

Department of Systems Design Engineering - University of Waterloo
SY DE 121 – Digital Computation
Prof. D.A. Clausi

DETAILED COURSE OUTLINE

Section 1 – Introduction

- 1.1 Computing Basics
- 1.2 Computer History
- 1.3 Computer Systems
- 1.4 Algorithms

Section 2 – Introductory C++ Concepts

- 2.1 A Demo Program
- 2.2 Variables
- 2.3 Expressions
- 2.4 Input/Output
- 2.5 Programming Errors
- 2.6 Programming Style (refer to Style Guide)

Example implementation: marks2.cpp

Section 3 – Decision Structures

- 3.1 'if ... else'
 - 3.1.1 Logical Operators
 - 3.1.2 Examples
- 3.2 Logical Operators
 - 3.2.1 and/or Expressions
 - 3.2.2 Multiple Expressions
 - 3.2.3 Short-Circuit Evaluation
 - 3.2.4 ! (not) Expressions
- 3.3 Advanced Decisions
 - 3.3.1 Multi-branching
 - 3.3.2 Nested Decisions
 - 3.3.3 'switch' Statement
- 3.4 Enumeration Types
- 3.5 Ternary Operator
- 3.6 Precedence Summary

Example implementation: marks3.cpp

Section 4 – Iteration Structures

- 4.1 'while' Loop
 - 4.1.1 Example
 - 4.1.2 Syntax
- 4.2 'do-while' Loop
- 4.3 'for' Looping
 - 4.3.1 Introductory Example
 - 4.3.2 Syntax
 - 4.3.3 Additional Examples
 - 4.3.4 Comma Operator
- 4.4 Prefix/Postfix
- 4.5 Nested Loops
- 4.6 'break' Command
- 4.7 Loop Design
 - 4.7.1 Summary of Iteration Structures
 - 4.7.2 Off-by-one Errors
 - 4.7.3 Types of Data Input Loops
 - 4.7.4 Infinite Loops

Example implementation: marks4.cpp

Section 5 – Pointers and Functions

- 5.1 Predefined Functions
- 5.2 Pointers
 - 5.2.1 Pointer Definition
 - 5.2.2 Pointer Variable Declaration
 - 5.2.3 Pointer Operations
- 5.3 Programmer Defined Functions
 - 5.3.1 Single Value Returned
 - 5.3.2 No Value Returned
 - 5.3.3 Multiple Values Returned
- 5.4 Overloading Functions
- 5.5 Data Scope (Local Versus Global Variables)
- 5.6 Recursion & Iteration
- 5.7 Miscellaneous
 - 5.7.1 Inline Functions
 - 5.7.2 Default Arguments
 - 5.7.3 Use of 'const'

Example implementation: marks5.cpp

Midterm Examination

Section 6 – Structures

- 6.1 What are structures?
- 6.2 Passing and Returning Structs
- 6.3 Use of Same Member Names in Different Structures
- 6.4 Hierarchical Structs

Example implementation: structs.cpp

Section 7 – Namespaces, File Streams, Arrays, Strings

- 7.1 Standard Libraries and Namespaces
 - 7.1.1 Using Namespaces
 - 7.1.2 Creating Namespaces
- 7.2 Numerical Arrays
 - 7.2.1 Declarations
 - 7.2.2 Accessing
- 7.3 Character Arrays
 - 7.3.1 Declaring
 - 7.3.2 Using get/put/getline
 - 7.3.3 Predefined functions for c-strings
 - 7.3.4 'string' ANSI class
- 7.4 Arrays and Functions
 - 7.4.1 Passing Arrays Into Functions
 - 7.4.2 Use of 'const'
- 7.5 Applications
 - 7.5.1 Sorting
 - 7.5.2 Searching
- 7.6 Arrays and Structs
 - 7.6.1 Arrays of Structs
 - 7.6.2 Strings and Structs
 - 7.6.3 Arrays of Structs with Arrays
- 7.7 n-d Arrays
 - 7.7.1 Arrays of Strings
 - 7.7.2 Numerical Multi-dimensional Arrays

Example implementation: arraysandstructs.cpp, multiarrays.cpp

Section 8 – File Streams

- 8.1 File Streams
 - 8.1.1 Basics

- 8.1.2 Defining a Filestream
- 8.1.3 Checking a Filestream
- 8.1.3 Using an Input Filestream
- 8.1.4 Using an Output Filestream
- 8.1.5 Terminating a Filestream
- 8.2 I/O Stream Class
 - 8.2.1 I/O Stream Class Hierarchy
 - 8.2.2 Calling Member Functions Using an I/O Stream

Example implementation: filestreams.cpp

Section 9 – Introduction to Classes

- 9.1 Introduction to Classes
- 9.2 A Simple Class Example (Counter class)
 - 9.2.1 Class Definition
 - 9.2.2 Implementation of Counter Class
 - 9.2.3 Using Counter Class
- 9.3 Generating Constructors
 - 9.3.1 Three Methods for Generating Constructors
 - 9.3.2 Constructor Rules
- 9.4 Implicit Versus Explicit Objects
- 9.4 Other Class Design Considerations
 - 9.4.1 Protecting Object Data Members
 - 9.4.2 Class Destructor
 - 9.4.3 Summary of Good Programming Practices for Classes

Example implementation: Counter class

Section 10 – Advanced Class Implementations

- 10.1 Friend Functions
- 10.2 Operator Overloading
 - 10.2.1 Two Methods to Perform Operator Overloading
 - 10.2.2 Automatic Type Conversion
 - 10.2.3 Overloading '>>' and '<<'
- 10.3 Inheritance
 - 10.3.1 Inheritance Using Stream Classes
 - 10.3.2 Implementing Inheritance
- 10.4 Dynamic Memory Allocation
 - 10.4.1 typedef
 - 10.4.2 Dynamic Memory Usage
- 10.5 Classes and Dynamic Memory
 - 10.5.1 Destructors
 - 10.5.2 Copy Constructors
 - 10.5.3 Overloading '='
- 10.6 Templates
 - 10.6.1 Function Templates
 - 10.6.2 Class Templates
 - 10.6.3 <vector> class

Example implementation: Date class, Person class, Student class