checked and **unchecked** Statements

C# statements can execute in either checked or unchecked context. In a checked context, arithmetic overflow raises an exception. In an unchecked context, arithmetic overflow is ignored and the result is truncated by discarding any high-order bits that don't fit in the destination type.

- `checked` keyword Specify checked context.
- `unchecked` keyword Specify unchecked context.

The following operations are affected by the overflow checking:

- Expressions using the following predefined operators on integral types:
  `++`, `--`, unary `-`, `+`, `-`, `*`, `/`
- Explicit numeric conversions between integral types, or from float or double to an integral type.

checked Statement

The `checked` keyword is used to explicitly enable overflow checking for integral-type arithmetic operations and conversions. By default, an expression that contains only constant values causes a compiler error if the expression produces a value that is outside the range of the destination type. If the expression contains one or more non-constant values, the compiler does not detect the overflow.

Example:

```
class OverFlowTest
{
    // Set maxIntValue to the maximum value for integers.
    static int maxIntValue = 2147483647;
    // Using a checked expression.
    static int CheckedMethod()
    {
        int z = 0;
        try
        {
            // The following line raises an exception because it is checked.
            z = checked(maxIntValue + 10);
        }
        catch (System.OverflowException e)
        {
            // The following line displays information about the error.
            Console.WriteLine("CHECKED and CAUGHT:  " + e.ToString());
        }
        // The value of z is still 0.
        return z;
    }
    // Using an unchecked expression.
    static int UncheckedMethod()
    {
        int z = 0;
        try
        {
            // The following calculation is unchecked and will not
            // raise an exception.
            z = maxIntValue + 10;
        }
        catch (System.OverflowException e)
        {
            // The following line will not be executed.
            Console.WriteLine("UNCHECKED and CAUGHT:  " + e.ToString());
        }
        // Because of the undetected overflow, the sum of 2147483647 + 10 is
        // returned as -2147483639.
        return z;
    }
    static void Main()
    {
        Console.WriteLine("\nCHECKED output value is: {0}",
                          CheckedMethod());
        Console.WriteLine("UNCHECKED output value is: {0}",
                          UncheckedMethod());
    }
    /*
   Output:
   CHECKED and CAUGHT:  System.OverflowException: Arithmetic operation resulted
   in an overflow.
      at ConsoleApplication1.OverFlowTest.CheckedMethod()

   CHECKED output value is: 0
   UNCHECKED output value is: -2147483639
 */
}
```

unchecked Statement

The unchecked keyword is used to suppress overflow-checking for integral-type arithmetic operations and conversions. In an unchecked context, if an expression produces a value that is outside the range of the destination type, the overflow is not flagged. For example, because the calculation in the following example is performed in an unchecked block or expression, the fact that the result is too large for an integer is ignored, and int1 is assigned the value -2,147,483,639.

```
unchecked
{
    int1 = 2147483647 + 10;
}
int1 = unchecked(ConstantMax + 10);
```

If the unchecked environment is removed, a compilation error occurs. The overflow can be detected at compile time because all the terms of the expression are constants. Expressions that contain non-constant terms are unchecked by default at compile time and run time.