Министерство образования РФ

Самарский государственный аэрокосмический университет имени академика С. П. Королева

ПРОГРАММИРОВАНИЕ НА ЯЗЫКЕ СИ

Методические указания к лабораторным работам по курсу "Информатика" для студентов 1 курса радиотехнического факультета

Самара - 2001

Составитель **А.А.Тюгашев**

Программирование на языке Си: Метод.указания к лабораторным работам/Самарский аэрокосм.ун-т; Сост. А.А.Калентьев, А.А.Тюгашев. Самара, 2001. 14 с.

В данных методических указаниях продолжается рассмотрение комплекса проблем, связанных с использованием одного из самых популярных на сегодняшний день языков программирования – Си. Рассмотрены типовые программы для работы с массивами и матрицами, сортировки, нахождения сумм и максимумов, работы с файлами и структурами данных.

Предназначены для студентов радиотехнического факультета, выполняющих лабораторные работы по курсу "Информатика". Могут быть использованы в практической деятельности. Подготовлены на кафедре компьютерных систем.

Печатаются по решению Редакционного Совета Самарского государственного аэрокосмического университета

1. ПРИНЦИПЫ ВВОДА-ВЫВОДА В ЯЗЫКЕ СИ

Говоря о принципах организации ввода-вывода информации в языке программирования Си прежде всего необходимо отметить тот факт, что сам язык не включает в себя операторов ввода-вывода, а все поддерживающие эти процессы операции осуществляются с помощью стандартных функций. Этот подход был применен разработчиками с целью достижения переносимости между различными платформами.

В языке Си, поскольку он генетически связан с операционной системой UNIX, поддерживается концепция стандартных потоков

ввода (input), вывода (output) и ошибок (errors). При этом по умолчанию стандартный ввод связан с клавиатурой компьютера, стандартный вывод – с экраном дисплея, и поток ошибок – также с экраном. Стандартные операции ввода-вывода поддержаны библиотечными функциями, описание которых содержится в заголовочном файле stdio.h.

Это прежде всего printf для форматного вывода и scanf для ввода информации.

Функция printf имеет следующий синтаксис:

printf(“<format string>”[,<variables list>]**);**

Здесь жирными символами обозначены обязательные части функции, обычным – необязательные. Список переменных <variables list> может отсутствовать, в этом случае printf используется просто для вывода на экран текстовой строки. <format string> представляет собой строку символов, которые необходимо вывести на экран, вперемежку с форматными символами, описывающими формат вывода переменных (если они есть). При этом количество форматных символов должно строго соответствовать числу выводимых переменных в списке, в котором они разделяются запятыми, например:

printf(“Результаты вычислений: a= %7.2f b=%7d\n”,a,b);

В приведенном примере выводятся значения двух переменных – вещественной a и целой b, после этого происходим переход на новую строку.

Форматные символы в общем случае имеют синтаксис:

%[признаки][ширина][.точность]тип

Все параметры, приводимые в квадратных скобках, являются необязательными. Обязательно правильное указание типа выводимого значения из ниже перечисленных.

d – целое число

u – целое число без знака

c – одиночный символ

s – символьная строка

f – число с плавающей точкой

e – выводится вещественное число в экспоненциальной форме, например 1.23e+2

Признаки могут быть следующие:

- (знак минуса) указывает на выравнивание по левому краю

+ указывает на необходимость вывода знака числа

Ширина задает, сколько позиций символов на экране отводится для вывода значения данной переменной, этот параметр очень удобен для выравнивания при печати данных, например, в таблицах.

В случае вывода чисел, имеющих дробную часть после десятичной точки, возможно указание параметра точности, который определяет, сколько знаков внутри определенного параметром ширины поля будет отведено на дробную часть. Например, если значение переменной равно –12.4567, то при применении к ней форматной строки %7.2f будет выведено “ –12.45”.

Внутри форматной строки возможно появление специальных символов. Например, используются следующие комбинации:

\7 – для вывода звукового сигнала

\n – для перехода на новую строку

\r – для вывода символа “возврат каретки”

\t – для горизонтальной табуляции

\\ - для печати символа \

%% - для печати символа %

\” – для печати символа кавычек

Функция scanf используется для ввода данных с клавиатуры (со стандартного устройства ввода). Её синтаксис имеет следующий вид.

scanf(“<format string>”,<pointers list>);

В отличие от функции printf, список <pointers list> здесь является обязательным. При этом он представляет собой список адресов, то есть указателей на переменные, а не самих переменных. В языке Си для взятия адреса переменной во всех случаях, кроме случая символьных строк применяется операция &. Имена переменных, описывающих строки, фактически и так являются указателями на массивы символов.

Строка <format string> содержит форматные символы для соответствующих выводимых переменных, формируемые по тем же правилам, что и в случае функции printf. Их количество должно строго соответствовать числу переменных, например:

scanf(“%d %f”,&a,&b);

2. РАБОТА С ФАЙЛАМИ

Помимо ввода-вывода, работающего со стандартными потоками ввода, вывода и ошибок (фактически это клавиатура и экран дисплея), в стандартных библиотеках языка программирования Си присутствуют средства для работы с информацией, находящейся на магнитных носителях – дисках, и организованной соотвественно, в виде файлов. Существенно облегчает понимание при этом тот факт, что эти функции используют те же основные принципы, что и функции для ввода-вывода со стандартных устройств.

Для работы с файлами в языке Си используется понятие файлового дескриптора. Специальное ключевое слово FILE зарезервировано для обозначения указателя на файл:

```
FILE *fp;
```

описывает дескриптор файла fp. После того, как объявлен файловый дескриптор, его необходимо связать с реальным файлом на диске (а если файл еще не существует, то он создается путем открытия его на запись или добавление). Для открытия файла используется стандартная функция fopen, ее синтаксис следующий:

```
fp=fopen("<имя файла>","<режим>");
```

<имя файла> - строка с именем файла, соответствующая его наименованию в операционной системе.

<режим> может быть следующим:

"w" – запись;

"r" – чтение;

"a" – добавление.

При этом после символа, задающего режим доступа, может идти дополнительный символ: t для текстового доступа, b для доступа в

бинарном режиме. По умолчанию после запуска системы подразумевается текстовый режим доступа.

В случае, если файл открывается на чтение, и он отсутствует на диске, fopen возвращает NULL, в противном случае – указатель на файл (дескриптор). Открытие на запись уже существующего файла предполагает перезапись его содержимого. Открытие на добавление существующего файла предполагает приписывание информации в его конец.

Для работы с текстовыми файлами имеется ряд функций, соответствующих функциям работы со стандартным вводом-выводом, в частности:

fprintf – для вывода в файл;

fscanf – для чтения из файла.

Отличие этих функций от fscanf и fprintf заключается только в том, что первым аргументом, идущим после открывающей круглой скобки, должен быть дескриптор файла, например:

fprintf(fp,”2 + 2 = %f\n”, a);

После использования файла его необходимо закрыть. Это осуществляется с использованием функции fclose, имеющей следующий синтаксис:

fclose(<дескриптор файла>);

Нижеследующая программа осуществляет ввод строки с клавиатуры, сохранение ее в текстовом файле, чтение строки из файла и вывод ее на экран.

```
#include <stdio.h>
int main()
{
  FILE *f;
  char str[40];
  printf (“\nВведите строку:”);scanf(“%s”,str);
  f=fopen(“file.txt”,”w”);
  if (f==NULL) return -1;
  else
  {
    fprintf(f,”%s\n”,str);
    fclose(f);
```

```
    f=fopen(“file.txt”,”r”);
    fscanf(f,”%s”,str);
    fprintf(“\nстрока из файла: %s”,str);
    fclose(f);
    return 0;
  }
}
```

Из программы видно, что файл одновременно можно открыть только на чтение или только на запись. Поэтому два раза используется вызов функции fopen с той же переменной – указателем на файл, но с разными режимами доступа.

3. МАССИВЫ И МАТРИЦЫ

Помимо простых одинарных переменных одного из базовых типов, язык Си позволяет объявлять массивы. Под массивом понимается набор (упорядоченное множество) элементов одного типа. При этом возможен доступ к элементам массива путем указания индекса (т. е. номера элемента в массиве). При этом, в отличие от многих языков программирования, в языке Си

принято соглашение о том, что элементы массива нумеруются, начиная с нуля, то есть первый элемент – элемент с номером 0.

Объявляются массивы там, же где и обычняе переменные, то есть в начале функции, но в квадратных скобках указывается размерность (количество элементов), например:

```
int a,b,c;
float x[20];
char c,str[40];
```

Здесь описаны массив вещественных чисел из 20 элементов, а также массив символов (строка) размерностью 40. Обратим внимание на то, что в языке Си строки представляют собой массивы символов, при этом на конец строки указывает символ с кодом (значением), равным нулю.

При работе с массивами часто используют цикл по его элементам, при этом значение переменной, используемой в качестве индекса, либо увеличивается, либо уменьшается на единицу. Обратим внимание на тот факт, что поскольку нумерация элементов начинается с ноля, переменная-индекс пробегает значение от 0 до N-1, где N – количество элементов массива.

Массивы могут фигурировать и в качестве формальных и фактических параметров функций. Пример:

```
#include <stdio.h>
#include <math.h>
int Pecat_kornei(float args[],int kol)
{
  int i;
  float res;
  for (i=0;i<kol;i++)
  {
    if (args[i]>0)
    {
      res=sqrt(args[i]);
      printf("Корень из %d – го элемента равен %f\n",i,res);
    }
    else
      printf("Элемент массива номер %d меньше нуля !",i);
```

```
  }
}
```

Используется стандартная библиотека математических функций, описанных в заголовочном файле math.h. Функция sqrt() возвращает значение квадратного корня числа.

В приведенной программе используется цикл for языка программирования Си, который является весьма гибким инструментом. После ключевого слова for в круглых скобках, разделенные точками с запятой, идут три выражения. Первое – группа операторов языка, разделяемых запятыми, которые должны выполниться до первого выполнения цикла. Второе – логическое условие, проверяемое перед каждым выполнением цикла, и определяющее, продолжать выполнение или нет. Наконец, третье выражение – действия (через запятую), которые выполняются по окончании каждой итерации. Само тело цикла, то есть действия, выполняемые циклически, заключается в фигурные скобки, если это – не один оператор, в противном случае просто это следующий за круглыми скобками оператор.

В примере используется также условный оператор языка Си. Его синтаксис выглядит следующим образом:

```
if (<условное выражение>)
  <действие 1>
else
  <действие 2>;
```

В случае истинности условного выражения (в котором могут фигурировать несколько условий, связанных логическими операциями И -&&, ИЛИ - ||), выполняются <действия 1>, в качестве которых может выступать один оператор, или группа операторов, заключенных в операторные скобки (фигурные скобки). Если условие не выполняется, то управление передается на <действие 2>.

При этом ветвь ИНАЧЕ (else) может отсутствовать. Этот случай соответствует сокращенному условному выражению, когда в случае ложности условия просто ничего не делается.

Помимо простых, или одномерных массивов, в языке Си возможна работа с так называемыми многомерными массивами, то есть массивами, элементы которых также, в свою очередь, являются массивами. В простейшем случае двумерный массив (аналогом

которого в математике служит двумерная матрица) может быть описан, например, следующим образом:

```
float matr[10][20];
```

Здесь объявляется массив из 10 строк по 20 элементов в каждой. Доступ к элементам массива осуществляется очевидно:

```
Y+=matr[i][j];
```

В этом примере к переменной Y добавляется значение элемента массива matr, находящееся на пересечении i-ой строки и j-го столбца матрицы.

Аналогично возможна работа с трех- и более мерными массивами.

4. СТРУКТУРЫ ДАННЫХ В ЯЗЫКЕ СИ

В различных современных языках программирования возможно создание на основе простых (базовых) типов данных более сложных конструкций. В одних языках такого рода данные называются записями, в других – структурами, и т.д.

Идея структуры состоит в том, что под одним именем группируются сразу несколько параметров (элементов данных). Можно попытаться описать, например, гоночный автомобиль следующим набором параметров: мощность двигателя (целое число лошадиных сил), изготовитель (строка символов), тип мотора (строка символов), количество передач (целое число), максимальная скорость (число с плавающей точкой).

В языке программирования Си возможно объявление соответствующей данному описанию структуры:

```
struct
{
  int Power;
  char Manufacturer[40];
  char Motor[40];
  int transmissons;
  float MaxSpeed;
} Auto;
```

После этого объявления возможно обращение к отдельным элементам данных переменной Auto структуры, делается это, например, следующим образом:

```
Auto.Power=340;
```

```
Auto.MaxSpeed=380.2;
scanf("%s",Auto.Manufacturer);
```

Помимо простого объявления структуры, в языке Си возможно объявление структуры как определение нового типа данных с новым именем – именем структуры. Для этого служит специальное ключевое слово typedef. В приведенном примере, например, заголовок мог бы выглядеть следующим образом: typedef struct {…} Auto;. После этого возможным становится объявлять переменные нового типа Auto, например:

```
Auto car,car2;
```

Дальнейшее обращение к членам данных переменных car и car2 осуществляется стандартным образом: car.Power, car.Motor и т.д. Понятие структуры является прототипом более сложного понятия, используемого в объектно-ориентированных языках программирования (например, C++) – понятия класса.

5. РАБОТА С ТЕКСТОВЫМИ СТРОКАМИ

Текстовая информация представляет собой наиболее часто встречающуюся разновидность информации, обрабатываемой на современных компьютерах.

В языке программирования Си текстовые строки рассматриваются как одномерные массивы символов, при этом ограничителем строки при выводе, например, на экран или в файл считается нулевой символ (символ с кодом, равным нулю).

Таким образом, для объявления строки с максимальным числом хранимых символов 19, возможно следующее описание:

```
char x[20];
```

Стандартная библиотека функций для работы с текстовыми строками, описанная в заголовочном файле string.h, имеет следующие наиболее употребительные функции для работы со строками:

strcpy(<строка 1>,<строка 2>) - для копирования содержимого <строки 2> в <строку 1>;

strcmp(<строка 1>,<строка 2>) - для сравнения <строки 1> и <строки 2>, функция возвращает 0 при равенстве строк;

strcat(<строка 1>,<строка 2>) - для приписывания (конкатенации) <строки 2> в конец <строки 1>;

strstr(<строка 1>,<строка 2>) – для поиска вхождения <строки 2> в <строку 1>, возвращает ноль или адрес начала вхождения <строки 2> в <строку1>.

Список лабораторных работ

1. Знакомство с интегрированной средой разработки программ Borland C++. Ввод, трансляция и выполнение простейшей программы на языке программирования Си. Анализ результатов.
2. Типы данных. Математические вычисления с использованием языка программирования Си. Знакомство

с функциями математической библиотеки. Основные операции ввода-вывода. Условные выражения.

3. Операторы цикла в языке Си. Работа с массивами и матрицами на языке Си. Поиск максимумов, минимумов.
4. Написание программ сортировки. Знакомство с отладчиком интегрированной среды. Установка точек останова. Просмотр значений переменных в ходе исполнения программы.
5. Работа с текстовыми строками в языке Си. Поиск подстроки в строке.
6. Структуры данных в языке Си. Описание и работа со структурами.
7. Работа с файлами. Открытие файлов, вывод информации в файл, запись и добавление.
8. Разработка информационной системы с вводом, хранением, поиском и заменой информации о студентах на основе структур и текстовых файлов.