Compiler Construction

CS-471L

By Laeeq Khan Niazi

CLOs

- Understand and Apply Mathematical Formalisms like Regular Expressions, Grammars, FAs & PDAs.
- Understand the working principles of various phases of a compiler and how they are integrated to produce a working.
- Compare Various Implementations Techniques of Phases of Compiler, in particular Lexical Analyser, Parser
- Be able to work in a team (preferably alone) to design, develop, test, and deliver analysis phases of a compiler

Books

- Compiler – Principal, Techniques and Tools by Aho, Sethi and Ullman
- Engineering a Compiler – Keith D. Cooper & Linda Torczon 2nd Edition

Agenda

- Memory Management
- Object-Oriented Programming (OOP)
- STL (Standard Template Library) Containers
- Templates
- Algorithms
- Concurrency

Note

Please document your all tasks with task name, the solution and the screenshot. At end of this lab generate the pdf and upload on google classroom.

Memory Management

- **Pointers**: Variables that store memory addresses.
- **Dynamic Memory Allocation**: Using new, delete for managing heap memory.

- Read the concepts from the following page
- https://www.programiz.com/cpp-programming/memory-management

Object-Oriented Programming (OOP)

Implement one example of each concept

- **Classes and Objects**: Creating user-defined data types.
- **Struct**: Create user defined structure and store the data
- **Inheritance**: Reusing and extending functionality of base classes.
- **Virtual Functions**: For runtime polymorphism.

STL (Standard Template Library) Containers

- **Vectors**: Dynamic arrays that can resize automatically.
- **Lists**: Doubly linked lists that provide efficient insertion and deletion.
- **Deque**: Double-ended queue that allows insertion/removal at both ends.
- **Stack**: LIFO (Last In First Out) structure.
- **Queue**: FIFO (First In First Out) structure.
- **Priority Queue**: A type of queue that orders elements based on priority.
- **Set**: Collection of unique elements, generally implemented as binary search trees.
- **Map**: Collection of key-value pairs, usually implemented as balanced trees.
- **Unordered Map/Set**: Similar to maps/sets but based on hash tables for faster access.

Some problems

- You are tasked with developing a dynamic array-based solution to store an unknown number of student grades. Implement a program that allows insertion, deletion, and retrieval of elements in a vector. Demonstrate how the vector resizes dynamically when new elements are added beyond its current capacity
- Create a doubly linked list to manage a sequence of webpages visited by a user in a browser. Implement functions for moving forward, backward, adding a new page, and deleting a page from the list. How would the design change if you were using a singly linked list instead?

Some problems

- Implement a task scheduling system where tasks can be added from both the front and the back of a deque. Tasks with higher priority should be added at the front, while regular tasks should be added at the back. Demonstrate the operations of inserting, removing, and accessing elements from both ends.
- Design a program using a stack to check for balanced parentheses in a mathematical expression (e.g., {}, [], ()). Your solution should be able to handle expressions with nested parentheses and return true or false based on whether the expression is balanced

Some problems

- Implement a ticketing system for a cinema using a queue, where people are served in a first-come-first-served manner. Your program should allow customers to join the queue, process their tickets when they reach the front, and allow a VIP customer to be served at the next available opportunity.
- You are building a hospital emergency room system where patients are attended based on the severity of their condition. Implement a priority queue where higher-severity patients are treated first, even if they arrive later than others with lower-severity conditions.

Some problems

- Write a program to determine the unique elements in a list of customer email addresses. Use a set to eliminate duplicates and efficiently store the unique email addresses. Demonstrate set operations such as insertion, deletion, and searching..
- Implement a student record management system using a map, where the student ID is the key and their details (name, grades, etc.) are stored as the value. The system should allow efficient retrieval, insertion, and deletion of student records by ID

Some problems

- Create a word frequency counter using an unordered map that counts how often each word appears in a given text. Compare the performance of this solution with an ordered map in terms of insertion and lookup time. When would you prefer to use an unordered map over a regular map?

Templates

- **Learn about templates programming in C++**
  - **Function Templates**: Generic functions that work with any data type.
  - **Class Templates**: Generic classes to handle multiple data types.
  - **Template Specialization**: Customizing template behavior for specific types.

Algorithms

Sorting: std::sort, std::stable_sort

Searching: std::binary_search, std::find

Questions

- You are given a list of student names and their corresponding grades. Use std::sort to sort the list in descending order based on their grades. Modify the solution to sort the list in ascending order of grades in case of ties (same grades).
- You are working with a dataset of employee records where employees are first grouped by department, and within each department, they are ordered by their hire date. Use std::stable_sort to sort the dataset by department, ensuring that employees within the same department remain sorted by their hire date

Concurrency

- Multithreading: Using std::thread for parallel execution.
- Mutexes and Locks: Ensuring thread-safe access to shared resources.

Write a program that spawns two threads in C++: one thread prints numbers from 1 to 5, and the other thread prints numbers from 6 to 10. Use std::thread to run both threads concurrently.