

Experiment no.3

Aim: Implement stack using a linked list. Use this stack to perform evaluation of a postfix expression.

Objective -

- 1) To understand the concept of abstract data type
- 2) How different data structures such as arrays & stacks are represented as an ADT.

Theory:

A stack is an Abstract Data type (ADT) is special kind of datatype, whose behavior is defined by a set of values & set of operations. The keyword "Abstract" is used as we can use those datatypes, we can perform different operations. But how those operations are working that is totally hidden from the user. The ADT is made of with primitive datatypes, but operation logic are hidden.

Functions of the Stack ADT

isFull(), Check whether stack is full or not.

peek(), get the top data element of the stack, without removing it.

isEmpty(), Check whether stack is empty or not.

push(x), This is used to push x into the stack.

pop(), Used to delete one element from top of the stack.

Size(), Used to get number of elements present into the stack.

Algorithm

Procedures to support stack functions-


```

bool isFull() {
    if (top == MAXSIZE)
        return true;
    else
        return false;
}

```

Implementation of isEmpty() function in C programming language is slightly different. We initialize top at -1, as the index in array starts from 0. So we check if the top is below zero or -1 to determine if the stack is empty.

```

bool isEmpty() {
    if (top == -1)
        return true;
    else
        return false;
}

```

Push Operation: The process of putting a new data element onto stack is known as a Push Operation.

```

void push(int data) {
    if (!isFull()) {
        top = top + 1;
        stack[top] = data;
    } else {
        printf("Could not insert data, Stack is full.\n");
    }
}

```


Pop Operation: In an array implementation of pop() operation, the data element is not actually removed, instead top is decremented to a lower position in the stack to point to the next value. But in linked-list implementation, pop() actually removes data element & deallocates memory space.

```
int pop(int data){
    if (!isempty()){
        data = Stack[top];
        top = top - 1;
        return data;
    } else {
        printf("Could not retrieve data, Stack is empty.\n");
    }
}
```

Program :

```
#include <iostream>
#include <cstring>
#include <string>
#include <stack>
using namespace std;

struct Node
{
    int data;
    struct Node *next;
} *top = NULL;

void push(int);
void pop();
void display();
void postfix();

void push(int value)
{
    struct Node *newNode;
    newNode = (struct Node *)malloc(sizeof(struct Node));
    newNode->data = value;
    if (top == NULL)
        newNode->next = NULL;
    else
        newNode->next = top;
    top = newNode;
    cout << ("\nInsertion is Success!!!\n");
}

void pop()
{
    if (top == NULL)
        cout << ("\nStack is Empty!!!");
    else
    {
        struct Node *temp = top;
        cout << ("\nDeleted element: %d", temp->data);
        top = temp->next;
        free(temp);
    }
}

void display()
{
    if (top == NULL)
        cout << ("\nStack is Empty!!!\n");
    else
    {
        struct Node *temp = top;
        while (temp->next != NULL)
```

```

    {
        cout << ("\t", temp->data) << ", ";
        temp = temp->next;
    }
    cout << ("\t", temp->data) << ", ";
}
}

struct Stack
{
    int top;
    unsigned capacity;
    int *array;
};

// Stack Operations
struct Stack *createStack(unsigned capacity)
{
    struct Stack *stack = (struct Stack *)malloc(sizeof(struct Stack));
    if (!stack)
        return NULL;
    stack->top = -1;
    stack->capacity = capacity;
    stack->array = (int *)malloc(stack->capacity * sizeof(int));
    if (!stack->array)
        return NULL;
    return stack;
}

int isEmpty(struct Stack *stack)
{
    return stack->top == -1;
}

char peek(struct Stack *stack)
{
    return stack->array[stack->top];
}

char pop(struct Stack *stack)
{
    if (!isEmpty(stack))
        return stack->array[stack->top--];
    return '$';
}

void push(struct Stack *stack, char op)
{
    stack->array[++stack->top] = op;
}

int evaluatePostfix(char *exp)
{
    struct Stack *stack = createStack(strlen(exp));
    int i;
    if (!stack)
        return -1;
    for (i = 0; exp[i]; ++i)
    {

```

```

    if (isdigit(exp[i]))
        push(stack, exp[i] - '0');
    else
    {
        int val1 = pop(stack);
        int val2 = pop(stack);
        switch (exp[i])
        {
            case '+':
                push(stack, val2 + val1);
                break;
            case '-':
                push(stack, val2 - val1);
                break;
            case '*':
                push(stack, val2 * val1);
                break;
            case '/':
                push(stack, val2 / val1);
                break;
        }
    }
}
return pop(stack);
}

```

```

int main()
{
    int choice, value;
    cout << ("\n***** MENU *****\n");
    cout << "* 1. Push in stack      *" << endl;
    cout << "* 2. Pop from stack     *" << endl;
    cout << "* 3. Display stack      *" << endl;
    cout << "* 4. Postfix Evaluation *" << endl;
    cout << "* 5. Exit               *" << endl;
    cout << "*****" << endl;

    do
    {
        cout << ("\nEnter your choice: ");
        scanf("%d", &choice);
        switch (choice)
        {
            case 1:
                cout << ("Enter the value to be insert: \n");
                scanf("%d", &value);
                push(value);
                break;
            case 2:
                pop();
                break;
            case 3:
                display();

```

```

        break;
    case 4:
    {
        char exp[] = "10 20 * 30 60 10 / - +";
        cout << "Postfix Evaluation: " << evaluatePostfix(exp)<<"\n";
        return 0;
    }
    case 5:
        exit(0);
        break;
    default:
        cout << ("\nWrong selection!!! Please try again!!!\n");
    }
} while (choice != 4);
return 0;
}

```

Output :

```

C++ Assignment3.cpp > main()
1 #include <iostream>
2 #include <cstring>

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
orion@OMEN-15:/mnt/d/College/2 Second year/SY SEM 3/Data Structures and Algorithms (DSA)/Lab manual/assign 3$ ./Assignment3

***** MENU *****
* 1. Push in stack *
* 2. Pop from stack *
* 3. Display stack *
* 4. Postfix Evaluation *
* 5. Exit *
*****

Enter your choice: 1
Enter the value to be insert:
6

Insertion is Success!!!

Enter your choice: 1
Enter the value to be insert:
3

Insertion is Success!!!

Enter your choice: 1
Enter the value to be insert:
9

Insertion is Success!!!

Enter your choice: 3
9, 3, 6,
Enter your choice: 2
9
Enter your choice: 3
3, 6,
Enter your choice: 4
Postfix Evaluation: 72
orion@OMEN-15:/mnt/d/College/2 Second year/SY SEM 3/Data Structures and Algorithms (DSA)/Lab manual/assign 3$

```