа записи

A) Начинается с имени записи и заканчивается именем последнего поля;

B) Начинается с первого идентификатора поля записи и заканчивается идентификатором последнего поля;

C) Начинается с зарезервированного слова record и заканчивается зарезервированным словом end;

D) Начинается с описания типа записи и заканчивается описанием типов полей записи;

E) Начинается с имени записи и зарезервированного слова record и заканчивается зарезервированным словом end.*

32. Дан фрагмент программы:

```
for i:=1 to n do
        for j:=1 to m do
                a[i,j]:=(sin(i)+cos(j))/2;
```

Назначение данного фрагмента – это

A) Заполнение двумерного массива с использованием формулы; *

B) Линейная сортировка двумерного массива по возрастанию;

C) Вычисление элементов двумерного массива по формуле;

D) Заполнение и сортировка одномерного массива с использованием формул;

E) Сортировка двумерного массива методом пузырька.

33. Если множество m1>=m2, это означает

A) Все элементы множества m1 содержатся в m2;

B) Значения элементов множества m2 больше значений элементов множества m1;
C) Количество элементов множества m1 больше количества элементов множества m2;
D) Значения элементов множества m1 больше значений элементов множества m2;
E) Все элементы множества m2 содержатся в m1 *

34. Для определения местоположения элемента одномерного массива необходимо определить

A) Его значение;
B) Его индекс; *
C) Количество его элементов;
D) Его размер;
E) Его тип.

35. Объем памяти, требуемый для размещения записи, складывается из

A) Из объема памяти, занимаемой полями записи, плюс длина имени записи;
B) Из объема памяти, занимаемого полями записи; *
C) Из объема памяти, отводимой переменной определяемой от записи;
D) Специально для типа данных записи объем памяти не отводится;
E) Из сумм значений полей записи.

36. Дан фрагмент программы

```
var st:string;
begin
        st:=’abcdef’;
        delete(st, 2, 3);
end;
```

Значение st после операции равно

A) “abef”;
B) “abf”;

C) “aef”; *
D) “bcd”;
E) “cde”.

37. Для преобразования числа, выраженного в числовом представлении, в его строковое представление служит стандартная процедура

A) Pos;
B) Var;
C) Copy;
D) Str; *
E) Insert.

38. Массивы бывают

A) Одномерные и разномерные;
B) Одномерные и многомерные; *
C) Числовые и вещественные;
D) Символьные или файловые;
E) Строчные и числовые.

39. Для отрицания принадлежности элемента указанному множеству M необходимо записать операцию

A) x not in M;
B) not (x in M); *
C) x not (in M);
D) (x not) in M;
E) (x not in) M.

40. Переменные st, k определены следующим образом в разделе переменных: var st: string[4]; k:integer; Назначение стандартной функции k:=length(st)

A) Копирование значения переменной st в переменную k;
B) Вычисление длины в символах переменной st; *
C) Перевод значения переменной st из символьного значения в числовое;
D) Вычисление объема памяти, занимаемой переменной St;
E) Перемещение значения переменной st в переменную k.

41. Последовательность символов представляет собой

A) Скалярный тип данных;
B) Строковый тип данных; *

C) Пользовательский тип данных;
D) Символьный тип данных;
E) Простой тип данных.

42. Массив – это

A) Фиксированный набор данных, имеющий общее имя, содержащий переменные разного типа данных;
B) Упорядоченная оп номерам последовательность однотипных данных, расположенных на носителе информации; *
C) Нефиксированная структура данных, содержащих переменные разного типа;
D) Структура данных, содержащих переменные одинакового типа и имеющих одинаковые значения;
E) Нефиксированная структура данных, содержащих переменные одного типа.

43. Даны множества m1 и m2: m1:=[1, 2, 3, 4]; m2:=[3, 4, 1]. Для них определена операция m3:=m1-m2. Множество m3 равно

A) [1, 2,3];
B) [-2,2,2];
C) [1,3,4];
D) [0,2,0,4];
E) [2]. *

44. Запись определена следующим образом:

```
type
car=record
        nom, oklad, stag: integer;
        fio, adres: string[25];
end;
var m:car;
```

Неправильное обращение к полям записи – это

A) m.nom:=1;

B) m.stag:=5;
C) m.oklad:=8000;
D) m.fio:=’Абаева Н.Т.’;
E) adres:=’Семеновой, 15-20’. *

45. Запись относится к комбинированным типам данных, потому что

A) Состоит из нескольких полей одного типа;
B) Состоит из разных компонент, имеющих различные значения;
C) Может состоять из нескольких полей различного типа; *
D) Состоит из разных компонент разных длин;
E) Состоит из фиксированного числа компонент одного и того же типа.

46. Основные операции с массивами – это

A) Перебор элементов массива, нахождение номера индекса, нахождение размерности
B) Определение типа элемента массива, сравнение элементов массива, ввод количества элементов массива
C) Подсчет индекса, количества элементов
D) Ввод и вывод значений в элементы массива, поиск максимального или минимального элемента, нахождение суммы элементов массива *

5. ВВОД – ВЫВОД ДАННЫХ

5.1. Основные понятия. Классификация файлов

В практике программирования часто встречаются задачи, при решении которых удобно хранить обрабатываемые данные на внешнем носителе. В этом случае данные оформляются в виде внешних файлов. До сих пор мы рассматривали задачи, в которых исходные данные поступали с клавиатуры, дисплея в память ЭВМ, а результаты вычислений не сохранялись. Всякий раз при выполнении одной и той же программы приходится заново вводить исходные данные. Создание внешних файлов для постоянного хранения на магнитном диске и возможность обрабатывать эти файлы в программе является эффективным средством программирования.

Под файлом понимается либо именованная область внешней памяти ПК (ж.д., г.д. и т.д), либо логическое устройство – потенциальный источник или приемник информации. В языке Паскаль файл представляет собой последовательность элементов одного типа (простого или сложного). В отличие от массива длина файла, т.е. количество элементов, не задается, место элемента не определяется индексом и каждый элемент становится доступным только после перебора всех предыдущих элементов. При считывании файла в оперативную память машины символы файла преобразуются в тот тип данных, который объявлен в программе.

Любой файл имеет 3 характерные особенности. Во-первых, у него есть имя, что дает возможность программе работать одновременно с несколькими файлами. Во-вторых, он содержит компоненты одного типа. Типом может любой тип Паскаля, кроме файлов. В третьих, длина файла никак не оговаривается при его объявлении и ограничивается только емкостью устройств внешней памяти. Файловый тип или переменную файлового типа можно задать одним из трех способов:

<имя>=file of <тип>;
<имя>=text;
<имя>=file;

<имя> - имя файлового типа, **file, of** – зарезервированные слова, text – имя стандартного типа текстовых файлов, <тип> - любой тип ТП, кроме файлов.

В зависимости от объявления можно выделить 3 вида файлов:

Типизированные файлы (задаются предложением **file of**);

Текстовые файлы (определяются типом text);

Нетипизированные файлы (определяются типом file).

Пример 132. Примеры описания файловых переменных

```
type
product=record
        name: string;
        code: word;
        cost: comp
end;
text80=file of string[80];
var
        f1: file of char;
        f2: text;
        f3: file;
        f4: text80;
        f5: file of product;
```

В нашем примере f1, f4, f5 – типизированные файлы, f2 – текстовый файл, f3 – нетипизированный файл.

Вид файла определяет способ хранения информации в файле.

Любые файлы, а также логические устройства становятся доступны программе только после выполнения особой процедуры открытия файла. Эта процедура заключается в связывании ранее объявленной файловой переменной с именем существующего или вновь создаваемого файла, а также в

указании направления обмена информацией: чтение из файла или запись в него.

Файловая переменная связывается с именем файла в результате обращения к стандартной процедуре assign:

assign (<ф.п.>, <путь к файлу с именем и расширением .txt, .dat, .pas>);

Инициация файла

Инициировать файл означает указать для этого файла направление передачи данных. В ТП можно открыть файл для чтения, для записи информации, а также для чтения и записи одновременно. Для чтения файл инициируется с помощью стандартной процедуры reset:

reset (<ф.п.>);

дисковый файл или логическое устройство подготавливаются к чтению информации.

Если делается попытка инициировать чтение из несуществующего файла, возникает ошибка периода исполнения, которая может быть сообщена программе ненулевым значением встроенной функции ioresult типа word.

Пример 133. Исполните следующую программу. После первого исполнения создайте в папке BIN документ myfile.dat и еще раз исполните программу.

```
uses crt;
var f: file of char;
begin
        clrscr;
        assign (f, 'myfile.dat');
        {$I-}   {отключаем контроль ошибок ввода-вывода}
        reset(f);
        {$I+}   {включаем контроль ошибок ввода-вывода}
        if ioresult<>0 then
                writeln('файл не существует')
        else
```

```
            writeln('файл существует');
        readln
end;
```

Стандартная процедура

rewrite (<ф.п.>);

инициирует запись информации в файл, связанный ранее с файловой переменной <ф.п.>. С помощью нее нельзя инициировать запись информации в ранее существующий дисковый файл: при выполнении этой процедуры старый файл уничтожается и никаких сообщений при этом в процедуру не передается. Новый файл подготавливается к приему информации и его указатель принимает значение 0.

Стандартная процедура

append (<ф.п.>);

инициирует запись в ранее существующий текстовый файл для его расширения. Указатель устанавливается в конец файла. Работает только для текстовых файлов.

Процедура

close (<ф.п.>);

закрывает файл. Обязательна по окончании работы с файлом.

Процедура

rename (<ф.п.>, <новое имя>);

переименовывает закрытый файл.

Процедура

erase (<ф.п.>);

уничтожает закрытый файл.

Функция

eof (<ф.п.>): boolean;

логическая функция, тестирующая конец файла. True, если указатель файла в конце.

5.2. Текстовые файлы

Текстовые файлы предназначены для хранения текстовой информации. Именно в такого типа файлах хранятся, например, исходные тексты программ. Компоненты (записи)

текстового файла могут пиметь п6кременную длину, что существенно влияет на характер работы с ними.Доступ к каждой строке текстового файла возможен лишь последовательно, начиная с первой. При создании текстового файла в конце каждой записи (строки) ставится специальный признак EOLN (End Of Line – конец строки), а в конце всего файла – признак EOF (End Of File – конец файла). При формировании текстовых файлов используются следующие системные соглашения:

EOLN – последовательность кодов ASCII #13(CR) и #10(LF);

EOF – код #26 стандарта ASCII.

Для доступа к записям применяются процедуры READ, READLN, WRITE, WRITELN. Они отличаются возможностью обращения к ним с переменным числом фактических параметров, в качестве которых могут использоваться символы, строки и числа. Первым параметром в любой из перечисленных процедур может стоять файловая переменная. В этом случае осуществляется обращение к дисковому файлу или логическому устройству, связанному с переменной процедурой ASSIGN. Если файловая переменная не указана, происходит обращение к стандартным файлам INPUT и OUTPUT.

Процедура READ. Обеспечивает ввод символов, строк и чисел. Формат обращения:

READ (<ф.п.>, <сп. ввода>) или READ(<сп. ввода>)

Здесь <сп. ввода> – список ввода: последовательность из одной или более переменных типа CHAR, STRING, а также любого целого или вещественного типа. Это переменные, в которые записываются данные из файла.

Процедура READ прекрасно приспособлена к вводу чисел. При обращении к ней за вводом очередного целого или

вещественного числа процедура «перескакивает» маркеры конца строк, т.е. фактически весь файл рассматривается ею как одна длинная строка, содержащая текстовое представление чисел. В сочетании с проверкой конца файла функцией EOF процедура READ позволяет организовать простой ввод массивов данных.

Пример 134. Считывание из файла массива данных и вывода его на экран. Предварительно создайте файл 'prog.txt' в папке BIN и заполните его вещественными числами в строку через пробел.

```
uses crt;
const
N = 1000; Максимальная длина ввода var
f : text;
m : array [1..N] of real;
i : integer;
begin
        clrscr;
        assign(f, 'prog.txt') ; reset(f);
        i := 1;
        while not EOF(f) and (i <= N) do
        begin
                read(f ,m[i] ) ;
                write(m[i]:6:2);
                inc(i)
        end;
        close(f);
        readln
end.
```

Процедура READLN. Обеспечивает ввод символов, строк и чисел. Эта процедура идентична процедуре READ за исключением того, что после считывания последней переменной оставшаяся часть строки до маркера EOLN пропускается, поэтому следующее обращение к READLN или READ начинается с первого символа новой строки. Кроме того, эту процедуру можно вызвать без параметра <cn.ввода> (см.

процедуру READ), что приведет к пропуску всех символов текущей строки вплоть до EOLN.

Процедура WRITE. Обеспечивает вывод информации в текстовый файл или передачу ее на логическое устройство. Формат обращения:

WRITE (<ф.п.>, <сп.вывода>) или WRITE (<сп.вывода>)

Здесь <сп.вывода> - список вывода: последовательность из одного или более выражений типа CHAR, STRING, BOOLEAN, а также любого целого или вещественного типа.

Файловая переменная <ф.п.>, если она указана, должна быть предварительно описана как е65менная типа TEXT и связана с именем файла или логическим устройством процедурой ASSIGN. Если файловая переменная отсутствует, подразумевается вывод в стандартный файл OUTPUT, который обычно связан с экраном ПК.

Процедура WRITELN. Эта процедура полностью идентична процедуре WRITE за исключением того, что выводимая строка символов завершается кодами CR и LF. При вызове WRITELN можно опускать параметр <сп.вывода>: в этом случае в файл передается маркер EOLN, что при выводе на экран приведет к переводу курсор» в начало следующей строки.

Логическая функция EOLN возвращает TRUE, если во входном текстовом файле достигнут маркер конца строки. Формат обращения:

EOLN<ф.п.>

Если параметр <ф.п.> опущен, функция проверяет стандартный файл INPUT.

Логическая функция EOF возвращает TRUE, если во входном текстовом файле достигнут маркер конца файла. Формат обращения:

EOF<ф.п.>

Пример 135. В следующем примере, иллюстрирующем работу с текстовым файлом, подсчитывается общее количество символов в файле:

```
uses crt;
var
        f : text;
        s: String;
const
        Sum: LongInt = 0; {Здесь будет количество символов}
begin
        write('Имя файла: ');{Запрашиваем...}
        readln(s); {и вводим имя файла.}
        assign(f,s);
        reset (f); {Открываем файл}
        while not EOF(f) do {Подсчитываем...}
                begin {количество. . .}
                        readln(f,s); {символов...}
                        inc(Sum, length(s)) {в файле}
                end ;
        close(f); {Закрываем файл}
        writeln('Объем = ', Sum:6,' символов.');
        readln
end.
```

Пример 136. А теперь попробуем наоборот записать в файл введенный пользователем текст.

```
uses crt;
var
f : text;
s: String;
begin
        clrscr;
        writeln('Введите текст');
        readln(s);
        assign(f,'text.txt');
        rewrite(f);
```

```
        writeln(f,s);
        close(f);
        writeln('Ваш текст в файл записан, проверьте');
        readln
end.
```

Пример 137. Прочитайте внимательно текст программы и определите всю последовательность ее работы, после чего запустите программу на исполнение и проверьте правильность своих рассуждений.

```
uses crt;
var
f : text;
m : array [1..20] of real;
i : integer;
n,s:real;
begin
        clrscr;
        assign(f, 'myfile.txt') ;
        rewrite(f);
        writeln('Введите в строку дробные числа');
        while not eoln do
        begin
           read(n);
           writeln(f,n);
        end;
        close(f);
        writeln;
        {$I-}
        if ioresult<>0 then
        writeln('Такого файла нет')
        else
         reset(f);
```

```
        {$I+}
        i := 1;
        s:=0;
        while not EOF(f) do
        begin
                readln(f ,m[i] );
                if int(m[i])<>m[i] then
                s:=s+m[i];
                i:=i+1;
        end;
        close(f);
        writeln('s=',s:5:2);
        readln;
        readln
end.
```

Пример 138. Прочитайте внимательно текст программы и определите всю последовательность ее работы, после чего запустите программу на исполнение и проверьте правильность своих рассуждений.

```
uses crt;
var
    f : text;
    s,a: String;
    sum: integer;
begin
        clrscr;
        write('Имя файла: ');
        readln(s);
        assign(f,s);
        {$I-}
        if ioresult<>0
        then writeln('Ошибка')
        else reset(f);
        {$I+}
        sum:=0;
        while not EOF(f) do
```

```
        begin
                readln(f,a);
                if length(a)>10
                then sum:=sum+1;
        end ;
        close(f);
        writeln('sum= ', sum) ;
        readln;
end.
```

Пример 139. В следующей программе создается список и записывается в файл. Имеется возможность вывести на экран список из файла и добавить запись в файл. Исполните данную программу, следя за работой каждого оператора.

```
program files;
uses crt;
type
      buro=record
         address: string[20];
         type_dom:string[10];
         etage:integer;
         chislo_kom:integer;
         cena:real;
       end;
var
     f:text;
     k,I,v:integer;
     a,b:buro;
procedure sozd;
begin
        assign(f,'c:\buro.txt');
        rewrite(f);
```

```
        writeln('Введите количество домов');
        readln(k);
        for i:=1 to k do
          begin
                writeln('Введите адрес');
                readln(a.address);
                writeln('Введите тип дома');
                readln(a.type_dom);
                writeln('Введите этаж');
                readln(a.etage);
                writeln('Введите число комнат');
                readln(a.chislo_kom);
                writeln('Введите цену');
                readln(a.cena);
                writeln(f,a.address);
                writeln(f,a.type_dom);
                writeln(f,a.etage);
                writeln(f,a.chislo_kom);
                writeln(f,a.cena);
          end;
          close(f);
end;

procedure print;
begin
        assign(f,'c:\buro.txt');
        reset(f);
        i:=1;
        writeln('номер  адрес    тип дома    этаж  число комнат
        цена');
        while not eof(f) do
          begin
                readln(f,a.address);
                readln(f,a.type_dom);
                readln(f,a.etage);
                readln(f,a.chislo_kom);
```

```
                readln(f,a.cena);
                writeln(i:4,a.address:10,a.type_dom:12,a.etage:10,a.ch
                islo_kom:10,a.cena:8);
                i:=i+1;
            end;
            close(f);
            readln;
end;

procedure dob;
begin
        assign(f,'c:\buro.txt');
        append(f);
        writeln('Введите адрес');
        readln(b.address);
        writeln('Введите тип дома');
        readln(b.type_dom);
        writeln('Введите этаж');
        readln(b.etage);
        writeln('Введите число комнат');
        readln(b.chislo_kom);
        writeln('Введите цену');
        readln(b.cena);
        writeln(f,b.address);
        writeln(f,b.type_dom);
        writeln(f,b.etage);
        writeln(f,b.chislo_kom);
        writeln(f,b.cena);
        close(f);
        writeln('Операция завершена');
end;
```

```
begin
    clrscr;
    v:=0;
    while v<>4 do
    begin
        writeln('Меню программы');
        writeln('1-создать новый файл');
        writeln('2-просмотреть существующий файл');
        writeln('3-добавить запись в файл');
        writeln('4-выйти из программы');
        writeln('Ваш выбор');
        readln(v);
        case v of
            1:sozd;
            2:print;
            3:dob;
        end;
    end;
end.
```

5.3. Типизированные файлы

Длина любого компонента типизированного файла строго постоянна, что дает возможность организовать прямой доступ к каждому из них (т.е. доступ к компоненту по его порядковому номеру).

Перед первым обращением к процедурам ввода-вывода указатель файла стоит в его начале и указывает на первый компонент с номером 0. После каждого чтения или записи указатель сдвигается к следующему компоненту файла. Переменные в списках ввода-вывода должны иметь тот же тип, что и компоненты файла. Если этих переменных в списке несколько, указатель будет смещаться после каждой операции обмена данными между переменными и дисковым файлом.

Процедура READ. Обеспечивает чтение очередных компонентов типизированного файла. Формат обращения:

READ (<ф.п.>,<сп.ввода>)

Здесь <сп.ввода> - список ввода, содержащий одну или более переменных такого же типа, что и компоненты файла.

Файловая переменная <ф.п.> должна быть объявлена предложением FILE OF... и связана с именем файла процедурой ASSIGN. Файл необходимо открыть процедурой RESET. Если файл исчерпан, обращение к READ вызовет ошибку ввода-вывода.

Процедура WRITE**.** Используется для записи данных в типизированный файл. Формат обращения:

WRITE (<ф.п.>,<сп.вывода>)

Здесь <сп.вывода> - список вывода, содержащий одно или более выражений того же типа, что и компоненты файла.

Процедура SEEK. Смещает указатель файла к требуемому компоненту. Формат обращения:

SEEK (<ф.п.>,<N компонента>)

Здесь <N компонента> - выражение типа LONGINT, указывающее номер компонента файла.

Первый компонент файла имеет номер 0. Процедуру нельзя применять к текстовым файлам.

Функция FILESIZE. Возвращает значение типа LONGINT, которое содержит количество компонентов файла. Формат обращения:

FILESIZE (<ф.п.>)

Функцию нельзя использовать для текстовых файлов. Чтобы переместить указатель в конец типизированного файла, можно написать:

seek (FileVar, FileSize(FileVar));

где FILEVAR - файловая переменная.

Функция FILEPOS. Возвращает значение типа LONGINT, содержащее порядковый номер компонента файла,

который будет обрабатываться следующей операцией ввода-вывода. Формат обращения:

FILEPOS (<ф.п.>)

Функцию нельзя использовать для текстовых файлов. Первый компонент файла имеет порядковый номер 0.

Процедура TRUNCATE. Отсекает часть файла, расположенную ниже указателя. Формат обращения:

TRUNCATE (<ф.п.>)

Первый компонент файла имеет номер 0. Процедуру нельзя применять к текстовым файлам.

Пример 140. Записать в типизированный файл текст и вывести его на экран.

```
program f;
uses crt;
var
        f1: file of string;
        s, a: string;
begin
        clrscr;
        writeln (‘Введите текст’);
        readln (a);
        assign (f1, ‘c:\45.dat’);
        rewrite (f1);
        write (f1, a);
        close (f1);
        assign (f1, ‘c:\45.dat’);
        reset (f1);
        while not eof (f1) do
                read (f1, s);
        writeln (‘Текст из файла: ’, s);
        close (f1);
        readln;
end.
```

С типизированными файлами очень легко можно произвести такие действия как добавление, удаление, изменение

записи, произвести расчеты. Привожу алгоритмы каждого из перечисленных действий.

Раздел описания: Тип запись, файловая переменная типа запись, две переменные типа запись

Процедура создания списка:

- ✓ Связываем файловую переменную с именем файла на диске. Открываем файл для записи.
- ✓ Запрашиваем у пользователя количество записей.
- ✓ В цикле вводим поля записи в первую переменную типа запись и записываем запись в файл.
- ✓ Закрываем файл.

Процедура печати:

- ✓ Связываем файловую переменную с именем файла на диске. Открываем файл для чтения.
- ✓ Выводим шапку таблицы.
- ✓ Включаем счетчик.
- ✓ В цикле просмотра файла читаем запись, выводим ее на экран и наращиваем счетчик (счетчик необходим для нумерации записей).
- ✓ Закрываем файл.

Процедура добавления записи:

- ✓ Связываем файловую переменную с именем файла на диске. Открываем файл для чтения.
- ✓ Вводим поля добавляемой записи во вторую переменную типа запись.
- ✓ Устанавливаем указатель файла в конец файла.
- ✓ Записываем в файл добавляемую запись.
- ✓ Закрываем файл.

Процедура удаления записи:

- ✓ Запрашиваем номер удаляемой записи.
- ✓ Связываем файловую переменную с именем файла на диске. Открываем файл для чтения.

- ✓ Устанавливаем указатель перед последней записью.
- ✓ Читаем запись в переменную типа запись.
- ✓ Устанавливаем указатель перед удаляемой записью.
- ✓ Записываем в файл запись, хранящуюся в переменной типа запись.
- ✓ Устанавливаем указатель перед последней записью.
- ✓ Отсекаем часть файла ниже указателя.

Процедура изменения записи:

- ✓ Запрашиваем номер изменяемой записи.
- ✓ Связываем файловую переменную с именем файла на диске. Открываем файл для чтения.
- ✓ Устанавливаем указатель перед изменяемой записью.
- ✓ Читаем запись в переменную типа запись.
- ✓ Спрашиваем пользователя, желает ли он изменить первое поле записи. Если ответ положительный, просим пользователя ввести значение данного поля и записываем это значение в переменную типа запись.
- ✓ Спрашиваем пользователя, желает ли он изменить второе поле записи. Если ответ положительный, просим пользователя ввести значение данного поля и записываем это значение в переменную типа запись.
- ✓ и т.д. до конца
- ✓ Устанавливаем указатель перед изменяемой записью и записываем в файл запись, хранящуюся в переменной типа запись.

Процедура расчетов

- ✓ Просматирваем в цикле каждую запись файла (точно также как в процедуре печати;
- ✓ В том же цикле ставим нужные условия и производим необходимые расчеты.

Пример 141. Создать список «Бюро недвижимости» и записать его в типизированный файл. Распечатать список из файла, добавить запись в список, удалить запись из списка, изменить запись, найти квартиру, удовлетворяющую критерию пользователя с необходимым количеством квартир и этажом

```
program files;
uses crt;
type
      buro=record
         address: string[20];
         type_dom:string[10];
         etage:integer;
         chislo_kom:integer;
         cena:real;
         metka1,metka2:integer;
       end;
var
     f:file of buro;
     m,l,n,k,i,v,r:integer;
     min:real;
     a,b:buro;
     s:char;

procedure sozd;
begin
        assign(f,'c:\buro.dat');
        rewrite(f);
        writeln('Введите количество домов');
        readln(k);
        for i:=1 to k do
          begin
                writeln('Введите адрес');
                readln(a.address);
                writeln('Введите тип дома');
                readln(a.type_dom);
                writeln('Введите этаж');
                readln(a.etage);
```

```
                writeln('Введите число комнат');
                readln(a.chislo_kom);
                writeln('Введите цену');
                readln(a.cena);
                a.metka1:=0;
                write(f,a);
                writeln;
          end;
          close(f);
          writeln;
end;

procedure print;
begin
        assign(f,'c:\buro.dat');
        reset(f);
        i:=1;
        writeln('номер  адрес    тип дома    этаж  число комнат
        цена');
        while not eof(f) do
          begin
                read(f,a);
                writeln(i:4,a.address:10,a.type_dom:12,a.etage:10,a.ch
                islo_kom:10,a.cena:8);
                i:=i+1;
          end;
          close(f);
          readln;
end;

procedure dob;
begin
        assign(f,'c:\buro.dat');
        reset(f);
        writeln('Введите адрес');
        readln(b.address);
        writeln('Введите тип дома');
```

```
        readln(b.type_dom);
        writeln('Введите этаж');
        readln(b.etage);
        writeln('Введите число комнат');
        readln(b.chislo_kom);
        writeln('Введите цену');
        readln(b.cena);
        seek(f,filesize(f));
        write(f,b);
        close(f);
        writeln('Операция завершена');
end;

procedure izm;
begin
        writeln('Номер изменяемой записи');
        readln(n);
        assign(f,'c:\buro.dat');
        reset(f);
        seek(f,n-1);
        read(f,a);
        writeln('Изменить адрес?(y/n)');
        readln(s);
        if (s='Y') or (s='y') then
          begin
                writeln('Введите адрес');
                readln(a.address);
          end;
        writeln('Изменить тип дома? y/n');
        readln(s);
        if (s='y') or (s='Y') then
```

```
        begin
                writeln('Введите тип дома');
                readln(a.type_dom);
        end;
      writeln('Изменить этаж? y/n');
      readln(s);
      if (s='y') or (s='Y') then
        begin
                writeln('Введите этаж');
                readln(a.etage);
        end;
      writeln('Изменить число комнат? y/n');
      readln(s);
      if (s='y') or (s='Y') then
        begin
                writeln('Введите число комнат');
                readln(a.chislo_kom);
        end;
      writeln('Изменить цену? y/n');
      readln(s);
      if (s='y') or (s='Y') then
        begin
                writeln('Введите цену');
                readln(a.cena);
        end;
      seek(f,n-1);
      write(f,a);
      close(f);
end;

procedure ydal;
begin
      writeln('Номер удаляемой записи');
      readln(n);
      assign(f,'c:\buro.dat');
      reset(f);
      seek(f,filesize(f)-1);
```

```
        read(f,b);
        seek(f,n-1);
        write(f,b);
        seek(f,filesize(f)-1);
        truncate(f);
        writeln('операция завершена');
end;

procedure poisk;
begin
        writeln('Введите этаж');
        readln(m);
        writeln('Введите число комнат');
        readln(l);
        assign(f,'c:\buro.dat');
        reset(f);
        r:=1;
        seek(f,0);
        while not eof(f) do
                begin
                        read(f,a);
                        if (a.etage=m) or (a.chislo_kom=l) then
                          begin
                                r:=0;
                                writeln(a.address:10, a.type_dom:8,
                          a.etage:10, a.chislo_kom:10, a.cena:25);
                          end;
                end;
        if r<>0 then writeln('Таких квартир нет');
end;
```

```
begin
    clrscr;
    v:=0;
    while v<>7 do
    begin
        writeln('Меню программы');
        writeln('1-создать новый файл');
        writeln('2-просмотреть существующий файл');
        writeln('3-добавить запись в файл');
        writeln('4-отредактировать запись');
        writeln('5-удалить запись');
        writeln('6-поиск квартиры по этажу и числу комнат');
        writeln('7-выйти из программы');
        writeln('Ваш выбор');
        readln(v);
        case v of
            1:sozd;
            2:print;
            3:dob;
            4:izm;
            5:ydal;
            6:poisk;
        end;
    end;
end.
```

5.4. Нетипизированные файлы

Нетипизированные файлы

Нетипизированные файлы объявляются как файловые переменные типа FILE и отличаются тем, что для них не указан тип компонентов. Отсутствие типа делает эти файлы, с одной стороны, совместимыми с любыми другими файлами, а с другой – позволяет организовать высокоскоростной обмен данными между диском и памятью.

При инициации нетипизированного файла процедурами RESET или REWRITE можно указать длину записи нетипизированного файла в байтах. Например, так:

```
var
    f: file;
begin
    .......
    assign(f, 'myfile.dat')
    reset (f, 512);
    .......
end.
```

Длина записи нетипизированного файла указывается вторым параметром при обращении к процедурам RESET или REWRITE, в качестве которого может использоваться выражение типа WORD. Если длина записи не указана, она принимается равной 128 байтам.

При работе с нетипизированными файлами могут применяться все процедуры и функции, доступные типизированным файлам, за исключением READ и WRITE, которые заменяются соответственно высокоскоростными процедурами BLOCKREAD и BLOCKWRITE . Для вызова этих процедур используются следующие предложения:

BLOCKREAD (<ф .п . >, <буф>, < [, <NN>])

BLOCKWRITE (<ф . п . >, <буф>, < [, <NN>])

Здесь <буф> - буфер: имя переменной, которая будет участвовать в обмене данными с дисками;

<D> - количество записей, которые должны быть прочитаны или записаны за одно обращение к диску;

<NN> - необязательный параметр, содержащий при выходе из процедуры количество фактически обработанных записей.

После завершения процедуры указатель смещается на <NN> записей. Процедурами SEEK, FILEPOS и FILESIZE

можно обеспечить доступ к любой записи нетипизированного файла.

Пример 142. Записать в нетипизированный файл запись с полями целого и строкового типа и вывести запись из файла на экран.

```
program r;
uses crt;
type
        e=record
                x: integer;
                y: string;
        end;
var
        f1: file;
        ee, ee1: e;
begin
        clrscr;
        writeln (‘Введите текст’);
        readln (ee.y);
        writeln (‘Введите целое число’);
        readln (ee.x);
        assign (f1, ‘c:\12.dat’);
        rewrite (f1);
        blockwrite (f1, ee, 1);
        close (f1);
        assign (f1, ‘c:\12.dat’);
        reset (f1);
        while not eof (f1) do
                blockread (f1, ee1, 1);
        writeln (‘Текст из файла: ’, ee1.x, ee1.y);
        close (f1);
        readln;
end.
```

ПРАКТИЧЕСКИЕ ЗАДАНИЯ

1. *** Школе необходим файл для учета выпускников. Создайте файл для канцелярии по учету выпускников. Храните в нем фамилию, имя, год выпуска, любимый вид спорта и нынешний род занятий выпускника. Для образца составьте файл на десять человек. Воспользуйтесь этим файлом и напечатайте приглашения на очередной домашний матч "Зенита" тем выпускникам, которые назвали футбол своим любимым видом спорта.
   Смотрите пример 141.
2. *** Компьютерная фирма ведет файл со сведениями о двадцати своих сотрудниках. Создайте файл, содержащий имя и адрес каждого сотрудника (с указанием улицы, дома, квартиры и почтового индекса). По содержимому файла напечатайте почтовые адреса для рассылки чеков еженедельной заработной платы.
   Смотрите пример 141.
3. *** Гидрометцентр ведет статистику выпадения снега по регионам, для каждого из которых заведен последовательный файл. Во всех файлах присутствуют три элемента данных: имя метеоролога, название региона, количество выпавшего за зиму снега в мм. Напишите программу ввода данных; заполните файлы для трех регионов. Просмотрите все три файла и подсчитайте средний уровень снежных осадков по трем областям. Результат выведите на экран.
   Смотрите пример 141.
4. *** Налоговая инспекция поощряет налогоплательщиков, вносящих подоходный налог до истечения апрельского контрольного срока, делая им скидку. Создайте файл, в котором содержались бы имена, сведения о сроках уплаты и размере налога для каждого налогоплательщика (ограничьтесь группой из шести человек). Пусть ваша программа читает файл и делает скидку в 10% для тех, кто

уплатил налог досрочно, а также выводит на экран их имена и размер скидки в рублях.
Смотрите пример 141.

5. *** Фабрика игрушек ведет учет фирм розничной торговли, сбывающих ее продукцию. Файл контрагентов содержит названия этих фирм, сведения об их местоположении и индекс кредитоспособности: низкая или высокая.
   a) Напишите программу, которая создала бы файл контрагентов.
   b) Напишите программу, которая создала бы два последовательных файла с именами good.dat и bad.dat соответственно для фирм с высокой и низкой кредитоспособностью.
   c) Пусть ваша программа спрашивает у бухгалтера, какой из двух списков ему представить, а затем выдает названия фирм и их местоположение из соответствующего файла.
   Смотрите пример 141.
6. *** Инспектор колледжа ведет файл академических занятий студентов. Создайте файл и заполните его фамилиями, названиями академических курсов и оценочным коэффициентом студентов. Выберите "умных" студентов, т. е. тех, кто имеет оценку выше 88, и запишите сведения о них в файл best.dat. Пусть программа помогает инспектору формировать на основе этого файла группы углубленного обучения. По названию курса она должна выдавать список "умных" студентов, зачисленных в такую группу.
   Смотрите пример 141.
7. В следующих задачах предусмотреть

- Создание файла.
- Просмотр файла.
- Добавление записи в файл.
- Удаление записи из файла.
- Редактирование записи в файле.

1) Дан файл «Улицы города», содержащий название улицы, № дома, количество этажей, количество квартир. Найти самый многоквартирного и самый многоэтажный дом.

2) Дан файл «Аэрофлот», содержащий номер рейса, пункт назначения, время вылета, время прибытия, количество свободных мест в салоне. Выдать справку по запросу о времени отправления самолетов в город X и о наличии свободных мест на рейс в город X со временем отправления У. Значения X и У вводятся в режиме диалога.
3) Дан файл «Аптеки города», содержащий № аптеки, ее адрес, название лекарства, цену на это лекарство. Выдать справку по запросу о наличии лекарства X в аптеках города. Справка должна содержать номера и адреса аптек, в которых есть это лекарство и цены на него. Значения X вводится в режиме диалога.
4) Дан файл «Нефтепродукты», содержащий название АЗС, адрес АЗС, цены на различные виды топлива (если топлива нет, то - 0). Выдать справку, которая должна содержать все наименования АЗС, их адрес, где цены на топливо не больше, чем введенная в режиме диалога.
5) Дан файл «Больница», содержащий фамилию пациента, пол, возраст, диагноз, отделение. Найти всех пациентов старше X лет с диагнозом У по заданному отделению. X, У, наименование отделения вводятся в режиме диалога.
6) Дан файл «Отдел кадров», содержащий день, месяц, год рождения, Ф.И.О. сотрудника, адрес. Выдать справку о всех сотрудниках, которые празднуют в этом году юбилей и какой (например, 25, 30, 40, 60 и т.д.).
7) Дан файл «Отдел кадров», содержащий Ф.И.О. сотрудника, адрес, стаж, количество детей. Выдать справку о всех сотрудниках, у которых количество детей больше 3 и тех, у которых самые маленькие по возрасту дети.
8) Дан файл «Игрушки», содержащий адрес магазина, название игрушки, ее цена, производитель, возрастные границы

9) (например, от 1 до 3, от 3 до 5 и т.д.). Выдать все магазины, в которых есть игрушки данного вида для ребенка данного возраста в порядке возрастания цены. Возраст ребенка и название игрушки вводятся в режиме диалога
Смотрите пример 141.

8. * Выполнить все примеры задач из данного раздела.
9. ** Создать текстовый файл. Записать в него элементы массива случайных чисел. Просмотреть файл, найти в нем все отрицательные числа и вывести их на экран.
10. ** Создать нетипизированный файл. Записать в него текст. Просмотреть данный файл и определить число символов в файле.
11. ** Компонентами файла являются действительные числа. Найти:
    a) Сумму компонент;
    b) Произведение компонент;
    c) Сумму квадратов компонент;
    d) Модуль суммы и квадрат произведения;
    e) Последнюю компоненту файла.
12. ** Компонентами файла являются действительные числа. Найти:
    a) наибольшее из значений компонент;
    b) сумму наибольшего и наименьшего значений компонент;
    c) разность первой и последней компонент.
13. ** Компонентами файла являются натуральные числа. Найти:
    a) Количество четных компонент;
    b) Количество квадратов нечетных компонент.

ТЕСТОВЫЕ ЗАДАНИЯ ДЛЯ ПРОВЕРКИ ТЕОРЕТИЧЕСКИХ ЗНАНИЙ

1. **Под файлом понимается**
   A) последовательность бит информации для хранения и считывания;
   B) именованная область внешней памяти компьютера либо логическое устройство – потенциальный источник информации; *
   C) именованная область внешней памяти компьютера для хранения и обработки данных;
   D) именованная область внутренней памяти компьютера либо логическое устройство – потенциальный источник информации;
   E) именованная область внешней или внутренней памяти компьютера либо логическое устройство – потенциальный источник информации.
2. **Можно выделить следующие виды файлов:**
   A) типизированные, внешние, внутренние;
   B) внешние, внутренние, текстовые;
   C) текстовые, типизированные, внешние;
   D) нетипизированные, типизированные, текстовые; *
   E) текстовые, внутренние, внешние.
3. **В примере представлено описание файловых переменных (f1-f5) следующих типов (перечислить соответственно):**

```
type
        product = record
                name: string;
                code: word;
                cost: comp
        end;
text80=file of string[80];
var
        f1: file of char;
        f2: text;
```

f3: file;
f4: text80;
f5: file of product;

A) текстовый, текстовый, нетипизированный, типизированный, нетипизированный
B) типизированный, текстовый, нетипизированный, типизированный, типизированный; *
C) типизированный, типизированный, нетипизированный, текстовый, типизированный;
D) нетипизированный, текстовый, нетипизированный, типизированный, типизированный;
E) нетипизированный, текстовый, нетипизированный, текстовый, типизированный.

4. С помощью процедуры assign

A) файловая переменная связывается с именем файла на диске; *
B) файловая переменная задает путь к файлу;
C) задается путь к файлу;
D) файл f связывается с адресом дискового пространства;
E) файл f связывается с именем файла на диске.

5. Стандартная процедура, которая инициирует запись информации в файл, связанный ранее с файловой переменной. С помощью нее нельзя инициировать запись информации в ранее существующий файл, так как он уничтожится.

A) reset;
B) rewrite; *
C) assign;
D) ioresult;
E) erase.

6. С помощью следующей последовательности операторов осуществляется:

```
type
    e=record
        x:integer;
        y:string;
```

```
        end;
var
        f1:file;
        ee:e;
begin
    readln(ee.y);
    readln(ee.x);
    assign(f1,'c:\12.dat');
    rewrite(f1);
    blockwrite('f1,ee,1');
    close(f1);
end;
```

A) запись в типизированный файл записи с полями целого и строкового типов;
B) считывание из типизированного файла записи с полями целого и строкового типов;
C) запись в нетипизированный файл записи с полями целого и строкового типов; *
D) считывание из нетипизированного файла записи с полями целого и строкового типов;
E) считывание из нетипизированного файла переменных целого и строкового типов.

7. С помощью следующей последовательности операторов осуществляется:

```
program f1;
var
    f1: text;
    s: real;
begin
    assign (f1, 'c:\text.txt');
    reset (f1);
    while not eof(f1) do
```

```
    begin
        readln (f1, s);
        writeln(s);
    end;
    close(f1)
end.
```

A) запись в текстовый файл действительных чисел;
B) считывание из текстового файла действительных чисел и вывод их на экран; *
C) запись в текстовый файл текста, вводимого пользователем;
D) считывание из текстового файла текста и вывод его на экран;
E) считывание из текстового файла целых чисел и вывод их на экран.

8. С помощью следующей последовательности операторов осуществляется:

```
write('Имя файла?');
readln(s);
assign(f, s);
{$I-}
reset(f);
{$I+}
if ioresult<>0 then
writeln('***');
```

A) считывание из файла текста;
B) считывание из файла целых чисел и вывод их на экран.
C) инициация файла, и если файл не существует, выдается ошибка ввода-вывода;
D) инициация файла, и если файл не существует, выдается сообщение об этом; *
E) файл открывается для считывания.

9. С помощью следующей последовательности операторов осуществляется:

```
assign(f, text.txt);
reset(f);
```

```
while not eof(f) do
        begin
                readln(f, s);
                inc(sum, length(s))
end;
close(f);
```

A) запись в текстовый файл каких-либо данных;
B) считывание данных из текстового файла;
C) запись в текстовый файл текста, вводимого пользователем;
D) подсчет суммы чисел, записанных в текстовый файл;
E) подсчет количества символов в текстовом файле. *

10. С помощью какой процедуры файл можно открыть для записи и очистить?

A) Erase(f);
B) Rename(f,s);
C) Reset(f);
D) Eof(f);
E) Rewrite(f). *

11. Как называются файлы, состоящие из компонентов одного типа, число которых заранее не определено и может быть любым?

A) Логические;
B) Интегрированные;
C) Текстовые;
D) Типизированные; *
E) Динамические.

12. Для чего используется процедура assign(f,s)?

A) Для удаления файла;
B) Для открытия файла;
C) Для установления связи между файловой переменной и именем файла на диске; *

D) Для закрытия файла;
E) Для переименования файла.

13. При выполнении какой операции необходимо выполнить следующее действие

1. описать файловую переменную;
2. связать ее с физическим файлом;
3. открыть файл для записи;
4. внести необходимую информацию в файл;
5. обязательно закрыть файл.

A) при записи в файл значений переменных; *
B) для чтения информации из файла;
C) при проверке, достигнут ли конец файла;
D) при создании файла;
E) при сохранении файла.

14. Какая процедура служит для удаления файла?

A) filesize(f)
B) write(f)
C) erase(f) *
D) filepos(f)
E) eof(f)

15. Файл – это

A) Это контейнер, в котором могут содержаться другие объекты;
B) Это упорядоченный набор величин, обозначаемых одним именем;
C) Это набор взаимосвязанных данных, воспринимаемых компьютером как единое целое ; *
D) Это определенная и отдельным образом оформленная часть информации, т.е. один текст, одна программа, одна картинка или, иначе говоря, любой набор данных;
E) Переменная, которая в качестве своего значения содержит адрес байта памяти.

16. Для чего используется процедура close(f)?

A) Для чтения информации из файла;
B) Для закрытия файла; *
C) Для удаления файла;

D) Для установления связи файловой переменной и именем файла на диске;
E) Для записи информации в файл.

17. С помощью какого служебного слова описывается текстовый файл?

A) Text; *
B) File;
C) String;
D) Array;
E) Record.

18. Каков результат фрагмента программы?

```
assign(f, ‘a:\number.txt’);
reset(f);
while not eof(f) do
begin
        readln(f, n);
        writeln(n);
end;
close(f);
```

(где n-переменная типа integer)

A) Добавляется в файл 5 чисел;
B) Записывает в файл 5 целых чисел;
C) Выводит содержимое файла; *
D) Удаляет содержимое файла;
E) Вычисляет среднее арифметическое 5-ти целых чисел.

19. Каков результат фрагмента программы:

```
assign(f,’a:\number.txt’);
rewrite(f);
for i:=1 to 5 do
begin
        readln(n);
        writeln(f,n);
```

```
end;
close(f);
```

(где n-переменная типа integer)

A) Добавляется в файл 5 чисел;
B) Записывает в файл 5 целых чисел; *
C) Выводит содержимое файла;
D) Удаляет содержимое файла;
E) Вычисляет среднее арифметическое 5-ти целых чисел.

20. Какая процедура служит для переименования файла?

A) Eof(f);
B) Erase(f);
C) Write(f);
D) Read(f);
E) Rename(f,s). *

21. Какая процедура отсекает часть открытого файла, начиная с текущего компонента, и подтягивает на его место конец файла?

A) Filesize(f);
B) Seek(f);
C) Filepos(f);
D) Truncate(f);
E) Append(f).

22. В каких случаях логическая функция eof (имя файла) принимает значение “true” (истина)?

A) При переводе указателя на запись с указанным номером;
B) В любых случаях;
C) Если при работе с файлом достигнут его конец; *
D) При возвращении значения типа longint;
E) Если при работе с файлом не достигнут его конец.

23. Каков результат фрагмента программы?

```
sum:=0;
kol:=0;
assign(f,’a:\number.txt’);
reset(f);
while not eof(f) do
begin
```

```
        readln(f,n);
        sum:=sum+n;
        kol:=kol+1;
    end;
    close(f);
    sa:=sum/kol;
    writeln(sa);
```

(где n-переменная типа integer)

A) Добавляется в файл 5 чисел;
B) Выводит на экран натуральные числа от 1 до 5 в порядке возрастания;
C) Находит сумму натуральных чисел от 1 до 5;
D) Записывает в файл 5 целых чисел;
E) Вычисляет среднее арифметическое 5-ти целых чисел.*

24. Какая процедура служит для удаления файла?

A) Erase(f); *
B) Filesize(f);
C) Filepos(f);
D) Eof(f);
E) Write(f).

25. Какая процедура используется, чтобы открыть файл для чтения?

A) Open(имя файла);
B) Eof(имя файла);
C) Reset(имя файла); *
D) Rewrite(имя файла);
E) Close(имя файла).

26. Как описываются типизированные файлы?

A) var <имя записи>:<тип записи>
B) var <имя>:file of <тип> *
C) type <имя типа>: file of <тип>
D) var <имя> (<список параметров>)

E) type <имя типа>=array (<тип индекса>)

27. Какие процедуры являются основными процедурами управления файлами?

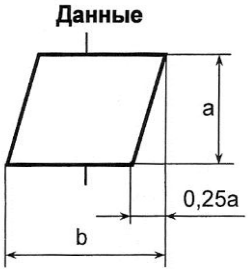
A) erase, clrscr, getdate, rewrite;
B) reset, rewrite, seek, eof; *
C) reset, delay, seek, delete;
D) rectangle, eof, rename, seek;
E) val, close, assign, sound.

28. Какая процедура возвращает значение типа longint – текущую позицию в файле f, который должен быть открыт.

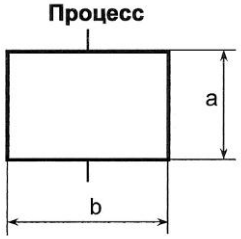
A) Append(f);
B) Filepos(f); *
C) Filesize(f);
D) Seek(f);
E) Truncate(f).

ПРИЛОЖЕНИЯ

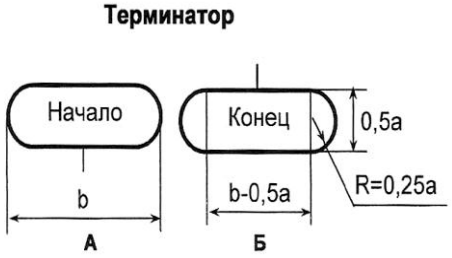
Приложение А. Графическое представление алгоритма



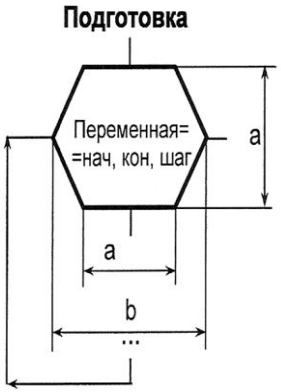
Символ отображает данные, когда носитель данных не определен



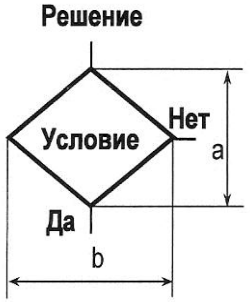
Символ отображает функцию обработки данных любого типа



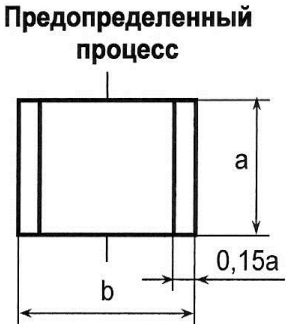
Символ отображает выход во внешнюю среду и вход из внешней среды (начало или конец схемы программы, внешнее использование и источник или пункт назначения данных).



Символ отображает модификацию команды или группы команд с целью воздействия на некоторую последующую функцию (установка переключателя, модификация индексного регистра).



Символ отображает решение или функцию переключательного типа, имеющую один вход и ряд альтернативных выходов, один из которых может быть активизирован после вычисления условий, определенных внутри этого символа.

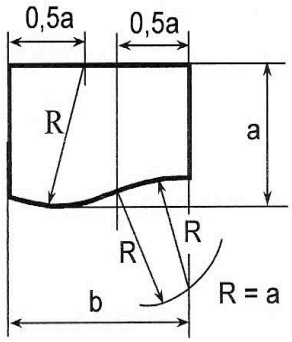


Символ отображает предопределенный процесс, состоящий из одной или нескольких операций или шагов программы, которые определены в другом месте

Символ, состоящий из двух частей, отображает начало и конец цикла. Обе части символа имеют один и тот же идентификатор. Условия для инициализации, приращения, завершения и т.д. помещают внутри символа в начале или в конце в зависимости от расположения операции, проверяющей условие.

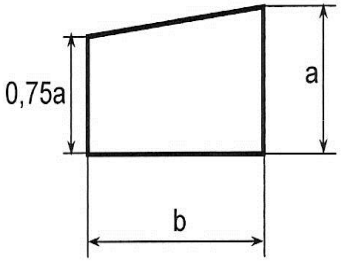
А – цикл с предусловием;
Б – цикл с постусловием.

Документ



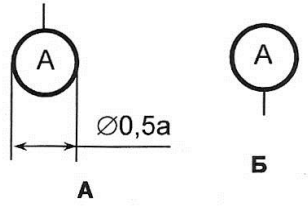
Символ отображает данные, представленные на носителе в удобочитаемой форме (машинограмма, документ для оптического и магнитного считывания, микрофильм, бланки ввода данных, рулон ленты с итоговыми данными).

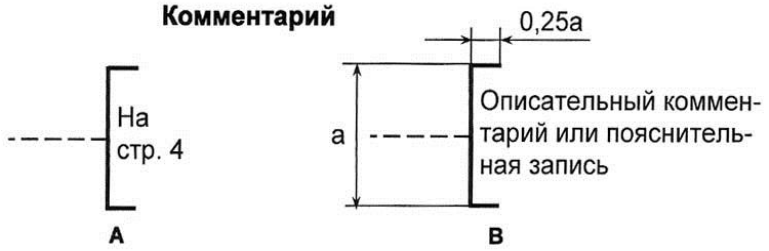
Ручной ввод



Символ отображает данные, вводимые вручную во время обработки с устройств любого типа (клавиатура, переключатели, кнопки, световое перо, полоски со штриховым кодом)

Соединитель





Символ используют для добавления описательных комментариев или пояснительных записей в целях объяснения или примечаний. Пунктирные линии в символе комментария связаны с соответствующим символом или могут обводить группу символов. Текст комментариев должен быть помещен около ограничивающей фигуры.

Линия

Символ отображает поток данных или управления. При необходимости или для повышения удобочитаемости могут быть добавлены стрелки-указатели.

Стрелками указываются отрицательные направления линий потока (справа налево и снизу вверх)

Символ отображает синхронизацию двух или более параллельных операций

Приложение Б. Таблицы кодов

Таблица №1

Символы ASCII с кодами 0..127

Сим-вол	код	Описание	Сим-вол	код	Сим-вол	код
1	2	3	4	5	6	7
	0	Null. Отображает как "пробел"	,	44	X	88
☺	1		-	45	Y	89
☻	2		.	46	Z	90
♥	3		/	47	[	91
♦	4	Конец передачи (Ctrl+D)	0	48	\	92
♣	5		1	49	]	93
♠	6		2	50	^	94
•	7	Звонок.	3	51	_	95
◘	8	**BackSpace (клавиша "←")**	4	52	`	96
○	9	**Табуляция (клавиша "TAB")**	5	53	a	97
◙	10	Перевод строки.	6	54	b	98
♂	11		7	55	c	99
♀	12		8	56	d	100
♪	13	Возврат каретки.	9	57	e	101
♫	14		:	58	f	102
☼	15		;	59	g	103
►	16		<	60	h	104
◄	17		=	61	I	105
↕	18		>	62	j	106
‼	19		?	63	k	107
¶	20		@	64	l	108
§	21		A	65	m	109
▬	22		B	66	n	110
↨	23		C	67	o	111
↑	24		D	68	p	112
↓	25		E	69	q	113
→	26		F	70	r	114
←	27	**Клавиша "ESC".**	G	71	s	115
∟	28		H	72	t	116
↔	29		I	73	u	117
▲	30		J	74	v	118
▼	31		K	75	w	119
	32	**Клавиша "Пробел".**	L	76	x	120
!	33		M	77	y	121
"	34		N	78	z	122
#	35		O	79	{	123

1	2	3	4	5	6	7
$	36		P	80	¦	124
%	37		Q	81	}	125
&	38		R	82	~	126
'	39		S	83	⌂	127
(	40		T	84		
)	41		U	85		
*	42		V	86		
+	43		W	87		

Таблица №2

Символы ASCII кодировки IBM cp866 с кодами 128..255

Символ	код	Символ	код	Символ	код
1	2	3	4	5	6
А	128		192	м	172
Б	129		193	н	173
В	130		194	о	174
Г	131		195	п	175
Д	132		196		176
Е	133		197		177
Ж	134		198		178
З	135		199		179
И	136		200		180
Й	137		201		181
К	138		202		182
Л	139		203		183
М	140		204		184
Н	141		205		185
О	142		206		186
П	143		207		187
Р	144		208		188
С	145		209		189
Т	146		210		190
У	147		211		191
Ф	148		212	ь	236
Х	149		213	э	237
Ц	150		214	ю	238
Ч	151		215	я	239
Ш	152		216	Ё	240
Щ	153		217	ё	241
Ъ	154		218	є	242
Ы	155		219	Є	243
1	2	3	4	5	6
Ь	156		220	Ї	244
Э	157		221	ї	245

1	2	3	4	5	6
Ю	158	▌	222	Ŷ	246
Я	159	▀	223	ŷ	247
а	160	р	224	°	248
б	161	с	225	·	249
в	162	т	226	·	250
г	163	у	227	√	251
д	164	ф	228	№	252
е	165	х	229	¤	253
ж	166	ц	230	■	254
з	167	ч	231		255
и	168	ш	232		
й	169	щ	233		
к	170	ъ	234		
л	171	ы	235		

Приложение В. Модуль SISTEM

Модуль *SYSTEM* является основной библиотекой Турбо Паскаля. Он реализует подпрограммы для всех встроенных возможностей, таких как ввод/вывод, обработка строк, эмуляция арифметического сопроцессора, управление оверлеями и динамическое распределение памяти. Модуль SYSTEM используется автоматически любым модулем или программой и никогда не указывается в предложении USES.

Стандартные процедуры и функции

Процедуры управления

Procedure Break Обеспечивает немедленный выход из операторов повторения.

Procedure Continue Завершает очередной итерационный цикл операторов повторения.

Procedure Exit Позволяет немедленно выйти из текущей подпрограммы. При вызове из тела основной программы завершает ее работу.

Procedure Halt [(ExitCode: Word)] Останавливает выполнение программы и возвращает управление в операционную систему. Необязательный параметр ExitCode определяет код завершения программы.

Procedure RunError [(ErrorCode: Byte)] Останавливает выполнение программы и генерирует ошибку периода выполнения программы. Необязательный параметр ErrorCode определяет код ошибки.

Функции преобразования

Процедуры Pack и UnPack, определенные в стандартном Паскале, в Турбо Паскале не реализованы. Function Chr(X: Byte): Char Возвращает символ с заданным порядковым номером X.

Function Ord(X) : LongInt Возвращает порядковый номер, соответствующий значению X порядкового типа.

Function Round (R: Real) : LongInt Округляет значение R вещественного типа до ближайшего целого.

Function Trunc(R: Real): LpngInt Усекает значение вещественного типа до значения типа LongInt путем отбрасывания дробной части.

Арифметические функции

При компиляции в режиме использования сопроцессора или его эмуляции арифметические функции возвращают значение типа EXTENDED, в противном случае - типа REAL.

Function Abs(R; Real): Real Возвращает абсолютное значение аргумента.

Function ArcTan(R: Real): Real Возвращает арктангенс аргумента.

Function Cos(R: Real): Real Возвращает косинус аргумента.

Function Exp(R; Real): Real Возвращает экспоненту аргумента.

Function Frac(R; Real): Real Возвращает дробную часть аргумента.

Function Int(R; Real): Real Возвращает целую часть аргумента.

Function Ln(R: Real) : Real Возвращает натуральный логарифм аргумента.

Function Pi: Real Возвращает значение числа pi=3.1415926535897932385.

Function Sin(R: Real): Real Возвращает синус аргумента.

Function Sqr(R: Real): Real Возвращает аргумент в квадрате.

Function Sqrt(R; Real): Real Возвращает квадратный корень аргумента.

Процедуры порядкового типа

Procedure Dec (var X [; DX: LongInt]) Уменьшает значение переменной X на величину DX, а если параметр DX не задан - на 1.

Procedure Inc (var X [; DX: LongInt]) Увеличивает значение переменной X на величину DX, а если параметр DX не задан - на 1.

Функции порядкового типа

Function Odd(X) : Boolean Проверяет, является ли аргумент нечетным числом.

Function Pred(X) Возвращает предшествующее значение аргумента. Тип результата совпадает с типом аргумента.

Function Succ(X) Возвращает последующее значение аргумента. Тип результата совпадает с типом аргумента.

Строковые процедуры

Procedure Delete (var S: String; Index, Count: Integer) Удаляет Count символов из строки S, начиная с позиции Index.

Procedure Insert (SubS: String; var S: String; Index: Integer) Вставляет подстроку SubS в строку 5, начиная с позиции Index.

Procedure Str(X [: width [: Decimals]]; var S: String) Преобразует численное значение X в его строковое представление S.

Procedure Val(S; String; var X; var Code: Integer) Преобразует строковое значение S в его численное представление X. Параметр Code -содержит признак ошибки преобразования (0 - нет ошибки).

Строковые функции

Function Concat(S1 [, S2,...,SN]): String Выполняет конкатенацию последовательности строк.

Function Copy(S: String; Index, Count: Integer): String Возвращает подстроку из строки S, начиная с позиции Index и длиной Count символов.

Function Length(S: String): Byte Возвращает текущую длину строки S.

Function Pos(SubS, S: String): Byte Возвращает позицию, начиная с которой в строке S располагается подстрока SubS (О - S не содержит SubS).

Процедуры разного назначения

Procedure Exclude (var S: set of T; I: T) Исключает элемент T из множества S.

Procedure Include (var S: set of T; I; T) Включает элемент T во множество 5.

Procedure Randomize Инициализирует случайным значением (текущим системным временем) встроенный генератор псевдослучайных чисел.

Функции разного назначения

Function Random [(Range: Word)] Возвращает псевдослучайное число. Если параметр Range опущен, функция возвращает вещественное число в диапазоне от 0 до 1, если указан - целое число в диапазоне от 0 до Range-1.

Function SizeOf(X): Word Возвращает число байт, занимаемых аргументом.

Function UpCase(C: char): Char Преобразует латинскую букву в заглавную.

Процедуры ввода/вывода

Procedure Assign (var F; Name: String) Связывает внешний файл Name с файловой переменной F.

Procedure ChDir(S: String) Устанавливает текущий каталог.

Procedure Close (var F) Закрывает открытый файл.

Procedure Erase (var F) Удаляет внешний файл.

Procedure GetDir(D: Byte; var S: String) Возвращает каталог по умолчанию S на заданном диске D.

Procedure MkDirtS: String) Создает подкаталог S.

Procedure Rename (var F) Переименовывает внешний файл.

Procedure Reset (var F) Открывает существующий файл для чтения или изменения.

Procedure Rewrite (var F) Создает и открывает новый файл.

Procedure RmDir(S: String) Удаляет пустой подкаталог.

Procedure Seek (var F; N: LongInt) Устанавливает текущую позицию файла на указанный элемент (не используется с текстовыми файлами).

Procedure Truncate (var F) Усекает размер файла до текущей позиции в файле (не используется с текстовыми файлами)

Функции ввода/вывода

Function EOF (var F) ; Boolean Возвращает для файла F признак конца файла.

Function FilePos (var F) **:** LongInt Возвращает текущую позицию в файле (не используется с текстовыми файлами)

Function FileSize(var F) : LongInt Возвращает текущий размер файла (не используется с текстовыми файлами).

Function IQResult; Integer Возвращает целое значение, являющееся состоянием последней выполненной операции ввода/вывода.

Процедуры для текстовых файлов

Procedure Append (var F: Text) Открывает существующий файл для расширения.

Procedure Flush (var F: Text) Выталкивает буфер файла вывода.

Procedure Read ([var F: Text;] V1 [, V2,...,VN]) Считывает одно или более значений из текстового файла в одну или более переменных.

Procedure Readln Выполняет те же действия, что и Read, а потом делает пропуск до начала следующей строки файла.

Procedure SetTextBuf (var F: Text; var Buf [; Size: Word]) Назначает буфер ввода/вывода для текстового файла. Параметр

Size определяет длину буфера в байтах (если Size опущен, длина буфера равна 128 байтам).

Procedure Write([var F: Text;] V1 [, V2,...,VN]) Записывает в текстовый файл одно или более значений.

Procedure WriteLn Выполняет те же действия, что и Write, а затем добавляет к файлу маркер конца строки.

Функции для текстовых файлов

Function Eolntvar F: Text): Boolean Возвращает признак конца строки.

Function SeekEof [(var F: Text)]: Boolean Возвращает признак конца файла. Предварительно пропускает все пробелы, символы табуляции и признаки конца строк.

Function SeekEoln [(var F: Text)]: Boolean Возвращает признак конца строки. В отличие от Eoln предварительно пропускает все пробелы и символы табуляции.

Процедуры для нетипизированных файлов

Procedure BlockRead(var F: File; var Buf; Count; Word [;var Result; Word]) Считывает в переменную Buf Count записей из файла F. Необязательный параметр Result содержит истинное количество считанных записей.

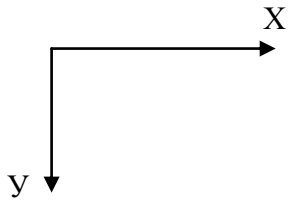
Procedure BlockWrite(var F; File; var Buf; Count: Word [;var Result: Word]) Передает Count записей из переменной Bufe файл F. Необязательный параметр Result содержит истинное количество переданных записей.

Приложение Г. Модуль CRT

Модуль Crt содержит подпрограммы управления текстовым выводом на экран дисплея, звуковым генератором и чтения клавиатуры. Для подключения модуля к программе необходимо сразу после заголовка программы написать предложение: uses crt; , после чего можно будет использовать функции и процедуры данного модуля.

В режиме текстового вывода используются следующие координаты экрана: левый верхний угол экрана имеет координаты 1,1; горизонтальная координата возрастает слева направо, вертикальная - сверху вниз. Если на экране определено окно, все координаты определяются относительно границ окна. Исключением являются координаты процедуры Window установки границ окна, которые всегда задаются относительно границ экрана.

Координатная плоскость экрана среды Паскаль в текстовом обычном режиме имеет следующий вид:



При этом координаты оси У – это целые числа, отсчет которых начинается сверху с единицы и заканчивается числом 25, а координаты X следуют слева направо и имеют диапазон значений 1..80. При обращении к точке экрана сначала указывается координата X, а затем У.

Для чтения клавиатуры используются две функции - **KeyPressed** и **ReadKey**. Функция KeyPressed определяет факт нажатия на любую клавишу и не приостанавливает дальнейшее исполнение программы. Функция ReadKey читает расширенный код нажатой клавиши. Если к моменту обращения к функции не была нажата ни одна клавиша, программа приостанавливает свою работу, ожидая действий пользователя.

Управление звуковым генератором строится по схеме Sound - Delay - NoSound. Процедура Sound включает звуковой генератор и заставляет его непрерывно генерировать звук нужного тона. Процедура Delay приостанавливает работу программы на заданное число миллисекунд реального времени. Процедура NoSound отключает звуковой генератор.

Например, использование функции KeyPressed delay и delline. В данной программе звездочка движется вдоль строк экрана до тех пор, пока не будет нажата какая-либо клавиша.

```
uses crt;
var x,y: integer;
begin
clrscr;
repeat
        write(‘*’)
        delay(9000);
        delline;
until keypressed;
end.
```

Пример использования readkey, Sound - Delay - NoSound. В программе в цикле включается режим ожидания нажатия клавиши и, если какая-либо клавиша нажата, звездочка будет двигаться вдоль строк экрана, данное действие происходит до тех пор, пока не будет нажат enter.

```
uses crt;
var x, y:integer;
        c:char;
begin
        clrscr;
        repeat
                c:=readkey;
                write(‘*’);
```

```
            delay(9000);
            delline;
        until c=#13;
    end.
```

Константы

Константы режима работы

```
const
BW40    = 0;{Черно-белый, 40 символов, 25 строк}
BW80    = 2;{Черно-белый, 80 x 25}
Mono    = 7;{Монохромный, 80 x 25}
C040    = 1;{Цветной, 40 x 25}
C080    = 3;{Цветной, 80 x 25}
Font8x8 = 256;{Для EGA/VGA режим 43 или 50 строк}
C40     = C040;{Для совместимости с версией 3.0}
C80     = C080;{Для совместимости с версией 3.0}
```

Константы цветов

```
const
Black = 0;{Черный}
Blue = 1;{Синий}
Green = 2;{Зеленый}
Cyan = 3;{Голубой}
Red = 4;{Красный}
Magenta = 5;{Фиолетовый}
Brown = 6;{Коричневый}
LightGray = 7;{Светло-серый}
DarkGray = 8;{Темно-серый}
LightBlue = 9;{Ярко-синий}
LightGreen = 10;{Ярко-зеленый}
LightCyan = 11;{Ярко-голубой}
LightRed = 12;{Розовый}
LightMagenta= 13;{Малиновый}
Yellow = 14;{Желтый}
```

White = 15;*{Белый}*
Blink = 128;*{Мерцание символа}*

Переменные

var
CheckBreak:Boolean;{Разрешает/запрещает контроль Ctrl-Break}
CheckEof:Boolean; {Разрешает/запрещает контроль Ctrl-Z}
CheckSnow:Boolean {Разрешает/запрещает контроль "снега"}
Directvideo:Boolean; {Разрешает/запрещает прямой доступ к видеопамяти}
LastMode:Word; {Хранит последний текстовый режим}
TextAttr:Byte; {Хранит текущий байт атрибутов}
WindMin:Word; {Координаты левого верхнего угла текущего окна} .
WindMax:Word; {Координаты правого нижнего угла}

Процедуры и функции

Функции

Function KeyPressed; Boolean Возвращает True, если на клавиатуре была нажата клавиша, и False в противном случае. Не задерживает исполнение программы.

Function ReadKey: char Читает символ с клавиатуры без эхоповтора на экране. Приостанавливает исполнение программы до нажатия на любую клавишу, кроме Shift, Ctrl, Alt, CapsLock, NumLock, ScrollLock.

Function WhereX: Byte Возвращает горизонтальную координату текущей позиции курсора относительно текущего окна.

Function WhereY: Byte Возвращает вертикальную координату текущей позиции курсора относительно текущего окна.

Процедуры

Procedure AssignCrt (var F: Text) Связывает с файловой переменной устройство CON (клавиатуру для ввода и экран для вывода).

Procedure ClrEol Удаляет все символы от текущей позиции курсора до конца строки без перемещения курсора.

Procedure ClrScr Очищает экран (окно) и помещает курсор в верхний левый угол.

Procedure Delay (D: word) Приостанавливает работу программы на указанное число D миллисекунд.

Procedure DelLine Удаляет строку, на которой находится курсор, и перемещает все строки ниже этой строки на строку вверх. Нижняя строка очищается.

Procedure GotoXY(X, Y; Byte) Перемещает курсор в нужное место экрана (окна).

Procedure Highvideo Устанавливает высокую яркость символов.

Procedure InsLine Вставляет пустую строку в позицию курсора. ,

Procedure LowVideo Устанавливает низкую яркость символов.

Procedure NormVideo Устанавливает нормальную яркость символов.

Procedure NoSound Выключает звуковой генератор.

Procedure Sound (F: word) Включает звуковой генератор. F - частота звука (Гц).

Procedure TextBackground (Color: Byte) Устанавливает цвет фона.

Procedure TextColor (Color: Byte) Устанавливает цвет символов.

Procedure TextMode (Mode: Word) Устанавливает нужный текстовый режим.

Procedure Window(X1, Y1, X2, Y2 : Byte) Определяет текстовое окно на экране. X1, Y1 - координаты левого верхнего угла, X2, Y2 - правого нижнего угла.

СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ

1. Абрамов С.А., Зима Е.В. Начала информатики. – М. Наука. Гл. ред. Физ. – мат. Лит., 1989;
2. Аладьев В.З., Хунт Ю.Я, Шишаков М.Л.. Основы информатики. Учебное пособие. Издание 2-е,переработанное и дополненное. – М.: Информационно-издательский дом «Филин», 1999;
3. Бурин Е.А., Введение в основы информатики и вычислительной техники. – Алма-Ата «Мектеп», 1989;
4. Васильев П.П. Турбо Паскаль в примерах и задачах: Освой самостоятельно: Учеб. Пособие. – М: Финансы и статистика, 2003;
5. Ермеков Н., Криворучко В., Кафтункина В. Информатика. Учебник для 9 класса. Алматы. «Жазушы», 2002;
6. Лахтин В.А. «Учебник программирования на языке Турбо Паскаль», Москва Учеб. Пособие. - 1999.
7. Фаронов В.В., Turbo Pascal 7.0. Начальный курс: учебное пособие. – М.: КНОРУС, 2005;
8. Шаньгин В.Ф., Поддубная Л.М., Голубев-Новожилов Ю.С. Программирование на языке «Паскаль». – Москва «Высшая школа», 1988.

СОДЕРЖАНИЕ