Miami Dade College

Course Description

COP2335 | Object Oriented Programming Using C++ | 4.00 credits

This second course in C++ programming is recommended for Computer Science and Computer Information Systems majors. Students will learn techniques and skills of object-oriented programming including object-oriented modeling, analysis, and design. Prerequisite: COP1334. Knowledge of high school algebra is recommended.

Course Competencies:

Competency 1: The student will demonstrate knowledge of object-oriented programming (OOP) principles and implementation by:

1. Understanding the difference between classes and objects
2. Defining and manipulating classes and objects to model real-world entities and behaviors
3. Utilizing constructors to initialize objects efficiently, ensuring proper resource allocation and setup
4. Applying constructor overloading to offer multiple ways of initializing objects
5. Overloading various operators (e.g. math operators, insertion, and extraction operators, ++ operators, == and = operators, subscript, and parentheses operators) to improve program performance
6. Employing destructors to manage resource deallocation and cleanup
7. Defining, creating, implementing, and accessing members of a class
8. Understanding the purpose of header and implementation files
9. Implementing polymorphism and inheritance to improve program design and performance
10. Creating and using classes and subclasses with overriding and overloading constructors, methods, and class access specifiers
11. Using constructor initialization lists and dynamic and static binding
12. Declaring and using friend functions with data from different classes
13. Understanding the use of composition in class design

Competency 2: The student will demonstrate proficient use of arrays and vectors by:

1. Creating and using arrays and vectors
2. Using efficient techniques to manipulate the size and content of vectors
3. Using arrays and vectors as function arguments
4. Using a loop to modify, copy, compare, or search vectors
5. Explaining fundamental introductory vector class operations
6. Adding or removing elements into a vector
7. Storing objects of a class as values in a vector

Competency 3: The student will demonstrate knowledge of pointers, dynamic data, and reference types by:

1. Declaring and using pointers
2. Understanding pointer arithmetic and dereferencing
3. Accessing class members and dynamic data using pointers
4. Understanding the use of references in C++. e) Using the "this" pointer

Competency 4: The student will demonstrate an understanding of exception handling by:

1. Understanding the limitations of traditional error-handling methods
2. Throwing and catching exceptions
3. Implementing try-catch blocks to handle exceptions
4. Using multiple throw statements and catch blocks to handle exceptions
5. Implementing the stack unwinding concept and handling memory allocation exceptions
6. Declaring custom exception classes

Competency 5: The student will demonstrate an understanding of how to leverage the capabilities of the Standard Template Library (STL) by:

1. Understanding the basics of STL containers, iterators, and algorithms for efficient utilization of pre-packaged data structures and operations
2. Using container class templates with objects to store and organize data
3. Using iterators class templates to access individual data elements in a container

Competency 6: The student will demonstrate proficiency in using and manipulating linked lists by:

1. Using dynamically allocated data structures that are linked together in memory to form a sequential chain, showcasing the fundamental concept of linked lists
2. Applying fundamental linked list operations, including appending nodes to the end, inserting nodes at specific positions, and deleting nodes
3. Navigating through a linked list to access and modify data
4. Employing best practices for safely and efficiently managing the destruction and cleanup of a linked list to prevent memory leaks
5. Designing and implementing templates for linked lists to enable the creation of type-agnostic lists capable of storing any data type
6. Using linked lists as a foundation for other data structures, such as stacks and queues

Competency 7: The student will demonstrate an understanding of recursion by:

1. Understanding the principles of recursion and how it is used to break down complex problems into simpler sub-problems
2. Understanding the base and recursive cases in functions to ensure correct algorithm termination and to avoid stack overflow errors
3. Differentiating between situations that require recursive solutions versus those that benefit from iterative approaches
4. Combining recursive and iterative approaches to optimize algorithms
5. Using recursion within class methods to efficiently manage data structures like linked lists, enabling operations such as traversal, node counting, and displaying node data
6. Designing recursive solutions for common algorithms, including factorial calculation, Fibonacci sequence generation, and binary search

Learning Outcomes:

- Use quantitative analytical skills to evaluate and process numerical data.
- Solve problems using critical and creative thinking and scientific reasoning.
- Use computer and emerging technologies effectively.