Console Class, Menus

Topics

- Console Apps
- Arrays
- Memory
- Console Input
- Converting Variables
- Console Output

Console Apps

Console Apps

- A console app is an application that use the console window as its input/output window.
- The app will have a .exe extension.
- The Main method is what the C# runtime will call first when running your app.

How-To

Console

Challenges

App Challenge

Console App How To

- Open Visual Studio
- Click the “Create new project” button
- Select C# in the Language dropdown
- Select Console in the “project type” dropdown
- Select .NET Core
- Click “Next”
- Enter a name for the project (make it meaningful)
- To make zipping up project easier, keep the “Place solution and project in the same directory” as unchecked.

App Challenge #1

- Create a C# Console App
  - Choose .NET Core

LINKS

Creating C# Console App

Slides

App Info

App How-To

Memory

Value Types, Reference Types, Stack, Heap

.NET Data Types

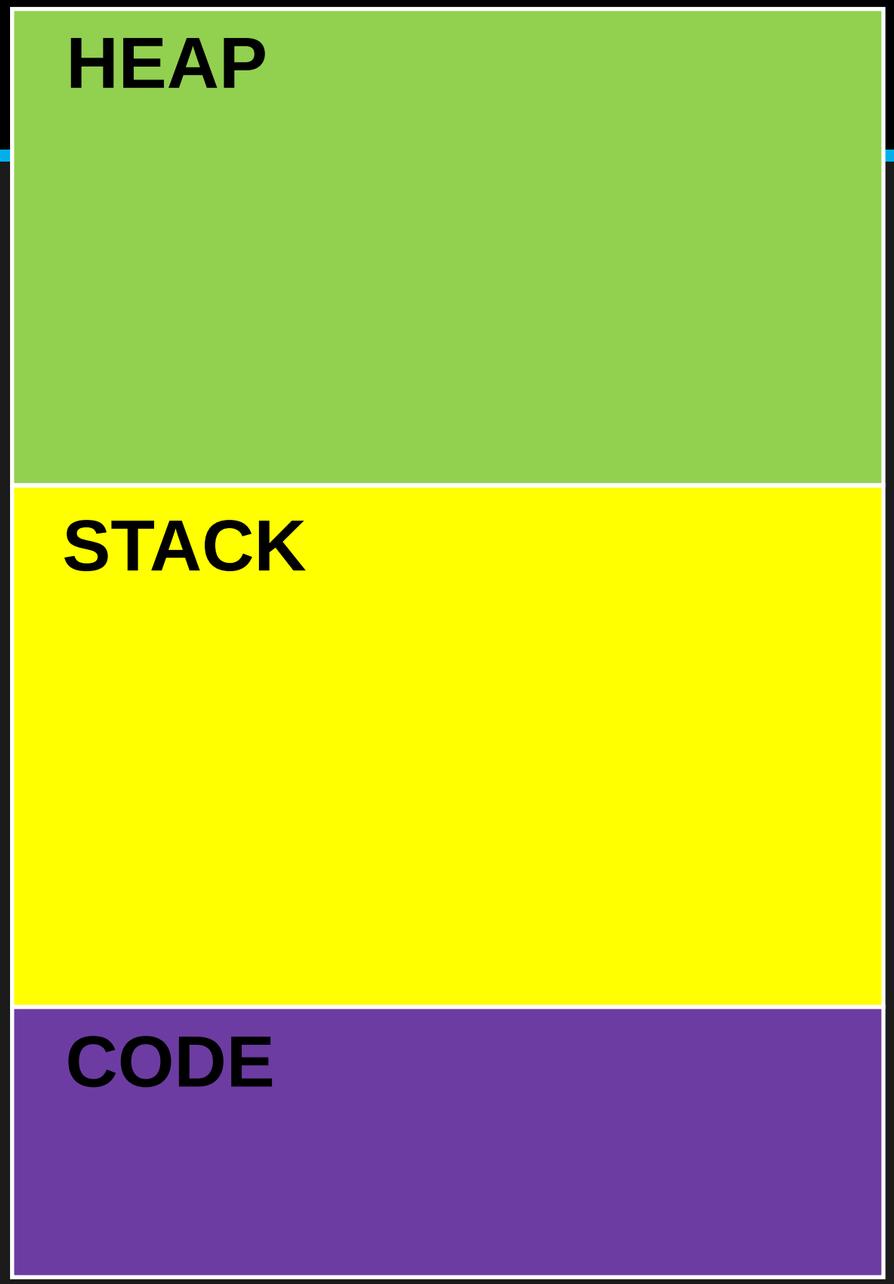
- In .NET, you have two kinds of data types: **value types** and **reference types**.
- **Value Types**
  - Built-in data types (bool, int, float, double…)
    - NOT string though. String is a reference type.
  - User-defined structs
  - Stores its contents in memory on the **stack**
- **Reference Types**
  - Classes and arrays
  - Stores its contents in memory allocated on the **heap**

Size of types

- Variables take up a certain amount of memory depending on the type of the variable
  - **Value Types**:
    - bool: 1 byte
    - char: 2 bytes
    - int: 4 bytes
    - float: 4 bytes
    - double: 8 bytes
  - **Reference Types**. Classes and arrays vary in size based on what is stored in them. However, the **variables** for reference types hold memory addresses so their sizes are the size of a memory location.
    - If a 32-bit build: 4 bytes (or 32 bits)
    - If a 64-bit build: 8 bytes (or 64 bits)

The Basics: Memory

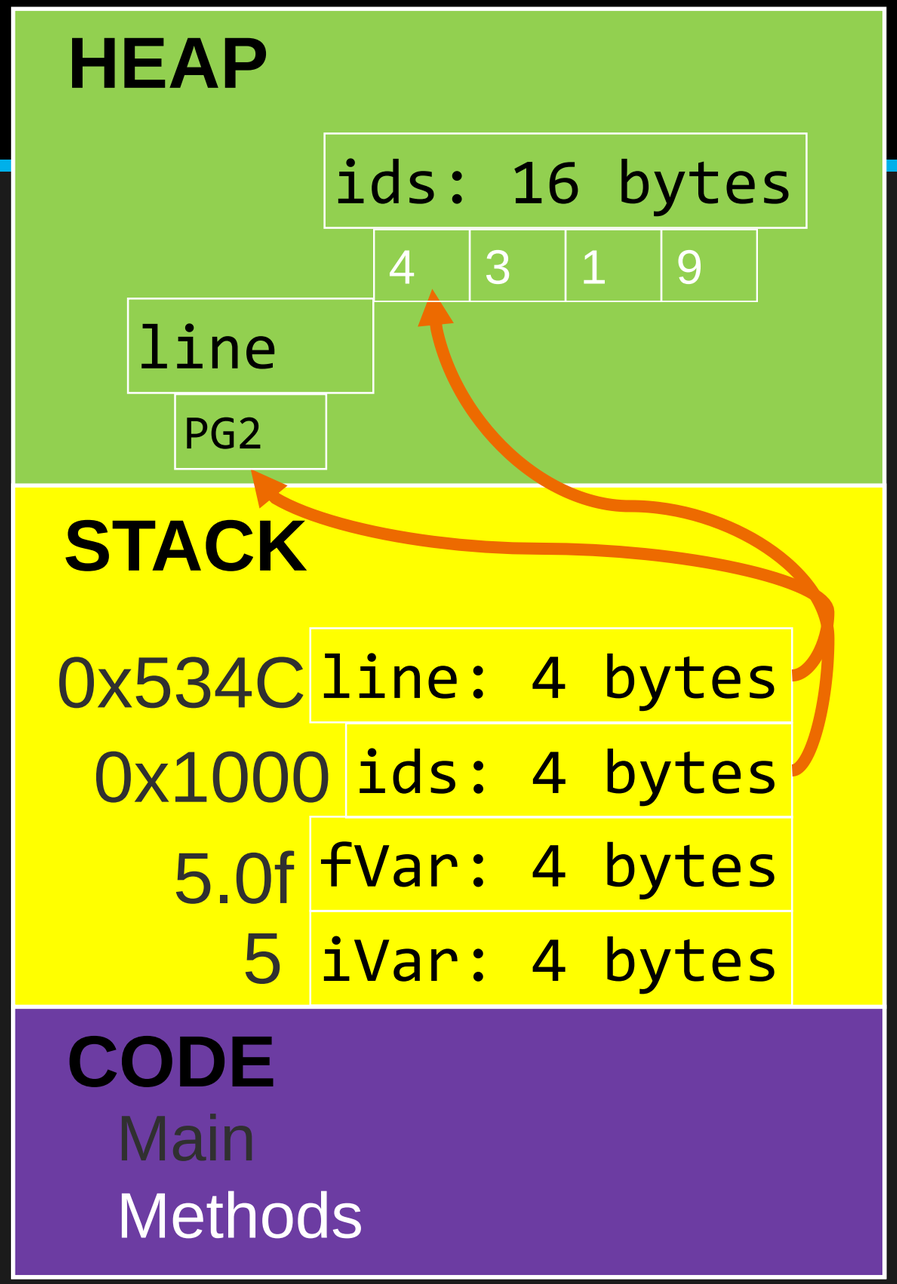
- Memory for **Reference types** (arrays, classes) are allocated on the **Heap** and are *not guaranteed* to be next to each other in memory.
- Memory for **Value types** (int, float, bool, struct, etc) are allocated on the **Stack** *next to each other* in the order that they are declared.
- Your application **code** is loaded by the operating system into the **Code** block of memory.



The Basics: Memory

- Let's create some variables…
- `int iVar = 5;`
- `float fVar = 5.0f;`
- `int[] ids = new int[4] { 4, 3, 1, 9 };`
- `string line = "PG2";`

- Assuming 32-bit, the reference variables (ids, line) take 1 word (or 32 bits or 4 bytes). For 64-bits that would be 8 bytes.



Console Input

Getting User Input

Console Input

- The two most common ways to get user input from the user:
- Console.ReadLine() – gets a string from the user. Returns when the user hits enter.
- Console.ReadKey() – returns on any key press.

Console.ReadLine

Console.ReadLine gets input from the user and returns a string.

Even if the user enters numbers, it is returned as a string.

EXAMPLE:

```
string ageString = Console.ReadLine();
```

Console.ReadKey

- Console.ReadKey gets the next character or function key pressed by the user
- Most of the time you'll use this to wait for any key press.
- It does return the key that was pressed.

EXAMPLE:

```
ConsoleKeyInfo key =  Console.ReadKey();

Console.ReadKey(); //if you don't care about what key is pressed
```

Menu Loop Challenge #1

1. In Main, create an array of strings with the following strings added to the array:
    - “1. Draw Challenge 1”
    - “2. Draw Challenge 2”
    - “3. Draw Challenge 3”
    - “4. Draw Challenge 4”
    - “5. Draw Challenge 5”
    - “6. Exit”
2. Create a do-while loop.
    - In the loop, clear the screen then print the array of strings. Call Console.ReadKey to stop the loop.
3. In the loop after printing the options, get input from the user.

LINKS

Arrays

Console.Clear

Do-while

for

Foreach

Console.ReadKey

Console.ReadLine

Slides

Converting Variables

Type Conversion

- **There are two ways to convert a value between variables of different types:**
  - `Implicit Conversion`
  - `Explicit Conversion`

https://www.tutorialspoint.com/csharp/csharp_type_conversion.htm

Type Conversion

- **Implicit** conversion (the system does it for you)
  - ```
    int num = 2147483647;
    long bigNum = num; //implicitly converts from int to long
    ```

- When does this happen?
  - When the second variable's type is bigger than the first variable's type.
    - long is bigger than int.
    ```

Type Conversion

- **Explicit** conversion (you tell the system the conversion to use)
  - ```
    double x = 1234.7;
    int a;
    a = (int)x; // Cast double to int.
    ```

- When does this happen?
  - When the second variable's type is smaller than the first variable's type.
    - int is smaller than double.
    ```

Parsing Strings

Parsing Strings

- Can we convert a string to another type like an int?

```
string number = “5”;
int num = number; //will this work?
```

- No! The system doesn’t know how to automatically or explicitly convert a string to an int.
- So how can we do this? We parse the string!
- There are 2 ways to parse to an int: int.Parse and int.TryParse

Int.Parse

- Int.Parse will take a string parameter and returns an int

```
string number = “5”;
int num = int.Parse(number); //num = 5
```

- What happens if the string parameter is not a number?

```
string number = “Steve”;
int num = int.Parse(number);
```

- It will throw an exception!
- You need to handle exceptions that are thrown by other code.

Handling Exceptions

- Use a try-catch block to handle exceptions that are thrown by other code.
- try //the code that might throw an exception goes in the try block

```
{
        string number = “Steev”;
        int num = int.Parse(number);
}
```

- catch(Exception ex) //the code that handles the exception is the catch block

```
{
        Console.WriteLine(“OOPS!”);
}
```

Int.TryParse

- Int.TryParse will take a string parameter, an out int parameter and returns a bool

```
string number = “5”;
bool success = int.TryParse(number, out int num);
```

- What happens if the string parameter is not a number?

```
string number = “Steve”;
 bool success = int.TryParse(number , out int num);
```

- It will ***NOT*** throw an exception! It will return false and set num to 0.
- Check the return value to see if it converted correctly.

Int.TryParse

- string number = “Steev”;
- if(int.TryParse(number, out int num))
  {
  …//gets here is TryParse returns true
  }
- else
  {
  …//gets here is TryParse returns false
  }

Menu Loop Challenge #2

1. Convert the input to an integer and store in an int variable.
2. Add a switch statement for the integer with a case statement for each menu option.
3. In each case statement, print the menu option for that case. (HINT: the # - 1 will be the index into the array)
4. Create a bool variable outside of the loop. Default it to true.
5. Use the variable in the condition of the do-while loop.
6. In the exit case of the switch, set the bool to false.

LINKS

int.TryParse

int.Parse

try-catch

Switch

Slides

Input Info

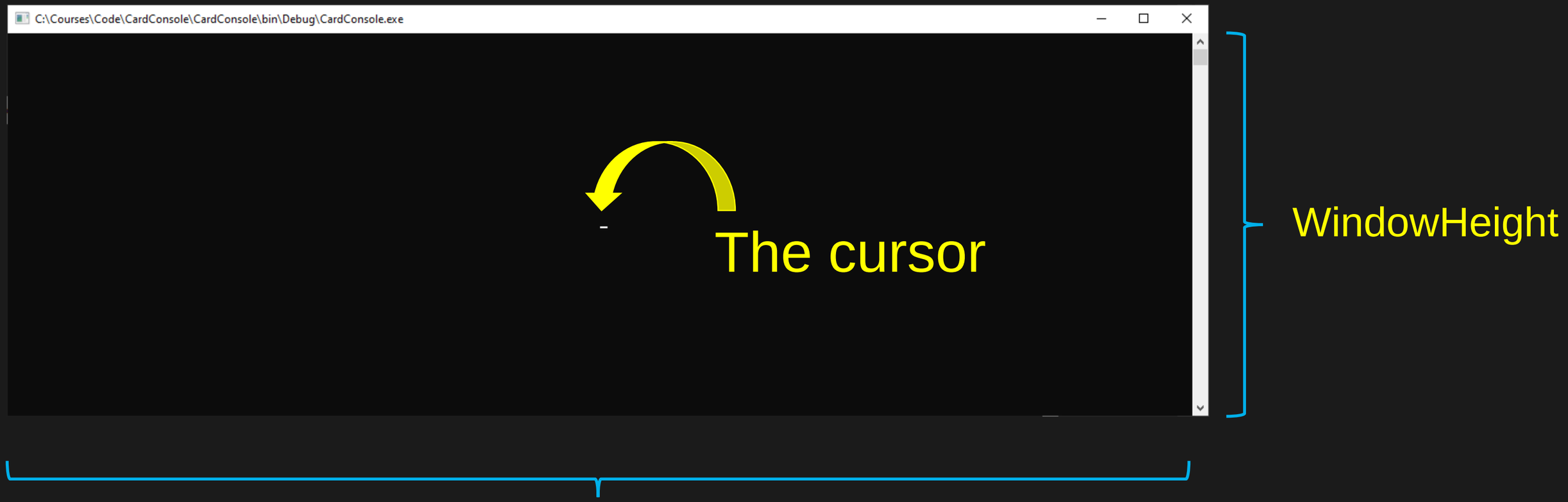
Input How-To

Int.TryParse How-To

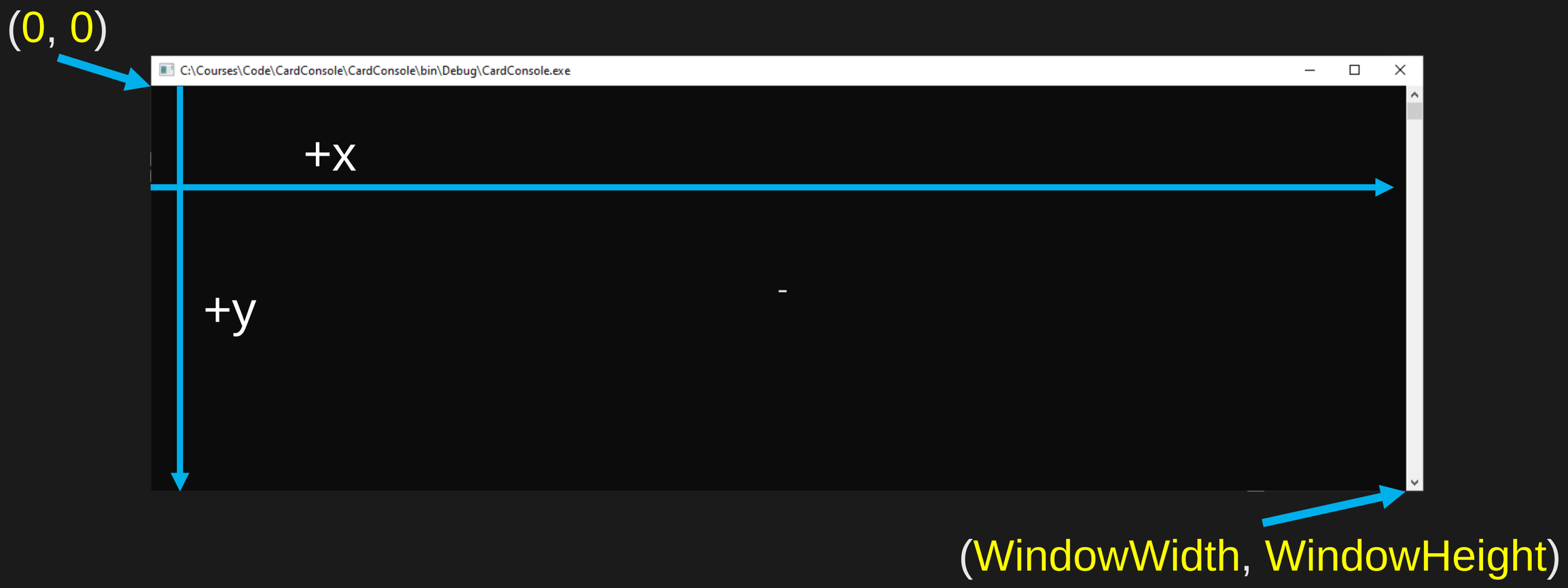
Console Output

Printing to the Screen

The Console Window



Window Coordinates



Cursor Position

- Printing to the console start wherever the cursor is located in the window.

- Console.CursorLeft – allows you to get/set the X position of the cursor
- Console.CursorTop – allows you to get/set the Y position of the cursor
- Console.SetCursorPosition(x,y) – moves the cursor to the x,y values that are passed in

- Console.WindowWidth – tells you the width of the Console window
- Console.WindowHeight – tells you the height of the Console window

Console Output

- Console.Write(string) – prints the string where the cursor is located
- Console.WriteLine(string) – prints the string where the cursor is located then moves the cursor to the beginning of the next line (sets CursorLeft to 0 and increments CursorTop)
- Console.Clear() – clears the console window to the current background color

Console Colors

- Console.ForegroundColor – gets/sets the color of the text printed to the console
- Console.BackgroundColor – gets/sets the background color of the console.
- Console.ResetColor() – resets the foreground and background colors to their defaults.

Draw Challenge #1

- Create a method called DrawChallenge1.
- In the method, clear the screen and draw a green box in the top-left corner of the Console window.
- Call DrawChallenge1 from the appropriate case statement in the menu loop.

LINKS

Console.BackgroundColor

Console.Write

Console.Clear

Console.ReadKey

static

Slides

Output How-To

Colors How-To

Draw Challenge #2

- Create a method called DrawChallenge2.
- In the method, clear the screen and draw a 15 green boxes horizontally in the top-left corner of the Console window.
- Call DrawChallenge2 from the appropriate case statement in the menu loop.

LINKS

For

Console.Write

Slides

Output How-To

Colors How-To

Draw Challenge #3

- Create a method called DrawChallenge3.
- In the method,
  - clear the screen
  - generate a random number that is **less than** the width of the Console window
  - draw the random number of green boxes horizontally in the top-left corner of the Console window.
- Call DrawChallenge3 from the appropriate case statement in the menu loop.

LINKS

Console.WindowWidth

Random

Slides

Cursor Position Info

Draw Challenge #4

- Create a method called DrawChallenge4.
- In the method,
  - clear the screen
  - generate a random number that is **less than** the width of the Console window
  - Generate a random ConsoleColor (hint: the color range is 0 – 15. You'll need to cast a random number.)
  - draw the random number of random color boxes horizontally in the top-left corner of the Console window.
- Call DrawChallenge4 from the appropriate case statement in the menu loop.

LINKS

Explicit casting

Random

Slides

Output How-To

Colors How-To

Draw Challenge #5

- Create a method called DrawChallenge5.
- In the method,
  - Clear the screen
  - Generate a random number that is **less than** the **width** of the Console window
  - Generate a random number that is **less than** the **height** of the Console window
  - Generate a random ConsoleColor (hint: the color range is 0 – 15)
  - Draw the random number of random color boxes horizontally in the random row of the Console window.
- Call DrawChallenge5 from the appropriate case statement in the menu loop.

LINKS

Console.WindowHeight

Console.CursorTop

Console.CursorLeft

Console.SetCursorPosition

Slides

Output How-To

Colors How-To