Chapter 2
Introduction to C Programming

Associate Prof. Yuh-Shyan Chen
Dept. of Computer Science and Information Engineering
National Chung-Cheng University

Outline

2.1 Introduction
2.2 A Simple C Program: Printing a Line of Text
2.3 Another Simple C Program: Adding Two Integers
2.4 Memory Concepts
2.5 Arithmetic in C
2.6 Decision Making: Equality and Relational Operators

2.1 Introduction

- C programming language
  - Structured and disciplined approach to program design
- Structured (結構化) programming
  - Introduced in chapters 3 and 4
  - Used throughout the remainder of the book

計算機實習

Week 1: 請熟悉 Window 系統 and Microsoft Visual C++ 的編譯環境. 請助教自行說明. 請助教確認組別. 週一(A, B組), 週二(C, D組), 週三(E, F組), 週四(G, H組). 助教上課名單確定. (自行準備磁碟片備份資料), 並撰寫你 (妳)的第一個 C 程式.
2.2 A Simple C Program: Printing a Line of Text

- Fig. 2.1: fig02_01.c
- A first program in C
- Source code to:
- int main()
- printf( "Welcome to C!\n" );
- return 0;
- Welcome to C!

Comments
- Text surrounded by /* and */ is ignored by computer
- Used to describe program
- include <stdio.h>
  - Preprocessor directive - tells computer to load contents of a certain file
  - <stdio.h> allows standard input/output operations

-int main()
  - C++ programs contain one or more functions, exactly one of which must be main
  - Parenthesis () used to indicate a function
  - int means that main "returns" an integer value
  - Braces indicate a block
  - The bodies of all functions must be contained in braces

 printf( "Welcome to C!\n" );
  - Instructs computer to perform an action
  - Specifically, prints string of characters within quotes
  - Entire line called a statement
  - All statements must end with a semicolon
  - \ - escape character
  - Indicates that printf should do something out of the ordinary
  - \n is the newline character

return 0;
  - A way to exit a function
  - return 0, in this case, means that the program terminated normally
  - Right brace }
  - Indicates end of main has been reached
● Linker
○ When a function is called, linker locates it in the library
○ Inserts it into object program
○ If function name misspelled, linker will spot error because it cannot find function in library

2.3 Another Simple C Program: Adding Two Integers

○ As before
○ Comments, `#include <stdio.h>` and `main`

```c
#include <stdio.h>

int main() {
    int integer1, integer2, sum; /* declaration */
    printf("Enter first integer
" ); /* prompt */
    scanf("%d", &integer1); /* read an integer */
    printf("Enter second integer
" ); /* prompt */
    scanf("%d", &integer2); /* read an integer */
    sum = integer1 + integer2; /* assignment of sum */
    printf("Sum is %d
", sum ); /* print sum */
    return 0; /* indicate that program ended successfully */
}
```

Enter first integer
45
Enter second integer
72
Sum is 117

○ `int`, `integer1`, `integer2`, `sum` - variable names (identifiers)
○ Identifiers: consist of letters, digits (cannot begin with a digit), and underscores, case sensitive
○ Declarations appear before executable statements
○ If not, syntax (compile) error
2.3 Another Simple C Program: Adding Two Integers (II)

- `scanf( "%d", &integer1 );`
  - Obtains value from user
  - `scanf` uses standard input (usually keyboard)
  - This `scanf` has two arguments
    - `%d` - indicates data should be a decimal integer
    - `&integer1` - location in memory to store variable

- `%d` is confusing in beginning - just remember to include it with the variable name in `scanf` statements
  - It will be discussed later
- User responds to `scanf` by typing in number, then pressing the enter (return) key

2.3 Another Simple C Program: Adding Two Integers (III)

- `=` (assignment operator)
  - Assigns value to a variable
  - Binary operator (has two operands)
    - `sum = variable1 + variable2;`
  - Variable receiving value on left
- `printf( "Sum is %d\n", sum );`
  - Similar to `scanf - %d` means decimal integer will be printed
    - `sum` specifies what integer will be printed
2.4 Memory Concepts

- Variables
  - Variable names correspond to locations in the computer's memory.
  - Every variable has a name, a type, a size and a value.
  - Whenever a new value is placed into a variable (through `scanf`, for example), it replaces (and destroys) previous value.
  - Reading variables from memory does not change them.
- A visual representation

Calculations can be performed inside `printf` statements

```c
printf( "Sum is %d\n", integer1 + integer2 );
```
2.5 Arithmetic

- Arithmetic calculations are used in most programs
  - Use * for multiplication and / for division
  - Integer division truncates remainder
    
    \[ 7 / 5 \] evaluates to 1
  - Modulus operator returns the remainder
    
    \[ 7 \% 5 \] evaluates to 2

- Operator precedence
  - Some arithmetic operators act before others (i.e., multiplication before addition)
  - Use parenthesis when needed
  - Example: Find the average of three variables \( a \), \( b \) and \( c \)
    
    Do not use: \( a + b + c / 3 \)
    Use: \( (a + b + c) / 3 \)

2.5 Arithmetic (II)

- Arithmetic operators:

<table>
<thead>
<tr>
<th>Operator</th>
<th>Mathematical</th>
<th>Algebraic Expression</th>
<th>C Expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>Addition</td>
<td>+</td>
<td>( x + y )</td>
<td>( x + y )</td>
</tr>
<tr>
<td>Subtraction</td>
<td>-</td>
<td>( p - c )</td>
<td>( p - c )</td>
</tr>
<tr>
<td>Multiplication</td>
<td>*</td>
<td>( b \times m )</td>
<td>( b \times m )</td>
</tr>
<tr>
<td>Division</td>
<td>/</td>
<td>( x / y )</td>
<td>( x / y )</td>
</tr>
<tr>
<td>Modulus</td>
<td>%</td>
<td>( r % s )</td>
<td>( r % s )</td>
</tr>
</tbody>
</table>

Rule of Operator Precedence

<table>
<thead>
<tr>
<th>Operator(s)</th>
<th>Operation(s)</th>
<th>Order of evaluation (precedence)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(</td>
<td>Parentheses</td>
<td>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.</td>
</tr>
<tr>
<td>* / or %</td>
<td>Multiplication Division</td>
<td>Evaluated second. If there are several, they are evaluated left to right.</td>
</tr>
<tr>
<td>+ -</td>
<td>Addition Subtraction</td>
<td>Evaluated last. If there are several, they are evaluated left to right.</td>
</tr>
</tbody>
</table>
# Decision Making: Equality and Relational Operators

- Executable statements
  - Perform actions (calculations, input/output of data)
  - Perform decisions
    - May want to print "pass" or "fail" given the value of a test grade
- **if** control structure
  - Simple version in this section, more detail later
  - If a condition is true, then the body of the **if** statement executed
    - 0 is false, non-zero is true

## Keywords
- Special words reserved for C
- Cannot be used as identifiers or variable names

## Relational Operators

| Relational Operators | Symbol | Example Condition | Meaning of Symbol
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Greater than</td>
<td>&gt;</td>
<td>x &gt; y</td>
<td>x is greater than y</td>
</tr>
<tr>
<td>Less than</td>
<td>&lt;</td>
<td>x &lt; y</td>
<td>x is less than y</td>
</tr>
<tr>
<td>Greater than or equal to</td>
<td>&gt;=</td>
<td>x &gt;= y</td>
<td>x is greater than or equal to y</td>
</tr>
<tr>
<td>Less than or equal to</td>
<td>&lt;=</td>
<td>x &lt;= y</td>
<td>x is less than or equal to y</td>
</tr>
</tbody>
</table>

## Equality Operators

<table>
<thead>
<tr>
<th>Equality Operators</th>
<th>Symbol</th>
<th>Example Condition</th>
<th>Meaning of Symbol</th>
</tr>
</thead>
<tbody>
<tr>
<td>Equal to</td>
<td>==</td>
<td>x == y</td>
<td>x is equal to y</td>
</tr>
<tr>
<td>Not equal to</td>
<td>!=</td>
<td>x != y</td>
<td>x is not equal to y</td>
</tr>
</tbody>
</table>

### Common Programming Error

- A syntax error will occur if the two symbols in any of the operators ==, !=, >= and <= are separated by spaces.

- A syntax error will occur if the two symbols in any of the operators !=, >=, and <= are reversed as in !=, =>, and =<, respectively.
Common Programming Error

- Confusing the equality operator == with the assignment operator =

2.6 Decision Making: Equality and Relational Operators (III)

<table>
<thead>
<tr>
<th>Keywords</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>auto</td>
<td>double</td>
<td>int</td>
</tr>
<tr>
<td>break</td>
<td>else</td>
<td>long</td>
</tr>
<tr>
<td>case</td>
<td>enum</td>
<td>register</td>
</tr>
<tr>
<td>char</td>
<td>extern</td>
<td>return</td>
</tr>
<tr>
<td>const</td>
<td>float</td>
<td>short</td>
</tr>
<tr>
<td>continue</td>
<td>for</td>
<td>signed</td>
</tr>
<tr>
<td>default</td>
<td>goto</td>
<td>sizeof</td>
</tr>
<tr>
<td>do</td>
<td>if</td>
<td>static</td>
</tr>
<tr>
<td>while</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

```c
#include <stdio.h>

int main()
{
    int num1, num2;
    printf("Enter two integers, and I will tell you
the relationships they satisfy: ");
    scanf("%d%d", &num1, &num2);

    if (num1 == num2)
        printf("%d is equal to %d\n", num1, num2);

    if (num1 != num2)
        printf("%d is not equal to %d\n", num1, num2);

    if (num1 < num2)
        printf("%d is less than %d\n", num1, num2);

    if (num1 > num2)
        printf("%d is greater than %d\n", num1, num2);

    if (num1 <= num2)
        printf("%d is less than or equal to %d\n", num1, num2);

    if (num1 >= num2)
        printf("%d is greater than or equal to %d\n", num1, num2);

    return 0;
}
```

---

Enter two integers, and I will tell you the relationships they satisfy: 3 7
3 is not equal to 7
3 is less than 7
3 is less than or equal to 7

Enter two integers, and I will tell you the relationships they satisfy: 22 12
22 is not equal to 12
22 is greater than 12
22 is greater than or equal to 12