Департамент образования, науки и молодёжной политики Воронежской области

ГБПОУ ВО «Воронежский государственный профессионально-педагогический колледж»

Методические указания к практическим работам по дисциплине «Основы алгоритмизации и программирования»

Савченко Е.А., Польников П.М.

Учебно-методическое пособие

Рекомендовано

советом учебно-методического цента

в качестве учебно-методического пособия по дисциплине «Информационные технологии» для студентов колледжа

специальности 051001 Профессиональное обучение (по отраслям),

профиля подготовки 230113 «Компьютерные системы и комплексы»

Воронеж, 2016 г.

УДК 373
ББК 74.57
С 12
Рецензенты:
Желобкова А.А. заведующая отделением ИТ ГБПОУ ВО «ВГППК»

Савченко Е.А., Польников П.М. Методические указания к практическим работам по дисциплине «Основы алгоритмизации и программирования»: учебно-методическое пособие для студентов СПО – Воронеж; ВГППК, 2016г.

Учебно-методическое пособие по дисциплине ОП.09 «Основы алгоритмизации и программирования» разработано на основе Федерального государственного образовательного стандарта по специальности среднего профессионального образования 051001 Профессиональное обучение (по отраслям) специальности профиля подготовки 230113 Компьютерные системы и комплексы.

Учебно-методическое пособие предназначено для студентов колледжа специальности 051001 Профессиональное обучение (по отраслям), профиля подготовки 230113 «Компьютерные системы и комплексы»

.Печатается по решению совета учебно-методического центра ГОБУ СПО ВО «Воронежский государственный профессионально-педагогический колледж»

Оглавление

Введение

Внедрение новых информационных технологий во все сферы современной жизни привело к тому, что умение работать на компьютере является необходимым атрибутом профессиональной деятельности любого специалиста и во многом определяет уровень его востребованности в обществе.

К специалистам по компьютерным технологиям предьявляются особые требования: знание языков программирования и умение сотавлять компьютерные программы любой сложности.

Знакомство с языком программирования начинается с изучения алфавита и структуры языка программирования. Не зная теоретических основ нельзя научиться программировать.

При проведении работ студенты пользуются подробными инструкциями, в которых указаны: тема, цель работы, порядок выполнения работы.

Работа с данным учебно-методическим пособием способствует формированию у студентов общих и профессиональных компетенций определенных федеральным государственным образовательным стандартом для дисциплины «Основы алгоритмизации и программирования»:

ОК 1. Понимать сущность и социальную значимость своей будущей профессии, проявлять к ней устойчивый интерес.

ОК 2. Организовывать собственную деятельность, определять методы решения профессиональных задач, оценивать их эффективность и качество.

ОК 3. Оценивать риски и принимать решения в нестандартных ситуациях.

ОК 4. Осуществлять поиск, анализ и оценку информации, необходимой для постановки и решения профессиональных задач, профессионального и личностного развития.

ОК 5. Использовать информационно-коммуникационные технологии для совершенствования профессиональной деятельности.

ОК 6. Работать в коллективе и команде, взаимодействовать с руководством, коллегами и социальными партнерами.

ОК 7. Ставить цели, мотивировать деятельность обучающихся, организовывать и контролировать их работу с принятием на себя ответственности за качество образовательного процесса.

ОК 8. Самостоятельно определять задачи профессионального и личностного развития, заниматься самообразованием, осознанно планировать повышение квалификации.

ОК 9. Осуществлять профессиональную деятельность в условиях обновления ее целей, содержания, смены технологий.

ОК 10. Осуществлять профилактику травматизма, обеспечивать охрану жизни и здоровья обучающихся.

ОК 11. Строить профессиональную деятельность с соблюдением правовых норм ее регулирующих.

ОК 12. Исполнять воинскую обязанность, в том числе с применением полученных профессиональных знаний (для юношей).

ПК 1.1. Определять цели и задачи, планировать занятия.

ПК 1.3. Проводить лабораторно-практические занятия в аудиториях, учебно-производственных мастерских и в организациях.

ПК 1.4. Организовывать все виды практики обучающихся в учебно-производственных мастерских и на производстве.

ПК 2.2. Определять цели и задачи, планировать деятельность по педагогическому сопровождению группы обучающихся.

ПК 2.3. Организовывать различные виды внеурочной деятельности и общения обучающихся.

ПК 2.4. Осуществлять педагогическую поддержку формирования и реализации обучающимися индивидуальных образовательных программ.

ПК 3.2. Систематизировать и оценивать педагогический опыт и образовательные технологии в области начального профессионального образования и профессиональной подготовки на основе изучения профессиональной литературы, самоанализа и анализа деятельности других педагогов.

Практическая работа № 1, 2 (4 часа)

Тема: Выражения в паскаль

Цель работы : Ознакомиться с системой программирования ABC Pascal, получить основные навыки работы с ней, освоить приёмы создания, компиляции и исполнения программы, научиться создавать программы для вычисления выражений.

Ход работы:

Теоретическое обоснование:

Язык программирования Паскаль был разработан Норбертом Виртом в 1971 году. Швейцарский профессор Норберт Вирт создал язык Паскаль как учебный язык структурного программирования.

Наибольший успех в распространении этого языка обеспечили персональные компьютеры. Фирма Borland International, Inc (США) разработала Систему программирования Турбо Паскаль для ПК. Турбо Паскаль – это не только язык и транслятор с него, но еще и операционная оболочка, обеспечивающая пользователю удобство работы. Турбо Паскаль вышел за рамки учебного предназначения и стал профессиональным языком.

По сути дела расхождения между алгоритмическим языком и языком Паскаль заключается в следующем: алгоритмический язык – русскоязычный, язык Паскаль – англоязычный. Синтаксис языка Паскаль определен строго и однозначно в отличие от сравнительно свободного синтаксиса алгоритмического языка.

Структура программы на языке Паскаль.

Программа состоит из заголовка программы, раздела описаний и раздела операторов. В конце программы ставится точка.

Program < имя программы>; – заголовок программы

Label < раздел меток>;

Const < раздел констант>;

Type < раздел типов >; – раздел описаний

Var < раздел переменных >;

Procedure (Function) < раздел подпрограмм>;

Begin
действия;
} – раздел операторов

End.

Заголовок программы начинается со слова Program, за которым следует произвольное имя, придуманное программистом.

Раздел описаний может содержать не все пункты.

Раздел операторов имеется в любой программе и является основным. Начало и конец раздела операторов отмечаются словами Begin (начало) и End (конец). Все команды в разделе операторов отделяются друг от друга точкой с запятой. В конце программы обязательно ставится точка.

Описание переменных.

Раздел описания переменных начинается со слова Var, за которым следует список переменных. Тип переменной указывается после двоеточия.

VAR имя переменной:тип;

В языке Паскаль используются следующие типы переменных:

INTEGER – целый (значениями являются целые числа)

REAL – вещественный (значениями являются целые и дробные числа)

CHAR – символьный (значениями являются символы, например ‘+’, ‘e’)

STRING – строковый (значениями являются строки символов, например ‘+*/+*+’, ‘мама’)

BOOLEAN – логический (принимает значения TRUE – истина и FALSE – ложь)

Пр. Var a:integer;
b:real;
c:string;

Описание констант.

CONST название константы=значение;

Пр. Const n=10;
M=’+’;

Константа – переменная, которая не меняет свое значение во время выполнения программы.

Команда ввода.

READ(названия переменных); – курсор после ввода остается в той же строке. или

READLN(названия переменных); – курсор после ввода переходит на следующую строку.

Пр. read(a,b);

readln(s,d);

Команда вывода.

WRITE(названия переменных или фраза); – курсор после вывода остается в той же строке. или

WRITELN(названия переменных или фраза); – курсор после вывода переходит на следующую строку.

Если переменная вещественного типа, то при выводе указывают формат вывода.

название переменной:0:количество цифр после запятой

Пр. write(a,d);

writeln('сумма=',s:0:2);

Арифметические операции в языке Паскаль.

Знак	Выражение	Типы операндов	Типы результатов	Операция
+	A+B	вещ., вещ. цел., цел. цел., вещ. вещ., цел.	вещественный целый вещественный вещественный	Сложение
-	A-B	вещ., вещ. цел., цел. цел., вещ. вещ., цел.	вещественный целый вещественный вещественный	Вычитание

*	A*B	вещ., вещ. цел., цел. цел., вещ. вещ., цел.	вещественный целый вещественный вещественный	Умножение
/	A/B	вещ., вещ. цел., цел. цел., вещ. вещ., цел.	вещественный вещественный вещественный вещественный	Вещественное деление
div	A div B	цел., цел.	целый	Целое деление
d	A mod B	цел., цел.	целый	Остаток от целого деления.

Стандартные функции языка Паскаль.

Обращение	Тип аргумента	Тип результата	Функция
Pi	----	вещественный	Число П. = 3,141592
Abs (x)	цел., вещ.	цел., вещ.	Модуль аргумента x
Sqr(x)	I,R	I, R	Квадрат x
Sqrt(x)	I,R	R	Корень квадрат. из x
Sin(x)	вещ.	вещественный	Синус x в радианах
Cos(x)	вещ.	вещественный	Косинус x (в радианах)
Exp(x)	цел., вещ.	вещественный	ex- экспонента
Ln (x)	цел., вещ.	вещественный	Натуральный логарифм x

В языке Паскаль нет стандартной операции возведение в степень, поэтому при возведении в вещественную степень пользуются формулой: $x^y = e^{y \ln x}$.

Пр. $x^{2,3} = \exp(2.3 * \ln(x))$

Если y – целое значение, то степень вычисляется через умножение, например: x3= x*x*x; большие степени следует вычислять умножением в цикле.

Работа в среде программирования PascalABC.

Загрузка АВС Паскаля

Загрузите АВС Паскаль с помощью ярлыка на рабочем столе или из главного меню ПУСК.

Краткое знакомство с интегрированной средой

После загрузки системы экран разделен на три части:

1. служебную область;

2. окно ввода алгоритма программы - рабочее окно;

3. окно ввода данных и вывода результатов программы.

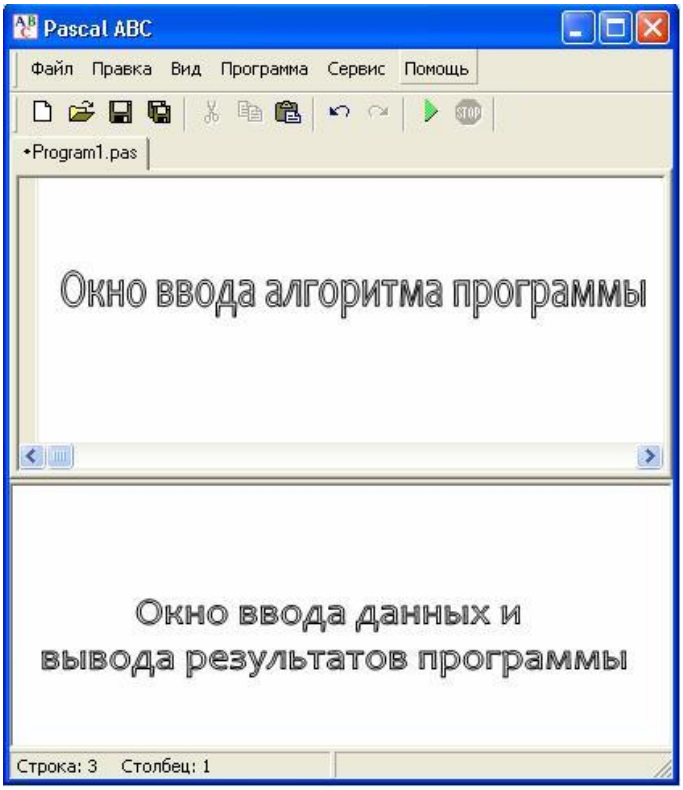


Рис. 1

Переход из основного окна в главное меню и обратно осуществляется посредством клавиши <F10>.

Первая программа

В рабочем окне редактора интегрированной среды наберем текст первой программы.

```
Program Myl_l ;
Var a, b, rez : Integer;
Begin
WriteLn ('Введите два числа через пробел');
```

```
  ReadLn (a, b);
  rez :=a*b;
  WriteLn ('Их произведение равно ', rez);
  WriteLn ('Нажмите <Enter>');
  ReadLn
End.
```

Подсказка: можно выделить текст программы мышкой, нажать правую кнопку мыши и выбрать Копировать.

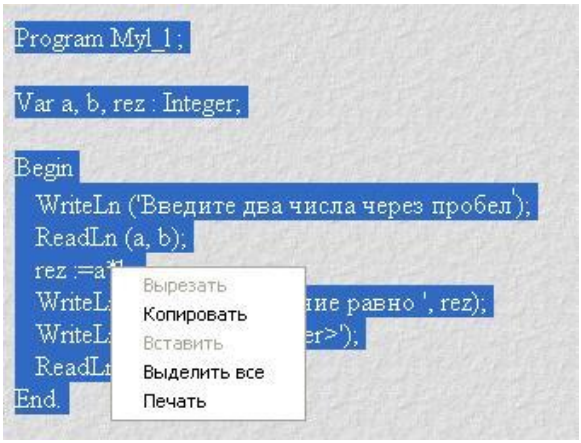


Рис. 2

Затем в системе Pascal ABC выбрать команды Правка/Вставить.

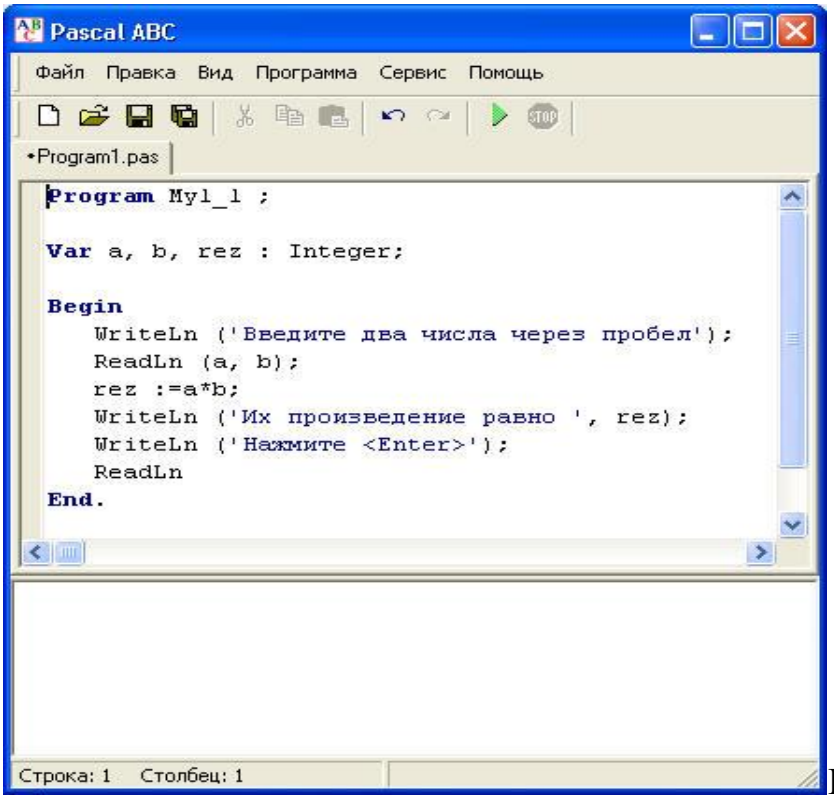


Рис. 3

В результате вы получите текст программы в рабочем окне.

Компиляция программы

Компилятор – специальная программа, с помощью которой исходный текст программы переводится в исполняемый двоичный файл программы.

Компиляция – процесс получения исполняемого двоичного файла из исходного текста программы, написанного на языке программирования.

Компиляция в Pascal-е происходит автоматически.

Запуск программы

Для того чтобы запустить программу, в главном меню выбираем пункт Программа/Выполнить или нажимаем значок  . На экране в нижней части окна появляется сообщение:

Введите Два целых числа через пробел

Курсор находится в следующей строке. Вводим два целых числа через пробел и нажимаем < Enter>, после этого появляется сообщение:

Их произведение равно …

нажмите <Enter>

Вместо точек будет выведено значение переменной rez, то есть число, равное произведению первого введенного числа на второе. Это сообщение останется на экране до тех пор, пока не будет нажата клавиша <Enter>.

Сохранение программы

Для того чтобы сохранить программу, необходимо:

- выйти в главное меню и выбрать режим Файл;
- в вертикальном меню выбрать пункт Сохранить как… ;
- в появившемся окне выбрать папку Children, затем выбрать свою папку (например, Ivanov), ввести имя файла (например, prim1_1) и нажать клавишу <Сохранить>.

Примечания.

1. В именах файлов нельзя употреблять следующие символы: *, =, +, [,], \, |, :, ., <, >, /, ?, символ пробела и буквы русского алфавита.

2. Для быстрого сохранения файла можно воспользоваться командами Сохранить или Сохранить все меню Файл.

Выход из системы программирования Pascal ABC

Для того чтобы закончить работу, необходимо:

- выйти в главное меню и выбрать пункт Файл;

• в вертикальном меню выбрать пункт Выход;

. • или нажать на кнопку  в правом углу строки заголовка.

Эксперименты с программой

1. Введите в качестве исходных данных достаточно большие числа, например, 4567 и 789. Убедитесь, что у вас получается неправдоподобный результат — отрицательное число (—1117). Найдите экспериментальным путем тот интервал значений переменных a и b, когда результат умножения правильный.

2. Вместо числа введите какой-нибудь символ. Убедитесь, что компьютер выдает сообщение об ошибке.

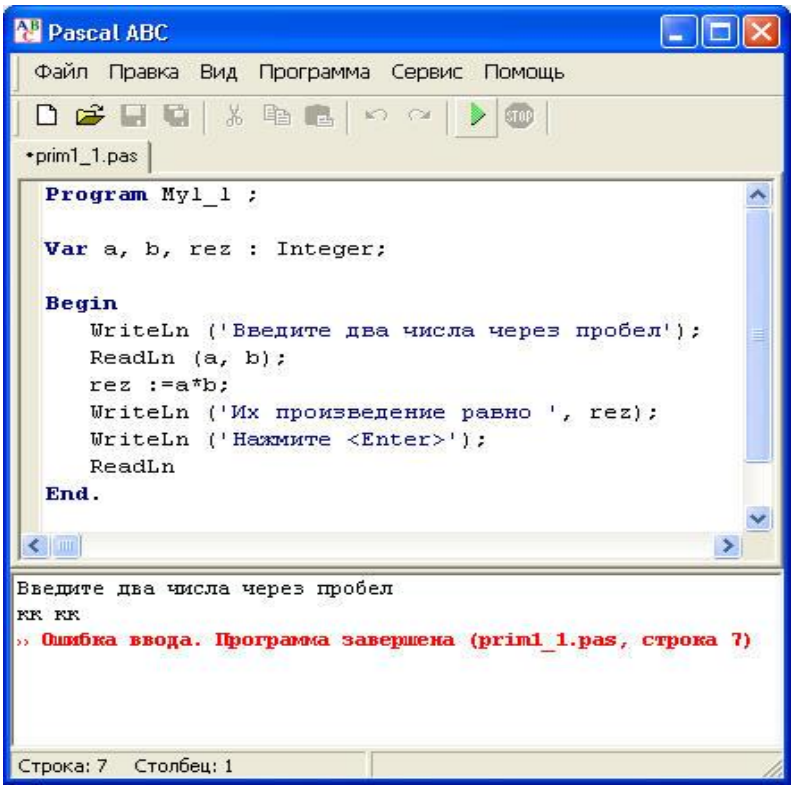


Рис. 4

3. Добавьте лишний знак апострофа в операторе WriteLn. Убедитесь, что программа не проходит компиляцию, а система сообщает об ошибке.

4. Измените оператор rez :=a*b на rez:=a*(a + b). Выполните алгоритм.

Примеры задач.

Даны стороны прямоугольника. Найти периметр и площадь.

```
Program prim;
    Var a,b,p,s: real;
    Begin
        writeln('введите стороны прямоугольника');
        readln (a,b);
        p:=(a+b)*2;
        s:=a*b;
        writeln('периметр=',p:0:2);
        writeln ('площадь=',s:0:2);
    End.
```

Даны 2 числа. Найти их сумму и разность.

```
    Program chisla;
    Var a,b,p,s: real;
    Begin
        writeln('введите 2 числа');
        readln (a,b);
        p:=a-b;
        s:=a+b;
        writeln('разность=',p:0:2);
        writeln ('сумма=',s:0:2);
    End.
```

Даны 3 числа. Найти сумму их кубов.

```
Program chisla;
    Var a,b,c,s: real;
```

```
Begin
    writeln('введите 3 числа');
    readln (a,b,c);
    s:=a*sqr(a)+b*sqr(b)+c*sqr(c);
    writeln('сумма кубов=',s:0:2);
End.
```

Дано целое число x. Вычислить $y=\sin(3x+8)$, $z=\ln y$, $k=\frac{x^2+13}{e^{2x}}$, $m=\sqrt{x+y-z}$, $n=x^3-|k-2|$

```
Program chisla;
Var x:integer;
     y,z,k,m,n: real;
Begin
    writeln('введите целое число x');
  readln (x);
  y:=sin(3*x+8);     z:=ln(y);
    k:=(sqr(x)+13)/exp(2*x);
    m:=sqrt(x+y-z);
    n:=x*sqr(x)-abs(k-2);
    writeln('y=',y:0:2,'   z=',z:0:2, '     k=',k:0:2);
    writeln ('m=',m:0:2, '    n=',n:0:2);
End.
```

Задания для самостоятельного решения

Вариант №1

Даны целые числа a, b, c. Вычислить их полусумму и произведение.

Даны стороны прямоугольника. Найти периметр и площадь прямоугольника.

Вычислить значение функции $y=\sqrt{x^2+8}$.

Вычислить значение функции $y=e^{x^2-8}+3$.

Найти сумму цифр трехзначного числа.

Вариант №2

Даны действительные числа x, y. Вычислить их полуразность и частное.

Дана сторона квадрата. Найти периметр, площадь и диагональ квадрата.

Вычислить значение функции $y = x^2 - 7x + 10$.

Вычислить значение функции $y = \cos^2(x+3)$.

Найти сумму цифр четырехзначного числа.

Вариант №3

Даны действительные числа c, d. Вычислить их полусумму и произведение.

Дан радиус окружности. Найти диаметр и длину окружности, площадь круга.

Вычислить значение функции $y = x^2 + \sqrt{x-1}$.

Вычислить значение функции $y = |x+3|$.

Найти сумму цифр пятизначного числа

Вариант №4

Даны целые числа a и b. Найти полусумму их квадратов.

Даны стороны треугольника. Найти периметр и площадь треугольника.

Вычислить значение функции $y = \sqrt{x} + 3x$.

Вычислить значение функции $y = \ln(x^2 + 4)$.

Найти сумму цифр шестизначного числа

Вариант №5

1. Ширина стороны прямоугольника B см, а длина в 2 раза больше. Найти площадь

2. Скорость первого автомобиля v1 км/ч, второго — v2 км/ч, расстояние между ними s км. Какое расстояние будет между ними через t ч, если автомобили движутся в разные стороны?

3.Вычислить значение функции $c = x - \ln x + \dfrac{y}{\cos x - \dfrac{x}{3}}$

4.Вычислить значение функции $c=\sin\sqrt{x+1} - \sin\sqrt{x-1}$

5. Дано натуральное четырехзначное число N. Найти разницу между сумой цифр и произведением цифр данного числа

Контрольные вопросы

1. Как в программе на языке Pascal описываются переменные?
2. Какие бывают типы переменных?
3. Какой вид имеет оператор присваивания?
4. Каким символом отделяются друг от друга операторы в программе?
5. В каких случаях после оператора не ставятся точка с запятой?
6. Какая процедура служит для вывода информации на печать?
7. Какая процедура служит для ввода значений с клавиатуры?
8. Какие функции служат для вычисления квадрата, квадратного корня, модуля, экспоненты числа или числового выражения?
9. Какие стандартные тригонометрические функции существуют в языке Pascal?
10. Как в среде Pascal запустить программу на выполнение?

Практическая работа № 3, 4 (4 часа)

Тема: Составление программ линейной структуры в Pascal/

Цель работы: Ознакомиться с операциями div и mod. Научиться составлять программы с использованием данных операций.

Ход работы:

Операции div и mod.

Целый тип данных

Теоретическое обоснование:

Переменные целого типа описываются посредством идентификатора Integer. Они могут принимать значения в диапазоне от -32768

до 32767. К данным целого типа можно применить операции "+"-сложение, "-"-вычитание, "*"-умножение и некоторые другие.

Так как в результате деления одного целого числа на другое не всегда получается целое число, то имеются операции:

div - целая часть от деления;

mod - остаток от деления.

Иногда нам требуется найти частное либо же остаток от деления. В такие моменты на помощь нам приходят такие операции, как div и mod. Заметим, что эти операции выполняются только над целыми числами.

Div Для того, чтобы найти частное от деления, мы используем операцию div.

Примеры:

25 div 20 = 1;

20 div 25 = 0;

39 div 5 = 7;

158 div 3 = 52.

Mod Для того, чтобы найти остаток от деления, мы используем операцию mod.

Примеры:

25 mod 20 = 5;

20 mod 25 = 0;

39 mod 5 = 4;

158 mod 3 = 2.

Примечание. Переменной целого типа присваивать значение, получаемое в результате выполнения обычной операции деления "/", нельзя, так как при делении одного числа на другое целое число результат не всегда является целым числом. При использовании операторов div и mod переменные описываются как переменные целого типа, т.е. ,например, integer или longint.

Чтобы окончательно понять, с чем мы имеем дело, решим следующую задачу:

Пример 1. Заданы два целых числа k и d. Используя только арифметические операции, найдите целую и дробную части от деления k на d

Этапы выполнения задания.

I. Определение исходных данных: переменные k,d.

II. Определение результатов: переменные cel, drobn.

III. Алгоритм решения задачи.

1. Ввод исходных данных

2. Вычисление значений cel=k div d, a=k/d, drobn=a-cel.

3. Вывод результата.

IV. Описание переменных:

Переменные a,drobn имеют тип real, a переменные k,d,cel имеют типinteger..

V. Программа:

```
var a,drobn:real;
k,d,cel:integer;
Begin
writeln('введите два числа'); readln(k,d);
cel:=k div d;                    {целочисленное деление}
a:=k/d;                          {обычное деление}
drobn:=a-cel;
writeln('целая часть - ',cel);
writeln('дробная часть - ',drobn);
End.
```

VI. Тестирование

1. Запустите программу и введите значения k=2, d=3

Проверьте, результат должен быть следующим:

cel= 1, drobn= 0.666666666667

Проверить правильность вычислений можно на калькуляторе.

Пример 2. Дано трехзначное число. Чему равны его цифры?

Этапы выполнения задания.

I. Определение исходных данных: переменная a (трехзначное число).

II. Определение результатов: переменные c1,c2,c3 (цифры числа).

III. Алгоритм решения задачи.

1. Ввод исходных данных

2. Выделение цифр числа:

```
c1:=a div 100;
c2:=(a mod 100) div 10;        {или  c2:=(a div 10) mod 10;}
c3:=a mod 10;
```

3. Вывод результата.

IV. Описание переменных:

Все переменные, определенные для решения задачи, имеют тип integer.

V. Программа:

```
var a,c1,c2,c3:integer;
Begin
writeln('введите трёхзначное число '); readln(a);
c1:=a div 100;
c2:=(a mod 100) div 10;        {или  c2:=(a div 10) mod 10;}
c3:=a mod 10;
writeln('первая цифра -',c1);
writeln('вторая цифра -',c2);
writeln('третья цифра -',c3);
End.
```

VI. Тестирование

1. Запустите программу и введите значения

a=234

Проверьте, результат должен быть следующим:

первая цифра - 2

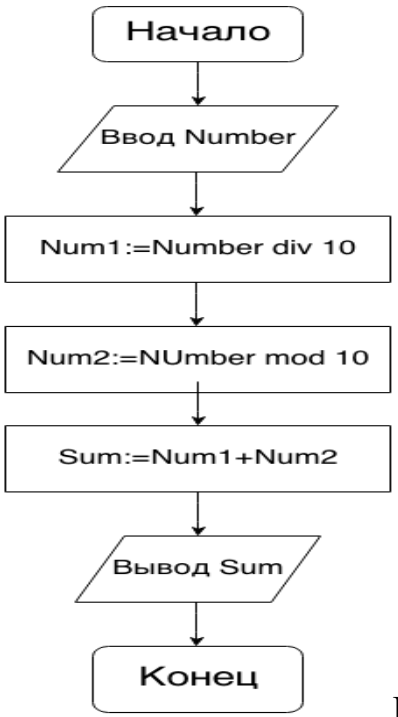
вторая цифра - 3

третья цифра - 4

2. Введите другие числа. Убедитесь в правильности работы программы.

Пример 3. Найти сумму цифр двухзначного числа.

Так как эта задача очень простая, составим блок-схему и программу.



Блок-схема

```
program Sumoftwo;
var Number,Num1,Num2,Sum: integer;
begin
write('Введите двухзначное число: ');
read(Number);  { Возьмем число 25 }
Num1 := Number div 10; { 25 div 10 = 2 }
Num2 := Number mod 10; { 25 mod 10 = 5 }
Sum := Num1 + Num2; { 2 + 5 = 7 }
write('Сумма двух чисел -- ',Sum);
end.
```

Задания для самостоятельного решения

Вариант №1

Дано трехзначное число. Определить: является ли сума его цифр двузначным числом

Дано трехзначное число. Определить: является ли произведение его цифр трехзначным числом

Дано пятизначное число . Нужно вычислить квадрат суммы всех цифр.

Вариант №2

Дано трехзначное число. Определить: кратна ли пяти сумма его цифр

Дано трехзначное число. Определить: кратна ли сумма его цифр числу a

Дано четырехзначное число. Определить: равна ли сумма двух первых его цифр сумме двух его последних цифр

Вариант №3

Дано четырехзначное число. Определить: кратно ли четырем произведение его цифр

Дано четырехзначное число. Определить: кратно ли произведение его цифр числу a

Дано трехзначное число . Нужно вычислить квадрат суммы всех цифр.

Вариант №4

Дано четырехзначное число . Нужно вычислить квадрат суммы всех цифр.

Дано четырехзначное число. Определить: кратна ли трем сума его цифр

Дано трехзначное число. Определить: больше ли числа a произведение его цифр

Вариант №5

Дано трехзначное число. Определить: кратна ли шести сумма его цифр

Дано трехзначное число. Определить: кратно ли произведение его цифр числу a

Дано четырехзначное число. Определить: равно ли произведение двух первых его цифр произведению двух его последних цифр

Контрольные вопросы

Что такое «тип данных»? Для чего нужен тип данных? Какие типы данных есть в Паскале?

Как записываются арифметические операции в языке Паскаль?

Для чего нужен формат вывода данных? Как он задается?

Для чего предназначены операторы div и mod?

Запишите встроенные математические функции языка Паскаль.

Практическая работа № 5

Тема: Составление программ ветвящейся структуры в Pascal

Цель работы: научиться составлять программы для решения задач, содержащих условия, продолжить освоение работы в системе программирования ABC Pascal.

Ход работы:

Теоретическое обоснование

Конструкция ветвления- это часть алгоритма, в которой в зависимости от выполнения или невыполнения некоторого условия выполняется либо одна, либо другая последовательность действий.

Алгоритм, в котором используется конструкция ветвления, называется алгоритмом с ветвлением..

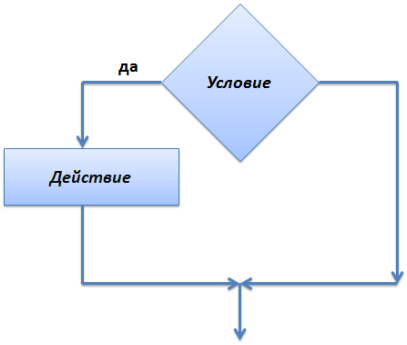
Условие бывает:

Неполное

Полное

Неполное условие

Блок-схема неполного условия выглядит следующим образом:



Неполное условие

ЕСЛИ УСЛОВИЕ ИСТИННО, ТО ВЫПОЛНЯЕТСЯ ДЕЙСТВИЕ, ИНАЧЕ НИЧЕГО НЕ ПРОИСХОДИТ

На языке Паскаль данная алгоритмическая конструкция выглядит следующим образом:

IF условие THEN действие;

Если условие, стоящее после IF истинно, то выполняется действие, стоящее после слова THEN.

Что такое условие?

Условие — это выражение, которое может быть либо истинным, либо ложным. Условие обязательно содержит логические операторы <, >, =, <=, >=, <> (не равно).

Например:

```
var a:integer;
begin
readln(a);
if a=7 then writeln('Привет!');
end.
```

Если введенное значение переменной a равно 7, то на экране мы увидим слово Привет! Если не равно — то на экран ничего выводится не будет.

Рассмотрим другой пример:

```
var a:integer;
begin
```

```
readln(a);
if a=7 then writeln('Привет!'); writeln('До встречи');
end.
```

Что мы увидим на экране, введя число 7? Увидим:

Привет!

До встречи

Что мы увидим на экране, введя число 10?

До встречи

Почему так? Почему До встречи выводится на экран в любом случае? Команда

```
writeln('До встречи');
```

не относится к конструкции If-Then

```
var a:integer;
begin
readln(a);
if a=7 then writeln('Привет!'); writeln('До встречи'); // условие подсвечено голубым цветом
end.
```

Как сделать так, чтобы оба действия относились к конструкции If-Then?

Необходимо заключить эти действия в так называемые операторные скобки: begin… end;

Получим:

```
var a:integer;
begin
readln(a);
if a=7 then begin                    writeln('Привет!');
            writeln('До встречи');
  end;
end.
```

Теперь, если мы введем число 10, то на экране ничего не увидим.

Составное (сложное) условие

Иногда приходится использовать сложное условие. Для его составления используются логические союзы: and или or.

Если мы используем and, то составное условие будет истинно, когда все простые условия истинны.

Если мы используем or, то составное условие будет истинно, когда хотя бы одно простое условие будет истинно.

Например:

a>7 и a<15 на языке паскаль будет записываться

(a>7) and (a<15)

т.е. оба условия должны выполняться одновременно, чтобы составное условие было истинно

Или:

(a=7) or (a>15)

Составное условие будет истинно, если: либо a=7, либо a>17.

Задача:

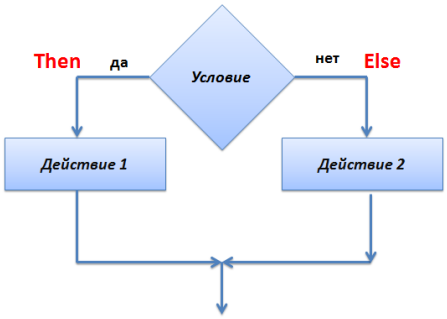
Используя конструкцию If-Then, найти максимальное среди трех введенных чисел.

Решение

```
var a, b, c:integer;
begin
readln(a);
readln(b);
readln(c);
if (a>b) and (a>c) then writeln('число ', a, ' максимальное');
if (b>a) and (b>c) then writeln('число ', b, ' максимальное');
if (c>a) and (c>b) then writeln('число ', c, ' максимальное');
end.
```

Полное условие

Блок-схема полного условия выглядит так:



Полное условие. Блок-схема

IF условие THEN действие_1 ELSE действие_2;

Если условие истинно, то выполняется действие, стоящее после слова Then.

Если условие ложно, то выполняется действие, стоящее после слова Else.

Если действий, которые выполняются, если условие истинно или ложно несколько — используются операторные скобки.

```
if условие then
begin
    действие;
    действие;
.............
end
        else
begin
    действие;
    действие;
.............
end;
```

(перед else точка с запятой не ставится)

Пример. Проверить является ли введенное число трехзначным, и вывести четные цифры числа.

Этапы выполнения задания.

I.Определение исходных данных: переменная a (трехзначное число).

II. Определение результатов: переменные a1, a2, a3, в том случае если они четные или сообщение, что четных цифр нет.

III. Алгоритм решения задачи.

1. Ввод исходных данных

2. Проверка является ли число трехзначным. Трехзначное число больше 99 и меньше 1000.

3 Если число трехзначное, то выделяем цифры числа и проверяем каждую из них на четность.

а) Для выделения первой цифры (переменная a1) трехзначного числа необходимо найти целую часть от деления числа на 100.

б) Для выделения второй цифры (переменная a2) трехзначного числа необходимо найти остаток от деления числа на 100 и от него найти целую часть при делении на 10.

в) Для определения последней цифры (переменная a3) трехзначного числа необходимо найти остаток от деления числа на 10.

г) Для проверки цифры на четность нужно проверить, равен ли нулю остаток от деления цифры на 2.

4. Вывод результата.

IV. Описание переменных:

Все переменные, определенные для решения задачи, имеют типinteger.

V. Программа:

```
Var a,a1,a2,a3: integer;
Begin
   Write(‘введите значение a=’);
   Readln(a);
   If (a>99) and (a<1000) then
   Begin
      a1:=a div 100;
      a2:=a mod 100 div 10;
      a3:=a mod 10;
```

```
    if a1 mod 2=0 then Writeln('цифра ',a1,' четная');
    if a2 mod 2=0 then Writeln('цифра ',a2,' четная');
    if a3 mod 2=0 then Writeln('цифра ',a3, ' четная');
    If (a1 mod 2=1) and (a2 mod 2=1) and (a3 mod 2=1)then
     Writeln('в числе нет четных цифр');
  End
  Else  Writeln('число не является трехзначным');
End.
```

Пример. Написать программу для решения геометрической задачи. Определить, является ли треугольник со сторонами a, b, c равнобедренным. Если "да", то вычислить его периметр.

Этапы выполнения задания.

I.Определение исходных данных: переменные a, b, c.

II. Определение результатов: переменная p, в том случае если треугольник равнобедренный или сообщение, что треугольник не равнобедренный.

III. Алгоритм решения задачи.

1. Ввод исходных данных

2. Проверка является ли треугольник равнобедренным.

3 Если треугольник равнобедренный, то вычислим его периметр p:=a+b+c, иначе выведем сообщение "треугольник не равнобедренный" .

4. Вывод результата.

IV. Описание переменных:

Все переменные, определенные для решения задачи, имеют типinteger.

V. Программа:

```
Var a,b,c,p: integer;
Begin
  Write('введите значения сторон треугольника a,b,c:');
  Readln(a,b,c);
```

```
If (a=b) or (b=c) then
Begin p:=a+b+c;   Writeln('p= ',p); End
Else  Writeln('треугольник не равнобедренный');
End.
```

Задачи для самостоятельного решения:

Вариант № 1

1. Дано целое число. Если оно является положительным то прибавить к нему 1, в противном случае вычесть из него два. Вывести полученное число.

2. Даны три стороны одного треугольника и три стороны другого треугольника. Определить, будут ли эти треугольники равновеликими, т. е. имеют ли они равные площади.

3. Ввести 2 числа. Если их произведение отрицательно, умножить его на –2 и вывести на экран, в противном случае увеличить его в 1,5 раза и вывести на экран. (Написать программу, начертить блок-схему).

Вариант № 2

1. Ввести число. Если оно четное, разделить его на 4, если нечетное - умножить на 5.

2. Ввести рост человека. Вывести на экран "ВЫСОКИЙ", если его рост превышает 180 см, и "НЕ ОЧЕНЬ ВЫСОКИЙ" в противном случае.

3. Составить программу, которая спрашивает возраст человека и, если ему 18 лет и больше, сообщает "Замечательно. Вы уже можете водить автомобиль", а в противном случае – "К сожалению, водить автомобиль Вам рановато".

Вариант № 3

1. Составить программу вычисления значений функции для любого x по желанию пользователя.

2. Вовочка, любитель стрелять из рогатки, 7 раз попадал в милицию. Ввести с клавиатуры целое положительное число – № попадания. Определить

результат: 4,6,7 – милиционеры вставляли новое стекло, 2,5 – новое стекло вставлял папа Вовочки, 1, 3 – стекло не разбилось.

3. Составить программу, которая запрашивает ввод трех значений температуры и проверяет, есть ли среди них температура таяния льда?

Вариант № 4

1. Вводятся три значения ускорения свободного падения. Программа должна проверить, есть ли среди них ускорение свободного падения.

2. Составить программу, которая запрашивает ввод формул трех кислот и проверяет, есть ли среди них формула серной кислоты?

3. Задача № 13: Составить программу, которая запрашивает ввод температуры тела человека и определяет, здоров он или болен (здоров при $36 < t < 37$)?

Вариант № 5

1. Составить программу вычисления значений функции для любого x по желанию пользователя.

2. Даны три стороны одного треугольника и три стороны другого треугольника. Определить, будут ли эти треугольники равновеликими, т. е. имеют ли они равные площади.

3. Составить программу, которая спрашивает возраст человека и, если ему 18 лет и больше, сообщает “Замечательно. Вы уже можете водить автомобиль”, а в противном случае – “К сожалению, водить автомобиль Вам рановато”.

Контрольные вопросы

1 Что понимают под алгоритмом ветвления?

2 Привести примеры случаев ветвления.

3 Как обозначается ветвление в блок-схемах?

4 Какие операторы ветвления существуют в языке ABC Pascal?

5 Какой формат имеет оператор IF?

6 Какие различия между полной и сокращенной формой оператора IF?

Практическая работа № 6

Тема: Составление программ ветвящейся структуры в Pascal

Цель: научиться составлять программы для решения задач, содержащих оператор выбора CASE, продолжить освоение работы в системе программирования ABC Pascal.

Ход работы:

Теоретическое обоснование

Оператор case

Оператор выбора позволяет выбрать одно из нескольких возможных продолжений программы.

Структура оператора выбора такова:

case <ключ выбора> of

<список выбора>

[else <операторы>]

end;

Пример 1. Вводиться число от 1 до 4, определяющее пору года. Дать название этой поры года (1 - зима, 2 - весна, 3 - лето, 4 - осень).

Этапы выполнения задания.

I.Определение исходных данных: переменная n.

II. Определение результатов: вывод названия поры года.

III. Алгоритм решения задачи.

1. Ввод исходных данных

2. Проверяем чему равно n и соответственно выводим название поры года.

IV. Описание переменных:

Переменная n типа byte.

V. Программа:

```
var
n:byte;
Begin
writeln('Введите номер поры года');
readln(n);
case n of
1: writeln('зима');
2: writeln('весна');
3: writeln('лето');
4: writeln('осень');
else
writeln('неправильно ввели номер поры года');
end;
End.
```

Пример 1

VI. Тестирование программы:

1. Проверьте работу программы для следующих значений

Ввод n	Вывод
5	неправильно ввели номер поры года
2	весна
1	зима

2. Поэкспериментируйте с программой вводя различные значения.

Пример 2. Составим программу "КАЛЬКУЛЯТОР", которая после ввода двух чисел и одного из знаков +, -, *, / произведёт вычисления, а результат выдаст на экран.

Этапы выполнения задания.

I.Определение исходных данных: переменные a,b,sim.

II. Определение результатов: переменная s.

III. Алгоритм решения задачи.

1. Ввод исходных данных

2. Проверяем чему равно sim и соответственно выполняем нужное действие.

IV. Описание переменных:

Переменные a,b,s типа real, sim типа char .

V. Программа:

```
var
a,b,s:real;
sim:char;
Begin
writeln('Введите два числа');
readln(a,b);
writeln('Введите знак операции');
readln(sim);
case sim of
'+': s:=a+b;
'-': s:=a-b;
'*': s:=a*b;
'/': s:=a/b;
end;
writeln ('результат   ',a,sim,b,' = ',s);
End.
```

Пример 2

VI. Тестирование программы:

1. Проверьте работу программы для следующих значений

Ввод a,b,sim	Вывод
Введите два числа 3 4 Введите знак операции *	результат 3*4 = 12
Введите два числа 3 4 Введите знак операции +	результат 3+4 = 7
Введите два числа 12 6 Введите знак операции -	результат 12-6 = 6

2. Поэкспериментируйте с программой вводя различные значения.

Замечание. В данной программе отсутствует часть else и поэтому, если ввести вместо рассматриваемых арифметических знаков, ввести любой символ, то программа будет работать, но будет работать неверно. Устраните этот недочет.

Вопросы для повторения:

1. Сколько строк может быть записано в списке выбора?
2. Может ли в операторе выбора отсутствовать часть else?
3. Сформулируйте, что может являться ключом выбора?

4. Можно ли оператор выбора заменить условным оператором if … then?

5. Сколько операторов if then понадобилось бы для решения примера 2?

Задачи для самостоятельного решения

Вариант 1

1. Дано целое число в диапазоне 1–7. Вывести строку — название дня не дели, соответствующее данному числу (1 — «понедельник», 2 — «вторник» и т. д.).
2. Дано целое число K. Вывести строку-описание оценки, соответствующей числу KK (1 — «плохо», 2 — «неудовлетворительно», 3 — «удовлетворительно», 4 — «хорошо», 5 — «отлично»). Если KK не лежит в диапазоне 1–5, то вывести строку «ошибка».
3. Написать программу перевода числовой оценки в текстовую

Вариант 2

1. Дан номер месяца — целое число в диапазоне 1–12 (1 — январь, 2 февраль и т. д.). Вывести название соответствующего времени года («зима», «весна», «лето», «осень»).
2. Дан номер месяца — целое число в диапазоне 1–12 (1 — январь, 2 февраль и т. д.). Определить количество дней в этом месяце для невисокосного года.
3. Ввести число. Определить четное или нечетное это число

Вариант 3

1. Арифметические действия над числами пронумерованы следующим образом: 1 — сложение, 2 — вычитание, 3 — умножение, 4 — деление. Дан номер действия NN (целое число в диапазоне 1–4) и вещественные числа AAи BB (BB не равно 0). Выполнить над числами указанное действие и вывести результат.
2. Единицы длины пронумерованы следующим образом: 1 — дециметр, 2 — километр, 3 — метр, 4 — миллиметр, 5 — сантиметр. Дан номер

единицы длины (целое число в диапазоне 1–5) и длина отрезка в этих единицах (вещественное число). Найти длину отрезка в метрах.

3. Создать простейший калькулятор

Вариант 4

1. Единицы массы пронумерованы следующим образом: 1 — килограмм, 2 — миллиграмм, 3 — грамм, 4 — тонна, 5 — центнер. Дан номер единицы массы (целое число в диапазоне 1–5) и масса тела в этих единицах (вещественное число). Найти массу тела в килограммах.
2. Даны два целых числа: DD (день) и MM (месяц), определяющие правильную дату невысокосного года. Вывести значения DD и MM для даты, предшествующей указанной.
3. Написать программу перевода римских цифр в арабские

Вариант 5

1. Даны два целых числа: DD (день) и MM (месяц), определяющие правильную дату невисокосного года. Вывести значения DD и MM для даты, следующей за указанной.
2. Робот может перемещаться в четырех направлениях («С» — север, «З» — запад, «Ю» — юг, «В» — восток) и принимать три цифровые команды: 0 — продолжать движение, 1 — поворот налево, –1 — поворот направо. Дан символ CC — исходное направление робота и целое число NN - посланная ему команда. Вывести направление робота после выполнения полученной команды.
3. Написать программу «Гороскоп»

Контрольные вопросы

1. Что такое оператор case?
2. Что позволяет выполнять оператор выбора?
3. С какими типами данных может работать оператор выбора?

Практическая работа № 7, 8 (4 часа)

Тема: Программирование циклов в Pascal

Цель работы: Изучить структуру оператора цикла с параметром языка Pascal, научиться составлять циклические алгоритмы, создавать программы, используя полученные знания.

Ход работы:

Теоретический обоснование

В программах, связанных с обработкой данных или вычислениями, часто выполняются циклически повторяющиеся действия.

Цикл представляет собой последовательность операторов, которая выполняется неоднократно. В языке Turbo Pascal различают три вида операторов цикла: цикл с предусловием (while), цикл с постусловием (repeat) и цикл с параметром (for).

Следует знать:

подавляющее большинство задач с циклами можно решить разными способами, используя при этом любой из трех операторов цикла;

часто решения, использующие разные операторы цикла, оказываются равноценными;

в некоторых случаях все же предпочтительнее использовать какой-то один из операторов;

самым универсальным из всех операторов цикла считается while, поэтому в случае затруднений с выбором можно отдать предпочтение ему;

цикл repeat имеет очень простой и понятный синтаксис, поэтому с него удобно начинать изучение циклов;

цикл for обеспечивает удобную запись циклов с заранее известным числом повторений;

при неумелом использовании циклов любого типа возможна ситуация, когда компьютер не сможет нормально закончить цикл. При работе в среде Turbo Pascal для выхода из подобной ситуации используется комбинация клавиш <Ctrl>+<Break>.

если это не помогает, есть и крайнее средство – < Ctrl >+<Alt>+<Del>. Одновременное нажатие этих трех клавиш или кнопки Reset, расположенной на системном блоке, позволяет перезагрузить компьютер, при этом данные, относящиеся к работающей программе, будут утеряны.

процедура continue позволяет прервать выполнение тела любого цикла и передает управление на его заголовок, заставляя цикл немедленно перейти

к следующему выполнению.

Циклы с параметром

Если число требуемых повторений заранее известно, то используется оператор, называемый оператором цикла с параметром или циклом со счетчиком.

*цикл по <счетчик> от <нач.знач.> до <конечн.знач.>

** действие

* конец цикла

Оператор цикла с параметром имеет два варианта записи:

1) вариант с увеличением счетчика

for <счетчик> := <начальное значение> to <конечное значение> do

<тело цикла>

2) вариант с уменьшением счетчика

for <счетчик> := <нач. значение> downto <конечное значение> do

<тело цикла>

Счетчик – параметр цикла, простая переменная целого типа; <тело цикла> - операторы или оператор. Цикл повторяется до тех пор пока значение параметра лежит в интервале между начальным и конечным значениями. В первом варианте при каждом повторении цикла значение параметра увеличивается на 1, во втором - уменьшается на 1.

При первом обращении к оператору for вначале определяются начальное и конечное значения, и присваивается параметру цикла начальное значение. После этого циклически повторяются следующие действия.

1. Проверяется условие: параметр цикла <= конечному значению.

2. Если условие выполняется, то оператор продолжает работу (выполняется оператор в теле цикла), если условие не выполняется, то оператор завершает работу и управление в программе передается на оператор, следующий за циклом.

3. Значение параметра изменяется (увеличивается на 1 или уменьшается на 1).

Если в теле цикла располагается более одного оператора, то они заключаются в операторные скобки begin … end;

Следует знать:

оператор For используется для организации циклов с фиксированным, заранее известным числом повторений;

количество повторений цикла определяется начальным и конечным значениями переменной-счетчика. Оператор For обеспечивает выполнение тела цикла до тех пор, пока не будут перебраны все значения параметра цикла: от начального до конечного;

переменная счетчик должна быть порядкового типа: чаще integer, реже – char, boolean. Использование вещественного типа недопустимо;

начальное и конечное значения параметра цикла могут быть константами, переменными, выражениями и должны принадлежать к одному и тому же типу данных. Начальное и конечное значение параметра цикла нельзя изменять во время выполнения цикла;

параметр цикла For может изменяться (увеличиваться или уменьшаться) каждый раз при выполнении тела цикла только на единицу. Если нужен другой шаг, предпочтительнее использовать циклы с предусловием или с постусловием.

Задача 1. Вывести на экран натуральные числа от 1 до 9 в обратном порядке.

```
Program z1;
Var i:integer;
Begin
```

```
For i:=9 downto 1 do
   Writeln(i);
End.
```

Оператор цикла с предусловием

Если число повторений заранее неизвестно, а задано лишь условие его повторения (или окончания), то используются операторы while и repeat. Оператор While часто называют оператором цикла с предусловием. Так как проверка условия выполнения цикла производится в самом начале оператора.

*цикл пока <условие>

**<действие>

* конец цикла

Общий вид: While <условие продолжения повторений> do

<тело цикла>;

Тело цикла – простой или составной оператор или операторы. Если операторов в теле цикла несколько, то тело цикла заключается в операторные скобки begin…end. Ключевые слова While и do означают соответственно "пока" и "выполнять". Когда программа в процессе выполнения впервые достигает оператора while, осуществляется проверка истинности условия. Если условие истинно, то выполняется тело цикла (оператор). После этого происходит возврат к началу фрагмента while do, где проверка условия осуществляется вновь. Цикл будет выполняться до тех пор, пока логическое выражение будет истинным. Как только логическое выражение станет ложным, управление передается следующему за циклом оператору. Если при первом выполнении цикла значение логического оператора будет "ложь", то цикл не станет выполняться, а управление сразу же передается следующему за while оператору.

Следует знать:

число повторений операторов цикла while определяется в ходе работы программы и, как правило неизвестно;

после слова while записывается условие продолжения выполнения инструкций цикла;

условие – это выражение логического типа: простое выражение отношения или сложное выражение отношения, которое может принимать одно из двух значений: true или false;

для успешного завершения цикла while в его теле обязательно должны присутствовать инструкции, оказывающие влияние на условие выпонения инструкций цикла.

Задача 1. Найти сумму 10 произвольных чисел.

```
Program z1;
Const
N=10;
Var k, x, s: integer;
Begin
k:=0; s:=0;  {k- количество введенных чисел}
while k < n do
   begin
     k:=k+1;
     write(‘Введите число’);
      readln(x);
      s:=s+x;
    end;
writeln(‘Сумма чисел равна’, s);
end.
```

Циклы с постусловием

Оператор цикла repeat аналогичен оператору while, но отличается от него, во-первых, тем, что условие проверяется после очередного выполнения операторов тела цикла и таким образом гарантируется хотя бы однократное выполнение цикла. Во-вторых, тем, что критерием прекращения цикла является равенство выражения константе true . За это данный оператор часто называют

циклом с постусловием, так как он прекращает выполняться, как только условие, записанное после слова until, выполнится. Оператор цикла repeat состоит из заголовка, тела и условия окончания.

*цикл с постусловием

**<действие>

* конец цикла, если <условие>

Общий вид: Repeat

<оператор>

. . . . .

<оператор>

until <условие окончания цикла>

Вначале выполняется тело цикла, затем проверяется условие выхода из цикла. В любом случае этот цикл выполняется хотя бы один раз. Если условие не выполняется, т.е. результатом выражения является False, то цикл активизируется еще раз. Если условие выполнено, то происходит выход из цикла. Использования операторных скобок, в случае, если тело цикла состоит из нескольких операторов, не требуется.

Следует знать:

число повторений операторов цикла repeat определяется в ходе работы программы и, как правило неизвестно;

инструкции цикла repeat будут выполняться, пока условие, стоящее после until, будет оставаться ложным;

после слова until записывается условие завершения цикла;

условие – это выражение логического типа: простое выражение отношения или сложное выражение отношения, которое может принимать одно из двух значений: true или false;

для успешного завершения цикла repeat в его теле обязательно должны быть инструкции, выполнение которых влияет на условие завершения цикла, иначе цикл будет выполняться бесконечно – программа зациклится. Другими

словами, переменная, которая участвует в условии выхода из цикла, обязательно должна изменяться в теле цикла.

Задача 1. Составить программу, которая вводит и суммирует целые числа. Если введено значение 999, то на экран выводится результат суммирования.

```
Program z1;
Var x, s:integer;
Begin
S:=0;
Repeat
  Write('Ввести число');
  Readln(x);
  If x<>999 then s:=s+x;
Until x=999;
Writeln('Сумма введенных чисел', s);
End.
```

Вложенные циклы

В теле любого оператора цикла могут находиться другие операторы цикла. При этом цикл, содержащий в себе другой, называют внешним, а цикл, находящийся в теле первого – внутренним (вложенным). Правила организации внешнего и внутреннего циклов такие же, как и для простого цикла.

При программировании вложенных циклов необходимо соблюдать дополнительное условие: все операторы внутреннего цикла должны полностью располагаться в теле внешнего цикла.

Задача 1.

Даны натуральные числа n и k. Составить программу вычисления выражения lk+2k+...+nk.

{Для вычисления указанной суммы целесообразно организовать цикл с параметром i, в котором, во-первых, вычислялось бы очередное значение y=ik и, во-вторых, осуществлялось бы накопление суммы прибавлением

полученного слагаемого к сумме всех предшествующих (s= s+y).}

```
program z1;
  uses crt;
  var n, к, y, i, s, m: integer;
begin
  clrscr;
  writeln ('n= k='); readln(n, k);
  s:=0;
  for i:=l to n do
    begin
      y:=1;
        for m:=l to к do
          begin
{Нахождение степени k числа i.}
            y: = y*i;
          end;
{Нахождение промежуточной суммы.}
      s:=s+y;
    end;
  writeln(' Ответ: ',s);
  readln;
end.
```

Задания для самостоятельного решения

Вариант 1

1. Вычисление p = n! (n факториал)
2. Приближенное вычисление суммы бесконечно убывающего ряда 1+x/1!+x2/2!+x3/3!+ ...
3. Использование цикла repeat для подсчета суммы вводимых чисел до первого отрицательного числа

4. Вычислить сумму 11+22+...+nn.

Вариант 2

1. Составить программу вычисления значения выражения y=1+1/2+1/3+...+1/20.
2. Составление таблицы значений функции y = sin x отрезке [0;3.14] с шагом 0,1
3. Составить программу, которая вводит и суммирует целые числа. Если введено значение 999, то на экран выводится результат суммирования
4. Написать программу, которая находит и выводит на печать все четырехзначные abcd, числа a, b, c, d - различные цифры, для которых выполняется: ab-cd=a+b+c+d.

Вариант 3

1. Из чисел от 10 до 99 вывести те, сумма цифр которых равна S(0 < S < 18).
2. Вычислить наибольший общий делитель двух натуральных чисел A и B
3. Составить программу планирования закупки товара в магазине на сумму, не превышающую заданную величину
4. Даны натуральные числа n и k. Составить программу вычисления выражения lk+2k+...+nk.

Вариант 4

1. Вывести на экран натуральные числа от 1 до 9 в обратном порядке.
2. Найти сумму 10 произвольных чисел
3. Написать программу нахождения наибольшего общего делителя (НОД) двух натуральных чисел
4. Если мы сложим все цифры какого-либо числа, затем все цифры найденной суммы и будем повторять много раз, мы, наконец, получим однозначное число (цифру), называемое цифровым корнем данного числа. Например, цифровой корень числа 34697 равен 2

(3+4+6+9+7=29; 2+9=11; 1 + 1=2). Составим программу для нахождения цифрового корня натурального числа

Вариант 5

1. Дано натуральное число n (1000<=N<=9999). Определить, является ли оно палиндромом ("перевертышем"), с учетом четырех цифр. Например, палиндромами являются числа: 2222, 6116, 1441.
2. Дано натуральное число n. Посчитать количество цифр в числе
3. Составить программу, которая вводит и суммирует целые числа. Если введено значение 999, то на экран выводится результат суммирования
4. Вычислить сумму 11+22+...+nn.

Контрольные вопросы

1 Что такое циклический алгоритм?

2 Какие обозначения используются в графическом представлении алгоритма для обозначения цикла?

3 Какие бывают циклы?

4 Что такое и когда используется цикл с предусловием?

5 Какой оператор в языке Паскаль используется для представления цикла с предусловием?

6 Как в блок-схемах изображаются цикл с предусловием?

7 Особенности использования цикла с предусловием.

8 Что такое и когда используется цикл с постусловием?

9 Какой оператор в языке Паскаль используется для представления цикла с постусловием?

10 Как в блок- схемах изображаются цикл с постусловием?

11 Особенности использования цикла с постусловием.

12 Что такое и когда используется цикл с параметром?

13 Какой оператор в языке Паскаль используется для представления цикла с параметром? 14 Как в блок-схемах изображаются цикл с параметром?

Практическая работа № 9

Тема: Одномерные массивы

Цель: Изучить структуру одномерных массивов

Ход работы:

Теоретическое обоснование

В повседневной и научной практике часто приходится встречаться с информацией, представленной в табличной форме. Вот, например, таблица, содержащая значения температуры, за определенный год.

Месяц	1	2	3	4	5	6	7	8	9	10	11	12
Температура	-21	-18	-7,5	5,6	10	18	22,2	24	17	5,4	-7	-18

Такую таблицу называют линейной. Она представляет собой последовательность упорядоченных чисел. Если требуется какая-то математическая обработка данных, то для их обозначения вводят индексную символику.

Например:

T1 – температура января;

T5 – температура мая;

В общем виде множество значений, содержащихся в таблице, можно обозначить так: { Ti }, I = 1,2,3, …,12.

Порядковые номера элементов называются индексами. Индексированные величины удобно использовать для записи их математической обработки: $T_{cp} = \frac{1}{12}\sum_{i=1}^{12} T_i$. По данной формуле можно подсчитать среднегодовую температуру.

В языке Паскаль для записи линейных таблиц используют структурированный тип данных, который называется одномерным массивом. Одномерный массив представляет собой совокупность пронумерованных однотипных значений, имеющих общее имя. Элементы массива обозначаются

переменными с индексами. Индексы записывают в квадратных скобках после имени массива.

Например: T[1], T[5], T[I] и т.п.

Описание одномерных массивов:

Переменная типа одномерный массив описывается в разделе описания переменных в следующем виде:

var имя массива: array[1..n] of тип;

n – количество элементов массива.

Пример: var T: array [1..12] of real;

Обработка массивов в программах производится поэлементно. Обращаются к элементам массива, указывая индекс.

Ввод массива (чтение массива):

```
writeln('введите ',n,' элементов массива');
for i:=1 to 12 do
  readln (t[i]);
```

Вывод массива:

```
writeln('массив');
for i:=1 to 12 do
  writeln ('t[',i,']=', t[i]:0:2);
```

Примеры задач.

Известны данные о среднемесячной температуре за год. Требуется вычислить среднегодовую температуру, а также ежемесячные отклонения от этой величины.

```
Program Example;
Const n=12;
Var t, dt:array[1..n] of real;
      ts:real;
      i:integer;
Begin {ввод исходных данных}
   writeln ('введите температуру за ',n,' месяцев');
```

```
   for i:=1 to n do
     readln (t[i]);
{вычисление средней температуры}
   ts:=0;
   for i:=1 to n do
     ts:=ts+t[i];
   ts:=ts/n;
{вычисление таблицы отклонений от среднего}
   for i:=1 to n do
     dt[i]:=t[i]–ts;
{вывод результатов}
   writeln (‘среднегодовая температура =’ , ts:0:2);
   writeln (‘отклонения от среднегодовой температуры:’);
   for i:=1 to n do
     writeln (i,‘ – месяц: ’, dt[i]:0:2);
End.
```

Дан массив из 8 элементов целого типа. Найти сумму положительных элементов массива.

```
Program summa;
Const  n=8;
Var a:array[1..n] of integer;
     i,s:integer;
Begin {ввод исходных данных}
   writeln (‘введите ’,i,‘ целых чисел’);
   for i:=1 to n do
     readln (a[i]);
   s: =0;
   for i:=1 to n do
      if a[i]>0
      then s:=s+a[i];
```

writeln (‘сумма положительных чисел =’,s);

End.

3. Задать массив B из 12 элементов по следующей формуле: $b_i = \begin{cases} i^2 - 15, & i < 6 \\ i + 8, & i \geq 6 \end{cases}$. Вывести полученный массив. Найти максимальный элемент массива.

```
Program maximum;
Const  n=12;
Var b:array[1.. n] of integer;
      i,max:integer;
Begin {получение массива}
   for i:=1 to n do
       if  i<6
       then  b[i]:=sqr(i)-15
       else  b[i]:=i+8;
{вывод массива}
   writeln(‘массив’);
   for i:=1 to n do
       writeln (‘b[’,i,‘]=’, b[i]);
{поиск максимального элемента массива}
   max:=b[1];
   for i:=1 to n do
       if  b[i]>max
       then max:=b[i];
    writeln (‘максимум=’,max);
End.
```

4. Получить массив C из 10 элементов следующего вида: $c_i = \frac{i^3 - 10i}{3}$. Вывести полученный массив. Сосчитать количество отрицательных элементов массива.

```
Program kolvo;
Const  n=10;
Var c:array[1.. n] of real;
      i,k:integer;
Begin {получение массива}
   for i:=1 to n do
       c[i]:=(i*sqr(i)-10*i)/3;
{вывод массива}
   writeln('массив');
   for i:=1 to n do
      writeln ('c[',i,']=', c[i]:0:2);
{подсчет количества отрицательных}
   k:=0;
   for i:=1 to n do
      if  c[i]<0
      then k:=k+1;
    writeln ('количество отрицательных чисел=',k);
End.
```

5. Дан массив из 10 элементов целого типа. Заменить отрицательные элементы массива на их квадрат, положительные оставить без изменения. Вывести полученный массив.

```
Program zamena;
Const  n=10;
Var a:array[1.. n] of integer;
      i:integer;
Begin {ввод исходных данных}
   writeln ('введите ',i,' целых чисел');
   for i:=1 to n do
      readln (a[i]);
{замена отрицательных чисел}
```

```
    for i:=1 to n do
        if  a[i]<0
        then a[i]:=sqr(a[i]);
{вывод массива}
    writeln(‘массив’);
    for i:=1 to n do
        writeln (‘a[’,i,‘]=’, a[i]);
End.
```

Задания для самостоятельного решения

Вариант 1

1. Дан массив из 10 целых чисел. Найти количество отрицательных элементов массива.
2. Дан массив из 8 чисел. Заменить все положительные числа на 0. Отрицательные числа оставить без изменения. Вывести элементы массива.
3. Дано натуральное число n. Задать массив из n элементов по формуле $a_i = \frac{i^2 - 3}{2}$. Вывести элементы массива. Найти минимальный элемент массива.

Вариант 2

1. Дан массив из 8 чисел. Найти максимальный элемент массива.
2. Дан массив из 6 целых чисел. Умножить все отрицательные числа на 5. Вывести элементы массива.
3. Дано натуральное число n. Задать массив из n элементов по формуле $a_i = \cos i$. Вывести элементы массива. Найти сумму положительных элементов массива.

Вариант 3

1. Дан массив из 10 целых чисел. Найти минимальный элемент массива.
2. Дан массив из 7 чисел. Заменить все отрицательные числа на их модуль. Вывести элементы массива.

3. Дано натуральное число n. Задать массив из n элементов по формуле $a_i = e^{i+2} - 4$. Вывести элементы массива. Найти количество отрицательных элементов массива.

Вариант 4

1. Дан массив из 8 чисел. Найти сумму отрицательных элементов массива.
2. Дан массив из 9 целых чисел. Умножить все отрицательные числа на -3. Вывести элементы массива.
3. Дано натуральное число n. Задать массив из n элементов по формуле $a_i = \frac{|i-7|}{4}$. Вывести элементы массива. Найти минимальный элемент массива.

Вариант 5

1. Дан массив из 14 целых чисел. Найти количество отрицательных элементов массива.
2. Дан массив из 8 чисел. Заменить все отрицательные числа на 1. Положительные числа оставить без изменения. Вывести элементы массива.
3. Дано натуральное число n. Задать массив из n элементов по формуле $a_i = \frac{i^2 - 6}{2}$. Вывести элементы массива. Найти максимальный элемент массива.

Контрольные вопросы

Что такое массив?

Что такое индекс?

Как объявляются массивы?

Как можно заполнить массив?

Каким образом можно вывести массив на экран?

Как найти сумму, произведение и количество элементов одномерного массива?

Как найти минимальный элемент в массиве?

Как найти максимальный элемент в массиве?

Как вывести двумерный массив в виде таблицы?

Практическая работа № 10

Тема: Двумерные массивы

Цель: Изучить структуру двумерных массивов

Ход работы:

Теоретическое обоснование

Двумерный массив в Паскале трактуется как одномерный массив, тип элементов которого также является массивом (массив массивов). Положение элементов в двумерных массивах Паскаля описывается двумя индексами. Их можно представить в виде прямоугольной таблицы или матрицы.

Рассмотрим двумерный массив Паскаля размерностью 3*3, то есть в ней будет три строки, а в каждой строке по три элемента:

Каждый элемент имеет свой номер, как у одномерных массивов, но сейчас номер уже состоит из двух чисел – номера строки, в которой находится элемент, и номера столбца. Таким образом, номер элемента определяется пересечением строки и столбца. Например, a 21 – это элемент, стоящий во второй строке и в первом столбце.

Описание двумерного массива Паскаля.

Существует несколько способов объявления двумерного массива Паскаля.

Мы уже умеем описывать одномерные массивы, элементы которых могут иметь любой тип, а, следовательно, и сами элементы могут быть массивами. Рассмотрим следующее описание типов и переменных:

Пример описания двумерного массива Паскаля

```
Type
Vector = array [1..5] of <тип_элементов>;
Matrix= array [1..10] of vector;
Var m: matrix;
```

Мы объявили двумерный массив Паскаля m, состоящий из 10 строк, в каждой из которых 5 столбцов. При этом к каждой i -й строке можно обращаться m [i], а каждому j -му элементу внутри i -й строки – m [i , j].

Определение типов для двумерных массивов Паскаля можно задавать и в одной строке:

```
Type
Matrix= array [1..5] of array [1..10] of < тип элементов >;
```

или еще проще:

```
type
matrix = array [1..5, 1..10] of <тип элементов>;
```

Обращение к элементам двумерного массива имеет вид: M [i , j]. Это означает, что мы хотим получить элемент, расположенный в i -й строке и j -м столбце. Тут главное не перепутать строки со столбцами, а то мы можем снова получить обращение к несуществующему элементу. Например, обращение к элементу M [10, 5] имеет правильную форму записи, но может вызвать ошибку в работе программы.

Основные действия с двумерными массивами Паскаля

Все, что было сказано об основных действиях с одномерными массивами, справедливо и для матриц. Единственное действие, которое можно осуществить над однотипными матрицами целиком – это присваивание. Т.е., если в программе у нас описаны две матрицы одного типа, например,

```
type
matrix= array [1..5, 1..10] of integer;
var
  a , b : matrix ;
```

то в ходе выполнения программы можно присвоить матрице a значение матрицы b (a := b). Все остальные действия выполняются поэлементно, при этом над элементами можно выполнять все допустимые операции, которые определены для типа данных элементов массива. Это означает, что если массив состоит из целых чисел, то над его элементами можно выполнять операции, определенные для целых чисел, если же массив состоит из символов, то к ним применимы операции, определенные для работы с символами.

Ввод двумерного массива Паскаля.

Для последовательного ввода элементов одномерного массива мы использовали цикл for, в котором изменяли значение индекса с 1-го до последнего. Но положение элемента в двумерном массиве Паскаля определяется двумя индексами: номером строки и номером столбца. Это значит, что нам нужно будет последовательно изменять номер строки с 1-й до последней и в каждой строке перебирать элементы столбцов с 1-го до последнего. Значит, нам потребуется два цикла for , причем один из них будет вложен в другой.

Рассмотрим пример ввода двумерного массива Паскаля с клавиатуры:

Пример программы ввода двумерного массива Паскаля с клавиатуры

```
type
  matrix= array [1..5, 1..10] of integer;
var
  a, : matrix;
  i, j: integer; { индексы массива }
begin
  for i :=1 to 5 do {цикл для перебора всех строк}
    for j :=1 to 10 do {перебор всех элементов строки по столбцам}
      readln ( a [ i , j ]); {ввод с клавиатуры элемента, стоящего в i -й строке и j -м столбце}
```

Двумерный массив Паскаля можно заполнить случайным образом, т.е. использовать функцию random (N), а также присвоить каждому элементу матрицы значение некоторого выражения. Способ заполнения двумерного массива Паскаля выбирается в зависимости от поставленной задачи, но в любом случае должен быть определен каждый элемент в каждой строке и каждом столбце.

Вывод двумерного массива Паскаля на экран.

Вывод элементов двумерного массива Паскаля также осуществляется последовательно, необходимо напечатать элементы каждой строки и каждого столбца. При этом хотелось бы, чтобы элементы, стоящие в одной строке, печатались рядом, т.е. в строку, а элементы столбца располагались один под другим. Для этого необходимо выполнить следующую последовательность действий (рассмотрим фрагмент программы для массива, описанного в предыдущем примере):

Пример программы вывода двумерного массива Паскаля

```
for i :=1 to 5 do {цикл для перебора всех строк}
begin
  for j :=1 to 10 do {перебор всех элементов строки по столбцам}
    write ( a [ i , j ]:4); {печать элементов, стоящих в i -й строке матрицы в одной экранной строке, при этом для вывода каждого элемента отводится 4 позиции}
  writeln ; {прежде, чем сменить номер строки в матрице, нужно перевести курсор на начало новой экранной строки}
end ;
```

Замечание (это важно!): очень часто в программах студентов встречается ошибка, когда ввод с клавиатуры или вывод на экран массива пытаются осуществить следующим образом: readln (a), writeln (a), где a – это переменная типа массив. При этом их удивляет сообщение компилятора, что переменную этого типа невозможно считать или напечатать. Может быть, вы поймете, почему этого сделать нельзя, если представите N кружек, стоящих в ряд, а у вас

в руках, например, чайник с водой. Можете вы по команде «налей воду» наполнить сразу все кружки? Как бы вы ни старались, но в каждую кружку придется наливать отдельно. Заполнение и вывод на экран элементов массива также должно осуществляться последовательно и поэлементно, т.к. в памяти ЭВМ элементы массива располагаются в последовательных ячейках.

Представление двумерного массива Паскаля в памяти

Элементы абстрактного массива в памяти машины физически располагаются последовательно, согласно описанию. При этом каждый элемент занимает в памяти количество байт, соответствующее его размеру. Например, если массив состоит из элементов типа integer , то каждый элемент будет занимать по два байта. А весь массив займет S^2 байта, где S – количество элементов в массиве.

А сколько места займет массив, состоящий из массивов, т.е. матрица? Очевидно: S i^S j , где S i - количество строк, а S j – количество элементов в каждой строке. Например, для массива типа

Matrix = array [1..3, 1..2] of integer ;

потребуется 12 байт памяти.

Как будут располагаться в памяти элементы этого массива? Рассмотрим схему размещения массива M типа matrix в памяти.

Под каждый элемент M [i,j] типа integer выделяется две ячейки памяти. Размещение в памяти осуществляется «снизу вверх». Элементы размещаются в порядке изменения индекса, что соответствует схеме вложенных циклов: сначала размещается первая строка, затем вторая, третья... Внутри строки по порядку идут элементы: первый, второй и т.д.

Как мы знаем, доступ к любой переменной возможен, только если известен адрес ячейки памяти, в которой хранится переменная. Конкретная память выделяется для переменной при загрузке программы, то есть устанавливается взаимное соответствие между переменной и адресом ячейки. Но если мы объявили переменную как массив, то программа «знает» адрес начала массива, то есть первого его элемента. Как же происходит доступ ко

всем другим элементам массива? При реальном доступе к ячейке памяти, в которой хранится элемент двумерного массива, система вычисляет ее адрес по формуле:

Addr + SizeElem * Cols *(I -1)+ SizeElem *(J -1),

где Addr – фактический начальный адрес, по которому массив располагается в памяти; I , J – индексы элемента в двумерном массиве; SizeElem – размер элемента массива (например, два байта для элементов типа integer); Cols – количество элементов в строке.

Выражение SizeElem * Cols *(I -1)+ SizeElem *(J -1) называют смещением относительно начала массива.

Сколько памяти выделяется для массива?

Рассмотрим не столько вопрос о том, сколько памяти выделяется под массив (это мы разобрали в предыдущем разделе), а о том, каков максимально допустимый размер массива, учитывая ограниченный объем памяти.

Для работы программы память выделяется сегментами по 64 Кбайт каждый, причем как минимум один из них определяется как сегмент данных. Вот в этом-то сегменте и располагаются те данные, которые будет обрабатывать программа. Ни одна переменная программы не может располагаться более чем в одном сегменте. Поэтому, даже если в сегменте находится только одна переменная, описанная как массив, то она не сможет получить более чем 65536 байт. Но почти наверняка, кроме массива в сегменте данных будут описаны еще некоторые переменные, поэтому реальный объем памяти, который может быть выделен под массив, находится по формуле: 65536- S , где S – объем памяти, уже выделенный под другие переменные.

Зачем нам это знать? Для того чтобы не удивляться, если при компиляции транслятор выдаст сообщение об ошибке объявления слишком длинного массива, когда в программе встретит описание (правильное с точки зрения синтаксиса):

Type myArray= array [1..50000] of integer;

Вы уже знаете, что, учитывая двухбайтовое представление целых чисел, реально можно объявить массив с количеством элементов равным 65536/2 – 1=32767. И то лишь в том случае, если других переменных не будет. Двумерные массивы должны иметь еще меньшие границы индексов.

Задания для самостоятельного решения

Вариант № 1

1. Сформировать с помощью датчика случайных чисел и вывести на экран матрицу, размером MxN. Элементы задаются на интервале [-20, 25].

2. В двумерном массиве, состоящем из n целых чисел, найти сумму элементов в каждой строке. Размер произвольный.

3. Найти наименьший элемент двумерного массива. Размер MXN. Элементы задаются на интервале [-30, 45].

Вариант № 2

1. В двумерном массиве, состоящем из целых чисел, найти наименьший элемент и номер строки, в которой он находится. Элементы вводятся с клавиатуры. Размер MXN.

2. Найти сумму элементов в каждой строке двумерного массива, состоящего из целых чисел. Размер MXN. Элементы задаются на интервале [-19, 30].

3. Подсчитать количество положительных элементов в каждой строке матрицы размером MxN, элементы которой вводятся с клавиатуры.

Вариант № 3

1. Сформировать матрицу типа

1 0 0 0
0 1 0 0
0 0 1 0
0 0 0 1

2. Найти номер столбца массива размером MxN, в котором находится наибольшее количество отрицательных элементов. Элементы вводятся с клавиатуры.

3. Упорядочить каждый столбец матрицы по возрастанию. Массив размером MxN, элементы которого задаются датчиком случайных чисел на интервале [-17;26].

Вариант № 4

1. Сформировать матрицу

1 1 1 1

2 2 2 2

3 3 3 3

2. Найти наибольшее нечетное число в матрице размером MxN, элементы которой задаются датчиком случайных чисел на интервале [-27, 38].

3. Найти наименьший элемент двумерного массива. Размер MXN. Элементы задаются на интервале [-60, 15].

Вариант № 5

1. Сформировать с помощью датчика случайных чисел и вывести на экран матрицу, размером MxN. Элементы задаются на интервале [-10, 45].

2. Найти номер столбца массива размером MxN, в котором находится наибольшее количество положительных элементов. Элементы вводятся с клавиатуры.

3. Подсчитать количество отрицательных элементов в каждой строке матрицы размером MxN, элементы которой вводятся с клавиатуры.

Контрольные вопросы

Что такое двумерный массив?

Что такое индекс?

Как объявляются двумерные массивы?

Как можно заполнить двумерный массив?

Каким образом можно вывести двумерный массив на экран?

Как найти сумму, произведение и количество элементов двумерного массива?

Как найти минимальный элемент в двумерном массиве?

Как найти максимальный элемент в двумерном массиве?

Как вывести двумерный массив в виде таблицы?

Практическая работа № 11

Тема: Строки в Pascal

Цель: Изучить строковый тип в Паскале

Ход работы:

Теоретическое обоснование

Строковый тип данных в Паскале

Строки в Паскале – это данные типа string. Они используются для хранения последовательностей символов. В Паскале длина стандартной строки ограничена 255 символами. Под каждый символ отводится по одному байту, в котором хранится код символа. Кроме того, каждая строка содержит еще дополнительный байт, в котором хранится длина строки.

Если заранее известно, что длина строки будет меньше 255 символов, то программист может сам задать максимальную длину строки.

Примеры описания строк:

```
type
    str_type = string[12];
const
    n = 50;
var
    s1: string;
    s2, s3: str_type;
    s4: string[n];
    s5, s6, s7: string[7];
…
```

Длина строки хранится в первом ее байте, индекс которого равен 0. Объявление типизированной константы для типа string осуществляется так:

```
const
        s: string = 'FreePascal'
…
```

Существует понятие пустой строки, т.е. строки, которая не имеет элементов. Пустая строка обозначается двумя рядом стоящими апострофами (например, st := '').

Операции над строками

Строки можно присваивать друг другу. Если максимальная длина переменной слева меньше длины присваиваемой строки, то лишние символы справа отбрасываются.

```
…
s1 := 'this is text';
s2 := s1;
…
```

Строки можно объединять с помощью операции конкатенации, которая обозначается знаком +.

```
…
s1 := 'John';
s2 := 'Black';
s1 := s1 + ' ' + s2;
…
```

Строки можно сравнивать друг с другом с помощью операций отношения. При сравнении строки рассматриваются посимвольно слева направо, при этом сравниваются коды соответствующих пар символов. Строки равны, если они имеют одинаковую длину и посимвольно эквивалентны. В строках разной длины существующий символ всегда больше соответствующего ему отсутствующего символа. Меньшей будет та строка, у которой меньше код первого несовпадающего символа (вне зависимости от максимальных и текущих длин сравниваемых строк).

'abc' > 'ab' (true)

'abc' = 'abc' (true)

'abc' < 'abc ' (false)

Имя строки может использоваться в процедурах ввода-вывода. При вводе в строку считывается из входного потока количество символов, равное длине строки или меньшее, если символ перевода строки (который вводится нажатием клавиши Enter) встретится раньше. При выводе под строку отводится количество позиций, равное ее фактической длине.

```
…
readln (s1);
write (s1);
…
```

К отдельному символу строки можно обращаться как к элементу массива символов, например s1[3]. Символ строки совместим с типом char, их можно использовать в выражениях одновременно, например:

```
…
s1[3] := 'h';
writeln (s2[3] + 'r');
…
```

Можно осуществлять коррекцию любого символа строковой переменной, для чего в соответствующем операторе достаточно указать имя переменной типа string, вслед за которым в квадратных скобках задается номер ее элемента (например, str[3]:='j').

Элементы строки нумеруются с единицы, т.к. в каждой строковой переменной имеется элемент с номером 0, в котором в виде символа хранится длина текущей строки. Чтобы узнать текущую длину, достаточно применить функцию ord к нулевому элементу строки. Например:

```
…
writeln(ord(st[0]))
…
```

Нулевой элемент строковой переменной можно корректировать. При этом будет изменяться текущая длина строки. Например, выражение str[0]:=#50 устанавливает текущую длину равной 50.

Процедуры и функции для работы со строками

При работе со строками, как правило, возникает необходимость выполнять их копирование, вставку, удаление или поиск. Для эффективной реализации этих действий в Паскале предусмотрены стандартные процедуры и функции. Они кратко описаны ниже.

Функция Concat (s1, s2, ..., sn) возвращает строку, являющуюся слиянием строк s1, s2, ..., sn.

Функция Copy (s, start, len) возвращает подстроку длиной len, начинающуюся с позиции start строки s.

Процедура Delete (s, start, len) удаляет из строки s, начиная с позиции start, подстроку длиной len.

Процедура Insert (subs, s, start) вставляет в строку s подстроку subs, начиная с позиции start.

Функция Length (s) возвращает фактическую длину строки s, результат имеет тип byte.

Функция Pos (subs, s) ищет вхождение подстроки subs в строку s и возвращает номер первого символа subs в s или нуль, если subs не содержится в s.

Процедуры преобразования типов

Процедура Str (x, s) преобразует числовое значение x в строку s, при этом для x может быть задан формат, как в процедурах вывода write и writeln. Например:

```
x := 123;
s := str(x:6,s);
```

Результат: s = ' 123'.

Процедура Val (s, x, errcode) преобразует строку s в значение числовой переменной x, при этом строка s должна содержать символьное представление

числа. В случае успешного преобразования переменная errcode равна нулю. Если же обнаружена ошибка, то errcode будет содержать номер позиции первого ошибочного символа, а значение x не определено.

Задача: (Символьные величины)

Составить программу, которая определяет количество букв «a» в заданном тексте

```
program fff;
var a,b:string;
i,n,k:integer;
begin
readln(a);
n:=length(a);
for i:=1 to n do
begin
if a[i]='a' then k:=k+1;
end;
writeln('в слове ',a,' буква А встречается ',k,' раз');
readln;
end.
```

Задача: (Символьные величины)

Составить программу, которая определяет количество слов в заданном тексте при условии, что слова разделены пробелом.

```
program fff;
var a,b:string;
i,n,k:integer;
begin
readln(a);
n:=length(a);
for i:=1 to n do
```

```
begin
if a[i]=' ' then k:=k+1;
end;
writeln('в тексте: ',a,', количество слов= ',k);
readln;
end.
```

Задача: (Массив с символьными переменными)

Составить массив из пяти фамилий, и вывести на экран столбиком, начиная с последней.

```
program fff;
var v:array[1..5] of string;
i:integer;
begin
writeln('введи пять фамилий');
for i:=1 to 5 do readln(v[i]);
writeln(' фамилии наоборот:');
for i:=5 downto 1 do writeln(v[i]);
readln;
end.
```

Задача: (Массив с символьными переменными)

Составить массив из пяти фамилий, и вывести на экран те из них, которые начинаются с определённой буквы, которая вводиться с клавиатуры.

Длину строки можно указать в разделе описания переменных:

<имя переменной, …>:string[n]

```
program fff;
var v:array[1..5] of string;
k:string[1];
i:integer;
begin
writeln('введи букву, с которой будет начинаться фамилия');
```

```
readln(k);
writeln('введи пять фамилий');
for i:=1 to 5 do readln(v[i]);
writeln('интересующие Вас фамилии:');
for i:=1 to 5 do
if (v[i])[1]=k then writeln(v[i]);
readln;
end.
```

Задача: (Массив с символьными переменными)

Из вводимого с клавиатуры слова вырезать каждую третью букву.

```
program aa;
var a:string;
k,x: integer;
begin
readln(a);
k:=length(a);
x:=3;
while x<=k do
begin
a[x]:=' ';
x:=x+3;
end;
writeln(a);
readln;
end.
```

Задача: (Массив с символьными переменными)

В вводимом слове с клавиатуры заменить все буквы «а» на букву «о».

```
program aa;
var a:string;
k,x: integer;
```

```
begin
readln(a);
k:=length(a);
for x:=1 to k do
if a[x]='a' then a[x]:='o';
writeln(a);
readln;
end.
```

Задача: (Массив с символьными переменными)

Заданны фамилия, имя, отчество учащегося, разделённые пробелами. Напишите программу, печатающую фамилии ученика и его инициалы.

```
program aaa;
uses crt;
var d,r:string;
k,i:integer;
begin
clrscr;
writeln('введи Ф.И.О.');
readln(d);
k:=length(d);
for i:=1 to k do
begin
if d[i]=' ' then d:=copy(d,1,i)+d[i+1]+'.';
end;
writeln(d);
readln;
end.
```

Все структурированные типы данных, с которыми мы уже познакомились, представляют собой совокупности однотипных величин.

Комбинированный тип данных – это структурированный тип, состоящий из фиксированного числа компонент (полей) разного типа. Комбинированный тип имеет еще и другое название – запись.

Обычно запись содержит совокупность разнотипных атрибутов, относящихся к одному объекту. Например, анкетные сведения о студенте могут быть представлены в виде информационной структуры.

Анкета студента:

Фамилия, Имя, Отчество;

Пол;

Дата рождения;

Адрес;

Курс;

Группа;

В Паскале эта информация может храниться в одной переменной типа Запись. Задать тип и описать соответствующую переменную можно следующим образом:

```
Type имя типа=record;
        поле 1:тип;
        поле 2:тип;
              …
        поле n:тип;
End;
```

```
Пр.  Type anketa=Record;
         FIO:String[50];
         Pol:Char;
         Dat:String[16];
         Adres:String[50];
         Curs:integer;
         Grup:string[5];
End;
```

Var student:anketa;

К каждому элементу записи можно обратится используя составное имя, которое имеет следующую структуру: имя переменной.имя поля

Например: student.fio, student.dat и т.п. Если, например полю курс присвоить значение 3, то это можно сделать следующим образом:

Student.Curs:=3;

Поля записи могут иметь любой тип, в частности сами могут быть записями. Такая возможность используется в том, случае, когда требуется представить многоуровневое дерево (более 2 уровней).

Любая обработка записей, в том числе ввод и вывод, производится поэлементно.

Например, ввод сведений о 500 студентах можно организовать следующим образом:

```
For i:=1 to 500 do
With Student [i]  do
Begin
    Write (‘Ф.И.О.’); Readln (FIO);
    Write (‘Пол (м/ж)’); Readln (Pol);
    Write (‘Дата рождения’); Readln (Dat);
    Write (‘Адрес‘); Readln (Adres);
    Write (‘Курс‘); Readln (Curs);
    Write (‘Группа’); Readln (Grup);
End;
```

В этом примере использован оператор присоединения, который имеет следующий вид:

```
With  имя переменной типа запись  do
  Begin
    Действия
  End;
```

Он позволяет, один раз указав имя переменной типа запись после слова With, работать в пределах оператора с именами полей как с обычными переменными, т.е. не писать громоздких составных имен.

Примеры задач.

В группе 30 студентов. О каждом студенте известно: фамилия и оценки по 8 предметам. Найти среднюю оценку каждого студента и выдать на экран среднюю оценку и фамилию лучшего студента.

```
Program zapis;
Const n=30;
Type sved=record;
        fio:string[50];
        p1, p2, p3, p4, p5, p6, p7, p8:integer;
        ocsr:real;
      End;
Var s:array[1..n] of sved;
    i,k: integer;
    max: real;
Begin
  writeln(‘введите данные о ’,n,‘ студентах’);
  For i:=1 to n do
    With spisok [i] do
      Begin
        Writeln (‘введите фамилию ученика‘);
        Readln (famil);
        Writeln (‘введите оценки по восьми предметам’);
        Readln (p1, p2, p3, p4, p5, p6, p7, p8);
        Clrscr;
      End;
{нахождение среднего балла }
```

```
  for i:=1 to n do
  with s[i] do
      ocsr:=(p1+p2+p3+p4+p5+p6+p7+p8)/8;
{нахождение максимального среднего балла}
  max:=s[1].ocsr;
  for i:=1 to n do
    if s[i].ocsr>max
    then
      begin
        max:=s[i].ocsr;
        k:=i
      end;
{печать сведений о лучшем ученике}
  writeln('Лучший ученик')
  writeln ('ФИО - ',s[k].fio,' средний балл - ' ,s[k].ocsr);
end.
```

Составить программу, которая позволит ввести данные о 10 автомобилях предприятия (марка, цвет, пробег, год выпуска, водитель) и выдаст сведения об автомобилях, старше 10 лет.

```
Program Avto;
Const n=10;
Type mash=record;
        marka:string[25];
        cvet:string[15];
        probeg:real;
        god_vip:integer;
        voditel:string[25];
      End;
 Var s:array [1.. n] of mash;
      i:integer;
```

```
Begin
  writeln (‘Введите данные о ’,n,‘ автомобилях’);
  for i:=1 to n do
    with s[i] do
     begin
       write(‘введите марку автомобиля‘); readln(marka);
       write(‘введите цвет автомобиля’); readln(cvet);
       write(‘введите пробег автомобиля’); readln(probeg);
       write(‘введите год выпуска’); readln(god_vip);
       write(‘введите фамилию водителя’); readln(voditel);
     end;
  writeln(‘автомобили старше 10 лет’);
  for i:=1 to n do
    begin
      with s[i] do
        begin
          if 2011-god_vip>10
          then
            begin
              writeln (‘марка машины - ’,marka);
              writeln (‘цвет машины - ’,cvet);
              writeln (‘пробег - ’,probeg);
              writeln (‘водитель - ’,voditel);
            end;
        end;
    end;
End.
```

Составить программу, которая позволяет ввести данные о 15 деталях, находящихся на складе (название, количество, стоимость одной детали). Вывести название и количество самой дорогой детали.

```
Program detali;
Const n=15;
Type detal=record;
          nazv:string[25];
          kol:integer;
          cena:real;
        End;
Var s:array [1.. n] of detal;
      i,k:integer;
      max:real;
Begin
  writeln (‘Введите данные о ’,n,‘ деталях’);
  for i:=1 to n do
    with s[i] do
      begin
        write(‘введите название детали’); readln(nazv);
        write(‘введите количество этой детали на складе’); readln(kol);
        write(‘введите цену 1-й детали’); readln(cena);
      end;
  max:=s[1].cena;
  k:=1;
  for i:=1 to n do
    begin
        with s[i] do
          begin
            if cena>max
            then
              begin
                max:=cena;
                k:=i
```

```
                end;
            end;
        end;
    writeln(‘самая дорогая деталь’);
    writeln (‘название - ’,s[k].nazv);
    writeln (‘количество - ’,s[k].kol);
End.
```

Составить программу, которая позволяет ввести данные о 20 кубиках (цвет, материал, длина ребра). Сосчитать количество кубиков с длиной ребра меньше 5 см.

```
Program kubiki;
Const n=20;
Type kub=record;
        cvet:string[25];
        material:string[25];
        dlina_rebra:real;
      End;
Var s:array [1.. n] of kub;
    i,k:integer;
Begin
  writeln (‘Введите данные о ’,n,‘ кубиках’);
  for i:=1 to n do
    with s[i] do
      begin
        write(‘введите цвет кубика’); readln(cvet);
        write(‘введите материал, из которого изготовлен’); readln(material);
        write(‘введите длину ребра’); readln(dlina_rebra);
      end;
  k:=0;
  for i:=1 to n do
```

```
        with s[i] do
            begin
              if dlina_rebra<5
              then k:=k+1
            end;
    writeln('количество кубиков с длиной ребра меньше 5 см. =',k);
End.
```

Даны две строки, выяснить совпадают ли их длины.

Дана строка, подсчитать количество вхождений слова «кот» в данную строку.

Найти номер позиции, с которой в заданной строке в первый раз встретится некоторое введенное слово.

Подсчитать количество запятых в заданном тексте с помощью массива символов.

Заменить в заданной строке все вхождения «ку» на «за» с помощью операции копирования, удаления и вставки.

Задается последовательность символов(строка, состоящая только из букв латинского алфавита). Определить, являются ли все символы строки уникальными. Уникальность символа означает, что он встречается в строке только один раз. Малые и большие буквы алфавита считаются разными

Задается последовательность символов(строка). Заменить все плюсы на минусы, а все минусы на плюсы.

Задается последовательность символов(строка). Определить количество пробелов

в левой части строки, (от начала строки до первого непробела). Пример: ;.....bnnb',

Дана последовательность символов (строка). Заменить на символ звездочки (*) все буквы английского алфавита, которые повторяются больше, чем один раз. Малые и большие буквы считаются эквивалентными.

Дана последовательность символов (строка). Удалить все знаки "+", если за ними непосредственно следует цифра. Вывести новый текст и количество удаленных символов.

Дан двумерный массив размером п*т, за полненный случайными целыми числами в диапазоне [-70,30] (после заполнения массива случайными числами массив вывести на экран). Определить, есть ли в данном массиве столбец, в котором имеются одинаковые элементы.

Дана последовательность символов (строка). Словом текста является последовательность цифр и букв алфавита (латинского). Будем называть особым такое слово, в котором количество цифр и букв одинаково. Найти колличество особых слов

Определить количество слов во введённом тексте, начинающихся с заданной буквы. Считать, что слова в тексте разделены пробелами.

Задания для самостоятельного решения

Вариант 1

1. Известны данные о 5 пассажирах: ФИО, количество вещей и общий вес вещей. Вывести ФИО и количество вещей пассажира с максимальным общим весом вещей.
2. Известны данные о 5 студентах: ФИО, рост и вес. Вывести ФИО и вес всех студентов, которые ниже 180 см.
3. Известны данные о 5 кубиках: цвет, материал и длина ребра. Сосчитать количество красных кубиков.

Вариант 2

1. Известны данные о 5 мячиках: цвет и радиус. Вывести цвет самого маленького мячика.

2. Известны данные о 5 спортсменах: ФИО, возраст и вид спорта. Вывести ФИО и возраст тех спортсменов, которые занимаются плаванием.
3. Известны данные о 5 рабочих: ФИО, стаж, оклад. Сосчитать количество рабочих с окладом больше 15 000 рублей.

Вариант 3

1. Известны данные о 5 сортах конфет: название, цена и фабрика изготовитель. Вывести название и фабрику изготовитель самых дорогих конфет.
2. Известны данные о 5 студентах: ФИО, год рождения и специальность. Вывести ФИО и специальность тех студентов, которые родились после 1994 года.
3. Известны данные о 5 пассажирах: ФИО, количество вещей и общий вес вещей. Сосчитать количество пассажиров с количеством вещей больше 2.

Вариант 4

1. Известны данные о 5 кубиках: цвет, материал и длина ребра. Вывести цвет и материал самого маленького кубика.
2. Известны данные о 5 учениках музыкальной школы: ФИО, класс, инструмент. Вывести ФИО и класс тех из них, кто занимается игрой на скрипке.
3. Известны данные о 5 спортсменах: ФИО, возраст, вид спорта. Сосчитать количество спортсменов младше 20 лет.

Вариант 5

1. Даны две строки, выяснить совпадают ли их длины.
2. Дана строка, подсчитать количество вхождений слова «кот» в данную строку.
3. Найти номер позиции, с которой в заданной строке в первый раз встретится некоторое введенное слово.

Контрольные вопросы

1. Что такое строковый тип?
2. С какими типами данных может работать строковый тип?
3. Что такое процедура Val?

Практическая работа № 12

Тема: Процедуры и функции

Цель: научиться применять подпрограммы в решении задач на Паскале, а также научиться понимать какой вид подпрограммы необходим при решении определенной задачи и рассмотреть основные приемы использования подпрограмм

Ход работы:

Теоретическое обоснование:

Иногда в разных местах программы приходится выполнять практически одни и те же последовательности действий с разными исходными данными. Такие последовательности действий можно оформить в виде так называемых подпрограмм (от англ, subroutine) – сгруппировать операторы в блок, к которому можно обратиться по имени, причем неоднократно.

Подпрограммы сокращают текст программы, существенно уменьшают время их исполнения, облегчают жизнь программистам, которые могут создавать программы модульно, т. е. собирая сложную программу из законченных кусочков более простых составляющих. Это позволяет группе программистов создавать большие программы, а группе школьников разрабатывать и реализовывать какие-либо глобальные проекты

Подпрограммы делятся на процедуры и функции.

Встроенные (стандартные) процедуры и функции являются частью языка и могут вызываться по имени без предварительного описания. Например, abs, sqrt, ln, sin…- функции (возвращают результат), readln, write… – процедуры (не

возвращают результат). Их наличие существенно облегчает разработку прикладных программ. Однако в большинстве случаев некоторые специфичные для данной программы действия не находят прямых аналогов в библиотеках Turbo Pascal, и тогда программисту приходится разрабатывать свои нестандартные процедуры и функции.

Процедуры пользователя пишутся самим программистом в соответствии с синтаксисом языка вразделе описания подпрограмм.

Структура процедуры повторяет структуру программы, это "программа в миниатюре" — она также представлена заголовком и телом.

Заголовок состоит из зарезервированного слова procedure, идентификатора (имени) процедуры.

```
VAR … // раздел описания переменных главной программы
procedure ИмяПроцедуры;
var …
begin
…
end;
begin
//тело главной программы
end.
```

Вызов процедуры для последующего выполнения записывается в теле главной программы.

Пример 1. Программа вычисления площади и периметра.

Открыть в Паскале файл. Оформим повторяющуюся часть программы в виде процедуры (программа внутри главной программы).

```
{***Вычисление****}
procedure tx;
begin
writeln;
writeln('Вычисление: ');
```

```
write('Введите стороны a, b: ');
end;
{******************}
```

Вызов в нужном месте:

```
tx;
```

При вызове процедуры работа главной программы приостанавливается и начинает выполнятьсявызванная процедура. Когда процедура выполнит свою задачу, программа продолжится с оператора, следующего за оператором вызова процедуры.

Т.е. мы “научили” ПК новой команде tx. Ею можно пользоваться только в этой программе и, причем, много раз.

Достоинства подпрограмм:

Программы, написанные с участием подпрограмм, легче тестировать и отлаживать, у них более четкая логическая структура.

Самостоятельный характер подпрограмм позволяет поручать их составление различным программистам. Так осуществляется разделение работы по программированию и, тем самым, ускоряется ее завершение;

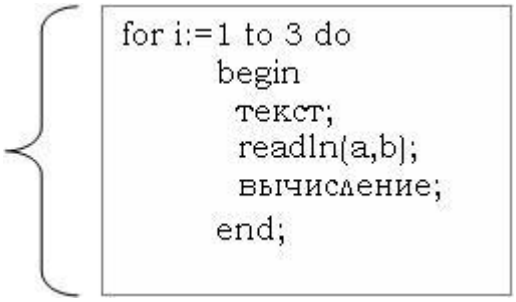
Использование подпрограмм позволяет экономить память. Память для хранения переменных, использующихся в подпрограмме, выделяется только на время ее работы и высвобождается, как только ее выполнение заканчивается.

Пример 2. Пользователь вводит две стороны трех прямоугольников. Вывести их площади.

Можно решить задачу так:

```
for i:=1 to 3 do
begin
writeln(‘Введите a и b:’);
readln(a,b);
writeln(‘Площадь=’,a*b);
end;
```

Хорошим стилем программирования считается использование процедур. Необходима процедура, которая будет вычислять площадь прямоугольника. Вот как схематично будет выглядеть главная программа:



текст

readln (a,b);

вычисление

текст

readln (a,b);

вычисление

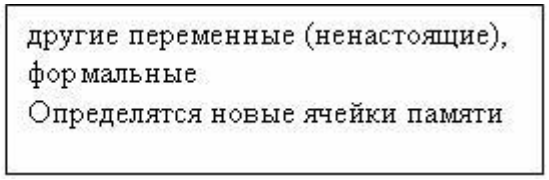
текст

readln (a,b);

вычисление

Процедура текста уже есть (см пример1). Создадим вторую процедуру, которая вычисляет площадь. Но для того чтобы вычислить S, надо знать 2 стороны, поэтому процедуре надо показать какие стороны она должна перемножать.

procedure pl (c,d: integer);



var S:integer;

begin

S:=c*d;

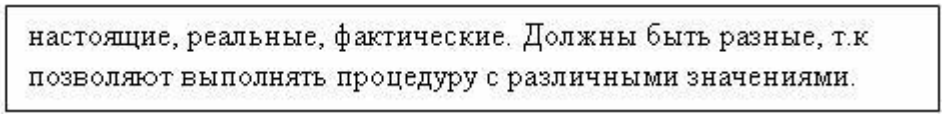
writeln(‘площадь прямоугольника со сторонами ’,c, ‘ ‘ ,d, ‘=‘,S);

end;

Параметр – это переменная, которой присваивается некоторое значение. Существуют формальные параметры, определенные в заголовке подпрограммы, и фактические параметры – выражения, задающие конкретные значения при обращении к подпрограмме.

Процедура выполнится, если вызвать ее по имени и указать фактические параметры, отделенные друг от друга запятыми и заключенных в круглые скобки:

p
l(4,5);



p
l(a,b);

Фактические параметры должны совпадать по типу и количеству с формальными.

Итак, главная программа:

```
for i:=1 to 3 do
begin
tx;
readln(a,b);
pl(a,b);
end;
```

Замечание. При решении этой задачи необходимо проверять введенные пользователем числа (они не должны быть отрицательными, иначе работа программы прервется).

Составим процедуру проверки:

```
procedure error (f,g:integer);
begin
if (f<0) or (g<0) then begin
writeln('стороны прямоугольника не могут быть отрицательными');
halt; //прерывание программы
end;
end;
```

Итак формат процедуры:

```
Procedure <имя> (формальные параметры);
Const ...;
```

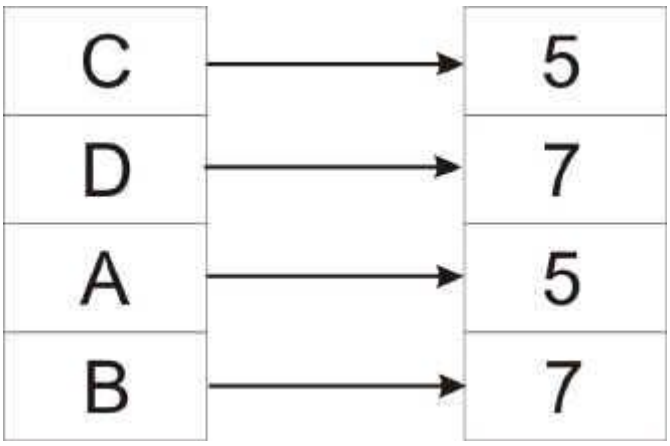
```
Type ...;
Var ...;
Begin
<операторы>;
end;
```

Пример 3. Составить программу обмена местами двух чисел c=5 и d=7.

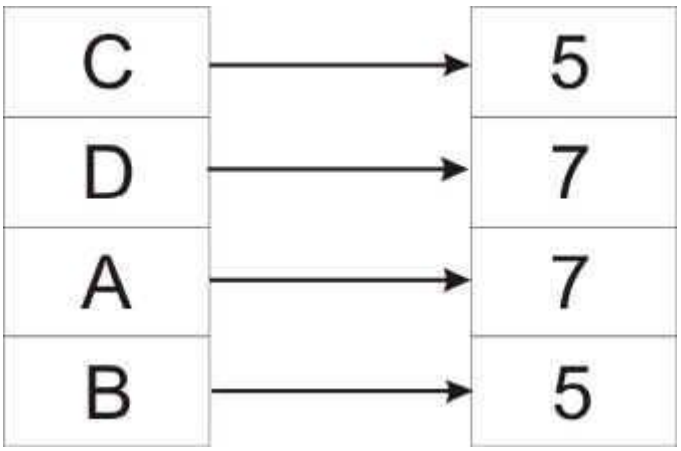
```
program obmenDan;
uses wincrt;
var c,d:integer;
{обмен}
procedure obmen ( a,b:integer);
var m:integer;
begin
m:=a; a:=b; b:=m;
writeln(a,b);
end;
begin
writeln ('Введите 2 числа: ');
readln(c,d);
obmen(c,d);
writeln(c,' ',d);
end.
```

После запуска программы видно, что поменялись местами формальные параметры (в процедуре), а фактические (которые используются в лавной программе) – не поменялись. Рассмотрим рисунок, на котором приведена часть оперативной памяти:

1) при вызове процедуры obmen с двумя параметрами 5 и 7, в переменные a и b помещаются тоже числа 5 и 7 соответственно:



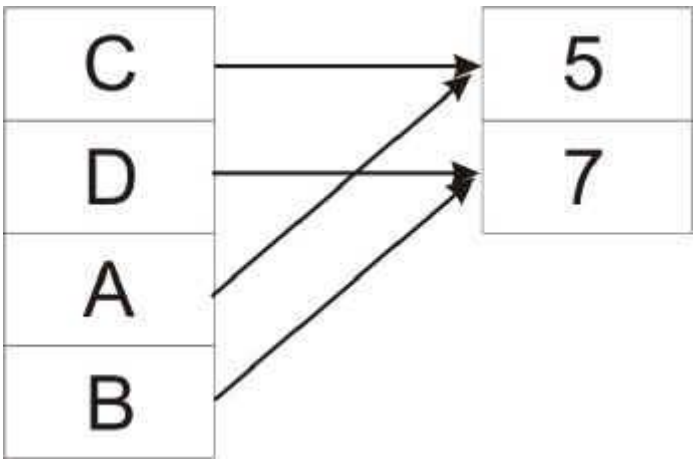
2) далее в процедуре осуществляется перестановка значений ячеек памяти a и b:



3) но в переменных c и d данные не поменялись, т.к. они находятся в других ячейках памяти.

Для того чтобы переменные c и d, a и b ссылались на одни и те же ячейки памяти (если изменятся значения a и b, то изменятся значения и c, d) необходимо при описании формальных параметров, перед нужными переменными добавить слово VAR:

```
procedure obmen (var a,b:integer);
```



Измените программу obmenDan:

obmen (1,4);

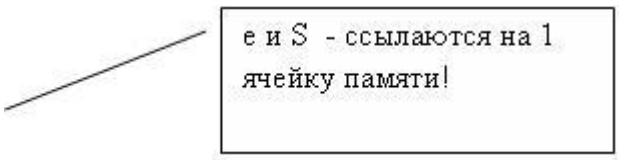
⇒ ошибка из-за var. Числа – константы, которые нельзя изменять в процедуре.

Пример 4. Найти площадь круга с использованием процедуры, которая производит только вычисление, но не отображает результат на экране.

```
procedure circle (r:real);
var S:real;
begin
S:=pi*r*r;
end;
```

Процедура должна возвращать результат:

```
procedure circle (r:real; var S:real);
begin
S:=pi*r*r;
end;
begin
readln(a, e);
writeln(e);
end.
```



Замечание: Переменная в процедуре S используется для возвращения результатов работы процедуры в основную программу. При ее изменении,

изменяется и фактический параметр в вызывающей программе, т.е. переменная e.

Чаще для этого в Паскале вместо процедур используют функции (подпрограммы, которые что-то возвращают).

Функция аналогична процедуре, но имеются два отличия.

Функция передает в программу результат своей работы – единственное значение, носителем которого является имя своей функции.

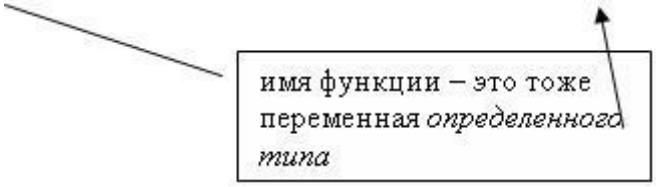
Имя функции может входить в выражение как операнд. Функция возвращает результат в точку своего вызова.

Например, sqr(x) – возведет в квадрат значение x и возвратит в точку вызова вычисленное значение квадрата числа x: y:=sqr(x);

Функция, определенная пользователем, состоит из заголовка и тела функции. Тело функции по структуре аналогично программе. Описание меток, констант, типов и т.д. действительны только в пределах данной процедуры.

Формат:

Function <имя> (формальные параметры) : <тип результата>;



Const ...;

Type ...;

Var ...;

Begin

<операторы>;

end;

В разделе операторов должен находиться хотя бы один оператор, присваивающий имени функции значение. В точку вызова возвращается результат последнего присваивания.

Пример 5. Переделаем задачу о площади круга.

function circle (r:real): real;

```
begin
circle:=pi*r*r;
end;
```

вызов:

```
a:=circle(5); (ОБЯЗАТЕЛЬНО присваиваем)
```

или

```
e:=circle(a);
writeln(e);
```

Пример 6. Найти 1!+2!+…+n!

Используем функцию нахождения факториала, т.к подаем на вход и получаем результат.

```
function fact (a:integer): integer;
var i: integer;
begin
fact:=1;
for i:=1 to a do
fact:=fact*I;
end;
```

В строке fact:=fact*I;

компилятор найдет ошибку, т.к fact должна вызываться с параметрами. Поэтому обычно вводят дополнительную переменную, в которую помещают результат. А потом в переменную fact присваивают этот результат:

```
program factorial;
uses wincrt;
var sum,n,j: integer;
function fact (a: integer): integer;
var i,d: integer;
begin
d:=1;
for i:=1 to a do
```

```
d:=d*i;
fact:=d;
end;
begin
sum:=0;
readln(n);
for j:=1 to n do
sum:=sum+fact(j);
writeln(sum);
end.
```

Практическая работа № 13

Тема: Работа с файлами

Цель: Изучить способы работы с файлами в Паскале

Ход работы:

Теоретическое обоснование

Файл - это упорядоченная последовательность однотипных компонентов, расположенных на внешнем носителе. Файлы предназначены только для хранения информации, а обработка этой информации осуществляется программами. Использование файлов целесообразно в случае:

долговременного хранения данных ;

доступа различных программ к одним и тем же данным;

обработки больших массивов данных, которые невозможно целиком разместить в оперативной памяти компьютера.

В Паскале определены текстовые файлы, типизированные и нетипизированные. Файл, не содержащий ни одного элемента, называется пустым. Создается файл путем добавления новых записей в конец первоначально пустого файла. Длина файла, т.е. количество элементов, не задается при определении файла.

Все файлы должны быть описаны в программе либо в разделе переменных VAR, либо в разделе типов TYPE. Под чтением файла понимают ввод данных из внешнего файла, находящегося на диске, в оперативную память машины. Запись в файл - вывод результатов работы программы из оперативной памяти на диск в файл.

Работа с файлами выполняется следующими процедурами:

Assign – устанавливает связь между именем файла в программе (файловой переменной) и физическим именем файла, принятым в ОС. Reset - открывает существующий файл для чтения. Rewrite – создает и открывает новый файл для записи на внеш нем устройстве (если файл ранее существовал, вся предыдущая информация из него стирается). Close - закрывает открытый файл.

Для определения конца файла используется стандартная встро енная функция EOF (файловая переменная), которая принимает значение True, если достигнут конец файла, и значение False в противном случае.

Текстовые файлы

Текстовые файлы – файлы на диске, состоящие из символов ASCII. Для разделения строк используются символы «конец строки». Текстовые файлы являются файлами с последовательным доступом. В любой момент времени доступна только одна запись файла. Другие записи становятся доступными лишь в результате последовательного продвижения по файлу. Текстовые файлы внутренне разделены на строки, длины которых различны. Для разделения строк используется специальный маркер конца строки. Объявляются текстовые файлы переменной типа text. Обрабатывать их можно только последовательно и с помощью процедур и функций:

Readln (f , st)- чтение строки st из файла f и переход на начало следующей ;
Writeln (f, st)- запись строки st в файл f и маркера конца строки ;
Append (f) - процедура, открывающая файл f для добавления строк в конец

файла;

Eoln (st)- логическая функция, результат выполнения которой равен TRUE, если достигнут маркер конца строки st.

Пример 1. Создать текстовый файл, в который записать 3 предложения. Прочитать этот файл, вывести его содержимое на экран. Определить длину каждого предложения.

```
Program File_text;
var
f1 : text;
st : string;
n: byte;
begin
assign   (f1, 'file1.txt'); {связать с файлом file1.txt  файловую переменную f1  }
rewrite  (f1); { создать новый файл  с именем file1.txt }
writeln ( f1, 'Очень полезно изучать'); { записать предложения в файл}
writeln ( f1, ' всем  студентам ');
writeln (f1, ' язык  Pascal ');
close (f1);    { закрыть файл для записи }
reset (f1); { открыть файл для чтения }
while not eof (f1) do { пока не конец файла  f1}
begin
readln (f1, st);    {читаем строку из файла f1 }
writeln(st);         { выводим на экран }
n:= length (st);     {определяем длину строки }
writeln ('  длина =',n);
end;
close (f1); { закрыть файл  для чтения}
end .
```

Типизированные файлы

Типизированные файлы – это файлы, состоящие из нумерованной последовательности объектов (записей) любого типа. С такими файлами можно работать в режиме прямого доступа, при котором выполняется непосредственное об ращение к любой записи файла. Каждая запись файла имеет свой номер, начиная с 0 и т.д.
Процедуры и функции обработки файлов:

1) Write и Read- записывают и читают информацию из указанного файла и перемещают указатель файла к сле дующей записи.
2) Seek (файловая переменная, номер записи); процедура перемещения указателя на запись файла с заданным номером.
3) Truncate (файловая переменная); процедура, усекающая файл по текущей позиции указа теля файла, т.е. все записи, находящиеся после указателя фай ла, удаляются.
4) Функция Filesize (файловая переменная); имеет тип Integer и определяет размер файла, т.е. число записей.
5) Функция Filepos (файловая переменная); имеет тип Integer и возвращает текущую позицию указателя файла.

Для добавления записей в конец файла используются процедуры:

```
Readln (a );
Seek (f, filesize (f));
Write (f, a);
```

При этом указатель устанавливается за конец файла, т.к. нуме рация записей начинается с нуля. После чего с помощью Write можно добавлять записи. Открывать файл можно только проце дурой Reset (f).
Для того, чтобы в режиме произвольного доступа считать, а затем изменить значение записи, следует выполнить два вы зова процедуры Seek.Один вызов перед операцией Read, а другой - перед операцией Write (т.к. Read после чтения записи переместит указатель к сле дующей записи).

Пример: Создать файл из списка 10 студентов с их оценками (номер, Ф.И.О. и три оценки). Вывести его содержимое на экран, изменить фамилию студента с номером, введенным с клавиатуры, заново прочитать файл.

```
Program  file;
Type
wed  =  record      {Тип wed включает 3 поля: n, fio, bal}
n : byte ;       fio : string[15] ;
bal : array [1..3] of byte;   {Поле bal – массив из 3 оценок }
end;
Var    spisok : wed ;       {Запись spicok  типа  wed}
sp : file  of wed;  {Файл записей типа wed}
procedure vvod;     { процедура создания файла}
var    i,j:byte;
begin
{ оператор assing  находится в основной прграмме }
rewrite ( sp);            {открытие файла для записи}
with spisok do
For   i:=1  to  10  do      begin
n:=i;
writeln (' Введите фамилию -  ',  i );    readln (fio);
writeln (' Введите 3 оценки  ', fio );    For  j:= 1 to 3 do  readln ( bal [j] );
write (sp , spisok);     { запись в файл информации о студенте}
end;
close (sp);      { закрытие файла для записи }
end;
procedure    print;  { процедура  чтения   и печати всего файла }
var    j : byte;
begin
reset ( sp);              {открытие файла для чтения}
writeln (‘                Список студентов:  ‘);
```

```
while not eof (sp) do
with spisok do
begin
Read (sp, spisok);   {чтение данных из файла}
write (n,'  ',fio);      {вывод записи на экран}
For  j:= 1 to 3 do  write (' ', bal [j] );
writeln ;
end;
readln;
close (sp) ;
end;
procedure       work;
var    num: integer;
begin
reset ( sp);              {открытие файла для чтения}
writeln ('номер= ');    readln (num);
seek (sp, num-1); {поиск записи с указанным номером (нумерация записей с 0)}
read (sp,spisok);{чтение и перемещение указателя к сле д. записи}
write ('fio='); writeln (spisok.fio);
seek (sp,filepos(sp)-1); {возвращение к изменяемой записи }
writeln (‘ Введите новую фамилию’  ); readln (spisok.fio);
write (sp, spisok);  {запись в файл измененной записи}
close (sp);
end;

begin      {начало основной программы}
assign (sp,'Vedom.DAT'); {связать файловую перем-ю  sp с файлом Vedom.dat}
vvod;      print; {процедуры создания и чтения файла}
work;   print;   {корректировка и чтение измененного файла}
```

```
readln
end.
```

Задания для самостоятельного решения

Вариант № 1

1. Преобразовать строку так, чтобы буквы каждого слова в ней были отсортированы по алфавиту.
2. Сформировать массив из случайных целых чисел в указанном пользователем диапазоне. Сообщить, есть ли в нем элемент, также указанный пользователем. Перед поиском элементы массива отсортировать (при этом оставив исходный массив без изменений), после чего воспользоваться **бинарным поиском**. В программе должны быть три процедуры - заполнение массива, сортировка, поиск элемента.
3. Найти наименьшее общее кратное (НОК) пар целых положительных чисел через наибольший общий делитель (НОД) по формуле lcm = ab / gcd(a; b), где lcm - НОК, gcd - НОД, a и b - числа.

Вариант № 2

1. Получить десять массивов случайных чисел. Найти среди них тот, сумма элементов которого наибольшая.
2. Написать функцию, заменяющую подстроку, которая начинается с первого вхождения в строку s открывающей квадратной скобки и заканчивается соответствующей ей закрывающей квадратной скобкой, на строку s1, и возвращающую подстроку, заключенную между скобками в качестве своего значения.
3. Дана квадратная матрица. Вычесть последнюю строку из каждой строки (реализовать с помощью подпрограмм).

Вариант № 3

1. В двумерном массиве случайных чисел [1..10,1..10] вычислить сумму элементов побочной диагонали.

(Побочная диагональ проходит из нижнего левого угла в верхний правый.)

Решить задачу двумя способами:

• Используя подпрограмму-функцию.

• Используя подпрограмму-процедуру.

2. Найти средние арифметические пяти массивов, состоящих их десяти целых чисел.
3. Число представленное в шестнадцатеричной системе счисления перевести в десятичную систему счисления.

Вариант № 4

1. Написать программу, которая переводит число из десятичной системы счисления в двоичную или восьмеричную.
2. Двоичное число, введенное пользователем программы, преобразовать в десятичное число. Результат вывести на экран.
3. Дана квадратная матрица. Вычесть последнюю строку из каждой строки (реализовать с помощью подпрограмм).

Вариант № 5

1. Получить десять массивов случайных чисел. Найти среди них тот, сумма элементов которого наибольшая.
2. Найти средние арифметические пяти массивов, состоящих их десяти целых чисел.
3. Найти наименьшее общее кратное (НОК) пар целых положительных чисел через наибольший общий делитель (НОД) по формуле lcm = ab / gcd(a; b), где lcm - НОК, gcd - НОД, a и b - числа.

Контрольные вопросы

1. Что такое процедура?
2. Что такое функция?

Практическая работа № 14

Тема: Организация диалога с пользователем

Цель: Научиться корректно вводить исходную информацию и отображать выходную.

Ход работы:

Теоретическое обоснование

Что такое диалог с компьютером

Если вы исполняли рассмотренные выше программы на компьютере, то почувствовали определенное неудобство при работе с машиной. Во-первых, непонятно, когда машина начинает ожидать ввода данных, какие данные и в каком порядке нужно вводить (это ведь можно и забыть). Во-вторых, результаты получаются в виде чисел на экране, без всяких пояснений их смысла. Ясно, что люди между собой так не общаются.

Любую программу составлять нужно так, чтобы ее исполнение имитировало диалог между компьютером и пользователем в понятной для человека форме.

Прежде чем начать составление программы, нужно продумать сценарий такого диалога.

Например, составим сценарий работы программы, вычисляющей сумму двух целых чисел. На экране компьютера последовательно должны появляться следующие строки (для примера предположим, что будем вводить числа 237 и 658):

Введите первое слагаемое: A = 237

Введите второе слагаемое: B = 658

A + B = 895

Пока!

Здесь курсивом записаны символы, которые выводит компьютер по программе, а прямым жирным шрифтом - символы, вводимые пользователем.

Любой вывод на экран происходит по оператору вывода, записанному в программе.

Следовательно, с помощью оператора вывода на экран выносятся не только результаты решения задачи, но и все элементы диалога со стороны компьютера.

Вот программа, которая реализует наш сценарий:

```
Program Summa;
var A, B : integer;
begin write ('Введите первое слагаемое: A = ');
    readln(A);
    write('Введите второе слагаемое: B = ');
    readln(B);
    writeln;
    writeln('A + B = ', A+B) ;
    writeln('Пока!')
end.
```

В этой программе используется возможность включать в список вывода символьные строки, заключенные в апострофы, и арифметические выражения. Выражение A+B сначала вычисляется, а потом полученное число выводится на экран. Конечно, для вычисления суммы можно было написать отдельный оператор присваивания, но можно и так, как в этом примере.

Еще обратите внимание на оператор writeln без списка вывода. Он обеспечивает пропуск строки на экране.

Пример программирования диалога

Компьютерная программа совсем не обязательно должна иметь математическое содержание. Вот пример сценария, судя по которому компьютер выполняет роль электронной няньки, заботящейся о здоровье школьника. Приводятся два варианта развития сценария, в зависимости от ответа ребенка.

Вариант 1:

```
Ты вчера был болен. Измерь-ка температуру! Сообщи, какая у тебя
температура: 36.5
```

Ты здоров, дружок! Можешь идти в школу.

Желаю успехов!

Вариант 2:

Ты вчера был болен. Измерь-ка температуру!

Сообщи, какая у тебя температура: 37.3

Ты еще болен! Раздевайся и ложись в постель.

Поправляйся, дружок!

Алгоритм этой программы содержит ветвление. Идея алгоритма состоит в том, что значение температуры ребенка сравнивается с величиной нормальной температуры человека: 36,6 °С. И если у ребенка температура выше, то он нездоров. Вот соответствующий алгоритм на АЯ:

```
алг НЯНЬКА
вещ Т
нач вывод "Ты вчера был болен. Измерь-ка температуру!"
    вывод "Сообщи, какая у тебя температура:
    ввод(Т)
    если Т> 36.6
       то вывод "Ты еще болен! Раздевайся и ложись в постель."
    вывод "Поправляйся, дружок!"
    иначе вывод "Ты здоров, дружок!
    Можешь идти в школу."
    вывод "Желаю успехов!"
    кв
кон
```

По этому алгоритму получается следующая программа на Паскале:

```
Program NANNY;
Var T: real;
begin writeln( 'Ты вчера был болен. Измерь-ка температуру! ');
    write ( 'Сообщи, какая у тебя температура: ');
    readln(T);
```

```
    if T>36.6 then begin
        writeln ( 'Ты еще болен! Раздевайся и ложись в постель. ');
        writeln( 'Поправляйся, дружок! ')
    end
    else begin
        writeln('Ты здоров, дружок! Можешь идти в школу. ') ;
        writeln( 'Желаю успехов! ')
    end
end.
```

Обратите внимание на два момента: во-первых, перед словом else ни в коем случае нельзя ставить точку с запятой; во-вторых, в записи и при вводе вещественных чиселцелая и дробная части числа отделяются десятичной точкой.

Составляя подобную программу, вы сами организуете интерфейс компьютера с пользователем вашей программы. Этот интерфейс обязательно должен быть дружественным. Содержание диалога должно быть понятным и удобным.

Задания для самостоятельного решения

Вариант № 1

Написать в Паскале программу диалога компьютером с пользователем. (Спросить имя.Поздороваться,обращаясь по имени.Узнать о возрасте.Любимой книге.Любимом предмете в школе. За ранее спасибо)

Вариант № 2

Постройте алгоритм и составьте программу, по которой будет реализован следующий сценарий: компьютер запрашивает номер дня недели, после ввода компьютер сообщает название этого дня. Например, если ввели 1, то выведется фраза "Это понедельник" и т. д.

Вариант № 3

Напишите программу, вычисляющая сумму двух целых чисел

Контрольные вопросы

1. Что обозначает понятие "диалоговый характер программы"?

2. Какими средствами программируется диалог между пользователем и компьютером?

3. Что обозначает понятие "дружественный интерфейс"?

4. Выполните на компьютере все программы, приведенные в данном параграфе.

Практическая работа № 15

Тема: Работа с графикой

Цель: Овладение начальными навыками работы с графическим режимом экрана, используя модуль GRAPH

Ход работы:

Теоретическое обоснование

При создании программ с оформлением, чтобы пользователю было удобно и просто работать, нужно владеть работой с текстовым и графическим режимом. Для установки текстовых режимов используется стандартная процедура: TextMode(Mode: integer). Для возвращения из графического режима в текстовый TextMode(LastMode).

Для полноценной работы в текстовом режиме нужно подключение модуля Crt .

Для работы в цветном текстовом режиме нужно знать следующие процедуры:

TextColor(Color:byte); - устанавливает цвет выводимых символов,

TextBackGround(Color:Byte); - устанавливает цвет фона.

Нижеописанная шкала поможет вам подобрать нужный цвет!

Цветовая шкала.

Темные цвета	Светлые цвета
0 (Black) – черный	8 (DackGray) - темно-серый

1 (Blue) – синий	9 (LightBlue) - светло-синий
2 (Green) – зеленый	10 (LightGreen) – светло-зеленый
3 (Cyan) – голубой	11 (LightGyan) – светло-голубой
4 (Red) – красный	12 (LightRed) – светло-красный
5 (Magenta) - фиолетовый	13 (LightMagenta) – светло-фиолетовый
6 (Brown) – коричневый	14 (Yellow) – светло-коричневый
7 (LightGray) - светло-серый	15 (White) - белый

Для написания многих программ используется процедура Window из модуля Crt для вывода окон:

Window(x1,y1,x2,y2); ,

где x1 и y1 – координаты левого верхнего угла, а x2 и y2 – координаты правого нижнего угла.

Размер максимального окна (полный экран) – (1,1,80,25).

Задача: Программа проверки таблицы умножения с демонстрацией цветного текстовый режим с окнами.

```
Program pr35;
uses CRT;
var i,x,y,z,j,o:integer;
begin
TextBackGround(1);
ClrScr;
TextColor(14);
GoToXY(15,3);
writeln('Проверь себя, как ты знаешь таблицу умножения?');
TextColor(12);
GoToXY(25,25);
writeln('программа создана в школе №9 г. Нерчинска !');
window(2,5,79,20);
```

```
TextBackGround(3);
for i:=1 to 10 do
begin
randomize;
x:=random(10);
y:=random(10);
ClrScr;
TextColor(14);
GoToXY(25,3);
writeln('Пример №',i);
TextColor(12);
GoToXY(25,7);
write(x,' x ',y ,' = ');
readln(z);
if x*y=z then j:=j+1;
end;
TextColor(14);
GoToXY(15,12);
writeln('Вы ответили на ',j,' вопросов из 10');
if j=10 then o:=5 else if j>7 then o:=4 else
if j>4 then o:=3 else o:=2;
GoToXY(15,13);
writeln('оценка ',o);
GoToXY(15,14);
writeln('Для выхода из программы нажмите <Enter>');
readln;
end.
```

Графический режим.

Для формирования графических изображений используется библиотечный модуль GRAPH, который имеет множество процедур и функций.

Для перехода в графический режим нужно осуществить инициализацию видеорежима:

```
program …;
uses Graph;
var Gd, Gm: Integer;
                ……
begin
Gd := Detect; { автоматическое распознавание типа дисплея (Gd:=0;)}
InitGraph (Gd, Gm, ' '); { файл egavga.bgi находится в одном каталоге с turbo.exe}
if GraphResult <> grOk then Halt(1); { обработка результатов установки}
```

Для завершения работы графического режима используется процедура CloseGraph.

Некоторые процедуры и функции графической системы.

Процедура	Действие
Arc(x,y,нач.угол,конеч.угол, a,в)	Рисует дугу от начального угла к конечному, используя (x,y) как центр
Bar(x1,y1,x2,y2)	Рисует полосу (закрашенный прямоугольник) в текущий стиль и цвет
Circle(x,y,r)	Рисует окружность, используя (x,y) как центр
CloseGraph	Закрывает графическую систему
Ellipse(x,y,нач.уг.,кон.угол,a ,в)	Рисует эллипт-ую дугу от нач. угла к конеч., используя (x,y) как центр
Line(x1,y1,x2,y2)	Рисует линию от (x1,y1) до (x2,y2)
LineRel(dx,dy)	Рисует линию от текущего указателя с указанным смещением
LineTo(x,y)	Рисует линию от текущего указателя до (x,y)
MoveRel(dx,dy)	Передвигает текущий указатель с указанным смещением

MoveTo(x,y)	Передвигает текущий указатель до (x,y)
OutTextXY(x,y,’текст’)	Выводит текст на экран
PieSlice(x,y,н.угол,к.угол,рад.)	Рисует и заполняет сектор, используя (x,y) как центр от нач. угла к конеч.
PutPixel(x,y,номер цвета)	Рисует точку (x,y)
Rectangle(x1,y1,x2,y2)	Рисует закрашенный прямоугольник в текущий стиль и цвет
Sector(x,y,н.угол,к.угол,рад.,а,в)	Рисует и заполняет сектор эллипса
SetColor(цвет)	Устанавливает основной цвет, которым будет осуществляться рисование
SetBkColor	Устанавливает цвет фона
SetFillStyle(цвет1,цвет2)	Устанавливает шаблон заполнения и цвет
Функция	Действие
GetColor	Возвращает текущий цвет
GetBkColor	Возвращает текущий фоновый цвет

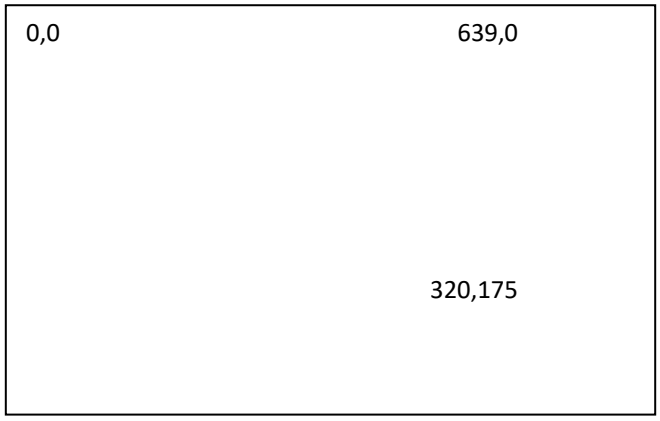
Задача: Построить мишень (ряд окружностей разного радиуса).

```
program pr36;
uses Graph;
var Gd, Gm: Integer;
Radius: Integer;
begin
Gd := Detect;
InitGraph(Gd, Gm, ' ');
if GraphResult <> grOk then Halt(1);
for Radius := 1 to 5 do
Circle(100, 100, Radius * 10);
Readln;
CloseGraph;
```

end.

Моделирование графических программ.

Для того, чтобы моделировать графическое изображение на экране используют систему координат. Отсчёт начинается с верхнего левого угла экрана, имеющий координаты (0,0). Значение X увеличивается слева направо, значение Y сверху вниз (см. рис. «схема экрана в режиме EGAHi»). Количество пикселей экрана зависит от разрешения монитора.



Задача «Звёздное небо».

```
Program pr37;
uses Graph;
var Gd,Gm:Integer;
i,x,y:Integer;
c:word;
begin
Gd := Detect;
InitGraph(Gd, Gm, ' ');
if GraphResult <> grOk then Halt(1);
randomize;
for i:=1 to 500 do   { количество точек «звёзд»}
begin
x:=random(640);
y:=random(480);
c:=random(16);
```

```
putpixel(x,y,c);
end;
readln;
CloseGraph;
end.
```

Задача: «Идёт снег».

```
Program pr38;
uses Graph,Crt;
var Gd,Gm:Integer;
i,x,y,r:Integer;
c:word;
begin
Gd := Detect;
InitGraph(Gd, Gm, ' ');
if GraphResult <> grOk then Halt(1);
setbkcolor(1);
randomize;
for i:=1 to 1000 do {количество «хлопьев снега» (секторов)}
begin
c:=random(16);
setcolor(c);
x:=random(640);
y:=random(480);
r:=random(5);
pieslice(x,y,10,120,r);
end;
Setfillstyle(1,9);
pieslice(100,480,20,160,50);
pieslice(150,480,20,160,50);
Bar(0,480,640,460);
```

```
readln;
CloseGraph;
end.
```

Задача: «Движение прямоугольгиков (убегающие курсоры)».

```
Program pr39;
uses Graph,Crt;
var Gd,Gm:Integer;
i,x1,y1,x2,y2:Integer;
c:word;
begin
Gd := Detect;
InitGraph(Gd, Gm, ' ');
if GraphResult <> grOk then Halt(1);
randomize;
i:=0;
x1:=0;
y1:=480;
x2:=10;
y2:=y1-8;
setbkcolor(1);
for i:=0 to 10 do      { задаётся количество прямоугольников}
begin
while (x1<640) and (y1>0) do  {движение до данных координат}
begin
Setfillstyle(1,12);
bar(x1,y1,x2,y2);
delay(500);
Setfillstyle(1,1);
bar(x1,y1,x2,y2);
x1:=x1+10;
```

```
y1:=y1-8;
x2:=x2+10;
y2:=y2-8;
end;
x1:=0;     {возвращене к текущим координатам}
        y1:=480;
        x2:=10;
        y2:=y1-8;
end;
readln;
CloseGraph;
end.
```

Задания для самостоятельного решения

Вариант № 1

Построить круговую диаграмму, отображающую процентное соотношение отличников, хорошистов и прочих. Для заливки секторов использовать различные шаблоны и цвета.

Вариант № 2

Построить столбиковую диаграмму, отображающую рост цен на бензин.

Вариант № 3

Построить график функции, заданной функцией y=x2.

Контрольные вопросы

1. Что такое CloseGraph?
2. Что такое Setfillstyle?
3. Что такое GraphResult?

Литература

1. О. А. Меженный / Turbo Pascal. Самоучитель. – 2-е изд., стер. – М. : издательский центр «Академия», 2011. – 208 с
2. Ю. Федоренко, Алгоритмы и программы на Turbo Pascal. Учебный курс:,Издательство: Питер, 2014.- 240 с.
3. Н. Культин, Turbo Pascal в задачах и примерах: задачник по программированию на языке Turbo Pascal./Издательство: БХВ-Петербург,2015,256 с.
4. http://pas1.ru/
5. http://pascal.guti.ru/
6. http://www.pascal.helpov.net/