***Колледж "Серебряный бор" при Центре "АГАТ"***

**Игошев А.Д.**

****

**Turbo Pascal v.6.0, 7.0**

***учебно-дидактическое пособие для средних учебных заведений***

**(процедурно - ориентированный блок)**

**г. Верхняя Пышма Свердловской обл.**

***1996 г.***

ВВЕДЕНИЕ.

В 1992 г. фирма Borland International выпустила два пакета программирования: Borland Pascal v. 7.0 и Turbo Pascal v. 7.0, последнему и посвящено данное пособие, т.к. BP предъявляет более строгие требования к аппаратуре, что может явиться камнем преткновения для учебных заведений.

Многие школы оснащены КУВТ "Корвет", на котором можно получить первичное представление о среде программирования Turbo Pascal- это среда Express Pascal (Turbo Pascal v. 4.0, адаптированная для процессора КР580 фирмой МикС). Опыт работы с этим пакетом изложен в пособии [11]. Часть учебных заведений оснащена IBM-совместимой техникой, на которую можно установить среду программирования Turbo Pascal v. 6.0 или 7.0, при наличии на винчестере около 5 Мб свободного дискового пространства, а вместе с документацией и TURBO VISION - около 10 МБайт и ОЗУ не менее 640 Кб. Обе версии обладают целым рядом достоинств, по сравнению с другими языками программирования, и наиболее подходят для учебных целей. В частности следует отметить следующее:

1. Turbo Pascal-постоянно совершенствующийся язык, выросший из стандартного Паскаля, ставшего популярным еще в 80-е годы.

2. Turbo Pascal основан на принципе структурного программирования и пошаговом методе проектирования программ, что способствует развитию культуры программирования.

3. Turbo Pascal удобен в разработке прикладных и системных программ.

4. Turbo Pascal-это строго типизированный язык, что позволяет легко разрабатывать представления для структур данных любой разрешимой задачи.

5. Turbo Pascal содержит все необходимые средства для объектно-ориентированного программирования.

6. Turbo Pascal имеет богатейшую библиотеку стандартных приложений.

7. Графический пакет Turbo Pascal один из самых мощных пакетов такого типа.

8. Turbo Pascal поддерживает модульный принцип программирования, который лежит в основе всех современных технологий разработки программ.

9. Turbo Pascal-это универсальная интегрированная среда, в которую погружен язык. На экране может одновременно присутствовать несколько окон редактирования, окна ввода, подсказки и т.п.

В данное пособие включен ряд программ, подобранных автором специально для учебных целей. В целом пособие соответствует стандартам образования для школ с углубленным изучением программирования и для учебных заведений, занимающихся профессиональной подготовкой пользователей ЭВМ.

Для облегчения ориентации в пособии используются следующие пиктограммы:

|  |  |
| --- | --- |
| Пример решения задачи или типичный фрагмент программы. |  |

|  |  |
| --- | --- |
| Пиктограммой “Тревога!” помечены те места, в которых могут возникнуть неприятные ситуации. |  |

|  |  |
| --- | --- |
| Сберегающие время советы предлагают способы более эффективного использования компьютера. |  |

|  |  |
| --- | --- |
| Задачи для самостоятельного решения. |  |

# Основная часть

## Алфавит языка.

### Символы используемые в именах файлов, подпрограмм, переменных и т.п.

Латинские буквы от A до Z и от a до z, цифры от 0 до 9, знак подчеркивания ( \_ ). В идентификаторах учитывается не более 63 первых символов.

### Разделители.

Пробел, управляющий символ (коды от 0 до 31), комментарии заключенные в фигурные скобки { } или в сложные скобки (\* \*).

### Знаки пунктуации:

[ ]-индексы ряда, элементы множества, размеры строки;

( )-выделение выражений, списка параметров;

''-апострофы для выделения символа или строки;

:= - знак присваивания значения переменной, типизированной константе или функции;

;(точка с запятой)- конец предложения в Паскале;

:- отделение переменной или типизированной константы от типа, а так же метки от оператора;

= отделение имени типа от описания типа или константы от ее значения;

, - разделение элементов списка;

.. - разделение границ диапазона;

. - конец программы, отделение целой части от дробной, полей в записи, идентификатора от имени модуля или объекта;

^ - тип указателя;

@- адрес переменной, типизированной константы, подпрограммы, метода;

#- символ по его коду;

$- директива компилятору или знак шестнадцатиричного числа;

### Знаки операций.

 + сложение;

 - вычитание;

 \* умножение;

 / деление;

 div деление целых чисел;

 mod остаток от деления целых чисел;

 not логическое отрицание;

 and логическое и;

 or логическое или;

 xor логическое исключающее или;

 shl сдвиг битов влево;

 shr сдвиг битов вправо;

 = равно;

 < меньше;

 > больше;

 <> не равно;

 <= не больше;

 >= не меньше;

#### Порядок выполнения операций.

 1. Вычисления в круглых скобках ()

 2. Значения функций Функции

 3. Унарные операции @,not,+,-

 4. "Умножение" \*,/,div, mod, and, shl, shr

 5. "Сложение" +, -, or, xor

 6. Отношения =,<,>,<=,>=,<>,in

### Зарезервированные слова.

 absolute признак абсолютной переменной

 and логическое И

 array описание ряда

 asm ассемблерный блок

 assembler программа на ассемблере

 begin начало

 case выбор

 const константа

 constructor конструктор объекта

 destructor деструктор объекта

 div деление нацело

 do выполнять

 downto от большего к меньшему

 else иначе

 end конец

 external директива внешней подпрограммы

 far форматирование дальнего адреса

 file файловый тип

 for цикл с параметром

 forward подпрограмма описана позже

 function подпрограмма-функция

 goto переход на метку

 if если

 implementation исполнительная часть модуля

 in проверка на принадлежность множеству

 inherited наследуемый метод

 inline подключение программы на ассемблере

 interface связь с модулем

 interrupt обработка прерываний

 label метка

 mod остаток от деления

 near форматирование ближнего адреса

 nil пустая ссылка

 not логическое НЕ

 object объект

 of тип компонента

 or логическое ИЛИ

 packed упакованный массив

 private приватная секция объекта

 procedure подпрограмма-процедура

 program программа

 public доступная извне секция объекта

 record запись

 repeat начало цикла с постусловием

 set множество

 shr сдвиг битов вправо

 shl сдвиг битов влево

 string строка

 then то

 to до

 type тип

 unit модуль

 until конец цикла с постусловием

 uses вызов модуля

 var описание переменных

 virtual виртуальный метод

 while цикл с предусловием

 with начало работы над записями

 xor логическое ИСКЛЮЧАЮЩЕЕ ИЛИ

### Неиспользуемые символы.

Русский алфавит, &, % и некоторые другие могут быть использованы только внутри комментариев и строковых величин.

## Структура программы на языке Паскаль.

Программа на языке TURBO PASCAL состоит из *заголовка*, который всегда записывается в самом начале, и *тела программы*. Тело программы, в свою очередь, состоит из шести разделов, располагающихся в следующем порядке:

1. раздел *меток*;
2. раздел *констант*;
3. раздел *типов*;
4. раздел *переменных*;
5. раздел *процедур* и *функций*;
6. раздел *операторов*.

|  |  |
| --- | --- |
| В программе должен обязательно присутствовать раздел операторов, а другие разделы могут и отсутствовать.  |  |

program .......; <имя программы>

uses .....;<открытие библиотек и вызов модулей>

<директивы компилятору>

const ........; <описание констант>

label .......; <описание меток>

type .........; <типы, определяемые пользователем>

var ..........; <описание глобальных переменных>

function <имя>(<параметры>:<типы параметров>):<тип результата>;

<локальное описание>

<тело функции>

procedure <имя>(<параметр>:<типы параметров>);

<локальное описание>

<тело процедуры>

{main file}

begin {начало главной программы}

... {тело программы, обязательная часть}

end. {main file}

## Типы данных.

### Простые типы данных.

#### Целые типы.

|  |  |  |  |
| --- | --- | --- | --- |
| Тип | представление | границы | примечание |
| Byte | байт | от 0 до 255 | упорядочен |
| Shortint | байт | от -128 до 127 | упорядочен |
| Word | 2 байта | от 0 до 65535 | упорядочен |
| Integer | 2 байта | от -32768 до 32767 | упорядочен |
| Longint | 4 байта | от -215 до 215-1 | не упорядочен |

#### Логический тип boolean.

Однобайтовый, упорядоченный принимает два значения:

True и False, при этом False< True.

Логические операции имеют следующую таблицу истинности:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| А | B | NOT(A) | (A) AND (B) | (A) OR (B) |
| TRUE | TRUE | FALSE | TRUE | TRUE |
| TRUE | FALSE | FALSE | FALSE | TRUE |
| FALSE | TRUE | TRUE | FALSE | TRUE |
| FALSE | FALSE | TRUE | FALSE | FALSE |
|  Примеры:x>56; NOT(a<d); (y≥x) AND (n>789); (k≠l) OR (g≤n). |  |

Значения логических выражений относятся к предопределенному булеву типу с идентификатором: boolean. Этот тип имеет всего два значения, которые обозначаются с помощью предопределенных констант TRUE и FALSE. Для этих значений выполняются следующие соотношения: TRUE > FALSE ; Ord(FALSE )=0; Ord(TRUE)=1. Знаки ≥, ≤ , ≠ на компьютере записываются соответственно >=, <=, <>.

|  |  |
| --- | --- |
| Определите значение логических выражений: при a=-2, х=6.a>=x; 5.1\*a\*x<0; (x\*x-5\*x+6)<0; 48<>3\*x\*a\*a\*(-2);Not( a<x); NOT(-9\*a=3\*x); NOT(78<a/x); NOT(a=0);(2/a>=0) OR (78/x<>13); (a>=2\*x) OR (x<>-3\*a); (1=2) OR (2>5)(2/a>=0) AND (78/x<>13); (a>=2\*x) AND (x<>-3\*a); (1=2) AND (2>5) |  |

#### Символьный тип char.

Однобайтовый упорядоченный принимает значения всех допустимых символов.

#### Перечисляемый тип

type <имя типа>=<идентификатор 1, идентификатор 2 .. >;

#### Отрезочный тип

type <имя типа>=<минимальное значение.. максимальное значение>;

#### Вещественные типы

|  |  |  |  |
| --- | --- | --- | --- |
| Тип | представление | границы | значащих цифр |
| Real | 6 байт | 2.9\*10-39..1.7\*1038 | 11-12 |
| Single | 4 байт | 1.5\*10-45..3.4\*1038 | 7 - 8 |
| Double | 8 байт | 5\*10-324..1.7\*10308 | 15-16 |
| Extended | 10 байт | 3.4\*10-4932..1.1\*104932 | 19-20 |
| Comp | 8 байт | -263+1..263 -1 | 19-20 |

### Структурированные типы.

#### Одномерные ряды.

В Pascal'е существует механизм обозначения большого числа однородных объектов - это ряды или таблицы( в Basic'е-массивы). Их описание делается с помощью слова array, например:

tabl:array[1..10] of integer; в ОЗУ будет выделено 10 участков по 2 байта каждый, для записи 10-и упорядоченных целых чисел. Если ряды будут применяться в подпрограммах с параметрами, то их нужно описывать с помощью своих типов, например:

const lim=10;

type tabl=array[1..lim] of integer;

var tablica:tabl;

В таблицах индексами могут служить не только натуральные числа, но и любые величины упорядоченного типа, например, при решении задачи о зарплате 5-и конкретных лиц можно так определить массив:

type familiya=(Iwanoff,Petroff,Sidoroff,Smirnoff,Egoroff);

 zarplata=array[familiya] of real;

var dochod:zarplata;

#### n-мерные.

Примером двумерной таблицы может служить прямоугольная таблица:

|  |  |  |  |
| --- | --- | --- | --- |
| a11 | a12 | ... | a1n |
| a21 | a22 | ... | a2n |
| . | . | . |  |
| am1 | am2 | ... | amn |

Данная таблица состоит из m строк и n столбцов. Каждый элемент имеет свой индекс состоящий из двух чисел. Первое число индекса это номер строки, второе число - номер столбца. При чтении идентификатора элемента таблицы надо указывать каждый индекс. Например: а[1,1] - читается “*а один, один*”; а[2,3] - “*а два, три*” и т.д.

Идентификатор двумерной таблицы состоит из имени совпадающем с именем таблицы и индекса, заключенного в квадратные скобки. Элементы индекса записываются через запятую. Например: a[10,2], taml[n,k].

Многомерные ряды могут быть определены двумя способами:

1. a:array[1..4,1..5] of char; - будет выделено 20 (4\*5)участков памяти, по 1 байту каждый для возможного занесения 20 символов.

2. Каждый ряд рассматривается, как "ряд рядов" (при этом число вложений не ограничено), например, для двумерного массива:

type studexames=array[1..4] of integer;

var grade:array[1..10] of studexames;

### Строковый тип string.

Turbo Pascal, в отличие от стандартного Pascal'я, предусматривает возможность работы с отдельными строками заданной длины, например: string[n], где n-целое число, это строка из n символов, занимающая в памяти n+1 байт. String можно рассматривать как ряд символов. В нулевом байте хранится длина строки.

### Запись, тип record.

Включает в себя ряд компонент, называемых полями, которые могут быть различных типов. Например:

 type

 Complex=record

 Re:real;

 Im:real;

 End;

 {подготовлена запись для обработки комплексных чисел}

Доступ к полям записи осуществляется указанием имени переменной и имени поля, записываемого через точку. Для того чтобы не выписывать каждый раз имя записи, применяют оператор With <имя записи> do <оператор>;

### Множество, тип set.

Тип множество содержит отдельные элементы, количество которых не может превышать 255. Проверка на принадлежность множеству осуществляется оператором in. Допустимые операции над множеством: объединение, разность, пересечение, проверка на эквивалентность, проверка на подмножество.

### Файловый тип.

Под файлом подразумевается любой упорядоченный список, предназначенный для хранения на внешнем носителе информации. В Паскале различают три типа файлов.

#### Типизированный файл

 type <имя типа>= file of <тип>;

 var <имя файловой переменной>:<имя типа>;

#### Текстовые файлы

 var F:text; {F-имя файловой переменной}

#### Не типизированные файлы

Могут состоять из разнотипных компонент. В этом случае работа осуществляется на физическом уровне

 var F:file;

##  Понятие переменной.

Переменную определяют три основных характеристики:

### Имя переменной.

Это адрес физического участка памяти, в котором хранится значение (одной!) переменной. Имена переменных могут состоять из любых букв латинского алфавита, знака подчеркивания ( \_ ), и цифр (два последних не должны быть первым символом имени переменной), всего до 63 символов. Это же справедливо и для имени программы, но в собственно имени до 8 символов, в расширении до трех.

Рассмотрим задачу: Вывести на экран 1-е n натуральных чисел через " - " их квадраты.

В нашей задаче всего три переменных: натуральное число, его квадрат и ограничитель. Можно бы было им дать имена x, y и z, но лучше использовать принципы "мнемотехники", дабы не ломать голову по прошествии некоторого времени: “И что же это я обозначил буквой x?”. В нашем случае возможно chislo, quchisla и konec.

### Типы переменных

 (описаны в пункте 3.)

Объявив в программе var chislo,quchisla,konec:integer;,получим три участка памяти по 2 байта каждый.

### Значение переменной.

Если в программе встретиться, например: chislo:=5;, то в соответствующий участок будет занесено значение 5, если же после этого встретится chislo:=chislo+2 , то значение 5 будет утеряно, а запишется значение 7. Наконец, приступим к написанию программы (построение математических моделей и алгоритмов не является целью этой работы). Внутри { } записаны комментарии.

 program primer1;

{распечатка квадратов 1-х n натуральных чисел}

uses crt;

 var chislo,quchisla,konec:integer;

begin

 clrscr;

 write('До какого числа распечатать? konec=');

 readln(konec);

 for chislo:=1 to konec do

 begin

 quchisla:=chislo\*chislo; { можно quchisla:=sqr(chislo)}

 writeln(chislo,'-',quchisla);

 end; { знак; не обязателен}

end.

## Константы.

Константами называются параметры программы, значения которых не меняются в процессе выполнения программы.

### Целые.

 const lim=20;

### Вещественные.

 const x=0.5;

 y=-1.13;

### Строковые и символьные.

 const s='TURBO';

 a='''';

 t='line1'#13#10'line2';

 r=^W^J;

### Константные выражения.

Выражения, которые могут быть вычислены на стадии компиляции, без запуска программы. Они являются частным случаем выражения и могут состоять из: констант, знаков операций, круглых скобок и некоторых функций(abs, chr, hi, high, length, lo, low, odd, ord, pi, pred, ptr, round, sizeof, succ, swap, trunc).

### Типизированные константы.

Это инициализированные переменные, которые используются наравне с другими переменными.

 const

 <имя константы>:<тип константы>=<значение константы>;

 Например:

 const x:integer=455;

## Операторы

### Простые операторы.

Операторы не содержащие других операторов называются простыми.

#### Оператор присваивания

 <имя переменной>:=<выражение или значение>;

При выполнении оператора присваивания по адресу, определенному именем переменной вносится новое значение переменной, при этом старое значение теряется.

При выполнении этого оператора значение переменной, имя которой стоит слева от знака := , будет изменено на значение выражения стоящего в правой части.

|  |  |  |
| --- | --- | --- |
| Оператор присваивания | Результат присваивания |  |
| x:=5+6\*3;a:=x;dd:= x=aa:=a-7;dd:= a>x;a:=sqrt(a);x:=x+a; | x=23a=23dd=TRUEa=16dd=FALSEa=4x=27 |  |
|  |  |  |
| Обратите внимание на то, что оператор присваивания это не математическое тождество и не уравнение ! |  |

#### Оператор безусловного перехода goto.

Переход на метку (label), которая может быть представлена числом в интервале 0..9999 или обычным идентификатором. Метка должна быть объявлена в заголовке программы, если она глобальная, или в заголовке подпрограммы, если она локальная.

#### Пустой оператор.

В программе никак не описывается и может быть использован, например, для безусловного перехода.

### Структурированные операторы.

#### Составной оператор.

Совокупность последовательно выполняемых операторов, заключенных в операторные скобки ,begin..end;

 begin

 <оператор 1>;

 <оператор 2>;

 ............

 <оператор n>;

 end;

#### Ветвление.

Широко используется при решении многих задач. В языке Pascal оформляется с помощью следующей конструкции:

 if <условие> then

 begin

 <серия команд>

 end

 [ else

 begin

 <серия альтернативных команд>

 end];

Полная форма команды ветвления и неполная форма команды ветвления


#### Цикл с параметром.

 for i:=a to b do

 begin

 <тело цикла>

 end;

при этом переменные a и b<или константы> только дискретного типа. Если а<b, то слово to нужно заменить на downto.


#### Цикл с предусловием.

 while <условие> do

 begin

 <тело цикла, в который нужно включить изменение параметров, по которым идет проверка условия>

 end;


#### Цикл с постусловием.

 repeat

 <тело цикла>

 until <условие>;

Команды, содержащиеся в теле цикла с постусловием, обязательно выполняются хотя бы один раз. Существенным отличием от двух предыдущих способов является также и то, что повторный входи в тело цикла осуществляется только при невыполнении условия.

#### Команда выбора(case).

При решении многих задач бывает нужно осуществить выбор по большому числу условий, например: комментарий к оценке; выбор изменю и т.д. Pascal предоставляет такую возможность с помощью оператора case. Возможно сделать это и с помощью вложенных команд ветвления, но уже при выборе большем, чем из двух, программа стает трудно читаемой. Оформляется оператор case следующим образом:

 var <имя переменной>: <дискретный тип>;

 ...

 ... case <имя переменной> of

 <допустимые значения переменной>: <1-я серия команд>;

 <допустимые значения переменной>: <2-я серия команд>;

 <допустимые значения переменной>: <3-я серия команд>;

 .....................................................

 <допустимые значения переменной>: <n-я серия команд>;

 [ else <n+1-я серия команд>;]

 end;

#### Процедуры без параметров досрочного выхода из цикла.

 6.2.7.1. Break позволяет досрочно выйти из цикла, не дожидаясь выполнения условия выхода.

 6.2.7.2. Continue начинает новую итерацию цикла, даже если предыдущая не завершена.

## Подпрограммы. Системный подход к программированию в системе TP.

### Декомпозиция.

Полученную задачу считаем глобальной (0-й уровень). Если задача проста, то можно сразу приступить к написанию программы. В противном случае провести разбивку глобальной задачи на ряд локальных(1-й уровень). Простые задачи 1-го уровня программируются, а сложные снова разбиваются на локальные(2-й уровень)... Когда прекращать детализацию алгоритма? Часто подробная схема алгоритма оказывается менее полезной, чем более ранний вариант декомпозиции, т.к. обилие деталей может затмить общую структуру программы. Если работа над алгоритмом ведется в виде блок-схемы, то лучше для каждой локальной задачи выделить отдельный лист и при программировании каждую локальную задачу рассматривать как подпрограмму, которая в свою очередь может вызывать подпрограммы, являющиеся решением локальных задач более низкого уровня.

Поскольку построению алгоритмов с помощью блок-схем уделено достаточное внимание в многочисленной литературе, да автор и не ставит перед собой задачи обучения основам алгоритмизации, все пояснения к написанию программ будут сделаны в комментариях или с помощью "псевдокодов".

*Рекомендации по структурному программированию:*

#### Комментарии, занимающие несколько строк, лучше оформлять в виде отдельного блока, например:

 {------------------------------------------}

 {Эта программа находит корни квадратного }

 { уравнения по введенным с клавиатуры зна- }

 { чения коэффициентов a, b, c }

 {------------------------------------------}

#### Как говорилось выше, переменным лучше давать мнемонические имена, размещая рядом еще и комментарий, например:

 var a,b,c :real; {коэффициенты ур-ия a\*+b\*x+c=0}

 diskriminant:real; {дискриминант кв. ур-ия=-4\*a\*c }

 sqdiskr :real; {корень квадратный из дискриминанта}

#### Для лучшей наглядности в каждой строке можно давать только одну команду. Отдельные блоки лучше отделять пустыми строками. Соответствующие begin и end надо располагать, начиная в одном столбце. Команду, следующую за begin, for..., while...,if..., и т.д. размещайте со сдвигом на колонку. Пункты then и else следует выравнивать относительно друг друга.

#### Данные, предназначенные для вывода, должны быть учтены по принципу "необходимо и достаточно". На печати придерживайтесь разумных форматов, организуйте ряды, столбцы, разграфление.

#### Если вами создана программа, достойная тиражирования, то грамотно оформляйте документацию. Она должна содержать:

- Формулировку задачи.

- Схему алгоритма.

- Листинг программы с комментариями.

- Инструкцию по применению.

- Полный набор тестов, позволяющий проверить все возможные ситуации, которые могут возникнуть при работе программы.

### Подпрограммы-функции.

#### Стандартные будут описаны в приложении.

#### Функции, определяемые пользователем.

Структура определена следующим образом:

 function <имя функции>(<формальный параметр>:<тип формального параметра>):<тип результата>;

 var <локальные переменные>:<типы локальных переменных>;

 begin

 ... <тело ... функции>

 end;

Вызов функции осуществляется по ее имени и формальным параметрам, например:

 ...

 function prim1(x:real):real;

 begin

 prim1:=x/5;

 end;

 ...

 {главная программа}

 begin

 ...

 n:=527;

 w:=prim1(n);

 writeln(w:6:2);

 ...

 end.

В результате обработки функции будет выведено 105.40

### Подпрограммы-процедуры.

Подпрограмма-функция способна выдавать только одно значение. Во многих же случаях требуется получить несколько значений или несколько действий. Основным инструментом для этого являются подпрограммы-процедуры. Также, как и функции они должны быть описаны до главной программы. Если же вызов подпрограммы идет из другой подпрограммы, то вызываемая должна быть описана раньше. Этого можно и избежать, применяя служебное слово forward. Подпрограммы-процедуры условно можно классифицировать по трем признакам.

#### Процедуры без параметров - это самостоятельные подпрограммы, не связанные с переменными главной программы, например, для оформления заголовка таблицы, выводимой главной программой. Пусть нужно получить следующий заголовок:

 Количество медалистов, окончивших школу за последние 10 лет

 Год Фамилия И.О. Медаль

program medali;

{печать списка медалистов за послед. 10 лет и...........}

uses .........; {вызов модулей и библиотек}

const ........; {описание констант}

type .........; {типы, определяемые пользователем}

var ..........; {описание глобальных переменных}

function ............{описание функций}

procedure .......... {описание процедур}

procedure printzagolowok;

 begin

 writeln('Количество медалистов, окончивших школу за последние 10 лет':60);

 writeln;

 writeln('Годы Фамилия И.О. Медаль');

 writeln;

 end;{printzagolowok}

{main file}

begin

...

...

printzagolowok;

...

end. {main file}

#### Процедуры с формальными параметрами.

Применяются в тех случаях, когда нужно выполнить ряд операций, не передавая значений в главную программу, но используя значения параметров из главной программы или подпрограмм, например, поиск и печать наименьшего из 2-х чисел.

program chisla;

var a,b,c,d:integer;

procedure findprintmin(x,y:integer);

 begin

 if x>y

 then writeln(у, ' меньше ',x)

 else writeln(x,' меньше ',y)

 end;{findprintmin}

 {main file}

begin

 a:=5; b:=7;

 findprintmin(a,b);

 c:=12; d:=16;

 findprintmin(c+2,d-7);

 findprintmin(a\*b,b+d);

end.

В результате будет отпечатано:

5 меньше 7

9 меньше 14

23 меньше 35

#### Подпрограмма, изменяющая свои параметры.

(Подпрограмма с var параметрами).

Очень часто бывает необходимо параметры, обработанные процедурой, поместить по новым адресам и сделать их доступными для главной программы, например, видоизменим задачу из п. 7.3.2: найти наименьшее и наибольшее из 2-х чисел и, не отпечатывая, записать их под другими именами, т.е. записать их по другим адресам, доступным главной программе.

program minmax;

{нахождение наименьшего и наибольшего из 2-х чисел}

var a,b,c,d:real;

procedure findminmax(x,y:real;var max,min:real);

 begin

 if x<y then begin

 min:=x;

 max:=y

 end {для then}

 else begin

 min:=y;

 max:=x;

 end {для else}

 end; {findminmax}

{main file}

begin

 clrscr;

 writeln('введите два числа');

 readln(a,b);

 findminmax(a,b,c,d); {здесь параметры c и d-только переменные}

 writeln('наибольшее-',c:6:2); {задан формат печати}

 writeln('наименьшее-',d:6:2); {задан формат печати}

end.{main file}

Еще раз выделим применение var-параметров. Ни один из параметров функции не должен быть var-параметром, т.к. ни один из них не должен получать новое значение в теле функции. В процедурах некоторым параметрам следует приписывать атрибут var, если в ходе выполнения процедуры эти параметры должны получить новые значения. Тогда для передачи их значений из главной программы в процедуру и обратно компилятор Pascal'я применит вызов по ссылке. Для остальных параметров действует механизм вызова по значению.

 В заключение рассмотрим классическую задачу:

 Решить квадратное уравнение a\*x\*x+b\*x+c=0.

 ------------------------------------------

program kwur;

{===================================================}

{ по введенным коэффициентам a,b,c находятся действительные}

{ корни уравнения ax2+bx+c=0 или печатается сообщение о их }

{ не существовании }

{===================================================}

uses crt;

var a,b,c,d:real;{коэффициенты и дискриминант кв. ур-ия}

 x1,x2 :real; {неравные корни}

 z :integer; {знак дискриминанта}

 ch:char;

function signum(y:real):integer;

 begin

 if y<0 then signum:=-1;

 if y=0 then signum:=0;

 if y>0 then signum:=1;

 end; {signum}

procedure negativ;

 begin

 writeln('Действительных корней нет! ');

 end; {negativ}

procedure nol(e,f:real);

 var x:real; {значение равных корней}

 begin

 x:=-f/(2\*e);

 writeln('x1=x2=',x:6:3);

 end; {nol}

procedure positiv(g,h,i:real;

 var j,k:real); {неравные корни}

 begin

 j:=(-h-sqrt(i))/(2\*g);

 k:=(-h+sqrt(i))/(2\*g);

 end; {positiv}

{main file}

begin

 TextColor(14);

 TextBackground(1);

 clrscr;

 writeln('Ведите коэффициенты квадратного уравнения ');

 readln(a,b,c);

 d:=b\*b-4\*(a\*c);

 z:=signum(d);

 case z of

 -1: negativ;

 0: nol(a,b);

 1: begin

 positiv(a,b,d,x1,x2);

 writeln('x1=',x1:6:3,' x2=',x2:6:3);

 end

 end; {case}

 ch:=ReadKey;

end. {main file}

## Процедурные типы.

Процедуры и функции можно рассматривать как некоторые параметры и можно использовать переменные, принимающие значения процедуры или функции. Объявляется процедурный тип также, как и заголовок подпрограммы. Например:

type proc1=procedure;

 proc2=procedure(var x, y:integer);

 func1=function(x:real):real;

var p1:proc1;

 p2:proc2;

 f1:func1;

Поле этого переменной p1 может быть присвоено значение любой процедуры без параметров, переменной p2- значение любой процедуры с двумя целыми параметрами по ссылке, переменной f1- значение любой функции с одним вещественным параметром.

На процедурные типы накладываются следующие ограничения:

- компиляция проходит с ключем {$F+} или иметь директиву Far для получения полного адреса подпрограммы;

- они не должны быть стандартными процедурами и функциями;

- они не должны объявляться внутри других процедур и функций;

- они не должны быть типа InLine или InterRupt

Процедурная переменная занимает 4 байта памяти. Процедурные переменные можно использовать также как и переменные других типов: в выражениях, в виде оператора, как компонента другой более сложной переменной, как передаваемый в подпрограмму параметр.

## Совместимости преобразования типов данных.

### Идентичность типов.

Типы считаются идентичными в двух случаях: Т1 и Т2 один и тот же идентификатор типа, один тип объявлен эквивалентным другому, например: Type T1=integer;

 T2=T1;

 T3=integer;

 В этом примере Т1,Т2,Т3,integer - идентичные типы.

### Совместимость типов.

Два типа Т1 и Т2 совместимы в следующих случаях:

- Т1 и Т2 идентичны;

- Т1 и Т2 вещественные типы;

- T1 и Т2 целые числа;

- один тип вещественный, другой - целый;

- один тип является диапазоном другого;

- оба типа являются диапазонами третьего типа;

- оба типа являются множествами с совместимыми базовыми типами;

- один тип является строкой, а другой - строкой или символом;

- один тип Pointer, а другой - любой тип-указатель;

- оба типа процедурные по сходным параметрам.

### Совместимость для присваивания.

Если значения объекта типа Т2 присваивается объекту типа Т1, то это возможно в следующих случаях:

- Т1 и Т2 идентичны;

- Т1 и Т2 совместимые порядковые типы и значение Т2 не выходит за пределы Т1;

- Т1 и Т2 вещественные типы и значение Т2 не выходит за пределы Т1;

- Т1 вещественный тип, а Т2 - целый;

- Т1 и Т2 - строки;

- Т1 - строка, а Т2 - символ;

- Т2 является подмножеством Т1;

- Т1 и Т2 - совместимые указатели;

- Т1 и Т2 - совместимые процедурные типы;

- Т1 и Т2 - объектные типы и Т2 потомок Т1;

## Директивы подпрограмм.

#### Forward

Если подпрограмма вызывает подпрограмму описанную ниже, то необходимо применять директиву Forward с указанием заголовка ниже описываемой процедуры.

#### Far & Near.

Обычно компилятор Pascal автоматически выбирает адресацию к подпрограмме, но в случае использования процедурных типов нужно использовать директиву Far для выбора "дальней" адресации, что равносильно ключу компилятора {$F+} .

#### External.

Позволяет использовать подпрограммы написанные на Assembler и скомпилированные отдельно. Основная программа компилируется с использованием ключа компилятора {$L<имя файла с расширением.OBJ>}.

#### Assembler.

Позволяет создать подпрограмму полностью написанную на языке Assembler.

#### InLine

Позволяет написать подпрограмму в машинных кодах.

#### InterRupt

Предназначена для написания процедур обрабатывающих прерывания

## Внешние устройства в качестве файлов.

 CON-консоль, т.е. дисплей

 LPT1, LPT2, LPT3- печатающее устройство, можно заменить на Lst, параллельные порты PRN-синоним LPT1

 COM1, COM2 - последовательные порты

 AUX-синоним COM1

 NUL-фиктивное внешнее устройство

## Модули

#### Заголовок модуля

 unit <имя модуля>;- модуль должен быть помещен в файл с расширением .pas и с тем же именем

#### Интерфейс модуля.

Через него осуществляется взаимодействие основной программы с модулем. В интерфейсе должны быть указаны константы, типы, переменные, процедуры и функции, которые используются в основной программе при вызове этого модуля. Начало со слова interface, затем после слов uses указываются имена модулей, которые используются данным модулем. Объявление процедур может содержать директиву inline. Если этой директивы нет, то в разделе объявления функций и процедур указывают лишь заголовки подпрограмм. Сами подпрограммы приводят в исполнительной части. Недопустимо циклическое обращение модулей друг к другу.

#### Исполнительная часть модуля.

Должна включать в себя все подпрограммы модуля, а так же локальные метки, константы, типы и переменные. Начинается со слова implementation, затем после слова uses указываются имена модулей, которые используются данной исполнительной частью. Допустимо циклическое обращение модулей друг к другу

## Стандартные модули.

### System.

Основная библиотека среды.

**ПРОЦЕДУРЫ**

#### Append(var F: text); - открытие текстового файла для до записи в конец

#### Assign(var F; name: string); - связь файловой переменной с внешним файлом

#### BlockRead(var F:file;var buf;N:word [; var result: word]); - читает компоненты не типизированного файла

#### BlockWrite(var F: file; var buf; N: word [; var result: word]); - запись в не типизированный файл

#### Break; - немедленный выход из цикла

#### ChDir (S: string); - изменение активного каталога

#### Close (var F); - закрытие файла

#### Continue; - очередная итерация цикла

#### Dec(var X[; N: longint]); - уменьшение значения переменной

#### Delete (var S: string; index: integer; count: integer); - удаление подстроки из строки, где index - номер первого удаляемого, count - число удаляемых символов

#### Dispose (var P[, destructor]); - освобождение памяти от динамической переменной

#### Erase (var F); - удаление файла

#### Exclude(var S:set of T; I:T); удаляет из множества S подмножество T

#### Exit; выход из текущего блока

#### FillChar(var X; count: word; value: <порядковый тип>); - заполнение смежных байтов

#### Flush(var F: text); - освобождение буфера текстового файла

#### FreeMem(var P; Size: word); - удаление динамической переменной заданного размера

#### GetDir(d: byte; var S: string); - перемещение из каталога в строку

#### GetMem(var P; Size: word); - создание новой динамической переменной

#### Halt(ExtCode: word); - остановка программы

#### Inc(var x[; N; Longint]); - увеличение значения X

#### Include(var S: set of T; I: T); - добавить подмножество I к множеству T

#### Insert(S: string; var s1: string; index: word); - вставить подстроку в строку

#### Mark(var P); - фиксация состояния динамической переменной

#### Move(var source, Dest; count: word); - копирование байтов из одной области в другую

#### New(var P[, constructor]); - создание новой динамической переменной

#### Randomize; - включение генератора случайных чисел

#### Read [ln] ( [var F: text] [v1, v2, ..vn]); - чтение из текстовых файлов (почти так же и для типизированных файлов)

#### Release(var P); - возвращает динамическую память в состояние, определяемое переменной P

#### Rename(var F; newname: string); - изменение имени файла на диске

#### Reset(var F[; Size: word] ); - открытие существующего файла

#### Rewrite(var F[; Size: word]); - открытие нового файла

#### RmDir(S: string); - удаление пустого каталога

#### RunError(ErrorCode: word); - стоп с выдачей сообщения

#### Seek(var F; Nom: longint); - настройка на требуемую компоненту файла

#### SetTextBuf(var F: text; var Buf; size: word); - назначение буфера ввода- вывода для текстового файла

#### Str(X; var S: string); - преобразование числа в символы

#### Truncate(var F); - удаление части файла от текущей позиции до конца

#### Val(S: string; var V; var Code: integer); - преобразование символьного представления числа в двоичную форму

#### Write [ln] ([var F: text; ] [v1, v2, ..vn]); - запись в текстовый файл (почти так же с типизированными файлами)

**ФУНКЦИИ**

#### Abs(x: NUM): NUM; - модуль числа

#### Addr(var x): pointer; - возвращает адрес аргумента

#### ArcTan(x: real): real; - арктангенс

#### Assignet(var P): boolean; - проверка nil

#### Chr(x: byte): char; - символ по коду

#### Concat(S1, ..SN: string): string; - сложение строк

#### Copy(S: string; index, count: iteger): string; - вырезка из строки

#### Cos(x: real): real; - косинус

#### Cseg: word; - значение регистра CS

#### Dseg:word; - Значение регистра DS

#### Eof(var F): boolean; - проверка на конец файла

#### Eoln(var F: text)boolean; - проверка на конец строки

#### Exp(x: real): real; - экспонента

#### FilePos(var F): longint; - номер текущей компоненты

#### FileSize(var F): longint; - размер файла

#### Frac(x: real): real; - дробная часть числа

#### Hi(x: integer): byte; - старший байт любого целого числа

#### High(x: <тип>): word; - максимальное значение

#### Int(x: real): real; - целая часть числа

#### IOResult: word; - сообщения об ошибках

#### Length(S: string): byte; - длинна строки

#### Ln(x: real): real; - натуральный логарифм

#### Lo(x: <целый тип>): byte; - младший байт

#### Low(x: <тип>): word; - минимальное значение

#### MaxAvail: longint; - размер макс. свободной области динамической памяти

#### MemAvail: longint; - размер всех свободных областей динамической памяти

#### New(<тип- указатель>, [constructor]): <тип указатель>; создает новую динамическую переменную

####  Odd(x: longint): boolean; - проверка на четность

####  Ofs(x): word; - смещение сегмента аргумента

####  Ord(x: <порядковый тип>): longint; - номер элемента

####  ParamCount: word; - число параметров в командной строке

####  ParamStr(Index: word): string; - указанный параметр командной строки

#### Pi: real; - 3.1415926535897932385...

#### Pos(s1, s2: string): byte; - место вхождения s1 в s2

#### Pred(x: <порядковый тип>): <порядковый тип>; - предыдущий элемент

#### Ptr(Seg.Ofs: word): Pointer; - преобразование адреса в указатель

#### Random(X: word): <число>; - случайное число на [0..x]

#### Round(x: real): longint; - округление

#### SeekEof(var F: Text): boolean; - фиксация конца файла

#### SeekEoln(var F: Text): boolean; - фиксация конца строки

#### Seg(x): word; - адрес сегмента аргумента

#### Sin(X: real): real; - синус

#### SizeOff(x): word; - аргумент в байтах

#### SPtr: word; - значение в регистре SP

#### Sqr(x: real): real; - квадрат числа

#### Sqrt(x: real): real; - корень квадратный

#### SSeg: word; - значение регистра SS

#### Succ(x: <порядковый тип>): <порядковый тип>; - последующий элемент

#### Swap(x: <целый тип>): <целый тип>; - меняет местами младшие и старшие байты

#### Trunc(x: real): longint; - выделение целой части

#### UpCase(ch: CHAR): char; - преобразование строчного в заглавный

### Модуль Strings

**ПРОЦЕДУРЫ**

#### StrDispose(str: PChar); - удаление строки из динамической памяти

**ФУНКЦИИ**

#### StrCat(d, s: PChar): PChar; - подсоединение к концу первой строки

#### StrComp(st1, st2: PChar): integer; - сравнение строк

#### StrCopy(d, s: PChar): PChar; - копирование второй строки в первую

#### StrECopy(d, s: PChar): PChar; - копирует вторую в первую, указатель на конце строки

#### StrEnd(st: PChar): PChar; - указатель в конец строки

#### StrIComp(st1, st2: PChar): integer; - сравнение строк без учета регистра

#### StrLCat(d, s: PChar; MaxLen: word): PChar; - объединение строк

#### StrLen(Str: PChar): word; - размер строки

#### StrLower(st: PChar): PChar; - прописные символы - в строчные

#### StrMove(d, s: PChar; c: word): PChar; - копирование символов из одной строки в другую

#### StrNew(st: PChar): PChar; - создание строки в динамической памяти

#### StrPas(St: PChar): string; - преобразование ASCIIZ- строки в строку типа string

#### StrPCopy(d: PChar; s: string): PChar; - преобразование строки string в строку dest

#### StrPos(st1, st2: PChar): PChar; - номер вхождения второй строки в первую

#### StrRScan(st: PChar; Chr: char): PChar; - вхождение второй строки в первую

#### StrUpper(st: PChar): PChar; - строчные в прописные

### Модуль CRT

**ПРОЦЕДУРЫ**

#### AssignCrt(var F: Text); - связывает текстовый файл с CRT

#### ClrEol; - удаляет все символы от курсора до конца строки

#### ClrScr; - очищает окно, перемещает курсор в левый верхний угол

#### Delay(Ms: Word); - задает выдержку времени в миллисекундах

#### DelLine; - удаляет строку в позиции курсора

#### GotoXY(X, Y: byte); - перемещение курсора

#### HighVideo; - высокая яркость экрана

#### InsLine; - вставляет пустую строку

#### LowVideo; - малая яркость

#### NormVideo; - первоначальные цвета

#### NoSound; - отключает звук

#### Sound(H: word); - звук с частотой Н

#### TextBackGround(c: byte); - цвет фона

#### TextColor(c: byte); - цвет символа

#### TextMode(m: word); - установка текстового режима

#### Window(x1, y1, x2, y2: byte); - установка окна

**ФУНКЦИИ**

#### KeyPressed: boolean; - проверка буфера на нажатие клавиши

#### ReadKey: char; - опрос клавиатуры

#### WhereX: byte; - текущая координата Х

#### WhereY: byte; - текущая координата Y

### Модуль GRAPH

*Стандартные типы*

|  |  |  |
| --- | --- | --- |
| **Имя** | **Определение** | **Комментарий** |
| PaletteTypeЦвета палитры | Record Size: byte;  Colors: array[0..MaxColors] of Shortint; end;  | Количество цветовИспользуемые цвета |
| LineSettingsTypeХарактеристики линии | Record LineStyle: Word;  Pattern: Word;  Thickness: Word; end;  | тип линиишаблон линиитолщина линии |
| TextSettingsTypeХарактеристики текста | Record Font: Word;  Direction: Word;  CharSize: Word;  Horiz: Word;  Vert: Word; end;  | тип шрифтанаправление текстаразмер символовгор. выравниваниеверт. выравнивание |
| FillSettingsTypeОрнамент и цвет заполнения | Record Pattern: Word;  Color: Word; end;  | шаблонцвет заполнения |
| **Имя** | **Определение** | **Комментарий** |
| FillPaletteTypeЗадаваемый пользователем орнамент заполнения | array[1..8] of byte;  |  |
| PointTypeКоординаты точки | Record X, Y: integer; end;  |  |
| ViewPortTypeОкно вывода графической информации | Record X1, Y1, X2, Y2: integer;  Clip: boolean; end;  | Координаты окнаПризнак усечения |
| ArcCordsTypДанные о дугеe | Record X, Y: integer;  Xstart, Ystart: integer;  Xend, Yend: integer; end;  | центр дугиначало дугиконец дуги |

*Коды типов линий для процедур GetLineStyle, SetLineStyle, GetLineSettings.*

|  |  |  |
| --- | --- | --- |
| **Имя** | **Значение** | **Назначение** |
| SolidLn | 0 | Сплошная |
| DottedLn | 1 | Пунктирная |
| CenterLn | 2 | Штрихпунктирная |
| DashedLn | 3 | Штриховая |
| UserBitLn | 4 | Заданная пользователем |

*Толщина линий для процедур GetLineStyle, SetLineStyle, GetLineSettings.*

|  |  |  |
| --- | --- | --- |
| **Имя** | **Значение** | **Назначение** |
| NormWidth | 1 | Нормальная |
| ThickWidth | 3 | Толстая |

*Управление отсечением фигур для процедуры SetViewPort*

|  |  |  |
| --- | --- | --- |
| **Имя** | **Значение** | **Назначение** |
| ClipOn | True | Отсекаются не помещающиеся в окне части вмгур |
| ClipOff | False | Не отсекаются не помещающиеся в окне части вмгур |

*Константы изображения параллелепипеда Bar3D*

|  |  |  |
| --- | --- | --- |
| **Имя** | **Значение** | **Назначение** |
| TopOn | True | Закрашивается верхняя грань |
| TopOff | False | Не закрашивается верхняя грань |

*Константы орнамента заполнения для процедур GetFillStyle и SetFillStyle*

|  |  |  |
| --- | --- | --- |
| **Имя** | **Значение** | **Назначение** |
| EmptyFill | 0 | Заполнение цветом фона |
| SolidFill | 1 | Однородное Заполнение цветом |
| LineFill | 2 | Заполнение - - -  |
| LtSlashFill | 3 | Заполнение /// |
| SlashFill | 4 | Заполнение /// толстыми линиями |
| BkSlashFill | 5 | Заполнение \\\ толстыми линиями |
| LtBkSlashFill | 6 | Заполнение \\\ |
| HatchFill | 7 | Заполнениеклеткой |
| XHatchFill | 8 | Заполнениекосой клеткой |
| InterleaveFill | 9 | Заполнение частой сеткой |
| WideDotFill | 10 | Заполнение редкими точками |
| CloseDotFill | 11 | Заполнение частыми точками |
| UserFill | 12 | Заполнение определяется пользователем |

*Характеристики шрифтов для процедур SetTextStyle и GetTextStyle*

|  |  |  |
| --- | --- | --- |
| **Имя** | **Значение** | **Назначение** |
| DefaultFont | 0 | Коды шрифтов |
| TriplexFont | 1 | Коды шрифтов |
| SmaltFont | 2 | Коды шрифтов |
| SansSerifFont | 3 | Коды шрифтов |
| GothicFont | 4 | Коды шрифтов |
| HorizDir | 0 | Горизонтальное расположение строки |
| VertDir | 1 | Вертикальное расположение строки |
| UserCharSize | 0 | Размер шрифта задается пользователем |

**ПРОЦЕДУРЫ**

#### Arc(x, y: integer; st, en, ra: word); - дуга окружности X, Y координаты центра, ST - начало, EN - конец, RA - радиус

#### Bar(x1, y1, x2, y2: integer); - закрашенный прямоугольник

#### Bar3D(x1, y1, x2, y2: integer; de: word; top: boolean); - параллелепипед

#### Circle(x, y: integer; r: word); - окружность

#### ClearDevice; - очистка экрана

#### ClearViewPort; - очистка окна

#### CloseGraph; - закрытие графического режима

#### DetectGraph(var GraphDriver, GraphMode: integer); - проверка оборудования

#### DrawPoly(n: word; var PolyPoints); - контур многоугольника

#### Ellipse(x, y: integer; st, en: word; XR, YR: word); - дуга эллипса

#### FillEllips(x, y: integer; XR, YR: word); - закрашенный эллипс

#### FillPoly(n: word; var PolyPoints); - закрашенный многоугольник

#### FloodFill(x, y: integer; b: word); - закрашивание замкнутого контура в - цвет контура

#### GetArcCoords(var ArcCoord; ArcCoordType); - параметр дуги или эллипса

#### GetAspectRatio(var x, y: word); - число точек на экране

#### GetDefaultPalette(var Pal: PaletteType); - исходная палитра

#### GetFillPalettern(var FillPalettern: FillPaletternType); - Возврат орнамента заполнения

#### GetFillSettings(var FillInfo: FillSettingsType); - возвращение орнамента и цвета

#### GetImage(x1, y1, x2, y2: integer; var BitMap); - сохранение изображения в буфере

#### GetLineSettings(var LineInfo: LineSettingType); - возврат параметров линии

#### GetModeRange(GraphDriver: integer; var LoMode, HiMode: integer); - минимум и максимум графических режимов драйвера

#### GetPalette(var Palette: PaletteType); - возврат текущей палитры и цветов

#### GetTextSettings(var TextInfo: TextSettingsType); - возврат параметра шрифта

#### GetViewSettings(var ViewPort: ViewPortType); - возврат параметров окна

#### GraphDefaults; - сбрасывает графические параметры

#### InitGrsph(var graphdriver, graphmode: integer; driverpath: string); - инициализация графики

#### Line(x1, y1, x2, y2: integer); - отрезок

#### LineRel(dx, dy: integer); - рисование отрезка от текущей точки

#### LineTo(x, y: integer); - рисование отрезка от текущей точки

#### MoveRel(dx, dy: integer); - перемещение указателя от текущей очки

#### MoveTo(x, y: integer); - перемещение указателя

#### OutText(s: string); - вывод текста

#### OutTextXY(x, y: integer; s: string); - вывод текста с указанной позиции

#### PieSlice(x, y: integer; s, e, r: word); - закрашенный сектор круга

#### PutImage(x, y: integer; var BitMap; BitBit: word); - вывод изображения из буфера

#### PutPixel(x, y: integer; c: word); - точка

#### Rectangle(x1, y1, x2, y2: integer); - прямоугольник

#### RestoreCrtMode; - выход в текстовый режим

#### Sector(x, y: integer; s, e, xr, yr: word); - закрашенный сектора эллипса

#### SetActivePage(p: word); - номер активной страницы

#### SetAllPalette(var Palette); - меняет все цвета палитры

#### SetAspectRatio(x, y: word); - изменение масштаба экрана

#### SetBkColor(c: word); - цвет экрана

#### SetColor(c: word); - цвет рисунка

#### SetFillPattern(p: FillPatternType; c: word); - задание орнамента и цвета

#### SetFillStyle(p, c: word); - стандартный орнамент и цвет

#### SetGraphBufSize(b: word); - размер буфера

#### SetGraphMode(n: integer); - графический режим

#### SetLineStyle(l, p, t: word); - параметры линии

#### SetPalette(cn, c: word); - изменение цвета одной компаненты

#### SetRGBPalette(c, r, g, b: integer); - задание цвета в палитре

#### SetTextJustify(h, v: word); - способ выравнивания текста

#### SetTextStyle(f, d, c: word); - тип, направление и размер шрифта

#### SetUserCharSize(mx, dx, my, dy: word); - коэффициенты увеличения шрифта

#### SetViewPort(x1, y1, x2, y2: word; c: boolean); - графическое окно

#### SetVisualPage(p: word); - номер отображаемой страницы

#### SetWriteMode(w: integer); - режим построения линий

**Функции**

#### GetBkColor: word; - цвет фона

#### GetColor: word; - цвет рисунка

#### GetDriverName: string; - имя текущего драйвера

#### GetGraphMode: integer; - номер режима

#### GetMaxColor: word; - максимальный номер палитры

#### GetMaxMode: word; - число режимов драйвера

#### GetMaxX: integer; - максимум Х

#### GetMaxY: integer; - максимум Y

#### GetModeName(MD: word): string; - строка с именем режима

#### GetPaletteSize: word; - количество цветов

#### GetPixel(x, y: integer): word; - цвет точки

#### GetX: integer; - координата Х

#### GetY: integer; - координата Y

#### GraphErrorMsg(c: integer): string; - сообщение об ошибке

#### GraphResult: integer; - код ошибки

#### ImageSize(x1, y1, x2, y2: integer): word; - размер памяти, необходимый для сохранения прямоугольной области

#### InstallUserDriver(Name: string; AutoDetectPtr: pointer): word; - размещает драйвер в таблице BGI

#### InstallUserFont(Name: string): integer; - установка нового шрифта

#### RegisterBgiDriver(d: pointer): integer; - регистрация BGI драйвера

#### RegisterBgiFont(Font: pointer); - регистрация шрифта

#### TextHeight(Text: string): word; - высота строки

#### TextWidth(Text: string): word; - длина строки

### Модуль DOS

**Процедуры:**

#### Exec(Path, Cmd: string); - выполнение программы с параметром

#### FindFirst(Path: string; attr: word; var s: searchrec); - поиск файла

#### FindNext(var s: searchrec); - поиск следующего

#### FSplit(Path: string; var Dir: DirStr; var name: NameStr; var Ext: ExtStr); - разбивает имя файла на 3 части

#### GetBreak(var Break: boolean); - проверка прерывания на Ctrl+Break

#### GetDate(var y, m, d, dw: word); - дата и день недели

#### GetFAttr(var F; var Attr: word); - атрибуты файла

#### GetFTime(var F; var Time: Longint); - дата и время обновления файла

#### GetIntVec(IntNo: byte; var Vector: Pointer); - адрес программы обработки прерывания

#### GetTime(var h, m, s, s100: word); - текущее время

#### GetVerify(var Verify: boolean); - состояние проверки правильности записи

#### Intr(IntNo: byte; var r: Registers); - выполнение прерывания

#### Keep(e: wort); - остановка программы, которая остается в памяти

#### MsDos(r: Registers); - вызов функции DOS

#### PackTime(var DT: Datetime; var Time: Longint); - Упаковывает заданные дату и время.

#### SetCBreak(Break: Boolean); - Устанавливает способ работы с комбинацией Ctrl+Break.

#### SetDate(Year, Month, Day: Word); - Устанавливает текущую дату в операционной системе.

#### SetFAttr(var F; Attr: Word); - Устанавливает атрибуты файла.

#### SetFTime(var F; Time: Longint); - Дата и время обновления файла в упакованном виде

#### SetIntVec(i: byte; vector: pointer); - адрес программы обработки прерывания

#### SetTime(h, m, s, s0: word); - установка текущего времени

#### SetVerify(v: boolean); - установка состояния флага проверки правильности записи на диск

#### SwapVectors; - меняет местами содержимое указателей SaveIntXX модуля System

#### UnPackTime(t: longint; var dt: DateTime); - Распаковка даты и времени

**ФУНКЦИИ**

#### DiskFree(d: word): longint; - свободное место на диске

#### DiskSize(d: word): longint; - объем памяти на текущем диске

#### DosExitCode: word; - код выхода из вычислительного процесса

#### DosVersion: word; - версия DOS

#### EnvCount: integer; - количество строк среды DOS

#### EnvStr(i: integer): string; - указанная строка среды операционной системы

#### FExpand(path: PathStr): PathStr; - расширение имени файла до полного

#### FSearch(path, dirlist: string): string; - поиск файлов в списке каталога

#### GetEnv(EnvVar: string): string; - значение указанной переменной среды DOS

### Модуль WINDOS

**ПРОЦЕДУРЫ**

#### CreateDir(d: string); - создание каталога со спецификацией

#### FindFirst(path: string; a: word; var s: TSearchRec); - поиск в заданном каталоге файла по атрибуту и маске

#### FindNext(var s: TSearchRec); - поиск следующего файла по п.6.2.

#### GetCBreak; - см. п.5.5.

#### GetDate; - см. п.5.6.

#### GetFAttr(var F; var a: word); - возвращает атрибуты файла

#### GetFTime - см. п.5.8.

#### GetIntVec - см. п.5.9.

#### GetTime - см. п.5.10.

#### GetVerify - см. п.5.11.

#### Intr(i: byte; var r: TRegisters); - программное прерывание

#### MsDos(var r: TRegisters) - вызывает функцию ДОС

#### PackTime(var DT: TDateTime; var T: Longint); - упаковка заданных даты и времени

#### RemoveDir(D: string); - уничтожает пустой каталог

#### SetCBreak - см. п.5.16.

#### SetCurDir(d: string); - задает новый текущий каталог

#### SetDate - см. п.5.17.

#### SetFAttr - см. п.5.13.

#### SetFTime - см. п.5.19.

#### SetIntVec - см. п.5.20.

#### SetTime - см. п.5.21.

#### SetVerify - см. п.5.22.

#### UnpackTime - см. п.5.24.

**ФУНКЦИИ**

#### DiskFree - см. п.5.25.

#### DiskSize - см. п.5.26.

#### DosVersion - см. п.5.28.

#### FileExpand(d, n: string): string; - расширяет имя файла, добавляя к нему путь. Результат - указатель на расширенное имя

#### FileSearch(d, n, l: string): string; - поиск файла в списке каталога

#### FileSplit(p, d, n, e: string): word; - разбивка имени файла на 3 части

#### GetArgCount: integer; - число параметров

#### GetArcStr(d: string; i: integer; m: word): string; - возврат указателя на параметр командной строки

#### GetCurDir(d: string; Drive: byte): string; - спецификация текущего каталога

#### GetEnvVar(Name: string): string; - возврат указателя на заданную переменную ДОС

### Модуль OVERLAY

**Процедуры**

#### OvrClearBuf; - очистка буфера

#### OvrInit(FileName: string); - открытие оверлейного файла

#### OvrInitEMS; - оверлейный файл в расширенную память

#### OvrSetBuf(s: LongInt); - размер оверлейного буфера

#### OvrSetRetry(s: LongInt); - размер испытательной области оверлейного буфера

**Функции**

#### OvrGetByf: LongInt; - размер оверлейного буфера

#### OvrGetRetry: LongInt; - размер испытательной области оверлейного буфера

### Модуль Printer .

Включает в себя текстовую переменную Lst, которая связана с принтером, и ее можно использовать в качестве файловой переменной в процедурах Write, WriteLn, т.к. эта переменная связывается с принтером автоматически, не следует выполнять процедуры Assign, ReWrite, а так же закрывать файл.

##  14. Работа в главном меню.

Вход--клавиша F10.

### Меню работы с файлами File.

#### New - создание нового файла;

#### Open - открытие списка файлов по шаблону (F3);

#### Save - сохранить файл под старым именем (F2);

#### Save As .. - сохранить файл под новым именем;

#### Save all - сохранить все активные файлы;

#### Change Dir - изменить текущий каталог;

#### Print - распечатать активное окно на принтере;

#### Printer Setup - установка параметров принтера;

#### DOS shell - временный выход в ДОС

#### Exit - выход в ДОС

###  Меню редактирования Edit.

#### Undo - удалить изменения;

#### Redo - восстановить изменения;

#### Cut - удалить фрагмент;

#### Copy - скопировать фрагмент;

#### Paste - вставить информацию;

#### Clear - удалить фрагмент;

#### Show clipboard - открыть окно промежуточного буфера;

### Меню поиска Search.

#### Find - найти фрагмент

#### Replace - заменить фрагмент

#### Search again - повторный поиск

#### Go to line number - переход на строку с заданным номером

#### Show last compile error - показать последнюю ошибку компилятора

#### Find error - найти ошибку (следует указать ее полный адрес в 16- ричной системе счисления)

#### Find procedure - найти подпрограмму

### Меню выполнения программы (RUN)

Меню запускает программу, выполняет отдельные части во время ее отладки.

#### RUN(запустит программу) Если текст программы был изменен, то при очередном запуске программы происходит ее перекомпиляция.

#### STEP OVER (трассировка без захода в подпрограммы) Трассировка подпрограмм не производится - они рассматриваются как единые операторы.

#### TRACE INTO(трассировка с заходом в подпрограммы) Она выполняется по отдельным строкам.

#### GO TO CURSOR(выполнить до строки, помеченной курсором) Для того чтобы прервать выполнение программы, следует нажать комбинацию клавиш Ctrl+Break.

#### PROGRAM RESET(прекратить выполнение программы)

#### PARAMETERS(параметры) Параметры должны задаваться перед запуском программы.

### Меню компиляции (COMPILE)

#### Compile (компиляция) Если обнаружена синтаксическая ошибка выдается сообщение об этой ошибке, а курсор помещается в место ее нахождение.

#### Make (собрать программу) Если тексты отдельных модулей были изменены, то соответствующие модули перекомпилируются.

#### Build(создать программу)

#### Destination(размещение файла)

#### Primary file(основной файл) Устанавливает основной файл программы для команд Make и Build

#### Clear primary file(отказаться от основного файла)

#### Information(получить информацию о файле)

### Меню отладки (Debug)

Меню отладки позволяет задать параметры.

#### Breakpoints (работа с точками остановка)

#### Call stack(окно используемых подпрограмм)

#### Register(окно регистров)

#### Watch(окно отладки)

#### Output(окно выходных результатов)

#### User screen(окно пользователя)

#### Evaluate/modify(вычислить/модифицировать)

#### Add watch(добавить наблюдаемый параметр)

#### Add breakpoint(добавить точку останова)

### Меню инструментальных средств (Tools)

Меню позволяет задать программы, которые можно запустить не выходя из интегрированной среды.

#### Messahes(сообщение) Открывает окно сообщение.

#### Go to next(на следующую строку)

#### Go to previous(на предыдущую строку) Меню может содержать строки с именами программ, которые можно запустить из интегрированной среды.

### Меню параметров среды (Options)

С помощью этого меню можно установить необходимые параметры интегрированной среды.

#### Compiler(параметры компилятора)

#### Memory sizes - устанавливает размеры используемой памяти

#### Linker - установка параметров компоновщика

#### Debugger - установка параметров отладчика

#### Directories - установка каталогов для файлов

#### Tools - установка программы вызываемой из среды программ

#### Environment - установка условий работы в среде

### Меню окон Window.

#### Tile - размещение окон встык.

#### Cascade - каскадное размещение окон.

#### Close All - удалить все окна.

#### Refresh display - обновить экран среды.

#### Size/Move - изменение размеров, перемещение окна.

#### Zoom - раскрыть окно.

#### Next - следующее окно.

#### Previous - предыдущее окно.

#### Close - закрыть окно.

#### List - список окон.

### Меню помощи Help.

#### Contents - справка о выводимой на экран информации

#### Index - выводит ключевые слова

#### Topic search - предметный поиск, о слове на котором находится курсор

#### Previous topic - предыдущая тема

#### Using help - справка о справке

#### Files - файлы контекстной помощи

#### Compiler directives - директивы компилятора

#### Reserved words - зарезервированные слова

#### Standart units - стандартные модули

#### Turbo Pascal language - основные элементы Turbo Pascal

#### Error message - сообщения об ошибках

#### About - сведения о системе пакета.

# Приложения.

## Перемещение курсора.

#### Ctrl+→; Ctrl+← - на слово

#### Ctrl+Home; Ctrl+End - экран

#### Ctrl+PgUp; Ctrl+PgDn - текст

#### Ctrl+Q B; Ctrl+Q K - выделенный блок

#### Ctrl+O O - вставка ключей компилятора

## Работа с блоками

### Выделение блока

#### Мышь

#### Shift + ←, ↑, →, ↓

#### Ctrl + K B - начало блока; Ctrl + K K - конец блока

#### Блок размером в одно слово Ctrl + K T

#### Ctrl + Ins - скопировать блок в буфер

#### Shift + Del - переместить блок в буфер

#### Shift + Ins - перенос из буфера в текст

#### Ctrl + K C - копировать блок на место курсора

#### Ctrl + K V - перенос блока

#### Ctrl + K H - выделение особым цветом или снятие его

#### Ctrl + K W - записать блок на диск

#### Ctrl + K R - чтение блока с диска

#### Ctrl + K P - печать блока

#### Ctrl + Q L - восстановление строки

### Команды компилятора

#### {$A+/- } - выравнивание по границе машинного слова

#### {$B+/- } - полное вычисление логических операций

#### {$D+/- } - отладочная информация

#### {$E+/- } - эмуляция сопроцессора

#### {$F+/- } - форматирование дальних и ближних адресов

#### {$G+/- } - форматирование команд для 80286

#### {$I+/- } - проверка результата ввода- вывода

#### {$L+/- } - информация о локальных параметрах модуля

#### {$N+/- } - использование сопроцессора

#### {$O+/- } - использование оверлеев

#### {$P+/- } - массивы как параметры

#### {$Q+/- } - проверка переполнения арифметики

#### {$R+/- } - проверка диапазонов

#### {$S+/- } - проверка стека

#### {$T+/- } - использование адресного оператора @

#### {$V+/- } - проверка строковых параметров

#### {$X+/- } - использование расширенного синтаксиса

### Ключи параметров

#### {$I filename} - включение файла

#### {$L filename} - компановка объектного файла

#### {$M staksize, heapmin, heapmax} - задание размеров памяти

#### {$O unitname} - компановка оверлейного модуля

# Образцы решения задач

#### Рассмотрим задачу по нахождению наибольшего значения ряда для следующей задачи: В трех классах (не более 25 уч- ся) проведены экзамены. Нужно распечатать протокол в виде:

**Число оценок**

**Перечисление оценок(для каждого класса отдельно)**

**Средний балл**

**Наибольший балл**

program ocenki;

uses Crt;

type graderange=0..5; {допустимая оценка}

 arraytype=array[1..25] of graderange; {таблица оценок}

var numx, numy, numz, i, k: integer; {число учеников по классам}

markx, marky, markz: arraytype; {оценки из табл.по классам}

avgmarkx, avgmarky, avgmarkz: real; {средние оценки по классам}

higradex, higradey, higradez: graderange; {max оценки по классам}

ch: char; {задержка для выхода по нажатию любой клавиши}

{заполнение таблицы оценок}

procedure readdate(var mark: arraytype; var num: integer);

var count: integer; {счетчик}

 begin

 writeln('Ведите число учеников');

 readln(num);

 for count: =1 to num do

 begin

 write('mark[', count, ']=');

 readln(k);

 mark[count]: =k;

 end; {for}

 end; {readdate}

{вывод таблицы оценок}

procedure printdate(var mark: arraytype; var num: integer);

var count: integer;

 begin

 i: =i+1;

 writeln(num, ' человек в ', i, '- ом классе');

 for count: =1 to num do

 write(mark[count]: 3);

 writeln;

 end; {printdate}

{сумма оценок в классе}

function sumarray(numstosum: arraytype; num: integer): integer;

var total, count: integer;

 begin

 total: =0;

 for count: =1 to num do

 total: =total+numstosum[count];

 sumarray: =total;

 end; {sumarray}

 {средняя оценка в классе}

function avgarray(numstoavg: arraytype; m: integer): real;

 begin

 avgarray: =sumarray(numstoavg, m)/m

 end; {avgarray}

{лучшая оценка в классе}

function findmax(marks: arraytype; num: integer): integer;

var largest\_so\_var, count: integer;

 begin

 largest\_so\_var: =marks[1];

 for count: =1 to num do

 if largest\_so\_var<marks[count] then

 largest\_so\_var: =marks[count];

 findmax: =largest\_so\_var

 end; {findmax}

{main file}

begin

 clrscr;

 writeln('1- й класс');

 readdate(markx, numx);

 writeln('2- й класс');

 readdate(marky, numy);

 writeln('3- й класс');

 readdate(markz, numz);

 i: =0;

 printdate(markx, numx);

 printdate(marky, numy);

 printdate(markz, numz);

 avgmarkx: =avgarray(markx, numx);

 writeln('Средняя в 1- ом классе=', avgmarkx: 5: 3);

 avgmarky: =avgarray(marky, numy);

 writeln('Средняя в 2- ом классе=', avgmarky: 5: 3);

 avgmarkz: =avgarray(markz, numz);

 writeln('Средняя в 3- eм классе=', avgmarkz: 5: 3);

 higradex: =findmax(markx, numx);

 writeln('max оценка в 1- ом классе=', higradex);

 higradey: =findmax(marky, numy);

 writeln('max оценка во 2- ом классе=', higradey);

 higradez: =findmax(markz, numz);

 writeln('max оценка в 3- ем классе=', higradex);

 ch: =readkey;

end. {main file}

#### Распечатать все допустимые символы.

program simwol;

uses crt;

 var x: byte; {код символа}

 y: char; {символ по коду}

begin

 clrscr;

 for x: =32 to 255 do {коды от 0 до 31- управляющие}

 begin

 y: =chr(x);

 write(y, ' ');

 end;

 y: =ReadKey;

end.

#### Ввести с клавиатуры строку (Не более 20 символов) и выдать ее на экран выбегающим текстом в центр экрана.

program begstr;

uses crt;

var l, i, n, j: integer; {длина стр., счетчик симв., последняя позиция}

 st : string[20]; {вводимая строка}

 wyr : string[1]; {вырезаемый символ}

 ch: char;

begin

 clrscr;

 writeln('Введите слово < 20 символов');

 readln(st);

 l: =length(st); {длина строки}

 n: =(64- l) div 2+l; {последняя позиция при выводе}

 for i: =l downto 1 do{номер перемещ.символа, начиная с последнего}

 begin

 for j: =2 to n do {число перемещений}

 begin

 wyr: =copy(st, i, 1); {взять 1 символ}

 gotoxy(j- 1, 8);

 write(' '); {стереть символ}

 gotoxy(j, 8);

 write(wyr); {напечатать текущий символ}

 end; {for j}

 n: =n- 1;

 end; {for i}

 ch: =ReadKey

end.

#### По введенным корням квадратного уравнения составить приведенное квадратное уравнение.

program korni;

uses crt;

var x1, x2, p, q: real; {корни и коэффициенты}

 ch: char;

begin

 clrscr;

 writeln('Введите корни кв.уравнения. x1, x2');

 readln(x1, x2);

 p: =- (x1+x2); {второй коэффициент}

 q: =x1\*x2; {свободный член}

 writeln('x\*x+(', p: 6: 2, ')\*x+(', q: 6: 2, ')=0');

 ch: =ReadKey

end.

#### Ввести с клавиатуры число в интервале от 1 до 27 и распечаать все трехзначные числа, сумма цифр которых равна введенному числу.

program sumcif;

uses crt;

var n, i, j, k, l: integer; {сумма цифр, цифры, счетчик}

ch: char;

begin

 textcolor(14);

 textbackground(1);

 clrscr;

 write('Введите сумму цифр n=');

 readln(n);

 l: =0;

 for i: =1 to 9 do

 for j: =0 to 9 do

 for k: =0 to 9 do

 if i+j+k=n then

 begin

 write(100\*i+10\*j+k: 4);

 l: =l+1;

 end;

 writeln;

 writeln('Всего ', l, ' чисел');

 ch: =readkey;

end.

#### Найти все совершенные числа не превосходящие введенного с клавиатуры. Число называется совершенным, если оно равно сумме своих делителей, исключая себя.

program sower1;

uses crt;

var lim, sum, prower: word; {до какого проверять, сумма цифр, счетчик}

ch: char;

{вычисление суммы делителей}

function sumdelit(n: word): word;

var delit, ostatok, s: word; {делитель, остаток от деления, накопитель}

 begin

 s: =1;

 for delit: =2 to n- 1 do

 begin

 ostatok: =n mod delit;

 if ostatok=0 then s: =s+delit

 end;

 sumdelit: =s

end;

{главная программа}

begin

TextColor(14);

TextBackground(1);

clrscr;

write('n=');

read(lim);

for prower: =3 to lim do

begin

 sum: =sumdelit(prower);

 if sum=prower then writeln(prower, '- совершенное')

 end;

 writeln('Пока все...');

 ch: =readkey;

 end.

#### Распечатать все тройки пифагоровых чисел, не превосходящие введенного с клавиатуры. При выводе не допускать повторений полученных на основе переместительного закона сложения. Тройка чисел a, b и c называется пифагоровой, если выполняется равенство: a2+b2=c2.

program pifagor;

uses crt;

var u, v: integer; {взаимно простые нечетные натуральные числа}

 { при этом u>v, а тройка пифагоровых чисел }

 {находится по формулам a=u\*v; b=(u\*u- v\*v)/2; }

 {c=(u\*u+v\*v)/2, (a\*a+b\*b=c\*c- пифагоровы) }

 max: integer; {ограничитель}

 ch : char;

{поиск взаимно простых}

function wzaimprost(x, y: integer): boolean;

{поиск наибольшего общего делителя}

 function nod(x, y: integer): integer;

 begin {nod}

 while (x<>0) and (y<>0) do

 if x>=y then x: =x mod y

 else y: =y mod x;

 nod: =x+y;

 end; {nod}

begin {wzaimprost}

 wzaimprost: =nod(x, y)=1

end; {wzaimprost}

{main file}

begin

 textcolor(14);

 textbackground(1);

 clrscr;

 write('Введите ограничение max=');

 read(max);

 v: =1; u: =3; {нечетные и взаимно простые}

 while (u\*u+v\*v) div 2<=max do

 begin

 while (u\*u+v\*v) div 2<=max do

 begin

 if wzaimprost(u, v) then

 writeln(u\*v, ' ', (u\*u- v\*v) div 2, ' ', (u\*u+v\*v) div 2);

 u: =u+2;

 end;

 v: =v+2;

 u: =v+2;

 end;

ch: =ReadKey

end.

#### Вычислить факториалы чисел, не превосходящих введенного с клавиатуры.

program faktor1; {использование цикла с предусловием}

 uses crt;

 var f, i, n: longint; {факториал, счетчик}

 ch: char;

 j: word;

begin

 clrscr;

 f: =1; i: =2;

 write('Введи n=');

 read(n);

 for j: =2 to n do

 begin

 while i<=j do

 begin

 f: =f\*i;

 i: =i+1;

 end;

 writeln(j, '!=', f);

 ch: =readkey;

 end;

 ch: =readkey;

end.

#### Вывести на экран таблицу умножения.

program umno;

uses crt;

var i, y, x: integer; {счетчик, множители}

ch: char;

procedure wertik;

 begin

 gotoxy(1, 3);

 for i: =1 to 10 do

 writeln(i, ' |');

 end;

procedure goriz;

 begin

 gotoxy(6, 1);

 for i: =1 to 10 do

 write(i: 4);

 gotoxy(2, 2);

 write('\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_');

 end;

 {main file}

 begin

 TextBackGround(1);

 TextColor(14);

 ClrScr;

 wertik;

 goriz;

 for i: =1 to 10 do

 for y: =0 to 9 do

 begin

 gotoxy(y\*4+6, i+2);

 x: =i\*(y+1);

 write(x: 4);

 end;

 ch: =readkey

 end.

#### Ввести с клавиатуры размер одномерного ряда(не более 25 элементов). Заполнить его с клавиатуры целыми числами. Распечатать ряд в строку. Найти и вывести на экран наименьший и наибольший элементы, с указанием их индексов. Упорядочить ряд по возрастанию, используя алгоритм линейной сортировки или сортировку методом "пузырька". Распечатать сортированные данные в одну строку.

program tablica; {имя программы}

 USES CRT; {открытие библиотеки}

 const lim=25; {максимальное число элементов}

 type size=1..lim; {номер элемента}

 tabl=array[size] of integer; {описание таблицы}

 var n, i: word; {число элементов и счетчик элементов}

 a: tabl; {переменная как ряд}

 ch: char; {переменная опроса клавиатуры}

{процедура ввода числа элементов и загрузка ряда}

 procedure zagt (var b: tabl; i: word); {вывод сообщения}

 begin

 write('Введите число элементов n='); {вывод сообщения}

 readln(n); {ввод размеров таблицы}

 for i: =1 to n do{начало цикла}

 begin

 write('a[', i, ']='); {подсказка по вводу элементов}

 readln (b[i]); {ввод текущего элемента}

 end; {конец цикла}

 end; {конец процедуры zagt}

{процедура распечатки ряда в одну строку}

 procedure printdate(var a: tabl; i: word);

 begin

 for i: =1 to n do{начало цикла}

 write(a[i], ' '); {печать текущего элемента, пробела и конец

 цикла}

 end; {конец процедуры printdate}

{процедура поиска наименьшего элемента}

 procedure findmin (var a: tabl; n: word);

 var k, min, i: integer; {номер наименьшего элемента, наименьший

 элемент, счетчик числа элементов}

 begin

 min: =a[1]; k: =1; {считаем первый элемент наименьшим}

 for i: =2 to n do{начало цикла}

 if min>a[i] then{если минимум больше, то ..}

 begin

 min: =a[i]; {минимальному присвоить новое значение}

 k: =i; {новый номер минимального элемента}

 end; {конец ветвления}

 writeln; {переход на новую строку}

 writeln('Наименьшее=', min, ' номер=', k); {вывод минимального

элемента и его номера}

 end; {конец процедуры findmin}

{поиск наибольшего элемента}

 procedure findmax (var a: tabl; n: word);

 var k, max, i: integer; {номер наибольшего элемента, наибольший

элемент, счетчик числа элементов}

 begin

 max: =a[1]; k: =1; {считаем первый элемент наибольшим}

 for i: =2 to n do{начало цикла}

 if max<a[i] then{если максимум меньше, то ..}

 begin

 max: =a[i]; {максимальному присвоить новое значение}

 k: =i; {новый номер максимального элемента}

 end; {конец ветвления}

 writeln('Наибольшее=', max, ' номер=', k); {вывод максимального

 элемента и его номера}

 end; {конец процедуры findmax}

{процедура линейной сортировки}

 procedure linearsort(var namb: tabl; n: size);

 var pass, cand: size; {номера предыдущего и последующего элементов}

 temp: integer; {временная переменная для обмена данными}

 begin

 for pass: =1 to n- 1 do{начало внешнего цикла}

 for cand: =pass+1 to n do{начало внутреннего цикла}

 if namb[pass]>namb[cand] then {если предыдущий больше пос-

 ледующего, то..}

 begin

 temp: =namb[pass]; {три строки для

 обмена данными}

 namb[pass]: =namb[cand];

 namb[cand]: =temp;

 end; {конец обмена данными }

 end; {конец процедуры linearsort}

{начало процедуры сортировки методом "пузырька" }

 procedure bubblesort(var numb: tabl; n: size); {}

 var pos : size;

 temp : integer;

 switch: boolean;

 begin

 repeat

 switch: =false;

 for pos: =1 to n- 1 do

 begin

 if numb[pos]>numb[pos+1] then

 begin

 switch: =true;

 temp: =numb[pos];

 numb[pos]: =numb[pos+1];

 numb[pos+1]: =temp;

 end;

 end;

 until switch=false;

 end; {bubblesort}

{main file}

begin

 textcolor(14);

 textbackground(1);

 clrscr;

 zagt(a, i);

 printdate(a, n);

 findmin(a, n);

 findmax(a, n);

 bubblesort(a, n);

 printdate(a, n);

 writeln;

 ch: =readkey;

end.

#### Ввести с клавиатуры количество колонок n и количество строк m двумерного ряда. Заполнить ряд случайными целыми числами. Вывести ряд на экран в виде прямоугольной таблицы, оформив ее нумерацией строк и колонок. Найти наибольший и наименьший элементы с указанием адресов. Провести сортировку ряда по возрастанию и в строках, и в столбцах. Вывести на экран сортированные данные.

program massiv;

uses crt;

label 1;

const row=17;

 col=30;

type strka=1..row;

 stol=1..col;

 spisoc=array[stol, strka] of integer;

var p1, p2, t, i, k, n, m, j, jc, jr, lc, lr: integer;

 sp: spisoc;

 ch: char;

procedure zagruz(var sp: spisoc; var n, m: integer) ;

 begin

 writeln('введите 2 размера таблицы');

 readln(n, m);

 for i: =1 to m do

 for k: =1 to n do

 begin

 write('sp[', k, ', ', i, ']=');

 t: =random(100);

 sp[k, i]: =t;

 Delay(100);

 writeln(sp[k, i]);

 end;

 end;

procedure printdate(var sp: spisoc; var m, n: integer);

 begin

 gotoxy(4, 3);

 for i: =1 to m do

 begin

 for k: =1 to n do

 begin

 Delay(1);

 write(sp[k, i]: 3);

 end;

 gotoxy(4, i+3);

 end;

 end;

procedure oformlenie(m, n: integer);

begin

 gotoxy(3, 1);

 for i: =1 to n do

 write(i: 3);

 gotoxy(1, 2);

 for i: =1 to 3\*(n+1) do

 write('=');

 gotoxy(1, 3);

 for j: =1 to m do

 writeln(j: 2, '|');

end;

procedure findminmax(sp: spisoc; m, n: integer);

var min, max: integer;

begin

 min: =sp[1, 1]; jc: =1; jr: =1;

 max: =sp[1, 1]; lc: =1; lr: =1;

 for i: =1 to m do

 for k: =1 to n do

 begin

 if min>sp[k, i] then

 begin

 min: =sp[k, i];

 jc: =k;

 jr: =i;

 end;

 if max<sp[k, i] then

 begin

 max: =sp[k, i];

 lc: =k;

 lr: =i;

 end;

 end;

 writeln;

 writeln('Наим=', min, ' его позиция- ', jc: 3, jr: 3);

 writeln('Наиб=', max, ' его позиция- ', lc: 3, lr: 3);

end;

procedure linearsort(var sp: spisoc; m, n: integer);

var temp: integer;

begin

 for i: =1 to m do

 for k: =1 to n do

 for p1: =1 to m do

 for p2: =1 to n do

 if sp[k, i]<sp[p2, p1] then

 begin

 temp: =sp[k, i];

 sp[k, i]: =sp[p2, p1];

 sp[p2, p1]: =temp;

 end;

end;

{main file}

begin

1:

 TextColor(14);

 TextBackGround(1);

 clrscr;

 randomize;

 zagruz(sp, n, m);

 ch: =readkey;

 clrscr;

 oformlenie(m, n);

 printdate(sp, m, n);

 findminmax(sp, m, n);

 writeln('Нажми что- нибудь');

 ch: =readkey;

 writeln('Сортированные данные');

 linearsort(sp, m, n);

 printdate(sp, m, n);

 findminmax(sp, m, n);

 writeln;

 WriteLn('Будете ещё y/n ');

 ch: =readkey;

 if (ch='y') or (ch='Y') then goto 1;

end.

#### Построить график функции y=a\*sin(b\*x+c). Предусмотреть возожность замены функции на другую, а так же возможность совмещеия графиков одной функции при различных значениях параметров.

program graphic;

uses crt, graph;

label 1;

const sx=320; sy=240; n=0; cx=639; cy=479; {s- сдвиг, n- начало, c- конец}

 kx=20; ky=20; {коэффициенты сжатия по осям}

var x, a, b, c: real; {независимая переменная и параметры}

 x2, x3, y2, y3, Gd, Gm: integer; {координаты начала и конца отрезка

 графика}

 l, r: integer; {границы построения графика}

 i, j: word; {счетчики}

 d: word; {цвет графика}

 ch: char;

{инициализация графики}

procedure initgr;

begin

 Gd: =Detect;

 InitGraph(Gd, Gm, ' ');

end;

procedure osi; {координатные оси}

 begin

 SetColor(15);

 line(n, sy, cx, sy); {ось х}

 line(sx, n, sx, cy); {ось у}

 end;

{координатная сетка}

procedure setka;

 begin

 setcolor(4);

 i: =20; {вертикальная}

 while i<639 do

 begin

 line(i, n, i, cy);

 i: =i+kx;

 end;

 j: =20; {горизонтальная}

 while j<479 do

 begin

 line(n, j, cx, j);

 j: =j+ky;

 end;

 setcolor(14);

 end;

{значения функции}

function f(a, b, c, x: real): real;

begin

 f: =a\*sin(b\*x+c); {достаточно изменить функцию здесь}

end;

{координата у точки принадлежащей графику}

function y1(x, a, b, c: real): integer;

 begin

 y1: =round(- f(a, b, c, x)\*ky+sy);

 end;

{координата x точки принадлежащей графику}

function x1(x: real): integer;

 begin

 x1: =round(x\*kx+sx);

 end;

{построение отрезков, составляющих график}

procedure grf(var a, b, c: real);

 begin

 outtextxy(10, 40, 'Левая граница=');

 read(l);

 x: =l;

 outtextxy(10, 50, 'Правая граница=');

 read(r);

 while x<=r do

 begin

 x2: =x1(x); {}

 y2: =y1(x, a, b, c);

 x: =x+0.2;

 x3: =x1(x); {}

 y3: =y1(x, a, b, c);

 line(x2, y2, x3, y3);

 end;

 end;

{main file}

begin

 initgr;

 setbkcolor(1);

 d: =15;

1: outtextxy(10, 10, 'Введите коэффициенты a, b и c');

 readln(a, b, c);

 setka;

 osi;

 setcolor(d); {}

 grf(a, b, c);

 outtextxy(10, 30, 'Еще ? (y/n)');

 ch: =readkey;

 if (ch='y') or (ch='Y') then

 begin

 d: =d- 1; {}

 goto 1;

 end;

 CloseGraph;

end.

#### (Шуплецов В.Ю.) Вывести на экран иммитацию электронных часов.

{\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*}

{ Turbo Crt, Graph, Dos }

{ Copyright (c) 1996 by Starson Internation, Inc. }

{\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*}

program Clock;

 uses Crt, Graph, Dos;

 const rx=145; ry=145;

 days: array[0..6] of string[9] =

 ('Sunday', 'Monday', 'Tuesday',

 'Wednesday', 'Thursday', 'Friday',

 'Saturday');

 mesac: array[1..12] of string[10] =

 ('January', 'February', 'March', 'April',

 'May', 'June', 'Jule', 'August', 'September',

 'October', 'November', 'Desember');

 mus: array[1..48] of word =

 (330, 294, 262, 294, 262, 294, 330, 392, 330, 294, 25000,

 330, 294, 262, 294, 262, 294, 330, 392, 330, 294, 25000,

 294, 330, 349, 330, 294, 262, 262, 248, 262, 440, 392, 25000,

 330, 294, 248, 262, 330, 294, 248, 262, 330, 294, 294, 294, 196,

 25000);

 dlit: array[1..48] of word = (10, 5, 5, 10, 5, 5, 10, 5, 5, 10, 5,

 10, 5, 5, 10, 5, 5, 10, 5, 5, 10, 1,

 5, 5, 10, 5, 5, 10, 5, 5, 10, 10, 10, 3,

 5, 5, 5, 5, 5, 5, 5, 5, 5, 10, 5, 5, 10, 5);

 var x, y, i, Gd, Gm, f: integer;

 year, mes, dte, week, hour, min, sec, cs: word;

 ch: char;

procedure Initi;

 begin

 Gd: =Detect;

 InitGraph(Gd, Gm, 'd: \tp7\bgi');

 end;

procedure Cif1;

 begin

 SetFillStyle(1, 0);

 SetColor(7);

 Bar3D(75, 195, 400, 280, 1, TopOff);

 end;

procedure Ciferblat;

 begin

 SetBkColor(0);

 SetColor(14);

 SetFillStyle(11, 4);

 Bar3D(0, 0, 639, 479, 1, TopOn);

 Cif1;

 end;

procedure WriteOut(S: string);

 begin

 OutTextXY(x, y, s);

 Inc(y, TextHeight('M')+2);

 end;

function Int2Str(L: LongInt): string;

 var S: string;

 begin

 Str(L, S);

 Int2Str: =S;

 end;

procedure Musica;

 begin

 Sound(mus[i]);

 Delay(dlit[i]\*100);

 end;

procedure Data;

 begin

 GetDate(year, mes, dte, week);

 GetTime(hour, min, sec, cs);

 end;

procedure Date1; Forward;

procedure Secound; Forward;

function LeadZero(w: word): string;

 var S1: string;

 begin

 Str(w: 0, S1);

 if Length(S1)=1 then S1: ='0'+S1;

 LeadZero: =S1;

 end;

procedure Chas;

 begin

 SetColor(0);

 x: =100; y: =200;

 if hour=0 then WriteOut(LeadZero(23))

 else WriteOut(LeadZero(hour- 1));

 SetColor(11);

 x: =100; y: =200;

 WriteOut(LeadZero(hour));

 if (hour=0) and (f=0) or (hour=12) and (f=0) then

 begin

 f: =1;

 Date1;

 end;

 end;

procedure Minute;

 begin

 SetTextStyle(GothicFont, HorizDir, 6);

 SetColor(0);

 x: =210; y: =200;

 if min=0 then WriteOut(LeadZero(59))

 else WriteOut(LeadZero(min- 1));

 Data;

 SetColor(12);

 x: =210; y: =200;

 WriteOut(LeadZero(min));

 if min<>0 then f: =0

 else

 begin

 Chas;

 if f=0 then

 begin

 Sound(3000); { Beep }

 Delay(1500); { For 1000 ms }

 NoSound; { Relief! }

 end;

 end;

 end;

procedure Date1;

 begin

 ClearDevice;

 Ciferblat;

 SetTextStyle(DefaultFont, HorizDir, 1);

 SetColor(15);

 OutTextXY(10, 450, 'c Starson Inc. 1996');

 Circle(13, 454, 6); Circle(13, 454, 5);

 Data;

 SetColor(10);

 SetTextStyle(GothicFont, HorizDir, 6);

 x: =10; y: =10;

 WriteOut(Int2Str(year));

 SetTextStyle(GothicFont, HorizDir, 8);

 SetColor(12);

 x: =300; y: =20;

 WriteOut(mesac[mes]);

 SetTextStyle(GothicFont, HorizDir, 12);

 SetColor(14);

 x: =440; y: =150;

 WriteOut(Int2Str(dte));

 SetTextStyle(GothicFont, HorizDir, 8);

 SetColor(11);

 x: =250; y: =350;

 WriteOut(days[week]);

 SetTextStyle(GothicFont, HorizDir, 6);

 SetColor(15);

 x: =180; y: =195;

 WriteOut(': : ');

 Chas;

 Minute;

 if f=1 then

 begin

 SetColor(13);

 x: =320; y: =200;

 WriteOut('00');

 for i: =1 to 48 do

 begin

 Musica;

 Secound;

 end;

 NoSound;

 end;

 end;

procedure Secound;

 begin

 SetTextStyle(GothicFont, HorizDir, 6);

 SetColor(0);

 x: =320; y: =200;

 WriteOut(LeadZero(sec));

 Data;

 SetColor(13);

 x: =320; y: =200;

 WriteOut(LeadZero(sec));

 if sec=0 then Minute;

 end;

{ Main program }

begin

 {SetDate(1996, 3, 16);

 SetTime(15, 41, 45, 0); }

 f: =0;

 Initi;

 Date1;

 repeat

 Secound;

 Delay(1995);

 until KeyPressed;

 CloseGraph;

 if (ParamCount <> 3) or (ParamStr(1)<>'S') or

 (ParamStr(2)<>'W') or (ParamStr(3)<>'N') then

 begin

 ch: =ReadKey;

 GotoXY(1, 20);

 TextColor(15);

 WriteLn(' (c) Starson Inc. 1996');

 WriteLn('Шуплецов В.Ю. тел. оф. 343- 2 / 13- 32- 45');

 WriteLn(' д. 139- 32- 45');

 WriteLn('Всегда рады ответить на ваши вопросы ! Удачи Вам !');

 if ch=#0 then ch: =ReadKey;

 ch: =ReadKey;

 end;

end. Benjamin S.

#### Написать программу для копирования текстового файла по его имени в файл zapis1.pas.

program chtenie;

uses crt;

var i: byte;

 a, r, l: string;

 G, F: text;

 ch: char;

begin

 clrscr;

 write('Введи имя файла для чтения=');

 read(l);

 r: ='a: \zapis1.pas';

 assign(F, l);

 assign(g, r);

 rewrite(g);

 reset(f);

 while not eof(F) do

 begin

 readln(F, a);

 writeln(g, a);

 writeln(a);

 end;

 ch: =readkey;

 close(f);

 close(g);

end.

#### Задано 20 слов на немецком языке. Слова должны перемещаться по экрану в определенном порядке, если слово имеет отношение к осенней тематике, нужно нажимать <пробел>, при нажатии на других словах должна фиксироваться ошибка. По окончанию должна выводиться оценка.

program Osenslowa;

uses Crt;

const Slowo: array[1..20] of string[12]=

 ('der Wind', 'der Vogel', 'es schneit', 'weiss', 'bunt',

 'das Gemuse', 'frieren', 'die Ernte', 'trube', 'heiss',

 'der Schneemann', 'das Neujahr', 'es taut', 'die Kleidung',

 'lernen', 'nass', 'hell', 'bunte Blatter',

 'Blatterfall', 'wehen');

var a: Real;

 b, c, d, e, g: Byte; {a- оценка, b- номер слова, d- число правильных нажатий, e- число неправильных нажатий}

 ch: Char;

 y: Boolean;

procedure Prowerka;

begin

 if (g=1) or (g=5) or (g=6) or (g=8) or (g=9) or (g=16)

 or (g=18) or (g=19) or (g=20) then

 begin

 d: =d+1;

 GotoXY(11, 15);

 WriteLn(d);

 end

 else

 begin

 e: =e+1;

 GotoXY(61, 15);

 WriteLn(e);

 end;

 repeat until KeyPressed;

 ch: =ReadKey;

 if ch=#0 then ch: =ReadKey;

 y: =True;

 end;

begin

 a: =0; d: =0; e: =0; y: =false;

 b: =1;

 TextBackground(1);

 TextColor(14);

 ClrScr;

 GotoXY(8, 3); WriteLn('Правильных');

 GotoXY(60, 3); WriteLn('Неправильных');

 for b: =1 to 20 do

 for c: =0 to 24 do

 begin

 y: =False;

 GotoXY(27, c+1);

 Write(Slowo[b]);

 GotoXY(1, 1);

 Delay(400);

 GotoXY(27, c+1);

 Write(' ');

 g: =b;

 if KeyPressed then Prowerka;

 if y=True then Break;

 end;

 a: =(d/9- e/11)\*5;

 WriteLn('Ваша оценка - ', a: 1: 2);

 ch: =ReadKey;

end.

## Задачи для самостоятельного решения.

### Линейные программы.

#### Для заданного a вычислить принадлежащий интервалу (pi; 2\*pi) корень уравнения ln(ctg(x)- 1)=a;

#### Вычислить дробную часть среднего геометрического трех заданных положительных чисел;

Примечание: Корень кубический из x можно записать: exp((1/3)\*log(x));

#### По координатам трех вершин треугольника вычислить его площадь и периметр;

#### Найти сумму цифр заданного четырехзначного числа.

#### Даны два действительных числа a и b. Получить их сумму, разность и произведение.

#### Дана длина ребра куба. Найдите объем куба и площадь его боковой поверхности.

#### Даны катеты прямоугольного треугольника. Найдите c и S.

#### Известен доход семьи. 57% бюджета идет на питание, 23% - на коммунальные услуги, 8% - на развлечения, остаток идет на накопления. Определите сумму расходов по статьям бюджета.

#### Дана сторона равностороннего треугольника. Определите площадь, высоту, радиус описанной и вписанной окружностей.

#### Известна длина окружности. Найдите площадь круга, ограниченного этой окружностью.

#### Найдите площадь кольца, внутренний радиус= r, а внешний - заданному числу R (R>r).

#### Дана арифметическая прогрессия. Найдите n- ый член и сумму первых n членов прогрессии, если известен первый член и разность арифметической прогрессии.

#### Дана геометрическая прогрессия. Найдите n- ый член и сумму первых n членов прогресси, если исвестнен первый член и знаменатель геометрической прогрессии.

### Операторы: условный, составной, пустой, перехода.

#### Решить уравнение а\*x4 +b\*x2 +c=0 при заданных a, b, c;

#### Заданы 3 числа a, b, c распечатать: 0, если треугольник с такими сторонами построить невозможно; 3, если треугольник равносторонний; 2, если равнобедренный; 1- прочий;

#### ≈ Даны 50 вещественных чисел. Найти величину наибольшего из них; (к массивам не прибегать !)

#### Составить программу нахождения корня уравнения ax+b=c, где a, b, c - данные числа.

#### Даны три действительных числа. Возвести в квадрат те из них, значения которых неотрицательны.

#### Даны действительные положительные числа . Выяснить, имеет существует ли треугольник с длинами сторон, если треугольник существует, то выяснить - является ли он остроугольным.

#### Даны три действительных числа. Выбрать из них те, которые принадлежат отрезку [- 5; 3].

#### Известны результаты соревнований трех лыжников. Выбрать результат победителя и занявшего третье место.

#### Вычислить f(a),

если: ;

#### Составьте программу решения квадратного уравнения ax2+bx+c=0, где a, b, c - данные числа.

#### Составьте программу решения биквадратного уравнения ax4+bx2+c=0, где a, b, c - данные числа.

#### Дано натуральное четырехзначное число. Верно ли, что все четыре цифры различны?

#### Дано натуральное число n (n<100), определяющее возраст человека ( в годах). Дать для этого числа наименование “год”, “года” или “лет”: например, 21 год, 44 года, 65 лет и т.д.

### Оператор цикла.

#### ≈ Написать три программы(без рекурсии) по вычислению f=10!, используя различные способы организации цикла.

#### ≈ С точностью 10- 5 вычислить наименьший положительный корень уравнения tg(x)=x, используя метод деления отрезка пополам

#### ≈ Вычислить по схеме Горнера: y=x10+2x9+3x8+...+10x+11;

#### Напечатать таблицу значений функций sin(x), cos(x) на отрезке [0, 1] с шагом 0.1 в следующем виде:

x sin(x) cos(x)

 - - - - - - - - - -

 1. 0000

 0. 9950

 . . . . . . . . .

 0. 5403

#### ≈ Распечатать те натуральные трехзначные числа, сумма цифр которых равна заданному n (1<=n<=27), а так же определить их количество;

#### Распечатать в порядке возрастания все трехзначные числа, в записи которых нет одинаковых цифр.

#### Распечатать таблицу истинности для функции: F=(A or B) and not(B and C) в виде:

 - - - - - - - - - - - -

A B C F

- - - - - - - - - - - -

true true true false

true true false true

 . . . . . . . . . . . . .

false false false false

- - - - - - - - - - - - -

#### Заданы целые числа a, b, c, d. Найти все целые корни уравнения: a\*x3+b\*x2+c\*x+ d=0;

Примечание: Если существуют целые корни(как отрицательные, так и положительные), то они являются делителями числа d.

#### Найти сумму цифр заданного натурального числа.

#### Вывести свою фамилию на экране монитора n раз.

#### Даны действительное число a, натуральное число n. Вычислить:

an; б) a(a+1)...(a+n); в) a(a- n)(a- 2n)...(a- n2).

#### Составьте программу вывода на экран дисплея квадратных корней чисел от 1 до n в виде таблицы.

#### Напечатать таблицу значений функций sin(x), cos(x) на отрезке [0°; 90°] c шагом n°.

#### Распечатать в порядке возрастания все трехзначные числа, в записи которых нет одинаковых цифр.

#### Найти сумму цифр заданного натурального числа.

#### Вычислить бесконечную сумму с заданной точностью ε (ε>0). Если очередное слагаемое оказалось по модулю меньше ε, то вычисления прекратить: а) ; б)

#### Дано натуральное число n. Вычислить произведение первых n сомножителей:

 а) б) 

#### Дано цело число m>1. Получить наибольшее k, при котором 4k<m.

### Литерный тип.

#### ≈ Напечатать в одну строку все символы от 'A' до 'Z';

#### Задан текст в латинской символике, при этом каждое слово заканчивается запятой, а текст- точкой. Определить количество слов, начинающихся с литеры 'w' или 'W';

#### Дана строка символов; подсчитать:

- сколько раз среди символов строки встречается данный символ;

- процентное содержание данного символа;

- заменить данный символ на знак “#”;

#### Составить программу, в которой во введенном тексте определяется количество знаков препинания.

#### Составить программу, в которой во введенном тексте определяется количество слов, начинающихся:

- на данную букву;

- на данный слог.

#### Составить программу, в которой во введенном тексте русские буквы заменяются на соответствующие латинские.

#### Дан текст. Напечатать сначала все цифры, входящие в него, а затем все остальные символы, сохраняя при этом взаимное расположение символов в каждой из этих двух групп.

#### Дан текст из цифр. Ввод прекратить после нажатия заданной клавиши. Напечатать цифру, наиболее часто встречающуюся в этом тексте (если таких цифр несколько, напечатать любую из них).

#### При подаче телеграмм их стоимость считается по количеству слов и знаков препинания. Стоимость одного слова(знака) равна k рублей. Составить программу, моделирующую процесс определения стоимости телеграммы.

#### Составьте программу вывода букв стоящих:

а) на нечетных местах; б) на четных местах; в) на местах, кратных 3.

#### Составьте программу, в которой все цифры строки заменяются на их названия. Например, если введена строка “2 апельсина”, выводится - “два апельсина”.

#### В текстовой строке содержится ряд чисел. Составьте программу, которая определяет среднее арифметическое этих чисел.

#### Составьте программу, которая вводит выражения вида cos( <число> [операция <число> ] ) и выводит результат. Считать, что значение в круглых скобках задается в градусах.

#### Получить на экране монитора изображение лампочки и обеспечить возможность “зажигать” и “гасить” нарисованную лампочку: включение и выключение лампочки должно выполняться с клавиатуры.

#### Получить на экране монитора изображение дома с окнами и обеспечить возможность “зажигать” и “гасить” свет в доме: включение и выключение света должно выполняться с клавиатуры, окна дома при зажженном при погашенном свете окрашивается в разные света.

#### Программно описать управление движением курсора по символьному экрану дисплея.

#### Программно описать управление движением круга по экрану дисплея.

#### Имеется список служащих и их оклады. Составить программу, которая обеспечит ввод этих данных в память компьютера и выводит список служащих и их оклады, если необходимо:

- увеличить все оклады на n рублей;

- добавить к окладу премию, равную m% от оклада.

#### Имеется список клиентов банка, номера счетов, тип вклада и размеры их вкладов. Составить программу, которая позволит ввести имеющиеся данные и вывести их в виде таблицы следующей структуры:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Номер счета | Тип вклада | Размер вклада | На сколько увеличится вклад | Новый размер вклада |
| (типы вкладов: срочный - 10% годовых; до востребования - 2%; детский - 85%; пенсионный - 50%) |

#### Имеется список оборудования и его стоимость. Составьте программу, которая выводит названия оборудования, имеющего стоимость выше средней более чем на 20%.

#### На соревнованиях по лыжному спорту после каждого финиша участника на табло появляется новый список, измененный с учетом этого спортсмена. Составьте программу для ЭВМ, обслуживающую такие соревнования.

#### В ЭВМ хранится набор английских слов и их перевод. Составить программу, которая переводит данное слово. Введенное слово может быть как на русском, так и на английском.

### Перечисляемые и ограниченные типы.

#### Для целого числа k, где 1<=k<=99 напечатать фразу: "мне k лет", учитывая, что иногда вместо лет нужно писать "год";

#### Напечатать все натуральные числа, меньшие 100, для которых число букв в записи числа на русском языке равно самому числу(пробелы не учитывать).

### Одномерные массивы.

#### ≈ Программа.Дан текст из 80 литер.Напечатать сначала все цифры, входящие в него, а затем все остальные литеры, сохраняя при этом взаимное расположение литер в каждой из этих двух групп;

#### Программа.Дан текст, содержащий от 1 до 70 букв, за которым следует точка.Напечатать этот текст в обратном порядке;

#### Программа.Дан непустой текст из цифр, за которым следует точка.Напечатать цифру, наиболее часто встречающуюся в этом тексте(если таких цифр несколько, напечатать любую из них);

const n=100;

var x: array[1..n] of real;

Преобразовать массив x по следующему правилу:

Элементы массива расположить в обратном порядке;

Элементы массива циклически сдвинуть на одну позицию влево, т.е. x[n]: =x[1], x[k]: =x[к+1]. n- задано;

#### ≈ Дана последовательность из 100 различных целых чисел. Найти сумму чисел этой последовательности, заключенных между max и min элементами этой последовательности;

#### Ввести текст из 100 литер(латинских). Напечатать данный текст, удалив из него все повторные вхождения каждой литеры.

#### Распечатать все "счастливые" автобусные билеты(шестизначные) и подсчитать их количество.

### Двумерные массивы.

#### Нечетные строки матрицы A заменить на x, если

 type vektor=array[1..20] of integer;

 matrica=arrray[1.20] of vektor;

 var A: matrica;

 x: vektor;

 B: array[1..20, 1..20] of integer;

#### Дан двумерный массив размером 20\*20. Упорядочить его строки по неубыванию их первых элементов.

#### Составить программу, в которой всем элементам одномерной таблицы присваивается случайное натуральное число, лежащее в данном промежутке.

#### Составьте программу, которая из заданной одномерной таблицы: а) сначала печатает отрицательные элементы, а затем неотрицательные;

#### - печатает элементы в обратном порядке;

#### - сначала печатает элементы, стоящие на четных места, затем - на нечетных.

#### Даны действительные числа a1, a2, ... , a15. Получить

 ;  .

#### Даны n действительных чисел x1, x2, ..., xn.

Вычислить ; .

#### Найти наибольший и наименьший элементы одномерной таблицы.

#### Все элементы с наибольшим значением в данной целочисленной квадратной таблице заменить нулями.

#### Дана таблица размером m×n. Найти сумму наибольших значений элементов ее строк.

#### Дана таблица размером m×n. Найти произведение наименьших элементов ее столбцов.

#### Дана квадратная таблица. Найти сумму элементов, располагающихся над главной диагональю(a11, a22,  ..., ann).

#### Дана действительная таблица размером m×n, все элементы которой различны. В каждой строке выбирается элемент с наименьшим значением, затем среди этих чисел выбирается наибольшее. Указать индексы элемента с найденным значением.

#### Элементы одномерной таблицы циклически сдвинуть на одну позицию влево, т.е. x[n]=x[1], x[k]=x[k+1].

#### Дана последовательность из n различных чисел. Найти сумму элементов заключенных между наименьшим и наибольшим элементами этой последовательности и распечатать их.

#### Дана таблица размером n×n. Упорядочить ее строки по возрастанию.

#### Нечетные строки матрицы А заменить на х, если

type vektor=array[1..20] of integer;

matrica=array[1..20] of vektor;

var A: matrica;

x: vektor;

B: array[1..20, 1..20] of integer; .

#### Найти сумму элементов двумерного массива, имеющих заданную разность индексов i- j=k. Число k может быть и отрицательным.

#### В заданном двумерном массиве заменить нулями элементы, стоящие в строках или столбцах где имеются нули.

#### Некоторый элемент двумерного массива назовем седловой точкой, если он является одновременно наименьшим в своей строке и наибольшим в своем столбце. Найти в массиве седловые точки.

#### Найти длину самой длинной последовательности элементов оджномерного массива, равных нулю.

#### Дана таблица размером n×n. Определить числа b1, b2, ...bn, равные соответственно:

- суммам элементов строк;

- наименьшим значениям элементов строк;

- произведениям элементов строк;

- средним арифметическим элементов столбцов;

- наибольшим значениям элементов столбцов;

- произведениям элементов столбцов.

### Функции и процедуры.

#### Даны три натуральных числа. Определить их наибольший общий делитель.

#### ≈ Даны отрезки a, b, c и d. Для каждой тройки этих отрезков, из которых можно построить треугольник, напечатать площадь данного треугольника.

#### Даны действительные числа s, t.

Получить: - 2s; 1, 17)+f(2, 2; t, s- t), где 

#### Даны числа a, b, c.

Получить 

#### Даны действительные числа s, t.

 Получить: +g(t; s)- g(2s- 1; st), где .

#### Даны действительные числа a, b.

Получить u=min(a; b), v=min(ab, a+b), min(u+v2; 3, 14).

#### Даны три числа. Отсортировать их по возрастанию.

#### Вычислить по длинам двух сторон треугольника и углу между ними длины третьей стороны и двух других сторон.

#### Проверить треугольники на предмет подобия по трем сторонам.

#### Проверить прямоугольники на предмет подобия по двум смежным сторонам.

#### Астрологи делят год на 12 периодов и каждому ставят в соответствие знак зодиака:

|  |  |  |  |
| --- | --- | --- | --- |
| 20.1 - 18.02 | Водолей | 19.02 - 20.03 | Рыбы |
| 21.3 - 19.04 | Овен | 20.04 - 20.05 | Телец |
| 21.5 - 21.06 | Близнецы | 22.06 - 22.07 | Рак |
| 23.7 - 22.08 | Лев | 23.08 - 22.09 | Дева |
| 23.9 - 22.10 | Весы | 23.10 - 22.11 | Скорпион |
| 23.11 - 21.12 | Стрелец | 22.12 - 19.1 | Козерог |

Составьте программу, которая по дате рождения определяет знак зодиака.

#### Даны натуральные числа m и n.

Получить 

#### Даны четыре пары координат точек. Проверить лежит ли они внутри окружности с центром в начале координат и радиусом R.

#### Составить программу, которая по номеру месяца выдает на экран дисплея его названия.

#### Числа вида a+bi, где a и b действительные числа, называются комплексными.

Сложение двух комплексных чисел производится по правилу: (a1+b1i)+(a2+b2i)=(a1+b1)+(b1+b2)i. Составьте программу сложения четырех комплексных чисел.

Произведение двух комплексных чисел (см. 4.14) производится по правилу: (a1+b1i)(a2+b2i)=(a1a2- b1b2)+(a1b2+a2b1)i. Составьте программу, которая находит произведение четырех комплексных чисел.

Найдите значение выражения: u+wu- w- xy, где u, w, x, y - комплексные числа. (см 4.14 и 4.15)

#### Дано натуральное число n. Среди чисел 1, 2, 3..., n найти все те, которые можно представить в виде суммы квадратов двух натуральных чисел.

#### Даны координаты четырехугольника ABCD. Найти периметр

#### Дано натуральное число n.

Получить а) последовательность f0, f1, ..., fn, где ; б) 

### Графические возможности

#### Построить и закрасить прямоугольник, центр которого совмещен с центром экрана. Стороны прямоугольника должны быть параллельны осям координат.

#### Построить окружность, центр которой совмещен с центром экрана.

#### Придумать и программно описать шаблоны следующих фигур: жука, букета цветов, робота, самолета, домика, будильника и т.д.

#### Дано натуральное число n (n≤999999). Записать его шестью цифрами, используя девятисегментный шаблон (как на почтовых открытках).

#### Получить на экране изображение действующих электронных часов, показывающих текущее время. Шаблоны используемых цифр должны соответствовать обычному для электронных часов семисегментному шаблону.

#### Получить на экране изображение часов с секундной (с минутной и часовой стрелками).

#### Получить на экране изображение забора

#### Получить на экране изображение узора:

#### m кругов по горизонтали и n по вертикали.

#### Получить на экране изображение мишени(мишени n колец).

#### Получить на экране изображение пирамиды: ( в пирамиде m кирпичей).

#### 6.9.11. Дано натуральное число R. Построить фигуры показанные на рисунке:

#### Фигуры образованы окружностью радиуса R и k точками, являющимися вершинами правильного многоугольника, вписанного в эту окружность и соединенными между собой, как показано на рисунке.

#### Дан квадрат. Его стороны поделены на n равных частей. Построить изображения приведенные на рисунке:

#### Столбчатая диаграмма (гистограмма)

|  |
| --- |
| представляет собой набор прямоугольников, основания которых равны, а высоты пропорциональны числовым величинам, взятым из некоторой совокупности. Для большей наглядности прямоугольники диаграммы обычно закрашивают в разные цвета. Постройте гистограмму для данных n действительных положительных чисел. |

#### Секторная диаграмма - это круг, площади

|  |
| --- |
| секторов которого, пропорциональны соответствующим числовым величинам, взятым из некоторой совокупности. Постройте секторную диаграмму для данных n действительных положительных чисел. |

#### Даны натуральные числа v1, v2, ..., v8, задающие число дней в году, в которых преобладало соответственно северное, северо- восточное, восточное, юго- восточное, южное, юго- западная, западное или северо- западное направление ветра. Построить розу ветров .

С

## Ответы на задачи помеченные знаком~.

####  7.1(6.2.4~)

 program maxreal;

 uses crt;

 label 4, 9;

 const n=50; {кол- во заданных чисел}

 var x, max: real; {очередное число, наиб. среди введенных}

 i: integer; {номер числа x}

begin

 write('Введите первое x=');

 readln(max); i: =1;

 4: i: =i+1;

 if i>n then goto 9;

 write('Введите ', i, '- ое x=');

 readln(x);

 if x>max then max: =x;

 goto 4;

 9: writeln('max=', max: 6: 2);

end.

####  7.2(6.3.1~)

program faktor1; {с предусловием}

uses crt;

 var f, i: integer; {факториал, счетчик}

begin

 f: =1; i: =2;

 while i<=10 do

 begin

 f: =f\*i;

 i: =i+1;

 end;

 writeln('10!=', f);

end.

program faktor2; {с постусловием}

uses crt;

 var f, i: integer; {факториал, счетчик}

begin

 f: =1; i: =2;

 repeat

 f: =f\*i;

 i: =i+1;

 until i>10;

 writeln('10!=', f);

end.

program faktor1; {с параметром}

 uses crt;

 var f, i: integer; {факториал, счетчик}

begin

 f: =1;

 for i: =2 to 10 do

 f: =f\*i;

 writeln('10!=', f);

end.

####  7.3(6.3.2~)

program uraw\_tg;

uses crt;

 const e=0.00001; {точность}

 var x: real; {середина отрезка}

 l: real; {левый конец отрезка}

 r: real; {правый конец отрезка}

begin

 l: =0.001; r: =1.5;

 repeat

 x: =(l+r)/2;

 if sin(x)/cos(x)<x then l: =x

 else r: =x;

 until r- l>e;

 x: =(l+r)/2;

 writeln('x=', x: 4: 5);

end.

####  7.4(6.3.3~)

program geron;

uses crt;

var x, y: real; {аргумент, значение функции}

 i : integer; {счетчик}

begin

 write('Введите x=');

 readln(x);

 y: =1;

 for i: =2 to 11 do

 y: =y\*x+i;

 writeln('y=', y: 10: 4);

end.

####  7.5(6.3.5~)

program sumcif;

uses crt;

var n, i, j, k, l: integer; {сумма цифр, цифры, счетчик}

begin

 write('Введите сумму цифр n=');

 readln(n);

 l: =0;

 for i: =1 to 9 do

 for j: =0 to 9 do

 for k: =0 to 9 do

 if i+j+k=n then

 begin

 write(100\*i+10\*j+k: 4);

 l: =l+1;

 end;

 writeln;

 writeln('Всего ', l, ' чисел');

end.

####  7.6(6.4.1~)

program latup;

var ch: char; {символ}

begin

 for ch: ='A' to 'Z' do

 write(ch: 2);

end.

####  7.7(6.6.1~)

program cifrbukw;

uses crt;

const n=80;

var x: array[1..n] of char;

 c: char; {вводимый символ}

 i, k: integer;

begin

 k: =0;

 for i: =1 to n do

 begin

 read(c);

 if (c>='0') and (c<='9') then write(c) {печать цифр}

 else

 begin

 k: =k+1;

 x[k]: =c; {запоминание букв}

 end;

 end;

 for i: =1 to k do

 write(x[i])

end.

**7.8(7.6.4~)**

для 6.6.4.1

program rewers;

uses crt;

const n=100; {ограничитель}

var x: array[1..n] of real;

 i, k: integer; {счетчики}

 r : real; {переменная для обмена значений}

begin

 for i: =1 to n do{загрузка}

 read(x[i]);

 for k: =1 to n div 2 do {преобразование}

 begin

 r: =x[k]; x[k]: =x[n+1- k]; x[n+1- k]: =r;

 end;

 for k: =1 to n do {распечатка}

 write(x[k]: 6: 2);

end.

для 6.6.4.2 достаточно блок {преобразование} заменить на следующий:

r: =x[1];

for k: =1 to n- 1 do

 x[k]: =x[k+1];

x[n]: =r;

####  7.9(6.8.2)

program treugol;

uses crt;

 var a, b, c, d: real;

procedure printpl(x, y, z: real);

 var p: real;

begin

 if (x+y>z) and (x+z>y) and (y+z>x) then

 begin

 p: =(x+y+z)/2;

 writeln('S=', sqrt(p\*(p- x)\*(p- y)\*(p- z)): 6: 2);

 end;

 end;

{main file}

begin

 writeln('Введите длины отрезков');

 read(a, b, c, d);

 printpl(a, b, c);

 printpl(a, b, d);

 printpl(a, c, d);

 printpl(b, c, d);

end.

## Функциональные клавиши.

### Простое нажатие.

 F1 - вызов подсказки

 F2 - запись в файл из активного окна

 F3 - открытие файла

 F4 - выполнение программы до позиции курсора

 F5 - раскрытие окна на весь экран

 F6 - активизация следующего окна

 F7 - трассировка программы

 F8 - трассировка без заходов в подпрограммы

 F9 - компилирование начиная с основного файла

 F10 - вызов главного меню

### Режим Shift+

 Shift+F1 - помощь по подсказке

 Shift+F2 -

 Shift+F3 -

 Shift+F4 -

 Shift+F5 -

 Shift+F6 - предыдущее окно

 Shift+F7 -

 Shift+F8 -

 Shift+F9 -

 Shift+F10 -

### Режим Alt+

 Alt+F1 - предыдущее окно подсказки

 Alt+F2 -

 Alt+F3 - закрытие активного окна

 Alt+F4 -

 Alt+F5 - активизация окна результатов выполнения программы

 Alt+F6 - возврат в предыдущее окно

 Alt+F7 - предыдущая строка окна сообщения

 Alt+F8 - следующая строка окна сообщения

 Alt+F9 - компиляция программы из активного окна

 Alt+F10 - вызов локального меню

 Alt+BkSp - отменить изменения в строке

 Alt+C - compile

 Alt+D - debug

 Alt+E - edit

 Alt+F - file

 Alt+H - help

 Alt+O - options

 Alt+R - run

 Alt+S - search

 Alt+T - tools

 Alt+W - window

 Alt+X - exit

 Alt+0 - список окон

### Режим Ctrl+

 Ctrl+F1 - получение контекстной подсказки

 Ctrl+F2 - остановка отладки программы

 Ctrl+F3 - вызов использованных подпрограмм

 Ctrl+F4 - вычисление или модификация выражений

 Ctrl+F5 - установка размера и позиции активного окна

 Ctrl+F6 -

 Ctrl+F7 - добавить контролируемый параметр

 Ctrl+F8 - точка останова

 Ctrl+F9 - компиляция и выполнение программы

 Ctrl+F10 -

## Встроенный ассемблер

 AH 8- битовый регистр данных (старший байт AX )

 AL 8- битовый регистр данных (младший байт AX )

 AND логическое И

 AX 16- битовый регистр данных

 BH 8- битовый регистр данных (старший байт BX )

 BL 8- битовый регистр данных (младший байт BX )

 BP 16- битовый регистр - указатель базы

 BX 16- битовый регистр данных

 BYTE 1 байт

 CH 8- битовый регистр данных

 CL 8- битовый регистр данных

 CS 16- битовый регистр сегмента комманд

 CX 16- битовый регистр данных

 DH 8- битовый регистр данных

 DI 16- битовый регистр - индекс приемника

 DL 8- битовый регистр данных

 DS 16- битовый регистр сегмента данных

 DWORD 4 байта

 DX 16- битовый регистр данных

 ES 16- битовый регистр дополнительного сегмента

 FAR фара дальней адресации

 HIGH старший байт

 LOW младший байт

 MOD остаток от деления

 NEAR ближняя адресация

 NOT логическое НЕ

 OFFSET смещение адреса сегмента

 OR логическое ИЛИ

 PTR задание соответствия типов

 QWORD 8 байт

 SEG адрес сегмента

 SHL сдвиг влево

 SHR сдвиг вправо

 SI 16- битовый регистр индекс источника

 SP 16- битовый регистр указатель стека

 SS 16- битовый регистр сегмента стека

 ST стек сопроцессора

 TBYTE 10 байт

 TYPE тип данных размер в байтах

 WORD 2 байта

 XOR логическое ИСКЛЮЧАЮЩЕЕ ИЛИ

#  СПИСОК РЕКОМЕНДУЕМОЙ ЛИТЕРАТУРЫ.

#### 1. mkSoft EXPRESS PASCAL V.2. Описание языка и руководство пользователя. МикС. 1994.

#### 2. Ж.Джонс, К.Харроу. Решение задач в системе Турбо Паскаль. Москва. ФиС. 1991.

#### 3. Е.Зуев. Программирование на языке TURBO PASCAL 6.0 и 7.0. Москва. РиС. 1993.

#### 4. В.Тумасонис и др. ПАСКАЛЬ. Руководство для программиста.Москва. РиС. 1992.

#### 5. В.Пильщиков. Сборник упражнений по языку Паскаль. Москва.Наука. 1989.

#### 6. А. Брудно, Л.Каплан. Московские математические олимпиады по программированию. Москва. Наука. 1990.

#### 7. В. Дагене и др. 100 задач по программированию. Москва.Просвещение. 1993 г.

#### 8. Т. Поддубная, И. Фукс. Информатика в задачах и упражнениях.Томск. МП "Раско". 1992 г.

#### 9. Н. Красовский. Школьник и КОМПЬЮТЕР: Учимся друг у друга.Москва. Наука. 1993 г.

#### 10. М.В. Сергиевский, А.В. Шалашев. ТУРБО ПАСКАЛЬ 7.0, язык, Среда программирования. Москва. Машиностроение. 1994 г.

#### 11. А.Д.Игошев, Г.Н.Кузнецова EXPRESS PASCAL для КУВТ "Корвет".

#### 12. Н.Рубенкинг ТУРБО ПАСКАЛЬ для WINDOWS.

#### 13. А.М. Епанешников, В.А. Епанешников. ПРОГРАММИРОВАНИЕ В СРЕДЕ TURBO PASCAL 7.0.Москва. Диалог МИФИ. 1995

***ОГЛАВЛЕНИЕ.***

1 Основная часть 4

1.1 Алфавит языка. 4

1.1.1 Символы используемые в именах файлов, подпрограмм, переменных и т.п. 4

1.1.2 Разделители. 4

1.1.3 Знаки пунктуации: 4

1.1.4 Знаки операций. 4

1.1.5 Зарезервированные слова. 5

1.1.6 Неиспользуемые символы. 7

1.2 Структура программы на языке Паскаль. 7

1.3 Типы данных. 8

1.3.1 Простые типы данных. 8

1.1.2 Структурированные типы. 9

1.1.3 Строковый тип string. 10

1.1.4 Запись, тип record. 10

1.1.5 Множество, тип set. 11

1.1.6 Файловый тип. 11

1.4 Понятие переменной. 11

1.4.1 Имя переменной. 11

1.4.2 Типы переменных 12

1.4.3 Значение переменной. 12

1.5 Константы. 12

1.5.1 Целые. 12

1.5.2 Вещественные. 13

1.5.3 Строковые и символьные. 13

1.5.4 Константные выражения. 13

1.1.5 Типизированные константы. 13

1.6 Операторы 13

1.6.1 Простые операторы. 13

1.6.2 Структурированные операторы. 14

1.7 Подпрограммы. Системный подход к программированию в системе TP. 19

1.7.1 Декомпозиция. 19

1.1.2 Подпрограммы-функции. 20

1.1.3 Подпрограммы-процедуры. 21

1.8 Процедурные типы. 25

1.9 Совместимости преобразования типов данных. 25

1.9.1 Идентичность типов. 25

1.9.2 Совместимость типов. 26

1.9.3 Совместимость для присваивания. 26

1.10 Директивы подпрограмм. 26

1.11 Внешние устройства в качестве файлов. 27

1.12 Модули 27

1.13 Стандартные модули. 28

1.13.1 System. 28

1.13.2 Модуль Strings 34

1.13.3 Модуль CRT 36

1.13.4 Модуль GRAPH 37

1.13.5 Модуль DOS 45

1.13.6 Модуль WINDOS 48

1.13.7 Модуль OVERLAY 50

1.13.8 13.8. Модуль Printer . 51

1.14 14. Работа в главном меню. 51

1.14.1 Меню работы с файлами File. 52

1.14.2 Меню редактирования Edit. 52

1.1.3 Меню поиска Search. 52

1.14.4 Меню выполнения программы (RUN) 53

1.14.5 Меню компиляции (COMPILE) 54

1.14.6 Меню отладки (Debug) 54

1.14.7 Меню инструментальных средств (Tools) 55

1.14.8 Меню параметров среды (Options) 55

1.14.9 Меню окон Window. 56

1.14.10 Меню помощи Help. 56

2 Приложения. 57

2.1 Перемещение курсора. 57

2.2 Работа с блоками 57

2.2.1 Выделение блока 57

2.2.2 Команды компилятора 58

2.2.3 Ключи параметров 59

3 Образцы решения задач 59

3.1.1 Рассмотрим задачу по нахождению наибольшего значения ряда для следующей задачи: В трех классах (не более 25 уч- ся) проведены экзамены. Нужно распечатать протокол в виде: 59

3.1.2 Распечатать все допустимые символы. 61

3.1.3 Ввести с клавиатуры строку (Не более 20 символов) и выдать ее на экран выбегающим текстом в центр экрана. 62

3.1.4 По введенным корням квадратного уравнения составить приведенное квадратное уравнение. 63

3.1.5 Ввести с клавиатуры число в интервале от 1 до 27 и распечаать все трехзначные числа, сумма цифр которых равна введенному числу. 63

3.1.6 Найти все совершенные числа не превосходящие введенного с клавиатуры. Число называется совершенным, если оно равно сумме своих делителей, исключая себя. 64

3.1.7 Распечатать все тройки пифагоровых чисел, не превосходящие введенного с клавиатуры. При выводе не допус 64

3.1.7 кать повторений полученных на основе переместительного закона сложения. Тройка чисел a, b и c называется пифагоровой, если выполняется равенство: a2+b2=c2. 64

3.1.8 Вычислить факториалы чисел, не превосходящих введенного с клавиатуры. 66

3.1.9 Вывести на экран таблицу умножения. 66

3.1.10 Ввести с клавиатуры размер одномерного ряда(не более 25 элементов). Заполнить его с клавиатуры целыми числами. Распечатать ряд в строку. Найти и вывести на экран наименьший и наибольший элементы, с указанием их индексов. Упорядочить ряд по возрастанию, используя алгоритм линейной сортировки или сортировку методом "пузырька". Распечатать сортированные данные в одну строку. 67

3.1.11 Ввести с клавиатуры количество колонок n и количество строк m двумерного ряда. Заполнить ряд случайными целыми числами. Вывести ряд на экран в виде прямоугольной таблицы, оформив ее нумерацией строк и колонок. Найти наибольший и наименьший элементы с указанием адресов. Провести сортировку ряда по возрастанию и в строках, и в столбцах. Вывести на экран сортированные данные. 70

3.1.12 Построить график функции y=a\*sin(b\*x+c). Предусмотреть воз- ожность замены функции на другую, а так же возможность совмещеия графиков одной функции при различных значениях параметров. 73

3.1.13 (Шуплецов В.Ю.) Вывести на экран иммитацию электронных часов. 75

3.2 Задачи для самостоятельного решения. 83

3.2.1 Линейные программы. 83

3.2.2 Операторы: условный, составной, пустой, перехода. 84

3.2.3 Оператор цикла. 86

3.2.4 Литерный тип. 88

3.2.5 Перечисляемые и ограниченные типы. 91

3.2.6 Одномерные массивы. 91

3.2.7 Двумерные массивы. 92

3.2.8 Функции и процедуры. 95

3.2.9 Графические возможности 97

3.3 Ответы на задачи помеченные знаком~. 99

3.4 Функциональные клавиши. 103

3.4.1 Простое нажатие. 103

3.4.2 Режим Shift+ 104

3.4.3 Режим Alt+ 104

3.4.4 Режим Ctrl+ 104

3.5 Встроенный ассемблер 105

4 СПИСОК РЕКОМЕНДУЕМОЙ ЛИТЕРАТУРЫ. 107