1. Расскажите как генерируется исключение.

Обычно система сама генерирует исключения при определенных ситуациях, например, при делении числа на ноль. Но язык C# также позволяет генерировать исключения вручную с помощью оператора throw. То есть с помощью этого оператора мы сами можем создать исключение и вызвать его в процессе выполнения.

1. Расскажите методику обработки исключений.

Язык C# предоставляет разработчикам возможности для обработки таких ситуаций. Для этого в C# предназначена конструкция **try...catch...finally**.

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 обработке исключения?

**try...catch...finally**. throw. Try проверяет условие catch выводит ошибку finally выводит сообщение после блока catch throw выбрасывает ошибку

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

обработать исключение?

Если в программе нет предложения catch, способного обработать исключение, то исключение будет выдано и программа прекратит свое выполнение. Это приведет к завершению работы программы и возможно выводу сообщения об ошибке.

6. Что такое фильтры исключения? Приведите пример

Фильтры исключений позволяют обрабатывать исключения в зависимости от определенных условий. Для их применения после выражения catch идет выражение when, после которого в скобках указывается условие: catch when(условие) { }

7. Могут ли исключения быть вложенными?

Один блок try может быть вложен в другой. Исключение, генерируемое во внутреннем блоке try и не перехваченное в соответствующем блоке catch, передается во внешний блок try:

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

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

9. Чем следует руководствоваться при размещении обработчиков

исключения?

Используйте минимально возможный блок try-catch

Улавливайте и обрабатывайте только те исключения, которые вы можете обработать

Предоставьте информативные сообщения об ошибках:

Используйте блок finally для очистки ресурсов

Возможно, используйте несколько блоков catch для разных типов исключений

Не подавляйте исключения без необходимости

Используйте иерархию исключений

10.Что будет выведено на консоль в результате выполнения фрагмента

листинга?

static void Main(string[] args)

{

string[] str = new string[5];

try {

str[4] = "anything";

Console.WriteLine("It's OK");

}

catch (IndexOutOfRangeException e)

{

Console.WriteLine("IndexOutOfRangeException");

}

catch (Exception e)

{

Console.WriteLine("Exception");

}

}

**}**

11.Как повторно сгенерировать то же самое исключение в блоке

обработчике catch?

Если возникает необходимость снова сгенерировать исключения из блока, который обрабатывает исключения, можно сделать это путем вызова throw без указания исключения. В результате текущее исключение будет передано во внешнюю последовательность try/catch обработки исключений.

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

использовать?

InnerException: хранит информацию об исключении, которое послужило причиной текущего исключения

Message: хранит сообщение об исключении, текст ошибки

Source: хранит имя объекта или сборки, которое вызвало исключение

StackTrace: возвращает строковое представление стека вызовов, которые привели к возникновению исключения

TargetSite: возвращает метод, в котором и было вызвано исключение