ESE-1025 Principles of Embedded Programming

Computer Studies

Course Number:	Co-Requisites:	Pre-Requisites:
ESE-1025	N/A	N/A
Prepared by:	Jay Nadeau, Outline Creator	
Approved by:	Chris Slade, Dean Computer Studies and International Education	
Approval Date:	Monday, June 11, 2018	
Approved for Academic Year:	2018-2019	
Normative Hours:	75.00	

Course Description

In this fundamental course, students utilize the C++ programming language to explore topics related to development for modern embedded systems. These topics include the use of the C++ standard library such as strings, vectors and other relevant objects and data structures, as well as object oriented and concurrent programming. Laboratory sessions are used to provide the student with the opportunity to design and implement topics taught during the course.

Course Learning Outcomes/Course Objectives

1. **Implement embedded C++ programs that utilize plain old datatypes (PODs), control statements, arrays, functions, overloaded functions and default arguments.**
   - 1.1 Explain the difference between branching statements such as if, else, else if and switch.
   - 1.2 Identify the differences between looping control structures such as while, do while and for loops.
   - 1.3 Write applications that use both the logical and mathematical operators.
   - 1.4 Describe the difference between logical and mathematical operators.
   - 1.5 Implement applications that use arrays, functions, default arguments and pass arrays to functions as arguments.
   - 1.6 Write programs that use overloaded functions, templated functions, and inline functions.
   - 1.7 Describe the differences between overloaded functions, template functions and inline functions.
   - 1.8 List the different types of variable scope.

2. **Design and implement classes that use object oriented programming techniques such as operator overloading, inheritance, virtual functions, polymorphism, and friend operators.**
   - 2.1 Write applications that use objects instantiated from student defined classes.

2.2 Design classes that contain default constructors, copy constructors, destructors and accessor and mutator methods.

2.3 Design and implement classes that contain private and public data and methods.

2.4 Design classes that use polymorphism, single and multiple inheritance and list some of the pitfalls of multiple inheritance.

2.5 Describe the ‘this’ pointer and how it relates to objects at runtime.

2.6 Define when a constructor is called and when a destructor is called.

2.7 Design and implement classes that use virtual functions and operator overloading.

2.8 Design and implement classes that use pure virtual functions and describe the difference between a class and an interface.

2.9 Write programs that use friend functions and friend classes.

3. Write applications that use generic programming, exception handling, and multiple threads.

3.1 Write programs that use templated functions.

3.2 Design and implement template classes.

3.3 Design classes that extend the STL exception class and provides a constructor that accepts an exception message of type string.

3.4 Explain the meaning of stack unwinding.

3.5 Implement programs that use try, catch, finally blocks and use functions that re-throw exceptions.

3.6 Describe when to use exceptions.

3.7 Write applications that create and manage STL thread objects.

4. Write embedded applications that correctly use condition variables, shared memory and inter-thread communication.

4.1 Explain the difference between semaphores, critical sections and mutexes.

4.2 Write applications that use condition variables to signal multiple threads.

4.3 Write programs that correctly perform inter-thread communication using a thread safe queue.

5. Create programs that handle input/output routines, streams, serial communication and socket communication.

5.1 Write applications that use stringstreams to read from and write to a file.

5.2 List the different ios flags and describe their purpose.

5.3 Describe the difference between random access files and sequential access files.

5.4 Design a stream management class that automatically opens and closes a stream.

5.5 Write programs that use the stream manipulators to set the width, precision and base of output data.

5.6 Write applications that create child processes and pipes data between parent and child processes.

5.7 Write applications that open, close, read and write to serial ports using the Linux device tree.

5.8 Explain the difference between synchronous and asynchronous I/O.

5.9 Write programs that use the socket API to send and receive data over a network.

6. **Apply the C++ Standard Template Library in embedded applications and list the benefits of using the STL containers and algorithms such as strings, lists, vectors, maps, iterators and sorting algorithms.**
   - 6.1 List the different STL containers, how they differ and what their main advantages are.
   - 6.2 List the different STL associative containers, how they differ and what their main advantages are.
   - 6.3 Implement programs that uses iterators to traverse and manipulate vectors, lists and queues.
   - 6.4 Write applications that use the sorting algorithms on STL containers of data.
   - 6.5 Write applications that search, merge and modify STL containers using such algorithms as find, for each, fill, merge, copy and max.

7. **Correctly allocate and manage memory using pointers, references, arrays, smart pointers, auto variables, callback functions, functors, static variables and implement move semantics in embedded C++ programs.**
   - 7.1 Explain how to declare and initialize pointers.
   - 7.2 Write functions that accept pointers and references as arguments.
   - 7.3 Explain how arrays and pointers are related.
   - 7.4 Explain the benefits of smart pointers.
   - 7.5 Design and implement a smart pointer class.
   - 7.6 Write applications that correctly use auto variables and auto pointers.
   - 7.7 Write applications that use callback functions.
   - 7.8 Design and implement a functor class.
   - 7.9 Describe how a static variable differs from automatic or local variables.
   - 7.10 Design and implement a class that uses move semantics. I.e., contains a move constructor and move assignment operator.

8. **Cross-compile, cross-debug and remotely execute embedded C++ applications.**
   - 8.1 Install and setup a cross-compiler and cross-toolchain.
   - 8.2 Use Eclipse or other similar IDEs to cross-compile an application.
   - 8.3 Explain the difference between the target machine and the host machine.
   - 8.4 Use breakpoints and step through an application using a cross debugger and a debug server on the target machine.
   - 8.5 Use the remote execution feature in Eclipse.

Learning Resources

a. Required

C++ 11 for Programmers by Paul Deitel and Harvey M. Deitel; Prentice Hall, 2nd ed. (Mar. 2013).

b. Supplemental

C++ Concurrency in Action: Practical Multithreading by Anthony Williams; Manning Publications; 1st ed. (Mar.

2012).
Real-Time C++: Efficient Object-Oriented and Template Microcontroller Programming by Christopher Kormanyos; Springer; 2nd ed. (Nov. 2015).

Student Evaluation

Tests - 40%
Assignments - 10%
Assignments (2 equally weighted) @ 5% each
Laboratory Sessions (10 equally weighted) - 50%

Grade Scheme

The round off mathematical principle will be used. Percentages are converted to letter grades and grade points as follows:

Mark (%)	Grade	Grade Point	Mark (%)	Grade	Grade Point
94-100	A+	4.0	67-69	C+	2.3
87-93	A	3.7	63-66	C	2.0
80-86	A-	3.5	60-62	C-	1.7
77-79	B+	3.2	50-59	D	1.0
73-76	B	3.0	0-49	F	0.0
70-72	B-	2.7			

Prior Learning Assessment and Recognition

Students who wish to apply for prior learning assessment and recognition (PLAR) need to demonstrate competency at a post-secondary level in all of the course learning requirements outlined above. Evidence of learning achievement for PLAR candidates includes:

- Not Applicable: Post graduate course and not eligible for PLAR.

Course Related Information

The course is designed primarily to deliver more emphasis on hands on experience via laboratory sessions and weekly assignments.

College Related Information

Academic Integrity
Lambton College is committed to high ethical standards in all academic activities within the College, including research, reporting and learning assessment (e.g. tests, lab reports, essays).
The cornerstone of academic integrity and professional reputation is principled conduct. All scholastic and academic activity must be free of all forms of academic dishonesty, including copying, plagiarism and cheating.

Lambton College will not tolerate any academic dishonesty, a position reflected in Lambton College policy. Students should be familiar with the Students Rights and Responsibilities Policy, located on the MyLambton

website. The policy states details concerning academic dishonesty and the penalties for dishonesty and unethical conduct.

Questions regarding this policy, or requests for additional clarification, should be directed to the Lambton College Centre for Academic Integrity

Students with Disabilities

If you are a student with a disability please identify your needs to the professor and/or the Accessibility Centre so that support services can be arranged for you. You can do this by making an appointment at the Accessibility Centre or by arranging a personal interview with the professor to discuss your needs.

Student Rights and Responsibility Policy

Acceptable behaviour in class is established by the instructor and is expected of all students. Any form of misbehaviour, harassment or violence will not be tolerated. Action will be taken as outlined in Lambton College policy.

Date of Withdrawal without Academic Penalty

Please consult the Academic Regulations and Registrar's published dates.

Waiver of Responsibility

Every attempt has been made to ensure the accuracy of this information as of the date of publication. The content may be modified, without notice, as deemed appropriate by the College.

Students should note policies may differ depending on the location of course offering. Please refer to campus location specific policies:

- Lambton College - Sarnia Campus: https://www.mylambton.ca/Policies/
- Lambton College - Non-Sarnia Study Locations: https://www.mylambton.ca/Lambton_in_GTA/Student_Policies/

Note: It is the student’s responsibility to retain course outlines for possible future use to support applications for transfer of credit to other educational institutions.