ЦЕЛЬ РАБОТЫ

Продемонстрировать работу с механизмом шаблонов в объектно-ориентированном программировании.

ПРАКТИЧЕСКОЕ ЗАДАНИЕ

Реализовать шаблон класса согласно варианту.

Объект описать минимально возможным количеством полей, остальные характеристики вычислять динамически с помощью методов. Класс должен содержать минимум 2 конструктора: конструктор по умолчанию и параметризованный. Перегрузить операторы записи и чтения потока, для обеспечения потокового ввода-вывода. Перегрузить минимум 2 стандартных оператора (унарный и бинарный).

Написать программу, выполняющую:

1. Создание 2-х экземпляров класса, с динамическим выделением памяти, первый с параметрами по умолчанию, второй с параметрами заданными пользователем;

2. Вывод всех характеристик заданного экземпляра класса на экран;

3. Демонстрацию работы перегруженных операторов;

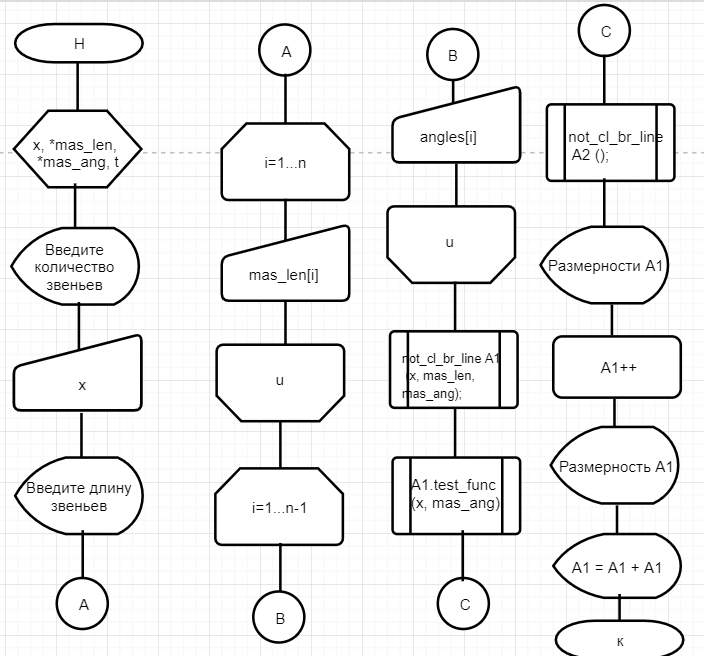
4. Освобождение выделенной памяти.

ИНДИВИДУАЛЬНОЕ ЗАДАНИЕ

Вариант 10. Ломаная линия

БЛОК-СХЕМА АЛГОРИТМА

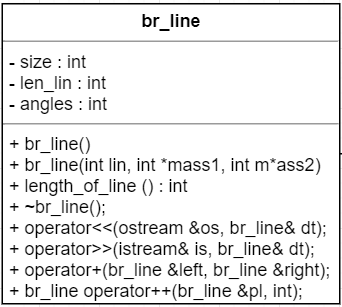
Блок-схема алгоритма представлена на рисунке 1.



*Рисунок 1 Блок-схема программы*

ДИАГРАММА КЛАССОВ

Диаграмма классов представлена на рисунке 2.



*Рисунок 2 Диаграмма классов*

ЛИСТИНГ

#include <iostream>

#include <string>

using namespace std;

template <class T> class br\_line {

private:

int size;//количество звеньев

T \*len\_lin;//массив из длин звеньев

T \*angles;//массив из углов между звеньями

public:

br\_line() { //конструктор по умолчанию

size = 3;

len\_lin = new int[size];

len\_lin[0] = 1; len\_lin[1] = 1; len\_lin[2] = 1;

angles = new int[size];

angles[0] = 1; angles[1] = 1;

}

br\_line(const br\_line &l) {

size = l.size;

len\_lin = new int[size];

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

len\_lin[i] = l.len\_lin[i];

}

angles = new int[size - 1];

for (int i = 0; i < (size - 1); i++) {

angles[i] = l.angles[i];

}

}

br\_line(int lin, int \*mass1, int \*mass2) { //параметризированный конструктор

size = lin;

len\_lin = new int[size];

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

len\_lin[i] = mass1[i];

}

angles = new int[size - 1];

for (int i = 0; i < (size - 1); i++) {

angles[i] = mass2[i];

}

}

~br\_line() {

delete[] len\_lin;

delete[] angles;

};

int length\_of\_line()

{

int sum = 0;

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

{

sum = sum + len\_lin[n];

}

return sum;

}//функция высчитывает длину ломанной

friend br\_line <T> operator+(br\_line &left, br\_line &right);

friend ostream& operator<<(ostream &os, br\_line& dt);

friend istream& operator>>(istream& is, br\_line& dt);

friend br\_line <T> operator++(br\_line &pl, int);

};

br\_line <int> operator+(br\_line<int> &left, br\_line<int> &right)//перегрузка бинарного прератора

{

br\_line <int> sum;

sum.size = left.size + right.size;

return sum;

}

br\_line <int> operator++(br\_line <int> &pl, int) //перегрузка унарного оператора

{

pl.size++;

return pl;

}

ostream& operator<<(ostream &os, br\_line <int> &dt) //перегрузка оператора вывода

{

os << "dlina linii" << endl;

os << dt.length\_of\_line() << endl;

os << "kolichestvo zven'ev" << endl;

os << dt.size << endl;

return os;

}

istream& operator>>(istream& is, br\_line <int> &dt) //перегрузка оператора ввода

{

is >> dt.size;

return is;

}

int main(void) {

setlocale(LC\_ALL, "Russian");

int x, \*mas\_len, \*mas\_ang;//x-количество звеньев, mas\_len-массив под длины звеньев

cout << "Введите количество звеньев" << endl;

cin >> x;

mas\_len = new int[x];

cout << "Введите длину звеньев" << endl;

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

{

cin >> mas\_len[i];

}

mas\_ang = new int[x - 1];

cout << "Введите размерность углов в градусах (на один меньше, звеньев)" << endl;

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

{

cin >> mas\_ang[i];

}

br\_line <int> A1 (x, mas\_len, mas\_ang);//вызов параметризированно конструктора

cout << A1 << endl; //демонстация работы перегрузки оператора вывода

cout << "Введите новое количество звеньев" << endl;

cin >> A1; //демонстрация работы перегрузки оператора ввода

cout << "Новое количество звеньев + 1" << endl;

A1++; //демонстрация работы перегрузки инкремента

cout << A1 << endl;

cout << "Вывод конструктора по умолчанию A2" << endl;

br\_line <int> A2; //вызов конструктора по умолчанию

cout << A2 << endl;

A2 = A2 + A2; //демонстрация работы перегрузки бинарного оператора

cout << "вывод перегрузки '+' A2" << endl;

cout << A2 << endl;

delete[] mas\_len;

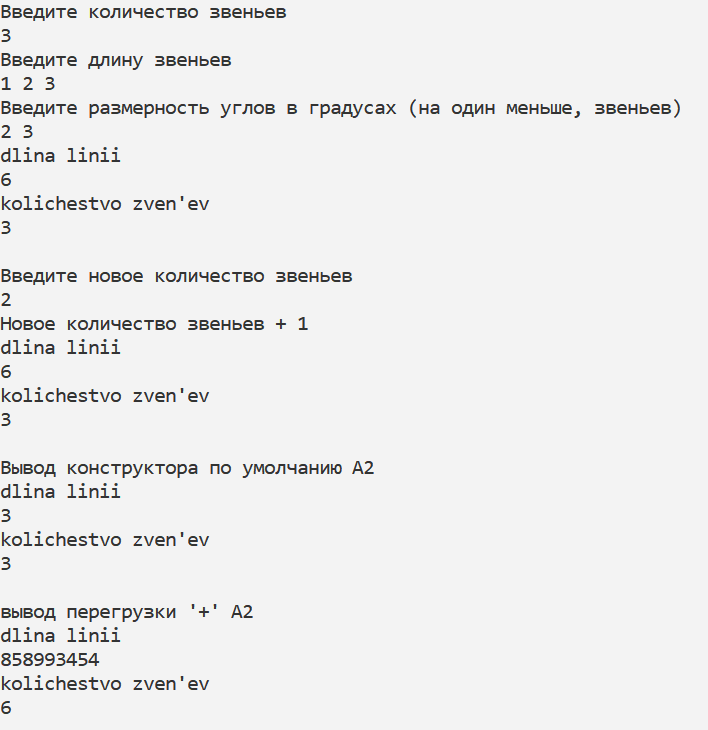
delete[] mas\_ang;

system("pause");

}

ПРОТОКОЛ ВЫПОЛНЕНИЯ ПРОГРАММЫ

Протокол выполнения программы представлен на рис.3.



*Рисунок 3 Протокол программы*