University of Macau
Faculty of Science and Technology
Department of Computer and Information Science
SFTW241 Programming Languages Architecture I
Syllabus
2nd Semester 2011/2012
Part A – Course Outline

Compulsory course in Computer Science

Course description:
(2-1-2) 4.5 credits. This course provides in-depth coverage of object-oriented programming principles and techniques using C++. Topics include classes and objects, vectors, overloading, inheritance, polymorphism, templates, stream I/O, file processing and exception handling. Optional topics include the comparison of C++ with other OOP languages such as Java.

Course type:
Theoretical with substantial laboratory/practice content

Prerequisites:
- SFTW120

Textbook(s) and other required material:
- Paul Deitel and Harvey Deitel. (2009) *C++ How to Program***.** Prentice Hall, US.

References:
- Stephen Prata. (2004). *C++ Primer Plus*. 5th ed., Sams.
- Bruce Eckel. (2006). *Thinking in Java*. Prentice Hall.

Major prerequisites by topic:
- Programming languages and algorithms

Course objectives:
- Learn the basic concepts of object oriented programming (OOP), demonstrated by the use of C++ [a]
- Able to develop solutions to problems demonstrating the use of standard language constructs [c,e,k]
- Able to develop solutions to problems demonstrating the use of data abstraction, encapsulation, overloading, inheritance, and polymorphism. [c,e,k]
- Apply the use of templates to enable software reuse [c,e,k]
- Practice OOP techniques on small (assignments) and medium-size projects [c,e,k,l]

Topics covered:
- **Introduction (5 hours)**: Review the concepts, history, and the different types of programming languages. Introduce a typical C++ program development environment. Give a brief introduction to the industry-standard object-oriented system modeling language, the UML. To test-drive C++ applications in GNU C++ on Linux.
- **Basic concepts of classes and objects (5 hours)**: Learn how to define a class and use it to create an object, define member functions to implement the class's behaviors, declare data members to implement the class's attributes, and initialize a class by constructor. Introduce the techniques to engineer a class to separate its interface from its implementation and encourage reuse.
- **Arrays, vectors, and pointers (5 hours)**: Learn to use array data structure and discuss the way to declare, initialize, and refer to array elements. Learn the basic searching and sorting techniques. Learn to declare and manipulate multidimensional arrays and the use of the C++ standard library class template. Review pointers. Discuss the similarities and differences between pointers and references, and the relationships between pointers and arrays.
- **Classes (10 hours):** Understand class scope and learn how to access class members via different ways. Discuss the use of constructors and destructors, and the order of calling. Learn the use of `const` objects and member functions, and the use of `friend` functions and `friend` classes. Discuss the concept of a container class and the

notion of iterator classes that walk through the elements of container classes. Study the use of proxy classes to hide implementation details from a class's clients.

- **OOP techniques - Overloading, Inheritance, and Polymorphism (15 hours)**: Learn what and how to use operator overloading to simplify programming. Discuss the way to overload operators for user-defined classes and operators, and to convert objects from one class to another. Introduce the `string` class. Discuss the use of inheritance; learn the notions of base classes and derived classes, and the relationships between them. Introduce the `protected` member access specifier. Learn to use constructors and destructors in inheritance hierarchies. Understand the differences between `public`, `protected`, and `private` inheritance. Study the use of polymorphism and its advantages in programming. Distinguish between abstract and concrete classes, and how to create abstract classes. Show how C++ implements `virtual` functions and dynamic binding, and how `virtual` destructors work.
- **Templates (5 hours)**: Study the use of templates to enhance software reuse. Discuss the differences between function templates and function-template specializations. Learn to use class templates to create groups of related types and functions. Learn to overload function templates, and to understand the relationships among templates, friends, inheritance, and static members.
- **Stream I/O and file processing (5 hours)**: Overview of the stream-I/O class hierarchy. Learn to use and to format input and output. Study the way to create, read, write and update files, both sequential file processing and random-access file processing. Discuss the differences between formatted-data and raw-data file processing.
- **Exception handling (5 hours)**: Introduce exceptions and the standard exception hierarchy. Learn to use `try`, `catch`, and `throw` to detect, handle, and indicate exceptions, respectively. Learn to process uncaught and unexpected exceptions.
- **From C++ to Java (10 hours)**: A brief overview of the Java programming language, compare the differences between C++ and Java in terms of the use of objects, string operations, references, memory management, and the way of error handling, etc. The Java deployment model will be introduced.

Class/laboratory schedule:

Timetabled work in hours per week			No of teaching weeks	Total hours	Total credits	No/Duration of exam papers
Lecture	**Tutorial**	**Practice**				
2	1	2	14	70	4.5	1 / 3 hours

Student study effort required:

Class contact:	
Lecture	28 hours
Tutorial	14 hours
Hands-on practice	28 hours
Other study effort	
Self-study	30 hours
Homework assignment	20 hours
Project / Case study	15 hours
Total student study effort	135 hours

Student assessment:

Final assessment will be determined on the basis of:

Homework	20%	Project	30%
Mid-term	20%	Final exam	30%

Course assessment:

The assessment of course objectives will be determined on the basis of:

- Homework, project and exams
- Course evaluation

Course outline:

Weeks	Topic	Course work
1	**Introduction** Programming languages and OOP, typical C++ development	

Weeks	Topic	Course work
	environment, object-oriented analysis and design	
2	**Introduction to classes and objects** Defining classes, objects, member functions, constructors, destructors	
3	**Arrays, vectors, and pointers** Declaring, initializing and using arrays, arrays and functions, searching and sorting, class template `vector`, declaring, initializing and using pointers, pointer operators	
4-5	**Classes** Class scope, accessing class members, separating interface from implementation, use of `const`, object members, `friend` functions and `friend` classes, data abstraction and information hiding	
6	**Operator overloading** Fundamentals and restrictions, overloading stream operators, unary operators and binary operators; dynamic memory management, converting between types	
7	**Inheritance** Base classes and derived classes, `protected` members, `public`, `protected`, and `private` inheritance	Midterm exam
8	**Polymorphism** Polymorphism, relationships among objects in an inheritance hierarchy, type fields, abstract classes `and` pure `virtual` functions	
9	**Templates** Function templates, overloading function templates, class templates	
10	**Stream I/O and file processing** Classic streams vs. standard streams, `iostream` library, stream input and putput, stream format states and manipulators, files and streams, create, access, update sequential and random-access files,	Project
11	**Exception handling** Processing exceptions, rethrow an exception, unexpected exceptions, stack unwinding, exceptions and inheritance, standard library exception hierarchy	
12-13	**From C++ to Java** Comparing the OOP with Java to C++ in the use of objects, string operations, references, memory management, error handling, etc. The Java deployment model.	
14	**Review for exam**	

Contribution of course to meet the professional component:
This course prepares students with fundamental knowledge and experiences to constructing a language processor.

Relationship to CS program objectives and outcomes:
This course primarily contributes to the Computer Science program outcomes that develop student abilities to:
(a) an ability to apply knowledge of mathematics, science, and engineering.
(c) an ability to design a system, component, or process to meet desired needs within realistic constraints such as economic, environmental, social, political, ethical, health and safety, manufacturability, and sustainability.
(e) an ability to identify, formulate, and solve engineering problems.
(k) an ability to use the techniques, skills, and modern engineering tools necessary for engineering practice.

The course secondarily contributes to the Computer Science program outcomes that develop student abilities to:
(l) an ability to use the computer/IT tools relevant to the discipline along with an understanding of their processes and limitations.

Relationship to CS program criteria:

Criterion	DS	PF	AL	AR	OS	NC	PL	HC	GV	IS	IM	SP	SE	CN
Scale: 1 (highest) to 4 (lowest)		4	2	4	2		4						3	

Discrete Structures (DS), Programming Fundamentals (PF), Algorithms and Complexity (AL), Architecture and Organization (AR), Operating Systems (OS), Net-Centric Computing (NC), Programming Languages (PL), Human-Computer Interaction (HC), Graphics and Visual Computing (GV), Intelligent Systems (IS), Information Management (IM), Social and Professional Issues (SP), Software Engineering (SE), Computational Science (CN).

Course content distribution:

Percentage content for			
Mathematics	Science and engineering subjects	Complementary electives	**Total**
0%	100%	0%	100%

Coordinator:
Prof. Zhiguo Gong

Persons who prepared this description:
Dr. Shirley W. I. Siu

Part B – General Course Information and Policies

2nd Semester 2011/2012

Instructor: Dr. Shirley W. I. Siu
Office: N327B
Office hour: *To be announced*
Phone: 8397 4378
Email: utakosiu@umac.mo

Time/Venue: *To be announced*

Grading distribution:

Percentage Grade	Final Grade	Percentage Grade	Final Grade
100 - 93	A	92 - 88	A–
87 - 83	B+	82 - 78	B
77 - 73	B–	72 - 68	C+
67 - 63	C	62 - 58	C–
57 - 53	D+	52 - 50	D
below 50	F		

Comment:
The objectives of the lectures are to explain and to supplement the text material. Students are responsible for the assigned material whether or not it is covered in the lecture. Students who wish to succeed in this course should read the textbook prior to the lecture and should work all homework and project assignments. You are encouraged to look at other sources (other texts, etc.) to complement the lectures and text.

Homework policy:
The completion and correction of homework is a powerful learning experience; therefore:
- There will be 4-6 homework assignments.
- Homework is due 10 days after assignment unless otherwise noted, no late homework is accepted.
- The course grade will be based on the average of the homework grades.

Course project:
The project is probably the most exciting part of this course and provides students with meaningful experience to design and implement a medium size system applying all the OOP techniques learnt throughout this course:
- You will work with group of four students for the course project.
- The requirements will be announced and discussed in class.
- The project will be presented at the end of semester.

Exam:
One 2-hour mid-term exam will be held during the semester. Both the mid-term and the final exam are closed book examinations. There will be occasional in-class quizzes.

Note:
- Check UMMoodle (https://ummoodle.umac.mo/) for announcement, homework and lectures. Report any mistake on your grades within one week after posting.
- No make-up exam is given except for CLEAR medical proof.
- Cheating is absolutely prohibited by the university.

Appendix:

Rubric for Program Outcomes

Rubric for (a)	5 (Excellent)	3 (Average)	1 (Poor)
Understand the theoretic background	Students understand theoretic background and the limitations of the respective applications.	Students have some confusion on some background or do not understand theoretic background completely.	Students do not understand the background or do not study at all.
Rubric for (c)	**5 (Excellent)**	**3 (Average)**	**1 (Poor)**
Design capability and design constraints	Student understands very clearly what needs to be designed and the realistic design constraints such as economic, environmental, social, political, ethical, health and safety, manufacturability, and sustainability.	Student understands what needs to be designed and the design constraints, but may not fully understand the limitations of the design constraints.	Student does not understand what needs to be designed and the design constraints.
Rubric for (e)	**5 (Excellent)**	**3 (Average)**	**1 (Poor)**
Identify applications in engineering systems	Students understand problem and can identify fundamental formulation.	Students understand problem but cannot apply formulation, or cannot understand problem.	Students cannot identify correct terms for engineering applications.
Modeling, problem formulation and problem solving	Students choose and properly apply the correct techniques.	Students model correctly but cannot select proper technique or model incorrectly but solve correctly accordingly.	Students at loss as to how to solve a problem.
Rubric for (k)	**5 (Excellent)**	**3 (Average)**	**1 (Poor)**
Use modern principles, skills, and tools in engineering practice	Student applies the principles, skills and tools to correctly model and analyze engineering problems, and understands the limitations.	Student applies the principles, skills and tools to analyze and implement engineering problems.	Student does not apply principles and tools correctly and/or does not correctly interpret the results.
Rubric for (l)	**5 (Excellent)**	**3 (Average)**	**1 (Poor)**
Use modern computer/IT tools relevant to the discipline	Student uses computer/IT tools relevant to the engineering discipline, and understands their limitations.	Student uses computer /IT tools relevant to the engineering discipline.	Student does not use computer/IT tools relevantly, and does not understand their limitations.