

Lecture 1. Information about course

Plan

- Introduction to C++. Data types and Objects
- Working with Git
- Control flow statements
- Expressions and Operators
- Functions and procedures
- C++ Memory model
- Classes
- Strings
- Standard library (STL)
- Operator overloading
- Templates
- Exceptions and Move semantics
- Inheritance
- Object-Oriented Analysis and Design
- Qt library
- C++ plus Python

HW, Midterm, Project

- HW: contests
- Midterm: app. at the end of March
- Project: command project with Qt, Git and project board. You will have a project defense at the seminar.

Marks & points

- $OA = 10 \cdot (RP+BP)/Rp_{max}$
- RP = regular points (contests, midterm, projects)
- BP = bonus points (work on seminars, task with * and so on, no more than 20%)
- Total = $0.7OA + 0.3Exam$

Our materials

- Our materials will be on Git <https://github.com/mgordenko/Course-C->

Where to write code

- I recommend VS Code for C++



Visual Studio Code

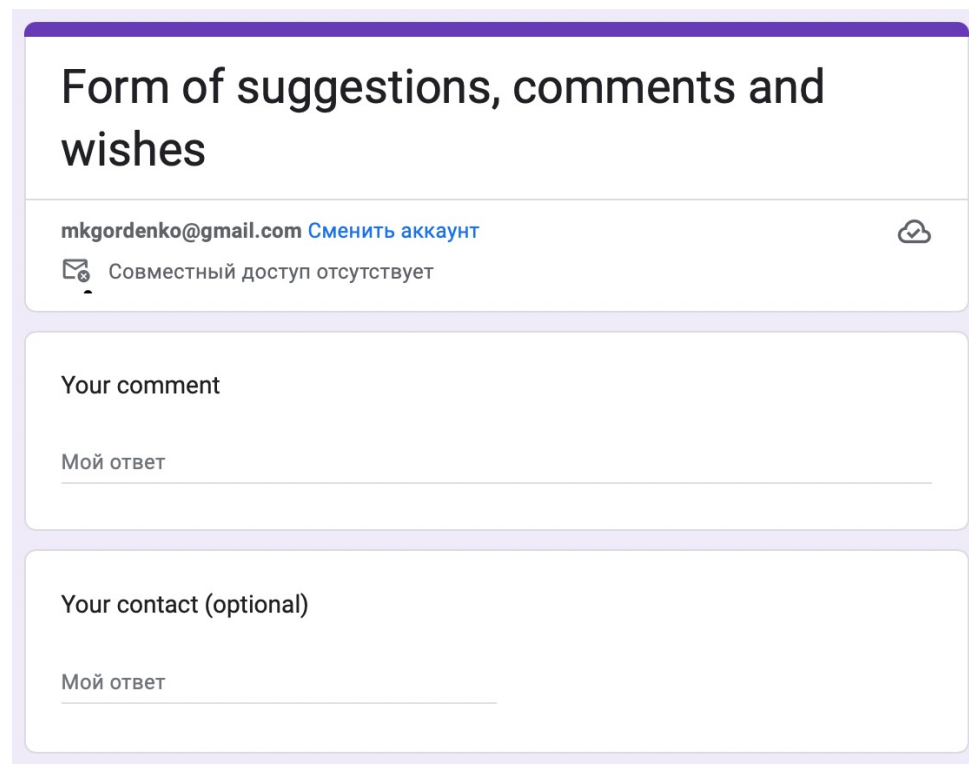
- Instructions <https://code.visualstudio.com/docs/languages/cpp>

What can you use as additional materials?

- [Optional] Stepik Course (on Russian)
<https://stepik.org/course/180462/syllabus>
- Davis, S. R. (2014). C++ For Dummies (Vol. 7th ed). Hoboken: For Dummies. Retrieved from <http://search.ebscohost.com/login.aspx?direct=true&site=eds-live&db=edsebk&AN=784132>
- Gregoire, M. (2018). Professional C++ (Vol. Fourth edition). Indianapolis, IN: Wrox. Retrieved from <http://search.ebscohost.com/login.aspx?direct=true&site=eds-live&db=edsebk&AN=1729638>
- Pilgrim, M. (2009). Dive Into Python 3. New York: Apress. Retrieved from <http://search.ebscohost.com/login.aspx?direct=true&site=eds-live&db=edsebk&AN=326208>


Forms for any comments


- https://docs.google.com/forms/d/e/1FAIpQLSeEKPcGePHom74giTpT9HnOe85vL5WrICnJYumQfL0qeAo46w/viewform?usp=sf_link



The image shows a Google Forms interface with a purple header bar. The form title is "Form of suggestions, comments and wishes". Below the title, the user's email "mkgordenko@gmail.com" is displayed with a link to "Сменить аккаунт" (Change account) and a checkmark icon. Below this, there is a section for "Your comment" with a text input field labeled "Мой ответ" (My answer). At the bottom, there is a section for "Your contact (optional)" with another text input field labeled "Мой ответ" (My answer).

Form of suggestions, comments and wishes

mkgordenko@gmail.com [Сменить аккаунт](#) 

 Совместный доступ отсутствует

Your comment

Мой ответ

Your contact (optional)

Мой ответ

A black and white photograph of Richard P. Feynman in a classroom or lecture hall. He is wearing a light-colored shirt and is captured in the middle of writing on a large chalkboard. His right arm is raised, holding a piece of chalk, and his left hand is also on the board. The chalkboard is filled with complex mathematical equations and diagrams, including integrals and Feynman diagrams. The lighting is dramatic, with strong highlights on Feynman's shirt and the chalkboard, and deep shadows elsewhere.

***“What I cannot create,
I do not understand”***

***Richard P.
Feynman***

“The only way to learn a new programming language is by writing programs in it”

Dennis Ritchie

Creator of the C programming language

Lecture 1. Introduction to C++.

C++ program structure.

Statements. Programs and
modules. Toolchains and
building C++ Programs.

Classification of programming languages

Programming languages

```
graph TD; A[Programming languages] --> B[By level of abstraction:]; A --> C[According to the method of converting the source code:]; A --> D[According to the implemented paradigm:];
```

By level of abstraction:

- Low level
- High level

According to the method of converting the source code:

- Compiled
- Interpretable
- Compile to intermediate code

According to the implemented paradigm:

- Procedural
- Object-oriented
- Functional
- Multi-paradigm











About C++

- C++ is a high-level programming language that was developed by Bjarne Stroustrup starting in 1979 at Bell Labs.
- C++ it was designed with a bias toward system programming and embedded, resource-constrained software and large systems, with performance, efficiency, and flexibility of use as its design highlights.

About C++

- Object-Oriented Programming (OOP)
- Low-Level Manipulation
- Standard Template Library (STL)
- Multi-paradigm Programming
- Compatibility with C
- Control over System Resources
- Community and Library Support

TIOBE Index for January 2024

Jan 2024	Jan 2023	Change	Programming Language		Ratings	Change
1	1			Python	13.97%	-2.39%
2	2			C	11.44%	-4.81%
3	3			C++	9.96%	-2.95%
4	4			Java	7.87%	-4.34%
5	5			C#	7.16%	+1.43%
6	7	⬆		JavaScript	2.77%	-0.11%
7	10	⬆		PHP	1.79%	+0.40%
8	6	⬇		Visual Basic	1.60%	-3.04%
9	8	⬇		SQL	1.46%	-1.04%
10	20	⬆⬆		Scratch	1.44%	+0.86%

C++ program structure

```
#include <iostream> // Header file for input/output
using namespace std; // Using standard namespace

// Function declaration
void greet();

// The main function
int main() {
    greet(); // Call function
    return 0; // Return statement
}

// Function definition
void greet() {
    cout << "Hello, World!" << endl; // Print message
}
```


C++ Programs and modules

- **Structure of a simple C++ program:**
 - **Header files:** Include the necessary C++ standard libraries and any custom header files.
 - **Main function:** The entry point of a C++ program. Execution starts here.
 - **Functions/Classes:** Define additional functions and classes used in the program.

C++ Modules

- Modules in C++ refer to a way of organizing code into separate, independent units. Each module typically contains related functions, classes, and data. The concept of modules helps in managing large codebases by breaking them down into smaller, more manageable pieces.

C++ Modules

1. Header files:

1. Files with .h or .hpp extension.
2. Declare the interfaces to your code (e.g., classes, functions, constants).
3. Included in source files with the #include directive.

2. Source files:

1. Files with .cpp extension.
2. Contain the implementation of the code declared in header files.
3. Each source file is compiled separately.

3. Use of modules:

1. **Encapsulation:** Modules encapsulate specific functionality, making the codebase more organized and maintainable.
2. **Reusability:** Functions and classes defined in one module can be reused in other parts of the program.
3. **Separation of interface and implementation:** Keeping declarations in headers and definitions in source files separates the interface from the implementation.

C++ Modules

In a C++ project, modules are compiled independently. The linker then combines these compiled units (object files) into a single executable. This process involves resolving references to functions and variables across different modules.

Toolchains and building C++ programs

Building a C++ program, especially a complex one with multiple modules, requires a toolchain. A toolchain is a set of programming tools used to perform a complex software development task or to create a software product. In the context of C++, a typical toolchain includes a compiler, a linker, and often a build system.

Compiler

- The most essential tool, it converts C++ code into machine code.

Common C++ compilers include:

- **GCC (GNU Compiler Collection)**: A free, open-source compiler for C and C++.
- **Clang**: Part of the LLVM project, known for its excellent diagnostics (error and warning messages).
- **MSVC (Microsoft Visual C++)**: The C++ compiler from Microsoft, used in Visual Studio.
- **Intel C++ Compiler**: Known for its optimization capabilities, especially on Intel processors.

Linker

- It combines object files (compiled source files) into a single executable or library. The linker resolves references to functions and variables across files.

Build system/make tool

- Automates the process of compiling and linking.
- Examples include **Make**, **CMake**, **Ninja**, and **MSBuild**.
- They use configuration files (Makefile, CMakeLists.txt, etc.) to manage build rules and dependencies.

Integrated development environment (IDE)

- Provides an interface for managing complex projects.
- Popular C++ IDEs include **Visual Studio**, **CLion**, **Qt Creator**, and **Eclipse CDT**.
- IDEs integrate compilers, debuggers, and often provide a GUI for build systems.

Best practices

- Use a version control system: Like Git, to manage your source code.
- Consistent coding standards: Maintain readability and maintainability.
- Regularly build and test: To catch issues early.
- Document your code: Especially public interfaces and complex algorithms.

C++ code style

- <https://google.github.io/styleguide/cppguide.html>

Input-output

- The `iostream` library includes classes for console I/O operations:
 - **`cin`**: Standard input stream (typically the keyboard).
 - **`cout`**: Standard output stream (typically the console screen).
 - **`cerr`**: Standard error output stream (also the console screen but can be redirected separately).
 - **`clog`**: Similar to `cerr`, but used for logging.

Input-output

```
#include <iostream>
using namespace std;

int main() {
    int number;
    cout << "Enter a number: ";
    cin >> number;
    cout << "You entered: " << number << endl;
    return 0;
}
```

Standard C++ Data Types

- Integer (signed/unsigned);
- Real (floating point);
- Symbolic;
- Logical;
- `void`;
- Pointer

Standard C++ Data Types

Literal is an entry in the source code of a program that represents a fixed value.

A variable is a named area of memory that stores data of a specific type.

Literals : 32U, 16, 0x10, 07, `'\n'`, `"qwerty"`

Variable declaration:

```
int a;
```

Initializing variables:

```
unsigned int b = 32U;
```

```
unsigned int c = b, d = 1;
```

Standard C++ Data Types

The size of a type can be found using the operator `sizeof(type)`

```
sizeof(int)
```

```
double a;
```

```
sizeof(a)
```


Standard C++ Data Types

Integer types:

- **long long int**
- **long int**
- **int**
- **short int**
- **char**

Standard C++ Data Types

The signed/unsigned nature of a type is specified by the `signed/unsigned` keywords. By default, types are considered signed.

- `unsigned long`
- `signed int`
- `signed short`
- `unsigned char`

Standard C++ Data Types

The size of an `int` type can be adjusted using the `short` and `long` keywords.

- `long int`
- `short int`
- `unsigned long long int`

It is acceptable not to specify the `int` type:

- `long`
- `short`
- `unsigned long long`

Standard C++ Data Types

Type size ratio:

$$\begin{aligned} \text{sizeof}(\text{char}) &\leq \text{sizeof}(\text{short}) \leq \text{sizeof}(\text{int}) \leq \text{sizeof}(\text{long}) \\ &\leq \text{sizeof}(\text{long long}) \end{aligned}$$

Standard C++ Data Types

Sizes of integer types on the x86-64 platform

Type	Size, bytes	
char	1	$-2^7 \dots 2^7 - 1$
short	2	$-2^{15} \dots 2^{15} - 1$
int	4	$-2^{31} \dots 2^{31} - 1$
long	4	$-2^{31} \dots 2^{31} - 1$
long long	8	$-2^{63} \dots 2^{63} - 1$

Standard C++ Data Types

Arithmetic operations with integer variables

=
+
-
*
/
%
++
--

+=
-=
*=
/=
%=

Standard C++ Data Types

```
int a = 77;
```

```
a = 77
```

```
int b = a % 5;
```

```
b = 2
```

```
a += 5;
```

```
a = 82
```

```
a++;
```

```
a = 83
```

Standard C++ Data Types

- Real data types:
- *float* (single precision)
- *double* (double precision)
- *long double* (extended precision)

$$\textit{sizeof}(\textit{float}) \leq \textit{sizeof}(\textit{double}) \leq \textit{sizeof}(\textit{long double})$$

Standard C++ Data Types

Size of real types

Type	Size, bytes
float	4
double	8

Arithmetic operations with floating point numbers are the same as with integer numbers. In this case, the modulo division operation is not defined.

Standard C++ Data Types

Boolean type:

- **bool**
- Takes one of two values: true or false.
- Typically, false is interpreted as 0, true as any non-zero value.
- The type size is 1 byte.