Datatypes and Variables

Principles of Computer Programming I

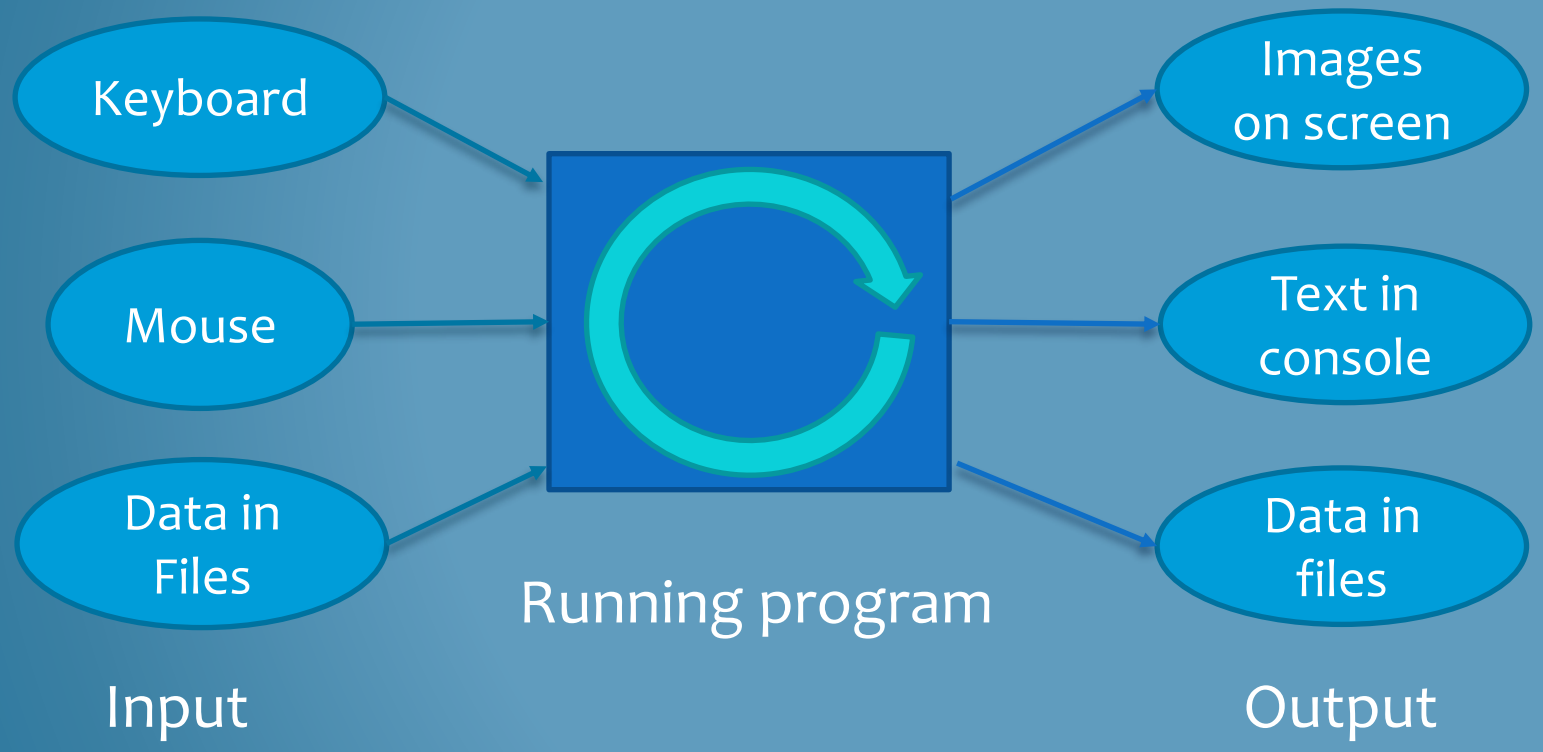
Spring/Fall 20XX

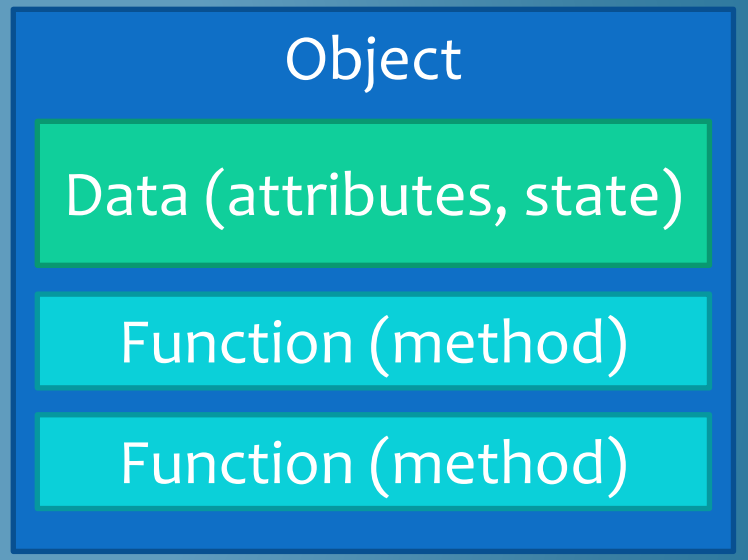
AUGUSTA UNIVERSITY

Outline

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

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
- Format Specifiers

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");
  }
}
```

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

String interpolation character

Print the value of myName and myAge inside this string

Outline

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

Variable Declaration Syntax

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

```
int myAge;
```

```
string myName;
```

```
double winChance;
```

Semicolon

Type keyword

Variable name

- 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

Direction of assignment

```
myAge = 29;
```

Name of variable

Assignment operator

New value to put in variable

- 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 = 30;
```

myAge now stores the value 29

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 = -5;
```

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

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

Displaying Variables

- The console/terminal 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

Outline

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

Output Formatting

- Lots of ways to print numbers, especially fractions
- C#'s default might not be what you want

```
decimal price = 19.99m;
decimal discount = 0.25m;
decimal salePrice = price - discount * price;
Console.WriteLine($"{price} with a discount of " +
      $"{discount} is {salePrice}");
```

```
19.99 with a discount of 0.25 is 14.9925
```

Better String Interpolation

- Can change how numbers are printed with a **format specifier**
- Goes inside braces, after a colon: `{numVar:N}`

Numeric value

Format specifier

Format specifier	Description
N or n	Adds a thousands separator, displays 2 decimal places (by default)
E or e	Uses scientific notation, displays 6 decimal places
C or c	Formats as currency: Adds a currency symbol, adds thousands separator, displays 2 decimal places
P or p	Formats as percentage with 2 decimal places

Format Specifier Example

- Returning to the sale price:

```
decimal price = 19.99m;
decimal discount = 0.25m;
decimal salePrice = price - discount * price;
Console.WriteLine($"{price:C} with a discount of " +
      $"{discount:P} is {salePrice:C}");
```

```
$19.99 with a discount of 25.00% is $14.99
```

Formats with Rounding

- For each format, can change the number of decimal places by adding **precision specifier**: `{numVar:N3}`

Numeric value

Format specifier

Precision specifier

```
double bigNumber = 1537963.666;
decimal discount = 0.1337m;
Console.WriteLine($"{bigNumber:N}");
Console.WriteLine($"{bigNumber:N3}");
Console.WriteLine($"{bigNumber:N1}");
Console.WriteLine($"{discount:P1}");
Console.WriteLine($"{discount:P4}");
```

```
1,537,963.67
1,537,963.666
1,537,963.7
13.4%
13.3700%
```