Курс-интенсив

Программирование на C++

Ввод/Вывод

academy.rubius.com
sergey@prohanov.com
Сергей Проханов

Строки

```
int main()
{
    std::cout << "Hello, world!" << std::endl;
    return 0;
}
```

```
#include <string>
std::string name1;
name1 = "Masha";
std::string name2("Sasha");
```

```
char name3[] = "Vasya";
```

Строки C-style

```
#include <cstring>

strcpy_s()
strcat()
strncat()
strcmp()
strncmp()
```

```
char name[15] = "Artem";
std::cout << "Name is: " << name << '\n';
std::cout << name << " has " << strlen(name) << " letters.\n";
std::cout << name << " has " << sizeof(name) << " characters in the array.\n";
```

```
#include <iostream>
#include <cstring>
int main()
{
    char buffer[255];
    std::cout << "Enter a string: ";
    std::cin.getline(buffer, 255);
    int spacesFound = 0;
       for (int index = 0; index < strlen(buffer); ++index)
    {
              if (buffer[index] == ' ')
           spacesFound++;
    }
    std::cout << "You typed " << spacesFound << " spaces!\n";
    return 0;
}
```

Строки

```
namespace std
{
    using string = basic_string<char, char_traits<char>,
        allocator<char>>

    using wstring = basic_string<wchar_t, char_traits<wchar_t>,
        allocator<wchar_t>>

    using u16string = basic_string<char16_t, char_traits<char16_t>,
         allocator<char16_t>>

    using u32string = basic_string<char32_t, char_traits<char32_t>,
        allocator<char32_t>>
}
```

Строки

```
std::find, std::find_if, std::find_if_not
std::adjacent_find
std::search, std::find_end, std::find_first_of
std::search_n, std::mismatch, ...
```

```
std::string::find, std::string::rfind,
std::string::find_first_of, std::string::find_last_of,
std::string::find_first_not_of, std::string::find_last_not_of
std::string::c_str
```

Строки

```
std::stoi, std::stol, std::stoll //строка в знаковое целое
std::stoul, std::stoull  //строка в беззнаковое целое
std::stof, std::stod, std::stold // строка в вещественное число

std::to_string //число в строку
```

```
std::string one{"1"};
std::string fifteen{"F"};
std::string fourteen{"1110"};

constexpr double real = 12.567890;
std::cout << "One to integer: " << std::stoi(one) << "\n";
std::cout << "Fifteen to integer: " << std::stol(fifteen, nullptr, 16) << "\n";
std::cout << "Fourteen to integer: " << std::stoull(fourteen, nullptr, 2) << "\n";
std::cout << "Real to string: " << std::to_string(real) << "\n";
```

Строки

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

int main()
{
    static const std::vector<std::string> fruits = {
        "apple", "pineapple", "orange", "melon", "banana", "grape"
    };

    std::sort(fruits.begin(), fruits.end());
    std::cout << "Sorted fruits: \n";

    for(const auto& str : fruits) {
        std::cout << str << "\n";
    }

    return 0;
}
```

Строки

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

int main()
{
	std::string str{ "String" };
	std::cout << "Some " + str << "\n";
	return 0;
}
```

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

int main()
{
	std::stringstream stream;
	std::string str{ "String" };
	stream << "Some " << str;
	std::cout << stream.str() << "\n";
	return 0;
}
```

Свойства символов

https://en.cppreference.com/w/cpp/string/char_traits

```
struct CaseInsensitiveTraits final : std::char_traits<char> {
    static bool lt(char lhs, char rhs) {
        return std::tolower(lhs) < std::tolower(rhs);
    }

    static bool eq(char lhs, char rhs) {
        return std::tolower(lhs) == std::tolower(rhs);
    }

     static int compare (const char* lhs, const char* rhs, size_t length);
};
```

std::string_view

```cpp
#include <string_view>

class User
{
public:
    std::string firstName() const;
    const std::string& lastName() const;
    std::string_view email() const;

private:
    std::string firstName_;
    std::string lastName_;
    std::string email_;
};
```

```cpp
void print(const std::string& string)
{
    std::cout << string << "\n";
}

void print(std::string string)
{
    std::cout << std::move(string) << "\n";
}

void print(const char* string)
{
    std::cout << string << "\n";
}
```

```cpp
void print(std::string_view string)
{
    std::cout << string << "\n";
}
```

std::string_view

```cpp
void print(std::string_view string)
{
    std::cout << string << "\n";
}

using namespace std::literals;

print("Plain C string literal");
print("C++ string literal"s);
print("C++ string_view literal"sv);

auto [array, size] = allocate_array();

// неправильно
print(array);

// правильно
print(std::string_view(array, size));
```

```cpp
std::pair<char*, size_t> allocate_array()
{
    constexpr size_t size = 5;
    char *array = new char[size];

    std::memcpy(
        array,
        "A lot of characters and random bytes",
        500
    );

    array[0] = 'H';
    array[1] = 'e';
    array[2] = 'l';
    array[3] = 'l';
    array[4] = 'o';

    return {array, size};
}
```

Регулярные выражения

Регуля́рные выраже́ния (англ. regular expressions, жарг. регэ́кспы или ре́гексы) — система обработки текста, основанная на специальной системе записи образцов для поиска. Образец (англ. pattern), задающий правило поиска, по-русски также иногда называют «шаблоном», «маской».

. ^ $ * + ? { } [ ] \ | ( ) - спец. символы

Регулярные выражения

https://en.cppreference.com/w/cpp/regex

```
#include<iostream>

#include<regex>

int main()
{
    std::string input("Hello, world! world");
    std::regex re("world");
    std::cout << regex_replace(input, re, "planet") << "\n";
    return 0;
}
```

Регулярные выражения

.	Один любой символ, кроме новой строки “\n”.
?	0 или 1 вхождение шаблона слева
+	1 и более вхождений шаблона слева
*	0 и более вхождений шаблона слева
\w	Любая цифра или буква (\W — все, кроме буквы или цифры)
\d	Любая цифра [0-9] (\D — все, кроме цифры)
\s	Любой пробельный символ (\S — любой не пробельный символ)
\b	Граница слова
[..]	Один из символов в скобках ([^..] — любой символ, кроме тех, что в скобках)
\	Экранирование специальных символов (\. означает точку)
^ и $	Начало и конец строки соответственно
{n,m}	От n до m вхождений ({,m} — от 0 до m)
a|b	Соответствует a или b
()	Группирует выражение и возвращает найденный текст
\t, \n, \r	Символы табуляции, новой строки и возврата каретки (соответственно)

Регулярные выражения

```
#include <iostream>
#include <regex>

int main()
{
     std::regex re(R"(l[aio]st)");
     if (std::regex_search("listing", re)) {
          std::cout << "Case 1: matched" << "\n";  // printed
     }
     if (std::regex_search("the last day", re)) {
          std::cout << "Case 2: matched" << "\n";  // printed
     }
     if (std::regex_search("Lost", re)) {
          std::cout << "Case 3: matched" << "\n";  // not printed
     }
     return 0;
}
```

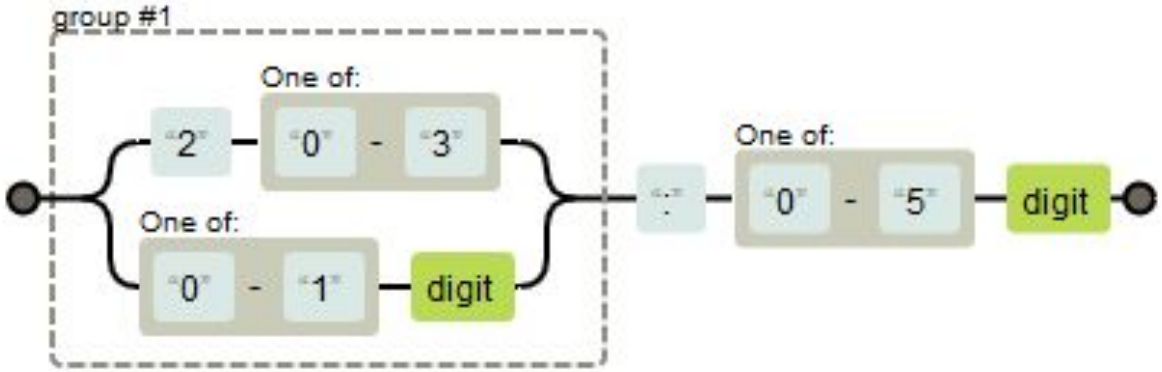
Регулярные выражения

```
#include <iostream>
#include <regex>

int main()
{
	std::string s("31.12.2018");
	std::regex r(R"((?:(\d\d)\.(\d\d)\.)?(\d{4}))");
	std::smatch sm;
	std::regex_match(s, sm, r);
	for (unsigned i = 0; i < sm.size(); ++i) {
		std::cout << sm[i] << std::endl;
	}

	return 0;
}
```

```
// "31.12.2018", "31", "12", "2018"
```

Регулярные выражения

Полезный ресурс
https://regexper.com/



примеры регулярок:

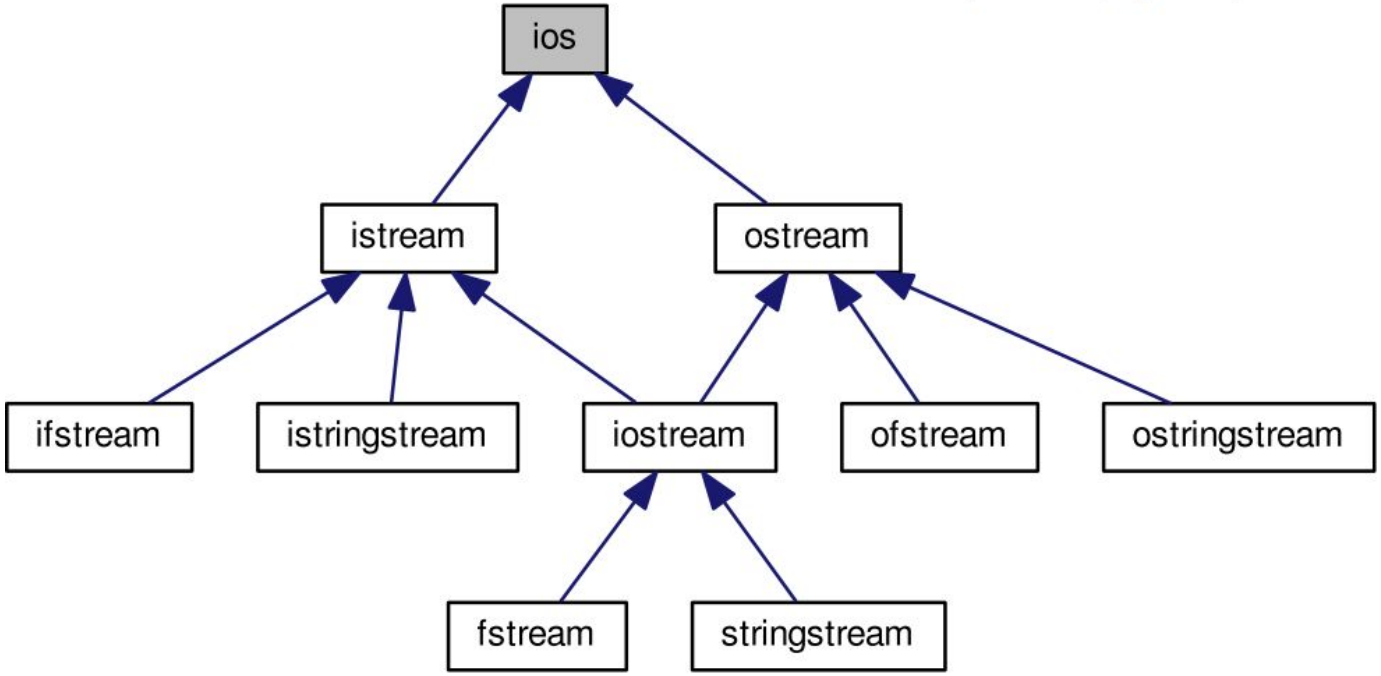
```
(\d{1,2}\/\d{1,2}\/\d{4})

(2[0-3]|[0-1]\d):[0-5]\d

(\W|^)[\w.\-]{0,25}@(yahoo|hotmail|gmail)\.com(\W|$)

\#([a-fA-F]|[0-9]){3, 6}

\< *[img][^\>]*[src] *= *[\"\']{0,1}([^\"\'\ >]*)
```

Потоки ввода/вывода

Универсальные средства ввода/вывода обеспечивают:

- Ввод
- Вывод
- Форматирование
- Работу с файлами
- Буферизацию
- RAII

Потоки ввода/вывода



Потоки ввода/вывода

```
#include <iostream>
```

std::cin - стандартный ввод
std::cout - стандартный вывод
std::cerr - стандартный вывод ошибок
std::clog - для логирования
std::wcin
std::wcout
std::wcerr
std::wclog

std::cout - буферизируемый
std::cerr - не буферизируемый

```
#include <iostream>

void TestWide()
{
	int i = 0;
	std::wcout << L"Enter a number: ";
	std::wcin >> i;
	std::wcerr << L"test for wcerr" << std::endl;
	std::wclog << L"test for wclog" << std::endl;
}

int i = 0;
std::cout << "Enter a number: ";
std::cin >> i;
std::cerr << "test for cerr" << std::endl;
std::clog << "test for clog" << std::endl;
TestWide();
```

Манипуляторы ввода/вывода

endl	Вывод символа новой строки с передачей в поток всех данных из буфера
dec	ввод/вывод в десятичной системы счисления
oct	восьмеричной
hex	шестнадцатеричной
setbase(int base)	Устанавливает базу счисления равной параметру base
flush	Передача в поток содержимого буфера
setfill(int ch)	Устанавливает символ заполнения равным ch
setprecision(int p)	Устанавливает число цифр после запятой
setw(int w)	Устанавливает ширину поля равной w
setiosflags(long f)	Устанавливает флаги, указанные в f
showpos	Показывает знак + для положительных чисел
scientific	Выводит число в экспоненциальной форме

Манипуляторы ввода/вывода

```
#include <iostream>
#include <iomanip>

int main()
{
    std::cout << std::setiosflags(std::ios::showpos);
    std::cout << std::setiosflags(std::ios::scientific);
    std::cout << 123 << " " << 123.23;
    return 0;
}
```

Чтение/запись файлов

- Большинство компьютерных программ работают с файлами, и поэтому возникает необходимость создавать, удалять, записывать читать, открывать файлы.

- Файл – именованный набор байтов, который может быть сохранен на некотором накопителе.

- Для работы с файлами необходимо подключить заголовочный файл <fstream>

- В <fstream> определены несколько классов и подключены заголовочные файлы <ifstream> - файловый ввод и <ofstream> - файловый вывод

Чтение/запись файлов

```
const std::string infile = "input.txt";
std::ifstream in(infile, std::ios::in);

const std::string outfile = "output.txt";
std::ofstream out(outfile, std::ios::out);

if (in.is_open() && out.is_open()) {
     std::string line;

     while (std::getline(in, line)) {
        out << line;
     }

     in.close();
     out.close();
} else {
     std::cerr << "Can’t open " << infile << "\n";
}
```

Режимы открытия файлов

Константа	Описание
ios_base::in	открыть файл для чтения
ios_base::out	открыть файл для записи
ios_base::ate	при открытии переместить указатель в конец файла
ios_base::app	открыть файл для записи в конец файла
ios_base::trunc	удалить содержимое файла, если он существует
ios_base::binary	открытие файла в двоичном режиме

Режимы открытия файлов

```
std::ofstream fout("cpp.txt", ios_base::app);
fout.open("cpp.txt", ios_base::app);
fout.open("cpp.txt", ios_base::out | ios_base::trunc);
```

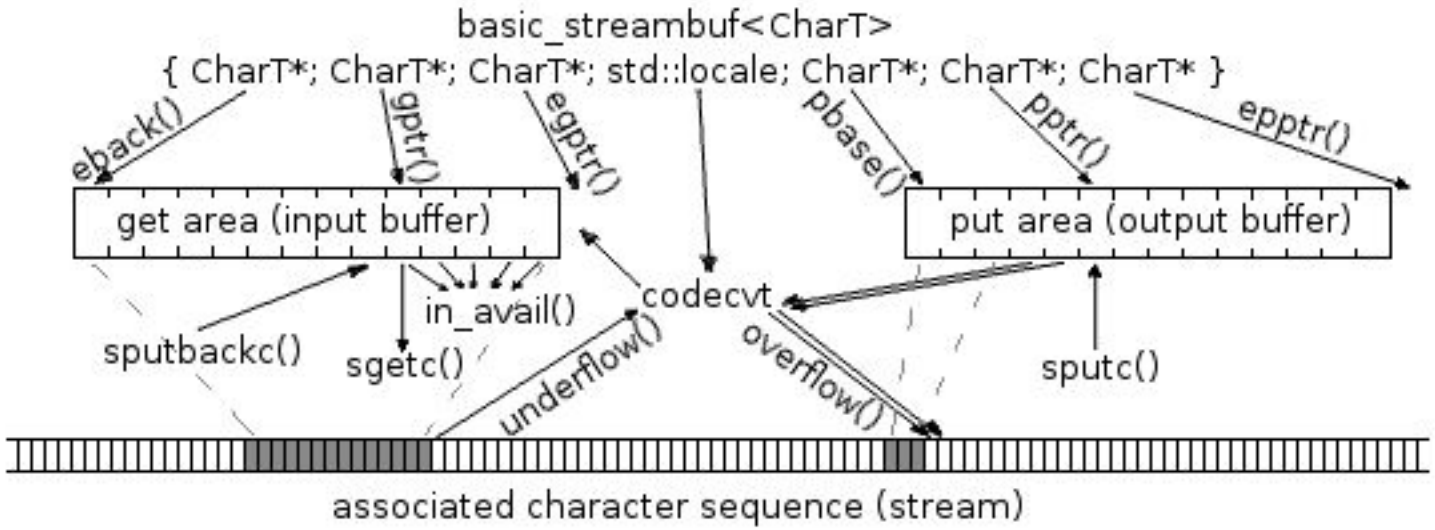
Произвольный доступ к файлам

```
basic_istream& seekg( pos_type pos );
basic_istream& seekg( off_type off, std::ios_base::seekdir dir);


basic_ostream& seekp( pos_type pos );
basic_ostream& seekp( off_type off, std::ios_base::seekdir dir);

pos_type tellg();
pos_type tellp();
```

```
f.seekg(10, ios::cur); // вперёд на 10 байт относительно текущего местоположения файлового указателя
f.seekg(-3, ios::cur); // назад на 3 байта относительно текущего местоположения файлового указателя
f.seekg(42, ios::beg); // 42-ому байту относительно начала файла
f.seekg(0, ios::end); // перемещение в конец файла
```

std::basic_streambuf

```
#include <streambuf>
```



Работа с файловой системой

https://en.cppreference.com/w/cpp/filesystem

```
#include <filesystem>
namespace fs = std::filesystem;

fs::create_directory("sandbox");
fs::create_directories("sandbox/a/b");
fs::create_symlink("a", "sandbox/syma");

fs::copy_file(
   "sandbox/file1.txt", "sandbox/file2.txt"
);
fs::remove_all("sandbox");
```

```
std::cout << fs::path("foo/bar").remove_filename() << "\n"
          << fs::path("foo/").remove_filename() << "\n"
          << fs::path("/foo").remove_filename() << "\n"
          << fs::path("/").remove_filename() << "\n";
```

```
for(auto& p: fs::recursive_directory_iterator("sandbox")) {
    std::cout << p.path() << '\n';
}
```

```
if (fs::is_regular_file(s)) std::cout << " is a regular file";
if (fs::is_directory(s)) std::cout << " is a directory";
if (fs::is_socket(s)) std::cout << " is a named IPC socket";
if (fs::is_symlink(s)) std::cout << " is a symlink";
if (!fs::exists(s)) std::cout << " does not exist";
```

Работа с файловой системой

```
HANDLE hFile = CreateFile(name, GENERIC_READ,
       FILE_SHARE_READ | FILE_SHARE_WRITE, NULL, OPEN_EXISTING,
       FILE_ATTRIBUTE_NORMAL, NULL);
   if (hFile==INVALID_HANDLE_VALUE)
       return -1;

   LARGE_INTEGER size;
   if (!GetFileSizeEx(hFile, &size))
   {
       CloseHandle(hFile);
       return -1;
   }

   CloseHandle(hFile);

auto filesize = size.QuadPart;
```

```
std::ifstream filename("a.out", ios:binary)
auto begin = filename.tellg();
filename.seekg(0, ios::end)
auto end = filename.tellg();

auto filesize = (end - begin);
```

```
auto filesize = size.QuadPart;
```

```
try {
      auto filesize =
std::filesystem::file_size("a.out")
}
catch (fs::filesystem_error& ex) {
      std::cout << ex.what() << "\n"
}
```

Работа с файловой системой

```
void DisplayDirectoryTree(const fs::path& pathToScan, int level = 0) {
   for (const auto& entry : fs::directory_iterator(pathToScan)) {
       const auto filenameStr = entry.path().filename().string();
       if (entry.is_directory()) {
           std::cout << std::setw(level * 3) << "" << filenameStr << '\n';
           DisplayDirectoryTree(entry, level + 1);
       }
       else if (entry.is_regular_file()) {
           std::cout << std::setw(level * 3) << "" << filenameStr
               << ", size " << fs::file_size(entry) << " bytes\n";
       }
       else
           std::cout << std::setw(level * 3) << "" << " [?]" << filenameStr << '\n';
   }
}
```

Курс-интенсив

Программирование на C++

Ввод/Вывод

academy.rubius.com
sergey@prohanov.com
Сергей Проханов