

Expressions, Variables, and Sequential Programs

*01204111 Computer and Programming
Department of Computer Engineering
Kasetsart University*

Outline

- Simple sequential programs
- Arithmetic expressions
- Basic output
- Variables and important data types

Task: *Dining Bill*



- At a dining place with your friends, there are five items you ordered:

Item	Price
Salad	82
Soup	64
Steak	90
Wine	75
Orange Juice	33



- Write a program to compute the total cost of the bill

Dining Bill – Ideas and Steps



- We simply want to know the summation of all the five numbers
- Somehow, we need to tell the computer to

“Show me the result of all these numbers added up.”



Dining Bill – First Program in C#

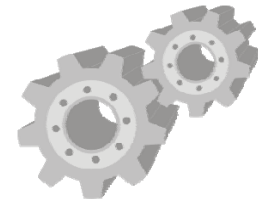


```
using System;

class Program
{
    static void Main()
    {
        Console.WriteLine( 82+64+90+75+33 );
    }
}
```

- Notes: the outermost “namespace” declaration is omitted

Explanation



```
using System;
```

Tell the computer we want to use the **System** library

```
class Program
```

In C#, a program must be inside a **class**.
(The class name can be anything.)

```
{
```

```
    static void Main()
```

Program's entry point is indicated by the **Main()** method.

```
{
```

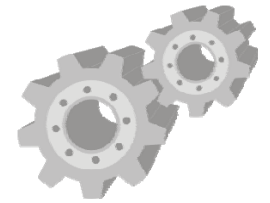
```
    Console.WriteLine( 82+64+90+75+33 );
```

```
}
```

This line is our only statement (i.e., command) in the program

```
}
```

What Is a Statement?



- A (programming) **statement** is a complete command to order the computer to do something
- In our example, it is the line

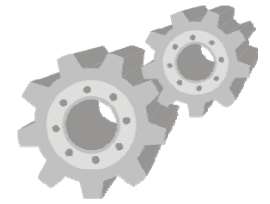
```
Console.WriteLine( 82+64+90+75+33 );
```

- This is equivalent to giving a command

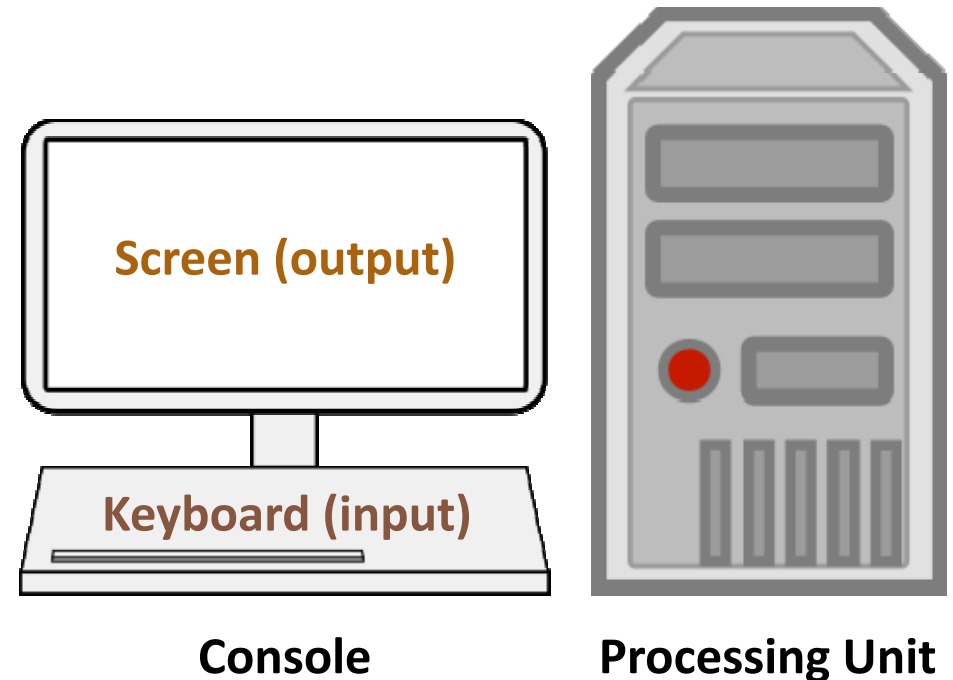
*“Hey **Console!** Please **write a line** with the value of the expression **82+64+90+75+33.**”*

- In C#, a single statement ends with ; (semicolon)
- A program usually consists of many statements

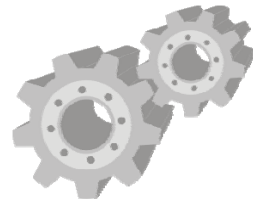
What Is the Console?



- The ***console*** is a simple text entry and display device for the computer
 - Keyboard for text input
 - Screen for text output
- In C#
 - The method `Console.Write` or `Console.WriteLine` outputs text to screen
 - The method `Console.ReadLine` inputs text from keyboard

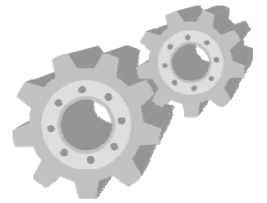


What Is a Method?



- A **method** is a set of steps prepared to perform a certain task
 - Also known as a **procedure** or a **function** in some programming languages
- Examples are
 - `Console.WriteLine` – writes something on the screen
 - `Console.ReadLine` – reads input from the keyboard
- Methods that give back values can be used as **expressions**
 - E.g., `Console.ReadLine` gives a string value that user entered on the keyboard
- We will learn how to write your own method in the next class
 - For now, we will use only methods already provided by C#

What Is an Expression?



- An ***expression*** is something that can be evaluated to a value
 - An ***arithmetic expression*** can be evaluated to a numerical value
- In our example, it is the part

`82+64+90+75+33`

- This part gets evaluated by the computer. The result, 344, is then given to the **Console.WriteLine** method.

Other Statement Examples



`Console.WriteLine(20);`

- displays a single value, 20, and move the cursor to the new line

`Console.WriteLine();`

- simply move the cursor to the new line

`Console.WriteLine("Hello");`

- displays the text HELLO and move the cursor to the new line
- "HELLO" is a *string* expression

`Console.Write("Hi");`

- displays the text Hi without moving the cursor to the new line

Dining Bill – Revised Program



- Let us modify our previous example to make it output more informative

```
using System;

class Program
{
    static void Main()
    {
        Console.Write("Total cost is ");
        Console.WriteLine(82+64+90+75+33);
    }
}
```

- Our program now has two statements, executed from top to bottom

Better Than a Calculator?



- Of course, using a simple calculator for this task might seem much easier. However,
 - Repeating the whole task is tedious, especially with many numbers
 - When making a small mistake, the whole process must be restarted from the beginning
- With a program, all steps can be easily repeated and modified



Task: *Discounted Dining Bill*



- Based on the previous scenario, one of your friends just happens to carry a member card with 20% discount
- Modify the program to compute the final cost



Discounted Dining Bill – Ideas



- Start with the same summation expression
- With 20% discount, the final cost will be 80% of the original
- Therefore, we just multiply the original expression by 0.8
- In most programming languages, ***** means ***multiply***


Discounted Dining Bill – 1st Attempt



- Will this work?

```
using System;

class Program
{
    static void Main()
    {
        Console.Write("Total cost is ");
        Console.WriteLine( 82+64+90+75+33 * 0.8);
    }
}
```

An orange arrow points to the multiplication operation in the code, specifically to the space between the sum and the discount factor 0.8.

Caveats – Operator Precedence



- In C# (and most programming languages), different operators have different precedence in order of operations
- For example, $*$ has precedence over $+$ in this expression, no matter how many spaces are used

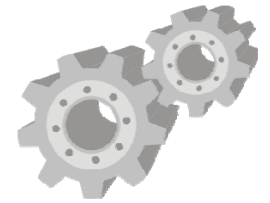
$82+64+90+75+33 * 0.8$

- Therefore, the above expression is equivalent to:

$82+64+90+75+(33*0.8)$

which is wrong

Operator Precedence



- C# (and most programming languages) evaluates expressions in this order

Operators	Precedence
()	Highest
* / %	:
+ -	Lowest

- Operations of the same precedence are evaluated from left to right
- When not sure, always use parentheses

Operator Precedence: Examples



Expression	Equivalent to
$2 * 3 + 4 * 5$	$(2 * 3) + (4 * 5)$
$1 + 2 + 3 + 4$	$((1 + 2) + 3) + 4$
$(2 + 3) / 5 * 4$	$((2 + 3) / 5) * 4$
$3 - 2 - 5 - (7 + 6)$	$((3 - 2) - 5) - (7 + 6)$
$10 + 9 \% 2 + 30$	$(10 + (9 \% 2)) + 30$
$10 / 2 * 5 \% 3$	$((10 / 2) * 5) \% 3$

Discounted Dining Bill – Revised Program



```
using System;

class Program
{
    static void Main()
    {
        Console.Write("Total cost is ");
        Console.WriteLine( (82+64+90+75+33) * 0.8);
    }
}
```

Task: *Shopping Bill*



- At a grocery store, you are putting these items in your shopping cart

Item	Price per item	How many
Apple	12	6
Orange	15	3
Banana	8	4
Tomato	10	5
Melon	30	2



- Write a program to compute the total cost of items in your shopping chart

Shopping Bill – Program #1



- Knowing that ***** has precedence over **+**, this program will work
 - But the expression looks confusing

```
using System;

class Program
{
    static void Main()
    {
        Console.Write("Total cost is ");
        Console.WriteLine(12*6+15*3+8*4+10*5+30*2);
    }
}
```

Shopping Bill – Program #2



- This is the same program, but parentheses and spaces can make the program much easier to read and spot errors

```
using System;

class Program
{
    static void Main()
    {
        Console.Write("Total cost is ");
        Console.WriteLine(
            (12*6) +
            (15*3) +
            ( 8*4) +
            (10*5) +
            (30*2) );
    }
}
```

Task: *Bill Sharing*



- At the same restaurant, five people are splitting the bill and share the total cost
- Write a program to compute the amount each person has to pay

Item	Price
Salad	82
Soup	64
Steak	90
Wine	75
Orange Juice	33



Bill Sharing – Ideas



- Just compute the total and divide it by 5
- The result should be the amount each person has to pay

Bill Sharing – First Attempt



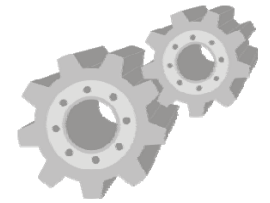
```
using System;

class Program
{
    static void Main()
    {
        Console.Write("Total amount: ");
        Console.WriteLine( 82+64+90+75+33 );
        Console.Write("Each has to pay: ");
        Console.WriteLine( (82+64+90+75+33) / 5 );
    }
}
```

- This is the output. Is it correct?

```
Total amount: 344
Each has to pay: 68
```

Integer vs. Floating Point Division



- In C#, when dividing two integers (whole numbers), the result is also a whole number
 - Similar to a long division taught in primary school
 - The fraction part is discarded
- To obtain the result with fraction, at least one of the numbers must be floating point

Expression	Evaluated to
10/4	2 (not 2.5)
10.0/4	2.5
10/4.0	2.5
10.0/4.0	2.5

Bill Sharing – Revised Program



```
using System;

class Program
{
    static void Main()
    {
        Console.Write("Total amount: ");
        Console.WriteLine( 82+64+90+75+33 );
        Console.Write("Each has to pay: ");
        Console.WriteLine( (82+64+90+75+33) / 5.0 );
    }
}
```

- The result should be correct
- However, the summation expression gets repeated twice

Bill Sharing – Revised Program#2

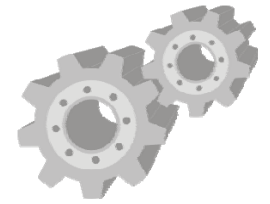


- We now store the result of the total amount in a ***variable***

```
using System;

class Program
{
    static void Main()
    {
        int total;
        total = 82+64+90+75+33;
        Console.Write("Total amount: ");
        Console.WriteLine(total);
        Console.Write("Each has to pay: ");
        Console.WriteLine(total/5.0);
    }
}
```

What Is a Variable?



- A **variable** is a storage location for storing a value
- In C#, a variable needs to be declared with a data type before used

This variable's name is **total**

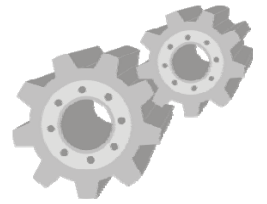
```
int total;
```

It is used to store an **integer**

- Once declared, it can store a value with an assignment statement (=)

```
total = 82+64+90+75+33;
```

More on Variable Declaration



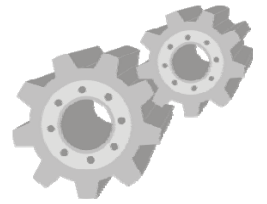
- Declaration and assignment can be done in one statement

```
int total = 82+64+90+75+33;
```

- Multiple variables can be declared (and assigned) in one statement

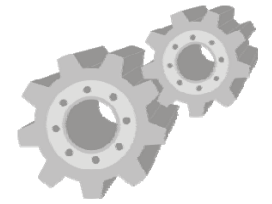
```
int width=30, height=50, area;
```

Naming Variables



- Different programming languages may have slightly different rules for naming a variable
- Some common rules are
 - A name consists of only alphanumeric characters (A-Z, a-z, 0-9) and underscores (_)
 - The first character cannot be a number
 - A name must not be a reserved word
 - Lowercase and uppercase letters mean different things

Naming Variables: Examples



Name	Correct?	Reason
radius	✓	OK
pay_rate	✓	OK
G_force	✓	OK
while	✗	is a reserved word
jack&jill	✗	contains a symbol &
8bus	✗	starts with a number
buggy-code	✗	contains a symbol -
class	✗	is a reserved word
Class	✓	OK
_class	✓	OK

Readability Counts!



- Your program is written not only for computer, but also human, to read
- Variable names should be meaningful

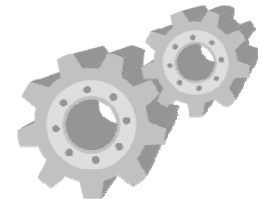
Bad Example

```
int a = 30;  
int b = 50;  
int c = a * b;
```

Good Example

```
int height = 30;  
int width = 50;  
int area = height * width;
```

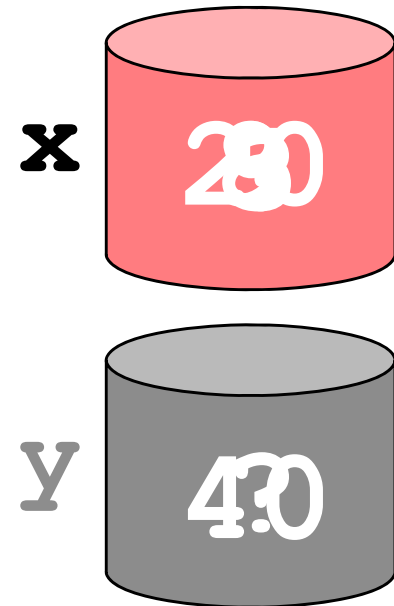
How a Variable Stores Value?



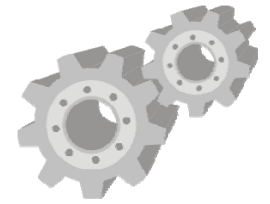
- A variable can be declared only once
- Assignment can be done over and over
 - However, a variable can store one value at a time
- A variable serves as an expression evaluated to its stored value

Declare	<code>int x;</code>
Assign	<code>x = 5;</code>
Assign	<code>x = 20;</code>
Declare	<code>int y;</code>
Assign	<code>y = x*2;</code>
Assign	<code>x = 8;</code>
Assign	<code>x = x+1</code>

→ No longer affect the value of y



Important Data Types



- Sometimes you need to store something other than an integer in a variable
- C# provides several data types for different purposes
- Some important types are listed here

Type	Purpose	Usage Example
int	storing a whole number (positive, negative, zero)	<code>int total = 25;</code>
double	storing a number with fraction	<code>double g_force = 9.81;</code>
char	storing a single character	<code>char first = 'A';</code>
string	storing a sequence of character	<code>string name = "John";</code>

Conclusion

- A program consists of one or more statements
- Each statement can be an assignment that assigns the value of an expression to a variable, or a method call that takes zero or more expressions for performing certain tasks
- An expression is a portion of code that can be evaluated to a value
- A variable is a storage in the memory for storing a single value
 - The type of the stored value must match the type of the variable itself
- The method **Console.WriteLine** can be used to displayed the value of an expression on screen

References



- Basic C# syntax, variables, and expressions
[https://msdn.microsoft.com/en-us/library/hh147285\(v=vs.88\).aspx](https://msdn.microsoft.com/en-us/library/hh147285(v=vs.88).aspx)
- Operator precedence and order of evaluation
<https://msdn.microsoft.com/en-us/library/2bxt6kc4.aspx>
- C# reserved words
<https://msdn.microsoft.com/en-us/library/x53a06bb.aspx>
- Data types
[https://msdn.microsoft.com/en-us/library/cs7y5x0x\(v=vs.90\).aspx](https://msdn.microsoft.com/en-us/library/cs7y5x0x(v=vs.90).aspx)

Syntax Summary I

- C# program structure

Without namespace

```
using System;

class ProgramName
{
    static void Main()
    {
        statement1;
        statement2;
        :
    }
}
```

With namespace

```
using System;

namespace NamespaceName
{
    class ProgramName
    {
        static void Main()
        {
            statement1;
            statement2;
            :
        }
    }
}
```

Syntax Summary II

- Variable declaration

- Without initial value

```
DataType variableName;
```

- With initial value

```
DataType variableName = value;
```

- Multiple declarations

```
DataType variableName1 = value1, variableName2 = value2, ...;
```