01 Introduction to C# and Data Types

Understanding Data Types

Test your Knowledge

1. What type would you choose for the following "numbers"?
A person's telephone number: string
A person's height: float
A person's age: int
A person's gender (Male, Female, Prefer Not To Answer): string
A person's salary: decimal
A book's ISBN: long
A book's price: float
A book's shipping weight: float
A country's population: int
The number of stars in the universe: ulong
The number of employees in each of the small or medium businesses in the United Kingdom (up to about 50,000 employees per business): ushort

2. What are the difference between value type and reference type variables? What is boxing and unboxing?

Value Types	Reference Types
Directly hold the value	Hold the memory address or reference for its value
Be stored in stack memory	Be stored in heap memory
Will not be collected by garbage collector	Will be collected by garbage collector

Boxing: Convert a value type into a reference type.

Unboxing: Convert a reference type back to value type.

3. What is meant by the terms managed resource and unmanaged resource in .NET

Managed resources are those that are pure .NET code and managed by the runtime and are under its direct control.

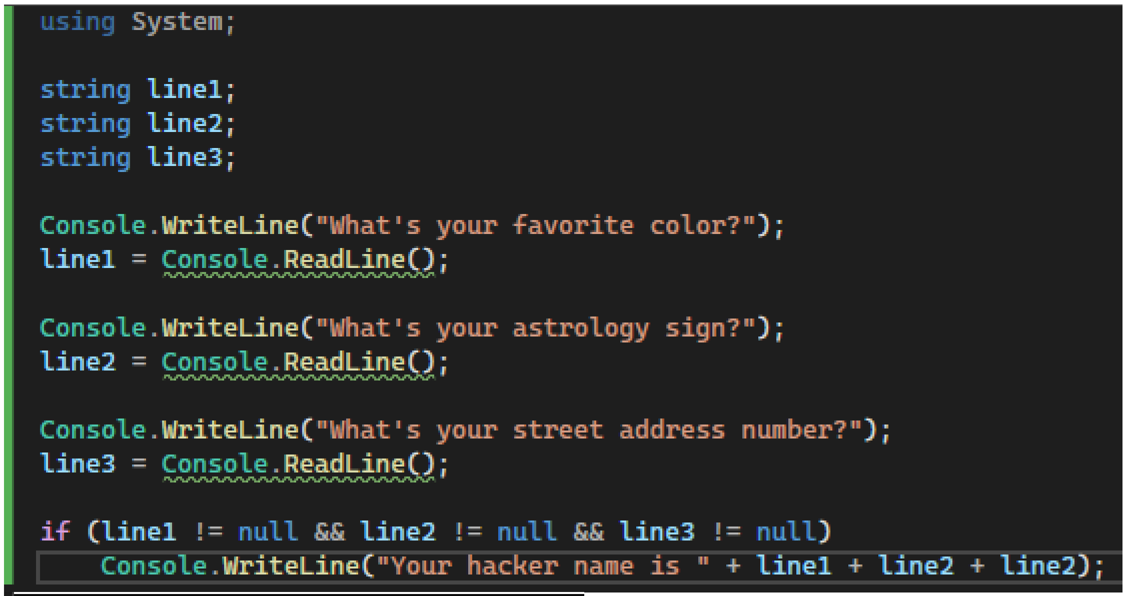
Unmanaged resource is used to describe something not directly under the control of the garbage collector.

4. What’s the purpose of Garbage Collector in .NET?

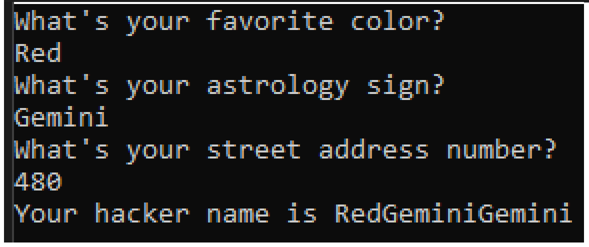
The garbage collector manages the allocation and release of memory for an application.

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."



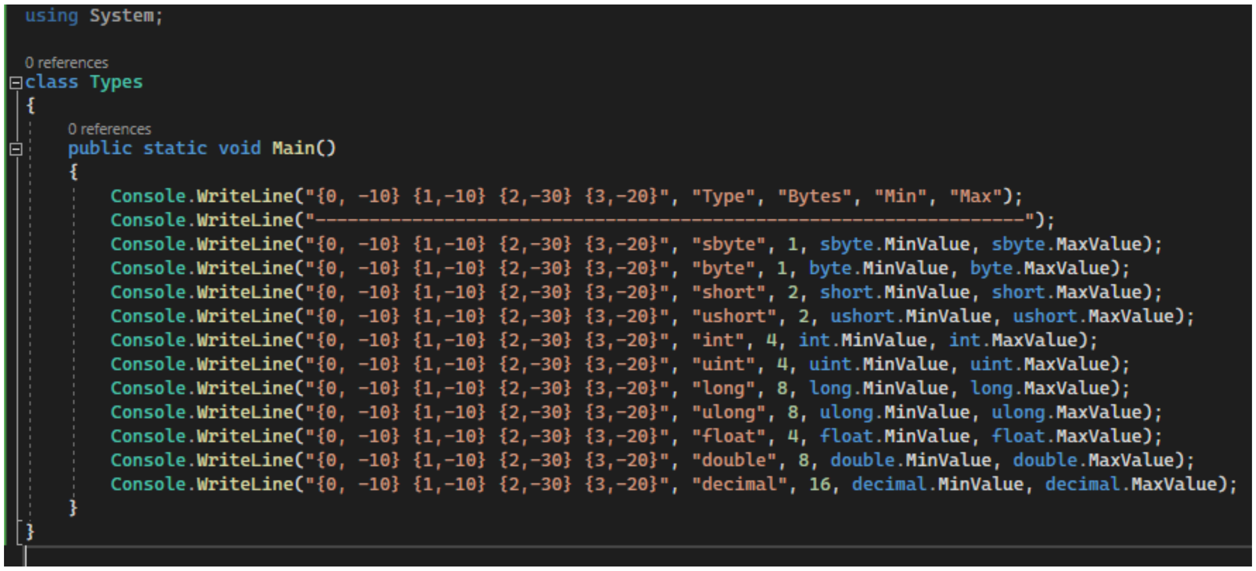
```
using System;

string line1;
string line2;
string line3;

Console.WriteLine("What's your favorite color?");
line1 = Console.ReadLine();

Console.WriteLine("What's your astrology sign?");
line2 = Console.ReadLine();

Console.WriteLine("What's your street address number?");
line3 = Console.ReadLine();

if (line1 != null && line2 != null && line3 != null)
    Console.WriteLine("Your hacker name is " + line1 + line2 + line2);
```



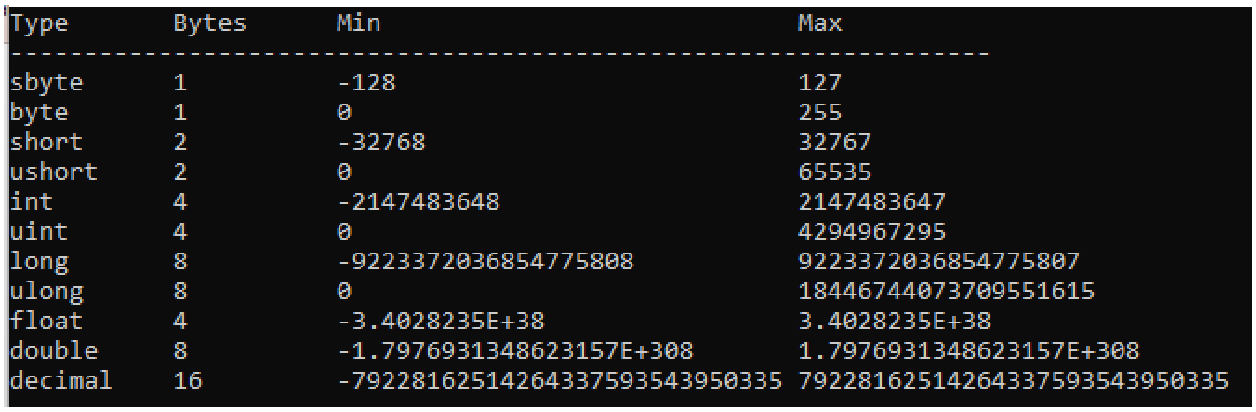
```
What's your favorite color?
Red
What's your astrology sign?
Gemini
What's your street address number?
480
Your hacker name is RedGeminiGemini
```

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.

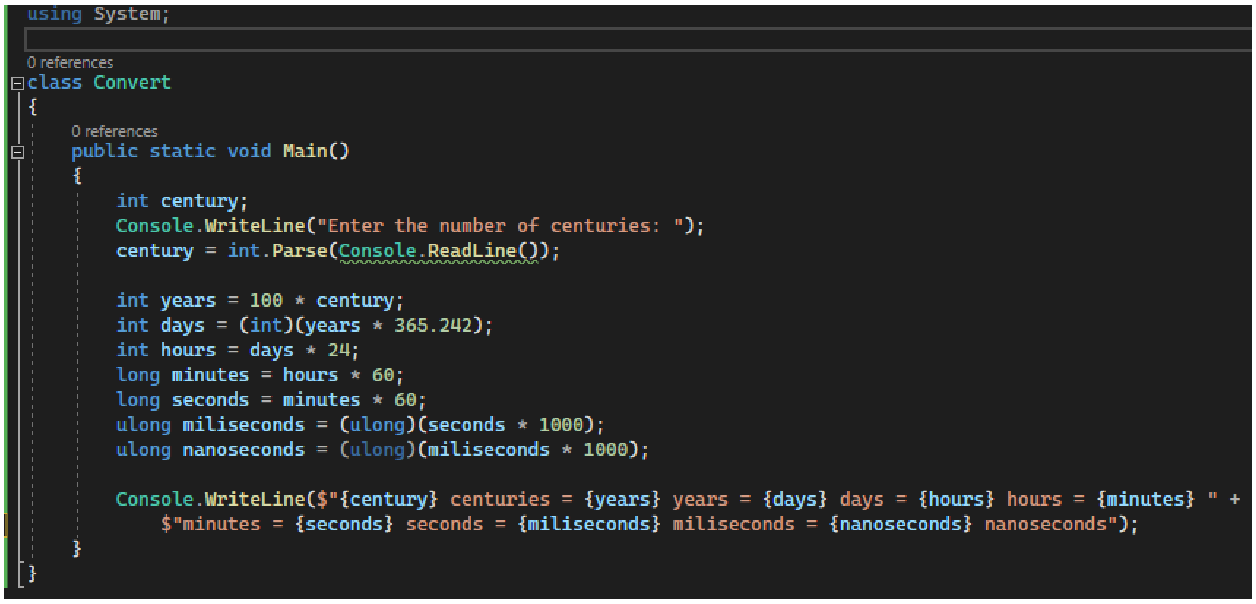


```
using System;

0 references
class Types
{
    0 references
    public static void Main()
    {
        Console.WriteLine("{0, -10} {1,-10} {2,-30} {3,-20}", "Type", "Bytes", "Min", "Max");
        Console.WriteLine("-----------------------------------------------------------------");
        Console.WriteLine("{0, -10} {1,-10} {2,-30} {3,-20}", "sbyte", 1, sbyte.MinValue, sbyte.MaxValue);
        Console.WriteLine("{0, -10} {1,-10} {2,-30} {3,-20}", "byte", 1, byte.MinValue, byte.MaxValue);
        Console.WriteLine("{0, -10} {1,-10} {2,-30} {3,-20}", "short", 2, short.MinValue, short.MaxValue);
        Console.WriteLine("{0, -10} {1,-10} {2,-30} {3,-20}", "ushort", 2, ushort.MinValue, ushort.MaxValue);
        Console.WriteLine("{0, -10} {1,-10} {2,-30} {3,-20}", "int", 4, int.MinValue, int.MaxValue);
        Console.WriteLine("{0, -10} {1,-10} {2,-30} {3,-20}", "uint", 4, uint.MinValue, uint.MaxValue);
        Console.WriteLine("{0, -10} {1,-10} {2,-30} {3,-20}", "long", 8, long.MinValue, long.MaxValue);
        Console.WriteLine("{0, -10} {1,-10} {2,-30} {3,-20}", "ulong", 8, ulong.MinValue, ulong.MaxValue);
        Console.WriteLine("{0, -10} {1,-10} {2,-30} {3,-20}", "float", 4, float.MinValue, float.MaxValue);
        Console.WriteLine("{0, -10} {1,-10} {2,-30} {3,-20}", "double", 8, double.MinValue, double.MaxValue);
        Console.WriteLine("{0, -10} {1,-10} {2,-30} {3,-20}", "decimal", 16, decimal.MinValue, decimal.MaxValue);
    }
}
```

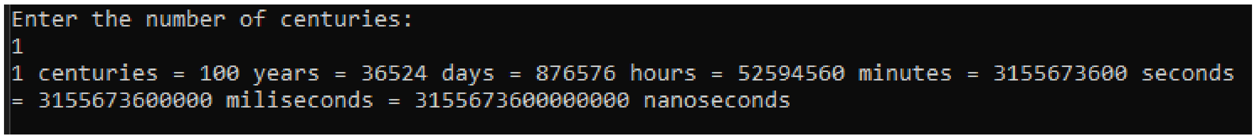


```
Type       Bytes      Min                            Max
-----------------------------------------------------------------
sbyte      1          -128                           127
byte       1          0                              255
short      2          -32768                         32767
ushort     2          0                              65535
int        4          -2147483648                    2147483647
uint       4          0                              4294967295
long       8          -9223372036854775808           9223372036854775807
ulong      8          0                              18446744073709551615
float      4          -3.4028235E+38                 3.4028235E+38
double     8          -1.7976931348623157E+308       1.7976931348623157E+308
decimal    16         -79228162514264337593543950335 79228162514264337593543950335
```

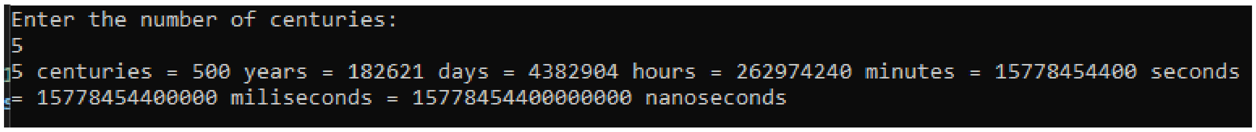
2. Write program to enter an integer number of centuries and convert it to years, days, hours, minutes, seconds, milliseconds, microseconds, nanoseconds. Use an appropriate data type for every data conversion. Beware of overflows!



```
using System;

0 references
class Convert
{
    0 references
    public static void Main()
    {
        int century;
        Console.WriteLine("Enter the number of centuries: ");
        century = int.Parse(Console.ReadLine());

        int years = 100 * century;
        int days = (int)(years * 365.242);
        int hours = days * 24;
        long minutes = hours * 60;
        long seconds = minutes * 60;
        ulong miliseconds = (ulong)(seconds * 1000);
        ulong nanoseconds = (ulong)(miliseconds * 1000);

        Console.WriteLine($"{century} centuries = {years} years = {days} days = {hours} hours = {minutes} " +
            $"minutes = {seconds} seconds = {miliseconds} miliseconds = {nanoseconds} nanoseconds");
    }
}
```



```
Enter the number of centuries:
1
1 centuries = 100 years = 36524 days = 876576 hours = 52594560 minutes = 3155673600 seconds
= 3155673600000 miliseconds = 3155673600000000 nanoseconds
```



```
Enter the number of centuries:
5
5 centuries = 500 years = 182621 days = 4382904 hours = 262974240 minutes = 15778454400 seconds
= 15778454400000 miliseconds = 15778454400000000 nanoseconds
```

Controlling Flow and Converting Types

Test your knowledge

1. What happens when you divide an int variable by 0?
Will get a DivideByZeroException.

2. What happens when you divide a double variable by 0?
Will get the result of infinity.

3. What happens when you overflow an int variable, that is, set it to a value beyond its range?
Will get the OverflowException.

4. What is the difference between x = y++; and x = ++y;?

The first x's value will be increased by 1 after y assign to x. The second x's value will increase by 1 before y assign to x. If y =1, the first x = 1, the second x = 2.

5. What is the difference between break, continue, and return when used inside a loop statement?
Break will terminate the closest enclosing iteration statement. Continue will start a new iteration of the closest enclosing iteration statement. Return will terminate execution of the function in which it appears and returns control and the function's result.

6. What are the three parts of a for statement and which of them are required?
a. Initialization; b. Condition; c. Increment/decrement

7. What is the difference between the = and == operators?
'=' is used to assign the value to the variables. '==' is the operator that checks whether the two operands are equal or not. If so, it returns true. Otherwise it returns false.

8. Does the following statement compile? for (; true;) ;
Yes. It will create an infinite loop.

9. What does the underscore _ represent in a switch expression?
The underscore '_' replaces the default keyword to signify that it should match anything if reached.

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

Practice loops and operators

1. FizzBuzzis 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:
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?

The max-value of byte type variable is 255. This value will always less than max. So, after compile, it will get an infinite loop.

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

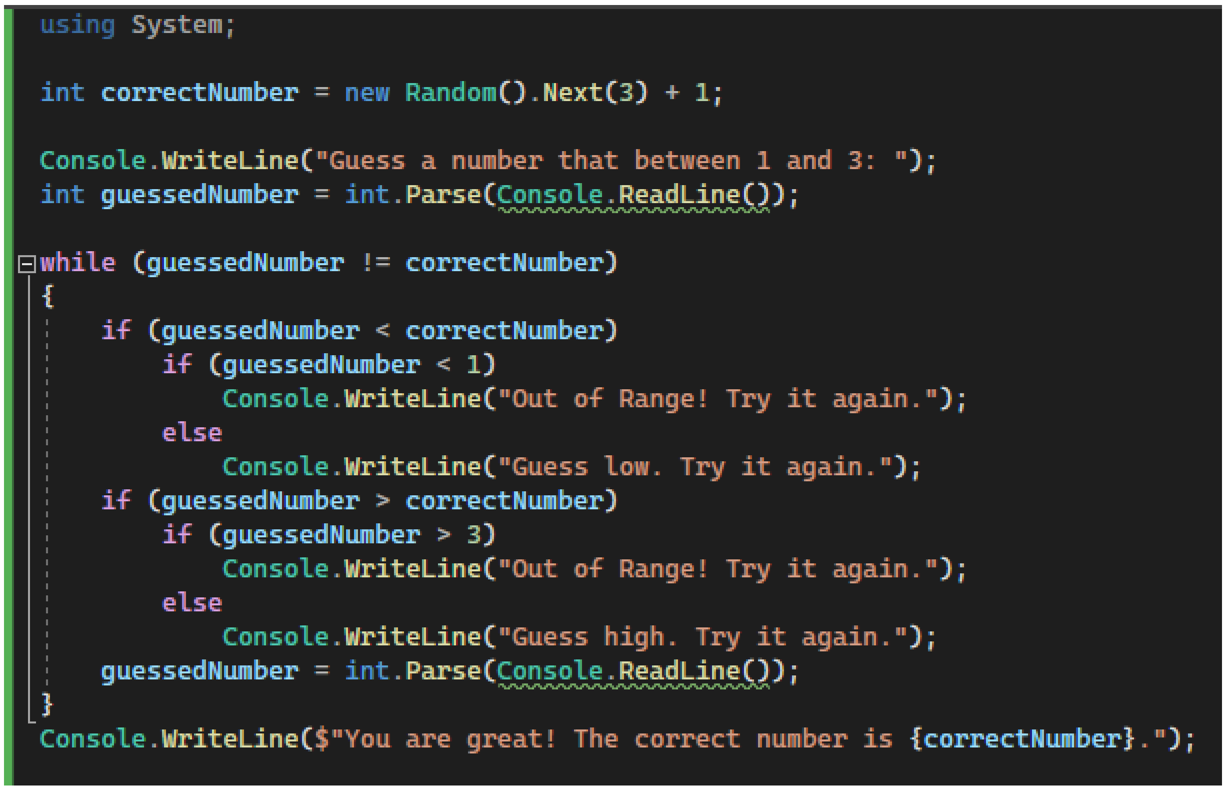
Add an 'if' statement to check: if i > 255, throw a warning exception.

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

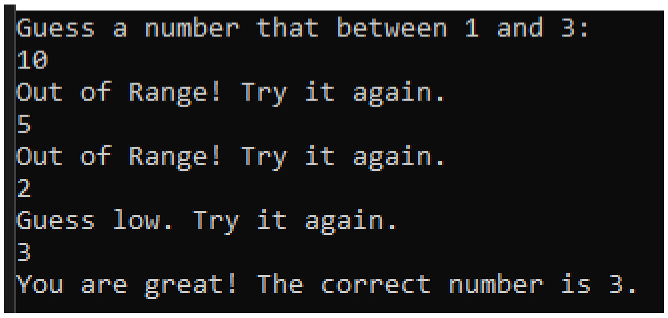
int correctNumber = new Random().Next(3) + 1;
Write a program that generates a random number between 1 and 3 and asks the user to guess what the number is. Tell the user if they guess low, high, or get the correct answer. Also, tell the user if their answer is outside of the range of numbers that are 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.



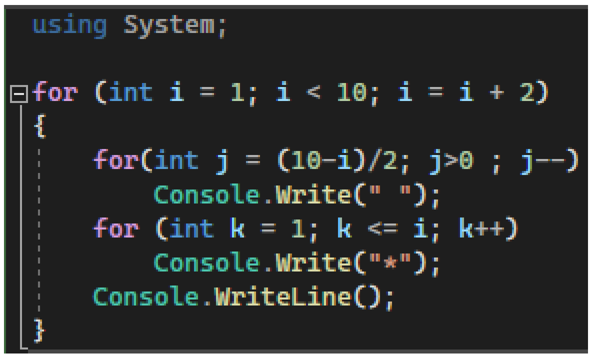
```
using System;

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

Console.WriteLine("Guess a number that between 1 and 3: ");
int guessedNumber = int.Parse(Console.ReadLine());

while (guessedNumber != correctNumber)
{
    if (guessedNumber < correctNumber)
        if (guessedNumber < 1)
            Console.WriteLine("Out of Range! Try it again.");
        else
            Console.WriteLine("Guess low. Try it again.");
    if (guessedNumber > correctNumber)
        if (guessedNumber > 3)
            Console.WriteLine("Out of Range! Try it again.");
        else
            Console.WriteLine("Guess high. Try it again.");
    guessedNumber = int.Parse(Console.ReadLine());
}
Console.WriteLine($"You are great! The correct number is {correctNumber}.");
```



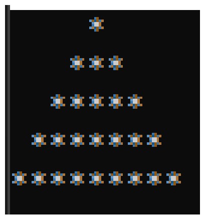
```
Guess a number that between 1 and 3:
10
Out of Range! Try it again.
5
Out of Range! Try it again.
2
Guess low. Try it again.
3
You are great! The correct number is 3.
```

3. 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 print the correct number of spaces, while the second inner loop prints out the correct number of stars.

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

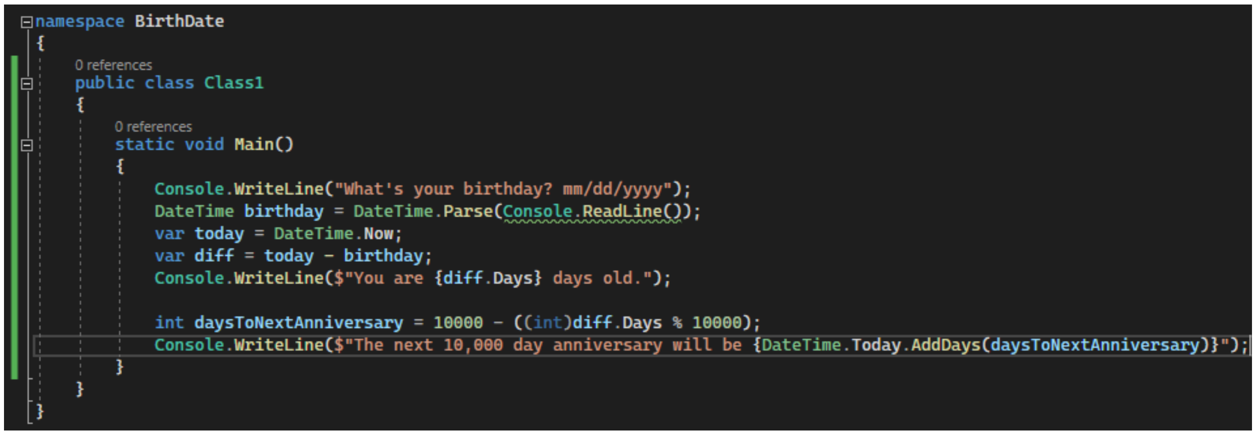


```
using System;

for (int i = 1; i < 10; i = i + 2)
{
    for(int j = (10-i)/2; j>0 ; j--)
        Console.Write(" ");
    for (int k = 1; k <= i; k++)
        Console.Write("*");
    Console.WriteLine();
}
```

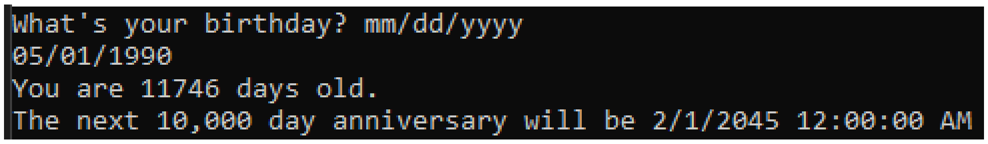


```
    *
   ***
  *****
 *******
*********
```

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); .

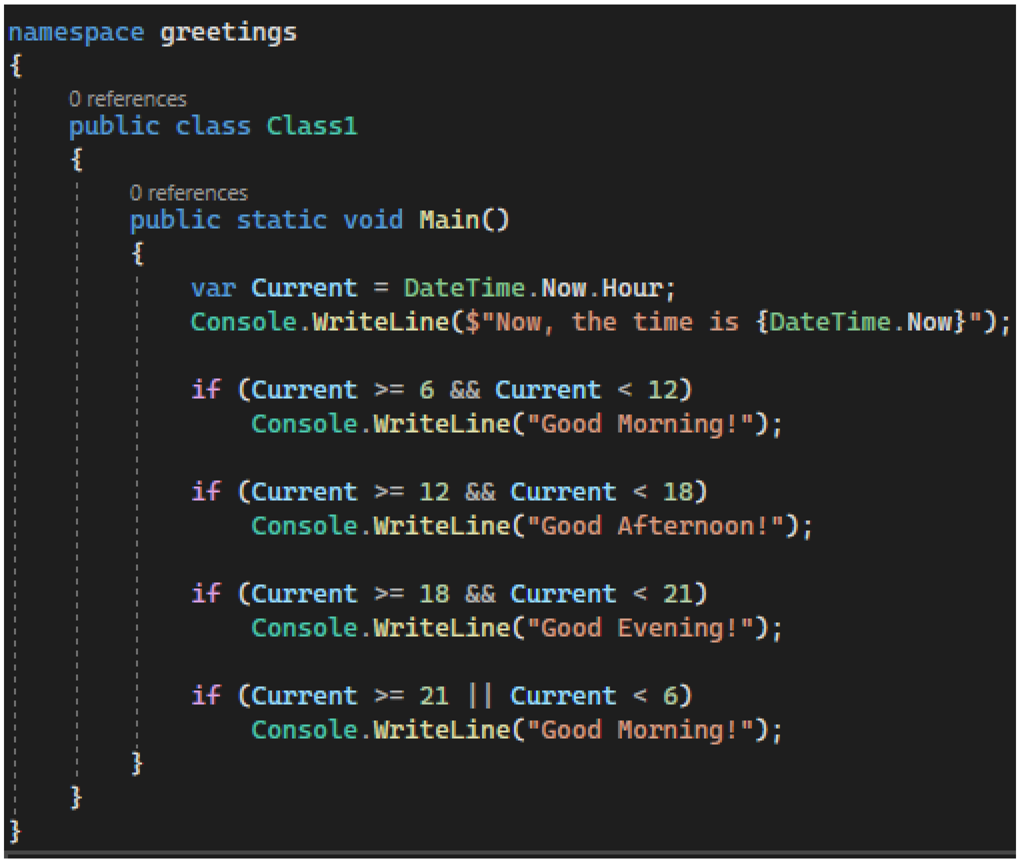


```
namespace BirthDate
{
    0 references
    public class Class1
    {
        0 references
        static void Main()
        {
            Console.WriteLine("What's your birthday? mm/dd/yyyy");
            DateTime birthday = DateTime.Parse(Console.ReadLine());
            var today = DateTime.Now;
            var diff = today - birthday;
            Console.WriteLine($"You are {diff.Days} days old.");

            int daysToNextAnniversary = 10000 - ((int)diff.Days % 10000);
            Console.WriteLine($"The next 10,000 day anniversary will be {DateTime.Today.AddDays(daysToNextAnniversary)}");
        }
    }
}
```

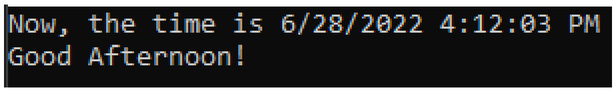


```
What's your birthday? mm/dd/yyyy
05/01/1990
You are 11746 days old.
The next 10,000 day anniversary will be 2/1/2045 12:00:00 AM
```

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).



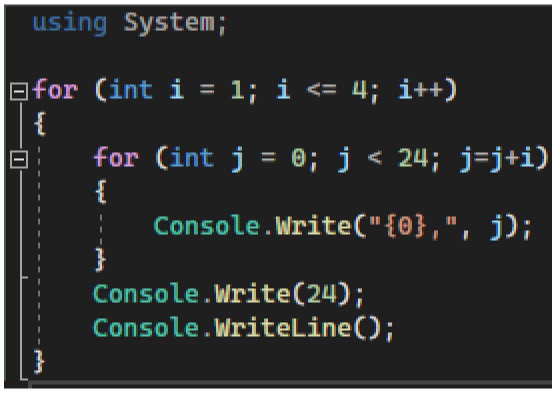
```
namespace greetings
{
    0 references
    public class Class1
    {
        0 references
        public static void Main()
        {
            var Current = DateTime.Now.Hour;
            Console.WriteLine($"Now, the time is {DateTime.Now}");

            if (Current >= 6 && Current < 12)
                Console.WriteLine("Good Morning!");

            if (Current >= 12 && Current < 18)
                Console.WriteLine("Good Afternoon!");

            if (Current >= 18 && Current < 21)
                Console.WriteLine("Good Evening!");

            if (Current >= 21 || Current < 6)
                Console.WriteLine("Good Morning!");
        }
    }
}
```



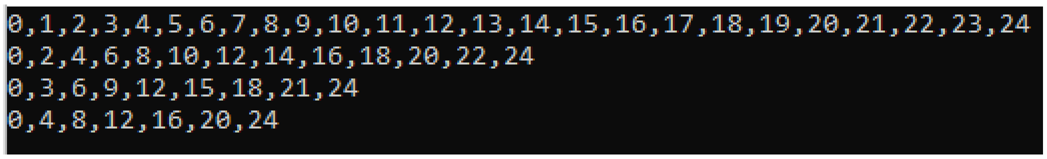
```
Now, the time is 6/28/2022 4:12:03 PM
Good Afternoon!
```

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. You 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:

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



```
using System;

for (int i = 1; i <= 4; i++)
{
    for (int j = 0; j < 24; j=j+i)
    {
        Console.Write("{0},", j);
    }
    Console.Write(24);
    Console.WriteLine();
}
```



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