**Ответы на теоретические вопросы**

**№1**

1. **unchecked**:
   * Когда операции выполняются в контексте unchecked, переполнение арифметики игнорируется, и результат обрезается, чтобы он соответствовал типу назначения.
   * Например, при сложении, если результат не помещается в целевой тип, он оборачивается от максимального значения к минимальному.
   * В вашем коде, вы используете unchecked для вычисления суммы x + y. Результат будет обрезан, если он выходит за пределы диапазона byte.
2. **checked**:
   * В контексте checked переполнение вызывает исключение System.OverflowException.
   * Если переполнение происходит в константном выражении, возникает ошибка времени компиляции.
   * В вашем коде, вы используете checked для вычисления суммы x + y. Если результат выходит за пределы диапазона byte, будет выброшено исключение

Если мы уберем unchecked, то в данном случае результат сложения x + y будет обрезан до значения, которое может храниться в типе byte. В данном конкретном примере, результат будет равен 144 (200 + 200 = 400, обрезаем до 144).

**№2**

1. checked блок:

* В этом блоке включена проверка на переполнение.
* Если результат вычисления выходит за пределы допустимого диапазона, будет выброшено исключение System.OverflowException.
* В данном случае, результат вычисления факториала числа 9 превышает максимальное значение типа byte, и будет выброшено исключение.

1. unchecked блок:

* В этом блоке отключена проверка на переполнение.
* Результат вычисления обрезается, если он выходит за пределы допустимого диапазона. В данном случае, результат также превышает максимальное значение типа byte, но вместо исключения, он обрезается до значения, которое может храниться в типе byte.

Таким образом, в результате выполнения программы, мы увидим следующее:

1: 1: 128

Возникло переполнение!

2: Возникло переполнение!

В первом блоке (checked) обрезалась до 128. Во втором блоке возникло переполнение.

№3

1. **Почему не было сгенерировано исключение**DivideByZeroException:
   * Исключение DivideByZeroException не было сгенерировано, потому что в вашем коде вы используете тип данных double для вычисления выражения number1 / number2.
   * В языке C#, деление числа с плавающей запятой на ноль не вызывает исключение DivideByZeroException. Вместо этого результатом будет положительная бесконечность (Infinity), отрицательная бесконечность (-Infinity) или не число (NaN), согласно правилам арифметики IEEE 754.
   * В данном случае, вы используете деление с плавающей запятой, поэтому исключение не возникает.
2. **Собственное исключение**DivideByZeroException:
   * Давайте создадим собственное исключение DivideByZeroException и обработаем его для ситуации, когда f - 1 < 0.000001.
   * Вот обновленный код:

Листинг кода:

using System;

class Program

{

static void Main()

{

try

{

genException();

}

catch (DivideByZeroException)

{

Console.WriteLine("Собственное исключение: Деление на ноль!");

}

}

static void genException()

{

Console.WriteLine("a = ");

double a = double.Parse(Console.ReadLine());

Console.WriteLine("b = ");

double b = double.Parse(Console.ReadLine());

int f = 1;

try

{

for (double i = a; i <= b; ++i)

{

f = checked((int)(f \* i));

double result = 100 / (f - 1);

if (result < 0.000001)

{

throw new DivideByZeroException();

}

Console.WriteLine("y({0}) = {1:f6}", i, result);

}

}

catch (ArithmeticException)

{

Console.WriteLine("ERROR");

throw; // Повторная генерация исключения

}

}

}

№4

a = 4; b = 2;

Вывод:

4 / 2 = 2

a = 3; b = g;

Вывод:

Нужно ввести число!

a = d; b = 1;

Вывод:

Нужно ввести число!

a = 2; b = 0;

Вывод:

Делить на нуль нельзя!

a=123456789987654321; b = 1;

Вывод:

Какая-то ошибка