Introduction to Programming
計算機程式
(應數系)

Lecture 4
Spring 2005
March 11, 2005

Topics
• Review of if statement
• The switch statement
• Repetition and Loop Statements
  – For-Loop
  – Condition-Loop
• Reading: Chap. 5.7~ Chap. 6
Review: if 叙述 (if-else statements)

- if-else 叙述的語法格式如下:

```c
if (expr)
{
    Statement1;
}
else
{
    Statement2;
}
```

- if-else 叙述的流程圖:

```
 expr
    ↓
  false
    ↓
Statement2
    ↓
true
    ↓
Statement1
```

Use Logical Operators

```c
int main()
{
    int num;
    printf("Please input a number between 1 to 100 : ");
    scanf("%d", &num);

    if (num >= 1) {
        if (num <= 100) // nested if
            printf("Valid number %d\n",num);
        else
            printf("Invalid number %d\n",num);
    }
    else
        printf("Invalid number %d\n",num);
    printf("Bye Bye !\n");
    system("PAUSE");
    return 0;
}
```

If ((num >= 1) && (num <= 100)) { …
NESTED IF with relational op

```c
if (( time >= 0) && (time < 12))
    printf("Good Morning");
else if ((time >= 12) && (time < 18))
    printf("Good Afternoon");
else if ((time >= 18) && (time < 24.))
    printf("Good Evening");
else printf("Time is out of range");
```

NESTED IF with logical op

```c
if ( x == 5 ) {
    if  ( y == 5 ) printf ( "Both are 5. \n ");
    else printf ( "x is 5, but y is not. \n ");
} else {
    if  ( y == 5 ) printf ( "y is 5, but x is not. \n ");
    else printf ( "Neither is 5. \n ");
}
```

Income Tax Example

<table>
<thead>
<tr>
<th>income</th>
<th>tax</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt; 15,000</td>
<td>0%</td>
</tr>
<tr>
<td>15,000, &lt; 30,000</td>
<td>18%</td>
</tr>
<tr>
<td>30,000, &lt; 50,000</td>
<td>22%</td>
</tr>
<tr>
<td>50,000, &lt; 100,000</td>
<td>28%</td>
</tr>
<tr>
<td>100,000</td>
<td>31%</td>
</tr>
</tbody>
</table>

```c
if ( income < 15000 ) {
    printf("No tax.");
}
if ( income >= 15000 && income < 30000 ) {
    printf("18% tax.");
}
if ( income >= 30000 && income < 50000 ) {
    printf("22% tax.");
}
if ( income >= 50000 && income < 100000 ) {
    printf("28% tax.");
}
if ( income >=100000 ) {
    printf("31% tax.");
}
```
Cascaded ifs

```c
if ( income < 15000 ) {
  printf( "No tax" );
} else {
  if ( income < 30000 ) {
    printf( "18%% tax." );
  } else if ( income < 50000 ) {
    printf( "22%% tax." );
  } else if ( income < 100000 ) {
    printf( "31%% tax." );
  } else {
    printf( "28%% tax." );
  }
}
Order is important.
Conditions are evaluated in order given.
```

DeMorgan’s Laws

- DeMorgan’s laws help determine when two complex conditions are equivalent
- They state:
  - ! ( P && Q ) is equivalent to ( !P || !Q )
  - ! ( P || Q ) is equivalent to ( !P && !Q )

- This applies for any Boolean expressions P and Q, which might themselves be complex expressions

```c
if ( ( age < 25 && sex == 'M' ) )
  printf ( "Cheap rates. \n" );
```
Gotcha! = versus ==

```c
int a = 2;
if ( a = 1 ) /* semantic (logic) error */
{
    printf("a is one\n");
}
else if ( a == 2 )
{
    printf("a is two\n");
}
else
{
    printf("a is %d", a);
}
```

(C Has No Boolean values)

Multiple Selection

- So far, we have only seen **binary selection**.

```c
if ( age >= 18 )
{
    printf("Vote!\n");
}
else
{
    printf("Maybe next time!\n");
}
```
演練：3個數字的排序流程圖

流程圖
巢狀 if：3 個數字的排序

程式 Ch05_10.c 輸入三個數，然後由大到小的順序輸出結果

```c
#include <stdio.h>

int main(void)
{
    int i,j,k;
    int bigger,smaller;
    int max,mid,min;
    printf("請依序輸入 3 個數字作排序\n");
    scanf("%d %d %d",&i,&j,&k);
```

巢狀 if：3 個數字的排序

```c
if (i>j) /* 第一個數與第二個數比較，較大者 */
    /* 為 bigger，較小者為 smaller */
    bigger = i;
    smaller = j;
else
    bigger = j;
    smaller = i;
```

```c
if (bigger>k) /* bigger 與第三個數比較 */
```
巢狀 if：3 個數字的排序

```c
24    max=bigger;    /* bigger 若比第三個數大，則為最大值 max */
25    if (k>smaller)    /* 第三個數再與 smaller 比較 */
26    {
27        mid = k;    /* 如第三個數比較大，則為第二大 */
28        min = smaller;    /* 值 mid, smaller 爲最小值 min */
29    }
30    else
31    {
32        mid = smaller;    /* 如第三個數比較小，smaller 爲第 */
33        min = k;    /* 二大值 mid, 第三個數為最小值 min */
34    }
35    }
36    else
37    {
```

```c
38    max = k;        /* 如 bigger 比第三個數小，則第 */
39    mid = bigger;    /* 三數為最大值 max, bigger 爲 */
40    min = smaller;    /* 次大 mid, smaller 爲最小值 min */
41    }
42    printf("由大到小排列 %d %d %d\n", max, mid, min);
43 44    return 0;
45    }
```

巢狀 if：3 個數字的排序
Multiple Selection (con’t)

• Sometimes it is necessary to **branch** in more than two directions.
• We do this via **multiple selection**.
• The multiple selection mechanism in C is the **switch** statement.
Multiple Selection with if

```c
if (day == 0) {
    printf("Sunday") ;
}
if (day == 1) {
    printf("Monday") ;
}
if (day == 2) {
    printf("Tuesday") ;
}
if (day == 3) {
    printf("Wednesday") ;
}
if (day == 4) {
    printf("Thursday") ;
}
if (day == 5) {
    printf("Friday") ;
}
if (day == 6) {
    printf("Saturday") ;
}
if ((day < 0) || (day > 6)) {
    printf("Error - invalid day.
") ;
}
```

(continued)

Multiple Selection with if-else

```c
if (day == 0) {
    printf("Sunday") ;
} else if (day == 1) {
    printf("Monday") ;
} else if (day == 2) {
    printf("Tuesday") ;
} else if (day == 3) {
    printf("Wednesday") ;
} else if (day == 4) {
    printf("Thursday") ;
} else if (day == 5) {
    printf("Friday") ;
} else if (day == 6) {
    printf("Saturday") ;
} else {
    printf("Error - invalid day.
") ;
}
```

This if-else structure is more efficient than the corresponding if structure. Why?
The **switch** Multiple-Selection Structure

```
switch ( integer expression )
{
    case constant1 :
        statement(s)
        break;
    case constant2 :
        statement(s)
        break;
    ...
    default:
        statement(s)
        break;
}
```

**switch Example**

```
switch ( day )
{
    case 0:  printf ("Sunday\n") ; break ;
    case 1:  printf ("Monday\n") ; break ;
    case 2:  printf ("Tuesday\n") ; break ;
    case 3:  printf ("Wednesday\n") ; break ;
    case 4:  printf ("Thursday\n") ; break ;
    case 5:  printf ("Friday\n") ; break ;
    case 6:  printf ("Saturday\n") ; break ;
    default: printf ("Error -- invalid day.\n") ; break ;
}
```

Is this structure more efficient than the equivalent nested if-else structure?
switch Statement Details

- The last statement of each case in the switch should *almost always* be a break.
- The break causes program control to jump to the closing brace of the switch structure.
- Without the break, the code flows into the next case. This is almost never what you want.
- A switch statement will compile *without a default case*, but always consider using one.
- Include a **default** case to catch invalid data.
- Inform the user of the type of error that has occurred (e.g., “Error - invalid day.”).

A Different Switch Example

```c
#include <stdio.h>
int main(void) {
    int mo;
    scanf("%d", &mo); /* get month number from user */
    switch (mo) {
        case 2: printf("28 day month"); break;
        case 4: /* 30 days hath sep, apr, jun, nov */
            case 6:
            case 9:
            case 11: printf("30 day month"); break;
        default: printf("31 day month"); break;
    }
    return 0;
}
```
Why Use a switch Statement?

• A nested if-else structure is just as efficient as a switch statement.
• However, a switch statement may be easier to read.
• Also, it is easier to add new cases to a switch statement than to a nested if-else structure.

The char Data Type

• The char data type holds a single character.
  
  ```
  char ch;
  ```
  
• Example assignments:
  
  ```
  char grade, symbol;
  grade = 'B';
  symbol = '$';
  ```

• The char is held as a one-byte integer in memory. The ASCII code is what is actually stored, so we can use them as characters or integers, depending on our need.
The **char** Data Type (con’t)

- Use
  
  ```c
  scanf ("%c", &ch) ;
  ```

  to read a single character into the variable `ch`. (Note that the variable does not have to be called “ch”.)

- Use
  
  ```c
  printf("%c", ch) ;
  ```

  to display the value of a character variable.

---

**char Example**

```c
#include <stdio.h>
int main ( )
{
    char ch ;

    printf("Enter a character: ");
    scanf("%c", &ch);
    printf("The value of %c is %d\n", ch, ch);
    return 0;
}
```

If the user entered an `A`, the output would be:

```
The value of A is 65.
```
switch statement with chars

Example:

```c
switch (c) {
    case '+': i = i+2; break;
    case '-': i = i-2; break;
    case '*': i = i*2; break;
    case '/': i = i/2; break;
    default: i = 0;
}
```

switch statement ---- 範例

```c
int main()
{
    char grade;
    printf("Input grade : ");
    scanf("%c", &grade);
    switch (grade) {
    case 'a'
    case 'A': printf("Excellent ! \n"); break; /* 輸入A或A時*/
    case 'b'
    case 'B': printf("Good ! \n"); break; /* 輸入B或B時*/
    case 'c'
    case 'C': printf("Be study hard ! \n"); break; /* 輸入C或C時*/
    default: printf("Failed\n"); /* 輸入其他字元時*/
    }
    return 0;
}
```
**switch statement**

範例2:

電力公司的電費計算方式分成三類：
(1) 家庭用電：0-300度，每度3.3元；
    301度(含)以上每度4.2元。
(2) 工業用電：基本費為每一契約馬力150元，
    實際用電費每度1.9元。
(3) 營業用電：0-300度，每度6元；
    301度(含)以上每度6.8元。

輸入用電類別及使用度數後，計算應繳電費為何？
switch statement

範例 2 程式碼 (1):

```c
int T;
float Deg;
floa C;
float Fee;

// 用電類別 */
// 用電度數 */
// 工業用電契約馬力 */
// 儲費 */

printf("1. 家用用電 ");
printf("a");
printf("2. 工業用電 ");
printf("a");
printf("3. 宿舍用電 ");
printf("a");
printf(" 請輸入用電類別(1-3): ");
scanf("%d", &T);

if (T>=1 && T<=3) { // 判別輸入是否正確
    printf("用電度數 = ");
    scanf("%f", &Deg);

    switch(T) {
    case 1: if (Deg<=100)
        Fee = Deg*3.3;
    else
        Fee = (Deg-300)*4.2 + 300*3.3;
    break;
    case 2: printf("契約馬力 = ");
        scanf("%f", &C);
        Fee = C*150 + Deg*1.9;
    break;
    case 3: if (Deg<=300)
        Fee = Deg*6;
    else
        Fee = (Deg-300)*6.8 + 300*6;
    break;
    }
    printf("契約費 = ");
    printf("%.2f", Fee);
    printf("元");
}
```

Ch2_1_2.c

switch statement

範例 2 程式碼：

```c
switch(T) {
    case 1: if (Deg<=100)
        Fee = Deg*3.3;
    else
        Fee = (Deg-300)*4.2 + 300*3.3;
    break;
    case 2: printf("契約馬力 = ");
        scanf("%f", &C);
        Fee = C*150 + Deg*1.9;
    break;
    case 3: if (Deg<=300)
        Fee = Deg*6;
    else
        Fee = (Deg-300)*6.8 + 300*6;
    break;
}
```
switch statement

範例2 程式碼：

```c
switch(Fee)
{
    case 2:
        printf("%f", Fee);
        break;
    default:
        printf("\n");
}
```

綜合演練

- if 與 switch 的綜合應用：星座對照表

流程圖
if 與 switch 的綜合應用：
星座對照表

程式 Ch05_09.c 用生日查星座

01 #include <stdio.h>
02 03 int main(void)
04 { 05 int month, day;
06 07 printf("請輸入您的生日 \n");
08 printf("幾月 \n");
09 scanf("%d", &month);
10 printf("幾日 \n");
11 scanf("%d", &day);

if 與 switch 的綜合應用：
星座對照表

13 if (day<0) /* 檢查輸入的日期，不合理的日期就將月份設為 13，*/
14 month=13; /* 警告列的程式會輸出生辰差異 */
15 if (month==2 & day>29) /* 2 月不能超過 29 天 */
16 month=13;
17 else if ((month==4 | month==6 | month==9 | month==11) &
18 day>30) /* 4, 6, 9, 11 月不能超過 30 天 */
19 month=13;
20 else if (day>31) /* 其它月份不能超過 31 天 */
21 month=13;
22 23 switch (month) /* 以出生月份作為條件變數 */
24 { 25 case 1: /* 生在第一月 */
26 if (day < 20)
27 printf("魔羯座 \n");
28 else
if 與 switch 的綜合應用：
星座對照表

```c
29  printf("水瓶座\n");
30  break;
31  case 2: /* 生日在二月份 */
32      if (day < 20)
33          printf("水瓶座\n");
34      else
35          printf("雙魚座\n");36
37          break;
38  case 3: /* 生日在三月份 */
39      if (day < 21)
40          printf("雙魚座\n");
41      else
42          printf("白羊座\n");
43          break;
44  case 4: /* 生日在四月份 */
45      if (day < 20)
46          printf("白羊座\n");
47      else
48          printf("金牛座\n");
49          break;
50  case 5: /* 生日在五月份 */
51      if (day < 21)
52          printf("金牛座\n");
53      else
54          printf("雙子座\n");
55          break;
56  case 6: /* 生日在六月份 */
57      if (day < 20)
58          printf("雙子座\n");
```
if 與 switch 的綜合應用：
星座對照表

```c
58     else
59         printf("巨蟹座\n");
60         break;
61     case 7:    /* 生日在七月份 */
62         if (day < 22)
63             printf("巨蟹座\n");
64         else
65             printf("獅子座\n");
66         break;
67     case 8:    /* 生日在八月份 */
68         if (day < 23)
69             printf("獅子座\n");
70         else
71             printf("處女座\n");
72         break;
73     case 9:    /* 生日在九月份 */
74     if (day < 23)
75         printf("處女座\n");
76     else
77         printf("天秤座\n");
78     break;
79     case 10:    /* 生日在十月份 */
80         if (day < 23)
81             printf("天秤座\n");
82         else
83             printf("天蠍座\n");
84         break;
85     case 11:    /* 生日在十一月份 */
86         if (day < 22)
87             printf("天蠍座\n");
```
if 與 switch 的綜合應用：
星座對照表

```c
else
    printf("射手座 \n");
    break;

case 12:    /* 生日在十二月份 */
    if (day < 22)
        printf("射手座 \n");
    else
        printf("魔羯座 \n");
    break;

default:    
    printf("你是外星人嗎？\n");
```

執行結果

請輸入您的生日
幾月
4
幾日
19
白羊座
Repetition Statements

- 重複敘述 (Repetition statements, Loop, Iteration): (有條件地)反覆執行指定的敘述
- C 語言的 Loop statements 包含
  - for loop: 計次型 (Counting loop)
  - while loop: 條件型 (Condition loop)
  - do-while loop: 條件型 (Condition loop)

- Reading: Chap. 5

while loop

- Syntax for while loop:

```java
while ( expr )
{
    statements;
}
next-statements;
```

```java
children = 7; cookies = 1;
while ( children > 0 )
{
    children = children - 1;
    cookies = cookies * 2;
}
```
**while loop**

- Flow chart for while loop

Note: C has no boolean value; C uses int for boolean values—0 for false and non-zero for true

**while Loop: Example 1**

- **Problem**: Write a program that calculates the *average exam grade* for a class of 10 (or *n*) students.
- **What are the program inputs?**
  - the exam grades
- **What are the program outputs?**
  - the average exam grade
The Pseudocode

<total> = 0
<grade_counter> = 1 // 紀錄學生數

While (<grade_counter> <= 10)
    Display “Enter a grade: ”
    Read <grade>
    <total> = <total> + <grade>
    <grade_counter> = <grade_counter> + 1
End_while
<average> = <total> / 10
Display “Class average is: “, <average>

The C Code

#include <stdio.h>
int main ( )
{
    int counter, grade, total, average ;
    total = 0 ;
    counter = 1 ;
    while ( counter <= 10 )
    {
        printf ("Enter a grade : ");
        scanf ("%d", &grade);
        total = total + grade ;
        counter = counter + 1 ;
    }
    average = total / 10 ;
    printf ("Class average is: %d\n", average) ;
    return 0 ;
}
Versatile?

• How versatile is this program?
• *It only works with class sizes of 10.*
• We would like it to work with any class size.
• A better way:
  – Ask the user how many students are in the class. Use that number in the condition of the while loop and when computing the average.

New Pseudocode

<total> = 0
<grade_counter> = 1

Display “Enter the number of students: “
Read <num_students>
While (<grade_counter> <= <num_students>)
  Display “Enter a grade: “
  Read <grade>
  <total> = <total> + <grade>
  <grade_counter> = <grade_counter> + 1
End_while
<average> = <total> / <num_students>
Display “Class average is: “, <average>
#include <stdio.h>

```c
int main ( )
{
    int numStudents, counter, grade, total, average;
    total = 0;
    counter = 1;
    printf ("Enter the number of students: ");
    scanf ("%d", &numStudents);
    while ( counter <= numStudents ) {
        printf ("Enter a grade : ");
        scanf ("%d", &grade);
        total = total + grade;
        counter = counter + 1;
    }
    average = total / numStudents;
    printf ("Class average is: %dn", average);
    return 0;
}
```

Why Bother to Make It Easier?

- Why do we write programs?
  - So the user can perform some task
- The more versatile the program, the more difficult it is to write. BUT it is more useable.
- The more complex the task, the more difficult it is to write. But that is often what a user needs.
- Always consider the user first.
Using a **Sentinel Value**

- We could let the user keep entering grades and when he’s done enter some special value that signals us that he’s done.
- This special signal value is called a **sentinel value**.
- We have to make sure that the value we choose as the sentinel isn’t a legal value. For example, we can’t use 0 as the sentinel in our example as it is a legal value for an exam score.

The Priming Read

- When we use a sentinel value to control a while loop, we have to get the first value from the user before we encounter the loop so that it will be tested and the loop can be entered.
- This is known as a **priming read**.
- We have to give significant thought to the initialization of variables, the sentinel value, and getting into the loop.
New Pseudocode

<total> = 0
<grade_counter> = 1
Display “Enter a grade: “
Read <grade>
While (<grade> != SENTINEL)
  <total> = <total> + <grade>
  <grade_counter> = <grade_counter> + 1
  Display “Enter another grade: ”
  Read <grade>
End_while
<average> = <total> / <grade_counter>
Display "Class average is: ", <average>

New C Code

#include <stdio.h>
#define SENTINEL -99
int main ()
{
    int counter, grade, total, average;
    total = 0;
    counter = 1;
    printf("Enter a grade: ");
    scanf("%d", &grade);
    while (grade != SENTINEL) {
        total = total + grade;
        counter = counter + 1;
        printf("Enter another grade: ");
        scanf("%d", &grade);
    }
    average = total / counter;
    printf("Class average is: %d\n", average);
    return 0;
}
#include <stdio.h>
#define SENTINEL -99

int main ( )
{
    int counter;     /* counts number of grades entered */
    int grade;        /* individual grade */
    int total;           /* total of all grades */
    int average;    /* average grade */

    /* Initializations */
    total = 0;
    counter = 1;

    /* Get grades from user */
    /* Compute grade total and number of grades */
    printf("Enter a grade: ");
    scanf("%d", &grade);
    while (grade != SENTINEL) {
        total = total + grade;
        counter = counter + 1;
        printf("Enter another grade (or %d to quit): ", SENTINEL);
        scanf("%d", &grade);
    }

    /* Compute and display the average grade */
    average = total / counter;
    printf("Class average is: %d\n", average);

    return 0;
}
while loop: Example 2

Example 2 Flow chart:

Get the greatest common divisor (gcd) of two numbers

\[ \text{gcd}(i, j) = \text{gcd}(j, i \% j) \]

while loop: gcd example

```c
#include<stdio.h>

void main()
{
    int i, j, tmp;
    printf("1: Please input two positive integer for finding their GCD:\n");
    scanf("%d %d", &i, &j);
    while ( j != 0 )
    {
        tmp = i % j;
        i = j;
        j = tmp;
    }
    printf("2: The GCD is %d \n",i);
}
```
Using a while Loop to Check User Input

```c
#include <stdio.h>
int main ()
{
    int number;
    printf ("Enter a positive integer : ");
    scanf ("%d", &number);
    while ( number <= 0 )
    {
        printf ("That's incorrect. Try again.\n");
        printf ("Enter a positive integer: ");
        scanf ("%d", &number);
    }
    printf ("You entered: %d\n", number);
    return 0;
}
```

The **do-while** Repetition Structure

```c
do
{
    statement(s)
} while ( condition ) ;
```

• The body of a **do-while** is *ALWAYS executed at least once*. 
Check User Input Using do-while

do
{
    printf("Enter a positive number: ");
    scanf("%d", &num);
    if ( num <= 0 )
    {
        printf("That is not positive.  Try again\n");
    }
} while ( num <= 0 );

Counter-Controlled loop: for loop*

*但也可作爲 condition-controlled loop
Counter-Controlled Repetition (Definite Repetition)

• If it is known in advance exactly how many times a loop will execute, it is known as a counter-controlled loop.

```c
int i = 1;
while ( i <= 10 )
{
    printf("i = %d\n", i);
    i = i + 1;
}
```

Counter-Controlled Repetition (con’t)

• Is the following loop a counter-controlled loop?

```c
while ( x != y )
{
    printf("x = %d", x);
    x = x + 2;
}
```
Event-Controlled Repetition
(Indefinite Repetition)

• If it is NOT known in advance exactly how many
times a loop will execute, it is known as an event-
controlled loop or condition-controlled loop.

```c
sum = 0;
printf("Enter an integer value: ");
scanf("%d", &value);
while (value != -1) {
    sum = sum + value;
    printf("Enter another value: ");
    scanf("%d", &value);
}
```

Event-Controlled Repetition (con’t)

• An event-controlled loop will terminate
when some event occurs (which makes
the condition true).

• The event may be the occurrence of a
sentinel value, as in the previous example.

• There are other types of events that may
occur, such as reaching the end of a data file.
The 3 Parts of a Loop

```c
#include <stdio.h>
int main ()
{
    int i = 1; /* count from 1 to 100 */
    while ( i < 101 )
    {
        printf( "%d ", i );
        i = i + 1;  /* test of loop termination condition */
    }
    return 0;
}
```

The `for` Loop Repetition Structure

- The `for` loop handles details of the counter-controlled loop “automatically”.
- The initialization of the loop control variable, the termination condition test, and control variable modification are handled in the `for` loop structure.

```c
for ( i = 1; i < 101; i = i + 1 )
{
    initialization
    test
    modification
}
```
for迴圈 (for loop)

- 運算式 1 (expr1) 為迴圈控制變數的初值。 (initialize)
- 運算式 2 (expr 2) 為迴圈是否重複執行的條件。 (test)
- 運算式 3 (expr 3) 為迴圈控制變數的修正值。 (modify)

```
for (expr1; expr2; expr3)
{
    statements;
}
next-statements;
```

❖ 重覆敘述--for迴圈(for loop)流程圖

- for迴圈的流程圖
When Does a *for-loop* Initialize, Test and Modify?

- Just as with a while loop, a for loop
  - *initializes* the loop control variable *before* beginning the first loop iteration,
  - *modifies* the loop control variable at the very *end* of each iteration of the loop, and
  - performs the loop *termination test before* each iteration of the loop.

- The for loop is easier to write and read for counter-controlled loops.

### Various forms of *for* loop counting

```c
for ( i = 0;  i < 10;  i = i + 1 )
{
    printf(“%d\n”, i ) ;
}

for ( i = 9;  i >= 0;  i = i - 1 )
{
    printf(“%d\n”, i ) ;
}

for ( i = 0;  i < 10; i = i + 2 )
{
    printf(“%d\n”, i ) ;
}

for ( i = 9;  i >= 0;  i = i - 2 )
{
    printf(“%d\n”, i ) ;
}
```
for loop---範例

範例1：計算1累加至100的總和，並輸出其總和

\[
\sum_{k=1}^{100} k
\]

```c
#include<stdio.h>
main()
{
    int i, sum=0;                               /*宣告變數i、sum為整數*/
    for (i=1;  i<=100;  i=i+1)
    {
        sum= sum+ i;
    }
    printf("sum = %d 
", sum);       /*週圈結束時，印出sum的值*/
}
```
for loop---範例2

範例2 程式碼：

```c
#include<stdio.h>
main()
{
    int i , N , sum=0;
    printf("Please input an integer: ");                     /*由鍵盤輸入數值*/
    scanf("%d", &N);
    for (i=1; i<=N; i=i+1)
    {
        sum= sum+ i;                                             /*計算1+2+…+N*/
    }
    printf("1+2+…+%d = %d 
",N, sum);       /*輸出sum的值*/
}
```

for loop---範例3

範例3： f(n)=n*(n-1)*(n-2)*(n-3)…*3*2*1 --階乘函數Factorial

```c
/*
* Computes n!
* Pre-condition: n is greater than or equal to zero
*/
Int factorial(int n)
{
    int i,         /* local variables */
        product;   /* accumulator for product computation */
    product = 1;
    /* Computes the product n x (n-1) x (n-2) x ... x 2 x 1 */
    for (i = n; i >= 1; --i) {
        product = product * i;
    }
    /* Returns function result */
    return (product);
}
```
The increment operator ++

The decrement operator --

If we want to add one to a variable, we can say:

```
count = count + 1;  // (count = count - 1;
```

Programs often contain statements that increment variables, so to save on typing, C provides these shortcuts:

```
count++;  // (count--;  OR  ++count;  // (--count;
```

Both do the same thing. They change the value of count by adding one to it.

*But post or pre increment (decrement) have different implications. (come back later)*

---

Nested Loops

Loops inside other loops!
Nested for loops--范例

範例： 九九乘法表（雙層迴圈的應用）

```c
#include<stdio.h>
main()
{
    int i, j, k;
    for (i=1; i<=9; i++)
    {
        for (j=1; j<=9; j++)
        {
            k = i*j;
            printf("%d*%d = %2d ", i, j, k);
        }
    }
}
```

Nested for loop---範例

執行結果：

```
1*1= 1
2*1= 2
3*1= 3
4*1= 4
5*1= 5
6*1= 6
7*1= 7
8*1= 8
9*1= 9
2*2= 4
2*3= 6
2*4= 8
2*5=10
2*6=12
2*7=14
2*8=16
2*9=18
3*3= 9
3*4=12
3*5=15
3*6=18
3*7=21
3*8=24
3*9=27
4*4=16
4*5=20
4*6=24
4*7=28
4*8=32
4*9=36
5*5=25
5*6=30
5*7=35
5*8=40
5*9=45
6*6=36
6*7=42
6*8=48
6*9=54
7*7=49
7*8=56
7*9=63
8*8=64
8*9=72
9*9=81
```

若要將表格排列如下，應如何修改？

```
1*1= 1
2*2= 4
3*3= 9
4*4=16
5*5=25
6*6=36
7*7=49
8*8=64
9*9=81
```

42
Revised Nested for loops--範例
範例：九九乘法表（雙層迴圈的應用）

```c
#include<stdio.h>
main()
{
    int i, j, k;
    for (i=1; i<=9; i++)
    {
        for (j=1; j<=9; j++)
        {
            k = i*j;
            printf("%d*%d = %2d ", i, j, k);
        }
        printf("\n");
    }
}
```

Nested for loops--範例

```c
for ( i = 1; i < 5; i = i + 1 )
{
    for ( j = 1; j < 5; j = j + 1 )
    {
        if ( j % 2 == 0 )
        {
            printf("O") ;
        }
        else
        {
            printf("X") ;
        }
    }
    printf("\n");
}
```

How many times is the “if” statement executed?
What is the output?

XOXO
XOXO
XOXO
XOXO
Nested for loops—範例變形

```c
for ( i = 1; i < 7; i = i + 1 )
{
    for ( j = i; j < 7; j = j + 1 )
    {
        if ( j % 2 == 0 )
            printf ("O") ;
        else
            printf ("X") ;
    }
    printf ("\n") ;
}
```

How many times is the “if” statement executed?

What is the output?

```
XOXOXO
OXOXO
XOXO
XO
O
```

Figure 5.14
Validating Input Using nested do-while statement
Nested for loop---Lab 練習

練習：請用巢狀for-loop 列出下列的圖形

```plaintext
#
###
####
#####
######
#######
########
#########
```
// 8 rows
```
// 15 #
```

Assignment 2

1. 前面nested for loops
2. 日期日計算
   1. 輸入月日
   2. 計算那是當年(2005)的第幾日