Міністерство освіти і науки України
Харківський національний автомобільно-дорожній університет
Кафедра інформаційних технологій та мехатроніки

МЕТОДИЧНІ ВКАЗІВКИ

до лабораторних робіт з дисципліни

“Інформаційні технології”

“Програмування на мові С++ у середовищі Microsoft Visual Studio 2010”

для студентів напряму підготовки 6.050702 ” Електромеханіка ”,
галузь знань 0507 ” Електротехніка та електромеханіка ”

Розроблено та надруковано доц. Симбірським Г.Д.

Харків, 2018

Лабораторная работа № 8
ИССЛЕДОВАНИЕ ВОЗМОЖНОСТЕЙ VISUAL C++ 2010 ПО ОБРАБОТКЕ МАССИВОВ СИМВОЛЬНЫХ ПЕРЕМЕННЫХ (СТРОК)

1. Описание строк в среде Visual C++ 2010

В языке C++ массив типа **char** - это одномерный массив, состоящий из символов:

char A[11];

Символьная строка – это последовательность символов, дополненная специальным символом-ограничителем, указывающим конец строки. Ограничивающий символ записывается управляющей последовательностью **"\0"**. Для такой символьной строки применяют название "строка. С" (было предложено разработчиком языка). В других ветвях языка C++ существуют другие представления символьных строк.

Каждый символ в строке занимает один байт.

Символьная константа **"\0"** , ограничивающая символьную строку, называется нулевым байтом. Ее следует учитывать при определении соответствующего массива символов: если строка должна содержать **N** символов, то в определении массива следует указать **N + 1** элемент.

Например, определение

char A[11];

означает, что строка содержит 10 элементов типа **char** (символов), а последний байт зарезервирован для нулевого байта.

В качестве символов могут использоваться.

1. Прописные буквы латинского и российского алфавитов.
2. Строчные буквы латинского и российского алфавитов.
3. Цифры от 0 до 9.
4. Символы пунктуации: **. ;** и т. п.
5. Символьные константы.
6. Управляющие символы.
7. Пробел.
8. Шестнадцатеричные цифры.

Символьные массивы при их определении могут инициализироваться как обычный массив:

char A[13]={'K','h','a','r','k','o','v','-','2','0','1','4'};

а могут – как символьная строка, т. е. последовательность символов, заключенных в двойные кавычки:

char A[13]="Kharkov-2014";

Отличие этих двух способов заключается в том, что во втором случае автоматически будет прибавлен еще и нулевой байт. К тому же второй способ короче.

Для выделения места в памяти под символьный массив произвольного размера необходимо указать количество символов в строке (если оно известно) или задать явно больший размер массива.

char B[80]= "Это инициализация массива символов";

В данном случае указан размер массива 80, хотя для размещения этой строки необходимо было указать 35 (с учетом нулевого байта).

Инициализировать символьный массив можно и без указания его размера:

char B[ ]= "Это инициализация массива символов";

В этом случае компилятор сам определит необходимый размер памяти под этот массив.

Задание 8.1. Исследовать программу, в которой вводятся и выводятся символьные переменные. При выполнении задания опробуйте ввод и вывод различных символов, в том числе с пробелами.

```
#include "stdafx.h"
#include <conio.h>
#include "iostream"
using namespace std;
int _tmain(int argc, _TCHAR* argv[])
{
    char stroka[30], A[50];                                  //Объявление символьных переменных
    cout<<"Vvedite ctroku < 30 simvolov:"<<endl;
    cin>>stroka;                                             //Ввод символьной переменной stroka
    cout<<"Vu vveli stroku:"<<endl;
    cout<<stroka<<endl;                                      //Вывод символьной переменной stroka
    cout<<"Vvedite novuyu ctroku < 30 simvolov::"<<endl;
    cin>>stroka;                                             //Ввод новой символьной переменной stroka
    cout<<"Vu vveli novuyu stroku: "<<stroka<<endl;          //Вывод символьной переменной stroka
    cout<<"Vvedite novuyu ctroku < 50 simvolov::"<<endl;
    cin>>A;                                                  //Ввод символьной переменной A
    cout<<"Vu vveli novuyu stroku: "<<A<<endl;               //Вывод символьной переменной A
    cout<<"A0= "<<A[0]<<endl;                                //Вывод 0-го элемента переменной A
    cout<<"A8= "<<A[8]<<endl;                                //Вывод 8-го элемента переменной A
getch();
```

```
return 0;
}
```

После выполнения программы экран будет иметь следующий вид:

```
Vvedite ctroku  <  30  simvolov:
wwwwwwwwwwwqqqqqqqqq
Vu vveli stroku:
wwwwwwwwwwwqqqqqqqqq
Vvedite novuyu ctroku  <  30  simvolov::
aaaaaddddddffff
Vu vveli novuyu stroku: aaaaaddddddffff
Vvedite novuyu ctroku  <  50  simvolov::
ssssddddgggg
Vu vveli novuyu stroku: ssssddddgggg
A0= s
A8= g
```

Создайте в текстовом процессоре **Word** файл **Результат_Фамилия_Лр8**. Поля документа сделайте по 0,5 см.

Поместите окно DOS с результатами решения **Задания 8.1** в центральной части окна **Microsoft Visual Studio** ниже программного кода **Lr8-1.cpp** (см. рис. 1.2) и нажмите клавишу **<Prt Scr**>, после чего вставьте полученную копию экрана в файл **Результат_Фамилия_Лр8**. Над вставленным рисунком проставьте номер задания – **8-1**. Файл результатов не закрывайте до получения оценки за выполненную практическую часть работы в тетрадь с отчетом.

Закройте окно DOS, откройте пункт меню **Файл** и выполните команду **Закрыть решение.**

!Внимание! Результаты следующих заданий данной лабораторной работы сохраняйте строго в соответствии с приведенным выше порядком действий!

При вводе символов с пробелами, последние игнорируются операторами ввода >> и вывода <<. Поэтому при работе со строками вместо этих операторов целесообразней использовать следующую функцию:

getline(ИмяСимвольнойПеременной, РазмерСимвольнойПеременной);

где **ИмяСимвольнойПеременной** указывает на строку, в которую осуществляется ввод; **РазмерСимвольнойПеременной** – число символов, подлежащих вводу.

Задание 8.2. Исследовать программу, в которой вводятся и выводятся символьные переменные. При выполнении задания опробуйте ввод и вывод различных символов, в том числе с пробелами.

```
#include "stdafx.h"
#include <conio.h>
#include "iostream"
using namespace std;
int _tmain(int argc, _TCHAR* argv[])
{
char stroka[30], A[50];                          //Объявление символьных переменных
cout<<"Vvedite ctroku  <  30  simvolov:"<<endl;
cin.getline(stroka,20);                          //Ввод символьной переменной  stroka
cout<<"Vu vveli stroku:"<<endl;
cout<<stroka<<endl;                              //Вывод символьной переменной  stroka
cout<<"Vvedite novuyu ctroku  <  30  simvolov:"<<endl;
cin.getline(A,20);                               //Ввод символьной переменной  A
cout<<"Vu vveli novuyu stroku: "<<endl<<A;       //Вывод символьной переменной  A
getch();
return 0;
}
```

После выполнения программы экран будет иметь следующий вид:

```
Vvedite ctroku  <  30  simvolov:
www sss ttttt
Vu vveli stroku:
www sss ttttt
Vvedite novuyu ctroku  <  30  simvolov:
qqq yyyyyyy ppppppppp
Vu vveli novuyu stroku:
qqq yyyyyyy ppppppppp_
```

При использовании функции **getline() РазмерПеременной** меньше или равен размеру объявленной символьной строки.

Объявленная в вышеприведенной программе строка **stroka** может принять 70 символов. Например, если в функции **getline(stroka, 20)** указано число 20, то при вводе строки с 37 символами введется строка из 30 символов. Остальные символы будут отброшены.

2. Функции для обработки строк в среде Visual C++ 2010

Для работы со строками существуют специальные функции, описание которых находится в заглавном файле **string.h**, который необходимо включать в программу оператором **include**:

#include <string.h>;

Рассмотрим функции, которые используются наиболее часто.

2.1. Определение длины строки

Очень часто при работе со строками необходимо знать, сколько символов содержит строка. Для получения информации о длине строки используется функция **strlen()**. Вызов функции имеет вид:

strlen(Имя массива);

Функция возвращает значение на единицу меньшее, чем отводится под массив (без учета нулевого байта).

Задание 8.3. Исследовать программу, в которой вводятся и выводятся символьные переменные. При выполнении задания опробуйте ввод и вывод различных символов, в том числе с пробелами.

```
#include "stdafx.h"
#include <conio.h>
#include "iostream"
using namespace std;
int _tmain(int argc, _TCHAR* argv[])
{
char A[80];
int k;
cout<<"Vvedite ctroku  <  30  simvolov:"<<endl;
cin.getline(A,30);                          //Вызов функции getline() для ввода массива A
cout<<"Vu vveli stroku: "<<endl<<A;
k=strlen(A);                                //Вызов функции strlen(A) для определения количества
                                            //символов в массиве A
cout<<endl<<"k= "<<k<<endl;                 //Вывод переменной k (кол. символов в A)
getch();
return 0;
}
```

Вид экрана после работы программы:

```
Vvedite ctroku  <  30  simvolov:
wwwwwwwww  RRRRRRRRR  xxxxxxxxxx
Vu vveli stroku:
wwwwwwwww  RRRRRRRRR  xxxxxxx
k= 29
```

2.2. Копирование и присоединение строк

Значения строк могут копироваться из одной строки в другую. Копирование осуществляется с помощью следующих функций.

Функция **strcpy(S1,S2)** используется для побайтного копирования строки **S2** в строку **S1**. Копирование прекращается при достижении нулевого байта. Поэтому длина строки **S1** должна быть достаточно большой, чтобы в нее поместилась строка **S2**.

Задание 8.4. Исследовать использование функции **strcpy()**.

```
#include <string.h>                         //Добавление библиотеч. файла для работы со строками
#include "stdafx.h"
#include <conio.h>
#include "iostream"
using namespace std;

int _tmain(int argc, _TCHAR* argv[])
{
char A[80];
int k;
cout<<"Vvedite ctroku  <  30  simvolov:"<<endl;
cin.getline(A,30);                          //Ввод символьной переменной  A
cout<<"Vu vveli stroku: "<<endl<<A;
strcpy(A, "Proverka kopirovaniya");         //Вызов функции strcpy(A) для копирования строки в строку
cout<<endl<<"Novaya stroka: "<<A;           //Вывод новой символьной переменной  A
getch();
return 0;
}
```

Вид экрана после работы программы:

```
Vvedite ctroku  <  30  simvolov:
aaaaaaaaaaaaaaa ddddddd
Vu vveli stroku:
aaaaaaaaaaaaaaa ddddddd
Novaya stroka: Proverka kopirovaniya_
```

Функция **ctrncpy()** отличается от функции **ctrcpy()** тем, что включает еще один параметр. Он указывает количество символов, которые необходимо копировать из строки **S2** в строку **S1**. Функция имеет вид:

strncpy(S1, S2, n);

где **n** – количество символов (целое без знака).

Если длина **S1** меньше длины **S2**, то происходит урезание символов.

Задание 8.5. Исследовать использование функции **strncpy()**.

```
#include <string.h>
#include "stdafx.h"
#include <conio.h>
#include "iostream"
using namespace std;
int _tmain(int argc, _TCHAR* argv[])
{
char A[]="0123456789";                  //Ввод символьной переменной  A
char B[]="qwertyuiop";                  //Ввод символьной переменной  B
cout<<"S2= "<<A<<endl;                  //Вывод символьной переменной  A
cout<<"S1= "<<B<<endl;                  //Вывод символьной переменной  B
strncpy(B,A,4);
cout<<"S2new= "<<B<<endl;               //Вывод новой символьной переменной  B
getch();
return 0;
}
```

Вид экрана после работы программы:

```
S1= 0123456789
S2= qwertyuiop
S1new= 0123456789
S2new= 0123tyuiop
```

То есть, из строки **A** в строку **B** будут скопированы 4 первых символа и размещены в начале строки **B**.

2.3. Присоединение строк

Присоединение (**конкатенация**) строк используется для образования новой строки символов из двух и более исходных строк. Для этой цели используются функции

strcat(S1, S2) и strncat(S1, S2, n);

Функция **strcat(S1, S2)** присоединяет строку **S2** к строке **S1** и помещает ее в массив, где находилась строка **S1**. Строка **S2** не изменяется. Вновь полученная строка S1 автоматически завершается нулевым байтом.

Задание 8.6. Исследовать использование функции **strcat()**:

```
#include <string.h>
#include "stdafx.h"
#include <conio.h>
#include "iostream"
using namespace std;

int _tmain(int argc, _TCHAR* argv[])
{
char A[30], B[30];
strcpy(A, "Hello, ");                   //Копирование строки "Hello, " в строку A
strcpy(B, "World!");                    //Копирование строки "World!" в строку B
cout<<"A= "<<A<<endl;
cout<<"B= "<<B<<endl;
strcat(A, B);                           //Присоединение строки B к строке A
cout<<endl<<"A= "<<A<<endl;
cout<<"B= "<<B<<endl;
getch();
return 0;
}
```

Результат выполнения программы следующий

```
A= Hello,
B= World!

A= Hello, World!
B= World!
```

Функция **strncat(S1, S2, n)** также осуществляет присоединение строк, однако присоединяет лишь указанное в третьем параметре количество символов, например:

```
char S1[80]="Dlya prodolgeniya ";
char S2[80]="nagat knopku OK !";
strncat(S1,S2,7);
cout<<S1<<endl;
```

В результате на экран будет выведена строка:

Dlya prodolgeniya nagat knopku OK !

Задание 8.7. Самостоятельно создайте проект с использованием функций для работы со строками: **getline()**, **strlen ()**, **strcpy()** и **ctrncpy()**, **strcat()** и **strncat()**. Все функции должны быть последовательно использованы в одной программе. В качестве символьных строк необходимо использовать свою фамилию и фамилию следующего по списку в журнале студента (в латинском написании).

При разработке проекта отладьте программу вначале для одной функции, затем добавьте код для операций с другой функцией и отладьте программу уже для двух функций и т. д. После операций с каждой функцией должен быть организован вывод исходных строк и результата (по аналогии с **Заданиями 8.1 – 8.6**).

Контрольные вопросы

1. В программе на С++ определен массив **char A [11]**. Это означает, что строка содержит:
   1)10 символов
   2)11 символов
   3) 12 символов
2. В языке С++ для копирования строк используется функция:
   1) **strlen( )**
   2) **strcpy( )**
   3) **strcat( )**

3 В языке С++ конкатенация строк – это:
   1) Копирование одной строки в другой
   2) Сравнение двух строк
   3) Присоединение одной строки к другому

4. Что такое символьная строка?
5. Что такое нулевой байт?
6. Как инициализируется символьный массив?
7. Как объявляется символьный массив?
8. Для чего применяется функция **getline()**?
9. Какие аргументы используются в функции **getline()**?
10. Для чего применяется функция **strcpy()**?
11. Какие аргументы используются в функции **strcpy()**?
12. Для чего применяется функция **strcat()**?
13. Какие аргументы используются в функции **strcat()**?
14. Для чего применяется функция **strncat()**?
15. Какие аргументы используются в функции **strncat()**?