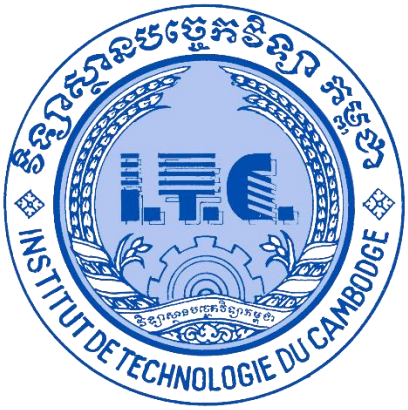


វិទ្យាស្ថានបច្ចេកវិទ្យាកម្ពុជា

Institute of Technology of Cambodia

TP-16

Stacks and Queues in C++

Academic Year: 2022 - 2023

1. Stacks

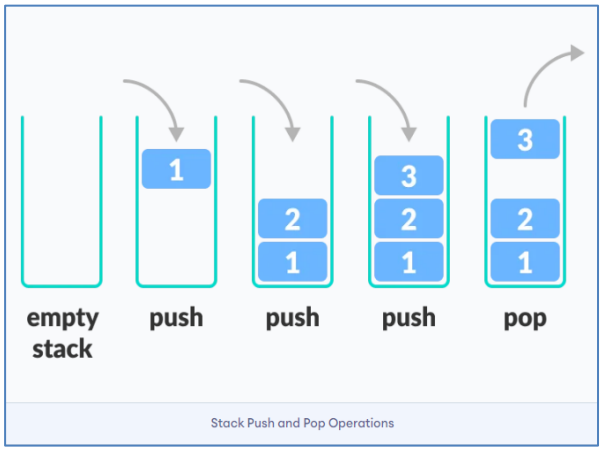
- **Definition**

  A stack is a linear data structure that follows the principle of **Last In First Out** (**LIFO**). This means the last element inserted inside the stack is removed first.

Stack as the pile of plates

Stack representation similar to a pile of plate

Stack in programming terms



Stack Push and Pop Operations

- **Operations of Stack**

  There are some basic operations that allow us to perform different actions on a stack, such as:

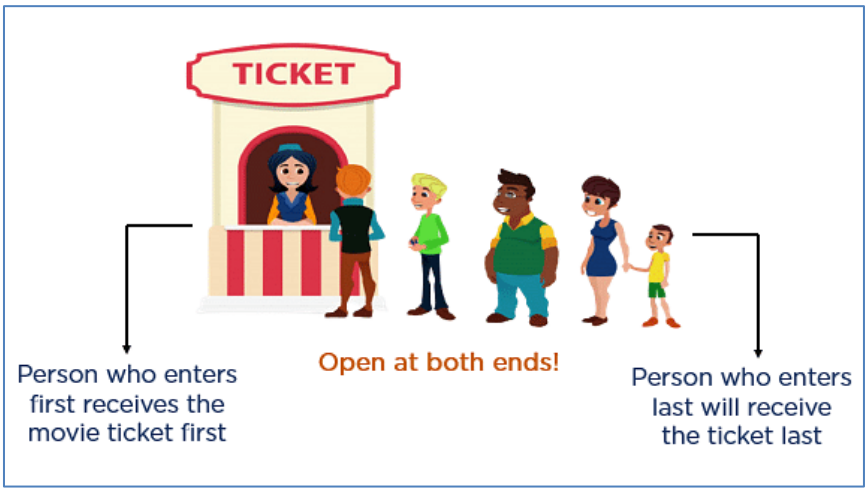
  - **Push**: Add an element to the top of a stack.
  - **Pop**: Remove an element from the top of a stack.
  - **IsEmpty**: Check if the stack is empty.
  - **IsFull**: Check if the stack is full.
  - **Peek**: Get the value of the top element without removing it.

2. Queues

- **Definition**
  - Queue follows the **First In First Out** (**FIFO**) rule - the item that goes in first is the item that comes out first.

  - A queue is a useful data structure in programming. It is similar to the ticket queue outside a cinema hall, where the first person entering the queue is the first person who gets the ticket.

Queue as the ticket receivers



Queue in programming terms

- **Operations of Queue**

  A queue is an object (an abstract data structure - ADT) that allows the following operations:

  - **Enqueue**: Add an element to the end of the queue.
  - **Dequeue**: Remove an element from the front of the queue.
  - **IsEmpty**: Check if the queue is empty.
  - **IsFull**: Check if the queue is full.
  - **Peek**: Get the value of the front of the queue without removing it.

- **Difference Stacks and Queues**

Stack	Queue
1. Elements are inserted and removed at the same end. 2. In stack only one pointer is used that is called top of the stack. 3. Stack operations are called push and pop. 4. Stacks are visualized as vertical collections. 5. Example: Collections of plates on the dining table	1. Elements are inserted and removed at the different end. 2. In queue two different pointers are used that are called front and rear. 3. Queue operations are called en-queue and de-queue. 4. Queues are visualized as horizontal collections. 5. Example: People standing at ticket counter.

Write program for each problem below:

1. Create an implementation of Queue and Stack as Linked list. Make them as header files. **Hint**: You can reuse the Single Linked List code and rename the operations and variable names to the terms used by Stack and Queue.

> **Note**: There will be two header files: MyStack.h, MyQueue.h

2. A letter means push and an asterisk (*****) means pop in the following sequence. Give the sequence of values returned by the pop operations when this sequence of operations is performed on an initially empty LIFO stack.

> **Note**: Show a warning and exit if the user tries to pop from an empty stack or push onto a full one.
>
> **E A S * Y * Q U E * * * S T * * * I O * N * * * * ***

3. A letter means put and an asterisk (*****) means get in the following sequence. Give the sequence of values returned by the dequeue operation when this sequence of operations is performed on an initially empty FIFO queue.

> **E A S * Y * Q U E * * * S T * * * I O * N * * ***

4. Write a program that reads in a sequence of characters and prints them in reverse order. Use a stack.

5. Write a program that reads in a sequence of characters, and determines whether its parentheses, braces, and curly braces are "balanced."

Hint: for left delimiters, push onto stack; for right delimiters, pop from stack and check whether popped element matches right delimiter.

6. Write a program that reads in a positive integer and store the binary representation of that integer into a stack. Finally, display the binary representation from the stack.

Hint: divide the integer by 2.

7. Create a function **void STACKmultipop(int k)** that pops **k** elements from the stack, or until the stack is empty.

Remark: For the exercise #2, implement Stack or Queue as array. The other exercises #1, #3-#7, implement those Stack or Queue as Linked List.

Reference:
https://www.geeksforgeeks.org/stack-data-structure-introduction-program/
https://www.programiz.com/dsa/stack
https://www.programiz.com/dsa/queue
https://www.geeksforgeeks.org/header-files-in-c-cpp-and-its-uses/