КАЗАНСКИЙ (ПРИВОЛЖСКИЙ) ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ
ИНСТИТУТ ВЫЧИСЛИТЕЛЬНОЙ МАТЕМАТИКИ И
ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ

ПРАКТИКУМ ПО РАЗРАБОТКЕ КОНСОЛЬНЫХ ПРИЛОЖЕНИЙ В СРЕДЕ DELPHI

УЧЕБНО-МЕТОДИЧЕСКОЕ ПОСОБИЕ

КАЗАНЬ 2012

УДК 519.682

Печатается по решению Редакционно-издательского совета ФГАОУВПО «Казанский (Приволжский) федеральный университет»

методической комиссии института вычислительной математики и информационных технологий
Протокол № 9 от 10 мая 2012 г.

заседания кафедры информатики и вычислительных технологий
Протокол № 9 от 20 апреля 2012 г.

Рецензенты:
канд. физ.-мат. наук, с.н.с. (ИММ КазНЦ РАН) Топорков Д.Ю.
канд. пед. наук, доц. КФУ Фаткуллов И.Р.

Халитова З.Р., Хисматуллина Н.А.

Практикум по разработке консольных приложений в среде Delphi. Учебно-методическое пособие / З.Р. Халитова, Н.А.Хисматуллина. – Казань: Казанский федеральный университет, 2012. – 85 с.

Предлагаемое пособие предназначено для обеспечения аудиторных и самостоятельных занятий студентов направления подготовки «Педагогическое образование» (специальность "информатика"). Оно содержит примеры решения задач по программированию на языке Object Pascal с подробными комментариями и задания для самостоятельной работы.

Оглавление

Введение

Предлагаемый практикум предназначен для студентов высших учебных заведений, изучающих программирование на языке Object Pascal, он содержит задачи с решениями на программирование с использованием как базовых алгоритмических структур, так и основных типов данных. Решения задач оформлены следующим образом. Сначала перечисляются обозначения, используемые в программе для исходных, промежуточных величин и результатов, затем указываются входные и выходные данные. Далее приводятся подробное словесное описание алгоритма, программа на языке Object Pascal, при необходимости – комментарии к ней и примеры ее исполнения с различными наборами входных данных. Подробное описание языка программирования Object Pascal можно найти, например, в [1]. В конце каждого параграфа приводятся задачи для самостоятельного решения.

1. Программирование линейных алгоритмов

Задача 1.1. Дано целое число x. Вычислить: $y = 5x^3 - |2x - 1| + 4$.

Входные данные: x. **Выходные данные:** y.

Вводится целое число x и значение y вычисляется по формуле $y = 5x^3 - |2x - 1| + 4$. Результат выводится на экран.

```
program Project1_1;                    {заголовок программы}
{$apptype console}                     {директива компилятора}
uses  SysUtils;                        {подключенный к программе модуль}
var     x, y: integer;                       {описание переменных}
begin
   write('x='');          { оператор предназначен для вывода  сообщения x=}
   readln(x);                  { оператор предназначен для ввода x}
   y:= 5*x*sqr(x)-abs(2*x-1)+4; {оператор предназначен для вычисления y }
   writeln('y=',y);            { оператор предназначен для вывода y }
   readln              { оператор предназначен для задержки окна вывода }
end.
```

Пример исполнения программы: Если в качестве значения переменной x ввести число -1, то далее вычисляем значение переменной y:= 5*x*sqr(x)-abs(2*x-1)+4 и y получает значение, равное -4; результат выводим на экран.

Задача 1.2. Перевести величину заданного угла из градусной меры в радианную.

Введем обозначения: x – величина данного угла в градусах; y – величина этого же угла в радианах.

Входные данные: x. **Выходные данные:** y.

Для перевода значения величины угла из градусной меры в радианную используется формула: $y = \frac{\pi \cdot x}{180}$.

```
program Project1_2;                {заголовок программы}
{$apptype console}                 {директива компилятора}
uses  SysUtils;                    {подключенный к программе модуль}
```

```
var     x, y:real;                  {описание переменных}
begin   write('x=');                {вывод  сообщения x=}
        readln(x);                  { ввода x}
        y:=pi*x/180;                {вычисление y, pi – стандартная функция}
        writeln('y=',y);            { вывод y }
        readln                      { задержка окна вывода }
end.
```

Комментарий к программе: Результатом является вещественное число y, оно выводится на экран с использованием экспоненциальной части. Например, если ввести x=45, то на экран выведется: y= 7.85398163397448E-0001.

Для получения результата в обычной форме с необходимым количеством цифр после запятой используется форматный вывод. При этом оператор writeln('y=',y:8:4) выведет результат y=␣␣0.7854 (четыре цифры в дробной части числа, точка занимает отдельную позицию), а оператор writeln('y=',y:8:6) - результат y=0.785398.

Задача 1.3. Даны действительные числа x,y,z. Вычислить значения выражений: $a = \dfrac{z - \sqrt[3]{13{,}5}}{\sin^2 y + z^2}$, $b = 2\sin\dfrac{\pi}{13} - \dfrac{\ln 5 + x}{1 + e^{y+z}}$.

Входные данные: x,y,z. **Выходные данные:** a,b.

```
program Project1_3;
{$apptype console}
uses  SysUtils;
var     x, y, z, a, b:real ;        {описание переменных}
begin   write('x=');                {вывод  сообщения x=}
        readln(x);                  {ввод x}
        write('y=');                {вывод сообщения y=}
        readln(y);                  {ввод y}
        write('z=');                {вывод сообщения z=}
        readln(z);                  {ввод z}
```

```
a:=(z-exp(1/3*ln(13.5)))/(sqr(sin(y))+sqr(z));          {вычисление  a  }
b:=2*sin(pi/13)-(ln(5)+x)/(1+exp(y+z));                 {вычисление  b  }
writeln('a=',a:8:2);                          {вывод a }
writeln('b=',b:8:2);                          {вывод b }
readln                                        {задержка экрана }
```

end.

Комментарий к программе: Исходные данные x,y,z можно ввести с использованием одного оператора ввода: readln(x,y,z). Тогда первые шесть строк раздела операторов приведенной программы заменяются на два оператора:

```
write('vvod x,y,z:');          {вывод сообщения vvod x,y,z:}
readln(x,y,z) ;                {ввод  x,y,z}
```

Результат можно вывести в одной строке экрана. Например:

```
writeln('a=',a:8:2,' b=',b:8:2);  {после вывода значения переменной b курсор переводится в первую позицию следующей строки экрана}
```

Задача 1.4. Дано трехзначное натуральное число n. Вычислить среднее арифметическое цифр этого числа.

Введем обозначения:

n – заданное число; a – первая цифра числа n; b – вторая цифра числа n; c – третья цифра числа n; d – число, составленное из первых двух цифр числа n; s – сумма всех цифр числа n; sred - среднее арифметическое цифр числа n.

Входные данные: n. **Выходные данные:** sred.

Для выделения цифр заданного числа n нужно использовать целочисленные операции mod (остаток от деления целых чисел) и div (целочисленное деление). Третья цифра (последняя цифра в записи числа) вычисляется с помощью оператора c:= n mod 10. Число d состоит из первых двух цифр числа n: d:= n div 10. Для вычисления второй цифры числа d можно применить оператор b:= d mod 10, первой цифры a:= d div 10. Сумма всех трех цифр s:= a+b+c , а среднее арифметическое sred:=s/3.

```
program Project1_4;
{$apptype console}
```

```
uses  SysUtils;
var      n,a,b,c,d,s: integer;  sred: real ;  {описание переменных}
begin write('n='');                          {вывод на экран сообщения n=}
      readln(n);                             {ввод n}
      c:= n  mod 10; d:= n div 10;
      b:= d  mod 10; a:= d  div 10;
      s:=a+b+c; sred:=s/3;
      writeln('sred=',sred:6:1);             {вывод результата}      readln
end.
```

Пример исполнения программы: если ввести n= 147, то

c=147 mod 10 =7; d=147 div 10 =14; b=14 mod 10 =4; a=14 div 10 =1;

s=1+4+7 =12; sred=12/3 = 4 (результат вещественного типа).

Задачи для самостоятельного решения

1. Даны действительные числа x,y. Вычислить: $z = \sin 2y - \sqrt[4]{e^x + 2}$, $t = \dfrac{5x - 1}{|\cos y + 3|} - 3{,}5$.

2. Даны действительные числа x,y. Вычислить: $f = \sqrt{5\sin^2 \dfrac{3x}{2} + 2{,}4x^4 + 8}$, $g = ctgy + 2\cos\dfrac{\pi}{5}$.

3. Найти сумму и произведение цифр данного двухзначного натурального числа.

4. Даны действительные числа x,y,z,t. Найти их среднее арифметическое.

5. Даны действительные числа x_1,y_1,x_2,y_2. Найти расстояние между двумя точками с координатами (x_1,y_1) и (x_2,y_2).

6. Дано трехзначное натуральное число. Найти сумму квадратов всех его цифр.

7. Дано четырехзначное натуральное число n. Вычислить сумму первой и последней цифры этого числа.

8. Даны длины сторон равностороннего треугольника. Вычислить площадь и высоту треугольника.

9. Треугольник задан координатами своих вершин. Вычислить его периметр и площадь.

10. Найти объем *V* и площадь *S* поверхности шара заданного радиуса *R*:

$V = \frac{4}{3}\pi R^3; S = 4\pi R^2$.

11. Найти объем *V* и площадь *S* поверхности цилиндра заданной высоты *h* и радиуса основания *R*: $V = \pi R^2 h; S = 2\pi R(h + R)$.

2. Программирование разветвляющихся алгоритмов

Задача 2.1. Даны действительные числа a,b. Вычислить:

$$c = \begin{cases} 7ab - 3{,}5 & \text{, если } a < b \\ \dfrac{1}{a^2 + b^2 + 5} & \text{, если } a \geq b \end{cases}$$

Входные данные: a, b. **Выходные данные:** c.

Вводятся действительные числа a и b. Проверяется условие a<b, если это условие выполняется, то значение c вычисляется по формуле $c = 7ab - 3{,}5$, иначе (т.е. выполняется противоположное условие a≥b) c вычисляется по формуле $c = \frac{1}{a^2 + b^2 + 5}$. Результат выводится на экран.

```
program Project2_1;
{$apptype console}
uses  SysUtils;
var    a,b,c:real ;
begin   write('a='');  readln(a); write('b='); readln(b);
        if  a<b then c:= 7*a*b-3.5
                 else c:= 1/(sqr(a)+sqr(b)+5);
        writeln('c=',c:5:1);readln
end.
```

Примеры исполнения программы:

1) введем a=5, b=8; вычислим значение логического выражения 5<8, равное true; выполним оператор, записанный после служебного слова then (c:= 7*a*b-3.5): переменная c получит значение 276.5, условный оператор завершит работу; на экран выводим результат.

2) введем a=6, b=3; вычислим значение логического выражения 6<3, равное false; выполним оператор, записанный после служебного слова else (c:= 1/(sqr(a)+sqr(b)+5)): переменная c получит значение 0.02, условный оператор завершит работу; на экран выводим результат.

Задача 2.2. Дано действительное число x. Вычислить:

$$y = \begin{cases} 4 - 3ctgx^2, & \text{если } x \le -2 \\ \sqrt{|x^3 - 4x + 1|}, & \text{если } -2 < x \le 5 \\ 2\ln\frac{x}{2} - 11{,}8, & \text{если } x > 5 \end{cases}$$

Входные данные: x. **Выходные данные:** y.

Вводится действительное число x. Проверяется условие x ≤-2, если это условие выполняется, то значение y вычисляется по формуле $y = 4 - 3ctgx^2$ и условный оператор завершает свою работу, иначе (т.е. выполняется противоположное условие x>-2) остается проверить условие x ≤5. Если это условие выполняется, то значение y вычисляется по формуле $y = \sqrt{|x^3 - 4x + 1|}$, если нет (т.е. выполняется противоположное условие x>5), то y вычисляется по формуле $y = 2\ln\frac{x}{2} - 11{,}8$. Результат выводится на экран.

```
program Project2_2;
{$apptype console}
uses  SysUtils;
var    x,y:real ;
begin   write('x='); readln(x);
```

```
if  x<=-2  then y:= 4-3* cos(sqr(x))/sin(sqr(x))
                else
                    if  x<=5  then y:= sqrt(abs(x*sqr(x)-4*x+1))
                              else y:=2* ln(x/2)-11.8;
writeln('y=',y:6:1);readln
end.
```

Комментарий к программе: В условном операторе после служебного слова then записан один оператор присваивания y:= 4-3* cos(sqr(x))/sin(sqr(x)), а после else - один условный оператор в полной форме **if** x<=5 **then** y:= sqrt(abs(x*sqr(x)-4*x+1)) **else** y:=2* ln(x/2)-11.8, поэтому никаких операторных скобок **begin end** здесь не требуется.

Примеры исполнения программы:

1) введем x=4; вычислим значение логического выражения 4<=-2, равное false; выполним условный оператор, записанный после служебного слова else, т.е. вычислим значение логического выражения 4<=5, равное true; выполним оператор, записанный после служебного слова then (y:= sqrt(abs(x*sqr(x)-4*x+1))): переменная y получает значение 7.0, выполнение условных операторов завершается; на экран выводим результат.

2) введем x=10; вычислим значение логического выражения 10<=-2, равное false; выполним условный оператор, записанный после служебного слова else, т.е. вычислим значение логического выражения 10<=5, равное false; выполним оператор, записанный после служебного слова else (y:=2* ln(x/2)-11.8): переменная y получает значение –8.6, выполнение условных операторов завершается; на экран выводим результат.

3) введем x= -3; вычислим значение логического выражения -3<=-2, равное true; выполним оператор, записанный после служебного слова then (y:= 4-3* cos(sqr(x))/sin(sqr(x)))): переменная y получает значение 10.6, выполнение условных операторов завершается; на экран выводим результат.

Задача 2.3. Даны три действительных числа x, y, z. Вычислить max(x, y, z) и min(x, y ,z).

Введем обозначения: x, y, z – заданные числа; max – наибольшее из чисел x, y, z; min – наименьшее из чисел x, y, z.

Входные данные: x, y, z. **Выходные данные:** max, min.

Сравниваются два заданных числа x>y, если это условие выполняется, то наибольшим из этих двух чисел будет x (max:=x), а наименьшим y (min:=y). Если это условие не выполняется, то наибольшим из этих двух чисел будет y (max:= y), а наименьшим x (min:= x). Таким образом, переменная max получила значение наибольшего из первых двух чисел x и y, а переменная min – наименьшего из этих чисел. Для того чтобы найти наибольшее из трех чисел x, y и z, нужно сравнить третье число z с уже найденным наибольшим из чисел x и y (т.е. max). Если выполняется условие z>max, то за наибольшее из чисел x, y, z нужно взять z (max:=z). Если же окажется, что z<=max, то значение переменной max менять не нужно, это означает, что при этом следует использовать неполную форму условного оператора. Аналогично число z сравнивается с наименьшим из двух чисел x и y (т.е. min). Если выполняется условие z<min, то за наименьшее из чисел x, y, z нужно взять z (min:=z).

```
program Project2_3;
{$apptype console}
uses  SysUtils;
var      x, y, z, min, max:real ;                    {описание переменных}
begin   write('x,y,z:'); readln(x, y, z);
        if x>y then                    {условный оператор в полной форме}
                    begin   max:=x; min:=y  end
              else    begin  max:=y; min:=x  end;
        if  z>max then max:=z;   {условный оператор в сокращенной форме}
        if  z<min then min:=z;    {условный оператор в сокращенной форме}
        writeln('max=',max:5:2);                {вывод max }
         writeln('min=', min:5:2);                {вывод min } readln
end.
```

Пример исполнения программы: Введем x=5, y=8, z=13; в первом условном операторе вычисляем значение логического выражения 5>8, равное false; выполняем составной оператор, записанный после else (max=8, min=5). Найдены наибольшее и наименьшее из чисел 5 и 8. Во втором условном операторе вычисляем значение логического выражения 13>8, равное true; выполняем составной оператор, записанный после then (max=13). Найдено наибольшее из трех чисел 5, 8, 13. В третьем условном операторе вычисляем значение логического выражения 13<5, равное false, переменная min остается равной 5.

Задача 2.4. Дано двухзначное натуральное число n. Встречается ли в записи этого числа цифра 5?

Введем обозначения: n – заданное число; a – первая цифра числа n; b – вторая цифра числа n.

Входные данные: n. **Выходные данные:** сообщение ‘da’ или ‘net’.

Вторая цифра (последняя цифра в записи числа) вычисляется по формуле: b:=n mod 10, первая цифра a:= n div 10. Если первая или вторая цифра равна 5, то в записи числа n встречается цифра 5, иначе – не встречается.

```
program Project2_4;
{$apptype console}
uses  SysUtils;
var      n,a,b: integer;
begin
        write(‘n=’); readln(n);
        b:= n  mod 10; a:= n  div 10;
        if  (a=5) or (b=5)  then writeln(‘da’)
                                else   writeln(‘net’); readln
end.
```

Комментарий к программе: В условном операторе логическое выражение содержит логическую операцию **or** и операции сравнения. Так как приоритет операции **or** выше приоритета операций сравнения, в логическом выражении a=5 **or** b=5 сначала выполняется логическая операция **or** с числовыми опе-

рандами: 5 **or** b. Компилятор не выдаст сообщения об ошибке, но выполнение этой операций не соответствует содержанию задачи, а следовательно, и ее алгоритму. Для изменения порядка выполнения операций нужно использовать круглые скобки: (a=5) **or** (b=5).

Пример исполнения программы:

Если ввести n=52, то b= 52 mod 10 =2; a= 52 div10 =5; вычисляем значение логического выражения (5=5) **or** (2=5), равное true (т.к. результат операций сравнения: 5=5 – true, 2=5 – false, результат логической операции true **or** false будет true); на экран выводим сообщение ‘da’.

Если же первоначально n=40, то b= 40 mod 10 =0; a= 40 div10 =4, значение логического выражения (4=5) **or** (0=5) равно false, на экран выводим ‘net’.

Задача 2.5. Даны действительные числа x, y. Определить, принадлежит ли точка с координатами (x, y) заштрихованной части плоскости (рис.1):

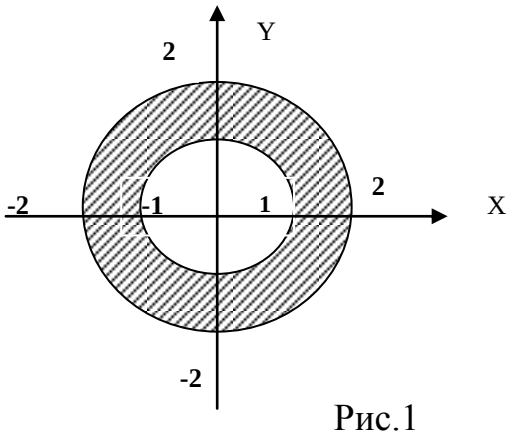

Рис.1

Входные данные: x, y. **Выходные данные:** сообщение ‘da’ или ‘net’.

Точки заштрихованной на рис.1 части плоскости принадлежат кольцу, ограниченному двумя окружностями, радиусы которых соответственно равны 1 и 2. Уравнение окружности радиуса r с центром в начале координат имеет вид: $x^2 + y^2 = r^2$. Все точки плоскости с координатами (x, y), удовлетворяющие условию $x^2 + y^2 \le 4$, лежат внутри круга радиуса 2, а удовлетворяющие условию $x^2 + y^2 \ge 1$ - вне круга радиуса 1 (точки на окружностях принадлежат заштрихованной на рис.1 части плоскости). Таким образом, если координаты точки,

удовлетворяют условию $(x^2 + y^2 \le 4)$ **and** $(x^2 + y^2 \ge 1)$, то они принадлежат заштрихованной части плоскости, иначе - не принадлежат.

```
program Project2_5;
{$apptype console}
uses  SysUtils;
var      x, y:real ;
begin
     write('x='); readln(x); write('y='); readln(y);
     if (sqr(x)+sqr(y)<=4) and (sqr(x)+sqr(y)>=1) then   writeln('da')
                                                  else   writeln('net');
     readln
end.
```

Примеры исполнения программы: 1) введем x=1, y=1; вычислим значение логического выражения $(1^2 + 1^2 \le 4)$ and $(1^2 + 1^2 \ge 1)$, равное true; на экран выводим сообщение 'da'. Это означает, что точка с координатами (1,1) принадлежит заштрихованной части плоскости.

2) введем x=1, y= -5; вычислим значение логического выражения $(1^2 + (-5)^2 \le 4)$ and $(1^2 + (-5)^2 \ge 1)$, равное false; на экран выводим сообщение 'net'. Следовательно, точка с координатами (1, -5) не принадлежит заштрихованной части плоскости.

Задача 2.6. Даны действительные числа x, y. Определить, принадлежит ли точка с координатами (x, y) заштрихованной части плоскости (рис.2).

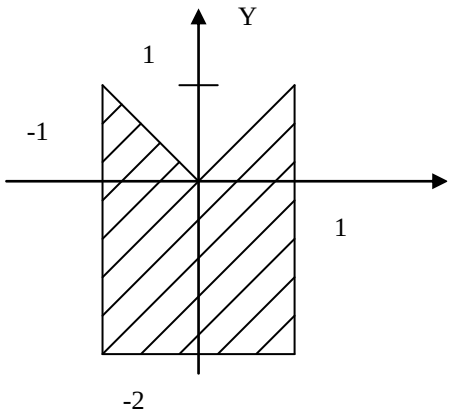

Рис.2

Входные данные: x, y. **Выходные данные:** сообщение ‘da’ или ‘net’.

Прямая x=0 делит заштрихованную на рис.2 часть плоскости на две части. Правая часть этой области (при x>=0) ограничена прямыми y=x, x=1 и y= -2. Все точки плоскости с координатами (x, y), удовлетворяющие условию (y<=x) **and** (x<=1) **and** (y>= -2), принадлежат правой части области. Левая часть (при x<0) ограничена прямыми y=-x, x= -1 и y= -2. Точки с координатами (x, y), удовлетворяющие условию (y<= -x) **and** (x>= -1) **and** (y>= -2), принадлежат левой части области.

```
program Project2_6;
{$apptype console}
uses  SysUtils;
var    x, y:real ;
begin   write(‘x=’); readln(x); write(‘y=’); readln(y);
        if x>=0 then
                if (y<=x) and (x<=1) and (y>=-2) then writeln(‘da’)
                                                 else   writeln(‘net’)
                 else
                   if (y<= -x) and (x>= -1) and (y>= -2) then writeln(‘da’)
                                                         else   writeln(‘net’);
        readln
end.
```

Примеры исполнения программы:

1) Введем x=1, y=0.5. Вычислим значение логического выражения 1>=0, равное true; выполним условный оператор, записанный после then, в нем вычислим значение логического выражения (y<=x) **and** (x<=1) **and** (y>=-2), равное true, т.к. значение выражения (0.5<=1) **and (**1<=1) **and** (0.5>= -2) равно true; на экран выводим сообщение ‘da’. Это означает, что точка с координатами (1, 0.5) принадлежит заштрихованной части плоскости.

2) Введем x= -0.5, y= -3. Вычислим значение логического выражения -0.5>=0, оно равно false; поэтому выполним условный оператор, записанный

после else. При этом сначала вычислим значение логического выражения, записанного после if: (-3<= - (-0.5)) **and** (-0.5>= -1) **and** (-3>= -2) равно false; на экран выводим сообщение ‘net’. Это означает, что рассматриваемая точка с координатами (-0.5, -3) не принадлежит заштрихованной части плоскости.

Задача 2.7. Дан номер месяца. Определить количество дней в этом месяце.

Введем обозначения: n- номер месяца; k – количество дней этого месяца.

Входные данные: n. **Выходные данные:** k.

```
program Project2_7;
{$apptype console}
uses  SysUtils;
var      n,k: integer ;
begin   write(‘n=’); readln(n);
          case n of
          1,3,5,7,8,10,12:k:=31;
          2:k:=28;       {год не високосный}
          4,6,9,11:k:=30
          end;
         writeln(‘k=’,k); readln
end.
```

Пример исполнения программы: введем n=2; значение переменной n ищем сначала в списке констант 1,3,5,7,8,10,12; затем в следующем списке, выполняем оператор, записанный после найденной константы 2 (k:=28), после этого оператор выбора завершает работу; на экран выводим результат.

Задачи для самостоятельного решения

1. Даны три действительных числа x, y, z. Вычислить max(x+2, y+z).
2. Даны действительные числа x,y,z. Существует ли треугольник с длинами сторон x,y,z?
3. Даны действительные числа x,y,z. Определить, является ли треугольник со сторонами x,y,z равнобедренным.

4. Решить линейное уравнение ax=b.

5. Дано действительное число x. Вычислить:

а) $y = \begin{cases} \dfrac{7\sqrt{7}\sin^2 x}{9{,}2\cos 3x + 1}, & \textit{если} \quad x \geq 0 \\ 3\sqrt{3}\sin x^2 + \dfrac{5{,}2}{x-1}, & \textit{если} \quad x < 0 \end{cases}$

б) $y = \begin{cases} 4 - \sqrt{|2x-6|}, & \text{если } x \geq 5 \\ \ln^2 x - 1, & \text{если } 2 \leq x < 5 \\ 3x^3 - 4x - 5, & \text{если } x < 2 \end{cases}$

в) $y = \begin{cases} x^3 - \dfrac{\sin^2 x}{2}, & \text{если } x < 1 \\ \sqrt{2x + \cos x^2}, & \text{если } 1 \leq x < 5 \\ \ln\dfrac{x}{2} - \dfrac{x}{x+2}, & \text{если } x \geq 5 \end{cases}$

6. Дано трехзначное натуральное число n. Имеется ли в нем цифра 1?

7. Дано двузначное натуральное число n. Верно ли, что первая цифра в два раза меньше, чем вторая?

8. Дано трехзначное натуральное число. Вычислить произведение ненулевых цифр этого числа.

9. Верно ли, что сумма цифр данного двузначного натурального числа является четной?

10. Дано трехзначное натуральное число n. Сосчитать количество единиц в записи этого числа.

11. Даны три действительных числа. Увеличить на пять те из них, значения которых неотрицательны.

12. Определить, есть ли среди цифр заданного целого трёхзначного числа одинаковые.

13. Дано натуральное число n. Проверить, является ли оно нечетным двухзначным.

14. Определить, имеется ли среди заданных целых чисел a, b, c хотя бы одно чётное.

15. Единицы длины пронумерованы: 1 – сантиметр, 2 – миллиметр, 3 – метр, 4- километр, 5 –дециметр. Дан номер единицы длины и длина отрезка в этих единицах. Найти длину отрезка в метрах.

16. Единицы измерения информации пронумерованы: 1 – бит, 2 – байт, 3 – килобайт, 4- мегабайт, 5 –гигобайт. Дан номер единицы информации и его значение в этих единицах. Вычислить количество информации в битах.

17. Определить, является ли заданный символ символом арифметической операции (+,-, *,/), строчной латинской буквой или цифрой.

18. Даны действительные числа x,y. Определить, принадлежит ли точка с координатами x,y заштрихованной на рис. 3 части плоскости.

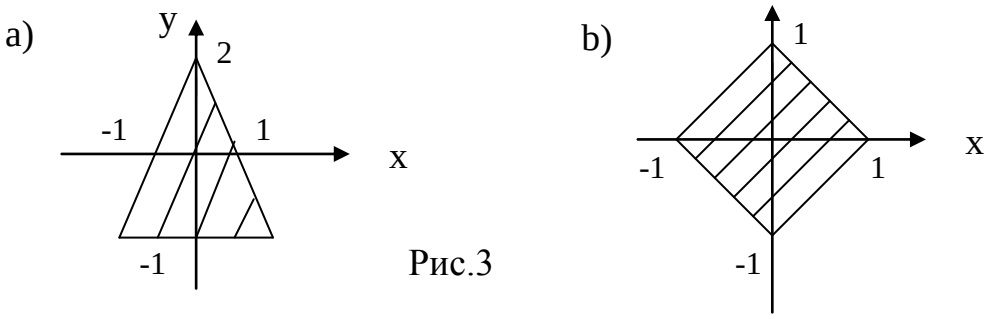

Рис.3

3. Программирование циклических алгоритмов

Задача 3.1. Даны действительные числа a, b, h. Вычислить значение функции $y = 2x^3 + x - 7$ при $a \le x \le b$ с шагом h.

Входные данные: a,b,h. **Выходные данные:** x, y.

Необходимо вычислить значение функции y для x = a; a+h; a+2h; …; b-h; b. Переменная x получает начальное значение (x:=a). Пока x не превышает конечного значения b (x<=b) вычисляется значение функции, на экран выводятся значения x и y, затем переменная x получает следующее значение (x:=x+h). При этом переменная x справа и слева от знака присваивания имеет разные значения: x справа – уже вычисленное, «старое» значение x, x слева – следующее, «новое» значение x. В программе Project3_1a использован цикл с предусловием.

```
program Project3_1a;
{$apptype console}
 uses  SysUtils;
 var     a,b,h,x,y:real ;
 begin write('a,b,h:'); readln(a,b,h);
        x:=a;                          {начальное значение x}
        while x <=b+ 0.00001 do    {пока x<=b выполнить}
        begin y:=2*sqr(x)*x+x-7;
              writeln('x=',x:3:1,'y=',y:6:1);
              x:=x+h                   {следующее значение x}
        end; readln
  end.
```

Комментарий к программе: Если в программе записать условие продолжения цикла x<=b, то последнее значение функции при x= b не будет выведено на экран. Так как арифметические операции с вещественными числами выполняются по правилам, определенным для чисел с плавающей точкой, а операция сравнения на равенство x= b производится побитово, и результат ее всегда будет равным false. Поэтому для того, чтобы цикл выполнился нужное число раз, к величине b нужно прибавить какое-нибудь число, меньшее h. В программе Project3_1b использован цикл с постусловием:

```
program Project3_1b;
{$apptype console}
uses  SysUtils;
var       a,b,h,x,y:real ;
begin   write('a,b,h:'); readln(a,b,h);
       x:=a;                  {начальное значение x}
       repeat                 { выполнить }
             y:=2*sqr(x)*x+x-7; writeln('x=',x:3:1,'y=',y:6:1);
             x:=x+h           {следующее значение x}
       until x >b+ 0.00001; { до соблюдения условия x>b } readln
```

```
end.
```

Переменная x не может использоваться в качестве параметра цикла **for,** так как ее тип real не является порядковым. Цикл **for** можно использовать с переменной i целого типа, которая будет считать, сколько раз уже вычислилось значение функции y. Переменная n содержит количество повторений: $n = \left[\frac{b-a}{h} \right] + 1$, здесь [ ] - целая часть. В программе Project3_1c использован цикл с параметром (цикл по переменной i):

```
program Project3_1c;
{$apptype console}
uses  SysUtils;
var      a,b,h,x,y:real ; n,i:integer;
begin   write('a,b,h:'); readln(a,b,h);
        n:=trunc((b-a)/h)+1; x:=a;
        for i:=1 to n do
        begin y:=2*sqr(x)*x+x-7;
              writeln('x=',x:3:1,'y=',y:6:1);
              x:=x+h
        end; readln
end.
```

Задача 3.2. Дано натуральное число n, действительное число x. Вычислить: $s = \sum_{i=1}^{n} \frac{x^i}{i}$.

Введем обозначения: n- количество слагаемых; i – номер слагаемого в сумме; s – сумма; a – i-е слагаемое суммы; b –числитель i-го слагаемого.

Входные данные: n,x. **Выходные данные:** s.

Запишем искомую сумму подробнее: $s = \sum_{i=1}^{n} \frac{x^i}{i} = x + \frac{x^2}{2} + \frac{x^3}{3} + ... + \frac{x^n}{n}$.

Искомая величина s накапливается постепенно. Первоначально сумма равна нулю: не просуммировано ни одно слагаемое, т.е. s:=0. Далее в цикле пока не

вычислена сумма n слагаемых, к уже накопленной сумме добавляется очередное слагаемое a, т.е. s:=s+a. Здесь слева и справа от знака присваивания записано имя одной и той же переменной s, именно это обеспечивает постепенное накопление суммы: s справа – уже вычисленное известное значение суммы, s слева – ее новое, вычисляемое значение. Очередное слагаемое $a = \frac{b}{i}$, где $b = x^i$. Вычисление значения b можно запрограммировать с помощью выражения $\exp(i * \ln(x))$, где x>0. Но, во-первых, по условию задачи x –любое действительное число, т.е. может быть и отрицательным, что приведет к ошибке при вычислении логарифма, во-вторых, вычисление значений функций exp и ln в теле цикла в данном случае является неоправданным и приводит к неэффективной программе. Поэтому для вычисления x^i лучше использовать рекуррентные соотношения, т.е. очередное слагаемое выразить через предыдущее:

при i=1 $b = x = (1) \cdot x$

при i= 2 $b = x^2 = (x) \cdot x$

... .

при i= n-1 $b = x^{n-1} = (x^{n-2}) \cdot x$

при i= n $b = x^n = (x^{n-1}) \cdot x$

Таким образом, i–е значение равно (i-1)–му, умноженному на x: b:=b*x. Как и ранее, для накопления произведения одно и то же имя переменной используется в левой и правой части оператора присваивания.Первоначально b=1.

При решении этой задачи можно применить любой из циклов: цикл с предусловием, цикл с постусловием или цикл с параметром. Если используется цикл с предусловием или цикл с постусловием, то начальное значение для параметра цикла i:=1 и его изменение i:=i+1 обязательно указываются. В программе Project3_2a использован цикл с предусловием.

```
program Project3_2a;
{$apptype console}
uses  SysUtils;
var    n,i:integer; x,a,b, s:real ;
```

```
begin write('n='); readln(n);    write('x='); readln(x);
      s:=0; b:=1; i:=1;    {начальные значения s, b, i}
      while i<=n do       {пока i<=n выполнить}
      begin b:=b*x; a:=b/i; s:=s+a;
            i:=i+1         {следующее значение i}
      end; writeln('s=',s:6:1); readln
end.
```

В программе Project3_2b используется цикл с постусловием:

```
program Project3_2b;
{$apptype console}
uses  SysUtils;
var   n,i: integer; x,a,b,s: real ;
begin write('n='); readln(n); write('x=');    readln(x);
      s:=0; b:=1; i:=1;
      repeat                {выполнить}
            b:=b*x; a:=b/i; s:=s+a;    i:=i+1
      until i>n;            { до выполнения условия i>n }
      writeln('s=',s:6:1); readln
end.
```

Использование цикла с параметром (программа Project3_2c):

```
program Project3_2c;
{$apptype console}
uses  SysUtils;
var   n,i:integer; x,a,b,s:real ;
begin write('n='); readln(n); write('x=');    readln(x);
      s:=0; b:=1;
      for i:=1 to n do
      begin b:=b*x; a:=b/i; s:=s+a end;
      writeln('s=',s:6:1); readln
end.
```

Задача 3.3. Вычислить значение бесконечной суммы $s = \sum_{i=1}^{\infty} \frac{1}{i^2}$ с заданной точностью ε. Считать, что требуемая точность достигнута, если очередное слагаемое оказалось по модулю меньше, чем заданная точность, это и все последующие слагаемые не учитывать.

Введем обозначения: s – искомая сумма; a – очередное слагаемое в сумме; i – его номер; eps – точность, с которой вычисляется бесконечная сумма.

Входные данные: eps. **Выходные данные:** s.

При вычислении бесконечной суммы количество слагаемых заранее не известно, но известно условие завершения вычисления этой суммы: как только очередное слагаемое оказалось по модулю меньше, чем заданная точность, суммирование должно прекратиться. Точность вычисления – это некоторое положительное малое число, например, 10^{-5}, 10^{-6} и др. Для решения задачи можно использовать цикл с предусловием или цикл с постусловием.

Первоначально сумма равна нулю. Вычислим первое слагаемое при i=1 a=1, если оно оказалось по модулю больше или равно заданной точности, то это слагаемое добавляется в сумму и вычисляется следующее слагаемое и т.д. Если очередное слагаемое оказалось по модулю меньше, чем заданная точность, то бесконечная сумма вычислена с указанной точностью, иначе оно снова добавляется в сумму.

В программе Project3_3a использован цикл с предусловием.

```
program Project3_3a;
{$apptype console}
uses  SysUtils;
var    i:integer; eps,a, s:real ;
begin          write('eps='); readln(eps);
       s:=0; a:=1; i:=1;              {начальные значения s, a, i}
       while abs(a)>=eps do      {пока |a| ≥ eps выполнить}
       begin s:=s+a;
             i:=i+1          {следующее значение i}
```

```
        a:=1/sqr(i);   {следующее слагаемое}
    end;
    writeln('s=',s:1:5); {число знаков при выводе зависит от eps }
    readln
end.
```

Пример исполнения программы: Введем eps=0.2. Переменные получают значения: s:=0; a:=1; i:=1. Далее вычисляем значение логического выражения abs(1)>=0.2, равное true, поэтому выполняем тело цикла: первое слагаемое добавляем в сумму, в результате s=1, увеличиваем номер слагаемого: i=2 и определяем следующее слагаемое a=1/4; снова вычисляем значение выражения abs(1/4)>=0.2, равное true; выполняем тело цикла: второе слагаемое добавляем в сумму s=1.25, увеличиваем i=3 и определяем a=1/9. После этого снова вычисляем значение выражения abs(1/9)>=0.2, оно оказывается равным false. Следовательно, выполнение цикла while завершается и значение бесконечной суммы, вычисленное с точностью eps=0.2, выводим на экран: s=1.3.

В программе Project3_3b использован цикл с постусловием:

```
program Project3_3b;
{$apptype console}
uses  SysUtils;
var    i:integer; eps,a, s:real ;
begin  write('eps='); readln(eps);  s:=0; i:=1; a:=1;
    repeat                {выполнить}
        s:=s+a;  i:=i+1; a:=1/sqr(i);
    until abs(a)<eps;          { до соблюдения условия |a| < eps }
    writeln('s=',s:1:5);readln
end.
```

В этой программе первое слагаемое добавляется в сумму всегда, даже если условие abs(a)<eps выполняется с самого начала.

Задача 3.4. Дано натуральное число n. Вычислить факториал этого числа.

$n!=1 \cdot 2 \cdot 3 \cdot 4 \cdot \ldots \cdot (n-1) \cdot n$

Введем обозначения: n – данное число; i – множитель; p – факториал (n!).

Входные данные: n. **Выходные данные:** p.

Для вычисления факториала натурального числа n нужно найти произведение n множителей. Первый множитель –1, второй – 2, …, i–й множитель – i, где i=1,2,…,n. Произведение вычисляется с помощью оператора p:=p*i. При этом p справа и p слева имеют разные значения: p справа – уже известное, вычисленное ранее значение произведения, p слева – новое, вычисляемое его значение. Первоначально произведение равно единице, т.е. p:=1. В программе Project3_4 использован цикл с параметром:

```
program Project3_4;
{$apptype console}
uses  SysUtils;
var    n,i:integer; p:real ;
begin write('n='); readln(n); p:=1;
        for i:=1 to n do p:=p*i;
        writeln('p=',p:8:2); readln
end.
```

Задача 3.5. Дано натуральное число n. Определить количество и сумму цифр этого числа.

Введем обозначения: n – первоначально: заданное натуральное число, в процессе выполнения алгоритма n меняется: из его записи постепенно удаляются цифры по одной справа налево; a – последняя цифра числа n; k – искомое количество цифр; s – искомая сумма цифр.

Входные данные: n. **Выходные данные:** k, s.

Цифры заданного натурального числа удобнее выделять справа налево. Первоначально количество и сумма цифр числа полагаются равными нулю. До тех пор, пока n не станет равным нулю, повторяются следующие действия: определяется последняя цифра в записи числа n (a:= n mod 10), эта цифра добавляется к сумме (s:=s+a), количество цифр увеличивается на единицу (k:=k+1), и из числа n удаляется очередная обработанная последняя цифра (n:=n div 10).

```
program Project3_5;
{$apptype console}
uses  SysUtils;
var      n,a,k,s: integer;
begin   write('n='); readln(n); k:=0; s:=0;
        while n<>0 do
        begin    a:= n  mod 10;    s:=s+a;      k:=k+1;
                 n:= n  div 10;
        end;     writeln('k=',k, ' s=',s); readln
end.
```

Задача 3.6. Дано натуральное число n. Получить его перевертыш. Например, n= 4319, его перевертыш - 9134.

Введем обозначения. Так же, как в задаче 3.5, n – первоначально: заданное натуральное число, в процессе выполнения алгоритма n меняется: из его записи постепенно удаляются цифры по одной справа налево; a – последняя цифра числа n; p – разряд цифры a; m – перевертыш числа n.

Входные данные: n. **Выходные данные:** m.

Любое натуральное число можно представить в виде суммы степеней числа 10 с коэффициентами, равными соответствующим цифрам этого числа. Например: $m = 9134 = 9 \cdot 1000 + 1 \cdot 100 + 3 \cdot 10 + 4 \cdot 1$. Следовательно, нужно определить десятичный разряд, в котором будет расположена первая цифра числа m, тогда вторая цифра будет в разряде, в 10 раз меньшем и т.д. Для вычисления десятичного разряда (p) заданное число n запоминается в другой переменной, например, t (t:=n). Первоначально p – разряд единиц. Пока t не равно нулю, вычисляется десятичный разряд p (p:=p*10). Это значение оказывается в 10 раз большим, чем необходимо, поэтому p уменьшается в 10 раз (p:=p div 10).

Число m – это сумма некоторого количества слагаемых, первоначально полагаем ее равной нулю, т.е m:=0. До тех пор, пока заданное число n не станет равным нулю, повторяются следующие действия: определяется его последняя цифра (a:= n mod 10), эта цифра добавляется к числу m в разряде p (m:=m+a*p),

десятичный разряд p уменьшается в 10 раз (p:=p div 10), а в числе n удаляется последняя цифра (n:=n div 10). После завершения цикла значение m выводится на экран.

```
program Project3_6;
{$apptype console}
uses  SysUtils;
var      n,a,p,m,t: integer;
begin   write('n='); readln(n); t:=n; p:=1;
        while t<>0 do
        begin p:=p*10;  t:= t  div 10 end;
        p:=p div 10; m:=0;
        while n<>0 do
        begin a:= n  mod 10;
              m:=m+a*p; p:=p div 10;
              n:= n  div 10;
        end; writeln('m=',m); readln
end.
```

Задача 3.7. Даны положительные действительные числа a, ε. В последовательности $y_1, y_2,\ldots$, образованной по правилу:

$$y_1 = \frac{a+1}{2}, \quad y_i = \frac{1}{2}\left(y_{i-1} + \frac{a}{y_{i-1}} \right), \quad i = 2,3,...$$

найти первый член y_n, для которого выполняется неравенство $|y_n - y_{n-1}| < \varepsilon$ (ε - точность вычисления).

Введем обозначения: eps– точность вычисления; a – заданное число; x – (i-1)-й член последовательности y_{i-1}; y – i–й член последовательности y_i.

Входные данные: a, eps. **Выходные данные:** y.

В вычислениях участвуют лишь два члена последовательности y_{i-1} и y_i, и по условию задачи нет необходимости сохранять все члены последовательности. Поэтому вместо y_{i-1} можно использовать простую переменную x, а вместо y_i –

переменную y. Тогда i-й член последовательности вычисляется через (i-1)-й член по формуле: $y = \frac{1}{2}\left(x + \frac{a}{x}\right)$. Но тогда для вычисления следующего члена последовательности нужно изменить значение переменной x: присвоить ей уже вычисленное значение y (x:=y). Как только выполнится условие $|y - x| < \varepsilon$, искомый член последовательности будет найден и его значение (y) выведется на экран.

Первый член последовательности получает значение $x = \frac{a+1}{2}$, второй член вычисляется по формуле $y = \frac{1}{2}\left(x + \frac{a}{x}\right)$. Пока выполняется условие $|y - x| \geq \varepsilon$, переменная x получает значение y и вычисляется следующий член последовательности.

```
program Project3_7;
{$apptype console}
uses  SysUtils;
var    eps, x, a, y: real ;
begin write('eps='); readln(eps); write('a='); readln(a);
      x:=(a+1)/2;  y:=(x+a/x)/2;
      while abs(y-x)>=eps do
      begin  x:=y; y:=(x+a/x)/2  end;
      writeln('y=',y:6:1);readln
end.
```

Задача 3.8. Даны натуральные числа m и n. Вычислить $1^n + 2^n + ... + m^n$.

Введем обозначения: m - количество слагаемых; n – степень каждого слагаемого; i – номер очередного слагаемого; j – номер множителя при вычислении степени; a – очередное слагаемое; s – сумма.

Входные данные: n, m. **Выходные данные:** s.

В искомой сумме m слагаемых, для каждого из них надо вычислить n-ю степень. Внешний цикл организуется по номеру слагаемого (i), в этом цикле

вычисляется i-е слагаемое и накапливается сумма. Для вычисления i-го слагаемого суммы $a = i^n$ нужно найти произведение $a = \underbrace{i \cdot i \cdot ... \cdot i}_{n \ раз}$. Внутренний цикл организуется по переменной j, которая обозначает порядковый номер множителя в этом произведении, т.е. j =1,2,…,n. В этом цикле вычисляется i-е слагаемое по формуле: a:=a*i. При каждом i начальное значение a равно единице.

Использование цикла с параметром (программа Project3_8a):

```
program Project3_8a;
{$apptype console}
uses  SysUtils;
var    n,m,i,j:integer; s,a:real ;
begin write('n='); readln(n); write('m=');    readln(m);
        s:=0;
        for i:=1 to m do                    {начало внешнего цикла}
        begin a:=1;
              for j:=1 to n do a:=a*i;    {внутренний цикл}
              s:=s+a
        end;                                 {конец внешнего цикла}
        writeln('s=',s:6:1); readln
end.
```

Использование цикла с предусловием (программа Project3_8b):

```
program Project3_8b;
{$apptype console}
uses  SysUtils;
var    n,m,i,j:integer; s,a:real ;
begin write('n='); readln(n); write('m=');    readln(m);
        s:=0; i:=1;
        while i<= m do
        begin a:=1; j:=1;
              while j<= n do
```

```
            begin a:=a*i; j:=j+1 end;
            s:=s+a; i:=i+1
        end;
        writeln('s=',s:6:1); readln
    end.
```

Задачи для самостоятельного решения

1. Вычислить значение функции $y = \sqrt[3]{2{,}3 + \cos 5x} + \frac{\sqrt{x^2 + 3}}{3x + 5}$ в точках -3, -2.5, -2, …, 2, 2.5,3.

2. Вычислить значение функции $y = \frac{3{,}2x \sin x \cos 5x + 1}{3\ln(x^2 + 1)}$ в точках 5, 5.1, 5.2, …,5.8, 5.9, 6.

3. Вычислить значение функции :

$$y = \begin{cases} x^3 \cos 2x + \dfrac{x + 2}{4}, & \text{если } x < -1 \\ \sin^2 x - \dfrac{x^2 - 2}{x + 4}, & \text{если } x \geq -1 \end{cases}$$ на отрезке [-2,2] с шагом 0,2

4. Даны действительные числа a, b, h. Вычислить значение функции:

$$y = \begin{cases} \dfrac{\sin(x + 2) + 3{,}2 \cos \dfrac{x}{2}}{5{,}7 + \sin x}, & \text{если } x \leq \pi \\ \sin x \sin 2x + 3, & \text{если } x > \pi \end{cases}$$ на отрезке [a,b] с шагом h.

5. Дано положительное число x. Вычислить:

a) $\sqrt{x} + \sqrt{x + 2} + \ldots + \sqrt{x + 20}$ b) $\frac{x + 16}{3} + \frac{x + 25}{4} + \cdots + \frac{x + 625}{24}$

c) $\frac{x + 1}{1} + \frac{2x + 1}{4} + \frac{3x + 1}{9} \ldots + \frac{20x + 1}{400}$

6. Дано действительное число x. Вычислить:

a) $(1 + 2x)(1 + 4x) \ldots (1 + 20x)$ b) $\frac{x + 1}{46} + \frac{x + 1}{44} + \ldots + \frac{x + 1}{6}$

c) $\frac{1}{x^2+2}\cdot\frac{2}{x^2+4}\cdot\ldots\cdot\frac{10}{x^2+20}$ d) $\sin x+3\sin 3x+\ldots+15\sin 15x$

7. Дано натуральное число n и действительное число x. Вычислить значения сумм:

a) $\frac{\cos 2x}{3}+\frac{\cos 4x}{15}+\ldots+\frac{\cos 2nx}{4n^2-1}$ b) $\sin x+\frac{\sin 3x}{3}+\cdots+\frac{\sin(2n-1)x}{2n-1}$

c) $1+\frac{2x}{1!}+\cdots+\frac{(2x)^n}{n!}$ d) $x+\frac{x^3}{3!}+\frac{x^5}{5!}+\ldots+\frac{x^{2n+1}}{(2n+1)!}$

e) $1+\frac{x^2}{2!}+\frac{x^4}{4!}+\ldots+\frac{x^{2n}}{(2n)!}$ f) $\sin x-\frac{\sin 2x}{2}+\cdots+(-1)^{n+1}\frac{\sin nx}{n}$

g) $-\cos x+\frac{\cos 2x}{2^2}+\cdots+(-1)^n\frac{\cos nx}{n^2}$ j) $x+\frac{x^5}{5}+\cdots+\frac{x^{4n+1}}{4n+1}$

8. Дано действительное число x. Вычислить:

a) $\sum_{m=1}^{\infty}\frac{1}{(m+1)m}$ b) $\sum_{k=7}^{\infty}\frac{k^2+k}{(k-4)!}$ c) $\sum_{k=0}^{\infty}\frac{(-3)^{k+1}}{(k+2)!}$

d) $\sum_{i=1}^{\infty}\frac{(-2)^i}{i!}$ e) $\sum_{m=1}^{\infty}\frac{x^m}{(m+1)m}$

9. Дано натуральное число n, действительное число a. Вычислить значение произведения a(a+1)(a+2)…(a+n).

10. Дано натуральное число n. Последовательность $a_0, a_1, \ldots$ образована по правилу: $a_0=3;\ \ a_k=2a_{k-1}+k/3,\ k=1,2,\ldots$ Вычислить $\sum_{k=0}^{n}\frac{a_k}{k+1}$.

11. Дано натуральное число n. Последовательность $x_1, x_2, \ldots$ образована по правилу: $x_1=x_2=1; x_i=x_{i-2}+\frac{x_{i-1}}{i}, i=3,4,\ldots$ Вычислить $x_1+x_2+\ldots+x_n$.

12. Дано натуральное число n. Последовательности $x_1, x_2, \ldots$ и $y_1, y_2, \ldots$ образованы по правилу: $x_1=y_1=1;\ \ x_i=0{,}2x_{i-1};\ \ y_i=3y_{i-1}+x_{i-1},\ \ i=2,3,\ldots$ Найти $\sum_{i=1}^{n}x_i y_i$.

13. Дано действительное число $\varepsilon > 0$. Последовательность $y_1, y_2, \ldots$ образована по правилу: $y_0 = 0$; $y_k = \frac{y_{k-1} + 1}{y_{k-1} + 2}$, $k = 1,2,\ldots$ Найти первый член y_n, для которого выполнено $y_n - y_{n-1} < \varepsilon$.

14. Последовательность $a_1, a_2, \ldots$ образована по правилу: $a_1 = 1$; $a_i = \frac{2 - a_{i-1}^3}{5}$, $i = 2,3,\ldots$ Найти первый член a_n, для которого $|a_n - a_{n-1}| < 10^{-5}$.

15. Дано натуральное число n. Вычислить произведение и количество нечетных цифр числа.

16. Дано натуральное число n. Вычислить сумму четных цифр числа.

17. Даны натуральные числа m и n. Найти их наибольший общий делитель.

18. Найти все трехзначные простые числа.

19. Найти все натуральные числа до 200, сумма цифр которых равна 13.

20. Даны натуральные числа m и n. Получить все натуральные числа, меньшие n, квадрат суммы цифр которых равен m.

21. Найти все целые числа из промежутка от 1 до 100, в написании которых встречается цифра 7.

22. Даны натуральные числа *n* и *m*. Вычислить: $\sum_{i=1}^{n} \sum_{j=1}^{m} \cos(i^3 + j)$.

23. Найти все натуральные числа до 200, сумма цифр которых равна 13.

24. Даны натуральные числа a и b (a<b). Вывести все целые числа от a до b включительно, при этом каждое число выводится столько раз, каково его значение (например, число a выводится a раз).

25. Дано натуральное число n. Напечатать разложение этого числа на простые множители. Каждый простой множитель должен быть напечатан один раз.

26. Дано натуральное число n. Напечатать разложение этого числа на простые множители. Каждый простой множитель должен быть напечатан столько раз, сколько раз он входит в разложение.

4. Обработка массивов

Задача 4.1. Дан целочисленный одномерный массив. Подсчитать количество положительных, отрицательных и нулевых элементов массива.

Введем обозначения: n – количество элементов в массиве; a – имя массива; i – индекс элемента массива; a_i – i-й элемент массива a; pol – количество положительных, ot – отрицательных, nul – нулевых элементов.

Входные данные: n, a. **Выходные данные**: pol, ot, nul.

Количество положительных, отрицательных и нулевых элементов массива в начальный момент полагаем равным нулю, т. е. pol:=0, ot:=0, nul:=0. Если очередной элемент массива положителен, то количество таких элементов нужно увеличить на единицу, т. е. pol:=pol+1. Если это не так, то элемент либо отрицательный, либо равен нулю. Надо проверить одну из оставшихся возможностей: если он отрицательный, то увеличить количество отрицательных элементов на единицу, т. е. ot:= ot+1, если нет, значит, элемент нулевой, при этом никаких дополнительных проверок и сравнений не требуется, количество нулевых элементов следует увеличить на единицу, т. е. nul:= nul+1. Таким образом обрабатываются все элементы массива.

```
program Project4_1;
{$apptype console}
uses  SysUtils;
var a: array[1..20] of integer; i,n,pol,ot,nul: integer;
begin    write('n='); readln(n);
         for i:=1 to n do  readln(a[i]); {ввод массива}
         pol:=0; ot:=0; nul:=0;
         for i:=1 to n do      {обработка массива}
              if a[i]>0 then   pol:= pol+1
                        else  if a[i]<0 then ot:=ot+1
                                        else nul:=nul+1;
         writeln('pol=',pol,' ot=',ot,' nul=',nul); readln
end.
```

В программе Project4_1 использованы циклы с параметром. При решении этой задачи можно применить и другие виды циклов: цикл с предусловием или цикл с постусловием. В следующей программе используются циклы с предусловием:

```
program Project4_1a;
{$apptype console}
uses  SysUtils;
var  a: array[1..20] of integer;   i,n,pol,ot,nul: integer;
begin  write('n='); readln(n);
        i:=1; while i<= n do begin  readln(a[i]); i:=i+1 end;
       pol:=0; ot:=0; nul:=0;
        i:=1; while i<= n  do
             begin  if  a[i]>0 then   pol:= pol+1
                                 else  if a[i]<0 then ot:=ot+1 else nul:=nul+1;
                    i:=i+1
             end;  writeln('pol=',pol,' ot=',ot,' nul=',nul); readln
end.
```

В первом цикле с предусловием пока индекс элемента не превышает количества элементов (i<= n), вводятся элементы массива. Во втором подсчитывается количество положительных, отрицательных и нулевых элементов массива.

Задача 4.2. Дан целочисленный одномерный массив. Встречается ли в нем отрицательный элемент?

Введем обозначения: m – количество элементов в массиве; x – имя массива; i – индекс элемента массива; x_i – i -й элемент массива x.

Входные данные: m,x. **Выходные данные**: сообщение 'yes' или 'no'.

Рассмотрим два способа решения этой задачи.

Первый способ. Введем еще одно обозначение: k – количество отрицательных элементов в массиве. В программе будем подсчитывать количество отрицательных элементов массива. Если это количество не равно нулю, то отрица-

тельные элементы есть. Если это условие не выполняется, т. е. количество таких элементов равно 0, то в массиве отрицательных элементов нет.

```
program Project4_2a;
{$apptype console}
uses  SysUtils;
const m=8;
var x: array[1..m] of integer; i,k: integer;
begin
      for i:=1 to m do  readln(x[i]); {ввод массива}
      k:=0;
     for i:=1 to m do  if x[i]<0 then  k:=k+1;
     if  k<>0 then   writeln('yes') else   writeln('no'); readln
end.
```

Этот способ решения предполагает просмотр всего массива до конца, даже если отрицательный элемент найден раньше.

Второй способ. Введем обозначение: s – признак наличия или отсутствия отрицательного элемента в массиве. Переменная логического типа s принимает значение false, если в массиве нет отрицательного элемента и значение true, если такой элемент есть. Первоначально полагаем, что в массиве нет отрицательного элемента: s:=false. Как только при просмотре массива найдется отрицательный элемент, переменная s получает значение true, и остальные элементы массива не просматриваются (осуществляется досрочный выход из цикла). Если отрицательный элемент в массиве не найден, то значение переменной s останется без изменения, т.е. false.

В цикле for нельзя осуществить досрочный выход из цикла (пока не просмотрены все элементы массива) без нарушения структурности алгоритма, поэтому лучше использовать циклы while или repeat. Из цикла нужно выйти, если найден отрицательный элемент, т.е. переменная s=true, или если уже проверены все элементы массива (i > m). Поэтому условие продолжения вычислений в цикле имеет вид: (i <= m) **and not** s. Если переменная s после завершения цикла

имеет значение true, то на экран выводится сообщение ‘yes’, иначе - сообщение ‘no’

```
program Project4_2b;
{$apptype console}
uses  SysUtils;
const m=8;
var x: array[1..m] of integer; i: integer; s:boolean;
begin
     for i:=1 to m do  readln(x[i]); {ввод массива}
    s:=false; i:=1;
    while (i<= m) and not s do
        begin  if x[i]<0 then  s:=true; i:=i+1 end;
    if s then   writeln(‘yes’) else   writeln(‘no’); readln
end.
```

Задача 4.3. Сформировать массив:

$a_1 = a_2 = 1; a_k = 2a_{k-1} + a_{k-2}^2; k = 3,4,..., n.$

Введем обозначения: a – имя массива; n – количество элементов в массиве; k – порядковый номер элемента в массиве; a_k – k -й элемент массива a;

Входные данные: n. **Выходные данные:** a.

```
program Project4_3;
{$apptype console}
uses  SysUtils;
var a: array[1..20] of integer; n, k : integer;
begin  write(‘n =’);  readln(n);
    a[1]:=1; a[2]:=1; {формирование массива a }
    for k:=3 to n do a[k]:= 2*a[k-1]+sqr(a[k-2]);
   for k:=1 to n do writeln (a[k] ); {вывод массива a } readln
end.
```

Задача 4.4. Дан целочисленный одномерный массив. Из его нечетных элементов сформировать новый одномерный массив.

Введем обозначения: n – количество элементов в заданном массиве; x – его имя; i – порядковый номер элемента этого массива; x_i – i-й элемент массива x; y – имя формируемого массива; j – порядковый номер элемента в массиве y; y_j – j-й элемент массива y.

Входные данные: n,x. **Выходные данные:** y.

Первоначально в формируемом массиве y нет элементов, поэтому порядковый номер текущего элемента полагаем равным нулю: j: =0. Если очередной элемент заданного массива x нечетный, то он записывается в новый массив y. При этом порядковый номер элемента массива y предварительно увеличивается на единицу j:=j+1. После того, как массив x просмотрен до конца, оказывается, что массив y состоит из j элементов, все они выводятся на экран.

```
program Project4_4;
{$apptype console}
uses  SysUtils;
var x,y: array[1..20] of integer; n,i,j : integer;
begin  write('n='');  readln(n);
        for i:=1 to n do  read(x[i]);
        j:=0; for i:=1 to n do
                if  x[i] mod 2<>0 then   begin j:=j+1; y[j]:= x [i] end;
        for i:=1 to j do writeln (y[i]:6:1); readln
end.
```

Задача 4.5. Дан целочисленный двумерный массив размера n×m. Найти среднее арифметическое ненулевых элементов третьей строки этого массива.

Введем обозначения: n - количество строк; m - количество столбцов в массиве; i - номер строки массива; j - номер столбца массива; x – имя массива; x_{ij} - элемент массива x, находящийся в i-й строке и j-м столбце; k, s, sr - соответственно количество, сумма и среднее арифметическое ненулевых элементов третьей строки массива.

Входные данные: n,m,x. **Выходные данные:** sr.

Первоначально количество и сумма ненулевых элементов третьей строки массива равны 0. Просматриваются элементы третьей строки массива, для этого организуется цикл по номеру столбца j. Если очередной элемент третьей строки массива x[3,j] является ненулевым, то количество таких элементов увеличивается на единицу k:=k+1, а к уже накопленной сумме добавляется этот элемент s:= s+ x[3,j].

```
program Project4_5;
{$apptype console}
uses  SysUtils;
const    n=4;  m=3;
var x: array[1..n,1..m] of  integer; i, j, k ,s: integer; sr:real;
begin  for i:=1 to n do    for j:=1 to m do  read(x[i,j]); {ввод массива}
          k:=0; s:=0;
          for j:=1 to m do
              if  x[3,j] <>0 then  begin k:=k+1; s:=s+x[3,j]  end;
              sr:=s/k; writeln ('sr=',sr:8:2); readln
end.
```

Решение этой же задачи с использованием цикла с постусловием:

```
program Project4_5a;
{$apptype console}
uses  SysUtils;
const    n=4;  m=3;
var x: array[1..n,1..m] of  integer; i, j, k,s : integer; sr:real;
begin for i:=1 to n do for j:=1 to m do  read(x[i,j]);
       k:=0; s:=0; j:=1;
      repeat   if  x[3,j] <>0 then begin k:=k+1; s:=s+x[3,j]  end;
                  j:=j+1
      until j >m;
      sr:=s/k; writeln ('sr=',sr:8:2); readln
end.
```

Комментарий к программе: При обращении к элементам третьей строки первый индекс равен 3. Просмотр элементов третьей строки двумерного массива начинается с первого столбца (j:=1) и продолжается (j:=j+1) до тех пор, пока не будут проверены все столбцы массива (j >m).

Задача 4.6. Дан двумерный символьный массив размера n×n. Все элементы, лежащие над главной диагональю, заменить символом ‘%’.

Введем обозначения: x – имя массива; n – количество его строк и столбцов; i – номер строки, j – номер столбца массива; x_{ij} – элемент массива x.

Входные данные: n,x. **Выходные данные:** x.

Просматриваются все элементы символьного массива. Элемент расположен над главной диагональю, если номер строки меньше номера столбца i <j, такой элемент заменяется символом ’%’.

```
program Project4_6;
{$apptype console}
uses  SysUtils;
const    n=3;
var x: array[1..n,1..n] of  char; i, j : integer;
begin  {ввод символьного массива в виде матрицы}
        for i:=1 to n do
        begin for j:=1 to n do  read(x[i,j]); readln     end;
        {обработка массива}
        for i:=1 to n do
              for j:=1 to n do if  i <j then   x[i,j]:=’%’;
        {вывод массива в виде матрицы}
        for i:=1 to n do
        begin  for j:=1 to n do  write(x[i,j]); writeln end;
        readln
 end.
```

Комментарии к программе: Обработать нужно лишь элементы массива, расположенные над главной диагональю. В последней строке такого элемента

нет, поэтому достаточно просмотреть строки с первой по предпоследнюю. В i-й строке элементами, расположенными над главной диагональю, являются элементы столбцов с номерами i+1,i+2, …n. Поэтому j меняется от i+1 до n.

```
program Project4_6a;
{$apptype console}
uses  SysUtils;
const    n=3;
var x: array[1..n,1..n] of  char;  i, j : integer;
begin  for i:=1 to n do
          begin  for j:=1 to n do  read(x[i,j]);  readln   end;
        for i:=1 to n –1 do  for j:=i+1 to n do   x[i,j]:=’%’;
        for i:=1 to n do
            begin   for j:=1 to n do  write(x[i,j]); writeln  end; readln
 end.
```

Задачи для самостоятельного решения

1. Дан одномерный массив, состоящий из n элементов. Найти сумму и количество элементов, меньших первого элемента этого массива.

2. Дан одномерный массив, состоящий из n элементов. Найти количество и произведение его элементов в диапазоне от 5 до 25.

3. Дан целочисленный одномерный массив, состоящий из n элементов. Верно ли, что четных элементов в нем больше, чем нечетных?

4. Дан одномерный массив, состоящий из k элементов. Найти среднее арифметическое элементов массива, индексы которых являются степенями двойки (1, 2, 4, 8, 16, ….).

5. Дан одномерный массив, состоящий из n элементов. Найти среднее арифметическое всех его элементов, кроме первого и последнего.

6. Дан одномерный массив, состоящий из n элементов. Найти сумму и количество его элементов, расположенных правее минимального.

7. Дан целочисленный одномерный массив, состоящий из n элементов. Верно ли, что произведение его четных элементов и произведение нечетных имеют разные знаки?

8. Дан одномерный массив $a_1, a_2, .., a_m$. Найти порядковый номер первого отрицательного элемента массива.

9. Какой элемент заданного одномерного массива встречается раньше: минимальный или максимальный?

10. Дан одномерный массив, состоящий из n элементов. Найти произведение и количество ненулевых элементов, расположенных левее максимального.

11. Дан одномерный массив, состоящий из n элементов. Найти произведение его максимального элемента и минимального среди положительных.

12. Дан одномерный массив, состоящий из n не равных между собой элементов. Найти сумму и количество его элементов, расположенных между максимальным и минимальным.

13. Дан одномерный целочисленный массив $a_1, ..., a_n$. Сформировать новый массив, элементы которого: 1, a_1, 2, a_2, ...,n, a_n.

14. Дано целое число a и массив $x_1, x_2, .., x_n$. Из элементов, меньших a, сформировать новый одномерный массив.

15. Дан одномерный массив a, состоящий из k элементов. Сформировать новый массив, элементы которого $b_i = \begin{cases} 3a_i^{\,3} + 5, если\, a_i < 7 \\ \sin a_i - 2, если\, a_i \geq 7 \end{cases}$.

16. Дан одномерный массив $a_1, a_2, .., a_m$. Сформировать новый одномерный массив, содержащий элементы заданного массива в обратном порядке.

17. Положительные элементы, расположенные в первой половине заданного одномерного массива, увеличить на 5, а расположенные во второй половине – уменьшить на 5.

18. Дан двумерный массив размера n×m. Найти среднее арифметическое его элементов.

19. Дан двумерный массив размера k×p. Вывести индексы элементов, кратных трем.

20. Дан двумерный массив размера n×m. Найти сумму минимальных элементов всех столбцов массива.

21. Дан двумерный массив размера n×n. Заменить диагональные элементы на максимальные в соответствующих строках.

22. Дан двумерный символьный массив размера n×m. Из символов-цифр образовать одномерный числовой массив.

23. Дан двумерный массив размера n×n. Поменять местами минимальный элемент главной диагонали с третьим элементом второй строки массива.

24. Дана двумерная матрица размера n×n. Заменить нулями все ее элементы, расположенные ниже побочной диагонали.

25. Сформировать двумерный целочисленный массив размера n×m по правилу: все элементы первой строки равны 10, а элементы каждой из последующих строк на 1 меньше элементов предыдущей.

26. Дано число *a*. Сформировать двумерный массив размера n×n, в котором элементы главной и побочной диагонали равны *a*, а остальные элементы равны сумме номеров строки и столбца, в которых они располагаются.

5. Обработка строк

Задача 5.1. Дана строка символов. Подсчитать в ней количество букв “a”.

Введем обозначения: s – заданная строка; i – номер символа в строке;

s[i] – i-й символ строки s; k – количество букв “a” в строке.

Входные данные: s. **Выходные данные:** k.

Первоначально (до начала обработки) количество букв “a” полагается равным нулю k:=0. Просматриваются все символы заданной строки с первого до последнего (длины строки). Если i-й символ строки s[i] совпадает с буквой “a”, то k увеличивается на единицу, иначе - остается без изменения.

```
program Project5_1;
{$apptype console}
```

```
uses  SysUtils;
var s:string; i,k:integer;
begin  readln(s);  k:=0;
        for i:=1 to length(s) do if s[i]='a' then k:=k+1;
        writeln('k=',k); readln
end.
```

Задача 5.2. Дана строка символов. Заменить сочетание “da” на “no”.

Введем обозначения: s – заданная строка; i – номер символа в строке.

Входные данные: s. **Выходные данные:** s.

Из строки s последовательно выделяются подстроки длиной 2 символа и сравниваются с сочетанием “da”. Если результат сравнения истинен, то i–й символ заданной строки заменяется на “n”, а (i+1)–й - на “o”, а i для продолжения обработки увеличивается на 2. В противном случае проверка продолжается со следующей пары символов, для этого i увеличивается на 1. Строка просматривается, начиная с первого символа до предпоследнего (в последний раз подстрока составляется из предпоследнего и последнего символов строки).

```
program Project5_2;
{$apptype console}
uses  SysUtils;
var s:string ; i:integer;
begin  writeln ('s= '); readln(s);
        i:=1;  while i<= length(s)-1  do
                if  copy(s, i, 2)='da'
                   then  begin  s[i]:='n'; s[i+1]:='o';  i:=i+2 end else i:=i+1;
         writeln('s= ', s); readln
end.
```

Задача 5.3. Дана строка символов. Вставить после символа “a” символ “o”.

Введем обозначения: s – заданная строка; i – номер символа в строке.

Входные данные: s. **Выходные данные:** s.

Длина строки в процессе обработки меняется, поэтому для организации цикла следует использовать операторы while или repeat. Оператор цикла for в данном случае использовать нельзя, так как если записать заголовок цикла в виде **for** i:=1 **to** length(s) **do**, то в качестве числа повторений цикла запоминается величина length(s), вычисленная до первого выполнения тела цикла. После этого число повторений цикла не изменится, хотя длина строки меняется в процессе обработки в теле цикла.

Вводится заданная строка. Начиная с первого символа, проверяется: если i–й символ строки совпадает с символом “a” (s[i]=’a’), то в строку s вставляется символ “o”, начиная со следующей (i+1)–й позиции, для проверки следующего символа в этом случае i нужно увеличить на 2 (i: = i +2). Если же i–й символ строки не совпадает с символом “a”, надо перейти к проверке следующего символа, т. е. i увеличить на 1 (i: = i +1). Проверка продолжается до последнего символа строки.

```
program Project5_3;
{$apptype console}
uses  SysUtils;
var s:string;   i:integer;
begin   writeln (‘s=’); readln(s);
        i:=1;
        while i<= length(s)  do
          if  s[i]=’a’ then  begin insert(’o’,s, i+1); i:=i+2  end
                       else i:=i+1;
        writeln(’s= ’, s);
        readln
end.
```

Задача 5.4. Дана строка символов. Сформировать новую строку, исключив из заданной строки буквы “a”.

Введем обозначения:

s – заданная строка; i – номер символа в строке; snew – новая строка.

Входные данные: s. **Выходные данные:** snew.

Первоначально формируемая строка snew не содержит ни одного символа, поэтому полагаем ее пустой. Просматриваются все символы заданной строки s. Если i-й символ (s[i]) не совпадает с буквой “a”, то он добавляется к новой строке snew.

```
program Project5_4;
{$apptype console}
uses  SysUtils;
var s, snew :string; i:integer;
begin write(‘s= ’); readln(s);
      snew:=’’;  for i:=1 to length(s) do if  s[i]<>’a’ then snew:= snew + s[i];
      writeln(‘snew=’, snew); readln
end.
```

Задачи для самостоятельного решения

Дана строка символов.

1.Встречается ли в строке сочетание “yes”?

2.Какой символ встречается в ней раньше: “+” или “-”?

3.Все ли символы этой строки являются строчными латинскими буквами?

4.Заменить буквы “a”, “o”, “e” на символ “*”.

5.Удалить из нее символы “+”, “-”, “*”, “/”.

6.Подсчитать в ней количество сочетаний “to”.

7.Подсчитать сумму четных цифр.

8.Удалить символы данной строки, расположенные между круглыми скобками. Считать, что внутри каждой пары скобок нет других скобок.

9.Из строчных латинских букв данной строки сформировать новую строку.

10. Сформировать новую строку из символов данной строки, расположенных на четных местах.

11. Верно ли, что в данной строке имеются все буквы слова “for”?

12. Подсчитать количество чисел в данной строке.

6. Программирование вспомогательных алгоритмов

Задача 6.1. Даны натуральные числа n, m и действительные числа x, y. Вычислить значение функции: $z = 2{,}5\sum_{i=1}^{n}\frac{x^2+7}{3i} + \sum_{i=1}^{m}\frac{y^2+4}{5i}$.

Входные данные: x,y,n,m. **Выходные данные:** z.

Для определения значения функции z нужно дважды вычислить сумму. Оформим ее вычисление в подпрограмме. Подпрограммы бывают двух видов - процедуры и функции. Покажем использование обоих видов. В подпрограмме–процедуре summa вычисляется сумма:

$$s = \sum_{i=1}^{k}\frac{p^2+a}{bi}, \qquad (*)$$

где k – количество слагаемых; $\frac{p^2+a}{bi}$ - слагаемое; i – его номер; s – сумма.

Процедура summa имеет пять формальных параметров: k, a, b, p – параметры-значения (исходные данные этой процедуры); s – параметр-переменная (выходная величина, результат этой процедуры). Переменная i – локальный параметр процедуры (промежуточная величина). В программе Project6_1a используется процедура summa:

```
program Project6_1a;
{$apptype console}
uses  SysUtils;
var   x,y,s1,s2,z: real; n,m: integer;
procedure summa (k,a,b:integer; p:real; var s:real);
var i:integer;
begin  s:=0;
          for i:=1 to k do s:=s+(sqr(p)+a)/(b*i)
 end;
begin  writeln (‘vvod n,x: ’);  readln(n,x);
         summa(n, 7,3,x,s1); {оператор вызова процедуры}
                                {n, 7,3,x,s1- фактические параметры}
```

```
    writeln (‘vvod m,y: ’);  readln(m,y);
    summa(m, 4,5,y,s2); {оператор вызова процедуры}
                        {m, 4,5,y,s2- фактические параметры}
    z:=2.5*s1+s2;  writeln(‘z=’,z:8:2); readln
end.
```

Вычисление суммы (*) можно оформить и в виде подпрограммы-функции. Функция sum имеет четыре формальных параметра k, a, b, p и два локальных параметра i, s. В программе Project6_1b используется функция sum:

```
program Project6_1b;
{$apptype console}
uses  SysUtils;
var   x,y,s1,s2,z: real; n,m: integer;
function sum (k,a,b:integer; p:real): real;
var i:integer; s:real;
begin
    s:=0; for i:=1 to k do s:=s+(sqr(p)+a)/(b*i);
     sum:=s        {имени функции sum присваивается результат}
end; {конец описания функции}
begin  writeln (‘vvod n,x: ’);  readln(n,x);
    s1:=sum(n, 7,3,x);   {n, 7,3,x- фактические параметры}
    writeln (‘vvod m,y: ’);  readln(m,y);
    s2:=sum(m, 4,5,y);  {m, 4,5,y- фактические параметры}
    z:=2.5*s1+s2;  writeln(‘z=’,z:8:2); readln
end.
```

При использовании функции в программе можно сразу вычислить значение z, не используя переменных s1 и s2, с помощью оператора z:=2.5*sum(n,7,3,x)+sum(m,4,5,y).

Задача 6.2. Даны действительные числа x и y. Вычислить:

$$z = 2f(\frac{x}{2}, y) + f^2(5, x + y), \quad \text{где} \quad f(a,b) = \frac{\sqrt{|a + b^3 - 1{,}5|}}{a^2 + 3b^2 + 1}$$

Входные данные: x,y. **Выходные данные:** z.

Для определения z нужно дважды вычислить значение функции f, но с разными значениями аргументов: $f(\frac{x}{2}, y)$ и $f(5, x + y)$. Вычисление этой функции можно осуществить в подпрограмме, а в самой программе дважды вызвать ее с нужными значениями фактических параметров. Подпрограмма–функция f имеет два формальных параметра a и b. При вызове этой функции f(x/2,y) и f(5,x+y) вместо формальных параметров подставляются фактические параметры: x/2 и y; 5 и x+y соответственно.

```
program Project6_2;
{$apptype console}
uses  SysUtils;
var   x,y,z:real ;
function f (a,b:real):real;
begin  {имени функции f присваивается результат}
        f:=sqrt(abs(a+sqr(b)*b-1.5))/(sqr(a)+3*sqr(b)+1)
end;
begin
         writeln (‘vvod x,y: ’);  readln(x,y);
         z:=2*f(x/2,y)+sqr(f(5,x+y));
         writeln(‘z=’,z:8:2); readln
 end.
```

Задача 6.3. Даны две строки символов. В первой строке удалить букву “a”, во второй строке - букву “o”.

Введем обозначения: s1,s2 – заданные строки; i – номер символа; s[i] – i-й символ строки s; c –переменная, значением которой является удаляемая буква.

Входные данные: s1,s2. **Выходные данные:** s1,s2.

В первой и во второй строке необходимо удалить определенные буквы. В программе вводятся заданные строки, дважды вызывается подпрограмма, осуществляющая удаление нужных букв и выводится результат.

В подпрограмме в строке символов s удаляются все буквы, совпадающие со значением переменной c. Для этого просматриваются все символы заданной строки - с первого символа до последнего (его номер равен длине строки). Если i-й символ строки s[i] совпадает с разыскиваемой буквой (c), то этот символ удаляется, т.е. в строке s удаляется, начиная с i–го символа, один символ - delete(s,i,1). Длина строки уменьшается на единицу, при этом символ, находившийся на (i +1)–й позиции окажется на i–й и должен проверяться следующим. Если же i–й символ не совпадает с разыскиваемой буквой (c), то проверяется следующий по порядку символ строки, для этого i увеличивается на единицу (i:=i+1). Так как длина строки меняется в процессе обработки, для организации цикла следует использовать операторы while или repeat.

В подпрограмме-процедуре два формальных параметра s,c: s,c – исходные данные для этой процедуры; s является одновременно и результатом процедуры. Поэтому c должен быть описан как параметр-значение, а s – как параметр-переменная. При вызове процедуры формальные параметры заменяются на соответствующие фактические параметры.

```
program Project6_3a;
{$apptype console}
uses  SysUtils;
var  s1,s2:string;
procedure del (var s:string; c:char);
var i:integer;    { i – локальный параметр}
  begin i:=1;
        while i<= length(s)  do      if  s[i]=c then  delete(s,i,1)  else i:=i+1;
end; {конец описания процедуры}
begin  write('first string: '); readln(s1);
        del (s1, 'a');            {s1,'a' - фактические параметры}
        writeln('s1=',s1);
        write('second string: '); readln(s2);
        del (s2, 'o');            {s2,'o' - фактические параметры}
```

```
        writeln('s2=',s2); readln
    end.
```

В подпрограмме-функции два формальных параметра-значения s и c. Изменившаяся строка s будет значением функции del_fun. В программе Project6_3b используется функция:

```
program Project6_3b;
{$apptype console}
uses  SysUtils;
var  s1,s2:string;
function del_fun (s:string; c:char): string;
var i:integer;    { i – локальный параметр}
  begin i:=1;
        while i<= length(s)  do
          if  s[i]=c then  delete(s,i,1)  else i:=i+1;
        del_fun:=s
end;
begin  write('first string: '); readln(s1);
      s1:=del_fun (s1, 'a');          {s1,'a' - фактические параметры}
      writeln('s1=',s1);          write('second string: '); readln(s2);
      s2:=del_fun (s2, 'o');          {s2,'o' - фактические параметры}
      writeln('s2=',s2); readln
  end.
```

Задача 6.4. Даны два целочисленных одномерных массива $x_1, x_2, ..., x_n$ и $y_1, y_2, ..., y_m$. Для первого массива вычислить сумму элементов, кратных трем, для второго массива – кратных пяти.

Введем обозначения: x и y –имена заданных массивов; n- количество элементов в массиве x; m – количество элементов в массиве y; sx – сумма элементов массива x, кратных трем, sy - сумма элементов массива y, кратных пяти.

Входные данные: n,m,x,y. **Выходные данные:** sx, sy.

Даны два одномерных массива x и y, состоящие из разного количества элементов. В программе вводится количество элементов в первом (n) и втором (m) массивах. Для ввода одномерных массивов создается процедура vvod с двумя формальными параметрами k и z, где k - количество элементов в массиве z. В этой процедуре k - параметр-значение (входная величина); z - параметр-переменная (выходная величина, результат).

Тип формального параметра процедуры должен быть идентификатором, поэтому заголовок процедуры нельзя записать в виде:

procedure vvod (k:integer; **var** z: ~~array[1..30] of integer~~);

Правильное описание заголовка этой процедуры имеет вид:

procedure vvod (k:integer; **var** z: mas);

которому должно предшествовать описание **type** mas= **array**[1..30] **of** integer;

Для ввода элементов массива понадобится локальная переменная i, определяющая порядковый номер элемента в массиве. Описание процедуры vvod может иметь вид:

```
procedure vvod (k:integer; var z: mas);
var i:integer;
begin   for i:=1 to k do readln(z[i])  end;
```

В процедуре summa вычисляется сумма (s) элементов массива, кратных p. В заголовке описаны четыре формальных параметра: k, p, z – параметры-значения (входные величины этой процедуры); s – параметр-переменная (выходная величина, результат). Переменная i – локальный параметр.

```
program Project6_4;
{$apptype console}
uses  SysUtils;
type mas= array[1..30] of integer;
var   x,y:mas; n, m, sx, sy: integer;
procedure vvod (k:integer; var z: mas);
var   i:integer;
begin   for i:=1 to k do readln(z[i])  end;
```

```
procedure summa (k,p :integer; z: mas; var s:integer);
var  i :integer;
begin   s:=0;
        for i:=1 to k do     if  z[ i ] mod p=0 then  s:=s+z[i];
end;
begin   write('n='); readln(n);
        vvod(n, x);            { n, x - фактические параметры}
        summa (n,3, x, sx); { n,3, x, sx - фактические параметры}
        writeln('sx=',sx);
        write('m='); readln(m);
        vvod(m, y);            { m, y - фактические параметры}
        summa (m,5, y, sy); { m,5, y, sy - фактические параметры}
        writeln('sy=',sy); readln;
end.
```

Задача 6.5. Даны два одномерных массива. В первом из них поменять местами минимальный элемент с третьим, а во втором – с последним элементом массива.

Введем обозначения: a и b –имена заданных массивов; n- количество элементов в массиве a; m – количество элементов в массиве b; i – номер элемента в массиве; a[i] – i-й элемент массива a; b[i] – i-й элемент массива b.

Входные данные: n,m,a,b. **Выходные данные:** a,b.

В программе вводится количество элементов в первом и втором массивах, дважды вызывается процедура vvod, осуществляющая ввод элементов массива, и процедура obr, меняющая местами минимальный элемент с заданным элементом массива, и выводятся изменившиеся одномерные массивы.

В процедуре obr входными величинами являются: c –одномерный массив; k- количество элементов в массиве c; p - номер элемента, с которым меняется местами минимальный элемент. Результатом этой процедуры является изменившийся массив c. Таким образом, массив $c_1, c_2, ..., c_k$ является и входной, и выходной величиной этой процедуры, его необходимо описать как параметр-

переменную. В процедуре obr три формальных параметра: k,p – параметры-значения; c – параметр-переменная. Для того чтобы поменять местами минимальный элемент с элементом c_p, нужно найти минимальный элемент (min), его порядковый номер (t) и поменять местами элементы c_t и c_p. Переменные min, t, i – локальные параметры процедуры obr.

```
program Project6_5;
{$apptype console}
uses  SysUtils;
type massiv= array[1..30] of real;
var a,b: massiv; i,n,m: integer;
procedure vvod (k: integer; var c: massiv);
var i:integer;
begin   for i:=1 to k do readln(c[i])  end;
procedure obr (k,p: integer; var c: massiv);
var  i, t :integer;  min:real;
begin   min:=c[1]; t:=1;
        for i:=2 to k do     if c[ i ]<min then   begin min:=c[ i ]; t:= i end;
        c[t]:=c[p];  c[p]:=min
end;
begin   write('n='); readln(n);
        vvod(n, a);   obr(n,3,a);
        for i:=1 to n do write(a[ i ]:6:1); writeln;
        write('m='); readln(m);
        vvod(m, b);  obr(m,m,b);
        for i:=1 to m do write(b[ i ]:6:1);writeln; readln;
end.
```

В программе Project6_5 дважды выводится одномерный массив, для его вывода можно создать процедуру vivod. В этой процедуре будет два формальных параметра-значения k, c:

```
procedure vivod (k:integer; c: massiv);
```

```
var i:integer; { i - локальный параметр}
begin   for i:=1 to k do write(c[i]:6:1); writeln
end;
```

Тогда в разделе операторов программы можно дважды вызвать процедуру vivod: vivod (n, a); vivod (m, b).

Задача 6.6. Даны две символьные матрицы размера nxm и кхр. Проверить, встречается ли во втором столбце первой матрицы символ ‘+’, а в третьем столбце второй матрицы - символ ‘%’.

Введем обозначения: n- количество строк матрицы a; m – количество ее столбцов; к- количество строк матрицы b; p – количество ее столбцов; sa - признак наличия или отсутствия символа ‘+’ во втором столбце матрицы a; sb- признак наличия или отсутствия символа ‘%’ в третьем столбце матрицы b; i – номер строки матрицы; j – номер столбца матрицы; x[i,j] - элемент массива x, находящийся в i -той строке, j -том столбце; t- номер обрабатываемого столбца; c - переменная, значением которой является разыскиваемый символ; s - признак наличия или отсутствия разыскиваемого символа.

Входные данные: n,m,k,p,a,b. **Выходные данные:** сообщение “yes” или “no”.

В программе вводится количество строк и столбцов первой и второй матриц; дважды вызывается процедура vvod, осуществляющая ввод элементов матрицы, и процедура poisk, проверяющая, встречается ли в заданном столбце нужный символ. Если во втором столбце первой матрицы встречается символ ‘+’ и в третьем столбце второй матрицы - символ ‘%’, то выводится сообщение “yes”, иначе - “no”.

В процедуре vvod построчно вводится символьная матрица x размера nxm. Элементы одной строки вводятся друг за другом без каких-либо разделителей, строки матрицы разделяются клавишей ввода (этот символ считывается при помощи readln). В этой процедуре три формальных параметра n, m, x: n, m – параметры-значения (исходные данные); x - параметр-переменная (результат).

В процедуре poisk элементы t-го столбца матрицы x сравниваются со значением переменной c. Переменная логического типа s принимает значение true,

если в столбце есть нужный элемент и значение – false, если такого элемента нет. Первоначально полагаем, что среди элементов t-го столбца матрицы x нет элемента, равного значению переменной c, и переменной s присваиваем значение false. Если при просмотре элементов t-го столбца матрицы найдется элемент, равный c, то переменная s получит значение true, остальные элементы t-го столбца не просматриваются. При этом если такой элемент не отыщется, то значение переменной s останется без изменения, т. е. false. В процедуре poisk пять формальных параметров n, t, x, c, s, среди них: n, t, x, c – параметры-значения (исходные данные); s - параметр-переменная (результат процедуры). В программе Project6_6 дважды вызывается процедура poisk: poisk (n,2, a, '+', sa); poisk (k,3, b, '%', sb). Если переменные sa и sb одновременно получают значение true, то на экран выводится сообщение 'yes', иначе - сообщение 'no'.

```
program Project6_6;
{$apptype console}
uses  SysUtils;
type mas= array[1..10,1..10] of  char;
procedure vvod (n,m: integer; var x: mas);
var i, j:integer;        { i, j – локальные параметры}
begin for i:=1 to n do {ввод символьной матрицы}
            begin   for j:=1 to m do  read(x[i,j]);   readln     end;
end;
procedure poisk (n,t: integer;x: mas;c:char; var s:boolean);
var i:integer;  { i – локальный параметр}
begin s:=false; i:= 1;
      while (i<=n) and not s do
      begin  if  x[i, t]= c then   s:=true;
                i:=i+1
      end;
end;
var  a,b: mas;  n, m, k,p : integer; sa,sb:boolean;
```

```
begin
  write('n='); readln(n); write('m='); readln(m);
  writeln('massiv a:'); vvod (n,m,a);      { n,m,a- фактические параметры}
  write('k='); readln(k); write('p='); readln(p);
  writeln('massiv b:'); vvod (k,p,b);      { k,p,b - фактические параметры}
  poisk (n,2, a, '+', sa);                 {n,2, a, '+', sa - фактические параметры}
  poisk (k,3, b, '%', sb);                 {k,3, b, '%', sb - фактические параметры}
  if sa and sb   then   writeln ('yes' )
                 else   writeln ('no' ); readln
end.
```

Задачи для самостоятельного решения

1. Даны стороны двух треугольников. Найти сумму периметров и сумму площадей этих треугольников. Использовать процедуру, которая по сторонам *a*, *b*, *c* треугольника вычисляет его периметр $p = a + b + c$ и площадь $s = \sqrt{\frac{p}{2}(\frac{p}{2} - a)(\frac{p}{2} - b)(\frac{p}{2} - c)}$.

2. Даны натуральные числа n, m и действительные числа x, y. Вычислить значение функции: $z = \sum_{i=1}^{n} \frac{1 + \cos x}{i + 2} + \sum_{i=1}^{m} \frac{\cos y}{2i + 3}$.

3. Вычислить значение функции $f(x) = \sin(2x) + \frac{x^4 + 1}{2}$ при следующих значениях *x*=-5, -4.5, -4, …, 1, 1.5,2.

4. Даны действительные числа s и t. Вычислить значение выражения $g(4s, t - 1) + \frac{1 - 5g(t, s^2)}{2}$, где $g(x, y) = \frac{\sqrt{|\sin x - 3xy|}}{2x^2 + y^4 + 1}$.

5. Даны натуральные числа n, m и действительное число x. Вычислить значение функции: $y = 6{,}2\sum_{i=1}^{n} \frac{2x + 7}{i(5 + i)} + 1{,}8\sum_{i=1}^{m} \frac{x + 6}{i(7 + i)}$.

6. Даны натуральные числа n, m и действительные числа x, y. Вычислить значение функции: $s = \sum_{i=1}^{n} \frac{x^i}{i!} + \sum_{i=1}^{m} \frac{y^i}{i!}$.

7. Даны координаты вершин двух треугольников. Определить, какой из них имеет меньшую площадь.

8. Даны два одномерных массива $a_0, a_1, .., a_n$ и $b_1, b_2, .., b_m$. В каждом из них найти максимальный и минимальный элемент.

9. Даны два одномерных массива. Проверить, верно ли, что минимальный элемент первого из них находится на пятом месте, а второго - на третьем.

10. Даны два одномерных массива $c_0, c_1, .., c_n$ и $b_1, b_2, .., b_m$. В каждом из них вычислить произведение и количество элементов, меньших среднего арифметического элементов соответствующего массива.

11. Даны два целочисленных одномерных массива. Встречаются ли в первом массиве три нулевых элемента, а во втором - два элемента, равные k?

12. Даны два двумерных массива размера n x m и p x к. Найти сумму четных элементов, расположенных в первых двух строках первой матрицы и первых s строках второй матрицы.

13. Даны два двумерных массива размера n×n и k×k. Для каждого массива заменить последний элемент последней строки суммой элементов, расположенных ниже главной диагонали.

14. Даны два двумерных массива размера n×n и k×k. Среди элементов, лежащих выше побочной диагонали, найти максимальный элемент и поменять его местами в первом массиве с элементом второй строки g-го столбца, а во втором массиве с элементом h-й строки третьего столбца.

15. Даны два одномерных массива. В каждом из них найти максимальный элемент и поменять его местами в первом массиве с последним, а во втором - с q-м элементом.

16. Даны два двумерных массива размера n×m и k×3. Из элементов первого массива, не превышающих 5, и элементов второго, не превышающих 10, сформировать новые одномерные массивы.

17. Даны два одномерных массива. В каждом из них вычислить сумму и произведение ненулевых элементов.

18. Даны два целочисленных массива. В первом массиве заменить четные элементы на 105, а во втором нечетные - на 200.

19. Даны два одномерных массива $a_0, a_1, .., a_n$ и $b_1, b_2, .., b_m$. В первом из них посчитать количество элементов, равных нулю, а во втором - семи.

20. Даны два двумерных массива размера n×m и t×s. Все ли элементы второй строки первого массива кратны g, а элементы третьей строки второго - h.

21. Даны два двумерных массива размера `N×M` и `K×L`. В первом массиве найти максимальный среди первых t элементов p-го столбца, во втором – среди первых q элементов r-го столбца.

22. Даны две строки символов. В первой из них посчитать количество цифр, во второй – количество латинских строчных букв.

23. Даны две строки символов. В первой из них вставить после сочетания `"net"` сочетание `"no"`, а во второй - после сочетания `"da"` вставить `"yes"`.

24. Даны две строки символов. В первой из них удалить сочетания `"da"`, во второй - `"net"`.

25. Даны две строки символов. В первой из них подсчитать количество букв `"a"` и `"o"`, а во второй - `"e"` и `"i"`.

26. Даны две строки символов. В первой из них заменить букву `"a"` на букву`"e"`, а во второй - `"o"` на `"к"`.

7. Программирование с использованием модулей пользователя

Задача 7.1. Дан одномерный массив $a_1, a_2, .., a_n$. Поменять в нем местами максимальный и минимальный элементы. Создать и использовать два модуля, один из которых содержит процедуру для ввода одномерного массива, а другой – процедуру, меняющую местами два элемента массива.

Создаются два модуля: Unit1 и Unit2. В модуле Unit1 описана процедура vvod для ввода одномерного массива a, состоящего из n элементов, в этой процедуре два формальных параметра: n и a (параметры-переменные). В модуле Unit2 описана процедура change, меняющая местами максимальный и мини-

мальный элементы массива, в ней два формальных параметра: n (параметр-значение) и a (параметр-переменная). В программе Project7_1 последовательно вызываются процедуры vvod, change и измененный массив выводится на экран.

Для создания модуля нужно выполнить команду File→ New→ Unit. Модуль Unit1 имеет вид:

```
unit Unit1;     { заголовок модуля}
interface       {интерфейс модуля}
type mas=array[1..20] of real;
var n:integer; a:mas;
procedure vvod ( var n:integer; var a:mas);
implementation      { исполняемая часть модуля}
procedure vvod;
var i:integer;
begin write('n=');readln(n); writeln('vvod a:');
        for i:=1 to n do readln(a[i])
end;  end.                  {конец модуля}
```

Модуль Unit2 имеет вид:

```
unit Unit2;
interface
uses Unit1;                         {используемый модуль}
procedure change (n:integer; var a:mas);
implementation
procedure change;
var max,min,i:integer; c:real;      {max- номер максимального элемента}
begin  max:=1; min:=1;              {min- номер минимального элемента}
          for i:=1 to n do
                begin if a[i]>a[max] then max:=i;
                      if a[i]<a[min] then min:=i;
                end;                { a[max] - максимальный элемент}
                                    { a[min] - минимальный элемент}
```

```
        c:=a[max]; a[max]:=a[min]; a[min]:=c
end;
end.
```

В программе Project7_1 подключены модули SysUtils, Unit1, Unit2. Модуль SysUtils является стандартным модулем Delphi. Модуль Unit1 позволяет получить доступ к переменным n и a, типу mas, процедуре vvod; модуль Unit2 - к процедуре change.

```
program Project7_1;
{$apptype console}
uses
  SysUtils, Unit1,Unit2;
var i: integer;
begin   vvod(n,a);
        change (n, a);
        for i:=1 to n do writeln(a[i]); readln
end.
```

Программу и модули, используемые ею, сохранить на диске в одной папке. Каждый модуль должен быть сохранен в файле, имя которого совпадает с именем модуля, а расширение - .pas, соответственно: Unit1.pas и Unit2.pas.

Задача 7.2. Дан двумерный массив размера n x m. Вычислить сумму элементов массива. Создать и использовать модуль, в котором содержатся процедура для ввода двумерного массива и функция для вычисления суммы элементов массива.

В модуле U1 описана процедура vvod для ввода двумерного массива x, состоящего из n строк и m столбцов, и функция summa, в которой вычисляется сумма всех элементов массива x. В программе P1 вызывается процедура vvod и значение функции summa выводится на экран.

Модуль U1 имеет вид:

```
unit U1;
interface
```

```
    type mas=array[1..10,1..10] of real;
    procedure vvod(var n,m:integer; var x: mas);
    function summa(n,m:integer; x:mas):real;
implementation
    procedure vvod;
    var i,j:integer;
    begin write('n=');readln(n);write('m ='); readln(m);
        for i:=1 to n do for j:=1 to m do readln(x[i,j])
    end;
    function summa;
    var i,j:integer; s:real;
    begin s:=0; for i:=1 to n do
            for j:=1 to m do s:=s+x[i,j];
        summa:=s
    end;
end.
```

Модуль U1 позволяет получить доступ к типу mas, процедуре vvod и функции summa.

```
program P1;
{$apptype console}
uses  SysUtils,  U1;
var n,m:integer; x:mas;
begin vvod(n,m,x);  writeln(summa(n,m,x)); readln
end.
```

Задачи для самостоятельного решения

1. Дан одномерный массив. Все отрицательные элементы этого массива заменить на среднее арифметическое положительных. Создать и использовать модуль, содержащий процедуры ввода и вывода одномерного массива.

2. Дан одномерный массив $a_1, a_2, .., a_m$. Вычислить $\max(a_1^2, \ldots, a_m^2)$. Создать и использовать модуль, содержащий процедуры ввода одномерного массива и поиска максимального элемента.

3. Дан двумерный массив размера n x m. Отрицательные элементы третьей строки увеличить в два раза. Создать и использовать модуль, содержащий процедуры ввода и вывода двумерного массива.

4. Дан двумерный массив размера n x m. Определить сумму элементов второго столбца. Создать и использовать модуль, содержащий процедуру вычисления суммы элементов одномерного массива.

5. Дан двумерный массив размера n x n. Определить сумму элементов, расположенных на главной и побочной диагоналях, с использованием модуля, содержащего функцию вычисления суммы элементов одномерного массива.

6. Дан целочисленный двумерный массив размера n×k. Найти первый нечетный элемент второго столбца. Создать и использовать модуль, содержащий процедуру ввода двумерного массива, и модуль, содержащий процедуру поиска первого нечетного элемента одномерного массива.

7. Дана строка символов. Удалить из нее символы "+", "-", с использованием модуля, содержащего процедуру удаления заданного символа из строки.

8. Создать модуль, содержащий процедуры для выполнения четырех арифметических операций с комплексными числами. Использовать этот модуль в программах для вычисления значений следующих выражений

a) $c = \dfrac{2a + 3b}{4ab}$ b) $c = (a + 2b)(a - 3b)$ c) $c = \dfrac{a}{a + b} - (b - 8)\dfrac{1 - 3a}{b - a}$, где a и b – заданные комплексные числа.

9. Создать модуль, содержащий процедуры ввода одномерного массива, вычисления суммы его элементов, нахождения максимального по абсолютной величине элемента этого массива. Использовать этот модуль для решения следующих задач:

a) вычислить скалярное произведение двух данных векторов по формуле $(a,b)=\sum_{i=1}^{n} a_i b_i$ где a и b – векторы с n компонентами;

b) даны три вектора с различным числом компонент; определить, какой из них имеет наибольшую и наименьшую нормы. Норму вектора вычислять по формуле $\|x\| = \max_{1\le i\le n} |x_i|$, где n – количество компонент вектора x.

8. Работа с файлами

Задача 8.1. Создать типизированный файл, содержащий фамилию, имя, пол и возраст школьников, количество которых задано. Вывести на экран список мальчиков и подсчитать количество школьников старше 10 лет.

Введем обозначения: f – файловая переменная; n –количество школьников; i – порядковый номер школьника; a– запись; a.fam , a.name, a.pol, a.year – поля записи a, содержащие фамилию, имя, пол и возраст школьника; k – количество школьников старше 10 лет.

Входные данные: n,a. **Выходные данные:** k, a.fam, a.name .

Описывается запись uchenic с полями для фамилии, имени, пола и возраста школьника. Типизированный файл состоит из компонент типа uchenic, т.е. из записей. В программе сначала устанавливается связь между файловой переменной f и файлом a.121 корневого каталога диска d, затем открывается новый пустой файл a.121 для записи, при этом указатель текущей компоненты файла указывает на начало файла. С клавиатуры сначала вводится количество школьников, а затем в цикле вся информация об i–м школьнике (поля записи a), которая записывается в типизированный файл с помощью процедуры write(f, a), при этом указатель файла устанавливается в конце очередной записанной компоненты. Таким образом, в типизированном файле d:\a.121 будет информация о n школьниках. Далее этот файл открывается для чтения, при этом указатель файла указывает на его начало. Пока не достигнут конец созданного файла d:\a.121, из него считывается информация о школьнике (запись a) с помощью процедуры read(f,a), при этом указатель файла устанавливается в конце очередной прочи-

танной компоненты. Если считана информация о мальчике, то его фамилия и имя выводятся на экран (a.fam и a.name), если возраст школьника старше 10 лет, то количество таких школьников увеличивается на единицу (k:=k+1). Процедура closefile(f) закрывает соответствующий файл на диске.

```
program Project8_1;
{$apptype console}
uses  SysUtils;
type  uchenic = record
            fam,name:string[30]; pol:char; year:integer;
      end;
var a: uchenic; f: file of uchenic; n,i,k: integer;
begin   assignfile(f, 'd:\a.121');rewrite(f); write('n='); readln(n);
        for i:=1 to n do
            begin readln(a.fam); readln(a.name); readln(a.pol); readln(a.year);
              write(f, a); {записывает в файл a.121 запись a}
            end;
      k:=0; reset(f);  {открывается существующий файл a.121 для чтения}
      while not eof(f) do      {пока не конец файла выполнить}
      begin   read(f,a);      {читает из файла a.121 запись a}
              if a.pol='m' then writeln(a.fam, ' ',a.name);
              if a.year>10 then k:=k+1;
      end;     writeln('k=',k); closefile(f); readln
end.
```

Задача 8.2. Создать текстовый файл, содержащий следующие сведения о жильцах дома: фамилию, место работы, зарплату. Вывести на экран место работы жильца с наибольшей зарплатой.

Введем обозначения: f – файловая переменная; n- количество людей; i – номер человека в файле; fam – фамилия человека; work – место работы человека; zarplata – зарплата человека; max – наибольшая зарплата; workmax – место работы с наибольшей зарплатой.

Входные данные: n, fam, work, zarplata. **Выходные данные:** workmax .

С клавиатуры вводится количество жильцов. Устанавливается связь между файловой переменной f и файлом d:\b.txt, затем этот файл открывается для записи. Вводятся с клавиатуры фамилия, место работы, зарплата и записываются в текстовый файл, с которым связана файловая переменная f (каждая информация записывается в новую строку файла). Таким образом, в файле d:\b.txt будет информация об n людях. Далее этот файл открывается для чтения, при этом указатель файла указывает на его начало.

Из текстового файла считываются фамилия, место работы, зарплата первого человека. Его зарплату можно взять за наибольшую (max:= zarplata) и запомнить место работы (workmax:=work). Далее пока не достигнут конец созданного текстового файла d:\b.txt, из него снова считываются фамилия, место работы, зарплата очередного человека (каждое данное считывается из новой строки файла), при этом если зарплата этого человека больше чем max, то за наибольшую нужно взять его заплату и запомнить место работы (max:= zarplata, workmax:=work).

```
program Project8_2;
{$apptype console}
uses  SysUtils;
var f: textfile; fam, work, workmax:string; n,i :integer; max, zarplata:real;
begin write('n=');readln(n); assignfile(f, 'd:\b.txt'); rewrite(f);
      for i:=1 to n do
      begin readln(fam); readln(work); readln(zarplata);
            writeln(f,fam); writeln(f, work); writeln(f, zarplata)
      end;
      reset(f); readln(f,fam); readln(f, work); readln(f, zarplata);
      max:= zarplata; workmax:=work;
      while not eof(f) do
      begin  readln(f,fam); readln(f, work); readln(f, zarplata);
            if zarplata > max then begin  max:= zarplata; workmax:=work  end
```

end; writeln(workmax); closefile(f); readln;

end.

Задача 8.3. Создать текстовый файл, содержащий в первой строке количество строк (n) и столбцов (m) матрицы, в следующих строках элементы целочисленной матрицы x. Например:

3 4
-5 7 12 2
52 0 13 -8
-3 9 0 -45

Введем обозначения: n – количество строк; m – количество столбцов; i – номер строки; j – номер столбца; x- матрица; f – файловая переменная.

Входные данные: n,m,x. **Выходные данные:** f .

Устанавливается связь между файловой переменной f и файлом d:\a.txt, затем этот файл открывается для записи. С клавиатуры вводится количество строк и столбцов матрицы, они записываются в файл: writeln(f, n,' ',m), здесь числа попадают в одну строку файла, они разделяются пробелом, а строка завершается признаком конца строки. Указатель файла в этом случае устанавливается в первую позицию следующей строки. Далее с клавиатуры вводятся элементы матрицы и записываются в файл: write(f,x[i, j], ' '). При этом элементы одной строки матрицы записываются друг за другом в одной и той же строке файла длиной до 255 символов (если в этой строке хватает места для всех этих элементов), используя в качестве разделителя пробел. Процедура writeln(f) устанавливает указатель файла в первую позицию следующей строки, поэтому элементы следующей строки матрицы записываются в файл с новой строки. Процедура closefile(f) закрывает файл на диске.

```
program Project8_3;
{$apptype console}
uses  SysUtils;
var f: textfile; x:array[1..10,1..10] of integer; n, m, i,j :integer;
begin assignfile(f, 'd:\a.txt'); rewrite(f);
      write('n='); readln(n); write('m=');readln(m);
```

```
        writeln(f, n,' ',m);
        for i:=1 to n do
        begin  for j:=1 to m do
                  begin read(x[i, j]); write(f,x[i, j], ' ')  end;
                  writeln(f)
        end; closefile(f); readln;
end.
```

Задача 8.4. Дан текстовый файл, содержащий в первой строке количество строк (n) и столбцов (m) матрицы, в следующих строках элементы целочисленной матрицы x (как в задаче 5.3). Вычислить произведение ненулевых элементов матрицы.

Введем обозначения: n – количество строк; m – количество столбцов; i – номер строки; j – номер столбца; x- матрица; p – произведение ненулевых элементов; f – файловая переменная.

Входные данные: f. **Выходные данные:** p .

Устанавливается связь между файловой переменной f и файлом d:\a.txt, затем этот файл открывается для чтения. Из файла считываются количество строк и столбцов матрицы, включая признак конца строки файла, указатель файла при этом сдвигается к началу следующей строки. Далее из файла считываются элементы матрицы. После чтения каждого элемента указатель файла сдвигается к началу следующего значения. Процедура readln(f) считывает признак конца текущей строки файла и сдвигает указатель файла к началу следующей строки. Процедура closefile(f) закрывает файл.

```
program Project8_4a;
{$apptype console}
uses  SysUtils;
var f: textfile; x:array[1..10,1..10] of integer; n, m, i,j :integer;p: real;
begin assignfile(f, 'd:\a.txt');
        reset(f); readln(f,n,m); p:=1;
        for i:=1 to n do
```

```
        begin for j:=1 to m do
                begin read(f, x[i, j]);        if x[i,j] <>0 then p:=p*x[i,j] end;
                readln(f)
        end; writeln('p=',p:6:1); closefile(f); readln;
end.
```

Решим эту же задачу с использованием функций eoln(f) и eof(f). Для обработки последовательности значений (элементов массива) можно взять простую переменную x. В этом случае значения n и m не нужны, т.е. нет необходимости считывать информацию из первой строки текстового файла, поэтому с помощью процедуры readln(f) указатель файла сдвигается ко второй строке. Пока не достигнут конец текущей строки файла, из нее считываются элементы массива x, при этом указатель файла сдвигается к следующему элементу, и вычисляется произведение ненулевых элементов. Как только достигли конца строки текстового файла, из файла с помощью процедуры readln(f) считывается признак конца этой строки и указатель файла сдвигается к началу следующей строки. Так просматриваются все строки файла до тех пор, пока не будет достигнут конец файла.

```
program Project8_4b;
{$apptype console}
uses  SysUtils;
var f: textfile; x: integer; p: real;
begin assignfile(f, 'd:\a.txt');
        reset(f); readln(f); p:=1;
        while not eof(f) do
        begin while not eoln(f) do
                begin read(f, x);
                        if x <>0 then p:=p*x
                end; readln(f)
        end; writeln('p=',p:6:1); closefile(f); readln;
end.
```

Задача 8.5. Дан текстовый файл, содержащий следующую информацию:

количество строк
3
количество столбцов
5
матрица
f+h5p
4g6*a
cdef+

Посчитать количество элементов ‘+’ в каждом столбце символьной матрицы, хранящейся в описанном файле.

Введем обозначения: n – количество строк; m – количество столбцов; i – номер строки; j – номер столбца; a- матрица; k – количество элементов ‘+’; s – файловая переменная.

Входные данные: s. **Выходные данные:** k .

Устанавливается связь между файловой переменной s и файлом d:\a.txt, затем этот файл открывается для чтения. Процедура readln(s) считывает признак конца первой строки файла и сдвигает указатель файла ко второй строке, из нее процедурой readln(s,n) считывается количество строк матрицы, включая признак конца строки и указатель файла при этом сдвигается к третьей строке. Аналогично обрабатывается информация из третьей, четвертой и пятой строк файла. Далее из файла считываются элементы матрицы: read(s, a[i, j]). После чтения каждого символа указатель файла сдвигается к следующему символу. Процедура readln(s) считывает признак конца текущей строки файла и сдвигает указатель файла к началу следующей строки. Введенная символьная матрица обрабатывается по столбцам. Процедура closefile(s) закрывает файл.

```
program Project8_5;
{$apptype console}
uses  SysUtils;
var s: textfile; a: array[1..10,1..10] of char; n, m,i,j,k :integer;
begin assignfile(s, ‘d:\a.txt’); reset(s);
      readln(s); readln(s,n);
      readln(s); readln(s,m); readln(s);
```

```
for i:=1 to n do
begin for j:=1 to m do read(s, a[i, j]);  readln(s)  end;
for j:=1 to m do
begin k:=0;
        for i:=1 to n do
                if  a[i, j]='+' then k:=k+1;
        writeln('k=',k: 5);
  end; closefile(s); readln;
end.
```

Задача 8.6. Создать типизированный файл, компонентами которого являются действительные числа. Вычислить абсолютную величину суммы всех компонентов файла.

Введем обозначения: n – количество компонент файла; i – номер компоненты; a- компонента файла; s – сумма компонент; f – файловая переменная.

Входные данные: n,a. **Выходные данные:** abs(s).

```
program Project8_6;
{$apptype console}
uses  SysUtils;
var a, s: real; f: file of real; n,i: integer;
begin   assignfile(f, 'd:\c.125');rewrite(f); write('n='); readln(n);
        for i:=1 to n do
        begin read(a);
                write(f, a); {записывает в файл c.125 число a}
        end;
        s:=0; reset(f);  {открывается существующий файл c.125 для чтения}
        while not eof(f) do      {пока не конец файла выполнить}
        begin read(f,a);      {читает из файла c.125 число a }
                s:=s+a;
        end; writeln('modul s=', abs(s): 6:1);  closefile(f); readln;
end.
```

Задачи для самостоятельного решения

1. Сформировать типизированный файл, содержащий следующие сведения о студентах: фамилию, имя, номер школы, год окончания школы. Удалить из этого файла всю информацию о выпускниках 2003 года школы № 8.

2. Сформировать текстовый файл, содержащий следующие сведения об учащихся: фамилию, имя, возраст, класс. Вывести на экран список учащихся 5 класса и количество учащихся старше x лет (x вводится с клавиатуры).

3. Сформировать типизированный файл, компонентами которого являются целые числа. Найти наименьшую компоненту файла.

4. Сформировать текстовый файл, содержащий следующие сведения об учителях: фамилию, стаж работы, количество детей. Если в файле встречается учитель, имеющий более двух детей, вставить перед информацией об этом учителе сообщение: "семья многодетная", а на экран вывести список учителей, имеющих стаж работы более 10 лет.

5. Сформировать типизированный файл со следующими сведениями: фамилия и адрес проживания. Вывести на экран все сведения из файла и количество людей, проживающих по адресу, введенному с клавиатуры.

6. Сформировать текстовый файл, содержащий следующие сведения о пенсионерах: фамилию, имя, размер пенсии. Вывести на экран фамилию пенсионера, имеющего наименьшую пенсию.

7. Сформировать типизированный файл, содержащий следующие сведения: фамилию, имя, возраст, профессию. Вывести на экран все сведения о врачах, старше x лет (x вводится с клавиатуры).

8. Создать текстовый файл, содержащий следующие сведения о детях: фамилию, имя, рост. Вывести на экран сведения о самом маленьком ребенке.

9. Сформировать типизированный файл, содержащий следующие сведения об учителях: фамилию, имя, стаж работы, количество детей. Вывести на

экран список учителей, имеющих стаж более 25 лет. По введенным с клавиатуры фамилии и имени вывести количество детей.

10. Сформировать текстовый файл со следующими сведениями о больных: фамилия, пол, возраст, диагноз. С клавиатуры вводятся фамилия и возраст. Если в файле есть информация об этом больном, то вывести ее на экран, если нет, то запросить пол и диагноз и добавить новую информацию в файл.

11. Сформировать типизированный файл, содержащий следующие сведения о родителях: фамилию, имя, место работы, зарплату. Проверить, встречаются ли в файле сведения о родителях, работающих в школе, если да, то вывести зарплату таких родителей.

12. Сформировать типизированный файл, содержащий следующие сведения: фамилию, имя, стаж работы, количество детей. Посчитать количество многодетных (более двух детей) и вывести список людей со стажем более 20 лет.

13. Сформировать типизированный файл, содержащий следующие сведения: фамилия, имя, номер школы, год окончания школы. Посчитать количество выпускников школы № 2 и вывести список выпускников 2002 года.

14. Сформировать текстовый файл, содержащий m действительных чисел. Найти сумму наибольшего и наименьшего из чисел файла.

15. Сформировать текстовый файл, содержащий следующие сведения: фамилия, должность, зарплата. Учителям и врачам увеличить зарплату на 30%.

16. Сформировать типизированный файл, содержащий следующие сведения о пенсионерах, количество которых задано:фамилию, имя, размер пенсии. Увеличить в файле пенсии на 15% и вывести всю информацию на экран.

17. Сформировать типизированный файл, содержащий следующие сведения о больных: фамилию, возраст, место проживания (в Казани или нет), диагноз каждого из них. Подсчитать количество не проживающих в Казани и вывести список больных старше x лет с диагнозом y (x и y вводятся с клавиатуры).

18. Сформировать текстовый файл, содержащий следующие сведения об учащихся, количество которых задано: фамилию, имя, возраст, класс. Подсчи-

тать количество учащихся 10 класса и вывести в новый тексовый файл список учащихся старше у лет (у вводится с клавиатуры).

19. Сформировать квадратную матрицу по правилу: $x_{ij} = 2i + j$; i, j=1,2,…,m. Элементы матрицы построчно записать в текстовый файл.

20. Сформировать типизированный файл, содержащий следующие сведения о студентах: фамилия, имя, факультет, средний балл в сессии. Всю информацию о студенте с максимальным средним баллом перенести в начало файла.

9. Указатели

Пример 9.1. Что выводит следующий фрагмент программы:

```
var p1, p2: ^integer; {указатели на переменные типа integer}
begin  new(p1); p1^:=42; p2:=p1;  writeln(p1^, ‘ ’, p2^);
       p2^:=53; writeln(p1^,‘ ’, p2^);
       new(p1); p1^:=88; writeln(p1^, ‘ ’, p2^);
       dispose(p1);  dispose(p2);
end.
```

Решение: Процедура new(p1) выделяет память для динамической переменной типа integer, адрес первого байта этой памяти является значением переменной-указателя p1. В выделенный участок памяти помещается целое число 42 (динамическая переменная p1^ получает значение 42). Указателю p2 присваивается значение указателя p1 (p2:=p1), указатели p1 и p2 указывают на одну и ту же динамическую переменную p1^, поэтому динамическая переменная p2^ получает то же значение, т.е. 42. Оператор p2^:=53 присваивает переменной p2^ значение 53, но тогда и переменная p1^ получит это же значение (p1 и p2 указывают на один и тот же байт памяти). Далее процедура new(p1) выделяет новый участок памяти для динамической переменной p1^ и переменная-указатель p1 получает адрес первого байта этой памяти. В выделенный участок памяти помещается целое число 88 (динамическая переменная p1^ получает значение 88), при этом переменная p2^ остается без изменения. Процедуры dispose освобождают память, занимаемую динамическими переменными.

Результат: 42 42

53 53

88 53

Пример 9.2. Что выводит следующий фрагмент программы:

```
a:=0; p:=@a;          {@a – адрес переменной a }
p^:=10;  writeln(a,' ',p^);   {вывод переменных a и p^ }
```

Решение: Первоначально значение переменной a равно нулю (a:=0), переменная-указатель p получает в качестве значения адрес переменной a (p:=@a), т.е указывает на переменную a. Оператор p^:=10 присваивает переменной a значение 10 (p^ –обращение к переменной, на которую указывает указатель p).

Результат: 10 10

Пример 9.3. Что выводит следующий фрагмент программы:

```
var p1, p2: ^integer; ;        {указатели на переменные типа integer}
begin {выделяется память для динамических переменных типа integer}
   new(p1); new(p2); {p1, p2 указывают на разные переменные}
   {динамические переменные p1^, p2^ получают значения}
   p1^:=10; p2^:=20; writeln(p1^, ' ', p2^);
   p1:=p2; {p1 и p2 указывают на динамическую переменную p2^}
   writeln(p1^, ' ', p2^);     {p1^= p2^ =20}
   p1^:=30; {динамическая переменная p1^ получает значение 30}
   writeln(p1^, ' ', p2^);     {p1^= p2^ =30}
   {освобождается память, занимаемая динамическими переменными}
   dispose(p1);dispose(p2);
end.
```

Решение: Оператор p1:=p2 присваивает указателю p1 значение указателя p2, переменные p1 и p2 содержат адрес первого байта памяти, выделенной для динамической переменной p2^, они указывают на одну и ту же динамическую переменную. Оператор p1^:=30 присваивает динамической переменной p1^ значение 30, тогда и переменная p2^ получит это же значение.

Результат: 10 20

20 20

30 30

Пример 9.4. Что выводит следующий фрагмент программы:

```
var p1, p2: ^integer;
begin   new(p1); new(p2);
    {динамические переменные p1^, p2^ получают значения}
    p1^:=10; p2^:=20;          writeln(p1^, ‘ ’, p2^);
    {динамическая переменная p1^ получает значение переменной p2^ }
    p1^:=p2^; writeln(p1^, ‘ ’, p2^);          p1^:=30; writeln(p1^, ‘ ’, p2^);
    dispose(p1);dispose(p2);
end.
```

Решение: Оператор p1^:=p2^ присваивает переменной p1^ значение переменной p2^ (p1^=20, p2^=20), т.е. динамические переменные p1^ и p2^ имеют одинаковые значения, но переменные-указатели p1 и p2 указывают на разные байты памяти (адреса разные). Оператор p1^:=30 меняет значение динамической переменной p1^, значение переменной p2^ не меняется.

Результат: 10 20

20 20

30 20

10. Использование динамических структур данных

Возможность выделения и освобождения областей памяти позволяет создавать структуры данных с переменным числом элементов – динамические структуры. Наиболее часто используются следующие структуры:

1. Очередь – это структура переменной длины, организованная по принципу: первым вошел – первым вышел. Примером такой структуры является очередь в магазине. Новый покупатель встал (добавился) в конец очереди, а покупатель, который пришел раньше всех и уже купил товар, ушел (удалился) из начала очереди. Итак, у очереди есть голова и хвост. Элемент, добавляемый в очередь, оказывается в её хвосте, удаляемый элемент находился в её голове.

Очередь – упорядоченный набор элементов, в котором извлечение элементов происходит с одного его конца, а добавление новых элементов – с другого.

2. Стек – структура переменной длины, организованная по принципу: последним вошел – первым вышел. Стек можно уподобить стопке книг, из которой взять можно только верхнюю, а новую книгу можно положить тоже только на самый верх. Стек – упорядоченный набор элементов, в котором добавление новых элементов и удаление существующих производится с одного и того же конца, называемого вершиной стека.

3. Список – обобщение очереди и стека. Допускает вставку нового элемента в любое место, исключение любого элемента, доступ к любому элементу. Например, список студентов математического факультета.

Для программирования динамических структур данных используется динамическая память и указатели. Элементы этих структур размещаются в памяти в виде связанной структуры данных. Такое размещение предполагает, что каждый элемент может помещаться в любое свободное место памяти, но при этом ему сопутствует указатель на следующий (или предыдущий) элемент структуры. При исключении элементов структуры или вставки нового элемента все элементы остаются на месте, меняется только одно значение типа указатель. Использование динамических структур позволяет получить большой выигрыш машинного времени, хотя и усложняет программу.

Каждый элемент структуры представляет собой значение типа запись, которое состоит по крайней мере из двух полей: одно из них – указатель, который указывает на следующий (или предыдущий) элемент структуры, остальные поля – непосредственно хранимое значение (информационные поля). При этом началом структуры является указатель, указывающий на первый элемент, а самый последний элемент в качестве значения указателя имеет nil. Это признак конца структуры.

Например:

```
type pzap=^zap;          {тип указатель на данное типа zap }
     zap=record          {тип элемента – запись zap }
```

```
        data:тип элемента;    {информационное поле data }
        next:pzap;            {указатель на следующий элемент}
        end;
    var p:pzap;               {переменная-указатель p }
```

Следует отметить, что идентификатор zap используется до его описания, чем нарушается одно из главных правил Паскаля. Это исключение сделано только для описания указателей, которые могут ссылаться на еще не описанный тип данных для реализации возможности создания связанных структур.

Задача 10.1. Создать стек для хранения целых чисел. Предусмотреть подпрограммы создания пустого стека, добавления нового элемента в стек, доступа к вершине стека и удаления элемента из стека. Записать в этот стек последовательно три числа; удалить элемент из стека и вывести на экран значение элемента в вершине стека.

Введем обозначения: a,b,c – целые числа, записываемые в стек,werh – указатель на вершину стека; i – заданное целое число (информационное поле записи); p- указатель на следующий элемент.

Входные данные: a,b,c. **Выходные данные:** werh^.i.

Для работы со стеком надо определить тип элементов стека и переменную-указатель на вершину стека:

```
type pstack=^stackint;
     stackint =record
     i:integer; p: pstack;
     end;
var werh: pstack;
```

где stackint - тип элементов стека, pstack - тип указатель на переменную типа stackint, werh - переменная-указатель на вершину стека.

Процедура sozdst создает пустой стек, указатель werh получает значение nil (werh – глобальный параметр). Процедура inst вставляет новый элемент (целое число к) в вершину стека. Для этого в памяти выделяется место для переменной типа stackint, адрес первого байта выделенной памяти получает пере-

менная-указатель pnew. Информационное поле переменной pnew^ получит значение k (pnew^.i:=k); поле p – значение werh (адрес элемента, находящегося в вершине стека); переменная werh получает новое значение – адрес выделенной области памяти pnew (новый элемент будет в вершине стека).

Процедура outst извлекает элемент из стека (из его вершины). Для этого переменная-указатель pold получает значение werh (адрес элемента, находящегося в вершине стека); werh получает значение указателя следующего элемента стека (именно этот следующий элемент становится вершиной стека); элемент, ранее находившийся в вершине стека (pold) удаляется из динамической памяти.

Функция werhst позволяет получить доступ к информационному полю (целому числу) элемента, находящегося в вершине стека, именно это число является результатом функции werhst.

В программе вводятся с клавиатуры значения a,b,c, затем последовательно вызываются процедуры: sozdst, создающая пустой стек; затем трижды - inst, добавляющая в стек числа a,b,c; writeln для вывода на экран элемента из вершины стека (c); outst, удаляющая элемент из вершины (c); writeln и функция werhst для вывода на экран элемента из вершины стека (b).

```
program Project10_1;
{$apptype console}
uses  SysUtils;
type pstack=^stackint;
     stackint =record
     i:integer; p: pstack;
     end;
var werh: pstack; a,b,c: integer;
procedure sozdst;            {создание пустого стека}
begin werh:=nil; end;
function werhst: integer;    {доступ к числу, находящемуся в вершине стека}
begin werhst:=werh^.i;       {по адресу werh  ссылка  к полю i} end;
procedure inst(k:integer); {добавление числа k в стек}
```

```
var pnew: pstack;
begin new(pnew);          { pnew получает адрес свободной ячейки памяти}
      pnew^.i:=k;         {по этому адресу  в поле i записывается число к}
      pnew^.p:=werh;      {по адресу pnew  в поле p записывается werh }
      werh:=pnew;         { werh получает новое значение}
end;
procedure outst;     {удаление числа из стека}
var pold: pstack;
begin pold:=werh;
  werh:=werh^.p; {werh получает значение указателя следующего элемента}
  dispose(pold);
end;
begin readln(a,b,c); sozdst; inst(a); inst(b); inst(c); writeln(werhst);
      outst; writeln(werhst); readln
end.
```

Таким же образом можно реализовать и другие динамические структуры, например, очередь и список. В случае очереди так же, как и в случае стека, внутренние элементы структуры не обрабатываются, но здесь нужны две переменные для обозначения головы и хвоста очереди. В случае списка можно реализовать доступ к каждому элементу. Например, для получения элемента с номером t организуется цикл, в котором указатель t раз получает значение поля указателя очередного элемента, таким образом осуществляется передвижение по списку и достигается нужный элемент. Список, в котором обеспечивается связь элемента только со следующим или предыдущим, называется односвязным (однонаправленным).

Примеры описания однонаправленного и двунаправленного списков:

```
type
   ListOne = ^Spisok;       {однонаправленный список}
   Spisok = record
```

```
    Info: Integer;          {информационное поле}
    Next: ListOne;          {указатель на следующий элемент}
  end;
ListTwo = ^SpisokTwo;       {двунаправленный список }
  SpisokTwo = record
    Info: Integer;          {информационное поле}
    Next: ListTwo;          { указатель на следующий элемент}
    Prev: ListTwo;          { указатель на предыдущий элемент}
  end;
```

Задача 10.2. Создать символьный список. Формирование списка завершить при вводе символа ‘*’. Вывести список на экран.

Введем обозначения: p – указатель на первый, при выводе– на очередной элемент списка; y – указатель на новый элемент; x – указатель на последний элемент; data – информационное поле записи (символ); next – поле указатель на следующий элемент; letter - значение элемента.

Входные данные: letter **Выходные данные:** p^.data.

Элемент списка можно добавить в начало, конец, либо между существующими элементами списка. С помощью типа указатель pspis создается связанный однонаправленный список, элементы которого вводятся с клавиатуры и добавляются в его конец. Элементы списка имеют тип запись spis. Сначала список пуст, т.е. p:=nil. С клавиатуры вводится элемент списка (символ) letter. Пока это значение не является символом ’*’, формируется символьный список: выделяется память для нового элемента (y), полям динамической переменной-записи y^ присваиваются значения: указателю - значение nil (y^.next:=nil), а информационному полю - значение введенного символа (y^.data:= letter). Если список пуст, то в него добавляется первый элемент (p:=y), иначе новый элемент добавляется в конец списка (x^.next:=y). Последним становится новый элемент. Адрес последнего элемента списка содержит переменная-указатель x, поэтому ей присваивается значение y (x:=y).

Для вывода списка организуется цикл, в котором используется указатель p, содержащий первоначально адрес первого элемента списка. В теле цикла его значение меняется. Пока значение этого указателя не равно nil, элемент списка (p^.data) выводится на экран, указателю p присваивается значение поля next той записи, на которую указывает p. В результате значением p становится адрес следующего элемента списка. Так указатель перемещается по списку.

```
program Project10_2;
{$apptype console}
uses  SysUtils;
type pspis=^spis;     {указатель на данное типа spis }
     spis =record     {описание типа элемента списка }
     data:char;       {информационное поле}
     next: pspis;     {указатель на следующий элемент}
     end;
var p: pspis;   {указатель на первый элемент списка } x,y:pspis; letter:char;
begin   writeln('введите список'); p:=nil;
        writeln('введите элементы списка. Конец ввода символ *') ;
        read(letter);
        while letter <>'*' do
        begin new(y); y^.next:=nil; y^.data:= letter;
              if  p=nil then p:=y else x^.next:=y;
              x:=y; read(letter);
        end;
        while p<>nil do begin write(p^.data,' '); p:=p^.next end; readln
end.
```

Задача 10.3. Создать очередь, содержащую n целых чисел. Удалить два элемента из очереди и вывести их значения на экран.

Введем обозначения: pob- указатель начала очереди (голова); poe – указатель конца очереди (хвост); data – информационное поле записи (целое число);

next – указатель на следующий элемент; n – количество элементов; i – номер элемента; k – значение элемента очереди; c - удаленный из очереди элемент.

Входные данные: n, k. **Выходные данные:** c.

Добавление элемента в очередь производится так же, как добавление элемента в конец списка, а извлечение элемента из очереди аналогично удалению элемента из начала списка. В процедуре instoch элемент добавляется в очередь, в процедуре outoch - удаляется из очереди. При этом, так как в задаче требуется вывести на экран значение удаляемого элемента, это значение сохраняется в статической переменной c (иначе к уже удаленному из структуры элементу нет доступа).

В программе вводится количество элементов очереди (n); указатели pob и poe получают значение nil (очередь пуста); формируется очередь: с клавиатуры n раз вводится целое число (k) и вызывается процедура instoch, добавляющая это число в конец очереди; дважды вызывается процедура outoch, удаляющая элемент из очереди; удаленные элементы (c) выводятся на экран.

В процедуре outoch для непустой очереди в статической переменной c сохраняется значение удаляемого элемента (c:=pob^.data), после чего этот элемент удаляется из очереди. Для этого переменная-указатель po получает значение pob (адрес начала очереди); pob - значение указателя следующего элемента очереди (этот следующий элемент будет новым началом очереди); из динамической памяти удаляется элемент, ранее находившийся в начале очереди (po).

```
Program Project10_3;
{$apptype console}
uses  SysUtils;
type poch=^och;
     och =record
     data:integer; next: poch;
     end;
var pob, poe: poch; i,k,n,c:integer;
procedure instoch(var pob, poe:poch; k:integer);
```

```
var po:poch; {po- указатель нового элемента очереди}
begin new(po); po^.data:= k; po^.next:=nil;
      if pob=nil then pob:=po else poe^.next:=po;
      poe:=po;
end;
procedure outoch(var pob:poch;var c:integer);
var po:poch; {po- указатель удаляемого элемента очереди}
begin if  pob=nil then writeln('очередь пуста')
                  else begin c:=pob^.data; po:= pob;
                             pob:=pob^.next; dispose(po)
                       end;
end;
begin write('n=');readln(n); pob:=nil; poe:=nil;
      for i:=1 to n do
      begin write('k='); readln(k);  instoch(pob, poe, k) end;
      outoch(pob,c); writeln('удаленный элемент- ',c);
      outoch(pob,c); writeln('удаленный элемент- ',c); readln;
end.
```

Задачи для самостоятельного решения

1. Создать список вещественных чисел. Найти его минимальный элемент.

2. Создать список вещественных чисел. Вычислить их среднее арифметическое.

3. Создать список целых чисел. Встречается ли в нем элемент с заданным значением?

4. Создать список, каждый элемент которого содержит фамилию студента и номер его группы. Подсчитать количество студентов 121 группы.

5. Создать список целых чисел. Удалить из него нечетные числа.

6. Создать очередь, содержащую целые числа. Вычислить сумму и произведение элементов очереди.

7. Создать очередь, содержащую целые числа. Вычислить количество простых чисел в очереди.

8. Создать список целых чисел. После минимального элемента списка вставить элемент с заданым значением.

9. Создать список строк. После к-й строки вставить заданную строку.

10. Создать список строк. Удалить пятую строку списка.

11. Создать список, состоящий из целых чисел. Удалить из него числа, кратные 3.

12. Создать список действительных чисел. Упорядочить элементы списка по возрастанию.

13. Создать очередь, содержащую целые числа. Добавить в очередь число, равное сумме всех элементов очереди.

14. Создать две очереди и объединить их в одну.

15. Создать список и продублировать в нем каждый его элемент так, чтобы равные элементы оказались рядом.

16. Создать список и удалить его элементы, начиная с элемента с номером p и кончая элементом с номером q. В программе предусмотреть случай, когда список содержит менее q элементов.

Литература

1. Аганин А.А., Халитова З.Р., Хисматуллина Н.А. Изучение основ языка программирования Object Pascal. – Казань: ТГГПУ, 2006. - 80 с.

2. Аганин А.А., Халитова З.Р., Хисматуллина Н.А. Задачи по програмированию. Часть I: Программирование базовых алгоритмических структур. – Казань: ТГГПУ, 2009. - 32 с.

3. Аганин А.А., Халитова З.Р., Хисматуллина Н.А. Задачи по програмированию. Часть II: Массивы и строки. – Казань: ТГГПУ, 2009. - 28 с.

4. Аганин А.А., Халитова З.Р., Хисматуллина Н.А. Задачи по програмированию. Часть III: Процедуры и функции, текстовые файлы. – Казань: ТГГПУ, 2009. - 32 с.

5. Культин Н.Б. Delphi 6. Программирование на Object Pascal. СПб: БХВ - Петербург, 2002.

6. Культин Н.Б. Turbo Pascal в задачах и примерах. СПб: БХВ - Петербург, 2002. – 256 с.

7. Немнюгин С.А. Turbo Pascal. Практикум. Программирование на языке высокого уровня. – СПб.:Питер, 2003. -496 с.

8. Фаронов В. В. Delphi 6: учебный курс. – СПб.:Питер, 2002. - 512 с.

9. Фаронов В. В. Turbo Pascal 7.0: практика программирования. – М.:КноРус, 2011. - 414 с.