01 Introduction to C# and Data Types

Understanding Data Types

Test your Knowledge

1. What type would you choose for the following “numbers”?
    a. A person’s telephone number
       string
    b. A person’s height
       float
    c. A person’s age
       uint
    d. A person’s gender (Male, Female, Prefer Not To Answer)
       string
    e. A person’s salary
       decimal
    f. A book’s ISBN
       string
    g. A book’s price
       decimal
    h. A book’s shipping weight
       float
    i. A country’s population
       ulong
    j. The number of stars in the universe
       ulong
    k. The number of employees in each of the small or medium businesses in the United Kingdom (up to about 50,000 employees per business)
       uint
2. What is the difference between value type and reference type variables?
   Value Type:
   1. store its value in stack memory
   2. will collect by the garbage collector.
   3. cannot be null
   4. create by structure, emun

   Reference type:
   1. store the memory address in heap memory,
   2. will collect by the garbage collector.
   3. cannot be null
   4. create by classes, interfaces, delegates, array
3. What are boxing and unboxing?
   Boxing: convert a value type into a reference type.
   Unboxing: convert a reference type into a value type.
4. What is meant by the terms managed resource and unmanaged resource in .NET?
   Manageable means it is under the control of the garbage collector.

5. What is the purpose of Garbage Collector in .NET?
   To release heap memory by deleting reference values that are no longer in use.

Playing with Console App

Modify your console application to display a different message. Go ahead and intentionally add some mistakes to your program, so you can see what kinds of error messages you get from the compiler. The more familiar you are with these messages, and what causes them, the better you'll be at diagnosing problems in your programs that you /didn't/ intend to add!

Using just the `ReadLine` and `WriteLine` methods and your current knowledge of variables, you can have the user pass in quite a few bits of information. Using this approach, create a console application that asks the user a few questions and then generates some custom output for them. For instance, your program could generate their "hacker name" by asking them their favorite color, their astrology sign, and their street address number. The result might be something like "Your hacker name is RedGemini480."

Practice number sizes and ranges

1. Create a console application project named /02UnderstandingTypes/ that outputs the number of bytes in memory that each of the following number types uses, and the minimum and maximum values they can have: sbyte, byte, short, ushort, int, uint, long, ulong, float, double, and decimal.
   Composite Formatting to learn how to align text in a console application

```
Console.WriteLine(
   "sbyte:\t-128 to +127\n" +
   "byte:\t0 to 255\n" +
   "short:\t-32,768 to 32,767\n" +
   "ushort:\t0 to 65,535\n" +
   "int:\t-2,147,483,648 to 2,147,483,647\n" +
   "uint:\t0 to 4,294,967,295\n" +
   "long:\t-9,223,372,036,854,775,808  to 9,223,372,036,854,775,807\n" +
   "ulong:\t0 to 18,446,744,073,709,551,615\n" +
   "float:\t±1.0e-45 to ±3.4e38\n" +
   "double:\t±5e-324 to ±1.7e308\n" +
   "decimal:±1.0 ×10e-28 to ±7.9e28\n");
```

2. Write program to enter an integer number of centuries and convert it to years, days, hours, minutes, seconds, milliseconds, microseconds, nanoseconds. Use an appropriate data type for every data conversion. Beware of overflows!

```
Input: 1

Output: 1 centuries = 100 years = 36524 days = 876576
hours = 52594560 minutes
```

```
= 3155673600 seconds = 3155673600000 milliseconds =
3155673600000000
microseconds = 3155673600000000000 nanoseconds

Input: 5

Output: 5 centuries = 500 years = 182621 days = 4382904
hours = 262974240
minutes = 15778454400 seconds = 15778454400000
milliseconds = 15778454400000000
microseconds = 15778454400000000000 nanoseconds
```

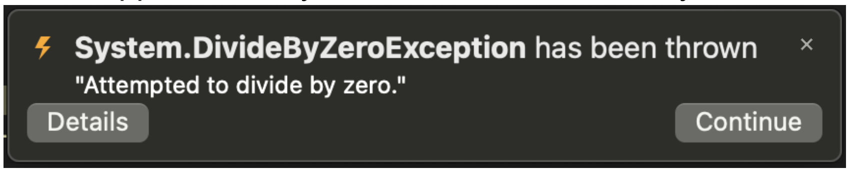
```
string val;
Console.Write("Enter integer: ");
val = Console.ReadLine();
int a = Convert.ToInt32(val);
long b = Convert.ToInt64(val);

Console.Write(
   $"Output:{a} centuries = {a * 100} years = {a * 100 * 365.24} days\n" +
   $"= {a * 100 * 365.24 * 24} hours\n" +
   $"= {b * 100 * 365.24 * 24 * 60} minutes\n" +
   $"= {b * 100 * 365.24 * 24 * 60 * 60} seconds\n" +
   $"= {b * 100 * 365.24 * 24 * 60 * 60 * 1000} milliseconds\n" +
   $"= {b * 100 * 365.24 * 24 * 60 * 60 * 1000 * 1000} microseconds\n" +
   $"= {b * 100 * 365.24 * 24 * 60 * 60 *1000 * 1000 *1000}
nanoseconds\n");
```

Explore following topics
- C# Keyword
- Main() and command-line argument
- Types (C# Programming Guide)
- Statements, Expressions, and Operators
- Strings (C# Programming Guide)
- Nullable Types (C# Programming Guide)
- Nullable reference types

Controlling Flow and Converting Types

Test your Knowledge

1. What happens when you divide an int variable by 0?



System.DivideByZeroException has been thrown
"Attempted to divide by zero."

2. What happens when you divide a double variable by 0?
   Infinity.

3. What happens when you overflow an int variable, that is, set it to a value beyond its range?
   Unable to compile the code. A red underline occurs at the assignment statement.
4. What is the difference between x = y++; and x = ++y;?
   “x = y++;”: the x will be assigned with the y value first, then the y variable increased by one.
   “x = ++y;”: the y variable increased by one, then x will be assigned with the y value.
5. What is the difference between break, continue, and return when used inside a loop statement?
   break: leave the current loop.
   continue: jump to the next iteration.
   return: exit the current method.
6. What are the three parts of a for statement and which of them are required?
   Initialize, test, and update.
7. What is the difference between the = and == operators?
   = Assignment
   == Equal to
8. Does the following statement compile? for (; true;) ;
   Yes, it’s an infinite loop.
9. What does the underscore _ represent in a switch expression?

10. What interface must an object implement to be enumerated over by using the foreach statement

Practice loops and operators

1. FizzBuzz is a group word game for children to teach them about division. Players take turns to count incrementally, replacing any number divisible by three with the word /fizz/, any number divisible by five with the word /buzz/, and any number divisible by both with /fizzbuzz/.
   Create a console application in Chapter03 named Exercise03 that outputs a simulated FizzBuzz game counting up to 100. The output should look something like the following
   screenshot:

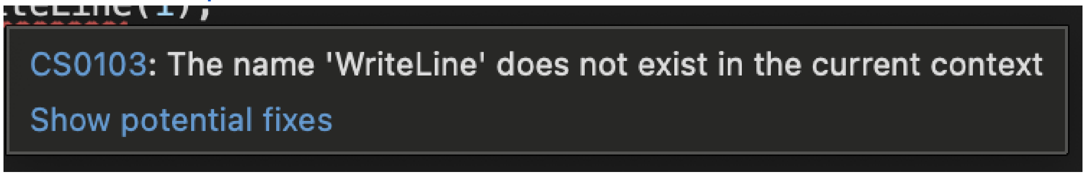
```
//FizzBuzzis
int c = 0;

for (byte i = 1; i < 101; i++) {
    c++;
    if (c % 15 == 0)
    {
        Console.WriteLine("FizzBuzz");
    }
    else if (c % 5 == 0)
    {
        Console.WriteLine("Buzz");
    }
    else if (c % 3 == 0)
    {
        Console.WriteLine("Fizz");
    }
    else {
        Console.WriteLine(c);
    }
}
```

What will happen if this code executes?

```
int max = 500;
for(byte i = 0; i < max; i++)
{
     WriteLine(i);
}
```

Create a console application and enter the preceding code. Run the console application and view the output. What happens?

Unable to compile.



CS0103: The name 'WriteLine' does not exist in the current context

Show potential fixes

What code could you add (don't change any of the preceding code) to warn us about the problem?

Your program can create a random number between 1 and 3 with the following code:

```
int correctNumber = new Random().Next(3)+1;
```

Write a program that generates a random number between 1 and 3 and asks the user to guess what the number is. Tell the user if they guess low, high, or get the correct answer. Also, tell the user if their answer is outside of the range of numbers that are valid guesses (less than 1 or more than 3). You can convert the user's typed answer from a `string` to an `int` using this code:

```
int guessedNumber = int.Parse(Console.ReadLine());
```

Note that the above code will crash the program if the user doesn't type an integer value.
For this exercise, assume the user will only enter valid guesses.

2. Print-a-Pyramid. Like the star pattern examples that we saw earlier, create a program that will print the following pattern: If you find yourself getting stuck, try recreating the two examples that we just talked about in this chapter first. They're simpler, and you can compare your results with the code included above.
   This can actually be a pretty challenging problem, so here is a hint to get you

going. I used three total loops. One big one contains two smaller loops. The bigger loop goes from line to line. The first of the two inner loops prints the correct number of spaces, while the second inner loop prints out the correct number of stars.

```
    *
   ***
  *****
 *******
*********
```

```
int num = 5;

for (int i = 1; i<=num; i++){
    int starts = i * 2 - 1;
    string layer = new String(' ', num-i)+ new
String('*', starts)+ new String(' ', num - i);
    Console.WriteLine(layer);
}
```

3. Write a program that generates a random number between 1 and 3 and asks the user to guess what the number is. Tell the user if they guess low, high, or get the correct answer.
   Also, tell the user if their answer is outside of the range of numbers that are valid guesses (less than 1 or more than 3). You can convert the user's typed answer from a `string` to an `int` using this code:

   ```
   int guessedNumber = int.Parse(Console.ReadLine());
   ```

   Note that the above code will crash the program if the user doesn't type an integer value.
   For this exercise, assume the user will only enter valid guesses.

4. Write a simple program that defines a variable representing a birth date and calculates how many days old the person with that birth date is currently.
   For extra credit, output the date of their next 10,000 day (about 27 years) anniversary.
   Note: once you figure out their age in days, you can calculate the days until the next anniversary using
   `int daysToNextAnniversary = 10000 - (days % 10000);`

5. Write a program that greets the user using the appropriate greeting for the time of day. Use only `if`, not `else` or `switch`, statements to do so. Be sure to include the following greetings:
   - "Good Morning"
   - "Good Afternoon"
   - "Good Evening"
   - "Good Night"

   It's up to you which times should serve as the starting and ending ranges for each of the greetings. If you need a refresher on how to get the current time, see DateTime Formatting. When testing your program, you'll probably want to use a `DateTime` variable you define, rather than the current time. Once you're confident the program works correctly, you can substitute `DateTime.Now` for your variable (or keep your variable and just assign `DateTime.Now` as its value, which is often a better approach).

6. Write a program that prints the result of counting up to 24 using four different increments. First, count by 1s, then by 2s, by 3s, and finally by 4s. Use nested `for` loops with your outer loop counting from 1 to 4. Your inner loop should count from 0 to 24, but increase the value of its /loop control variable/ by the value of the /loop control variable/ from the outer loop. This means the incrementing in the / afterthought/ expression will be based on a variable. Your output should look something like this:

```
0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,2
2,23,24
0,2,4,6,8,10,12,14,16,18,20,22,24
0,3,6,9,12,15,18,21,24
0,4,8,12,16,20,24
```

Explore following topics

- C# operator
- Bitwise and shift operator
- Statement keyword
- Casting and type conversion
- Fundamentals of garbage collection
- $ - string interpolation
- Formatting types in .NET
- Iteration statements
- Selection statements