**C# Data types and sizes**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| C# type/keyword | Range | Precision | Size | .NET type |
| Integral numeric types |  |  |  |  |
| sbyte | -128 to 127 |  | Signed 8-bit integer | System.SByte |
| byte | 0 to 255 |  | Unsigned 8-bit integer | System.Byte |
| short | -32,768 to 32,767 |  | Signed 16-bit integer | System.Int16 |
| ushort | 0 to 65,535 |  | Unsigned 16-bit integer | System.UInt16 |
| int | -2,147,483,648 to 2,147,483,647 |  | Signed 32-bit integer | System.Int32 |
| uint | 0 to 4,294,967,295 |  | Unsigned 32-bit integer | System.UInt32 |
| long | -9,223,372,036,854,775,808 to 9,223,372,036,854,775,807 |  | Signed 64-bit integer | System.Int64 |
| ulong | 0 to 18,446,744,073,709,551,615 |  | Unsigned 64-bit integer | System.UInt64 |
| Floating-point numeric types |  |  |  |  |
| float | ±1.5 x 10−45 to ±3.4 x 1038 | ~6-9 digits | 4 bytes | System.Single |
| double | ±5.0 × 10−324 to ±1.7 × 10308 | ~15-17 digits | 8 bytes | System.Double |
| decimal | ±1.0 x 10-28 to ±7.9228 x 1028 | 28-29 digits | 16 bytes | System.Decimal |
| char |  |  |  |  |
| char | U+0000 to U+FFFF |  | 16 bit | System.Char |

<https://docs.microsoft.com/en-us/dotnet/csharp/language-reference/builtin-types/integral-numeric-types>

**Example for bitwise XOR**

using System;

class HW {

static void Main() {

bool x = true, y = false, z = true;

Console.WriteLine(x ^ y); // true,

Console.WriteLine(x ^ z); // false,

Console.WriteLine(z ^ y); // true,

}

}

**Example for && and & (short & long AND)**

using System;

class HW {

static void Main() {

int x, y, z; x = 100; y = 1; z = 0;

Console.WriteLine((x/y != 0) && (x/z != 0));

//Console.WriteLine((x/y != 0) & (x/z != 0));

}

}

**Example for five geometrical shapes and operations**

using System;

class HW {

static void Main() {

int a = 4, b = 6, c = 8, r = 5;

Console.WriteLine(areaRectangle(a, b));

Console.WriteLine(perimeterRectangle(a, b));

Console.WriteLine(areaSquare(a));

Console.WriteLine(perimeterSquare(a));

Console.WriteLine(areaCircle(r));

Console.WriteLine(circumferenceCircle(r));

Console.WriteLine(areaTriangle1(a));

Console.WriteLine(perimeterTriangle1(a));

Console.WriteLine(areaTriangle2(a, b, c));

Console.WriteLine(perimeterTriangle2(a, b, c));

}

static public int areaRectangle(int a, int b) {

return a\*b;

}

static public int perimeterRectangle(int a, int b) {

return 2\*(a+b);

}

static public int areaSquare(int a) {

return a\*a;

}

static public int perimeterSquare(int a) {

return 4\*a;

}

static public double areaCircle(int r) {

return r\*r\*Math.PI;

}

static public double circumferenceCircle(int r) {

return 2\*Math.PI\*r;

}

//equal-sized

static public double areaTriangle1(int a) {

return a\*a/2\*Math.Sqrt(3);

}

static public int perimeterTriangle1(int a) {

return 3\*a;

}

//right-angled

static public int areaTriangle2(int a, int b, int c) { //c -> hypothenus

return a\*b/2;

}

static public int perimeterTriangle2(int a, int b, int c) {

return a+b+c;

}

}