

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РФ  
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ  
БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ  
ВЫСШЕГО ОБРАЗОВАНИЯ  
«ВОРОНЕЖСКИЙ ГОСУДАРСТВЕННЫЙ  
УНИВЕРСИТЕТ»

Д.В. Груздев

**ПРОГРАММИРОВАНИЕ C++ (1 КУРС)**

Учебное пособие

Воронеж  
Издательский дом ВГУ  
2017

# C++

## Часть 1

### Исторические сведения

C++ восходит главным образом к C. C сохранено как подмножество, поэтому сделанного в C акцента на средствах низкого уровня достаточно, чтобы справляться с самыми насущными задачами системного программирования. C, в свою очередь, многим обязано своему предшественнику BCPL; на самом деле, комментарии // (заново) введены в C++ из BCPL.

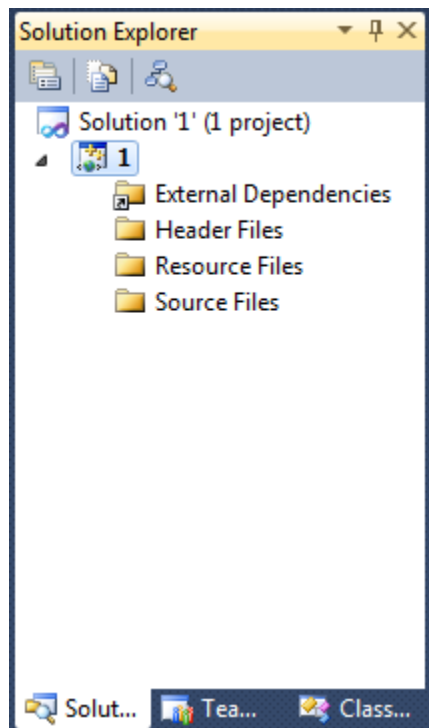
Название C++ - изобретение совсем недавнее (лета 1983его). Более ранние версии языка использовались начиная с 1980ого и были известны как "C с Классами". Первоначально язык был придуман потому, что автор хотел написать модели, управляемые прерываниями, для чего был бы идеален Simula67, если не принимать во внимание эффективность. "C с Классами" использовался для крупных проектов моделирования, в которых строго тестировались возможности написания программ, требующих минимального (только) пространства памяти и времени на выполнение. В "C с Классами" не хватало перегрузки операций, ссылок, виртуальных функций и многих деталей. C++ был впервые введен за пределами исследовательской группы автора в июле 1983его; однако тогда многие особенности C++ были еще не придуманы.

Название C++ выдумал Рик Масситти. Название указывает на эволюционную природу перехода к нему от C. "++" - это операция приращения в C.

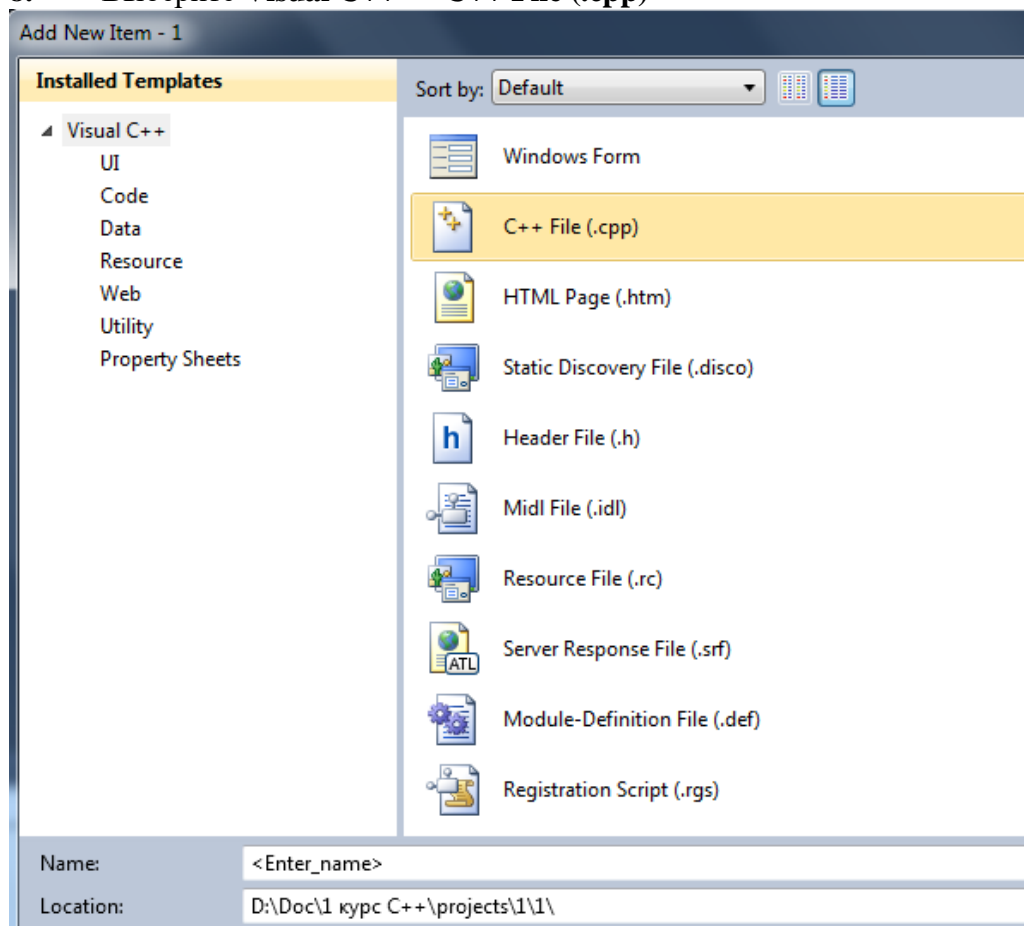
Изначально C++ был разработан, чтобы автору и его друзьям не приходилось программировать на ассемблере, C или других современных языках высокого уровня. Основным его предназначением было сделать написание хороших программ более простым и приятным для отдельного программиста. Плана разработки C++ на бумаге никогда не было; проект, документация и реализация двигались одновременно. Разумеется, внешний интерфейс C++ был написан на C++. Никогда не существовало "Проекта C++" и "Комитета по разработке C++". Поэтому C++ развивался и продолжает развиваться во всех направлениях чтобы справляться со сложностями, с которыми сталкиваются пользователи, а также в процессе дискуссий автора с его друзьями и коллегами.

В качестве базового языка для C++ был выбран C, потому что он (1) многоцелевой, лаконичный и относительно низкого уровня; (2) отвечает большинству задач системного программирования; (3) идет везде и на всем; и (4) пригоден в среде программирования UNIX. В C есть свои сложности, но в наспех спроектированном языке тоже были бы свои, а сложности C нам известны. Самое главное, работа с C позволила "C с Классами" быть полезным (правда, неудобным) инструментом в ходе первых месяцев раздумий о добавлении к C Simula-образных классов.

C++ стал использоваться шире, и по мере того, как возможности, предоставляемые им помимо возможностей C, становились все более существенными, вновь и вновь поднимался вопрос о том, сохранять ли совместимость с C. Ясно, что отказавшись от определенной части наследия C можно было бы избежать ряда проблем. Это не было сделано, потому что (1) есть миллионы строк на C, которые могли бы принести пользу в C++ при условии, что их не нужно было бы полностью переписывать с C на C++; (2) есть сотни тысяч строк библиотечных функций и сервисных программ, написанных на C, которые можно было бы использовать из или на C++ при условии, что C++ полностью совместим с C по загрузке и синтаксически очень похож на C; (3) есть десятки тысяч программистов, которые знают C, и которым, поэтому, нужно только научиться использовать новые особенности C++, а не заново изучать его основы; и (4), поскольку C++ и C будут использоваться на одних и тех же системах одними и теми же людьми, отличия должны быть либо очень большими, либо очень маленькими, чтобы свести к минимуму ошибки и недоразумения. Позднее была проведена проверка определения C++, чтобы удостовериться в том, что любая конструкция, допустимая и в C и в C++, действительно означает в обоих языках одно и то же.



## 8. Выберите **Visual C++ C++ File (.cpp)**



Укажите имя (**Name**) и расположение (**Location**). Нажмите кнопку **Add**.

## 9. Новый проект готов

## Простая программа: печать строки текста

```
#include <iostream>
using namespace std;
int main()
{
    cout << "Добро пожаловать в C++!\n";
    return 0;
}
```

```
#include <iostream>
```

сообщает компилятору, чтобы он включил стандартные возможности потока ввода и вывода, находящиеся в файле `iostream`

```
main() { ... }
```

определяет функцию, названную `main`. Каждая программа должна содержать функцию с именем `main`, и работа программы начинается с выполнения этой функции.

Левая фигурная скобка `{` должна начинать *тело* каждой функции. Соответствующая *правая фигурная скобка* `}` должна заканчивать каждую функцию.

Строка

```
cout << "Добро пожаловать в C++!\n";
```

является командой компьютеру напечатать на экране строку символов, заключенную в кавычки. Полная строка, включающая `cout`, операцию «`<<`», строку `"Добро пожаловать в C++!\n"` и точку с запятой (`;`), называется *оператором*. Каждый оператор должен заканчиваться точкой с запятой (известной также как *признак конца оператора*). Все вводы и выводы в C++ выполняются над *потоками* символов.

Операция «`<<`» называется *операцией поместить в поток*. При выполнении этой программы значение справа от оператора, правый *операнд*, помещается в поток вывода. Символы правого операнда обычно выводятся в точности так, как они выглядят между двойными кавычками. Заметим, однако, что символы `\n` не выводятся на экране. Обратный слэш (`\`) называется *знаком перехода* или *escape-символом* (*эскейп*). Он свидетельствует о том, что должен выводиться «специальный» символ. Когда обратный слэш встречается в цепочке символов, следующий символ комбинируется с обратным слэшем и формирует *управляющую последовательность* (*escape-последовательность*). Управляющая последовательность `\n` означает *новую строку*. Она вызывает перемещение курсора (т.е. индикатора текущей позиции на экране) к началу следующей строки на экране.

Управляющая последовательность	Описание
<code>\n</code>	Новая строка. Позиционирование курсора к началу следующей строки.
<code>\t</code>	Символ горизонтальной табуляции. Перемещение курсора к следующей позиции табуляции.
<code>\r</code>	Возврат каретки. Позиционирование курсора к началу текущей строки; запрет перехода к следующей строке.
<code>\a</code>	Сигнал тревоги. Звук системного звонка.

\\	Обратный слэш. Используется для печати символа обратного слэша.
\"	Двойные кавычки. Используют для печати символа двойных кавычек.

Строка

`return 0;`

показывает, что программа успешно окончена

### простая программа: сложение двух целых чисел

```
// Программа сложения
#include <iostream>
using namespace std;
int main()
{
    int integer1, integer2, sum;    //объявление
    cout << "Введите первое целое число\n";    //приглашение
    cin >> integer1;    //чтение целого
    cout << "Введите второе целое число\n";    //приглашение
    cin >> integer2;    //чтение целого
    sum = integer1 + integer2;    //присваивание значения сумме
    cout << "Сумма равна " << sum << endl;    //печать суммы
    return 0;
}
```

Оператор

`cout << "Введите первое целое число\n";`

печатает на экране буквенное сообщение **Введите первое целое число** и позиционирует курсор на начало следующей строки. Это сообщение называется *приглашением*, потому что оно предлагает пользователю выполнить некоторое действие. О предыдущем операторе можно сказать так: `<<cout` *получает* символьную строку **"Введите первое целое число\n"**.

Оператор

`cin >> integer1;`

использует объект входного потока `cin` и операцию взять из потока », чтобы получить от пользователя значение. Объект **cin** забирает вводимую информацию из стандартного потока ввода, которым обычно является клавиатура. О предыдущем операторе можно сказать так: `<<cin` *дает* значение первого целого числа».

Когда компьютер выполняет предыдущий оператор, он ждет от пользователя ввода значения переменной **integer1**. В ответ пользователь набирает на клавиатуре целое число и затем нажимает *клавишу возврата* — **return** (называемую иногда *клавишей ввода* — **enter**), чтобы послать это число в компьютер. Компьютер затем присваивает это число, или *значение*, переменной **integer1**. Любое последующее обращение в программе к **integer1** будет использовать это значение.

Объекты потоков **cout** и **cin** вызывают взаимодействие между пользователем и компьютером. Поскольку это взаимодействие напоминает диалог, часто говорят о *диалоговом расчете* или *интерактивном расчете*.

Оператор

`cout << "Введите второе целое число\n";`

печатает на экране сообщение **Введите второе целое число** и затем позиционирует курсор на начало следующей строки. Этот оператор приглашает пользователя выполнить действие.

Оператор

`cin >> integer2;`

получает от пользователя значение переменной **integer2**.

Оператор присваивания

```
sum = integer1 + integer2;
```

рассчитывает сумму переменных **integer1** и **integer2** и присваивает результат переменной **sum**, используя *операцию присваивания* =.

Оператор читается так: **sum** получает значение, равное **integer1 + integer2**. Оператор присваивания используется в большинстве расчетов.

Операция = и операция + называются *бинарными операциями*, потому что каждая из них имеет по два *операнда*. В случае операции + этими операндами являются **integer1** и **integer2**. В случае операции = двумя операндами являются sum и значение выражения **integer1 + integer2**.

Оператор

```
cout << "Сумма равна " << sum << endl;
```

печатает символьную строку Сумма равна, затем численное значение переменной sum, за которым следует endl (аббревиатура словосочетания \*end line\* — конец строки) — так называемый *манипулятор потока*. Манипулятор endl выводит символ новой строки и затем «очищает буфер вывода». Это просто означает, что в некоторых системах, где выводы накапливаются в вычислительной машине до тех пор, пока их не станет достаточно, чтобы «имело смысл печатать на экране», endl вызывает немедленную печать на экране всего накопленного.

### Арифметика

Большинство программ выполняет арифметические вычисления. Множество *арифметических операций* показано на рис.. Отметим использование в них разнообразных специальных символов, не используемых в алгебре. Звездочка (\*) обозначает умножение, а знак процента (%) — это операция *вычисления остатка*, которая вкратце будет еще обсуждаться. Арифметические операции на рис. являются бинарными операциями. Например, выражение **integer1 + integer2** содержит бинарную операцию + и два операнда **integer1** и **integer2**.

Операция C++	Арифметическая операция	Алгебраическое выражение	Выражение на C++
Сложение	+	$f + 7$	$f + 7$
Вычитание	-	$P - c$	$P - c$
Умножение	*	$bm$	$b * m$
Деление	/	$x/y$	$x/y$
Вычисление	%	$r \bmod s$	$r \% s$

Целочисленное деление дает целый результат; например, выражение  $7 / 4$  равно 1, а выражение  $17 / 5$  равно 3. Заметим, что любая десятичная часть при целочисленном делении просто отбрасывается (т.е. усекается) — округление не производится. В C++ имеется операция вычисления остатка %, которая дает в качестве результата остаток от целочисленного деления. Выражение  $x \% y$  дает остаток от деления x на y. Таким образом,  $7 \% 4$  равно 3,  $17 \% 5$  равно 2.

### “Лечение” русского языка

```
#include <iostream>
#include "Windows.h"
using namespace std;
```

```
int main()
{
    int integer1, integer2, sum;        //объявление
    SetConsoleCP(1251);
    SetConsoleOutputCP(1251);
    cout << "Введите первое целое число\n";    //приглашение
    cin >> integer1;        //чтение целого
    cout << "Введите второе целое число\n";    //приглашение
    cin >> integer2;        //чтение целого
    sum = integer1 + integer2;        //присваивание значения сумме
    cout << "Сумма равна " << sum << endl;    //печать суммы
    return 0;
}
```

Или

```
#include <iostream>
using namespace std;
int main()
{
    int integer1, integer2, sum;        //объявление
    setlocale(LC_ALL, "rus_rus.1251");
    cout << "Введите первое целое число\n";    //приглашение
    cin >> integer1;        //чтение целого
    cout << "Введите второе целое число\n";    //приглашение
    cin >> integer2;        //чтение целого
    sum = integer1 + integer2;        //присваивание значения сумме
    cout << "Сумма равна " << sum << endl;    //печать суммы
    return 0;
}
```

## Принятие решений: операции проверки на равенство и отношения

Этот раздел познакомит вас с простой версией *структуры if* в C++, которая позволяет программе принимать решение, основываясь на истинности или ложности некоторого *условия*.

Если условие удовлетворено, т.е. условие есть **true** (истина), то оператор в теле структуры **if** выполняется. Если условие не удовлетворяется, т.е. условие есть **false** (ложь), то оператор в теле не выполняется.

Условия в структурах **if** могут быть сформированы с использованием *операций проверки на равенство и отношения*, сводка которых приведена на рис.

Обычная алгебраическая операция проверки на равенство или отношения	Операция C++ проверки на равенство или отношения	Пример условия на C++	Значение условия C++
<b>Операции проверки на равенство</b>			
=	==	x == y	x равен y
*	!=	x != y	x не равен y
<b>Операции отношения</b>			
>	>	x > y	x больше y
<	<	x < y	x меньше y
>	>=	x >= y	x больше или
<	<=	x <= y	x меньше или