Abstract

• Today, we’ll outline the aims for this course and present a rough course plan. After some introduction, we’ll start reviewing C++ by looking at the simplest possible C++ program and outline how it can be made into running code.
Techniques for managing complexity

- Black box abstraction
  - Primitive objects
    - Primitive procedures and primitive data
  - Means of combination
    - Procedure composition and compound data
  - Means of abstraction
    - Procedure (function) definition
    - Simple data abstraction
- Capturing common patterns
  - Higher order functions
  - Data as procedures, procedures as data
Techniques for managing complexity

- Conventional Interfaces
  - Generic operations
  - Large scale structure and modularity
  - Object oriented programming
  - Operations on aggregates
Techniques for managing complexity

- Meta-Linguistic Abstraction
  - Making new (domain specific) languages
Admin & Organizational

• Congrats
  • Why I like programming

• Course:
  • Depth first introduction, C++ Standard Library, C++ Data structures and algorithms
  • [http://www.cct.lsu.edu/~hkaiser/spring_2015/csc1254.html](http://www.cct.lsu.edu/~hkaiser/spring_2015/csc1254.html)
  • csc1254@cct.lsu.edu
  • hkaiser@cct.lsu.edu

• Reading:
  • Koenig’s Accelerated C++
  • Stroustrup’s Programming - Principles and Practice Using C++
• Homework, project, quizzes, grading, honesty
  • Server: classes.csc.lsu.edu, accounts & passwords
Honesty

- The LSU *Code of Student Conduct* defines plagiarism in Section 5.1.16:

  - "Plagiarism is defined as the unacknowledged inclusion of someone else's words, structure, ideas, or data. When a student submits work as his/her own that includes the words, structure, ideas, or data of others, the source of this information must be acknowledged through complete, accurate, and specific references, and, if verbatim statements are included, through quotation marks as well. Failure to identify any source (including interviews, surveys, etc.), published in any medium (including on the internet) or unpublished, from which words, structure, ideas, or data have been taken, constitutes plagiarism;"

- Plagiarism will not be tolerated and will be dealt with in accordance with and as outlined by the LSU Code of Student Conduct:  
Course Overview

• Programs == Algorithms + Data structures
Rough course outline

- Books:
  - Andrew Koenig, Barbara Moo, Accelerated C++ • Practical Programming by Example, ISBN 0-201-70353-X
  - Stanley Lippmann, Josée Lajoie, Barbara E. Moo • C++ Primer 5th Ed., ISBN 0321714113

- Part I: The basics
  - Working with strings
  - Looping and counting, working with batches

- Part II: Organizing Programs and Data
  - Sequential data containers
  - Using library algorithms
  - Associative data containers

- Part III: Defining new types
  - Managing memory and low level data structures
  - Defining abstract data types

- Part IV: Object oriented programming
  - Inheritance and dynamic binding
  - Automatic memory management
Rough course outline

• 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)
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.
Why C++?

- C++ is precisely and comprehensively defined by an ISO standard
  - C++11! Now also C++14!
  - And that standard is almost universally accepted
- 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
A First C++ Program

// Our first C++ program

#include <iostream>

int main() // main() is where a C++ program starts
{
    std::cout << "Hello, world!" << std::endl;
    // 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 deeper look

- **Expressions**
  - Compute something, yields result, may have side effects
  - Operands and operators – types!

  ```
  std::cout << "Hello, world!" << std::endl;
  ```

- **Scope**
  - Part of the program in which a name has its meaning
    - Namespace scope
    - Block scope
Details

• Program structure
  • Free form except string literals, #include, comments

• Types
  • Data structures and their operations
  • Built in and user defined

• Namespaces
  • Grouping related names
Details

• Special character literals
  • \n   newline character
  • \t   horizontal tabulator
  • \b   backspace character
  • \"   same as " but does not terminate string
  • \'   same as ' but does not terminate character literal
  • \\   same as \ but does not give special meaning to next character

• Definitions and headers

• The main() function

• Braces and semicolons

• Output
A First C++11 Program

// Our first C++11 program

#include <iostream>

auto main() -> int // main() is where a C++ program starts
{
    std::cout << "Hello, world!" << std::endl;
    // 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
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 std::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

C++ source code

C++ compiler

Object code

Library Object code

Linker

Executable program
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
    - Analysis, design, programming (coding), testing

- 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