Brandon Girard

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: uint

A person’s height: float

A person’s age: byte

A person’s gender (Male, Female, Prefer Not To Answer): char

A person’s salary: decimal

A book’s ISBN: ulong

A book’s price: decimal

A book’s shipping weight: float

A country’s population: uint

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 differences between value type and reference type variables?**

Value types are located on the stack, directly contains their data, has its own copy of data, and any operation on it will not affect another. Reference types are located on the heap, store the memory location of the data, can have more than one variable reference the same object, and operations on it can affect another.

**What is boxing and unboxing?**

Boxing is when a value type is converted to a reference and unboxing is vice versa.

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

Everything that’s executed under the common language runtime is considered a managed resource, while everything else is considered unmanaged.

**4.Whats the purpose of Garbage Collector in .NET?**

It's a background process that cleans unused managed resources.

**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.

Console.WriteLine($"sbyte - size:{sizeof(sbyte)} min:{sbyte.MinValue} max:{sbyte.MaxValue}");

Console.WriteLine($"byte - size:{sizeof(byte)} min:{byte.MinValue} max:{byte.MaxValue}");

Console.WriteLine($"short - size:{sizeof(short)} min:{short.MinValue} max:{short.MaxValue}");

Console.WriteLine($"ushort - size:{sizeof(ushort)} min:{ushort.MinValue} max:{ushort.MaxValue}");

Console.WriteLine($"int - size:{sizeof(int)} min:{int.MinValue} max:{int.MaxValue}");

Console.WriteLine($"uint - size:{sizeof(uint)} min:{uint.MinValue} max:{uint.MaxValue}");

Console.WriteLine($"long - size:{sizeof(long)} min:{long.MinValue} max:{long.MaxValue}");

Console.WriteLine($"ulong - size:{sizeof(ulong)} min:{ulong.MinValue} max:{ulong.MaxValue}");

Console.WriteLine($"float - size:{sizeof(float)} min:{float.MinValue} max:{float.MaxValue}");

Console.WriteLine($"double - size:{sizeof(double)} min:{double.MinValue} max:{double.MaxValue}");

Console.WriteLine($"decimal - size:{sizeof(decimal)} min:{decimal.MinValue} max:{decimal.MaxValue}");

**Composite Formatting to learn how to align text in a console application**

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

const int yearsMult = 100, daysMult = 365, hoursMult = 24, minutesMult = 60, secondsMult = 60,

millisecondsMult = 1000, microsecondsMult = 1000, nanosecondsMult = 1000;

ulong years = 0, days = 0, hours = 0, minutes = 0, seconds = 0,

milliseconds = 0, microseconds = 0, nanoseconds = 0;

int centuries = int.Parse(Console.ReadLine());

years = (ulong)(centuries \* yearsMult);

days = (ulong)(years \* daysMult);

hours = days \* hoursMult;

minutes = hours \* minutesMult;

seconds = minutes \* secondsMult;

milliseconds = seconds \* millisecondsMult;

microseconds = milliseconds \* microsecondsMult;

nanoseconds = microseconds \* nanosecondsMult;

Console.WriteLine($"{centuries} centuries = {years} years = {days} days = {hours} hours = " +

$"{minutes} minutes = {seconds} seconds = {milliseconds} milliseconds = {microseconds} microseconds = {nanoseconds} nanoseconds");

**Explore following topics**

**C# Keywords**

**Main() and command-line arguments:**

Main is a reserved word and indicates the entry point to the program. The command line arguments are received when the cs file containing the Main function is executed in the command with additional values.

**Types (C# Programming Guide):**

There are value types which directly hold a value and reference types which hold the memory address for a value.

**Statements, Expressions, and Operators:**

Statements are the actions that a program takes, expressions are combinations of operands and operators that can be evaluated to a single value, and operators are symbols used to perform operations on operands.

**Strings (C# Programming Guide):**

Built in type of object that contains an array of characters.

**Nullable Types (C# Programming Guide):**

A nullable type allows null to be assigned to value types.

**Nullable reference types:**

A reference that can be assigned null

**Controlling Flow and Converting Types**

**Test your Knowledge**

**1.What happens when you divide an int variable by 0?** A DivideByZeroException will be thrown.

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

Results in infinity.

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

The value wraps back starting at the min integer value -2,147,483,648.

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

For y++, the value y is incremented after it’s assigned to x. For ++y, the value y is incremented then assigned to x.

**5.What is the difference between break, continue, and return when used inside a loop statement?**

A break stops the current scoped loop and continues with the rest of the code after the loop. Continue, skips the current iteration of the loop. Return cancels the current loop and function, while also returning a value.

**6.What are the three parts of a for statement and which of them are required?**

There’s the loop variable, the condition, and the expression that modifies the loop variable after each loop cycle. None are required.

**7.What is the difference between the = and == operators?**

= assigns a value to a variable, while == compares values.

**8.Does the following statement compile? for ( ; true; ) ;**

Yes.

**9.What does the underscore \_ represent in a switch expression?**

\_ represents the default case.

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

**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**

int number = 1;

bool running = true;

Console.WriteLine("Starting at, 1 input a divisible number. " +

"\nIf the number is divisible by 3, enter fizz. " +

"\nIf the number is divisible by 5, enter buzz. " +

"\nIf the number is divisible by both 3 and 5, enter fizzbuzz.");

while (running)

{

if(number > 100) break;

Console.WriteLine(number + " ?");

string input = Console.ReadLine();

bool isFizz = IsDivisible(3, number);

bool isBuzz = IsDivisible(5, number);

bool isFizzbuzz = (isFizz && isBuzz);

if(!isFizz && !isBuzz && !isFizzbuzz)

{

int inputNum = 0;

try

{

inputNum = int.Parse(input);

}

catch(FormatException ex)

{

input = "";

continue;

}

if (!IsDivisible(inputNum, number))

continue;

}

else

{

if ((isFizzbuzz && !input.Equals("fizzbuzz")) ||

(isFizz && !input.Equals("fizz")) ||

(isBuzz && !input.Equals("buzz")))

continue;

}

number++;

}

bool IsDivisible(int a, int b)

{

if((b == 1 && a != 1) || (b != 1 && a == 1)) return false;

return (a % b) == 0;

}

**2.What will happen if this code executes?**

An error will be thrown due to WriteLine() not being defined.

Int max = 500;

for

(byte i = 0; i <max;i++)

{

WriteLine(i);

}

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

The program crashes and an error with description “The name 'WriteLine' does not exist in the current context” is thrown

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

You could wrap the code in a try catch block that will catch a NotImplementedException.

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

bool running = true;

Console.WriteLine("Guess a number between 1 and 3: ");

while (running) {

int input = int.Parse(Console.ReadLine());

if(input != correctNumber)

{

if(input < 1 || input > 3) Console.WriteLine("Your guess was out of bounds. Try again: ");

else if (input < correctNumber) Console.WriteLine("Your guess was too low. Try again: ");

else if (input > correctNumber) Console.WriteLine("Your guess was too high. Try again: ");

input = 0;

continue;

}

Console.WriteLine("You guessed correct!");

running = false;

}