Вар 1  
Проверить правильность расстановки скобок в арифметическом выражении (допустимы четыре типа скобок: круглые, квадратные, фигурные и угловые). Для слежения за скобками воспользоваться стеком.

#include <iostream>

#include<string>

using namespace std;

struct krug

{

int q; // Информационный элемент

krug\* next, \* head; // Начало стека и указатель на следущий элемент

};

struct kvadrat

{

int w; // Информационный элемент

kvadrat\* nexT, \* heaD; // Начало стека и указатель на следущий элемент

};

struct figur

{

int e; // Информационный элемент

figur\* neXt, \*heAd; // Начало стека и указатель на следущий элемент

};

struct treug

{

int r; // Информационный элемент

treug\* nExt, \* hEad; // Начало стека и указатель на следущий элемент

};

void Addkrug(char q, krug\*& Mykrug) //круг скобка

{

krug\* Temp = new krug; // Элемент стека и указатель на стек

Temp->q = q; // Выделяем память для элемента

Temp->next = Mykrug->head; // Указываем, что следущий элемент это предыдущий

Mykrug->head = Temp; //Сдвигаем началный эелемент вперед

}

void Addkvadrat(int w, kvadrat\*& Mykvadrat)

{

kvadrat\* temp = new kvadrat; // Элемент стека и указатель на стек

temp->w = w; // Выделяем память для элемента

temp->nexT = Mykvadrat->heaD; // Указываем, что следущий элемент это предыдущий

Mykvadrat->heaD = temp; //Сдвигаем началный эелемент вперед

}

void Addfigur(int e, figur\*& Myfigur)

{

figur\* temP = new figur; // Элемент стека и указатель на стек

temP->e = e; // Выделяем память для элемента

temP->neXt = Myfigur->heAd; // Указываем, что следущий элемент это предыдущий

Myfigur->heAd = temP; //Сдвигаем началный эелемент вперед

}

void Addtreug(int r, treug\*& Mytreug)

{

treug\* teMp = new treug; // Элемент стека и указатель на стек

teMp->r = r; // Выделяем память для элемента

teMp->nExt = Mytreug->hEad; // Указываем, что следущий элемент это предыдущий

Mytreug->hEad = teMp; //Сдвигаем началный эелемент вперед

}

void proverka(krug\*& Mykrug)

{

krug\* Temp = Mykrug->head; //Объявляем указатель и указываем ему, что его позиция в голове стека

while (Temp != NULL) //Условие пока указатель имеет значение

{

if (Temp->q == 41) // Если элемент стека равен )

{

Temp = Temp->next; //Переходим к следующему элементу

if (Temp->q == 40) // Если элемент стека равен (

{

Temp = Temp->next; //Переходим к следующему элементу

if (Temp == NULL) // Если у указателя больше нет значений

{

cout << "Круглые скобки верны" << endl;

}

}

else

{

cout << "Круглые скобки неверны! Перепроверьте выражение" << endl; break;

}

}

else

{

cout << "Круглые скобки неверны! Перепроверьте выражение" << endl; break;

}

}

}

void proverka1(kvadrat\*& Mykvadrat)

{

kvadrat\* temP = Mykvadrat->heaD; //Объявляем указатель и указываем ему, что его позиция в голове стека

while (temP != NULL) //Условие пока указатель имеет значение

{

if (temP->w == 93) // Если элемент стека равен ]

{

temP = temP->nexT; //Переходим к следующему элементу

if (temP->w == 91) // Если элемент стека равен [

{

temP = temP->nexT; //Переходим к следующему элементу

if (temP == NULL) // Если у указателя больше нет значений

{

cout << "Квадратные скобки верны" << endl;

}

}

else

{

cout << "Квадратные скобки неверны! Перепроверьте выражение" << endl; break;

}

}

else

{

cout << "Квадратные скобки неверны! Перепроверьте выражение" << endl; break;

}

}

}

void proverka2(figur\*& Myfigur)

{

figur\* teMp = Myfigur->heAd; //Объявляем указатель и указываем ему, что его позиция в голове стека

while (teMp != NULL) //Условие пока указатель имеет значение

{

if (teMp->e == 125) // Если элемент стека равен }

{

teMp = teMp->neXt; //Переходим к следующему элементу

if (teMp->e == 123) // Если элемент стека равен {

{

teMp = teMp->neXt; //Переходим к следующему элементу

if (teMp == NULL) // Если у указателя больше нет значений

{

cout << "Фигурные скобки верны" << endl;

}

}

else

{

cout << "Фигурные скобки неверны! Перепроверьте выражение" << endl; break;

}

}

else

{

cout << "Фигурные скобки неверны! Перепроверьте выражение" << endl; break;

}

}

}

void proverka3(treug\*& Mytreug)

{

treug\* tEmp = Mytreug->hEad; //Объявляем указатель и указываем ему, что его позиция в голове стека

while (tEmp != NULL) //Условие пока указатель имеет значение

{

if (tEmp->r == '>') // Если элемент стека равен >

{

tEmp = tEmp->nExt; //Переходим к следующему элементу

if (tEmp->r == '<') // Если элемент стека равен <

{

tEmp = tEmp->nExt; //Переходим к следующему элементу

if (tEmp == NULL) // Если у указателя больше нет значений

{

cout << "Треугольные скобки верны" << endl;

}

}

else

{

cout << "Треугольные скобки неверны! Перепроверьте выражение" << endl; break;

}

}

else

{

cout << "Треугольные скобки неверны! Перепроверьте выражение" << endl; break;

}

}

}

int main()

{

setlocale(LC\_ALL, "Russian"); //Подключаем русский язык

krug\* Mykrug = new krug; // Выделяем память для стека

kvadrat\* Mykvadrat = new kvadrat; // Выделяем память для стека

figur\* Myfigur = new figur; // Выделяем память для стека

treug\* Mytreug = new treug; // Выделяем память для стека

string s;

int g;

char n,v;

Mykrug->head = NULL; //Инициализируем первый элемент

Mykvadrat->heaD = NULL; //Инициализируем первый элемент

Myfigur->heAd = NULL; //Инициализируем первый элемент

Mytreug->hEad = NULL; //Инициализируем первый элемент

cout << "Введите число:";

cin >> s; // Вводим S

cout << "s= " << s << endl;

n = s.length(); // n = длине строки S

v = 0;

for (int m = 0; m < n; m++)

{

v = s[m]; // V = элементу строки S под номером m

if (v == '(') // Если равен (

{

Addkrug(v, Mykrug); // Обращаемся к функции

}

if (v == ')' ) // Если равен )

{

Addkrug(v, Mykrug); // Обращаемся к функции

}

if (v == '{') // Если равен {

{

Addfigur(v, Myfigur); // Обращаемся к функции

}

if (v == '}') // Если равен }

{

Addfigur(v, Myfigur); // Обращаемся к функции

}

if (v == '[') // Если равен [

{

Addkvadrat(v, Mykvadrat); // Обращаемся к функции

}

if (v == ']') // Если равен ]

{

Addkvadrat(v, Mykvadrat); // Обращаемся к функции

}

if (v == '<') // Если равен <

{

Addtreug(v, Mytreug); // Обращаемся к функции

}

if (v == '>') // Если равен >

{

Addtreug(v, Mytreug); // Обращаемся к функции

}

}

proverka(Mykrug); // Выводим результат

proverka1(Mykvadrat); // Выводим результат

proverka2(Myfigur); // Выводим результат

proverka3(Mytreug); // Выводим результат

system("pause");

}

