# Welcome

## Object Oriented Programming with C++
## CIS 265

Week 1 – First Class

Christopher K. Burns

# Agenda

Tonight's agenda

- Introduction
- Review Syllabus
- Text book – Chapter 1 – 5
- Visual C++ 6.0, and .NET 2003 Tutorial

# Administrative Info

## Important Dates:

2005 Academic Calendar

| Winter Quarter | |
|---|---|
| Winter Quarter Begins | Monday, Jan. 10 |
| Add/Drop Period | Monday, Jan. 10 – Tuesday, Jan. 18 |
| Martin Luther King, Jr. Birthday (University Closed) | Monday, Jan. 17 |
| Presidents' Birthday (University Closed) | Monday, Feb. 21 |
| Easter (University Closed) | Friday–Sunday, March 25-27 |
| Winter Quarter Ends | Monday, Mar. 28 |

## Important Phone Numbers:

Snow Closings and other info:

http://studentserver.strayer.edu/campus_links.htm

**1 800 909 9211**

# Introduction

**CIS 265**
**Object-Oriented Programming I**
*Prerequisite: CIS 242 or equivalent*
Covers the traditional C language and object-oriented extensions that are found in the C++ language. Describes concepts of objects, encapsulation, data hiding, polymorphism, and inheritance as well as the C++ techniques that implement them.

# Introduction

History of C++

The first object oriented programming languages were SIMULA I (1962) and SIMULA 67, developed by Ole-Johan Dahl and Kristen Nygaard. They supported the basic concepts of objects as we know them: classes, inheritance, and virtual procedures. Although originally designed for simulation work with UNIVAC, the language quickly found other applications and quickly gained popularity. SIMULA 67 is still used in many places around the world.



http://heim.ifi.uio.no/~kristen/FORSKNINGSDOK_MAPPE/F_OO_start.html

# History of C++

*{around the same time}*
BCPL was created by Martin Richards while he was visiting MIT in the mid 1960s. BCPL was influenced by several languages of the time including CPL and PL/I (where the // style comments came from). BCPL stands for Basic Combined Programming Language.

Steve Johnson, Brian Kernighan and Ken Thompson of Bell Labs developed the B programming language in the late 60s. B was based on BCPL and fixed several technological. B was primarily used for experimentation as its early compilers did not create machine code; thus too slow for most utilities.

Legend says that the name B is either from Ken Thompson's **B**on language, or **B**CPL.



http://www.cs.bell-labs.com/who/dmr/

# History of C++

Around 1970, the Unix project had become a major force at Bell Labs. The B language was desirable but had several fatal flaws that would not let it port to the new machines. Dennis Ritchie began working on ”New B” or NB. NB kept many of the same semantics that existed in BCPL and B. When the design of NB made a critical leap with the invention of a true pointer, it became C, and C was good.

# History of C++

With the advent of Unix, C quickly became popular as a general purpose language. The early OO programming languages had their purpose and were admired for what they could do, but they were not being used for mainstream development.

In the mid 80s, Bjarne Stroustrup wanted to create a language that had the efficiency of C but also had the elegance of SIMULA67. This language became “C with Classes”.

It was later renamed C++. C++ was standardized by ISO on November 14, 1997 (ISO/IEC 14882)

# Future of C++

C++ was the language of choice for all applications and quickly pushed C off of the throne. C++'s reign lasted several years, from the late 80's until the mid-90's.

Java and, more recently C#, have challenged C++ supremacy. Many RAD or end user apps are being written in these now. Both Java and C# improved upon the C++ by removing complicated code syntax, and garbage collection.

The net result of these languages is that programmers tend to make less mistakes than with C++, and can write code more quickly. Also, code written in the languages is less efficient C++.

# Future of C++

Will C++ ever go away?

Unlikely, too many lines of code have been written in C and C++ for it to just go away.

Perhaps, if there was a language with the ease of Java and the power of C#, that was code compatible with C++…

Once there was B

Then it was improved by… C

C++

Java

C#

What could be next down the road?

D

# Future of programming language

The D programming language was created by Walter Bright to bring the advantages of Java and C# to C++.

D promises to be code and binary compatible with C++, thus like C programs could be compiled in C++, C++ programs could be compiled in D.

This is a great idea, but it is still under development and its future remains to be seen.  D does not have a Bell Labs, or a Sun, or a Microsoft behind; but this is the age of open source...

# C++ Today

C++ is still an important language, and will likely remain so for some time.

Becoming a good C++ programmer takes work, but it is worth it. You'll often hear senior developers saying, if you can handle C++ then you can handle any imperative language. Good C++ programmers are also hard to find…

# Syllabus

**CIS 265**
**Object Oriented Programming with C++**
Tuesday 6:30pm – 10:15pm @Fredericksburg 109

**Christopher Burns**
Office Hours: 9am – 1pm Saturdays
Email Address: ckburns@cox.net

**COURSE DESCRIPTION:**
This course covers the basics of the C++ language and object-oriented programming detail that are found in the C++ language. It describes the dynamic, interactive instructions and C++ templates, concepts of objects, encapsulation, data hiding, polymorphism, and inheritance, as well as the C++ techniques that implement them.

# Syllabus

**EXPECTED LEARNING OUTCOMES:**
**Upon the successful completion of this course, the student will be able to:**

- know the basic concepts in OOP and its implementation with C++;
- know the syntax of C++; and
- be able to write, debug, and document programs using the recommended structured approach.

# Syllabus

**INSTRUCTIONAL MATERIALS:**

*Required Text Resources*

Deitel, Harry and Paul. C++ How to Program. 4th Ed, Prentice Hall, 2003.

*Optional Text Resources (not required)*

Stroustrup, Bjarne. The C++ Programming Language. 3rd Ed, Addison-Wesley, 2001.

Myers, Scott. Effective C++. 2nd Ed, Addison-Wesley, 1998.

*Web Resources*

Homepage for Deitel text: http://www.deitel.com/books/cppHTP4/

Slides and examples for Deitel text: http://www.deitel.com/books/downloads.html

Getting Started with Visual C++: http://www.deitel.com/books/cppHTP4/DiveIntoVCPP6.zip

Microsoft Developer's Network (MSDN): http://msdn.microsoft.com/visualc/

# Syllabus

**TEACHING STRATEGIES:**

The course will be conducted with lectures based on the text and hands-on applications concepts. The professor will provide additional course-work if and when necessary. Students will be required to design the solution to a minimum of eight programming problems.

# Syllabus

| Week | | Content |
|---|---|---|
| 1 | 1/11 | *Review*<br>Chapter 1 Intro to Computers and C++ Programming<br>Chapter 2 Control Structures<br>Chapter 3 Functions<br>Chapter 4 Arrays |
| 2 | 1/18 | Additional Array and Function Topics<br>Chapter 5 Pointers and Strings<br>Lab 1 – Functions, Arrays, and Strings<br>Homework 1 Assigned |
| 3 | 1/25 | Chapter 6 Classes and Data Abstraction<br>Homework 1 Due |
| 4 | 2/1 | Chapter 7 Classes: Part I<br>Lab 2 – Classes 1 |
| 5 | 2/8 | ***MID-TERM EXAMINATION*** *(Chapters 1 through 7*)*<br>**only portions of chapter 7 that were covered in class*<br>*Final Project Assigned* |
| 6 | 2/15 | Chapter 7 Classes: Part II<br>Chapter 8 Operator Overloading<br>Homework 2 Due |
| 7 | 2/22 | Chapter 8 Operator Overloading<br>Lab 3 – Classes 2 |
| 8 | 3/1 | Chapter 9 Inheritance: Part I<br>Homework 3 Due |
| 9 | 3/8 | Chapter 9 Inheritance: Part II<br>Chapter 10 Polymorphism<br>Lab 4 – Inheritance |
| 10 | 3/15 | Chapter 10 Polymorphism |
| 11 | 3/22 | ***FINAL PROJECTS DUE*** |

# Syllabus

**COURSE REQUIREMENTS:**

1. Mid-term examination
2. Final Project
3. Completion of all text assignments and a minimum of 8 programs
   (4 Labs, 3 Homeworks, 1 Project)
4. Active class participation
5. Regular class attendance

# Syllabus

**EVALUATION METHODS:**

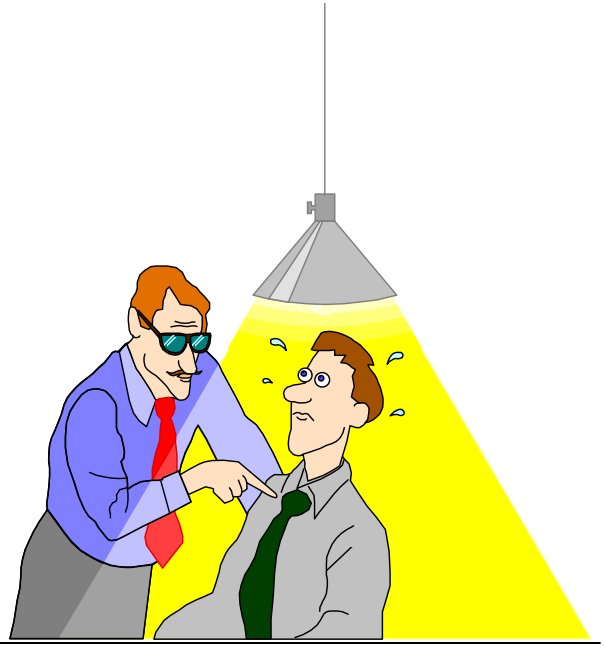
| | |
|---|---|
| Lab Assignments | 20% |
| Homework | 20% |
| Midterm Exam | 25% |
| Final Project | 35% |

**GRADING SCALE:**

| | |
|---|---|
| 90-100 % | A |
| 80-89 % | B |
| 70-79 % | C |
| 60-69 % | D |
| Below 60 % | F |

# C++ Review

Questions?



# Object Oriented Programming

## Home Work

*Review Chapters 1-5* – You will need to be able to apply this material in order to successfully complete the upcoming labs and homework. You will also be responsible for this material on the mid-term.

*Read Chapter 6*