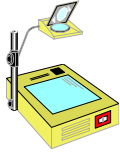




Object-Oriented Software Construction with C++



Introduction

Topics

- Introduction
- Basics
- Classes
- The SDL API
- Building a GUI w/ SDL
- Standard C++ Library, STL
 - Inheritance
 - Polymorphism
 - Templates
 - Specialization
 - Metaprogramming
 - typelists
 - Policy classes
- Design Patterns
 - Singleton
 - Command
 - Strategy
 - Observer
 - Factory
 - Visitor Pattern
 - decorator
- Memory Management
- Reference Counting
- CGI Programming
- Threads

Course Objectives

- Learn the correct use of C++
- Learn Object-Oriented Programming
- Learn the Standard C++ Library
- Learn Design Patterns
- Learn Generic Programming
- Learn how to build/program a GUI
- Learn scripting: CGI, JavaScript, Python, threads
- Learn how to be a good programmer

Starting Point

I know no C++

Starting Point

I know no C++

I know a little C++

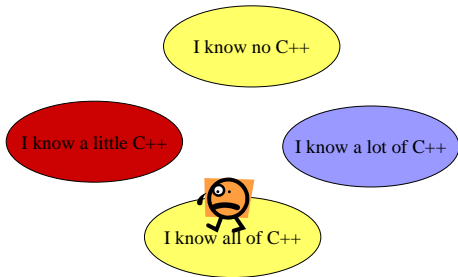
Starting Point

I know no C++

I know a little C++

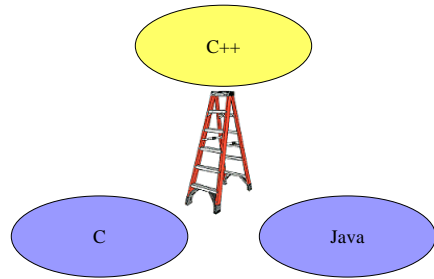
I know a lot of C++

Starting Point



7

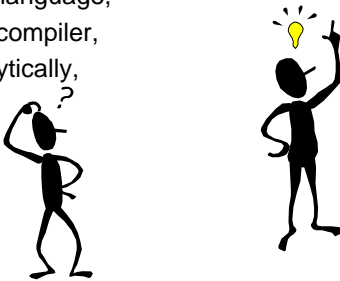
Languages & Paradigms



8

Learn to be a good programmer:

- Know the language,
- know the compiler,
- think analytically,
- work hard



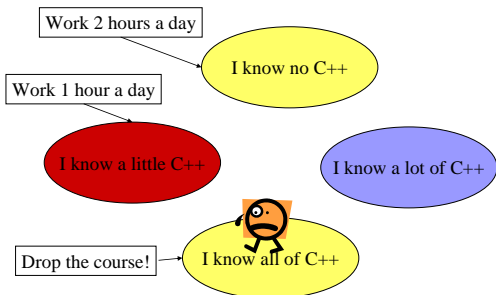
9

Programming takes practice



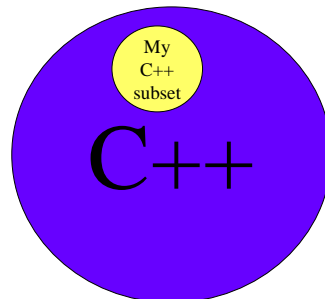
10

Write code/Compile/Execute!



11

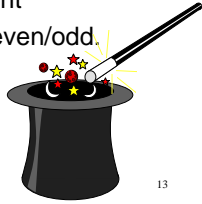
Practitioners use a subset



12

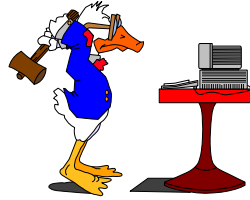
Programming: tricks help

- Swapping 2 numbers,
- find the largest number in a list,
- convert a char digit to an int
- determine if a number is even/odd.



13

Programming can be frustrating



**If you've never been here,
then you ain't programmed!**

14

Resources

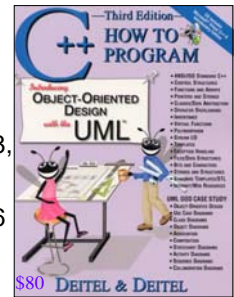
- Lecture
 - presentations
 - **demonstrations**
 - discussion
- graded assignments
- exams
- The WWW, and
- Textbooks.



15

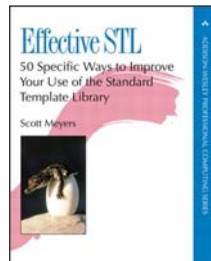
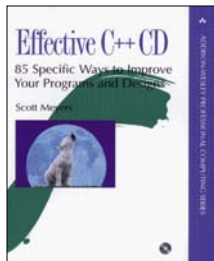
Some will need an intro text

C++ How to Program
Deitel & Deitel,
Second Edition,
Prentice Hall, 1998,
1000 pages,
ISBN: 0135289106



16

The Meyers Texts

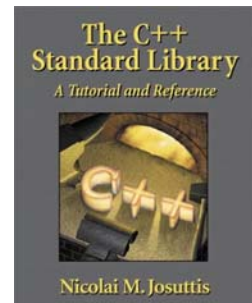


17

STL text

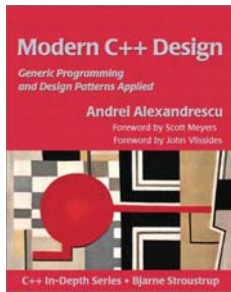
The Standard Template
Library

Addison-Wesley



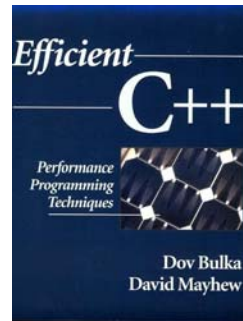
18

Alexandrescu: Generic Programming



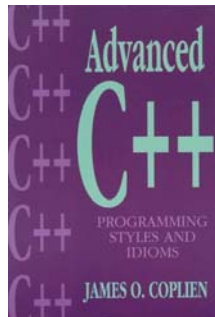
19

C++ Speed



20

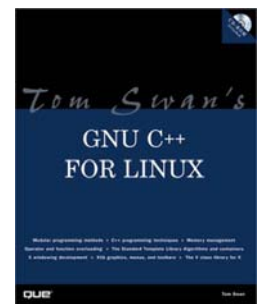
Coplien: Idioms



21

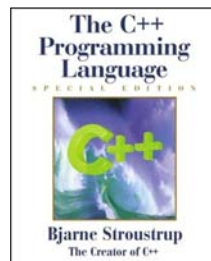
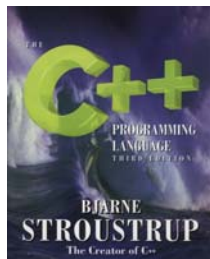
Intro text

GNU C++ for Linux
Tom Swan
QUE: a division of
Macmillan, USA
ISBN: 0-7897-2153-8



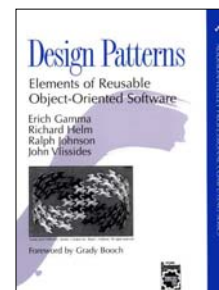
22

Stroustrup 3rd Edition



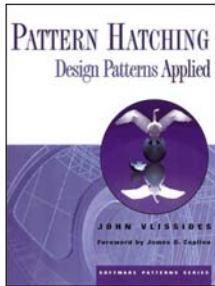
23

Design Patterns: GoF



24

One of the “gang”



25

3 Kinds of Programmers

- Those who can't
- those who can without pointers
- those who can with pointers

<http://www.joelonsoftware.com/articles/fog0000000006.html>

26

5 Kinds of Programmers

- Those who can't
- those who can without pointers
- those who can with pointers
- those who know object technology: OO
- those who can use generics/templates

27

Compiler

- g++ for unix or mingwin32 for Win95
 - GNU foundation
 - it's free
 - close to the standard
 - fairly complete
- others
 - VC++ 7.1, and .net
 - Borland
 - See my web page



28

Syllabus: **Two phases**

- **Phase 1**
 - C++
 - **Phase 2**
 - OO
 - GUIs
 - Design Patterns
 - Generic Programming
 - CGI programming
-

29

Motivation for OO with C++

- C++ most widely used language,
- OO is the way of the present,
- familiarity increases marketability,
- viable for many years,
- excellent expressivity, and
- excellent speed.
- Easy to learn another language.



30

Why GUI's and CGI

- GUIs:
 - Everybody's doin' it!
 - GUIs make it look like you did something
 - amazing: few students have used a GUI library!
- CGI:
 - Study of CGI exposes some interesting low level features of www
 - CGI mixes nicely with C++, OO and ...

31

Summary

- This course is an unusual opportunity
- May require tremendous work, study and dedication
 - Programming requires analytical skills
 - may have to learn new language, paradigm or skills

32

Warning

- If you came into the course not knowing any C++:
 - You will need to work hard
 - You will need to practice a lot
 - you will leave knowing a lot of C++, but still needing lots of practice!

33

Warning

- This course is for people who want to know C++ and OO well
- This course is about learning to write large programs that are easier to read, easier to extend and modify
- Really "hard core" C programmers usually do not like/need this course
- Please consider dropping to let others, who are "open" to OO, take the course

34

Lots of "stuff"

- There is an intimidating amount of C++ stuff
- C++ was designed to be a powerful tool for professional programmers solving real problems in diverse domains
- C++ was NOT designed for academia!
- C++ was NOT designed to be a nice, "pure" language, good for teaching students how to program

35

How well do you want to know C++



36

Design Goals & Rationale

- Compatibility with C
- Efficiency
- Compatibility with traditional tools and environments

37

Examples

- **Why do implicitly generated copy constructors implement shallow copy?** Because that's how C copies and assigns structs!
- **Why aren't destructors automatically virtual?** Because it would impose a performance penalty for classes that never have derived classes.
- **Why can implementation details appear in class definitions?** For efficiency: auto inline.

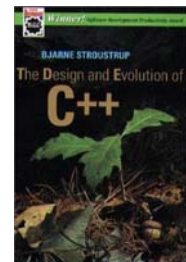
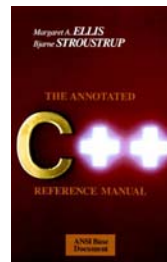
38

More examples

- **Why can't C++ detect initialization dependencies between non-local static objects?** Because C++ supports separate translation: the ability to compile source modules separately, then link several object files together to form an executable.
- **Why doesn't C++ free programmers from tiresome duties like memory management and low level pointer manipulation?** Because some programmers need these capabilities.

39

Design Goals & Rationale Explained



40