МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

Юго-Западный государственный университет

(ЮЗГУ)

Кафедра вычислительной техники

РАБОТА С МАССИВАМИ

Методические указания по выполнению лабораторной работы

для студентов направления подготовки 09.03.01

Курск 2016

УДК 621.3
Составитель: Э.И. Ватутин

Рецензент
Кандидат технических наук, доцент *В.С. Панищев*

Работа с массивами: методические указания по выполнению лабораторных работ по дисциплине «Программирование» / Юго-Зап. гос. ун-т; сост.: Э.И. Ватутин; Курск, 2016. 13 с.

Методические рекомендации содержат сведения по разработке программ с использованием массивов на современных языках программирования высокого уровня.

Предназначены для студентов направления подготовки 09.03.01 «Информатика и вычислительная техника».

Текст печатается в авторской редакции

Подписано в печать _____. Формат 60x84 1/16.
Усл. печ. л. Уч. – изд.л. Тираж 30 экз. Заказ . Бесплатно.
Юго-Западный государственный университет
305040, Курск, ул. 50 лет Октября, 94.

Содержание

Введение

Целью работы является получение практических навыков при работе с массивами различного типа.

Достаточно часто в процессе хранения и обработки данных приходится сталкиваться с необходимостью хранения не одиночных значений в виде переменных простых типов, а целых наборов или таблиц значений. Для этого обычно используются структуры данных, именуемые *массивами*. Они представляют из себя именованные последовательности однотипных элементов, причем в качестве типа элемента массива может выступать как простой тип, так и практически любой другой тип, за исключением файлового. Каждый массив обладает *размером* (количество элементов либо объем занимаемой памяти) и *размерностью* (количество индексов).

Размер массива теоретически не ограничен, на практике однако он ограничивается объемом доступной памяти (2 ГБ для данных программы в 32-х разрядных версиях Windows). Размерность массива не ограничена, однако на практике очень редко применяются массивы размерности более трех. В учетом введенного понятия размерности массивы подразделяются на *одномерные* и *многомерные* (двумерные, трехмерные, четырехмерные и т.д.).

Если размер массива известен заранее, то массив называется *статическим*. Память под него резервируется во время компиляции и не изменяется в процессе выполнения программы (массив имеет один и тот же адрес и размер). Если на этапе компиляции размер массива не, то такой массив называется *динамическим* и память под него выделяется в процессе выполнения программы по мере необходимости. Во время разных запусков программы подобный массив может располагаться в памяти под разными адресами.

Динамические массивы можно размещать с использованием стандартных средств – *динамические массивы, поддерживаемые компилятором*, либо вручную с использованием механизма динамической

памяти – *динамические массивы, размещаемые вручную*. Первый вариант является наиболее предпочтительным, т.к. позволяет выявлять большое количество ошибок на более ранних этапах. Сфера применения второго варианта совместно с массивами в настоящее время существенно ограничена, однако он может быть применен, например, при обработке списков.

Массивы являются представителями структурных типов, которые могут быть объявлены непосредственно при определении переменной или константы (секция `var` или `const`), либо при определении типов (секция `type`). В дальнейших разделах рассмотрены типовые приемы использования массивов различных типов.

Массивы данного типа записываются в программе следующим образом:

```
array [Дипазон_изменения_индекса] of Тип_элементов
```

В качестве диапазона может выступать любой порядковый тип (обычно используются ограниченные типы). Тип элементов может быть любым программно доступным типом, кроме файлового. Приведем пример:

```
type
  TArr = array [10..20] of Integer;
```

В примере в обращение вводится тип `TArr`, являющийся массивом целых чисел с номерами позиций с 10 до 20 включительно.

Тип элементов должен быть определен заранее и не может совпадать с вводимым в обращение типом, т.е. приведенная ниже конструкция будет ошибочной.

```
type
  TErrorArr = array [1..10] of TErrorArr;  // Ошибка: рекурсивные определения типов запрещены!
```

Достаточно удобной является возможность создавать массивы с произвольными номерами элементов, не обязательно начинающимися с единицы или нуля. В языке Си, например, такой возможности нет.

Размер статического массива должен быть известен на этапе компиляции. В приведенных выше примерах он составляет 11 и 10 элементов соответственно $(20-10+1=11, 10-1+1=10)$. Благодаря этому правилу нельзя указывать переменные в качестве границ диапазона изменения индекса:

```
var
  N: Integer;
  A: array [1..N] of Integer;  { Ошибка: размер массива не определен! }
```

При объявлении переменной или константы типа массив можно задать ее начальное значение путем перечисления элементов массива через запятую, указываемого в круглых скобках через запятую:

```
var
  A: array [1..5] of Byte = (10, 20, 30, 40, 50);
```

Количество указываемых значений должно в точности соответствовать количеству элементов в массиве, иначе будет выдано сообщение компилятора об ошибке.

Над массивами целиком разрешена только операция копирования, выполняемая с использованием оператора присваивания и внешне ничуть не отличающаяся от копирования значения переменной любого другого типа. Все остальные необходимые действия осуществляются при помощи поэлементного доступа, для чего применяется операция индексации:

```
Имя_массива[Номер_элемента]
```

Результатом применения операции индексации является значение отдельного элемента массива с указанным индексом. В качестве индекса в общем случае может быть использовано выражение, совпадающее по типу с индексом,

указанным при определении массива. Например, для нахождения суммы всех элементов введенного выше массива A можно записать следующее выражение:

```
S := A[1] + A[2] + A[3] + A[4] + A[5];
```

Обычно в подобных ситуациях при обработке массивов применяются циклы, что будет неоднократно рассмотрено ниже.

При обработке массивов необходимо внимательно следить за тем, чтобы индекс массива находился в указанных в описании типа массива рамках. При обращении к несуществующему элементу массива программа будет работать некорректно: возможно возникновение ошибок типа «Access Violation» (нарушение доступа), возникающих при обращении к памяти, не принадлежащей адресному пространству процесса, ошибочное изменение значений переменных, располагающихся в памяти в непосредственной близости от массива и т.д. На этапе отладки рекомендуется установка опции Range checking или использование в тексте программы директивы компилятора `{$R+}`. Как уже отмечалось выше, по завершении отладки подобные проверки лучше отключать.

Правилом хорошего тона является занесение размера массива в константу, объявленную в разделе `const`.

Динамические массивы, поддерживаемые компилятором, описываются следующим образом:

```
array of Тип_элементов;
```

(по сравнению со статическими массивами отсутствует описание индексов). Память под подобные массивы выделяется в процессе выполнения программы путем вызова процедуры

```
procedure SetLength(var A: Динамический_массив; NewLength: Integer);
```

В качестве первого параметра процедура принимает динамический массив, вторым параметром указывается новая длина массива. Данную процедуру можно использовать как для выделения памяти, так и для изменения количества элементов динамического массива. Перед использованием динамического массива необходимо обязательно выделить под него память.

Индекс динамических массивов может быть только целочисленным, причем индексация всегда начинается с нулевого элемента.

После выделении памяти под динамический массив `X` путем вызова `SetLength(X, N)` в массиве будут содержаться элементы `X[0]`, `X[1]`, … `X[N−1]`. Элемента `X[N]` в массиве нет, а попытка обращения к нему является достаточно распространенной ошибкой.

Текущую длину массива можно узнать при помощи функции `Length`, возвращающей размер массива:

```
function Length(A: Динамический_массив): Integer;
```

По завершении работы с массивом память освобождается автоматически при выходе из текущей области видимости (подпрограммы или основной программы), что является достаточно удобным и защищает от возможных утечек памяти. В случае необходимости принудительного освобождения памяти (например, если в программе используется несколько массивов размером в несколько сотен мегабайт, то может возникнуть нехватка оперативной памяти и память из-под освободившихся динамических массивов желательно освободить как можно ранее) наиболее быстрым способом является следующий:

```
X := nil;
```

(`X` – динамический массив). Значение `nil` является указателем, который никуда не указывает.

Если при решении поставленной задачи можно обойтись без использования динамических массивов, то необходимо поступать именно так, т.к. операции выделения (англ. alloc), освобождения (англ. free) и изменения размера (англ. realloc) области динамической памяти не очень быстры. По этой же причине рекомендуется избегать частых изменений длины динамического массива, т.к. эта операция может «съесть» б**о**льшую часть времени выполнения фрагмента программы.

Пример программы

Задача. Найти и сохранить в массиве все делители заданного числа.

Решение поставленной задачи может быть представлено в следующем виде.

```
var
  X, I: Integer;
  Divisors: array of Integer;

begin
  { Ввод числа }
  Write('X = ');
  Readln(X);

  { Сканирование делителей }
  for I := 1 to X-1 do
    if X mod I = 0 then begin
      SetLength(Divisors, Length(Divisors)+1); { Увеличение размера массива на 1 }
      Divisors[High(Divisors)] := I;           { Запоминание найденного делителя }
    end;

  { Вывод результата }
  Writeln(#13#10'Divisors: ');
  for I := 0 to High(Divisors) do
    Write(Divisors[I], ' ');

  { Освобождение памяти (необязательно) }
  Divisors := nil;

  Readln;
end.
```

Индивидуальные задания

1. Информация о среднесуточной температуре за месяц (30 дней) задана в виде массива. Определить, сколько дней температура была ниже среднемесячной.

2. Имеются данные о погоде за месяц (30 дней): дождливо, пасмурно, солнечно. Определить, сколько дней было солнечно и сколько дней шел дождь. Для хранения данных о погоде использовать перечислимый тип.

3. Задан массив *A*, длина массива вводится с клавиатуры. Определить сумму и количество отрицательных элементов массива.

4. Имеются результаты наблюдения погоды за месяц (30 дней) в виде двух массивов. В одном содержится температура, в другом – количество осадков. Определить количество выпавшего снега. Считать, что снег выпадает только при отрицательной температуре.

5. Задан массив вещественных чисел. Вычислить сумму четных элементов и произведение нечетных.

6. Рост учеников класса представлен в виде массива, причем рост мальчиков кодируется положительными значениями, девочек – отрицательными. Определить средний рост мальчиков и девочек.

7. Результаты переписи населения хранятся в виде массива, содержащего даты рождения. Вывести номера людей, родившихся раньше заданного года.

8. Ртутные термометры используются для измерения температуры до –39,4 °С. Определить, можно ли использовать такие термометры в Курской области, если имеются данные о минимальной температуре за последние 10 лет.

9. Имеется информация о направлении ветра за месяц (30 дней) в виде следующих значений: северный, южный, западный, восточный, северо-западный, юго-западный, северо-восточный, юго-восточный. Определить какой процент дней дул северный и юго-западный ветер. Для кодирования направления ветра использовать перечислимый тип.

10. В области 10 районов. Для каждого из них известны посевная площадь (до 200 га) и средняя урожайность (50 ц/га). Определить среднюю урожайность в области и количество выращенной пшеницы.

11. В школе N классов (задается с клавиатуры). Определить общее количество учащихся в школе и вывести классы, в которых количество учащихся превосходит заданное количество.

12. Для сборки изделия используется N типов деталей (задается с клавиатуры) различной стоимости. В состав изделия может входить до 5 однотипных деталей. Определить общее количество деталей и суммарную стоимость изделия.

13. Имеется два отсортированных массива A и B. Необходимо объединить данные массивов в один массив C с сохранением отсортированности.

14. Упорядочить массив целых чисел в порядке возрастания количества делителей элементов.

15. С клавиатуры вводится два больших числа (количество цифр больше 20), найти их сумму. Числа представлять в виде массивов цифр.

16. С клавиатуры вводится два больших числа (количество цифр больше 20), найти их произведение. Числа представлять и обрабатывать в виде массивов цифр.

17. С клавиатуры вводится большое число A (количество цифр больше 20) и целое число n. Определить значение A^n. Числа представлять и обрабатывать в виде массивов цифр.

18. С клавиатуры вводится два больших числа (количество цифр больше 20), найти их частное и остаток от деления. Числа представлять и обрабатывать в виде массивов цифр.

19. Дан массив целых чисел. Определить число уникальных (не совпадающих) чисел и максимальное число совпадающих. Оформить программу так, чтобы при необходимости тип обрабатываемых данных можно было легко изменить (например, с целых чисел на вещественные).

Например, для массива «1, 2, 1, 3, 4, 5, 1, 2, 7» число уникальных чисел – 6 (числа 1, 2, 3, 4, 5, 7), а максимальное число совпадающих – 3 (число 1).

20. Дан массив неотрицательных целых чисел. Расположить элементы массива так, чтобы нулевые элементы располагались правее ненулевых, используя при этом минимальное число обменов. Дополнительный массив не использовать. Например, массив «1 0 2 0 4 2 5 0 0 0 7» может быть преобразован к виду «1 7 2 5 4 2 0 0 0 0 0».

Содержание отчета

1. Титульный лист.
2. Индивидуальное задание.
3. Краткое описание стратегии решения.
4. Листинг программы.
5. Тестовые примеры, результаты тестирования.
6. Выводы.

Контрольные вопросы

1. Для чего применяются массивы?
2. Какие типы массивов существуют?
3. Что такое размерность и размер массива? Чем отличаются указанные понятия?
4. В чем отличия массивов, хранимых в статической и динамической памяти?
5. Что такое операция индексации?
6. Что происходит при обращении к несуществующему элементу массива?

Библиографический список

1. Емельянов С.Г., Ватутин Э.И., Панищев В.С., Титов В.С. Процедурно-модульное программирование на Delphi: учебное пособие. М.: Аргамак-Медиа, 2014. 352 с.
2. Зотов И.В., Ватутин Э.И., Борзов Д.Б. Процедурно-ориентированное программирование на C++: учебное пособие. Курск: КурскГТУ, 2008. 211 с.