**МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ КЫРГЫЗСКОЙ РЕСПУБЛИКИ**

**КЫРГЫЗСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ ИМ. И.РАЗЗАКОВА**

**ФАКУЛЬТЕТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ**

**КАФЕДРА ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ КОМПЬЮТЕРНЫХ СИСТЕМ**

**ОТЧЕТ**

**ПО ДИСЦИПЛИНЕ: ООП**

**ПО ТЕМЕ: Исключения**

**Лабораторная работа №8**

**ВЫПОЛНИЛ:** студент гр. ПИ(б)-3-18

Гадайбаев М.А.

**ПРОВЕРИЛА:** Каткова С.Н.

**Бишкек 2020**

**ВОПРОСЫ К ЗАЩИТЕ ЛАБОРАТОРНОЙ РАБОТЫ № 8**

1. Исключение в большинстве случаев возникает из-за:

* Программиста, написавшего исходный код приложения;
* Создателя класса, написавшего его метод;
* **Ошибка выполнения;**
* Сбоя в операционной системе.

1. При работе с механизмом исключений в С++ используются следующие ключевые слова: **catch**, **throw**, **try**.
2. Исключения передаются:

* Из блока-ловушки в блок повторных попыток;
* Из выражения, создавшего исключительную ситуацию, в блок повторных попыток;
* Из точки, где возникла ошибка, в блок-ловушку.
* **Из выражения, в котором возникла ошибка, в блок-ловушку.**

1. Для следующих ошибок обычно генерируются исключения:

* **Чрезмерное количество данных грозит переполнить массив;**
* Пользователь нажал Ctr+C для закрытия программы;
* Скачок напряжения в сети привел к перезагрузке системы;
* «New» не может зарезервировать необходимый объем памяти.

1. Истинно ли утверждение о том, что программа может продолжить свое выполнение после возникновения исключительной ситуации?

В результате возникновения исключительной ситуации программа не может корректно продолжить своё выполнение.

1. Поясните механизм исключений.

Механизм перехвата исключений позволяет генерировать исключения в том месте, в котором оно возникает.

**Задание №1.**

**Постановка задачи:**

Добавьте класс исключений к программе ARROVER1, чтобы индексы, выходящие за пределы массива, вызывали генерацию исключения. Блок-ловушка может выводить пользователю сообщение об ошибке.

**Описание входных данных:**

Limit – размер массива, типа cons int = 5.

safeArray – класс заполняющий массив, поля: **arr[Limit]**, методы: **putel()** – установка значения элемента массив; **getel()** – получение значения элемента массива; Arrover1 – класс исключений для safeArray.

sal – объект класса safeArray.

val – элемент массива.

i – индекс массива.

**Функция main() Рис.1.**

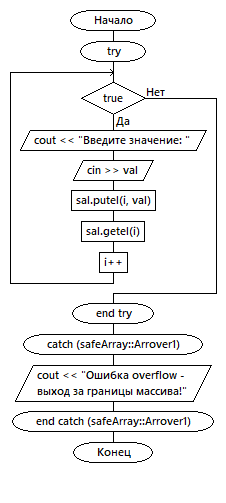


Рис.1.

**Класс safeArray Рис.2.**



Рис.2.

**Метод putel() Рис.3. Метод getel() Рис.4.**

Рис.3. Рис.4.

**Код:**

#include <iostream>

#include <process.h> //для функции exit

using namespace std;

const int Limit = 5; //размер массива

class safeArray {

private:

int arr[Limit];

public:

//класс ислючений для safeArray

class Arrover1 {};

//установка значения элемента массива

void putel(int i, int elValue) {

if (i < 0 || i >= Limit) {

throw Arrover1();

}

arr[i] = elValue;

}

//получение значения элемента массива

int getel(int i) const {

if (i < 0 || i >= Limit) {

throw Arrover1();

}

return arr[i];

}

};

int main() {

setlocale(LC\_ALL, "rus");

safeArray sal;

int val, i = 1;

try {

//задаём и показываем значения элементов

while (true) {

cout << "Введите значение: ";

cin >> val;

sal.putel(i, val);

sal.getel(i);

i++;

}

}

catch (safeArray::Arrover1) { //обработчик

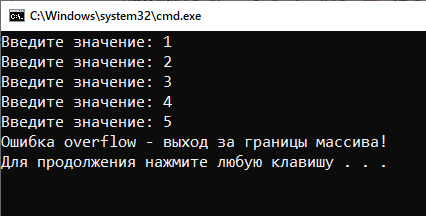
cout << "Ошибка overflow - выход за границы массива!" << endl;

}

return 0;

}

**Результат:**

****

**Задание №2.**

**Постановка задачи:**

Задание 3 из лабораторной работы №4 выполнить с использованием механизма исключений.

**Описание входных данных:**

Area – класс вычисления площади прямоугольника и квадрата, поля: **width –** ширина; **length** – длина, методы: **Area()** – перегруженный аргументами конструктор; **rect\_area()** – перегруженный метод, вычисляющая площадь; OverflowINT – класс исключений для Area, вызывается если число вышло за границы int.

MinusValue – класс исключений для Area, вызывается если были введены отрицательные значения.

area1, area2 – объекты класса safeArray.

**Функция main Рис.2.**

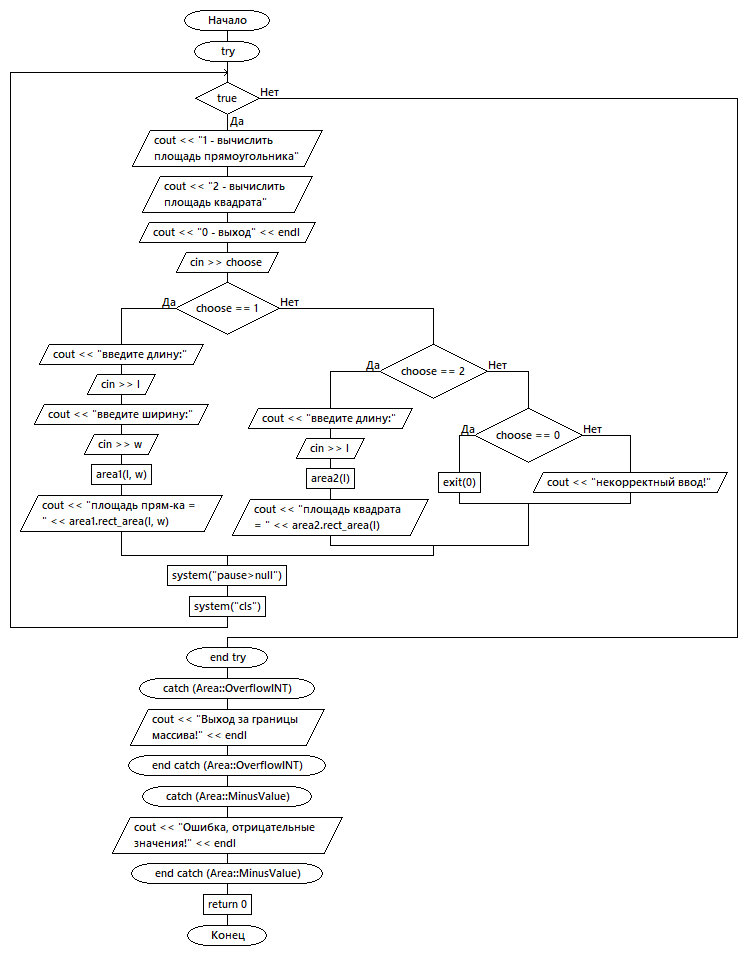
****

Рис.2.

**Класс Area Рис.2.**



Рис.2.

**Метод rect\_area(a, b) Рис.3. Метод rect\_area(a) Рис.4.**

Рис.3. Рис.4

**Код:**

#include <iostream>

using namespace std;

// Класс вычисления площади прямоугольника и квадрата

class Area {

private:

// поля класса

int width; // ширина

int length; // длина

public:

// класс исключений для Area

class OverflowINT {};

class MinusValue {};

// перегруженный аргументами конструктор

Area(int a, int b) {

width = a;

length = b;

}

Area(int a) {

length = a;

}

// перегруженный метод вычисляющая площадь

int rect\_area(int a, int b) {

long int ab = a \* b;

if (ab >= INT16\_MAX) {

throw OverflowINT();

}

else if (a < 0 || b < 0) {

throw MinusValue();

}

return a \* b;

}

int rect\_area(int a) {

long int aa = a \* a;

if (aa >= INT16\_MAX) {

throw OverflowINT();

}

else if (a < 0) {

throw MinusValue();

}

return a \* a;

}

};

int main() {

setlocale(LC\_ALL, "rus");

int choose, w, l;

try {

while (true) {

cout << "1 - вычислить площадь прямоугольника" << endl;

cout << "2 - вычислить площадь квадрата" << endl;

cout << "0 - выход" << endl;

cout << "ответ пользователя: ";

cin >> choose;

if (choose == 1) {

cout << "введите длину:";

cin >> l;

cout << "введите ширину:";

cin >> w;

Area area1(l, w);

cout << "площадь прямоугольника = " << area1.rect\_area(l, w) << endl;

}

else if (choose == 2) {

cout << "введите длину:";

cin >> l;

Area area2(l);

cout << "площадь квадрата = " << area2.rect\_area(l) << endl;

}

else if (choose == 0) {

exit(0);

}

else {

cout << "\nнекорректный ввод. используйте цифры 1 - 2 для управления и 0 для выхода\n";

}

system("pause>null");

system("cls");

}

}

catch (Area::OverflowINT) { //обработчик

cout << "Ошибка overflow - выход за границы массива!" << endl;

}

catch (Area::MinusValue) { //обработчик

cout << "Ошибка, отрицательные значения!" << endl;

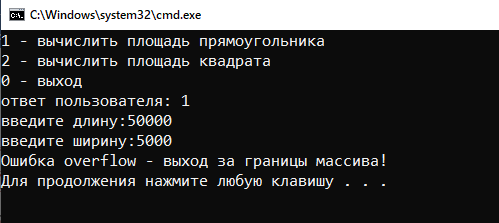
}

return 0;

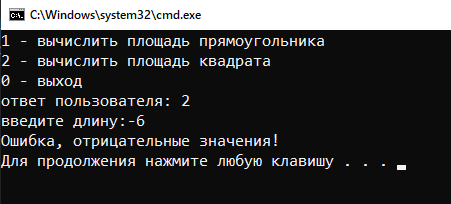
}

**Результат:**

**Исключение OverflowINT**



**Исключение MinusValue**

****