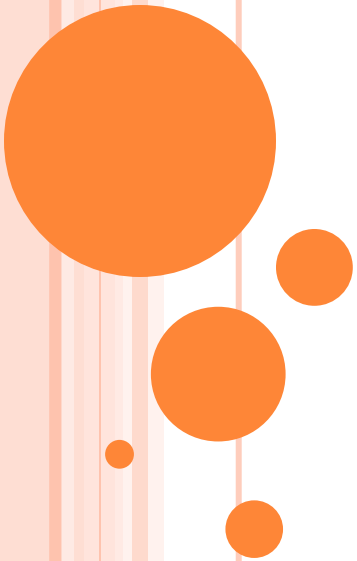


DATA STRUCTURES

LAB III

Infix to Postfix Conversion,
Postfix Evaluation
Queue ADT

Outline

- Infix to postfix conversion.
- Evaluating postfix expression.
- Queue
- Exercise.
- Prelab questions.

Infix to Postfix

- **Infix expression:**The expression of the form a op b. When an operator is in-between every pair of operands.
- **Postfix expression:**The expression of the form a b op. When an operator is followed for every pair of operands.

Why postfix representation of the expression?

The compiler scans the expression either from left to right or from right to left.

- Consider the below expression: a op1 b op2 c op3 d
  If op1 = +, op2 = *, op3 = +
- The compiler first scans the expression to evaluate the expression b * c, then again scan the expression to add a to it. The result is then added to d after another scan.
- The repeated scanning makes it very in-efficient. It is better to convert the expression to postfix(or prefix) form before evaluation.
- The corresponding expression in postfix form is: abc*+d+. The postfix expressions can be evaluated easily using a stack. We will cover postfix expression evaluation in a separate post

Algorithm for Infix to Postfix conversion

- **1.** Scan the infix expression from left to right.
- **2.** If the scanned character is an operand, output it.
- **3.** Else,
  .....**3.1** If the precedence of the scanned operator is greater than the precedence of the operator in the stack(or the stack is empty or the stack contains a '('), push it.
  .....**3.2** Else, Pop all the operators from the stack which are greater than or equal to in precedence than that of the scanned operator. After doing that Push the scanned operator to the stack. (If you encounter parenthesis while popping then stop there and push the scanned operator in the stack.)

Algorithm for Infix to Postfix conversion

- **4.** If the scanned character is an '(', push it to the stack.
- **5.** If the scanned character is an ')', pop the stack and output it until a '(' is encountered, and discard both the parenthesis.
- **6.** Repeat steps 2-6 until infix expression is scanned.
- **7.** Print the output
- **8.** Pop and output from the stack until it is not empty.

Evaluation of Postfix expression

- The Postfix notation is used to represent algebraic expressions. The expressions written in postfix form are evaluated faster compared to infix notation as parenthesis are not required in postfix.

Algorithm for evaluation of Postfix expression

- 1) Create a stack to store operands (or values).
- 2) Scan the given expression and do following for every scanned element.
  .....a) If the element is a number, push it into the stack
  .....b) If the element is a operator, pop operands for the operator from stack. Evaluate the operator and push the result back to the stack
- 3) When the expression is ended, the number in the stack is the

Queue

- Queue is an abstract data structure, somewhat similar to Stacks.
- Unlike stacks, a queue is open at both its ends. One end is always used to insert data (enqueue) and the other is used to remove data (dequeue). Queue follows First-In-First-Out methodology, i.e., the data item stored first will be accessed first.

Queue Definition

- A Queue is an ordered list in which insertions are done at one end (rear) and deletions are done at other end (front). The first element to be inserted is the first one to be deleted. Hence it is called First IN First Out (FIFO) or Last In Last Out (LIFO) list.

Queue ADT

The following operations make a queue ADT. Insertions and Deletions in the queue must follow the scheme.

Main Queue operations:

- EnQueue(int data): Inserts an element at the end of the queue.
- int DeQueue(): Removes and returns the element at the front of the queue.
- int IsEmptyQueue(): Indicates whether no elements are stored in the queue or not.
- int IsFullQueue(): Indicates whether the queue is full or not.
- Int Peek(): Returns the element at the front without removing it.

Applications

- Operating systems scheduling jobs
- Simulation of real-world queues such as lines at ticket counter or any other first-come first-served scenario requires a queue.
- Multiprogramming
- Waiting times of customers at call center

Enqueue Operation

- Queues maintain two data pointers, **front** and **rear**. Therefore, its operations are comparatively difficult to implement than that of stacks.
- The following steps should be taken to enqueue (insert) data into a queue –

  Step 1 – Check if the queue is full.

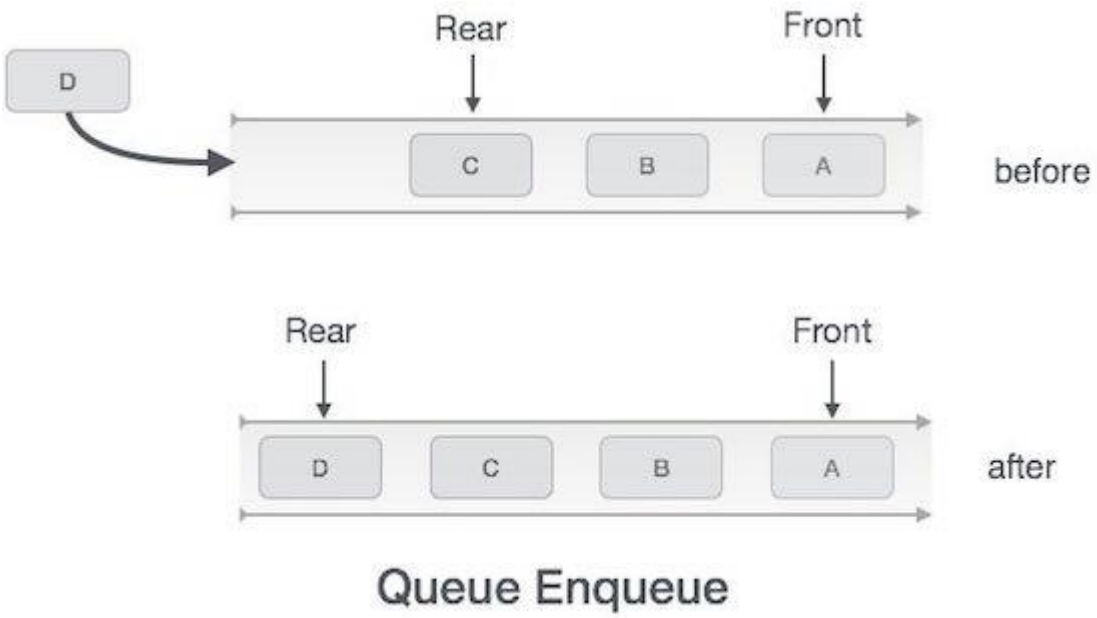
  Step 2 – If the queue is full, produce overflow error and exit.

  Step 3 – If the queue is not full, increment **rear** pointer to point the next empty space.

  Step 4 – Add data element to the queue location, where the rear is pointing.

  Step 5 – return success

Enqueue Operation



Dequeue Operation

- Accessing data from the queue is a process of two tasks – access the data where **front** is pointing and remove the data after access. The following steps are taken to perform **dequeue** operation –

  Step 1 – Check if the queue is empty.

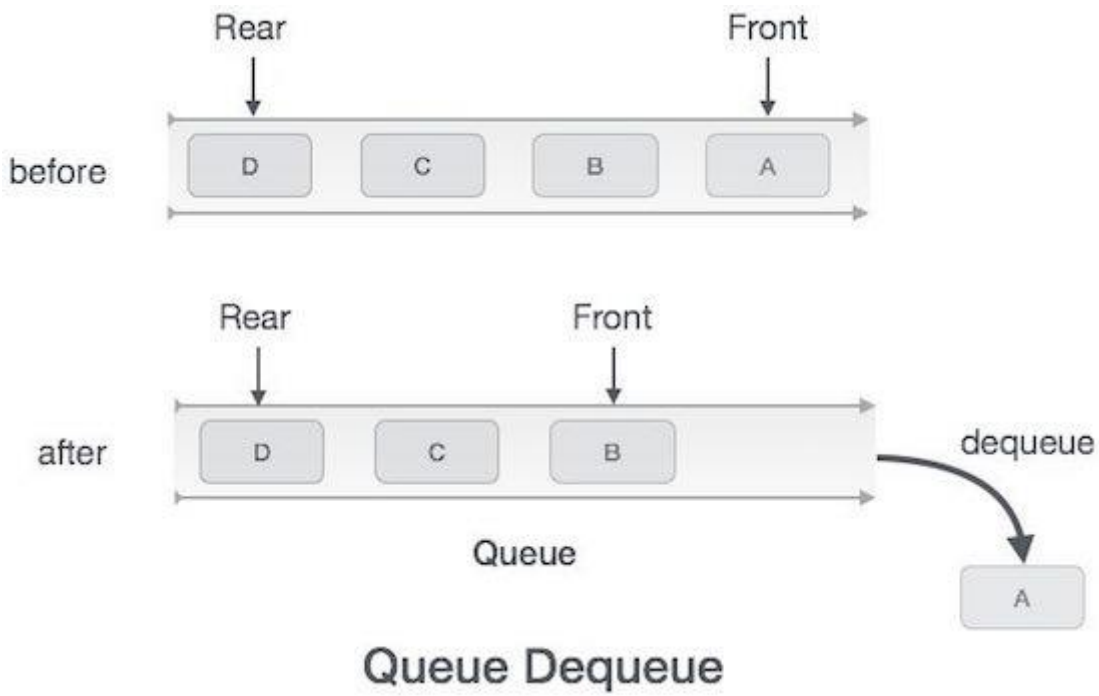
  Step 2 – If the queue is empty, produce underflow error and exit.

  Step 3 – If the queue is not empty, access the data where **front** is pointing.

  Step 4 – Increment **front** pointer to point to the next available data element.

  Step 5 – Return success.

Dequeue Operation



Queue Dequeue

Exercise

- Write a program to implement infix to postfix conversion using stack.
- Write a program to implement evaluation of postfix expression.
- Write a program to implement infix to postfix conversion using stack when right to left associativity exists.
- Write a menu driven C program that implements Queue using arrays

  a)create b)insert c)delete d)peek e)display

Prelab Questions

- Write a function to free the memory allocated to stack and stack array at run time.
- What are the complexities for stack operations.
- Write a function to double the size of the stack array at runtime.
- After converting from infix to postfix expression the order of operands or operators will affect? Give example.
- Consider an empty stack of integers. Let the numbers 1,2,3,4,5,6 be pushed on to this stack in the order they appear from left to right. Let S indicate a push and X indicate a pop operation. Can they be permuted into the order 325641(output) and order 154623(output).
- Define queue.
- List out the applications of queue.