Lab 10

Bibek Dhungana

**CODE**

**main.c**

/\*

FILENAME: main.c

NAME: BIBEK DHUNGANA

DATE: APRIL 7 2021

SPECIFICATION:This simulates the operation of a calculator by scanning an integer

expression in postfix form and displaying its result.

FOR: CS-1412-504

\*/

//including all the required libraries.

#include <stdio.h>

#include <string.h>

#include <ctype.h>

#include <stdlib.h>

#include "stack.h"

//defining the macro constant.

#define SIZE 100

//defining the function prototype

int evaluatePostfix(char\* exp);

int main()

{

//initializing the variables.

char inputExpression[SIZE];

//getting postfix expression from the user.

printf("Please enter the postfix expression:");

fgets(inputExpression,SIZE,stdin);

//printing out the result.

printf ("%d\n", evaluatePostfix(inputExpression));

return 0;

}

/\*

NAME:evaluatePostfix

INPUT: Char\*

RETURN TYPE:int

SPECIFICATION: This takes the postfix expression, perform all the calculation in the stack.

Then, it finally return the result.

\*/

int evaluatePostfix(char\* expression)

{

//creating the stack

Stack\* stack = createStack((int)strlen(expression));

if (!stack){

return -1;

}

    for (int i = 0; i < strlen(expression) - 1; ++i)

{

      //checking if the character is '' or '?'

  if(expression[i]==' ' || expression[i]=='?'){

          continue;

        }

      else if (isdigit(expression[i])){

  int num=0;

//checking if the character is digits

while(isdigit(expression[i])){

num=num\*10 + (int)(expression[i]-'0');

i++;

}

i--;

  push(stack,num);

}

else{

        //getting element from the stack

        int first = pop(stack);

int second = pop(stack);

        //using switch structure

switch (expression[i]){

  case '+':

push(stack, second + first);

break;

case '-':

push(stack, second - first);

break;

case '\*':

push(stack, second \* first);

break;

case '/':

push(stack, second/first);

break;

}

}

}

return pop(stack);

}

**Stack.c**

//including all the required libraries.

#include <stdio.h>

#include <stdlib.h>

#include "stack.h"

//implementation of createStack function

Stack\* createStack(int size)

{

//using malloc function

Stack\* stack = (Stack\*) malloc(sizeof(Stack));

if (!stack){

return NULL;

}

//creating the stack.

stack->head = -1;

stack->size = size;

stack->chain = (int\*) malloc(stack->size \* sizeof(int));

if (!stack->chain){

return NULL;

}

return stack;

}

//implementation of isEmpty function

int isEmpty(Stack\* stack)

{

return stack->head == -1 ;

}

//implementation of pop function

int pop(Stack\* stack)

{

if (!isEmpty(stack))

return stack->chain[stack->head--] ;

return 0;

}

//implementation of push function

void push(Stack\* stack,int input)

{

stack->chain[++stack->head] = input;

}

**Stack.h**

//defining the structure

struct Stack

{

int head;

int size;

int\* chain;

};

//type aliasing the structure Stack with name stack.

typedef struct Stack Stack;

/\*

NAME:createStack

INPUT: int

RETURN TYPE:Stack\*

SPECIFICATION: This function takes int as input argument and return Stack pointer.

\*/

extern Stack\* createStack(int size);

/\*

NAME: isEmpty

INPUT:Stack\*

RETURN TYPE: int

SPECIFICATION: This function takes Stack\* and check if the linked list is empty

\*/

extern int isEmpty(Stack\* stack);

/\*

NAME:pop

INPUT:Stack\*

RETURN TYPE:int

SPECIFICATION:This function takes Stack\* as input and return the top element of the stack

\*/

extern int pop(Stack\* stack);

/\*

NAME:push

INPUT:Stack\*, int

RETURN TYPE:void

SPECIFICATION: This function takes Stack\* and int. Then it pushes that int at top of linked list.

\*/

extern void push(Stack\* stack,int op);

**OUTPUT**

**Text

Description automatically generated**

**Text

Description automatically generated**