

Chapters 1 & 2 Programming and Programs

Bjarne Stroustrup

www.stroustrup.com/Programming

Abstract

Today, we'll outline the aims for this course and present a rough course plan. We'll introduce the basic notion of programming and give examples of areas in which software is critical to our civilization. Finally, we'll present the simplest possible C++ program and outline how it can be made into running code.

Overview

- Course aims and outline
- Programming
- "Hello, world!"
- Compilation

This is a course

- In Programming
- For beginners
 - who want to become professionals
 - i.e., people who can produce systems that others will use
 - who are assumed to be bright
 - Though not (necessarily) geniuses
 - who are willing to work hard
 - Though do need sleep occasionally, and take a normal course load
- Using the C++ programming language

Not!

- A Washout course
 - “If you can get into the science/engineering parts of a university, you can handle this course”
- A course in
 - The C++ programming language
- For students
 - who want to become language lawyers
 - We try not to get bogged down in technical obscurities
 - who are assumed to be a bit dim and fairly lazy
 - We try not to spoon feed
- Using
 - Some untested software development methodologies and a lot of unnecessarily long words

The Aims

- Teach/learn
 - Fundamental programming concepts
 - Key useful techniques
 - Basic Standard C++ facilities
- After the course, you'll be able to
 - Write small colloquial C++ programs
 - Read much larger programs
 - Learn the basics of many other languages by yourself
 - Proceed with an “advanced” C++ programming course
- After the course, you will ***not*** (yet) be
 - An expert programmer
 - A C++ language expert
 - An expert user of advanced libraries

The Means

- Lectures
 - Attend every one
- Notes/Chapters
 - Read a chapter ahead (about one per lecture)
 - Read the chapter again after each lecture
 - Feedback is welcome (typos, suggestions, etc.)
- Do the assigned work!
 - Projects
 - That's where the most fun and the best learning takes place
 - Homework
 - Exams

Why C++ ?

- You can't learn to program without a programming language
- The purpose of a programming language is to allow you to express your ideas in code
- C++ is the language that most directly allows you to express ideas from the largest number of application areas
- C++ is the most widely used language in engineering areas
 - <http://www.stroustrup.com/applications.html>

Why C++ ?

- C++ is precisely and comprehensively defined by an ISO standard
 - And that standard is almost universally accepted
 - The most recent standard in ISO C++ 2014
- C++ is available on almost all kinds of computers
- Programming concepts that you learn using C++ can be used fairly directly in other languages
 - Including C, Java, C#, and (less directly) Fortran

Rough course outline

- Part I: The basics
 - Types, variables, strings, console I/O, computations, errors, vectors functions, source files, classes
- Part II: Input and Output
 - File I/O, I/O streams
 - Graphical output
 - Graphical User Interface
- Part III: Data structures and algorithms
 - Free store, pointers, and arrays
 - Lists, maps, sorting and searching, vectors, templates
 - The STL
- Part IV: Broadening the view
 - Software ideals and history
 - Text processing, numerics, embedded systems programming, testing, C, etc.

Rough course outline (Cont.)



- Throughout
 - Program design and development techniques
 - C++ language features
 - Background and related fields, topics, and languages
- Note: Appendices
 - C++ language summary
 - C++ standard library summary
 - Index (extensive)
 - Glossary (short)

Promises

- ***Detail:*** We will try to explain every construct used in this course in sufficient detail for real understanding
 - There is no “magic”
- ***Utility:*** We will try to explain only useful concepts, constructs, and techniques
 - We will not try to explain every obscure detail
- ***Completeness:*** The concepts, constructs, and techniques can be used in combination to construct useful programs
 - There are, of course, many useful concepts, constructs, and techniques beyond what is taught here

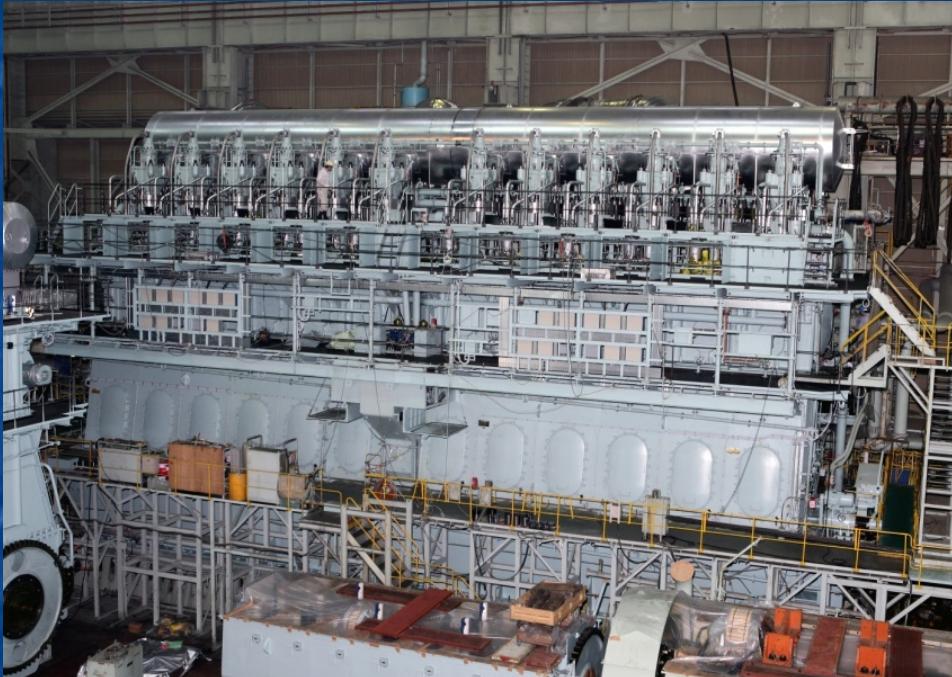
More Promises

- **Realism:** The concepts, constructs, and techniques can be used to build “industrial strength” programs
 - i.e., they have been used to ...
- **Simplicity:** The examples used are among the simplest realistic ones that illustrate the concepts, constructs, and techniques
 - Your exercises and projects will provide more complex examples
- **Scalability:** The concepts, constructs, and techniques can be used to construct large, reliable, and efficient programs
 - i.e., they have been used to ...

Why programming?

- Our civilization runs on software
 - Most engineering activities involve software
- Note: most programs do not run on things that look like a PC
 - a screen, a keyboard, a box under the table

Ships



- Design
- Construction
- Management

- Monitoring
- Engine
- Hull design
- Pumps

Aircraft



- Communication
- Control
- Display

- Signal processing
- “Gadget” control
- Monitoring

Phones



- Voice quality
- User interfaces
- Billing
- Mobility
- Switching
- Reliability
- Provisioning
- Images

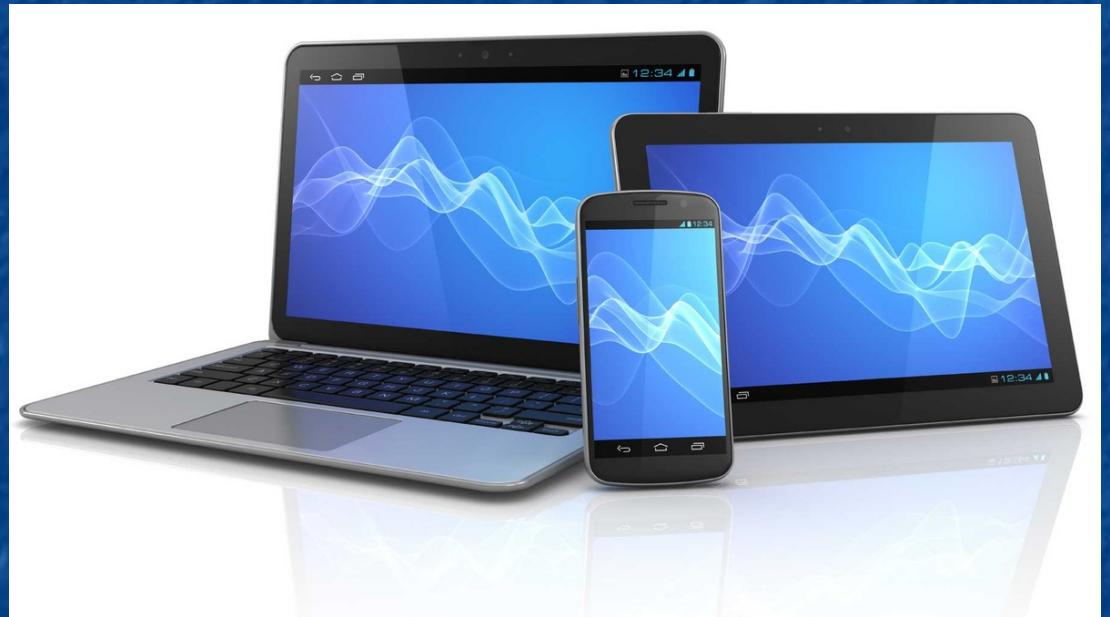
Energy



- Control
- Monitoring
- Analysis
- Design

- Communications
- Visualization
- Manufacturing

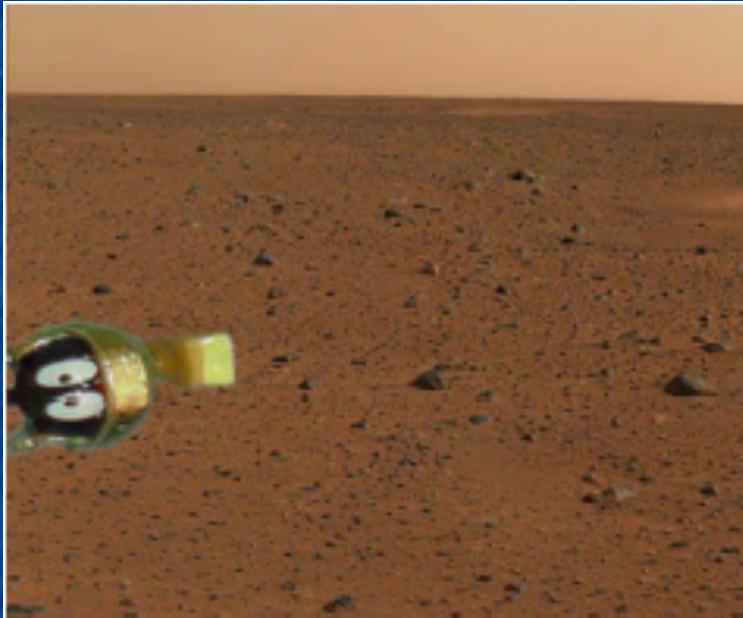
PC/tablet/workstation



- There's a lot more to computing than games, word processing, browsing, and spreadsheets!

Where is C++ Used?

- Just about everywhere



Mars rovers, animation, graphics, Photoshop, GUI, OS, compilers, slides, chip design, chip manufacturing, semiconductor tools, etc.

See www.stroustrup.com/applications.html

A first program – just the guts...

```
// ...
```

```
int main()                                // main() is where a C++ program starts
{
    cout << "Hello, world!\n";               // output the 13 characters Hello, world!
                                            // followed by a new line
    return 0;                               // return a value indicating success
}
```

```
// quotes delimit a string literal
// NOTE: “smart” quotes “ ” will cause compiler problems.
//         so make sure your quotes are of the style "
// \n is a notation for a new line
```

A first program – complete

```
// a first program:  
#include <iostream>           // get the library facilities needed for now  
  
using namespace std;  
  
int main()                  // main() is where a C++ program starts  
{  
    cout << "Hello, world!\n";   // output the 13 characters Hello, world!  
                                // followed by a new line  
    return 0;                  // return a value indicating success  
}  
  
// note the semicolons; they terminate statements  
// braces { ... } group statements into a block  
// main( ) is a function that takes no arguments ()  
// and returns an int (integer value) to indicate success or failure
```

Hello, world!

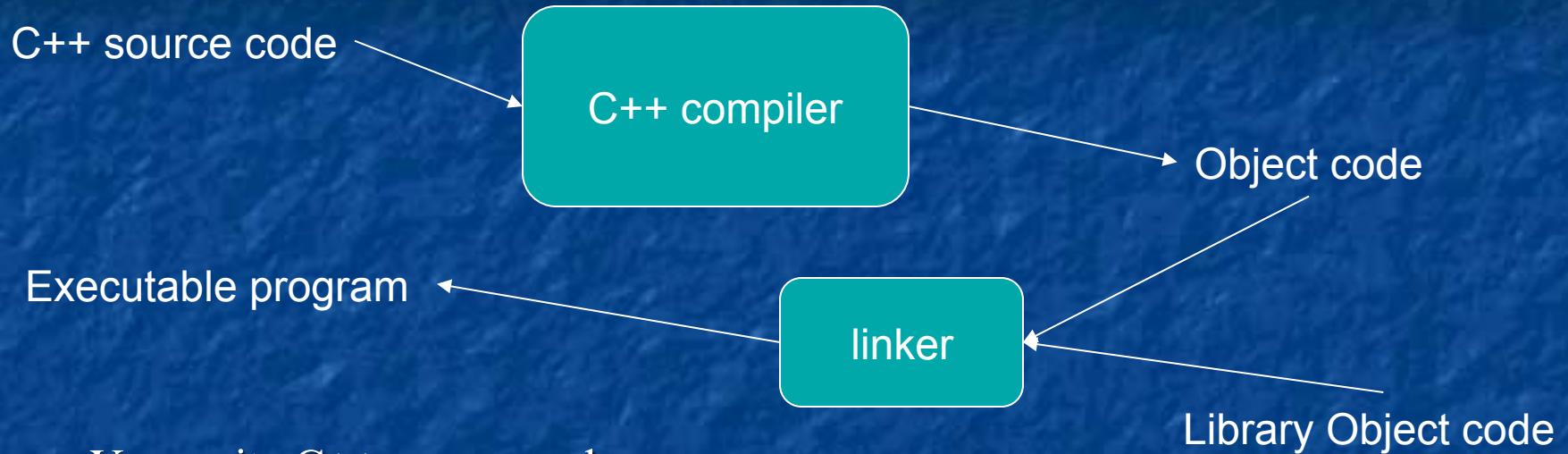
- “Hello world” is a very important program
 - Its purpose is to help you get used to your tools
 - Compiler
 - Program development environment
 - Program execution environment
 - Type in the program **carefully**
 - After you get it to work, please make a few mistakes to see how the tools respond; for example
 - Forget the header
 - Forget to terminate the string
 - Misspell **return** (e.g., **retrun**)
 - Forget a semicolon
 - Forget { or }
 - ...

Hello world

- It's almost all "boiler plate"
 - Only `cout << "Hello, world!\n"` directly does anything
- That's normal
 - Most of our code, and most of the systems we use simply exist to make some other code elegant and/or efficient
 - "real world" non-software analogies abound
- "Boiler plate," that is, notation, libraries, and other support is what makes our code simple, comprehensible, trustworthy, and efficient.
 - Would you rather write 1,000,000 lines of machine code?
- This implies that we should *not* just "get things done"; we should take great care that things are done elegantly, correctly, and in ways that ease the creation of more/other software:

Style Matters!

Compilation and linking



- You write C++ source code
 - Source code is (in principle) human readable
- The compiler translates what you wrote into object code (sometimes called machine code)
 - Object code is simple enough for a computer to “understand”
- The linker links your code to system code needed to execute
 - E.g., input/output libraries, operating system code, and windowing code
- The result is an executable program
 - E.g., a **.exe** file on windows or an **a.out** file on Unix

So what is programming?

- Conventional definitions
 - Telling a **very** fast moron *exactly* what to do
 - A plan for solving a problem on a computer
 - Specifying the order of a program execution
 - But modern programs often involve millions of lines of code
 - And manipulation of data is central
- Definition from another domain (academia)
 - A ... program is an organized and directed accumulation of resources to accomplish specific ... objectives ...
 - Good, but no mention of actually doing anything
- The definition we'll use
 - Specifying the structure and behavior of a program, and testing that the program performs its task correctly and with acceptable performance
 - Never forget to check that "it" works
- Software == one or more programs

Programming

- Programming is fundamentally simple
 - Just state what the machine is to do
- So why is programming hard?
 - We want “the machine” to do complex things
 - And computers are nitpicking, unforgiving, dumb beasts
 - The world is more complex than we’d like to believe
 - So we don’t always know the implications of what we want
 - “Programming is understanding”
 - When you can program a task, you understand it
 - When you program, you spend significant time trying to understand the task you want to automate
 - Programming is part practical, part theory
 - If you are just practical, you produce non-scalable unmaintainable hacks
 - If you are just theoretical, you produce toys

The next lecture

- Will talk about types, values, variables, declarations, simple input and output, very simple computations, and type safety.