Лабораторная работа № 6. Обработка исключительных ситуаций

1. Цель работы

Научиться обрабатывать исключительные ситуации в процессе работы программы. Вызов неуправляемого кода.

2. Базовые сведения

Исключение представляет собой ошибку, происходящую во время выполнения программы. С помощью подсистемы обработки исключений для C# можно обрабатывать такие ошибки, не вызывая краха программы.
Обработка исключений в C# выполняется с применением четырех ключевых слов: try, catch, throw и finally. Эти ключевые слова образуют взаимосвязанную подсистему, в которой использование одного из ключевых слов влечет за собой использование других.
Основа обработки исключений основана на использовании блоков try и catch.
Синтаксис:

```
try {
Блок_кода_для_которого_выполняется_мониторинг_ошибок
catch (ExcepType1 exOb) {
Обработчик_исключений_ExcepType1 }
catch (ExcepType2 exOb) {
Обработчик_исключений_ExcepType2 }
```

Основные системные исключения приведены в следующей таблице:

Исключение	**Значение**
ArrayTypeMismatchException	Тип сохраненного значения несовместим с типом массива.
DivideByZeroException	Предпринята попытка деления на ноль.
IndexOutOfRangeException	Индекс массива выходит за пределы диапазона.
InvalidCastException	Некорректное преобразование в процессе выполнения
OutOfMemoryException	Вызов new был неудачным из-за недостатка памяти.
OverflowException	Переполнение при выполнении арифметической операции.
StackOverflowException	Переполнение стека.

Тип исключения в операторе **catch** должен соответствовать типу перехватываемого исключения. Неперехваченное исключение непременно приводит к досрочному прекращению выполнения программы.

Для выполнения перехвата исключений вне зависимости от их типа (перехват всех исключений) возможно использование **catch** без параметров.

Возврат из исключения

Так как оператор **catch** не вызывается из программы, то после выполнения блока **catch** управление не передается обратно оператору программы, при выполнении которого возникло исключение. Выполнение программы продолжается с операторов, находящихся после блока **catch**.

С целью предотвращения этой ситуации возможно указание блока кода, который вызывается после выхода из блока **try/catch**, с помощью блока **finally** в конце последовательности **try/catch**. Общая форма конструкции **try/catch**, включающей блок **finally**, показана ниже:

```
try {
// Блок кода, выполняющий мониторинг ошибок }
catch (ExcepTypel exOЫ) {
/I Обработка исключения ExcepTypel. }
catch (ExcepType2 exOЬ2) {
II Обработка исключения ExcepType2 .
finally {
// Код блока finally. }
```

Блок finally будет вызываться независимо от того, появится исключение или нет, и независимо от причин возникновения такового.

Генерация исключений

Исключения автоматически генерируются системой. Однако исключение может быть сгенерировано и посредством оператора **throw**.

Синтаксис:

```
throw exceptOb;
```

Исключение, перехваченное одним оператором **catch**, может генерироваться повторно, благодаря чему оно может перехватываться внешним оператором **catch**. Для этого указывается ключевое слово **throw** без имени исключения.

Наследование классов исключений:

Можно создавать заказные исключения, выполняющие обработку ошибок в пользовательском коде. Генерирование исключений не представляет особых сложностей. Просто определите класс, наследуемый из класса Exception. В качестве общего правила руководствуйтесь тем, что определенные пользователем исключения наследуются из класса ApplicationException, так как они представляют собой иерархию зарезервированных исключений, связанных с приложениями. Наследуемые классы не нуждаются в фактической реализации в каком-либо виде, поскольку само их существование в системе типов данных позволяет воспользоваться ими в качестве исключений.

Создаваемые пользователем классы исключений автоматически получают доступные для них свойства и методы, определенные в классе Exception.

3. Задание

Реализовать в коде лабораторной работы перехват и обработку исключений, при этом переопределив с помощью наследования исключение (номер варианта делим по модулю 7):

1) StackOverflowException
2) ArrayTypeMismatchException
3) DivideByZeroException
4) IndexOutOfRangeException
5) InvalidCastException
6) OutOfMemoryException
7) OverflowException

Дополнить исключение своим блоком данных. Заполнить его в момент генерации. Отобразить в момент перехвата.

Реализовать свой тип исключения. Сгенерировать и выполнить его перехват. На возникновение исключения отреагировать выводом системного окна сообщения посредством вызова MessageBox из библиотеки user32.dll.

4. Варианты заданий

Придерживаться предметной области лабораторной работы №1.

5. Требования к оформлению отчета:

- титульный лист;
- название;
- цель работы;
- лабораторное задание;
- описание метода решения задачи;
- листинг (текст программы);
- пояснительный текст к программе;
- результаты работы программы;
- выводы.

6. Контрольные вопросы

1) Что такое обработка исключений?
2) Оператор try.
3) catch - блок обработки исключений.
4) finally - блок завершения
5) Каким образом возможно осуществить явную генерацию исключений?
6) Каким образом обеспечивается обращение к свойствам и методам системных исключений?

7. Список рекомендованной литературы

1. Васильев А. C#. Объектно-ориентированное программирование: Учебный курс. – СПб.: Питер, 2012. – 320 с.: ил.

2. Гамма Э., Хелм Р., Джонсон Р., Влиссидес Дж. Приемы объектно-ориентированного программирования. Паттерны проектирования. – СПб: Питер, 2009. – 366 с.: ил.
3. Герберт Шилдт. C# 3.0. Полное руководство. - Изд. Вильямс, 2010.
4. Нейгел К., Ивьен Б. и др. C# 2008 и платформа NET 3.5 для профессионалов. – Изд. Диалектика, 2008.
5. Трей Нэш. C# 2010. Ускоренный курс для профессионалов. - Изд. Вильямс, 2010.
6. Троелсен Э. Язык программирования C# 2008 и платформа .NET 3.5- Изд. Вильямс, 2010.
7. Стиллмен Э., Грин Дж. Изучаем C# [пер. с англ. И. Рузмайкина]. - 2-е изд. - Москва: Питер, 2012. – 694 с. : ил.