Методические указания

Лабораторная работа 11

Одномерный массив. Работа с элементами.

Описание массивов

Общие сведения

Обобщим полученные ранее сведения о массивах.

По определению ***массив*** – это *статическая* структура данных, которая представляет собой *однородную,* фиксированную по размеру и конфигурации совокупность элементов, *упорядоченных* по номерам (*индексам*).

Рассмотрим каждое из выделенных в определении свойств массива:

- *статическая* структура данных означает, что взаиморасположение и взаимосвязи элементов всегда остаются постоянными;
- *однородность* означает, что все элементы массива одного типа;
- элементы массива *упорядочены* (пронумерованы), и обратиться к каждому из них можно по индексу элемента (или по нескольким индексам, если массив многомерный).

Кроме того, *имя* массива является общим для всех его элементов.

Характеристики массива:

- *тип* – общий тип элементов массива;
- *размерность (ранг)* – количество индексов массива;
- *диапазон изменения индекса (индексов)* – определяет количество элементов в массиве.

Одномерный массив (вектор) – это массив, в котором элементы нумеруются одним индексом (т.е. ранг равен единице).

В памяти компьютера все элементы массива занимают одну непрерывную область (массив), отсюда и произошло это название.

Описание массива

Синтаксис для описания одномерного массива:

```
var
  ИмяМассива : array[НижнГран..ВерхГран] of ТипЭлементов;
```

В данном случае пользовательский тип данных определен непосредственно в разделе описания переменных `var`. Такой тип называется *анонимным,* т.к. не имеет имени.

Более строгим считается предварительное описание типа-массива:

```
type
  ИмяТипа = array[НижнГран..ВерхГран] of ТипЭлементов;
var
  ИмяМассива : ИмяТипа;
```

Такое описание типа-массива считается хорошим стилем программирования, оно **необходимо** при использовании имени массива в качестве параметра процедуры или функции.

В качестве индекса массива (НижнГран**..**ВерхГран) в общем случае используется диапазон значений **порядкового типа**. Чаще всего индекс массива имеет тип integer, реже char или boolean. Но возможны и следующие описания:

```
type
  month = (Jan,Feb,Mar,Apr,May,Jun,Jul,Aug,Sep,Oct,Nov,Dec);
  income : array[month] of real;
var
  company1, company2 : income;
```

тогда переменная company1[Oct] будет обозначать прибыль первой компании за октябрь.

Возможно даже такое описание:

```
var  arr : array[byte] of real;
```

В этом случае массив arr состоит из 256 элементов, а индекс изменяется от 0 до 255 (диапазон типа byte).

Выход за границы массива

Поскольку в качестве индексов элементов массива используются переменные и выражения, то возможна ситуация выхода за границы массива. Например:

```
var  a : array[1..100] of integer;
begin
  a[0]:=10;      {выход за границы массива}
  a[200]:=20;    {выход за границы массива}
end.
```

Ошибка выхода за границы массива выдается на этапе компиляции, только если включена ***директива компилятора*** {R+}. Если эта директива отключена (а по умолчанию она как раз отключена), то программа откомпилируется, но в ходе ее выполнения ***выход за границы массива может привести к непредсказуемым ошибкам в программе*** в связи с некорректным доступом к соседним ячейкам памяти.

Поэтому чтобы получать сообщения об ошибке выхода за границы при отладке программ с массивами рекомендуется использовать директиву {R+}, включая ее в самом начале программы:

```
{R+}
var  a : array[1..100] of integer;
begin
  a[200]:=20; {при компиляции появится сообщение об ошибке}
end.
```

Действия над массивами

Операции

Над массивом, как целым допускается только одна операция – массивы **одного типа** (типа **с тем же именем**) могут быть операндами в операторе присваивания. Идентичный по структуре тип, имеющий такое же описание, но другое имя, не подходит. Пример:

```
type
  mas = array[1..100] of real;
  vec = array[1..100] of real;
var
  a, b : mas;
  c : vec;
begin
  ...
  a:=b; {массив a – точная копия массива b}
  a:=c; {появится сообщение об ошибке несовместимости типов}
  ...
end.
```

Поэтому ***все операции по обработке массивов следует проводить поэлементно,*** используя циклические конструкции.

Типичные задачи обработки массивов

К типичным задачам обработки относятся:

- заполнение массива данными;
- вывод элементов массива;
- поиск элементов с заданным значением;
- подсчет количества элементов, удовлетворяющих некоторому условию;
- поиск максимального (минимального) элемента и его номера (или всех таких элементов, если их несколько);
- изменение значений элементов;
- удаление элементов, со сдвигом оставшихся влево (т.е. по направлению к элементам с меньшим индексом);
- вставка элементов, со сдвигом оставшихся вправо;
- упорядочивание (***сортировка***) элементов массива по значению некоторого признака и другие.

Удаление и вставка элементов

Так как элементы массива располагаются в соседних ячейках памяти, то без сдвига можно удалять только последний элемент или добавлять новый элемент в конец массива. Во всех остальных случаях придется ***сдвигать*** элементы массива.

Оставшиеся ненужными (например, после удаления элементов) ячейки в конце массива ***обнуляются,*** т.е. заполняются нулями.

Пример. Удаление и вставка элементов в начале массива:

```
const  n=10;
type  mas=array[1..n] of integer;
var  a:mas;
     i,b:integer;
```

```
begin
  ...
  {Удаление первого элемента массива:}
  for i:=1 to n-1 do a[i]:=a[i+1];
  a[n]:=0; {обнуление ненужной последней ячейки}
  ...
  {Вставка элемента в начало массива:}
  write('Введите значение добавляемого элемента:',b);
  readln(b);
  for i:=n downto 2 do a[i]:=a[i-1];
  a[1]:=b;
  ...
end.
```

Массивы как параметры подпрограмм

Передача массивов в подпрограммы

Для использования массива в качестве параметров процедур и функций необходимо заранее **описать тип массива** в разделе объявления типов. Причем тип фактического и формального параметров должен быть **один и тот же.**

Пример. Инициализация (обнуление) и вывод на экран одномерного массива:

```
const n=10;
type  TVector=array[1..n] of integer;

procedure init(var a:TVector); {массив как параметр-переменная}
  var i:integer;
  begin
    for i:=1 to n do a[i]:=0;
  end;

procedure print(a:TVector); {массив как параметр-значение}
  var i:integer;
  begin
    for i:=1 to n do write(a[i]:6,' ');
    writeln;
  end;

var  v: TVector;
begin
  init(v);  {инициализация массива}
  v[3]:=123;  v[5]:=2345;  v[n]:=34;
  print(v);  {вывод массива на экран}
end.
```

Открытые параметры-массивы

Для передачи в процедуры и функции массивов различной размерности (и с различным количеством элементов) используются ***открытые параметры-массивы.*** В этом случае, при описании параметра-массива в заголовке процедуры или функции, индексы массива не указываются:

```
procedure ИмяПроцедуры(ИмяМассива:array of ТипЭлементов);
```

Такое определение возможно **только** при описании формальных параметров подпрограмм.

Все формальные параметры-массивы в рамках подпрограммы автоматически описываются как ***одномерные массивы с нулевой базой*** (нулевой нижней границей) указанного в заготовке типа. То есть приведенное выше описание параметра-массива аналогично следующему:

```
ИмяМассива:array[0..N-1] of ТипЭлементов;
```

где **N** – количество элементов массива, указанного в качестве **фактического параметра** при вызове подпрограммы `ИмяПроцедуры`.

Для определения реального количества элементов фактического параметра используются следующие функции:

`high(ИмяМассива)`	для обычного массива возвращает **верхнюю границу** массива, для открытого – **максимальное** значение индекса
`low(ИмяМассива)`	для обычного массива возвращает **нижнюю границу** массива, для открытого – **минимальное** значение индекса

Изменим процедуры из предыдущего примера для работы с массивами различного размера:

```
const n=10;
      m=15;
type  TVector1=array[1..n] of integer;
      TVector2=array[1..m] of integer;

procedure init(var a:array of integer);
  var i:integer;
  begin
    for i:=low(a) to high(a) do a[i]:=0;
  end;

procedure print(a:array of integer);
  var i:integer;
  begin
    for i:=low(a) to high(a) do write(a[i]:6,' ');
    writeln;
  end;

var  v1: TVector1;
     v2: TVector2;
begin
  init(v1);  {инициализация массива из n=10 элементов}
  init(v2);  {инициализация массива из m=15 элементов}
  v1[3]:=123;  v1[5]:=2345;  v1[n]:=34;
  v2[2]:=45;  v2[8]:=567;  v2[12]:=6789;
  print(v1);  {вывод массива из n=10 элементов на экран}
  print(v2);  {вывод массива из m=15 элементов на экран}
end.
```

Содержание