MEC 510: Object-Oriented Programming for Scientists and Engineers

Xiaolei Yang

Fall, 2018

E-mail: xiaolei.yang@stonybrook.edu
Phone: (631) 632-9164
Office Hours: Tu 2:00PM - 4:00PM
Class Hours: Tu 4:00PM - 6:50PM
Office: 161 Light Engineering
Class Room: Earth & Space 131

Course Description

Practical introduction to C++ and object-oriented programming for a first programming course for scientists and engineers. Covers basics of application software development such as problem decomposition, structure charts, object modeling, class diagrams, incremental code building, and testing at a beginner's level. Features the concepts of abstract data types (ADT), encapsulation, inheritance, composition, polymorphism, operator and function overloading besides studying UML (Unified Modeling Language) as a graphical representational design technique. The course follows the evolution of programming ideas from the use of a single function to the use of structural charts and functions to modularize and finally to the use of object-oriented programming.

Prerequisites

B.S. in science or engineering.

Required Materials

Textbook

- Object-Oriented Programming in C++ (4th Edition). Author: Lafore, Robert. Publisher: Sams Publishing.

Compilers

- IDE (integrated development environment): 1) Microsoft Visual studio; 2) Xcode.
- Without IDE: Install GNU or Intel C++ compiler, edit your code using vim or emacs, compile and execute your code on the command line.

Visualization

- Microsoft Excel.
- Matlab.

Course Learning Outcomes/Objectives

Upon successful completion of this course, the students will be able to:

1. Understand the idea of object-oriented programming (OOP).
2. Know C++ programming basics.
3. Use loops, decision-making statements, arrays, pointers and functions.
4. Use structures to create user-defined data types.
5. Use classes for organizing C++ programs.
6. Overload operators and perform type conversion.
7. Use stream class for file I/O.
8. Handle exceptions in a C++ code.
9. Use standard template library.
10. Solve simple problems in research and engineering applications using OOP.

Grading Policy

The typical grading scale on blackboard will be used. The grade will count the assessments using the following proportions:

- **40%** of your grade will be determined by 8 homework assignments (5% each).
- **25%** of your grade will be determined by the mid-term exam.
- **30%** of your grade will be determined by the final project.
- **5%** of your grade will be determined by class attendance.

Course Policies

During Class

Computers will be allowed in class. Please refrain from using computers for anything but activities related to the class. Phones are prohibited.

Make-up Exams and Late Homework

Make-up exams are considered only for students who provide documentation of a compelling reason for missing the exam, such as doctor's certificate for medical emergency. No make-up exams for reasons that are within your control, such as pre-arranged vacation, travel, or other engagements.

Late homework will be accepted for no penalty if a valid excuse is communicated to the instructor before the deadline. After the deadline, homework will be accepted for a 50% deduction to the score up to 2 days after the deadline. After this any homework handed in will be given 0.

Attendance

For complete attendance policy, please see https://www.stonybrook.edu/commcms/registrar/policies/tracking.php. Attendance is expected in all lectures. Valid excuses for absence will be accepted before class. In extenuating circumstances, valid excuses with proof will be accepted after class.

Student Accessibility Support Center Statement

If you have a physical, psychological, medical or learning disability that may impact your course work, please contact Student Accessibility Support Center, ECC (Educational Communications Center) Building, Room 128, (631)632-6748. They will determine with you what accommodations, if any, are necessary and appropriate. All information and documentation is confidential.

Students who require assistance during emergency evacuation are encouraged to discuss their needs with their professors and Student Accessibility Support Center. For procedures and information go to the following website: http://www.stonybrook.edu/ehs/fire/disabilities.

Academic Integrity Statement

Each student must pursue his or her academic goals honestly and be personally accountable for all submitted work. Representing another person's work as your own is always wrong. Faculty is required to report any suspected instances of academic dishonesty to the Academic Judiciary. Faculty in the Health Sciences Center (School of Health Technology & Management, Nursing, Social Welfare, Dental Medicine) and School of Medicine are required to follow their school-specific procedures. For more comprehensive information on academic integrity, including categories of academic dishonesty please refer to the academic judiciary website at http://www.stonybrook.edu/commcms/academic_integrity/index.html

Critical Incident Management

Stony Brook University expects students to respect the rights, privileges, and property of other people. Faculty are required to report to the Office of University Community Standards any disruptive behavior that interrupts their ability to teach, compromises the safety of the learning environment, or inhibits students' ability to learn. Faculty in the HSC Schools and the School of Medicine are required to follow their school-specific procedures. Further information about most academic matters can be found in the Undergraduate Bulletin, the Undergraduate Class Schedule, and the Faculty-Employee Handbook.

Schedule

The schedule is tentative and subject to change.

Week 01, 08/27 - 08/31: Introduction; C++ programming basics

- Importance of computing in engineering applications.
- Idea of object-oriented programming (OOP).
- Basic ideas of objects and classes.
- Relation of C++ and C.
- Unified Modeling Language (UML).
- A major building block of C++: function.
- Output and input: <<, >>.
- Data types: char, int, float, double, bool.
- Keyword: const.
- Arithmetic operators (+, -, *, and /) and remainder operator (%).
- Preprocessor directives: #include directive, #define directive.
- Use a library in a program.

Week 02, 09/03 - 09/07: Loops, decisions; Structures

- Relational operator.
- Loops: for loop, while loop, do loop.
- Decision-making statements: if, if...else, else if, swith.
- Logical operator, AND, OR, NOT.
- break statement, continue statement, goto statement.
- Precedence: unary, arithmetic, relational, logical, conditional, assignment.
- *Structures* for grouping several data items together to form a single entity.
- *Enumeration*: a programmer-defined type that is limited to a fixed list of values.

Week 03, 09/10 - 09/14: Functions; Objects, classes

- Functions: help organize programs, reduce program size, can be executed from other parts of the program.
- Function *declarations*, function *calls*, function *definitions*.
- Pass *by value*, pass *by reference*.
- *Overloaded* functions, *inline* functions.
- *Storage class*: local variables, *global* variables, *static* local variables.
- A class is a specification or blueprint for a number of objects.
- Objects consist of both data and functions that operate on that data.
- Keywords: private, public.
- Member function, constructor, destructor.

Week 04, 09/17 - 09/21: Arrays, strings

- Arrays contain a number of data items of the same type.
- The address of an array can be used as an argument to a function; the array itself is not copied.
- C-strings are arrays of type char.
- The string class.

Week 05, 09/24 - 09/28: Operator overloading

- Keyword operator used for overloading an operator.
- *Type conversion*: routine in destination, routine in source.

Week 06, 10/01 - 10/05: Inheritance

- Inheritance: *derived class, base class*.
- Accessibility: protected.
- *Multiple inheritance*.

Week 07, 10/08 - 10/12: No class

Week 08, 10/15 - 10/19: Mid-term

Week 09, 10/22 - 10/26: Pointers

- Everything in the computer's memory has an address.
- Addresses are pointer constants.
- The *address-of* operator: &.
- Pointers are variables that hold address values.
- Use * (*pointer to*) to define pointers.
- Dereference operator * (*contents of the variable pointed to by*).
- Array elements can be accessed using array notation with brackets or pointer notation with an asterisk.
- Pass the address of a variable to a function.
- A string constant can be defined as an array or as a pointer.
- The new operator, delete operator.
- Access operator ->.
- Pointers to pointers.

Week 10, 10/29 - 11/02: Virtual functions

- Virtual functions provide a way for a program to decide while it is running what function to call.
- The use of one or more pure virtual functions in a class makes the class *abstract*.
- A friend function can access a class?s private data.
- A static function is one that operates on the class in general, rather than on objects of the class.
- Overloaded assignment operator.
- The this pointer.
- The dynamic_cast operator.
- The typeid operator.

Week 11, 11/05 - 11/09: Streams and files; Multi-file system

- Stream classes for I/O.
- File I/O: ofstream, ifstream, fstream.
- Handle I/O errors.
- read(), write() functions.
- Overload extraction operator $>>$ and the insertion operator $<<$.
- Class declarations in an .H header file.
- Member function definitions in an .OBJ object file or .LIB library file.

Week 12, 11/12 - 11/16: Templates and exceptions; Standard template library

- Templates allow you to generate a family of functions, or a family of classes, to handle different data types.
- Exceptions are a mechanism for handling C++ errors in a systematic, OOP-oriented way.
- The class member function discovers the error and throws an exception, which is caught by the program using the class, in exception-handler code following the try block.
- Standard template library (STL): containers, algorithms, and iterators.

Week 13, 11/19 - 11/23: No class

Week 14, 11/26 - 11/30: Final project presentation

Week 15, 12/03 - 12/07: Final project presentation

Week 16, 12/10 - 12/14: Final project due

Coding Rules

- Write human-friendly code: programming is like writing a poem.
  - Follow consistent name conventions. If work collaboratively, discuss name conventions before start.
  - The variable and function names should describe what the code does perfectly.
  - Use indentation and line breaks to reveal the code structure clearly.
  - Write comments to describe what is happening, how it is being done, what parameters mean, and any restrictions or bugs.
    - A brief introductory description of the program is needed at the beginning of the program
    - All functions must have a description of what they are doing
    - Block comments (/* xxxx */) describe a function or a block of the code. They should be aligned with that block of the code.
    - Short comments are added using "//"
  - Avoid hard-coding (put specific numbers in the code)
- Code organization. Separate code into different files according to their primary functions.
- Planning before coding.