HEINZ NIXDORF INSTITUT
UNIVERSITÄT PADERBORN

C++ Programming

Lecture 0

Secure Software Engineering Group

Philipp Dominik Schubert

SECURE SOFTWARE ENGINEERING GROUP

The C++ Programming Language

C++ is easy.

It’s like riding a bike.

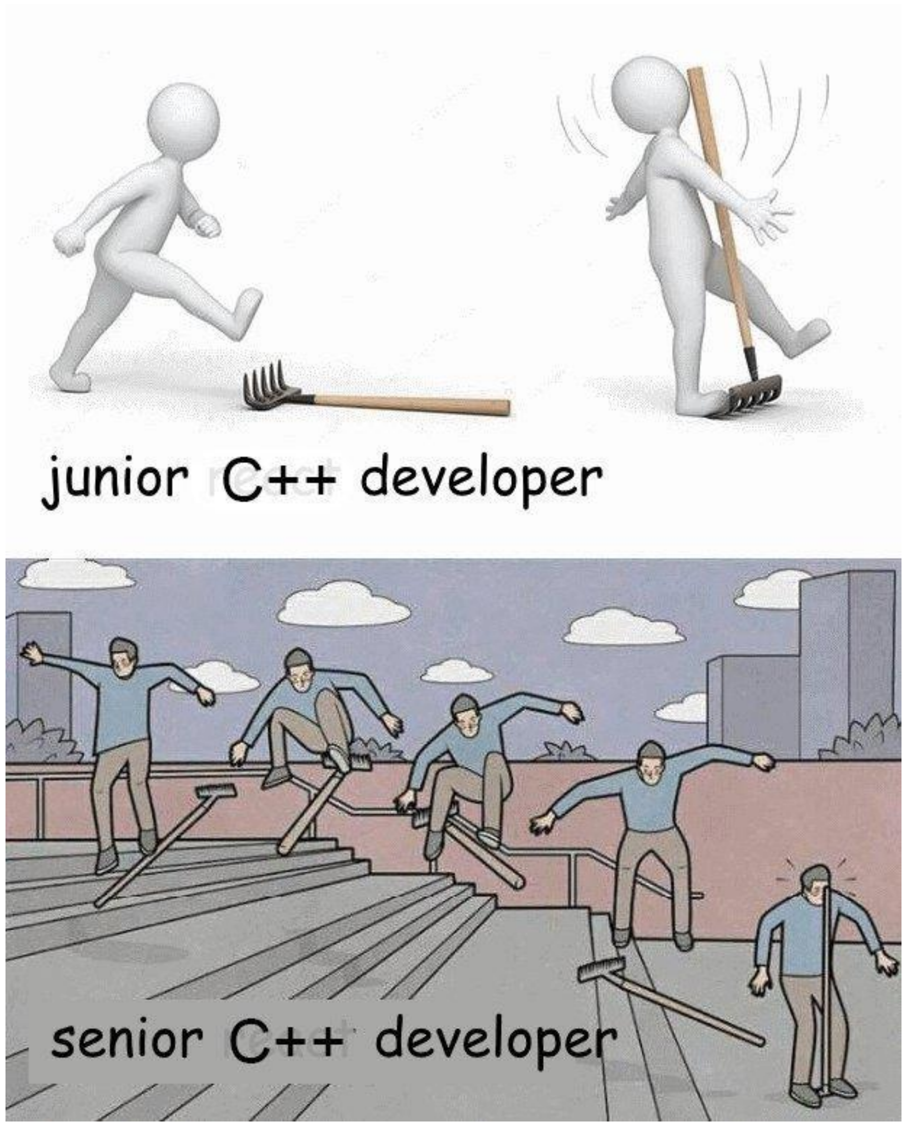
Except the bike is on fire,

and you’re on fire

and everything is on fire

because you’re in hell.

The C++ Programming Language



Contents

Organization

- "Rooms"
  - Lecture: recorded (Panda/YouTube), available on Fridays ~14:00
  - Exercises: livestream (Twitch), Fridays 16:00-18:00
- Instructor
  - Philipp Schubert @home in BI
  - E-Mail philipp.schubert@upb.de
  - Web https://www.hni.uni-paderborn.de/sse/lehre/cppp/
- Prerequisites
  - No programming experience
  - Knowledge on how to use a computer
    - Text editor
    - Operating system (Linux/Windows/Mac)

SECURE SOFTWARE ENGINEERING GROUP

Fraunhofer IEM

GaZAR

Organization

- Benefits
  - Be confident to take advanced courses that require C++
  - Realize programming projects
  - Will be useful for computational thinking
  - Better understanding on how a computer works
  - Well-paid jobs
- Studium Generale (SG) EIM-I
  - Computer science students will not receive credit points
  - Electrical engineering students will not receive credit points
  - When in doubt ask your examination office
  - All (?/most) other students will receive 4 credit points
  - Everyone obtains a nice certificate for their CV

Get the book

theboostcpplibraries.com

$45.95 Print
$9.99 Kindle
$9.99 E-book
$9.99 PDF

… oder das Training

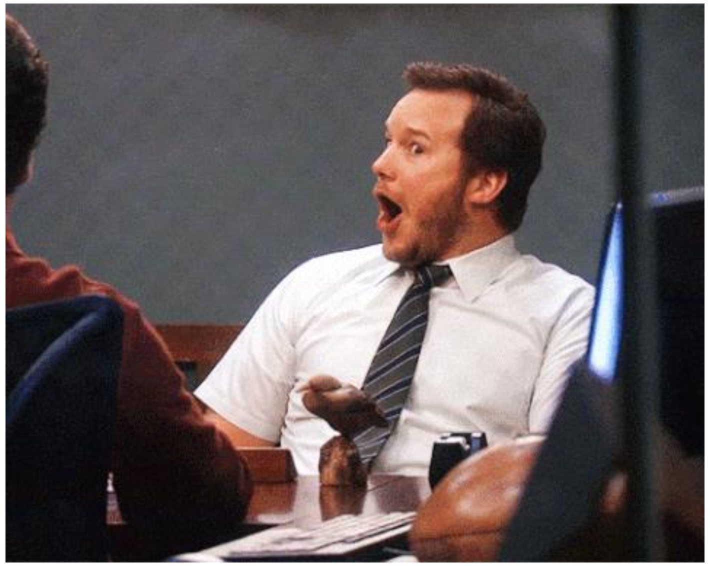
boost-cpp-master-class.eventbrite.de

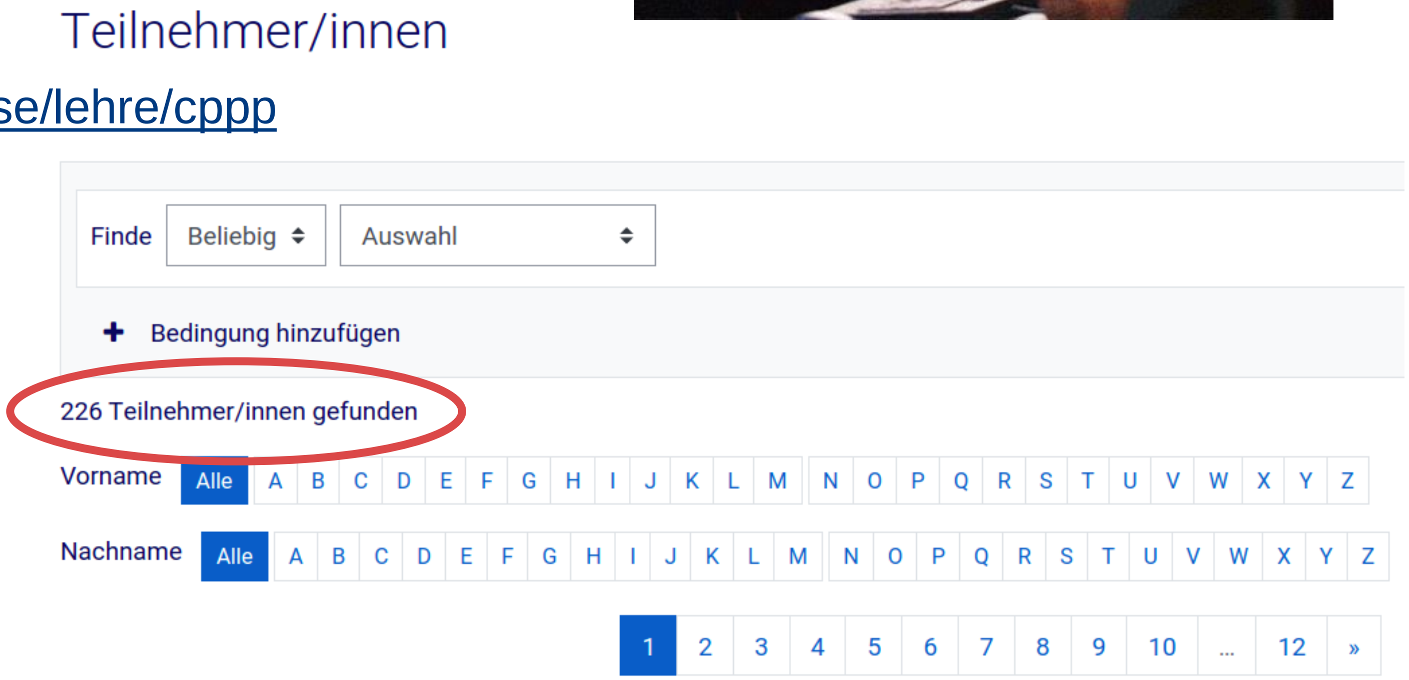
Boost C++ Master Class, mit Boris Schäling, 6-8 Dez, Berlin, 2490 EUR

[Figures taken from https://c2.staticflickr.com/8/7693/17123251389_bed3c3a1ba_b.jpg, https://theboostcpplibraries.com/

Organization

- Some of you have not yet registered?
  - Register to this course in Panda
    - https://panda.uni-paderborn.de/course/view.php?id=22691
    - I will send emails with additional materials
  - External students
    - https://www.hni.uni-paderborn.de/sse/lehre/cppp




Teilnehmer/innen

226 Teilnehmer/innen gefunden

Finde
Beliebig
Auswahl
Bedingung hinzufügen
Vorname Alle A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
Nachname Alle A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
1 2 3 4 5 6 7 8 9 10 ... 12 »

Contents

Course outline

- Basic introduction
  - History of C & C++
  - Compilers
  - Development environments
  - Basic terms and concepts
- Basic C++ programming
  - Primitive data types, strings, vectors, arrays, pointers
  - Expressions, statements
  - Structures, unions, enumerations
  - Functions, classes

Course outline

- How to organize a project
  - Tooling
  - Namespaces
  - Forward declarations
- C++' Standard Template Library (STL)
  - IO, containers, generic algorithms
  - Static / dynamic memory
  - Smart pointers
- Advanced techniques
  - Copy control, standard class members
  - Operator overloading
  - Object-oriented programming
  - Templates and generic programming

Course outline

- Useful libraries
  - OpenMP, OpenCV, OpenCL, OpenGL/Vulkan, …
  - Qt
  - Google test
  - Google protobuf
  - Abseil
  - Boost
  - And other useful libraries
  - Where to find the desired information you need
  - Don’t reinvent the wheel, use libraries

Literature

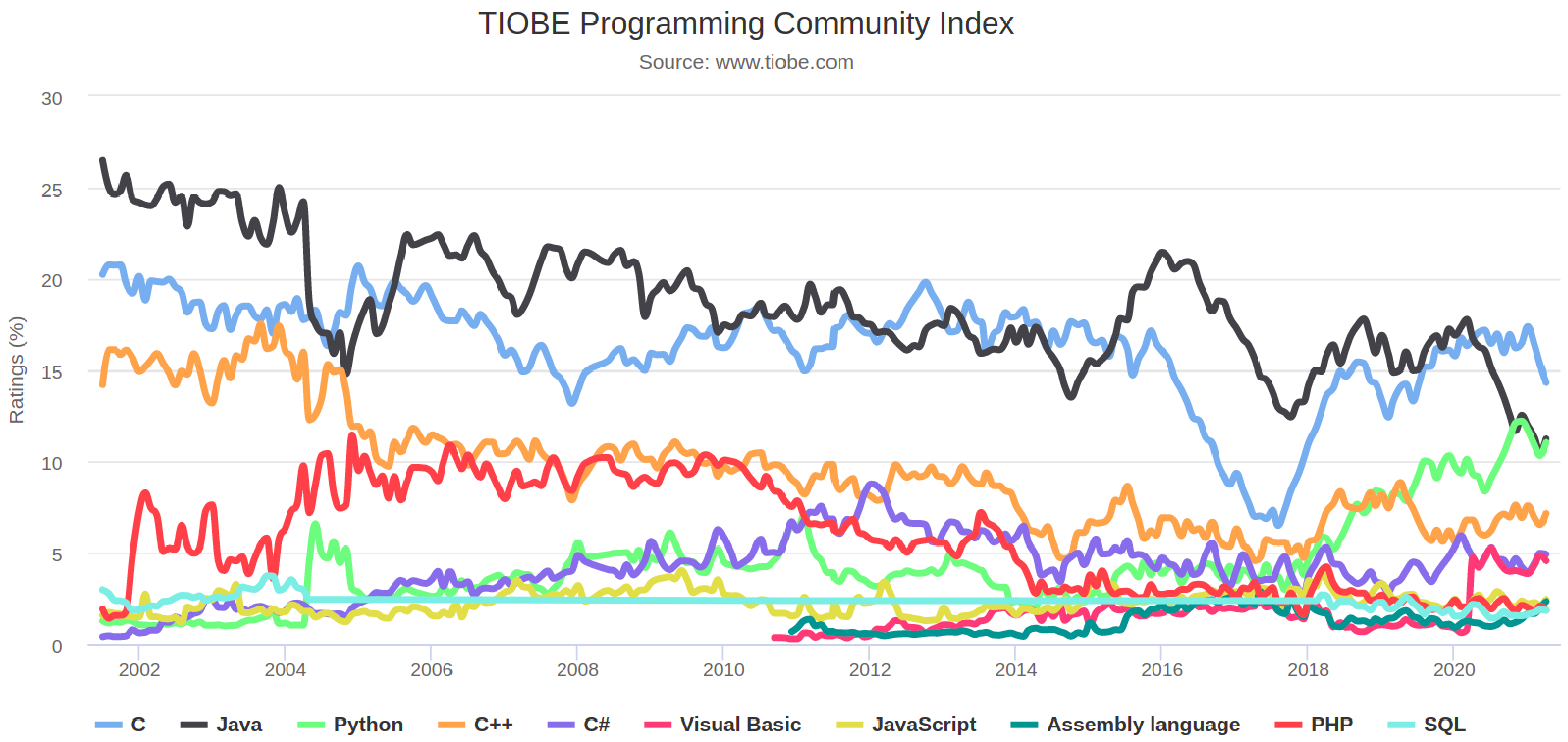
- [1] A Tour of C++, Stroustrup 2013
- [2] Programming – Principles and Practice using C++, Stroustrup 2015
- [3] The C++ Programming Language (4th Edition), Stroustrup 2013
- [4] C++ reference, http://en.cppreference.com/
- [5] CppCon, https://www.youtube.com/user/cppcon/
- [6] Effective Modern C++, Meyers 2015
- [7] Online tutorial: http://www.cplusplus.com/doc/tutorial/
- Various different input channels are important:
  - Lecture
  - Exercises
  - I'll try to make links to books and YouTube videos
  - Talk to each other and look things up

Exercises

- Weekly exercises
  - Theoretical and practical exercises
- Submissions are graded
  - You need to achieve 50% during semester
- Final project
  - Solve a programming task
- Certificate (+ credit points)
  - Pass exercises + project solved
  - No final exams
- Plagiarism is prohibited (Plage Source Code Copying Detector https://sourceforge.net/projects/plage/)
- Adhere to the notes on the exercise sheets
- Questions so far?

Contents

What is C++?


TIOBE Programming Community Index
Source: www.tiobe.com
Ratings (%)
30
25
20
15
10
5
0
2002
2004
2006
2008
2010
2012
2014
2016
2018
2020
C
Java
Python
C++
C#
Visual Basic
JavaScript
Assembly language
PHP
SQL

What is C++?

- An object-oriented programming language
- Generic Programming
- Template meta-programming
- Buffer overflows
- Classes
- Too big
- Host for DSLs
- A hybrid language
- Embedded systems
- Low level
- A random collection of features
- Class hierarchies
- Multi-paradigms
- A failed attempt to build Java
- It’s **C**
- Too complicated

Stroustrup: The Essence of C++

What is C++?

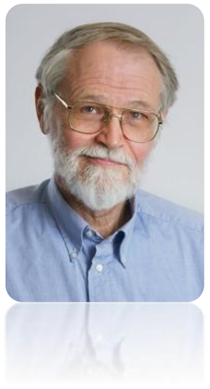
THE VIRGIN RUST
need to use unsafe blocks to make programming exciting
cannot write a compile-time raytracer yet
no feeling of reward when dealing with complicated library dependency graphs
only requires you to write function definitions once instead of three times
doesn't let you swap array elements
assumes you can't keep track of object lifetimes in your head
you don't have to creatively implement pattern matching yourself by abusing lambdas
THE CHAD++
multiple compilers, each one with different bugs
provides several ways of initializing variables to promote diversity
Effective C++ Third Edition
sometimes segfaults to keep you on your toes
ISO
literally has 3 or 4 accidentally turing-complete languages embedded into it
allows you to create compile-time tetris
provides job security
never removes features to preserve backward compatibility
fast like sonic
it's probably UB
@supahvee1234

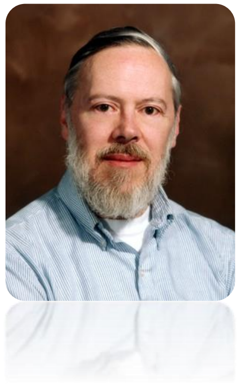
Advice

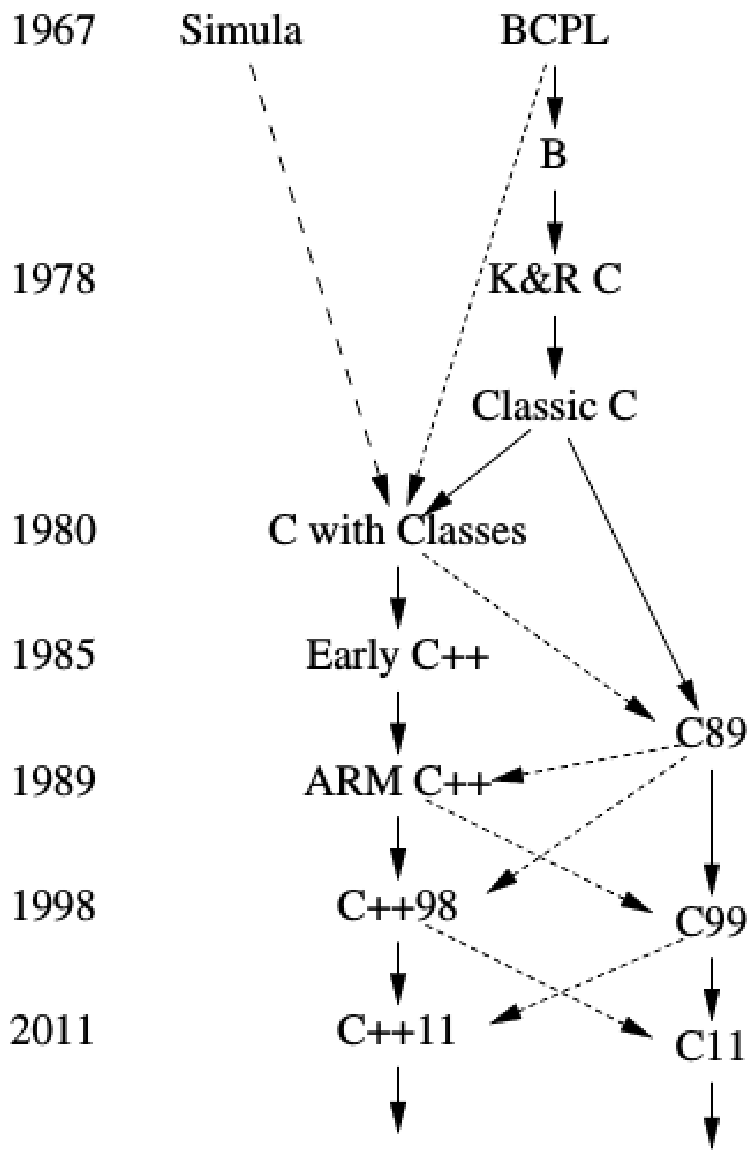
- Don’t be afraid
- Learning a new language takes time
- Practice, practice, practice
- Read a lot about it (books and C++ forums / as well as code)
- Do the exercises
- Always ask yourself: why does this work?
    - If you are curious about something → use google
        - … and share your knowledge and discuss with friends
- Programming will be fun when understood

History of C++

- All started with **BCPL**
  - Basic Combined Programming Language
  - Has no data types
- **B** – a language to implement operating systems
- **C** – better than **B**
  - Brian Wilson Kernighan
  - Dennis MacAlister Ritchie
- **C** with Classes
  - Bjarne Stoustrup
- **C++**
  - Dynamically evolving
  - **C++14/C++17/C++20**



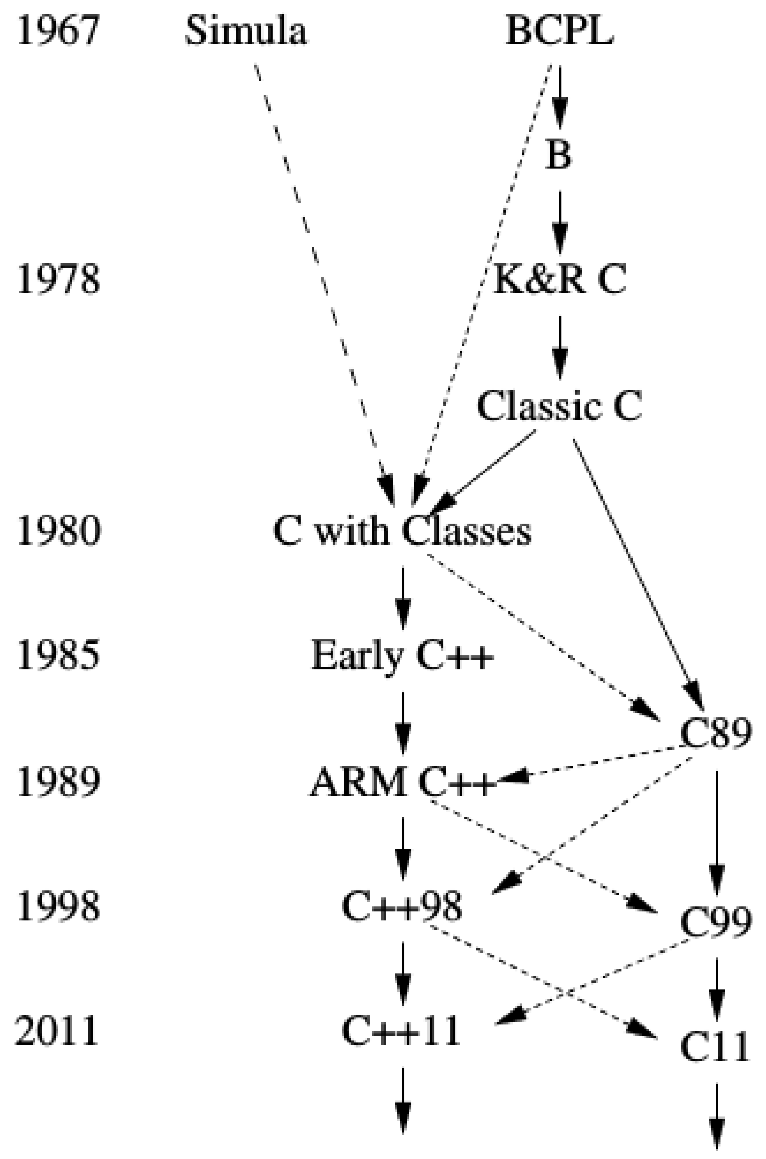




[Figure and images taken from images.google.de/ and A Tour of C++, Stroustrup 2013]

History of C++

- But why are we not learning C++20?
- I cannot teach five courses in one
- Adaption needs time
  - Concepts and ideas first
  - Compiler implementations follow
  - // void …
  - Industry usually adapts ~ 5-10 years later
    - There are reasons for that
      - Concepts have to be proven as useful
      - Compilers have to mature over time



[Figure from A Tour of C++, Stroustrup 2013]

History of C++

- BCPL, B, C,
  - Why not D after C?
  - C was and is still tremendously successful
  - Lots of existing code was and is still written in C
  - Don’t break compatibility!
  - Be an increment rather than a new language
  - A language called D exists
    - D is no longer compatible with C
  - Be aware: Modern C++ is not C

Contents

What is a compiler?

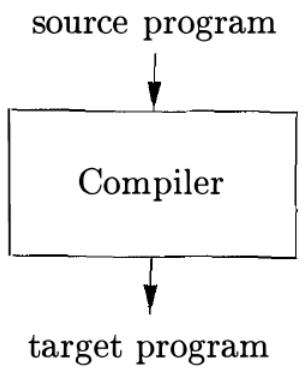


Figure 1.1: A compiler

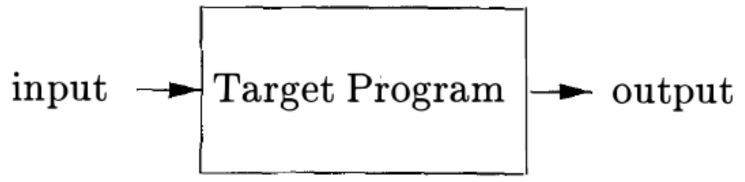


Figure 1.2: Running the target program

Compilers: Principles, Techniques, & Tools, Aho, Lam, Sethi, Ullman 2007

Are there other forms? Interpreter

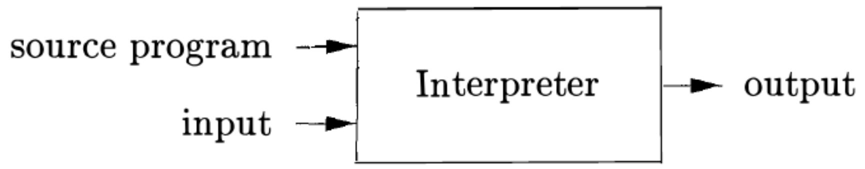


Figure 1.3: An interpreter

Even more: hybrid compilers

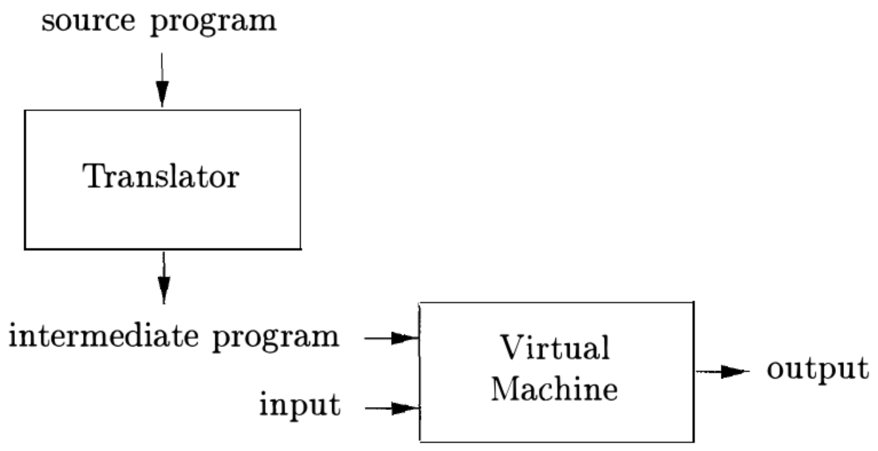


Figure 1.4: A hybrid compiler

C++ compilers

- **G**nu **C**ompiler **C**ollection **GCC**
  - Includes C and C++ front-ends
  - Standard on most Linux dists.
  - “Most used C/C++ compiler in the world”
  - Fist stable release was v1.17 (1988)
  - Monolithic design
  - Written by bootstrapping
    - Written by *something else* until its powerful enough to compile itself

- **Clang**
  - Compiler front-end for C-like languages (including C and C++)
  - Used by Google, Apple, Oracle …
  - Started as a Ph.D. thesis by Chris Lattner
  - Stable version in 2009
  - Part of a reusable compiler infrastructure (LLVM project)
  - Written in C++

There are a lot more: Intel icc, IBM C++, MSVS C++, Oracle ++, Apple C++, Bloodshed Dev-C++, EDG C++

GCC and Clang are language processing systems

- C++ is (usually) a compiled language
- C++ compilers are language processing systems / compiler tool chains

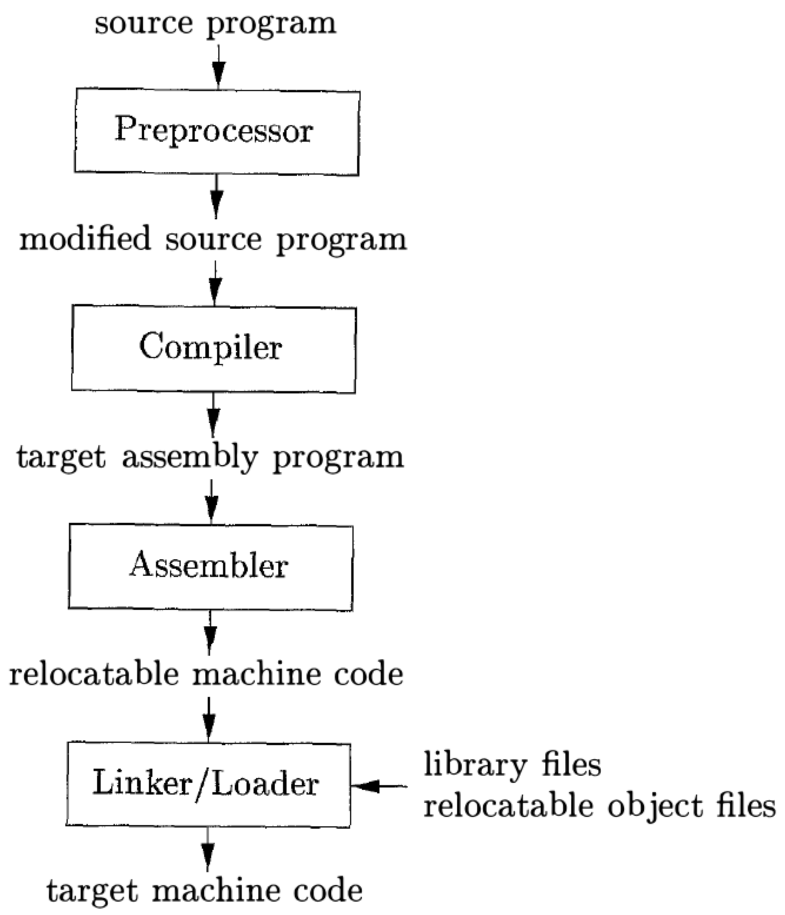


Figure 1.5: A language-processing system

Compilers: Principles, Techniques, & Tools, Aho, Lam, Sethi, Ullman 2007

Remark on what follows

- “Keep simple things simple,

  as simple as possible, but not simpler!” (Einstein)
- Problem: where to start when learning a programming language?
  - It all seems like magic
  - In order to be able to start **at all** we have to …
    1. take certain things for granted
    2. learn the WHY over time

Contents

A “Hello, World!” program

- Shortest valid C++ program
- A “Hello, World!” program
  - Uses a header file
  - A comment
  - `main()` function (with arguments)
  - Uses a namespace
  - **::** scope and **<<** shift operator
  - Uses a string literal and a variable (`cout`)
  - `return 0;` a value that is returned to the OS
    - ‘`0`’ indicates success
    - Values other than ‘`0`’ indicate failure

```
int main() { return 0; }
or int main() {}
```

```
#include <iostream>
// This function prints Hello, World!
int main(int argc, char **argv) {
    std::cout << "Hello, World!\n";
    return 0;
}
```

A “Hello, World!” program

- Tell the compiler to translate ‘hello.cpp’ into executable machine code
- Command:
  - `cc hello.cpp -o hello`
  - You can execute the program ‘`hello`’ with `./hello`
- Replace `cc` with `g++` or `clang++`

Edit a text file, e.g. ‘`hello.cpp`’, with the following contents:

```
#include <iostream>
int main(int argc, char **argv){
    std::cout << "Hello, World!\n";
    return 0;
}
```

A “Hello, World!” program

- Some useful compiler flags

`-Wall`	turns on compiler warning
`-Wextra`	turns on even more warnings
`-g`	insert debugging symbols
`-Ox`	turn on compiler optimization (x is a number: 0,1,2,3)
`-o`	specify the output file
`-std=X`	specify the C++ standard e.g. `-std=c++17` or `-std=c++20`

Edit a text file, e.g. ‘`hello.cpp`’, with the following contents:

```
#include <iostream>
int main(int argc, char **argv){
    std::cout << "Hello, World!\n";
    return 0;
}
```

- E.g.

```
g++ -Wall -Wextra -std=c++17 hello.cpp -o hello
```

A “Hello, World!” program

- #-directives are instructions for the preprocessor
  - Preprocessor runs over the program first
  - Then compiler starts its job
- `#include` directives just perform textual insertion
- `std::` is a namespace
  - Namespaces hold code
  - Helps to avoid collisions (e.g. variable names, function names, …)

```
#include <iostream>
int main(int argc, char **argv) {
   std::cout << "Hello, World!\n";
   return 0;
}
```

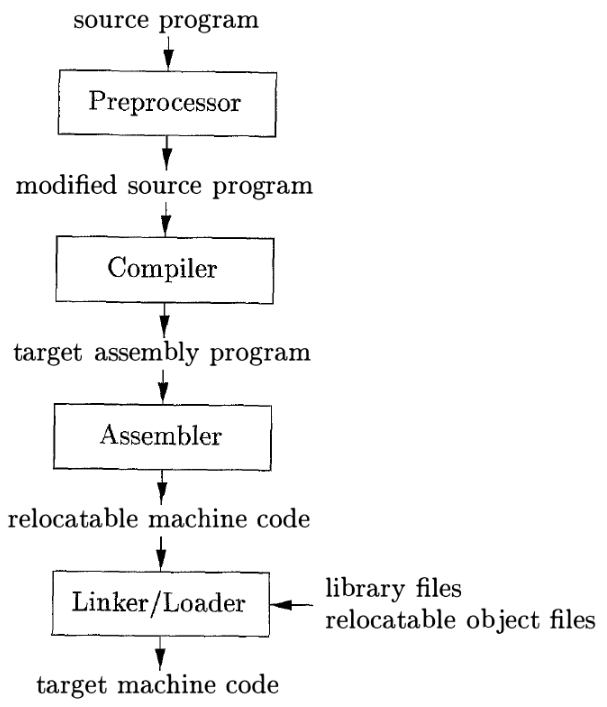


Figure 1.5: A language-processing system

[Figure from Compilers: Principles, Techniques, & Tools, 2007]

A “Hello, World!” program

- Compiler option `-S` shows the assembly code
- `cc hello.cpp -S -o hello.as`

```
        .file    "hello.cpp"
        .local   _ZStL8__ioinit
        .comm _ZStL8__ioinit,1,1
        .section         .rodata
.LC0:
        .string  "Hello World"
        .text
        .globl   main
        .type    main, @function
main:
.LFB971:
        .cfi_startproc
        pushq  %rbp
        .cfi_def_cfa_offset 16
        .cfi_offset 6, -16
        movq   %rsp, %rbp
        .cfi_def_cfa_register 6
        subq   $16, %rsp
        movl   %edi, -4(%rbp)
        movq   %rsi, -16(%rbp)
        movl   $.LC0, %esi
        movl   $_ZSt4cout, %edi
        call   _ZStlsISt11char_traitsIcEERSt13basic_ostreamIcT_ES5_PKc
        movl   $_ZSt4endlIcSt11char_traitsIcEERSt13basic_ostreamIT_T0_ES6_, %esi
        movq   %rax, %rdi
        call   _ZNSolsEPFRSoS_E
        movl   $0, %eax
…… // code still continues
```

```cpp
#include <iostream>
int main(int argc, char **argv) {
    std::cout << "Hello, World!\n";
    return 0;
}
```

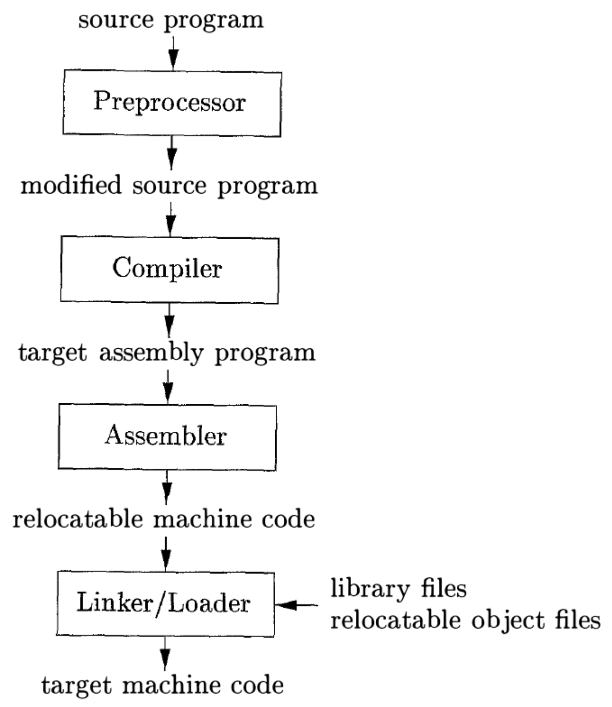
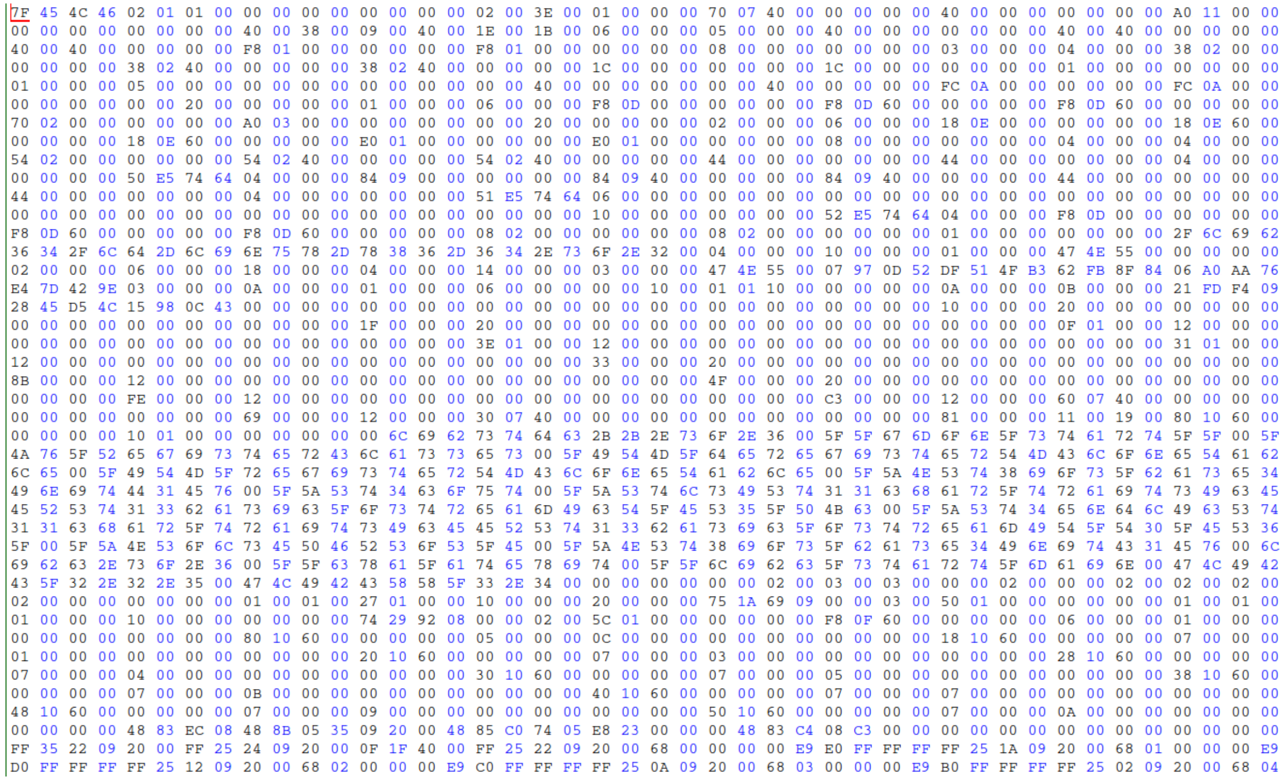


Figure 1.5: A language-processing system

A “Hello, World!” program

- Compile to binary directly
  - `cc hello.cpp -o hello`
- Content of hello looks like that

```cpp
#include <iostream>
int main(int argc, char **argv) {
    std::cout << "Hello, World!\n";
    return 0;
}
```



```
7F 45 4C 46 02 01 01 00 00 00 00 00 00 00 00 00 02 00 3E 00 01 00 00 00 70 07 40 00 00 00 00 00 40 00 00 00 00 00 00 00 A0 11 00 00
00 00 00 00 00 00 00 00 40 00 38 00 09 00 40 00 1E 00 1B 00 06 00 00 00 05 00 00 00 40 00 00 00 00 00 00 00 40 00 40 00 00 00 00 00
40 00 40 00 00 00 00 00 F8 01 00 00 00 00 00 00 F8 01 00 00 00 00 00 00 08 00 00 00 00 00 00 00 03 00 00 00 04 00 00 00 38 02 00 00
00 00 00 00 38 02 40 00 00 00 00 00 38 02 40 00 00 00 00 00 1C 00 00 00 00 00 00 00 1C 00 00 00 00 00 00 00 01 00 00 00 00 00 00 00
01 00 00 00 05 00 00 00 00 00 00 00 00 00 00 00 00 00 40 00 00 00 00 00 00 00 40 00 00 00 00 00 FC 0A 00 00 00 00 00 00 FC 0A 00 00
00 00 00 00 00 00 20 00 00 00 00 00 01 00 00 00 06 00 00 00 F8 0D 00 00 00 00 00 00 F8 0D 60 00 00 00 00 00 F8 0D 60 00 00 00 00 00
70 02 00 00 00 00 00 00 A0 03 00 00 00 00 00 00 00 00 20 00 00 00 00 00 02 00 00 00 06 00 00 00 18 0E 00 00 00 00 00 00 18 0E 60 00
00 00 00 00 18 0E 60 00 00 00 00 00 E0 01 00 00 00 00 00 00 E0 01 00 00 00 00 00 00 08 00 00 00 00 00 00 00 04 00 00 00 04 00 00 00
54 02 00 00 00 00 00 00 54 02 40 00 00 00 00 00 54 02 40 00 00 00 00 00 44 00 00 00 00 00 00 00 44 00 00 00 00 00 00 00 04 00 00 00
00 00 00 00 50 E5 74 64 04 00 00 00 84 09 00 00 00 00 00 00 84 09 40 00 00 00 00 00 84 09 40 00 00 00 00 00 44 00 00 00 00 00 00 00
44 00 00 00 00 00 00 00 04 00 00 00 00 00 00 00 51 E5 74 64 06 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 10 00 00 00 00 00 00 00 52 E5 74 64 04 00 00 00 F8 0D 00 00 00 00 00 00
F8 0D 60 00 00 00 00 00 F8 0D 60 00 00 00 00 00 08 02 00 00 00 00 00 00 08 02 00 00 00 00 00 00 01 00 00 00 00 00 00 00 2F 6C 69 62
36 34 2F 6C 64 2D 6C 69 6E 75 78 2D 78 38 36 2D 36 34 2E 73 6F 2E 32 00 04 00 00 00 10 00 00 00 01 00 00 00 47 4E 55 00 00 00 00 00
02 00 00 00 06 00 00 00 18 00 00 00 04 00 00 00 14 00 00 00 03 00 00 00 47 4E 55 00 07 97 0D 52 DF 51 4F B3 62 FB 8F 84 06 A0 AA 76
E4 7D 42 9E 03 00 00 00 0A 00 00 00 01 00 00 00 06 00 00 00 00 00 10 00 01 01 10 00 00 00 00 00 0A 00 00 00 0B 00 00 00 21 FD F4 09
28 45 D5 4C 15 98 0C 43 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 10 00 00 00 20 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 1F 00 00 00 20 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 0F 01 00 00 12 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 3E 01 00 00 12 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 31 01 00 00
12 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 33 00 00 00 20 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
8B 00 00 00 12 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 4F 00 00 00 20 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 FE 00 00 00 12 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 C3 00 00 00 12 00 00 00 60 07 40 00 00 00 00 00
00 00 00 00 00 00 00 00 69 00 00 00 12 00 00 00 30 07 40 00 00 00 00 00 00 00 00 00 00 00 00 00 81 00 00 00 11 00 19 00 80 10 60 00
00 00 00 00 10 01 00 00 00 00 00 00 00 6C 69 62 73 74 64 63 2B 2B 2E 73 6F 2E 36 00 5F 5F 67 6D 6F 6E 5F 73 74 61 72 74 5F 5F 00 5F
4A 76 5F 52 65 67 69 73 74 65 72 43 6C 61 73 73 65 73 00 5F 49 54 4D 5F 64 65 72 65 67 69 73 74 65 72 54 4D 43 6C 6F 6E 65 54 61 62
6C 65 00 5F 49 54 4D 5F 72 65 67 69 73 74 65 72 54 4D 43 6C 6F 6E 65 54 61 62 6C 65 00 5F 5A 4E 53 74 38 69 6F 73 5F 62 61 73 65 34
49 6E 69 74 44 31 45 76 00 5F 5A 53 74 34 63 6F 75 74 00 5F 5A 53 74 6C 73 49 53 74 31 31 63 68 61 72 5F 74 72 61 69 74 73 49 63 45
45 52 53 74 31 33 62 61 73 69 63 5F 6F 73 74 72 65 61 6D 49 63 54 5F 45 53 35 5F 50 4B 63 00 5F 5A 53 74 34 65 6E 64 6C 49 63 53 74
31 31 63 68 61 72 5F 74 72 61 69 74 73 49 63 45 45 52 53 74 31 33 62 61 73 69 63 5F 6F 73 74 72 65 61 6D 49 54 5F 54 30 5F 45 53 36
5F 00 5F 5A 4E 53 6F 6C 73 45 50 46 52 53 6F 53 5F 45 00 5F 5A 4E 53 74 38 69 6F 73 5F 62 61 73 65 34 49 6E 69 74 43 31 45 76 00 6C
69 62 63 2E 73 6F 2E 36 00 5F 5F 63 78 61 5F 61 74 65 78 69 74 00 5F 5F 6C 69 62 63 5F 73 74 61 72 74 5F 6D 61 69 6E 00 47 4C 49 42
43 5F 32 2E 32 2E 35 00 47 4C 49 42 43 58 58 5F 33 2E 34 00 00 00 00 00 00 00 02 00 03 00 03 00 00 00 02 00 00 00 02 00 02 00 02 00
02 00 00 00 00 00 00 00 01 00 01 00 27 01 00 00 10 00 00 00 20 00 00 00 75 1A 69 09 00 00 03 00 50 01 00 00 00 00 00 00 01 00 01 00
01 00 00 00 10 00 00 00 00 00 00 00 74 29 92 08 00 00 02 00 5C 01 00 00 00 00 00 00 F8 0F 60 00 00 00 00 00 06 00 00 00 01 00 00 00
00 00 00 00 00 00 00 00 80 10 60 00 00 00 00 00 05 00 00 00 0C 00 00 00 00 00 00 00 00 00 00 00 18 10 60 00 00 00 00 00 07 00 00 00
01 00 00 00 00 00 00 00 00 00 00 00 20 10 60 00 00 00 00 00 07 00 00 00 03 00 00 00 00 00 00 00 00 00 00 00 28 10 60 00 00 00 00 00
07 00 00 00 04 00 00 00 00 00 00 00 00 00 00 00 30 10 60 00 00 00 00 00 07 00 00 00 05 00 00 00 00 00 00 00 00 00 00 00 38 10 60 00
00 00 00 00 07 00 00 00 0B 00 00 00 00 00 00 00 00 00 00 00 40 10 60 00 00 00 00 00 07 00 00 00 07 00 00 00 00 00 00 00 00 00 00 00
48 10 60 00 00 00 00 00 07 00 00 00 09 00 00 00 00 00 00 00 00 00 00 00 50 10 60 00 00 00 00 00 07 00 00 00 0A 00 00 00 00 00 00 00
00 00 00 00 48 83 EC 08 48 8B 05 35 09 20 00 48 85 C0 74 05 E8 23 00 00 00 48 83 C4 08 C3 00 00 00 00 00 00 00 00 00 00 00 00 00 00
FF 35 22 09 20 00 FF 25 24 09 20 00 0F 1F 40 00 FF 25 22 09 20 00 68 00 00 00 00 E9 E0 FF FF FF FF 25 1A 09 20 00 68 01 00 00 00 E9
D0 FF FF FF FF 25 12 09 20 00 68 02 00 00 00 E9 C0 FF FF FF FF 25 0A 09 20 00 68 03 00 00 00 E9 B0 FF FF FF FF 25 02 09 20 00 68 04
```

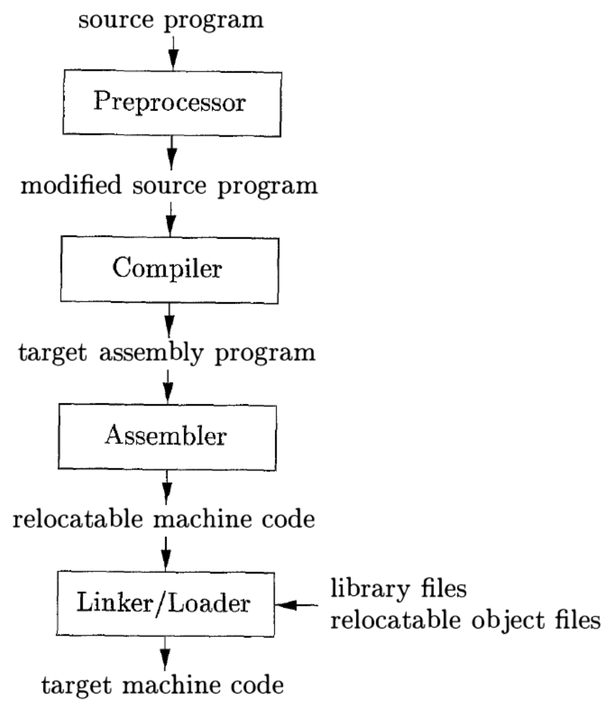


Figure 1.5: A language-processing system

[Figure from Compilers: Principles, Techniques, & Tools, 2007]

Contents

Calling the compiler by hand is *wasteful*

- Makefile, CMake, and friends
  - Help to organize a software's source code
  - Text files containing rules that describe how to invoke the compiler
  - Rules are read, identified, and executed on-demand
  - Flexible and powerful
  - Hard to write for complex tasks
    - Start with a template
  - You see what's going on
    - Nothing is hidden under the carpet

- Integrated Development Environment (IDE)
  - Handles the project and corresponding source files for you
  - Handles compiler invocations
  - Easier to use than Makefile, CMake, etc.
  - Will find syntax errors on-the-fly
  - More complex tasks are painful
    - Lack of control
  - Hides complexity

- **I'm using a combination of both!**

Makefile, an example

- Using the compiler 'by hand' is fiddly
- Use files describing the compiler commands
  - Makefile
    - Contains executable "targets"
    - Consist of a bunch of declarative rules
    - Processed by `make`
    - Flexible
    - Easy to use
    - Hard to write
      - There are books on `make`

- Project directory: `MyProject/`
  - `Makefile`

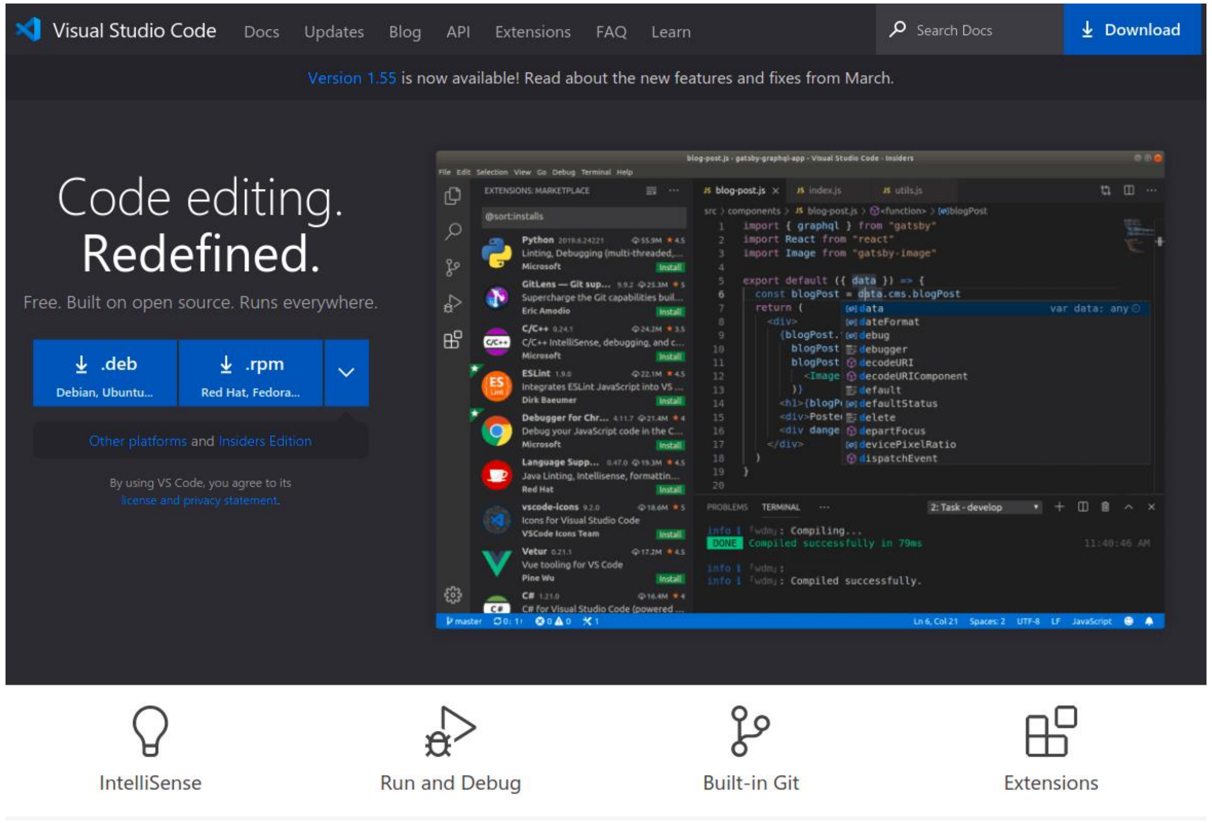
```
PROGNAME := hello
CC := g++
FLAGS := -std=c++17
FLAGS += -Wall
all: main.cpp
	$(CC) $(FLAGS) *.cpp -o $(PROGNAME)
clean:
	rm -f $(PROGNAME)
```

  - `hello.cpp`:

```
#include <iostream>
int main() {
    std::cout << "Hello, World!\n";
    return 0;
}
```

Integrated Development Environment (IDE) and other editors

- Visual Studio Code
  - Compact editor
  - Windows / Linux / Mac
- Or use vim, emacs, etc. (hardcore ;-)
- Use whatever feels best to you
  - Depending on your programming level and experience


Visual Studio Code
Docs
Updates
Blog
API
Extensions
FAQ
Learn
Search Docs
Download
Version 1.55 is now available! Read about the new features and fixes from March.
Code editing. Redefined.
Free. Built on open source. Runs everywhere.
.deb
Debian, Ubuntu...
.rpm
Red Hat, Fedora...
Other platforms and Insiders Edition
By using VS Code, you agree to its license and privacy statement.
blog-post.js - gatsby-graphql-app - Visual Studio Code - Insiders
blog-post.js
index.js
utils.js
IntelliSense
Run and Debug
Built-in Git
Extensions

```
import { graphql } from "gatsby"
import React from "react"
import Image from "gatsby-image"

export default ({ data }) => {
  const blogPost = data.cms.blogPost
  return (
    <div>
      {blogPost.
        blogPost
        blogPost
          <Image
        })
      <h1>{blogP
      <div>Poste
      <div dange
    </div>
  )
}
```

```
info ℹ ｢wdm｣: Compiling...
DONE  Compiled successfully in 79ms                    11:40:46 AM

info ℹ ｢wdm｣:
info ℹ ｢wdm｣: Compiled successfully.
```

[Figures taken from eclipse.org and code.visualstudio.com]

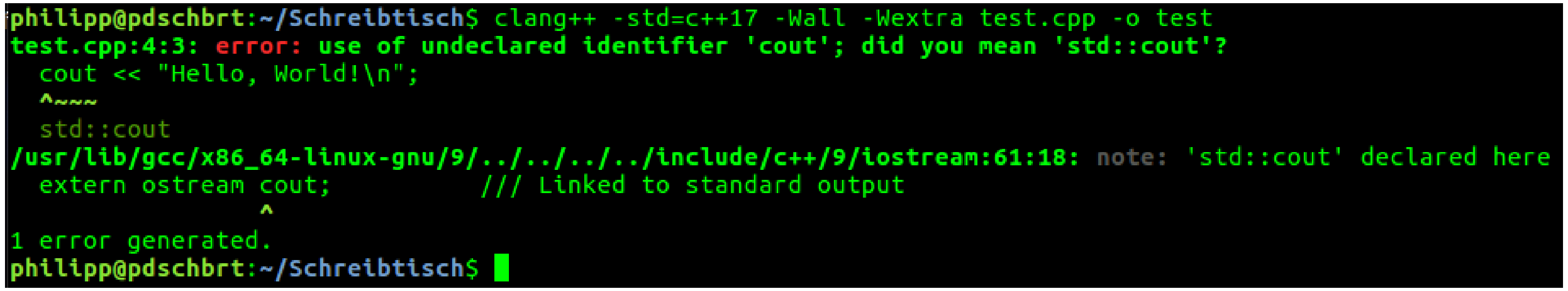
Set up a development environment

- Set up a development environment?
  - I will provide a virtual machine
  - Password: cppp
  - Ubuntu 20.04, ~20 GB (sorry)
  - Ships with everything that is needed

```cpp
#include <iostream>
int main() {
   cout << "Hello, World!\n";
   return 0;
}
```

- Remark on compiler errors
  - Errors are the default case
  - Don’t panic and read them
  - Read them carefully
  - Google will help
  - So does stack overflow

    (a programming forum)

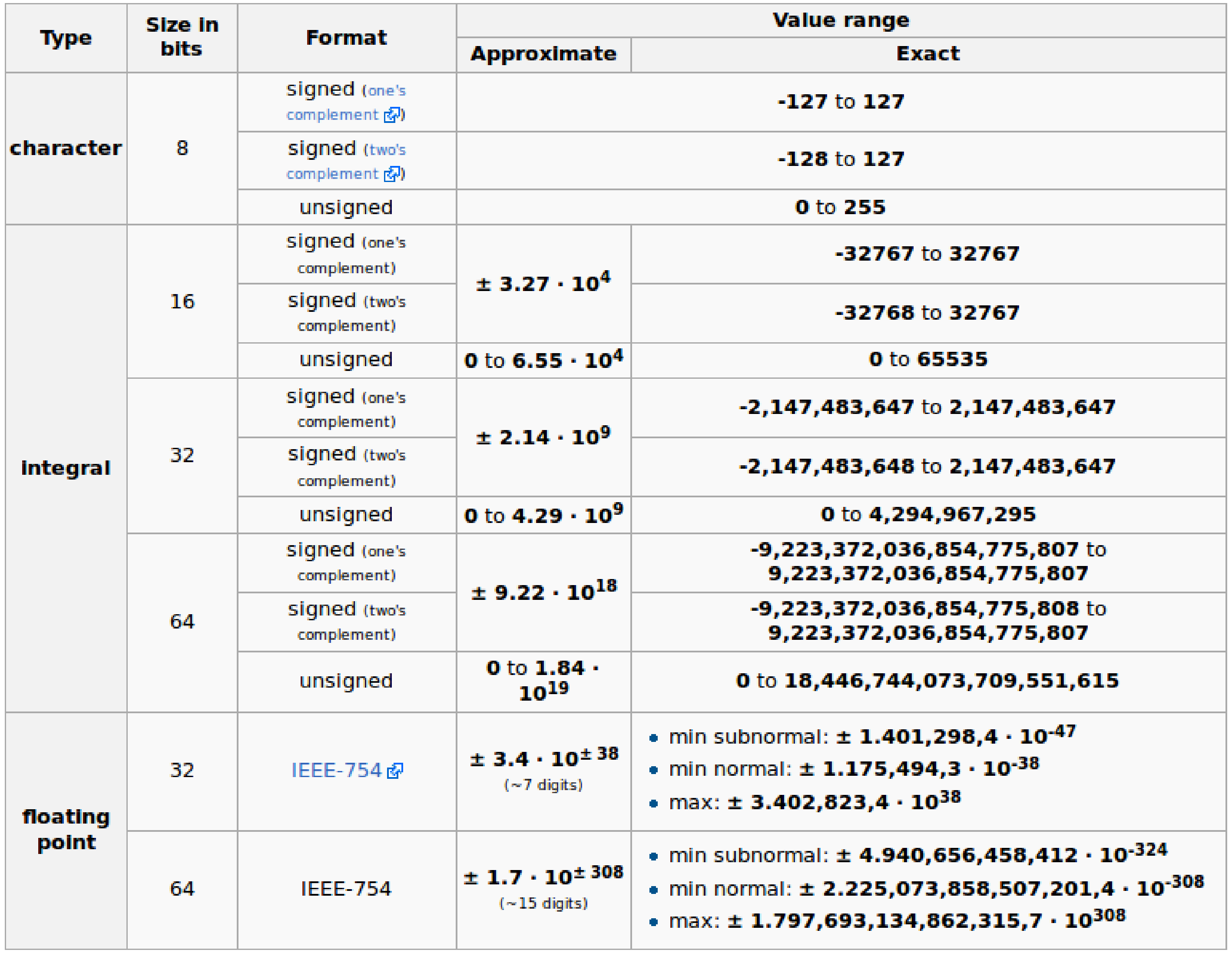


```
philipp@pdschbrt:~/Schreibtisch$ clang++ -std=c++17 -Wall -Wextra test.cpp -o test
test.cpp:4:3: error: use of undeclared identifier 'cout'; did you mean 'std::cout'?
  cout << "Hello, World!\n";
  ^~~~
  std::cout
/usr/lib/gcc/x86_64-linux-gnu/9/../../../../include/c++/9/iostream:61:18: note: 'std::cout' declared here
  extern ostream cout;          /// Linked to standard output
                 ^
1 error generated.
philipp@pdschbrt:~/Schreibtisch$
```

Contents

Primitive / built-in data types

- Boolean types
  - `bool`
  - Can hold `true` or `false`
- Character types
  - `char`
- Integer types
  - `int`
  - Modifiers and sizes (integer types only)
    - `signed` and `unsigned`
    - `short` / `long` / `long long`
- Floating point types
  - `float`
  - `double`
  - `long double`



Type	Size in bits	Format	Value range	
			Approximate	Exact
character	8	signed (one's complement)		-127 to 127
		signed (two's complement)		-128 to 127
		unsigned		0 to 255
integral	16	signed (one's complement)	$\pm 3.27 \cdot 10^4$	-32767 to 32767
		signed (two's complement)		-32768 to 32767
		unsigned	0 to $6.55 \cdot 10^4$	0 to 65535
	32	signed (one's complement)	$\pm 2.14 \cdot 10^9$	-2,147,483,647 to 2,147,483,647
		signed (two's complement)		-2,147,483,648 to 2,147,483,647
		unsigned	0 to $4.29 \cdot 10^9$	0 to 4,294,967,295
	64	signed (one's complement)	$\pm 9.22 \cdot 10^{18}$	-9,223,372,036,854,775,807 to 9,223,372,036,854,775,807
		signed (two's complement)		-9,223,372,036,854,775,808 to 9,223,372,036,854,775,807
		unsigned	0 to $1.84 \cdot 10^{19}$	0 to 18,446,744,073,709,551,615
floating point	32	IEEE-754	$\pm 3.4 \cdot 10^{\pm 38}$ (~7 digits)	• min subnormal: $\pm 1.401,298,4 \cdot 10^{-47}$ • min normal: $\pm 1.175,494,3 \cdot 10^{-38}$ • max: $\pm 3.402,823,4 \cdot 10^{38}$
	64	IEEE-754	$\pm 1.7 \cdot 10^{\pm 308}$ (~15 digits)	• min subnormal: $\pm 4.940,656,458,412 \cdot 10^{-324}$ • min normal: $\pm 2.225,073,858,507,201,4 \cdot 10^{-308}$ • max: $\pm 1.797,693,134,862,315,7 \cdot 10^{308}$

[Figure taken from Wikipedia]

Integer encoding

- `unsigned char`
  - 1 byte = 8 bit
- Dual number encoding with `unsigned`

1	**0**	**1**	**1**	**0**	**0**	**1**	**1**

Decimal value: $1 \cdot 2^7 + 0 \cdot 2^6 + 1 \cdot 2^5 + 1 \cdot 2^4 + 0 \cdot 2^3 + 0 \cdot 2^2 + 1 \cdot 2^1 + 1 \cdot 2^0$

$= 128 + 32 + 16 + 2 + 1 = 179$

Integer encoding

- `signed char` or `char`
  - 1 byte = 8 bit
- Two's complement encoding with `signed` or as default

1	0	1	1	0	0	1	1

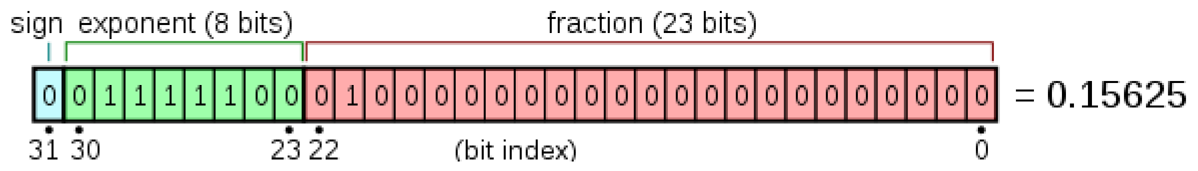
  - Highest bit encodes sign
  - Other bits encode value
  - Here: sign bit 1, number is negative: take two's complement (negate and add 1)

-	1	0	0	1	1	0	0	Take complement
-	1	0	0	1	1	0	1	Add one

Decimal value: $1 \cdot 2^6 + 0 \cdot 2^5 + 0 \cdot 2^4 + 1 \cdot 2^3 + 1 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0 = 64 + 8 + 4 + 1 = 77 \rightarrow -77$

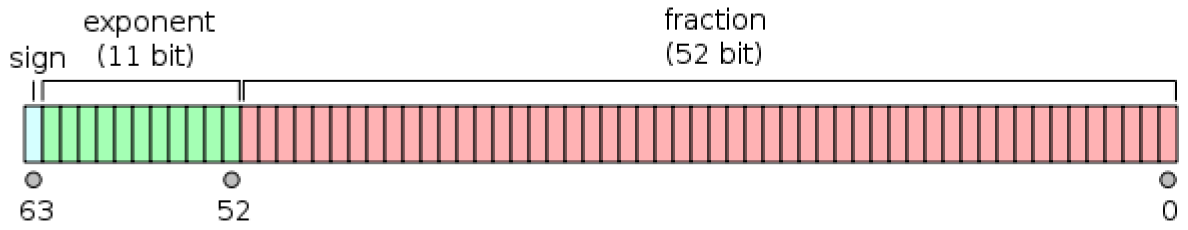
Floating point number encoding

- IEEE-754 single-precision binary floating-point format



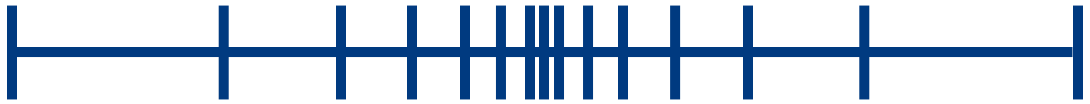
$$\text{value} = (-1)^{\text{sign}} \times \left(1 + \sum_{i=1}^{23} b_{23-i} 2^{-i}\right) \times 2^{(e-127)}$$

- IEEE-754 double-precision binary floating-point format



$$(-1)^{\text{sign}} \left(1 + \sum_{i=1}^{52} b_{52-i} 2^{-i}\right) \times 2^{e-1023}$$

- Remark
  - Use double as default, float usually far too imprecise
  - Floating point numbers are not distributed equidistant



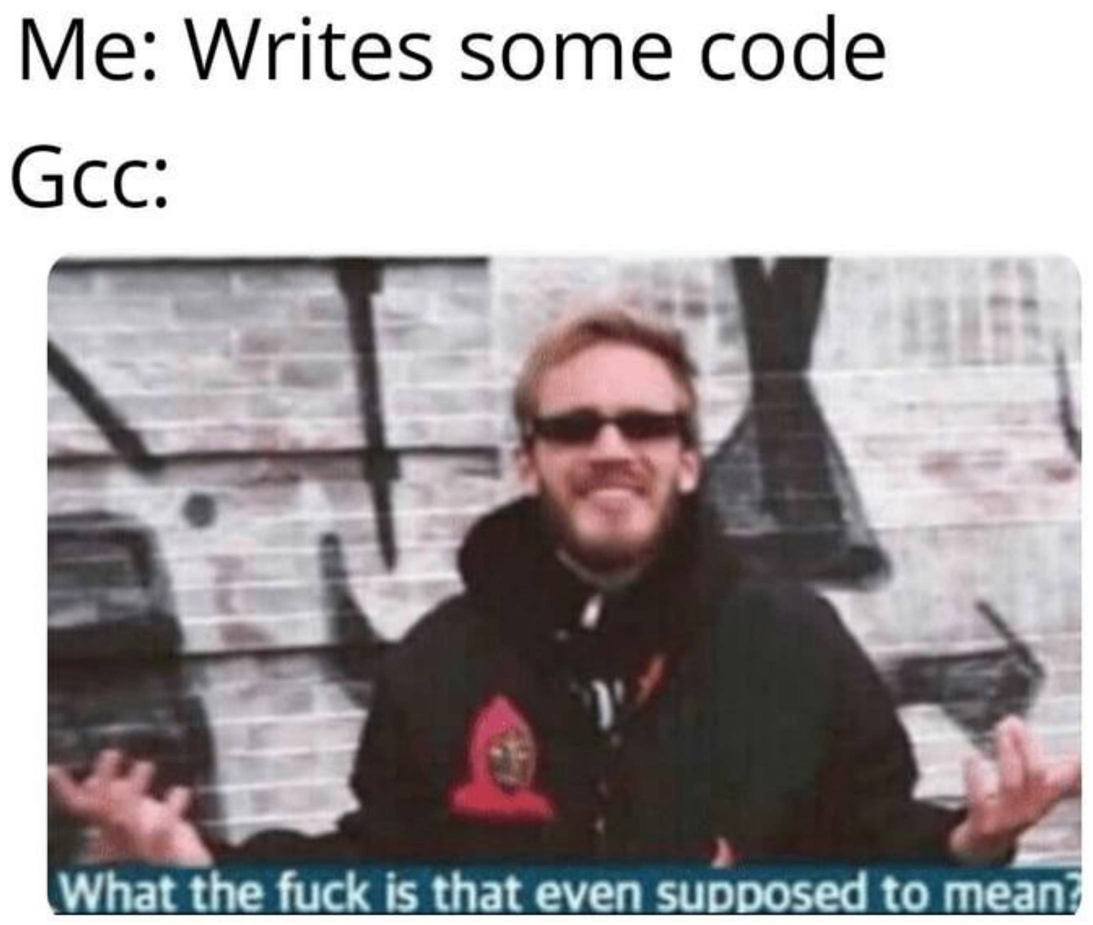
[Figures from Wikipedia]

Comments in C++

- Comments tell other people what your code does
- Comments tell yourself what your code does
  - Or at least what it is supposed to do
- Code can be hard to understand
- Examples
  - `// a single-line comment`

  -
    ```
    /*
     A multi-line
     comment
    */
    ```

  -
    ```
    /* … */
    … *
    … */
    ```
    this is wrong



Integer literals in C++

- `100` `// int decimal`
- `123456` `// int decimal`
- `5L` `// long, decimal`
- `123u` `// unsigned int, decimal`
- `777uL` `// unsigned long, decimal`
- `-02O` `// int, octal`
- `0x1fff` `// int, hexadecimal`
- `0x1ffful` `// unsigned long, hexadecimal`

Character literals in C++

`'A'`	`// character A`
`'*'`	`// symbol *`
`'\0'`	`// end of a string`
`'\n'`	`// new line`
`'\t'`	`// tabulator`
`'\''`	`// apostrophe`
`'\\'`	`// backslash`

String literals in C++

`"This is a string literal!"`	`// a string literal`

- More on strings later

Floating-point literals in C++

`-9.876`	`// double`
`123.456E-7`	`// double`
`1e12`	`// double`
`.001`	`// double`
`1.23f`	`// float`
`1.23L`	`// long double`

Defining variables in C++

- Variables have a
  - Type
  - Name
  - Optional: an initial value

```
int i = 42;
int j;
int k = 10, l = 42, m;
double d = 1;
double e;
double f = 1.23456;
float g = 12.5f;
float h = 42.13;
char c = 'A';
char c[] = "A string";      // later on
char *c = "Another string"; // later on
char x = -10;
unsigned int ui = 123;
unsigned int huge = -13; // DON'T!!!
```

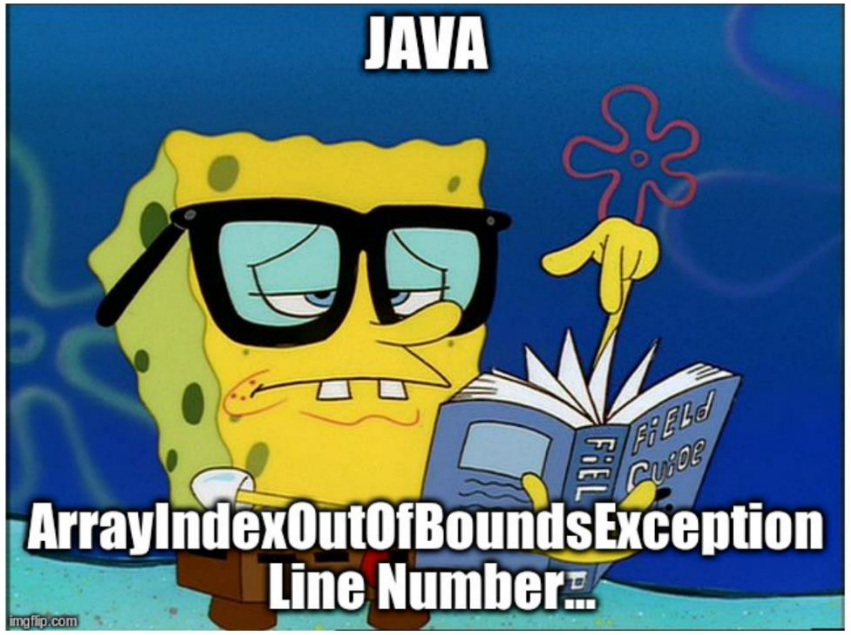
```
int main() {
    // see left side
    return 0;
}
```

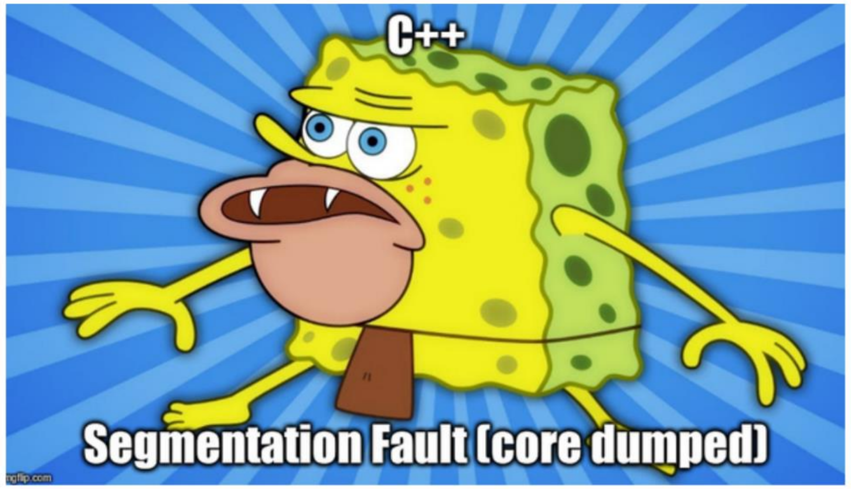
- Initialize your variables, unless you know what you are doing!

Variables in C++

- `unsigned int huge = -13;      // DON'T!!!`
  - Dangerous
  - Integer overflow

- C++ is famous for its undefined behavior
  - C++ standard allows undefined behavior in some situations
    ```
    int i;
    int j = i + 42;
    ```
  - Anything can happen
  - Depends on the compiler’s implementation
  - Why?
    - Compilers can produce faster machine code when assuming that certain things cannot happen





[Figure taken from https://www.reddit.com/r/ProgrammerHumor/comments/8p54sk/reporting_errors/]

Variables in C++

- auto keyword
  - Automatic type deduction
  - Compiler finds the correct type
  - Always be verbose
    - If type name gets 'too long' or type is obvious use auto
- What type is x?
  - `auto x = 13L;       // long`
  - `auto x = 1.2345;    // double`

```
#include <vector>
// C++98 style ☹
std::vector<int> v;
v.push_back(1);
v.push_back(2);
v.push_back(3);
for (std::vector<int>::iterator it =
        v.begin(); it != v.end(); ++it) {
    std::cout << *it << '\n';
}

// using modern C++
std::vector<int> w = {1, 2, 3};
for (auto i : w) {
    std::cout << i << '\n';
}
```

Making a point: there are ~50 ways to initialize a simple integer

- `int a = 1;`
- `int b(2);`
- `int c{3};`
- `int d = {4};`

- `auto i = 5;`
- `auto j(6);`
- `auto k{7};`
- `auto l = {8};`

IO streams

- `#include <iostream>`
  - Part of the STL
  - Content lives in namespace `std`
  - Use `std::`
  - Important variables
    - `cin` standard input stream
    - `cout` standard output stream
    - `cerr` standard error stream
    - `clog` general information
    - `<<` and `>>` are shift operators defined (i.e., overloaded) on the stream variables

- Example

```
#include <iostream>

int main() {
    int i = 0;
    std::cout<< "Enter an integer: ";
    std::cin >> i;
    std::cout<< "The value of i is: "
             << i << '\n';
    return 0;
}
```

Recap

- Course outline
- What is C++?
- History of C++
- Compilers
- “Hello, World!”
- Built-in types
- Information encoding
- Variables
- IO streams

- Any questions?

And now?

- Quick demo: the development environment and how to write a “Hello, World!” program
  1. Visual Studio Code
  2. How to get a C++ job?

Thank you for your attention

Questions?