# Intro to C# Curriculum

## Content Week 1: Console.WriteLine() and Variables

### Overview

As an introductory to the class, expose students to the Console Application. With this first step into C# students will need to learn the basic operations to use within a Console Application such as **WriteLine()**.

Students will also need to become familiar with the concept of a **Variable**, a named location in memory used to store data. In order to use variables students will need know the different datatypes that we will be associating with variables, **int, long, bool, double and string.** Students should develop an understanding of what each datatype is and how to use them appropriately.

Recommended: In the past, we have always introduced methods as the last topic and we always wished we had more time with them. To gain more exposure it may be worth talking about methods at the most basic level at the start and maybe use simple methods to write messages to the console window. As the class progresses introduce return types and parameters as it becomes appropriate, (recommended lessons to introduce these topics are in later lesson plans)

**Class Objective:**

1. Build a ‘Hello World Program’
   1. Use WriteLine() to write “Hello World” to console window
2. Explain what a variable is
   1. A named location in memory
3. Declare each of the following variables and assign appropriate values to each:

* int (1)
* bool (true/false)
* string (“Random Text”)
* double ( 1.234)

### Pre-Work

Place student IP address on screen and have students log into their VM’s and open visual studio. Students should write the IP address on name card or store in another accessible and secure location. After this lesson, students should be able to login on their own as they enter the room in future classes.

NOTE: worth showing students how to use both monitors when connecting to their VM’s

### Lecture

Lecture/guided lesson with PowerPoints ‘1 Building your first C# program’ and ‘2 Variables & Data Types’

Code along – ‘Hello World!’ program & declaring and assigning variables of different datatypes

### Group Activities

Using Kahoot as an Exit evaluation:

<https://create.kahoot.it/#quiz/1f90d1ad-60b6-4536-a9fd-1987dda96f62> (Datatypes)

### Challenge

Week one challenge work: ShareBase, Intro C# week 1

Create a new project, declare/assigned values to variables and write to console window

### Post-Work

To students: Write any outstanding questions to Piazza

### Suggested Schedule

1. Get students logged into VM’s
2. Get students to open visual studio and create Console Application
3. Show students where project files are saved
4. Introduce Console.WriteLine() with ‘Hello World’ activity
5. Modify ‘Hello World’ activity to use method to write to console window
6. Show other datatypes
7. Start Week 1 Challenge Work
8. Exit Kahoot

### Wrap-Up Message (Piazza)

Items to cover in Piazza message:

1. Define a **Variable**
2. Recap each datatype and give an example of each
3. Recap Console.WriteLine() is how we write to console window
4. Review format of a simple method to write to console window

## Content Week 2: Food for thought and User input

### Overview

Start by talking about why C# is important, the high-level language is human readable, its value in the real world (who uses it? What is it used to create?).

Students should be able to do the following:

1. Students should understand that C# is a high level programing language
2. Students should understand that C# is a specific language; things cannot have more than one meaning.
3. Ask for user input and store the data in appropriate variables.
   1. When need, students should be able to convert the user input to the appropriate data type.

### Pre-Work

Students should finish the week 1 challenges as they enter the classroom

### Lecture

Lecture: PowerPoint ‘3 Food for Thought’

* C# is a high level programing language. It’s designed to be human readable.
  + Instead of typing 10000101010001010, we can write in statements that make sense
* C# is structured, specific. Things CANNOT have more than one meaning.

Guided lesson: PowerPoint ‘4 Reading User Input’ & ‘5 Conditional Statements’

4.

a) ReadLine() returns a string value, this value can be saved to a string variable

b) To read in other types we must convert. Use Convert.ToInt32(<string value>); or bool

5.

a) Construct basic ‘If’ statement (plug in true) that runs a WriteLine();

b) Next, gather user input (how old are you?) and run a conditional statement. If (age > 16) WriteLine(old enough to drive)

c) Introduce else statement

d) Introduce else if statement

Can break into someone like, ask if:

Sun is blue, If/else (we have a problem)

What is the temperature outside? (Temp > 70 wear shorts

Else, wear pants

What kind of animal is that? (dog/cat/rabbit/whatever else students want to add) (Else/if)

### Individual Exercises

Intro C# week 2

Read user input/ If/else logic challenges

### Post-Work

Items to cover in Piazza message:

### Suggested Schedule

1. Get students logged into VM’s (someone won’t have their info written down still)
2. Give students 15 minutes to finish bell work and then review solutions as a class
3. Spent time talking about what C# is and why its important (ppt. 3)
4. Introduce Console.Readline() with simple activity (read input and write it back)
5. Expand on ReadLine activity by storing user input to string variable
6. If time allows: Introduce conditional statements and data types (this is main focus for next week)
7. Work on week 2 Challenges

### Wrap-Up Message (Piazza)

Items to cover in Piazza message:

1. Cover what it means by C# being a high level programing language
2. Review format for reading in user input
3. Cover how we store this information in a variable and use it later

## Week 3: If statements and Boolean operators

### Overview

In week 3 we will introduce the concept of adding logic to applications using **if()** statements. Students will be introduced to the structure of the logic, ‘if something is true (or false) then do this work’. Since if() statements assess some type of **conditional** statement, students will need to learn the simple **Boolean operators, <, >, =, <=, >=**. These conditional statements take a set of values or parameters and equate them to a **true or false** value. Starting conditionals will be if and int == int or string == string.

After students have some exposure with if(), introduce them to **else if()** and the catch-all **else**. Students should develop an understanding of when an if(), else if(), else() logic tree executes the code in each segment (why we skip blocks after we enter one) and why using a serious of if() statements doesn’t work.

Recommended: After students have demonstrated they understand how to use if/elseif logic, introduce them to Switches. We do not expect students to master the use of Switches, but rather expose them to the concept now and during student projects if they have a project that could benefit from the use of a Switch (such as choose your own story or some type of menu) help the student work that into their project.

Ppt: “Conditional Statements”

* Use conditional statements to traverse different logic paths.
  + If()
  + Else if()
  + Else()
* Use simple Boolean operators to check conditional statements
  + <, >, ==
* Introduce Switch statements, Exposure over mastery for this topic

### Pre-Work

Students should complete the provided bell work as they enter the classroom

### Lecture

Guided lesson: PowerPoint ‘5 Conditional Statements’

a) Construct basic ‘If’ statement (plug in true) that runs a WriteLine();

b) Next, gather user input (how old are you?) and run a conditional statement. If (age > 16) WriteLine(old enough to drive)

c) Introduce else statement

d) Introduce else if statement

Can break into something like, ask if:

Sun is blue, If/else (we have a problem : well that’s good)

What is the temperature outside? (Temp > 70 wear shorts

Else, wear pants

What kind of animal is that? (dog/cat/rabbit/whatever else students want to add) (Else/if)

If(dog)

“Must be Moose!”

Else if(cat)

“Is it Grumpy Cat?”

Else if(Monkey)

“It’s Abu!”

1. If time allows, modify whatever type of if/else logic your created to use a switch

### Group Activities

Using Kahoot as an Exit evaluation:

<https://play.kahoot.it/#/k/7bcbcc28-adaf-46d9-847b-0291f528e62a>

### Individual Exercises

Week 3 challenge work: ShareBase, Intro C# week 3

Conditional logic (If/else)

Boolean Operators

### Post-Work

To students: Write any outstanding questions to Piazza

### Suggested Schedule

1. Students enter class and start bell work
2. Review bell work
3. Start lecture on if()
4. Code Along application using if()
5. Give examples of conditional statements being converted to Booleans (use both int and string)
6. Lecture on else if()/else
7. Code Along application using else if()/else
8. Introduce Switch statements
9. Convert in class project to use Switches
10. Start week 3 challenge work

### Wrap-Up Message (Piazza)

Items to cover in Piazza message:

1. Explain what a **Conditional Statement** is
2. Recap the ways to evaluate a conditional statement to Boolean statement (>,<,==)
3. Explain the purpose of If/else branches and how conditional statements are used to evaluate them
4. If covered in class, recap what a switch statement is. Again, switches should be an exposure over mastery topic. Some students will have good projects at the end of the semester to utilize switches

## Week 4: Boolean Operators

### Overview

Last week students were introduced to the basic **Boolean operators**, this week introduce them to the **advanced Boolean operators**. These operators are as followed: **<=, >=, !=, ||, &&**. Students need to understand that these operators still come back as either **true** or **false** to satisfy a conditional statement.

Extra attention will be required for **&&** and **||**. Students need to understand the difference between them, and understand that with the or operator, as soon as one side returns **true** the other side becomes irrelevant since the conditional statement already returned true.

Work with students to create more complex and structured if/ ifelse statements to continue mastery.

### Pre-Work

Students should complete the provided bell work as they enter the classroom

### Lecture

Guided lesson: PowerPoint ‘6 Boolean Operators’

* Use Advanced Boolean operators to check conditional statements
  + <=, >=, !=, ||, &&

Continue to use more complex if/elseif logic to re-enforce these concepts.

### Individual Exercises

Week 4 Challenge work

### Challenge

Week 4 challenge work: ShareBase, Intro C# week

Boolean Operators

### Post-Work

To students: Write any outstanding questions to Piazza

# Intro to C# Curriculum

## Content Week 5: Basic Arithmetic

### Overview

While arithmetic is not a long topic to cover, it is still an important one and most students will not have much difficult understanding this topic. Start by reviewing the basic operators that students would typically encounter, + (addition), - (subtraction), \* (multiplication) and / (division). Once students have seen each introduce them to the modulus operator (%), a few students in the past have found a practical application for this operator during final projects. Once students have used our basic operators to perform simple operations show them how we can condense these lines buy using +=, -=, ++, --, ect..

If students have been introduced to Functions and Methods (which is highly encouraged) this would be a great time to introduce them to Parameters and Return types. A great in class activity would be some type of simple calculator (attached solution under lecture).

If time permits introduce For loops.

### Pre-Work

Students should complete the provided bell work as they enter the classroom

### Lecture

Guided lesson: PowerPoint ‘7 Basic Arithmetic’

* Use arithmetic statements
  + +, -, \*, /
  + +=, -=, \*=, /=
  + ++, --



(If time permits)

Guided Lesson: PowerPoint ‘8 For & While Loops’

* Identify the parts of a for statement:

For (int I = 0 ; I < 10 ; i++)

Int I = 0 --> variable (Temp initialized variable)

I < 10 --> comparison (continue while this condition is true)

I++ --> modifier (increment after every run)

### Individual Exercises

Week 5 Challenge work

### Post-Work

To students: Write any outstanding questions to Piazza

### Suggested Schedule

1. Allow students some time to finish bell work (20 minutes or so)
2. Review bell work and answer any last questions about topics
3. Introduce basic mathematical operators (+, -, \*, /)
4. Modify in class project to use condensed operators where appropriate (=+, -+, ect)
5. Introduce modulus operator (%)
6. Expand on methods with parameters and return type (simple calculator attached)
7. Start Week \* challenge work

### Wrap-Up Message (Piazza)

Items to cover in Piazza message:

1. Address and reaffirm any common questions brought up during bell work review
2. Recap the simple mathematical operators and give a simple use case for each
3. Reaffirm how the modulus operator works, may be new to a few students
4. Explain what a return type is and how we used parameters to expand our understanding of methods
5. Review the structure of a For loop if covered

# Intro to C# Curriculum

## Content Week 6

### Overview

So far each project created has been designed to run once and then exit out, what if we need a process to run multiple times? Time to introduce the students to Loops, For and While to be specific. The first loop to introduce to the students will be the For loop. While most useful in situations where you know specifically how many times something should execute, we can explain its function using a simple project such as print a value and increment. Starting with explaining the 3 components to the loop (variable, comparison, increment), and drive home that the comparison simply translates to a Boolean statement.

Once students have worked with For loops introduce the While loop. Students seemed to have an easier time understanding the functionality of a while loop vs. the For. Enforce while loop translates the conditional statement to either true or false and continues to run as long as it equates to true. Other example to demonstrate is a bad conditional statement can result in an undesired infinite loop. If an infinite loop is desired, set the conditional to true and use a break statement to exit the loop.

If time allows introduce Arrays. Note: foreach loops are covered when talking about collections.

### Pre-Work

Students should complete FizzBuzz bell work as they enter the room

### Lecture

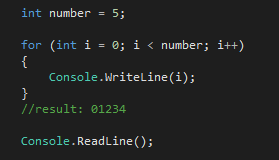
Guided lesson: PowerPoint ‘8 For & While Loops

**For Loop:**

Four major parts of a for loop for (1, 2, 3) { 4 }

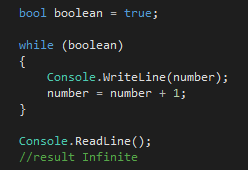
1. Variable – A temporary numeric variable we must assign a value to
2. Comparison – compare against our temp variable (ex. i > 15), as long as it equates to true re-run the loop
3. Modifier – increment our temp value by some amount so we will eventually break out of our loop (ex. i++)
4. While not commonly defined as a key part of our For loop, enforce that whatever exists in this code block will only execute while the comparison returns true. If our condition is never true this code will never be executed

Simple example:

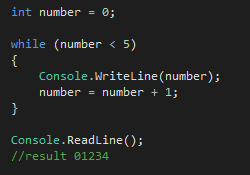


**While Loop:**

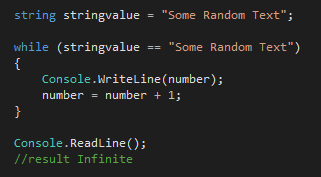
The while loop simply executes a block of code as long as the condition you give it is true. The conditional statement can be a Boolean variable (true/false – can be a Boolean variable) (note infinite loop)



Numeric comparison (variable i > 10)



or even string comparison (variable s == “Some Random Text”) (note infinite loop)



Each of these are translated to either true or false and loop executes as long as it equates to true. The string and Boolean example above create infinite loops, demonstrate why these maybe useful and how to break out using a break; statement.

### Group Activities

Using Kahoot as an Exit evaluation:

<https://create.kahoot.it/details/intro-to-c-up-to-arrays/d6367a98-e228-446e-972f-7b17f8064d6f> (Up to Arrays)

### Individual Exercises

Week \* Challenge work

### Post-Work

To students: Write any outstanding questions to Piazza

### Suggested Schedule

1. work on and then review bell work
2. Introduce **For Loops**
3. Code along example using For loops (For loops will likely take about half the class time)
4. Introduce **While Loops**
5. Code along example for while loops
6. Have students work on challenge work.

### Wrap-Up Message (Piazza)

Items to cover in Piazza message:

1. What a For loop is
2. What are the major components of a for loop
3. What a While loop is
4. What is an infinite loop? And what can we do to break out of it

## Content Week 7: Collections (Arrays & Lists)

### Overview

Collections allow users to store multiple like values within a single object. Two common types of collections will be discussed, Arrays and Lists. Think of arrays has like items in a 1 by x size container, you as the user have the ability to place an object in one of the containers and at any given time be able to return to a specific spot and retrieve or replace an object in a specific slot. In programing, arrays store multiple values of the same data type, so an array can store multiple integers or multiple strings (or any data type for that matter!). The size of the array (or how many items it can hold) must be determined at creation and unfortunately, we are unable to modify its size after creation.

However, that is where Lists become useful. While the syntax is slightly different, Lists have many of the same advantages as Arrays but we can modify how many items the list can hold at any given time. In order to access a specific item in a list you must address it by using its index value.

### Pre-Work

Students should complete the provided bell work as they enter the classroom

### Lecture

Guided lesson:

Array:  
 Declaration:

Structure: dataType[] variablename = new datatype[sizeOfArray];

1. long[] numArray = new long[10]; Each index set to default value for the data type, we don’t know what we want the values to be, but we know we need to hold 10 items.
2. long[] numArray = new long[] { 0, 1, 1, 2, 3, 5, 8 }; we know exactly what values we need to store! Since we are supplying what we want the values to be we do not need to declare the size of the array, it will figure that out on its own!

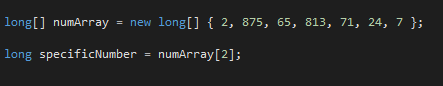
How do we set and access values? Remember, collections are zero bases, so the first item in the array is actually index 0!

Set:

### 

Set index 3 (really the 4th item in the list) equal to 25.

Retrieve:



We have set values to the array during declaration, so get the value of index 2 (really the 3rd item in the array), that value would be 65.

Lists:

Declaration:

Structure: List<string> listName = new List<string>();

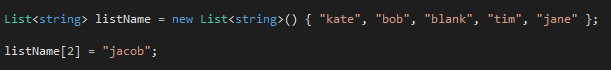
1. List<string> listName = new List<string>();

This is fine! Lists don’t need to be told how big they need to be, we can add and remove items any time we want!

1. List<string> listName = new List<string>() { "random", "text", "goes", "here" }; Like arrays, if we know exactly what values we need we can just add them!

How do we set/add and access values? Remember, collections are zero bases, so the first item in the array is actually index 0!

Set:



To change the value of an already existing slot, just do as you would an array, index 2 (which is really the 3rd item in the collection) now equals “Jacob”

Retrieve: 

We have set values to the array during declaration, so get the value of index 0 (really the 1st item in the array), that value would be “kate”.

### Individual Exercises

Week \* Challenge work

### Post-Work

To students: Write any outstanding questions to Piazza

### Suggested Schedule

1. Work on and review bell work
2. Collections: Array(structure and declaration)
3. Array code along
4. Collections: Lists(structure and declaration):
5. List code along
6. Advantages Lists have over Arrays
7. Foreach loop

### Wrap-Up Message (Piazza)

Items to cover in Piazza message:

1. What are the two ways we can declare an array?
2. How about a List?
3. Arrays cannot grow in size, but a List can
4. Collections start at index 0!

## Final 3 weeks: Project time!

### Show us what your learned

The final 3 weeks are for students to work on individual projects.

The first 2 weeks are for students to create their projects, what specifically they create is up to them! Examples in the past are Hangman, Create your own story, Mad-lib generators, simple databases with menu options, and rock paper scissors! All we ask is students follow these required objectives:

1. Use a loop! Can be For, Foreach or while. Up to them!
2. Use *at least* one method, show us you understand how to use them

By case: If you see a student using a project that would benefit from using a collection, drive them to use one!

Past experiences:  
 It is advised students work alone, in group projects it has been difficult to see who actually did the majority of the work and if anyone was lacking understanding. Working solo will help better advice who should move on to advanced and who should retake

The last week:

Present project to class! Run through the application once to show us what it does. Then have students show and explain what their loop and method do (just so we know they use them)