ЛАБОРАТОРНАЯ РАБОТА № 16

СИМВОЛЬНЫЕ МАССИВЫ

Цель работы: изучение правил описания символьных массивов, строковых выражений, операций над строковыми данными, функций для обработки строковых данных. Овладение практическими навыками составления программ с использованием символьных массивов.

Краткие теоретические сведения

Строка представляет собой массив значений типа **char**, завершающийся нулевым байтом.

Строка описывается как символьный массив:

имя_типа имя_массива[константное выражение];

Например:

```
int a[10];          /* массив с именем a, содержащий 10 элементов
                       целого     типа */
char b[10*5];  /* массив b, содержащий 50 элементов типа char */
```

Символьная строка – это последовательность символов, в которой на каждый символ отводится 1 байт. Каждый символ имеет свой код (от 0 до 255), эти коды определяются по специальной таблице. Символ с кодом ноль не имеет никакого изображения, в программе его записывают как **'\0'.** Символ с кодом ноль (обозначается как '**\0**') и цифра ноль (обозначается **'0'**, имеет код 48) – это два разных символа.

Начальное значение строки можно задать при объявлении в двойных кавычках после знака равенства:

```
char s[80] = "Привет, Вася!";
```

Символы в кавычках будут записаны в начало массива **s**, а затем – признак окончания строки '**\0**'. Оставшиеся символы не меняются, и в локальных строках там будет «мусор». Можно написать и так

```
char s[] = "Привет, Вася!";
```

В этом случае компилятор подсчитает символы в кавычках, выделит памяти на 1 байт больше и занесет в эту область саму строку и завершающий ноль.

Если строка не будет изменяться во время работы программы, то можно объявить константу (постоянную строку) так:

const char PRIVET[] = "Привет, Вася!";

Стандартный ввод и вывод

Для ввода и вывода строк с помощью функций scan**f** и printf используется специальный формат "%s":

```
#include<stdio.h>
void main()
{
   char Name[50];
   printf("Как тебя зовут? ");
   scanf("%s", Name);
   printf("Привет, %s!", Name);
}
```

Заметьте, что в функцию scanf надо передать просто имя строки (без знака **&**), ведь имя массива является одновременно адресом его начального элемента.

Однако у функции scanf есть одна особенность: она заканчивает ввод, встретив первый пробел. Если надо ввести всю строку целиком, включая пробелы (то есть до нажатия на клавишу Enter), то применяется функция gets(s);

Для вывода строки на экран можно (кроме printf) использовать и функцию puts, которая после вывода строки еще и дает команду перехода на новую строку.

Пример. Ввести символьную строку и заменить в ней все буквы 'А' на буквы 'Б'.

Будем рассматривать строку как массив символов. Надо перебрать все элементы массива, пока мы не встретим символ '\0' (признак окончания строки) и, если очередной символ – это буква 'А', заменить его на 'Б'. Для этого используем цикл while, потому что мы заранее не знаем длину строки. Условие цикла можно сформулировать так: «пока не конец строки».

```
#include<stdio.h>
void main()
{
    char s[80];
    int i;
    puts( "Введите строку" );
    gets(s);
    i=0;  // начать с первого символа, s[0]
    while (s[i] != '\0' )  // пока не достигли конца строки
       {
          if ( s[i] = = 'A' )  s[i]='Б';
          i++;
        }
    puts ( s );
}
```

Заметьте, что одиночный символ записывается в апострофах, а символьная строка – в кавычках.

Функции для работы со строками

Для использования этих функций надо включить в программу заголовочный файл

#include <string.h>

***Длина строки* – strlen**

Эта функция определяет текущую длину строки (не считая '**\0**').

```
int len;
char s[] = "Hello, world! ";
len = strlen(s);
printf ( "Длина строки %s равна %d", s, len );
```

Сравнение строк – strcmp

Для сравнения двух строк используют функцию **strcmp**. Функция возвращает ноль, если строки равны (то есть «разность» между ними равна нулю) и ненулевое значение, если строки различны. Сравнение происходит по кодам символов, поэтому

функция различает строчные и заглавные буквы – они имеют разные коды.

```
char s1[] = "Вася",  s2[] = "Петя";
if ( 0 = = strcmp(s1,s2) )
   printf("Строки %s и %s одинаковы", s1, s2);
else printf("Строки %s и %s разные", s1, s2);
```

Если строки не равны, функция возвращает «разность» между первой и второй строкой, то есть *разность кодов первых различных символов*. Эти числа можно использовать для сортировки строк – если «разность» отрицательна, значит первая строка «меньше» второй, то есть стоит за ней в алфавитном порядке.

Задача. Ввести две строки и вывести их в алфавитном порядке.

```
#include <stdio.h>
#include <string.h>
void main()
{ char s1[80], s2[80];
  printf ("Введите первую строку");
  gets(s1);
  printf ("Введите вторую строку");
  gets(s2);
  if ( strcmp(s1,s2) <= 0 )
     printf("%s\n%s", s1, s2);
 else printf("%s\n%s", s2, s1);
}
```

Иногда надо сравнить не всю строку, а только первые несколько символов. Для этого служит функция **strncmp** (с буквой **n** в середине). Третий параметр этой функции – количество сравниваемых символов. Принцип работы такой же – она возвращает нуль, если заданное количество первых символов обеих строк одинаково.

```
char s1[80], s2[80];
printf ("Введите первую строку");
gets(s1);
printf ("Введите вторую строку");
gets(s2);
```

```
 if ( 0 = = strncmp(s1, s2, 2) )
  printf("Первые два символа %s и %s одинаковы", s1, s2);
 else
  printf("Первые два символа %s и %s разные", s1, s2);
```

Один из примеров использования функции **strcmp** – проверка пароля.

Пример. Составить программу, которая определяет, сколько цифр в символьной строке. Программа должна работать только при вводе пароля «куку».

```
#include<stdio.h>
#include<string.h>
main()
{
char pass[] = "куку", // правильный пароль
s[80];                    // вспомогательная строка
int i, count = 0;
printf ("Введите пароль ");
gets(s);
if ( strcmp ( pass, s ) != 0 )
  {
    printf ( "Неверный пароль" );
    return 1;    // выход по ошибке, код ошибки 1
  }
printf ("Введите строку");
gets(s);
i = 0;
while ( s[i] != '\0' )   {
   if ( s[i] >= '0' && s[i] <= '9' )
      count ++;   }
printf("\nНашли %d цифр", count);
}
```

В этой программе использован тот факт, что коды цифр расположены в таблице символов последовательно от '0' до '9'.

Копирование строк

В копировании участвуют две строки, они называются «*источник*» (строка, откуда копируется информация) и «*приемник*» (куда она записывается или добавляется).

При копировании строк надо проверить, чтобы для строки-приемника было выделено достаточно места в памяти.

Простое копирование выполняет функция strcpy. Она принимает два аргумента: сначала строка-приемник, потом – источник (порядок важен!).

```
char s1[50], s2[10];
gets(s1);
strcpy ( s2, s1); // s2 (приемник) <- s1 (источник)
puts ( s2 );
```

Следующая строка скопирует строку s2 в область памяти строки s1, которая начинается с ее 6-ого символа, оставив без изменения первые пять:

```
strcpy ( s1+5, s2 );
```

Функция позволяет скопировать только заданное количество символов, она называется strncpy и принимает в третьем параметре количество символов, которые надо скопировать. Важно помнить, что эта функция *НЕ записывает завершающий нуль*, а только копирует символы (в отличие от нее strcpy всегда копирует завершающий нуль). Функция strncpy особенно полезна тогда, когда надо по частям собрать строку из кусочков.

```
char s1[] = "Ку-ку", s2[10];
strncpy ( s2, s1, 2 );   // скопировать 2 символа из s1 в s2
puts ( s2 );                  // ошибка! нет последнего '\0'
s2[2] = '\0';          // добавляем символ окончания строки
puts (s2);            // вывод
```

Объединение строк

Функция – strcat позволяет добавить строку-источник в конец строки-приемника (завершающий нуль записывается автоматически).

```
char s1[80] = "Могу, ",
s2[] = "хочу, ", s3[] = "надо!";
strcat ( s1, s2 );   // дописать s2 вконец s1
```

```
puts ( s1 );          // "Могу, хочу, "
strcat ( s1, s3 );   // дописать s3 вконец s1
puts ( s1 );          // "Могу, хочу, надо!"
```

Порядок выполнения работы

1. Изучить теоретические сведения.
2. Ответить на контрольные вопросы.
3. Выполнить задание.

Контрольные вопросы

1. Что понимается под строкой в языке Си?
2. Как описываются строковые данные?
3. Что является признаком конца строки?
4. Какие стандартные функции используются для обработки строковых данных?

Задания для выполнения

1. Ввести строку. Преобразовать ее, заменив точками все двоеточия (**:**), встречающиеся среди первых n/2 символов (n – длина введенной строки), и заменив точками все восклицательные знаки, встречающиеся среди символов, стоящих после n/2 символов. Вывести преобразованную строку.

2. Если в заданной строке есть хотя бы один символ “*”, то подсчитать количество цифр, встречающихся до первого символа “*”, иначе - вывести соответствующее сообщение.

3. Если в заданной строке есть хотя бы один символ “*”, то подсчитать количество точек, встречающихся после первого символа “*”, иначе - вывести соответствующее сообщение.

4. Ввести набор слов, разделенных одним пробелом. Подсчитать в нем количество слов, заканчивающихся заданной буквой (ввести с клавиатуры).

5. Ввести строку, в которой слова разделены пробелами. Если в ней есть хотя бы один символ “:”, то подсчитать количество слов, начинающихся с большой буквы, расположенных после первого символа “:” или вывести сообщение об отсутствии указанного символа.

6. Ввести три строки, сцепить их. В результирующей строке подсчитать количество заглавных букв. Заменить все запятые точками. Результат вывести на экран.

7. Ввести строку из английских слов, разделенных пробелами. Вывести на экран слова, начинающиеся с гласных букв.

8. Ввести строку, в которой слова разделены одним пробелом. Подсчитать количество слов, состоящих из не более чем четырех букв.

9. Ввести строку, в которой слова разделены одним пробелом. Вывести на экран слова, начинающиеся и заканчивающиеся одинаковой буквой.

10. Ввести строку, в которой слова разделены одним пробелом. Определить, сколько букв содержит самое длинное слово в строке.

11. Ввести набор слов, разделенных одним пробелом. Подсчитать в нем количество слов, начинающихся с заданной буквы (ввести с клавиатуры).

12. Ввести строку, в которой слова разделены одним пробелом. Вывести на экран слова, состоящих из пяти букв.

13. Если в заданной строке есть хотя бы один символ “*”, то подсчитать количество слов после первого символа “*”.

14. Ввести строку, содержащую скобки. Подсчитать количество символов между первой и последней скобками.

15. Ввести строку, в которой слова разделены одним пробелом. Вывести на экран слова, состоящих из M (ввести с клавиатуры) букв, и подсчитать их количество.