**Name: Abdul Ghaffar Kalhoro**

**Registration # 194699**

**Class: BSCS-6C**

DATA STRUCTURES AND ALGORITHMS

ASSIGNMENT 3

**Objective: Evaluating Postfix Expression in C++:**

**Algorithm for postfix evaluation:**

This implementation is done by using arrays and stacks:

* Create an empty stack
* Take the postfix expression as an input
* Read all the symbols one by one from left to right from given postfix expression
* If the reading symbol is operator like: +, -, \* etc.
* then perform TWO pop operations
* and store two pop operations in two different variables like: num1 and num2
* Now perform different symbol operations using num1 and num2 and the result back on the stack
* Now at the end perform pop operation to display the final result of the expression.

**Source Code**

//Assignment#3

#include<cmath>

#include<iostream>

using namespace std;

# define length\_Stack 50 /\* total stack size \*/

# define length\_PostFix 50 /\* total number of characters in postfix expression \*/

//stacking class where stack functions are created

class stacking{

public:

/\* declaring stack array\*/

int stack[length\_Stack];

int max = -1; /\* As array index in C begins at 0 \*/

//function PushData

void PushData(int value)

{

if (max >= length\_Stack - 1)

{

cout << "stack over flow";

return;

}

else

{

++max;

stack[max] = value;

}

}

//PopData function

int PopData()

{

int value;

if (max <0)

{

cout << "stack under flow";

}

else

{

value = stack[max];

--max;

return value;

}

}

};

int main()

{

//object for class stacking

stacking object;

int counter;

char any\_char;

int number;

int num1, num2;

// array of characters to store postfix expression

char array[length\_PostFix];

cout << "~~Evaluation of Postfix expressions~~" << endl;

cout << "Using stacks and arrays contains the operations (+,-,\*,/,^)";

cout << " \nEnter postfix expression,\npress 'z' for end expression : ";

for (counter = 0; counter <= length\_PostFix - 1; counter++)

{ //input of postfix expression

cin >> array[counter];

if (array[counter] == 'z') /\* elimination condition \*/

{

break;

} /\* execute from loop \*/

}

/\* evaluate postfix expression \*/

for (counter = 0; array[counter] != 'z'; counter++)

{

any\_char = array[counter];

if (isdigit(any\_char))

{

object.PushData(any\_char - '0'); //take digit value from character code

}

else if (any\_char == '+' || any\_char == '-' || any\_char == '\*' || any\_char == '/' || any\_char == '^')

{

//pop num1 and num2 from stack

num1 = object.PopData();

num2 = object.PopData();

switch (any\_char) // any\_char is an operator

{

case '/': number = num2 / num1; break;

case '\*': number = num2 \* num1; break;

case '+': number = num2 + num1; break;

case '-': number = num2 - num1; break;

case '^': number = pow(num2, num1); break;

}

//Pushing the data from the stack

object.PushData(number);

}

}

cout << " \n Result of postfix evaluation is : " << object.PopData() << endl;

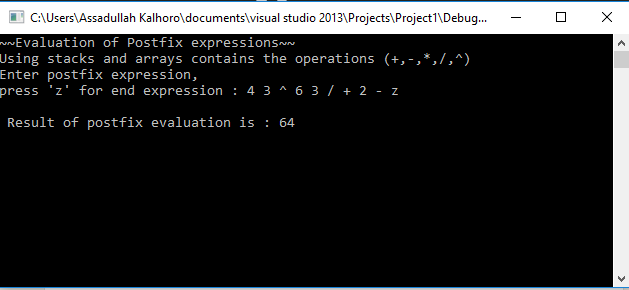
getchar();

getchar();

return 0;

}

**OUTPUT**

****