**THE INSTITUTE OF FINANCE MANAGEMENT**



**FACULTY OF COMPUTING MATHEMATICS - FCM**

**DEPARTMENT OF COMPUTER SCIENCE**

**BACHELOR DEGREE IN COMPUTER SCIENCE**

**CSU 07317:DATA STRUCTURE AND ALGORITHMS - ASSIGNMENT**

|  |  |  |
| --- | --- | --- |
| **S/N** | **NAME** | **REGISTRATION NUMBER** |
| **1.** | **CHRISTINA S KISAMBU** | **IMC/BCS/2216198** |
| **2.** | **RODGERS J SWAI** | **IMC/BCS/2226482** |
| **3.** | **HABIBA A DAUDI** | **IMC/BCS/2211330** |
| **4.** | **MWESIGA G MTEMBEI** | **IMC/BCS/2227037** |
| **5.** | **SARAH J MKISI** | **IMC/BCS/2214387** |
| **6.** | **SAUDA A KASSIM** | **IMC/BCS/2210043** |
| **7.** | **SHAMSA A BELEKO** | **IMC/BCS/2216303** |
| **8.** | **KELVIN S MGUHI** | **IMC/BCS/2222452** |
| **9.** | **ASHIRI J MWANGIA** | **IMC/BCS/2226943** |
| **10.** | **NELSON L HENDRY** | **IMC/BCS/2222246** |

**YEAR TWO 2023/2024**

A **stack** is a linear data structure in which the insertion of a new element and removal of an existing element takes place at the same end represented as the top of the stack.

It follows an order in which the operations are performed. The order may be LIFO (Last in First Out) or FILO (First in Last Out). LIFO implies that the element that is inserted last, comes out first and FILO implies that the element that is inserted first, comes out last.

Real-life example is the pile of plates kept on top of each other. The plate which we put last is on the top and since we remove the plate that is at the top, we can say that the plate that was put last comes out first.

To implement the stack, it is required to maintain the **pointer to the top of the stack,** which isthe last element to be inserted because **we can access the elements only on the top of the stack.**

**Types of stacks.**

**Fixed Size Stack**: A fixed size stack has a fixed size and cannot grow or shrink dynamically. If the stack is full and an attempt is made to add an element to it, an overflow error occurs. If the stack is empty and an attempt is made to remove an element from it, an underflow error occurs.

**Dynamic Size Stack**: When the stack is full, it automatically increases its size to accommodate the new element, and when the stack is empty, it decreases its size. This type of stack is implemented using a linked list, as it allows for easy resizing of the stack.

In addition to these two main types, there are several other variations of Stacks, including:

* **Infix to Postfix Stack**: This type of stack is used to convert infix expressions to postfix expressions.
* **Expression Evaluation Stack**: This type of stack is used to evaluate postfix expressions.
* **Recursion Stack**: This type of stack is used to keep track of function calls in a computer program and to return control to the correct function when a function returns.
* **Memory Management Stack**: This type of stack is used to store the values of the program counter and the values of the registers in a computer program, allowing the program to return to the previous state when a function returns.
* **Balanced Parenthesis Stack**: This type of stack is used to check the balance of parentheses in an expression.
* **Undo-Redo Stack**: This type of stack is used in computer programs to allow users to undo and redo actions.

**The following are some basic operations on Stack data structure.**

* Push.

is the operation used to insert an element into the stack.

* Pop.

The operation used to remove an element from the stack. The items are popped in the reversed order in which they are pushed. If the stack is empty, then it is said to be an **Underflow condition.**

* Top.

Returns the top element of the stack.

* is Empty.

Returns true if stack is empty else false.

* Size.

Returns the size of stack.

**Applications of the stack:**

* Used in redo-undo features at many places like photoshop.
* Forward and backward features in web browsers
* In memory management, any modern computer uses a stack as the primary management for a running purpose. Each program that is running in a computer system has its own memory allocations.
* Stack also helps in implementing function call in computers. The last called function is always completed first. Stacks are also used to implement the undo/redo operation in text editor.
* Stack is used in many algorithms like histogram problems, tree traversals and stock plan problems.
* Stack can be used to store operands and operators as they are processed.

**Implementation of Stack**

A stack can be implemented using an array or linked list. In an array-based implementation, the push operation is implemented by incrementing the index of the top element and storing the new element at that index. The pop operation is implemented by decrementing the index of the top element and returning the value stored at that index.

In a linked list-based implementation, the push operation is implemented by creating a new node with the new element and setting the next pointer of the current top node to the new node. The pop operation is implemented by setting the next pointer of the current top node to the next node and returning the value of the current top node.

**Implementation of stack using array**

#include <stdio.h>

int stack[1000],i,j,choice=0,n,top=-1;

void push();

void pop();

void show();

void main ()

{

printf("Enter the number of elements in the stack ");

scanf("%d",&n);

printf("\*\*\*\*\*\*\*\*\*Stack operations using array\*\*\*\*\*\*\*\*\*");

printf("\n----------------------------------------------\n");

while(choice != 4)

{

printf("Chose one from the below options...\n");

printf("\n1.Push\n2.Pop\n3.Show\n4.Exit");

printf("\n Enter your choice \n");

scanf("%d",&choice);

switch(choice)

{

case 1:

{

push();

break;

}

case 2:

{

pop();

break;

}

case 3:

{

show();

break;

}

case 4:

{

printf("Exiting....");

break;

}

default:

{

printf("Please Enter valid choice ");

}

};

}

}

void push ()

{

int val;

if (top == n )

printf("\n Overflow\n");

else

{

printf("Enter the value?");

scanf("%d",&val);

top = top +1;

stack[top] = val;

printf("element inserted....\n");

}

}

void pop ()

{

if(top == -1)

printf("Underflow\n");

else

top = top -1;

}

void show()

{

printf("elements in stack are:\n");

for (i=top;i>=0;i--)

{

printf("%d\n",stack[i]);

}

if(top == -1)

{

printf("Stack is empty");

}

}

