

Программирование на языке C++

...

C++ String

std::string

Строки - объекты, представляющие
собой последовательность
СИМВОЛОВ

Не является встроенным типом
языка, но входит в состав
стандартной библиотеки.

```
#include <string>
#include <iostream>

int main()
{
    std::string str = "I'm C++ string!";
    std::cout << str << std::endl;
}
```

```
I'm C++ string!
```

Размер строки

`size_t size() const;`
`size_t length() const;`

```
#include <string>
#include <iostream>

int main()
{
    std::string str = "I'm C++ string!";
    std::cout << "Str      = " << str << std::endl;
    std::cout << "Str size = " << str.size() << std::endl;

    str = "New text..";
    std::cout << "Str      = " << str << std::endl;
    std::cout << "Str size = " << str.length() << std::endl;
}
```

```
Str      = I'm C++ string!
Str size = 15
Str      = New text..
Str size = 10
```

Проверка пустой строки

`bool empty() const;`

Флаги форматирования

потока вывода:

`std::boolalpha`

`std::noboolalpha`

```
#include <string>
#include <iostream>
```

```
int main()
{
    std::string str1 = "Not empty string";
    std::string str2 = "";

    std::cout << "Is str1 empty: " << str1.empty() << std::endl;
    std::cout << "Is str2 empty: " << str2.empty() << std::endl;

    std::cout << std::boolalpha;
    std::cout << "Is str1 empty: " << str1.empty() << std::endl;
    std::cout << "Is str2 empty: " << str2.empty() << std::endl;
}
```

```
Is str1 empty: 0
Is str2 empty: 1
Is str1 empty: false
Is str2 empty: true
```

Конкатенация строк

```
#include <string>
#include <iostream>

int main()
{
    std::string str1 = "First";
    std::string str2 = "Second";

    std::string resultStr = str1 + " and " + str2;
    resultStr += " and 3rd";

    std::cout << resultStr << std::endl;
}
```

First and Second and 3rd

Работа с потоком ввода

std::cin

operator>> возвращает символы до ближайшего пробельного символа.

Остальные символы остаются в
std::cin

```
#include <string>
#include <iostream>

int main()
{
    std::string title;
    std::cout << "Enter the title of the book: ";
    std::cin >> title;

    std::cout << "Title: " << title << std::endl;
}
```

```
Enter the title of the book: Effective Modern C++
Title: Effective
```

Работа с потоком ввода

std::cin

std::getline принимает 2 параметра
- поток ввода и строку назначения

std::ws - манипулятор ввода.
Игнорирует лидирующие
пробельные символы

```
#include <string>
#include <iostream>

int main()
{
    std::string title, author;
    std::cout << "Enter the title of the book: ";
    std::getline(std::cin >> std::ws, title);

    std::cout << "Enter the author of the book: ";
    std::getline(std::cin >> std::ws, author);

    std::cout << "Title: " << title << std::endl;
    std::cout << "Description: " << author << std::endl;

    return 0;
}
```

```
Enter the title of the book: Effective Modern C++
Enter description: Scott Meyers
Title: Effective Modern C++
Description: Scott Meyers
```

Преобразование в целые числа

```
int stoi( const std::string& str,  
std::size_t* pos = nullptr, int base = 10 );
```

```
long stol( const std::string& str,  
std::size_t* pos = nullptr, int base = 10 );
```

```
long long stoll( const std::string& str,  
std::size_t* pos = nullptr, int base = 10 );
```

```
#include <string>  
#include <iostream>
```

```
int main()  
{  
    std::string str{ "164" };  
    int intValue = std::stoi(str);  
    std::cout << "Integer value: " << intValue << std::endl;  
}
```

```
Integer value: 164
```


Преобразование в целые числа

Методы `stoi`, `stol` и `stoll` в качестве
третьего необязательного
параметра принимают
основание системы счисления
(по умолчанию 10)

```
#include <string>
#include <iostream>

int main()
{
    std::string str{ "0xA4" };
    int intValue = std::stoi(str, 0, 16);
    std::cout << "Integer value: " << intValue << std::endl;
}
```

```
Integer value: 164
```

Преобразование в целые числа

Методы `stoi`, `stol` и `stoll` в качестве
третьего необязательного
параметра принимают
основание системы счисления
(по умолчанию 10)

```
#include <string>
#include <iostream>

int main()
{
    std::string str{ "10100100" };
    int intValue = std::stoi(str, 0, 2);
    std::cout << "Integer value: " << intValue << std::endl;
}
```

```
Integer value: 164
```

Некорректное преобразование

```
#include <string>
#include <iostream>
```

```
int main()
{
    std::string str{ "Text" };
    int intValue = std::stoi(str);
    std::cout << "Integer value: " << intValue << std::endl;
}
```

Signal SIGABRT (signal SIGABRT)

Преобразование в числа с плавающей точкой

```
float stof( const std::string& str,  
            std::size_t* pos = nullptr );
```

```
double stod( const std::string& str,  
             std::size_t* pos = nullptr );
```

```
long double stold( const std::string& str,  
                  std::size_t* pos = nullptr );
```

```
#include <string>  
#include <iostream>  
#include <iomanip>  
  
int main()  
{  
    std::string str{ "3.14159265358979323846264338327950288419" };  
  
    const float fValue = std::stof(str);  
    const double dValue = std::stod(str);  
    const long double ldValue = std::stold(str);  
  
    std::cout << std::setprecision(32);  
    std::cout << "Float value      : " << fValue << std::endl;  
    std::cout << "Double value     : " << dValue << std::endl;  
    std::cout << "Long Double value : " << ldValue << std::endl;  
}
```

```
Float value      : 3.1415927410125732421875  
Double value     : 3.1415926535897931159979634685442  
Long Double value : 3.1415926535897932385128089594062
```

Получение C-string из std::string

```
const CharT* c_str() const;  
const CharT* data() const;
```

```
#include <string>  
#include <iostream>  
  
class dbConnection;  
  
dbConnection* OpenDBConnection(const char* dbName);  
void CloseDBConnection(dbConnection* connection);  
  
int main()  
{  
    std::string dbName = "...";  
    // ...  
    dbConnection* connection = OpenDBConnection(dbName.c_str());  
    // ...  
    CloseDBConnection(connection);  
}
```

Внутреннее представление std::string

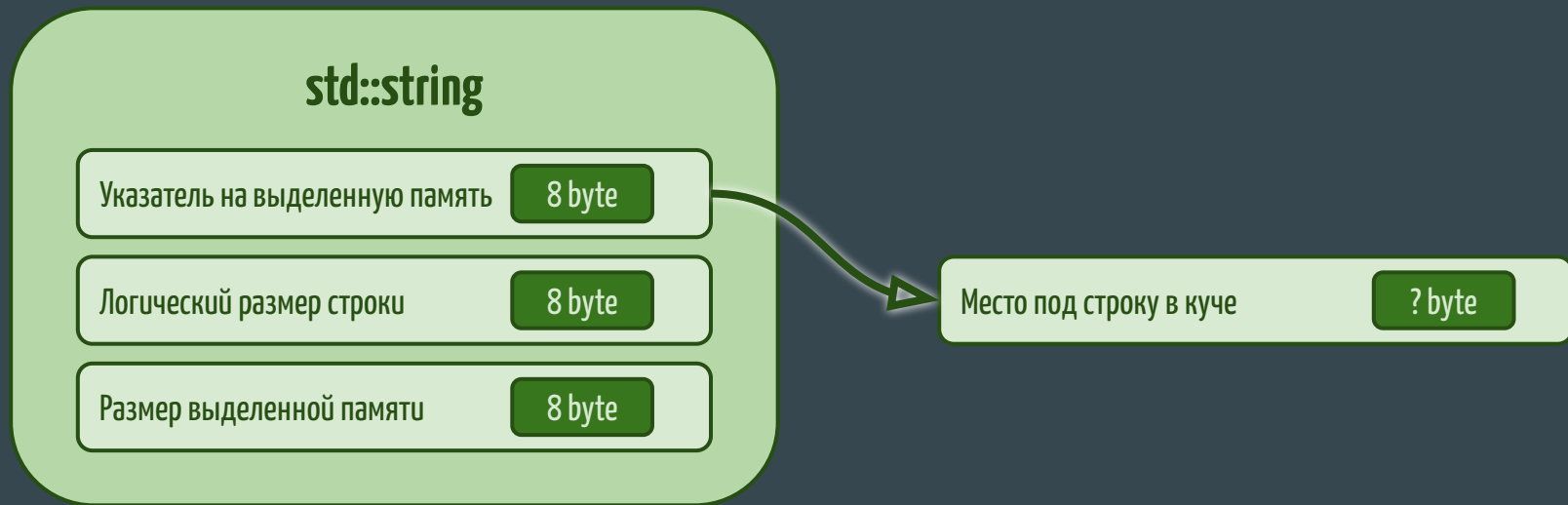
```
#include <cstring>
#include <iostream>

int main()
{
    std::string emptyString;
    std::cout << "emptyString = " << emptyString << "; size = " << sizeof(emptyString) << std::endl;

    std::string filledString{ "Some important information" };
    std::cout << "filledString = " << filledString << "; size = " << sizeof(filledString) << std::endl;
}
```

```
emptyString = ; size = 24
filledString = Some important information; size = 24
```

Внутреннее представление std::string



Обращение к символу по индексу

reference `operator[]`(size_type pos);

reference `at`(size_type pos);

```
#include <string>
#include <iostream>
#include <cassert>

int main() {
    std::string str{"abcdefgh"};

    assert(str[0] == 'a');
    std::cout << "2th symbol of \"" << str << "\"" is " << str[1] << '\n';

    str[2] = 'w';
    char &ch = str[3];
    ch = 'z';

    std::cout << "Result string is \"" << str << "\"" << std::endl;
}
```

```
2th symbol of "abcdefgh" is b
Result string is "abwzefgh"
```


Посимвольный анализ строки

```
#include <string>
#include <map>
#include <iostream>

int main()
{
    std::string famousQuote = "Life did not intend to make us perfect.
    Whoever is perfect belongs in a museum. -- Erich Maria Remarque";

    std::map<char, int> symbCounter;
    for(char& ch : famousQuote) {
        if(std::isalpha(ch)) {
            ++symbCounter[std::tolower(ch)];
        }
    }

    for(auto& symbData : symbCounter) {
        cout << symbData.first << " : " << symbData.second << endl;
    }
}
```

```
a : 5
b : 1
c : 3
d : 3
e : 14
f : 3
g : 1
h : 2
i : 7
k : 1
l : 2
m : 5
n : 5
o : 4
p : 2
q : 1
r : 7
s : 4
t : 5
u : 4
v : 1
w : 1
```

Функции стандартной библиотеки (cctype) для анализа кода символа

```
int isalnum ( int c );
```

```
int isalpha ( int c );
```

```
int isblank ( int c );
```

```
int iscntrl ( int c );
```

```
int isdigit ( int c );
```

```
int isgraph ( int c );
```

```
int islower ( int c );
```

```
int isprint ( int c );
```

```
int ispunct ( int c );
```

```
int isspace ( int c );
```

```
int isupper ( int c );
```

```
int isxdigit ( int c );
```

```
int tolower ( int c );
```

```
int toupper ( int c );
```

Разбор std::string на отдельные слова

```
#include <string>
#include <iostream>
#include <sstream>

int main()
{
    std::string famousQuote = "Life did not intend to make us perfect.
Whoever is perfect belongs in a museum. -- Erich Maria Remarque";
    std::istringstream is{famousQuote, std::istringstream::in };

    std::string longestWord;
    std::string currentWord;
    while(is >> currentWord) {
        std::cout << currentWord << std::endl;

        if(currentWord.size() > longestWord.size()) {
            longestWord = currentWord;
        }
    }

    std::cout << "Longest word is \"<string>\" << longestWord << "\"" which contains ";
    std::cout << longestWord.size() << " letters." << std::endl;
}
```

```
Life
did
not
intend
to
make
us
perfect.
Whoever
is
perfect
belongs
in
a
museum.
--
Erich
Maria
Remarque
Longest word is "perfect."
which contains 8 letters.
```

Алгоритм замены символов

```
#include <string>
#include <iostream>
#include <algorithm>

int main()
{
    std::string famousQuote = "Success is not final; failure is not fatal: It is the courage to continue that counts. -- Winston Churchill";

    std::cout << "Source string = " << famousQuote << std::endl;
    std::replace_if(famousQuote.begin(), famousQuote.end(), [](char& ch){
        return !std::isalpha(ch);
    }, ' ');
    std::cout << "Result string = " << famousQuote << std::endl;
}
```

```
Source string = Success is not final; failure is not fatal: It is the courage to continue that counts. -- Winston Churchill
Result string = Success is not final  failure is not fatal  It is the courage to continue that counts      Winston Churchill
```

Поиск и замена слов в строке

```
#include <string>
#include <iostream>
#include <algorithm>

std::string StringReplacer(const std::string& inputStr, const std::string& src, const std::string& dst)
{
    std::string result{ inputStr };

    size_t pos = result.find(src);
    while(pos != std::string::npos) {
        result.replace(pos, src.size(), dst);
        pos += dst.size();
        pos = result.find(src, pos);
    }

    return result;
}

int main()
{
    std::string str = "<t>Header</t><t>Description section</t>";
    str = StringReplacer(str, "<t>", "<text>");
    str = StringReplacer(str, "</t>", "<end text>");
    std::cout << "Result string: " << str << std::endl;
}
```

Поиск и замена слов в строке

```
std::string str = "<t>Header</t><t>Description section</t>";  
str = StringReplacer(str, "<t>", "<text>");
```

```
std::string StringReplacer(
```

```
    const std::string& inputStr,
```

<	t	>	H	e	a	d	e	r	<	/	t	>	<	t	>	T	e	x	t	<	/	t	>
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

```
    const std::string& src,
```

<	t	>
---	---	---

```
    const std::string& dst)
```

<	t	e	x	t	>
---	---	---	---	---	---

```
std::string StringReplacer(
```

```
    const std::string& inputStr,
```

< t > H e a d e r < / t > < t > T e x t < / t >

```
    const std::string& src,
```

< t >

```
    const std::string& dst)
```

< t e x t >

```
std::string result{ inputStr };
```

< t > H e a d e r < / t > < t > T e x t < / t >

```
size_t pos = result.find(src);
```

 < t > H e a d e r < / t > < t > T e x t < / t >


```
result.replace(pos, src.size(), dst);
```

 < t e x t > H e a d e r < / t > < t > T e x t < / t >

```
pos += dst.size();
```

 < t e x t > H e a d e r < / t > < t > T e x t < / t >

```
pos = result.find(src, pos);
```

< t e x t > H e a d e r < / t >  < t > T e x t < / t >