БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ФАКУЛЬТЕТ РАДИОФИЗИКИ И КОМПЬЮТЕРНЫХ ТЕХНОЛОГИЙ
КАФЕДРА ИНФОРМАТИКИ И КОМПЬЮТЕРНЫХ СИСТЕМ

Н. В. Серикова

ПРАКТИЧЕСКОЕ РУКОВОДСТВО
к лабораторному практикуму

«СТРОКИ»

ПО ДИСЦИПЛИНЕ

«ПРОГРАММИРОВАНИЕ НА C++»

2024
МИНСК

Практическое руководство к лабораторному практикуму «СТРОКИ» по дисциплине «ПРОГРАММИРОВАНИЕ НА С++» предназначено для студентов, изучающих базовый курс программирования на языке С++, специальностей «Компьютерная безопасность», «Прикладная информатика», «Радиофизика».

Руководство содержит некоторый справочный материал, примеры решения типовых задач с комментариями.

Автор будет признателен всем, кто поделится своими соображениями по совершенствованию данного пособия.

Возможные предложения и замечания можно присылать по адресу:
E-mail: Serikova@bsu.by

ОГЛАВЛЕНИЕ

СТРОКИ

Строка – последовательность символов кодовой таблицы ПЭВМ, обрабатываемая как единая структура. Строка может включать буквы, цифры и разнообразные специальные символы.

Такая последовательность должна быть ограничена, то есть нужно знать, где она заканчивается. Есть два варианта связать со строкой её размер: хранить его в отдельной переменной, либо ограничить строку специальным символом (элементом последовательности), дойдя до которого мы будем знать, что достигли конца.

Языки программирования имеют различные типы данных для представления строк и библиотеки функций для работы со строками.

Строки используются во многих приложениях для содержания текстовых данных, для организации интерфейса программы пользователя.

Элементом строки является символ.

Количество символов в строке определяет длину и размер строки.

Размер строки - общая длина строки, которая характеризует размер памяти, выделяемый строке при описании.

Длина строки - текущая длина строки (всегда меньше или равна размеру строки), которая показывает количество символов строки в каждый конкретный момент времени.

В каждом языке программирования существует **функция для определения текущей длины строки.** Размер строки всегда больше, чем длина строки. Это связано с тем, что в памяти, отводимой для строковой переменной, дополнительно хранится либо размер строки, либо нуль-символ.

Нуль-символ (\0) отмечает конец нуль-терминированной строки. Хотя нуль-символ записывается в виде двух символов, компилятор его интерпретирует как один символ, который может храниться в переменной символьного типа.

Строка, в которой нет символов, называется **пустой**. Длина пустой строки равна 0. Размер пустой строки равен 1. В языках программирования существует разница между символом и строкой длинны 1 символ.

При **объявлении строковых переменных**, как правило, необходимо указывать максимальную длину строки. В том случае, если длина строки не задана, для строковой переменной резервируется максимально возможный размер памяти, который отводится в данном языке программирования для строковых переменных.

Инициализация строковой переменной может происходить при объявлении или непосредственно в программе. Размер инициализирующей строки не должен превышать максимальной длины строковой переменной.

Операции сравнения выполняются над строками в лексикографическом порядке.

Лексикографический порядок – порядок сравнения строк, принятый во многих языках программирования.

При лексикографическом сравнении последовательно, начиная с первого символа, сравниваются коды символов строки. Те символы считаются больше, чьи коды больше, и наоборот. Сравнение происходит до первого несовпадения символов. Та строка считается большей, у которой код первого несовпадающего символа больше, и наоборот. Если начала строк совпадают, то более короткая строка считается меньше более длинной.

Кодировка (кодовая таблица) - это однозначное соответствие между целым число (кодом) и символом. Кодировки обычно составляют так, что символы следуют в алфавитном порядке и символ 'a' имеет наименьший код, а символ 'z' наибольший. Таким образом, выполняя лексикографическое сравнение строк можно разместить их в алфавитном порядке.

СТРОКИ СИМВОЛОВ В С++

Способы представления строк символов в С++

- в виде одномерного массива символов (строки, завершающиеся нулевым байтом) - С-строки;
- в виде объекта класса string;
- другие способы в зависимости от реализации.

С-Строки (завершающиеся нулевым байтом)

- Описываются как одномерный массив, каждый элемент которого имеет тип char.
- Символы строки последовательно располагаются в элементах массива, начиная с нулевого.
- В элемент массива, следующий за последним символом, автоматически записывается элемент с ASCII кодом 0: '\0'

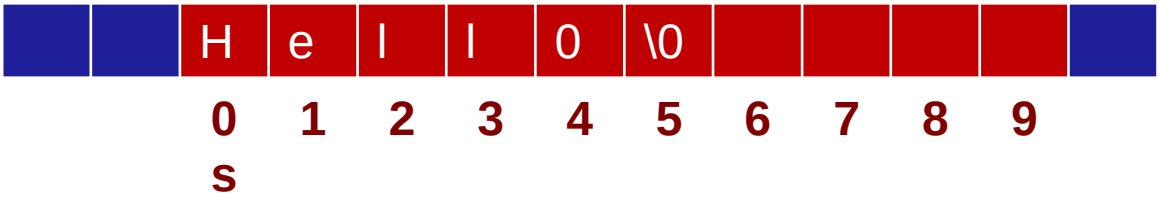
Описание строк, завершающихся нулевым байтом

- Строка описывается как обычный массив символов char

```
char s[ ] = "Hello";
char s[6] = "Hello";
char s[6] = {'H', 'e', 'l', 'l', 'o', '\0'};
```

- Размер массива должен быть на 1 больше длины самой длинной строки, которую предполагается в этом массиве хранить

```
char s[10] = "Hello";
```



Основные отличия строк, завершающих нулевым байтом, от массивов

- В процессе выполнения программы отслеживается текущая длина строки.
- Строка в целом может быть проинициализирована значением строкового литерала (а не только поэлементно символьными значениями).
- При помещении идентификатора символьного массива в поток cout выводится не адрес массива, а его содержимое от 0-го элемента до последнего перед завершающим символом \0.

Основные ограничения на действия со строками, завершающимися нулевым байтом (С-строками)

- Строки, как и обычные массивы, нельзя присваивать друг другу, используя оператор присваивания (включая присвоение строке строкового литерала где-либо, кроме начальной инициализации).
- К строкам, как к обычным массивам, нельзя применять операции сравнения.
- Для строк нет перегруженных операций (например, + для операции конкатенации).
- Для выполнения операций копирования, конкатенации, сравнения строк и ряда других операций приходится использовать встроенные функции, описанные в заголовке <string.h>

ФУНКЦИИ ПРЕОБРАЗОВАНИЯ (С-СТРОКИ)

STDLIB.H

	ИМЯ ФУНКЦИИ	ЗАПИСЬ	ОПИСАНИЕ
1	**atof**	double **atof**(char *str)	Преобразует строку str в вещественное число
2	**atoi**	int **atoi**(char *str)	Преобразует строку str в десятичное число
3	**atol**	long **atol** (char *str)	Преобразует строку str в длинное десятичное число
4	**_fcvt_s**	errno_t **_fcvt_s**(char* buffer, size_t strSize, double value, int count, int* dec, int* sign)	Преобразует число с плавающей точкой типа value в строку buffer. В случае успеха – возвращает значение 0. buffer – строка, содержащая результат конвертации. strSize – размер буфера buffer в байтах. value – число для конвертации. count – число цифр после запятой. dec – количество цифр целой части. sign – знак числа (0 или 1)
5	**_itoa_s**	char* **_itoa_s**(int value, char *str, size_t strSize, int baz)	Преобразует целое value в строку str. strSize – размер буфера str в байтах. baz – основание системы счисления 2<=baz<=36.
6	**_ltoa_s**	char * **_ltoa_s** (long value, char* str, size_t strSize, int baz)	Преобразует длинное целое value в строку str. strSize – размер буфера str в байтах. baz – основание системы счисления 2<=baz<=36.
7	**strtod**	double **strtod**(char* str1, char **str2)	Преобразует строку str1 в вещественное число, параметр str2 возвращает указатель на первый символ, который не может быть интерпретирован как часть числа (= nullptr, если преобразование выполнено корректно).
8	**strtol**	long **strtol**(char *str1, char **str2, int base)	Преобразует строку str1 в десятичное целое по основанию base, параметр str2 возвращает указатель на первый символ, который не может быть интерпретирован как часть числа (= nullptr, если преобразование выполнено корректно).
9	**_ultoa_s**	char *_**ultoa_s**(unsigned long value, char *str, size_t strSize, int baz)	Преобразует беззнаковое длинное целое value в строку str. strSize – размер буфера str в байтах. baz – основание системы счисления 2<=baz<=36.

ФУНКЦИИ ДЛЯ РАБОТЫ С С-СТРОКАМИ

STRING.H

	ИМЯ ФУНКЦИИ	ЗАПИСЬ	ОПИСАНИЕ
1	**strcat_s**	errno_t **strcat_s**(char *str1, size_t n, char *str2)	Выполняет конкатенацию (объединение) строк, записывая результат по адресу первого аргумента. **n** - размер массива str1.
2	**strchr**	char * **strchr** (char *str, char c)	Находит в строке str первое вхождение символа c. Если его нет, то возвращает nullptr.
3	**strcmp**	int **strcmp** (char *str1, char *str2)	Сравнивает строки str1 и str2. Результат <0, если str1<str2, =0, если str1=str2, >0, если str1>str2
4	**strcpy_s**	char* **strcpy_s**(char *str1, size_t n, char *str2)	Копирует строку str2 в строку str1. **n** - размер массива str1.
5	**strcspn**	int **strcspn**(char *str1, char *str2)	Определяет длину первого сегмента строки str1, содержащего символы, не входящие во множество символов строки str2. Если его нет, то возвращает nullptr.
6	**_strdup**	char* **_strdup**(char *str)	Дублирование строки str с выделением ей памяти
7	**strlen**	unsigned int **strlen**(char *str)	Вычисляет длину строки str, не включая нуль-символ.
8	**_strlwr_s**	char* **_strlwr_s**(char *str, size_t n)	Преобразует буквы верхнего регистра в строке str в буквы нижнего регистра. **n** - размер массива str.
9	**strncat_s**	errno_t **strncat_s**(char *str1, size_t n, char *str2, size_t kol)	Приписывает kol символов строки str2 к строке str1. **n** - размер массива str1.
10	**strncmp**	int **strncmp**(char *str1, char *str2, size_t kol)	Сравнивает строки str1 и str2., причем рассматриваются только первые kol символов. Результат <0, если str1<str2, =0 если str1=str2, >0, если str1>str2
11	**strncpy_s**	char* **strncpy_s**(char *str1, size_t n, char *str2, size_t kol)	Копирует kol символов строки str2 в строку str1. Функция не выполняет ни усечения, ни заполнение строки. **n** - размер массива str1.

	ИМЯ ФУНКЦИИ	ЗАПИСЬ	ОПИСАНИЕ
12	**strpbrk**	char * **strpbrk**(char *str1, char *str2)	Находит в строке str1 первое появление любого из множества символов, входящих в строку str2. Если его нет, то возвращает nullptr.
13	**strrchr**	char * **strrchr**(char *str, char c)	Находит в строке str последнее вхождение символа c. Если его нет, то возвращает nullptr.
14	**strrev**	char * **strrev**(char *str)	Инвертирование (реверс) строки str
15	**strspn**	int **strspn**(char *str1, char *str2)	Определяет длину первого сегмента строки str1, содержащего символы, входящие во множество символов строки str2. Если его нет, то возвращает nullptr.
16	**strstr**	char ***strstr**(char *str1, char *str2)	Определяет адрес первого вхождения в строку str1 подстроки str2. Если подстрока не найдена, то возвращается указатель nullptr.
17	**strtok_s**	char * **strtok_s**(char *str1, char *str2, char** sm)	Функция разбивает переданную ей строку (первый аргумент вызова) на лексемы в соответствии с заданным набором разделителей (второй аргумент). Функция возвращает указатель на первую из найденных лексических единиц или значение nullptr, если таких единиц в строке-аргументе нет. После выделения лексической единицы из аргумента str1 непосредственно за ней размещается символ с нулевым кодом. Функция изменяет разбиваемую на лексемы строку. **sm** – используется для хранения информации во время вызова функции. (Указатель на ячейку памяти)
18	**_strupr_s**	char* **_strupr_s**(char *str, size_t n)	Преобразует буквы нижнего регистра в строке str в буквы верхнего регистра. **n** - размер массива str.

Функции для работы со строками

Функции как правило возвращают 0, если функция выполняется корректно.

Для строк не определена операция присваивания, поскольку строка является не основным типом данных, а массивом. Присваивание выполняется с помощью функций стандартной библиотеки **strcpy_s** или **strncpy_s** или посимвольно «вручную».

Функция **strcpy_s**(char *s1**,** size_t n, char *s2) копирует все символы строки, указанной вторым параметром, включая завершающий 0, в строку, указанную первым параметром.

Функция **strncpy_s**(char *s1**,** size_t n, char *s2, size_t m) выполняет то же самое, но не более m символов. Если нуль-символ встретился в исходной строке раньше, то копирование прекращается, а оставшиеся до m символы строки s2 заполняются нуль-символами. В противном случае (если m меньше или равно длине строки s2) завершающий нуль-символ в s2 не добавляется. n – размер массива s1.

Функция **strlen**(char *s) возвращает фактическую длину строки s, не включая нуль-символ.

Функции **strcat_s** и **strncat_s** используются для объединения строк.

Функция **strcat_s**(char *s1**,** size_t n, char*s2) добавляет копию строки s2, включая завершающий нуль-символ, в конец строки s1. Первый символ строки s2 размещается в байте, в котором был записан нуль-символ строки s1. Если строка s1 заполнена, а s2 – не нулевая строка, то вызов strcat_s(s1, n, s2) перезапишет конец строки. Следите за длиной результирующей строки при конкатенации строк при вызове функции strcat_s(. . .). n – размер массива s1.

Функция **strncat_s**(char *s1**,** size_t n, char *s2**,** size_t m) позволяет добавить не более чем m символов строки s2 в конец строки s1.

Функция **strcmp**(char *s1, char *s2) позволяет сравнить 2 строки. Функция возвращает 0, если строки равны, отрицательное значение, если первая строка меньше, чем вторая, и положительное значение, если первая строка больше, чем вторая. Функция **strncmp**(char *s1, char *s2, n) проводит сравнение только первых n символов двух строк.

Функции **stricmp** и **strnicmp** позволяют сравнивать строки без учета регистра, т.е. без различия между прописными и строчными буквами.

Функция **strchr(const char *s, int c)** реализует поиск одиночного символа в строке. Она возвращает значение nullptr, если символ-аргумент в строке не найден, иначе возвращает указатель на этот символ.

Функция **strcspn**(char *s1, char *s2) - определяет и возвращает длину начальной части строки s1, состоящей из символов, не содержащихся в строке s2.

Функция **strspn**(const char *s1, const char *s2) - определяет и возвращает длину начальной части строки s1, состоящей только из символов, содержащихся в строке s2.

Функция **strpbrk**(char *s1, char *s2) - определяет позицию первого вхождения в строку s1 любого из символов строки s2. Если символ не найден, то возвращается указатель nullptr.

Функция **strstr**(char *s1, char *s2) - определяет позицию первого вхождения в строку s1 подстроки s2. Если подстрока не найдена, то возвращается указатель nullptr.

Функция char* **strtok_s**(char *s1, const char *s2, char** sm) интерпретирует строку s1 как последовательность лексических единиц (слов), разделённых ограничителями, которые заданы в строке-аргументе, соответствующей параметру s2. Функция возвращает указатель на первую из найденных лексических единиц или значение nullptr, если таких единиц в строке-аргументе нет. После выделения лексической единицы из аргумента s1 непосредственно за ней размещается символ с нулевым кодом. Таким образом, функция разбивает переданную ей строку (первый аргумент вызова) на лексемы в соответствии с заданным набором разделителей (второй аргумент). Эту функцию можно использовать для выделения в строке лексем-слов. Функция изменяет разбиваемую на лексемы строку, поэтому необходимо позаботиться о предварительном сохранении строки. sm – буфер, используемый между вызовами функции **strtok_s** для хранения начала следующей лексемы строки.

ВВОД С-СТРОК

При вводе строки с клавиатуры (помещение в поток cin) следует контролировать длину вводимой строки (чтобы она не превышала длину массива, где будет храниться строка

```
const int MAX = 100;
char str[MAX];
cin >> setw (MAX) >> str;  // Ввод до MAX-1 символов
```

При помещении строки в поток cin пробелы и символы новой строки конвертируются в завершающий нулевой байт

```
const int MAX = 100;
char str[MAX];
cin >> setw(MAX) >> str;
                // Введем строку "Good Morning"
cout << str;                  // Получим "Good"
```

Чтобы ввести строку, содержащую пробелы, необходимо вызвать встроенную функцию getline для потока cin:

```
char str[MAX];
cin.getline(str, MAX);
```

или можно задать свой разделитель строк:

```
char str[MAX];
char SEP = '$';
                // Задаем свой разделитель строк
cin.getline(str, MAX, SEP);
```

ПРИМЕР 1. Инициализация С-строки

```
#include <iostream>  // for cin cout
using namespace std;

void main()
{
   const int MAX = 8;
//нулевой символ добавляется к концу строки автоматически
   char str1[MAX] = "stroka1";

//нулевой символ добавляется к концу строки автоматически
   char str2[] = "stroka2";
                                       //  нет \0
   char str3[] = {'s', 't', 'r', 'o', 'k', 'a', '3'};

   char str4[] = {'s','t','r','o','k','a','4','\0'};

   char str5[MAX] = {0};
   char *str6 = "stroka6";

   cout << " 1 " << str1 << endl;
   cout << " 2 " << str2 << endl;
   cout << " 3 " << str3 << endl;
   cout << " 4 " << str4 << endl;

              //  !!!    нельзя - константный указатель
   str5 = "stroka5";
   cout << " 5 " << str5 << endl;
   cout << " 6 " << str6 << endl;

                 //  !!!     можно  !!!
   str6 = "stroka";
   cout << " 6 " << str6 << endl;
}
```

ПРИМЕР 2. Ввод С-строки. Инструкция cin

```
#include <iostream>  // for cin cout
using namespace std;

void main()
{
   const int MAX =  80;
   char str[MAX] = {0};

   cout << "Enter a string: ";
        //ввод выполняется до первого пробельного символа
   cin >> str;
   cout << "You entered: " << str << endl;
}
```

ПРИМЕР 3. Ввод С-строки. Функция gets

```
#include <stdio.h>              // for gets puts
#include <iostream>             // for cin cout
using namespace std;

void main()
{
   const int N =  80;
   char s[N] = {0};

   cout << "Enter a string: ";
   gets(s);
 // gets_s(s, n); // если массив задан через указатель,
                  // второй параметр позволяет ограничить
                  // количество вводимых символов,
                  // чтобы не перезаписать случайно
                  // память вне пределов выделенного
                  // буфера s
 // gets_s(s); // если массив статический
                // с длиной заданной константой,
                // то размер такого массива будет передан
                // в функцию gets_s автоматически.
                // Функция gets не имеет такой защиты
                // от переполнения буфера.
   cout << "You entered: ";
   puts(s);  // для этой функции указание
             // размера буфера s не требуется
}
```

ПРИМЕР 4. Ввод С-строки. Метод get

```
#include <iostream>  // for cin cout
using namespace std;

void main()
{
   const int MAX = 80;
   char str[MAX] = {0};
   char c = 0;

   cout << "1    Enter a string: ";
   // символ перевода строки '\n' остается в потоке
   // в строковую переменную добавляется '\0'
   cin.get(str, MAX);
   cout << "1    You entered: " << str << endl;

   cin.get();          // удаление из потока символа '\n'.
   cout << endl << "2    Enter a string: ";

              //   свой разделитель для ввода строки
   cin.get(str, MAX, '$');
   cout << "2    You entered: " << str << endl;
   c = cin.get();   // удаление из потока символа '$'.
   cout << c << endl;
   c = cin.get();    // удаление из потока символа '\n'.
   cout << int(c) << endl;

   cout << endl << "3    Enter a string: ";
   // символ перевода строки '\n' остается в потоке
   // в строковую переменную добавляется '\0'
   cin.get(str, MAX);
   cout << "3    You entered: " << str << endl;

   cin.get();     // удаление из потока символа '\n'.
}
```

ПРИМЕР 5. Ввод С-строки. Метод getline

```
#include <iostream>  // for cin cout
using namespace std;

void main()
{
   const int MAX = 80;
   char str[MAX] = {0};

   cout << endl << "1    Enter a string: ";
// при этом '\n' также считывается (удаляется) из потока
// и вместо него в строковую переменную записывается '\0'
   cin.getline(str, MAX);
   cout << "1    You entered: " << str << endl;

   cout << endl << "2    Enter a string: ";
// свой разделитель для ввода строки
// при этом '\n' также считывается (удаляется) из потока
// и вместо него в строковую переменную записывается '\0'
   cin.getline(str, MAX, '&');
   cout << "2    You entered: " << str << endl;
}
```

ПРИМЕР 6. Ввод С-строк в цикле

```
#include <string.h>
#include <iostream>  // for cin cout
using namespace std;

void main()
{
   const int N = 80;
   char s[N] = {0};

   // выход из цикла – конец файла – Ctrl+z
   while (cin.getline(s, N))
   {
      if (!strcmp(s, ""))
         break;
      cout << s << endl;
  //          .........              обработка строки
   }
}
```

ПРИМЕР 7. Ввод С-строки с русскими буквами

```
#include <windows.h>
#include <iostream>   // cin cout

using namespace std;

void main()
{
   char buff[80] = {0};

   cin.getline(buff,80);         // "русский текст"

   cout << "Моя программа" << endl;
                              // на экране набор знаков
   cout << buff << endl;
                         // на экране : "русский текст"

   setlocale(LC_ALL, "rus");

   cout << "Моя программа" << endl;
                        // на экране: "Моя программа"
   cout << buff << endl;
                        // на экране набор знаков

   OemToAnsi(buff, buff);

   cout << "Моя программа" << endl;
                         // на экране: "Моя программа"
   cout << buff << endl;
                         // на экране : "русский текст"
}
```

ПРИМЕР 8. Преобразование двоичного числа в десятичное

```
#include <string.h>
#include <iostream>  // for cin cout

using namespace std;

void main()
{
   char str[80] = {0};
   long int dec = 0;
   int v = 1;

            // двоичное число задаем в виде строки
   cout << "Enter number v 2 ss: ";
   cin.getline(str, 80);

   dec = 0;
   for (int i = strlen(str)-1; i >= 0; i--)
   {            // значение элемента строки 0 или 1
      if (str[i] == '1')
      {
         dec += v;        // добавляем к результату
         cout << v << " ";
      }
      v *= 2;          //   вес следующего разряда
   }
   cout << endl << "result v 10 ss = " << dec << endl;
}
```

ПРИМЕР 9. Преобразование десятичного числа в двоичное

```
#include <iostream>  // for cin cout
using namespace std;

void main()
{
   char str[80] = {0};
   int n = 0,
       i = 0;
   cout << "vvedite number v 10 ss: " << endl;
   cin >> n;
   do
   {
      int r = n % 2; // остаток от деления на основание
                     // с/с
      n = n / 2; // частное от деления на основание с/с
      str[i] = char('0' + r);
      i++;
   } while (n > 0);

   //  цифры результата в строке str в обратном порядке
   //  меняем порядок цифр на обратный
   for(int j = 0; j < i / 2; j++)
   {
      int c = str[j];
      str[j] = str[i - j - 1];
      str[i - j - 1] = c;
   }
   str[i] = '\0';  // конец строки i - номер позиции
   cout << "result v 2 ss = " << str << endl;
}

                  //  другое решение задачи
void main()
{
   const char MAX = 80;
   char str[MAX] = {0};
   int n = 0;

   cout << "vvedite number v 10 ss: " << endl;
   cin >> n;
   _itoa_s(n, str, MAX, 2);
   cout << "result v 2 ss = " << str << endl;
}
```

ПРИМЕР 10. Работа с символами С-строки

Задана строка символов. Получить новую строку из исходной, удалив символы '*' и удвоив букву A (a).

```
#include <string.h>
#include <iostream>   // for cin cout
using namespace std;


void main()
{
   const int MAX = 80;
   char  str1[MAX] = {0},
         str2[MAX] = {0};
   unsigned int  k = 0;

   cout << "Enter a string: ";
   cin.getline(str1, MAX);

   for (unsigned int i = 0; str1[i]; i++)
   {
      if (str1[i] == '*')     // если * пропускаем
         continue;

                  // если A или a удваиваем
      if (str1[i] == 'A'|| str1[i] == 'a')
         str2[k++] = str1[i];

// в остальных случаях записываем в результирующую строку
      str2[k++] = str1[i];
   }

   str2[k] = '\0';            // конец строки
   cout << "Rezult = " << str2 << endl;
}
```

ПРИМЕР 11. Копирование С-строк. Функция strcpy_s

```
#include <string.h>
#include <iostream>  // for cin cout

using namespace std;

void main()
{
    const unsigned int MAX = 20;
    char st[MAX] = "********************";    // 19 '*'
    unsigned int err = 0;

    for (unsigned int i = 0; i < MAX;  i++)
        cout << st[i];
    cout << endl << st << " " << strlen(st) << endl;

    strcpy_s(st, MAX, "Good Morning");

    for (unsigned int i = 0;  i < MAX;  i++)
        cout << st[i];
    cout << endl << st << " " << strlen(st) << endl;

    err = strcpy_s(st, MAX, "Hello World");
    if (!err)
        for (unsigned int i = 0;  i < MAX;  i++)
            cout << st[i];
    cout << endl << st << " " << strlen(st) << endl;


    strcpy_s(st, MAX, "Hello");

    for (unsigned int i = 0; i < MAX;  i++)
        cout << st[i];
    cout << endl << st << " " << strlen(st) << endl;
    cout << endl;
}
```

ПРИМЕР 12. Сравнение С-строк. Функции strcmp и strlen

```
#include <string.h>
#include <iostream>  // for cin cout

using namespace std;

void  main()
{
    const int MAX = 80;
    char buffer[MAX] = "";

    while (true)
    {
       cout << "Enter a string of less than 80 chars:\n";
       cin.getline(buffer, MAX);

       if (!strcmp(buffer, ""))
           break;

       int count = 0;
       for (count =0; buffer[count] != '\0'; count++);

       cout << count << "  " << strlen(buffer) << endl;
    }
}
```

ПРИМЕР 13. Объединение С-строк. Функции strcat_s, strcncat_s

```
#include <string.h>          // for strcat_s strncat_s
#include <iostream>             // for cin cout
using namespace std;

void main()
{
   const int MAX = 80;
   char s1[MAX] = "1";
   char s2[MAX] = "22222";
   char s3[MAX] = "333";

   cout << "1 " << endl;
   cout << " str1 = " << s1 << endl;
   cout << " str2 = " << s2 << endl;
   cout << " str3 = " << s3 << endl;

   cout << "2 " << endl;
   cout << " str1 = " << s1 << endl;
   strcat_s(s1, MAX, s2);
   cout << " str1 = " << s1 << endl;

   cout << "3 " << endl;
   unsigned int err = strcat_s(s1, MAX, s2);
   if (!err)
      cout << " str1 = " << s1 << endl;

   cout << "4 " << endl;
   strncat_s(s3, MAX, s2, 2);
   cout << " str3 = " << s3 << endl;

   cout << "5 " << endl;
   err = strncat_s(s3, MAX, s2, 2);
   if (!err)
      cout << " str3 = " << s3 << endl;
}
```

ПРИМЕР 14. Вхождение символа в С-строку. Функция strchr

```
#include <string.h>
#include <iostream>  // for cin cout

using namespace std;

void main()
{
   const int MAX = 80;
   // строка в которой будем искать символы
   char s[MAX] = "1234567890abcdefghABCDEFGH";
   char c = 0;

   cout << "c = ";
   cin >> c; // вводим искомый символ

   while(c != '!') // пока не введён символ '!'
   {
      char* str = strchr(s, c);
      if (str != nullptr)
      {     //  определение позиции найденного символа
         int n = int(str - s) ;
         cout << "simvol = " << c
              << " v " << n << "pozicii"
              << endl;
      }
      else
         cout << "simvola = " << c << " net" << endl;

      cout << "c = ";
      cin >> c; // вводим искомый символ
   }
}
```

ПРИМЕР 15. Вхождение С-подстроки в С-строку. Функция strstr

```
#include <string.h>
#include <iostream>  // for cin cout

using namespace std;

void main()
{
   const int MAX = 80;
   char s1[MAX] = "",
        s2[MAX] = "";

   cout << "s1 = ";
   cin.getline(s1, MAX);

   while(true)
   {
      cout << "s2 = ";
      cin.getline(s2, MAX);

      if (!strcmp(s2, ""))
         break;

      char* str = strstr(s1, s2);

      if (str != nullptr)
      {
            //  определение позиции найденной подстроки
         int n = int(str - s1);
         cout << "s2 v s1 c = " << n << "pozicii" << endl;
      }
      else
         cout << "s2 no s1" << endl;
   }
}
```

ПРИМЕР 16. Вхождение символов в С-строку. Функции strspn, strcspn

```
#include <string.h>
#include <iostream>  // for cin cout

using namespace std;

void  main()
{
   const int MAX =  80;
   char s1[MAX] = "",
        s2[MAX] = "";

   cout << "s1 = ";
   cin.getline(s1, MAX);

   while(true)
   {
      cout << "s2 = ";
      cin.getline(s2, MAX);

      if (!strcmp(s2, ""))
         break;

      int n1 = strspn(s1, s2);
      cout << "dlina pervoy 4asti s1 "
           << "tolko iz simvolov s2 = " << n1 << endl;

      int n2 = strcspn(s1, s2);
      cout << "dlina pervoy 4asti s1 "
           << "ne iz simvolov s2 = " << n2 << endl;
   }
}
```

ПРИМЕР 17. Определение С-подстроки из С-строки. Функция strpbrk

```
#include <string.h>
#include <iostream>  // for cin cout

using namespace std;

void main()
{
   const int MAX = 80;
   char s1[MAX] = "",
        s2[MAX] = "";

   cout << "s1 = ";
   cin.getline(s1, MAX);

   while(true)
   {
      cout << "s2 = ";
      cin.getline(s2, MAX);

      if (!strcmp(s2, ""))
         break;

      char* str = strpbrk(s1, s2);

      if (str != nullptr)
         cout << " podstroka s1 " << str << endl;
      else
         cout << " v s1 net simvolov iz s2" << endl;

   }
}
```

ПРИМЕР 18. Замена символов в С-строке из нижнего регистра в верхний. Функция _strupr_s

```
#include <string.h>
#include <iostream>  // for cin cout
using namespace std;

void main()
{
   const int MAX = 80;
   char str[MAX] = "";

   cout << "str = ";
   cin.getline(str, MAX);
   int err = _strupr_s(str, MAX);
   if (!err)
      cout << "rezult = " << str << endl;
}
```

ПРИМЕР 19. Замена символов в С-строке из верхнего регистра в нижний. Функция _strlwr_s

```
#include <string.h>
#include <iostream>  // for cin cout
using namespace std;

void main()
{
   const int MAX = 80;
   char str[MAX] = "";

   cout << "str = ";
   cin.getline(str, MAX);
   int err = _strlwr_s(str, MAX);
   if (!err)
      cout << "rezult = " << str << endl;
}
```

ПРИМЕР 20. Преобразование С-строки в число. Функции atoi, atof

```
#include <string.h>        // for strcmp
#include <stdlib.h>        // for atoi atof
#include <iostream>  // for cin cout

using namespace std;

void main()
{
   const int MAX = 80;
   char str[MAX] = "abcd";

   int i = atoi(str);
   cout << i << endl;

   while (true)
   {
      cout << "1 Enter a string: ";
      cin.getline(str, MAX);

      if (!strcmp(str, ""))
         break;

      i = atoi(str);
      cout << i << endl;
   }

   while (true)
   {
      cout << "2 Enter a string: ";
      cin.getline(str, MAX);

      if (!strcmp(str, ""))
         break;

      double f = atof(str);
      cout << f << endl;
   }
}
```

ПРИМЕР 21. Преобразование С-строки в число. Функция strtol

```
#include <string.h>        // strcmp
#include <stdlib.h>        // strtol
#include <iostream>        // cin cout
using namespace std;
void main()
{
   const int MAX = 80;
   char str[MAX] = "abcd";
   char *err = nullptr;
   int  base = 0;

   cout << "   1    " << endl;
   long l = strtol(str, &err, 10);
   cout << l << endl; // полученное число в 10-ой с/с
   cout << err << endl;
         // строка, содержащая ошибочное значение, если
         // перевод выполнен некорректно
         // = nullptr, если перевод выполнен корректно
   cout << "   2    " << endl;
   l = strtol(str, &err, 16);
   cout << l << endl;      // полученное число в 16-ой с/с
   cout << err << endl;

   while (true)
   {     cout << "Enter a string: ";
      cin.getline(str, MAX);

      if (!strcmp(str, ""))
         break;

      cout << endl << "Enter a base: ";
      cin >> base;
                           // число по основанию base
      l = strtol(str, &err, base);
      cout << l << endl;
      cout << err << endl;
   }
}
```

ПРИМЕР 22. Преобразование С-строки в число. Функция strtod

```
#include <string.h>        // strcmp
#include <stdlib.h>        // strtol
#include <iostream>        // cin cout
using namespace std;
void main()
{  const int MAX = 80;
   char  *err = nullptr;

   cout << "   1   " << endl;
   char* str = "1.1";
   doubled = strtod(str, &err);
   cout << d << endl;     // полученное число
   cout << err << endl;
         // = nullptr, если перевод выполнен корректно
   cout << "   2   " << endl;
   str = "1,1";
   d = strtod(str, &err);
   cout << d << endl;      // полученное число
   cout << err << endl;
         // строка, содержащая ошибочное значение, если
         // перевод выполнен некорректно
   cout << "   3   " << endl;
   str = "1e-3";
   d = strtod(str, &err);
   cout << d << endl;      // полученное число
   cout << err << endl;
            // = nullptr, если перевод выполнен корректно
   cout << "   4   " << endl;
   str = "e-3";
   d = strtod(str, &err);
   cout << d << endl;      // полученное число
   cout << err << endl;
         // строка, содержащая ошибочное значение, если
         // перевод выполнен некорректно
}
```

ПРИМЕР 23. Преобразование числа в С-строку. Функции _itoa_s, _ltoa_s, _utoa_s

```
#include <string.h>
#include <stdlib.h>
#include <iostream>  // cin cout
using namespace std;

void main()
{
   const int MAX =  80;
   char str[MAX] = "";
   int a = 0;
   // преобразование целого числа в строку
   // функции
   // _itoa_s(int value, char *str,
   //         size_t strSize, int radix);
   // _ltoa_s(long value, char *str,
   //         size_t strSize, int radix );
   // _ultoa_s(unsigned long val, char *str,
   //          size_t strSize, int radix );
   // strSize – длина массива str
   // 2<=radix<=36 - основание с/с в которой получим
число

   cout << "Enter number: ";
   cin >> a;
   _itoa_s(a, str, MAX, 2);
   cout << " number in 2 s/s = " << str << endl;

   _ltoa_(a, str, MAX, 10);
   cout << " number in 10 s/s = " << str << endl;

   _ultoa_s(a, str, MAX, 16);
   cout << " number in 16 s/s  = " << str << endl;

   _itoa_s(a, str, MAX, 22);
   cout << " number in 22 s/s = " << str << endl;
}
```

ПРИМЕР 24. Преобразование числа в С-строку. Функция _fcvt_s

```
#include <string.h>
#include <stdlib.h>
#include <iostream>  // cin cout
using namespace std;

void main()
{
   // преобразование вещественного числа в строку
   // _fcvt_s(char* buffer, size_t sizeBuffer,
   //         double value, int count,
   //         int *dec, int *sign )
   // В случае успеха – возвращает значение 0
   // buffer – строка, содержащая результат конвертации
   // sizeBuffer – размер строки buffer в байтах
   // value – число для конвертации
   // count – число цифр после запятой
   // dec – количество цифр целой части
   // sign – знак числа (0 или 1)

   const int MAX =  80;
   int  decimal, sign, err;
   char *buffer;
   double source;

   cout << "Enter number: ";
   cin >> source;

   err = _fcvt_s(buffer, MAX, source, 7, &decimal, &sign);
                 // результат в buffer
                 // в случае успеха возвращает 0 в err
   if (err != 0)
      cout << "error" << err << endl;
   else
      cout << source << " " << buffer << " "
           << decimal << " " << sign << endl;
}
```

ПРИМЕР 25. Выделение лексем. Функция strtok_s

```
#include <string.h>
#include <iostream>  // for cin cout
using namespace std;
void main()
{
   char str[80];
                        //строка из разделителей
   const char razd[] = " ,?!;-.";
   char *context = nullptr;

   cout << "Enter a string: ";
   cin.getline(str, 80);

   // первый вызов функции первый параметр – строка,
   // второй – строка из разделителей лексем
   char* Ptr = strtok_s(str, razd, &context);

   int i = 0;
   while (Ptr != nullptr)
   {
      cout << Ptr << endl;    // очередная лексема
      i++;                 // количество лексем
         // вызов функции для выделения следующей лексемы
         // если первый параметр = nullptr, то продолжаем
         // работать со строкой переданной ранее,
         // указатель между обращениями к функции strtok_s
         // сохраняется в переменной context
      Ptr = strtok_s(nullptr, razd, &context);
   }
   cout << "vvedeno " << i << " lecsem" << endl;
   cout << str << endl;
                  // изменилась ли исходная строка ???
}
```

ПРИМЕР 26. Указатели и С-строки

```
#include <string.h>
#include <iostream>  // for cin cout
using namespace std;
void main()
{
   char str1[] = "stroka - array";
   char *str2 = "stroka - pointer";

   cout << str1 << endl;
   cout << str2 << endl;
     //  вычисление значения функции strlen
     //  на каждой итерации цикла
   for (int i = 0; i < strlen(str1); i++)
       cout << str1[i];          //  вывод посимвольно
   cout << endl;
     //  вычисление значения функции  strlen
     //  на каждой итерации цикла
   for (int i = 0; i < strlen(str2); i++)
       cout << str2[i];              // вывод посимвольно
   cout << endl;
            // вывод посимвольно через указатель  !!!
            //  эффективнее предыдущего
   for (char *p = str1; *p;  p++)
       cout << *p;
   cout << endl;
            // вывод посимвольно через указатель  !!!
            //  эффективнее предыдущего
   for (char *p = str2; *p;  p++)
       cout << *p;
   cout << endl;

   //  str1++;  //   !!!  нельзя  константный указатель
   str2++;      //   !!!  можно
   cout << str2 << endl; //  строка без первого символа

   str2 = "new stroka - pointer";    //  !!!   можно
   cout << str2 << endl;
                  //  !!! нельзя  константный указатель
   // str1 = "new stroka - array";
}
```

ПРИМЕР 27. Массивы С-строк

```
#include <iostream>  // for cin cout
using namespace std;

void main()
{
   char* pointer[] = {"11111",
                      "22222222",
                      "33333333333",
                      "44444444444"};

   char array[][12] = {"11111",
                       "22222222",
                       "33333333333",
                       "44444444444"};

   for (int i = 0; i < 4; i++)
      cout << pointer[i] << endl;
   cout << endl;

   for (int i = 0; i < 4; i++)
      cout << array[i] << endl;
   cout << endl;

   for (int i = 0; i < 4; i++)
   {
      for (int j = 0; j < 4; j++)
         cout << pointer[i][j];
      cout << endl;
   }
   Cout << endl;

   for (int i = 0; i < 4; i++)
   {
      for (int j = 0; j < 4; j++)
         cout << array[i][j];
      cout << endl;
   }
}
```

ПРИМЕР 28. Массивы С-строк

Запустите программу и объясните результаты.

```
#include <iostream>  // for cin cout
using namespace std;
void main()
{
   char   *c[] = {"11111",
                  "22222222",
                  "333333333333",
                  "444444444444"};
   char **cp[] = {c + 3, c + 2, c + 1, c};
   char ***cpp = cp;

   cout << "    1    " << endl;
   cout << "  &cpp = " <<   &cpp << endl;
   cout << "   cpp = " <<    cpp << endl;
   cout << "  *cpp = " <<   *cpp << endl;
   cout << " **cpp = " <<  **cpp << endl;
   cout << "***cpp = " << ***cpp << endl;

   cout << endl << "    2    " << endl;
   for (int i = 0; i < 4;  i++)
   {
      cout << " i = " << i << endl;
      cout << "  *cpp[i] = " <<  *cpp[i] << endl;
      cout << " **cpp[i] = " << **cpp[i] << endl;
      cout << "  *cp[i] =  " <<  *cp[i]  << endl;
      cout << " **cp[i] =  " << **cp[i]  << endl;
      cout << "   c[i] =   " <<   c[i]   << endl;
      cout << "  *c[i] =   " <<  *c[i]   << endl << endl;
   }

   cout << endl << "    3    " << endl;
   for (int i = 0; i < 4;  i++)
   {
      cout << " i = " << i << endl;
      cout << "  **(cpp+i)= " <<  **(cpp+i) << endl;
      cout << " ***(cpp+i)= " << ***(cpp+i) << endl;
      cout << "  **(cp+i)=  " <<  **(cp+i)  << endl;
      cout << " ***(cp+i)=  " << ***(cp+i)  << endl;
      cout << "   *(c+i)=   " <<   *(c+i)   << endl;
      cout << "  **(c+i)=   " <<  **(c+i)   << endl;
   }
}
```

ПРАВИЛА ПЕРЕВОДА ЧИСЕЛ ИЗ ОДНОЙ СИСТЕМЫ СЧИСЛЕНИЯ В ДРУГУЮ

1. 2 → 10

Для перевода двоичного числа в десятичное необходимо его записать в виде многочлена, состоящего из произведений цифр числа и соответствующей степени числа 2, и вычислить по правилам десятичной арифметики:

$$X_2 = A_n \cdot 2^{n-1} + A_{n-1} \cdot 2^{n-2} + A_{n-2} \cdot 2^{n-3} + ... + A_2 \cdot 2^1 + A_1 \cdot 2^0 + A_0 \cdot 2^{-1} + A_{-1} \cdot 2^{-2} + ...$$

Степени числа 2

n (степень)	0	1	2	3	4	5	6	7	8	9	10
2^n	1	2	4	8	16	32	64	128	256	512	1024

Пример. Число 11101000.01_2 перевести в десятичную систему счисления.

$$11101000.01_2 = 1 \cdot 2^7 + 1 \cdot 2^6 + 1 \cdot 2^5 + 0 \cdot 2^4 + 1 \cdot 2^3 + 0 \cdot 2^2 + 0 \cdot 2^1 + 0 \cdot 2^0 +$$

$$0 \cdot 2^{-1} + 1 \cdot 2^{-2} = 232.25_{10}$$

2. 16 → 10

Для перевода шестнадцатеричного числа в десятичное необходимо его записать в виде многочлена, состоящего из произведений цифр числа и соответствующей степени числа 16, и вычислить по правилам десятичной арифметики:

$$X_{16} = A_n \cdot 16^{n-1} + A_{n-1} \cdot 16^{n-2} + A_{n-2} \cdot 16^{n-3} + ... + A_2 \cdot 16^1 + A_1 \cdot 16^0 + A_0 \cdot 16^{-1} + ...$$

Степени числа 16

n (степень)	0	1	2	3	4	5	6
16^n	1	16	256	4096	65536	1048576	16777216

Пример. Число $FDA1_{16}$ перевести в десятичную систему счисления.

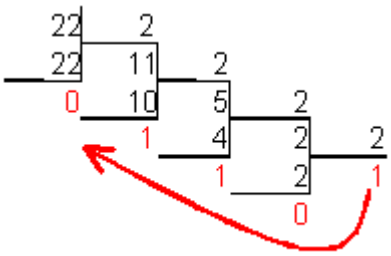
$$FDA1_{16} = 15 \cdot 16^3 + 13 \cdot 16^2 + 10 \cdot 16^1 + 1 \cdot 16^0 = 64929_{10}$$

3. 10 → 2

Вещественное число, в общем случае содержит целую и дробную часть, всегда можно представить в виде суммы целого числа и правильной дроби.

Для перевода целого десятичного числа в двоичную систему его необходимо последовательно делить на 2 до тех пор, пока не останется остаток, меньший или равный 1. Число в двоичной системе записывается как последовательность последнего результата деления и остатков от деления в обратном порядке.

Пример. Число 22_{10} перевести в двоичную систему счисления.



$22_{10}=10110_2$

Алгоритм перевода правильных дробей из 10 в другую систему р.

1. умножить исходную дробь в 10-ной системе счисления на основание р, выделить целую часть - она будет первой цифрой новой дроби;
2. отбросить целую часть;
3. для оставшейся дробной части операцию умножения с выделением целой и дробной части повторить, пока в дробной части не окажется 0 или не будет достигнута желаемая точность конечного числа;
4. записать дробь в виде последовательности цифр после ноля с разделителем в порядке их появления.

Пример. Число 0.3_{10} перевести в двоичную систему счисления.

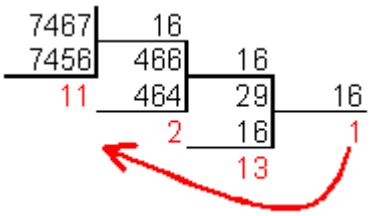
$0.3_{10} = 0.010011..._2$

0.3*2=0.6 → 0
0.6*2=1.2 → 1
0.2*2=0.4 → 0
0.4*2=0.8 → 0
0.8*2=1.6 → 1
0.6*2=1.2 → 1
...

4. 10 → 16

Для перевода целого десятичного числа в шестнадцатеричную систему его необходимо последовательно делить на 16 до тех пор, пока не останется остаток, меньший или равный 15. Число в шестнадцатеричной системе записывается как последовательность цифр последнего результата деления и остатков от деления в обратном порядке.

Пример. Число 7467_{10} перевести в шестнадцатеричную систему счисления.



$7467_{10}=1D2B_{16}$

Алгоритм перевода правильных дробей из 10 в другую систему р .

1. умножить исходную дробь в 10-ной системе счисления на основание р, выделить целую часть - она будет первой цифрой новой дроби;
2. отбросить целую часть;
3. для оставшейся дробной части операцию умножения с выделением целой и дробной части повторить, пока в дробной части не окажется 0 или не будет достигнута желаемая точность конечного числа;
4. записать дробь в виде последовательности цифр после ноля с разделителем в порядке их появления.

Пример. Число 0.125_{10} перевести в шестнадцатеричную систему счисления.

$0.125_{10} = 0.2_{16}$

0.125*16=2 → 2

5. 2 → 16

Чтобы перевести целое число из двоичной системы в шестнадцатеричную, его нужно разбить на тетрады (четвёрки цифр), начиная с младшего разряда, в случае необходимости дополнив старшую тетраду нулями, и каждую тетраду заменить соответствующей шестнадцатеричной цифрой.

Чтобы перевести дробную часть числа из двоичной системы в шестнадцатеричную, его нужно разбить на тетрады (четвёрки цифр), начиная со старшего разряда, в случае необходимости дополнив младшую тетраду нулями, и каждую тетраду заменить соответствующей шестнадцатеричной цифрой.

Пример. Число 1011100011.11_2 перевести в шестнадцатеричную систему счисления.

$$0010111000011.1100 = 2E3.C_{16}$$

6. 16 → 2

Для перевода шестнадцатеричного числа в двоичное необходимо каждую цифру заменить эквивалентной ей двоичной тетрадой.

Пример. Число $EE8.1F_{16}$ перевести в двоичную систему счисления.

$$EE8.1F_{16} = 111011101000.00011111_2$$

ЗНАЧЕНИЯ ДВОИЧНЫХ КОДОВ ДЛЯ ШЕСТНАДЦАТЕРИЧНЫХ ЦИФР

цифра	код	цифра	Код
0	0000	8	1000
1	0001	9	1001
2	0010	A	1010
3	0011	B	1011
4	0100	C	1100
5	0101	D	1101
6	0110	E	1110
7	0111	F	1111

СЛОВАРЬ ПОНЯТИЙ, ИСПОЛЬЗУЕМЫХ В ЗАДАНИЯХ

Текст – непустая последовательность символов.

Слово – непустая последовательность любых символов, кроме символов-разделителей.

Предложение – последовательность слов, разделенных одним или несколькими символами-разделителями.

Символы-разделители: «пробел», «.», «,», «:», «;», «!», «?», «-», «'», «(», «)».

Подслово – непустая подпоследовательность слова.

Обращение слова – слово, получающееся из исходного записью его букв в обратном порядке. Слово называется **симметричным**, если оно совпадает со своим обращением.

Вхождением слова (последовательности) *v* в слово (последовательность) *w* называется любая часть слова (последовательности) *w*, которая является подсловом (подпоследовательностью) *v* слова (последовательности) *w*.

Слова-серии – слова, составленные из повторяющихся подслов. Например, 123123123 – слово-серия.