**МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ**

**КЫРГЫЗСКОЙ РЕСПУБЛИКИ**

**КЫРГЫЗСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ им. И.Раззакова**

**ФАКУЛЬТЕТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ**

Кафедра: **Программное обеспечение компьютерных систем**

Курс «Объектно-ориентированное программирование»

**ОТЧЕТ**

**Лабораторная работа №9**

Выполнил: студент группы ПИ-3-21

Копжашаров Азамат

Проверил: Мусабаев Э.Б.

**Бишкек 2024**

# **Задание №1**

**Постановка задачи:**

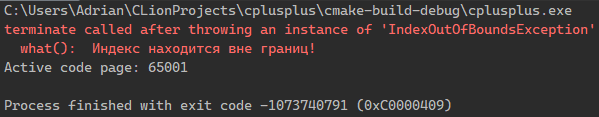
Добавьте класс исключений к программе ARROVER1, чтобы индексы, выходящие за пределы массива, вызывали генерацию исключения. Блок-ловушка может выводить пользователю сообщение об ошибке.

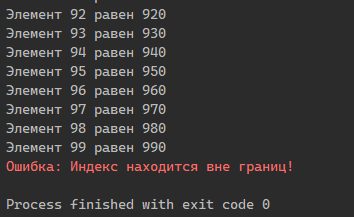
**Исходные данные:**

**Исходный код программы на C++:**

// 9.1  
// arrover1.cpp  
// демонстрация создания безопасного массива, проверяющего  
// свои индексы при использовании  
// используются отдельные функции для установки и получения значения  
#include <iostream>  
#include <stdexcept>  
using namespace std;  
#include <process.h> // для функции exit  
const int LIMIT = 100; // размер массива  
///////////////////////////////////////////////////////////////  
  
class IndexOutOfBoundsException : public exception { // Создание пользовательского класса исключения  
public:  
 const char\* what() const noexcept override {  
 return "Индекс находится вне границ!";  
 }  
};  
  
class safearray  
{  
private:  
 int arr [ LIMIT ];  
public:  
 // установка значения элемента массива  
 void putel(int n, int elvalue)  
 {  
 if ( n < 0 || n >= LIMIT ){  
 throw IndexOutOfBoundsException();  
 }  
 arr [ n ] = elvalue;  
 }  
 // получение значения элемента массива  
 int getel( int n ) const  
 {  
 if ( n < 0 || n >= LIMIT ) {  
 throw IndexOutOfBoundsException();  
 }  
 return arr [ n ];  
 }  
};  
///////////////////////////////////////////////////////////////  
int main( )  
{  
 system("chcp 65001");  
 safearray sa1;  
  
 // задаем значения элементов  
 for(int j = 0; j < LIMIT; j++)  
 sa1.putel(j, j \* 10);  
  
 // показываем элементы  
 for (int j = 0; j < LIMIT; j++) {  
 try {  
 int temp = sa1.getel(j);  
 cout << "Элемент " << j << " равен " << temp << endl;  
 } catch (const IndexOutOfBoundsException& e) {  
 cerr << "Ошибка: " << e.what() << endl; // Вывод сообщения об ошибке при возникновении исключения  
 }  
 }  
  
 return 0;  
}

**Тесты:**



****

# **Задание №2**

**Постановка задачи:**

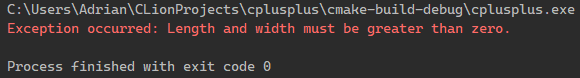
Задание 3 из лабораторной работы №4 выполнить с использованием механизма исключений.

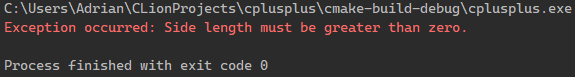
**Исходные данные:**

**Исходный код программы на C++:**

// 9.2  
#include <iostream>  
#include <stdexcept>  
using namespace std;  
  
class Rectangle {  
private:  
 double length;  
 double width;  
  
public:  
 Rectangle(double l, double w) {  
 if (l <= 0 || w <= 0) {  
 throw invalid\_argument("Length and width must be greater than zero.");  
 }  
 length = l;  
 width = w;  
 }  
  
 Rectangle(double size) {  
 if (size <= 0) {  
 throw invalid\_argument("Side length must be greater than zero.");  
 }  
 length = width = size;  
 }  
  
 double rect\_area() {  
 return length \* width;  
 }  
  
 double rect\_area(double side) {  
 if (side <= 0) {  
 throw invalid\_argument("Side length must be greater than zero.");  
 }  
 return side \* side;  
 }  
};  
  
int main() {  
 try {  
 Rectangle rectangle(4.0, 3.0);  
 Rectangle square(5.0);  
   
 cout << "Area of rectangle: " << rectangle.rect\_area() << endl;  
 cout << "Area of square: " << square.rect\_area(5.0) << endl;  
 } catch (const invalid\_argument& e) {  
 cerr << "Exception occurred: " << e.what() << endl;  
 }  
 return 0;  
}

**Тесты:**





# **Ответы на вопросы:**

1. Исключение в большинстве случаев возникает из-за:

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

Ошибка выполнения.

2. При работе с механизмом исключений в С++ используются следующие ключевые слова: \_\_\_\_\_\_\_\_\_\_\_\_\_, \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_, \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_.

Try, throw, catch

3. 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.

Из точки, где возникла ошибка, в блок-ловушку.

Из выражения, в котором возникла ошибка, в блок-ловушку.

4. 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.

«New» не может зарезервировать необходимый объем памяти.

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

Да

6. Поясните механизм исключений.

Представим себе приложение, создающее объекты какого-либо класса и работающее с ними. В нормальной ситуации вызовы методов не приводят к ошибкам. Но иногда в программе возникает ошибка, которую обнаруживает метод. Он информирует программу об ошибке. При использовании исключений это событие называется генерацией исключительной ситуации. В приложении имеется отдельная секция кода, в которой находится операции по обработке ошибок. Этот код называется обработчиком исключительных ситуаций или улавливающим блоком. В нем отлавливаются исключения, сгенерированные методами. Любой код приложения, использующий объекты класса, заключается в блок повторных попыток. Соответственно, ошибки, возникшие в последнем, будут пойманы улавливающим блоком.

Давайте подведем итоги и покажем последовательность действий программы при возникновении ошибок:

1. Код нормально выполняется вне блока повторных попыток.
2. Управление переходит в блок повторных попыток.
3. Какое-то выражение в этом блоке приводит к возникновению ошибок в методе.
4. Метод генерирует исключение.
5. Управление переходит к обработчику ошибок (улавливающему блоку), следующему сразу за блоком повторных попыток.