Уральский федеральный университет

ИННОВАЦИОННАЯ ОБРАЗОВАТЕЛЬНАЯ ПРОГРАММА

Уральский федеральный университет

Составные типы в языке С

Строки

Лекция 10

Иллюстративный материал к лекциям по Информатике

Автор Саблина Н.Г. 2011 г.

Уральский федеральный университет

Содержание

Строки. Массивы символов

В языке С ***нет отдельного типа для строк***.

Работа со строками реализована ***с помощью одномерных символьных массивов*** (типа char).

Но есть разница между массивом символов и символьным массивом, содержащим строку.

Символьная строка - это одномерный массив типа char, ***заканчивающийся нулевым байтом*** – «нультерминальные строки».

Признак завершения строки

Нулевой байт - это байт, каждый бит которого равен нулю.

Для нулевого байта определена специальная символьная константа **'\0'.**

Если строка должна содержать N символов, то в описании массива следует указать N+1 элемент – дополнительный символ для завершающего нуля.

Строковые и символьные константы

Строковая константа - это список литер, заключенных в двойные кавычки.

Например,"Borland C++", "Это строковая константа".

В конец строковой константы компилятор добавляет символ '\0'. В памяти будет выглядеть так:

B	o	r	l	a	n	d		C	+	+	\0

Символьная константа содержит один символ клавиатуры, заключается в одинарные кавычки.
Например, ‘F’, ‘f’, ‘!’, ‘4’, ‘’ – символьные константы*

Примеры описаний символьных массивов

```
char S1[9]="ИРИТ-РТФ";
char S2[8]={'И', 'Р', 'И', 'Т', '-', 'Р', 'Т', 'Ф'};

char S3[ ]="ИРИТ-РТФ";
char S4[ ]={'И', 'Р','И', 'Т', '-','Р','Т','Ф'};
```

Обработка этих массивов

```
// работа с массивом символов
for (int i=0; i< sizeof(S1); i++)  printf ("%c" , S1[i]);
//работа с символьной строкой
printf("%s" , S2);
```

Ввод и вывод строк на консоль (1)

Ввод строк

- Использование функции **scanf( )** со спецификатором ввода %s, добавляет в конец вводимой строки ‘\0’.

```
char S1[80]; scanf (“%s”, S1);
```

- Использование команды **cin**

```
cin>>S1;
```

При этом вводятся символы до первого пробела.

Ввод и вывод строк на консоль (2)

Использование специальной библиотечной функцией **gets( )** (объявленна в файле stdio.h).

- Позволяет вводить строки, содержащие пробелы.
- Enter - окончание ввода. Добавляет в конец вводимой строки ‘\0’.

Вывод строк

- функции printf( ) или puts( ). Обе функции выводят содержание массива до первого нулевого байта.
- команда cout

Использование указателей для работы со строками (1)

Вся работа со строками символов в C++ осуществляется с помощью указателей на символ (тип char*).

Т.е. доступ к строке осуществляется через указатель на ее первый символ

Описание и инициализация

```
char *st = “Язык C++”; //st – хранит адрес строковой константы

char *st1 = st;      //При этом сама строка не копируется,
                     //копируется только указатель на нее
```

Использование указателей для работы со строками (2)

char *st ; ***//st указатель, будет хранить адрес***

gets(st); ***//опасная ситуация, память не выделена,***

//указатель не инициализирован

Нужно сначала

char *st ;

st= new char [80];

Потом можно

gets(st);

- *Динамично выделенную память нужно освобождать:* **delete [ ]st;**

Функции для работы со строками

- Подключить специальную библиотеку (заголовочный файл string.h)
- Наиболее часто используются функции
  - ✔ strlen() – определение длины строки
  - ✔ strcpy() – копирование строк
  - ✔ strcat() – сложение строк
  - ✔ strcmp() – сравнение строк
  - ✔ strstr() – поиск подстроки в строке

Функция strlen()

Вычисляет длину строки в символах (байтах).

Прототип функции имеет вид:

```
int strlen (const char * str);
```

аргумент - указатель на строку, длину которой нужно вычислить.

Длина строки определяется **без учета** ноль-символа.

Пример использования функции strlen()

Пример 1

```
char *str="Студент";
int i;
i=strlen(str);
printf("Ваша строка содержит %d символов.",i);
```

В результате работы программы на экран будет выведена фраза:

Ваша строка содержит 7 символов.

Пример 2

```
char st[100] ;
gets(st);
printf(“длина введенной строки %d символов.", strlen(st));
```

Функция strcpy()

- Прототип функции

char * strcpy (char *s1, const char *s2);

- Копирует содержимое строки s2 в строку s1.
- Возвращает адрес s1
- Массив s1 должен быть достаточно большим, чтобы в него поместилась строка s2.
- Если места мало,
  - компилятор не выдает указания на ошибку или предупреждения;
  - не прерывается выполнение программы,
- Но может привести к порче других данных или самой программы и неправильной работе программы в дальнейшем.

Пример использования функции strcpy()

```
char s1 [ ] = "Язык C++";
char s2[100] ;
strcpy(s2,s1);
```

- Теперь s2 содержит те же символы, что и s1, но указывает на другую область памяти, так что

s1 != s2.

С использованием указателей

```
char *s1 = "Язык C++";
char *s2 = new char [strlen(s1)+1]; // с учетом
                                    //завершающего '\0'
strcpy(s2,s1);
```

Функция strcat()

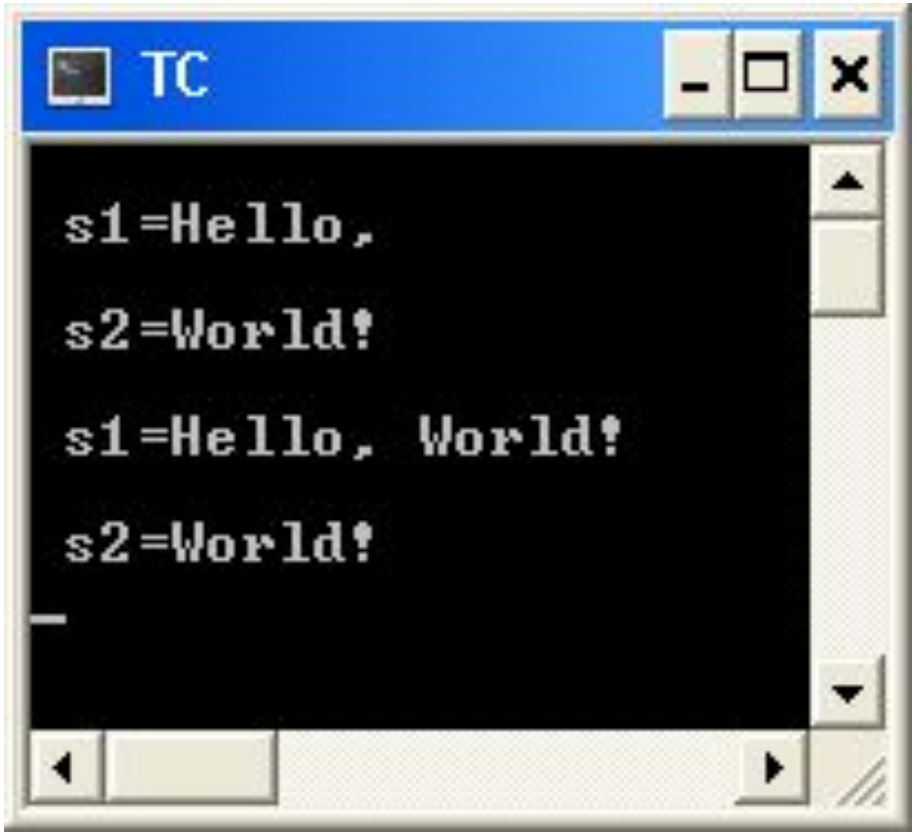
Прототип функции

char * strcat (char *s1, const char *s2);

- присоединяет строку s2 к строке s1 и помещает ее в s1, при этом строка s2 не изменяется.
- Нулевой байт, который завершал строку s1, будет заменен первым символом строки s2.
- Возвращает адрес s1

Пример использования функции strcat()

```
# include <string.h>
# include <stdio.h>
# include <conio.h>
main()
{clrscr();
char s1[20],s2[20];
strcpy (s1, "Hello, ");
strcpy (s2, "World!");
printf("\n s1="); puts(s1);
printf("\n s2="); puts(s2);
strcat(s1, s2);
printf("\n s1="); puts(s1);
printf("\n s2="); puts(s2);
getche();
}
```



Функция strcmp()

Прототип функции

int strcmp (const char *s1, const char *s2);

сравнивает строки s1 и s2 и возвращает значение 0, если строки равны, т. е. содержат одно и то же число одинаковых символов.

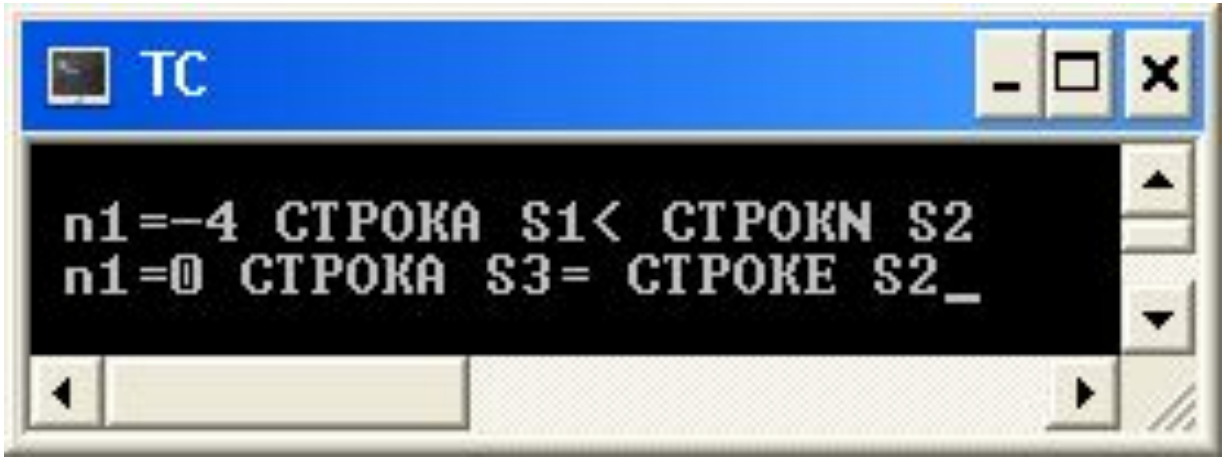
Результат сравнения – целое число, равное разности кодов первых несовпадающих символов.

Если символ первой строки больше, то возвращается положительное значение, если меньше, то отрицательное.

Если строки равны, возвращается 0.

Пример использования функции strcmp()

```
# include <string.h>
# include <stdio.h>
main()
{char s1[ ]="WENA",s2[ ]="WERA";  char s3[20];
int n1=strcmp(s1,s2); printf("\n n1=%d СТРОКА S1< СТРОКN S2", n1);
strcpy(s3,s2);
n1=strcmp(s3,s2);   printf("\n n1=%d СТРОКА S3= СТРОКЕ S2", n1);
}
```



Функция strstr()

Прототип функции

char* strstr (const char *s1, const char *s2);

- Ищет в строке *s1* подстроку *s2*.
- Результат – указатель на начало подстроки в строке или NULL (если подстрока не найдена).

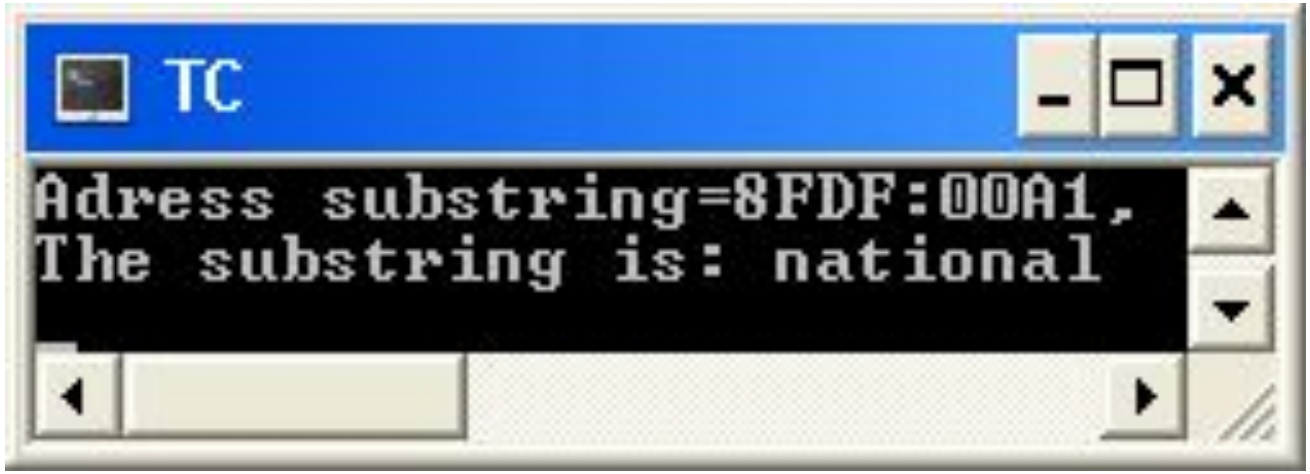
Поиск символа в строке

char* strchr (const char *s1, int c);

- Ищет в строке *s1* символ c.
- Результат – указатель на найденный символ или NULL (если символ не найден).

Пример использования функции strstr()

```
#include <stdio.h>
#include <string.h>
int main(void)
{char *str1 = "Borland International", *str2 = "nation", *ptr;
   ptr = strstr(str1, str2);
  printf("Adress substring=%p, \nThe substring is: %s\n",ptr, ptr);
return 0;
}
```



Перевод строковых данных в числовые

- **Строка в целое число**

int atoi (const char *s);

параметр – указатель на исходную строку

возвращает целое число

- **Строка в действительное число**

double atof (const char *s);

параметр – указатель на исходную строку

возвращает действительное число

Перевод числовых данных в строковые

целое в строку

char * itoa (int value, char *string, int radix);

длинное целое в строку

char * ltoa (long value, char *string, int radix);

длинное беззнаковое целое в строку

char * ultoa (unsigned long value, char *string, int radix);

radix – *основание системы счисления*

любое

```
char buffer[80];
sprintf ( buffer, «Число  Pi=%f\n", M_PI);
puts(buffer);
```

Примеры копирования строк посимвольно

```
char so[]="Строки", sn[10];
 int len=strlen(so);
   for (int i=0; i<len; i++) sn[i]=so[i];
    sn[i]='\0';

 for (int i=0; i<len; i++) sn[i]=*(so+i);
```

Примеры копирования строк посимвольно

```
//копирование строк 2
char so[ ]=”РИ-РТФ”,  sn[10];
char *sn1=sn, *so1=so;
    while (*sn1++=*so1++)  ;
   *sn1='\0';

do;
while (*sn1++=*so1++);
```

Массивы строк (двумерные символьные массивы)

- Часто двумерные массивы используются для работы с таблицами, содержащими текстовую информацию.
- Также очень часто используются массивы строк.

Пример заполнения массива

```
#include <stdio.h>
# include <string.h>
main()
{
char text[5][20];
strcpy (text[0], "Turbo Basic");
strcpy (text[1], "Turbo Pascal");
strcpy (text[2], "Borland C++");
strcpy (text[3]. "Turbo Prolog");
strcpy (text[4], "Paradox");
}
```

T	u	r	b	o		b	a	s	i	c	\0							
T	u	r	b	o		p	a	s	c	a	l	\0						
B	o	r	l	a	n	d		C	+	+	\0							
T	u	r	b	o		P	r	o	l	o	g	\0						
P	a	r	a	d	o	x	\0											

Задача

- Дан текст из нескольких строк. Из символов заданной строки составить новую строку, удалив из исходной строки все символы ‘a’.
- **Исходные данные**
- **n** – количество строк в тексте, целое, вводится с клавиатуры.
- **T** – исходный текст – массив строк, двумерный символьный массив.
- **k** – номер модифицируемой строки
- **Выходные данные**
- **S** – модифицированная строка

Метод решения

1. Создать массив для хранения исх. текста. Память выделять динамично в зависимости от кол-ва строк
2. Заполнить этот массив с клавиатуры (построчно).
3. Выбрать строку по ее номеру
4. Переписывать по букве в новую строку, пропуская букву ‘а’

Исходный текст программы (1)

```
#include <stdio.h>
#include <conio.h>
#include <string.h>
main()
{ int n,k,j;     char **;
   char s[80];
   printf ("\nСколько строк в тексте n=");
    scanf("%d",&n);
    fflush(stdin); //очистка буфера входного потока
```

Исходный текст программы (2)

```
printf("\nВведите  текст\n");
T=new char* [n];
for ( int i=0; i<n; i++)
{ T[i]=new char [80];  gets(T[i]); }
printf ("\nИсходный текст ");
for (i=0; i<n; i++) puts(T[i]);

 printf("\nКакую строку в тексте менять k=");
scanf("%d",&k);
```

Исходный текст программы (3)

```
printf ("\n Модифицированная строка \n");
 j=0;
 for (i=0; i<strlen(T[k]); i++)
    if ( T[k] [i] != 'a') s [j++]= T[k][i];
 s [j]='\0';
 puts(s);
}
```

Итоги

Рассмотренные вопросы:

- Типы в языке С
- Массивы символов
- Строки
- Указатели
- Описание строк

Библиографический список

- Подбельский В.В. Язык СИ++. Учебное пособие. М.: Финансы и статистика, 2003. – 560 с.
- Павловская Т.А. C/C++. Программирование на языке высокого уровня: учебник для студентов вузов, обучающихся по направлению "Информатика и вычисл. техника" СПб.: Питер, 2005. - 461 с.
- Березин Б.И. Начальный курс C и C++ / Б.И. Березин, С. Б. Березин. - М.: ДИАЛОГ-МИФИ, 2001. - 288 с
- Каширин И.Ю., Новичков В.С. От С к С++. Учебное пособие для вузов. – М.: Горячая линия – Телеком, 2005. – 334 с.

Автор:

Саблина Наталья Григорьевна

Ст. преподаватель

каф. РТС УГТУ-УПИ