Лекция 5

Строки
Формат строки в языке С
Принципы работы со строками
Функции для работы со строками

Строка

Строка – это последовательность ASCII или UNICODE символов.

Строки в большинстве языков программирования высокого уровня рассматриваются как отдельный тип, входящий в систему базовых типов языка.

Так как язык С по своему происхождению является языком системного программирования, то строковый тип данных в нем как таковой отсутствует, а в качестве строк используются обычные массивы символов.

Форматы хранения строк

Исторически сложилось два представления формата строк:

- формат ANSI;
- строки с завершающим нулем (используется в языке С).

Формат ANSI устанавливает, что значением первой позиции в строке является ее длина, а затем следуют сами символы строки. Например, представление строки “Моя строка!” будет следующим:

11	‘М’	‘о’	‘я’	‘ ’	‘с’	‘т’	‘р’	‘о’	‘к’	‘а’	‘!’

Форматы хранения строк

В строках с завершающим нулем, значащие символы строки указываются с первой позиции, а признаком завершения строки является значение ноль. Представление рассмотренной ранее строки в этом формате имеет вид:

‘М’	‘о’	‘я’	‘ ’	‘с’	‘т’	‘р’	‘о’	‘к’	‘а’	‘!’	0

Объявление строк

Строки в языке С реализуются посредством массивов символов. Поэтому объявление ASCII строки на языке С имеет следующий синтаксис:

char имя[длина];

Объявление строки имеет тот же синтаксис, что и объявление одномерного символьного массива. Длина строки должна представлять собой целочисленное значение (в стандарте **C89** – константа, в стандарте **C99** может быть выражением). Длина строки указывается с учетом одного символа на хранение завершающего нуля, поэтому максимальное количество значащих символов в строке на единицу меньше ее длины.

Например, строка может содержать максимально двадцать символов, если объявлена следующим образом:

char str[21];

Объявление строк

Инициализация строки осуществляется при ее объявлении, используя следующий синтаксис:

char str[длина] = строковый литерал;

Строковый литерал – строка ASCII символов заключенных в двойные кавычки.

Примеры объявления строк с инициализацией:

```
char str1[20] = “Введите значение: ”,
     str2[20] = “”;
```

Объявление строк

Объявление константных строковых переменных начинается с ключевого слова **const**, за которым следует объявление строки с инициализацией.

Пример:

```
const char message[] = “Сообщение об ошибке!”;
```

Работа со строками

Так как строки на языке С являются массивами символов, то к любому символу строки можно обратиться по его индексу. Для этого используется синтаксис обращения к элементу массива, поэтому первый символ в строке имеет индекс ноль.
Например, в следующем фрагменте программы в строке str осуществляется замена всех символов *‘a’* на символы *‘A’* и наоборот.

```
for(int i=0;str[i]!=0;i++){
  if(str[i] == ‘a’) str[i] = ‘A’;
    else if(str[i] == ‘A’) str[i] = ‘a’;
}
```

Массивы строк

Объявление массивов строк в языке С также возможно. Для этого используются двумерные массивы символов, что имеет следующий синтаксис:

char имя[количество][длина];

Первым размером матрицы указывается количество строк в массиве, а вторым – максимальная (с учетом завершающего нуля) длина каждой строки. Например, объявление массива из пяти строк максимальной длиной 30 значащих символов будет иметь вид:

char strs[5][31];

Массивы строк

При объявлении массивов строк можно производить инициализацию:

```
char имя[количество][длина] =
    {строковый литерал №1, ... строковый литерал №N};
```

Число строковых литералов должно быть меньше или равно количеству строк в массиве. Если число строковых литералов меньше размера массива, то все остальные элементы инициализируются пустыми строками. Длина каждого строкового литерала должна быть строго меньше значения длины строки (для записи завершающего нуля).

Массивы строк

Например:

```
char month[12][10] = {
    ”Январь”, ”Февраль”, ”Март”, ”Апрель”, ”Май”,
    ”Июнь”,   ”Июль”, ”Август”, ”Сентябрь”,”Октябрь”,
    ”Ноябрь”, ”Декабрь”
 };
```

Массивы строк

При объявлении массивов строк с инициализацией допускается не указывать количество строк в квадратных скобках. В таком случае, количество строк в массиве будет определено автоматически по числу инициализирующих строковых литералов.

Например, массив из семи строк:

```
char days[][12] = {
  ”Понедельник”, ”Вторник”, ”Среда”, ”Четверг”,
  ”Пятница”, ”Суббота”, ”Воскресенье”
};
```

Функции для работы со строками

Все библиотечные функции, предназначенные для работы со строками, можно разделить на три группы:

- ввод и вывод строк;
- преобразование строк;
- обработка строк.

Ввод и вывод строк

Для ввода и вывода строковой информации можно использовать функции форматированного ввода и вывода (**printf** и **scanf**). Для этого в строке формата при вводе или выводе строковой переменной необходимо указать спецификатор типа %s. Например, ввод и последующий вывод строковой переменной будет иметь вид:

```
char str[31] = “”;
printf(“Введите строку: ”);
scanf(“%30s”,str);
printf(“Вы ввели: %s”,str);
```

Ввод и вывод строк

Недостатком функции **scanf** при вводе строковых данных является то, что символами разделителями данной функции являются:

- перевод строки,
- табуляция;
- пробел.

Поэтому, используя данную функцию невозможно ввести строку, содержащую несколько слов, разделенных пробелами или табуляциями. Например, если в предыдущей программе пользователь введет строку:

"*Сообщение из нескольких слов*",

то на экране будет выведено только "*Сообщение*".

Ввод и вывод строк

Для ввода и вывода строк в библиотеке **stdio.h** содержатся специализированные функции **gets** и **puts**.

Функция **gets** предназначена для ввода строк и имеет следующий заголовок:

char * gets(**char** **buffer*);

Функция **puts** предназначена для вывода строк и имеет следующий заголовок:

int puts(**const char** **string*);

Ввод и вывод строк

Простейшая программа: ввод и вывод строки с использованием функций **gets** и **puts** будет иметь вид:

```
char str[100] = “”;
printf(“Введите строку: ”); gets(str);
printf(“Вы ввели: ”); puts(str);
```

Ввод и вывод строк

Помимо функций ввода и вывода в потоки в библиотеке **stdio.h** присутствуют функции форматированного ввода и вывода в строки. Функция форматированного ввода из строки имеет следующий заголовок:

int sscanf(**const char** * **restrict** *buffer*,
const char * **restrict** *string*,[*address*] ...);

Ввод и вывод строк

Функции форматированного вывода в строку имеют следующие заголовки:

int sprintf(**char** * **restrict** *buffer*,
const char * **restrict** *format*, [*argument*] ...);

int snprintf(**char** * **restrict** *buffer*, size_t *maxsize*,
const char * **restrict** *format*, [*argument*] ...);

Ввод и вывод строк

В следующем фрагменте программы осуществляется ввод целых чисел и вычисление их суммы (ввод значений продолжается пока не будет введена пустая строка):

```
int summa = 0;
while(1){
  char str[15];
  printf("Введите число или пустую строку: ");
  gets(str);
  if(str[0]==0) break;
  int n;
  sscanf(str,"%d",&n);
  summa += n;
}
printf("Сумма чисел: %d\n",summa);
```

Преобразование строк

Для преобразования строк, содержащих числа, в численные значения в библиотеке **stdlib.h** предусмотрен следующий набор функций:

double atof(**const char** **string*);
int atoi(**const char** **string*);
long int atol(**const char** **string*);
long long int atoll(**const char** **string*);

Преобразование строк

Корректное представление вещественного числа в текстовой строке должно удовлетворять формату:

[ ][{**+**|**-**}][цифры][.цифры][{**e**|**E**}[{**+**|**-**}]цифры]

После символов E, e указывается порядок числа.

Корректное представление целого числа в текстовой строке должно удовлетворять формату:

[ ] [{**+**|**-**}] цифры

Преобразование строк

Помимо приведенных выше функций в библиотеке **stdlib.h** доступны также следующие функции преобразования строк в вещественные числа:

float strtof(**const char** * **restrict** *string*,
char ** **restrict** *endptr*);

double strtod(**const char** * **restrict** *string*,
char ** **restrict** *endptr*);

long double strtold(**const char** * **restrict** *string*,
char ** **restrict** *endptr*);

Преобразование строк

Аналогичные функции присутствуют и для преобразования строк в целочисленные значения:

```
long int strtol(const char * restrict string,
        char ** restrict endptr, int base);

unsigned long strtoul(const char * restrict string,
            char ** restrict endptr, int base);

long long int strtoll(const char * restrict string,
            char ** restrict endptr, int base);

unsigned long long strtoull(
                const char * restrict string,
                 char ** restrict endptr, int base);
```

Преобразование строк

Функции обратного преобразования (численные значения в строки) в библиотеке **stdlib.h** присутствуют, но они не регламентированы стандартом, и рассматриваться не будут. Для преобразования численных значений в строковые наиболее удобно использовать функции **sprintf** и **snprintf**.

Обработка строк

В библиотеке **string.h** содержатся функции для различных действий над строками.

Функция вычисления длины строки:

size_t strlen(**const char** **string*);

Пример:

```
char str[] = “1234”;
int n = strlen(str);    //n == 4
```

Обработка строк

Функции копирования строк:

char * strcpy(**char** * **restrict** *dst*,
const char * **restrict** *src*);

char * strncpy(**char** * **restrict** *dst*,
const char * **restrict** *src*, size_t *num*);

Пример:
char str[] = “abcdefg”, dst1[10] =“”, dst2[10] = “”;
strcpy(dst1,str); //dst1 == “abcdefg”
strncpy(dst2,str,5); //dst2 == “abcde”

Обработка строк

Функции сравнения строк:

int strcmp(**const char** **string1*, **const char** **string2*);

int strncmp(**const char** **string1*, **const char** **string2*,
size_t *num*);

Функции осуществляют сравнение строк по алфавиту и возвращают:

- положительное значение – если *string1* больше *string2*;
- отрицательное значение – если *string1* меньше *string2*;
- нулевое значение – если *string1* совпадает с *string2*.

Пример:

char str1[] = “Компьютер”, str2[] = “Компьютерный”;

int n1 = strcmp(str1,str2); //n1 < 0

int n2 = strncmp(str1,str2,9); //n2 == 0

Обработка строк

Функции объединения (конкатенации) строк:

char * strcat(**char** * **restrict** *dst*,
const char * **restrict** *src*);

char * strncat(**char** * **restrict** *dst*,
const char * **restrict** *src*, size_t *num*);

Пример:
char str1[20] = “серверный ”, str2[20] = “Главный ”,
str3[] = “зал”;
strncat(str2,str1,6); //str2 == “Главный сервер”
strcat(str1,str3); //str1 == “серверный зал”;

Обработка строк

Функции поиска символа в строке:

char * strchr(**const char** **string*, **int** *c*);
char * strrchr(**const char** **string*, **int** *c*);

Пример:
char str[] = “Строка для поиска”;
char *str1 = strchr(str,’к’); //str1 == “ка для поиска”
char *str2 = strrchr(str,’к’);//str2 == “ка”

Обработка строк

Функция поиска строки в строке:

char * strstr(**const char** **str*, **const char** **substr*);

Пример:

```
char str[] = “Строка для поиска”;
char *str1 = strstr(str,“для”);  //str1 == “для поиска”
```

Обработка строк

Функция поиска первого символа в строке из заданного набора символов:

size_t strcspn(**const char** **str*, **const char** **charset*);

Пример:

char str[] = “Компьютер”;

char ch = str[strcspn(str,”трм”)]; //ch == ‘м’

Обработка строк

Функции поиска первого символа в строке не принадлежащему заданному набору символов:

size_t strspn(**const char** **str*, **const char** **charset*);

Пример:
char str[] = “Компьютер”;
char ch = str[strspn(str,”Кмьоп”)]; //ch == ‘ю’

Обработка строк

Функции поиска первого символа в строке из заданного набора символов:

char * strpbrk(**const char** **str*, **const char** **charset*);

Пример:
char str[] = “Компьютер”;
char *ptr = strpbrk(str,”трм”);
char ch = *ptr; //ch == ‘м’

Обработка строк

Функция поиска следующего литерала в строке:

char * strtok(**char** * **restrict** *string*,
const char * **restrict** *charset*);

Например, необходимо подсчитать количество слов в строке. Слова разделяются пробелами или табуляциями. Строка содержится в переменной str. Фрагмент программы:

```
char *ptr = strtok(str,” \t”);
unsigned num = 0;
while(ptr != NULL){
  num++;
  ptr = strtok(NULL,” \t”);
}
```

Пример 1

Дана строка (максимум 15 символов), содержащая целое незнаковое число в троичной системе исчисления. Перевести число в семеричную систему исчисления. Полученный результат вывести на экран.

Пример 1

```
#include <stdio.h>
int main(int argc, char *argv[])
{
  char str[16];
  printf("Введите число: ");
  gets(str);
  unsigned num = 0, i;
  for(i=0;str[i]!=0;i++){
    num += str[i] - 48;
    if(str[i+1] != 0) num *= 3;
  }
```

Пример 1

```
for(i=0;num>0;i++){
  str[i] = num%7 + 48;
  num /= 7;
}
str[i] = 0;
for(int j=0;j<i/2;j++){
  char ch = str[j];
  str[j] = str[i-j-1];
  str[i-j-1] = ch;
}
printf("Результат: ");
puts(str);
return 0;
}
```

Пример 2

Дана строка (максимально 100 символов), содержащая слова, разделенные одним или несколькими пробелами, или знаками табуляции. Заменить все знаки табуляции знаком пробела, удалить двойные пробелы из строки. При реализации программы функции из библиотеки **string.h** не использовать.

Пример 2

```
#include <stdio.h>
int main(int argc, char *argv[])
{
  char str[101];
  printf("Введите строку: ");  gets(str);
  for(int i=0;str[i]!=0;i++)
    if(str[i] == '\t') str[i] = ' ';
  int j = 1;
  for(int i=1;str[i] != 0;i++){
    if((str[i] == ' ') && (str[i-1]==' ')) continue;
    str[j++] = str[i];
  }
  str[j] = 0;
  printf("Результат: ");  puts(str);
  return 0;
}
```

Пример 3

Дана строка (максимально 100 символов), содержащая слова, разделенные одним или несколькими пробелами, или знаками табуляции. Заменить все знаки табуляции знаком пробела, удалить двойные пробелы из строки. При реализации программы использовать функции из библиотеки **string.h**.

Пример 3

```
#include <stdio.h>
#include <string.h>
int main(int argc, char *argv[])
{
  char str[101];
  printf("Введите строку: ");  gets(str);
  do{
    int ind = strcspn(str,"\t");
    if(str[ind] == 0) break;
    str[ind] = ' ';
  }while(1);
  do{
    char *ptr = strstr(str,"  ");
    if(!ptr) break;
    strcpy(ptr,ptr+1);
  }while(1);
  printf("Результат: "); puts(str);
  return 0;
}
```

Пример 4

Дана строка (максимальная длина 100 символов), содержащая слова, разделенные пробелами или знаками табуляции. Число слов в строке не превышает 20, а длина каждого слова не более 10 символов. Упорядочить слова в строке по алфавиту.

Пример 4

```
#include <stdio.h>
#include <string.h>

int main(int argc, char *argv[])
{
  char str[101];
  printf("Введите строку: "); gets(str);
  char array[20][11];
  int count = 0;
  char *ptr = strtok(str,"\t ");
  for(count=0; ptr!=NULL; count++){
    strcpy(array[count],ptr);
    ptr = strtok(NULL,"\t ");
  }
```

Пример 4

```
int flag = 1;
while(flag){
  flag = 0;
  for(int i = 0;i<count-1;i++)
    if(strcmp(array[i],array[i+1])>0){
      char buffer[11];
      strcpy(buffer,array[i]);     strcpy(array[i],array[i+1]);
      strcpy(array[i+1],buffer);  flag = 1;
    }
 }
 strcpy(str,"");
 for(int i=0;i<count;i++){
  strcat(str,array[i]); strcat(str," ");
 }
 printf("Результат: "); puts(str);
 return 0;
}
```

Пример 5

Дан массив строк (максимально 25 строк, каждая строка не более 80 символов). Строки вводятся пользователем, признак завершения ввода – ввод пустой строки. Упорядочить строки по длине или по алфавиту (по выбору пользователя).

Пример 5

```
#include <stdio.h>
#include <string.h>
int main(int argc, char *argv[])
{
  char strs[25][81];
  unsigned count = 0;
  printf("Вводите строки:\n");
  for(int i=0;i<25;i++,count++){
    gets(strs[i]);
    if(strcmp(strs[i],"") == 0) break;
  }
  int type = 0;
  printf(" 0 – сортировать по алфавиту, \
            \n !0 – сортировать по длине: ");
  scanf("%d",&type);
```

Пример 5

```
int flag = 1;
 while(flag){
   flag = 0;
   for(int i = 0;i<count-1;i++)
     if((type&&(strlen(strs[i]) > strlen(strs[i+1])))||
        (!type&&(strcmp(strs[i],strs[i+1])>0))){
       char buffer[81];
       strcpy(buffer,strs[i]);
       strcpy(strs[i],strs[i+1]);
       strcpy(strs[i+1],buffer);
       flag = 1;
     }
 }
 printf("Результат:\n");
 for(int i=0;i<count;i++) puts(strs[i]);
 return 0;
}
```