ЦЕЛЬ РАБОТЫ

Продемонстрировать работу с механизмом наследования в объектно-ориентированном программировании.

ПРАКТИЧЕСКОЕ ЗАДАНИЕ

Реализовать дочерний класс 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;

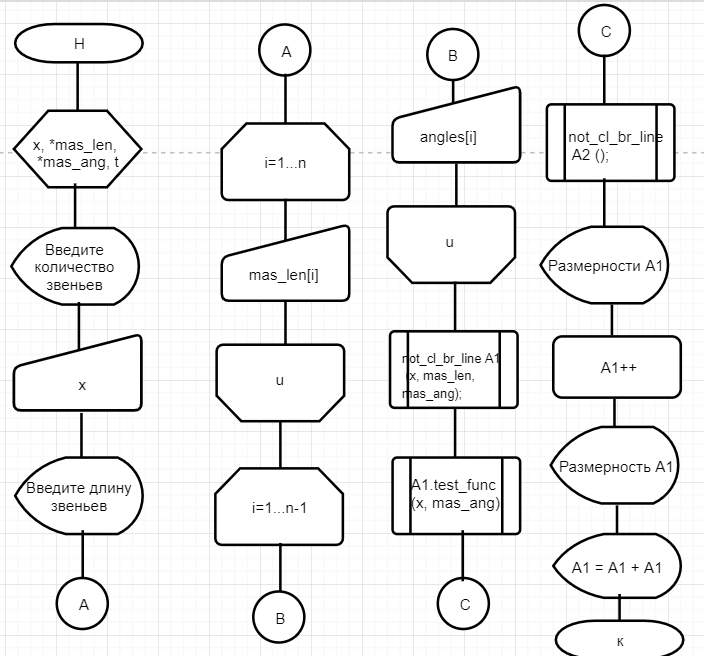
4. Освобождение выделенной памяти.

ИНДИВИДУАЛЬНОЕ ЗАДАНИЕ

Вариант 10. Ломаная линия

БЛОК-СХЕМА АЛГОРИТМА

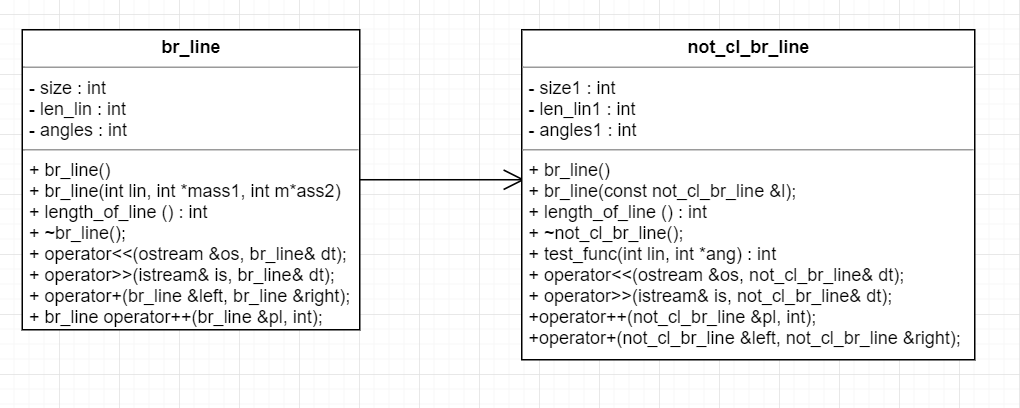
Блок-схема алгоритма представлена на рисунке 1.



*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;

class br\_line{

private:

int size;//количество звеньев

int \*len\_lin;//массив из длин звеньев

int \*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(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];

}

}//lin\_k-количество звеньев

~br\_line() {};

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 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 operator++(br\_line &pl, int);

};

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

{

br\_line sum;

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

return sum;

}

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

{

pl.size++;

return pl;

}

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

{

os << "Длина линии" << endl;

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

os << "Количество звеньев" << endl;

os << dt.size << endl;

return os;

}

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

{

is >> dt.size;

return is;

}

class not\_cl\_br\_line : public br\_line {//класс незамкнутая ломаная линия

private:

int size1;

int \*len\_lin1;

int \*angles1;

public:

not\_cl\_br\_line() {

size1 = 3;

len\_lin1 = new int[size1];

len\_lin1[0] = 1; len\_lin1[1] = 1; len\_lin1[2] = 1;

angles1 = new int[size1];

angles1[0] = 100; angles1[1] = 200;

}

not\_cl\_br\_line(const not\_cl\_br\_line &l) {

size1 = l.size1;

len\_lin1 = new int[size1];

for (int i = 0; i < size1; i++) {

len\_lin1[i] = l.len\_lin1[i];

}

angles1 = new int[size1 - 1];

for (int i = 0; i < (size1 - 1); i++) {

angles1[i] = l.angles1[i];

}

}

not\_cl\_br\_line(int lin, int \*mass1, int \*mass2) {

size1 = lin;

len\_lin1 = new int[size1];

for (int i = 0; i < size1; i++) {

len\_lin1[i] = mass1[i];

}

angles1 = new int[size1 - 1];

for (int i = 0; i < (size1 - 1); i++) {

angles1[i] = mass2[i];

}

}

~not\_cl\_br\_line() {

delete[] len\_lin1;

delete[] angles1;

};

int test\_func(int lin, int \*ang) //возвращает 1, если есть ошибка в данных

{

lin = lin - 1;

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

if ((i % 2) != 0) {//если i-нечетное

if (ang[i] > 180) {

cout << "Ошибка при записи "<< i << " градуса, переопределите массив"<< endl;

return 1;

}

}

else { //если угол четный

if (ang[i] < 180) {

cout << "Ошибка при записи " << i+1 << " градуса, переопределите массив" << endl;

return 1;

}

}

}

return 0;

}

int length\_of\_line()

{

int sum = 0;

for (int i = 0; i < size1; i++)

{

sum = sum + len\_lin1[i];

}

return sum;

}

friend ostream& operator<<(ostream &os, not\_cl\_br\_line& dt);

friend istream& operator>>(istream& is, not\_cl\_br\_line& dt);

friend not\_cl\_br\_line & operator++(not\_cl\_br\_line &pl, int);

friend not\_cl\_br\_line & operator+(not\_cl\_br\_line &left, not\_cl\_br\_line &right);

};

istream& operator>>(istream& is, not\_cl\_br\_line &dt) //перегрузка оператора ввода

{

is >> dt.size1;

return is;

}

ostream& operator<<(ostream &os, not\_cl\_br\_line& dt) //перегрузка оператора вывода

{

for (int i = 0; i < dt.size1; i++) {

os << dt.len\_lin1[i] << " ";

}

os << endl;

for (int i = 0; i < dt.size1-1; i++) {

os << dt.angles1[i] << " ";

}

return os;

}

not\_cl\_br\_line & operator++(not\_cl\_br\_line& pl, int) {//перегрузка унарного оператора, увеличивает линию

for (int i = 0; i < pl.size1; i++) {

pl.len\_lin1[i]++;

}

return pl;

}

not\_cl\_br\_line & operator+(not\_cl\_br\_line& left, not\_cl\_br\_line& right)//перегрузка бинарного прератора

{

not\_cl\_br\_line sum;

sum.size1 = left.size1 + right.size1;

return sum;

}

int main(void)

{

setlocale(LC\_ALL, "Russian");

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

int t=0;//переменная для проверки углов

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, чем звеньев)" << endl;

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

{

cin >> mas\_ang[i];

}

not\_cl\_br\_line A1 (x, mas\_len, mas\_ang);

\_rename:

if (t == 1) {

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

{

cin >> mas\_ang[i];

}

}

t=A1.test\_func(x, mas\_ang);

if (t == 1) { goto \_rename; }

not\_cl\_br\_line A2 ();

cout << "Размерности отрезков и углы параметризированной незамкнутой ломаной линии" << endl;

cout << A1 << endl;

A1++;

cout << "Размерность отрезков линии, увеличенные на один" << endl;

cout << A1 << endl;

cout <<"Вывод массива, после увеличения его размерности вдвое"<< endl;

A1 = A1 + A1;

cout << A1 <<endl;

delete[] mas\_len;

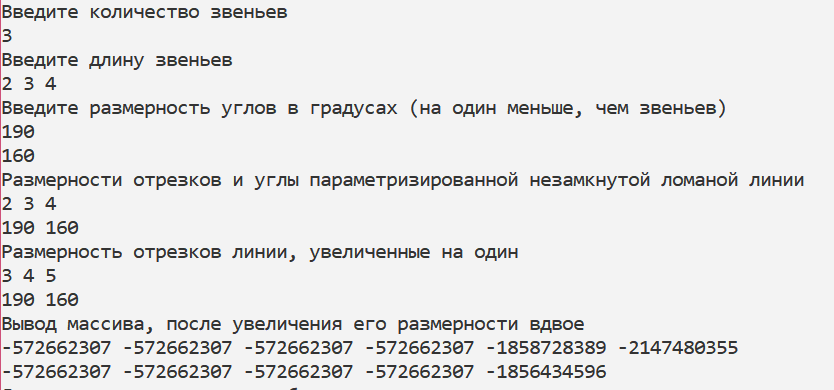
delete[] mas\_ang;

system("pause");

}

ПРОТОКОЛ ВЫПОЛНЕНИЯ ПРОГРАММЫ

Протокол выполнения программы представлен на рис.3.



*Рисунок 3 Протокол программы*