**ВОПРОСЫ К ЛАБОРАТОРНОЙ РАБОТЕ №7:**

1. Расскажите как генерируется исключение.

Исключительная ситуация – состояние ошибки, обнаруженное в программе в ходе ее выполнения (/0, невозможно выделение памяти…)

if (b==0)  
 throw new Exception(“Zero div”);

2. Расскажите методику обработки исключений.

try – контролируемый блок  
throw – генерация искл. ситуации внутри try  
catch – обработчик исключений  
finally – код, очищающий ресурсы и др. д-вия (выполнение всегда, 1 на 1 try)

3. Какое ключевое слово служит для обозначения блока кода, в котором можно генерировать исключение?

**Throw**

4. Какие ключевые слова используются для обработки и генерации исключений? Расскажите об механизме обработки исключения?

1) Исключения нет  
 try до конца  
 catch пропуск  
 finally вып.  
2) Исключение есть  
 try стоп  
 ищем catch (на соотв. по типу исключение)  
\* есть catch:  
 ближ. catch  
 объект-исключение в качестве параметра (если предусмотрено)  
 - finally  
 - нет finally -> выполнение программы после посл. catch данного try  
\* нет catch:  
 разматываем стек, лок. объекты выходят из обл. видимости  
 снова генер. искл  
 сообщение «Необработанное исключение», программа стоп

5. Что будет, если в программе нет предложения catch, способного обработать исключение?

Если подходящих обработчиков **catch** не существует, то выполнение программы переходит к следующему блоку **try**. Если до конца программы не найдены соответствующие обработчики **catch**, то программа завершает свое выполнение с ошибкой исключения.

Код в блоке **finally** выполняется, даже если создано исключение. Используйте блок **finally**, чтобы освободить ресурсы, например закрыть потоки и файлы, которые были открыты внутри блока **try**.

6. Что такое фильтры исключения? Приведите пример.

Фильтр искл. – позв. указать доп. условия, при кот. исп. обработчик исключения  
Эти условия bool, перед ними **WHEN:**

catch (Exception ex) when  
 (ex.GetType() != typeof(System.FormatException))  
 … //обработка исключений  
 //кроме того, кот. называется FormatException

7. Могут ли исключения быть вложенными?

**try** может

8. Какой синтаксис нужно использовать в C# для отлова любого возможного исключения?

Блок **catch**, отлавливающий исключения типа System.Exception. Также можно просто использовать catch{}, не указывая тип исключения.

9. Чем следует руководствоваться при размещении обработчиков исключения?

\* сначала более специфичное исключение  
\* далее наследуется от System.Exception или System.ApplicationException  
\* мб один catch без аргумент (нежелательно)  
\* finally вып всегда  
\* при lock, using, foreach блоки try/finally созд. авто-  
\* генерация исключения в finally нежелательно

11. Как повторно сгенерировать то же самое исключение в блоке обработчике catch?

try  
 int res = x / y  
catch (DivideByZeroException)  
 Console.WriteLine(“Деление на ноль!”);  
 throw; //при повт. вызове исключения с пом. throw  
. //инфа о нач. точке ост. в стеке

12. Какие методы содержатся в классе Exception? Где и как их можно использовать?

**Exception** — это базовый класс, от которого наследуются исключения. Например, иерархия класса InvalidCastException имеет следующий вид:

*Object*

*Exception*

*SystemException*

*InvalidCastException*

Все определяемые на уровне пользователя и системы исключения в конечном итоге всегда наследуются от базового класса System.Exception, который, в свою очередь, наследуется от класса System.Object.

|  |  |
| --- | --- |
| **Свойство** | **Описание** |
| *Data* | Это свойство, доступное только для чтения, позволяет извлекать коллекцию пар "ключ/значение" (представленную объектом, реализующим интерфейс IDictionary), которая предоставляет дополнительную, определяемую программистом, информацию об исключении. По умолчанию эта коллекция является пустой. |
| *HelpLink* | Это свойство позволяет получать или устанавливать URL-адрес, по которому доступен справочный файл или веб-сайт с детальным описанием ошибки. |
| *InnerException* | Это свойство, доступное только для чтения, может применяться для получения информации о предыдущем исключении или исключениях, которые послужили причиной возникновения текущего исключения. Запись предыдущих исключений осуществляется путем их передачи конструктору самого последнего исключения. |
| *Message* | Это свойство, доступное только для чтения, возвращает текстовое описание соответствующей ошибки. Само сообщение об ошибке задается в передаваемом конструктору параметре. |
| *Source* | Это свойство позволяет получать или устанавливать имя сборки или объекта, который привел к выдаче исключения. |
| *StackTrace* | Это свойство, доступное только для чтения, содержит строку с описанием последовательности вызовов, которая привела к возникновению исключения. Как нетрудно догадаться, это свойство очень полезно во время отладки или для сохранения информации об ошибке во внешнем журнале ошибок. |
| *TargetSite* | Это свойство, доступное только для чтения, возвращает объект MethodBase с описанием многочисленных деталей метода, который привел к выдаче исключения (вызов вместе с ним ToString() позволяет идентифицировать этот метод по имени) |