

# Научно- исследовательская практика

---

ВВЕДЕНИЕ В C++

# Что мы будем изучать


---

Основная цель практики – получение опыта разработки ПО на языке C++

В дальнейшем – эти знания понадобятся в дисциплинах «Системы технического зрения» и «Проектирование ПРТС»



# Откуда брать материалы

[https://gitlab.com/bmstu\\_underwater\\_robotics](https://gitlab.com/bmstu_underwater_robotics)


 **bmstu\_underwater\_robotics** This group Search + 📄 🔗 🔒 🌐

[Group](#) [Issues 0](#) [Merge Requests 0](#) [Members](#) [Contribution Analytics](#) [Settings](#)

[Home](#) [Activity](#)


  
**bmstu\_underwater\_robotics**   
Group for BMSTU Underwater Robotics students.  
Группа для учебных проектов студентов кафедры "Подводные роботы и аппараты" МГТУ им. Баумана.  
[Leave group](#) [🔔 Global](#)

[Projects](#) [Subgroups](#)  Last updated New Project

 **prts**


Материалы по курсу "Проектирование подводных робототехнических систем"

updated 5 days ago

 **ni\_prakt**

Материалы по дисциплине "Научно-исследовательская практика"

updated 5 days ago

 **rpk\_prts\_2017**

Проекты студентов по курсу "Разработка программных комплексов ПРТС" за 2017 год

updated 4 months ago

# Почему C++, а не Asm, Fortran и т.д.?

---

Ассемблер : быстрый код, высокая сложность

Pascal, Fortran, Lisp и т.д. – большое количество накопленных решений, «высокоуровневые» языки, усложнён доступ к машинным ресурсам

# Почему C++, а не Java, Ruby и т.д.?

---

Современные объектные языки, сильно упрощающие работу программиста

Большие сложности с реальным временем

# Почему C++, а не C?

---

C – проще

Но куча библиотек, фреймворков и т.д. использует объектно-ориентированный подход

При это объектно-ориентированный подход упрощает разработку (что спорно)

# Несколько общих слов о C/C++

---

C – компилируемый процедурный язык

C++ - компилируемый мультипарадигмальный язык: объектно ориентированный, обобщённый процедурный...

C не является подмножеством C++!!!

- Разные определения спецификатора `inline`, разные типы `complex`, `bool` и т.д.

# Несколько общих слов о C/C++

---

Несколько версий стандарта языка:

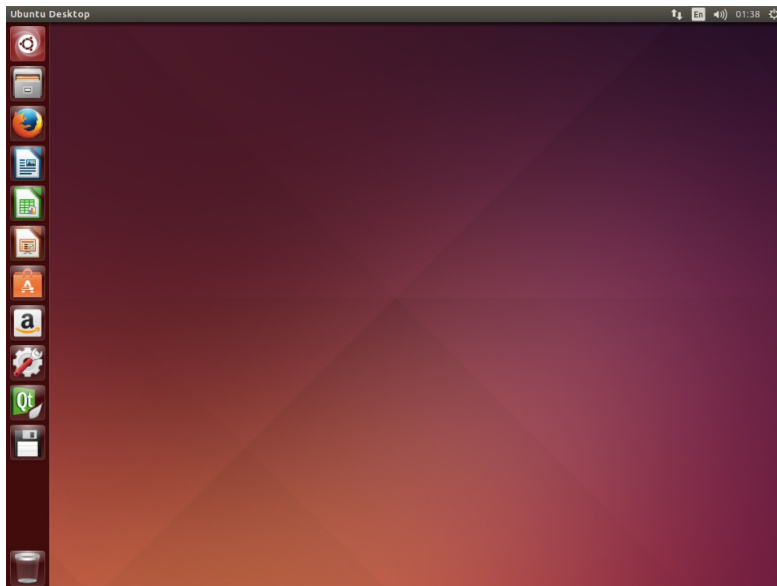
- ...
- C++98 – в старых учебниках (не надо их читать!)
- C++03 – наш «основной» клиент
- C++11 – много изменений, с частью мы познакомимся
- C++14 – небольшие изменения относительно предыдущего
- C++17 – недавно принят и мало поддерживается
- C++21 – следующий стандарт



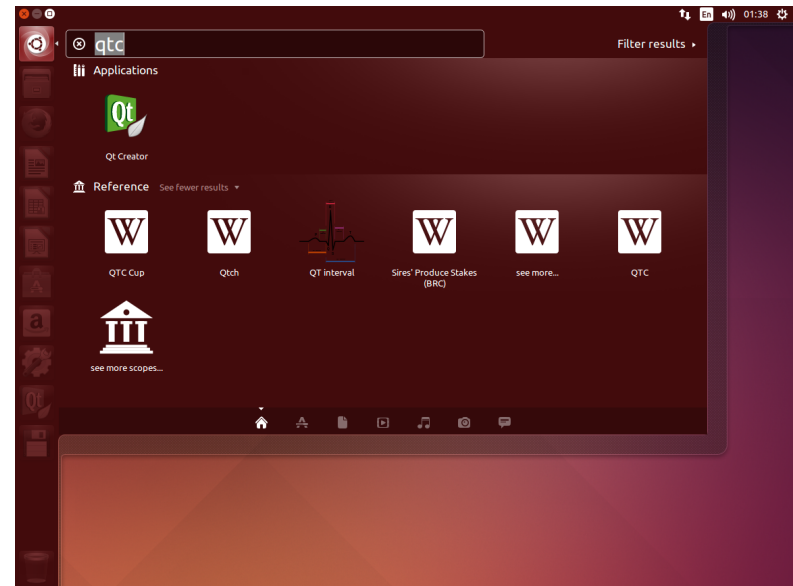
# Ubuntu/ Unity

---

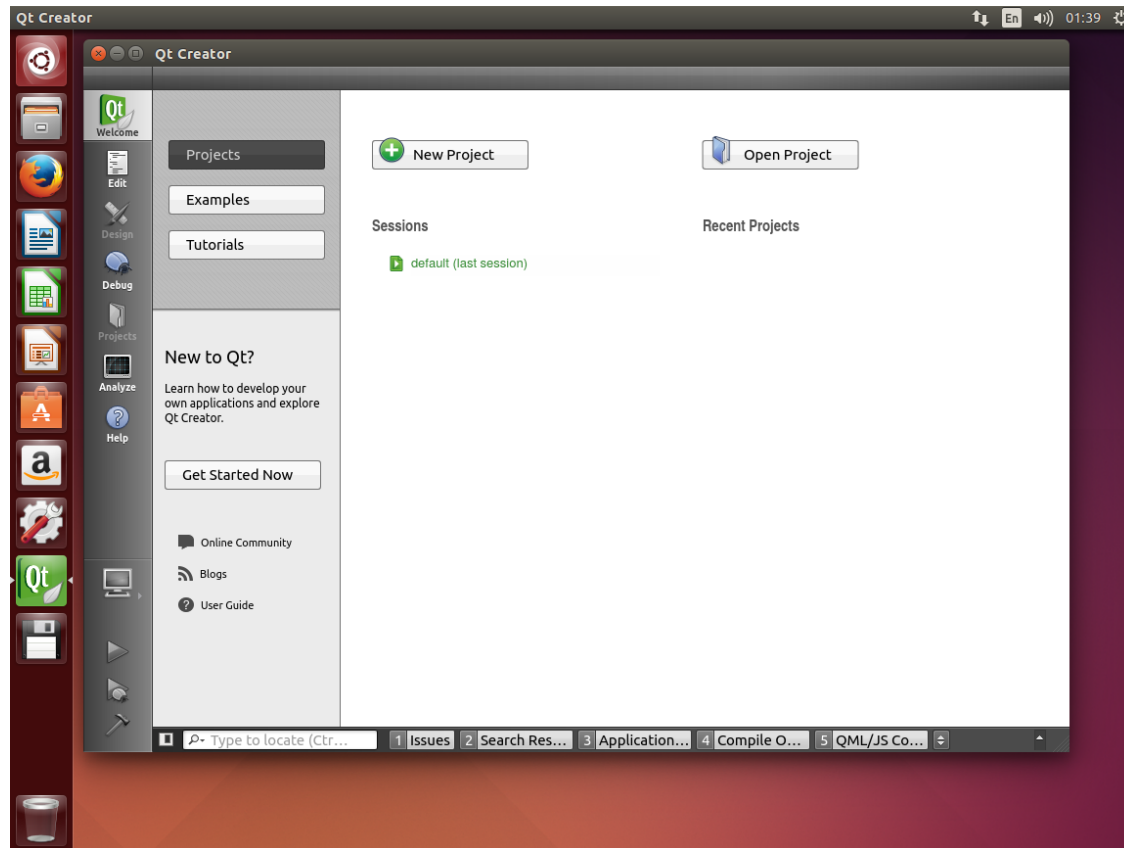
РАБОЧИЙ СТОЛ



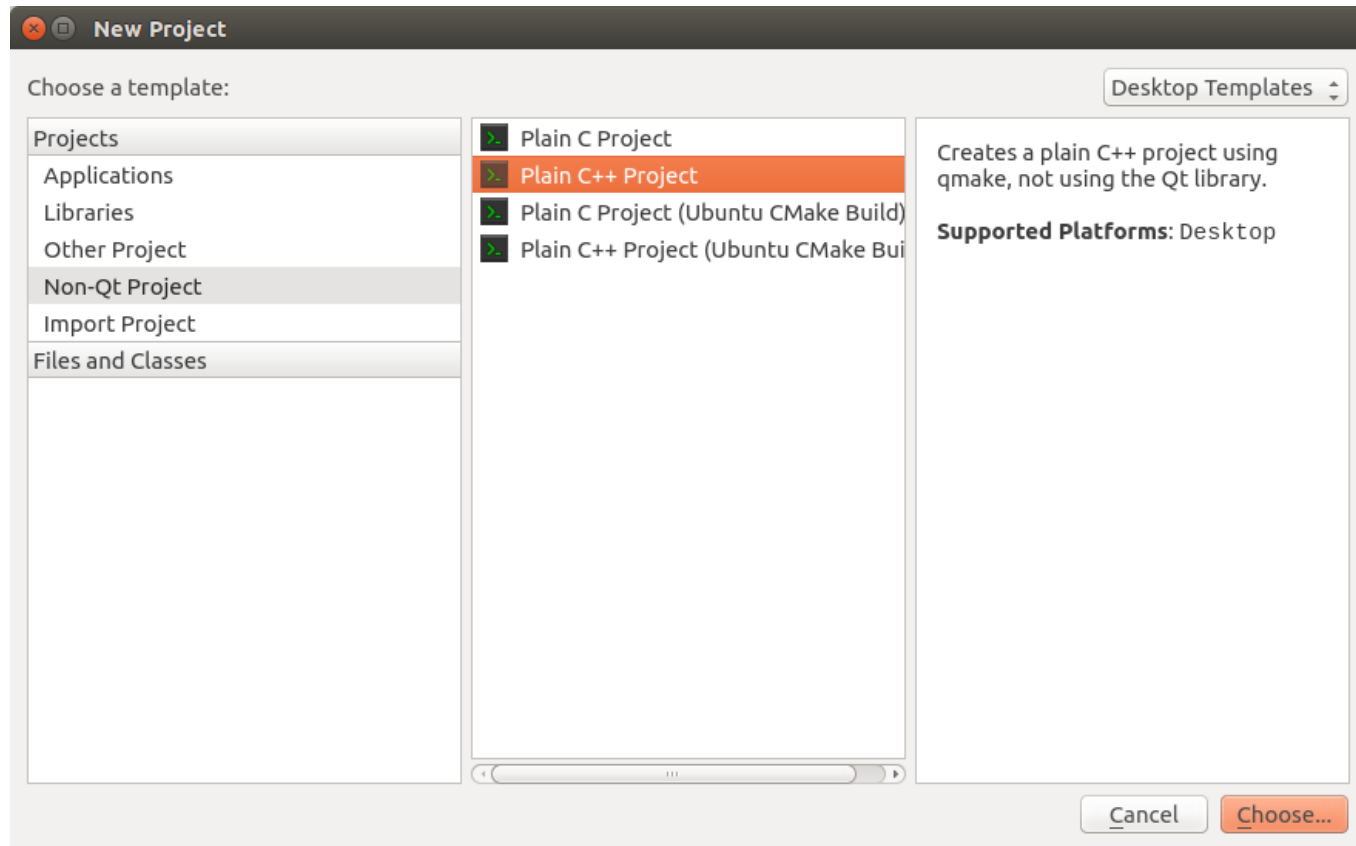
ДОК



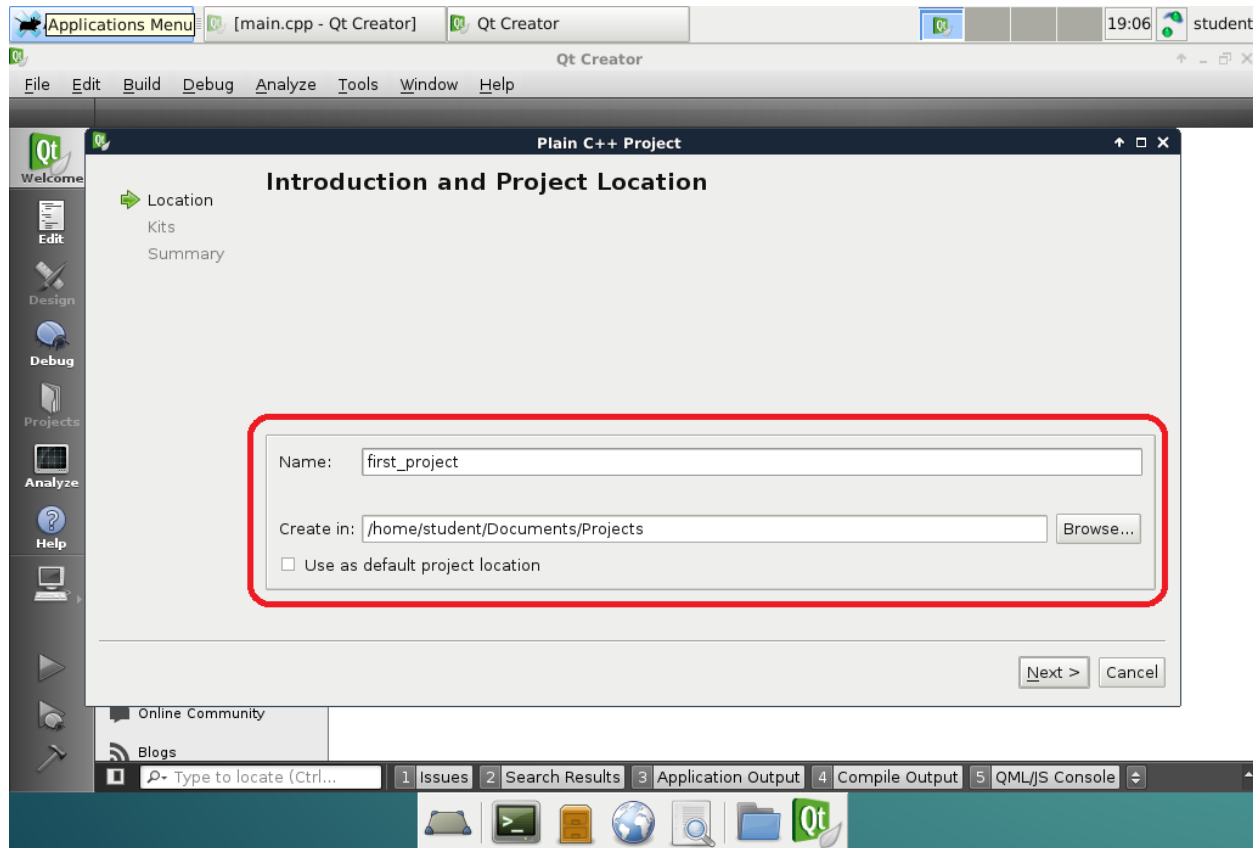
# QtCreator



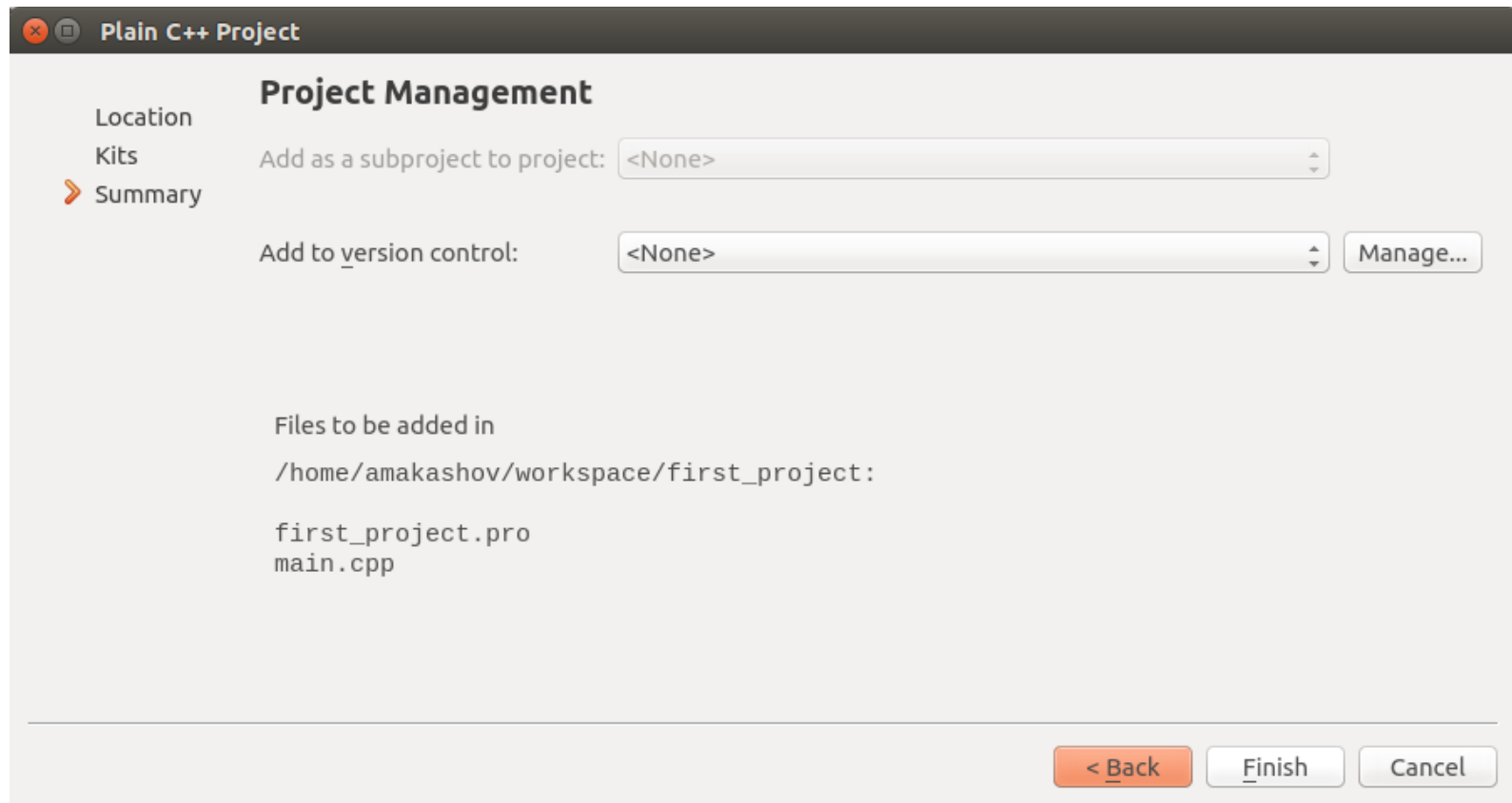
# Проект



# Имя и расположение

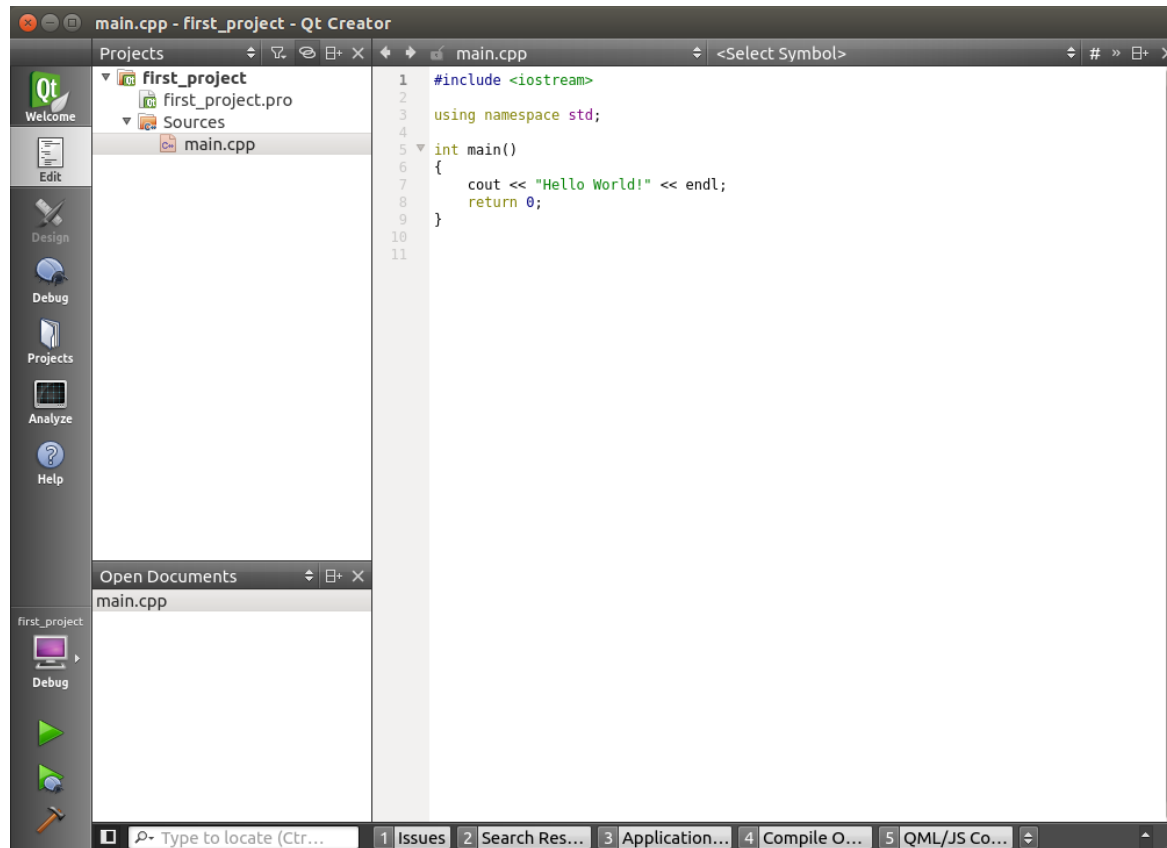


# Контроль версий (который мы пока не используем)



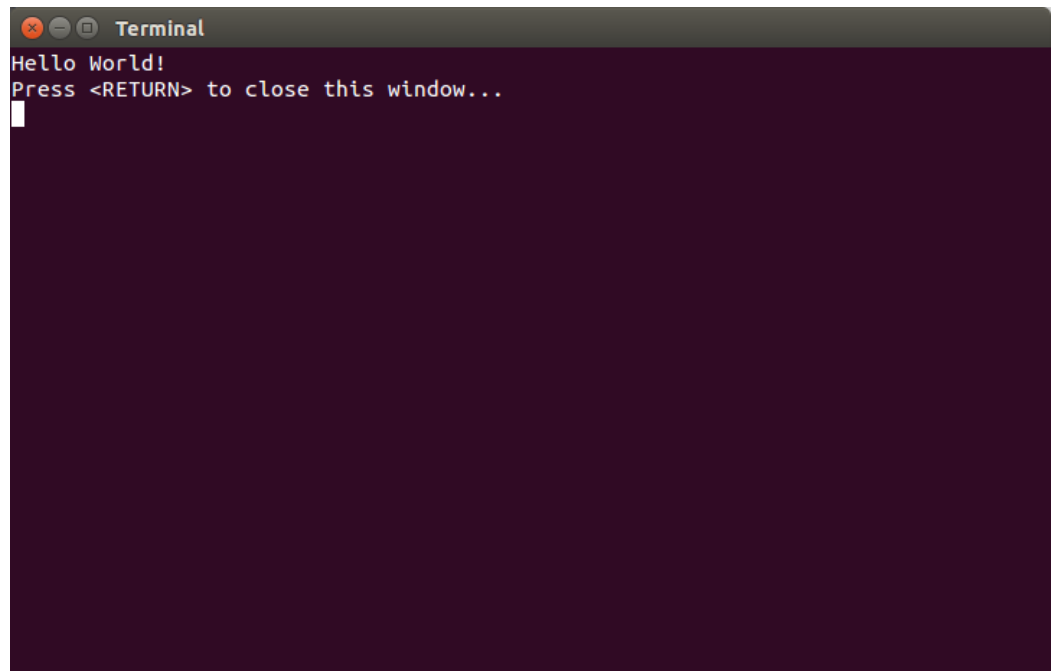
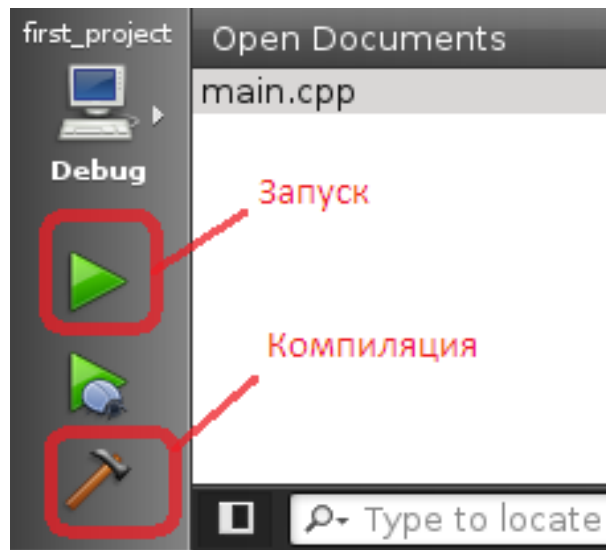
# Hello world

---



# Компиляция и вывод

---



# Hello world

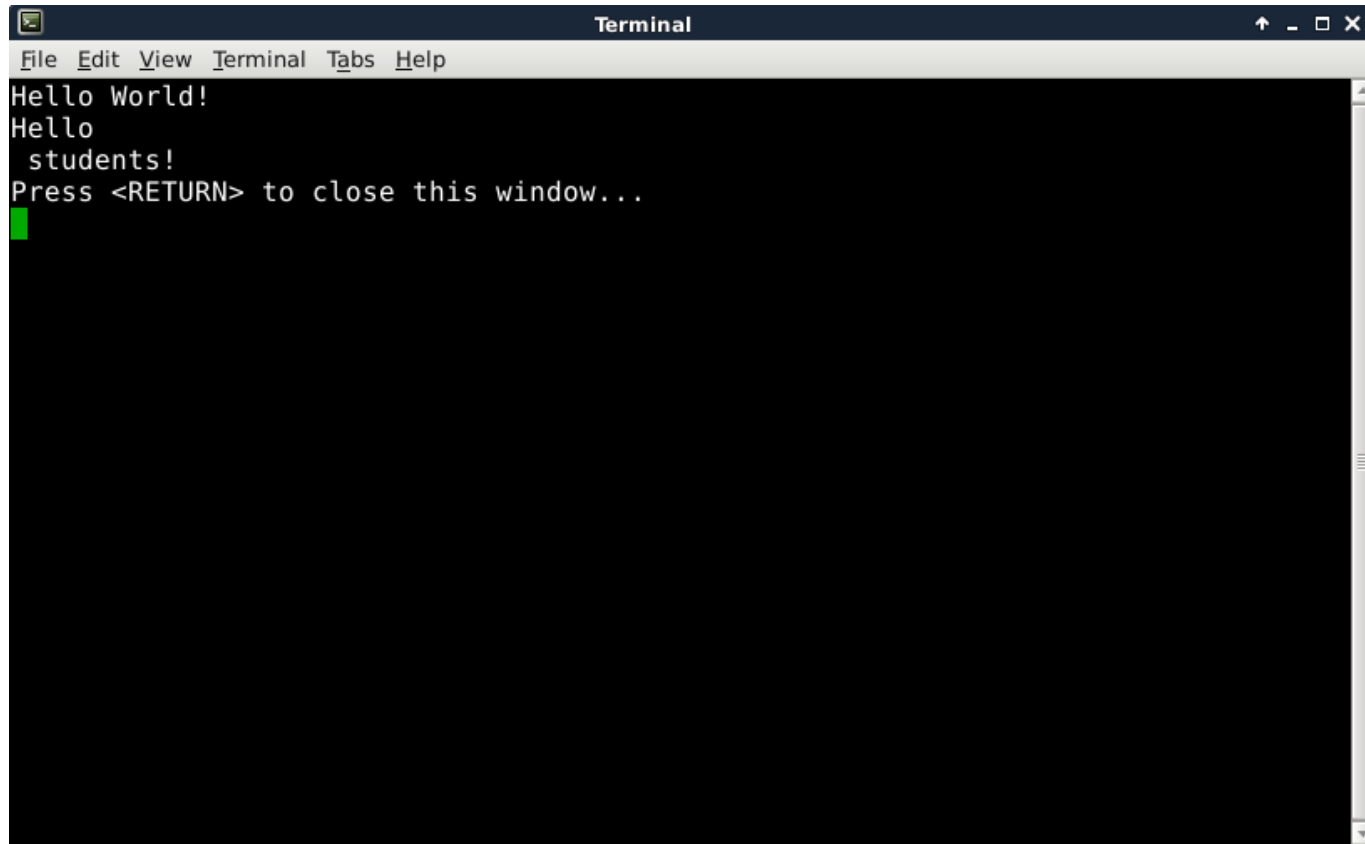
---

```
1 // - с такого символа начинается комментарий
2 #include <iostream> // Заголовочный файл для потокового ввода-вывода на экран
3
4 using namespace std; // Включение использования по умолчанию пространства имён std
5
6 int main()
7 {
8     cout << "Hello World!" << endl; // Собственно, функция потокового вывода
9     // cout - вывод в консоль
10    // endl - завершение строки
11    // Аргументы разделяются оператором <<
12
13    /* А вот с такого символа начинается комментарий произвольной длины
14     cout << "This string won't be printed anyway" << endl;
15     который заканчивается аналогичным закрывающим символом */
16
17    std::cout << "Hello\n students!" /*Просто комментарий внутри строки */<< std::endl;
18    /* вот так пришлось бы писать без подключения пространства std
19     При этом внутри строки у нас тоже есть символ перехода на новую строку \n
20     */
21    return 0; // Функция main должна вернуть целочисленное значение
22 }
23
24
```



# Hello world

---



```
Terminal
File Edit View Terminal Tabs Help
Hello World!
Hello
  students!
Press <RETURN> to close this window...
█
```

A screenshot of a terminal window titled "Terminal". The window has a menu bar with "File", "Edit", "View", "Terminal", "Tabs", and "Help". The terminal output shows "Hello World!" on the first line, "Hello" on the second line, " students!" on the third line (with two spaces), and "Press <RETURN> to close this window..." on the fourth line. A green cursor is visible on the line "Press <RETURN> to close this window...".

# Типы

---

## Целочисленные типы данных

- short (int) [-32768..32767] – 2 байта
- int [-2147483648.. 2147483647] – 4 байта
- long long [ $\pm 9.22 \cdot 10^{18}$ ] – 8 байт (C++11)

## Беззнаковые типы - unsigned

- unsigned short [0..65535]
- unsigned int [0..4294967295]
- unsigned long long [0..  $1.9 \cdot 10^{19}$ ] (C++11)

# Типы

---

Неприятное исключение:

- long (long int)
  - В Win32, Win64 и Linux x86 4-байтный тип (int)
  - В Linux x64 – 8-байтный тип (long long)

# Типы

---

## Дробные числа – с плавающей запятой

- float (одинарная точность) – 4 байта
  - 7 знаков мантиссы
  - Порядок  $\sim 10^{38}$
  - Диапазон  $[\pm 1.17 \cdot 10^{-38} \dots \pm 3.402 \cdot 10^{38}]$
- double (двойная точность) – 8 байт
  - 15 знаков мантиссы
  - Порядок  $\sim 10^{308}$
  - Диапазон  $[\pm 2.22 \cdot 10^{-308} \dots \pm 1.797 \cdot 10^{308}]$

# Типы

---

## «Символьный» тип char

- 1 байт
- signed [-128...127]
- unsigned [0...255]
- В консоль выводится ASCII-символ

## «Бинарный» bool

- Тот же char
- Значения false (0) и true (не 0)

# Типы

В этой длинной простыне мы будем пытаться вывести на экран различные типы данных, а заодно посмотрим (немножко)

```
#include <iostream>
#include <limits>    // Нужен для "размеров" типов

using namespace std;

int main()
{
    int a = 2147483647; // 4 байта
    int b = a+124;      // Вот тут у нас должно получиться переполнение
    short s = 32767;    // 2-байтный тип
    unsigned short us = s+32; // Переполнения не произойдёт
                                // потому, что беззнаковый тип в диапазоне
                                // 0..65535

    float f = 3.1415926535897932384626433832795; // Одинарная точность - 4 байта
    double d = 3.1415926535897932384626433832795; // Двойная точность - 8 байт
    cout << "a=" << a << " and b=" << b << endl;
    cout << "Signed short s=" << s << " and unsigned short us=" << us << endl;
    cout << "Single precision\t" << f << endl;
    cout << "Double precision\t" << d << endl;
    // Теперь опробуем распечатать, установив максимально возможное для каждого типа
    // количество символов в десятичном представлении
    cout.precision(std::numeric_limits<float>::digits);
    cout << "Single precision\t" << fixed << f << endl;
    cout.precision(std::numeric_limits<double>::digits);
    cout << "Double precision\t" << fixed << d << endl;
    cout << "Scientific precision\t" << scientific << d << endl;
    s = a;
    cout << "S = a " << s << " and b " << a << endl;
    cout << "S = f " << (s=f) << " and f " << f << endl;
    cout.precision(std::numeric_limits<double>::digits10);
    cout << "Limits for floats:" << endl;
    cout << "Min=" << std::numeric_limits<float>::min() << endl;
    cout << "Max=" << std::numeric_limits<float>::max() << endl;
    cout << "Epsilon " << numeric_limits<float>::epsilon() << endl;
    cout << "Limits for double, min max eps:" << endl;
    cout << "Min=" << std::numeric_limits<double>::min() << endl;
    cout << "Max=" << std::numeric_limits<double>::max() << endl;
    cout << "Epsilon " << numeric_limits<double>::epsilon() << endl;

    char c = 52; // 1 байт
    cout << "Character " << c << " has ASCII-code " << static_cast<int>(c) << endl;
    bool b1 = true, b2 = false;
    cout << std::boolalpha;
    cout << b1 << "=" << static_cast<int>(b1) << endl;
    cout << b2 << "=" << static_cast<int>(b2) << endl;

    return 0;
}
```

# Типы

---

```
a=2147483647 and b=-2147483525
Signed short s=32767 and unsigned short us=32799
Single precision      3.14159
Double precision      3.14159
Single precision      3.141592741012573242187500
Double precision      3.14159265358979311599796346854418516159057617187500000
Scientific precision  3.14159265358979311599796346854418516159057617187500000e
+00
S = a -1 and b 2147483647
S = f 3 and f 3.14159274101257324218750000000000000000000000000000000e+00
Limits for floats:
Min=1.175494350822288e-38
Max=3.402823466385289e+38
Epsilon 1.192092895507812e-07
Limits for double, min max eps:
Min=2.225073858507201e-308
Max=1.797693134862316e+308
Epsilon 2.220446049250313e-16
Character 4 has ASCII-code 52
true=1
false=0
Press <RETURN> to close this window...
```

# Операторы

---

В C/C++ используются обычные математические бинарные операторы:

+ -, \*, /

= - оператор присваивания

== - проверка равенства

(a==5) true если a равно 5

(a<=4) true если a меньше или равно 4

(a>=0) true если a больше или равно 0

Аналогично > и <



# Операторы

---

Унарные операторы:

- Оператор пост-инкремента `a++`
- Оператор пре-инкремента `++a`
- Оператор пост-декремента `a--`
- Оператор пре-декремента `--a`
- Оператор `+=` `a+=2` увеличивает `a` на 2
- ...

# Цикл For

---

Цикл со счётчиком

for (*начальное условие; условие прекращения; условие перехода*)

Например

```
for (i = 0; i<5; i++)
```

```
{
```

```
    ...      //      тело цикла
```

```
}
```

1. Начинаем с  $i=0$
2. Продолжаем, пока  $i<5$
3. Каждый раз увеличиваем  $i$  на 1

# Цикл For

---

Но тут могут быть более сложные конструкции

```
for (int i = 0; i<5; )  
{  
    ...      //      тело цикла  
    i +=2;  
}
```

# Области видимости

---

```
#include <iostream>

using namespace std;

int a = 5;
int b = 7;

int main()
{
    float b = 3.5;
    int c = a+2;    // Нет ошибки - перекрыли глобальную переменную
    cout << "First we'll print a=" << a << " and c=" << c << endl;
    cout << "Next let's print b=" << b << endl;

    a = 5;
    cout << "Pre-incremented a=" << ++a << endl;
    cout << "Now a=" << a << endl;
    a = 5;
    cout << "Post-incremented a=" << a++ << endl;
    cout << "Now a=" << a << endl;

    for (int i=0; i<10; i++)
    {
        cout << i << " a+i=" << a+i << endl;
        int c = i;    // Нет ошибки - переменная внутри блока перекрывает объявление в функции
        cout << "C variable in for cycle " << c << endl;
        int c = i+1;    // Ошибка - повторное определение
    }
}
```

```
First we'll print a=5 and c=7
Next let's print b=3.5
Pre-incremented a=6
Now a=6
Post-incremented a=5
Now a=6
0 a+i=6
C variable in for cycle 0
1 a+i=7
C variable in for cycle 1
2 a+i=8
C variable in for cycle 2
3 a+i=9
C variable in for cycle 3
4 a+i=10
C variable in for cycle 4
5 a+i=11
C variable in for cycle 5
6 a+i=12
C variable in for cycle 6
7 a+i=13
C variable in for cycle 7
8 a+i=14
C variable in for cycle 8
9 a+i=15
C variable in for cycle 9
Press <RETURN> to close this window...
```

# Области ВИДИМОСТИ

---

# Задача

---

Числа Фибоначчи

- 1, 1, 2, 3, 5, ...
- $a_i = a_{i-1} + a_{i-2}$
- $a_0 = a_1 = 1$

Напишите программу для вывода первых 10