**CMP1201 Computer Programming Fundamentals**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Period per  Week | | | Contact  Hour per  Semester | Weighted  Total Mark | Weighted  Exam Mark | Weighted  Continuous Assessment Mark | Credit  Units |
| LH | PH | TH | CH | WTM | WEM | WCM | CU |
| 45 | 30 | 00 | 60 | 100 | 60 | 40 | 4 |

**Rationale**

Competency in a programming language is prerequisite to the study of computer engineering. Object-oriented programming, event-driven applications, and the use of extensive APIs (application programming interfaces) are fundamental tools that computer engineering students need early in their academic program.

**Objectives**

 To introduce the student to the fundamental aspects of computer program writing

 To relate with the history and evolution of computer programming

 To expose the student to the state-of-the-art computer programming tools

 To provide the student with the basics of computer program design

**Course Content**

***1*. *History and Overview***

 Indicate some reasons for studying programming fundamentals

 Influential people; important areas such as programming constructs, algorithms, problem solving, data structures, programming paradigms, recursion, object-oriented programming, event-driven programming, and concurrent programming

 Contrast between an algorithm and a data structure

 Distinguish between a variable, type, expression, and assignment

 Highlight the role of algorithms in solving problems

 Describe some of the fundamental data structures such as array, record, stack, and queue

 Explain how divide-and-conquer strategies lend themselves to recursion

 Explore some additional resources associated with programming fundamentals

 Explain the purpose and role of programming fundamentals in computer engineering

***2. Programming Languages***

 Definition and History

 Characteristics (Pragmatics, Semantics and Syntax)

 Distinction between Text-based and Visual Programming

 Classification (Categorical, Chronological and Generational)

 Comparison of common programming languages (C, C++, C#, Java)

 Programming errors and warnings (syntax, logical, etc.)

***3. Programming Paradigms***

 Definition and rationale of a programming paradigm

 Types: Structured, Unstructured, Procedural, Object-oriented, Event-Drive, Generic etc.

 Separation of behavior and implementation

***4. ISO/ANSI C++ Programming Fundamentals***

 Bjarne Stroustrup Design rules

 Console applications basics (Source file, Basic I/O, Standard I/O Consoles, Function main( ))

 Fundamental data types

 Expressions and operators

 Control constructs (Conditional and Iterative)

 Pointers and Named collections (Arrays, Enumerators, Bit-fields, Unions)

 User-defined data types (Structures and Classes)

 Functions (In-built and User-defined)

 Object –oriented programming (Abstraction, Encapsulation, Inheritance, Composition, Polymorphism, Friend and Virtual Functions)

 File I/O

***5. Algorithms and Problem-Solving***

 Problem-solving strategies

 The role of algorithms in the problem-solving process

 Implementation strategies for algorithms

 Debugging strategies

 The concept and properties of algorithms

 Structured decomposition

***6. The Integrated Development Environment (IDE)***

 Definition

 Toolchains

 Advantages of IDEs

 Comparison of IDEs

 Using a typical IDE (Visual Studio)

**Learning Outcomes**

On completion of this course the student should be able to:

 Describe how computer engineering uses or benefits from programming fundamentals.

 Identify the appropriate paradigm for a given programming problem.

 Describe the way a computer allocates and represents these data structures in

 Use a suitable programming language to implement, test, and debug algorithms for solving simple problems.

memory.

 Outline the philosophy of object-oriented design and the concepts of encapsulation, subclassing, inheritance, and polymorphism.

**Recommended and Reference Books**

*[1]* Herbert Schildt, 2003. *C++ from the Ground Up,* Third Edition, McGraw- Hill/Osborne, ISBN 0-07-222897-0

*[2]* Chuck Easttom, 2003. *C++ Programming Fundamentals*, Charles River

Media, ISBN 158402371

*[3]* Bjarne Stroustrup, 2000. *The C++ Programming Language,* Addison-Wesley, ISBN 0-201-70073-5

*[4]* Michael T. Goodrich, Roberto Tamassio, David Mount, 1995. *Data Structures and Algorithms in C++,* John Wiley, ISBN 0-471-20208-8

*[5]* Robert Sedgewick, 2001. *Algorithms in C++,* Addison – Wesley, ISBN

0201510596

*[6]* Nell Dale, 2003. *C++ Data Structures,* Jones and Bartlett Publishers