**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра Вычислительной техники**

отчет

**по лабораторной работе №2**

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

Тема: «Поддержка обработки исключительных ситуаций»

|  |  |  |
| --- | --- | --- |
| Студенты гр. 1305 |  | Данилов А.С.  Серкин Д.А. |
| Преподаватель |  | Колинько П.Г. |

Санкт-Петербург

2023

**Цель работы**

Изучение способов обработки исключительных ситуаций.

**Задание**

Переработать программу работы с библиотекой фигур, дополнив её механизмом контроля исключительных ситуаций.

Реализовать генерацию и перехват не менее двух типов ошибок разного уровня сложности.

**Набор и вид классов для фиксации особых ситуаций**

1. Попадание фигуры за пределы экрана

struct OutError: runtime\_error // Выход за пределы экрана

{

OutError(const std::string& s): runtime\_error(s.c\_str( )) { }

};

1. Повторный поворот

struct RotationError: runtime\_error // Попытка повторного поворота

{

RotationError(const std::string& s): runtime\_error(s.c\_str( )) { }

};

Классы ошибок наследуют от класса runtime\_error (который наследует от класса exception) и имеют конструктор, принимающий строку.

**Место расположения операторов throw и блоков контроля**

Операторы throw расположены в функциях on\_screen и check\_point:

void on\_screen(int a, int b) // проверка попадания точки на экран

{

if (not(0 <= a && a < XMAX && 0 <= b && b < YMAX))

throw OutError("Выход за границы:");

}

void check\_rotation(int a, int b) // проверка двойного поворота

{

if (a + b == 2)

throw RotationError("Повторный поворот:");

}

Блоки контроля:

1. Для линии

void draw( )

{

try

{

on\_screen(w.x,w.y);

on\_screen(e.x,e.y);

put\_line(w, e);

}

catch (OutError &ex)

{

std::cout << ex.what() << " line::draw()\nТеперь фигура на середине экрана.";

std::cin.get();

this->move(XMAX / 2,YMAX / 2);

put\_line(w, e);

}

}

1. Для прямоугольника

try

{

on\_screen(sw.x,sw.y);

on\_screen(ne.x,ne.y);

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

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

}

catch (OutError &ex)

{

std::cout << ex.what() << " rectangle::draw()\nТеперь фигура на середине экрана.";

std::cin.get();

this->move(XMAX / 2, YMAX / 2);

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

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

}

1. Для параллелограмма

void parallelogram\_x :: draw( )

{

try

{

check\_rotation(r\_rotated, l\_rotated);

}

catch (RotationError &ex)

{

l\_rotated = 0;

std::cout << ex.what() << " draw()\nФигура повёрнута вправо.";

std::cin.get();

}

if(r\_rotated) // поворот вправо

{

flip\_horisontally();

put\_line(rot\_corner1(), rot\_corner2());

put\_line(rot\_corner3(), rot\_corner2());

put\_line(rot\_corner4(), rot\_corner3());

put\_line(rot\_corner4(), rot\_corner1());

}

else if(l\_rotated) // поворот влево

{

put\_line(rot\_corner1(), rot\_corner2());

put\_line(rot\_corner3(), rot\_corner2());

put\_line(rot\_corner4(), rot\_corner3());

put\_line(rot\_corner4(), rot\_corner1());

}

else

{

put\_line(corner1(), corner2()); // параллелограмм

put\_line(corner3(), corner2());

put\_line(corner4(), corner3());

put\_line(corner4(), corner1());

}

put\_line(west(), east()); // крест

put\_line(south(), north());

}

**Результаты тестирования**

1. Выход фигуры за пределы экрана



****

1. Повторный поворот



**Выводы**

В процессе выполнения данного задания мы изучили механизм исключительных ситуаций. Подключив программу к системному механизму перехвата исключений, мы использовали находящийся в нём механизм для сообщений.

**Список использованных источников**

1. П.Г. Колинько – «Пользовательские контейнеры» учебно-метод. пособие, 2023 г.

**Приложение**

В файле shape.h.