OBJECTIVES

- Identify and explain the fundamental concepts and components of .NET.
- Read and write code that uses variables.
- Declare a variable and assign a value to a variable.
- Know and use industry acceptable naming conventions for variables.
- Choose the appropriate primitive data type to represent different kinds of data in a program.
- Utilize arithmetic operators to form mathematical expressions.
- Explain the concept of data conversion (or casting), when it occurs, why it's used.
- Explain the purpose and use of literal suffixes and why they should be used.
- Code and test a simple Java or .NET program using an Integrated Development Environment (IDE).
- Perform simple tasks in an IDE such as:
    - Organizing code into projects.
    - Understand feedback on syntax errors.
    - Utilize the "IntelliSense" feature of an IDE to assist in code development.

Fixing Merge Conflict

- **Go to the repo**
- **Open with Code**
- **Select a file with a conflict**
- **Select which version of the changes you want.**
- **Save the file.**
- **Add, commit, push**

FILE Structure of our GIT setup

Walk through File Explorer to today's lecture folder

Cadence of class

Pull in morning for lecture start
Code in student-lecture
I will push LECTURE-FINAL , the video, and other info after that day's class ends
PULL in the LECTURE-FINAL

Follow along with me!

Please speak up if you have any questions or are lost…. Others are too!!!

Open Today's solution

Talk about VS and the different windows

Comments

What is an IDE?

```
/**
 * TERMS:
 *  -IDE: Integrated Development Environment
 *      -We will be using Visual Studio
 *      -where you write your code
 *      -tools to make things easier
 *      -organizes code into SOLUTIONS and PROJECTS
 *  -C#: programming language we will use
 *  -.NET Framework: A collection of pre-written code, a code library
 *      -Includes code for security, user interfaces, networking, etc.
 *      -We will be using .NET CORE FRAMEWORK, which runs on a lot of different OSs(windows, Linux)
 *      -CLR: Common Language Runtime
 *          -part of .NET that compiles and executes our code
 *  -compiler: a program that reads your code and turns it into bytecode (something the computer can understand)
 *      -compilers check for syntax
```

.net Languages: C#, F#, VB

Print out Hello!

```
Console.WriteLine("Hello!");
```

How to run a program?

- Hit enter to close the window because of READLINE()

Problem 1

Create the variable and uncomment the WRITELINE

Variables

```
/* VARIABLES & DATA TYPES
 * -A variable is a storage container with a name (a name you give it)
 * -Variable declarations start with a data type
 * -Variable declarations end with a semi-colon;
 *
 * = is the assignment operator and is used to put a value in a variable
 */
```

storage container paired with a name. A piece of memory with a name….this is name that YOU want to refer to it as
Variables start with a datatype
End with ;

Datatype and variable name combined are called DECLARATION or STATEMENT
Lets the computer know what the variable is and how much memory it needs (Sticky Notes!)

= is assignment.

INT is a datatype

Possible datatypes:

```
   /**
* DATA TYPE   RANGE
*    bool    true or false
*    byte    0 to 255
*    char    U+0000 to U+FFFF                 single letter ('a','b'...)
*    int     -2^31 to 2^31                    whole number (1,2,3...)
*    long     -2^63 to 2^63                   same as int, but bigger!
*    float   -3.4*10^38  to 3.4*10^38         numbers w/ decimals(1.1, 2.55...)
*    double  ±5.0 × 10^−324 to ±1.7 × 10^308          same as float, but bigger!
*    decimal (-7.9×10^28 to 7.9×1028) / (10^0 to 10^28)  like float/double, but much more precise!
*/
```

2 & 3

2: talk about naming conventions
Camelcase variable names… first letter is lowercase
Meaningful
Do not use keywords

3: strings…. Lowercase…. DOUBLE QUOTES!

```
*    string                                   a bunch of CHARs strung together in order
```

4..5..6

SHOW THE DEBUGGER

7..8..9
cw+tab

EXPRESSIONS

*** Expression: An expression is statement of code which can be evaluated to produce a result**
*** -each expression is evaluated separately, even if they are all in the same line**

Expressions are used with Operators

*** Operators: a symbol (like +) that tells the computer to perform something and produce a result**
*** -Addition +**
*** -Subtraction -**
*** -Multiplication ***
*** -Division /**
*** -remainder %**
*** - () groups operands and operators**

10..11..12

String concatenation
appending/prepending one string to another
13
Re-assigning to the same variable
14

+= operator
add/append and assign
15

Explicit conversion from INT to STRING
16

```
string saw = "saw";
int two = 2;
saw += two;
Console.WriteLine(saw);
```

17

```
saw += 0;
Console.WriteLine(saw);
```

18

```
double divide44By22 = 4.4 / 2.2;
Console.WriteLine(divide44By22);
```

19

```
Console.WriteLine(5.4 / 2);
```

20: widening/narrowing

```
double quotient = 5 / 2;
Console.WriteLine(quotient);
```

Computer took an INT, then another INT and made a 3rd INT, then tried to widen it into a DOUBLE

The computer can widen (int to long, float to double) automatically.
The computer has to be told to narrow it (long to int, double to float)

21: if one of the arguments is wider, then the computer uses the wider type
***debug these!!!

```
quotient = 5.0 / 2;
Console.WriteLine(quotient);

//convert the INT to a FLOAT
quotient = (float)5 / 2;
Console.WriteLine(quotient);
```

```
//REMEMBER HOW TO GET PRECISION FROM DIVISION!!

//convert the INT 12 to a double 12.0 and divide by 9
quotient = (double)12 / 9;
Console.WriteLine(quotient);

quotient = 5 - 2.2;
Console.WriteLine(quotient);
```

22: decimal

```
//Decimal is more accurate, thus used for financial numbers
decimal bankBalance = 1234.56M;
```

What is the M?

We tell the computer INTs with 1,2 & 3. We tell the computer double with 1.1 & 2.2
We use the M to tell the computer this is a more accurate DECIMAL and not a double.
We use f to tell the computer something is a FLOAT

```
float aFloat = 123.5f;


//Can also cast
decimal anotherBankBalance = (decimal)123.45;
```

Why use DECIMAL instead of FLOAT or DOUBLE?

```
//DECIMAL is more accurate than DOUBLE or FLOAT
double tenthPlusZeroFive = .1 + .05;
Console.WriteLine(tenthPlusZeroFive);
decimal tenthPlusZeroFiveAsDecimal = .1M + .05M;
Console.WriteLine(tenthPlusZeroFiveAsDecimal);
```

23: modulo

```
//remainder
Console.WriteLine(5%2);
Console.WriteLine(10 % 7);
```

24: overflow

```
int three = 3;
int billionAsInt = 1000000000;          //too big for an INT!!!
Console.WriteLine(three * billionAsInt);

long billionAsLong = 1000000000;
Console.WriteLine(three*billionAsLong);
```

25& 26: boolean

```
doneWithExercises = true;
Console.WriteLine(doneWithExercises);
```

CONST

```
//CONST: indicates a constant which means the value cannot be changed
const int DAYS_IN_WEEK = 7;
Console.WriteLine("Days in a week: " + DAYS_IN_WEEK);
DAYS_IN_WEEK = 4;   //compiler won't let us do this!
```

How to create our own projects?

HelloWorld

Click VS
New Project -> Console App (.NET Core)
Name: HelloWorld
Location: C:\Users\Student\git\benkennedy-c\Module-1\02_Variables_Data_Types\student-lecture

Run the program

Walk through a couple of homework exercises

Show how to run Tests

How to see what is wrong with the test

Use DECIMAL for currency/money. Otherwise use double