**Строки**

Прежде чем пользоваться строковыми операциями и вообще иметь возможность описывать переменные типа string необходимо включить в заголовок программы описание:

#include <string>

Также во всех примерах предполагается, что в начале программы подключено пространство имен std:

using namespace std;

Описание строки полностью аналогично описанию любой другой переменной. Например:

string example\_str;

Над строками возможны следующие операции:

* [Присваивание](https://informatics.msk.ru/mod/page/view.php?id=25345#str_assign)
* [Сравнение](https://informatics.msk.ru/mod/page/view.php?id=25345#str_compare)
* [Ввод](https://informatics.msk.ru/mod/page/view.php?id=25345#str_input)
* [Запрос длины строки (size, length)](https://informatics.msk.ru/mod/page/view.php?id=25345#str_size)
* [Соединение (+)](https://informatics.msk.ru/mod/page/view.php?id=25345#str_plus)
* [Удаление фрагмента (erase)](https://informatics.msk.ru/mod/page/view.php?id=25345#str_erase)
* [Вставка фрагмента (insert)](https://informatics.msk.ru/mod/page/view.php?id=25345#str_insert)
* [Копирование фрагмента (substr)](https://informatics.msk.ru/mod/page/view.php?id=25345#str_substr)
* [Поиск фрагмента (find)](https://informatics.msk.ru/mod/page/view.php?id=25345#str_find)
* [Преобразование к числу (sscanf, sstream)](https://informatics.msk.ru/mod/page/view.php?id=25345#str_to_int)
* [Преобразование числа к строке (sprintf, sstream)](https://informatics.msk.ru/mod/page/view.php?id=25345#int_to_str)
* [Строка как массив символов](https://informatics.msk.ru/mod/page/view.php?id=25345#str_as_array)
* [Примечание(\*)](https://informatics.msk.ru/mod/page/view.php?id=25345#str_comment)

**Присваивание строк**

Примеры с комментариями:

string example\_str = "Do bats eat cats?"; // присваивание при описании

example\_str = ""; // присваивание пустой строки -

// две кавычки рядом, без пробелов

work\_str = example\_str; // присваивание между строками

Помимо этого возможно вот такое присваивание:

char a[50];

example\_str = a;

**Сравнение строк**

Две строки равны друг другу, если у них одинаковые длины и символы на соответствующих местах полностью совпадают (то есть 1-й символ строки A равен 1-му символу строки B, 2-й символ строки A равен 2-му символу строки B и так далее)

Например строка "abc" равна строке "abc", но не равна строке "abcd".

Выяснение какая из строк больше выполняется посимвольно, аналогично алфавитному порядку русского языка. Если строка начинается с символа, чей код больше, что и сама строка больше. Если первые символы одинаковы, то сравниваются вторые символы по тому же правилу. Если и они одинаковы, сравниваются третьи и так далее. Если в процессе сравнения какая-то из строк закончилась, то она меньше другой.

Строку также можно сравнивать с пустой (обозначается двумя кавычками подряд, без пробелов).

while (example\_str != "") {

// какие-то команды

}

**Ввод строк**

Для ввода строк рекомендуется использовать функцию *getline()*. Данная функция читает строку целиком (с пробелами и прочими символами) до знака перевода строки.

Пример:

getline(cin, example\_str);

**Длина строки**

Функции size() или length() возвращают целое число - количество символов, которое сейчас находится в строке.

Пример:

example\_str = "Password";

int x = example\_str.size();

// после этого фрагмента x = 8

Пример:

example\_str = "";

if (example\_str.length() > 0) {

// команды этого блока сейчас не выполнятся,

// так как длина пустой строки равна 0

};

**Соединение строк**

Операция + возвращает строку, получившуюся в результате соединения (дописывания одной строки после конца другой) одной или более строк. Строки соединяются в том порядке, как они указаны в операции.

Пример:

example\_str = "password";

article = "The "

comment\_str = "anyone can guess it."

example\_str = article + example\_str + " is so simple, so " + comment\_str;

// example\_str = "The password is so simple, so anyone can guess it."

**Удаление фрагмента**

Функция erase() удаляет указанный фрагмент строки (если это возможно). Первым параметром указывается начальный символ (с него начнется удаление), вторым - количество символов. Символы в строке нумеруются, начиная с 0.

Пример:

example\_str = "The star is so distant";

example\_str.erase(3, 6);

// example\_str = "Theis so distant"

// будет удалено слово star вместе с обоими пробелами

Пример:

example\_str = "The star is so distant";

example\_str.erase(3, 80);

// example\_str = "The"

// будет удалено столько символов, сколько возможно

**Вставка фрагмента**

С помощьью функции insert() можно добавить заданный фрагмент в любое место строки. Первым параметром функции является номер начальной позиции для фрагмента (нумерация начинается с 0), вторым - фрагмент. Функция изменяет исходную строку.

Пример:

example\_str = "123456";

example\_str.insert(3, "+");

// теперь example\_str = "123+456"

**Копирование фрагмента**

Функция substr() возвращает фрагмент строки (который тоже является строкой). Первым параметром указывается номер начального символа, вторым - количество. Символы нумеруются, начиная с 0. Функция не оказывает влияния на исходную строку.

Пример:

example\_str = "Do bats eat cats?";

new\_str = example\_str.substr(3, 5);

// new\_str = "bats "

// будет скопировано слово "bats" вместе с последуюшим пробелом

Пример:

example\_str = "Do bats eat cats?";

new\_str = example\_str.substr(12, 100);

// new\_str = "cats?"

// в строке нет 100 символов, поэтому будет скопировано сколько есть

**Поиск фрагмента**

С помощью функции find() можно определить, встречается ли данный фрагмент в строке. Функция возвращает специальное значение string::npos, если фрагмент не найден или номер первого (начиная слева) места, откуда начинается подобный фрагмент

Пример:

example\_str = "comandante@cia.dark.org";

x = example\_str.find("z");

// в этом примере x равно string::npos,

// так как в строке нет ни одной буквы z

Пример:

example\_str = "comandante@cia.dark.org";

x = example\_str.find("a");

// в этом примере x равно 3,

// символов "а" в строке несколько, но find

// вернет номер только для первого слева

Пример:

example\_str = "comandante@cia.dark.org";

if (example\_str.find("te") != string::npos) {

// команды этого блока выполнятся, так как

// в строке есть фрагмент "te"

}

**Преобразование к числу**

1. Ввод/вывод в stringstream. Требуется создать специальный поток ввода/вывода, который будет вводить или выводить данные в некоторый буффер в памяти. При этом операции будут сходны со стандартным вводом/выводом.

Пример:

#include <sstream>

stringstream container;

// создали специальный поток ввода/вывода,

// который "работает" в некоторой области памяти

string data = "918";

int x;

container.clear();

// на случай, если в container что-то лежало, чтобы

// случайно не попало в результат преобразования

container << data;

// кладем в поток строку

container >> x;

// достаем из потока это же число в виде int

2. Функция sscanf() (scan from string as fromatted). Данная функция также из обычного C (не C++) поэтому "умеет" работать только с символьными массивами. В грубом приближении у функции 3 параметра: символьный массив, строка формата и переменная (или переменные), куда следует поместить результаты преобразования.

Пример:

#include <cstio>

string some\_value = 825;

x = sscanf(some\_value.data(), "%d", &x);

// в x записано число типа int, извлеченное из строки

// функция data() позволяет получить символьный массив

// в формате языка C (не C++), с таким же содержанием,

// что и строка

**Преобразование числа к строке**

1. Ввод/вывод в stringstream. Требуется создать специальный поток ввода/вывода, который будет вводить или выводить данные в некоторый буффер в памяти. При этом операции будут сходны со стандартным вводом/выводом.

Пример:

#include <iostream>

#include <sstream>

#include <string>

using namespace std;

stringstream container;

// создали специальный поток ввода/вывода,

// который "работает" в некоторой области памяти

string data;

int x = 16;

container.clear();

// объясняем потоку, что в нем нет данных

// чтоб не получить при чтении то, что не клали туда сами

container << x + 8;

// кладем в поток число типа int

container >> data;

// достаем из потока это же число в виде строки

2. sprintf() (print formatted to string). Эта функция также является функцией из стандартного C, и тоже работает только с массивами из char). Параметры (вкратце): символьный массив, куда будет выведен результат, строка формата, данные.

Пример:

#include <iostream>

#include <string>

#include <cstdio>

int x;

string s;

char z[100];

x = 92;

sprintf(z, "%d", x);

s = z;

**Строка как массив символов**

Строку можно рассматривать как массив символов и применять к ней методы работы, как с массивом.

Пример:

string example\_str;

for (int i = 0; i < example\_str.size(); i++) {

if (example\_str[i] == '\*') {

// некие команды, которые будут выполнятся для всех

// символов '\*', находящихся в строке example\_str

}

}

**Примечание(\*)**

На самом деле, здесь представлены далеко не все функции работы со строками. Более того, некоторые функции, например find(), допускают разные способы использования с разным количеством и комбинациями параметров. Более подробно см. документацию по языку C++. Однако, для решения большинства задач перечисленных функций вполне достаточно.