В C# имеется специальное средство, связанное с генерированием исключений, возникающих

при переполнении в арифметических вычислениях. Как вам должно быть

уже известно, результаты некоторых видов арифметических вычислений могут превышать

диапазон представления чисел для типа данных, используемого в вычислении.

В этом случае происходит так называемое *переполнение.*

**Применение ключевых слов checked и unchecked**

Рассмотрим в качестве примера следующий фрагмент кода.

byte a, b, result;

а = 127;

b = 127;

result = (byte) (а\* b);

В этом коде произведение значений переменных а и b превышает диапазон представления

чисел для типа byte. Следовательно, результат вычисления данного выражения

приводит к переполнению для типа данных, сохраняемого в переменной

result.

В C# допускается указывать, будет ли в коде сгенерировано исключение при переполнении,

с помощью ключевых слов checked и unchecked. Так, если требуется указать,

что выражение будет проверяться на переполнение, следует использовать ключевое

слово checked, а если требуется проигнорировать переполнение — ключевое

слово unchecked. В последнем случае результат усекается, чтобы не выйти за пределы

диапазона представления чисел для целевого типа выражения.

У ключевого слова checked имеются две общие формы. В одной форме проверяется

конкретное выражение, и поэтому она называется *операторной.* А в другой форме

проверяется блок операторов, и поэтому она называется *блочной.* Ниже приведены обе

формы:

checked (выражение)

checked {

// проверяемые операторы

}

где *выражение* обозначает проверяемое выражение. Если вычисление проверяемого

выражения приводит к переполнению, то генерируется исключение

OverflowException.

У ключевого слова unchecked также имеются две общие формы. В первой, операторной

форме переполнение игнорируется при вычислении конкретного выражения.

А во второй, блочной форме оно игнорируется при выполнении блока операторов:

unchecked (выражение)

unchecked {

// операторы, для которых переполнение игнорируется

}

где *выражение* обозначает конкретное выражение, при вычислении которого переполнение

игнорируется. Если же в непроверяемом выражении происходит переполнение,

то результат его вычисления усекается.

(***glava15\_13***)

class CheckDemo

{

static void Main()

{

byte a, b;

byte result;

a = 127;

b = 127;

try

{

result = unchecked((byte)(a \* b));

Console.WriteLine("Unchecked overflow result: " + result);

result = checked((byte)(a \* b)); //this operation leads to exception

Console.WriteLine("Checked overflow result: " + result); //no output

}

catch(OverflowException exc)

{

Console.WriteLine(exc);

}

}

}

Как видите, результат вычисления непроверяемого выражения был усечен. А вычисление

проверяемого выражения привело к исключительной ситуации.

В представленном выше примере программы было продемонстрировано применение

ключевых слов checked и unchecked в одном выражении. А в следующем примере

программы показывается, каким образом проверяется и не проверяется на переполнение

целый блок операторов.

(***glava15\_13.1***)

class CheckBlocks

{

static void Main()

{

byte a, b;

byte result;

a = 127;

b = 127;

try

{

unchecked

{

result = unchecked((byte)(a \* b));

Console.WriteLine("Unchecked overflow result: " + result);

a = 125;

b = 5;

result = unchecked((byte)(a \* b));

Console.WriteLine("Unchecked overflow result: " + result);

}

checked

{

a = 2;

b = 7;

result = checked((byte)(a\*b)); //correct

Console.WriteLine("Checked overflow result: " + result);

a = 127;

b = 127;

result = checked((byte)(a \* b));//this leads to exception

Console.WriteLine("Checked overflow result: " + result);//wount output

}

}

catch (OverflowException exc)

{

Console.WriteLine(exc);

}

}

}

Как видите, результаты выполнения непроверяемого на переполнение блока операторов

были усечены. Когда же в проверяемом блоке операторов произошло переполнение,

то возникла исключительная ситуация.

Потребность в применении ключевого слова checked или unchecked может возникнуть,

в частности, потому, что по умолчанию проверяемое или непроверяемое состояние

переполнения определяется путем установки соответствующего параметра

компилятора и настройки самой среды выполнения. Поэтому в некоторых программах

состояние переполнения лучше проверять явным образом.