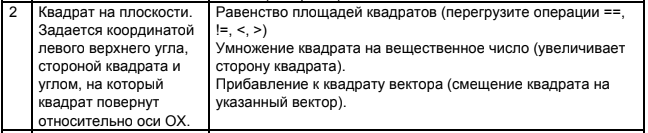
Лабараторная работа 3

Клишевич Вадим, M3105

Main.cpp

#include <bits/stdc++.h>  
#include "Square.h"  
#include "Array.h"  
  
using namespace std;  
  
int main() {  
 Square a;  
 Square b(1, 2, 3, 4);  
 Square c;  
  
 cout << "a "; a.print(); cout << "b "; b.print(); cout << "c "; c.print();  
 c = a \* 3.14;  
 cout << "a "; a.print(); cout << "b "; b.print(); cout << "c "; c.print();  
 c = a.operator\*(3.14);  
 cout << "a "; a.print(); cout << "b "; b.print(); cout << "c "; c.print();  
 c = a + make\_pair(1, 2);  
 cout << "a "; a.print(); cout << "b "; b.print(); cout << "c "; c.print();  
 c = a.operator+(make\_pair(2, 1));  
 cout << "a "; a.print(); cout << "b "; b.print(); cout << "c "; c.print();  
  
 cout << "a == b\t->\t" << (a == b) << " : " << a.operator==(b) << "\n";  
 cout << "a != b\t->\t" << (a != b) << " : " << a.operator!=(b) << "\n";  
 cout << "a < b\t->\t" << (a < b) << " : " << a.operator<(b) << "\n";  
 cout << "a > b\t->\t" << (a > b) << " : " << a.operator>(b) << "\n";  
  
  
 cout << "\n\n\n";  
  
  
 Array d;  
 Array e({1, 2, 3, 4, 5});  
 Array f;  
  
 cout << "d "; d.print(); cout << "e "; e.print(); cout << "f "; f.print();  
 f = d + e;  
 cout << "d "; d.print(); cout << "e "; e.print(); cout << "f "; f.print();  
  
  
 cout << "d == e\t->\t" << (d == e) << " : " << d.operator==(e) << "\n";  
 cout << "d != e\t->\t" << (d != e) << " : " << d.operator!=(e) << "\n";  
 cout << "d < e\t->\t" << (d < e) << " : " << d.operator<(e) << "\n";  
 cout << "d > e\t->\t" << (d > e) << " : " << d.operator>(e) << "\n";  
}

Square.h

//  
// Created by vadim on 3.03.20.  
//  
#include <bits/stdc++.h>  
using namespace std;  
  
#pragma once  
  
  
class Square {  
private:  
 pair<float, float> pos;  
 float len;  
 float alpha;  
  
public:  
 Square();  
 Square(const float &, const float &, const float &, const float &);  
 Square(const Square &);  
  
 bool operator==(const Square &) const;  
 bool operator!=(const Square &) const;  
 bool operator<(const Square &) const;  
 bool operator>(const Square &) const;  
 void print();  
  
 Square operator\*(const float &) const;  
 Square operator+(const pair<float, float> &) const;  
};

Square.cpp

//  
// Created by vadim on 3.03.20.  
//  
  
#include "Square.h"  
  
Square::Square() {  
 pos = {0, 0};  
 len = 3;  
 alpha = 0;  
}  
  
Square::Square(const float &posx, const float &posy, const float &len\_, const float &alpha\_) {  
 pos = {posx, posy};  
 len = len\_;  
 alpha = alpha\_;  
}  
  
Square::Square(const Square &x) {  
 pos = x.pos;  
 len = x.len;  
 alpha = x.alpha;  
}  
  
bool Square::operator==(const Square &x) const {  
 return len == x.len;  
}  
  
bool Square::operator!=(const Square &x) const {  
 return len != x.len;  
}  
  
bool Square::operator<(const Square &x) const {  
 return len < x.len;  
}  
  
bool Square::operator>(const Square &x) const {  
 return len > x.len;  
}  
  
void Square::print() {  
 cout << "position = {" << pos.first << ", " << pos.second << "};\tlen = " << len <<  
 ";\tangle between square side and OX = " << alpha \* 180. / M\_PI << " degrees\n";  
}  
  
Square Square::operator\*(const float &x) const {  
 Square a(\*this);  
 a.len \*= x;  
 return a;  
}  
  
Square Square::operator+(const pair<float, float> &x) const {  
 Square a(\*this);  
 a.pos.first += x.first;  
 a.pos.second += x.second;  
 return a;  
}

Array.h

//  
// Created by vadim on 3.03.20.  
//  
#include <bits/stdc++.h>  
using namespace std;  
  
#pragma once  
  
class Array {  
private:  
 vector<int> a;  
  
public:  
 Array();  
 Array(const int &);  
 Array(const vector<int> &);  
  
 bool operator==(const Array &) const;  
 bool operator!=(const Array &) const;  
 bool operator<(const Array &) const;  
 bool operator>(const Array &) const;  
 void print();  
  
 Array operator+(const Array &) const;  
};

Array.cpp

//  
// Created by vadim on 3.03.20.  
//  
  
#include "Array.h"  
  
Array::Array() {  
 a = {1, 2, 3};  
}  
  
Array::Array(const int &len) {  
 a.assign(len, 0);  
}  
  
Array::Array(const vector<int> &x) {  
 a = x;  
}  
  
bool Array::operator==(const Array &x) const {  
 return a.size() == x.a.size();  
}  
  
bool Array::operator!=(const Array &x) const {  
 return a.size() != x.a.size();  
}  
  
bool Array::operator<(const Array &x) const {  
 return a.size() < x.a.size();  
}  
  
bool Array::operator>(const Array &x) const {  
 return a.size() > x.a.size();  
}  
  
void Array::print() {  
 cout << "Array: len = " << a.size() << "\n";  
 for (auto &x : a) {  
 cout << x << " ";  
 }  
 cout << "\n";  
}  
  
Array Array::operator+(const Array &x) const {  
 Array temp(\*this);  
 for (auto &y : x.a) {  
 temp.a.push\_back(y);  
 }  
 return temp;  
}