Оглавление

Практическая работа № 1

Тема: «Структура программы на языке PASCAL. Процедуры ввода-вывода, оператор присваивания»

Структура программы на языке Pascal

Структура программы на языке PASCAL состоит из трёх частей:

1) заголовка;
2) описательного блока;
3) исполнительного блока;

Program <имя программы>;
Uses <имя 1>[,<имя 2>...];
Label m1,m2,...;
Const [<константа 1 = значение 1>,...,< константа n = значение n >];
Type [<имя типа1 = тип>,...,< имя типа n = тип>];
Var <переменная 1>[,<переменная 2>,...,<переменная n>]:<тип>;
Procedure <имя процедуры>[(параметры)];
Begin
<тело процедуры>;
End;
Function <имя> (аргументы):<тип значения>;
Begin
<тело функции>;
End;
Begin
<тело программы>;
End.

Процедуры ввода/вывода языка Pascal

Для выполнения ввода/вывода информации существуют четыре стандартные процедуры:

Read (x1,x2,x3) - ввод переменных x_1, x_2, x_3.
Readln (x1,x2,x3) - ввод переменных x_1, x_2, x_3 с переходом курсора на новую строку.
Write (x1,x2,x3) - вывод на экран значения переменных x_1, x_2, x_3;
Writeln (x1,x2,x3) - вывод на экран значения переменных x_1, x_2, x_3 с переходом курсора на новую строку.
Например, **Write** ('x1=') - выводит на экран комментарий, заключённый в апострофах;

Процедура вывода также выводит на экран результат вычисления арифметического выражения, заключённого в скобках, например:
Write (x1+x2).

Оператор присваивания

Оператор присваивания – это один из основных операторов языка Turbo Pascal. В левой части указывается имя переменной, правая часть – это выражение того же типа, что и переменная. Символы «:=» связывают левую и правую части оператора присваивания и означают «присвоить значение». Данные символы рассматриваются как один специальный символ и пишутся слитно.

Например: a**:=** b + c;

Примеры решений задач

1. Составить программу «Приветствие», результат показать преподавателю.

```
Program Hello;
begin
   writeln('Здравствуй, компьютер!');
   write('Привет,');writeln('студент.')
end.
```

2. Найти сумму двух чисел (в программе есть ошибка)

```
Program Summa;                          {заголовок программы}
Var                                     {раздел объявления пере-
   X,Y, Summa: Real;                    менных}
Begin                                   {тело программы}
  Write('Введите числа X и Y');         {вывод сообщения на экран}
  Readln (X,Y);                         {чтение двух чисел}
  Summa:=X+Y;                           {определение суммы}
  Writeln('Сумма чисел X и Y равна', Summa)   {вывод результата}
End.
```

Практическая работа № 2

Тема: «Простые числовые типы данных»

Тип данных Integer

Действия с целыми числами

В типе данных **Integer** (целое) существует пять подтипов, различающихся:

– множеством значений;

– количеством занимаемой памяти;

Тип	*Название*	*Диапазон значений*	*Размер*
Shortint	Короткое целое	-128,127	8 бит
Integer	Целое	- 32768, 32767	16 бит
Longint	Длинное целое	-2147483648, 2147483647	32 бита
Byte	Байт	0, 255	8 бит
Word	Слово	0, 65535	16 бит

Операции над целыми числами

Арифметические операции применимы только к величинам целых и вещественных типов. У операций умножение, деление, деление нацело более высокий приоритет по сравнению со сложением и вычитанием, то есть они автоматически вычисляются первыми.

Операция **Div** – выполняет целочисленное деление. Операция **Mod** находит остаток от целочисленного деления.

Тип данных Real

Действия с действительными числами

Для представления вещественных чисел имеется следующий набор типов:

Тип	*Название*	*Диапазон значений*	*Размер*
Real	Вещественный	2.9E-39, 1.7E+38	6 байт
Single	Одинарный	1.5E-35, 3.4E+38	4 байта
Double	Двойной	5.0E-324, 1.7E+308	8 байт
Extended	Расширенный	3.4E-4932, 1.1E+4932	10 байт
Comp	Комплексный	-9.2E+18, 9.2E+18	8 байт

Стандартные функции и операции

Abs (x) соответствует **|x|**; **ArcTan (x)** соответствует **arctg(x)**; **Cos (x)** соответствует **cos(x)**;
Sin (x) соответствует **sin(x)**; **Exp (x)** соответствует $\mathbf{e^x}$; **Ln (x)** соответствует **ln(x)**;

Sqr (x) соответствует $\mathbf{x^2}$; **Sqrt (x)** соответствует $\sqrt{x}$; **Frac (x)** - дробная часть: X – INT(X);

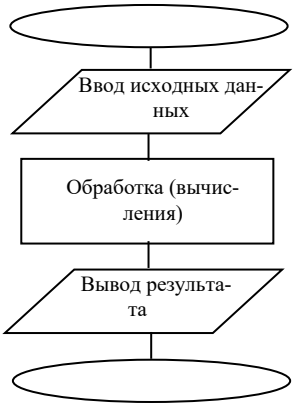
Int (x) возвращает целую часть числа;

Round (x) возвращает число равное целой части (округление по правилам арифметики);

Trunc(x) возвращает округленное число, отбрасывая дробную часть числа x.

Программы с линейной структурой

Программы с линейной структурой являются простейшими и используются, как правило, для реализации простых вычислений по формуле. В программах с линейной структурой инструкции выполняются последовательно. Алгоритм программы с линейной структурой может быть представлен в виде схемы:



Примеры решений задач

1. Составить программу, вычисляющую значение переменной m при данных i, j, k: $m = \frac{i+j}{k+1}$.

```
Program example_1;
Var i, j, k: Integer;
      m: Real;
Begin
 Write('Введите значения для i, j и k =>:');
 Readln(i, j, k);
 m:=(i+j)/(k+1);
 Writeln ('Значение для m=',m);
End.
```

Задачи для самостоятельной работы

1. Дан радиус окружности, подсчитать длину окружности.
2. Дан радиус окружности, подсчитать площадь круга.
3. Дан произвольный треугольник. Известны стороны a и b и угол между ними x. Найти третью сторону c.
4. Дан прямоугольный треугольник с катетами a и b. Найти гипотенузу c.
5. Дан произвольный треугольник со сторонами a, b и c. Найти площадь треугольника.
6. Найти среднеарифметическое 2-х чисел.
7. Вычислить объём шара радиуса R.
8. Найти расстояние между двумя точками с данными координатами.
9. Найти среднее арифметическое трёх заданных чисел.
10. По ребру найти площадь грани, площадь боковой поверхности и объём куба.
11. Для заданного целого числа a напечатать следующую таблицу:
    a
    a^3 a^6
    a^6 a^3 a
12. Даны два действительных числа a и b. Получить их сумму, разность и произведение.
13. Дана длина ребра куба. Найти объём куба и площадь его боковой поверхности.
14. Даны два действительных положительных числа. Найти среднеарифметическое и среднегеометрическое этих чисел.
15. Даны катеты прямоугольного треугольника. Найти его гипотенузу и площадь.
16. Дана сторона равностороннего треугольника. Найти площадь этого треугольника.
17. Найти среднегеометрическое 2-х чисел.
18. Даны гипотенуза и катет прямоугольного треугольника. Найти катет и радиус вписанной окружности.
19. Найти сумму членов арифметической прогрессии по данным значениям: a, d, n.
20. Треугольник задан длинами сторон. Найти длины высот.
21. Треугольник задан длинами сторон. Найти длины биссектрис.
22. Треугольник задан длинами сторон. Найти длины медиан.
23. Треугольник задан длинами сторон. Найти радиусы вписанной и описанной окружностей.
24. Вычислить расстояние между двумя точками с координатами x_1,y_1 и x_2,y_2.
25. Дано действительное число x. Получить дробную часть x; затем число x, округлённое до ближайшего целого; затем число без дробных цифр.
26. Треугольник задан координатами своих вершин. Найти периметр и площадь треугольника.

Практическая работа № 3

Тема: «Условный оператор. Оператор многозначного ветвления»

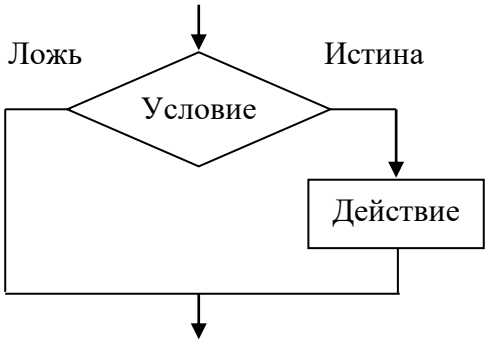
При описании разветвляющихся процессов обычно используют понятие условного и безусловного перехода. Если в программе требуется нарушить порядок выполнения операторов без предварительных проверок каких-либо условий, переход называется безусловным. Для реализации такого перехода служит оператор **goto n** (n – метка). В Паскале метка должна быть описана в разделе **label**, например:

label m, metka, 123

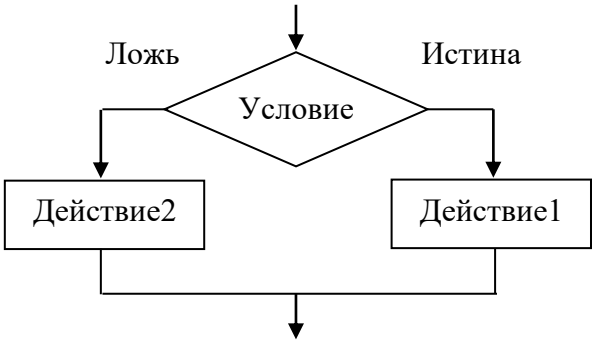
Однако современный стиль программирования предполагает как можно более редкое применение безусловного перехода, а еще лучше - полный отказ от него.

Условный оператор ***IF*** предназначен для изменения порядок выполнения операторов в зависимости от истинности или ложности некоторого условия. Он предписывает выполнять некоторое действие только в том случае, когда выполняется заданное условие. Это условие записывается в виде логического выражения, а действие, которое нужно выполнить, задается в виде последовательности операторов. Существует две конструкции оператора ветвления – простая и расширенная:

Простая конструкция



Расширенная конструкция



Полная развилка:

If < условие> **then** < оператор 1 > { если выполняется только 1 оператор }
else < *оператор2 >;* { после then и else }

или

If < условие> **then**
begin
< оператор 1 >; { если выполняется несколько операторов}
< оператор 2 >;
…
< оператор n >;
end
else
begin
< оператор 1 >; { если выполняется несколько операторов}
< оператор 2 >;
…
< оператор n >;
end;

Укороченная развилка

If < условие> **then** < оператор 1 > ;

Если необходимо проверить несколько условий, то можно использовать составное условие, соединив несколько условий при помощи логических операторов **or** (или), **and** (и).

If (условие1) **and** (условие2) **then** < оператор > ; // если должны одновременно выполняться несколько условий

или

If (условие1) **or** (условие2) **then** < оператор > ; // если должно выполняться хотя бы 1 из условий

Условия в составе составного условия **обязательно** нужно брать в скобки**.**

Оператор выбора *CASE*

В случае необходимости разветвить вычислительный процесс в зависимости от выполнения или невыполнения того или иного условия на более чем две ветви используется оператор выбора (случая, селектора, переключателя). Его использование оказывается более удобным по сравнению с использованием оператора if.

Case S of

C1: <Оператор1>

C2: <Оператор2>

. . . . . .

CN: <ОператорN>

Else <Оператор>

End;

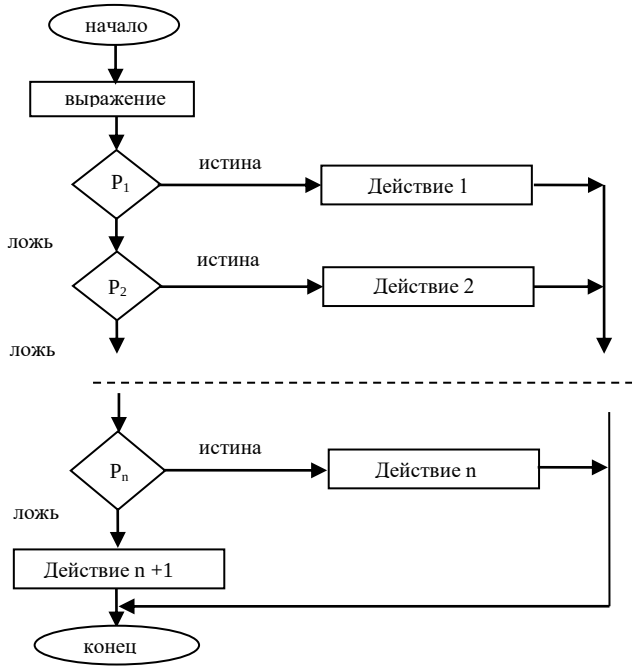
S - выражение порядкового типа значение которого вычисляется;

C1, C2,…,CN – константы, с которыми сравнивается значение выражения S;

<Оператор1>, <Оператор2>, <Оператор N> - операторы, из которых выполняется тот, с константой которого совпадает значение выражения S. Ветвь оператора *else* является необязательной. Если она отсутствует и значение выражения S не совпадает ни с одной константой, весь оператор рассматривается как пустой.

Если для нескольких констант нужно выполнить один и тот же оператор, их можно перечислить через запятую, сопроводив одним оператором.

Схематически такую конструкцию можно изобразить следующим образом:



Примеры решений задач

1. Даны действительные числа x и y. Получить max(x,y).

```
Program maximum;
 Var
  x,y:real;
 Begin
  Write('Введите  x и y');
  Readln(x,y);
  If x>y then m:=x else m:=y;
  Write(' Максимальное m=', m);
 End.
```

2. Ввести число от 1 до 100, если введенное число попадет в диапазон [1..10] определить его четность.

```
 Program Chisla;
Var
  I: integer;
Begin
  Write('Введите  число');
  Readln( i );
  Case i of
        2, 4, 6, 8: Writeln('Четная цифра')
```

```
    1, 3, 5, 7, 9: Writeln('Нечетная цифра')
    10..100: Writeln('Число от 10 до 100')
  else
  Writeln ('Отрицательное число или больше 100')
  end;
end.
```

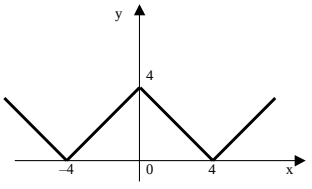
Список обязательных задач

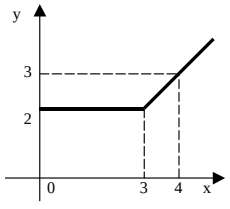
1. Даны действительные числа x,y. Получить: max(x,y), min(x,y).
2. Даны действительные числа x, y, z. Вычислить: $\max(x + y + z, x \cdot y \cdot z)$, $\min^2 (x + y + z/2, x \cdot y \cdot z) + 1$.
3. Даны действительные числа a, b, c. Проверить выполняется ли неравенство $a < b < c$.
4. Найти min значение из трёх величин, определяемых арифметическими выражениями a = sin(x), b = cos(x), c = ln(x) при заданных значениях x.
5. Даны действительные числа a, b, c. Удвоить эти числа, если $a > b > c$ и заменить их противоположными значениями, если это не так.

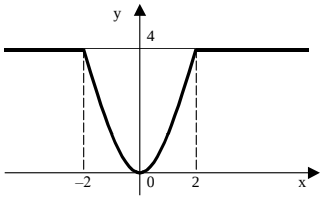
Индивидуальные задания

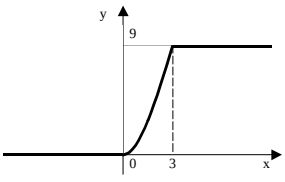
1. Найти max значение из трёх величин, определяемых арифметическими выражениями a = tg(x), b = ctg(x), c = ln(x) при заданных значениях x.
2. Даны действительные числа a, b, c. Проверить выполняется ли неравенство $c < b < a$.
3. Даны действительные числа a, b, c. Утроить эти числа, если $a > b > c$ и заменить их абсолютными значениями, если это не так.
4. Даны два действительных числа. Заменить первое число нулём, если оно меньше или равно второму, и оставить числа без изменения иначе.
5. Даны действительные числа x,y. Меньшее из этих двух чисел заменить их полусуммой, а большее – их удвоенным произведением.
6. Даны действительные числа a, b, c, d. Если $a < b < c < d$, то каждое число заменить наибольшим из них; если $a > b > c > d$, то числа оставить без изменения; иначе все числа заменяются их квадратами.
7. Даны действительные числа a, b, c. Выяснить, имеет ли уравнение $ax^2+bx+c=0$ действительные корни. Если действительные корни имеются, то найти их. В противном случае ответом должно служить сообщение, что действительных корней нет.
8. Даны действительные положительные числа a, b, c, x, y. Выяснить, пройдёт ли кирпич с рёбрами a, b, c в прямоугольное отверстие со сторонами x и y. Просовывать кирпич в отверстие разрешается только так, чтобы каждое из рёбер было параллельно или перпендикулярно каждой из сторон отверстия.
9. Даны два действительных числа. Вывести первое число, если оно больше второго, и оба числа, если это не так.
10. Даны три действительных числа. Выбрать из них те, которые принадлежат интервалу (1,3).
11. Даны три действительных числа. Выбрать из них те, которые принадлежат интервалу (4,6).
12. Даны три действительных числа. Возвести в квадрат те из них, значения которых не отрицательны.
13. Если сумма трёх попарно различных действительных чисел x,y,z меньше единицы, то наименьшее из этих трёх чисел заменить полусуммой двух других; иначе заменить меньшее из x и y полусуммой двух оставшихся значений.
14. Даны два числа. Если первое число больше второго по абсолютной величине, то необходимо уменьшить первое в 5 раз, иначе оставить числа без изменения.
15. Даны действительные положительные числа x, y, z. Выяснить, существует ли треугольник с длинами сторон x, y, z.
16. Составить программу определения большей площади из двух фигур: круга или квадрата. Известно, что сторона квадрата равна a, радиус круга равен r. Вывести и напечатать значение площади большей фигуры.
17. Определить, является ли целое число чётным.
18. Определить, верно ли, что при делении неотрицательного целого числа a на положительное целое число b, получается остаток равный одному из двух заданных чисел r или s.

Вывести значение y(x) в зависимости от введенного значения аргумента (19-24):

19. 

20. 

21. 

22. 

23. $y(x)=\begin{cases} x^2, & \text{при} |x|<1 \\ x-1, & \text{при} |x|\geq 1 \end{cases}$, 24. $y(x)=\begin{cases} x^2+5, & \text{при } x<0 \\ x, & \text{при } x\geq 0 \end{cases}$

25. Даны две точки $A(x_1,y_1)$ и $B(x_2,y_2)$. Составить программу, определяющую, которая из точек находится ближе к началу координат.

26. На плоскости XOY задана своими координатами точка *A*. Указать, где она расположена (на какой оси или в каком координатном угле).

27. Написать программу нахождения суммы большего и меньшего из трех чисел.

Практическая работа № 4

Тема: «Операторы повтора»

При реализации многократного повторения некоторых операций линейной конструкцией необходимо снова и снова повторять одни и те же операторы. Для более компактной реализации этих операций во всех языках используются циклические конструкции, суть которых заключается в том, что вместо многократного переписывания одних и тех же строк программы управление в нужном месте передается предыдущим операторам с тем, чтобы они повторялись.

Имеется два вида циклических алгоритмов: цикл с предусловием (цикл ПОКА) и цикл с постусловием (цикл ДО). Оператор цикла **REPEAT** организует выполнение цикла, состоящего из любого числа операторов, с неизвестным заранее числом повторений. Тело цикла выполняется ***хотя бы один раз***. Выход из цикла осуществляется при истинности некоторого логического выражения. Структура оператора:

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

Оператор цикла **WHILE** организует выполнение одного оператора неизвестное заранее число раз. Выход из цикла осуществляется, если некоторое логическое выражение окажется ложным. Так как истинность логического выражения проверяется в начале каждого повтора, ***тело цикла может не выполняться ни разу.*** Структура оператора цикла имеет вид:

while <условие> **do** <тело цикла>;

или

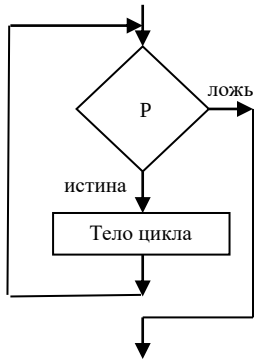
while <условие> **do**

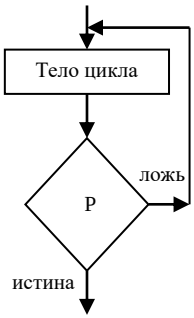
begin

<тело цикла>;

end;

Блок-схемы циклических конструкций могут быть изображены следующим образом:





Оператор цикла **FOR** организует выполнение одного оператора заранее известное число раз. Существует два варианта оператора:

1. от меньшего начального значения до большего конечного значения

for <переменная>:=<нач. значение> **to** <кон. значение> **do**<оператор>;

или

for <переменная>:=<нач. значение> **to** <кон. значение> **do**

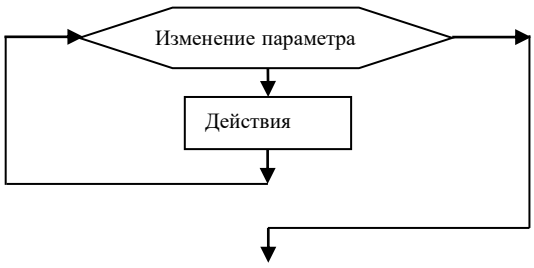
begin

<оператор1>;

<оператор2>;

…

end;



2. от большего начального значения до меньшего конечного значения

for <переменная>:=<нач. значение> **downto** <кон. значение> **do** <оператор>;

или

for <переменная>:=<нач. значение> **downto** <кон. значение> **do**

begin

<оператор1>;

<оператор2>;

…

end;

Выполнение очередного повтора включает в себя сначала выполнение оператора, а затем присваивание переменной цикла следующего большего значения (в первом случае) или следующего меньшего (во втором варианте). Особенностью цикла ***For*** (цикла с параметром) является то, что число повторений операторов цикла должно быть известно заранее.

Примеры решений задач.

1. Вычислить N! (факториал, т.е. произведение всех чисел от 1 до N).

```
Program Faktorial;
  Var n, i, f: integer;
Begin
    f:=1;
    Write('Введите n=');
    Readln(n);
    For i:=2 to n do   // для i от 2 до n делать
        f:=f*i;        //  f приравнять к предыд. знач f, умн. на текущее значение i
    Writeln(n,'!=',f);
  End.
```

2. Найти сумму цифр в записи данного натурального числа;

```
 Program SUM;
 uses crt;
 Var a,b,s,k:Integer;
 Begin
   Readln(a);
   s:=0;
   While a<>0 do    // пока а не равно 0
   begin          // выполнять следующие операторы
      b:=a mod 10;    // b приравнять к остатку от деления а на 10 (это всегда последняя цифра числа)
      s:=s+b;      // в s будут суммироваться последние цифры
      a:=a div 10 // а приравняем к а, деленному без остатка на 10
   end;           // (это отбросит последнюю цифру числа), и так пока а не станет равно 0
   Writeln(s);
End.
```

Список задач

Часть 1.

1. Написать программу, которая выводит на экран ваши имя и фамилию нужное количество раз.
2. Найти сумму квадратов чисел от m до n.
3. Найти сумму квадратов нечётных чисел в интервале, заданном значениями переменных m и n;
4. Найти сумму квадратов четных чисел в интервале, заданном значениями переменных m и n;
5. Найти сумму целых положительных чисел, кратных 4 и меньших 100.
6. Вывести на экран числовой ряд действительных чисел от n до m с шагом 0,2.
7. Написать программу, которая выводит таблицу квадратов первых n целых положительных чисел. Ниже приведен рекомендуемый вид экрана во время работы программы (для первых 10 чисел).

```
Таблица квадратов.
Число    Квадрат
---------------
1          1
2          4
3          9
4          16
5          25
6          36
7          49
8          64
9          81
10         100
```

8. Дано целое число a и натуральное (целое неотрицательное) число n. Вычислить a^n. Решить можно, умножив *a* на себя *n* раз в цикле.
9. Составить программу, печатающую квадраты всех натуральных чисел от 0 до заданного натурального n.
10. Даны целые числа K и N (N > 0). Вывести N раз число K.
11. Даны два целых числа A и B (A < B). Вывести в порядке возрастания все целые числа, расположенные между A и B (включая сами числа A и B), а также количество N этих чисел.
12. Даны два целых числа A и B (A < B). Вывести в порядке убывания все целые числа, расположенные между A и B (не включая числа A и B), а также количество N этих чисел.
13. Даны два целых числа A и B (A < B). Найти сумму всех целых чисел от A до B включительно.
14. Даны два целых числа A и B (A < B). Найти произведение всех целых чисел от A до B включительно.

15. Даны два целых числа A и B (A < B). Найти среднее арифметическое всех целых чисел от A до B включительно.
16. Дано число n. В интервале от 1 до n сложить все четные и перемножить все нечетные числа.
17. Дано число n. В интервале от 1 до n сложить все числа, которые делятся на 3 без остатка.
18. Дано число n. В интервале от 1 до n сложить все числа, которые делятся на 5 без остатка.
19. Дано число n. В интервале от 1 до n сложить все числа, которые делятся на 3 и не делятся на 2.
20. Дано число n. В интервале от 1 до n сложить все числа, которые делятся на 5 и не делятся на 3.
21. Вывести на экран числовой ряд действительных чисел от 10 до 20 с шагом 0,2.
22. Дано число n. В интервале от 1 до n сложить все числа, которые делятся на 3, и перемножить все, делящиеся на 2.

Часть 2.

Дано натуральное число n (n<9999). Найти (1-6):

1. кол-во цифр в числе n (см. пример с нахождением суммы цифр числа).
2. последнюю и первую цифру числа (см. пример с нахождением суммы цифр числа).
3. дано число m. Найти сумму m-последних цифр числа n (см. пример с нахождением суммы цифр числа).
4. выяснить, входит ли цифра 3 в запись числа n (см. пример с нахождением суммы цифр числа).
5. Выяснить, сколько четных цифр в числе (см. пример с нахождением суммы цифр числа).
6. Выяснить, сколько нечетных цифр в числе (см. пример с нахождением суммы цифр числа).

Вычислить (7-12)

7. $\sum_{i=1}^{100} \frac{1}{i^2}$;

8. $\sum_{i=1}^{50} \frac{1}{i^3}$;

9. $\sum_{i=1}^{10} \frac{1}{i!}$;

10. $\sum_{i=1}^{128} \frac{1}{(2i)^2}$;

11. $\prod_{i=1}^{52} \frac{i^2}{i^2+2i+3}$;

12. $\prod_{i=2}^{100} \frac{i+1}{i+2}$;

Практическая работа № 5

Тема: «Функции и процедуры»

Вспомогательные алгоритмы организуются двумя способами в зависимости от использования:

1) ПРОЦЕДУРЫ:

Программа процедура предназначена для выполнения какой-то законченной последовательности действий. Любая процедура начинается с заголовка. В отличии от основной программы заголовок в процедуре обязателен. Он состоит из зарезервированного слова **Procedure,** за которым следует идентификатор имени процедуры, а далее в круглых скобках список формальных параметров:

Procedure <имя процедуры> (список формальных параметров)

За заголовком могут идти такие же разделы, что и в основной программе. В отличии от основной программы процедура завершается не точкой, а точкой с запятой.

Procedure < имя процедуры >[(< аргументы >: < тип аргументов>]; **Var** <значение>:< тип значения >])];
Var < переменная 1>[,< переменная 2>,...]: тип;

Begin

< тело процедуры>;

End;

Для вызова процедуры из основной программы или другой подпрограммы следует записать оператор, состоящий из имени процедуры и списка фактических параметров, которые должны совпадать по количеству и типу с формальными параметрами процедуры.

2) ФУНКЦИИ:

Подпрограмма-функция предназначена для вычисления какого-либо параметра. У подпрограммы-функции два основных отличия от процедуры:

а) заголовок состоит из слова **function,** за которым следует имя функции, далее в круглых скобках - список формальных параметров, затем через двоеточие записывается тип функции - тип возвращаемого параметра. Функция может возвращать параметры следующих типов: любого порядкового, любого вещественного, стандартного строкового и символьного.

б) в теле функции хотя бы раз имени функции должно быть присвоено значение.

Function < имя функции >[(< аргументы >: < тип аргументов>):< тип значения функции>];

Var < переменная 1>[,< переменная 2>,...]: тип;

Begin

<тело функции>;

<имя функции>:=<выражение значение функции>;

End;

Примеры решений задач.

1. Пример программы с созданием функции для вычисления факториала.*(факториал числа n обозначается n!=1*2*3*...*n, т.е. произведение всех чисел от 1 до n)*

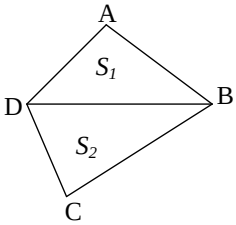
```
program f1;
uses crt;
var a:integer;

Function Factorial(N:Byte):Longint; // определение функции
Var
   Fact:longint;
   I:byte;
Begin
   Fact:=n;
   For i:=n-1 downto 2 do
            Fact:=fact*i;
   Factorial:=fact;  //имени функции обязательно присваиваем значение, которое она будет возвращать
End;

begin
     write('Введите число:');
     readln(a);
     writeln('факториал вашего числа равен ',Factorial(a));
end.
```

Пример запуска программы:

```
Введите число:3
факториал вашего числа равен 6
```



2. Вычислить площадь произвольного четырехугольника, у которого заданы значения четырех сторон и диагонали. Для решения задачи необходимо разбить четырехугольник на два треугольника и вычислить их площади по формуле Герона.

Вариант 1. *Использование процедуры без параметров.*

```
Program Square1;
Var AB, BC, CD, DA, DB, a, b, c, p, s, S1, S2: Real; // объявление глобальных переменных
Procedure geron;  // определение процедуры без параметров
Begin        // a, b, c – текущие стороны треугольника, p – текущий полупериметр
   p:= (a+b+c)/2;  // вычисление полупериметра
   s:= sqrt(p*(p-a)*(p-b)*(p-c));  // вычисление площади и её запись в глобальную переменную s
```

```
End;
Begin
   WriteLn ('Введите длины сторон AB, BC, CD, DA, DB');
   ReadLn (AB, BC, CD, DA, DB);
   a:= AB; b:=DB; c:=DA; // текущими становятся стороны первого треугольника
   geron;   // вызов процедуры без параметров для первого треугольника
   S1:=s;    // сохранение в переменную S1 значения площади первого треугольника
   a:= BC; b:=CD; c:=DB; // текущими становятся стороны второго треугольника
   geron;  // вызов процедуры без параметров для второго треугольника
   S2:=s; // сохранение в переменную S2 значения площади второго треугольника
   WriteLn ('Площадь равна:', S1+S2:5:2) // вывод результирующей площади на экран в формате 5 знаков
                                         //под число и 2 знака под дробную часть
End.
```

Вариант 2. *Использование процедуры без параметров. (все то же самое, но переменная со значением полупериметра является локальной для процедуры)*

```
Program Square2;
Var AB, BC, CD, DA, DB, a, b, c, s, S1, S2: Real;
Procedure geron;
Var p:Real;
Begin
   p:= (a+b+c)/2;
   s:= sqrt(p*(p-a)*(p-b)*(p-c));
End;
Begin
   WriteLn ('Введите длины сторон AB, BC, CD, DA, DB');
   ReadLn (AB, BC, CD, DA, DB);
   a:= AB; b:=DB; c:=DA;
   geron;   S1:=s;
   a:= BC; b:=CD; c:=DB;
   geron;  S2:=s;
   WriteLn ('Площадь равна:', S1+S2:5:2)
End.
```

AB, BC, CD, DA, DB, a, b, c, s, S1, S2 – глобальные переменные;
p – локальная переменная.

Вариант 3. *Использование процедуры с параметрами.*

```
Program Square3;
Var AB, BC, CD, DA, DB, s, S1, S2: Real;
Procedure geron (a, b, c: real); // определение процедуры с параметрами a, b, c
Var p:Real;
Begin
   p:= (a+b+c)/2;
   s:= sqrt(p*(p-a)*(p-b)*(p-c)); // s все еще глобальная переменная, которой присваивается значение
                                  // в теле процедуры
End;
Begin
   WriteLn ('Введите длины сторон AB, BC, CD, DA, DB');
   ReadLn (AB, BC, CD, DA, DB);
   geron (AB, DB, DA);  // вызов процедуры с параметрами для первого треугольника с нужными сторонами
   S1:=s;        // сохранение в переменную S1 значения площади первого треугольника
   geron (BC, CD, DB);  // вызов процедуры с параметрами для второго треугольника с нужными сторонами
   S2:=s;      // сохранение в переменную S2 значения площади второго треугольника
   WriteLn ('Площадь равна:', S1+S2:5:2);
End.
```

a, b, c – формальные параметры (используются в определении функции).
AB, BC, CD, DA, DB – фактические параметры (подставляются во время вызова функции);

Вариант 4. *Использование процедуры с возвращением значения.*

```
Program Square4;
Var AB, BC, CD, DA, DB, S1, S2: Real;
Procedure geron (a, b, c: real; Var s:Real); //определение процедуры с параметрами a, b, c, s,
                                             //причем s – возвращаемый параметр
Var p:Real;
Begin
   p:= (a+b+c)/2;  s:= sqrt(p*(p-a)*(p-b)*(p-c));
End;
Begin
   WriteLn ('Введите длины сторон AB, BC, CD, DA, DB');
   ReadLn (AB, BC, CD, DA, DB);
   geron (AB, DB, DA, S1); // вызов процедуры с параметрами для первого треугольника с нужными
                           //сторонами, в S1 сохраняется значение площади первого треугольника
   geron (BC, CD, DB, S2); // вызов процедуры с параметрами для второго треугольника с нужными
                           //сторонами, в S1 сохраняется значение площади второго треугольника
```

```
   WriteLn ('Площадь равна:', S1+S2:5:2)
End.
```

Вариант 5. *Использование функции.*

```
Program Square5;
Var AB, BC, CD, DA, DB, S: Real;
Function geron (a, b, c:  Real): Real; // определение функции с параметрами a, b, c, сама функция
                                       //будет возвращать значение площади типа real
Var p:Real;
Begin
   p:= (a+b+c)/2;
   geron:= sqrt(p*(p-a)*(p-b)*(p-c)); // переменной с именем функции присваиваем значение,
                                      //которое она будет возвращать
End;
Begin
   WriteLn ('Введите длины сторон AB, BC, CD, DA, DB');
   ReadLn (AB, BC, CD, DA, DB);
   S:= geron (AB, DB, DA) + geron (BC, CD, DB);  // в S сохраняем сумму площадей двух треугольников
   WriteLn ('Площадь равна:', S)
End.
```

ОБЯЗАТЕЛЬНАЯ ЗАДАЧА

Напишите процедуру-заставку к программам в виде

```
******** ********** ********** ****

*                  Программа                  *

*              <название программы>           *

*                  Автор: <ФИО>               *

******** **** ********** **********
```

Заставка должна выводиться в начале всех программ, которые Вы будете сдавать.

Список задач (в каждой нужно создать процедуру или функцию**)**

1. Даны действительные числа s, t. Получить: $f(t, -2s, 1.17) + f(2.2, t, s-t)$, где $f(a, b, c) = \dfrac{2a - b - \sin c}{5 + |c|}$
2. Даны действительные числа s, t. Получить: $f(2t, 3s, 5) + f(1, -t, s+t)$, где $f(a, b, c) = \dfrac{2a + b + \sin c}{7.5 - c}$
3. Даны действительные числа s, t. Получить: $f(t, s+t, -7) + f(3, s-t, 0.5s)$, где $f(a, b, c) = \dfrac{a + 2b + \ln c}{a + c}$
4. Даны действительные числа s, t. Получить: $g(1/2, s) + g(t, s) - g(2s-1, st)$, где $g(a, b) = \dfrac{a^2 + b^2}{a^2 + 2ab + 3b^2 + 4}$
5. Даны действительные числа s, t. Получить: $g(1/3, s+t) + g(t, s) - g(2s, st)$, где $g(a, b) = \dfrac{a^3 + b^3}{a^2 + 3ab + 4b^3 - 5}$
6. Даны действительные числа a, b, c. Получить: $\dfrac{\max(a, a+b) + \max(a, b+c)}{1 + \max(a + bc, 1, 15)}$
7. Даны действительные числа a, b, c. Получить: $\dfrac{\max(a, a*b) + \max(c, b/c)}{8 + \max(c + ba, 3.25)}$
8. Даны действительные числа a, b, c. Получить: $\dfrac{\min(c, a-b) - \min(b, a+2c)}{100 - \min(a + c, 2b)}$
9. Даны действительные числа a, b. Получить: u=min (a,b), v=min (ab, a+b), f= min ($u+v^2$,3.14)
10. Даны действительные числа a, b, c. Получить: u=min (a,b,c), v=min (abc, a+b+c), f= min (u^2+v^2,u*v)
11. Даны действительные числа a, b, c. Получить: u=min (a,b,c), v=min (2a+2b+2c, a*b+c), f= min (u^2+v , u+v)

12. Напишите функцию возведения в степень по формуле: $a^n = Exp(Ln(a)*n)$. Используйте ее в программе для возведения в нужную степень введенного с клавиатуры числа.
13. Дано натуральное число n. Среди чисел 1, 2, 3, ..., n найти все те, которые можно представить в виде сумм квадратов двух натуральных чисел. Определить процедуру, позволяющую распознавать полные квадраты.
14. Напишите программу поиска максимального из четырех чисел с использованием подпрограммы поиска большего из двух.
15. Дано число n. Вычислите сумму: 1! + 2! + 3! + ... +n!, создав функцию вычисления факториала числа.
16. Создайте программу для подсчета числа четных цифр, используемых в записи введенного числа M.
17. Создайте программу для подсчета числа нечетных цифр, используемых в записи введенного числа M.
18. Создайте программу вычисления суммы трехзначных чисел, в десятичной записи которых нет четных цифр.
19. Создайте программу вычисления суммы трехзначных чисел, в десятичной записи которых нет нечетных цифр.
20. Вычислить площадь правильного шестиугольника со стороной a, используя подпрограмму вычисления площади треугольника.
21. Составить программу, определяющую, в каком из данных двух чисел больше цифр (создать подпрограмму для вычисления кол-ва цифр в числе).
22. Дано натуральное число *n.* Выяснить, можно ли представить его в виде произведения трех последовательных натуральных чисел (сделать функцию для данных целей).

Практическая работа № 6

Тема: «Одномерные массивы»

Массив это упорядоченная совокупность конечного числа данных одного типа. Простейшим примером массива может служить линейная таблица. Значения, образующие линейную таблицу, являются элементами массива, а их порядковые номера в таблице называются индексами. Можно сказать, что одномерный массив соответствует понятию вектора. Индекс определяет положение элемента массива относительно его начала.

Общая форма описания переменной:

Var <имя>: Array [<тип-индексов>] of<тип-элементов>

Если массив используется в качестве возвращаемого параметра в процедуре, то необходимо его описать с помощью раздела ***type***:

type <имя типа>=**array**[<список индексных типов>] **of** <тип компонент>;
var <идентификатор массива> : <имя типа>;

Выбор отдельной компоненты одномерного массива осуществляется указанием идентификатора массива, за которым в квадратных скобках следует индексное выражение. Индексное выражение должно давать значения, лежащие в диапазоне, определяемом типом индекса.

Например: M[1],M[2],…,M[N].

Для организации автоматического ввода значений следует воспользоваться функцией библиотеки CRT **Random**. Эта функция возвращает случайное число из диапазона от 0 до n-1. Для этого необходимо в основной программе инициализировать датчик случайных чисел командой **Randomize**.

Формат вызова функции Random:

Random (m), где m – значение, указывающее на правую границу диапазона выдаваемых значений.

Например: создание массива размерностью 10, числами из диапазона от 0 до 9.

```
Program Vector_Full;
uses crt;
Const n=10;
Type vector = array [1..n] of Integer;
Var v : vector;
       i : integer;

Procedure Enter (Var vect: vector);
Var i: Integer;
  Begin
    For i:=1 to n do
      vect[i]:=Random(10);
  End;
Begin
  Randomize;
  Enter (v);
  for i:=1 to 10 do
     write(v[i]:3);
End.
```

Пример решения задачи

Найти максимальное число в массиве.

```
program maximum;
uses crt;
Var
a:array [1..10] of integer;
i: byte;
m: integer;

begin
  randomize;
```

```
  for i:=1 to 10 do
  begin
    a[i]:=random(10);
    write(a[i]:3);
  end;
  writeln;
  m:=a[1];
  for i:=2 to 10 do
      if a[i]>m then m:=a[i];
  writeln ('maximum = ', m);
end.
```

Список задач

Часть 1

1. Дан массив из n чисел. Определить количество нечетных чисел в массиве.
2. Дан массив из n чисел. Определить количество чисел, кратных 3 и некратных 5.
3. Дан массив из n чисел. Найти те элементы массива, которые являются удвоенными нечётными числами.
4. Дан массив из n чисел. Найти те элементы массива, которые при делении на 7 дают остаток 1,2 или 5.
5. Дан массив из n чисел. Получить сумму элементов массива, которые кратны 5.
6. Дан массив из n чисел. Получить сумму элементов массива, которые нечётны и отрицательны.
7. Дан массив из n чисел. Найти количество и сумму тех членов последовательности, которые делятся на 5 и не делятся на 7.
8. Дан массив действительных чисел, размерность которого < 100. Подсчитать, сколько в нем отрицательных, положительных и нулевых элементов.
9. Дан массив из n целых чисел и целые числа p, q (p>q>0). В массиве заменить нулями элементы, модуль которых при делении на p даёт в остатке q.
10. Дан массив из n целых чисел и целое число p. Получить сумму элементов массива, кратных p.
11. Дан массив из n чисел. Получить удвоенную сумму всех его положительных элементов.
12. Дан массив из n чисел. Все отрицательные числа увеличить на 0.5, а все неотрицательные на 0.1.
13. Дан массив из n чисел. В последовательности все элементы, меньше 2, заменить нулями.
14. Дан массив из n чисел. Получить сумму элементов, принадлежащих отрезку [3,7], а также число таких элементов.
15. Дан массив из n действительных чисел. В последовательности все неотрицательные элементы, не принадлежащие отрезку [1,3], заменить на 1. Кроме того, получить число отрицательных элементов и число элементов, принадлежащих отрезку [1,3].
16. Дан массив из n чисел. Получить сумму положительных и число отрицательных элементов последовательности.
17. Дан массив из n чисел. Заменить все, большие 7, элементы последовательности числом 7. Вычислить количество таких элементов.
18. Дан массив из n чисел и число A. Если в последовательности есть хотя бы один элемент, равный A, то получить сумму всех элементов, следующих за первым таким элементом.
19. Дан массив из n чисел. Получить другой массив, который отличается от исходного тем, что все нечётные элементы удвоены.
20. Дан массив из n чисел и число A. Определить, каким по счёту идёт в последовательности элемент, равный A. Если такого элемента нет, то ответом должно быть число 0.

Дан массив из n чисел. Получить (21-23):

21. $\max(A_1,...,A_n)$; $\min(A_1,...,A_n)$;
22. $\max(A_2, A_4, ...)$; $\min(A_1, A_3,...)$;
23. $\min(A_2, A_4,...) + \max(A_1, A_3, ...)$;

Часть 2

1. Дан массив из n чисел. Найти наименьшее из чётных чисел массива.
2. Дан массив из n чисел. Найти наибольшее из нечетных чисел массива.
3. Дан массив из n чисел. Выяснить, имеются ли в последовательности два идущих подряд нулевых элемента.
4. Дан массив из n чисел. Выяснить, какое число встречается в массиве раньше – положительное или отрицательное. Если все элементы массива равны нулю, то сообщить об этом.
5. Дан массив из n чисел. Найти номер первого чётного элемента массива; если чётных элементов нет, то ответом должно быть число 0.
6. Дан массив из n чисел. Найти номер последнего нечётного элемента последовательности; если нечётных элементов нет, то ответом должно быть число 0.

7. Дан массив из n чисел. Вычислить произведение элементов в массиве до первого отрицательного.
8. Дан массив из n чисел. Вычислить сумму элементов в массиве после первого отрицательного.
9. Дан массив из n чисел. Вычислить произведение элементов массива до первого нуля.
10. Дан массив из n чисел. Вычислить сумму элементов после первого нуля.
11. Дан массив из n чисел и число A. Подсчитать количество элементов, больших A.
12. Дан массив из n чисел. Составить программу для вычисления суммы элементов массива, имеющих чётные индексы.
13. Дан массив из n чисел. Составить программу для вычисления произведения элементов массива, имеющих нечётные индексы.

Даны массив (A_1, A_2, ..., A_{17}). Получить новый массив, состоящий из (14-16):

14. A_{11}, A_{12}, ..., A_{17}, A_1, A_2, ..., A_{10};
15. A_{11}, A_{12}, ..., A_{17}, A_{10}, A_9, ..., A_1;
16. A_1, A_3, ..., A_{17}, A_2, A_4, ..., A_{16}.

17. Дан массив из n чисел. Составить программу для нахождения среднеарифметического элементов массива.
18. Дан массив из n чисел. Все четные элементы умножить на 3, а к нечетным прибавить 2.
19. Дан массив из n чисел и число A. Заменить все элементы, делящиеся на A без остатка, на -11.
20. Дан массив из n чисел. Найти минимальный элемент и вычесть его из всех остальных элементов массива.
21. Дан массив из n чисел. Поменять в массиве местами наибольший и наименьший элементы.
22. Дан массив из n чисел. Поменять в массиве местами наибольший и последний элементы.
23. Дан массив из n чисел. Поменять в массиве местами наименьший и первый элементы.

Практическая работа № 7

Тема: «Строковый тип данных»

Тип-строка – последовательность символов произвольной длины (до 255). Строку можно рассматривать как массив символов, однако в связи с широким использованием строк и некоторыми особенностями по сравнению со стандартными массивами они выделены в отдельный тип данных.

У типа-строки в квадратных скобках может быть указан его размер (от 1 до 255). Если размер строки не указан, он считается равным 255, например:

```
Var
      Str: string[80]; {объявление строки с длиной 80}
      MaxStr: string; {объявление строки с длиной 255}
Const
      January: string[10]='Январь'; {объявление именованной константы со значением 'Январь'}
```

К любому элементу строки можно обратиться как к элементу массива, например:

```
Var    str: string;
begin
      str:='Hello, world';
      writeln(str[5]);      // на экране появится буква o, т.к. она – пятый символ в строке
end.
```

Процедуры и функции обработки строковых переменных*:*

Length(S: **String**): **Integer;** - функция определения длины строкового выражения.

Пример использования в программе:

```
Var
      S: String;
begin
      Readln (S);
      Writeln(' " ', S, ' " ');
      Writeln('длина строки = ', Length(S));
end.
```

Copy(S: **String**; Index: **Integer**; Count: **Integer**): **String**; – функция выделения подстроки из строки (из строки S начиная с элемента с номером Index копируется кол-во элементов, равное Count).

Пример использования в программе:

```
Var S: String;
begin
      S := 'ABCDEF';
      S := Copy(S, 2, 3); { в результате S будет равно  'BCD' }
end.
```

Concat(s1 [, s2,..., sn]: **String): String;** - функция склейки строк; аналогично операции «+» - последовательное соединение строк.

Пример использования в программе:

```
Var
      S: String;
begin
      S := Concat('ABC', 'DEF'); { в результате S будет равно  'ABCDEF' }
end.
```

Delete(**Var** S: **String;** Index: **Integer**; Count: **Integer**); **-** процедура удаления подстроки из строки (из строки S удаляется Count элементов, начиная с элемента с номером Index).

Пример использования в программе:

```
Var
s: string;
begin
```

```
        s := 'Honest Abe Lincoln';
        Delete(s,8,4);
        Writeln(s); { в результате выведется  'Honest Lincoln' }
end.
```

Insert(Word1: **String**; **Var** Word2: **String**; Index: **Integer**); - процедура вставки в строку *Word2* подстроки *Word1* с позиции Index.

Пример использования в программе:

```
Var
        S: String;
begin
        S := 'Honest Lincoln';
        Insert('Abe ', S, 8);  { в результате S будет равно  'Honest Abe Lincoln' }
end.
```

Pos(Substr: **String**; S: **String**): **Byte**; - функция определения позиции подстроки в строке (возвращает номер элемента в строке S, с которого начинается подстрока Substr);

Пример использования в программе:

```
        Var S: String;
        begin
                S := '   123.5';
                while Pos(' ', S) > 0 do // пока в строке S встречается пробел
                S[Pos(' ', S)] := '0';    // заменять его нулем
        end.
```

Примеры решений задач

1. Объединение двух строк.

```
Program  assign;
Var
        Str, str1,str2: string[80];
Begin
        Str1:='Turbo';
        Str2:='Pascal';
        Str:=str1+str2;
        Writeln (str);
End.
```

Список задач

Часть 1.

1. Дана строка, подсчитать сколько раз встречается буква a.
2. Выделить символы, заключённые в фигурные скобки.
3. Подсчитать наибольшее число букв a, идущих подряд в введенной последовательности символов.
4. Удалить символы, заключённые в фигурные скобки.
5. Дана строка. Подсчитать, сколько раз среди введенных символов встречается буква b.
6. Дана строка, заканчивающаяся точкой. Подсчитать, сколько слов в строке.
7. Дана строка, содержащая английский текст. Найти количество слов, начинающихся с буквы b.
8. Дана строка. Подсчитать, сколько в ней букв r, к, t.
9. Дана строка. Определить, сколько в ней символов * ; :
10. Дана строка, содержащая текст. Найти длину самого короткого слова и самого длинного слова.
11. Дана строка символов, среди которых есть двоеточие. Определить, сколько символов ему предшествует.
12. Дана строка, содержащая текст, заканчивающийся точкой. Вывести на экран слова, содержащие три буквы.
13. Удалить группу символов, расположенных между круглыми скобками включая сами скобки.
14. Если в заданный текст входит каждая из букв слова key, тогда напечатать yes, иначе no.
15. Напечатать заданный текст, удалив из него лишние пробелы, т.е. из нескольких подряд идущих пробелов оставить только один.
16. Логической переменной b присвоить значение **true**, если между литерами 'a' и 'z' нет иных символов, кроме строчных латинских букв, и значение **false** иначе.
17. Дана строка. Подсчитать сколько раз среди символов встречается символ «+» и сколько раз символ «*».
18. Дана строка. Подсчитать общее число вхождений символов +, – и * в строку.
19. Дана строка, заменить в ней все восклицательные знаки точками.

20. Дана строка, заменить в ней каждую точку многоточием.
21. Дана строка. Преобразовать ее, удалив каждый символ «*» и повторив каждый символ, отличный от «*».
22. Дана строка, среди символов которой есть двоеточие. Получить все символы, расположенные до первого двоеточия включительно.
23. Дана строка, среди символов которой есть двоеточие. Получить все символы, расположенные после первого двоеточия включительно.
24. Дана строка, среди символов которой есть двоеточие. Получить все символы, расположенные между первым и вторым двоеточиями. Если второго двоеточия нет, то получить все символы после первого двоеточия.
25. Дана строка. Подсчитать наибольшее количество идущих подряд пробелов.

Часть 2.

Дана строка. Группы символов, разделённых пробелами (одним или несколькими) и не содержащим пробелов внутри себя будем называть словами. Задания (1-9):

1. подсчитать количество букв "а" в последнем слове данной последовательности.
2. найти количество слов, начинающихся с буквы "с".
3. найти количество слов, у которых первый и последний символы совпадают.
4. подсчитать количество слов в данной последовательности.
5. преобразовать данную последовательность, заменяя всякое вхождение слова "это" на слово "то".
6. найти длину самого короткого слова.
7. найти длину самого длинного слова.
8. удалить все символы, не являющиеся буквами.
9. заменить все малые буквы одноимёнными большими.

10. Дана строка. Определить, сколько раз входит в нее группа букв abc.
11. Дана строка. Подсчитать, сколько различных символов встречается в ней. Вывести их на экран.
12. Дана строка. Подсчитать самую длинную последовательность подряд идущих букв a.
13. Найти последнее самое короткое слово предложения.
14. Имеется строка, содержащая буквы латинского алфавита и цифры. Вывести на экран длину наибольшей последовательности цифр, идущих подряд.
15. Дан набор слов, разделенных точкой с запятой (;). Набор заканчивается двоеточием (:). Определить, сколько в нем слов, заканчивающихся буквой a.
16. Найти первое самое короткое слово предложения.
17. Дана строка. Указать те слова, которые содержат хотя бы одну букву к.
18. В строке заменить все двоеточия (:) точкой с запятой (;). Подсчитать количество замен.
19. В строке удалить символ «двоеточие» (:) и подсчитать количество удаленных символов.
20. В строке между словами вставить вместо пробела запятую и пробел.
21. Составить процедуру, заменяющую в исходной строке символов все единицы нулями и все нули единицами. Замена должна выполняться, начиная с заданной позиции строки.
22. Найти самое длинное слово в предложении.
23. Найти первое симметричное слово в предложении.
24. Заменить заданное слово предложения на другое слово.

Практическая работа № 8

Тема: «Двумерные массивы. Преобразование и построение матриц. Матричная алгебра»

Двумерные массивы являются аналогами матриц. Первый индекс элемента двумерного массива определяет номер строки, а второй – номер столбца, на пересечении которых расположен элемент. Строки и столбцы нумеруются либо от единого заранее установленного минимального значения индекса, либо от граничного значения, заданного одновременно с объявлением массива**.**

Описание двухмерного массива производится следующим образом:

```
Const
   N = ранг_матрицы;
Type
   matr=array [1..N,1..N] of тип_элементов_матрицы;
Var
   Имя_матрицы : matr;
```

Или по упрощенной схеме:

```
Var    <имя массива>: array [1.. кол-во_строк, 1..кол-во_столбцов] of тип_элементов_массива;
```

Для задания значений двухмерного массива можно воспользоваться процедурой с использованием датчика случайных чисел (см. лаб. раб №6).

Например:

```
Program Array_Full;
Const n=10;
Type mas = array [1..n, 1..n] of Integer;
Var m : mas;
Procedure Enter (Var tabl: mas);
Var i, j: Integer;
  Begin
     For i:= 1 to n do
        For j:= 1 to n do
          tabl[i,j]:=Random(10);
  End;
Begin
  Randomize;
  Enter (m);
End.
```

С помощью процедуры Enter осуществляется ввод двумерного массива размерностью 10×10 случайными числами из диапазона от 0 до 9. Задание диапазона значений определяется константой n=10.

Примеры решений задач

Пример программы ввода/вывода двумерного массива с использованием процедур (в одной из процедур используется процедура GotoXY, содержащаяся в модуле CRT. Поэтому необходимо при использовании этой процедуры указать имя библиотеки CRT в разделе описания библиотек **Uses**). Дополнительно программа ищет максимальный элемент в массиве.

```
Program Mas_example;
Uses Crt;
Const n=10;
Type mas = array [1..n, 1..n] of Integer; // объявляем тип данных для массива
Var m : mas;                              // объявляем переменную типа mas, т.е. массив
    k1,k2:integer;               // объявляем переменные для кол-ва строк и столбцов

Procedure Enter (Var tabl: mas; x,y: integer); // процедура ввода данных в массив, параметры -
Var i, j: Integer;                             // массив и кол-во строк и столбцов
  Begin
     For i:= 1 to x do           // цикл для строк
       For j:= 1 to y do         // цикл для столбцов
         tabl[i,j]:=Random(10);  // заполняем текущий элемент массива случайным числом из промежутка
  End;                           // от 0 до 9

Procedure List(var tabl:mas; x,y: integer);    // процедура вывода массива на экран, параметры -
   Var  i, j: integer;                         // массив и кол-во строк и столбцов
```

```
   Begin
       For i:=1 to x do
          For j:=1 to y do
          begin
               gotoxy(3*j, i+3); //вызываем процедуру gotoxy, она перемещает курсор в нужную позицию
                                 // текстового экрана, 1-й параметр – по оси OX (т.е. для столбцов),
                                 // второй – по оси OУ (т.е. для строк)
               write (tabl[i,j]); // выводим на экран очередной элемент
          end;
       writeln;      // делаем переход на новую строку
   End;

Procedure mx_elmnt(var tabl:mas; x,y: integer); // процедура поиска максимального эл-та в массиве
   Var  i, j, max: integer;
   Begin
       max:=tabl[1,1];  // делаем по умолчанию максимальным 1-й элемент массива
       For i:=1 to x do
          For j:=1 to y do
            if max<tabl[i,j] then max:=tabl[i,j];  // ищем максимальный
       writeln('Максимальный элемент в массиве: ', max); // выводим его
   end;

Begin
  Randomize;
  write('Введите кол-во строк массива (<10): ');
  readln(k1);
  write('Введите кол-во столбцов массива (<10): ');
  readln(k2);
  Enter(m,k1,k2);   // вызываем нашу процедуру с нужными параметрами для ввода значений в массив
  list(m,k1,k2);    // вызываем нашу процедуру с нужными параметрами для вывода массива на экран
  mx_elmnt(m,k1,k2);// вызываем процедуру для нахождения максимального элемента
End.
```

На экране появится примерно следующее:

```
Введите кол-во строк массива   (<10):   3
Введите кол-во столбцов массива   (<10):   4
0  6  4  5
3  6  1  5
5  8  7  6
```

Матричная алгебра

Очень часто необходимо совершить над матрицами математические действия. Рассмотрим их на примерах.

Задача: найти произведение двух матриц.

$$A=\begin{pmatrix}-1 & -2 & -4\\ -1 & -2 & -4\\ 1 & 2 & 4\end{pmatrix} \quad \text{и} \quad B=\begin{pmatrix}1 & 2 & 3\\ 2 & 4 & 6\\ 3 & 6 & 9\end{pmatrix}.$$

Решение: Вычислим произведения AB. Согласно правилу умножения матриц элемент матрицы AB, стоящий в i-ой строке и j-м столбце (c_{ij}) равен сумме произведений элементов i-й строки матрицы A на соответствующие элементы j- го столбца матрицы B. Так, например,

$$C_{23}=(-1)*3+(-2)*6+(-4)*9=-51$$

Подсчитав таким образом все элементы матрицы AB, находим:

$$AB=\begin{pmatrix}-17 & -34 & -51\\ -17 & -34 & -51\\ 17 & 34 & 51\end{pmatrix}.$$

Мы видим на этом примере, что произведение матриц зависит от порядка сомножителей.

Пример решения задачи

Составить программу нахождения произведения двух матриц A и B размером 2x3 и 3x3 соответственно.

Элементы результирующей матрицы C (размером 2x3) определяются по формуле: $c_{ij} = \sum_{k=1}^{m} a_{ik} b_{kj}; i = 1,2,...n;\ j = 1,2,...p$,

где n- число строк матрицы A; m – число столбцов матрицы A и число строк матрицы B; p – число столбцов B. В общем случае результирующая матрица C имеет n строк и p столбцов.

```
Program UM;
uses crt;
Const n=10;
Type mas = array [1..n, 1..n] of Integer;
Var    A,b,C : mas;
       K: integer;

Procedure Enter (Var tabl: mas; x,y: integer);
Var i, j: Integer;
  Begin
     For i:= 1 to x do
        For j:= 1 to y do
          tabl[i,j]:=Random(2);
  End;

Procedure List(var tabl:mas; x,y: integer);
   Var  i, j: integer;
   Begin
       For i:=1 to x do
       begin
          For j:=1 to y do
             write (tabl[i,j]:4);
          writeln;
       end;
   writeln;
   End;

procedure proizved(var a,b,c:mas; x,y,z: integer);
var i,j,k: integer;
begin
For I:=1 to x do
       For J:=1 to y do
       Begin
              C[I,J]:=0;
              For K:=1 to z do
              C[I,J]:=C[I,J]+A[I,K]*B[K,J];
      End;
end;

Begin
 enter(a,2,3);
 enter(b,3,3);
 writeln('Матрица A');
 list(a,2,3);
 writeln('Матрица B');
 list(b,3,3);

 proizved(a,b,c,2,3,3);
 writeln('Матрица C');
 list(c,2,3);
End.
```

Список задач

Часть 1

1. Даны действительные числа A1, ..., An (можно использовать массив из n элементов, заполняемый случайным образом), действительная квадратная матрица порядка n (n>=6). Получить действительную матрицу размера n×(n+1), вставив в исходную матрицу между пятым и шестым столбцами новый столбец с элементами A1, ..., An.

2. Дана целочисленная матрица размера 6x9. Получить новую матрицу, получающуюся из данной перестановкой столбцов – первого и последнего, второго с предпоследним и т.д.;
3. Даны целые числа A1, ..., A10 (можно использовать массив из 10 элементов, заполняемый случайным образом), целочисленная квадратная матрица порядка n. Заменить нулями в матрице те элементы с четной суммой индексов, для которых имеются равные среди A1, ..., A10.
4. Дана целочисленная матрица размера 6x9. Получить новую матрицу, получающуюся из данной перестановкой строк – первой и последней, второй с предпоследней и т.д.;
5. Дана действительная квадратная матрица порядка n. Преобразовать матрицу по правилу: строку с номером n сделать столбцом с номером n, столбец с номером n сделать строкой с номером n.
6. Дана квадратная матрица A[n, n]. Записать на место отрицательных элементов матрицы нули, а на место положительных – единицы.
7. В данной действительной квадратной матрице порядка n найти наибольший по модулю элемент. Получить квадратную матрицу порядка n-1 путем выбрасывания из исходной матрицы какой-нибудь строки и столбца, на пересечении которых расположен элемент с найденным значением.
8. Дана действительная квадратная матрица порядка n. Найти наибольший элемент среди стоящих на главной и побочной диагоналях и поменять его местами с элементом, стоящим на пересечении этих диагоналей.
9. Вычислить сумму и число положительных элементов матрицы A[N, N], находящихся над главной диагональю.
10. Дана матрица A размером n × m. Определить k — количество особых элементов массива A, считая его элемент особым, если он больше суммы остальных элементов его столбца.
11. Задана квадратная матрица. Поменять местами строку с максимальным элементом на главной диагонали со строкой с заданным номером m.
12. Дана матрица B[N, M]. Найти в каждой строке матрицы максимальный и минимальный элементы и поменять их местами с первым и последним элементом строки соответственно.
13. Дана целая квадратная матрица n-го порядка. Определить, является ли она магическим квадратом, т.е. такой, в которой суммы элементов во всех строках и столбцах одинаковы.
14. Элемент матрицы назовем седловой точкой, если он является наименьшим в своей строке и одновременно наибольшим в своем столбце или, наоборот, является наибольшим в своей строке и наименьшим в своем столбце. Для заданной целой матрицы размером n × m напечатать индексы всех ее седловых точек.
15. Дана матрица размером n × m. Переставляя ее строки и столбцы, добиться того, чтобы наибольший элемент (или один из них) оказался в верхнем левом углу.
16. Определить, является ли данная целая квадратная матрица n-го порядка симметричной (относительно главной диагонали).
17. Дана действительная матрица размером n × m. В каждой строке выбирается элемент с наименьшим значением, затем среди этих чисел выбирается наибольшее. Указать индексы элемента с найденным значением.
18. Найти наибольший и наименьший элементы прямоугольной матрицы и поменять их местами.
19. Дана прямоугольная матрица. Найти строку с наибольшей и наименьшей суммой элементов. Вывести найденные строки и суммы их элементов.
20. В данной действительной квадратной матрице порядка n найти сумму элементов строки, в которой расположен элемент с наименьшим значением.
21. Пусть дана действительная матрица размером n × m. Требуется преобразовать матрицу следующим образом: поэлементно вычесть последнюю строку из всех строк, кроме последней.
22. Определить наименьший элемент каждой четной строки матрицы A[M, N].
23. Определить номера строк матрицы R[M, N], хотя бы один элемент которых равен c, и элементы этих строк умножить на d.

Часть 2

1. Даны матрицы A и B размера k×m и m×l соответственно. Найти произведения AB.
2. Дана квадратная матрица порядка n. Получить матрицу A^2.
3. Даны матрицы A и B порядка n. Получить матрицу AB-BA.

Дана квадратная матрица A порядка n. Получить матрицу AB; элементы матрицы B вычисляются по формуле (4-6):

4. $b_{ij}=1/(i+j-1)$;

5. $$b_{ij} = \begin{cases} 1/(i+j-1), & \text{если } i <= j \\ 1/(i+j+1), & \text{иначе} \end{cases}$$

6. $b_{ij} = \begin{cases} 1/(i+j-1), \text{если } i < j \\ 0 \quad\quad\quad , \text{если } i = j \\ 1/(i+j+1), \text{если } i > j \end{cases}$

Даны квадратная матрица A порядка n и вектор c n элементами. Получить вектор (7-9):

7. Ab;
8. A^2b;
9. (A-E)b, где E - единичная матрица порядка n.

Дана квадратная матрица порядка n. Получить вектор Ab, где вектор b- вектор, элементы которого вычисляются по формуле (10-11):

10. $b_i=1/(i^2+2)$;
11. $\begin{cases} 1/(i2+2), & \text{если } i - \text{четное} \\ 1/i \quad\quad , & \text{иначе} \end{cases}$

12. Даны квадратная матрица A порядка n и вектор x и y c n элементами. Получить вектор A(x+y).
13. Даны квадратная матрица A, B, C порядка n. Получить матрицу (A+B)C.
14. Даны квадратные матрицы A и B порядка n. Получить матрицу A(B–E)+C, где E- единичная матрица порядка n, а элементы матрицы C вычисляются по формуле: $c_{ij}=1/(i+j)$, i,j=1, 2, ..., n.

Практическая работа № 9

Тема: «Тип данных множество»

Множество – набор однотипных элементов базового типа, каким-то образом связанных друг с другом. Базовый тип – это порядковый тип, кроме word, integer, longint.

Число элементов исходного множества не может быть больше 256, а порядковые номера элементов должны находиться в пределах от 0 до 255.

Type

<*имя множества*> = **set of** <*тип компонент*>;

Var

<*переменная*>:<*имя множества*>;

Допустимые операции с множествами:

\+ объединение;

– разность;

***** пересечение;

= , <= , >= проверка эквивалентности двух множеств;

< > проверка неэквивалентности двух множеств;

in логический оператор проверки присутствия компоненты в множестве.

Примеры решений задач

1. Ввести строку символов, состоящую из латинских букв, цифр и пробелов. Осуществить проверку правильности введенных символов (т.е. чтобы там были только цифры, латинские буквы и пробелы).

```
Program stroka;
Var
       Str: string;
       L: byte;
       Test: boolean;
Begin
       Writeln ('Введите строку');
       Readln (str);
       L:=Length (Str);
       Test:= L>0;
       While Test and (L>0) do
     {выход из цикла будет осуществлен, если переменная test станет равна false или L станет равна 0}
       Begin
            Test:=Str[L] in ['0'..'9','A'..'Z','a'..'z',' '];
{test = true, если str[L] является одним из перечисленных символов}
       {иначе test = false}
            Dec (L)    {аналог L:=L - 1}
        End;
       If Test then
             WriteLn ('Правильная строка')
       Else
             WriteLn ('Неправильная строка');
End.
```

Другое решение данной задачи

```
Program stroka;
Var
       Str: string;
       L,i: byte;
       Test: boolean;
Begin
   Writeln ('Введите строку');
   Readln(str);
    L:=Length (Str);
  for i:=1 to L do
  Begin
     test:=Str[i] in ['0'..'9','A'..'Z','a'..'z',' '];
```

```
    if test=false then break;
   End;
   If Test then  WriteLn ('Правильная строка')
      Else   WriteLn ('Неправильная строка');
End.
```

Еще одно решение этой задачи

```
Program stroka;
Var
      Str: string;
      L,i,k: byte;
Begin
  Writeln ('Введите строку');
   Readln(str);
   L:=Length (Str);
   for i:=1 to L do
   Begin
     if Str[i] in ['0'..'9','A'..'Z','a'..'z',' '] then k:=k+1;
    End;
    If k=L then
          WriteLn ('Правильная строка')
               Else
         WriteLn ('Неправильная строка');
End.
```

2. *Заполнить множество A путем ввода n значений:*

```
Program mnozh;
var   A: set of 1..200;
      j,x,n: byte;
begin
     write('Введите кол-во эл-ов в мн-ве: ');
     readln(n);
     A:=[];         // задаем пустое мн-во
     for j := 1 to n do
     begin
          write('Введите эл-нт мн-ва: ');
          readln(x);
          A:=A+[x]         // добавляем в мн-во введенный элемент
     end;
     write('Вот ваше мн-во: ');
     for x := 1 to 200 do
         if x in A then write(x:3);   //  если x в мн-ве, то напечатать его
     writeln;
end.
```

Список задач (*решить с применением множеств*)

1. Дано множество, элементами которого являются буквы от a до f и от x до z. Требуется ввести с клавиатуры некую последовательность символов, и выяснить, какие из них входят в заданное множество.
2. Имеется множество, содержащее натуральные числа из некоторого диапазона. Сформировать два множества, первое из которых содержит все простые числа из данного множества, а второе — все составные.
3. Дан текст из строчных латинских букв, за которым следует точка. Определить, каких букв – гласных (a, e, i, o, u) или согласных – больше в этом тексте.
4. Напечатать в возрастающем порядке все цифры, не входящие в десятичную запись натурального числа n.

   Дан текст из строчных латинских букв, за которым следует точка. Напечатать (5-7):
5. все буквы, входящие в текст не менее двух раз;
6. все буквы, входящие в текст по одному разу;
7. первые вхождения букв в текст, сохраняя их исходный взаимный порядок.

8. Дана последовательность целых чисел. Определить, является ли эта последовательность перестановкой заданного отрезка элементов натурального ряда.
9. Подсчитать количество чётных цифр в исходной символьной строке и распечатать все, кроме пробелов, знаков операций и знаков препинания.

10. Сформировать множество, в которое входят только латинские буквы, встретившиеся во входной строке, и множество знаков препинания из входной строки.
11. Сформировать множество, в которое входят только цифры, встретившиеся во входной строке.
12. Сформировать множество, в которое входят только большие латинские буквы.
13. Дан текст на русском языке. Напечатать в алфавитном порядке все гласные буквы, которые входят в каждое слово.
14. Дан текст на русском языке. Напечатать в алфавитном порядке все звонкие согласные буквы, которые входят более чем в одно слово.
15. Подсчитать количество различных цифр в десятичной записи натурального числа.

Практическая работа № 10

Тема: «Тип данных запись»

Тип запись включает ряд компонент, называемых полями, которые могут быть разных типов. При задании типа-записи после зарезервированного слова **record** следует перечислять все поля типа записи с указанием через двоеточие их типов и завершить задание типа словом **end.** Поля отделяются друг от друга точкой с запятой. Количество полей записи может быть любым.

Тип данных запись описывается следующим образом:

Type

Record

<имя поля 1>:<тип поля 1>;

<имя поля 2>:<тип поля 2>;

. . .

<имя поля n>:<тип поля n>;

End;

Если тип нескольких полей совпадает, то имена полей могут быть просто перечислены. После объявления в программе переменной типа «запись» к каждому ее полю можно обратиться, указав сначала идентификатор переменной-записи, а затем через точку – имя поля. Поле записи может иметь практически любой тип. Доступ к вложенным элементам таких структур осуществляется по тем же правила, как и обычно.

Переменная типа «запись» может участвовать только в операциях присваивания. Но поле записи может принимать участие во всех операциях, применимых к типу этого поля.

Присвоение значений переменным типа запись производится одним из двух способов:

1) *<имя переменной>.<имя поля 1>:=<выражение>*;

<имя переменной>.<имя поля 2>:=<выражение>;

. . . . . .

<имя переменной>.<имя поля n>:=<выражение>;

2) для облегчения работы с полями записей вводится оператор присоединения.

With *<имя переменной>* **do**

begin

<имя поля 1>:=<выражение>

<имя поля 2>:=<выражение>

. . . . .

<имя поля n>:=<выражение>

end;

Примеры решений задач

1. Для каждого студента указаны фамилия и оценки в баллах по пяти дисциплинам. Требуется вычислить средний балл каждого студента и максимальный средний балл.

```
Program BAL;
Uses crt;
      Type zap=Record            // создаем zap типа Record
                  Fam: string;        // поле с фамилией
                  B1, B2, B3, B4, B5: 2..5;  // поля с оценками (тип множество от 2 до 5)
                  SB: real;                  // поле для хранения среднего бала
            End;
Var
      Tbl: array[1..10] of zap;  // создаем массив типа zap
      I,n: integer;
      max: real;                 // для хранения максимального значения
      max_name: string;
Begin
  write('Введите кол-во записей: ');
  readln(n);
  For I:=1 to n do
  Begin
    Write('Введите данные через Enter (фамилия и 5 оценок)');
// считываем данные для каждого поля каждого элемента массива
    Readln (Tbl[I].Fam, Tbl[I].B1, Tbl[I].B2, Tbl[I].B3, Tbl[I].B4,Tbl[I].B5);
  end;
  max:=0;     // зануляем текущий максимум
```

```
  For I:=1 to n do
  begin
    Tbl[I].SB:=(Tbl[I].B1+Tbl[I].B2+Tbl[I].B3+Tbl[I].B4+Tbl[I].B5)/5; // считаем ср. балл
    Writeln('Средняя оценка у ', Tbl[I].FAM,' равна ',Tbl[I].SB); //выводим ср. балл
    if Tbl[I].SB>max then // если текущий эл-нт массива больше максимума
    begin
        max:=Tbl[I].SB;  // записываем текущий максимум
        max_name:=Tbl[I].FAM; //записываем фамилию студента с текущим максимальным ср. баллом
    end;
  end;
  writeln('Максимальный средний балл у студента с фамилией ',max_name, ', он равен ',max);

End.
```

Список задач

1. Сформировать переменную типа запись, в которой расположены данные о каждом отдельном ученике в следующем порядке: имя (15 символов), фамилия (15 символов), год обучения (целое число), буква (символ). Требуется перенести эти данные в другую переменную, выводя первую букву имени и фамилию ученика:
   И. Петров
   П. Иванов
   и т. д.
2. Переменная содержит сведения об учениках некоторой школы (см. задачу 1).
   а) Собрать сведения об учениках девятых классов школы,
   б) Выяснить, на сколько человек в восьмых классах больше, чем в девятых.

   Багаж пассажира характеризуется количеством вещей и общим весом вещей. Сформировать переменную Bagaj, содержащую сведения о багаже нескольких пассажиров. Сведения о багаже каждого пассажира представляют собой запись с двумя полями: одно поле целого типа (количество вещей) и одно – действительное (вес в килограммах). Задание (3-7):
3. Найти багаж, средний вес одной вещи в котором отличается не более, чем на 0,3 кг от общего среднего веса одной вещи.
4. Найти число пассажиров, имеющих более двух вещей и число пассажиров, количество вещей которых превосходит среднее число вещей.
5. Выяснить, имеется ли пассажир, багаж которого состоит из одной вещи весом менее 30 кг.
6. Требуется удалить из данной переменной Bagaj сведения о багаже, общий вес вещей в котором меньше, чем 10 кг. Использовать вспомогательную переменную F.
7. Переписать сведения о багаже из переменной Bagaj в переменную Bag. В переменной Bag сведения о багаже каждого пассажира представляются массивом из двух целых чисел - числа вещей и общего веса вещей, выраженного в граммах. Составить также программу обратного преобразования: переписи сведений о багаже из переменной Bag в переменную Bagaj.

8. Сформирована переменная bibl, содержащий сведения о книгах. Сведения о каждой из книг – это фамилия автора, название и год издания.
   а) Найти названия книг данного автора, изданных с 1960 года
   б) Определить имеется ли книга с названием "Информатика". Если да, то сообщить фамилию автора и год издания. Если таких книг несколько, то сообщить сведения обо всех этих книгах.
9. Дана переменная T, которая содержит номера телефонов сотрудников учреждения: Указывается фамилия сотрудника, его инициалы и номер телефона. Найти номер телефона сотрудника по его фамилии и инициалам.
10. Сформирована переменная типа запись, содержащая различные даты. Каждая дата - это число, месяц и год. Найти:
    а) год с наименьшим номером.
    б) все весенние даты.
    в) самую позднюю дату.
11. Сформировать переменную Tovar, содержащую сведения об экспортируемых товарах: Указывается наименование товара, страна импортирующая товар, и объем поставляемой партии в штуках. Составить список стран, в которые экспортируется данный товар, и общий объем его экспорта.
12. Сформирована переменная Assortim, содержащая сведения об игрушках: указано название игрушки, ее стоимость в рублях, и возрастные границы. Получить следующие сведения:
    а) название игрушек, цена которых не превышает 4 руб., и которые подходят детям 5 лет.
    б) цену самого дорогого конструктора.
13. Сформирована переменная Assortim, содержащая сведения об игрушках: указано название игрушки, ее стоимость в рублях, и возрастные границы. Получить следующие сведения:
    а) Название наиболее дорогих игрушек (цена которых отличается не более чем на 1 руб. от самой дорогой).
    б) название игрушек, которые подходят как детям 4 лет, так и детям 10 лет.

Практическая работа № 11

Тема: «Тип данных файл»

У понятия файл есть две стороны. С одной стороны, файл – это область памяти на внешнем носителе, в котором хранится некоторая информация. Файл в таком понимании называют физическим файлом, т.е. существующим физически на некотором материальном носителе информации. С другой стороны, файл - это одна из структур данных, используемых в программировании. Файл в таком понимании называют логическим файлом, т.е. существующим в нашем логическом представлении при написании программы.

Структура физического файла представляет собой простую последовательность байт памяти носителя информации. Структура логического файла – это способ восприятия файла в программе.

Любой файл имеет следующие характеристики-требования:

– у него есть имя (набор из восьми, допустимых для имени файла, символов плюс расширение, указываемое после точки в имени файла состоящее из трех символов);
– он должен содержать данные одного типа (любой тип Паскаля, кроме типа Файл, то есть не существует типа «Файл файлов»);

Длина создаваемого файла никак не регламентируется при создании файла и ограничивается только емкостью носителя информации.

Работа с файлами в Паскале осуществляется следующим образом: сначала объявляется переменная файлового типа, с указанием свойств переменной (то есть типом содержимого), затем данная файловая переменная связывается («ассигнуется») с именованным дисковым пространством (то есть непосредственно с конкретным файлом, содержащим или, который будет содержать данные того же типа, что и связываемая переменная-файл) или логическим устройством)

Переменная файлового типа может быть объявлена одной из следующих строк:

<имя> = **file of** <тип>;
<имя> = **text;**
<имя> = **file;**

где <имя> – имя переменной-файла;
file of – зарезервированные слова (файл, из);
text – имя стандартного типа текстовых файлов;
<тип> – имя любого стандартного типа Паскаля, кроме типа файл.

Например файл, содержащий список учеников и их возраст:

```
type
  pupil = record
    surname : string;
    name : string;
    age : word
  end;
Var
  journal : file of pupil;
```

В зависимости от способа объявления можно выделить три вида файлов:

- типизированные файлы (задаются предложением **file of**);
- текстовые файлы (определяются типом **text**);
- нетипизированные файлы (определяются типом **file).**

Файловые переменные имеют специфическое назначение. Такие операции, как присвоение значения, сравнение и т.д. над переменными типа файл осуществлять нельзя.

Текстовые файлы – это файлы, содержащие символы, разделенные на строки. Причем в конце каждой строки стоит признак конца строки. Текстовые файлы не имеют прямого доступа. При чтении и записи числа преобразуются автоматически. К ним применима процедура **Append**(<имя переменной текстового файла>). Она открывает текущий файл, с которым связана данная переменная, текущий указатель помещает в конец для добавления новой информации.

Нетипизированные файлы предназначены для низкоуровневой работы с файлами. С их помощью можно обратиться к файлу любого типа и логической структуры. За одно обращение считывается/записывается число байт, приблизительно равное величине буфера ввода/вывода. В качестве буфера может выступать любая переменная. Для записи и чтения используются процедуры BlockRead, BlockWrite.

Процедуры и функции обработки файловых переменных

Assign (<*имя файловой переменной*>,'<*путь и имя файла на диске*>') – связь переменной файлового типа с конкретным внешним файлом.

Reset (f) – процедура открытия существующего файла и подготовка к чтению файла, связанного с файловой переменной f. Указатель текущей позиции файла устанавливается в его начало.

Rewrite (f) – процедура создания нового физического файла и подготовка к записи файла, связанного с файловой переменной f**.** Если такой файл существует, то он удаляется, и на этом месте создается новый пустой файл. Указатель текущей позиции файла устанавливается в его начало.

Readln (f) – пропуск строки файла до начала следующей;

Writeln (f) – запись признака конца строки и переход на следующую;

Read (f, x) – процедура чтения компоненты файла. Данные выводятся из файла.

Write (f, x) – процедура записи значения переменной в файл, который хранится на диске. Указатель перемещается на следующий элемент. Если указатель текущей позиции файла находится за последним элементом, т.е. в конце файла, то файл расширяется.

Eof – признак конца файла – логическая функция для определения, достигнут ли конец файла.

Close (f) – процедура закрытия файла (**!!! <u>именно в этот момент происходит реальная запись в файл</u>**).

Примеры решений задач

1. Прочитать из текстового файла A все записанные в него целые числа (их можно ввести в него через Блокнот и сохранить файл), преобразовать их в вещественные и вывести в текстовый файл B по 4 числа в строку.

```
Program File1;
Var F1,F2: text; // объявляем 2 переменные типа текстовых файлов
      X: real;
      s: string;
      I:integer;
Begin
  Assign (F1,'A.txt'); //связываем переменную F1 с файлом A.txt (находится там же, где и программа)
  Reset(F1);        // открываем его для чтения (т.е. он уже должен быть создан ранее)
  Assign (F2,'B.txt'); //связываем F2 с файлом B.txt (будет находиться там же, где и программа)
  Rewrite (F2);     // открываем его для записи
  writeln('содержимое файла A.txt:');
  Repeat                                  // цикл
      For I:=1 to 4 do     // цикл от 1 до 4, т.к. нужно в файле B формировать по 4 числа в строке
         If not seekeof(F1) then    // если еще не конец файла, связанного с F1
         Begin                    // то делаем
             Read (F1,x);    //считываем из файла, связанного с F1, 1 компонент и помещаем его в
                                  //переменную x
             write(x:-5:0);       // выводим на экран, что считали из файла в текущий момент,
                           //-5 означает, что в поле из 5 позиций с выравниванием по левому краю
             Write (F2,x:-5:2)     // выводим текущее считанное значение в файл, связанный с F2
         End;
      Writeln(F2);        // делаем в файле переход на новую строку (чтобы следующее значение
                                  //выводилось с новой строки)
  Until seekeof (F1);             // будет повторяться, пока файл не закончится
  writeln;
  Close (F1);       // закрываем файл, связанный с F1
  Close (F2);       // закрываем файл, связанный с F2
  Reset(F2);        // открываем файл, связанный с F2, для чтения
  writeln('содержимое файла B.txt:');
  Repeat                                  //цикл
      Readln(F2,s); // считываем по целой строке из B и записываем в строковую переменную s
      writeln(s);
  Until seekeof (F2);                     // пока не конец файла, связанного с F2
  Close (F2);

End.
```

Пример работы программы:

содержимое файла A.txt:

1 2 3 4 5

содержимое файла B.txt:

1.00 2.00 3.00 4.00

5.00

2. Работа с числовыми файлами. Записать в файл числа (кол-во вводит пользователь, сами числа задаются генератором случайных чисел). Записать в другой файл числа, кратные 5 или 4.

```
program file2;
var f1,f2: file of integer; //файловые переменные типа integer (в файле будут только целые числа)
    i,n,x: integer;

begin
  assign(f1,'a.dat'); // связываем файловую переменную f1 с файлом a.dat
  rewrite(f1); // открываем его на запись или перезапись (если он уже существует)
  assign(f2,'b.dat'); // связываем файловую переменную f2 с файлом b.dat
  rewrite(f2); // открываем его на запись или перезапись (если он уже существует)
  write('Введите кол-во чисел: ');
  readln(n);
  for i:=1 to n do  // будем в цикле записывать данные в файл
  begin
    x:=random(30);
    write(f1,x);    // текущее значение x записывается в файл
  end;
  close(f1);  // закрываем файл, связанный с f1 (это нужно, чтобы все записанные в него данные
              // сохранились)
  reset(f1); // и открываем его для чтения
  writeln('Вот числа из первого файла:');
  for i:=1 to n do
  begin
    read(f1,x);     // считываем 1 значение из файла, связанного с f1, и записываем его в x
    write(x:5);     // выводим x на экран
    if (x mod 5 =0) or (x mod 4 =0) then write(f2,x);  // если считанный элемент подходит по
                    //условию, то записываем его в файл, связанный с f2
  end;
  close(f1); // закрываем файл, связанный с f1
  close(f2); // закрываем файл, связанный с f2
  reset(f2); //  открываем файл, связанный с f2, для чтения
  writeln;
  writeln('Вот числа из второго файла:');
  while (not eof(f2)) do  // т.к. мы не знаем, сколько было записано элементов в файл,
                    //связанный с f2, то используем цикл с условием, который будет работать
                    // пока не будет считан признак конца файла (while (not eof(f2)) do)
  begin
    read(f2,x);  // считываем 1 элемент из файла, связанного с f2, и записываем его в x
    write(x:5);  // выводим x на экран
  end;
  close(f2); // закрываем файл, связанный с f2
  end.
```

Пример работы программы:

Введите кол-во чисел: 20

Вот числа из первого файла:

1 20 13 1 12 17 7 8 7 21 10 5 19 9 4 17 13 3 6 11

Вот числа из второго файла:

20 12 8 10 5 4

Список задач

1. Текстовые файлы.

1. Дан файл, содержащий текст на русском языке. Составить список всех слов, встречающихся в этом тексте.
2. Дан файл, содержащий текст, набранный заглавными русскими буквами. Провести частотный анализ текста, т. е. указать (в процентах), сколько раз встречается та или иная буква.
3. Дан символьный файл f. В файле не менее двух компонент. Определить являются ли два первых символа файла цифрами. Если да, то установить, является ли число, образованное этими цифрами, чётным.
4. Дан символьный файл f. Записать в файл g компоненты файла f в обратном порядке.
5. Дан файл, содержащий текст, записанный строчными русскими буквами. Получить в другом файле тот же текст, записанный заглавными буквами.
6. Дан файл, содержащий произвольный текст. Выяснить, чего в нем больше: русских букв или цифр.
7. Дан файл, содержащий текст на русском языке. Выяснить, входит ли данное слово в указанный текст, и если да, то сколько раз.
8. Дан файл, содержащий текст на русском языке. Выбрать из него те символы, которые встречаются в нем только один раз, в том порядке, в котором они встречаются в тексте.
9. Даны файл, содержащий текст на русском языке, и некоторые буквы. Найти слово, содержащее наибольшее количество указанных букв.

10. Даны файл, содержащий текст на русском языке, и некоторая буква. Подсчитать, сколько слов начинается с указанной буквы.
11. Дан файл, содержащий текст, включающий русские и английские слова. Подсчитать, каких букв в тексте больше – русских или латинских.
12. Дан текстовый файл. Удалить из него все лишние пробелы, оставив между словами не более одного пробела. Результат поместить в новый файл.
13. Дан файл, содержащий текст на русском языке. Подсчитать количество слов, начинающихся и заканчивающихся на одну и ту же букву.

2. Числовые файлы.

1. Дан файл f, компоненты которого являются действительными числами. Найти:
   а) сумму компонент файла;
   б) произведение компонент файла;
2. Дан файл f, компоненты которого являются действительными числами. Найти из значений компонент:
   б) наименьшее из значений компонент с чётными номерами;
   в) наибольшее из значений компонент с нечётными номерами;
3. Дан файл f, компоненты которого являются целыми числами. Найти:
   а) количество чётных чисел среди компонент;
   б) сумму нечётных чисел;
4. Дан файл f, компоненты которого являются целыми числами. Получить в файле g все компоненты файла f:
   а) являющиеся чётными числами;
   б) делящиеся на 3 и не делящиеся на 7;
5. Даны файлы f и g, заполненные случайными числами. Записать в файл h сначала компоненты файла f, затем компоненты файла g с сохранением порядка.
6. Заполнить файл f целыми числами, полученными с помощью генератора случайных чисел. Переписать в файл g те компоненты файла f, которые являются четными.
7. Заполнить файл последовательного доступа f целыми числами, полученными с помощью генератора случайных чисел. Получить в файле g все компоненты файла f, которые делятся на m и не делятся на n.
8. Заполнить файл f целыми числами, полученными с помощью генератора случайных чисел. Из файла f получить файл g, исключив повторные вхождения чисел. Вывести файл g на экран.
9. Записать в файл f N произвольных натуральных чисел. Переписать в другой файл те элементы, которые кратны К. Вывести полученный файл на экран.
10. Заполнить файл f N действительными случайными числами. Найти сумму минимального и максимального элементов этого файла.
11. Записать в файл f N случайных действительных чисел. Найти разность первого и последнего компонентов файла.
12. Заполнить файл f случайными целыми числами. Переписать в файл g те компоненты файла f, которые являются нечетными.
13. Заполнить файл f целыми числами, полученными с помощью генератора случайных чисел. Переписать их в файл g в обратном порядке.

Практическая работа №12.

Тема: Работа с графикой в PascalABC.

Для работы с графикой в среде программирования PascalABC и PascalABC.NET нужно подключить модуль GraphABC.

Процедуры модуля GraphABC

procedure SetPixel(x,y,color: integer);

Закрашивает один пиксел с координатами (x,y) цветом color.

procedure MoveTo(x,y: integer);

Передвигает невидимое перо к точке с координатами (x,y); эта функция работает в паре с функцией LineTo(x,y).

procedure LineTo(x,y: integer);

Рисует отрезок от текущего положения пера до точки (x,y); координаты пера при этом также становятся равными (x,y).

procedure Line(x1,y1,x2,y2: integer);

Рисует отрезок с началом в точке (x1,y1) и концом в точке (x2,y2).

procedure Circle(x,y,r: integer);

Рисует окружность с центром в точке (x,y) и радиусом r.

procedure Ellipse(x1,y1,x2,y2: integer);

Рисует эллипс, заданный своим описанным прямоугольником с координатами противоположных вершин (x1,y1) и (x2,y2).

procedure Rectangle(x1,y1,x2,y2: integer);

Рисует прямоугольник, заданный координатами противоположных вершин (x1,y1) и (x2,y2).

procedure RoundRect(x1,y1,x2,y2,w,h: integer);

Рисует прямоугольник со скругленными краями; (x1,y1) и (x2,y2) задают пару противоположных вершин, а w и h – ширину и высоту эллипса, используемого для скругления краев.

procedure Arc(x,y,r,a1,a2: integer);

Рисует дугу окружности с центром в точке (x,y) и радиусом r, заключенной между двумя лучами, образующими углы a1 и a2 с осью OX (a1 и a2 – вещественные, задаются в градусах и отсчитываются против часовой стрелки).

procedure Pie(x,y,r,a1,a2: integer);

Рисует сектор окружности, ограниченный дугой (параметры процедуры имеют тот же смысл, что и в процедуре Arc).

procedure Chord(x,y,r,a1,a2: integer);

Рисует фигуру, ограниченную дугой окружности и отрезком, соединяющим ее концы (параметры процедуры имеют тот же смысл, что и в процедуре Arc).

procedure TextOut(x,y: integer; s: **string**);

Выводит строку s в позицию (x,y) (точка (x,y) задает верхний левый угол прямоугольника, который будет содержать текст из строки s).

procedure FloodFill(x,y,color: integer);

Заливает область одного цвета цветом color, начиная с точки (x,y).

procedure FillRect(x1,y1,x2,y2: integer);

Заливает прямоугольник, заданный координатами противоположных вершин (x1,y1) и (x2,y2), цветом текущей кисти.

procedure Polygon(points: **array of** Point);

Рисует заполненный многоугольник, координаты вершин которого заданы в массиве points.

procedure Polyline(points: **array of** Point);

Рисует ломаную по точкам, координаты которых заданы в массиве points.

function PenX: integer;

function PenY: integer;

Возвращают текущие координаты пера.

procedure SetPenColor(color: integer);

Устанавливает цвет пера, задаваемый параметром color.

function PenColor: integer;

Возвращает текущий цвет пера.

procedure SetPenWidth(w: integer);

Устанавливает ширину пера, равную w пикселам.

function PenWidth: integer;

Возвращает текущую ширину пера.

procedure SetPenStyle(ps: integer);

Устанавливает стиль пера, задаваемый параметром ps.

function PenStyle: integer;

Возвращает текущий стиль пера.

Стили пера задаются следующими именованными константами:

psSolid psClear psDash psDot psDashDot psDashDotDot

procedure SetPenMode(m: integer);

Устанавливает режим пера, задаваемый параметром m.

procedure SetBrushColor(color: integer);

Устанавливает цвет кисти, задаваемый параметром color.

function BrushColor: integer;

Возвращает текущий цвет кисти.

procedure SetBrushStyle(bs: integer);

Устанавливает стиль кисти, задаваемый параметром bs.

function BrushStyle: integer;

Возвращает текущий стиль кисти.

Стили кисти задаются следующими именованными константами:

bsSolid bsCross bsClear bsDiagCross bsHorizontal bsBDiagonal bsVertical bsFDiagonal

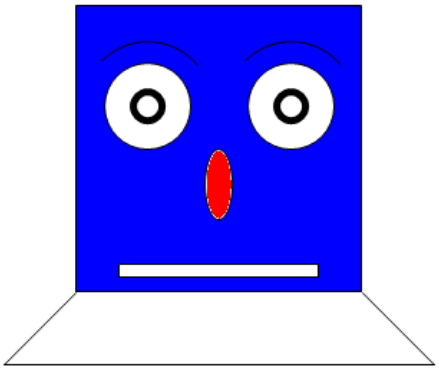
procedure ClearWindow;

Очищает графическое окно белым цветом.

procedure ClearWindow(c: ColorType);

Очищает графическое окно цветом c.

Пример. Составим программу, рисующую голову робота ——————> Рисунок содержит два прямоугольника, 4 окружности, две дуги, эллипс, три прямые линии. Заранее определяются все координаты и размеры элементов рисунка.

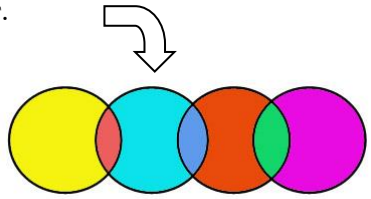


```
uses graphABC;
begin
 Rectangle(100,100,300,300);      {голова}
 floodfill(105,120,clBlue);  // закрашиваем голову синим
 Circle(150,170,30);        {левый глаз}
 Circle (250,170,30);       {правый глаз}
 Arc (150,170,45,135,40);  {левая бровь}
 Arc (250,170,45,135,40);  {правая бровь}
 Ellipse(190,200,210,250); {нос}
 floodfill(195,220,clRed); // закрашиваем нос красным
 Rectangle (130,280,270,290);     {рот}
 MoveTo (100,300);  {установка вниз влево}
 LineTo(50,350);    {три}
 LineTo(350,350);   {линии}
 LineTo(300,300);   {шеи}
 SetPenWidth(5);    // устанавливаем толщину линии в 5 пикселей
 Circle(150,170,10);{левый зрачок}
 Circle(250,170,10);{правый зрачок}
end.
```

Задачи

1. Нарисуйте три окружности с заключёнными в них треугольником, квадратом и звездой. Все фигуры должны быть разного цвета, разного типа заливки и типа линий.
2. Нарисуйте радугу.
3. Нарисуйте олимпийский флаг.
4. Нарисуйте контур прямоугольника, в котором нарисуйте линиями разного цвета и стиля своё имя.
5. Нарисуйте контур прямоугольника, в котором нарисуйте линиями разного цвета и стиля свою фамилию.
6. Нарисуйте сетку на экране линиями разного цвета и стиля.
7. Нарисуйте разноцветную мишень.
8. Нарисуйте 3 круговые диаграммы, состоящие из 10 заполненных секторов, используя различные орнаменты и цвета заполнения.
9. Нарисуйте шахматное поле.
10. Нарисовать на экране 10 заполненных звёзд разного цвета и стиля заполнения.

11. Нарисовать на экране 11 заполненных ромбов разного цвета и стиля заполнения.
12. Нарисовать на экране 12 заполненных параллелограммов разного цвета и стиля заполнения.
13. Нарисуйте 10 окружностей разного цвета, пересекающихся в 2-х точках. Цвет областей пересечения должен отличаться от цветов окружностей, которые эту область образуют.



Практическая работа №13.

Тема: Создание модулей в языке Pascal.

Структура модуля

Структура модуля аналогична структуре программы, однако есть несколько существенных различий.

```
unit <идентификатор>;
interface
uses <список модулей>;       { Необязательный }
  { глобальные описания }
implementation
  uses <список_модулей>;     { Необязательный }
  { локальные описания }
  { реализация процедур и функций }
begin
  { код инициализации }
end.
```

Заголовок модуля начинается зарезервированным словом unit, за которым следует имя модуля (идентификатор). Следующим элементом в модуле является ключевое слово interface. Оно обозначает начало интерфейсной секции модуля – части, доступной всем другим модулям или программам, в которых он используется.

Программный модуль может использовать другие модули, для этого они определяются в операторе uses. Оператор uses (если он имеет место) может содержаться в двух местах. Во-первых он может следовать сразу после ключевого слова interface. В этом случае любые константы и типы данных, описанные в интерфейсной секции этих модулей, могут использоваться в любом описании в интерфейсной части данного модуля.

Во-вторых, он может следовать немедленно за ключевым словом implementation. В этом случае все описания из этих модулей могут использоваться только в секции реализации.

Интерфейсная секция

Интерфейсная часть - "общедоступная" часть в модуле - начинается зарезервированным словом interface, следует сразу после заголовка модуля и заканчивается перед зарезервированным словом implementation. Интерфейс определяет, что является "видимым" (доступным) для любой программы (или модуля), использующей данный модуль.

В интерфейсной части (секции) модуля можно определять константы, типы данных, переменные, процедуры и функции.

Процедуры и функции, видимые для любой программы, использующей данный модуль, описываются в секции интерфейса, однако их действительные тела - реализации - находятся в секции реализации. В интерфейсной части перечисляются все заголовки процедуры и функции. Секция реализации содержит программную логику процедур и функций.

Секция реализации

Секция реализации – "приватная" часть - начинается зарезервированным словом implementation. Все, что описано в секции интерфейса, является видимым в секции реализации: константы, типы, переменные, процедуры и функции. Кроме того, в секции реализации могут быть свои дополнительные описания, которые не являются видимыми для программ, использующих этот модуль. Программа не знает об их существовании и не может ссылаться на них или обращаться к ним. Однако, эти скрытые элементы могут использоваться (и, как правило, используются) "видимыми" процедурами и функциями, то есть теми подпрограммами, чьи заголовки указаны в секции интерфейса.

Оператор uses может содержаться в секции реализации (implementation) и должен непосредственно следовать за ключевым словом implementation.

Обычные процедуры и функции, описанные в интерфейсной секции, должны повторно указываться в секции реализации.

Если вы хотите написать модуль, содержащий некоторые полезные подпрограммы, и использовать эти подпрограммы в своих программах, напишите модуль и сохраните его под именем, заданным в заголовке модуля. В исходном файле может содержаться только один модуль.

Компиляция модуля

В среде программирования PascalABC.NET компиляция модуля осуществляется командой Программа - Компилировать (CTRL+F9). При этом создается файл с расширением pcu, который должен располагаться либо в стандартном каталоге модулей, либо в одной папке с использующей его программой.

Если вы поместите свой модуль в заданный каталог модулей, то сможете ссылаться на этот модуль, даже если он не находится в текущем каталоге или в библиотеках исполняющей системы.

Пример

Теперь напишем небольшой модуль. Назовем его IntLib и вставим в него две простые подпрограммы для целых чисел – процедуру и функцию:

```
unit IntLib;
interface  {объявляем процедуры и функции, которые будут реализованы позже}
   procedure ISwap(var I,J : integer); {меняет значения переменных: I на значение J, J на значение I}
   function IMax(I,J : integer) : integer; {находит максимум из двух переменных}
implementation  {расписываем процедуры и функции, объявленные ранее}
procedure ISwap;  {меняет значения двух переменных}
var   Temp : integer;
begin
    Temp := I; I := J; J := Temp
end;  { конец процедуры ISwap }

function IMax; {находит максимум}
begin
  if I > J then IMax := I else IMax := J
end;  { конец функции IMax }
end.  { конец модуля IntLib }
```

Наберите этот модуль, сохраните его в файл INTLIB.PAS, а затем скомпилируйте (ctrl+F9). В результате получим код модуля в файле INTLIB.pcu.

Перешлем его в каталог модулей (если такой имеется), или оставив в том же каталоге, где находится следующая программа, которая использует модуль IntLib:

```
program IntTest;
uses IntLib;
var
  A,B  : integer;
begin
  Write('Введите два целочисленных значения: ');
  Readln(A,B);
  ISwap(A,B);
  Writeln('A = ',A,' B = ',B);
  Writeln('Максимальное значение равно ',IMax(A,B));
end.  { конец программы IntTest }
```

Задания:

1. Напишите программу, использующую модуль, в котором описаны 2 функции: первая возвращает сумму двух чисел, вторая их разницу.
2. Напишите программу, использующую модуль, в котором описаны 2 процедуры: первая возвращает произведение двух чисел, вторая их частное.
3. Напишите программу, использующую модуль, в котором описаны 2 функции: первая находит среднеарифметическое 2-х чисел, вторая – среднегеометрическое.
4. Напишите программу, использующую модуль, в котором описаны 2 функции: первая находит наибольший общий делитель двух чисел, вторая – наименьшее общее кратное двух чисел.
5. Напишите программу, использующую модуль, в котором описаны 2 процедуры: первая возвращает сумму первых n членов арифметической прогрессии для введенных значений a1, n d, вторая процедура находит сумму первых n членов геометрической прогрессии для введенных значений a1, n, d.
6. Напишите программу, использующую модуль, в котором описаны 2 функции: первая находит максимальное из 3-х чисел, вторая – минимальное из 3-х чисел.
7. Напишите программу, использующую модуль, в котором описаны 2 процедуры: первая вычисляет факториал введенного числа, вторая – сумму всех чисел до введенного числа включительно.
8. Напишите программу, использующую модуль, в котором описаны 2 процедуры: первая находит гипотенузу прямоугольного треугольника по двум введенным катетам, вторая находит площадь прямоугольного треугольника.
9. Напишите программу, использующую модуль, в котором описаны 2 функции: первая по веденной стороне куба находит его объем, вторая по стороне куба находит площадь его поверхности.
10. Напишите программу, использующую модуль, в котором описаны 2 процедуры: первая подсчитывает в строке гласные, вторая – согласные.
11. Напишите программу, использующую модуль, в котором описаны 2 функции: первая выясняет, делится ли введенное число на 3 без остатка, вторая находит сумму цифр числа.
12. Напишите программу, использующую модуль, в котором описаны 2 функции: первая выясняет, делится ли введенное число на 7 без остатка, вторая находит произведение цифр числа.
13. Напишите программу, использующую модуль, в котором описаны 2 процедуры: первая находит периметр треугольника по введенным сторонам, вторая находит площадь треугольника по введенным сторонам.
14. Напишите программу, использующую модуль, в котором описаны 2 функции: первая находит периметр прямоугольника по введенным сторонам, вторая находит площадь прямоугольника по введенным сторонам.

Практическая работа №14.

Тема: Создание линейных списков в языке Pascal.

Цель: получить практические навыки работы с динамическими переменными и динамическими структурами данных.

Общие сведения

Для работы с динамическими структурами данных используются указатели. Указатели представляют собой специальный тип данных. Они принимают значения, равные адресам размещения в оперативной памяти соответствующих динамических переменных.

Списком называется структура данных, каждый элемент которой посредством указателя связывается со следующим элементом. На самый первый элемент (голову списка) имеется отдельный указатель.

Из определения следует, что каждый элемент списка содержит поле данных (оно может иметь сложную структуру) и поле ссылки на следующий элемент. После ссылки последнего элемента должно содержать пустой указатель (nil).

Число элементов связанного списка может расти или уменьшаться в зависимости от того, сколько данных мы хотим хранить в нем. Чтобы добавить новый элемент в список, необходимо:

1. Получить память для него;
2. Поместить туда информацию;
3. Добавить элемент в конец списка (или начало).

Элемент списка состоит из разнотипных частей (хранимая информация и указатель), и его естественно представить записью. Перед описанием самой записи описывают указатель на нее:

```
Type { описание списка из целых чисел }
  PList = ^TList;
  TList = record
     Inf : Integer;
     Next : PList;
  end;
```

Примеры

Создание списка.

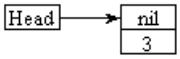
Задача. Сформировать список, содержащий целые числа 3, 5, 1, 9.

Определим запись типа TList с полями, содержащими характеристики данных – значения очередного элемента и адреса следующего за ним элемента

```
PList = ^TList;
  TList = record
     Data : Integer;
     Next : PList;
  end;
```

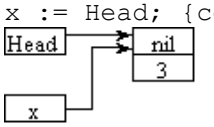
Чтобы список существовал, надо определить указатель на его начало. Опишем переменные.

```
Var
  Head, x : PList;
{Создадим первый элемент: }
New(Head); { выделяем место в памяти для переменной Head }
Head^.Next := nil; { указатель на следующий элемент пуст (такого элемента нет) }
Head^.Data := 3; { заполняем информационное поле первого элемента }
```

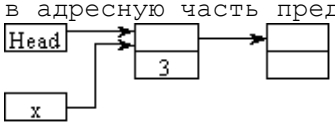


{Продолжим формирование списка, для этого нужно добавить элемент в конец списка. Введем вспомогательную переменную указательного типа, которая будет хранить адрес последнего элемента списка: }

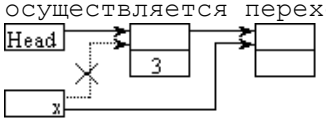
```
x := Head; {сейчас последний элемент списка совпадает с его началом}
```



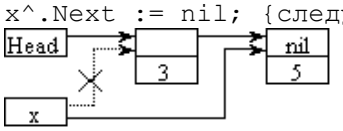

```
New(x^.Next); { выделим области памяти для следующего (2-го) элемента и поместим его адрес
в адресную часть предыдущего (1-го) элемента }
```




```
x := x^.Next ; { переменная x принимает значение адреса выделенной области. Таким образом
осуществляется переход к следующему (2-ому) элементу списка }
```



```
x^.Data := 5; { значение этого элемента }
x^.Next := nil; {следующего значения нет }
```



Остальные числа заносятся аналогично:

```
New(x^.Next); { выделим области памяти для следующего элемента }
x := x^.Next ; { переход к следующему (3-му) элементу списка }
x^.Data := 1; { значение этого элемента }
x^.Next := nil; {следующего значения нет }
New(x^.Next); { выделим области памяти для следующего элемента }
x := x^.Next; { переход к следующему (4-му) элементу списка }
x^.Data := 9; { значение этого элемента }
x^.Next := nil; {следующего значения нет }
```

Замечание. Как видно из примера, отличным является только создание первого (Head) элемента – головы списка. Все остальные действия полностью аналогичны и их естественно выполнять в цикле.

Присоединение нового элемента к голове списка производится аналогично:

```
New(x); { ввод значения элемента x^.Data := … }
x^.Next := Head; Head := x; ……..
```

В этом случае последний введенный элемент окажется в списке первым, а первый – последним.

Просмотр списка.

Просмотр элементов списка осуществляется последовательно, начиная с его начала. Указатель List последовательно ссылается на первый, второй и т. д. элементы списка до тех пор, пока весь список не будет пройден. При этом с каждым элементом списка выполняется некоторая операция – например, печать элемента. Начальное значение List – адрес первого элемента списка (Head).

```
List := Head;
While List <> nil do
  begin
     WriteLn(List^.Data);
     List := List^.Next; { переход к следующему элементу; аналог для массива i:=i+1 }
  end;
```

Удаление элемента из списка.

При удалении элемента из списка необходимо различать три случая:

1. Удаление элемента из начала списка.
2. Удаление элемента из середины списка.
3. Удаление из конца списка.

Digit – значение удаляемого элемента.

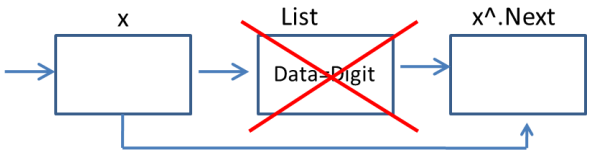
Удаление элемента из начала списка.

```
List := Head; { запомним адрес первого элемента списка }
Head := Head^.Next; { теперь Head указывает на второй элемент списка }
Dispose(List); { освободим память, занятую переменной List^ }
```

Удаление элемента из середины списка.

Для этого нужно знать адреса удаляемого элемента и элемента, находящегося в списке перед ним.

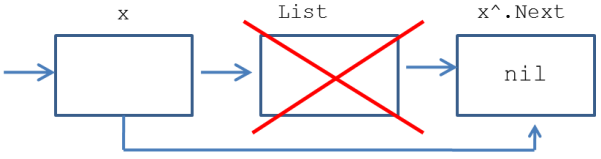
```
List := Head;
While (List< >nil) and (List^.Data< >Digit) do
  begin
     x := List;
     List := List^.Next;
  end;
x^.Next := List^.Next;
Dispose(List);
```



Удаление из конца списка.

Оно производится, когда указатель x показывает на предпоследний элемент списка, а List – на последний.

```
List := Head; x := Head;
While List^.Next<>nil do
  begin
     x := List;
     List := List^.Next;
  end;
x^.Next := nil;
Dispose(List);
```



Контрольные вопросы:

1. *Что такое указатель?*
2. *Что такое список?*
3. *Как реализуется описание списка?*
4. *На что должен указывать последний элемент списка?*
5. *Как создаются элементы списка?*
6. *Как удаляются элементы списка?*

Варианты заданий

Часть 1.

1. Сформировать список строк и а) сохранить его в текстовом файле; б) сохранить его в обратном порядке в текстовом файле.
2. Сформировать список строк из текстового файла.
3. Написать функцию, которая вычисляет среднее арифметическое элементов непустого списка.
4. Написать процедуру присоединения списка L2 к списку L1.
5. Написать функцию, которая создает список L2, являющийся копией списка L1, начинающегося с данного узла (задает пользователь).
6. Написать функцию, которая подсчитывает количество вхождений элемента, который вводит пользователь, в списке.
7. Написать функцию, которая удаляет из списка все вхождения элемента, который вводит пользователь.
8. Сформировать список целых чисел и удалить из него все четные.
9. Сформировать список вещественных чисел и вычислить сумму.
10. Написать функцию, которая проверяет, упорядочены ли элементы списка по алфавиту.
11. Написать функцию, подсчитывающую количество слов в списке, которые начинаются с той же буквы, что и следующее слово.
12. Написать функцию, которая использует исходный список L и создает два новых списка L1 и L2. L1 содержит нечетные узлы, а L2 – четные.
13. Написать функцию, которая использует исходный список L и создает два новых списка L1 и L2. L1 содержит нечетные числа, а L2 – четные.

Часть 2.

1. Составить программу, которая вставляет в список *L* новый элемент *F* за каждым вхождением элемента *E.*
2. Составить программу, которая вставляет в список *L* новый элемент F перед первым вхождением элемента *E,* если *E* входит в *L.*
3. Составить программу, которая удаляет из списка *L* все элементы *E,* если таковые имеются.
4. Составить программу, которая удаляет из списка *L* за каждым вхождением элемента *E* один элемент, если таковой имеется и он отличен от *E.*
5. Составить программу, которая удаляет из списка *L* все отрицательные элементы.
6. Составить программу, которая проверяет, есть ли в списке *L* хотя бы два одинаковых элемента.
7. Составить программу, которая переносит в конец непустого списка *L* его первый элемент.
8. Составить программу, которая вставляет в список *L* за первым вхождением элемента *E* все элементы списка *L,* если *E* входит в *L.*
9. Составить программу, которая переворачивает список *L,* т.е. изменяет ссылки в этом списке так, чтобы его элементы оказались расположенными в обратном порядке.
10. Составить программу, которая в списке *L* из каждой группы подряд идущих одинаковых элементов оставляет только один.
11. Составить программу, которая формирует список *L,* включив в него по одному разу элементы, которые входят одновременно в оба списка *L1* и *L2.*
12. Составить программу, которая формирует список *L,* включив в него по одному разу элементы, которые входят в список *L1,* но не входят в список *L2.*
13. Составить программу, которая формирует список *L,* включив в него по одному разу элементы, которые входят в один из списков *L1* и *L2,* но в то же время не входят в другой.