G. H. Raisoni College Of Engineering And Management, Wagholi Pune

2021- 2022

Assignment no :- 3

Department	CE [SUMMER 2022 (Online)]		
Term / Section	III/B	Date Of **submission**	12-10-2021
Subject Name /Code	Data Structures and Algorithms/ UCSL201/UCSP201		
Roll No.	SCOB77	Name	Pratham Rajkumar pitty
Registration Number	2020AC0E1100107		

Assignment No.3

Aim: Implementing 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 array and a stackes are represented as an ADT.

Theory:→

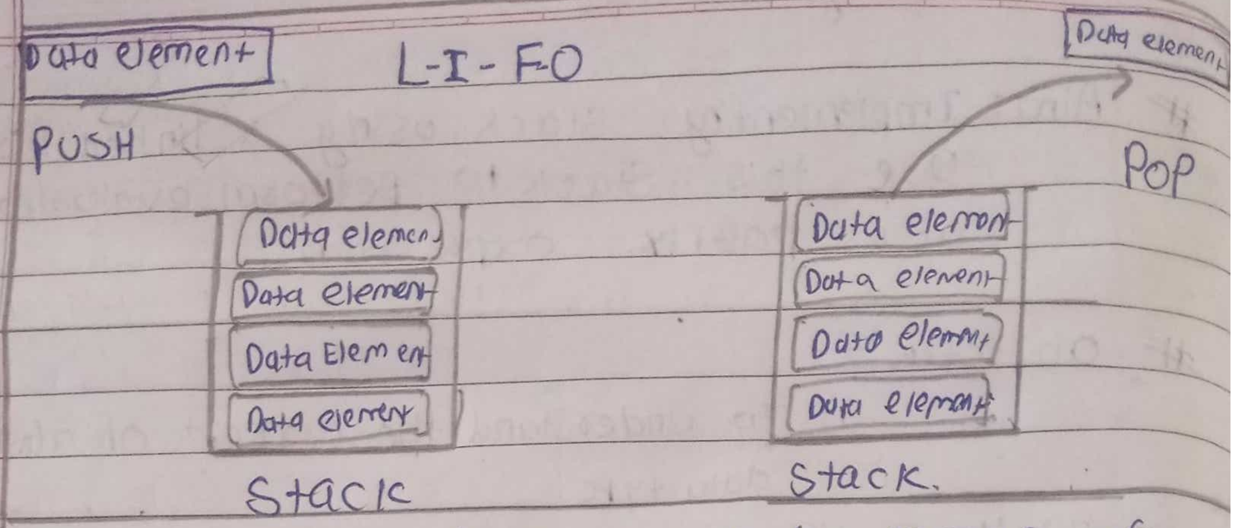
A Stack is an Abstract Data Type (ADT), commonly used in most programming language. It is named Stack as it behaves like a real-world stack, e.g- A deck of cards or a pile of plates, etc.

A real-world stack allows operations at one end only. Ex. we can place or remove a card or plate from a top of a stack only.
Likewise, ADT allows all data operations at one end only.

This feature make it LIFO (Last-in-First-out) data structure. Hear, the element which is placed (inserted or added) last, is accessed first.
Insertion operation is called PUSH operation.
removal operation is called POP operation.

▶ Stack Representation:-

The following digram depicts a stack and its operation.



A Stack can be implemented by means of Array, structures, pointers & Linked list. Stack can either be a fixed size ~~using~~ one or it may have a sense of dynamic resizing.
Stack using Array, which makes it a fixed size stack implementation.

▷ Basic operations

• Stack operations may involves:-
(1) initalizing a stack, using it
2) ~~its~~ de-initalizing a stack

• Primary operation in stack
(1) push() → Pushing (Storing) an element on the stack
(2) POP() → removing (accessing) an element ~~on the~~ from stack when data is pushed into stack

• To check the status of stack following
• Functionality is added to Stacks—
(1) peek() - get the top element of stack without Removing it
(2) isFull() - Check if stack is full.
(3) isEmpty() - Check if Stack is empty.

Pointer is maintained to the last pushed data on the stack. As pointer always always represents the top of the stack, hence named top. The top pointer provides top value of the stack without actually removing it.

▷ ~~to~~ Implementation of isfull()

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

▷ implementation of isempty()

```
bool isempty() {
if (top==-1)
  return true;
else
  return False ;}
}
```

we initalize top at -1, as the index in array starts from 0. we check if the top is below zero or -1 to determine if the stack is empty ~~empty~~.

▷ 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 Array implementation of ~~In~~ pop, 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 and deallocates memory space.

*

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

Evaluation postfix expression **using a linked list**

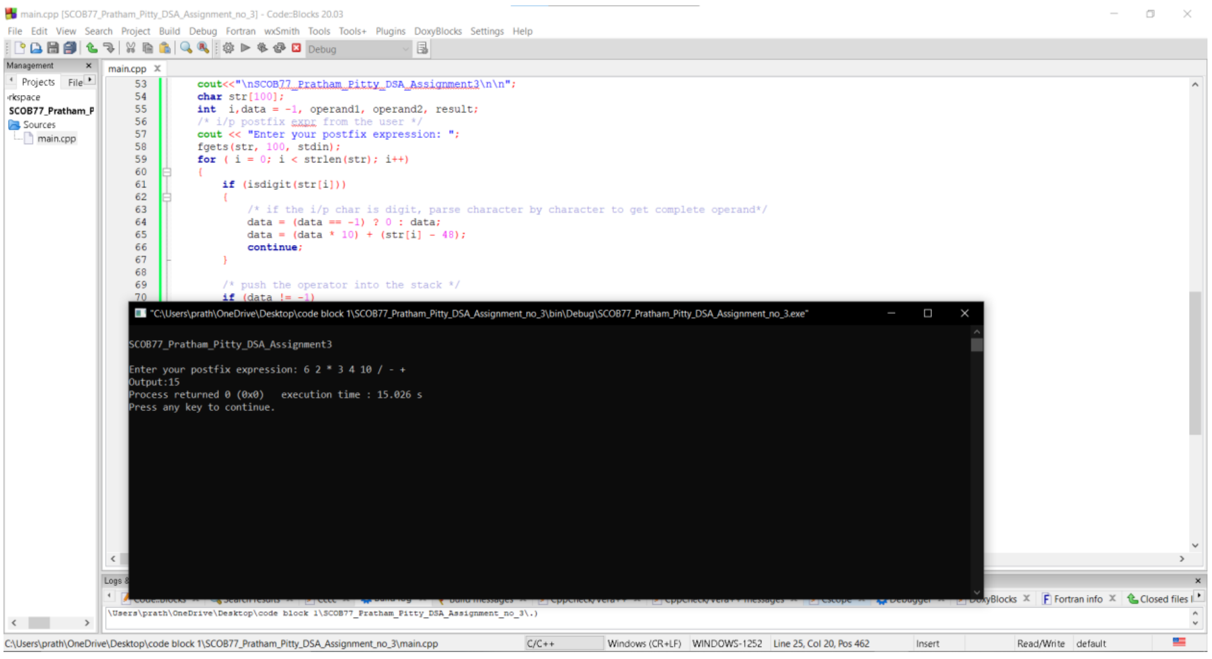
Program code

```
#include <iostream>

#include <stdlib.h>

#include <string.h>

#include <ctype.h>


using namespace std;


struct node

{

   int data;

   struct node *next;

};


struct node *top = NULL;


/* create a new node with the given data */

 struct node *createNode(int data)

{

   struct node *ptr = (struct node *)malloc(sizeof(struct node));

   ptr->data = data;

   ptr->next = NULL;

}


/* push the input data into the stack */

void push(int data)

{

   struct node *ptr = createNode(data);

   if (top == NULL)

   {

      top = ptr;
```

```
        return;
    }
    ptr->next = top;
    top = ptr;
}

/* pop the top element from the stack */
int pop()
{
    int data;
    struct node *temp;
    if (top == NULL)
        return -1;
    data = top->data;
    temp = top;
    top = top->next;
    free(temp);
    return (data);
}

int main()
{
    cout<<"\nSCOB77_Pratham_Pitty_DSA_Assignment3\n\n";
//   6 2 * 3 4 10 / - +
    char str[100];
    int  i,data = -1, operand1, operand2, result;
    /* i/p postfix expr from the user */
    cout << "Enter your postfix expression: ";
    fgets(str, 100, stdin);
    for ( i = 0; i < strlen(str); i++)
    {
        if (isdigit(str[i]))
        {
```

```
    /* if the i/p char is digit, parse character by character to get complete operand*/
    data = (data == -1) ? 0 : data;
    data = (data * 10) + (str[i] - 48);
    continue;
}

/* push the operator into the stack */
if (data != -1)
{
    push(data);
}

if (str[i] == '+' || str[i] == '-' || str[i] == '*' || str[i] == '/')
{
    /*
              * if the i/p character is an operator,
              * then pop two elements from the stack,
              * apply operator and push the result into
              * the stack
              */
    operand2 = pop();
    operand1 = pop();
    if (operand1 == -1 || operand2 == -1)
        break;
    switch (str[i])
    {
    case '+':
        result = operand1 + operand2;
        /* pushing result into the stack */
        push(result);
        break;
    case '-':
        result = operand1 - operand2;
```

```
                push(result);

                break;

            case '*':

                result = operand1 * operand2;

                push(result);

                break;

            case '/':

                result = operand1 / operand2;

                push(result);

                break;

            }

        }

        data = -1;

    }

    if (top != NULL && top->next == NULL)

        cout << "Output:"<<top->data;

    else

        cout << "You have entered wrong expression\n";

    return 0;

}
```


main.cpp [SCOB77_Pratham_Pitty_DSA_Assignment_no_3] - Code::Blocks 20.03
File Edit View Search Project Build Debug Fortran wxSmith Tools Tools+ Plugins DoxyBlocks Settings Help
Debug
Management
Projects
Sources
main.cpp

```
    cout<<"\nSCOB77_Pratham_Pitty_DSA_Assignment3\n\n";
    char str[100];
    int  i,data = -1, operand1, operand2, result;
    /* i/p postfix expr from the user */
    cout << "Enter your postfix expression: ";
    fgets(str, 100, stdin);
    for ( i = 0; i < strlen(str); i++)
    {
        if (isdigit(str[i]))
        {
            /* if the i/p char is digit, parse character by character to get complete operand*/
            data = (data == -1) ? 0 : data;
            data = (data * 10) + (str[i] - 48);
            continue;
        }

        /* push the operator into the stack */
        if (data != -1)
```

"C:\Users\prath\OneDrive\Desktop\code block 1\SCOB77_Pratham_Pitty_DSA_Assignment_no_3\bin\Debug\SCOB77_Pratham_Pitty_DSA_Assignment_no_3.exe"

```
SCOB77_Pratham_Pitty_DSA_Assignment3

Enter your postfix expression: 6 2 * 3 4 10 / - +
Output:15
Process returned 0 (0x0)   execution time : 15.026 s
Press any key to continue.
```

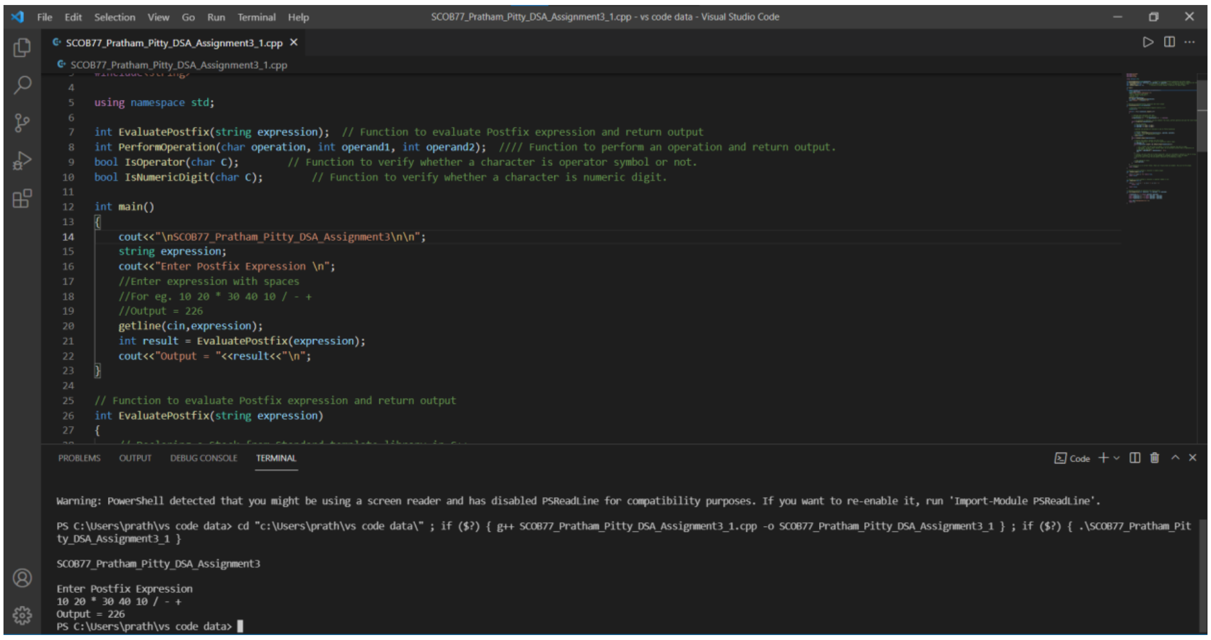
2nd method for postfix expression

Program code

```
#include<iostream>

#include<stack>

#include<string>


using namespace std;


int EvaluatePostfix(string expression);  // Function to evaluate Postfix expression and return output

int PerformOperation(char operation, int operand1, int operand2);  //// Function to perform an
operation and return output.

bool IsOperator(char C);        // Function to verify whether a character is operator symbol or not.

bool IsNumericDigit(char C);        // Function to verify whether a character is numeric digit.


int main()

{

   cout<<"\nSCOB77_Pratham_Pitty_DSA_Assignment3\n\n";

   string expression;

   cout<<"Enter Postfix Expression \n";

   //Enter expression with spaces

   //For eg. 10 20 * 30 40 10 / - +

   //Output = 226

   getline(cin,expression);

   int result = EvaluatePostfix(expression);

   cout<<"Output = "<<result<<"\n";

}


// Function to evaluate Postfix expression and return output

int EvaluatePostfix(string expression)
```

```
{
    // Declaring a Stack from Standard template library in C++.
    stack<int> S;

    for(int i = 0;i< expression.length();i++)
    {

        // Scanning each character from left.
        // If character is a delimiter, move on.
        if(expression[i] == ' ' || expression[i] == ',') continue;

            // If character is operator, pop two elements from stack, perform operation and push the
result back.
        else if(IsOperator(expression[i]))
        {
            // Pop two operands.
            int operand2 = S.top(); S.pop();
            int operand1 = S.top(); S.pop();

            //operand1 and operand2 are reversed in case of Prefix Expression

            // Perform operation
            int result = PerformOperation(expression[i], operand1, operand2);
            //Push back result of operation on stack.
            S.push(result);
        }
        else if(IsNumericDigit(expression[i]))
        {
            // Extract the numeric operand from the string
            // Keep incrementing i as long as you are getting a numeric digit.
            int operand = 0;
            while(i<expression.length() && IsNumericDigit(expression[i]))
            {
               // For a number with more than one digits, as we are scanning from left to right.
               // Everytime , we get a digit towards right, we can multiply current total in operand by 10
               // and add the new digit.
               operand = (operand*10) + (expression[i] - '0');
               i++;
            }
            // Finally, you will come out of while loop with i set to a non-numeric character or end of
string
            // decrement i because it will be incremented in increment section of loop once again.
            // We do not want to skip the non-numeric character by incrementing i twice.
            i--;

            // Push operand on stack.
            S.push(operand);
         }
      }
      // If expression is in correct format, Stack will finally have one element. This will be the output.
      return S.top();
}

// Function to verify whether a character is numeric digit.
bool IsNumericDigit(char C)
{
      if(C >= '0' && C <= '9') return true;
      return false;
}

// Function to verify whether a character is operator symbol or not.
bool IsOperator(char C)
```

```
{
    if(C == '+' || C == '-' || C == '*' || C == '/')
        return true;

    return false;
}

// Function to perform an operation and return output.
int PerformOperation(char operation, int operand1, int operand2)
{
    if(operation == '+') return operand1 +operand2;
    else if(operation == '-') return operand1 - operand2;
    else if(operation == '*') return operand1 * operand2;
    else if(operation == '/') return operand1 / operand2;

    else cout<<"Unexpected Error \n";
    return -1;
}
```



File Edit Selection View Go Run Terminal Help
SCOB77_Pratham_Pitty_DSA_Assignment3_1.cpp - vs code data - Visual Studio Code
SCOB77_Pratham_Pitty_DSA_Assignment3_1.cpp
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
Code

```
using namespace std;

int EvaluatePostfix(string expression);  // Function to evaluate Postfix expression and return output
int PerformOperation(char operation, int operand1, int operand2);  //// Function to perform an operation and return output.
bool IsOperator(char C);        // Function to verify whether a character is operator symbol or not.
bool IsNumericDigit(char C);        // Function to verify whether a character is numeric digit.

int main()
{
    cout<<"\nSCOB77_Pratham_Pitty_DSA_Assignment3\n\n";
    string expression;
    cout<<"Enter Postfix Expression \n";
    //Enter expression with spaces
    //For eg. 10 20 * 30 40 10 / - +
    //Output = 226
    getline(cin,expression);
    int result = EvaluatePostfix(expression);
    cout<<"Output = "<<result<<"\n";
}

// Function to evaluate Postfix expression and return output
int EvaluatePostfix(string expression)
{
```

```
Warning: PowerShell detected that you might be using a screen reader and has disabled PSReadLine for compatibility purposes. If you want to re-enable it, run 'Import-Module PSReadLine'.

PS C:\Users\prath\vs code data> cd "c:\Users\prath\vs code data\" ; if ($?) { g++ SCOB77_Pratham_Pitty_DSA_Assignment3_1.cpp -o SCOB77_Pratham_Pitty_DSA_Assignment3_1 } ; if ($?) { .\SCOB77_Pratham_Pitty_DSA_Assignment3_1 }

SCOB77_Pratham_Pitty_DSA_Assignment3

Enter Postfix Expression
10 20 * 30 40 10 / - +
Output = 226
PS C:\Users\prath\vs code data>
```