CSC 240 - Introduction to Different Programming Languages
Spring 2024 Course Syllabus

Instructor: William Meacham Phone: 480 423-6664
Email: William.Meacham@Scottsdalecc.Edu
Office Hours

Class Format and Time

Title	Section	Days	Time	Location	Start / End Date
CSC240	16765	MW	12 noon - 1:50 p.m..	Live-Online	1/29 - 5/10
CSC240	34459	Online		Online	1/29 - 5/10

Course Description Introduction to procedural (C/C++), applicative (LISP), and declarative (Prolog) languages. Prerequisites: A grade of C or better in CSC205 or permission of Instructor.

MCCCD Official Course Competencies

1. Write C/C++ programs that use branches and loops. (I)
2. Write C/C++ programs that call functions and procedures from the C/C++ standard library. (I)
3. Write C/C++ programs that call functions and procedures from user-defined classes. (I)
4. Describe Object Oriented Design as used in C++. (I)
5. Write LISP programs that use recursion. (II)
6. Describe how LISP programs use backtracking. (II)
7. Write Prolog programs that use recursion. (III)
8. Describe how Prolog programs use backtracking. (III)
9. Describe the advantages and disadvantages of strong or weak typing. (IV)
10. Describe how C/C++ use pointers. (I)

MCCCD Official Course Outline

I. C/C++
- A. Control structures
  - 1. Branches
  - 2. Loops
- B. Recursion
- C. Subroutine and functions
  - 1. Internal
  - 2. Libraries
  - 3. Classes
- D. Object-Oriented Design Concepts
  - 1. Inheritance
  - 2. Overloading
  - 3. Polymorphism
- E. Pointers
  - 1. Dynamic memory allocation
  - 2. Pointer dereferencing
  - 3. Pointer arithmetic

II. LISP
- A. Control structures
- B. Recursion
- C. Backtracking
- D. Execution strategies

III. Prolog
- A. Control structures
- B. Recursion
- C. Backtracking
- D. Execution strategies

IV. Comparison of computer languages
- A. History and development
- B. Strong versus weak typing
- C. Polymorphism

LMS: Course Materials, grades, assignments, slides, etc, will be via Canvas.

Textbooks (Recommended): **Introduction to Programming Languages: Programming in C, C++, Scheme, Prolog, C#, and SOA**, any edition, by Yinong Chen. Note: Older editions will work as well. Or you may find resources online.

Operating System: All Programming Projects will be completed and submitted via an online IDE called Replit.com. Replit provides a Linux Environment with all programming software needed for this class. (There is no cost to use Replit). In some cases, it may be easier to install your own versions of this software on your computer, complete the project there and then port your work to Replit. I recommend that you install your own so that it will be convenient to experiment in your own time and your own space. The good news is that you can get everything for free. Below are the recommended solutions.

C and C++ GNU GCC (GNU Compiler Collection)
https://gcc.gnu.org/

and Replit Online IDE

LISP/Scheme: We will be discussing a variation of traditional LISP called Scheme. The vendor that created the version that was used in the past has given their version a new name: Racket.
http://racket-lang.org/download/

and Replit Online IDE

Prolog: SWI Prolog:
http://www.swi-prolog.org/

Attendance: This class will be delivered in a Hy-Flex Format. Students have the option of attending the class live-online via ZOOM or watching the videos of those classes. The live-online class will open up at 11:55 a.m. and students are asked to be online by 11:58. Attendance is required. To be counted as present, you are required to complete and Demo's we do as a class for each session. You may be withdrawn from the course if you accumulate three or more absences (missing Demo's).

You are responsible for all missed notes, concepts and assignments if you miss class. Check with your classmates for anything you may have missed.

Class Demo's/Homework/Quizzes: (10% of your grade). There will be homework, Quizzes and in-class work throughout the course. In-Class work will consist of programs written in class and will be collected (electronically) periodically. Homework will either be online thru Canvas or will need to be uploaded electronically as a pdf. Written work must be completed on engineering paper. Quizzes will be taken online.

Programming Assignments: (45% of your grade). Programming Assignments will be posted on Canvas. They will be graded according to the Rubric provided. These guidelines will detail rules for writing your program, formatting requirements and submission requirements (how you send your code to me).

Due Dates: Due dates for each assignment (HW, Programs) are on Canvas. Assignments turned in within 2 days of the due date will be penalized 10%. Assignments turned in within 2 weeks of the due date will be penalized 50%. Submissions turned in later than two weeks late or after the final exam will not be accepted.

Exams: (45% of your grade). There will be three exams during the semester, (worth 15%, 15% & 15%). All will be given via Lockdown Browser. Tests will be timed, and will be available from 10 a.m. on Friday to 11:59 p.m. on Sunday for the weekend assigned. There will not be a cumulative final exam. However, if you do not take the third exam, you will be withdrawn from the class (with a grade of 'W'). You will not be able to withdraw from the class after you take the third exam. You are expected to take each test at the scheduled time. If an emergency arises that is beyond your control, you must contact me on or before the day of a test to arrange for a make-up test. Make-up tests are possible only in special circumstances, at the instructor's discretion. All make-ups, if granted, must be completed before the exams are returned to the class. Failure to adhere to this policy may result in a grade of zero for the test missed. You are only allowed **one** make-up test per semester.

Office Hours: There is not a true set time for office hours. I can meet during the afternoon Monday through Thursday and Friday morning. I can also meet Sunday afternoon. Meetings need to be scheduled through Google Calendar. Instructions on how to do so will be provided.

Mandatory Check-In Meetings: You will be required to meet with me for at least one mandatory check-in meeting during the semester. One will be after exam 1 as an opportunity to discuss your academic performance, offer insightful feedback, and address any other related issues during this time. You will need to schedule your virtual or in person meeting using the Google Calendar in your student account. It can be at any time I am shown as available on the calendar.

The other will be during the last two weeks of class. It could be to work together on an assignment or to discuss topics in the class.

Grading:

Point Allocation	
45%	Programming Assignments
45%	Exams
10%	Demo's/Quizzes/Homework

Grading Scale	
A	90% - 100%
B	80% - 89%
C	70% - 79%
D	60% - 69%
F	0% - 59%

Speaking of Ethics… I encourage students to help each other as needed and to get help from our Tutors often. But please **make sure you write your own code on all programming assignments for this class.** You will have to write similar code on all of the tests, so it is important that you have worked through the code. Students handing in code they did not write will have their grade for the course lowered 10% depending on the complexity of the assignment. Students handing in code downloaded from sites such as CourseHero, GitHub or any AI generated code will be given an immediate "F" for the class.

Schedule (Approximate).

	Module 0 and 1 Basic Principles	Module 2 Imperative - C	Module 3 OOP – C++	Module 4 Functional - Scheme	Module 5 Logic - Prolog

Date	Tues	Thurs
	printf("%s %d thru %d\n","September, Weeks ",1, 4);	
1/30 2/1	• Syllabus and Course Structure • Introduction to Replit, Racket and SWI-Prolog • Linux Basics	• Hello World in each of the languages
2/6 2/8	• Different paradigms of programming languages	• Imperative Programming Languages (C and C++) • Input/Output • scanf/printf
2/13 2/15	• Introduction to the structures of programming languages (cont) • Regular Expressions	• Introduction to the Functional Programming Paradigm • Terminology and History • Arithmetic Expressions and Prefix Notation
2/20 2/22	• BNF, Syntax Graphs • Data types and type equivalence • Strong vs Weak Typing	• Introduction to the Logic Programming Paradigm • Terminology and History • Facts, Rules and Goals
	std::cout << "October, Weeks " << 5 << " thru " << 8 << std::endl;	
2/27 2/29	• Program processing: interpretation, compilation • Macro Processing	• Basic data types and data declarations, scope rule and forward declaration • Functions and parameter passing
3/5 3/7	• Recursion	• Arrays, Strings, Built-in Functions, Math • Pointers and Constant
10/19	• Lambda Expressions	• Streams • Command Line Arguments • File Operations
3/19 3/21	• Compound Data Types: Enums, Structs, Unions	• Arrays, Pointers and Structures Combined • Dynamic Memory Allocation
	• (displayln "November, Weeks 9 thru 12")	
3/26 3/28	• Intro to Object Oriented Languages (C++) • Principles and Features of OOP • Input/Output	• Intro to OOP Continued
4/2 4/4	• Memory Management • Constructor, Destructor and Overloading • The String Type	• Inheritance • Multiple Inheritance
4/9 4/11	• Virtual Classes, Virtual Methods and Dynamic Binding • Exception Handling	• Scheme Data Types and Functions • Scope, Define, Let-form • High Order Functions: map, reduce and filter
4/16 4/18	• Structured Data, Pairs, Lists • Use of Quote	• Lists Continued
	week :- write("December, Weeks 13 and 14"), nl.	
4/23 4/25	• Proving Goals, Variables and Complex Goals, Backtracking	• Proving Goals, Variables and Complex Goals, Backtracking • Arithmetic operations
4/30 5/2	• Graphs • Prolog functions and recursive rules	• Structured Data, Pairs and Lists • Lists and List Operations • Flow Control Operations
5/7 5/9	Review and Test 3	

Disclaimer: Students are responsible for the information contained in this syllabus. The information in this syllabus is subject to change based on the discretion of the instructor.