Массивы

Ещё о типах данных

Типы, определяемые пользователем

Программист может объявить свой собственный тип данных. По существу это будет определение идентификатора, который можно будет дальше использовать в коде программы в качестве имени типа наряду со стандартными типами языка Паскаль. Естественно, определение пользовательского типа данных должно предшествовать его использованию.

Синтаксис определения пользовательского типа следующий

type *newTypeName* = *type1;*

Здесь *newTypeName* — идентификатор, обозначающий имя определяемого программистом типа данных. Если *type1* — существующий тип данных, то *newTypeName* будет псевдонимом типа *type1*. Компилятор их не будет различать.

Рассмотрим пример.

```
Program Example1;
Type int = SmallInt;
     big = LongInt;

Var a,b : int;
    res : big;

Begin
  WriteLn('Введите 2 целых числа');
  Read(a,b);
  res:=a*b;
  WriteLn(a,' x ',b,' = ',res)
End.
```

В этом примере объявленный пользователем тип *int* является псевдонимом типа *SmallInt*, а тип *big* — псевдонимом *LongInt*.

Этот пример может давать ошибку, если в качестве типа результата вычислений будет использован *Int64*. Для остальных типов он будет работать корректно. Как сказано в документации по Delphi[1], арифметические операции над целыми числами дают результат типа *Integer*, который эквивалентен *LongInt*. Результат имеет тип *Int64*, только если хотя бы один из операндов типа *Int64*. Чтобы даже в этом случае программа работала правильно, преобразуем её.

```
Program Example2;
Type int = LongInt;
     big = Int64;

Var a,b : int;
    res : big;

Begin
  WriteLn('Введите 2 целых числа');
  Read(a,b);
  res:=big(a)*b;
  WriteLn(a,' x ',b,' = ',res)
End.
```

[1] Help к Borland Developer Studio 2006 (BDS2006)

Для преобразования типа первого множителя использовано выражение big(a). В данном случае это эквивалентно Int64(a). Но приведённый приём с типами пользователя позволяет в случае необходимости просто изменить определение типов в разделе *type*, не изменяя остальную часть программы.

Типы-диапазоны

Тип-диапазон представляет собой подмножество значений другого порядкового типа, называемого базовым типом. Чтобы определить тип-диапазон, нужно указать минимальное и максимальное значение базового типа, между которыми пишутся две точки подряд. Пробел между точками не допускается. Минимальное значение не может быть больше максимального.

<мин. значение> .. <макс. значение>

Поскольку вещественные типы не относятся к порядковым, для них тип-диапазон определён быть не может. Не следует также думать, что минимальное и максимальное значение в определении типа-диапазона можно брать сколь угодно большими по модулю. Они должны быть допустимыми хотя бы в одном базовом типе, поэтому всегда ограничены.

Примеры определения типа-диапазона и его использования при описании переменных.

```
Type Digit = 0..9;
     Cent = -100..100;
Var  c : Digit;
     D : Cent;
```

Тип-диапазон не обязательно описывать в разделе описания типов. Того же эффекта, что в примере выше, можно было достигнуть и так:

```
Var  c : 0..9;
     D : -100..100;
```

В документации FreePascal сказано, что некоторые предопределённые целочисленные типы данных определены как типы-диапазоны.

Массивы

Описание статических[2] массивов

Структурированный тип — это тип, содержащий множество значений в одной переменной. Одним из таких типов является массив. Массивом называется конечная именованная последовательность однотипных величин, именуемых элементами массива. Доступ к элементу массива осуществляется по индексу, который записывается в квадратных скобках рядом с именем массива. Например, a[5] — элемент массива a с индексом 5.

Тип одномерного (иначе называемого линейным) статического массива определяется следующим образом.

```
Type имя_типа_массива = array[тип_индекса] of тип_элементов_массива;
```

В качестве типа индекса можно использовать любой порядковый тип. Чаще всего используют тип-диапазон. Примеры описаний массивов.

```
Type  TMyArr = array[Byte] of Real;
      TIntArr = array[-5..5] of Integer;
      TA = array[1..100] of ShortInt;

Var a,b : TMyArr;
      c : TIntArr;
```

[2] В Delphi и FreePascal есть ещё динамические массивы. О них речь пока не идёт.

```
    d : TA;
```

Здесь массивы *a* и *b* содержат элементы вещественного типа, которые индексируются переменной типа byte. Поскольку значение переменной типа byte может изменяться в пределах от 0 до 255 включительно, то и элементы этих массивов будут нумероваться в указанных пределах, а память будет выделена для каждого из этих массивов под 256 элементов. Массив *c* будет содержать значения типа Integer, а нумерация элементов массива определена от -5 до +5 включительно.

Вовсе не обязательно было вводить новый тип данных для массива. Например, массив *c* можно было описать следующим образом:

```
Var c : array[-5..5] of Integer;
```

Однако при написании подпрограмм[3] обработки массивов без определения нового типа данных для массива не обойтись.

При описании массива можно задать начальные значения его элементам. Они указываются в круглых скобках через запятую. Перед открывающей скобкой после типа элементов ставится знак равенства. Например,

```
Var c : array[1..5] of Integer=(10,20,30,40,50);
```

При этом первый элемент массива станет равным 10, второй 20, …, пятый 50. Некоторые считают, что при описании переменной, в том числе и массива, она по умолчанию получает значение 0. Это неверно. Вот цитата из документации по FreePsacal: «By default, variables in Pascal are not initialized after their declaration. Any assumption that they contain 0 or any other default value is erroneous: They can contain rubbish.»[4]

Многомерные массивы — это массивы массивов, т.е. массивы, элементами которых являются другие массивы. Например,

```
type TMatrix = array[1..10] of array[1..50] of Real;
```

Это эквивалентно следующему описанию

```
type TMatrix = array[1..10, 1..50] of Real;
```

В этом примере описан двумерный массив. Его часто представляют как таблицу, строки которой пронумерованы от 1 до 10, а столбцы от 1 до 50. Одномерный массив часто представляют как таблицу, состоящую только из одной строки или только из одного столбца. В Паскале также допускаются массивы большего числа измерений, но они используются очень редко.

Операции над массивами

Все действия над массивами выполняются поэлементно. Единственным действием над статическим массивом как целым является присваивание.

Присваивание массивов

Если два массива *a* и *b* **одного и того же типа**, то один из них можно присвоить другому: *b*:=*a*. При этом массив *b* станет точной копией массива *a*[5]. Вот пример.

```
Program Example3;
Const N=5;
Type
   TMyArray = array[1..N] Of Integer;
Var
   a : TMyArray = (10,20,30,40,50);
```

[3] О подпрограммах речь будет идти на одном из следующих занятий.
[4] Reference guide for Free Pascal, version 2.4. Пункт 4.4 «Initialized variables»
[5] Для динамических массивов это неверно.

```
   b : TMyArray;
   i : Integer;
Begin
  b:=a;  //Копирование массива a в массив b

  For i:=1 To N Do //Вывод массива b
  Write(b[i],' ');

  WriteLn //Чтобы опустить курсор на новую строку
End.
```

Заметьте, что оба массива должны быть одного и того же типа. Если описать массивы так

```
Var a : array[1..100] of Integer;
b : array[1..100] of Integer;
```

то при попытке b:=a компилятор Delphi выдаст ошибку несовместимости типов. Поведение компилятора FreePascal в этом случае зависим от режима работы. В режиме совместимости с Delphi тоже будет ошибка. Ошибки не будет, если для массива определить свой тип данных или описать массивы так

```
Var a,b : array[1..100] of Integer;
```

Ввод и вывод линейных (одномерных) массивов

Ввод и вывод массивов осуществляется поэлементно. Обычно это делается в цикле.

В следующем примере вводится одномерный массив a из N элементов, а затем выводится в обратном порядке. Известно, что $N \leq 1000$, а его элементы — целые числа, не превосходящие по модулю 10^6.

Массив описываем из максимально возможного в этой задаче количества элементов, т.е. из тысячи, а использовать будем N первых из них. Тип элементов массива — integer. Чтобы программу легко было перестроить на другое максимальное количество элементов, опишем константу NMAX. В дальнейшем для перестройки на другое предельное число элементов нужно будет только изменить значение этой константы.

```
{$mode delphi}
Program Example4;
Const NMAX=1000;

Type Massiv = array[1..NMAX] Of Integer;

Var i,N : Integer;
    a   : Massiv;
Begin
  WriteLn('Задайте количество элементов массива');
  Read(N);

  //Проверка введённого значения N
  If (N > NMAX) or (N <= 0) Then
  Begin
    WriteLn('Количество элементов должно быть от 1 до ', NMAX);
    halt  //Прекращение выполнения программы
  End;

  //Ввод массива
  WriteLn('Введите ', N, ' целых чисел');
  For i:=1 To N Do
  Read(a[i]);

  //Вывод массива в одну строку
```

```
  For i:=N DownTo 1 Do
  Write(a[i],' ');

  WriteLn //Перевести курсор на новую строку
End.
```

Процедура halt, использованная в этой программе, вызывает прекращение её работы и передачу управления вызвавшей программе.

Ввод двумерных массивов

Пусть задан массив из N строк и M столбцов.

```
Var a : array[1..100,1..100] of Real;
```

Для его ввода применяют вложенные циклы.

```
…
For i:=1 To N Do
For j:=1 To M Do
Read(a[i,j])
```

Вывод двумерных массивов

Для вывода массива в виде таблицы также применяют вложенные циклы.

В следующем примере формируется двумерный массив из случайных вещественных чисел из диапазона $0 \leq x < 1$, а потом выводится на экран. Точность вывода — 5 знаков после запятой. Для генерации случайных чисел применяем функцию random. Чтобы при каждом запуске программы случайные, точнее, псевдослучайные, последовательности чисел генерировались разные, перед первым вызовом random вызываем процедуру randomize. Она устанавливает стартовую точку генератора псевдослучайных чисел по системным часам, т.е. случайным образом.

```
{$mode delphi}
Program Example5;
Const NMAX = 10;
      MMAX = 10;

Type Massiv = array[1..NMAX,1..MMAX] Of Real;

Var a   : Massiv;
    N,M : Integer;
    i,j : Integer;
Begin
  WriteLn('Введите количество строк и столбцов массива');
  Read(N,M);

  If (N<1) or (N>NMAX) or (M<1) or (M>MMAX) Then
  Begin
    WriteLn('Неверно задана размерность массива');
    halt //Прерывание выполнения программы
  End;

  Randomize;  //Инициализация генератора случайных чисел

  //Заполнение массива псевдослучайными числами
  For i:=1 To N Do
  For j:=1 To M Do
  a[i,j]:=Random;

  //Вывод массива
  For i:=1 To N Do
  Begin
```

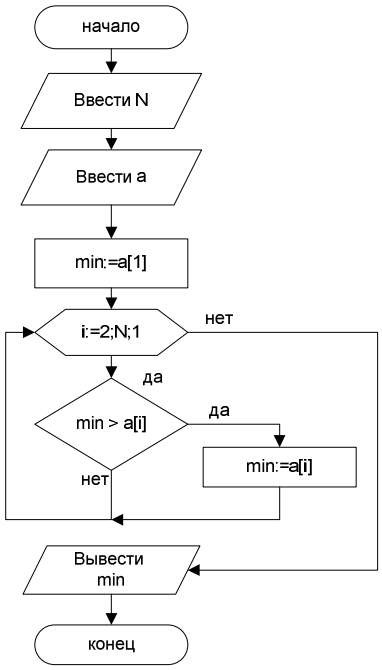
```
    //Вывод одной строки
    For j:=1 To M Do
    Write(a[i,j]:7:5,' ');
    //Переход на следующую
    WriteLn
  End
End.
```

Задачи на обработку массивов

1. Поиск минимального элемента массива

Найти минимальный элемент массива из N чисел.

Для решения этой задачи сначала предположим, что самый первый элемент массива и является минимальным. Для массива из одного элемента так оно и есть. Если бы элементов массива было два, нужно было бы сравнить предполагаемый минимальный элемент (первый) со вторым, и, если второй окажется меньше, то теперь минимальным считать его. Потом таким же образом добавить третий элемент, четвёртый и т.д. Получаем следующий алгоритм.



На этой блок-схеме для краткости ввод массива обозначен одним блоком «Ввести a», хотя в программе это надо делать в цикле. Программа для массива из не более 1000 целых чисел типа integer приведена ниже.

```
{$mode delphi}
Program MinElem;
Const NMAX = 1000;
Type TIntArray = array[1..NMAX] Of Integer;
Var a       : TIntArray;
    N       : Integer;
    i, min  : Integer;
Begin
  WriteLn('Введите количество элементов массива');
  Read(N);

  If (N<1) or (N>NMAX) Then
  Begin
    WriteLn('Неверно задана размерность массива');
    halt
  End;
```

```
  //Ввод массива
  WriteLn('Введите ',N,' целых чисел');
  For i:=1 To N Do
  Read(a[i]);

  //Собственно поиск минимального элемента
  min:=a[1];
  For i:=2 To N Do
  If min > a[i] Then
  min:=a[i];

  WriteLn('Минимальный элемент ', min)
End.
```

С тем же успехом для решения этой задачи можно было пройти массив в обратном порядке или взять в качестве начального значения min любой элемент массива и в цикле теперь уже по *всем* возможным значениям индекса, а не от двух, сравнить его с остальными элементами. Можно было бы также в качестве начального значения min взять любое заведомо завышенное, т.е. не меньшее возможных значений элементов массива, число. В данном случае таким числом является максимально возможное значение числа типа integer. В языке Паскаль для обозначения этого числа используется константа MaxInt. То есть можно было бы написать

```
//Собственно поиск минимального элемента
  min:=MaxInt;
  For i:=1 To N Do
  If min > a[i] Then
  min:=a[i];
```

Кто-то может заметить, что нет необходимости в одном цикле вводить массив, а в другом его обрабатывать для поиска минимального элемента. Так оно и есть. Просто целью этого примера было продемонстрировать именно алгоритм поиска минимального элемента массива. А если бы нужно было найти минимальное число из N вводимых с клавиатуры, то и массив никакой не был бы нужен. Попробуйте справиться с такой задачей сами.

2. «Популярные» цифры

Какие цифры в десятичной записи натурального числа N используются чаще всего? $N \leq 9 \cdot 10^{18}$ Лидирующих нулей в записи числа нет, т.е. запись типа 001 невозможна.

Примеры входных и выходных данных

Вход	**Выход**
525375	5
12323	2 3
123	1 2 3

Решение. Заметим, что это число, хоть оно и велико, попадает в диапазон допустимых значений для типа Int64. Для подсчёта количества вхождений каждой из десяти возможных различных цифр в записи этого числа будем использовать массив *a*, элементы которого пронумеруем от 0 до 9. Элемент *a[i]* будет показывать, сколько раз в записи числа встретилась цифра *i*.

1. Вначале все *a[i]* должны быть равны нулю.
2. Разбирая число на цифры при помощи целочисленного деления (div и mod), заполним этот массив.
3. Найдём в массиве *a* максимальный элемент. Обозначим его *max*.
4. Учтём, что элементов, равных *max*, может быть несколько. Пройдя в цикле по массиву ещё раз, выведем все *i*, для которых *a[i]=max*.

```
{$mode delphi}
Program Popular;
```

```
Type Digit = 0..9;

Var a   : array[Digit] Of Byte;
    i   : Digit;
    N   : Int64;
    max : Byte;
Begin
  Write('N=');
  Read(N);

  (* А вдруг кто-то путает целые числа с натуральными? *)
  If N<=0 Then
  Begin
    WriteLn('Это число не натуральное');
    Halt
  End;

  (* Очищаем массив *)
  For i:=0 To 9 Do
  a[i]:=0;

  (* Разбираем число и заполняем массив *)
  While N > 0 Do //Пока в записи N есть хоть одна цифра
  Begin
    i:= N mod 10; //Последняя цифра в текущей записи числа N
    a[i]:=a[i]+1; //Сосчитали: цифра i встретилась ещё раз
    N:=N div 10   //"Отрезали" последнюю цифру от N
  End;

  (* Ищем максимальный элемент массива a *)
  max:=a[0];
  For i:=1 To 9 Do
  If max<a[i] Then
   max:=a[i];

  (* Теперь ищем все цифры, встречающиеся в записи числа max раз *)
  For i:=0 To 9 Do
  If a[i]=max Then
    Write(i,' ');

  WriteLn
End.
```

Почему для элементов массива выбран тип byte? В записи числа N не может быть больше 19 цифр, значит и максимальное значение элемента массива равно 19. Это соответствует случаю, когда все цифры числа одинаковы.

3. Сумма элементов массива

Найти сумму положительных элементов целочисленной матрицы из M строк и N столбцов, не превышающих заданного числа K. ($M, N \leq 100$, элементы массива и число K не превышают по модулю 10^5.)

Решение. Напомним, что матрицей в математике называется прямоугольная таблица чисел. Будем просто проверять каждый элемент массива на соответствие диапазону [0,K] и при выявлении такого соответствия наращивать сумму: sum:=sum+a[i,j]. До начала прохода по массиву значение sum должно быть равно нулю. Оценим верхний предел получаемого результата. Пусть M, N и K имеют максимально возможное значение. Пусть также максимально возможное значение имеют все элементы массива. Тогда $sum = 10^5 \cdot 10^2 \cdot 10^2 = 10^9$. Это число может храниться в переменной типа integer.

```
{$mode delphi}
Program Task3;
```

```
Const NMAX=100;
      MMAX=100;
Var a       : array[1..MMAX,1..NMAX] Of Integer;
    M, N, K : Integer;
    i, j    : Integer;
    sum     : Integer;
Begin
  Write('Количество строк = ');
  Read(M);
  Write('Количество столбцов = ');
  Read(N);

  If (M<0) or (M>MMAX) or (N<0) or (N>NMAX) Then
  Begin
    WriteLn('Недопустимая размерность массива');
    Halt
  End;

  WriteLn('Задайте построчно матрицу из ',M,' строк и ',N,'
столбцов');
  For i:=1 To M Do
  For j:=1 To N Do
  Read(a[i,j]);

  Write('K=');
  Read(K);

  (* Собственно подсчёт суммы *)
  sum:=0;
  For i:=1 To M Do
  For j:=1 To N Do
  If (a[i,j]>0) and (a[i,j]<=K) Then
   sum:=sum+a[i,j];

  WriteLn('Искомая сумма ', sum)
End.
```

При решении задачи можно было бы ещё проверять на соответствие оговоренному в условии диапазону вводимые значения переменной K и элементов массива. Надеюсь, вы сможете справиться с этим сами.

Задачи для самостоятельного решения

1. Дан линейный массив из N действительных чисел ($N \leq 100$). Переставить все его элементы в обратном порядке. Например, из массива (1,2,3,4,5) получить массив (5,4,3,2,1). Второй массив не использовать.
2. Найти среднее арифметическое элементов линейного массива из N целых чисел. $N \leq 50$, элементы массива по модулю не превосходят 10^6.
3. Есть ли в десятичной записи двух натуральных чисел M и N одинаковые цифры? $M, N \leq 9 \cdot 10^{18}$.