

Datatypes and Variables

Principles of Computer Programming I

Spring/Fall 20XX

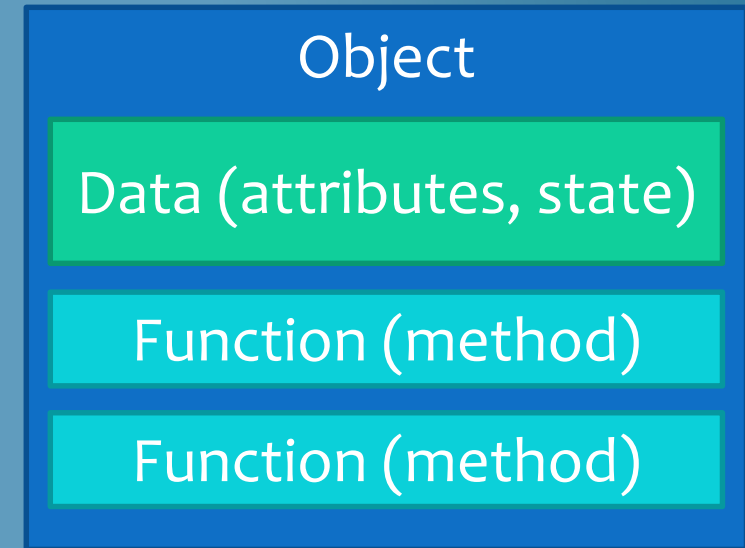
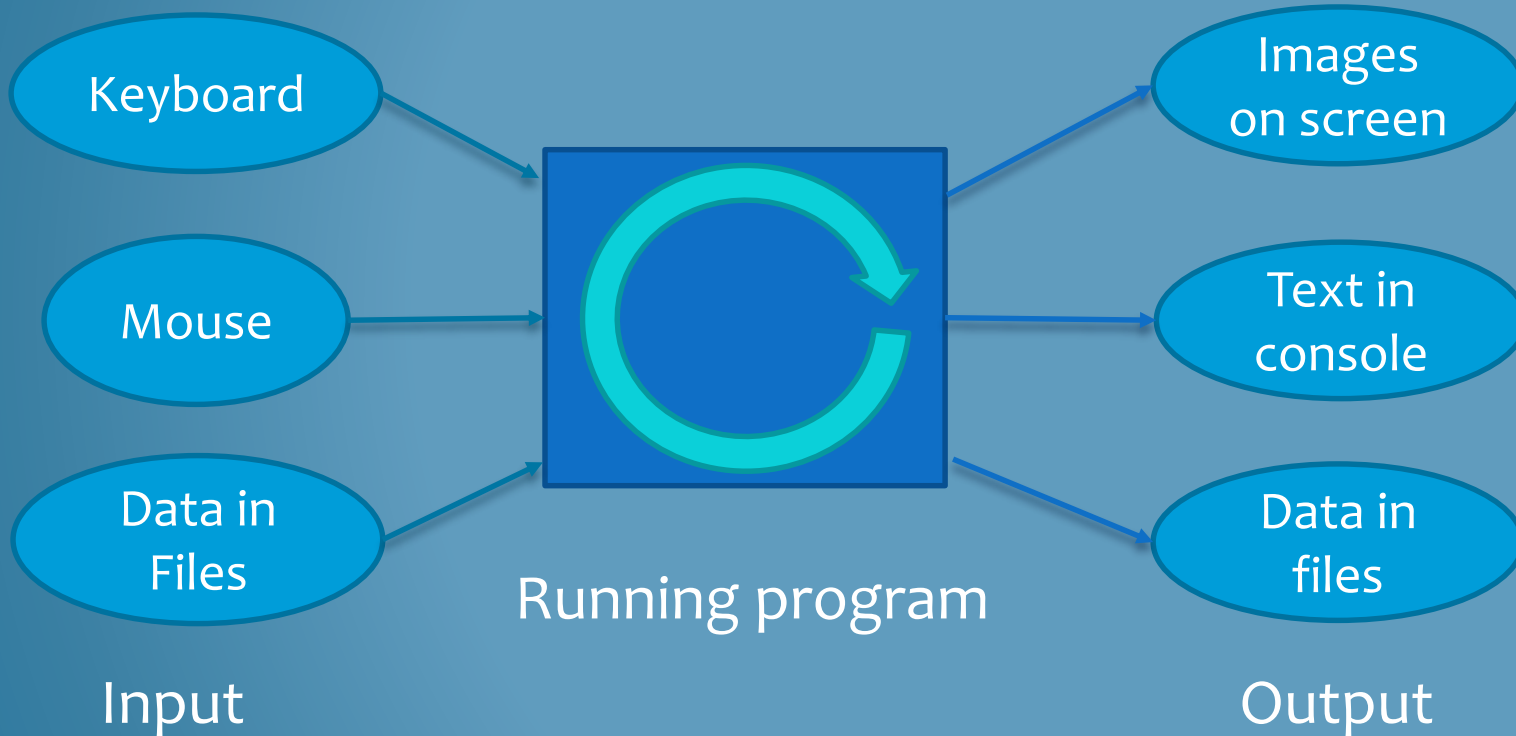


AUGUSTA
UNIVERSITY

Outline

- Basic C# Datatypes
- Literals and Variables
- Basic Variable Operations
 - Declaration
 - Assignment
 - Displaying

Programs Manipulate Data



Data Has Types

- Numbers vs. text (strings)
 - 2 is a number, “two” is text
 - Each letter in “two” is a character; string = string of characters
- Types of numbers
 - Natural numbers (\mathbb{N}): 0, 1, 2, ...
 - Integers (\mathbb{Z}): ... -2, -1, 0, 1, 2, ...
 - Real numbers (\mathbb{R}): 0.5, 1.333333..., -1.4, etc.

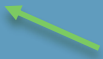
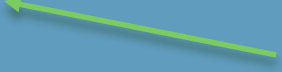
Some C# Datatypes and Keywords

- Text:
 - `string` = a string, like "Hello World!"
 - `char` = a single character, like 'e' or '\t'
- Numbers:
 - `int` = an integer, like -2 or 65536
 - `uint` = an *unsigned* integer, i.e. a natural number, like 42
 - `float` = a “floating-point” number, aka real number, like 3.85
 - `double` = also a real number (3.85), but with “double precision”
 - `decimal` = an “exact decimal” real number with 28 digits of precision

Outline

- Basic C# Datatypes
- **Literals and Variables**
- Basic Variable Operations
 - Declaration
 - Assignment
 - Displaying

Data Literals

- Literal = fixed value in code, “input” given by programmer
- Type can be indicated by syntax
- string literal: "text"
- char literal: 'e'  Single quote
- int literal: 42
- double literal: -4.5
- float literal: -4.5f  f suffix means “this is a float”
- decimal literal: 6.01m

A string literal

```
class Welcome
{
    static void Main()
    {
        Console.WriteLine("Hello World!");
    }
}
```

Variable Basics

- Store data that can **vary** (i.e. change) as program executes

```
class MyFirstVariables
```

```
{
```

```
    static void Main()
```

```
    {
```

```
        int myAge;
```

```
        string myName;
```

```
        myAge = 29;
```

```
        myName = "Edward";
```

```
        Console.WriteLine($"My name is {myName} and I am {myAge} years old");
```

```
    }
```

```
}
```

String interpolation
character

Declare an int variable named myAge

Declare a string variable named myName

Assign myAge a value of 29 using an int literal

Assign myName a value of "Edward" using a string literal

Print the value of myName and myAge inside this string

Outline

- Basic C# Datatypes
- Literals and Variables
- **Basic Variable Operations**
 - Declaration
 - Assignment
 - Displaying

Variable Declaration Syntax

- Specify the **name** of the variable and its **type**

```
int myAge;
```

```
string myName;
```

```
double winChance;
```

Type keyword

Variable name

Semicolon

- Name is an identifier – rules and conventions
 - Only letters and numbers
 - Must be unique
 - Should use camelCase

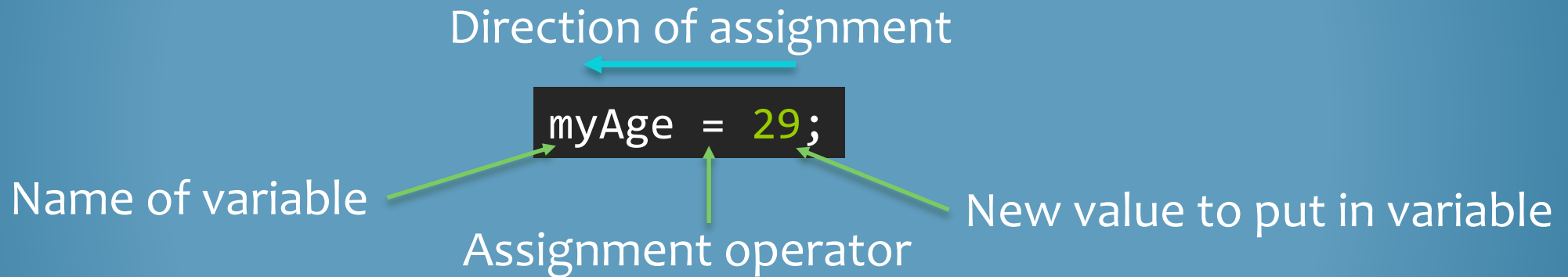
```
int myAge;  
double myAge;
```



Compile error!

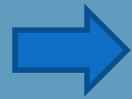
Variable Assignment

- Changes the value of a variable
- = is the “assignment operator”, not “equality” from math



- Value on right side **must** match the type of the variable

```
int myAge;  
myAge = "29";
```



Error! Can't assign a string to an int variable

Initialization Statements

- Combine declaration and first assignment

```
string myName = "Edward";
```

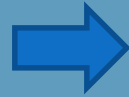
Type of variable

Name of variable

Initial value of variable

- Can only be used once: Variables can only be declared once

```
int myAge = 29;  
int myAge = 30;
```



Error! A variable named myAge already exists

How could we fix this?

```
int myAge = 29;  
myAge = 30;
```

More on Variable Assignment

- Assignment changes value; previous value is gone

```
int myAge;
```

```
myAge = 29;
```

myAge now stores the value 29

```
myAge = 30;
```

myAge now stores the value 30

- Can assign a variable to another variable
 - This takes a “snapshot” of the variable’s current value

```
int a = 12;
```

```
int b = a;
```

a has value 12 here, so b is assigned the value 12

```
a = -5;
```

a now has value -5, but b still has value 12

Displaying Variables

- `Console.WriteLine` can only print text
- To print a variable's value, it must be converted to a string
- **String interpolation:** convert variable data to string, insert it into another string

```
Console.WriteLine($"My name is {myName} and I am {myAge} years old");
```

String interpolation
character

Opening
brace

Variable
name

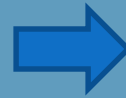
Closing
brace

Fetch value of `myAge`,
convert it to string
"29", insert here

Displaying Variables

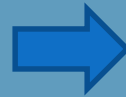
- `Console.WriteLine` will accept just a variable as argument
- Result: Convert this variable's value to a string and print it

```
Console.WriteLine(myAge);
```



29

```
Console.WriteLine($"{myAge}");
```



29

- All built-in C# types have string conversions
 - But when you write your own objects, you will need to write your own string conversions

That's All For Today

- Basic C# Datatypes
- Literals and Variables
- Basic Variable Operations
 - Declaration
 - Assignment
 - Displaying