Module Guide

Engineering Software C++

ENG_5_411

https://vle.lsbu.ac.uk/

School of Engineering

Level 5

Table of Contents

1. MODULE DETAILS

Module Title:	Engineering Software C++
Module Level:	5
Module Reference Number:	ENG_5_411
Credit Value:	20
Student Study Hours:	200 hours
Contact Hours:	26 hour lectures, 26 hour workshops
Private Study Hours:	148 hours
Pre-requisite Learning (If applicable):	Engineering Computing
Co-requisite Modules (If applicable):	None
Course(s):	2388, 2419, 4618, 4619, 4620, 4621, 4634 & 4635
Year and Semester	2019-2020, Semester 2
Module Coordinator:	Dr John Buckeridge
UC Contact Details (Tel, Email, Room)	020 7815 7420, j.buckeridge@lsbu.ac.uk, T803
Teaching Team & Contact Details (If applicable):	Dr John Buckeridge, j.buckeridge@lsbu.ac.uk, T803
Subject Area:	Engineering
Summary of Assessment Method:	Coursework (100%)
External Examiner:	Dr. Mihailo Ristic

2. SHORT DESCRIPTION

Engineering Software C++ is a module for students in Telecommunications Engineering and Computer Systems and Networks at level 5. This module introduces the syntaxes and semantics of programming language C++ and teaches students the intellectual knowledge in programming principles and programming skills with Object Oriented Programming (OOP) techniques. The practical skills include C++ program design with OOP and the use of the compiling tools for editing, compiling, linking and executing programs in workshops. After learning this module, students can pursue other software engineering and advanced programming courses and use OOP techniques to solve simple engineering problems.

How to use the module guide

This Module Guide is designed to help you find your way around the module. It tells you
- How teaching and learning is organised in this module
- What to do and when; and more importantly to give you a rounded view of the topic.

You are advised to use this module guide
- At the start of the module to plan ahead;
- Before each lecture to familiarize yourself with the content to be taught, this will enhance your confidence and help you to concentrate effectively during the lecture;
- After each lecture to check whether you have achieved the set objectives;
- After each phase to monitor your progress or weakness;
- An online submission in WEEK 7 to review and consolidate your basic programming knowledge and skill learnt in this module.

3. AIMS OF THE MODULE

This module is aimed to teach students design methodology and coding techniques in implementation of engineering programs with OOP techniques in C++.

4. LEARNING OUTCOMES

4.1 Knowledge and Understanding

On successful completion of this module, the students should be able to:

- know likeness and difference of C and C++
- know likeness and difference of structured programming methods and OOP techniques
- the syntax and semantics of C++.

4.2 Intellectual Skills

Students should be able to:

- work with the syntax and semantics of the language
- employ general principles of engineering design and practice
- use specification and design techniques
- decompose problems for software solutions
- understand the program design methodology using OOP techniques

4.3 Practical Skills

Students should be able to:

- use a software development environment for editing, debugging, compiling, linking and running programs
- design simple programs with OOP techniques to solve engineering problems
- draft program documentation

4.4 Transferable Skills

Students should be competent in:

- General programming knowledge
- Report-writing skills
- Problems solving ability

5. ASSESSMENT OF THE MODULE

Assessment is by Course Work contributing 100%. The Course Work consists of: logbook (30%), assignment (20%), phase test (50%).

The times for assessment components are summarised in the table below:

Assessment Component	*Type of assessment*	Timing of assessment	Contribution to module mark
1	Submission of Labs Report 1	Weeks 1 to 7	18%
2	Submission of Labs Report 2	Weeks 9 to 12	12%
3	Final OOP Assignment Report	Week 15	20%
4	Phase test	After Week 15	50%

5.1 Handing in Your Reports

All reports (aka Labs and Assignment) should be created and maintained in electronic format (e.g. Word.doc). Code used for each exercise should be pasted into the report as text and proof of program execution (DOS box image) should be pasted below as a clear to read (large) image offering proof of adequate execution.
Submission will be via MOODLE during a predetermined time window for each.

5.2 Plagiarism

Plagiarism is to hand in work which somebody else has done without clearly stating a reference for it. You can take work from books or papers or programs that you have been given as part of the module, but **all work which is not directly your own work should be referenced so that readers can see whose work it is and where the original work can be found**. Plagiarism in Logbooks and examinations will be penalised. You might work in groups in the laboratory but separate and identifiably distinct submissions must be made by each student unless specifically directed otherwise.

5.3 Late Submission

Reports which are handed in after the deadline date will incur a penalty unless a late submission form has been filled and signed. Work submitted within one week after the deadline will be awarded a maximum mark of 40%. Work submitted more than two weeks after the deadline awarded a zero mark.

6. FEEDBACK

Feedback will normally be given to students soon after the submission of their course works. Logbooks will be marked in class towards the end of the module. The students' comments on the teaching organisation will be fed back as soon as possible; and the relevant measures will be taken to satisfy the student requirements. In the case of a few students who are weak in programming skills, the feedback to their opinions or requirements will be enhanced.

7. INTRODUCTION TO STUDYING THE MODULE

7.1 Overview of the Main Content

The main content includes: syntaxes of C++, multiple files and data scope, pointers and the OOP techniques governing programming. The content will be covered in two phases, phase 1: programming fundamentals and skills and phase 2: OOP. Coding practice to enhance the knowledge above will be carried out in workshops through the programme.

7.2 Overview of Types of Classes

Topics will be presented in a mixture of lecture presentation and course notes. Where appropriate practical examples will be used to illustrate concepts and software design. Workshop sheets will be provided for appropriate topics.

The lecture notes will be used in lectures and workshops through the semester. A very practical approach is maintained with taught material and experiments to illustrate the structure and operation of essential elements of the programming language and to investigate the limitations and problems associated with some of the common constructs. You are encouraged to exploit opportunities to use laboratory facilities to follow some of these as self-directed and open-ended learning activities.

Workshop sessions support and extend the lecture material and are self-paced. It is more important that you have a very clear understanding of the fundamental elements of programming than you struggle to keep up with more experienced colleagues. Self-help is an important element in this work and you will find that you learn more yourself as you try to help others. The exercises move gradually from programs that you are given to enter and run, to the need to modify these and eventually to create your own software solutions to specified problems.

- Lectures: Two hours each week
- Workshops: Two hours each week, focusing on the use of compiling environment and the design of programs with OOP in C++.

7.3 Importance of Student Self-Managed Learning Time

Self-managed learning time will be demanded to support all the work on the module. Self-managed work must be recorded in the logbook. As well as being a comprehensive record of your work, your logbook will be useful to you in discussions with staff.

The Moodle site for this module contains a lot of information and should be consulted regularly.

Students should use their independent learning time to work through the core reading for this module. Some students who are weak in mathematics will gain additional advices from lecturers on improving their math by self-learning.

Email is likely to be particularly important for part-time students, whose only opportunity for team meetings is over lunch on a very busy day.

7.4 Employability

Enhancing employability is an important issue this module should not ignore. Using the knowledge acquired in class to solve engineering problems will be stressed through teaching. Some key issues associated to job hunting in the field of software engineering will be advised.

8. THE PROGRAMME OF TEACHING, LEARNING AND ASSESSMENT

This section includes lecture programme and workshop programme.

Lecture Programme
Week 1: Introduction to computers, programs, and C++
Lecture Topics: • what is a computer • programming languages • operating systems • a simple C++ program • C++ program-development cycle • programming style and document • programming errors
Week 2: Elementary programming
Lecture Topics: • Writing a simple program • Reading input from the keyboard • Identifiers • Variables • Assignment statements and assignment expressions • Named constants • Numeric data types and operations • Evaluating expressions and operator precedence • Displaying current time • augmented assignment operator • Increment and decrement operators • Numeric type conversions • software development process • common errors

Week 3: Conditional programming

Lecture Topics:

- The Boolean data type and Boolean expressions
- If statement, two-way if-else statements, nested if statements
- Common errors and pitfalls of if statements
- Generating random numbers
- Logical operators and combined Boolean expressions
- Switch statements
- Conditional expressions
- Operator precedence and associativity
- Debugging

Week 4: Mathematical functions, characters, and strings

Lecture Topics:

- Commonly used mathematical functions
- Character data type and operations
- String data type and operations
- Formatting console output
- Simple file input and output

Week 5: Loops

Lecture Topics:

- The while loop
- The do while loop
- The for loop
- Which loop to use
- Nested loops
- Minimizing numeric errors
- loop control using break and continue

Week 6: Functions

Lecture Topics:

Defining a function
Calling a function
Passing arguments by value
Modularizing code
Overloading functions
Function prototypes
Default arguments
Inline functions
Local, global and static local variables
Passing arguments by reference
Constant reference parameter
Function abstraction and stepwise refinement

Week 7: Online Submission of Labs Report 1

Week 8: Single-dimensional arrays and c-strings

Lecture Topics:
Array basics
Passing arrays to functions
returning arrays from functions
Searching arrays
Sorting arrays
C-string

Week 9: Objects and classes

Lecture Topics:
Defining classes for objects
Creating objects
Constructors
Constructing and using objects
Separating class definition and implementation
Preventing multiple inclusions
Inline functions in Classes
Data field encapsulation
The scope of variables
Class abstraction and encapsulation

Week 10: Object-oriented Thinking

Lecture Topics:
The string class
Passing objects to functions
Array of objects
Instance and static members
Constant member functions
Thinking in objects
Object composition
class design guidelines

Week 11: Pointers and dynamic memory management

Lecture Topics:
Pointer basics
Defining Synonymous types using the typedef keyword
Using const with pointers
Passing pointer arguments in a function call
Returning a Pointer from functions
Dynamic persistent memory allocation
Creating and accessing dynamic object
The this pointer
Destructor
Copy constructors customizing copy constructor

Week 12: Inheritance and Polymorphism

Lecture Topics:
Base classes and derived classed
Generic programming
Constructors and destructors
Redefining functions
Polymorphism
Virtual functions and dynamic bonding
The protected keyword
abstract classes and pure virtual functions

Week 13: online submission of Labs Report 2
Week 15: OOP assignment submission

Workshop Programme	
Week 1	Using Dev-C++, for editing, compiling and executing multiple file programs. Programming exercises for **Programming errors**.
Week 2	Programming exercises for **Elementary programming**.
Week 3	Programming exercises for **Conditional programming**
Week 4	Programming exercises for **Mathematical functions, characters, and strings**
Week 5	Programming exercises for **Loops**
Week 6	Programming exercises for **Functions**
Week 7	***Submission of First Report***
Week 8	Programming exercises **for Single-dimensional arrays and c-strings**
Week 9	Programming exercises for **objects and classes**
Week 10	Programming exercises for **Object-oriented Thinking**
Week 11	Programming exercises for **Pointers and dynamic memory management**
Week 12	Continuation of exercise for **Inheritance and Polymorphism**
Final Submissions in weeks 13 and 15	

9. LEARNING RESOURCES

There are many books on programming with OOP techniques in C++ in our library. These books were written by authors from different fields, e.g., Software engineering, industry, education or research, and so they are very different in perspectives from which to introduce and analyse problems. You should choose ones suitable for you. I recommend one books as core materials and three as optional materials in this module guide, but you are still advised to glance at more books to find one in favour of your interests and tastes.
Note: all the reading materials can be found in Perry library. The core material is in both printed and digital format.

9.1 Core Materials (text book)

- Y. Danel Liang, Introduction to Programming with C++, Pearson; 3 edition (25 April 2013)

9.2 Optional Materials

- Deitel, H., C++ How to Program (4th Edition), Pearson Prentice Hall, 2004.
- Harman & Jones, *First Course in C++,* McGraw-Hill, 1997.
- Parsons, D., *Object Oriented Programming with C++*, DP Publications, London, 1994.

There are also many online tutorials and Q&A forums that are free and easy to use:

9.3 Online Resources

http://www.cplusplus.com/doc/tutorial/
http://www.learncpp.com/
http://www.cprogramming.com/tutorial/c++-tutorial.html
https://www.sololearn.com/Course/CPlusPlus/
https://www.codeacademy.com