07-Mar-2022

PLAKSHA UNIVERSITY, MOHALI (PUNJAB)

Spring Semester 2021-22

Course Plan

Course: FM 121 Programming and Data Structures

Instructors: Gopinath Kanchi, Manoj Kannan

Teaching Assistant: Raghav Awasty

Credits: 4 (3 for lecture + 1 for lab)

Course Description:

Building on the Python course taught in the first semester, students will be introduced to a deeper examination of the Object Oriented Programming (OOP) paradigm, its differences with other programming paradigms and the trade-offs. The OOP paradigm will be demonstrated through problem solving examples using commonly used data structures. Broadly, the following topics in the course will be emphasized:

- Principles of OOP
- Classes, objects and methods
- Program structure
- Program Complexity, Recursion, Proofs by Induction
- Common Data structures and Algorithms
- Input/output, file handling

Learning Outcomes:

After taking this course, students will have:

1. **Knowledge** of OOP principles and the use of classes, objects and methods. They will also gain familiarity with programming in C++/Java by designing and implementing several abstract and concrete data structures that help in the realization of useful algorithms.
2. **Analytical skills** such as subdividing complex problems into smaller parts, analyse which computational approach works well for what kind of problems, and estimate computational effort required to solve problems. Also, they would know how to apply Big-O asymptotic notation to analyse and select efficient algorithms for solving a given problem with respect to time and memory usage.
3. **Comprehension** of which data structures and class of algorithms are most suited to a problem or a class of problems. Be able to compare the adequacy, computational efficiency, and generalizability of multiple solutions to problems.

Suggested books:

- Prata, Stephen. C++ Primer Plus (6/e). New Delhi: Pearson, 2015.
- Lippman, Stanley B. et. al. C++ Primer (4/e). New Delhi: Pearson, 2007.
- Lafore, Robert. Object-Oriented Programming in C++ (4/e). New Delhi: Pearson, 2008.
- Balagrusamy, E. Object-Oriented Programming with C++ (8/e). Chennai: McGraw Hill Education (India) Pvt. Ltd., 2021.

Detailed Topics:

The delivery and design of this course will build upon the first programming course in Python. Thus the treatment of data types, flow control, exception handling etc. will emphasize how C++/Java differ from Python in these aspects and the implications (and reasons), e.g., type safety. The list of topics to be covered include:

1. **Abstractions:**
   Abstractions in Computer Science: Procedures, Modules, Classes and Abstract Data Types. Type Safety. Motivation for Object Oriented Programming.
2. **Primitive data types and strings:**
   Numeric, and Character types; Operations on numeric, conversion between character and integer types; Range, precision, and rounding errors; Constant data; Operations on strings, Basic string class methods
3. **Control Flow:**
   If, else, switch, for, while statements; Boolean types; Boolean expressions and operations; Single- and multi-dimensional arrays; Arrays as a homogeneous collection
4. **Object Oriented Programming:**
   Classes; Objects; Methods; Constructors; Accessors; Mutators; Returning objects; Encapsulation and type safety
5. **Recursive Data Structures**
   Trees, Heaps and Priority Queues
6. **Inheritance and Polymorphism:**
   Class hierarchies, inherited data and methods; Polymorphism & Dynamic binding; Overloaded and overridden methods; Interfaces; Handling multiple inheritance
7. **Writing Good Programs, Debugging, Testing:**
   Good Programming Hygiene; Appropriate choice of variable and method names; How to design a class hierarchy -- implications of choices; Commenting; Debugging; How to fix and find errors; Use of a debugging tools; Introduction to testing (e.g., JUnit); Using a debugger (to trace a program, watch variables during execution etc.)

8. **Exception handling:**

   Writing robust code that deals with errors and uncertainties through exception handling; Understanding control flow during an exception: handling an exception versus propagating one; Using standard exception classes; Defining your own exception classes

9. **Input/Output (I/O):**

   Dealing with data outside the program: input and output; Dealing with Files (Text, binary); Streaming objects; Exchanging data over the network

10. **Graph Algorithms**

    Details: Shortest Paths, Minimum Spanning Trees (Dijkstra's Algorithm, Prim's Algorithm).

11. **Other Graph Algorithms**

    Details: Minors, Planarity

12. **Sorting Algorithms**

    Details: Insertion Sort, Selection Sort, Merge Sort, Heap Sort, Quick Sort, Radix Sort.

13. **Search Algorithms**

    Details: AVL Trees, Skip Lists

Practicum:

- Hardware and system dependencies
- Working with simple problems involving UTF-8 string
- Design classes for and with Elementary Data Structures such as Vectors, Linked Lists, Stacks, Queues
- Recursion and Time Complexity - simple examples of recursive procedures
- Designing classes (e.g., Binary Search, Binary Search Trees, Hashing) in the context of a specific problem for the course
- Test and validate the complete program for the course problem
- Working with the Linux OS

Evaluation Scheme:

Mid-semester Test – 15%
End-semester Examination – 25%
Assignments, Quizzes and Course Project– 45%
Laboratory and class participation – 10%
Learning Portfolio – 5% (see detailed description below)

Both the theory and laboratory components must be passed to obtain a valid grade in the course. While the minimum passing criterion for each of the component is 40% individually.

A **Learning Portfolio** is an in-depth analysis of yourself as a learner in a particular learning situation, in this case – in this course. You are required to create a Learning Portfolio structured as follows:

- Reflective Narrative (4-6 pages)
- Appendices: Documents, etc., that support those comments about learning discussed in the Narrative

Key Questions to be addressed in the narrative:

1. <u>WHAT did you learn in this course</u>?
   a. Of the major learning goals for the course, and you will honestly describe to what degree you believe you achieved each one.
   b. What *Foundational Knowledge* did you acquire? *Application*? *Integration*, etc.
2. <u>HOW did you learn</u>?
   a. What helped you learn? What didn't?
   b. What does this tell you about yourself as a learner? About the learning process itself?
3. <u>What is the VALUE to you, of what you learned</u>?
   a. For your: Personal life? Social interactions? Civic responsibilities? Professional life?
4. <u>What is your PLAN for FUTURE LEARNING</u>?
   a. WHAT ELSE do you want to learn?
   b. HOW would you learn THAT?

Academic Conduct Policy:
It is expected that all students follow the highest standard of academic practice when participating in any evaluation component. Having a zero–tolerance for academic dishonesty, any case of misconduct, however minor, will be dealt with appropriately, and may be reported to the Academic Integrity Committee for necessary administrative action.

Grading Policy:
Award of grades would be guided by the histogram of marks and course average. If a student does not give sufficient opportunity for being assessed by the instructor, either by missing an evaluation component (listed in the *Evaluation Scheme* specified above) entirely, or by not applying oneself to the task seriously, he/she may be awarded 'NC' report.

Make-up Policy:
Please refer to the Academic Policy for knowing details about seeking make-up for examinations. For all other evaluation components, a compensatory component may be provided only if the reason for absence is genuine as assessed by the instructors, whose say will be final.