Министерство образования и науки Российской Федерации
ФГАОУ ВО «КФУ им. В.И. Вернадского»
физико-технический институт (структурное подразделение)

Кафедра компьютерной инженерии и моделирования

Шенгелай Всеволод Михайлович

отчет по лабораторной работе №4
по дисциплине **«ОБЪЕКТНО-ОРИЕНТИРОВАННОЕ ПРОГРАММИРОВАНИЕ»**

Направление подготовки:
09.03.04 "Программная инженерия"

Оценка - 90

Симферополь, 2021

Лабораторная работа №4

Тема: Обработка исключительных ситуаций

Цель работы: Научиться на практике обрабатывать исключения, генерировать собственные исключения, отлавливать исключения различных типов, проверять числовые данные на выход за границы значений.

Описание ключевых понятий:

exception – Исключение, т.е. тип «ошибки» прерывающей исполнение программы;

try/catch – конструкция, позволяющая ловить и обрабатывать исключения, тем самым не прерывая исполнение программы;

finally – Возможное продолжение конструкции try/catch, обозначает часть программы, которую нужно обязательно исполнить после блока инструкций try, вне зависимости от выброшенного исключения;

checked – используется для явного включения проверки переполнения при выполнении арифметических операций и преобразований с данными целого типа;

unchecked – позволяет предотвратить проверку переполнения при выполнении арифметических операций и преобразований с данными целого типа.

throw – ключевое слово, позволяющее программисту сгенерировать и применить какое-то исключение, чтобы прервать исполнение программы.

Перед выполнением лабораторной работы изучена следующая литература:

1. Презентация лектора курса: «Обработка исключений в C#».
2. Шилдт Герберт. Полный справочник по C#. Пер. с англ. - М.: Издательский дом "Вильямс", 2008. - 752с.: ил. (глава 13).

3. Эндрю Троелсен. Язык программирования C# и платформы .NET и .NET CORE. Часть II. Глава 3.
4. Сайт MSDN Microsoft.
5. Сайт Metanit.com

Выполнены 3 задания, описанных в методических указания к выполнению лабораторных работ.

Задание 1: В этом упражнении мы добавили в приложение отказоустойчивую функциональность, чтобы оно продолжало работать даже в случае возникновения одного или нескольких исключений. Исключения генерируются методом Switch. Вызывая его методы, мы моделируем опрос датчиков систем ядерного реактора. Чтобы обеспечить отказоустойчивость нашей программы, мы используем конструкцию try/catch/finally, чтобы перехватить исключения, которые определены в методах класса Switch. При возникновении исключения мы выводим сообщение об ошибке и продолжаем опрос «датчиков».

```
// Step 2 - Verify the status of the Primary Coolant System
try
{
    primaryCoolantSystemStatus = sd.VerifyPrimaryCoolantSystem();
    if (primaryCoolantSystemStatus == SwitchDevices.CoolantSystemStatus.OK)
    {
        this.textBlock1.Text += "\nStep 2: Primary coolant system OK";
    }
    if (primaryCoolantSystemStatus == SwitchDevices.CoolantSystemStatus.Check)
    {
        this.textBlock1.Text += "\nStep 2: Primary coolant system requires manual check";
    }
    if (primaryCoolantSystemStatus == SwitchDevices.CoolantSystemStatus.Fail)
    {
        this.textBlock1.Text += "\nStep 2: Problem reported with primary coolant system";
    }
}
catch (SwitchDevices.CoolantTemperatureReadException M)
{
    this.textBlock1.Text += "\nStep 2: Problem reported with primary coolant system";
    this.textBlock1.Text += "\nERROR CODE: " + M.Message + '\n';
}
catch (SwitchDevices.CoolantPressureReadException M)
{
    this.textBlock1.Text += "\nStep 2: Problem reported with primary coolant system";
    this.textBlock1.Text += "\nERROR CODE: " + M.Message + '\n';
}
```

Рисунок 1. Проверка состояния первично системы охлаждения

Задание 2: По условию задания некоторое инженерное устройство выполняет несколько вычислений с использованием матриц. Он получает координаты наборов точек и загружает их в матрицы, затем, используя метод C#, перемножает их. При нормальных операциях ни один из элементов матриц не должен быть отрицательным. Иногда устройство генерирует значение -1 для точки данных.

Мы улучшили нашу программу умножения матриц, написанную во 2-й лабораторной работе, добавив возможность самостоятельного ввода пользователем строк второй матрицы.

В метод е MatrixMultiplication мы определяем 2 исключения ArgumentException, которые наступают в случае несовпадения количества столбцов первой матрицы и строк второй, и в случае, если одно из значений матриц меньше нуля.

В коде, где идёт вызов метода MatrixMultiplication мы перехватываем исключение, выводя в блоке catch описание исключения.

```
ссылка: 1
static double[,] MatrixMultiplication(double[,] matrixA, double[,] matrixB)
{

    if (matrixA.ColumnsCount() != matrixB.RowsCount())
    {
        throw new ArgumentException("Умножение не возможно! Количество столбцов первой матрицы не равно количеству строк второй матрицы.");
    }

    double[,] matrixC = new double[matrixB.ColumnsCount(), matrixA.RowsCount()];

    for (var i = 0; i < matrixB.ColumnsCount(); i++)
    {
        for (var j = 0; j < matrixA.RowsCount(); j++)
        {
            matrixC[i, j] = 0;

            for (int k = 0; k < matrixA.ColumnsCount(); k++)
            {
                //Считаем, что [x,y] - [строка, столбец]
                //Отсчёт начинаем с [1,1]
                if (matrixA[k, j] <= 0)
                {
                    throw new ArgumentException("Matrix1 contains an invalid entry in cell"+"["+ (j+1) +"," + (k+1) + "].");
                }
                else if (matrixB[i, k] <= 0)
                {
                    throw new ArgumentException("Matrix2 contains an invalid entry in cell" + "[" + (k+1) + "," + (i+1) + "].");
                }
                matrixC[i, j] += matrixA[k, j] * matrixB[i, k];
            }
        }
    }

    return matrixC;
}
```

Рисунок 2. Код метода MatrixMultiplication. Задание 2

Задание 3: По умолчанию проверяемое или непроверяемое состояние переполнения определяется путем установки соответствующего параметра компилятора и настройки самой среды выполнения. В нашем случае, при

помещении в переменную int результата выражения 2147483647*2, она примет значение -2. Используем ключевое слово checked для явного включения проверки переполнения при выполнении арифметических операций и преобразований с данными целого типа.

```
if (result_1 & result_2)
{
    try
    {
        //Включаем проверку переполнения
        int answer = checked (num1 * num2);
        label_1_Answer.Content = "Ответ: " + answer;
    }
    catch (OverflowException)
    {
        MessageBox.Show("Обработано исключение переполнения. Ответ не помещается в тип int.", "Сообщение",
        MessageBoxButton.OK, MessageBoxImage.Information);
    }
```

Рисунок 3. Проверка переполнения в задании 3.

Представлены 3 проекта, реализованных в Visual Studio Common Eddition 2019. Проекты представлены преподавателю в электронной форме, продемонстрирована их работоспособность, разъяснены детали программного кода.