№ 193

МИСиС

Е.В. Сигитов

Информатика

Методы программирования и структуры данных

Учебное пособие

ФЕДЕРАЛЬНОЕ АГЕНТСТВО ПО ОБРАЗОВАНИЮ

№ 193

ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ
МОСКОВСКИЙ ИНСТИТУТ СТАЛИ
И СПЛАВОВ
МИСиС

Кафедра инженерной кибернетики

Е.В. Сигитов

Информатика

Методы программирования и структуры данных

Учебное пособие

Рекомендовано редакционно-издательским
советом университета

Москва Издательский Дом МИСиС 2008

УДК 004.4
С34

Р е ц е н з е н т
профессор *Е.А. Калашников*

Сигитов Е.В.
С34 Информатика. Методы программирования и структуры данных: Учеб. пособие. – М.: Изд. Дом МИСиС, 2008. – 105 с.

Учебное пособие посвящено использованию языка Си и выбору структур данных (стеков, очередей, деревьев, графов) при решении различных задач.

В первой части пособия дается подробное описание языка Си, приводятся примеры, раскрывающие излагаемый материал.

Во второй части рассмотрены методы программирования, использующие основные динамические структуры данных, различные способы их представления в памяти и алгоритмы обхода, реализованные на языке Си.

В третьей части рассмотрены алгоритмы методов сортировки и реализация на языке Си.

Содержит контрольные вопросы для самопроверки и задания, направленные на приобретение навыков составления программ для задач, использующих структуры данных.

Соответствует программе курса «Методы программирования и структуры данных».

Предназначено для студентов специальности 230401 «Прикладная математика».

ОГЛАВЛЕНИЕ

1. ПРОГРАММИРОВАНИЕ НА ЯЗЫКЕ СИ

1.1. Структура программы

При решении задач из различных областей используются разнообразные структуры данных. Сочетание правильного выбора расположения информации (структуры данных) и современного алгоритмического языка высокого уровня позволяет создавать эффективные алгоритмы и программы.

Программа начинается перечислением директив подключения файлов из библиотек, необходимых при выполнении программы. Директива подключения файла из стандартной библиотеки имеет вид `#include <`*`имя файла`*`>`

Директива `#include <stdio.h>` (подключение файла из стандартной библиотеки, содержащей функции, которые выполняют ввод – вывод) присутствует практически в любой программе.

Далее следуют директивы определения `#define` (см. ниже); раздел описаний, где должны быть объявлены глобальные переменные; определения функций и, наконец, главная функция `main()`. Выполнение программы начинается с главной функции `main`.

1.2. Переменные. Типы данных. Выражения

Для именования объектов программы используются *идентификаторы*. Идентификатор содержит буквы, цифры и знак подчеркивания и начинается с буквы. Для внутренних переменных значащими является первый 31 символ. Большие и малые буквы различаются. Ключевые слова `if`, `else`, `struct` и т.п. зарезервированы и набираются малыми буквами.

Переменная – это именованная область памяти, в которой хранятся данные определенного типа. У переменной есть имя (идентификатор) и значение. Имя служит для обращения к области памяти, в которой хранится значение. Перед использованием любая переменная должна быть описана. Описание переменной задает тип, класс памяти и по умолчанию область действия, т.е. в какой части программы ее можно использовать.

Базовыми типами являются `char` (символьный, занимает в памяти 1 байт), `int` (целый, занимает в памяти 2 байта или 4 байта в зависимости от компьютера), `float` (вещественный, занимает в

памяти 4 байта), `double` (двойной точности, занимает в памяти 8 байтов). Базовые типы могут использоваться с квалификаторами: `short` и `long` применительно к целым, `signed` (со знаком) или `unsigned` (без знака) – к типу `char` или любому целому типу, `long` – к типу `double`.

Перечисленные типы (называемые стандартными) можно использовать для описания переменных непосредственно:

```
float x,y;
short m;
```

З а м е ч а н и е. Последнее аналогично `short int m;` (`int` обычно опускается).

Строки символов не выделяются как самостоятельный тип данных, а рассматриваются как массивы символов.

Логический тип также не выделяется в качестве самостоятельного типа данных, а любое ненулевое значение трактуется как “истина”, нулевое значение – как “ложь”. При проверке истинности различных условий формируется значение 1, если условие выполняется, и 0 – в противоположном случае. При явном использовании констант `true` и `false` их необходимо определить директивами:

```
#define  true   1
#define  false  0
```

В любой программе требуется проводить вычисления. Для вычисления значений используются *выражения*, которые состоят из операндов, знаков операций и скобок. Операнды задают данные для вычислений. Операции задают действия, которые необходимо выполнить. Каждый операнд является константой или переменной или, в свою очередь, выражением. Операции выполняются в соответствии с приоритетами. Для изменения порядка выполнения операций используются круглые скобки. В выражение могут входить операнды любого типа, и тогда выполняется преобразование типов. По смыслу выражение можно подразделить на арифметическое и логическое.

В арифметическом выражении используются операции +, –, *, /, % (остаток от деления для целых). Результат любой из этих операций будет целый, если оба операнда целые, в остальных случаях результат вещественный.

Заметим, что в Си отсутствует операция возведения в степень. Возведение в любую степень, например x^y, выполняется функцией `pow(x,y)`, которая расположена в файле `math.h`.

При вычислении арифметических выражений действуют следующие правила приведения типов:

1) любые операнды типа char, unsigned или short преобразуются к типу int;

2) если один из операндов имеет тип long double, то другой преобразуется к типу long double;

3) если пункт 2 не выполняется, а один из операндов имеет тип double, то другой преобразуется к типу double;

4) если пункты 2 и 3 не выполняются, а один из операндов имеет тип float, то другой преобразуется к типу float;

5) если пункты 2–4 не выполняются, а один из операндов имеет тип long, то другой преобразуется к типу long;

6) иначе оба операнда имеют тип int.

Тип результата тот же, что тип участвующих в выражении операндов.

Тип результата вычисления выражения можно выразить и явно:

(*тип*) *выражение*.

Например, `n/k` и `(double)n/k` имеют разный результат при целых `n` и `k`, так как во втором выражении дробная часть результата при делении не отбрасывается.

В Си очень большое количество разнообразных операций, поэтому для правильного применения операций важно знать их приоритет (перечислены в порядке убывания):

1) () [] . –>; 2) * & – ! ~ ++ -- sizeof; 3) * / %; 4) + –; 5) << >>; 6) < > <= >=; 7) = = !=; 8) &; 9) ^; 10) |; 11) &&; 12) ||; 13) ?:; 14) = *= /= += –=; 15) **,**

Значение и содержание каждой операции будет ясно из того контекста, где будет применена эта операция.

В логическом выражении могут использоваться три логические операции: `!` (отрицание), `&&` (И), `||` (ИЛИ). Логическим значением “истина” является любое ненулевое значение, а логическим значением “ложь” – нулевое значение. В связи с этим операндами логических выражений могут быть формально и числа, а не только отношения. Операции отношения >, >=, <, <= имеют одинаковый приоритет. Ниже приоритет операций сравнения на равенство = = и !=. Операции отношения имеют более низкий приоритет, чем арифметические операции, т.е. `i > j+1` то же, что и `i > (j+1)`. Приоритет логических операций еще более низкий, т.е. в логическом выражении сначала вычисляются арифметические выражения, затем операции

отношения, а затем выполняются логические операции. Логические операции выполняются слева направо в соответствии со следующими приоритетами: сначала `!`, далее `&&` и, наконец, `||`. Вычисления прекращаются, как только становится известна истинность результата.

1.3. Массивы и структуры

При использовании простых переменных каждой области памяти для хранения данных соответствует свое имя. Если с группой величин одинакового типа требуются произвести однообразные действия, им дают одно имя, а различают их по порядковому номеру. Такая группа образует м*ассив*. Описание массива отличается от описания простой переменной заданием числа его элементов, взятым в квадратные скобки. Тип элемента массива может быть одним из стандартных типов, типом другого массива, типом указателя или типом структуры. Элементы массива нумеруются с нуля. Для доступа к элементу массива после его имени указывается номер элемента (индекс) в квадратных скобках. Многомерные массивы задаются указанием каждого измерения в квадратных скобках.

Примеры описания массивов:

```
int maxs[10];
float Matr[4][4];
char Name[20];
```

Описан одномерный массив `maxs` из 10 элементов целого типа: `maxs[0]`, `maxs[1]`, ..., `maxs[9]`; двухмерный массив `Matr` из 16 элементов вещественного типа: `Matr[0][0]`, `Matr[0][1]`, ..., `Matr[0][3]`, `Matr[1][0]`, ..., `Matr[3][3]` и одномерный массив `Name` из 20 элементов символьного типа: `Name[0]`, `Name[1]`, ..., `Name[19]`. Массив `Name` представляет символьную строку, состоящую из 19 символов, причем последним элементом массива должен быть нулевой код ‘\0’, который служит ограничителем строки (заносится автоматически).

При обращении к элементу двухмерного массива в квадратных скобках указывается номер элемента по каждому измерению, например, `Matr[1][2]`. Для двухмерного массива можно обращаться и к строке, указав один индекс. Например, `Matr[3]` – ссылка на одномерный массив из 4 элементов, т.е. третья строка массива `Matr`. Ссылка `Matr[,3]` означает обращение к третьему столбцу этой матрицы.

При описании переменных и массивов им можно присвоить начальные значения (инициализировать). Например,

```
float eps = 0.00001;
int i = 0, j = 0;
int maxs[10]={1,3,5,-2,43,6,-8,1,0,3};
double Ms[2][3]={{1.0,0.2,0.8},{5.,3.1,4.9}};
```

Структура состоит из элементов различных типов, причем каждый элемент структуры имеет собственное имя. Элементы структуры называются полями структуры и могут иметь любой тип (простые переменные, массивы, структуры), кроме типа этой же структуры, но могут быть указателями на него. Доступ к полям структуры выполняется с помощью операции выбора **.** (точка) при обращении к полю через имя структуры и операции —> при обращении через указатель. Поля структуры размещаются в оперативной памяти одно за другим в той последовательности, в которой перечислены в описании.

Описание структуры имеет вид

```
struct
{
список описаний
} имя;
```

В фигурных скобках перечисляются типы и имена элементов структуры. Например, структура

```
struct
{
 char Name[20];/*фамилия и инициалы*/
    struct
  {
   int day; /*день*/
   char mon[9]; /*месяц*/
   int year; /*год рождения*/
  } date;

 char adress[40]; /*адрес*/

} student;
```

состоит из трех элементов (одномерного символьного массива, структуры и еще одного одномерного символьного массива). Второй элемент, в свою очередь, структура, состоящая также из трех элементов (переменной целого типа, одномерного символьного массива и