**Практическое задание №26**

**Тема: Разработка про­граммы с использованием одномерных массивов на языке Паскаль**

*Задание:*

1. Прочитать текст «Одномерные массивы», изучите примеры.
2. Составить блок-схемы к примерам решения задач (2.1, 2.2., 2.3, 2.4).
3. Записать блок-схемы и примеры в тетрадь (2.1, 2.2., 2.3, 2.4).

*Порядок выполнения:*

**Задание №1. Одномерные массивы**

**1.1.  Понятие «массив»**

С понятием «массив» приходится сталкиваться при решении научно-технических и экономических задач обработки совокупностей большого количества значений. В общем случае *массив* – это структурированный тип данных, состоящий из фиксированного числа элементов, имеющих один и тот же тип.

Название *регулярный тип* (или ряды) массивы получили за то, что в них объединены однотипные (логически однородные) элементы, упорядоченные (урегулированные) по индексам, определяющим положение каждого элемента в массиве.

*Массив* – это составной объект, образованный из элементов (*компонент*) одного и того же типа. Такой тип данных применяется в программировании для обозначения объектов, аналогичных числовым последовательностям в математике, где сразу целая группа чисел обозначается одним именем (чаще всего буквой), а для обращения к каждому отдельному числу данной последовательности используются различные *индексы* (номера элементов). В математике это может выглядеть, например, так:

*a1, а2, а3, …, аn.*

Таким образом, в программировании *массив* – это последовательность однотипных элементов, имеющих общее имя, причем каждый элемент этой последовательности определяется порядковым номером (индексом) элемента.

Примеры массивов:

* *Х1, Х2, …, Хn –* одномерный массив, состоящий из *n* элементов;
* *А0, А1, А2, …, А10 –* одномерный массив,состоящий из 11 элементов.

Массивы бывают одномерными  (один индекс), двумерными (два индекса) и т.д.

Структура массива всегда однородна. Массив может состоять из элементов типа *integer , real* или *char* , либо других однотипных элементов. Другая особенность массива состоит в том, что к любой его компоненте можно обращаться произвольным образом. Программа может сразу получить нужный ей элемент по его порядковому номеру (индексу). Номер элемента массива называется ***индексом.*** Индекс – это значение порядкового типа, определенного, как ***тип индекса*** данного массива. Тип индекса определяет границы изменения значений индекса.

Рассмотрим массив *Х1, Х2, …, Хn.*  Здесь

* *Х* – имя массива;
* *1, 2,…, n –* индексы (порядковые номера) элементов;
* *Х7 –* седьмой элемент массива *Х.*

**1.2. Описание одномерных массивов**

*Одномерный массив* – это фиксированное количество элементов одного типа, объединенных одним именем, причем каждый элемент имеет свой уникальный номер, и номера элементов идут подряд. Для описания подобных объектов в программировании предварительно следует ввести соответствующий тип в разделе описания типов.

Тип *массив* описывается следующим образом:

*Type*

*Имя типа = Array [тип индекса (ов)] Of  тип элементов;*

*Var*

*Имя переменной: имя типа;*

Переменную типа *массив* можно описать сразу в разделе описания переменных Var:

*Var   Имя переменной: array [тип индекса (ов)] Of  тип элементов;*

Здесь

* *Array* – служебное слово (в переводе с английского означает «массив»);
* *Of* – служебное слово (в переводе с английского означает «из»).
* Тип индекса – любой порядковый тип, кроме типов *integer, longint.*
* Тип же самих элементов может быть любым, кроме *файлового* типа.
* Количество элементов массива называется его *размерностью****.*** Несложно подсчитать, что при последнем способе описания множества индексов *размерность массива равна: максимальное значение индекса – минимальное значение индекса + 1.*

Например:

*Type*

*mas = array [1..20] of  real;*

*Var  X: mas;*

Массив *Х* – одномерный, состоящий из двадцати элементов вещественного типа. Элементы массива хранятся в памяти компьютера последовательно друг за другом.

При использовании переменных для обозначения индекса их значения к моменту использования должны быть определены, а в случае арифметических выражений их результат не должен выходить за границы минимального и максимального значения индексов массива.

Индексы элементов массива могут начинаться с любого целого числа, в том числе и отрицательного, например:

**Type** bb = **Array** [-5..3] **Of** Boolean;

Массивы данного типа будут содержать 9 логических переменных, пронумерованных от -5 до 3.

**1.3. Ввод и вывод одномерных массивов в Паскале**

Ввод и вывод массивов осуществляется поэлементно.

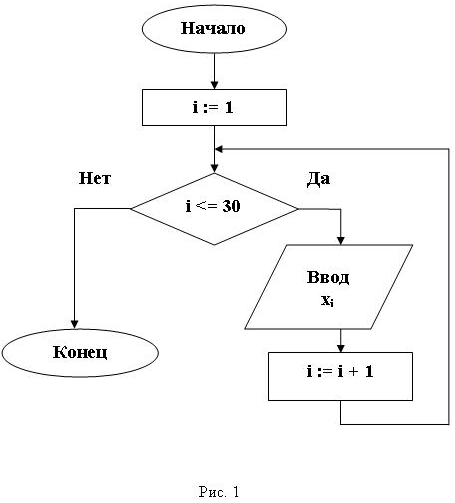
Введем одномерный массив *Х*, состоящий из 30 элементов, то есть необходимо ввести некую последовательность элементов *Х1, Х2, …, Х30.*

Пусть i *–* индекс (порядковый номер) элемента в массиве *Х*. Тогда

*Хi –* i-й элемент массива *Х*, где *i = 1, 2, …, 30.*

Для ввода массива можно использовать любой цикл.

*Первый вариант:* ввод массива с использованием цикла с предусловием (*Рисунок 1*).



**Program** Primer\_1;

**Var** i: integer;

     X: **array** [1..30] **of** Integer;

**Begin**

     i := 1;

**While** i <= 30 **Do**

**Begin**

            Read (X[i]); i := i + 1

**End;**

       Readln

**End.**

Можно ввести с клавиатуры элементы следующим образом:

-7 \_ 4 \_ -2 \_ 0 \_ 12 \_ -1 \_ -5 \_  9 \_ 11 \_ -3 \_ -5 \_ … \_15, то есть через пробел ввести в строчку и нажать клавишу Enter.

Можно было ввести элементы в столбец,   отделяя элементы клавишей Enter, то есть каждый элемент с новой экранной строки:

-7 Enter

 4 Enter

-2 Enter

  …

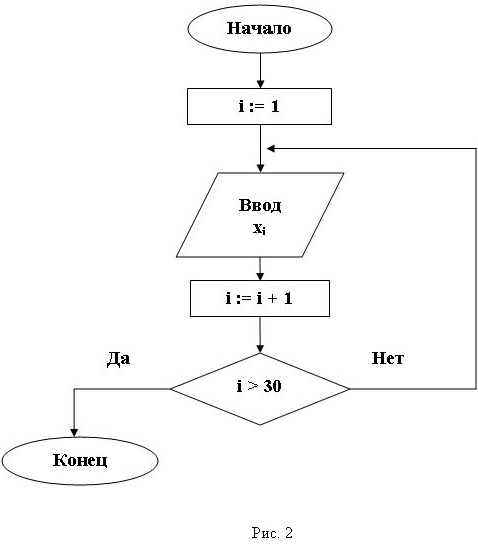
  …

  …

 15 Enter

В первом варианте ввод массива *Х* в программе осуществляется с использованием цикла с предусловием.

*Второй вариант:* ввод массива с использованием цикла с постусловием (*Рисунок 2*).



**Program** Primer\_2;

**Var** i: integer;

     X: **array** [1..30] **of** Integer;

**Begin**

     i := 1;

**Repeat**

           Read (X[i]); i := i + 1

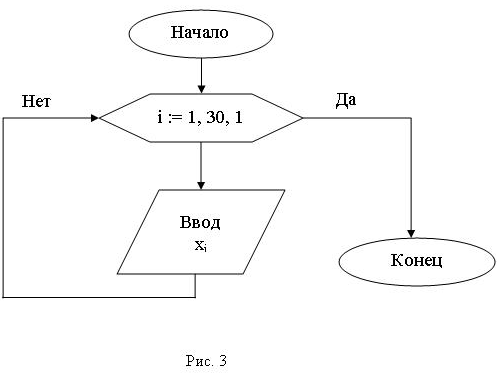
**Until** i > 30;

       Readln

**End.**

Массив X введен с использованием цикла с постусловием.

*Третий вариант:* ввод массива с использованием цикла с параметром (*Рисунок 3*).



**Program** Primer\_3;

**Var** i: integer;

     X: **array** [1..30] **of** Integer;

**Begin**

**For** i := 1 **To** 30 **Do** Read (X[i]);

     Readln

**End.**

Массив вводится с помощью цикла с параметром, где в качестве параметра используется индекс элемента массива (i).

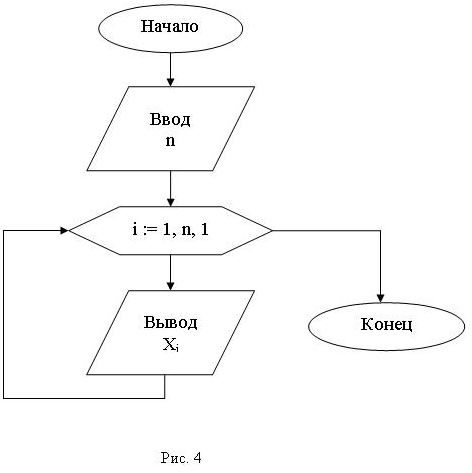
*Вывод одномерного массива* осуществляется аналогично.

В программе вместо операторов Read или Readln используются операторы Write или Writeln. Но просто заменить одни операторы на другие здесь недостаточно. Для того чтобы выводимые значения не сливались между собой, надо явным образом вставлять между ними разделитель – пробел или перевод строки. Приведем два возможных способа вывода массива:

* **For** i := 1 **To** n **Do** Write (X[i],’ ‘);
* **For** i := 1 **To** n **Do** Writeln (x[i]).

На первый взгляд второй способ может показаться более простым и удобным, но это далеко не всегда так. Результат работы такой программы зачастую неудобно, а то и просто невозможно анализировать. Ведь каждый элемент массива будет располагаться в отдельной строке, следовательно, мы не сможем увидеть более 25 элементов одновременно. Кроме того, очень часто массив требуется распечатать дважды, чтобы сравнить состояние массива до обработки и результат его обработки. В этом случае сравнение состояний массива гораздо удобнее проводить, если они распечатаны в двух соседних строках, а элементы выровнены по столбцам, то есть к варианту 1 должна быть добавлена еще и форматная печать (указано количество позиций, которое должно отводиться на печать одного элемента).

Например, выведем одномерный массив *Х1, Х2, …, Хn*, состоящий из элементов вещественного типа, используя цикл с параметром (*Рисунок 4*):



**Program** Primer\_4;

**Const** n = 30;

**Var** i: Integer;

     X: **Array** [1..n] **Of** Real;

**Begin**

**For** i:= 1 **To** n Do **Write** (X[i] : 6 : 2,  ‘ ‘);

          Writeln; {курсор переводит на новую строку}

        Readln

**End.**

**Задание №2.  Примеры решения задач**

**2.1. Вычисление суммы и произведения элементов массива, удовлетворяющих заданному условию.**

**Задача 1.** Дан целочисленный одномерный массив, состоящий из n элементов. Найти сумму и произведение нечетных элементов, кратных 3.

***Введем обозначения***: n – количество элементов в массиве; А – имя массива; i – индекс элемента массива; Ai – i-й элемент массива A; s – сумма нечетных элементов массива, кратных 3; p – произведение нечетных элементов массива, кратных 3.

***Входные данные: n, A.***

***Выходные данные: s, p.***

Первоначально сумма искомых элементов равна нулю: не просуммировано ни одно слагаемое, то есть s:=0. Далее, используя любой оператор цикла, просматриваем весь массив от первого и до последнего элемента. И если при этом элемент массива нечетный и кратен 3, то к уже накопленной сумме добавляется очередное слагаемое, т.е. s:= s + A[i]. 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  - ее новое, вычисляемое значение.

При просмотре массива можно сразу вычислить и произведение элементов массива, удовлетворяющих заданному условию. Произведение вычисляется с помощью оператора p:=p\*A[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 искомое произведение равно единице, т.е. p:=1.

При решении этой задачи можно использовать любой из видов циклов. Рассмотрим несколько вариантов решения задачи.

***Первый способ***.  Для решения используется цикл с параметром:

**Program** Primer1\_1;

**Var** A: **Array**[1..20] **Of** Integer;

      i, n, s, p: Integer;

**Begin**

     Write (‘n=’); Readln (n);

**For** i:=1 **To** n **Do** Readln (A[i]);   {ввод массива}

     s:= 0;  p:=1;

**Fo**r i:=1 **To** n **Do**   {обработка массива}

**If** (A[i] **mod** 2 <>0) **and** (A[i] **mod** 3 = 0) **Then**

**Begin**

          s:=s+A[i];  p:= p\*A[i]

**End;**

      Writeln (‘s=’, s, ‘p=’, p);

      Readln

**End.**

***Второй способ***. Для решения используется цикл с предусловием:

**Program** Primer1\_2;

**Var** A: **Array**[1..20] **Of** Integer;

      i, n, s, p: Integer;

**Begin**

     Write (‘n=’); Readln (n); i:=1;

**While** i <= n **Do**

**Begin**

      Readln (A[i]);   i:= i + 1

**End;**

     s:= 0;  p:=1; i:=1;

**While** i<=n  **Do**

**Begin**

**If** (A[i] **mod** 2 <>0) **and** (A[i] **mod** 3 = 0) **Then**

**Begin**

          s:=s+A[i]; p:= p\*A[i]

**End;**  i:= i + 1

**End;**

      Writeln (‘s=’, s, ‘p=’, p);

      Readln

**End.**

**2.2. Нахождение количества элементов, удовлетворяющих заданному условию**

**Задача 2.** Дан массив целых чисел. Найти количество тех элементов, значения которых положительны и не превосходят заданного натурального числа А.

***Введем обозначения:*** n – количество элементов в массиве; X – имя массива; *i* – индекс элемента массива; X*i* – *i*-й элемент массива X; А – заданное число; k – количество элементов, значения которых положительны и не превосходят заданного числа А.

***Входные данные:*** n, X, A.

***Выходные данные:*** k.

Вводим с клавиатуры значение числа А. Количество элементов, значения которых положительны и не превосходят заданного числа А, вначале полагаем равным нулю, то есть k:=0. Если очередной элемент массива положителен и не превосходят заданного числа A, то количество таких элементов нужно увеличить на единицу, то есть k:=k + 1. Таким образом, обрабатываются все элементы массива.

При решении этой задачи можно использовать любой из видов циклов. Рассмотрим несколько вариантов решения задачи.

***Первый способ***.  Для решения используется цикл с параметром:

**Program** Primer2\_1;

**Var** X: **Array**[1..20] **Of** Integer;

      i, n, k, A: Integer;

**Begin**

     Write (‘n=’); Readln (n);

**For** i:=1 **To** n **Do** Readln (X[i]);   {ввод массива}

      Write (‘A=’);  Readln (A); k:= 0;

**Fo**r i:=1 **To** n **Do**   {обработка массива}

**If** (X[i] >0) **and** (X[i] <= A) **Then** k:=k + 1;

      Writeln (‘k=’, k);

      Readln

**End.**

***Второй способ.*** Для решения используется цикл с постусловием:

**Program** Primer2\_2;

**Var** X: **Array**[1..20] **Of** Integer;

      i, n, k, A: Integer;

**Begin**

     Write (‘n=’); Readln (n); i:=1;

**Repeat**

           Read (X[i]); i := i + 1

**Until** i > n;

      Write (‘A=’); Readln (A); k:= 0; i:=1;

**Repeat**

**If** (X[i] >0) **and** (X[i] <= A) **Then**

          k:=k + 1; i := i + 1

**Until** I > n;

      Writeln (‘k=’, k);

      Readln

**End.**

**2.3. Нахождение номеров элементов, обладающих заданным свойством**

**Задача 3.** Найти номера четных элементов, стоящих на нечетных местах.

***Введем обозначения***: n – количество элементов в массиве; X – имя массива; i – индекс элемента массива; Xi – i-й элемент массива Х.

***Входные данные:*** n, X.

***Выходные данные:*** i.

Необходимо просмотреть весь массив. Если просматриваемый элемент является четным, а его порядковый номер – нечетный, то вывести его номер. При решении этой задачи можно использовать любой из видов циклов.

Составим программу с помощью цикла с параметром:

**Program** Primer3\_1;

**Var** X: **Array**[1..20] **Of** Integer;

      i, n:  Integer;

**Begin**

     Write (‘n=’); Readln (n);

**For** i:=1 **To** n **Do** Readln (X[i]);   {ввод массива}

**Fo**r i:=1 **To** n **Do**   {обработка массива}

**If** (X[i] **mod** 2 = 0) **and** (i **mod** 2<>0) **Then** Write (i:5);

      Readln

**End.**

**2.4.** **Найти номер последнего отрицательного элемента массива.**

***Введем обозначения***: n – количество элементов в массиве; А – имя массива; i – индекс элемента массива; Ai – i-й элемент массива A; m – номер последнего отрицательного элемента массива.

***Входные данные:*** n, A.

***Выходные данные:*** m.

Последний отрицательный элемент массива – это первый отрицательный элемент, который встретится при просмотре массива с конца. Если очередной элемент не является отрицательным, то нужно уменьшать значение текущего индекса (i:=i-1), пока он не станет меньше номера первого элемента или не будет найден отрицательный элемент. Переменная m получает значение i (номер отрицательного элемента), т.е. m := i. Если в массиве нет отрицательного элемента, то значение переменной m остается равным нулю.

**Program** Primer3\_2;

**Const** n=10;

**Var** A: **Array**[1..n] **Of** Integer;

      i, m: Integer;

**Begin**

**For** i:=1 **To** n **Do** Readln (A[i]);   {ввод массива}

      m := 0;  i:=n;

      While  (i >= 1) **and** (A[i] >=0) **Do** i:=i-1;    m:=i;

       Writeln (‘m=’, m);

      Readln

**End.**