# 27-28 ОАП Пр7 Работа со строками

## Теоретические сведения

Символьные строки хранят такую информацию, как имена файлов, названия книг, имена служащих и другие символьные сочетания. В C++ символьные строки хранятся в массиве типа char, который заканчивается символом NULL (или ASCII 0).

Для объявления символьной строки внутри программы просто объявите массив типа **char** с количеством элементов, достаточным для хранения требуемых символов. Например, следующее объявление создает переменную символьной строки с именем filename, способную хранить 64 символа (не забывайте, что символ NULL является одним из этих 64 символов):

**char** filename[64];

Главное различие между символьными строками и другими типами массивов заключается в том, как C++ указывает последний элемент массива.Программы на C++ представляют конец символьной строки с помощью символа NULL, который в C++ изображается как специальный символ *'\0'*. Когда мы присваиваем символы символьной строке, необходимо поместить его после последнего символа в строке. Например, следующая программа ALPHABET. CPP присваивает буквы от А до Я переменной alphabet, используя цикл **for**. Затем программа добавляет символ NULL в эту переменную и выводит ее с помощью cout.

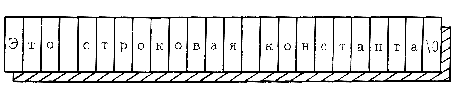
#include <iostream>   
**using namespace** std;  
**int** main(**int** argc, **char**\*\* atgv)   
{   
 **char** alphabet [34]; // 33 буквы плюс NULL char letter;   
 **int** index;   
 **for**(**char** letter=*'A'*, idx=0; letter<='Я'; letter++, idx++)  
 alphabet[idx]=letter;   
 alphabet[index]=NULL;   
 cout << *"Буквы "* << alphabet;   
 **return** 0;  
}

Когда выходной поток cout выводит символьную строку, он по одному выводит символы строки, пока не встретит символ NULL.

Практически все программы хотя бы раз используют константные строки, как показано ниже:

*"Это строковая константа"*

При создании символьной строковой константы компилятор C++ автоматически добавляет символ NULL.



Когда программы выводят символьные строковые константы с помощью выходного потока cout, cout использует символ NULL (который компилятор добавляет к строке) для определения последнего символа вывода.

Итак,

Символьная строка представляет собой массив символов, за которыми следует символ NULL ('\0'). При объявлении символьной строки вы объявляете массив типа char. Когда программа позднее присваивает символы строке, она отвечает за добавление символа NULL, который представляет конец строки.

Если вы используете строковые константы, заключенные в двойные кавычки, компилятор C++ автоматически добавляет символ NULL. Большинство функций C++ используют символ NULL для определения последнего символа строки.

При рассмотрении программ на C++ вы можете встретить символы, заключенные в одинарные кавычки (например, 'А') и символы, заключенные в двойные кавычки ("А"). Символ внутри одинарных кавычек представляет собой *символьную константу.* Компилятор C++ выделяет только один байт памяти для хранения символьной константы. Однако символ в двойных кавычках представляет собой *строковую константу —* указанный символ и символ NULL (добавляемый компилятором). Таким образом, компилятор будет выделять два байта для символьной строки.

Как известно из предыдущей темы, C++ позволяет инициализировать массивы при объявлении. Символьные строки C++ не являются исключением. Для инициализации символьной строки при объявлении необходимо указать требуемую строку внутри двойных кавычек, как показано ниже:

**char** title[64] = *"Учимся программировать на языке C++"*;

Если количество символов, присваиваемое строке, меньше размера массива, большинство компиляторов C++ будут присваивать символы NULL остающимся элементам строкового массива. Как и в случае с массивами других типов, если не указывать размер массива, который инициализируется при объявлении, компилятор C++ распределит достаточно памяти для размещения указанных букв и символа NULL:

**char** title[] = *"Учимся программировать на языке C++"*;

Для работы со строками используются функции библиотеки <cstring>

Вот некоторые из них.

1. **char** \*strcat(**char** \*str1, **const** **char** \*str2);  
   Функция strcat() присоединяет к строке str1 копию строки str2 и завершает строку str1 нулевым символом. Конечный нуль-символ, первоначально завершающий строку str1, перезаписывается первым символом строки str2. Строка str2 при этом не изменяется. Если заданные массивы перекрываются, поведение функции strcat() не определено.
2. **char** \*strchr(**const** **char** \*str, **int** ch);  
   Функция strchr() возвращает указатель на первое вхождение младшего байта параметра *ch* в строку *str*. Если указанный символ не найден, возвращается нулевой указатель.
3. **int** strcmp(**const** **char** \*str1, **const** **char** \*str2);  
   Функция strcmp() сравнивает в лексикографическом порядке две строки и возвращает целое значение, зависящее от результата сравнения.
4. **char** \*strcpy(**char** \*str1, **const** **char** \*str2);  
   Функция strcpy() копирует содержимое строки *str2* в строку *str1*. Параметр *str2* должен указывать на строку с завершающим нулевым символом. Функция strcpy() возвращает значение указателя *str1*.
5. size\_t strlen(**const** **char** \*str);  
   Функция strlen() возвращает длину строки, адресуемой параметром *str*, причем строка должна заканчиваться символом конца строки. Символ конца строки ( '0' ) не учитывается.
6. **char** \*strncat(**char** \*str1, **const** **char** \*str2, size\_t count);  
   Функция strncat() присоединяет к строке, адресуемой параметром *str1*, не более *count* символов строки, адресуемой параметром *str2*, завершая "результирующую" строку *str1* нулевым символом. Конечный нуль-символ, первоначально завершающий строку *str1*, перезаписывается первым символом строки *str2*. Строка *str2* в результате этой операции конкатенации не меняется. Если строки перекрываются, поведение функции strncat() не определено.
7. **int** strncmp(**const** **char** \*str1, **const** **char** \*str2, size\_t count);  
   Функция strncmp() сравнивает в лексикографическом порядке не более *count* символов из двух строк, заканчивающихся символом конца строки, и возвращает целое значение, зависящее от результата сравнения.
8. **char** \*strncpy(**char** \*str1, **const** **char** \*str2, size\_t count);  
   Функция strncpy() копирует не более *count* символов из строки, адресуемой параметром *str2*, в массив, адресуемый параметром *str1*. Параметр *str2* должен указывать на строку, заканчивающуюся символом конца строки.

Эта библиотека содержит и другие функции для работы со строками.

Следует, однако отметить, что такой способ работы со строками С++ унаследовал еще от С. От С ему досталась и библиотека <cstring>. В современный стандарт языка С++ входит библиотека STL (частью которой является, например, уже известный нам модуль <iostream>), в которой, в частности, есть и собственный класс для работы со строками.

#include <iosteam>  
#include <string>  
**using namespace** std;  
**int** main (i**nt** argc, **char**\*\* argv)  
{  
 string s;  
 s = *“мама мыла раму”*  
 cout << s;  
 **return** 0;  
}

## Контрольные вопросы

1. Как описываются строковые переменные?
2. Какие операции допустимы над строковыми данными?
3. В чем отличие строковой переменной от массива символов?
4. Какие стандартные функции для работы со строками вы знаете?
5. Чему равно значение x[0] после присваивания x='вопрос'?

## Задания к практической работе № 7

В следующих заданиях под словом «текст» понимается строка символов, слова в которой, разделены пробелами, ",", ".", "!", "?", ";", ":" (одним или несколькими).

1. Дан текст. а) Подсчитать количество слов в данной строке.
2. Подсчитать количество букв *а* в последнем слове данной строки.
3. Найти количество слов, начинающихся с буквы *б*.
4. Найти количество слов, у которых первый и последний символы совпадают между собой.
5. Найти длину самого короткого слова.
6. Составить программу циклической перестановки букв в словах текста так, что i-я буква слова становится i+1-ой, а последняя - первой.
7. В каждом слове текста замените "*а*" на букву "*е*", если "*а*" стоит на четном месте, и заменить букву "*б*" на сочетание "*ак*", если "*б*" стоит на нечетном месте.
8. Дан текст, содержащий от 2 до 30 слов, в каждом из которых от 2 до 10 латинских букв; между соседними словами – не менее одного пробела. Напечатать все слова, отличные от последнего слова, предварительно преобразовав каждое из них по следующему правилу: 1) перенести первую букву в конец слова; 2) перенести последнюю букву в начало слова.
9. Дан текст. Напечатать все слова, отличные от последнего слова, предварительно преобразовав каждое из них по следующему правилу: 1) оставить в слове только первые вхождения каждой буквы; 2) если слово нечетной длины, то удалить его среднюю букву
10. Составить таблицу слов данного текста, начинающихся с буквы "*А*", с указанием числа повторений каждого слова.
11. Составить программу для вычеркивания из слов текста всех букв, стоящих на нечетных местах после буквы "*а*".
12. Составить программы для перевода арабских чисел в римские и для обратной операции. Например, 255 = CCLV = сто + сто + пятьдесят + пять Замечание. Подобными алгоритмами перевода чисел из одной системы в другую мы пользуемся по нескольку раз на дню, когда ведем денежные расчеты. Сумма денег – это арабское число, которому соответствует определенный набор банкнот и монет (аналоги римских цифр).
13. Подсчитать, сколько букв надо исправить в слове Х, чтобы получилось слово Y (Х,Y – слова одинаковой длины).
14. Составить программу для подсчета числа одинаковых букв в словах X и Y равной длины, стоящих на одних и тех же местах.
15. Задано определенное количество конкретных сочетаний букв (например, УЩ, ЮЩ и др.). Определить, сколько таких групп символов содержится в тексте, вводимом с клавиатуры.
16. С клавиатуры вводится текст. Подсчитать и вывести на печать количество слов текста, начинающихся с гласной.
17. Для заданного текста определить длину содержащейся в нем максимальной серии символов, отличных от латинских букв.
18. Записать программу, выясняющую, можно ли из букв слова X составить слово Y.