//

// main.cpp

// Stack

// Created by Stephanie Marin Velasquez on 4/11/16.

// Copyright © 2016 Stephanie Marin Velasquez. All rights reserved.

#include <iostream>

#include "stack.h"

using namespace std;

void func(ArrayStack);

void menu();

void func2(int,ArrayStack);

//void display();

ArrayStack stack;

int main()

{

func(stack);

}

void func(ArrayStack a)

{

string newinput="";

string number1 = "";

string number2 = "";

int num1 = 0;

int num2 = 0;

int result = 0;

string expression = ' ';

//int temp2 = 0;

cout<<"Please enter in a postfix expression: ";

cin.clear();

cin>>newinput;

string temp[3];

temp[0] = newinput[0];

char temp2[3];

int count = 0;

for(int i=1; i<newinput.length();i++){

int j =0;

while (newinput[i] != ' ')

{

if (newinput[i]>='0' && newinput[i]<='9')

{

temp[j]= temp[j] + newinput[i];

}

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

{

expression=newinput[i];

}

}

j++;

}

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

{

if ((number1[i]>='0' && number1[i]<='9' ) || (number2 [i] >='0' && number2[i]<='9' )) // if char are in this category of char for numbers

{

num1= num1\*10+number1[i]-48;

a.push(num1);

num2= num2 \*10+number2[i]-48;

a.push(num2);

}

if(newinput[i] == '+' || newinput[i] == '-' || newinput[i] == '\*' || newinput[i] == '/'){

a.pop(num1);

a.pop(num2);

}

if(expression=='+')

result = num1+num2;

else if(expression == '-')

result = num1-num2;

else if(expression == '\*')

result = num1\*num2;

else if(expression == '/')

result = num1/num2;

cout<< result;

a.push(result);

}

}

/\*void evaluateExpression()

{

}\*/

/\*void evaluateExp(ifstream inpF, ostream& out, ArrayStack& stack, char& ch, bool& isExpOk)

{

char ch;

while(ch != '=')

{

switch (ch)

{

case '#':

}

}

}

\*/

void menu()

{

cout<<"1 - Push \n2 - Pop \n3 - Display List\n4 - Quit\nChoice: ";

}

try 3 with help

// main.cpp

// postfix

//

// Created by Stephanie Marin Velasquez on 5/9/16.

// Copyright © 2016 Stephanie Marin Velasquez. All rights reserved.

//

#include <iostream>

#include <string>

using namespace std;

int main(int argc, const char \* argv[]) {

string input="";

string number1 = "";

string number2 = "";

int num1=0;

int num2=0;

int result = 0;

char orador = ' ';

int temp = 0;

int temp2=0;

cin>>input;

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

if(input[i] == ' '){

temp = i+1;

}

if(input[i] == '+' || input[i] == '-' || input[i] == '\*' || input[i] == '/'){

temp2 = i;

}

}

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

if(i < temp)

number1=number1+input[i];

if(i > temp-1 && i < temp2)

number2=number2+input[i];

if(i == temp2)

orador = input[i];

}

num1 = stoi(number1);

num2 = stoi(number2);

if(orador=='+')

result = num1+num2;

else if(orador == '-')

result = num1-num2;

else if(orador == '\*')

result = num1\*num2;

else if(orador == '/')

result = num1/num2;

cout<<result;

return 0;

}

/ TRY 3

/ main.cpp

// Stack

// Created by Stephanie Marin Velasquez on 4/11/16.

// Copyright © 2016 Stephanie Marin Velasquez. All rights reserved.

#include <iostream>

using namespace std;

bool isOperator(char ch);

// Array-based Stack-----

const int MAX=10;

class ArrayStack

{

public:

ArrayStack(): top(-1){}; //in-line

~ArrayStack(){};

void push( int x, bool &success);

int pop(int &x, bool &success);

int getTop(int &x, bool &success)const;

void display(ArrayStack & s);

bool isEmpty();

bool isFull();

private:

int data[MAX];

int top;

};

ArrayStack stack;

void ArrayStack:: push(int x, bool &success)

{

success= (top < MAX);

if (success)

{

top++;

data[top]=x;

}

else

cout<<"You cannot add to a full stack."<<endl;

}

int ArrayStack:: pop(int &x, bool &success)

{

success= (top>=0);

if (success)

{

x=data[top];

top--;

}

return x;

}

int ArrayStack:: getTop(int &x, bool &success)const

{

success= top>-1;

if (success)

x=data[top];

return x;

}

bool ArrayStack:: isEmpty()

{

if (top==-1)

return true;

else

return false;

}

bool ArrayStack:: isFull()

{

if (top==MAX)

return true;

else

return false;

}

///////////////////////////////////////////////////////

bool isOperator(char ch)

{

if (ch=='+' || ch=='-' || ch=='\*' || ch=='/' || ch=='(' || ch==')' )

return true;

else

return false;

}

bool isOperand(char ch)

{

if ( ch >= '0' || ch <= '9')

return true;

else

return false;

}

char expression[100];

int cnt=0, op1, op2,result;

unsigned long len;

bool success=true;

int x=0;

char orador=' ';

void convert()

{

len= strlen(expression);

for (int i=0; i< len; i++)

{

if (expression[i]>='0' && expression[i]<='9') // if char are in this category of char for numbers

{

x= x\*10+expression[i]-48; // change from char to int

int num[cnt];

num[i]=x;

cnt++;

cout<< "the number is: "<< num[i];

/\*stack.push(x, success);

x= stack.pop(x, success);\*/

//int opt1;

//opt1= s.getTop(x, success)const;

//s.pop(x,success);

}

else if ( isOperator(expression[i]) )

{

orador=expression[i];

char ch[cnt];

ch[i]=orador;

cout<< ch[i]<<endl;

//opt1=s.getTop(x, success)const

}

cnt++;

}

// cout<<"\n counter= "<<cnt<<endl;

}

void calculator()

{

int num[cnt];

char ch[cnt];

len=strlen(expression);

for (int i=0; i<len; i++)

{

if (num[i] != isOperator(num[i]))

stack.push(x,success);

else if (num[i]== isOperator(num[i]))

{

op1= stack.getTop(x, success);

stack.pop(x,success);

}

}

}

int main()

{

// when the stack is empty

while(!stack.isFull())

{

if(stack.isEmpty())

cout<< "\n The stack is empty "<<endl;

cout<< "\*\* Enter the post-fix value" <<endl;

cin>> expression;

convert(); //changes from char to integer

}

}

void ArrayStack:: getArray()const

{

int x=NULL, cnt=0;

char ch;

for(int i=0; i< size; i++ )

{

if ( data[i]>='0' && data[i]<='9') // if char are in this category of char for numbers

{

x= x \*10+data[i]-48; // change from char to int

cout<< x;

}

else if ( isOperator( data[i]) )

{

ch=data[i];

cout<<ch;

}

cnt++;

}

}

// main.cpp

// Stack

// Created by Stephanie Marin Velasquez on 4/11/16.

// Copyright © 2016 Stephanie Marin Velasquez. All rights reserved.

#include <iostream>

using namespace std;

// Array-based Stack-----

const int size=10;

class ArrayStack

{

public:

ArrayStack(): top(-1){}; //in-line

~ArrayStack();

char getArray()const;

void push( int x, bool &success);

void pop(int &x, bool &success);

void getTop(int &x, bool &success)const;

void display(ArrayStack & s);

bool isEmpty();

private:

int data[size];

int top;

};

ArrayStack s;

char ArrayStack:: getArray()const

{

return data;

}

void ArrayStack:: push(int x, bool &success)

{

success= (top < size);

if (success)

{

top++;

data[top]=x;

}

}

void ArrayStack:: pop(int &x, bool &success)

{

success= (top>=0);

if (success)

{

x=data[top];

top--;

}

}

void ArrayStack:: getTop(int &x, bool &success)const

{

success= top>-1;

if (success)

x=data[top];

}

void ArrayStack:: display(ArrayStack &s)

{

ArrayStack t;

while (! s.isEmpty())

{

int x; bool success;

s.pop(x, success);

cout<<x;

t.push( x, success);

}

while (!t.isEmpty()) {

int x; bool success;

t.pop (x, success);

s.push( x, success);

}

}

bool ArrayStack:: isEmpty()

{

return (top==-1);

}

bool isOperator(char ch)

{

if (ch=='+' || ch=='-' || ch=='\*' || ch=='/' || ch=='(' || ch==')' )

return true;

else

return false;

}

bool isOperand(char ch)

{

if ( ch >= '0' || ch <= '9')

return true;

else

return false;

}

char expression[100];

int cnt=0, op1, op2,result;

unsigned long len;

bool success=false;

int x=NULL;

char orador=' ';

void convert()

{

cout<< "Enter the expression: \n";

cin>> expression;

len= strlen(expression);

for (int i=0; i< len; i++)

{

// int x=0;

if (expression[i]>='0' && expression[i]<='9') // if char are in this category of char for numbers

{

x= x\*10+expression[i]-48; // change from char to int

cout<<x;

// s.push(x, success);

//x= expression[i];

//int opt1;

//opt1= s.getTop(x, success)const;

//s.pop(x,success);

}

else if ( isOperator(expression[i]) )

{

orador=expression[i];

cout<< orador;

//opt1=s.getTop(x, success)const

}

cnt++;

}

// cout<<"\n counter= "<<cnt<<endl;

}

/\*

void calculator()

{

len=strlen(expression);

for (int i=0; i<len; i++)

{

if (expression[i]== x)

s.push(x,success);

else

{

// op2= s.getTop(x, success);

// s.pop(x,success);

}

}

}\*/

int main()

{

convert();

// calculator();

//for (int i=0; i< )

/\* int i=0;

char ch;

int val;

while (i<size)

{

}

\*/

}

Other idea for postfix expression- result

TRY 3// HERE IS A HELP

// main.cpp

// Stack

// Created by Stephanie Marin Velasquez on 4/11/16.

// Copyright © 2016 Stephanie Marin Velasquez. All rights reserved.

#include <iostream>

using namespace std;

int multiplication();

// Array-based Stack-----

const int size=10;

class ArrayStack

{

public:

ArrayStack(): top(-1){}; //in-line

~ArrayStack();

void push( int x, bool &success);

void pop(int &x, bool &success);

void getTop(int &x, bool &success)const;

bool isEmpty();

bool isFull();

private:

int data[size];

int top;

};

ArrayStack num;

void ArrayStack:: push(int x, bool &success)

{

success= (top < size);

if (success)

{

top++;

data[top]=x;

}

}

void ArrayStack:: pop(int &x, bool &success)

{

success= (top>=0);

if (success)

{

x=data[top];

top--;

}

}

void ArrayStack:: getTop(int &x, bool &success)const

{

success= top>-1;

if (success)

x=data[top];

}

bool priorityoperator(char value)

{

switch (value) {

/\*case '(':

case ')':

value=1;

break;\*/

case '/':

case '\*':

value = 2;

break;

case '+':

case '-':

value=3;

break;

default:

cout<< "invalide operator\n";

break;

}

return value;

}

void convert()

{

char expression[size];

int cnt=0;

cout<< "Enter the expression: \n";

cin>> expression;

int orand=NULL;

char orador=' ';

for (int i=0; i< sizeof(expression); i++)

{

if (expression[i]>='0' && expression[i]<='9') // if char are in this category of char for numbers

{

orand= orand\*10+expression[i]-48; // change from char to int

cout<<orand;

}

else if (expression[i] >='(' && expression[i] <='/')

{

orador=expression[i];

cout<< orador;

}

cnt++;

}

// cout<<"\n counter= "<<cnt<<endl;

}

void solution()

{

char expression[size];

int cnt=0;

int orand=NULL;

char orador=' ';

for (int i=0; i< sizeof(expression); i++)

{

if (expression[i]>=0 && expression[i]<=9) // if char are in this category of char for numbers

{

}

else if (expression[i] >='(' && expression[i] <='/')

{

orador=expression[i];

cout<< orador;

}

cnt++;

}

// cout<<"\n counter= "<<cnt<<endl;

}

int main()

{

convert();

int number[size];

int x; bool success=true;

for (int i=0; i< size; i++)

{

if (number [i]>=0 && number[i] <=9)

{

num.push( x, &success);

}

else if ()

}

}

TRY 2 //

// main.cpp

// Stack

// Created by Stephanie Marin Velasquez on 4/11/16.

// Copyright © 2016 Stephanie Marin Velasquez. All rights reserved.

#include <iostream>

using namespace std;

int multiplication();

// Array-based Stack-----

const int size=10;

class ArrayStack

{

public:

ArrayStack(): top(-1){}; //in-line

~ArrayStack();

void push( int x, bool &success);

void pop(int &x, bool &success);

void getTop(int &x, bool &success)const;

private:

int data[size];

int top;

};

ArrayStack num;

void ArrayStack:: push(int x, bool &success)

{

success= (top < size);

if (success)

{

top++;

data[top]=x;

}

}

void ArrayStack:: pop(int &x, bool &success)

{

success= (top>=0);

if (success)

{

x=data[top];

top--;

}

}

void ArrayStack:: getTop(int &x, bool &success)const

{

success= top>-1;

if (success)

x=data[top];

}

bool priorityoperator(char value)

{

switch (value) {

case '/':

case '\*':

value = 1;

break;

case '+':

case '-':

value=2;

break;

case '(':

case ')':

value=3;

break;

default:

cout<< "invalide operator\n";

break;

}

return value;

}

void convert()

{

char expression[size];

int cnt=0;

cout<< "Enter the expression: \n";

cin>> expression;

int orand=NULL;

char orador=' ';

int x;

//char c;

bool success;

for (int i=0; i< sizeof(expression); i++)

{

if (expression[i]>='0' && expression[i]<='9') // print numbers in int

{

orand= orand\*10+expression[i]-48; // change from char to int

num.push(x, success);

cout<<orand;

}

else if (expression[i] >='(' && expression[i] <='/')

{

orador=expression[i];

if ( priorityoperator(orador) == 1)

multiplication();

// num.pop(&x, &success);

cout<< orador;

}

cnt++;

}

// cout<<"\n counter= "<<cnt<<endl;

}

int multiplication()

{

int x; bool success;

num.pop(&x, &success);

num.pop(&x, bool &success);

}

int main()

{

convert();

int number[size];

for (int i=0; i< size; i++)

{

if (number [i]>=0 && number[i] <=9)

else if ()

}

}

TRY 1 --- THIS IS BETTER FOR PROBLEM 2 BECAUSE IT NEEDS THE PARENTHESIS

// main.cpp

// Stack

//

// Created by Stephanie Marin Velasquez on 4/11/16.

// Copyright © 2016 Stephanie Marin Velasquez. All rights reserved.

//

#include <iostream>

using namespace std;

// Array-based Stack-----

const int size=20;

class ArrayStack

{

ArrayStack(): top(-1){}; //in-line

~ArrayStack();

void push( int x, bool &success);

void pop(int &x, bool &success);

void getTop(int &x, bool &success)const;

private:

int data[size];

int top;

};

void ArrayStack:: push(int x, bool &success)

{

success= (top < size);

if (success)

{

top++;

data[top]=x;

}

}

void ArrayStack:: pop(int &x, bool &success)

{

success= (top>=0);

if (success)

{

x=data[top];

top--;

}

}

void ArrayStack:: getTop(int &x, bool &success)const

{

success= top>-1;

if (success)

x=data[top];

}

// end Array-based stack

// Pointer-based stack \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

/\*

class PoiStack{

PoiStack() : top(0){};

~PoiStack(); // have to erase manually

void push (int x);

void pop (int &x, bool &success);

void getTop(int &x, bool &success);

private:

struct stackNode{

int data;

stackNode \* next;

};

stackNode \*top;

};

PoiStack::~PoiStack()

{

stackNode \* delnode;

while (top)

{

delnode=top;

top=top->next;

delete delnode;

}

}

void PoiStack:: push(int x)

{

stackNode \* tmp= new stackNode;

tmp->data=x; //assign data field

tmp->next=top; // connect the new node to top

tmp=top; // assign top to tmp

}

void PoiStack:: pop(int &x, bool &success)

{

success= top!=0;

if (success)

{

stackNode \*delnode;

x=top->data;

delnode= top;

top= top->next;

delete delnode;

}

}

void PoiStack:: getTop(int &x, bool &success)

{

success= top!=0;

if (success)

x=top->data;

}

// end of pointer-based stack \*\*\*\*\*\*\*\*\*\*\*

\*/

bool priorityoperator(char symbol)

{

switch (symbol) {

case '/':

case '\*':

symbol = 1;

break;

case '+':

case '-':

symbol=2;

break;

case '(':

case ')':

symbol=3;

break;

default:

cout<< "invalide operator\n";

break;

}

return symbol;

}

/\*int a[5], top=-1;

void push (int value)

{

if (top==4)

{

cout<< "stack is full ";

}

else {

top=top+1;

a[top]=value;

}

}

void pop()

{

if (top==-1)

{

cout<< "stack is empty";}

else top=top-1;

}\*/

/\*void push( int value[])

{

for (int i=0; i<)

if (int value[])

}

\*/

int main()

{

int max=20;

char expression[max];

int cnt=0;

cout<< "Enter the expression: \n";

cin>> expression;

int value=0;

for (int i=0; i< sizeof(expression); i++)

{

if (expression[i]>='0' && expression[i]<='9') // print numbers in int

{

value= value\*10+expression[i]-48;

cnt++;

// push( &value[sizeof(expression)]);

cout<<value;

}

if (expression[i] == '+')

cout<< "+";

}

cout<<endl;

/\*

char symbol;

cout<< "operandor";

cin>> symbol;

bool priorityOperator( char symbol); \*/

}