Урок 8. Обработка символьных строк

В языках C/C++ символьные строки организуются как массивы символов, последним из которых является символ \0, внутренний код которого равен нулю. Размер строки символьного массива в С не имеет ограничения.

Строка описывается как символьный массив. Например:

```
char  STR[ 20 ];
```

Одновременно с описанием строка может инициализироваться. Возможны два способа инициализации строки — с помощью строковой константы и в виде списка символов:

```
char S[10] = "строка";
char S[ ] = "строка";
char S[ 10 ] ={'с', 'т', 'р', 'о', 'к', 'а', '\0' };
```

По результату первого описания под строку s будет выделено 10 байт памяти, из них первые 7 получат значения при инициализации (седьмой — нулевой символ). Второе описание сформирует строку из семи символов. Третье описание по результату равнозначно первому. Конечно, можно определить символьный массив и так:

```
char  S[10] = {'с', 'т', 'р', 'о', 'к', 'а'};
```

т.е. без нулевого символа в конце. Но это приведет к проблемам с обработкой такой строки, так как будет отсутствовать ориентир на его окончание. Отдельные символы строки идентифицируются индексированными именами. Например, в описанной выше строке s[0] = 'c', S[5] = 'a'.

Обработка строк обычно связана с перебором всех символов от начала до конца. Признаком конца такого перебора является обнаружение нулевого символа. В следующей программе производятся последовательная замена всех символов строки на звездочки и подсчет длины строки.

Пример 1.

```
//Замена символов на звездочки
#include <stdio.h>
#include <conio.h>
#include <clocale>

int  main( )
{
setlocale(LC_ALL,"RUS");
char S[ ] = "fh5j";
int  i = 0;
void  clrscr( );
puts( S );
while(S[ i ])
{
S[i++] = ' * ';
puts( S );
}
printf("\nДлина  строки = ",  i);
}
```

В результате выполнения программы на экране получим:

```
fh5j
```

```
*h5j
**5j
****
```
Длина строки = 4

В этой программе цикл повторяет свое выполнение, пока s[i] не получит значение нулевого символа.

Для вывода строки на экран в стандартной библиотеке stdio имеется функция puts(). Аргументом этой функции указывается имя строки. В этой же библиотеке есть функция ввода строки с клавиатуры с именем gets( ). В качестве аргумента указывается имя строки, в которую производится ввод.

Среди стандартных библиотек C/C++ существует библиотека функций для обработки строк. Ее заголовочный файл — string.h. В следующем примере используется функция определения длины строки из этой библиотеки. Имя функции — strlen( ). В качестве аргумента указывается имя строки.

Пример 2. Ввести символьную строку. Перевернуть (обратить) эту строку. Например, если ввели строку «abcdef», то в результате в ней должны получить «fedcba».

```
//Обращение  строки
#include <stdio.h>
#include <string.h>
#include  <conio.h>
#include  <clocale>

int  main( )
{
setlocale(LC_ALL, "RUS");
char C, S[ 10 ];
int  i;
void  clrscr( );
printf("Введите  строку");
gets( S );
for(i = 0;  i <= (strlen( S ) - l) / 2;  i++)
{
C = S[  ];
S[ i ] = S[strlen(S) - i - 1];
S[strlen( S ) - i - 1] = C;
}
printf(\nПеревернутая  строка:");
puts( S );
}
```

Идея алгоритма состоит в перестановке символов, расположенных на одинаковом расстоянии от начала и конца строки. Перебор элементов строки доходит до ее середины. Составляя подобные программы, не надо забывать, что индекс первого символа строки — 0, а индекс последнего на единицу меньше длины строки.

Строка как параметр функции. Использование строк в качестве параметра функции аналогично рассмотренному выше использованию массивов других типов. Необходимо помнить, что имя массива есть указатель на его начало. Однако для строк имеется одно существенное отличие от массивов других типов: имя строки является указателем-переменной, и, следовательно, его значение может подвергаться изменению. Стандарт

С рекомендует в заголовках функций, работающих со строками, явно использовать обозначение символьного указателя.

Пример 1. Запишем определение функции вычисления длины строки (аналог стандартной функции strlen( )).

```
int  length(char  *s)
{
int   k;
for(k = 0;  *s++! = '\0';  k++) ;
return  k;
}
```

Здесь функция использует явный механизм работы с указателем. Изменение значения указателя s допустимо благодаря тому, что он является переменной. Для числовых массивов этого делать нельзя! Если соблюдать данное ограничение и для строк, то условное выражение в операторе for следовало бы писать так:

```
 *(s + k)  ! = '\0  '  или  s [ k ] ! = ' \ 0 ' .
```

Пример 2. Оформим программу обращения строки в виде функции и напишем основную программу, использующую ее. Алгоритм обращения реализуем иначе, чем в рассмотренной выше программе. Для определения длины строки не будем пользоваться стандартной функцией. Для вывода строки на экран применим функцию printf( ) со спецификатором %s (работает аналогично функции puts( )).

```
//Обращение  строки
#include  <stdio.h>
#include <conio.h>
#include  <clocale>

//Прототипы  функций

int  length(char *str);
void  invers(char *str);

// Основная  программа

int main( )
{
setlocale(LC_ALL, "RUS");
char S[ ] = "123456789";
void  clrscr( )
invers(S);        //Вызов  функции обращения строки
printf("\n%s",  S);
}
//Функция вычисления длины строки
int  length(char *s)
{
int k;
for(k = 0;  *s++ != '\0';  k++) ;
return  k;
}

//Функция обращения  строки

void  invers(char  *e)
{
```

```
char c;
int i, j, m;
m = length(e);           //Вызов функции  length
for(i = 0,  j = m - 1 ;  i < j ;  i++,  j -- )
{
c = e[ i ];
e[ i ] = e[ j ];
e[ j ] = c;
}
}
```

В результате выполнения этой программы на экране получим строку:

9 8 7 6 5 4 3 2 1 0

Пример 3. Описать функцию вставки символа в строку. Параметры функции: строка, позиция вставки, вставляемый символ. Использовать эту функцию в основной программе, где исходная строка, номер позиции и вставляемый символ задаются вводом.

```
// Вставка символа в строку
#include <stdio.h>
#include <string.h>
#include <conio.h>
#include  <clocale>

void  INSERT(char  *str, int p, char c)
{
int  i;
for(i = strlen( str );
i>=p;  i--)
str[i + l] = str[i];
str[ p ] = c;
}

int  main( )
{
setlocale(LC_ALL, "RUS");
char c,  S[100];
int  n;
void   clrscr( );
puts("Введите  строку:");
get( S ) ;
puts("Введите  позицию  вставки:");
scanf("%d",  &n);
puts("Введите  символ:");
c = getche( );
INSERT(S, n, c)
puts("\nРезультат:");
puts(S);
}
```

Вариант диалога на экране при выполнении этой программы:
Введите строку:

0123456789

Введите позицию вставки:

4

Введите символ:

*

Результат:

0123*456789

В этой программе наряду с рассмотренными ранее функциями для ввода и вывода строки gets( ) и puts( ) используется функция чтения символа с клавиатуры getche( ) из библиотеки stdio.h. Ее прототип имеет вид: int getche(void). Она возвращает значение символа, введенного с клавиатуры, которое может быть присвоено символьной переменной.

Упражнения

1. Составить программу подсчета количества цифр в данной строке.
2. Составить программу, которая по данной символьной строке формирует числовой массив, содержащий коды символов, составляющих строку.
3. Составить функцию, определяющую тождественность двух данных строк.
4. Составить программу, удаляющую в данной строке каждый символ с четным номером.
5. Составить функцию конкатенации (слияния) двух строк. В основной программе использовать эту функцию для слияния четырех строк.
6. Составить функцию, переводящую десятичное целое число, представленное в символьном виде, в соответствующую величину целого типа.
7. Составить функцию, переводящую десятичное вещественное число, представленное в символьном виде, в соответствующую величину плавающего типа.