Discipline: Computer Information Systems

Originator: Mark Lehr

RIVERSIDE COMMUNITY COLLEGE DISTRICT
INTEGRATED COURSE OUTLINE OF RECORD

COMPUTER INFORMATION SYSTEMS 5

CIS-5 : Programming Concepts and Methodology I:C++

College: RIV
Lecture Hours: 54.000
Lab Hours: 54.000
Units: 4.00
Letter Grade

Course Description

Prerequisite: None

Advisory: CIS-1A

Course Credit Recommendation: Degree Credit

Introduction to the discipline of computer science incorporating problem definitions, algorithm development, and structured programming logic for business, scientific and mathematical applications. The C++ language will be used for programming problems. 54 hours lecture and 54 hours laboratory.

Short Description for Class Schedule

Introduction to computer science with programming for gaming, business, scientific and mathematical applications using C++. (Same as CSC-5)

Entrance Skills:

Before entering the course, students should be able to demonstrate the following skills:

1. **Identify the fundamental computer concepts and terminology used for input, processing, output, and storage.**
2. **Identify the key features of a variety of software such as operating systems, communications and graphics.**
3. **Create electronic presentations with presentation graphics.**
4. **Demonstrate the principles of Internet research.**
5. **Understand the principles of computer security, ethics and privacy.**

Student Learning Outcomes:

Upon successful completion of the course, students should be able to demonstrate the following skills:

1. **Describe the software development life-cycle**
   - **Critical Thinking:** Students will be able to demonstrate higher-order thinking skills about issues, problems, and explanations for which multiple solutions are possible. Students will be able to explore problems and, where possible, solve them. Students will be able to develop, test, and evaluate rival hypotheses. Students will be able to construct sound arguments and evaluate the arguments of others.
   - **Communication Skills:** Students will be able to communicate effectively in diverse situations. They will be able to create, express, and interpret meaning in oral, visual, and written forms. They will also be able to demonstrate quantitative literacy and the ability to use graphical, symbolic, and numerical methods to analyze, organize, and interpret data.
   - **Information Competency & Technology Literacy:** Students will be able to use technology to locate, organize, and evaluate information. They will be able to locate relevant information, judge the reliability of sources, and evaluate the evidence contained in those sources as they construct arguments, make decisions, and solve problems.
2. **Describe the principles of structured programming and be able to design, implement and test structured programs.**
   - **Critical Thinking:** Students will be able to demonstrate higher-order thinking skills about issues, problems, and explanations for which multiple solutions are possible. Students will be able to explore problems and, where possible, solve them. Students will be able to develop, test, and evaluate rival hypotheses. Students will be able to construct sound arguments and evaluate the arguments of others.

- **Information Competency & Technology Literacy:** Students will be able to use technology to locate, organize, and evaluate information. They will be able to locate relevant information, judge the reliability of sources, and evaluate the evidence contained in those sources as they construct arguments, make decisions, and solve problems.

3. **Explain what an algorithm is and its importance in computer programming.**
   - **Critical Thinking:** Students will be able to demonstrate higher-order thinking skills about issues, problems, and explanations for which multiple solutions are possible. Students will be able to explore problems and, where possible, solve them. Students will be able to develop, test, and evaluate rival hypotheses. Students will be able to construct sound arguments and evaluate the arguments of others.
   - **Communication Skills:** Students will be able to communicate effectively in diverse situations. They will be able to create, express, and interpret meaning in oral, visual, and written forms. They will also be able to demonstrate quantitative literacy and the ability to use graphical, symbolic, and numerical methods to analyze, organize, and interpret data.
   - **Information Competency & Technology Literacy:** Students will be able to use technology to locate, organize, and evaluate information. They will be able to locate relevant information, judge the reliability of sources, and evaluate the evidence contained in those sources as they construct arguments, make decisions, and solve problems.

4. **Summarize the evolution of programming languages illustrating how this history has led to the paradigms available today.**
   - **Critical Thinking:** Students will be able to demonstrate higher-order thinking skills about issues, problems, and explanations for which multiple solutions are possible. Students will be able to explore problems and, where possible, solve them. Students will be able to develop, test, and evaluate rival hypotheses. Students will be able to construct sound arguments and evaluate the arguments of others.
   - **Communication Skills:** Students will be able to communicate effectively in diverse situations. They will be able to create, express, and interpret meaning in oral, visual, and written forms. They will also be able to demonstrate quantitative literacy and the ability to use graphical, symbolic, and numerical methods to analyze, organize, and interpret data.
   - **Information Competency & Technology Literacy:** Students will be able to use technology to locate, organize, and evaluate information. They will be able to locate relevant information, judge the reliability of sources, and evaluate the evidence contained in those sources as they construct arguments, make decisions, and solve problems.

5. **Use pseudocode, flowcharts, and a programming language to implement, test, and debug algorithms for solving problems. Identify the information requirements, synthesize the algorithmic steps needed to transform the data input into the required output information, and organize the output format to facilitate user communication.**
   - **Critical Thinking:** Students will be able to demonstrate higher-order thinking skills about issues, problems, and explanations for which multiple solutions are possible. Students will be able to explore problems and, where possible, solve them. Students will be able to develop, test, and evaluate rival hypotheses. Students will be able to construct sound arguments and evaluate the arguments of others.
   - **Communication Skills:** Students will be able to communicate effectively in diverse situations. They will be able to create, express, and interpret meaning in oral, visual, and written forms. They will also be able to demonstrate quantitative literacy and the ability to use graphical, symbolic, and numerical methods to analyze, organize, and interpret data.
   - **Information Competency & Technology Literacy:** Students will be able to use technology to locate, organize, and evaluate information. They will be able to locate relevant information, judge the reliability of sources, and evaluate the evidence contained in those sources as they construct arguments, make decisions, and solve problems.

6. **Demonstrate different forms of binding, visibility, scoping, and lifetime management.**
   - **Critical Thinking:** Students will be able to demonstrate higher-order thinking skills about issues, problems, and explanations for which multiple solutions are possible. Students will be able to explore problems and, where possible, solve them. Students will be able to develop, test, and evaluate rival hypotheses. Students will be able to construct sound arguments and evaluate the arguments of others.
   - **Communication Skills:** Students will be able to communicate effectively in diverse situations. They will be able to create, express, and interpret meaning in oral, visual, and written forms. They will also be able to demonstrate quantitative literacy and the ability to use graphical, symbolic, and numerical methods to analyze, organize, and interpret data.
   - **Information Competency & Technology Literacy:** Students will be able to use technology to locate, organize, and evaluate information. They will be able to locate relevant information, judge the reliability of sources, and evaluate the evidence contained in those sources as they construct arguments, make decisions, and solve problems.

7. **Create computer programs using the principles of structured programming and demonstrate the use of an IDE with appropriate libraries. Design, implement, test, and debug programs that use**

fundamental programming constructs: basic computation, simple I/O, standard conditional and iterative structures, and functions.
    - **Critical Thinking:** Students will be able to demonstrate higher-order thinking skills about issues, problems, and explanations for which multiple solutions are possible. Students will be able to explore problems and, where possible, solve them. Students will be able to develop, test, and evaluate rival hypotheses. Students will be able to construct sound arguments and evaluate the arguments of others.
    - **Communication Skills:** Students will be able to communicate effectively in diverse situations. They will be able to create, express, and interpret meaning in oral, visual, and written forms. They will also be able to demonstrate quantitative literacy and the ability to use graphical, symbolic, and numerical methods to analyze, organize, and interpret data.
    - **Information Competency & Technology Literacy:** Students will be able to use technology to locate, organize, and evaluate information. They will be able to locate relevant information, judge the reliability of sources, and evaluate the evidence contained in those sources as they construct arguments, make decisions, and solve problems.
8. **Apply the principles of logical and programming concepts to develop solutions for gaming, business, scientific and mathematical problems.**
    - **Critical Thinking:** Students will be able to demonstrate higher-order thinking skills about issues, problems, and explanations for which multiple solutions are possible. Students will be able to explore problems and, where possible, solve them. Students will be able to develop, test, and evaluate rival hypotheses. Students will be able to construct sound arguments and evaluate the arguments of others.
    - **Communication Skills:** Students will be able to communicate effectively in diverse situations. They will be able to create, express, and interpret meaning in oral, visual, and written forms. They will also be able to demonstrate quantitative literacy and the ability to use graphical, symbolic, and numerical methods to analyze, organize, and interpret data.
    - **Information Competency & Technology Literacy:** Students will be able to use technology to locate, organize, and evaluate information. They will be able to locate relevant information, judge the reliability of sources, and evaluate the evidence contained in those sources as they construct arguments, make decisions, and solve problems.

General Education Outcomes:

- **District General Education -** A2 Language and Rationality - Communication & Analytical Thinking
- **UC/CSU Transfer Course -** Yes

Course Content:

Focus of the course topics is on the fundamentals of programming, problem solving, and software design. Models of applications from the areas of gaming, business, science and mathematics are presented throughout the course.

1. Overview of Programming Languages

    a. History of programming languages,
    b. Brief survey of programming paradigms,
    c. Procedural Languages,
    d. Object Oriented Languages.

    The intent is to; a) summarize the evolution of programming languages illustrating how this history has led to the paradigms available today, and b) identify at least one distinguishing characteristic for each of the programming paradigms covered.
2. Algorithms and problem-solving
    a. Problem solving strategies,
    b. The role of algorithms in the problem solving process,
    c. Implementation strategies for algorithms,
    d. Debugging strategies,
    e. The concept and properties of algorithms.

    The intent is to: a) discuss the importance of algorithms in the problem-solving process, b) identify the necessary properties of good algorithms, c) create algorithms for solving simple problems, d) use pseudocode, flowcharts, and a programming language implementation process, and e) describe strategies useful in debugging.
3. Introduction to C++
    a. Coverage of topics by instructors should include the components of programs such as key words, variables, operators, and punctuation. Programming design tools, such as pseudocode, flowcharts, and hierarchy charts are presented.

b. Students will be guided through C++ data types, identifiers, variable declarations, constants, comments, program output, simple arithmetic operations, and C-string. Programming style conventions and good programming style are modeled. The differences between ANSI standard and pre-standard C++ are explained. Students will write programs that input and handle numeric, character, and C-string data. The use of arithmetic operators and the creation of mathematical expressions are covered in detail, with emphasis on operator precedence
c. The intent amongst other things is to; a) explain the value of declaration models, especially with respect to programming-in-the-large, b) identify and describe the properties of a variable such as its associated address, value, scope, persistence and size, c) discuss type incompatibility, d) demonstrate different forms of binding, visibility, scoping, and lifetime management, and e) defend the importance of types and type-checking in providing abstraction and safety.

4. Decision Structures
    a. Relational operators, relational expressions and how to control the flow of a program with the if, if/else, and if/else if statements are explained. Crucial applications of the conditional operator and the switch statement are covered, such as menu-driven programs and the validation of input.
5. Iteration Structures
    a. Repetition control structures, such as the while loop, do-while loop, and for loop are taught, along with the common uses for these structures. Counters, accumulators, running totals, sentinels, and other application-related topics are modeled.
6. Functions
    a. Students learn how and why to modularize programs, using both void and value returning functions. Argument passing is covered, with emphasis on when arguments should be passed by value versus when they need to be passed by reference. Scope of variables is covered along with a discussion of local versus global variables and static local variables. Overloaded functions are introduced and demonstrated.
7. Arrays
    a. Students will create single and multidimensional arrays. Examples of array processing are provided including examples illustrating how to find the sum, average, highest and lowest values in an array and how to sum the rows, columns, and all elements of a two-dimensional array. Programming techniques using parallel arrays are demonstrated and students are shown how to use a data file as an input source to populate an array. STL vectors are introduced and compared to arrays. The bubble sort, selection sort, linear search, and binary search algorithms are modeled.
8. Pointers
    a. Students are introduced to pointers. Pointers are compared to and contrasted with reference variables. The topics of pointer arithmetic, initialization of pointers, relational comparison of pointers, pointers and arrays, pointers and functions, and dynamic memory allocation are discussed.
9. Structured Data
    a. Students are introduced to abstract data types and how to create them using structures, unions, and enumerated data types. Discussions and examples include using pointer to structures, passing structures to functions, and returning structures from functions.
10. File and I/0 Operations
    a. Sequential access, random access, text, and binary file use is modeled. The various modes for opening files are discussed, as well as the many methods for reading and writing file contents. Advanced output formatting is also covered.

The intent for 4.-10. is to; a) analyze and explain the behavior of programs involving fundamental programming constructs, b) create, modify and expand programs that use standard conditional, iterative control structures and functions, c) design implement, test, debug programs that use the above programming constructs, d) choose the appropriate construct for the solution required, e) apply structure decomposition to break programs into smaller pieces (divide and conquer), f) describe and implement the mechanics of functional parameter passing.

11. Classes
    a. The object-oriented paradigm is introduced. Member variables and functions are discussed. The student learns about private and public access specifications, and reasons to use each. The topics of constructors, overloaded constructors, and destructors are presented. Modeling classes with UML, and how to define the classes in a particular problem is discussed.

Methods of Instruction:

Methods of instruction used to achieve student learning outcomes may include, but are not limited to, the following activities:

- Presentation of class lectures/discussions/demonstrations in order to clarify computer programming, computer problem solving, and software design concepts.
- Presentation of class lectures/discussions/demonstrations in order to clarify the principles of structured programming.
- Web-based/web-enhanced/online/distance learning tasks/activities to reinforce understanding of concepts related to computer programming skills, computer problem solving, and software design.
- Online and Laboratory activities and application assignments in order to address areas of improvement in computer programming, computer problem solving, and software design.

- Projects in order to facilitate and demonstrate the acquisition of skills required to create computer programs.
- Collaborative projects/cooperative learning tasks in order to encourage students to develop and apply computer programming, computer problem solving, software design, and team work skills.

Methods of Evaluation:

Students will be evaluated for progress in and/or mastery of student learning outcomes using methods of evaluation which may include, but are not limited to, the following activities:

- Computer programs designed to demonstrate the acquisition of computer programming, computer problem solving, and software design concepts and skills.
- Quizzes/examinations designed to measure students' degree of mastery of fundamental computer programming and software design concepts and terminology.
- Collaborative projects designed to demonstrate successful understanding and application of computer programming, computer problem solving, software design, and team work skills.
- Computer Laboratory assignments/projects designed to clarify students' individual computer programming, computer problem solving, software design strengths and areas of improvement related to these skills.
- Common final project designed to evaluate students' overall achievement of course objectives in computer programming, computer problem solving, and software design concepts.

Sample Assignments:

Outside-of-Class Reading Assignments

- The primary assignments for this course involve the creation of programs using C++. To support that, students will be assigned textbook reading and/or other resource reading that covers programming concepts and demonstrates usage of C++.

Outside-of-Class Writing Assignments

- Assignments for this course involve writing C++ programming statements to form complete programs that carry out specific tasks. The programs are compiled, debugged and executed to display solutions.
- Students will also be asked to document their work with flowcharts and written explanations that clarify their programming code.

Other Outside-of-Class Assignments

- The primary assignments for this course involve the creation of programs using C++. Additional exercises in the computer lab will entail testing software and troubleshooting coded solutions to problems.

Course Materials:

All materials used in this course will be periodically reviewed to ensure that they are appropriate for college level instruction. Possible texts include the following:

Deitel, H. M. and P. J. Deitel. *C++ How to Program* . 8th Prentice Hall, 2015.
Gaddis, Tony. *Starting Out with C++ from Control Structures to Objects.* 8th Pearson Education, 2014.
Malik, D.S. . *C++ Programming: From Problem Analysis to Program Design.* 7th Course Technology, 2014.
Savitch, W.. *Problem Solving with C++.* 9th Pearson Addison Wesley, 2015.
Gliffy: Online Diagram Software and Flow Chart Software. Software. 2015. https://www.gliffy.com/, Free online accounts available over the internet.
Bloodshed Compiler and IDE. Software. Beta 9.2. http://www.bloodshed.net/, A free downloadable compiler.
MS Visual Studio 2015 . Software. 2015. Microsoft Corporation, Provided via the MSDNAA free downloads..
Cygwin GCC Compiler and Netbeans IDE. Software. 2015. http://www.cygwin.com/ and http://netbeans.org, Free downloads over the internet..
Eclipse IDE for C/C++ Developers. Software. 2015. http://www.eclipse.org/downloads/packages/eclipse-ide-cc-developers/lunar, Free downloads over the internet..
Git and Github. Software. 2015. https://git-scm.com/ and https://github.com/, Free online software and accounts available over the internet.

Codes/Dates:

CB05 MOV Transfer Status: Transfers to Both UC/CSU (A)
CB05 NOR Transfer Status: Transfers to Both UC/CSU (A)

CB05 RIV Transfer Status: Transfers to Both UC/CSU (A)
C-ID#: COMP 122

Board of Trustees Approval Date: 01/16/2016
COR Rev Date: 01/19/2016