**A Queue**

A queue is a linear list in which data can only be inserted at one end, called the REAR, and deleted from the other end, called the FRONT. These restrictions ensure that the data is processed through the queue in the order in which it is received. In an other words, a queue is a structure in which whatever goes fist comes out first (first in, first out(FIFO) structure).

REAR

FRONT

Figure : Queue representation

**2.5.1 Types of queues**

There exist two types of queues:

**• Linear queue**

**• Circular queue**

1. **Linear Queue**

Queue data structure is a linear data structure in which the operations are performed based on FIFO principle. **Queue Operations using Array** before we implement actual operations**, first follow the below steps to create an empty queue.**

• Step 1: Include all the header files which are used in the program and define a constant ‘SIZE’ with specific value.

• Step 2: Declare all the user defined functions which are used in queue implementation.

• Step 3: Create a one dimensional array with above defined SIZE (int queue[SIZE])

• Step 4: Define two integer variables ‘front’ and ‘rear’ and initialize both with ‘-1’. (int front = -1, rear=-1)

• Step 5: Then implement main method by displaying menu of operations list and make suitable function calls to perform operation selected by the user on queue.

1. **Circular Queue**

Circular Queue is a linear data structure in which the operations are performed based on FIFO (First In First Out) principle and the last position is connected back to the first position to make a circle.

FRONT

**Circular Queue Operation**

To implement a circular queue data structure using array, we first create it.

• Step 1: Include all the header files which are used in the program and define a constant ‘SIZE’ with specific value.

• Step 2: Declare all user defined functions used in circular queue implementation.

• Step 3: Create a one dimensional array with above defined SIZE (int cQueue[SIZE])

• Step 4: Define two integer variables ‘front’ and ‘rear’ and initialize both with ‘-1’. (int front = -1, rear = -1).

• Step 5: Implement main method by displaying menu of operations list and make suitable function calls to perform operation selected by the user on circular queue.

**2.5.2 Queue operations**

There are three main operations related to queues.

• Enqueue: the enqueue operation inserts an items at the rear of the queue

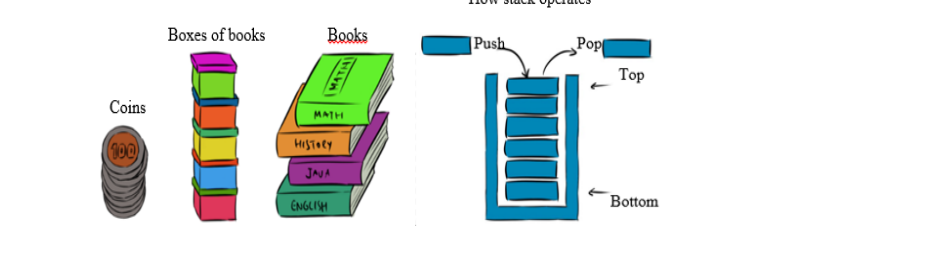
• Dequeue: the dequeue operation deletes the item at the front of the queue

• Display: show elements in the array

**2.6 Stack**

A stack is a restricted linear list in which all additions and deletions are made at one end, the top. If we insert a series of data items into a stack and then remove them, the order of the data is reversed. Data input as 5, 10, 15, 20, for example would be removed as 20, 15, 10, and 5. This reversing attribute is why stacks are known as Last in, First out (LIFO) data structures.

We use many different types of stacks in our daily lives. We often talk of a stack of coins, stack of books on a table and stack of plates in a kitchen. . Any situation in which Computer Science Senior 5 Student Book 44 we can only add or remove an object at the top is a stack. If we want to remove an object other than the one at the top, we must first remove all objects above it. The following charts illustrates cases of stack



A Stack is a Last in First out (LIFO) dynamic table or data structure. It has the following characteristics:

• List of the same kind of elements;

• Addition and deletion of elements occur only at one end, called the top of the stack;

• Computers use stacks to implement method calls;

• Stacks are also used to convert recursive algorithms into non recursive algorithm.

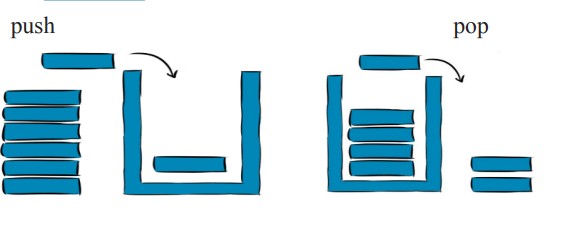


Figure: stacks’ operations (Push and Pop)

2.6.1 Operations performed on stacks The different operations performed on stacks are as follows:

• Push: adds an element to the stack

• Pop: removes an element from the stack

• Peek: display at top element of the stack

**Stack Operations using Array**

Before implementing actual operations, first follow the below steps to create an empty stack.

Step 1: Include all the header files  which are used in the program and define a constant ‘SIZE’ with specific value.

• Step 2: Declare all the functions used in stack implementation.

• Step 3: Create a one dimensional array with fixed size (int stack[SIZE])

• Step 4: Define a integer variable ‘top’ and initialize with ‘-1’. (int top = -1)

• Step 5: In main method display menu with list of operations and make suitable function calls to perform operation selected by the user on the stack.

**push(value) - Inserting value into the stack**

• Step 1: Check whether stack is FULL. (top == SIZE-1)

• Step 2: If it is FULL, then display “Stack is FULL!!! Insertion is not possible!!!” and terminate the function.

• Step 3: If it is NOT FULL, then increment top value by one (top++) and set stack [top] to value (stack[top] = value).

**pop() - Delete a value from the Stack**

• Step 1: Check whether stack is EMPTY. (top == -1)

• Step 2: If it is EMPTY, then display “Stack is EMPTY!!! Deletion is not possible!!!’” and terminate the function.

• Step 3: If it is NOT EMPTY, then delete stack [top] and decrement top value by one(top-).

**Display() - Displays the elements of a Stack**

• Step 1: Check whether stack is EMPTY. (top == -1)

• Step 2: If it is EMPTY, then display “Stack is EMPTY!!!” and terminate the function.

• Step 3: If it is NOT EMPTY, then define a variable ‘I’ and initialize with top. Display stack[i] value and decrement i value by one (i--).

• Step 3: Repeat above

2.6.2 Stack exceptions

Adding an element to a full stack and removing an element from an empty stack would generate errors or exceptions:

• Stack overflow exception: it occurs if you try to add an item to stack or queue which is already full.

• Stack underflow exception: it occurs if you try to remove an item from an empty queue or stack.

Notec 1: Stack and Queue data structures can be implement in two ways: By using array or by using linked list.

Notice 2: When stack or Queue is implemented using array, that stack or Queue can organize only a limited number of elements. When stack or Queue is implemented using linked list, that stack or Queue can organize an unlimited number of elements.