Міністерство освіти і науки України

Національний технічний університет України

“Київський політехнічний інститут”

Кафедра АСОІУ

**ЗВІТ**

про виконання комп’ютерного практикуму № 5

з дисципліни

“ООП”

Тема: «Исключительные ситуации в С++»

|  |  |  |
| --- | --- | --- |
| Прийняв: |  | Виконав: |
| Головченко Максим Миколайович |  | студент 2-го курсу  гр. ІП-52 ФІОТ  Набоков Едуард Максимович |

Київ – 2016

ЗМІСТ

[1 Мета роботи 3](#_Toc466999300)

[2 Постановка задачі (варіант 10, рівень В) 4](#_Toc466999301)

[3 Аналітичні викладки 5](#_Toc466999302)

[4 Діаграма класів 6](#_Toc466999303)

[5 Код програми 7](#_Toc466999304)

[6 Приклади 14](#_Toc466999305)

[Висновок 15](#_Toc466999306)

# Мета роботи

Цель работы – изучить особенности генерации и обработки исключительных ситуаций в C++.

# Постановка задачі (варіант 10, рівень В)

Спроектировать класс «Point», который содержит точку на плоскости и методы доступа к ней. Спроектировать класс «Points», который содержит: множество точек на плоскости – *A* (объектов класса Point), количество точек – *n.* Для него определить: операцию добавления новой точки «<<» и операцию удаления точки «>>». Для него определить: операцию объединения «+», ее же в сокращенной форме, которая находит объединение точек без дубликатов; операцию пересечения «\*», ее же в сокращенной форме, которая находит пересечение точек без дубликатов; операцию разности «-», ее же в сокращенной форме, которая находит разность точек без дубликатов. При необходимости разрешается определять другие операции (например «=») и методы (например, getter, setter и прочее). Продемонстрировать каждую операцию.

# Аналітичні викладки

# Діаграма класів

На рисунке 4.1 показана діаграмма класів

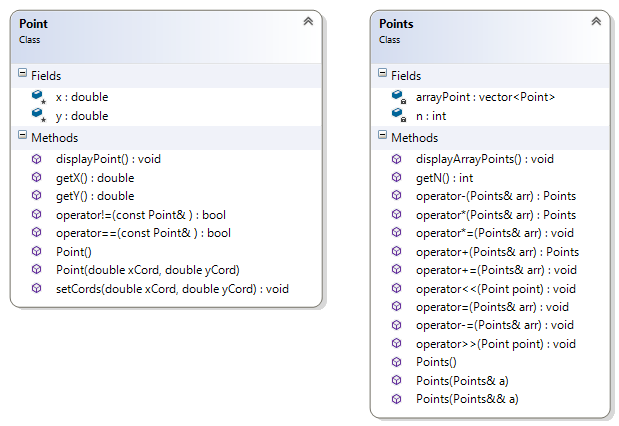


Рисунок 4.1

# Код програми

“main.cpp”

#include "helper.h"

int main() {

int answer;

display\_info();

Points arr1 = test\_set1();

Points arr2 = test\_set2();

cout << "Set 1:" << endl;

arr1.displayArrayPoints();

cout << endl;

cout << "Set 2:" << endl;

arr2.displayArrayPoints();

cout << endl;

answer = display\_menu();

while (true) {

if (answer == 6)

break;

work\_logic(answer, arr1, arr2);

answer = display\_menu();

}

}

“helper.h”

#ifndef HELPER\_H

#define HELPER\_H

#include "Points.h"

void display\_info();

int display\_menu();

double input\_number(string s);

void enter\_figure();

void work\_logic(int answer, Points&, Points&);

void add\_point(Points&);

void remove\_point(Point&);

Points test\_set1();

Points test\_set2();

void make\_action(Points&, Points&);

#endif

“helper.cpp”

#include "helper.h"

void display\_info() {

cout << setfill('"') << setw(10) << '"' << endl;

cout << "Edward Nabokov" << endl;

cout << "V - 10" << endl;

cout << "Level: B" << endl;

cout << setfill('"') << setw(10) << '"' << endl;

}

int display\_menu() {

int answer;

while (true) {

cout << setfill('-') << setw(40) << '-' << endl;

cout << "1 -> Add point to 1 set" << endl;

cout << "2 -> Add point to 2 set" << endl;

cout << "3 -> Remove point from 1 set" << endl;

cout << "4 -> Remove point from 2 set" << endl;

cout << "5 -> Actions" << endl;

cout << "6 -> Exit" << endl;

cout << setfill('-') << setw(40) << '-' << endl << endl;

cout << "Enter answer: ";

cin >> answer;

if (answer > 0 && answer < 7)

return answer;

cout << endl << "---Incorrect input!" << endl;

}

}

double input\_number(string s) {

double input;

cin >> input;

while (input <= 0) {

cout << "Incorrect input" << endl;

cin >> input;

}

return input;

}

void enter\_figure() {

}

Points test\_set1() {

Points arr;

Point test1(-1, -1), test2(3.25, 6.44), test3(-4, 5), test4(-1, -4), test5(-100, 100);

arr << test1;

arr << test2;

arr << test3;

arr << test4;

arr << test5;

return arr;

}

Points test\_set2() {

Points arr;

Point test1(-1, -1), test2(4, 13), test3(5, 23), test4(0, 0), test5(-100, 100);

arr << test1;

arr << test2;

arr << test3;

arr << test4;

arr << test5;

return arr;

}

void add\_point(Points &arr) {

double x, y;

cout << "Input X: ";

cin >> x;

cout << "Input Y: ";

cin >> y;

arr << Point(x, y);

}

void remove\_point(Points &arr) {

double x, y;

cout << "Input X: ";

cin >> x;

cout << "Input Y: ";

cin >> y;

arr >> Point(x, y);

}

void make\_action(Points &arr1, Points &arr2) {

cout << endl;

cout << "1 -> Union (+ operator)" << endl;

cout << "2 -> Intersection (\* operator)" << endl;

cout << "3 -> Difference (- operator)" << endl;

int answer;

Points temp;

cin >> answer;

switch (answer) {

case 1: {

temp = arr1 + arr2;

cout << endl;

cout << "RESULT:" << endl;

temp.displayArrayPoints();

cout << endl;

break;

}

case 2: {

temp = arr1 \* arr2;

cout << endl;

cout << "RESULT:" << endl;

temp.displayArrayPoints();

cout << endl;

break;

}

case 3: {

temp = arr1 - arr2;

cout << endl;

cout << "RESULT:" << endl;

temp.displayArrayPoints();

cout << endl;

break;

}

default:

break;

}

}

void work\_logic(int answer, Points &arr1, Points &arr2) {

switch (answer) {

case 1: {

add\_point(arr1);

break;

}

case 2: {

add\_point(arr2);

break;

}

case 3: {

remove\_point(arr1);

break;

}

case 4: {

remove\_point(arr2);

break;

}

case 5: {

make\_action(arr1, arr2);

break;

}

default:

break;

}

cout << "Set 1:" << endl;

arr1.displayArrayPoints();

cout << endl;

cout << "Set 2:" << endl;

arr2.displayArrayPoints();

cout << endl;

}

“Point.h”

#ifndef POINT\_H

#define POINT\_H

#include <iostream>

#include <iomanip>

using namespace std;

class Point {

protected:

double x, y;

public:

Point();

Point(double xCord, double yCord);

void setCords(double xCord, double yCord);

double getX() const;

double getY() const;

void displayPoint();

bool operator==(const Point&);

bool operator!=(const Point&);

friend bool operator==(Point&, Point&);

friend bool operator!=(Point&, Point&);

};

#endif

“Point.cpp”

#include "Point.h"

Point::Point() {

x = 0;

y = 0;

}

Point::Point(double xCord = 0, double yCord = 0) : x(xCord), y(yCord) {}

void Point::setCords(double xCord, double yCord) {

x = xCord;

y = yCord;

}

double Point::getX() const {

return x;

}

double Point::getY() const {

return y;

}

void Point::displayPoint() {

cout << "( " << x << " ; " << y << " )" << endl;

}

bool Point::operator==(const Point &a) {

if ((x == a.getX()) && (y == a.getY())) return true;

return false;

}

bool Point::operator!=(const Point &a) {

if ((x != a.getX()) || (y != a.getY())) return true;

return false;

}

bool operator==(Point &a, Point &b) {

if ((a.getX() == b.getX()) && (a.getY() == b.getY())) return true;

return false;

}

bool operator!=(Point &a, Point &b) {

if ((a.getX() != b.getX()) || (a.getY() != b.getY())) return true;

return false;

}

“Points.h”

#ifndef POINTS\_H

#define POINTS\_H

#include "Point.h"

#include <vector>

class Points{

int n;

vector<Point> arrayPoint;

public:

Points();

Points(Points &a);

Points(Points &&a);

void displayArrayPoints();

int getN();

void operator<<(Point point);

void operator>>(Point point);

Points operator+(Points &arr);

void operator+=(Points &arr);

Points operator\*(Points &arr);

void operator\*=(Points &arr);

Points operator-(Points &arr);

void operator-=(Points &arr);

void operator=(Points &arr);

};

#endif

“Points.cpp”

#include "Points.h"

Points::Points()

{

arrayPoint.clear();

n = 0;

}

Points::Points(Points &a)

{

n = a.n;

arrayPoint = a.arrayPoint;

}

Points::Points(Points &&a): n(move(a.n)), arrayPoint(move(a.arrayPoint))

{}

void Points::operator<<(Point point)

{

vector<Point>::iterator it;

it = find(arrayPoint.begin(), arrayPoint.end(), point);

if (it == arrayPoint.end()) {

arrayPoint.push\_back(point);

n++;

}

}

void Points::operator>>(Point point)

{

vector<Point>::iterator position;

position = find(arrayPoint.begin(), arrayPoint.end(), point);

if (position != arrayPoint.end()) {

arrayPoint.erase(position);

n--;

}

}

Points Points::operator+(Points &arr)

{

Points temp = arr;

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

temp << arrayPoint[i];

return temp;

}

void Points::operator+=(Points &arr)

{

\*this = \*this + arr;

}

Points Points::operator\*(Points &arr)

{

Points temp;

vector<Point>::iterator position;

for (int i = 0; i < arr.n; i++) {

position = find(arrayPoint.begin(), arrayPoint.end(), arr.arrayPoint[i]);

if (position != arrayPoint.end())

temp << arr.arrayPoint[i];

}

return temp;

}

void Points::operator\*=(Points &arr) {

\*this = \*this \* arr;

}

Points Points::operator-(Points &arr)

{

Points temp;

vector<Point>::iterator position;

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

position = find(arr.arrayPoint.begin(), arr.arrayPoint.end(), arrayPoint[i]);

if (position == arr.arrayPoint.end())

temp << arrayPoint[i];

}

return temp;

}

void Points::operator-=(Points &arr) {

\*this = \*this - arr;

}

void Points::operator=(Points &arr)

{

n = arr.n;

arrayPoint = move(arr.arrayPoint);

}

int Points::getN()

{

return n;

}

void Points::displayArrayPoints()

{

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

cout << "Point " << i + 1 << ": ";

arrayPoint[i].displayPoint();

}

# Приклади

На зображеннях 6.1 та 6.2 показани приклади роботи программи

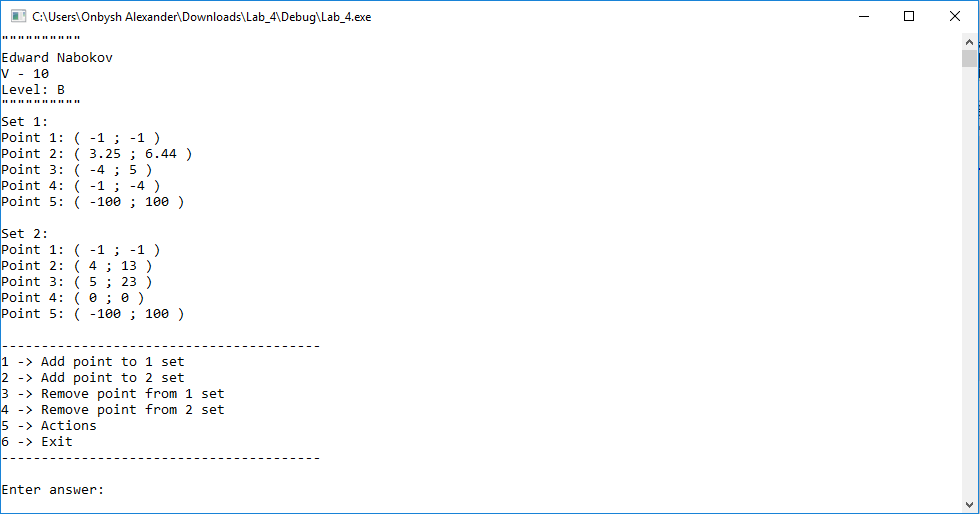


Рисунок 6.1

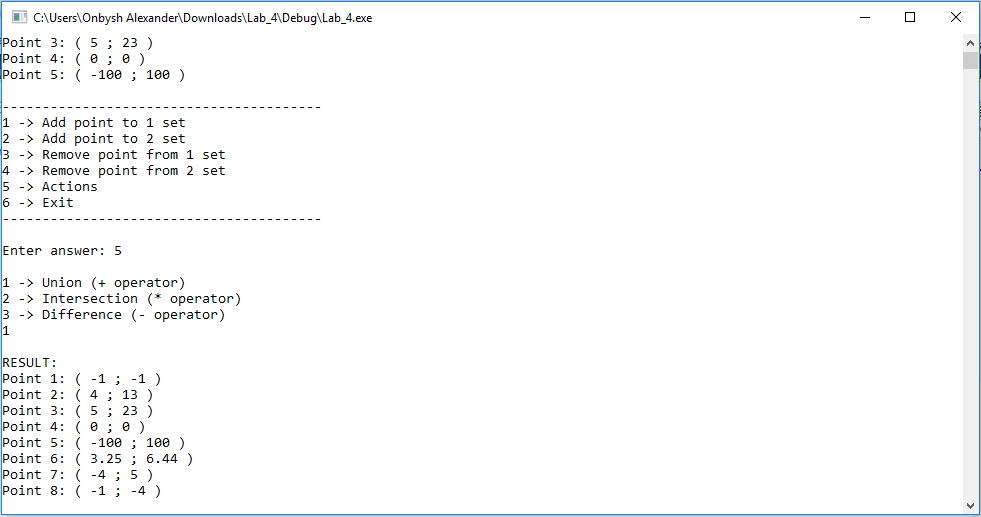


Рисунок 6.2

# Висновок

У цій лабораторній роботі були вивчені основи концепції використання виключаючих функцій. Розібрані особливості використання цих концепцій у мові С++. Була створена программа, яка реалізує клас точки, множину точок та ієрархію виключаючих класів(функцій). У класі множини точок були створені методи для виконання операцій об’єднання, перетину, віднімання множин, які покриті try{..}catch(){…}. Для цих методів були перевантажені оператори для зручного використання методів. Були оброблені виключні ситуаціі коли користувач міг ввести некоректні дані у програмі.