LONDON SOUTH BANK UNIVERSITY

Unit Guide

Engineering Software 2

ECS-2-823

http://www.blackboard.lsbu.ac.uk/

Faculty of Engineering,
Science and the Built
Environment

2008-09

become what you want to be

Template version: 5

Table of Contents

1. UNIT DETAILS

Unit Title:	Engineering Software 2
Unit Level:	2
Unit Reference Number:	ECS-2-823
Credit Value:	One
Student Study Hours:	144
Contact Hours:	24 hour lectures, 24 hour workshops
Private Study Hours:	4x24=96 hours
Pre-requisite Learning (If applicable):	Engineering Computing or Engineering Software 1
Co-requisite Units (If applicable):	None
Course(s):	TeCNE BEng TC2F TC2P
Year and Semester	2008-09, Semester 1
Unit Coordinator:	Shuwo Chen
UC Contact Details (Tel, Email, Room)	020 7815 7554, chensc@lsbu.ac.uk, T410
Teaching Team & Contact Details (If applicable):	Dr Shuwo Chen, chensc@lsbu.ac.uk, T410
Subject Area:	Communication and Computer Network Engineering
Summary of Assessment Method:	50% Coursework + 50% End-of-Unit Exam

2. SHORT DESCRIPTION

Engineering Software 2 is a unit for students in TeCNE in year 2. This unit introduces the knowledge and understanding of syntaxes and semantics of programming languages C++. The unit teaches students the intellectual skills in programming principles and methods with Object Oriented Programming (OOP) techniques in C++. The practical skills include the design of programs with OOP acquired in class and the use of VC++ 6.0 for editing, compiling, linking and executing programs. After learning this unit, students can purse other software engineering and advanced programming subjects and use OOP techniques to solve simple software engineering problems.

HOW TO USE THE UNIT GUIDE

This Unite Guide is to help you find your way around the unit. It is to tell you

- How teaching and learning is organised in this unit
- What to do and when; and more importantly to give you a rounded view of the topic.

You are advised to use this unit guide

- At the start of the unit 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;
- Before the examination to review and consolidate you knowledge and skill learnt in this unit.

3. AIMS OF THE UNIT

This unit 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 unit, the students should be able to:

- know likeness and difference of C and C++
- know likeness and difference of structured programming method 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 UNIT

Assessment includes one Course Work contributing 50% (consisting of a phase test 25% and a practical assignment 25%) and one closed book Examination contributing 50% in which all questions must be attempted.

5.1 Handing in Your Assignment

All assignments should be logged in to the course secretary in Room T313. The signature of the secretary should be obtained and kept as a proof. A standard front sheet must be used for all work submitted for assessment. Ensure that the front sheet is securely attached to the document. All course work must be

handed in before 4.00 p.m. on or before the date given in the assignment specification.

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 unit, 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 assignments 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

Assignments 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 unit. 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 UNIT

7.1 Overview of the Main Content

The main content include: syntaxes of C++, multiple files and data scope, pointers and the OOP techniques governing programming. 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 teacher 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 VC++ 6.0 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 unit. 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 Blackboard site for this unit 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 unit. 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 unit 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: Pointer variables

Learning Outcomes:

On completion of this section you will have the ability to
- recognise address, pointer and value of a variable
- use pointer variables to compute values.

Lecture Topics:
- address, pointer and value of a variable
- Declaration of pointer variables
- Wrong order of declaration of pointer variables
- Cancellation of pointer operators * and &
- Mathematical computations of pointer variables

Week 2: Pointer variables (cont.)

Learning Outcomes:

On completion of this section you will have the ability to
- Operate 1D arrays and 2D arrays.
- Find the address of any element of an array
- Find the value of any element of an array with the pointer variable.

Lecture Topics:
- pointer variables used in 1D arrays.
- pointer variables used in 2D arrays. (optional)

Week 3: Functional programming

Learning Outcomes:

On completion of this section you will have the ability to
- know difference of top-down programming method and functional programming method
- Perform complex tasks with functional programming method
- Use reference variables

Lecture Topics:
- Review of functional programming method with a simple sample
- Methods for passing actual parameters to formal parameters.
- Call a functions by passing actual parameters to formal parameters
- Reference variables
- Call a functions by passing actual parameters to reference as formal parameters

Week 4: Functional programming (cont.)

Learning Outcomes:

On completion of this section you will have the ability to
- Deal with issues associated to pass actual parameters to formal parameters.
- Use arrays as parameters
- Pass an actual array to a formal reference array
- Use inline functions and overloaded functions in programming.

Lecture Topics:
- Call a functions by passing actual parameter addresses to pointer variables of formal parameters
- Passing arrays as parameters
- References as arrays
- Default parameters
- Inline functions
- Overloaded function names
- Main()

Week 5: Multiple files and data scope

Learning Outcomes:

On completion of this section you will have the ability to
- Write programs with multiple files.
- Know to edit, compile, link and execute a project containing several files.

Lecture Topics:
- Structure of a multiple file program, header file, course code files
- Project
- Compile projects

Week 6: Multiple files and data scope (cont.)

Learning Outcomes:

On completion of this section you will have the ability to
- Identify scope of variable
- Define variables with proper data scope
- Use static and global variables

Lecture Topics:
- Data scope, data validity regions
- Program region, file region, function region and block region
- Static variables and global variables

Week 7: Phase Test

Week 8: Conceptual OOP

Learning Outcomes:

On completion of this section you will have the ability to
- Know difference between structured programming method and OOP techniques
- Understand OOP properties: protect data by encapsulation, enhance program power by inheritance and save memory and time by polymorphism.
- Use OOP to programs to solve simple problems.

Lecture Topics:
- Structured programming method and OOP techniques
- Class, object, member data, private:, protected: public:
- member functions, constructor.
- protect data by encapsulation, enhance program power by inheritance and save memory and time by polymorphism.

Week 9: Conceptual OOP (cont.)

Learning Outcomes:

On completion of this section you will have the ability to
- use Inheritance to write programs

Lecture Topics:
- Inheritance
- father class, son class.

Week 10: More programming techniques with OOP

Learning Outcomes:

On completion of this section you will have the ability to
- Use polymorphism to write programs
- Use overloading techniques
- Use complex methods to deal with objects.

Lecture Topics:
- Polymorphism
- Function overloading
- Destructor
- Some methods dealing with objects.

Week 11: More programming techniques with OOP (cont.)

Learning Outcomes:

On completion of this section you will have the ability to
- Use pointers to deal with objects and functions
- Use constant to deal with objects and functions
- Deal with objects with methods such as arrays as objects, reference variables to call object's member functions, dynamic management of objects, assignment of objects, copy of objects

Lecture Topics:
- some methods dealing with objects (cont.): arrays as objects, reference variables to call object's member functions, dynamic management of objects, assignment of objects, copy of objects
- pointers
- constant

Week 12: More programming techniques with OOP (cont.) and End-of-Unit Revision

Learning Outcomes:

On completion of this section you will have the ability to
- Use friend to access protected data or functions
- Use template to realise polymorphism

Lecture Topics:
- Friend
- Template
- Some method dealing with member data and functions

Week 13 & 14: End-of-Unit Examination

Workshop Programme	
Week 1	VC++ 6.0 for editing, compiling and executing multiple file programs. Files: creating, saving, copying and deleting. Programming exercises for pointer variables.
Week 2	Programming exercises for pointer variables.
Week 3	Programming exercises for functional Programming techniques
Week 4	Programming exercises for functional Programming techniques
Week 5	Programming exercises for multi file programs. Project, header file, source codes (.cpp).
Week 6	Programming exercises for data scope

Week 7	***Phase test***
Week 8	Programming exercises for conceptual OOP techniques: Class, object, member data, member function, constructor, private:, protected: public:.
Week 9	Programming exercises for conceptual OOP techniques: Inheritance – father class, son class.
Week 10	Programming exercises for more programming techniques with OOP: Overloading, destructor, some methods dealing with objects.
Week 11	Programming exercises for more programming techniques with OOP: some methods dealing with objects (cont.), pointers, constant
Week 12	Programming exercises for more programming techniques with OOP: Friend, template, some method dealing with member data and functions

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 analyze problems. You should choose ones suitable for you. I recommend two books as core materials and three as optional materials in this unit guide, but you are still advised to glance at more books to find one in favour of your interests and tastes.

Teacher Notes by Shuwo Chen has been used for three years in teaching. These notes are featured by introducing concepts and explaining design methods in a simple way and step by step, and so students who are weak in computer engineering can follow up.

Another core book is recommended here is: Bell, D., *The Essence of Programming Using C++*. According to my experience, this book is organised in the way that students like to use to learn new knowledge unlike some other books with too much theoretical comment that tends to make students lose their interests.

9.1 Core Materials

- S.Chen, Lecture notes on Engineering Software 2, ECCE, FESBE, LSBU, 2008.

- Bell, D., *The Essence of Programming Using C++,* Prentice-Hall, 1997.
  (Chen's comments: In spite of the advantages above, the disadvantage of this book is of less contents on OOP techniques. Fortunately you can read the Teacher Notes by Shuwo Chen for the more OOP techniques.)

9.2 Optional Materials

- Deitel, H., C How to Program (4th Edition), Pearson Prentice Hall, 2004.
  (Chen's comments: this book is comprehensive as to contents, students can use it as a 'dictionary' to look up terms or concepts that are explained clearly in the book.)

- Harman & Jones, *First Course in C++,* McGraw-Hill, 1997.

- Parsons, D., *Object Oriented Programming with C++*, DP Publications, London, 1994.

- Hubbard, *Programming with C++,* McGraw-Hill Shaum Outline Series, 1996.

9.3 More Relevant Materials

Most of the following books concentrate on the C++ language at the expense of the object-oriented concepts, but I try to give a short assessment of the book from both points of view. There are a vast number of books on C++ which I have not included in this list [indicative prices are included in square brackets, these change often and are unlikely to be completely up-to-date]. I have explicitly excluded books for one or more of the following reasons:

- they are addressed to C programmers who wish to 'convert' to C++;
- they are C++ programmers manuals with little tutorial support;
- they have almost no information about the object-oriented approach; or
- they cost over £25 [in 1997].

The books marked with '*' appear to be good but you may well discover another appropriate book by searching through libraries or bookshops.

Ameraal, L., 'C++ for Programmers', Addison-Wesley, 1991. [£17.95]

> Concentrates on the language and gives little support for the object-oriented approach. Assumes some programming knowledge, but not necessarily C.

Barclay, KA., & Gordon BJ., 'C++ Problem Solving and Programming', Prentice-Hall, 1994. [£19.95]

> Good thorough treatment of C++ including examples of its use in object oriented programming. Some discussion of object oriented design and abstraction.

Bergin, J., 'Data Abstraction: The Object Oriented Approach using C++', McGraw-Hill, 1994. [£22.95]

> Big book with lots of examples. Takes object oriented programming seriously. Good thorough treatment of C++. Includes disc.

Capper, DM., 'Introducing C++ for Scientists, Engineers and Mathematicians', Springer-Verlag, 1994 . [£20.95]

> Concentrates on the language and on its application to scientific and engineering problems. Little on the object-oriented approach. No programming knowledge assumed

***** Davies, CR., 'C++ for Dummies', IDG Books, 1994. [£18.99]

> Good discussions of the language and the object oriented approach. Despite its title, it looks quite a good book.

Eckel, B., 'C++ Inside and OUT', McGraw-Hill, 1993. [£23.95]

> This is a big book and looks like good value in terms of pages per £. Thorough and extensive treatment of the language, but rather short on object oriented approaches.

Ettinger, J., 'Programming in C++', Macmillan, 1994. [£14.95]

Small, cheap book giving a number of examples, but rather short on treatment of object oriented programming.

Flamig, B., 'Turbo C++ Step by Step', Wiley, 1993. [£19.95]

Thorough treatment of C++, but not a great deal on object orientation. Contains useful information on the use of the Turbo C++ environment.

Gorlen, KE., Orlow, SM., Plexico, PS., 'Data Abstraction and Object Oriented Programming in C++', Wiley, 1991. [£23.50]

Good thorough treatment of C++ with lots of examples of objects and their use in programs, but biased towards health administration applications.

Graham, N., 'Learning C++', McGraw-Hill, 1991. [£20.95]

Thorough, though incomplete (perhaps to do with its age) treatment of C++. Rather half-hearted attempt to deal with object orientation.

Hahn, B., 'C++: A Practical Introduction', NCC Blackwell, 1994. [£18.99]

Good treatment of the language but little on object oriented approaches.

*Henderson, P., 'Object Oriented Specification and Design with C++', Prentice- Hall, 1993. [£22.95]

Very good on object orientation and the programming language. Includes lots of examples and exercises. Includes a disc containing an implementable specification (prototyping) language.

Mitchell, RJ., 'C++ Object Oriented Programming', 1993 [£15.99]

Thorough treatment of C++ with plenty of examples, but is based on an extended graphics case study. Good discussion of object orientation.

Neibauer, AR., 'Your First C/C++ Program', Sybex, 1994. [£23.99]

This is a very basic book which addresses both C and C++. No object oriented techniques. Price includes a compiler (not TURBO C/C++) on disc.

*Smith, MA., 'Object Oriented Software in C++', Chapman-Hall, 1993. [£17.99]

This is clearly presented and written and contains lots of examples. There is some general discussion of object oriented approaches, though more might have been expected given the title.

Smith, NE., 'OOP Using Turbo C++', Wordware, 1991. [£13.95]

This is a small book with rather large print, so, even though its comparatively cheap, there do not seem to be very many words (or pages) to the £. It contains lots of explanation and small examples. There is some discussion of object orientated approaches.

Schildt, H., 'C++: The Pocket Reference', (2nd ed.), McGraw-Hill, 1992. [£7.95]

Small and cheap, but with "all essential functions". This is not a tutorial book or a text book. It is a reference book for the language only. There is nothing about the object oriented approach in general. Assumes a basic knowledge of C and C++.