ФЕДЕРАЛЬНОЕ АГЕНТСТВО ПО ОБРАЗОВАНИЮ
ВОРОНЕЖСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

Язык программирования Pascal
Линейные программы

Практикум

Специальность 010101 (010100) Математика

ВОРОНЕЖ
2005

Утверждено научно-методическим советом Математического факультета
–
(28 февраля 2005 года, протокол №6)

Составители: Васильев В.В., Хливненко Л.В.

Практикум подготовлен на кафедре математического моделирования математического факультета Воронежского государственного университета.

Рекомендуется для студентов вечернего отделения математического факультета Воронежского государственного университета.

1. Общие сведения о языке Паскаль

Язык программирования **Паскаль** был придуман **Никласом Виртом**. Позднее язык был модифицирован и появились версии языка. Мы будем работать с версией **Турбо Паскаль**, созданной корпорацией Borland. На базе **Турбо Паскаля** корпорацией Borland разработан язык Object Pascal, на котором пишутся программы-обработчики в визуальной среде программирования Delphi.

Язык программирования **Паскаль** относится к языкам **высокого уровня.** Для языков высокого уровня текст программы выглядит как набор инструкций, понимаемых ПК. Инструкции представляют собой служебные слова, которые трактуются в строго определенном смысле. После запуска программы на исполнение, программа автоматически переводится в машинные коды.

Программа, написанная на языке программирования **низкого уровня**, содержит непосредственно исполняемые машинные команды.

Трансляция – процесс перевода программы, написанной на языке высокого уровня, в машинные коды.

Все синтаксические ошибки обнаруживаются на этапе трансляции программы. Программа на Паскале выполняется целиком.

Вместе с трансляторами работают программы-компановщики (*линкеры*).

Помните, что компьютер отслеживает лишь синтаксические ошибки. Отвечать на вопрос, правильно ли решена задача, всё равно придётся Вам самим.

Как и во всех европейских языках, **текст программы на Паскале** состоит из слов и знаков препинания. Если между словами нет знака препинания, их должен разделять хотя бы один пробел или конец строки.

В **алфавит языка** Паскаль входят десятичные и шестнадцатеричные цифры, буквы, специальные символы + - * / = , ‘ . : ; < > [ ] ( ) { } @ $ #, *а также пары символов* <> <= >= := (* *) (. .), пробелы и зарезервированные слова (*инструкции, имеющие строго определенный смысл*).

Большие и малые буквы транслятором не различаются!

Характеристики объектов называются **величинами**. Например, для объекта **Дом** могут быть рассмотрены следующие величины – количество квартир, количество этажей, стоимость 1 кв. метра жилья, фамилии жильцов, номера телефонов жильцов, наличие горячей воды.

Каждая **величина** характеризуется своим именем, значением и типом.

Имя величины является уникальным идентификатором величины. **Идентификатор** - это последовательность латинских букв и цифр, начинающаяся с буквы, несовпадающая ни с каким зарезервированным словом. В именах можно использовать некоторые разрешенные символы, например, знак подчеркивания _. Знак подчеркивания удобно использовать для создания имен из нескольких слов.

Значение величины – фактическое значение характеристики объекта. Например, значением величины **Наличие горячей воды** может быть **Да** или **Нет**.

Если значение величины меняется в программе, т о величина называется **переменной**, в противном случае - **постоянной** (*константой*).

Константа – конкретное значение определенного типа.

Переменная – сплошной поименованный участок оперативной памяти, предназначенный для хранения информации определенного типа.

Тип величины - множество значений, принимаемых величиной. К простым типам данных относятся порядковые и вещественные типы.

Порядковые типы данных включают целые типы (*разные диапазоны целых чисел*), логический тип (*два значения – Да/Нет*), символьный тип (*значением величины является один символ*), перечисляемый тип (*задает список возможных значений*), тип диапазон (*задает интервал возможных значений*).

Структурированные типы данных включают массивы (*многомерные таблицы пронумерованных однотипных значений*), записи (*совокупность величин разного типа*), множества (*наборы однотипных ненумерованных значений*), файлы (*набор компонентов одного типа, в том числе и структурированного, который можно передать на диск или логическое устройство*).

В Паскале есть тип **Строка**, предназначенный для работы с текстами и похожий на одномерный массив, состоящий из символов.

В Паскале имена (*уникальные идентификаторы*) имеют не только переменные величины и константы, а также метки, процедуры и программы.

Метка – обозначение начала участка кода в программе, на который осуществляется принудительный переход при исполнении программы.

Процедура – подпрограмма, в которой создается новая инструкция, «понимаемая» программой. Величины, содержащие входные и выходные значения процедуры, называются **параметрами** процедуры.

В { } или в (* *) в программе приводятся комментарии (*поясняющий текст, возможно, на русском языке*). На ход работы программы комментарии не влияют.

Операторы – зарезервированные слова, содержащие указание что-либо сделать. Например, оператор перехода имеет вид: **goto метка**.

Операторные скобки – это пара зарезервированных слов **begin** (*начало*) и **end** (*конец*). Каждому **begin** должен соответствовать **end**. Операторные скобки используются при создании группы последовательно выполняемых операторов.

Символ ; (*точка с запятой*) обозначает конец инструкции. Этот символ может опускаться перед **end** и никогда не ставится перед служебным словом **else** (*иначе*). Лишний символ ; считается пустым оператором и к ошибке не приводит.

Структура программы на языке Паскаль:

Заголовок	**Program** имя_программы;
Раздел описаний	
• Описание меток	**Label**<числа или идентификаторы через запятую>;

• Описание констант	**Const**<имя1>=<значение>;...;<имяn>=<значение>;
• Описание типов	**Type** <имя>=<тип пользователя>;...;
• Описание переменных	**Var** <имя1>,...,<имяn>:<тип1>;...;
• Описание процедур: заголовок процедуры, описание внутренних переменных процедуры, раздел операторов.	**Procedure** <имя_процедуры> [(<список параметров с описанием типов>)]; **var** <имя1>,...,<имяn>:<тип1>;...; **begin** <операторы процедуры>; **end**;
• Раздел операторов	**Begin** <последовательность операторов>; **End**.

Вся программа на Паскале - это **заголовок** (*который можно опускать*), **блок** (*состоящий из раздела описаний и раздела операторов*), точка (*обозначающая конец кода программы*).

Описательная часть блока является необязательной, то есть раздел описаний может отсутствовать. Некоторые неиспользуемые описания могут быть опущены.

Исполнительная часть блока (*раздел операторов*) является обязательной. Таким образом, самой короткой программой на Паскале будет строчка **begin end**.

В программе на языке Паскаль надо описывать все используемые переменные! Для описанной переменной в оперативной памяти ПК резервируется область под хранение значений величины. Длина резервируемой области определяется типом переменной.

В начале выполнения программы все переменные имеют пустое или нулевое значение!

2. Числовые типы данных. Арифметические выражения. Операторы присваивания, ввод и вывод информации

Рассмотрим **целые типы**. Целые типы отличаются друг от друга количеством байт в оперативной памяти, отводимым под хранение значений целого типа. Длиной кода определяется и диапазон возможных значений типа.

Название целого типа	Длина в байтах	Диапазон значений
Byte	1	0…255
ShortInt	1	-128…+127
Word	2	0…65535
Integer	2	-32768…+32767
LongInt	4	-2 147 483 648…+2 147 483 647

В задачах среди всех целых типов часто отдают предпочтение типу **Integer**.

Предусмотрены следующие **арифметические операции**, у которых оба **операнда** (*числа, переменные, константы или арифметические выражения,*

с которыми проводится операция) целые и результат целый: **+** сложение, **-** вычитание, ***** умножение, **div** целочисленное деление, **mod** остаток от деления.

Обычное деление **/** применимо к операндам целого типа, но результат деления всегда считается вещественного типа.

Если операция применяется к операндам разного типа, то результат будет иметь тип более мощного операнда.

Выход за границы типа переменной при вычислении может приводить к некорректной работе программы.

Например, если вещественной переменной присвоить значение 32767+5, то при запуске программы будет выведено сообщение об ошибке.

Существуют **арифметические функции**, у которых аргумент целый и результат целый: **abs()** абсолютная величина (*модуль*), **sqr()** возведение в квадрат, **succ()** число на единицу большее, **pred()** число на единицу меньшее. Функция **odd()** возвращает значение **True** (*истина*), если аргумент функции – нечетное число. Функция **Random(n)** возвращает псевдослучайное число, равномерно распределенное в диапазоне **0**…**n-1** (*тип функции совпадает с типом аргумента*).

Аргумент функции всегда пишется в круглых скобках!

Рассмотрим **вещественные типы**. Вещественные числа имеют две **формы записи**.

Традиционная форма записи **с фиксированной точкой** представляет собой десятичное представление вещественного числа с целой и дробной частями, разделенными точкой. Например, 462.7

Форма записи вещественного числа **с плавающей точкой** выглядит так: вещественное число записывается в форме с фиксированной точкой, к числу добавляется справа буква "E" или "e" и целое число, обозначающее порядок числа. Например, число 462.7 можно записать как 4,627**E**2, что будет означать $4{,}627*10^2$.

Название вещественного типа	Длина в байтах	Количество значащих цифр	Диапазон десятичного порядка
Real	4	11…12	-39…+38
Double	6	15…16	-324…+308
Extended	8	19…20	-4951…+4932

При решении задач предпочтение часто отдается вещественному типу Real.

Двумя важными характеристиками вещественного типа являются **точность** (*максимально возможное количество значащих цифр*) и **диапазон** (*разброс возможного значения порядка чисел*). В памяти ПК вещественное число хранится в виде трех структурных частей: знак, порядок числа и **мантисса** числа (*набор значащих цифр*).

К вещественным операндам применимы все арифметические операции (+-*/).

Существуют **арифметические функции**, у которых аргумент и результат вещественные: уже известные нам **abs(), sqr(), random()**, а также **sqrt()** квадратный корень, **sin()** синус, **cos()** косинус, **arctan()** арктангенс, **ln()** натуральный логарифм, **exp()** экспонента, **frac()** дробная часть числа, **int()** целая часть числа, **pi** число p.

Аргументы стандартных тригонометрических функций задаются в радианах!

Существует две функции, преобразующие вещественный аргумент в целочисленный результат: **trunc()** отбрасывание дробной части, **round()** округление.

Арифметическое выражение – это запись, которая может содержать переменные, константы, арифметические операции и функции, числа и круглые скобки.

Арифметические выражения, содержащие операцию, записываются в строку.

Например, выражение $\dfrac{x^2+y^2}{1-\dfrac{x^2-y^2}{2}}$, записанное по правилам Паскаля, будет выглядеть так: **(sqr(x)+sqr(y))/(1-(sqr(x)-sqr(y))/2).**

*В арифметическом выражении нельзя опускать знак * операции умножения.*

Для сокращения числа круглых скобок в арифметических выражениях используется традиционное старшинство арифметических операций: в первую очередь выполняются операции ***, div, mod, /,** затем **+** и **-.** Идущие подряд операции одного приоритета выполняются слева направо.

Оператор присваивания представляет собой специальный символ, состоящий из пары символов **:=**. Оператор присваивания записывает в ячейку памяти, отведенную под переменную величину, фактическое значение этой величины.

Формат оператора присваивания (*способ записи, синтаксис*):

Переменная:=значение;

Оператор присваивания значение, стоящее в левой части, присваивает переменной, стоящей в правой части.

Если справа будет стоять арифметическое выражение, то вначале происходит вычисление значения арифметического выражения, а затем пересылка этого значения в память.

Переменной целого типа недопустимо присваивать значение вещественного типа. Обратная операция является допустимой.

Вывод информации – это процесс передачи данных из оперативной памяти во внешнюю. Буфером при выводе данных на дисплей ПК служит стан-

стандартный файл Output. Процедура вывода **Write()** обеспечивает вывод потока выходных данных через внутренний файл Output на экран ПК.

Выходной поток данных - это последовательность отображаемых на экране символов, которая наращивается при выводе данных процедурой **Write().**

По умолчанию информация в среде Турбо Паскаля выводится в текстовом режиме работы монитора. Для отображения информации в графическом режиме нужно включать в программу специальные команды.

Формат процедуры вывода (*способ записи, синтаксис*):

```
write('текст',переменная,арифметическое выражение);
```

Выводимые значения указываются в круглых скобках через запятую. Текст заключается в апострофы. Если в список вывода входят переменные или арифметические выражения, то на экран будут выведены их значения.

Если после вывода информации нужно перевести курсор на новую строку, то используют другую процедуру вывода :

```
writeln('текст',переменная,арифметическое выражение);
```

Процедура **writeln** без параметров просто переводит курсор в начало следующий строки. Таким образом, процедура **writeln(x)** эквивалентна паре процедур **write(x); writeln;**

Значения переменных вещественного типа по умолчанию выводятся на экран в форме записи с плавающей точкой. Можно выводить вещественные значения в форме записи с фиксированной точкой. Для этого следует указать количество выводимых значащих цифр и количество цифр в дробной части.

Пусть значение переменной **x** равно **4,627E2**. Чтобы вывести это значение в привычной для нас форме, нужно вызвать процедуру **write(x:4:1);**

Результатом исполнения оператора будет появление на экране числа **462,7**.

Ввод информации – это процесс передачи данных от внешнего носителя (*клавиатура*) в оперативную память. Буфером при вводе данных с клавиатуры служит стандартный файл Input. Процедура ввода **Read()** обеспечивает ввод потока входных данных через внутренний файл Input в оперативную память, доступную программе.

Входной поток данных - это последовательность символов, которая наращивается путем дописывания в ее конец набираемых на клавиатуре символов и сокращается при считывании символов процедурой **Read().** Нажатие на клавишу Enter добавляет во входной поток конец строки.

Формат процедуры ввода:

```
read(переменная1,…, переменнаяn);
```

Вводимые значения отделяются друг от друга пробелом или нажатием на клавишу Enter. Если после ввода информации нужно перевести курсор на новую строку, то используют другую процедуру ввода :

```
readln(переменная1,…, переменнаяn);
```

В этом случае вводимые значения отделяются друг от друга нажатием на клавишу Enter.

Действие процедуры **readln** *без параметров заключается в чтении входного потока до тех пор, пока не будет считан конец строки. Если при этом входной поток оказывается пустым, то происходит переход в режим ожидания, выход из которого происходит при нажатии* Enter.

Переменные в списках ввода и вывода могут иметь разный тип!

3. Интегрированная инструментальная среда Турбо Паскаль

Составим **программу для вычисления расстояния между двумя точками, координаты которых вводит пользователь**.

Программа на Паскале	Комментарии
`Program Distance;`	*{Заголовок программы.}*
`Var x1, y1, x2, y2, R: real;`	*{Описание переменных вещественного типа.}*
`Begin`	*{Начало раздела операторов.}*
`Write('Введите координаты начала отрезка (x1,y1):');`	*{Вывод в одной строке поясняющего текста. }*
`Readln(x1,y1);`	*{Ввод в той же строке значений переменных* x1, y1. *Переход на следующую строку. }*
`Write('Введите координаты конца отрезка (x2,y2):');`	*{Вывод в строке поясняющего текста. }*
`Readln(x2,y2);`	*{Ввод в той же строке значений переменных* x2, y2**..** *Переход на следующую строку. }*
`R:=sqrt(sqr(x2-x1)+sqr(y2-y1));`	*{Присваивание переменной* R *значения арифметического выражения* $\sqrt{(x_2-x_1)^2+(y_2-y_1)^2}$ *}*
`Writeln('Длина отрезка равна ', R);`	*{Вывод ответа на экран. }*
`End.`	*{Конец раздела операторов и программы. }*

Познакомимся с **интегрированной инструментальной средой Турбо Паскаль** (*включающей экранный редактор, компилятор с языка Паскаль, компоновщик, отладчик*) и проверим работу написанной выше программы.

На некоторых ПК Паскаль можно запустить только из оболочки MS DOS!

План примера 1

1-5. Создание файла distance.pas в своей папке.

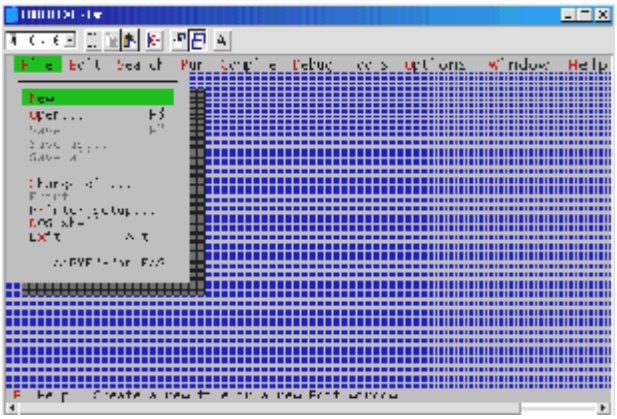
6-9. Набор текста программы и ее исполнение.

10-11. Организация задержки исполнения программы до нажатия клавиши Enter.

12-13. Вывод результата в виде числа с одним знаком после запятой.
14-18. Изменение цвета фона и символов. Очистка экрана.
19-21. Сохранение изменений в программе. Создание резервной копии программы. Выход из интегрированной инструментальной среды Турбо Паскаль.

Пример работы **1**: *Подсказка*

1. Запустите среду **Турбо Паскаля**	C:\Program Files\TP7\BIN\turbo.exe *или клик по ярлыку на рабочем столе*



На экране появится окно интегрированной среды Турбо Паскаль. Для активизации строки меню нужно нажать клавишу F10 или сделать левый клик на нужном пункте меню. Строка меню состоит из пунктов File (*файл*), Edit (*правка*), Search (*поиск*), Run (*исполнение*), Compile (*компиляция*), Debug (*отладка*), Tools (*инструменты*), Options (*настройки*), Window (*окно*), Help (*помощь*).

2. Откройте пункт меню File *(файл)*	*Клик по пункту, либо* F10 *и клавиши со стрелками*
3. Создайте новый файл	File | New

Тексты программ на языке Паскаль хранятся в файлах с расширением pas. Имя нового файла написано в верхней части окна документа - Noname00.pas.

Сохраним созданный файл под именем distance.pas в своей папке.

4. Укажите путь к своей папке	File | Change Dir... (**Файл | Сменить каталог**...)
5. Сохраните файл под именем distance.pas	File | Save as... (**Файл | Сохранить как**...)
6. Наберите текст программы для вычисления расстояния между двумя точками	*См. текст программы на стр.* 8-9. *Комментарии можно не набирать*
7. Отправьте программу на исполнение	Run | Run (**Исполнение | Исполнить**) *или* Ctrl+F9

Компьютер напечатал результат так быстро, что мы не успели ничего увидеть.

8. Отключите окно интегрированной среды и посмотрите результат выполнения программы	Alt+F5
9. Включите окно интегрированной среды	*Нажмите любую клавишу*

Задержать исполнение программы может пустой оператор **`readln`**, поставленный перед **end**. Программа будет исполняться до тех пор, пока Вы не нажмете на клавишу Enter.

10. Заставьте программу исполняться до на-	*Используйте* readln

жатия клавиши Enter	
11. Отправьте программу на исполнение	Run|Run (**Исполнение|Исполнить**) *или* Ctrl+F9

Успешным выполнением программы будет фрагмент черного экрана MS-DOS, на котором белыми символами (*после реквизитов Паскаля*) напечатаны выводимые программой символы.

Обратите внимание на то, что расстояние между двумя точками выводится в форме записи вещественного числа с плавающей точкой. Выведем это значение в форме с фиксированной точкой и одним знаком после запятой.

12. Выведите результат в виде числа с одним знаком после запятой	*Исправьте* **R:6:1**
13. Отправьте программу на исполнение	Run|Run (**Исполнение|Исполнить**) *или* Ctrl+F9

Проверьте правильность работы программы, подставляя разные значения, в том числе и отрицательные вещественные числа!

Научимся делать принудительную очистку экрана, управлять цветом фона и символов. Познакомимся с тремя процедурами, входящими в библиотеку Crt, поставляемую вместе с Паскалем.

`ClrScr` – очистка экрана.

`TextBackGround(`Цвет фона`)` - установка цвета фона.

`TextColor(`Цвет символов`)` – установка цвета символов.

В таблице даны константы цветов, определенные в библиотеке Crt.

Black	Черный	Magenta	Фиолетовый	LigtGreen	Светло-зеленый
Blue	Синий	Brown	Коричневый	LigtCyan	Светло-бирюзовый
Green	Зеленый	LigtGray	Светло-серый	LigtRed	Розовый
Cyan	Бирюзовый	DarkGray	Темно-серый	LigtMagenta	Малиновый
Red	Красный	LigtBlue	Светло-синий	Yellow	Желтый

Значением цвета фона и символов может быть целое число от 0 *(*Black*) – черный цвет до* 15 *(*White*) – белый цвет.*

Получим белые символы на синем фоне.

14. Подключите к программе библиотеку Crt	*Напишите после заголовка программы строку* Uses Crt;
15. Установите синий цвет фона	*Напишите после слова* Begin *команду* TextBackGround(Blue);
16. Очистите экран. Экран заполняется установленным цветом фона	*Напишите после* TextBackGround(Blue); *команду* ClrScr;
17. Установите белый цвет символов	*Напишите после* ClrScr; *команду* TextColor(White);
18. Отправьте программу на	Run|Run (**Исполнение|Исполнить**) *или* Ctrl+F9

исполнение	

Поэкспериментируйте с цветами фона и символов!

19. Сохраните изменения в программе	File|Save (**Файл|Сохранить**) *или* F2
20. Сохраните файл под именем distan_.pas	File|Save as... (**Файл|Сохранить как**...)
21. Выйдите из среды Турбо Паскаля	File|Exit (**Файл|Выход**) *или* Alt+F5

В примере работы 2 мы познакомимся с возможностями интегрированной среды Турбо Паскаль, часто использующимися при разработке программ.

План примера 2

1-3. Открытие текстов программ, хранящихся в Вашей папке.

4-8. Навигация по тексту программы. Поиск и замена фрагментов.

9-15. Выделение, перемещение, копирование и удаление блоков.

16-18. Переключение и закрытие открытых документов.

Пример работы **2**: *Подсказка*

1. Запустите среду **Турбо Паскаля**	C:\Program Files\TP7\BIN\turbo.exe *или клик по ярлыку на рабочем столе*
2. Укажите путь к своей папке	File|Change Dir... (**Файл|Сменить каталог**...)
3. Откройте программу distan_.pas	File|Open... (**Файл|Открыть**) *или* F3

Познакомимся с **клавишами навигации** по тексту программы.

- Ctrl+← - Перемещение на одно слово влево.
- Ctrl+→ - Перемещение на одно слово вправо.
- Home - Перемещение к началу строки.
- End - Перемещение к концу строки.
- Ctrl+Home - Переход на первую строку.
- Ctrl+End - Переход на последнюю строку.
- Page Up - Перемещение на одну страницу вверх.
- Page Down - Перемещение на одну страницу вниз.

4. Выполните все выше перечисленные способы навигации по тексту программы

Научимся **искать** нужный фрагмент кода программы **и заменять** его. Заменим в программе идентификатор x1 на x.

5. Перейдите в начало первой строки	Ctrl+Page Up
6. Укажите образец поиска и замены	*Нажмите* Ctrl+Q, *затем* **A**

В диалоговом окне Replace (*замена*) в поле Text to find (*текст для поиска*) наберите **x**1, а в поле New text (*новый текст*) – **x**. **ОК**. После подтверждения замены, автоматически будет изменен один идентификатор.

Для продолжения поиска и замены нужно нажимать Ctrl+L.

7. Замените все оставшиеся идентификаторы **x**1	*Нажимайте* Ctrl+L

на **x**	

Для осуществления поиска без замены надо нажать Ctrl+Q, *затем* F. *Для повторного поиска используется сочетание* Ctrl+L.

8. Найдите в тексте все идентификаторы **x**2	Ctrl+Q, F. Ctrl+L

Рассмотрим приемы работы с **блоком** (*выделенной последовательностью символов*). Научимся копировать, перемещать и удалять фрагменты кода.

Выделим блок, состоящий из процедур библиотеки Crt.

9. Поставьте курсор в начало блока. Отметьте начало блока	*Нажмите* Ctrl+K, *затем* B
10. Поставьте курсор в конец блока. Отметьте конец блока	*Нажмите* Ctrl+K, *затем* K

Блок будет выделен серым цветом.

11. Снимите (*восстановите*) выделение с блока	*Нажмите* Ctrl+K, *затем* H

Познакомимся со вторым способом выделения блока.

12. Выделите блок	*Используйте* Shift+←, Shift+→, Shift+↑, Shift+↓
13. Переместите блок	*Поставьте курсор перед оператором* readln. *Нажмите* Ctrl+K, *затем* V
14. Скопируйте блок	*Поставьте курсор после* Begin. *Нажмите* Ctrl+K, *затем* C

Допустимо копировать блок через Буфер обмена, по нажатию клавиш Ctrl+Insert, Shift+Insert. *Попробуйте!*

Переместить блок можно по нажатию клавиш Shift+Delete, Shift+Insert. *Попробуйте!*

15. Удалите блок	*Нажмите* Ctrl+K, *затем* Y. *Либо* Ctrl+Delete

Среда Турбо Паскаль позволяет работать с несколькими открытыми документами. Научимся работать с открытыми окнами.

16. Откройте программу distance.pas	File | Open... (**Файл** | **Открыть**) *или* F3
17. Переключитесь между открытыми документами	F6
18. Закройте окно документа distan_.pas	*Клик по квадрату в левом верхнем углу окна документа*

В следующей части мы вспомним этапы решения задач и познакомимся с вариантом оформления линейной программы.

4. Этапы решения задачи. Линейные программы

Чтобы научиться более эффективно решать задачи средствами Паскаля, желательно начинать процесс решения с обдумывания, а затем уже браться за кодировку.

Решение задачи включает в себя следующие этапы.

I. **Постановка задачи**. Определение цели решения задачи, набора исходных данных и результатов, общего подхода к решению задачи.

II. **Математическое описание задачи**. Создание математической модели решаемой задачи, которая может быть реализована средствами Паскаля.

III. **Алгоритмизация задачи**. Разработка алгоритма (*последовательности конечного числа действий*) решения задачи, приводящего к поставленной цели. Алгоритм может быть записан в виде блок-схемы или на псевдокоде.

IV. **Программирование**. Написание алгоритма с помощью конструкций языка программирования Паскаль.

V. **Разработка тестов**. Определение выходных данных для некоторых наборов исходных данных. Этап служит для проверки правильности разработанного алгоритма и программы.

VI. **Кодирование**. Набор программы в экранном редакторе интегрированной инструментальной среды Турбо Паскаль. Запись программы на диск.

VII. **Тестирование и отладка программы**. Обнаружение и исправление ошибок в тексте программы. При наличии ошибки среда Турбо Паскаль при компиляции выдает сообщение о причине ошибки.

VIII. **Получение и анализ результатов**. Проверка корректности выдаваемых программой результатов. Оценка правильности решения задачи при разных значениях исходных данных.

Сложные задачи решаются **методом последовательной детализации** (*решение делится на крупные блоки, большие блоки разбиваются на более мелкие, детально прорабатываются простые задачи, входящие в состав мелких блоков*).

Как правило, для наглядности блоки (*логически законченные фрагменты программы*) снабжаются комментариями.

В программах должна быть предусмотрена защита от ошибок при вводе. Если введенное значение не является допустимым, то нужно повторить ввод.

Все вводимые и выводимые данные должны сопровождаться поясняющим их назначение текстом. Программа должна иметь приятный, дружественный интерфейс.

Линейные программы представляют собой последовательность простых инструкций.

Рассмотрим пример решения задачи, приводящей к линейной программе.

1. **Тема**. Линейные программы.

2. **Условие задачи**. Целой переменной присвоить сумму цифр целого трехзначного числа.

3. **Постановка задачи**. Цель – вычислить сумму цифр заданного целого трехзначного числа. Исходная информация – целое трехзначное число. Искомую сумму можно найти, зная цифры заданного трехзначного числа. Цифры можно получить как остатки от последовательного целочисленного деления заданного числа на десять.

4. **Математическая модель**. Примем следующие обозначения для:

- заданного целого трехзначного числа – k,
- количества единиц в заданном числе – k1,

- количества десятков в заданном числе – k2,
- количества сотен в заданном числе – k3,
- частного от последовательного целочисленного деления числа на десять – r,
- суммы цифр заданного трехзначного числа – s.

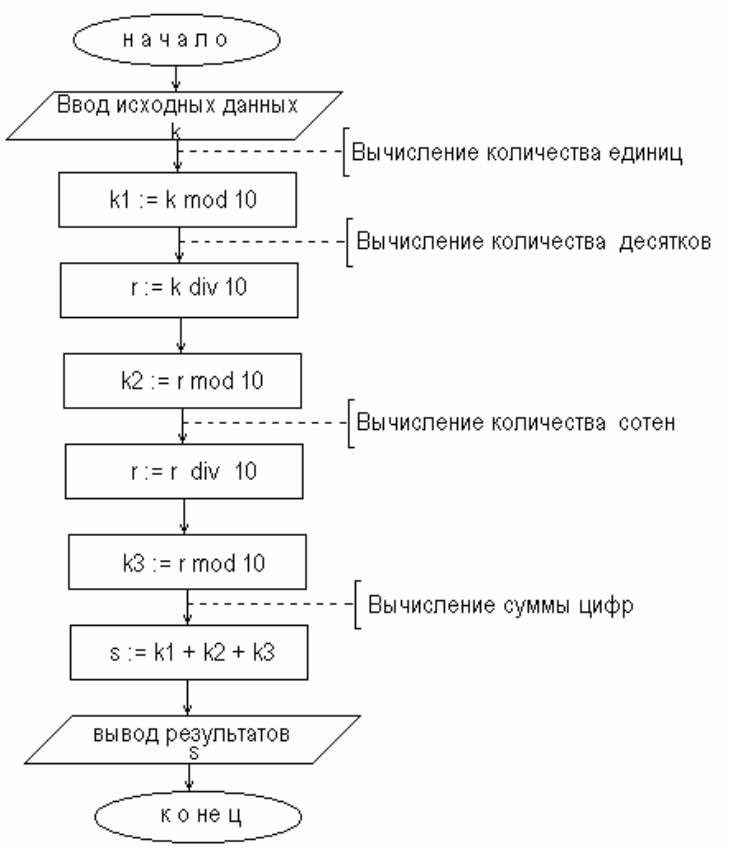
Тогда s=k1+k2+k3. Опишем процесс нахождения количества единиц, десятков и сотен в заданном числе k.

k1=остаток от деления k на 10.

r=частное от деления k на 10. k2=остаток от деления r на 10.

r=частное от деления r на 10. k3=остаток от деления r на 10.

5. **Блок-схема**.



6. **Программа**.

```
Program Summa_cifr;
Uses Crt;
Var k,k1,k2,k3,r,s:integer;
Begin
 {оформление фона и символов}
 textbackground(White);ClrScr; textcolor(Blue);
 {ввод целого трехзначного числа}
 write('Введите целое трехзначное число:'); readln(k);
 {вычисление количества единиц}
```

```
 k1:=k mod 10;
 {вычисление количества десятков}
 r:=k div 10; k2:=r mod 10;
 {вычисление количества сотен}
 r:=r div 10;k3:=r mod 10;
 {вычисление суммы цифр}
 s:=k1+k2+k3;
 {вывод суммы цифр целого трехзначного числа}
 writeln('Сумма цифр числа ',k,' равна ',s); readln
end.
```

7. **Тест**.

Введите целое трехзначное число:**758.**
Сумма цифр числа **758** равна **20.**

В задаче мы не сделали защиту данных при вводе, так как нужные для этого операторы ветвления и циклов будут рассмотрены нами в следующей лабораторной работе.

*Подготовьтесь к ответам на **все(!)** контрольные вопросы и выполните **все(!)** контрольные задания.*

Контрольные вопросы и задания

1. Общие сведения о языке Паскаль

1. Какие языки программирования называются языками высокого уровня?
2. Что такое трансляция программы?
3. Что такое интерпретация? Компиляция? Приведите примеры.
4. Какие языки программирования называются языками высокого уровня?
5. Из чего состоит текст программы на Паскале?
6. Что входит в алфавит языка Паскаль?
7. Как выглядят зарезервированные слова в окне редактора Турбо Паскаля?
8. Что такое величина? Чем она характеризуется?
9. Что такое идентификатор?
10. Какие величины называются переменными? Постоянными?
11. Что такое константа? Переменная?
12. Что такое тип величины?
13. Какие типы данных считаются простыми?
14. Охарактеризуйте порядковые типы данных.
15. Охарактеризуйте структурированные типы данных.
16. Что собой представляет строковый тип?
17. Что называется меткой?
18. Что такое процедура? Параметры процедуры?
19. Как написать комментарии в программе?
20. Что такое операторы? Приведите пример.
21. Что называется операторными скобками?
22. Как используется в программе точка с запятой?

23. Из каких частей состоит программа на Паскале?
24. Из каких частей состоит блок? Какая из них является обязательной?
25. Как выглядит самая короткая программа на Паскале?
26. Из чего состоит раздел описаний?
27. Как описать метки? Константы? Типы? Переменные?
28. Назовите структурные части процедуры?
29. Можно ли не описывать переменные и использовать их в тексте программы?
30. Для чего нужно описывать переменные? Как определить длину кода, предназначенного для хранения значений величины?
31. Какое значение имеют переменные в начале выполнения программы?

2. Числовые типы данных. Арифметические выражения. Операторы присваивания, ввод и вывод информации

32. Какие типы данных называются целыми? Чем они отличаются друг от друга?
33. Назовите все целые типы и характеризующие их параметры – длину кода и диапазоны значений.
34. Какой целый тип часто используют в задачах?
35. Что такое операнд?
36. Назовите 5 арифметических операций, у которых оба операнда и результат целые.
37. Чему равен результат вычисления выражения (a mod b)*b?
38. Какой тип имеет результат обычного деления?
39. Какой тип будет у результата операции, применяемой к операндам разного числового типа?
40. Что может произойти из-за выхода при вычислениях за границы типа?
41. Назовите 4 арифметические функции, у которых аргумент и результат целые.
42. Для чего предназначена функция odd()?Random(n)?
43. Как задается аргумент функции?
44. Что такое вещественные типы?
45. Охарактеризуйте две формы записи вещественных чисел.
46. Назовите все вещественные типы и их параметры – длину в байтах, количество значащих цифр и диапазон десятичного порядка.
47. Какому вещественному типу отдается предпочтение при решении учебных задач?
48. Назовите две характеристики вещественного типа. Что такое точность? Диапазон?
49. Как вещественное число хранится в памяти ПК?
50. Что такое мантисса вещественного числа?
51. Какие операции применимы к операндам вещественного типа? Какой тип имеет результат таких операций?
52. Как найти модуль числа? Квадрат? Квадратный корень? Экспоненту?

53. Как определить синус числа? Косинус? Арктангенс? Натуральный логарифм?
54. Для чего служат функции int() и frac()?
55. Можно ли аргументы тригонометрических функций задавать в градусах? Как это сделать?
56. Когда используются функции trunc()и round()?
57. Что такое арифметическое выражение?
58. Опишите расстановку приоритетов при выполнении арифметических операций.
59. Запишите по правилам Паскаля выражения:

а) $\dfrac{x^2+y^2}{1-\dfrac{x^3-y^3}{5}}$, б) $1+|x|+\sqrt{|x+1|}$, в) $\dfrac{1-\sqrt{1+|tg|x||}}{1{,}5-7{,}2x}$, г) 000.

60. Запишите на языке Паскаль следующие формулы:

а) $2\log_2\dfrac{x}{5}$, б) x^{300}, в) 5^x, г) $\arcsin x$, д) $\sqrt[5]{a^{-3}}$.

61. Напишите по правилам Паскаля несколько вариантов выражения, значение которого равно x^4. Для каждого из вариантов подсчитайте количество умножений, требующихся при вычислении значения выражения (*вычисление значения функции* **sqr** *требует одного умножения*).
62. Напишите выражения, записанные по правилам Паскаля, в традиционной математической форме:

а) $a*b/(c+d)-sqrt(\sin((a+b)/c))$, б) $\cos(abs(x*x*x+3)-1)/y/x$.

63. Вычислите значение выражений:

а) $tranc(4.8)-round(5.7)-1$; б) $tranc(-4.8)-round(-5.7)+abs(-3)$,

в) $30 div 60+succ(2)-pred(5)$; в) $24/(3*4)-24/3/4+24/3*4$.

64. Можно ли утверждать, что в Паскале значение выражения $(1/3)*3-1$ равно нулю?
65. Как записать в Паскале число e ? p ?
66. Как выглядит оператор присваивания? Каково его действие?
67. Что такое формат оператора?
68. Можно ли переменной вещественного типа присваивать значение переменной целого типа? А наоборот?
69. Какие из следующих последовательностей символов являются операторами присваивания:

а) a:=b;	**г) a*x+b:=0;**	**ж) z:=z+1,2;**
б)a=c+1;	**д) z:=0;**	**з) y:=y;**
в)a:b-sqr(2)	**е) z:=z+1;**	**и) -y:=y;**

70. Задать с помощью операторов присваивания следующие действия:

а) переменной **a** присвоить значение разности, а переменной **b** - полусуммы значений **x** и **y**;

б) переменной **a** присвоить значение удвоенного произведения значений переменных **x** и **y**, а переменной **b** - значение 0;

в) удвоить значение переменной **a** и изменить ее знак.

71. Дано **a**. Не используя никаких функций и никаких операций кроме умножения, получить a^{15} за пять операций. Допустимо использовать оператор присваивания.
72. Что такое вывод информации?
73. Для чего предназначен файл Output?
74. Что такое выходной поток данных?
75. Охарактеризуйте формат и действие двух процедур вывода данных.
76. Как используются апострофы в тексте программы?
77. Допустимо ли использовать процедуру вывода без параметров? Что при этом происходит?
78. Как организовать вывод вещественного числа в форме с фиксированной точкой? Приведите пример.
79. Что такое ввод информации?
80. Для чего предназначен файл Input?
81. Что такое входной поток данных?
82. Охарактеризуйте формат и действие двух процедур ввода данных.
83. Как отделяются друг от друга вводимые с клавиатуры значения?
84. В чем смысл процедуры **readln** без параметров?
85. Какие числа будут выведены в результате выполнения последовательности операторов `read(x,y); x:=x+y; y:=x-y; x:=x-y; write(x,y)`, если последовательность исходных данных была составлена из двух чисел:
    а) 3.5 и 2.4; б) 6.7 и -10.1?
86. Поменяйте местами значения переменных `x` и `y`.

3. Интегрированная инструментальная среда Турбо Паскаль

87. Покажите программу, вычисляющую расстояние между двумя точками. Объясните смысл ее кода.
88. Из чего состоит интегрированная инструментальная среда Турбо Паскаль?
89. Как запустить среду Турбо Паскаль?
90. Назовите пункты строки меню среды Турбо Паскаль.
91. Как выбрать команду из пункта меню с помощью клавиш на клавиатуре?
92. Как создать новый документ?
93. Как указать путь к нужной папке? Как сохранить файл в свою папку?
94. Как отправить программу на исполнение?
95. Как отключить окно среды Турбо Паскаль и посмотреть результат выполнения программы? Как вернуться в окно редактора?
96. Как можно задержать исполнение программы до нажатия клавиши ввода?
97. Что такое Crt? Как подключить к программе стандартную библиотеку?
98. Как очистить экран? Изменить цвет фона? Цвет символов?
99. Назовите константы цветов. Как задается цвет?

100. Напишите программу, вычисляющую гипотенузу и площадь прямоугольного треугольника по двум катетам. Сделайте желтые символы на зеленом фоне.
101. Как сохранить изменения в программе? Как сделать резервную копию программы?
102. Как выйти из среды Турбо Паскаль?
103. Как открыть текст программы из Вашей папки?
104. Как в программе переместиться на одно слово влево? Вправо?
105. Как перейти в начало строки? В конец строки?
106. Как попасть на первую строку? На последнюю?
107. Как переместиться на одну величину экрана вниз? Вверх?
108. Найдите и замените один идентификатор другим.
109. Как осуществить поиск нужного фрагмента по всему документу?
110. Что такое блок?
111. Покажите два способа выделения блока. Как снять-вернуть выделение?
112. Приведите два способа перемещения, копирования и удаления блока.
113. Как переключиться между открытыми окнами?
114. Как закрыть окно документа?

4. Этапы решения задачи. Линейные программы

115. Что продумывается при постановке задачи?
116. В чем состоит математическое описание задачи?
117. Как выглядит этап алгоритмизации задачи?
118. Чем отличаются этапы программирования и кодирования?
119. Зачем нужно разрабатывать тесты и проводить тестирование программы?
120. В чем состоит отладка программы?
121. Что оценивается на последнем этапе решения задачи средствами Паскаля?
122. В чем суть метода последовательной детализации?
123. Какие участки программы нужно снабжать комментариями? Зачем?
124. Как и зачем организовать защиту от ошибок при вводе?
125. Что такое дружественный интерфейс?
126. Какие программы называются линейными?
127. Даны два числа. Найти среднее арифметическое кубов этих чисел и среднее геометрическое модулей этих чисел.
128. Даны a, b, a . Найти площадь треугольника, две стороны которого равны a и b, а угол между этими сторонами равен a . Считать, что a - градусная мера угла.
129. Посчитать определитель третьего порядка. Предусмотреть удобный ввод.
130. Даны координаты двух векторов. Найти их векторное произведение.
131. Даны координаты трех векторов. Найти объем тетраэдра, построенного на этих векторах.

132. Даны координаты двух векторов. Определить угол между векторами.

133. **Индивидуальное(!) задание**, которое передается преподавателю перед началом собеседования по этой теме:

Номер индивидуального задания определяет преподаватель!

Опишите постановку задачи, создайте математическую модель ее решения, разработайте блок-схему и работающую программу, проведите тестирование и отладку программы, обдумайте полученные результаты.

Индивидуальные задания

1. Присвойте целой переменной третью от конца цифру в записи положительного целого числа.
2. Присвойте целой переменной первую цифру из дробной части положительного вещественного числа.
3. Идет k-я секунда суток. Определить, сколько полных часов (h) и полных минут (m) прошло к этому моменту.
4. Определить f - угол в градусах между положением часовой стрелки в начале суток и ее положением в h часов, m минут и s секунд ($0 \le h \le 11$, $0 \le m$, $s \le 59$).
5. Определить h – полное количество часов и m – полное количество минут, прошедших от начала суток до того момента (*в первой половине дня*), когда часовая стрелка повернулась на f градусов ($0 \le f < 11$, f – *вещественное число*).
6. Пусть k – целое число от 1 до 365. Присвоить целой переменной n значение 1, 2, …, 7 в зависимости от того, на какой день недели (**пн**, **вт**, **...**, **вс**) приходится k-й день невисокосного года, в котором 1 января – понедельник.
7. Поменяйте местами значения переменных **x**, **y** и z так, чтобы в **x** оказалось значение переменной **y**, в **y** – значение переменной z, а в z – прежнее значение переменной **x**.
8. Поменяйте местами значения целых переменных **x** и **y**, не используя дополнительные переменные.
9. Определите число, полученное выписыванием в обратном порядке цифр заданного целого трехзначного числа.
10. Вычислите дробную часть среднего арифметического и дробную часть среднего геометрического трех заданных чисел.
11. Дано время запуска ракеты в часах, минутах, секундах и время полета в секундах. Найти и напечатать время возвращения ракеты на землю.
12. Пусть даны целые числа m, n (*часы, минуты*), $0 \le m \le 11$, $0 \le n \le 59$, определяющие время суток. Определите наименьшее время (*число полных минут*), которое должно пройти до того момента, когда часовая и минутная стрелки на циферблате совпадут.
13. Пусть даны целые числа m, n (*часы, минуты*), $0 \le m \le 11$, 1, определяющие время суток. Определите наименьшее время (*число полных минут*), которое должно пройти до того момента, когда часовая и минутная стрелки на циферблате расположатся перпендикулярно друг другу.

Литература

1. Абрамов С.А. Начала информатики / С.А. Абрамов, Е.В. Зима. – М. : Наука, 1990.
2. Епанешников А. М. Программирование в среде Turbo Pascal 7.0 / А.М. Епанешников, В.А. Епанешников. – 3-е изд. – М. : Диалог-МИФИ, 1996.
3. Зубов В.С. Программирование на языке Turbo Pascal / В.С. Зубов. – М. : Филинъ, 1997.
4. Пильщиков В.Н. Сборник упражнений по языку Паскаль / В.Н. Пильщиков. - М. : Наука, 1989.
5. Программирование на языке Паскаль: задачник/под ред. Усковой О.Ф. – СПб: Питер, 2002.
6. Фаронов В.В. Turbo Pascal 7.0. Начальный курс / В.В. Фаронов. – М. : Нолидж, 1998.
7. Хершель Р. Турбо Паскаль / Р. Хершель. – Вологда : МП МИК, 1991.

©

Составители: Васильев Валерий Викторович,
Хливненко Любовь Владимировна

Редактор Тихомирова О.А.