CSIS-123A: C++ PROGRAMMING - LEVEL 2

Effective Term
Fall 2023

BOT Approval Date
06/23/2022

Discipline
CSIS - Computer Sci/Info Systems
Course Number
123A

Course Title
C++ Programming - Level 2

Short Title
C++ Programming - Level 2

Credit Status (CB 04)
D - Credit - Degree applicable

Units

Lecture Units
3.00

Total Units
3.00

Hours

Lecture Contact Hours
48-54
Out of Class Hours Lecture
96-108

Total Contact Hours
48 - 54

Total Out of Class Hours
96 - 108

Total Student Learning Hours
144 - 162

Distance Education
Yes

Distance Education

Distance Education Type
Both Fully Online and Hybrid Online

Fully Online Delivery Requirements:
a. Students must be notified via the college schedule of classes and the syllabus for the class if proctored tests are required for this course
b. Any planned face-to-face meetings, such as an orientation or study session, must be optional

c. The MSJC curriculum committee requires the use of accessible, asynchronous discussion as a component of every fully online course

Are you using publisher content?

No

Regular Substantive Interaction

Instructor-to-student, student-to-student, and student-to-content contact shall be distributed in a manner that will ensure regular substantive interaction (RSI) is maintained not only weekly but also for the duration of the course. Instructor interactions should be equivalent in time, quality and consistency to the engagement students would receive in a face-to-face course—for example, through announcements, discussion participation, personalized feedback, video messages, or other instructional communications. Instructors are expected to be responsive to student inquiries within a 24-72 hour time frame excluding holidays. RSI will be maintained through a variety of methods: • Orientation - Students must be oriented to the online and face-to-face portions at the start of the course. • Announcements - Announcements using the course management system must be posted at least weekly to keep students current on course events, due dates, materials, etc. • Discussion Forums - Content-related discussion forums within the designated CMS will include appropriate instructor participation and will be initiated and maintained, with timely feedback provided. Instructors are required to monitor the discussions and related activities to provide guidance and direction. Open-ended discussion boards may be used to further learning and interaction. • MSJC Communication – Instructor will use MSJC-administered communication tools (such as email or the designated CMS communication tools) to regularly communicate with and provide ease of access to students. • Timely Feedback - Timely feedback on student work will be provided by the instructor, which may include comments and/or rubrics, within the Grades area of the designated CMS. • Scheduled Meetings – Hybrid and synchronous courses will meet at regularly scheduled times. ADA Compliance - All course interactions and content must be designed with accessibility in mind and meet ADA requirements.

Catalog Description

This course presents advanced programming concepts in the C++ programming language. Advanced aspects of program design methodologies will be studied, evaluated, and applied in the design of complex C++ programs. C++ features that will be covered include classes and data abstraction, operator overloading, inheritance, polymorphism, templates, exception handling, and file structures.

Requisites

Prerequisite(s)

Prerequisite(s) (must be taken before)
CSIS-113A (with a grade of C or better).

Codes

TOP Code (CB 03)

0707.10* - *Computer Programming

CIP Code

11.0201 - Computer Programming/Programmer, General.

Repeatability Code

No

Grading Option

Letter grade OR P/NP

Learning Objectives

Learning Objectives

Learning Objective
1. Compose classes that use proper Object-Oriented Programming techniques
2. Explain how classes become objects
3. Create class constructors that properly initialize members of the class
4. Organize classes so that they use proper data encapsulation techniques
5. Construct O-O C++ programming solutions for re-use that incorporate encapsulation, data abstraction, and information hiding.

6. Develop functions that use operator overloading to simply object use.
7. Compose programs that use pointers to dynamically allocate memory.
8. Construct multiple-file, or multiple-module C++ programming solutions that use class hierarchies, inheritance, and polymorphism to reuse existing design and code.
9. Develop an exception handling strategy that utilize try-throw-catch.
10. Integrate templates into classes to create generic data types.
11. Evaluate the varying container types that are part of the STL
12. Identify and assess the benefits of diversity, equity and inclusion in the tech workplace and society at large (e.g. teamwork, creativity, innovation, competitive advantage, etc.).
13. Identify examples and potential sources of algorithmic bias in computer programming that can reinforce biases of race, gender, sexuality, and ethnicity in modern software systems.

Learning Outcome Information

Course Learning Outcomes

Learning Outcome
Create a class hierarchy that demonstrates the proper use of encapsulation and data hiding.
Demonstrate the use of templates to create generic data types
Construct and error handling scheme which separates the error handling from the code that functions normally
Demonstrate the use of multiple source code files as a way of organizing code.
Identify options and generate alternative approaches.
Engage in multiple and simultaneous outcomes and activities.

Program Learning Outcomes

Learning Outcome
Research and apply industry reference models and best practices in order to improve process designs.
Apply systems concepts in the investigation, evaluation, and resolution of information technology problems.

Institutional Learning Outcomes

Critical Thinking: The student will analyze problems, gather and synthesize relevant information, evaluate ideas, information, and evaluate alternative points of view to create, innovate and implement effective solutions.

Content

Course Lecture Content

1. Review of Object-Oriented Principles
   a. Introduction
   b. Objects
   c. Classes
   d. Abstract class
2. Extending Classes and Inheritance
   a. Data abstraction
   b. Class inheritance
   c. Polymorphism
   d. Abstract classes
   e. Casting objects
3. Advanced Data Structures
4. Exception Handling
   a. Types of exceptions
   b. Dealing with exceptions
   c. Exception objects
5. Streaming I/O

a. Defining a file
b. Character output streams
c. Byte output streams
d. Character input streams
e. Byte input streams

6. Utility Classes
   a. Collection classes
   b. Data structures
7. Using Preprocessor Directives
   a. Including files
   b. Conditional compilation
   c. Using macros
8. Testing Strategies
9. Advanced Object-oriented design principles
   a. Templates
   b. Stardard Template Library
10. Life-Long Learning
    a. Web resources
    b. Relevant publications
    c. Other Educational opportunities
11. Career Options
    a. How this course relates to standard job classifications
    b. Relation to industry and vendor certification paths
    c. Overview of technical skills
    d. Overview of soft skills
12. Diversity, Equity, and Inclusion Issues related to Information Technology and Computer Science.
    a. Digital Divide in Engineers and the Engineering workplace
    b. Racial and Gender Bias in Artificial Intelligence within large corporate systems
    c. The gender gap in industry programming and programmers

Methods of Instruction

Method

Discussion

Integration

Weekly discussion will be conducted as a means to compare and contrast the use and implementation of inheritance and an "isa" relationship, Unintended and unexpected outcomes (algorithmic biases), development of programs that use operator overloading as a public interface to private data, racial and gender bias in Artificial Intelligence and the proper uses for encapsulation and data abstraction.

DE Adaptations for MOI

Weekly discussion will be conducted as a means to compare and contrast the use and implementation of inheritance and an "isa" relationship, the unintended and unexpected outcomes (algorithmic biases) , development of programs that use operator overloading as a public interface to private data, racial and gender bias in Artificial Intelligence and the proper uses for encapsulation and data abstraction will be administered using the discussion tool supplied as part of the course management system.

Method

Lecture

Integration

Lecture, with supporting visual materials (presentation or multimedia) will introduce conceptual and practical skills such as reading and writing from data files, demonstrate the proper use of data abstraction and encapsulation, the digital divide, using inheritance to produce a class that has an "isa" relationship, and create programs that use operator overloading as public interfaces to private data.

DE Adaptations for MOI

Lecture, with supporting visual materials (presentation or multimedia) will introduce conceptual and practical skills such as reading and writing from data files, demonstrate the proper use of data abstraction and encapsulation, the digital divide, using inheritance to produce a class that has an "isa" relationship, and create programs that use operator overloading as public interfaces to private data. Lectures will be delivered using video capture software to embed voice into lecture slides. Internet technologies HTML, and CSS will be used to deliver additional lecture notes and sample materials. All lecture material will be located within the course management system.

Method

Observation and Demonstration

Integration

Guided practice and demonstration will illustrate lecture principles and reading assignments, with a focus on fundamental C++ object-oriented programming concepts and applying them to problem solving and potential solutions to homework assignments. Guided practice will include coding and implementing programs to read and write from data files, using inheritance to form an "isa" relationship between classes, the proper use of data abstraction and encapsulation, the gender gap in programming and programmers and using operator overloading to create public interfaces to private data.

DE Adaptations for MOI

Guided practice and demonstration will illustrate lecture principles and reading assignments, with a focus on fundamental C++ object-oriented programming concepts and applying them to problem solving and potential solutions to homework assignments. Guided practice will include coding and implementing programs to read and write from data files, using inheritance to form an "isa" relationship between classes, the proper use of data abstraction and encapsulation, the gender gap in programming and programmers and using operator overloading to create public interfaces to private data. Guided practice and demonstrations will be accomplished by using video / audio capture software (e.g., Camtasia) that will permit instructor demonstration to be captured and deployed. Hands-on activities can be similarly accommodated and supplemented with the use of software simulations and delivered via the course management system.

Methods of Evaluation

Method

Other

Integration

Participation in discussion and guided practice will be required. Topics to be covered will include the development and implementation of algorithms that read and write to data files; the proper use of data abstraction and encapsulation; the use of operator overloading to create a public interface to private data; the gender gap in programming and programmers; the implementation of inheritance to form an “isa” relationship. Guided practice / discussion projects will be evaluated based on the program running to completion without error or warning; how well directions are followed; program correctness; proper input and output.

DE Adaptations for MOE

Participation in discussion and guided practice will be required. Topics to be covered will include the development and implementation of algorithms that read and write to data files; the proper use of data abstraction and encapsulation; the use of operator overloading to create a public interface to private data; the gender gap in programming and programmers; the implementation of inheritance to form an “isa” relationship. Guided practice / discussion projects will be evaluated based on the program running to completion without error or warning; how well directions are followed; program correctness; proper input and output will be submitted via the assignment editor of the course management system. Optional methods of delivering guided practice are through the use of the discussion, blog, and wiki tools of the course management system. Feedback will be provided to students via the course management system within five days of submissions and a rubric will be utilized to provide students with evaluation criteria.

Method

Exams/Tests

Integration

A midterm and final exam will be given. The exams will be composed of short answer questions and small programming examples designed to evaluate the mastery of such topics as the digital divide; reading and writing from data files; using data abstraction and encapsulation; use of operator overloading to create public interfaces to private data; and the proper use of inheritance to form an "isa" relationship. Exams will be evaluated based on the program running to completion without error or warning; how well directions are followed; program correctness; proper input and output.

DE Adaptations for MOE

A midterm and a final exam to evaluate the students understanding of the digital divide; the use of inheritance to create an "isa" relationship; the development and implementation of algorithms that read and write to data files; the proper use of data abstraction and encapsulation; using operator overloading to create a public interface to private data will submitted via the assignment editor of the course management system. Feedback will be provided to students via the course management system within five days of submissions and a rubric will be utilized to provide students with evaluation criteria.

Method

Homework

Integration

Weekly programming assignments will be given that apply important concepts and skills to the student's experience outside the classroom. The focus might be on the development and implementation of algorithms that read and write to data files; the proper use of data abstraction and encapsulation; unintended and unexpected outcomes (algorithmic biases) ; the use of operator overloading to create a public interface to private data; or the implementation of inheritance to form an "isa" relationship. Weekly assignments will be evaluated based on the program running to completion without error or warning; how well directions are followed; program correctness; proper input and output.

DE Adaptations for MOE

Students will complete assignments consisting of the use of inheritance to create an "isa" relationship; the development and implementation of algorithms that read and write to data files; the proper use of data abstraction and encapsulation; unintended and unexpected outcomes (algorithmic biases) ; using operator overloading to create a public interface to private data. Assignments will be submitted using the assignment editor of the course management system. Optional methods for submitting assignments are through the use of the discussion, blog, and wiki tools of the course management system. Feedback will be provided to students via the course management system within five days of submissions and a rubric will be utilized to provide students with evaluation criteria.

Assignments

Assignment Type

Writing

In-class and/or outside of class

Outside of class

Formative or Summative

Formative

Assignment

Create a class called Byte that will store one byte of data. You should create a header called Byte.h where the class definition will be placed. You should also have a corresponding .cpp file called Byte.cpp

Private Section:

From the supplemental material you should have observed that a byte consists of 8-bits. To store these 8-bits of information we are going to need an array. In the data section of the class create an array called bits that will hold 8-integer values to store the 1's and 0's for the Byte.

Create the following utility function:

int bitsToInt() - this function will return the bit values stored in the data section as an int (whole number) value.

Public Section:

Create the following functions:

void setValue(int value) - this function will take as an argument an int type. The argument is going to be the value we want to set our bits to. To set the bits we are going to have to use some bitwise operators. Here is how it works.
A value like 65 has a binary value of
0 1 0 0 0 0 0 1
This will require you to create a loop and use AND along with the left shift operators to determine the value of each bit and store it in the array.
int at(int index) - takes as an argument an int value called index. This function will return the bit located in the array at index. This value should be returned as an int.
string toString() - returns the array as a string. Please note that with binary values they are in the opposite order from the way we are storing them. The array has index 0 going from left to right and binary numbers has bit 0 to the right going right to left. This means you will have to reverse the array when creating the string.
int toInt() - This function will return the value in the array as a whole number. This function should call the private function bitsToInt.

How assignment will be adapted in online format

The assignment will be created using the assignment editor found in the course management system. The assignment will be distributed and collected using the assignments area of the course management system. Feedback will be provided to students via the course management system within five days of submissions and a rubric will be utilized to provide students with evaluation criteria.

Assignment Type

Reading

In-class and/or outside of class

Outside of class

Formative or Summative

Summative

Assignment

Read the chapter and supplemental lecture material on operator overloading. Once you have read all material watch all videos that pertain to operator overloading. Once you have a good understanding of operator overloading go to your Byte class and demonstrate your knowledge by overloading the following operators:
+
-
*
/
=
==
!=
Each of these should work on complex (class) types as arguments.

How assignment will be adapted in online format

This assignment will be distributed and collected through the assignments area of the content management system. Feedback will be provided to students via the course management system within five days of submissions and a rubric will be utilized to provide students with evaluation criteria.

Assignment Type

Writing

In-class and/or outside of class

In-class
Outside of class

Formative or Summative

Formative

Assignment

For this assignment you are going to be asked to do some analysis on labor market data. Go to the labor department website for the state of California. Find the overall employment numbers for the last ten years of an IT field of your choosing. Select any race, or gender of a historically marginalized group that interests you and collect the jobs numbers for the last 10 years.
Create a class called LaborMarketData that has a data section that has the following fields
Year
Field
Gender
Race
TotalJobs
For this class create appropriate accessor and mutator functions. If you are looking not looking at Gender set the Gender field to undefined.
Create an array of these classes initialized with the data you collected from the labor market website.
Output the race, gender and then using a loop compute and output the following:
Total jobs
Average number of jobs for the last 10 years
How much the labor market is improving or decreasing by

How assignment will be adapted in online format

This assignment will be distributed and collected through the assignments area of the content management system. Feedback will be provided to students via the course management system within five days of submissions and a rubric will be utilized to provide students with evaluation criteria.

Course Materials

Textbooks

The C++ Programming Language, 4th Edition
Bjarne Stroustrup
Addison-Wesley 2022
ISBN-13: 978-0321958327
ISBN-10: 0321958322

Class Size Information

Class Size

35

Class Size Category

B - Facilitated learning 35

Diversity and Inclusion

Explain how diversity and inclusion(e.g., gender, culture/race, sexuality, etc.) is infused into the course.

Projects and assignments in Computer Science should include datasets and case studies that address diversity. Instruction should acknowledge the challenges that technology poses to diversity and equity (i.e. Digital Divide, racial and gender bias in Artificial Intelligence, etc.), as well as the opportunities that technology affords in confronting and overcoming racial, gender, and other bias. Diversity and inclusion is infused in the course through an acknowledgement of biases and prejudices that may exist, and a holistic view that each individual can make a positive impact in combating bias and prejudice in the workplace and society in their sphere of influence.

Describe how assignments, instruction, evaluation, course content, and interactions are contextualized for diverse learners promoting an equitable course.

A safe space will be created in the classroom to empower students to openly discuss equitable, inclusive, and diverse ideas. The safe space will be created by the development of appropriate language use, encouraging open and honest dialog, empowering students to share without ridicule, and by processing information shared by reconnecting it to the course content.

Assignments are constructed with flexibility and variety in mind to allow students to show what they have learned regardless of the academic strengths or familiarity with specific assignment types. An aspect of the assignments that will promote equity and inclusion is to allow the student to select from several different formats which will still meet the goal of the assignment.

Interactions

Explain how the course will incorporate student-to-student interaction.

Each lecture of the course will have several small programs to solve. Students are encouraged to work in groups to solve the problem as a team. In addition, two group projects will be assigned where students will be required to work with each other to derive a solution. Each unit of the course has its own discussion board where students can go to interact and help each other understand topics such as the differences between complex and primitive data types, branching, looping, pointers, arrays, public and private interfaces.
To further promote student to student interactions a discussion board will be created in the course management system titled “Student Lounge” where students can go to converse and share ideas with each other.

Explain how the course will incorporate instructor-to-student interaction.

A welcome letter will be sent to the students the first day of class which not only welcomes students to the class but also gives details of how to navigate the course. The announcements area of the course management system will be used to send weekly announcement about important concepts and due dates. A welcome to class announcement will be provided which contains a video that explains the details of the class.
A separate discussion board will be created in the course management system for each unit of the class. These discussion boards are a place where students can get feedback from the instructor on such topics as the difference between primitive and complex data types, arrays, pointers, functions, branching, and repetition.
In addition, a general questions discussion board will be provided where students are encouraged to speak with the instructor on any programming related topic.
Students are encouraged to email the instructor with any questions they may have that they do not want publicly published.

Explain how the course will incorporate student-to-content interaction.

All courses have an online lecture component. The content of the course has several “test your knowledge” pieces where students get instant feedback about their understanding of the material presented. In addition, a series of "video shorts" are given that reinforce topics given. Practice exams will be created as a way for students to get an understanding of the type of questions be asked on the exams and a way to practice skills before taking the exam. At the end of each lecture component several practice problems will be presented to give students a better understanding of the material offered in that unit. All practice problems will have the solution with the so students can reflect on their coding techniques and logic used to solve the problem.

Explain how the course will incorporate student-to-self interaction (metacognitive).

The course emphasizes the idea of thinking logically and to consider other logical interpretations of problem solving using a programming language. Through discussion and assignment, we ask students to pose questions about problem solving and how they approach problem solving through the application of a programming language. Through self-assessment we ask students to reflect on where they are successful and where they are having difficulty and how they can improve upon both their syntactical and logical skills.
We encourage students to think outside the box and investigate other problem-solving techniques from outside sources as a means to not only improve but to understand problem solving from other points of view.

Key: 1048