**Muhammad Abdullah**

**SE(3A) | 19F-0916**

DS LAB

EXPRESSION EVALUATION

**QUESTION # 1:**

**PROGRAM**

#include <iostream>

#include<string> // By M.Abdullah

//Without Using Stack Library, it is tougher for me to understand though i have tried to do it.

using namespace std;

void Vpush(int);

int Vpop(); // For Operand Stack

void Opush(char);

int Opop(); // For Operator Stack

int Precedence(char);

int Operations(int, int, char); // Required Functions

int Evaluation(string);

int \*VArray = NULL; // Operand Stack

int Vtop = -1;

char \*OArray = NULL; // Operator Stack

int Otop = -1;

int main()

{

cout << "Evaluated Infix Expression = " <<Evaluation("(1+3+4)\*(3+1)/(2+2)");

cout << endl << endl;

system("pause");

}

void Vpush(int C)

{

Vtop++;

VArray[Vtop] = C;

}

void Opush(char C)

{

Otop++;

OArray[Otop] = C;

}

int Vpop()

{

int x;

x = VArray[Vtop];

Vtop--;

return x;

}

int Opop()

{

char x;

x = OArray[Otop];

Otop--;

return x;

}

int Precedence(char Operator)

{

if (Operator == '+' || Operator == '-')

return 1;

else if (Operator == '\*' || Operator == '/')

return 2;

else if (Operator == '^')

return 3;

else

return 0;

}

int Operations(int X, int Y, char Operator)

{

switch (Operator)

{

case '+': return X + Y;

case '-': return X - Y;

case '\*': return X \* Y;

case '/': return X / Y;

case '^': return X ^ Y;

}

}

int Evaluation(string Expressions)

{

VArray = new int[Expressions.length()]; //Stack Creation

OArray = new char[Expressions.length()];

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

{

if (Expressions[i] == ' ')

continue; //Ignoring Spaces in Expression

else if (isdigit(Expressions[i]))

{

int exp = 0;

while (i < Expressions.length() && isdigit(Expressions[i]))

{

exp = (exp \* 10) + (Expressions[i] - '0'); // ASCII of 0 Because Expression is in character so

i++; // it needs to be change into int for solving and if

} // there is 2 digit number than this can control it

Vpush(exp);

i--;

}

else if (Expressions[i] == '(')

{

Opush(Expressions[i]);

}

else if (Expressions[i] == ')')

{

while (Otop!=-1 && OArray[Otop]!='(')

{

int X = Vpop();

int Y = Vpop(); // Runs Until () fullfil

char Z = Opop();

Vpush(Operations(Y, X, Z));

}

Opop(); //Poping ( at last

}

else if (Expressions[i] == '+' || Expressions[i] == '-' || Expressions[i] == '\*' || Expressions[i] == '/' || Expressions[i] == '^')

{

if (Otop == -1)

{

Opush(Expressions[i]);

}

else if (Otop != -1 && Precedence(OArray[Otop]) <= Precedence(Expressions[i]))

{

while (Otop != -1 && Precedence(OArray[Otop]) >= Precedence(Expressions[i]))

{

int X = Vpop();

int Y = Vpop(); //Reacting according to Precedence

char Z = Opop();

Vpush(Operations(Y, X, Z));

}

Opush(Expressions[i]);

}

else if(Otop != -1 && Precedence(OArray[Otop]) >= Precedence(Expressions[i]))

Opush(Expressions[i]);

}

}

if (Vtop != -1)

{

while (Otop != -1 )

{

int X = Vpop();

int Y = Vpop(); // If Any Operator Remains

char Z = Opop();

Vpush(Operations(Y, X, Z));

}

}

return Vpop();

}

A screenshot of a computer screen

Description automatically generated

**QUESTION # 2:**

**PROGRAM**

#include <iostream>

#include<string> // By M.Abdullah

//Without Using Stack Library, it is tougher for me to understand though I have tried to do it.

using namespace std;

void Vpush(int);

int Vpop(); // For Operand Stack

int Operations(int, int, char); // Required Functions

int Evaluation(string);

int \*VArray = NULL; // Operand Stack

int Vtop = -1;

int main()

{

cout << "Evaluated Prefix Expression = " <<Evaluation("++\*123-21");

cout << endl << endl;

system("pause");

}

void Vpush(int C)

{

Vtop++;

VArray[Vtop] = C;

}

int Vpop()

{

int x;

x = VArray[Vtop];

Vtop--;

return x;

}

int Operations(int X, int Y, char Operator)

{

switch (Operator)

{

case '+': return X + Y;

case '-': return X - Y;

case '\*': return X \* Y;

case '/': return X / Y;

case '^': return X ^ Y;

}

}

int Evaluation(string Expressions)

{

VArray = new int[Expressions.length()]; //Stack Creation

for (int i = Expressions.length() - 1 ; i >= 0; i--)

{

if (Expressions[i] == ' ')

continue; //Ignoring Spaces in Expression

else if (isdigit(Expressions[i]))

{

Vpush(Expressions[i] - '0'); // ASCII of 0 Because Expression is in character so

// it needs to be change into int for solving

}

else if (Expressions[i] == '+' || Expressions[i] == '-' || Expressions[i] == '\*' || Expressions[i] == '/' || Expressions[i] == '^')

{

int X = Vpop(); //When Operator found, instanly perform action

int Y = Vpop(); // on it then push value into stack of operands

char Z = Expressions[i];

Vpush(Operations(X, Y, Z));

}

else if (Expressions[i] == '(' || Expressions[i] == ')')

{

cout << "ERROR, BRACKETS ARE NOT ENCOUNTERED IN PREFIX EVALUATION !!";

break;

}

}

return Vpop();

}

A screenshot of a computer screen

Description automatically generated

**QUESTION # 3:**

**PROGRAM**

#include <iostream>

#include<string> // By M.Abdullah

//Without Using Stack Library, it is tougher for me to understand though I have tried to do it.

using namespace std;

void Vpush(int);

int Vpop(); // For Operand Stack

int Operations(int, int, char); // Required Functions

int Evaluation(string);

int \*VArray = NULL; // Operand Stack

int Vtop = -1;

int main()

{

cout << "Evaluated Postfix Expression = " <<Evaluation("21-321\*++");

cout << endl << endl;

system("pause");

}

void Vpush(int C)

{

Vtop++;

VArray[Vtop] = C;

}

int Vpop()

{

int x;

x = VArray[Vtop];

Vtop--;

return x;

}

int Operations(int X, int Y, char Operator)

{

switch (Operator)

{

case '+': return X + Y;

case '-': return X - Y;

case '\*': return X \* Y;

case '/': return X / Y;

case '^': return X ^ Y;

}

}

int Evaluation(string Expressions)

{

VArray = new int[Expressions.length()]; //Stack Creation

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

{

if (Expressions[i] == ' ')

continue; //Ignoring Spaces in Expression

else if (isdigit(Expressions[i]))

{

Vpush(Expressions[i] - '0'); // ASCII of 0 Because Expression is in character so

// it needs to be change into int for solving

}

else if (Expressions[i] == '+' || Expressions[i] == '-' || Expressions[i] == '\*' || Expressions[i] == '/' || Expressions[i] == '^')

{

int X = Vpop(); //When Operator found, instanly perform action

int Y = Vpop(); // on it then push value into stack of operands

char Z = Expressions[i];

Vpush(Operations(Y, X, Z));

}

else if (Expressions[i] == '(' || Expressions[i] == ')')

{

cout << "ERROR, BRACKETS ARE NOT ENCOUNTERED IN PREFIX EVALUATION !!";

break;

}

}

return Vpop();

}

A screenshot of a computer screen

Description automatically generated

**QUESTION # 4:**

**PROGRAM**

#include <iostream>

#include<string> // By M.Abdullah

//Without Using Stack Library, it is tougher for me to understand though I have tried to do it.

using namespace std;

void Evaluation(string Expressions);

int \*VArray = NULL; // Operand Stack

int Vtop = -1;

int main()

{

cout << "Given Expression is : "; Evaluation("21-321\*++") ;

cout << endl;

cout << "Given Expression is : "; Evaluation("++\*321-21");

cout << endl;

cout << "Given Expression is : "; Evaluation("(3+2+1)\*(2-1)");

cout << endl << endl;

system("pause");

}

void Evaluation(string Expressions)

{

VArray = new int[Expressions.length()]; //Stack Creation

int j = Expressions.length() - 1, i = 0;

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

{

if (Expressions[i] == ' ')

continue;

else if (Expressions[i] == '+' || Expressions[i] == '-' || Expressions[i] == '\*' || Expressions[i] == '/' || Expressions[i] == '^')

{

cout << "PreFix !";

break;

}

else if (Expressions[j] == '+' || Expressions[j] == '-' || Expressions[j] == '\*' || Expressions[j] == '/' || Expressions[j] == '^')

{

cout << "PostFix !";

break;

}

else

{

cout << "InFix !";

break;

}

}

}

A screenshot of a computer screen

Description automatically generated

**QUESTION # 5:**

**PROGRAM**

#include <iostream>

#include<string> // By M.Abdullah

//Without Using Stack Library, it is tougher for me to understand though I have tried to do it.

using namespace std;

char \*OArray = NULL; // Operator Stack

int Otop = -1;

void Opush(char C);

int Opop();

int Precedence(char Operator);

int Check(string Expressions); // Required Function

string In\_To\_Post(string Expression);

string In\_To\_Pre (string Expression);

string Pre\_To\_In (string Expression);

string Post\_To\_In(string Expression);

int main()

{

string Expression;

int opt,i;

cout << "----------------------------------------------" << endl;

cout << endl << "Write your Expression : ";

cin >> Expression;

cout << endl << "Given Expression is : "; i = Check(Expression);

cout << endl;

switch (i)

{

case 1:

{

cout << endl;

cout << " Press 1 to Change into Infix !! " << endl;

cout << " Press 2 to Change into PostFix !! " << endl;

cin >> opt;

break;

}

case 2:

{

cout << endl;

cout << " Press 1 to Change into Infix !! " << endl;

cout << " Press 2 to Change into Prefix !! " << endl;

cin >> opt;

break;

}

case 3:

{

cout << endl;

cout << " Press 1 to Change into Prefix !! " << endl;

cout << " Press 2 to Change into Postfix !! " << endl;

cin >> opt;

if (opt == 1)

{

}

else if (opt == 2)

{

cout << endl << "PostFix = " << In\_To\_Post(Expression);

}

else

cout << endl << "Invalid Entry !!" << endl;

break;

}

default:

cout << "Invalid Entry !! " << endl;

}

cout << endl << endl;

system("pause");

}

void Opush(char C)

{

Otop++;

OArray[Otop] = C;

}

int Opop()

{

char x;

x = OArray[Otop];

Otop--;

return x;

}

int Precedence(char Operator)

{

if (Operator == '+' || Operator == '-')

return 1;

else if (Operator == '\*' || Operator == '/')

return 2;

else if (Operator == '^')

return 3;

else

return 0;

}

int Check(string Expressions)

{

int j = Expressions.length() - 1, i = 0;

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

{

if (Expressions[i] == ' ')

continue;

else if (Expressions[i] == '+' || Expressions[i] == '-' || Expressions[i] == '\*' || Expressions[i] == '/' || Expressions[i] == '^')

{

cout << "PreFix !";

return 1;

break;

}

else if (Expressions[j] == '+' || Expressions[j] == '-' || Expressions[j] == '\*' || Expressions[j] == '/' || Expressions[j] == '^')

{

cout << "PostFix !";

return 2;

break;

}

else

{

cout << "InFix !";

return 3;

break;

}

}

}

string In\_To\_Post(string Expression)

{

OArray = new char[Expression.length()];

string Final;

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

{

if ((Expression[i] >= 'a' && Expression[i] <= 'z') || (Expression[i] >= 'A' && Expression[i] <= 'Z') ||

(Expression[i] >= '1' && Expression[i] <='9' ))

{

Final = Final + Expression[i];

}

else if (Expression[i] == '(')

{

Opush(Expression[i]);

}

else if (Expression[i] == ')')

{

while (Otop != -1 && OArray[Otop] == '(')

{

char opt = Opop();

Final = Final + opt;

}

Opop();

}

else

{

while (Otop != -1 && Precedence(OArray[Otop]) >= Precedence(Expression[i]))

{

char opt = Opop();

Final = Final + opt;

}

Opush(Expression[i]);

}

}

while (Otop != -1)

{

char opt = Opop();

Final = Final + opt;

}

return Final;

}

A screenshot of a computer screen

Description automatically generated