

## **CHAPTER 2**

### **DATA STRUCTURES: STACK & QUEUE**

#### **2.1 Stack**

- **Def:**
  - “A data structure, in which elements can be **added** and **removed** from the **one end** only.”
- **Example:**
  - Collection of Plates on the counter in cafeteria.
  - Ordering of pages in a file.
- **Characteristic:**
  - **LIFO: Last In First Out.**
  - Last inserted element (item) comes out First. (Explain this with an example.)
- **Representation:**
  - **Vertical:** (Provide figure as discussed in class.)
  - **Horizontal:** (Provide figure as discussed in class.)
- **Operations:**
  - **Push:** Inserts an element in a stack.
  - **Pop:** Removes / Deletes an element from a stack.
  - **Peep:** Returns  $i^{\text{th}}$  element from a stack.
  - **Update:** Updates / Changes  $i^{\text{th}}$  element in a stack.
- **Implementation:**
  - By using **Array** (Static Memory Allocation)
  - By using **Linked List / Pointer** (Dynamic Memory Allocation)

[Note: Use “DECROI” to remember above points.]

## 2.2 Stack Operations: Push

- **Def:** “Process of inserting an element in a stack.”
- **Explanation:** (Provide explanation as discussed in class.)
- **Stack Overflow:** “Situation, arising during **Push** operation, when Stack is **Full**.”
- **Function:**

```
void push ( int x )
{
    // Check for an Overflow...
    if ( top == MAX )
    {
        printf ( " Stack Overflow...\n " );
        getch ( );
        exit ( 0 );
    }

    // Store an element at top of stack...
    stck [ top ] = x;

    // Update top pointer...
    top ++ ;
}
```

- **Algorithm:**

❖ **PUSH ( X )**

- [Inserts given element 'X' in a stack.]

❖ **Variables:**

- i) **STCK:** Array having MAX elements.
- ii) **MAX:** No. of maximum elements in a stack.
- iii) **TOP:** Pointer to track top of stack.
- iv) **X:** Element to be inserted in a stack.

❖ **Steps:**

- **Step-1:** [Check for stack Overflow.]  
     IF (TOP = MAX) THEN  
         WRITE ('Stack Overflow...')  
         EXIT  
     END IF
- **Step-2:** [Store an element at top of stack.]  
     STCK [TOP] ← X
- **Step-3:** [Update top pointer.]  
     TOP ← TOP + 1
- **Step-4:** [Finish]  
     RETURN

## 2.3 Stack Operations: Pop

- **Def:**
  - “Process of removing an element from stack.”
- **Explanation:**
  - (Provide explanation as discussed in class.)
- **Stack Underflow:**
  - “Situation, arising during **Pop** operation, when Stack is **Empty**.”
- **Function:**

```

int    pop    ( )
{
    // Check for an Underflow...
    if ( top == 0 )
    {
        printf ( “ Stack Underflow...\n ”);
        getch ( );
        exit ( 0 );
    }

    // Update top pointer...
    top -- ;

    // Return an element from top of stack...
    return ( stck [top] );
}

```

- **Algorithm:**
  - ❖ **POP ( )**
    - [Removes / Deletes an element from top of stack.]
  - ❖ **Variables:**
    - i) **STCK:** Array having MAX elements.
    - ii) **MAX:** No. of maximum elements in a stack.
    - iii) **TOP:** Pointer to track top of stack.
  - ❖ **Steps:**
    - **Step-1:** [Check for stack Underflow.]
      - IF (TOP = 0) THEN
      - WRITE (‘Stack Underflow...’)
      - EXIT
      - END IF
    - **Step-2:** [Update top pointer.]
      - TOP ← TOP – 1
    - **Step-3:** [Return an element to be removed.]
      - RETURN ( STCK [TOP] )

## 2.4 Implementation of Stack

```
#include<stdio.h>
#include<conio.h>
#include<stdlib.h>

#define MAX 5

int stack[MAX];
int top = 0;

void main()
{
    int choice, x;

    void push(int);
    int pop();

    while(1)
    {
        printf("\n1. Push Operation.\n");
        printf("2. Pop Operation.\n");
        printf("3. Exit.\n");

        printf("\n Enter Ur Choice : ");
        scanf("%d",&choice);

        switch(choice)
        {
            case 1:
                printf(" Enter element to be pushed : ");
                scanf("%d",&x);
                push(x);
                break;

            case 2:
                x = pop();
                printf(" Poped element is : %d\n",x);
                break;

            case 3:
                printf("\n Program terminated successfully...");
                exit(0);

            default:
                printf("\n Invalid choice...\n");
        }
    }
}
```

**// define push function...**

```
void push(int x)
{
    // check for an Overflow...
    if (top == MAX)
    {
        printf("Stack Overflow...\n");
        getch();
        exit(0);
    }

    // store an element at top of stack...
    stack [ top ] = x;

    // update top pointer...
    top ++ ;
}
```

**// define pop function...**

```
int pop()
{
    // check for an Underflow...
    if (top == 0)
    {
        printf("Stack Underflow...\n");
        getch();
        exit(0);
    }

    // update top pointer...
    top -- ;

    // return an element from top of stack...
    return ( stack [ top ] );
}
```