Computer Programming
Lecture 2

Hung-Yu Wei
Department of Electrical Engineering
National Taiwan University

Last lecture

- A simple C++ program
  - Main program
  - Display text on the monitor
    - <iostream>
    - Std::cout << “your text to display”
  - Write comments(註解)
    - Always describe your programs at the beginning of the files
    - Add comments within the source codes to make the codes clear

Review: comments

- Inline comment (1 line)

```
// my comment here
```

- Multiple lines of comments

```
/*
      my comment here
         more comments
         more and more comments
 */
```

```cpp
// Fig. 2.1: fig02_01.cpp
// Text-printing program.
#include <iostream> // allows program to output data to the screen

// function main begins program execution
int main()
{
   std::cout << "Welcome to C++!\n"; // display message

   return 0; // indicate that program ended successfully

} // end function main
```

Review: main program

```
int main()
{
    this is your program

    return 0;
}
```

Review: Display text on screen

```
#include <iostream>



std::cout << “you text here";
```

What will we learn today?

- Display text (continued)
- Variables
- Input from keyboard
- Understand program operation in “memory”
- Arithmetic (+,-,*,/)

Display text differently

- Codes

```
std::cout << "Welcome to C++!\n";
```

  - Results

```
Welcome to C++!
```

- Codes

```
std::cout << "Welcome “;
std::cout << “to C++!\n";
```

  - Results

```
Welcome to C++!
```

Display text differently

- Codes

  ```
  std::cout << "Welcome\n to\n\n C++!\n";
  ```

  - Results

    ```
    Welcome
     to

     C++!
    ```

Declaration of variables

- Declaration

  ```
  int x;
  ```

- Variable
  - x
- Data types
  - int (integer)
  - float (real number)
  - double (real number, better precision)
  - char (character)

More declarations

- Declare several variables

```
int number1;   // number1 is …
int number2;   // number2 is …
int sum;       // sum is …
```

- Declare in 1-line

```
int number1,number2,sum;
```

How to name your variable?

- Variables in C++ is “case-sensitive”
  - xyz, Xyz, XYZ
  - X1, x1
- Name your variables with
  - Characters (a,b,c,….)
  - Number digits (1,2,3)
  - _ **(not -)**
- Not allowed
  - Keyword (main, int, …)
  - Not begin with a digit
    - 5566xyz → not allowed

Input text

- We know how to output text
  - std::cout <<
- Input some text
  - std::cin >>
- Don't forget to include <iostream>
- Example

```
std::cin >> x
```

```cpp
// Fig. 2.5: fig02_05.cpp
// Addition program that displays the sum of two numbers.
#include <iostream> // allows program to perform input and output

// function main begins program execution
int main()
{
   // variable declarations
   int number1; // first integer to add
   int number2; // second integer to add
   int sum; // sum of number1 and number2

   std::cout << "Enter first integer: "; // prompt user for data
   std::cin >> number1; // read first integer from user into number1

   std::cout << "Enter second integer: "; // prompt user for data
   std::cin >> number2; // read second integer from user into number2

   sum = number1 + number2; // add the numbers; store result in sum

   std::cout << "Sum is " << sum << std::endl; // display sum; end line

   return 0; // indicate that program ended successfully

} // end function main
```

Execution results

```
Enter first integer: 45
Enter second integer: 72
Sum is 117
```

```
Enter first integer: 3
Enter second integer: 5
Sum is 8
```

Line 21 (endl)

```
std::cout << "Sum is " << sum << std::endl;
```

- endl
  - Abbreviation of “end line”
  - Tell the program this is the end of a line
    - Just flush everything in a “buffer”
- Multiple <<
  - Also known as cascading
  - A flexible tool in C++ for output
- Read 2 integers at the same time

  ```
  cin >> number1 >> number2;
  ```

Review standard I/O

- Family of <iostream>
  - Std::cout <<
  - Std::cin >>
  - Std::endl
- Are you confused with << and >>?
  - Think about the '*direction*'
    - 象形 (箭頭的方向)

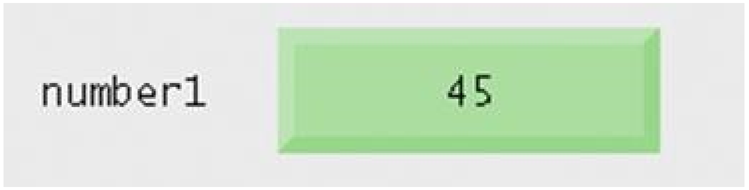
How does memory work?

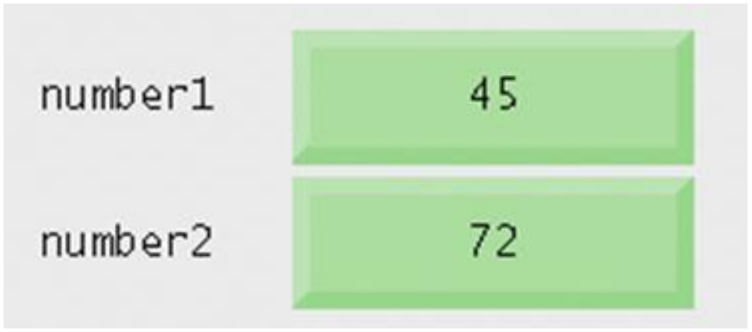
- Step 1

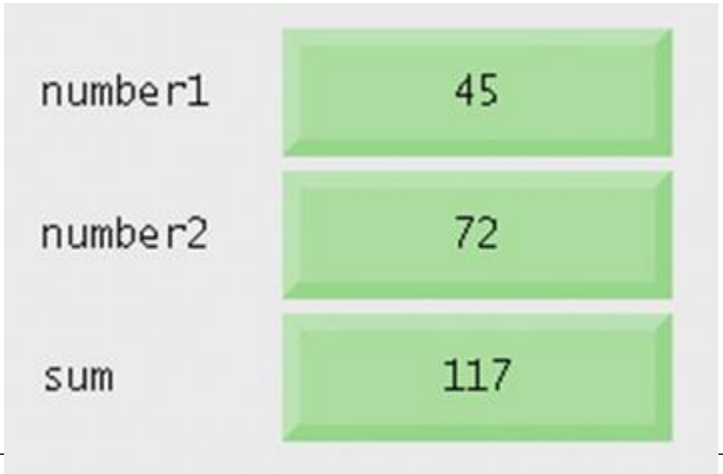
  std::cin >> number1;
- Step 2

  std::cin >> number2;
- Step 3

  sum=number1+number2;







Memory operation

- Destructive
  - The value stored in a memory location is destructed after an operation
  - For example, write a value to a variable

```
number1=15;
std::cin >> number1;
```

- Nondestructive
  - The value stored in a memory location is NOT destructed after an operation
  - For example, read a value from a variable

```
number1=1+2;
std::cout << number1;
number3=number1+number2;
```

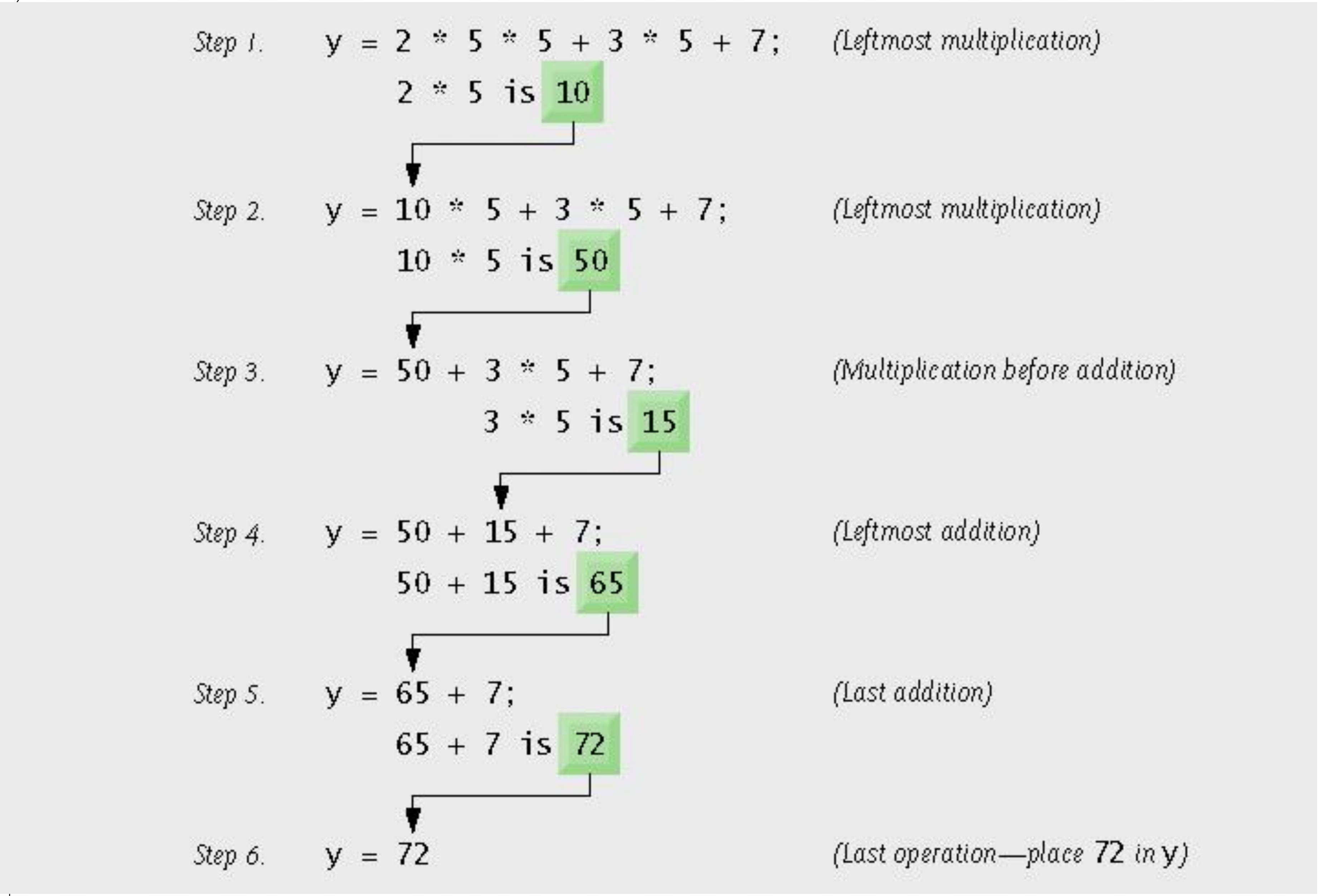
Arithmetic

- +, -
- *
  - Do not use “x” for multiplication
- /
  - Divide
  - give you an integer answer（無條件捨去)
    - 19/5=3
    - 17/4=4
- %
  - Modulus (取餘數)
    - 19%5=4
    - 17%4=1
- Y=aX+b
  - Y/X→a
  - Y%X→b

Arithmetic (continued)

- / and %
  - How would you like to use them together?
- Precedence of computation

Operator(s)	Operation(s)	Order of evaluation (precedence)
`( )`	**Parentheses**	**Evaluated first. If the parentheses are nested, the expression in the innermost pair is evaluated first. If there are several pairs of parentheses "on the same level" (i.e., not nested), they are evaluated left to right.**
`*`	**Multiplication**	**Evaluated second. If there are several, they are evaluated left to right.**
`/`	**Division**	
`%`	**Modulus**	
`+`	**Addition**	**Evaluated last. If there are several, they are evaluated left to right.**
`-`	**Subtraction**	



Step 1. `y = 2 * 5 * 5 + 3 * 5 + 7;` *(Leftmost multiplication)*

`2 * 5 is 10`

Step 2. `y = 10 * 5 + 3 * 5 + 7;` *(Leftmost multiplication)*

`10 * 5 is 50`

Step 3. `y = 50 + 3 * 5 + 7;` *(Multiplication before addition)*

`3 * 5 is 15`

Step 4. `y = 50 + 15 + 7;` *(Leftmost addition)*

`50 + 15 is 65`

Step 5. `y = 65 + 7;` *(Last addition)*

`65 + 7 is 72`

Step 6. `y = 72` *(Last operation—place* `72` *in* `y`*)*

"if" statement

- Syntax

  ```
  if (condition)
  statement to execute;
  ```

- Multiple statements to execute

  ```
  if (condition)
  {     statement to execute;
        more statements to execute;
  }
  ```

- C++ relational operation
  - >
  - <
  - >=
  - <=
  - == (equal)
  - != (not equal)

‘=‘ and ‘==‘

- Assignment (=)
  - x=y
  - x←y
  - Assign the value of y to x
- Comparison (==)
  - x==y
  - True(1): if x is equal to y
  - False(0): if x is not equal to y
- Example: z= x==y

example

- Compare x and y

```
if ( x == y )
    cout << x << " == " << y << endl;
 if ( x != y )
   cout << x << " != " << y << endl;
if ( x < y )
    cout << x << " < " << y << endl;
```

using

- <u>**using**</u> declaration

```
using std::cout;
using std::cin;
using std::endl;
```

- in the program

```
cout << x << " == " << y << endl;
```

- std::cout

```cpp
// Fig. 2.13: fig02_13.cpp
// Comparing integers using if statements, relational operators
// and equality operators.
#include <iostream> // allows program to perform input and output

using std::cout; // program uses cout
using std::cin;  // program uses cin
using std::endl; // program uses endl

// function main begins program execution
int main()
{
   int number1; // first integer to compare
   int number2; // second integer to compare

   cout << "Enter two integers to compare: "; // prompt user for data
   cin >> number1 >> number2; // read two integers from user

   if ( number1 == number2 )
      cout << number1 << " == " << number2 << endl;

   if ( number1 != number2 )
      cout << number1 << " != " << number2 << endl;

   if ( number1 < number2 )
      cout << number1 << " < " << number2 << endl;

   if ( number1 > number2 )
      cout << number1 << " > " << number2 << endl;

   if ( number1 <= number2 )
      cout << number1 << " <= " << number2 << endl;

   if ( number1 >= number2 )
      cout << number1 << " >= " << number2 << endl;

   return 0; // indicate that program ended successfully

} // end function main
```

```
Enter two integers to compare: 3  7
3 != 7
3 < 7
3 <= 7
```

```
Enter two integers to compare: 22  12
22 != 12
22 > 12
22 >= 12
```

```
Enter two integers to compare: 7  7
7 == 7
7 <= 7
7 >= 7
```

Precedence of operations

Operators	Associativity	Type
`()`	left to right	parentheses
`*` `/` `%`	left to right	multiplicative
`+` `-`	left to right	additive
`<<` `>>`	left to right	stream insertion/extraction
`<` `<=` `>` `>=`	left to right	relational
`==` `!=`	left to right	equality
`=`	right to left	assignment

What have we learned today?

- Reminder: download slides from Ceiba

- New materials in today's lecture
  - std::cin
    - using
  - +,-,*,/,%
  - =, ==, !=
  - if
- Reading: Chapter 2 (2.3~2.7)