# Laboratory 4

Title of the Laboratory Exercise: Stacks and queues

Introduction and Purpose of Experiment

Stacks and queues are very important data structures used in many real time applications. This experiment introduces the development of stack and queue ADT and applying them together.

1. Aim and Objectives

Aim

* To develop stack and queue ADT and to use them for string applications

Objectives

At the end of this lab, the student will be able to

* Design and develop and use stack and demonstrate its operations
* Design and develop and use queue and demonstrate its operations

1. Pseudo Codes

|  |  |
| --- | --- |
| * + - 1. Stack – Push   If top > MAX  Print Stack Overflow  End  Else  Set top = top + 1  Set stack[top] = value  End   * + - 1. Stack – Pop   If top < 0  Print Stack Underflow  End  Else  Set top = top - 1  Return stack[top + 1]  End   * + - 1. Stack – Display   Set i = 0  While i < top  Print stack[i]  Set i = i +1;  End | * + - 1. Queue – Enqueue   If tail > MAX  Print Queue Overflow  End  Else  Set tail = tail + 1  Set queue[tail] = value  End   * + - 1. Queue – Dequeue   If head > tail or tail == -1  Print Queue Underflow  End  Else  Set head = head + 1  Return queue[head - 1]  End   * + - 1. Queue – Display   Set i = head  While i < tail  Print queue[i]  Set i = i +1;  End |

1. Conclusions

Stacks and Queues are two of the very basic data structures, and used is simple applications, of which are described below:

Applications of Stack:

Backtracking is used in algorithms in which there are steps along some path (state) from some starting point to some goal.

* Find your way through a maze.
* Find a path from one point in a graph (roadmap) to another point.
* Play a game in which there are moves to be made (checkers, chess).

In all of these cases, there are choices to be made among a number of options. We need some way to remember these decision points in case we want/need to come back and try the alternative

Consider the maze. At a point where a choice is made, we may discover that the choice leads to a dead-end. We want to retrace back to that decision point and then try the other (next) alternative.

Again, stacks can be used as part of the solution. Recursion is another, typically more favored, solution, which is actually implemented by a stack.

Applications of Queue:

The simplest two search techniques are known as Depth-First Search(DFS) and Breadth-First Search (BFS). These two searches are described by looking at how the search tree (representing all the possible paths from the start) will be traversed.

Breadth-First Search with a Queue

In breadth-first search we explore all the nearest possibilities by finding all possible successors and enqueue them to a queue.

* Create a queue
* Create a new choice point
* Enqueue the choice point onto the queue
* while (not found and queue is not empty)
  + Dequeue the queue
  + Find all possible choices after the last one tried
  + Enqueue these choices onto the queue
* Return