**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра вычислительной техники**

отчет

**по лабораторной работе №1**

**по дисциплине «Алгоритмы и структуры данных»**

Тема: «Поддержка обработки исключительных ситуаций»

**Вариант №9**

|  |  |  |
| --- | --- | --- |
| Студенты гр. 6307 |  | Лазарев С.О.  Медведев Е.Р. |
| Преподаватель |  | Колинько П.Г. |

Санкт-Петербург

2018

СОДЕРЖАНИЕ

[ЗАДАНИЕ 3](#_Toc506283915)

[ДОБАВЛЕННЫЕ КЛАССЫ 4](#_Toc506283916)

[ПЕРЕОПРЕДЕЛЁННЫЕ ФУНКЦИИ-ЧЛЕНЫ 6](#_Toc506283917)

[КОНТРОЛЬНЫЙ ПРИМЕР РАБОТЫ ПРОГРАММЫ 8](#_Toc506283918)

[ВЫВОДЫ 11](#_Toc506283919)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 12](#_Toc506283920)

[ПРИЛОЖЕНИЕ: ТЕКСТ ПРОГРАММЫ 13](#_Toc506283921)

# 

# Цель

Получить практические навыки по работе с механизмом контроля исключительных ситуаций.

# ЗАДАНИЕ

Дополнить программу работы с библиотеками фигур механизмом контроля исключительных ситуаций.

Возможно выявление следующих ошибок:

* Непопадание точки на экран;
* Некорректные параметры при формировании фигуры;

**Классы для фиксаций особых ситуаций**

1. Попадание точки за пределы экрана

class offScreenPointError : public std::exception

{

private:

std::string m\_error;

public:

offScreenPointError(const std::string& a\_error) : m\_error(a\_error) {}

const char\* what() const noexcept { return m\_error.c\_str(); }

};

2. Некорректные параметры при инициализации фигуры

class errorIncorrectShapeInitializationParameters : public std::exception

{

private:

std::string m\_error;

public:

errorIncorrectShapeInitializationParameters(const std::string& a\_error) : m\_error(a\_error) {}

const char\* what() const noexcept { return m\_error.c\_str(); }

};

# Расположение операторов throw и блоков контроля

Оператор throw в функции put\_point (если точка не попадает в границы экрана, то происходит «выброс исключения»)

void put\_point(int a, int b)

{

if (on\_screen(a, b))

{

screen[b][a] = black;

}

else

{

throw offScreenPointError("trying to put a point off the screen: ");

}

}

Блоки контроля и защищённые части кода в функциях draw для различных фигур (если в защищённой части кода произошёл «выброс исключения», то происходит переход в блок catch (обработчик ошибочной ситуации))

void rectangle::draw()

{

try

{

put\_line(nwest(), ne); put\_line(ne, seast());

put\_line(seast(), sw); put\_line(sw, nwest());

}

catch (const std::exception& e)

{

std::cout << e.what() << " rectangle::draw\n";

std::cin.get();

}

}

}  
  
void parallelogram::draw() //Рисование

{

try

{

put\_line(ne, seast());

put\_line(sw, seast());

put\_line(sw, nwest());

put\_line(ne, nwest());

}

catch (const offScreenPointError& e)

{

std::cout << e.what() << " parallelogram::draw\n";

std::cin.get();

}

}

void myshape::draw()

{

try

{

rectangle::draw(); //Контур лица (глаза и нос рисуются сами!)

int a = (swest().x + neast().x) / 2;

int b = (swest().y + neast().y) / 2;

put\_point(point(a, b)); // Нос – существует только на рисунке!

}

catch (const offScreenPointError& e)

{

std::cout << e.what() << " myshape::draw\n";

std::cin.get();

}

}

void line::draw()

{

try

{

put\_line(w, e);

}

catch (const offScreenPointError& e)

{

std::cout << e.what() << " line::draw\n";

std::cin.get();

}

}

void h\_circle::draw() //Алгоритм Брезенхэма для окружностей

{ //(выдаются два сектора, указываемые значением reflected)

int x0 = (sw.x + ne.x) / 2, y0 = reflected ? sw.y : ne.y;

int radius = (ne.x - sw.x) / 2;

int x = 0, y = radius, delta = 2 - 2 \* radius, error = 0;

try

{

while (y >= 0) //Цикл рисования

{

if (reflected)

{

put\_point(x0 + x, y0 + y \* 0.7); put\_point(x0 - x, y0 + y \* 0.7);

}

else

{

put\_point(x0 + x, y0 - y \* 0.7); put\_point(x0 - x, y0 - y \* 0.7);

}

error = 2 \* (delta + y) - 1;

if (delta < 0 && error <= 0)

{

++x;

delta += 2 \* x + 1;

continue;

}

error = 2 \* (delta - x) - 1;

if (delta > 0 && error > 0)

{

--y; delta += 1 - 2 \* y; continue;

}

++x; delta += 2 \* (x - y); --y;

}

}

catch (const offScreenPointError& e)

{

std::cout << e.what() << " line::draw\n";

std::cin.get();

}

}

Оператор throw в защищённом блоке кода в конструкторах класса различных фигур

rectangle(point a, point b) : sw(a), ne(b)

{

try

{

if (sw.x > ne.x || sw.y > ne.y)

{

throw errorIncorrectShapeInitializationParameters("Incorrect Initialization rectangle");

}

}

catch (const errorIncorrectShapeInitializationParameters& e)

{

std::cout << e.what() << '\n';

std::cin.get();

}

}

parallelogram(point a, point b, int base) : sw(a), ne(b), sw\_se(base)

{

try

{

if (base < 0)

{

throw errorIncorrectShapeInitializationParameters("Incorrect Initialization parallelogram");

}

}

catch (const errorIncorrectShapeInitializationParameters& e)

{

std::cout << e.what() << '\n';

std::cin.get();

}

}

line(point a, int L) : w(point(a.x + L - 1, a.y)), e(a)

{

try

{

if (L < 0)

{

throw errorIncorrectShapeInitializationParameters("Incorrect Initialization line");

}

}

catch (const errorIncorrectShapeInitializationParameters& e)

{

std::cout << e.what() << '\n';

std::cin.get();

}

};

# Пример работы программы



Рис. 1. Сообщение о непопадание точки на экран при отрисовке объекта line

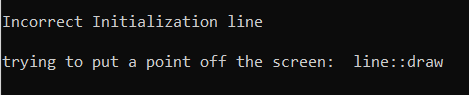


Рис. 2. Сообщение о некорректной конфигурации объекта line

# ВЫВОДЫ

В результате работы в программу работы с библиотекой фигур был добавлен механизм работы с исключительными ситуациями. Данный механизм отслеживает ошибки непопадания точки на экран и ошибки некорректной конфигурации фигуры, и выводит на экран сообщение с указанием ошибки и объекта, в котором эта ошибка произошла.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Алгоритмы и структуры данных. – Методические указания к лабораторным работам, практическим занятиям и курсовому проектированию, часть 2, глава 1 «Работа с иерархией объектов: наследование и полиморфизм».

# ПРИЛОЖЕНИЕ

**offScreenPointError.h**

#pragma once

#include <exception>

#include <string>

class offScreenPointError : public std::exception

{

private:

std::string m\_error;

public:

offScreenPointError(const std::string& a\_error) : m\_error(a\_error) {}

const char\* what() const noexcept { return m\_error.c\_str(); }

};

**errorIncorrectShapeInitializationParameters.h**

#pragma once

#include <exception>

#include <string>

class errorIncorrectShapeInitializationParameters : public std::exception

{

private:

std::string m\_error;

public:

errorIncorrectShapeInitializationParameters(const std::string& a\_error) : m\_error(a\_error) {}

const char\* what() const noexcept { return m\_error.c\_str(); }

};