**Course Outline**  **2022 – 2023**

|  |  |
| --- | --- |
| **Course:** | **COSC 292 - Advanced Programming 2** |
| **Course Description:** | You will study structured programming techniques, a procedural language, functions, pointers, file input/output, records and dynamic memory management to create applications. |
| **Pre Requisites:**  **Co Requisites:** | * COSC 286 |
| **Course Hours:**  **Credit Units:** | * 60 hours (4 hours/week x 15 weeks) * 4.0 credit units |
| **Student Assessment:**  **Grade/Passing Grade:**  **PLAR Method:** | * 3 assignments (20%), midterm exam (40%), final exam (40%) * 50% * Not developed |
| **Learning Resources:** | * Müldner, Tomasz. *C for Java Programmers* Addison Wesley, 2000. |
| **Learning Outcomes:** | 1. Design a solution to a problem using structured programming techniques. 2. Create a procedural language application from a structured programming design. 3. Use language libraries and functions when developing a procedural program. 4. Develop function libraries in a procedural language. 5. Develop code to reference and manipulate static memory with pointers. 6. Develop code to reference and manipulate dynamic memory with pointers. 7. Design records for the storage of non-homogeneous data. 8. Develop code to create, manipulate, and store records. |
| **Prepared/Updated by:** **Date:** | **Alex Wang**  Jan 2022 |
| **Approved by Program Head:**  **Date:** | **Joseph Herbert**  Jan 2022 |
| **Approved by Academic Chair:**  **Date:** |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 1. **Design a solution to a problem using structured programming techniques.** | 1. Compare differences between the object oriented and structured programming paradigm. 2. Distinguish between object oriented and structured programming terminology. 3. Explain the design and implementation steps in a structured programming solution to a problem statement. 4. Use design steps to identify functions and solution strategies for a simple problem. 5. Create algorithms for user defined functions from solution strategies. | | 1. Lecture, demonstration and in-class activity. 2. Lecture, demonstration and in-class activity. 3. Lecture, demonstration and in-class activity. 4. Lecture, demonstration and in-class activity. 5. Lecture, demonstration and in-class activity. | |
| Assessment Tools: Hand-in assignments, examinations |  | |  | |
| 1. **Create procedural language application from a structured programming design.** | 1. List data types in the procedural language. 2. State the difference between object encapsulation and file encapsulation. 3. Divide program into header files and module files. 4. Create a make file to compile multiple source code files. 5. Use Make command and a user Make file to compile multiple source code files. | | 1. Lecture, demonstration and in-class activity. 2. Lecture, demonstration and in-class activity. 3. Lecture, demonstration and in-class activity. 4. Lecture, demonstration and in-class activity. 5. Lecture, demonstration and in-class activity. | |
| Assessment Tools: Hand-in assignments, examinations |  | |  | |
| 1. **Use language libraries and functions when developing a procedural program.** | 1. Identify language libraries and their use in procedural language programs. 2. Employ library functions to implement design functions 3. Use I/O library functions to perform input and output. 4. Identify language defined functions in libraries that correlate to design functions. 5. Employ library functions to implement design functions in a procedural program. | 1. Lecture, demonstration and in-class activity. 2. Lecture, demonstration and in-class activity. 3. Lecture, demonstration and in-class activity. 4. Lecture, demonstration and in-class activity. 5. Lecture, demonstration and in-class activity. | |
| Assessment Tools: Hand-in assignments, examinations |  |  | |
| 1. **Develop function libraries in a procedural language.** | 1. Create coded user defined functions from solutions strategies and algorithms. 2. Combine user defined functions into function libraries. 3. Employ user defined library functions in a procedural program. | 1. Lecture, demonstration and in-class activity. 2. Lecture, demonstration and in-class activity. 3. Lecture, demonstration and in-class activity. | |
| Assessment Tools: Hand-in assignments, examinations |  |  | |
| 1. **Develop code to reference and manipulate static memory with pointers.** | 1. Identify the memory requirements of various data types. 2. Generate a paper-based representation of a computer program's memory usage from algorithm or code. 3. Explain indirection and its purpose emphasizing the difference between pointer address and the address it stores. 4. Create a pointer to a fundamental data type. 5. Use the cast operator to assign a fundamental data type to a pointer of dissimilar data type. 6. Explain the connection between arrays and pointers. 7. Manipulate base memory using pointers and pointer arithmetic. 8. Create functions that use pointer parameters and return types. | 1. Lecture, demonstration and in-class activity. 2. Lecture, demonstration and in-class activity. 3. Lecture, demonstration and in-class activity. 4. Lecture, demonstration and in-class activity. 5. Lecture, demonstration and in-class activity. 6. Lecture, demonstration and in-class activity. 7. Lecture, demonstration and in-class activity. 8. Lecture, demonstration and in-class activity. | |
| Assessment Tools: Hand-in assignments, examinations |  |  | |

|  |  |  |
| --- | --- | --- |
| 1. **Develop code to reference and manipulate dynamic memory with pointers.** | 1. Define dynamic memory using paper based model. 2. Allocate and release memory from the heap. 3. Employ memory management techniques. 4. Identify potential memory problems with emphasis on dangling references. 5. Manipulate dynamic memory using pointers and pointer arithmetic. 6. Identify bitwise operators and their purpose. 7. Use bit masks to extract desired bit values from a variable. 8. Define function pointers. 9. Employ a function pointer as an iterator. | 1. Lecture, demonstration and in-class activity. 2. Lecture, demonstration and in-class activity. 3. Lecture, demonstration and in-class activity. 4. Lecture, demonstration and in-class activity. 5. Lecture, demonstration and in-class activity. 6. Lecture, demonstration and in-class activity. 7. Lecture, demonstration and in-class activity. 8. Lecture, demonstration and in-class activity. 9. Lecture, demonstration and in-class activity. |
| Assessment Tools: Hand-in assignments, examinations |  |  |
| 1. **Design records for the storage of non-homogeneous data.** | 1. Employ records in the design of a program. 2. Define a structure. 3. Explain the relationship between a structure, a union and a record. 4. Compare the difference between a structure and an array, class, and fundamental data types. 5. Create a structure containing different data types including pointers and arrays. 6. Create functions to manipulate data stored in a structure. | 1. Lecture, demonstration and in-class activity. 2. Lecture, demonstration and in-class activity. 3. Lecture, demonstration and in-class activity. 4. Lecture, demonstration and in-class activity. 5. Lecture, demonstration and in-class activity. 6. Lecture, demonstration and in-class activity. |
| Assessment Tools: Hand-in assignments, examinations |  |  |

|  |  |  |
| --- | --- | --- |
| 1. **Develop code to create, manipulate, and store records.** | 1. Create programs to manipulate data using pointers to structures. 2. Create programs to store data in arrays of structures. 3. Create programs to read and write data in a structure to binary files. | 1. Lecture, demonstration and in-class activity. 2. Lecture, demonstration and in-class activity. 3. Lecture, demonstration and in-class activity. |
| Assessment Tools: Hand-in assignments, examinations |  |  |