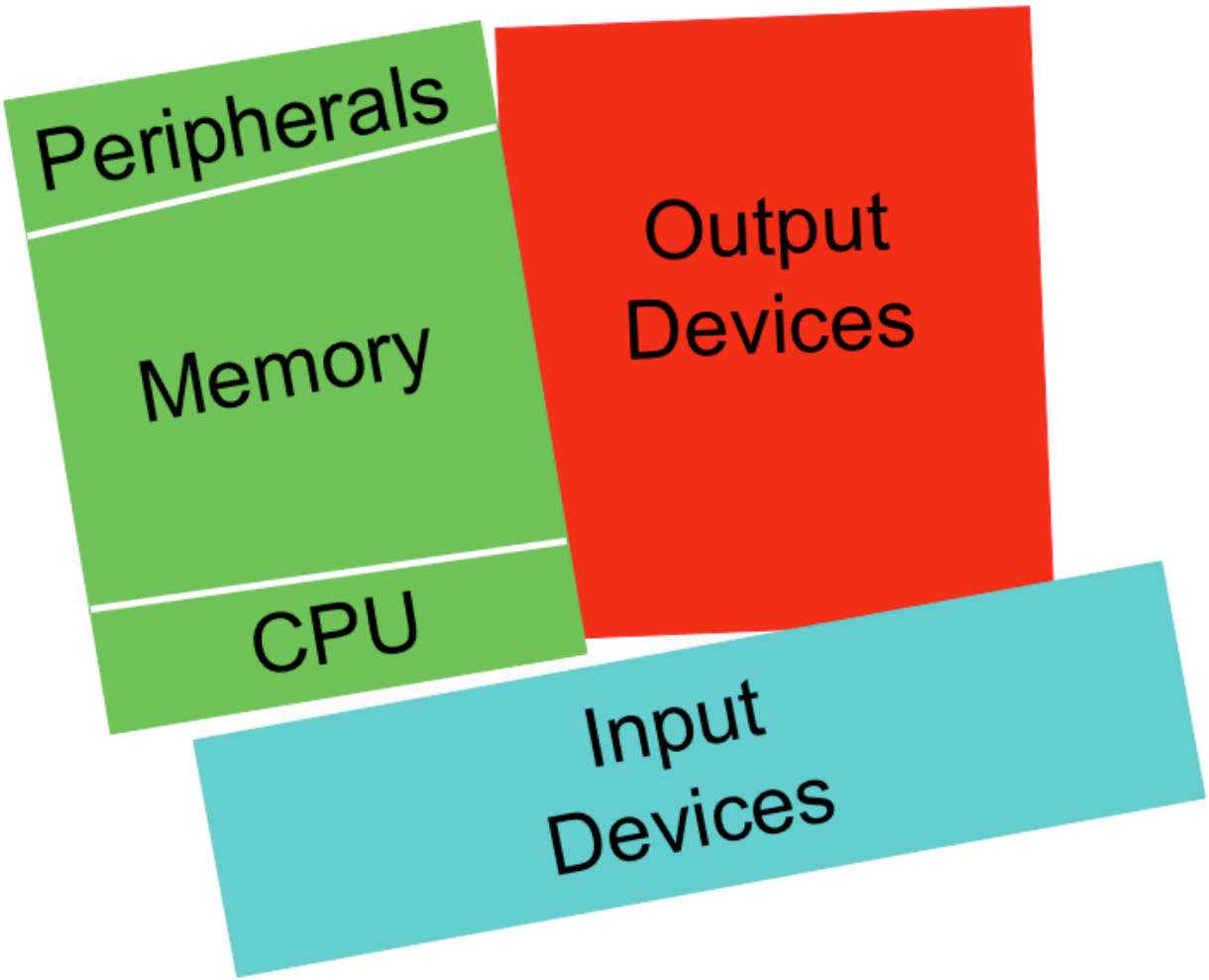
Computers, Variables and Types

Engineering 1D04, Teaching
Session 2

Typical Computer

An Abstract View of Computers

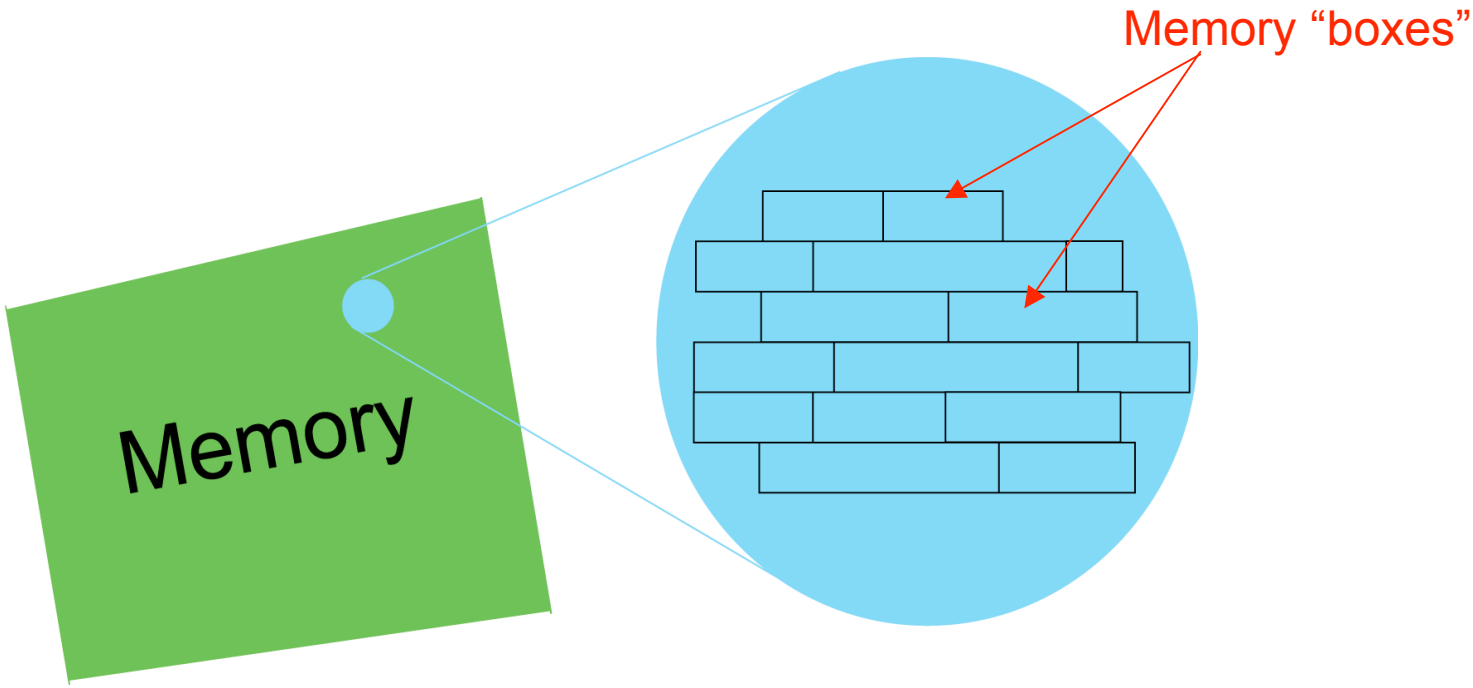
An Abstract View of Computers



Programming Languages

- Algorithms are communicated to the computer using a programming language
- Evolution of programming languages:
  - Machine language: 0100011100011010 0
  - Assembly code: mov, and, push, add, jmp, etc
  - High level languages: FORTRAN, Pascal, Basic, C, C++, Java, Visual C#, Visual Basic, Haskell, Ocaml, …
- Conversion of high level language to machine code is via a compiler

Abstract View of Memory



Introduction to Variables

- When we create algorithms that solve problems, we typically model the problem we are solving - we create an *abstraction* of the real problem and then use properties of that abstract model to get a solution that (hopefully) solves the original problem.
- We often create mathematical variables to represent physical entities.

Mathematical Variables

velocity of car is ***v***
starts at velocity v_0
t represents time
acceleration of car is ***a***

so, at this point in time, what is the car's velocity?

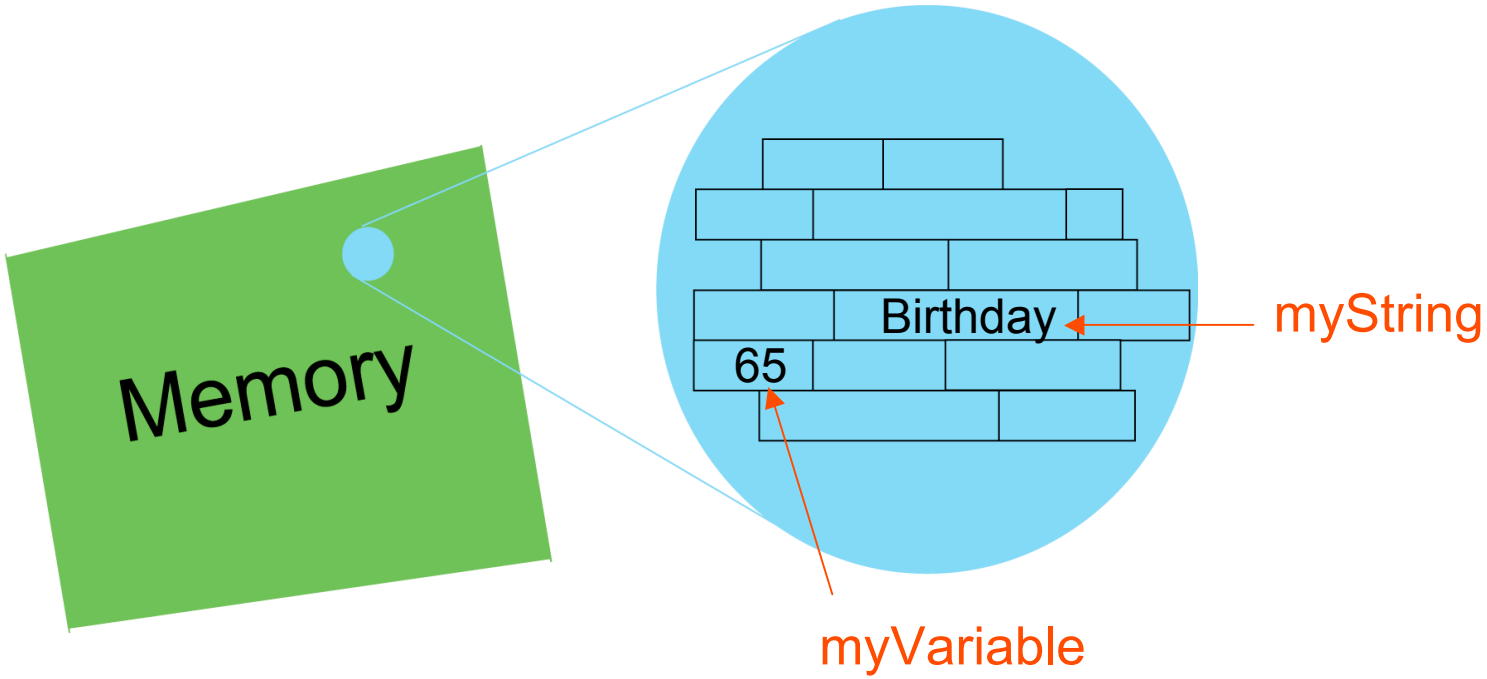
$$v = v_0 + at$$

Introduction to Program Variables

- Computer programs implement algorithms and we need to calculate values that represent physical entities or simply data items. We therefore need to manipulate and store data.

- A program variable is a named memory location that can be used to store data.

Introduction to Program Variables



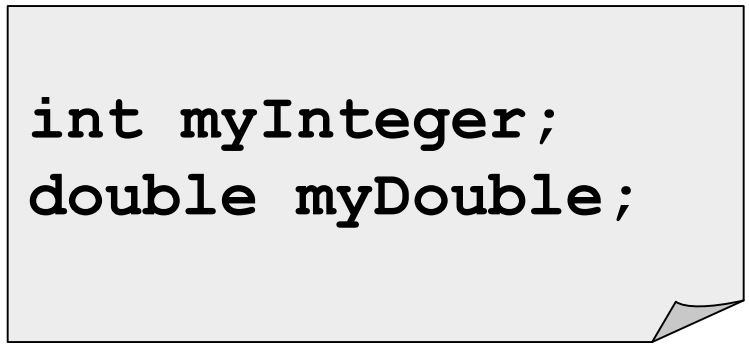
Introduction to Program Variables

- Variables can have different forms depending on the type of data you wish to store in them.
  - Int variables store integer numbers.
  - Float variables (double) store real numbers.
  - Char variables store a single character.
  - String variables store text (multiple characters).
  - Bool variables store true or false.

Introduction to Program Variables

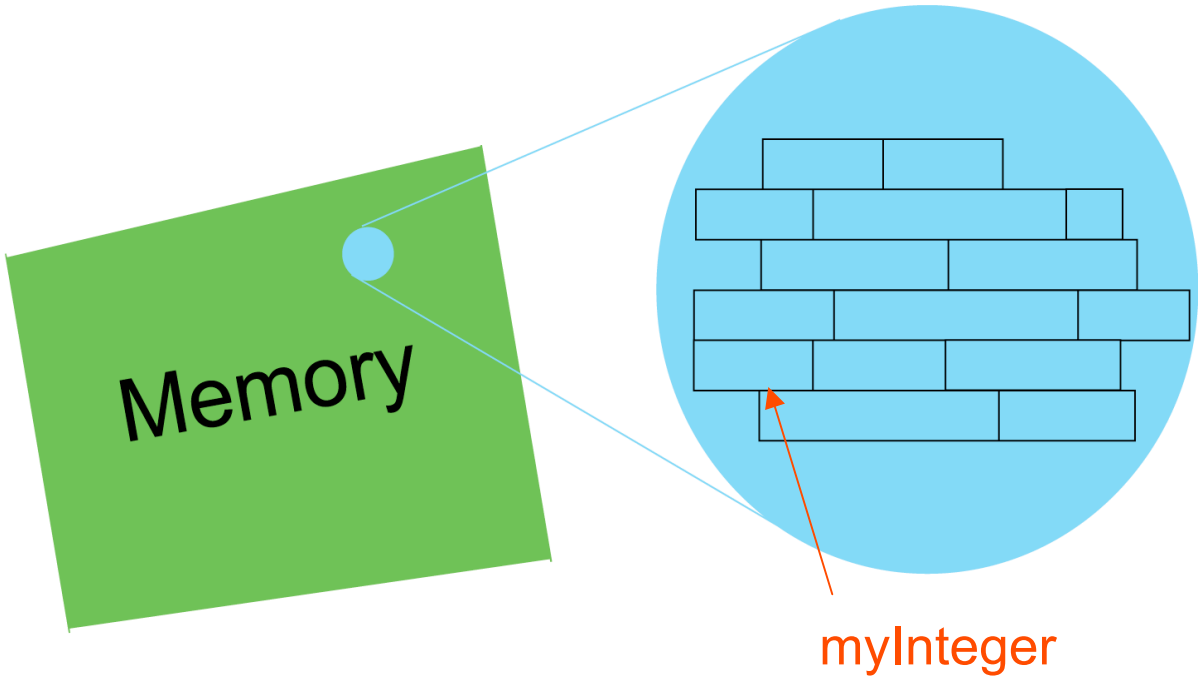
- Variables must be declared before they can be used.

- Declaring a variable assigns a name to a previously unused memory location.



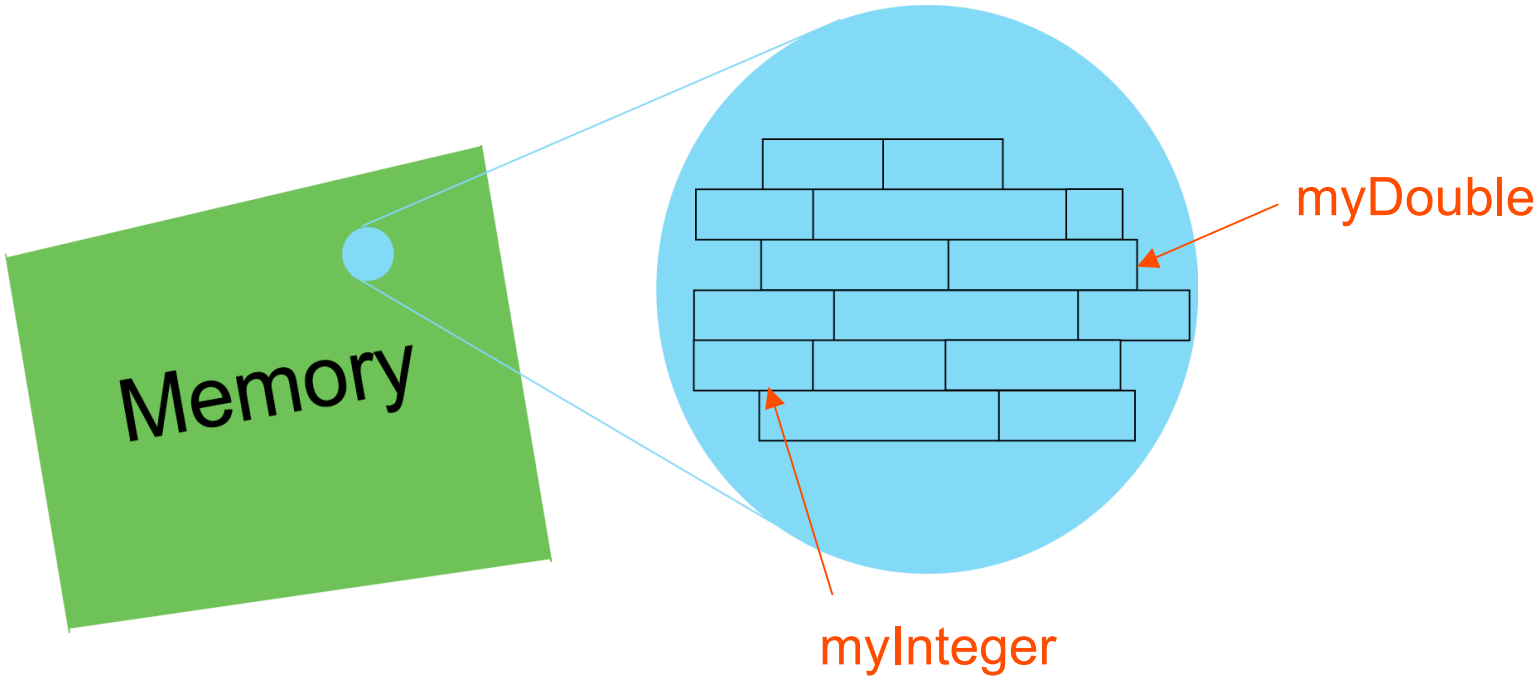
program segments will be shown in these “page” templates

Introduction to Program Variables




```
int myInteger;
double myDouble;
```

Introduction to Program Variables




```
int myInteger;
double myDouble;
```

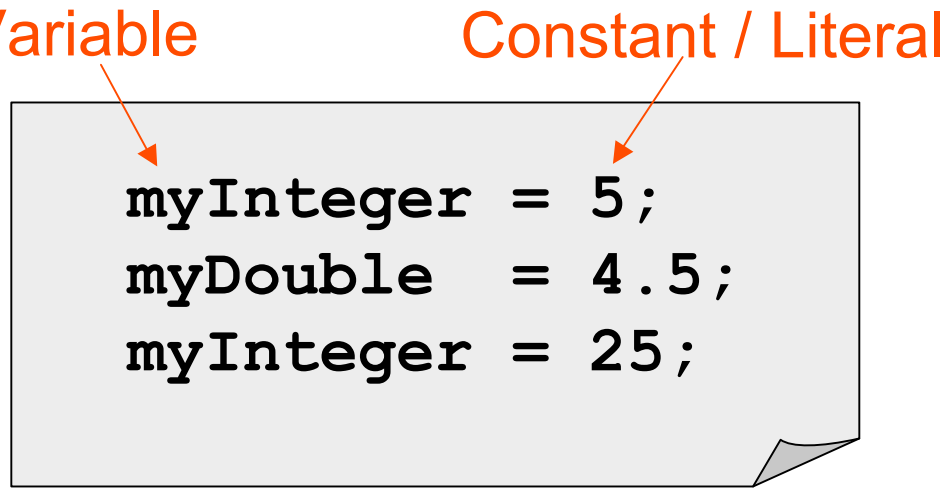
Introduction to Program Variables

- Multiple variables of the same type can be declared at the same time by separating them with a comma.

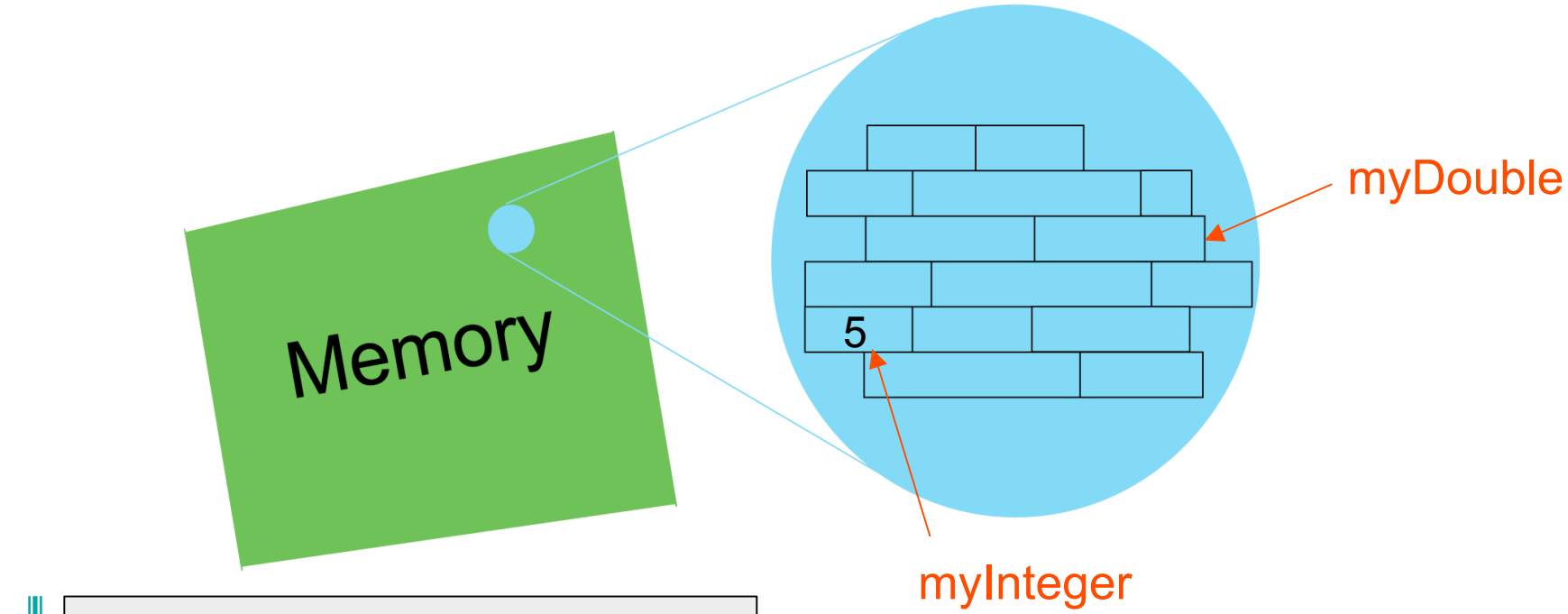
```
int myInteger, myOtherInteger, myThirdInt;
double myDouble;
```

Introduction to Program Variables

- Variables can be assigned a value with the assignment (=) operator.



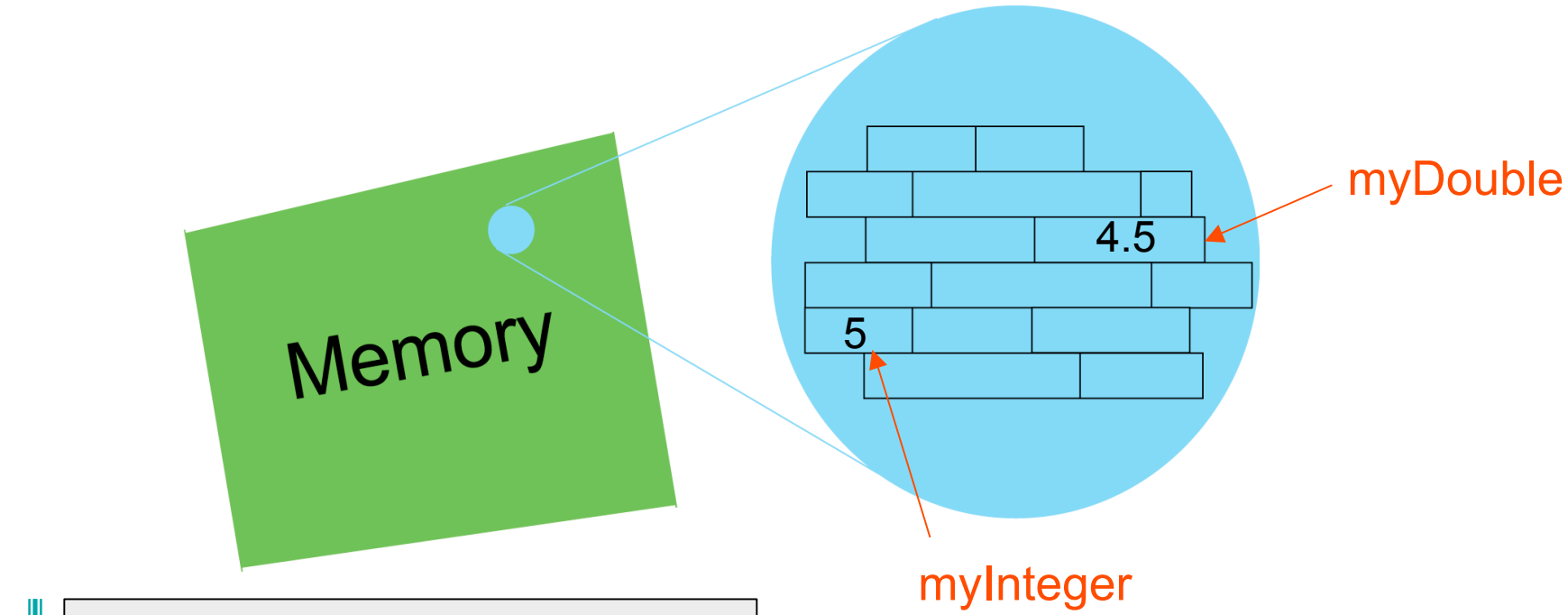
Introduction to Program Variables



sequence

```
myInteger = 5;
myDouble  = 4.5;
myInteger = 25;
```

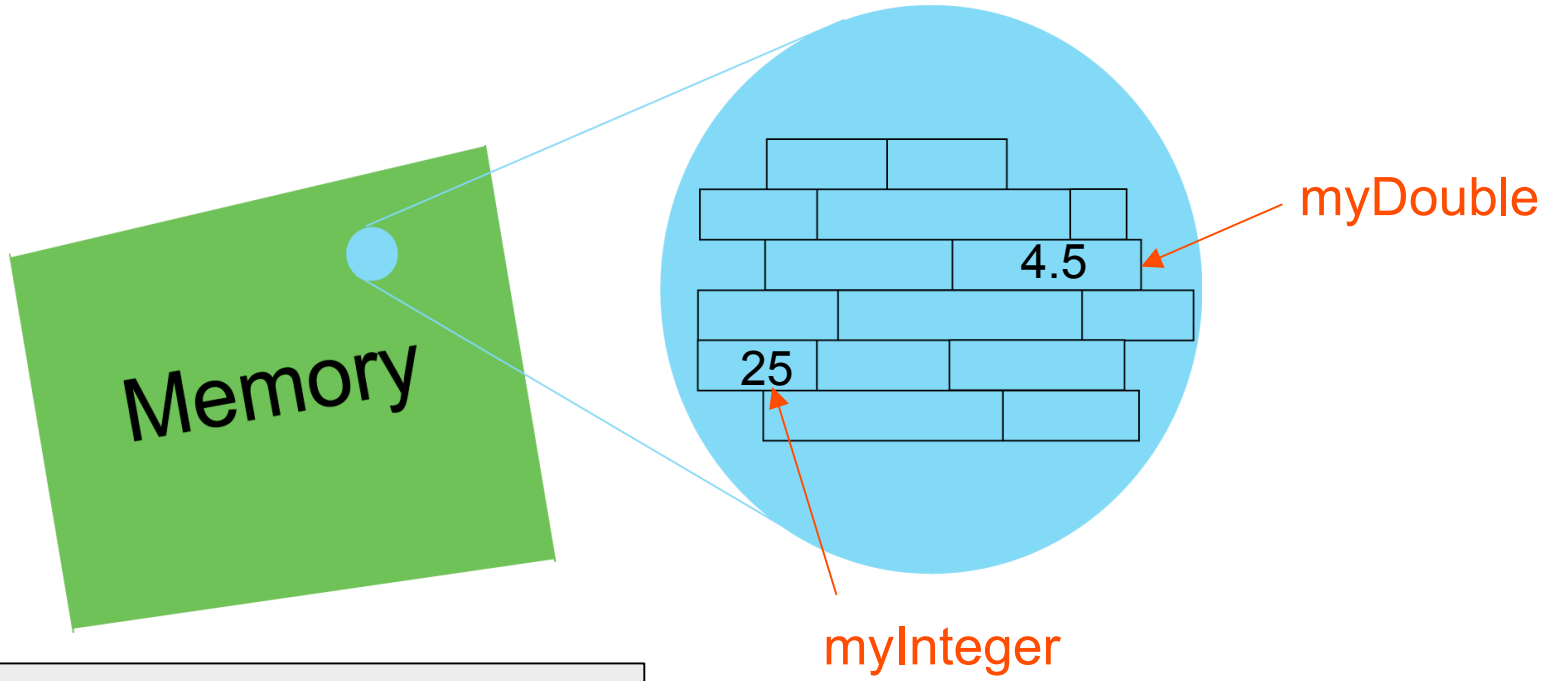
Introduction to Program Variables



sequence

```
myInteger = 5;
myDouble  = 4.5;
myInteger = 25;
```

Introduction to Program Variables



sequence

```
myInteger = 5;
myDouble  = 4.5;
myInteger = 25;
```

The value at any location can change from time to time.

Introduction to Program Variables

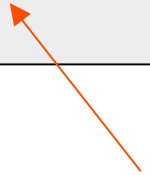
- Remember the definition of a variable:
  - A named memory location used to store data.
- The assignment operator (=) can be thought of as a left arrow, placing a value into a memory location.
  - myInteger ← 5; (put 5 in the myInteger box)
- But we don't have ← on our keyboard, so we use the equals sign instead.
  - myInteger = 5;

Introduction to Program Variables

- In our example, myInteger holds different values at different times.
  - Sequence is important here!
  - The current contents of a variable is what was most recently assigned to it.
- Assignment always occurs from right to left
  - myInteger = 5;
    - Storing 5 into “myInteger” makes sense.
  - 5 = myInteger;
    - Storing “myInteger” into the value of 5 makes no sense.

Introduction to Program Variables

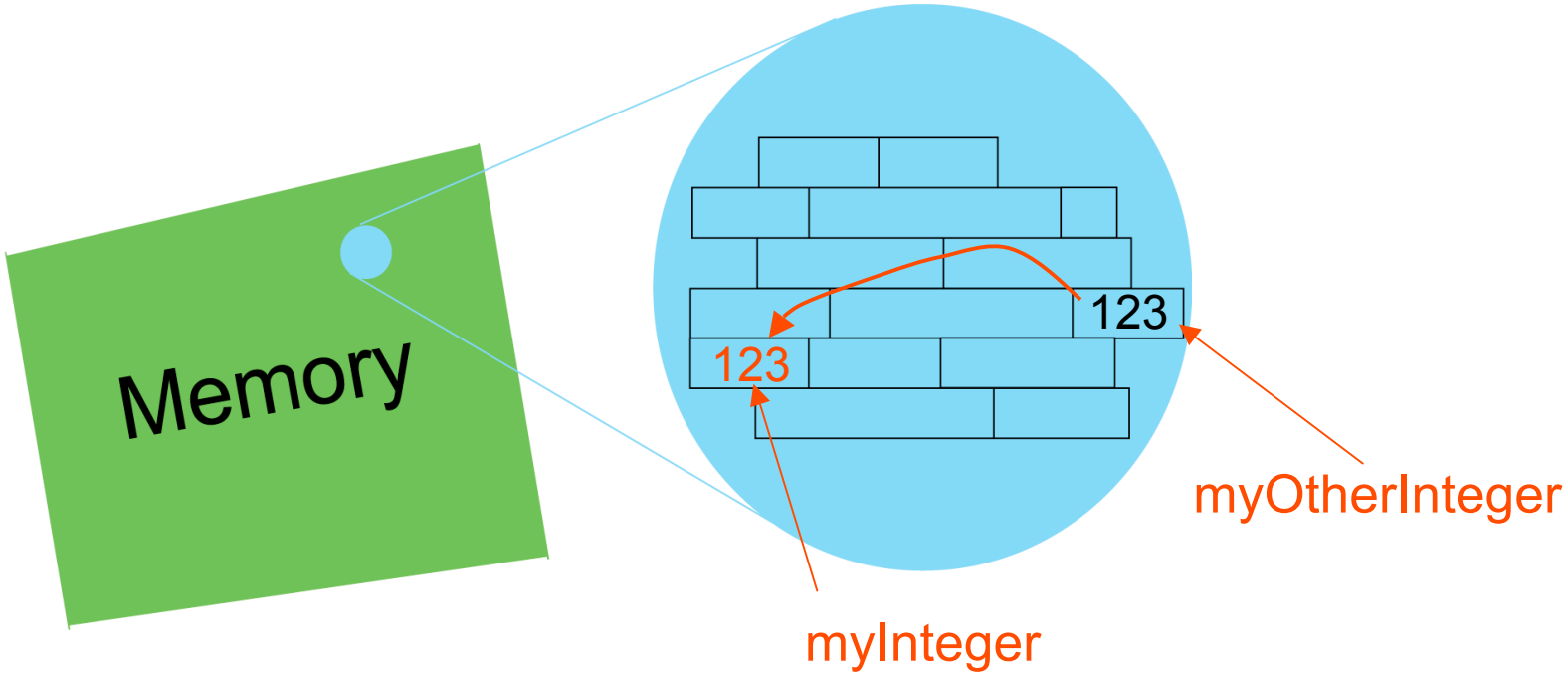
```
myInteger = myOtherInteger;
```



What does this do?

Introduction to Program Variables

```
myInteger = myOtherInteger;
```



Introduction to Program Variables

```
myInteger = myOtherInteger;
```

- When not being assigned to, a variable stands in for the value that it contains.
- In this case, if myOtherInteger contained 20 then this statement would be equivalent to the following:

```
myInteger = 20;
```

Introduction to Program Variables

```
myInteger = myInteger + 5;
```

What does this do?

Introduction to Program Variables

```
myInteger = myInteger + 5;
```

- This adds 5 to the current contents of myInteger and stores the result back into myInteger.
  - If myInteger used to contain 20, it now contains 25.
- Repeating this statement will result in repeatedly adding 5 to myInteger.

Example Program 1

```
double a, b, c, d, e, f, g;

a = 4; b = 25; c = 30;
d = 60; e = 10;

f = b;
f = f + c;
f = f + d;
f = f + e;
g = f / a;
```

What does this program do?

Example Program 1

- It's not intuitively obvious what this program does.

- It can be improved easily by giving the variables useful names.

Example Program 1

```
double b, c, d, e, sum, average;
int numTerms;

numTerms = 4;
b = 25; c = 30; d = 60; e = 10;

sum = b;
sum = sum + c;
sum = sum + d;
sum = sum + e;
average = sum / numTerms;
```

Program Variables

- There are a number of variable naming conventions.

- In 1D04, the following convention will be adopted:
  - All first words are lowercase.
  - Subsequent words have their first letter capitalized.
    - fred, theVariable, bobLovesMe

Program Variables

- Variables cannot start with a number.

- Extremely short or exceptionally long variable names can be hard to work with. Use an appropriate length.

Program Variables

- There is still room for improvement in the code that calculates the average. What can you do to make it “better”?

Program Variables

- There is still room for improvement in the code that calculates the average. In this case, we can simply add the values of b, c, d and e together without doing it sequentially.
- While we do this we need to bear in mind that:
  - Rules for the mathematical order of operations apply to operations on variables.
  - We should use brackets anywhere our intention might be unclear.

“Better” Version

```
double b, c, d, e, average;
int numTerms;

b = 25; c = 30; d = 60; e = 10;
numTerms = 4;

average = (b+c+d+e) / numTerms;
```

Program Variables

- Lastly, variables can be initialized when they are declared.
- Does that mean those variables will behave like constants?
- No
  - Their values can still be changed later in the program.

Initialized Version

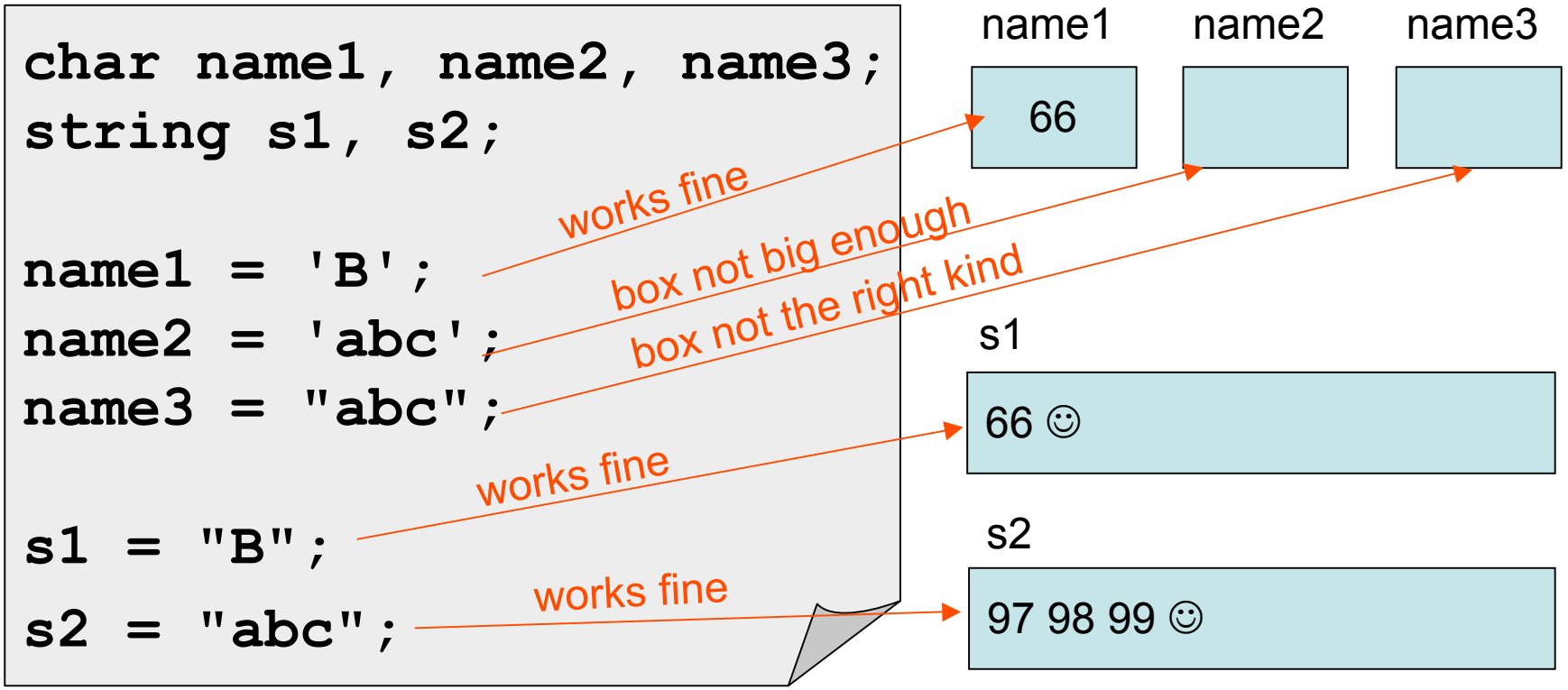
```
double b = 25, c = 30, d = 60,
       e = 10, average;
int numTerms = 4;

average = (b+c+d+e) / numTerms;
```

Program Variables

- Notice the use of quotes in characters and strings.
- The type of box that a literal is given is defined by its quotes.
- Single quotes get a character box. ‘x’ ‘#’
- Double quotes get a string box. “x” “1x3@”
- Double quotes are not single quotes typed twice.

Program Variables

- Examples:



Program Variables

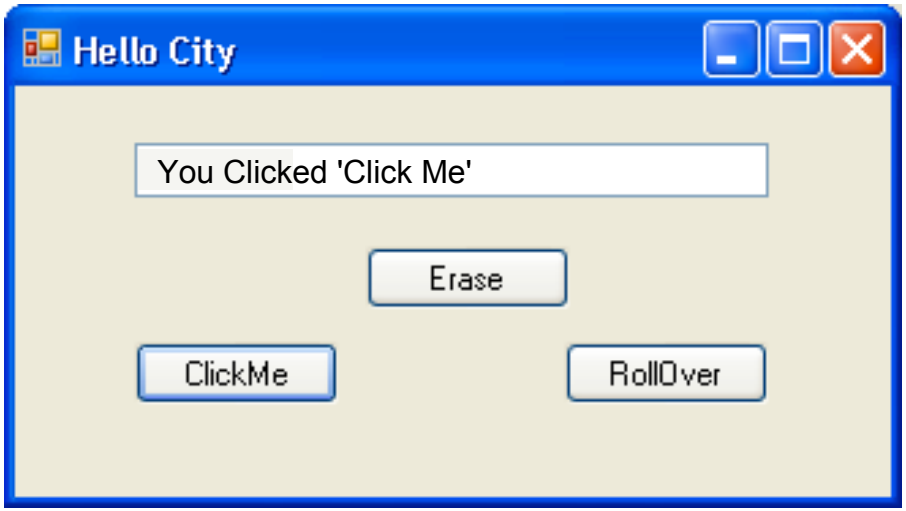
- Booleans can contain only the values true or false.
- They will become more useful as the course progresses.

Types

- The type of a variable provides very important information to the programmer and to the compiler
- The number of students in a tutorial is an integer, not a real number or double – you cannot have 23.45 students in a class!
- Some operations are only valid over certain types
- Type conversions when mixing types

Type Conversion

- Remember “Click Me” from lab 1?



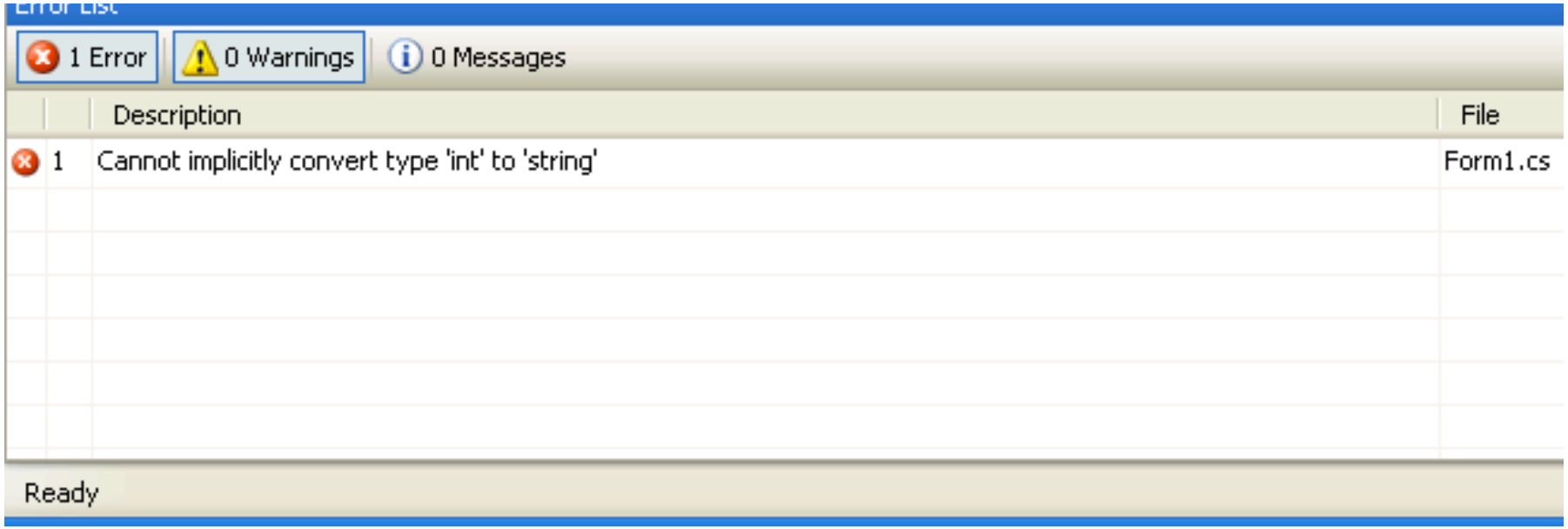

```
void …(…)
{
    textBox1.Text = "You Clicked 'Click Me'";
}
```

Type Conversion

- What happens when we run the method with an integer?

```
void …(…)
{
    int myInteger = 10;
    textBox1.Text = myInteger;
}
```

Type Conversion



Type Conversion

- The variables are in different size/kind of boxes.

- We need to do an explicit conversion from integer to string.

Example Program 2


```
void …(…)
{
    int myInteger = 10;
    textBox1.Text = Convert.ToString(myInteger);
}
```

Type Conversion

- The Convert methods are found in the System library.

- In order to use them as demonstrated in this tutorial the following line must be at the beginning of your source file:
  - using System;

- Visual Studio does this by default for you.

Example Program 3

- It's quite natural to want a numerical type along with a descriptive string in a textbox.

- How do we do that?

Example Program 3

```
private void ClickMe_Click(object sender,
                           EventArgs e)
{
      int sum;

      sum = 35 + 7;
      textBox1.Text = "The Answer to Life,
        The Universe, and Everything is ??";
}
```

We want the value of sum!

Example Program 3

```
private void ClickMe_Click(object sender,
                           EventArgs e)
{
      int sum;

      sum = 35 + 7;
      textBox1.Text = "The Answer to Life,
        The Universe, and Everything is sum";
}
```

Example Program 3


```
pri
    {
        sum = 35 + 7;
        textBox1.Text = "The Answer to Life,
          The Universe, and Everything is sum";
}
```

Example Program 3

- We need to convert sum into a string and concatenate it with “The Answer to …”.

- Convert.ToString(intValue) produces a string consisting of characters representing the value of intValue.

Example Program 3

```
private void ClickMe_Click(object sender,
                           EventArgs e)
{
      int sum;

      sum = 35 + 7;
      textBox1.Text = "The Answer to Life,
        The Universe, and Everything is " +
        Convert.ToString(sum);
}
```

Example Program 3


```
pri
        sum = 35 + 7;
        textBox1.Text = "The Answer to Life,
          The Universe, and Everything is " +
          Convert.ToString(sum);
}
```

Example Program 3

- Actually, this is a special case! We did not need to convert the integer to a string in this particular case.

- Why? Because the concatenation operator (+) implicitly converts numerical values to strings if necessary.

Example Program 3

```
private void ClickMe_Click(object sender,
                                    EventArgs e)
{
     int sum;

     sum = 35 + 7;
     textBox1.Text = "The Answer to Life,
      The Universe, and Everything is "+ sum;
}
```

Concatenation

- Multiple things can be concatenated at once.

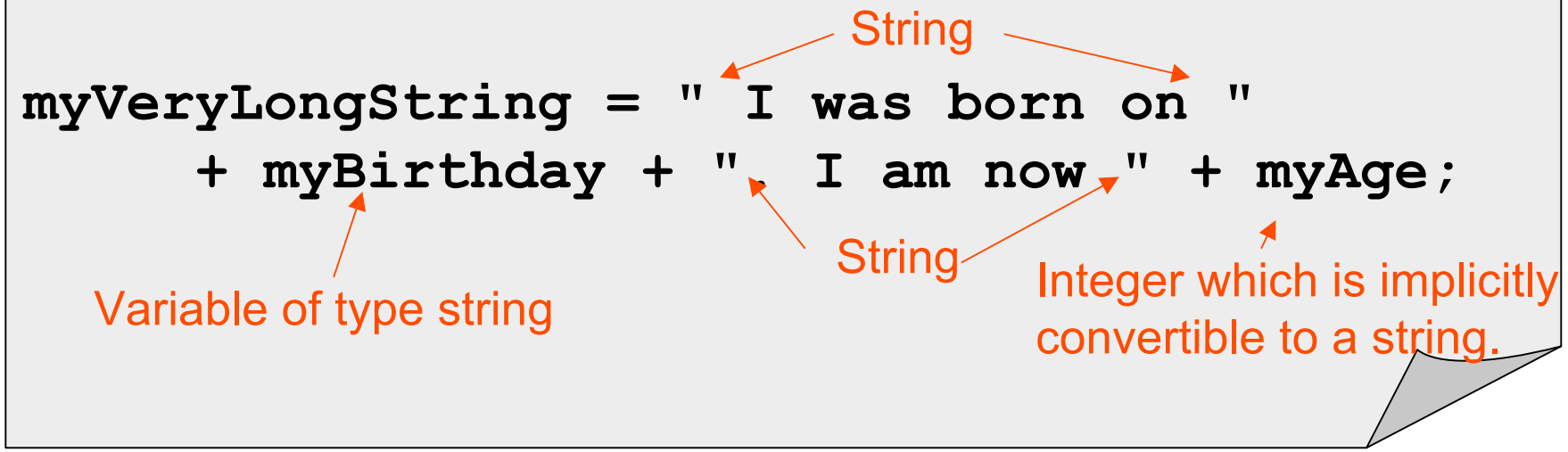
- All must be either strings or implicitly convertible to strings by the concatenation operator.

Example Program 4

```
string myVeryLongString;
int myAge = 22;
string myBirthday = "January 3, 1984";

myVeryLongString = " I was born on "
     + myBirthday + ". I am now " + myAge;
```

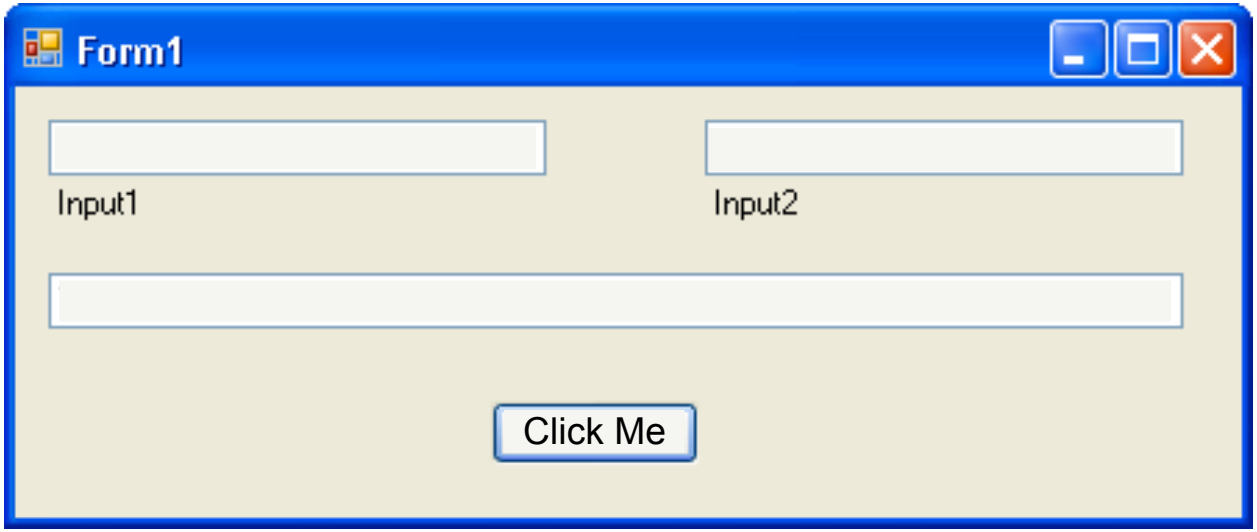
Example Program 4

```
string myVeryLongString;
int myAge = 22;
string myBirthday = "January 3, 1984";

myVeryLongString = " I was born on "
     + myBirthday + ". I am now " + myAge;
```



Inputting Data

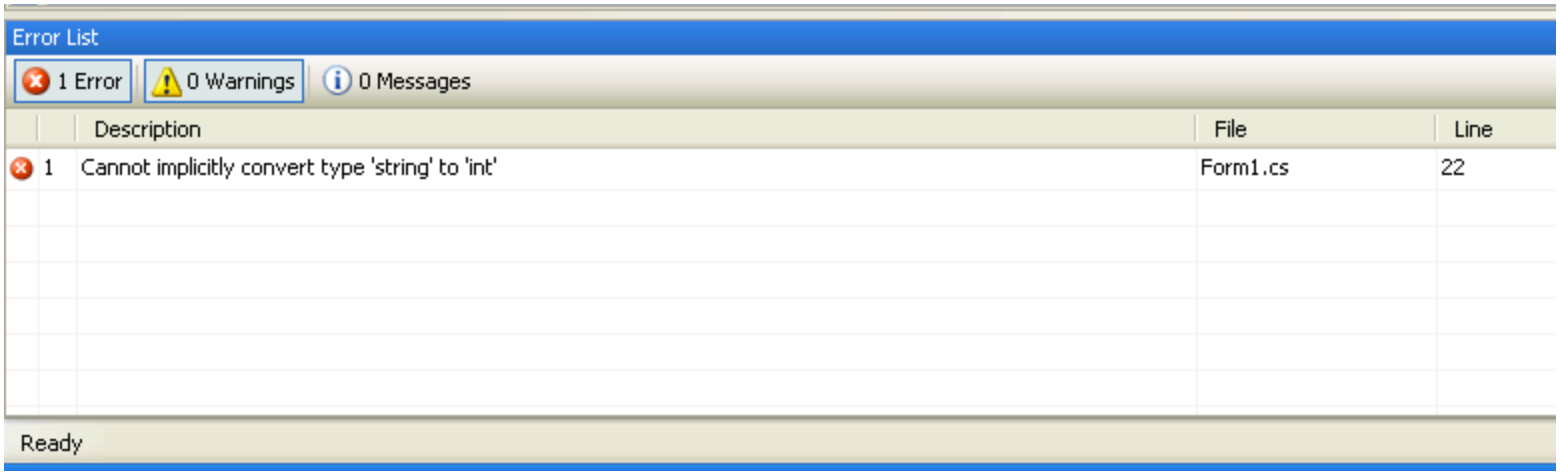
- Suppose we want to change the inputs used to calculate the answer to Life, the Universe, and Everything.
- We could read two numbers from two textboxes, add them, and then write the answer to a textbox.



Example Program 5

```
private void ClickMe_Click(object sender,
                           EventArgs e)
{
  int sum;

  sum = inputBox1.Text + inputBox2.Text;
  outputBox.Text = "The Answer to Life, The Universe,
      and Everything is " + sum;
}
```

Inputting Data



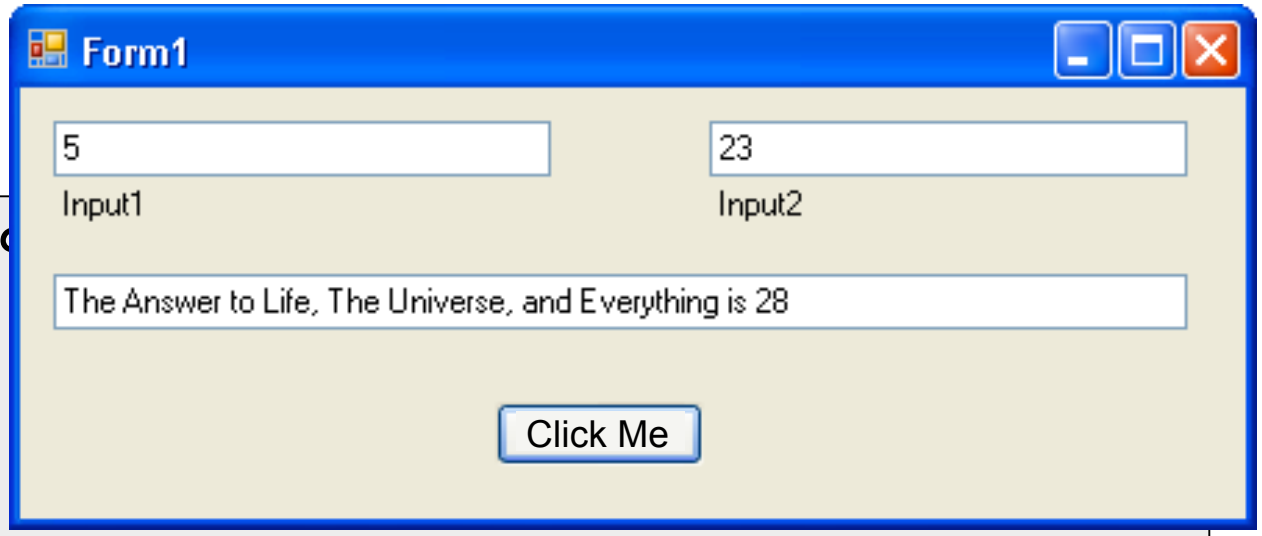
Inputting Data

- Textboxes deal only with Strings.
- We need our inputs to be integers.
- An explicit conversion is required.

Example Program 6

```
private void ClickMe_Click(object sender,
                           EventArgs e)
{
  int sum;

  sum = Convert.ToInt32(inputBox1.Text)
      + Convert.ToInt32(inputBox2.Text);

  outputBox.Text = "The Answer to Life, The Universe,
      and Everything is " + sum;
}
```

Example Program 6




```
private void Clic

{
  int sum;

  sum = Convert.ToInt32(inputBox1.Text)
      + Convert.ToInt32(inputBox2.Text);

  outputBox.Text = "The Answer to Life, The Universe,
      and Everything is " + sum;
}
```

Points to Ponder

- Program variables
- Variable typing
- Type conversions