CHAPTER 4 - PROGRAM CONTROL

4.1 Introduction

- 這個章節主要介紹另外幾個表示選擇與反覆控制結構的語法
  - 反覆控制結構
    - **for**
    - **do/while**
  - **switch** 多重選擇結構
  - **break**
  - **continue**

4.2 The Essentials of Repetition

- 控制迴圈反覆的次數基本有兩種方法
  - 若反覆次數是確定的就使用變數來計數反覆的次數。

```
int u = 0;
while (u < n) {
    u++;
}
```

  - 若反覆次數是不確定的，就要設計某個條件，當條件滿足(或不滿足)時，迴圈就停止。

```
int a = 0, b = 0;
while (scanf("%d“,&a) == 1) {
    //…
}
```

4.3 Essentials of Counter-Controlled Repetition

- 計數器控制的反覆結構(Counter-controlled repetition)
  - 使用變數來計數迴圈次數。

- 完成計數器控制的反覆結構需要下面設計

  (少了其中一個，你設計的程式就會不正常)
  - 必須有迴圈計數器變數(loop counter)
  - 迴圈計數器變數必須設定初值 (**initialization**)
  - 根據迴圈計數器變數的值測試迴圈結束的條件 (**repetition condition**)
  - 迴圈每執行一次迴圈計數器的更動方式
    - 增加或減少一個數值(**increment or decrement**)

4.3 Essentials of Counter-Controlled Repetition

- Example:

```
int counter = 1;            /* initialization */
while ( counter <= 10 ) { /* repetition condition */
   printf( "%d\n", counter );
   ++counter;               /* increment */
}
```

- `int counter = 1;`
  - 宣告迴圈計數器變數為整數型態
    - 若是宣告為浮點數呢？理論上是可行，但實際上會因為精確度關係，造成與欲執行次數有出入。

    例如：

    float counter = 0; int x = 0;

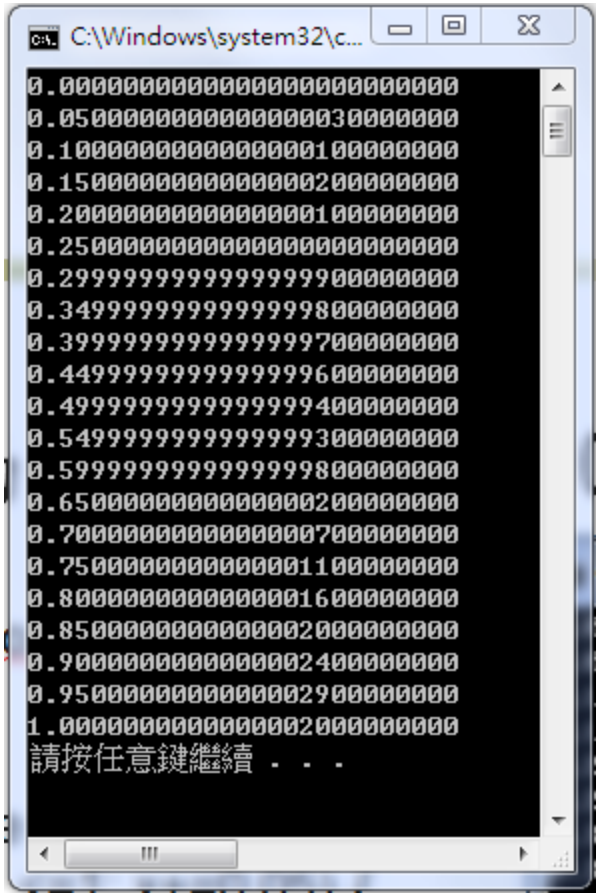
    while(counter<=1.) { x++; counter += 0.1; }
  - 設定初始值

範例

- 下面程式片段正確輸出0,0.05,0.1,0.15,…,1等21個數字嗎??

```
#include<stdio.h>
int main()
{
    long double x = 0;
    for(x = 0; x <=1; x+=0.05) {
        printf("%.25Lf\n",x);
    }
    printf("%.25Lf\n",x);
    return 0;
}
```

```
int u;
for(u=0,x=0; u<21; ++u,x+=0.05) {
    printf("%.25Lf\n",x);
}
for(u=0; u<=100; u+=5) {
    printf("%.25Lf\n",u/(long double)100.);
}
```



The while statement

- 語法

  while (expression) statement;

  while (expression) { statements }

```
//列印0,1,2,3,4,5
i=0;
while (i <= 5) {
    printf("%d\n",i);
    i++;
}
```

迴圈初始狀況

迴圈持續條件

迴圈每次疊代必須改變某些變數狀態，不然迴圈不會終止

```
while(1) { statements } // 無窮迴圈
```

Program: print a table of squares using a while-loop

```
#include <stdio.h>

int main(void)
{
  int i, n;

  printf("This program prints a table of squares.\n");
  printf("Enter number of entries in table: ");
  scanf("%d", &n);

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

  return 0;
}
```

Program: sum a series of numbers

```
#include <stdio.h>

int main(void)
{
  int n, sum = 0;

  printf("This program sums a series of integers.\n");
  printf("Enter integers (0 to terminate): ");

  scanf("%d", &n);
  while (n != 0) {
    sum += n;
    scanf("%d", &n);
  }
  printf("The sum is: %d\n", sum);

  return 0;
}
```

4.4 The `for` Repetition Structure

- 語法

  for (expr1; expr2; expr3) { statements }

迴圈初始敘述運算式

迴圈持續條件敘述運算式

迴圈狀態改變運算式

例子：

```
for(i=10; i > 0; i--) printf("%d\n", i);
```

- 一般可用while-loop改寫為

```
expr1;
while (expr2) {
        statements
        expr3;
}
```

The for statement (cont'd)

- for-loop其結構非常適合用來寫計數的迴圈
- 下面迴圈會疊代一樣多的次數
  - for(i = 0; i < n; i++) ..
  - for(i = 1; i <= n; i++) ..
  - for(i = n-1; i >=0; i--) ..
  - for(i = n; i >0; i--) ..

Omitting expression in a for statement

- for (expr1; expr2; expr3) expr1, expr2, expr3可省略(不一定要全寫)，分號不可省略 。
  - 例子一
    ```
    i=10;
    for(; i > 0; --i) printf("%d\n", i);
    ```
  - 例子二
    ```
    i=10;
    for(; i > 0;) printf("%d\n", --i);
    ```
  - 例子三: 無窮迴圈
    ```
    for( ; ; ) { statements }
    ```

The comma operator

- 在for-loop裡，我們可能需要在初始運算式裡寫超過一個運算式來初始化一些變數。這時，可以使用逗點隔開他們。

  例子：計算10!

  ```
  for(n = 1, i =1; i<=10; ++i) n *= i;
  ```

  當然也可以寫成

  ```
  n=1;
  for(i =1; i<=10; ++i) n *= i;
  ```

- 其實，在C語言裡運算式都可以用逗點隔開，而整體運算式的值是以最後一個運算式為準。

  ```
  i=10+1,12+2,13+3; // i is 16 now.
  ```

Program: print a table of squares without multiplication

```c
#include <stdio.h>

int main(void)
{
  int i, n, odd, square;

  printf("This program prints a table of squares.\n");
  printf("Enter number of entries in table: ");
  scanf("%d", &n);

  i = 1;
  odd = 3;
  for (square = 1; i <= n; odd += 2) {
    printf("%10d%10d\n", i, square);
    ++i;
    square += odd;
  }

  return 0;
}
```

```c
/* A straightforward method of printing a table of squares */
#include <stdio.h>
int main(void)
{
  int i, n;

  printf("This program prints a table of squares.\n");
  printf("Enter number of entries in table: ");
  scanf("%d", &n);

  for (i = 1; i <= n; i++)
    printf("%10d%10d\n", i, i * i);

  return 0;
}
```

i	1	2	3	4	5	6	7
odd	3	5	7	9	11	13	15
square	1	4	9	16	25	36	49

$$(i+1)^2 = i^2 + (2i+1)$$

```
(next) square= (current) square + (current) odd
(next) odd     = (current) odd + 2
```

The null statement

- 只有;的敘述也可以喔！

  i = 0; ; j = 1; //雖然第二個;前空無一物，但仍是對的。

- 要注意

```
for(i = 0; i < n; ++i)
  printf("%d\n", i);
```

與

```
for(i = 0; i < n; ++i); //這是一個for-loop，但無任何敘述在loop-body
  printf("%d\n", i);//這是一個在for-loop外的printf
```

Examples

- Calculate 1+2+…+n

```
#include<stdio.h>
int main()
{
    int n,s,i;

    printf("input n:");
    scanf("%d",&n);
    for(i =1,s=0; i <= n; ++i) {
       s+= i;
    }
    printf("1+...+%d=%d\n",n,s);
}
```

```
#include<stdio.h>
int main()
{
    int n;

    printf("input n:");
    scanf("%d",&n);
    printf("1+...+%d=%d\n",n,(1+n)*n/2);
}
```

下面運算如何用loop完成?　　當然也可以用公式算sum=n*(n+1)/2

$$sum = 1+2+3+4+\ldots\ldots+(n-1)+n$$

可以這麼算

$$sum = 1+2+3+4+\ldots\ldots+(n-1)+n$$

```
sum = 1; i = 2;
while(i<=n) { sum += i; i++; }
```

也可以這麼算

$$sum = 1+2+3+4+\ldots\ldots+(n-1)+n$$

```
sum = n; i = n-1;
while(i>=1) { sum += i; i--; }
```

還可以這麼算

$$sum = 1+2+3+4+\ldots\ldots+(n-1)+n$$

以後再說

Examples

- Calculate 1-2+3-4+…n

```
#include<stdio.h>
int main()
{
    int n,s,i;

    printf("input n:");
    scanf("%d",&n);
    for(i =1,s=0; i <= n; ++i) {
      s+= (i%2)?i:-i;
    }
    printf("1-2+3...%d=%d\n",n,s);
}
```

Analytical formula??

Examples

- Calculate $1^2+2^2+\ldots+n^2$

```
#include<stdio.h>
int main()
{
    int n,s,i;

    printf("input n:");
    scanf("%d",&n);
    for(i =1,s=0; i <= n; ++i) {
       s+= i*i;
    }
    printf("1+...+%d=%d\n",n,s);
}
```

```
//Analytical formula
#include<stdio.h>
int main()
{
    int n;

    printf("input n:");
    scanf("%d",&n);
    printf("1^1+2^2+...+%d^2=%d\n",n,
                    n*(n+1)*(2*n+1)/6);
}
```

- Calculate $1^3+2^3+\ldots+n^3$

4.7 The `switch` Multiple-Selection Structure

- 語法

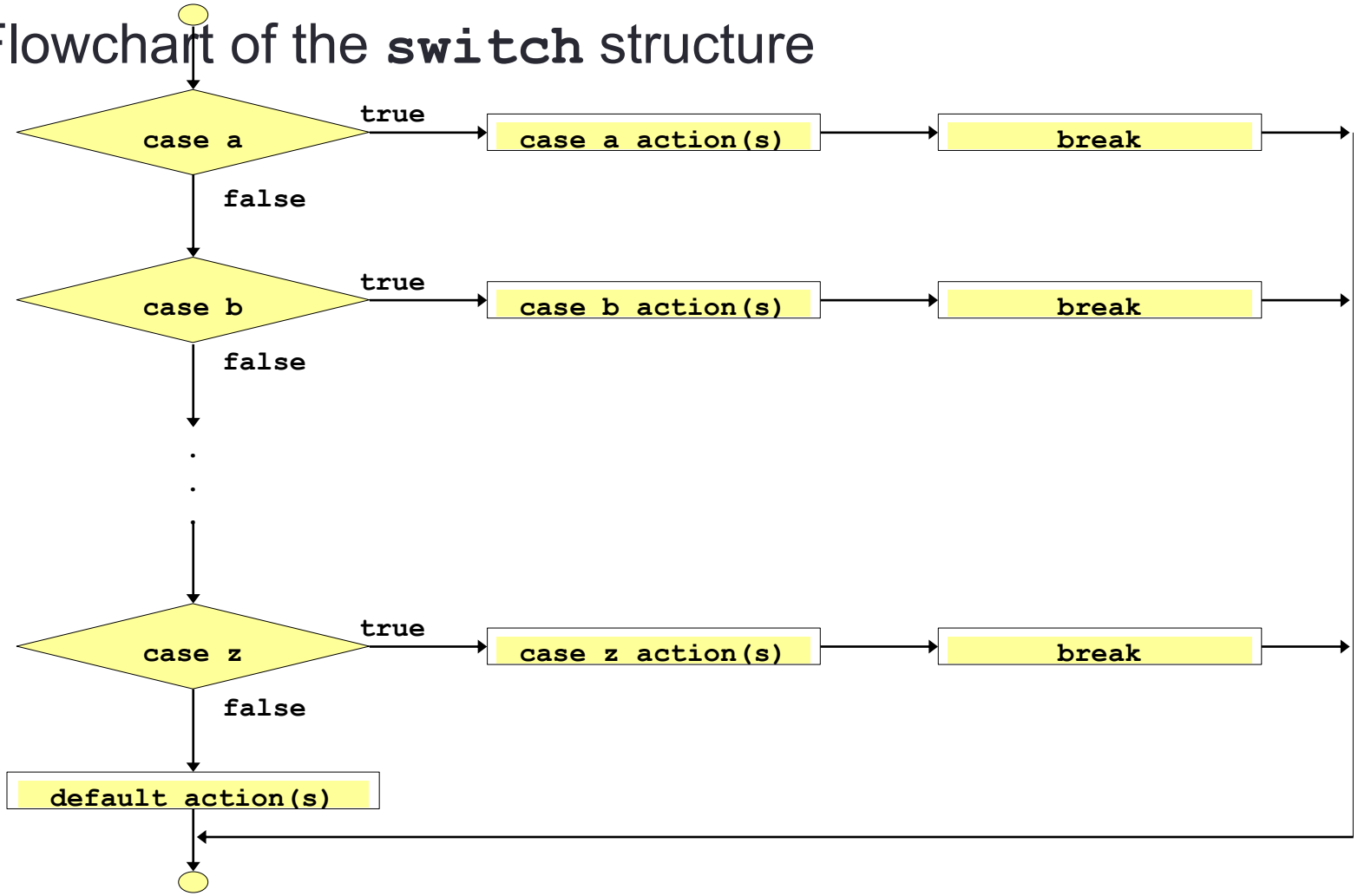
```
switch ( value ){
  case '1':
    actions
  case '2':
    actions
  default:
    actions
  }
```

都不是case裡的條件

- **`break;`** exits from structure

4.7 The `switch` Multiple-Selection Structure

- Flowchart of the **`switch`** structure



```c
/* Fig. 4.7: fig04_07.c
   Counting letter grades */
#include <stdio.h>

int main()
{
   int grade;
   int aCount = 0, bCount = 0, cCount = 0,
       dCount = 0, fCount = 0;

   printf(  "Enter the letter grades.\n"  );
   printf(  "Enter the EOF character to end input.\n"  );

   while ( ( grade = getchar() ) != EOF ) {

      switch ( grade ) {    /* switch nested in while */

         case 'A': case 'a':  /* grade was uppercase A */
            ++aCount;         /* or lowercase a */
            break;

         case 'B': case 'b':  /* grade was uppercase B */
            ++bCount;         /* or lowercase b */
            break;

         case 'C': case 'c':  /* grade was uppercase C */
            ++cCount;         /* or lowercase c */
            break;

         case 'D': case 'd':  /* grade was uppercase D */
            ++dCount;         /* or lowercase d */
            break;
```

1. Initialize variables

2. Input data

2.1 Use switch loop to update count

```
33
34          case 'F': case 'f':  /* grade was uppercase F */
35             ++fCount;         /* or lowercase f */
36             break;
37
38          case '\n': case' ':  /* ignore these in input */
39             break;
40
41          default:        /* catch all other characters */
42             printf( "Incorrect letter grade entered." );
43             printf( " Enter a new grade.\n" );
44             break;
45       }
46    }
47
48    printf( "\nTotals for each letter grade are:\n" );
49    printf( "A: %d\n", aCount );
50    printf( "B: %d\n", bCount );
51    printf( "C: %d\n", cCount );
52    printf( "D: %d\n", dCount );
53    printf( "F: %d\n", fCount );
54
55    return 0;
56 }
```

2.1 Use switch loop to update count

3. Print results

Program Output

```
Enter the letter grades.
Enter the EOF character to end input.
A
B
C
C
A
D
F
C
E
Incorrect letter grade entered. Enter a new grade.
D
A
B

Totals for each letter grade are:
A: 3
B: 2
C: 3
D: 2
F: 1
```

4.8 The do/while Repetition Structure

- **do/while** 迴圈
  - 語法:

```
do {
  statements;
} while ( condition );
```

至少執行一遍

4.8 The `do/while` Repetition Structure

- Example (letting counter = 1):

```
counter = 1;
do {
   printf( "%d  ", counter );
} while (++counter <= 10);
```

  - Prints the integers from **1** to **10**

```c
/* Fig. 4.9: fig04_09.c
   Using the do/while repetition structure */
#include <stdio.h>

int main()
{
   int counter = 1;

   do {
      printf( "%d  ", counter );
   } while ( ++counter <= 10 );

   return 0;
}
```

1. Initialize variable

2. Loop

3. Print

Program Output

```
1  2  3  4  5  6  7  8  9  10
```

Program: calculate the number of digits in an integer

```
#include <stdio.h>

int main(void)
{
  int digits = 0, n;

  printf("Enter a nonnegative integer: ");
  scanf("%d", &n);

  do {
    n /= 10;
    digits++;
  } while (n > 0);

  printf("The number has %d digit(s).\n", digits);

  return 0;
}
```

Example

1234 /10 $\xrightarrow{\text{Remainder (餘數)}}$ 4

$\downarrow$ Quotient (商)

123 /10 $\xrightarrow{\text{Remainder}}$ 3

$\downarrow$ Quotient

12 /10 $\xrightarrow{\text{Remainder}}$ 2

$\downarrow$ Quotient

1 /10 $\xrightarrow{\text{Remainder}}$ 1

$\downarrow$ Quotient

0

4.9 The `break` and `continue` Statements

- **`break`**
  - 立刻離開 **`while`**, **`for`**, **`do/while`** or **`switch`** 結構，去執行結構外下一行指令

- **`continue`**
  - 跳過**`while`**, **`for`** or **`do/while`** 結構內continue後的指令，直接進行下一次迴圈

注意

- **`while`** and **`do/while`**
  - continue指令執行後立刻測試迴圈條件

- **`for`**
  - **continue指令執行後先執行Increment那段，再測試迴圈條件**

```c
/* Fig. 4.12: fig04_12.c
   Using the continue statement in a for structure */
#include <stdio.h>

int main()
{
   int x;

   for ( x = 1; x <= 10; x++ ) {

      if ( x == 5 )
         continue;  /* skip remaining code in loop only
                       if x == 5 */

      printf( "%d ", x );
   }

   printf( "\nUsed continue to skip printing the value 5\n" );
   return 0;
}
```

1. Initialize variable

2. Loop

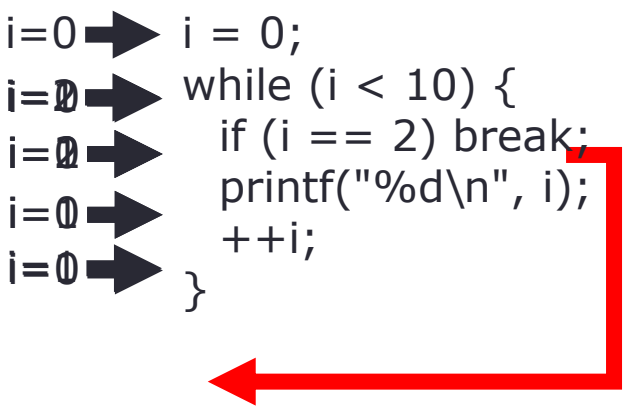
3. Print

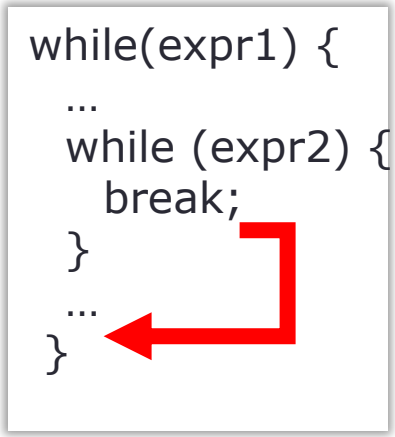
```
1 2 3 4 6 7 8 9 10
Used continue to skip printing the value 5
```

Program Output

Control of the loop flow

- break脫離目前的迴圈，跳到迴圈外下一行。
  - 適用於while-loop，do-while-loop，for-loop，與switch。



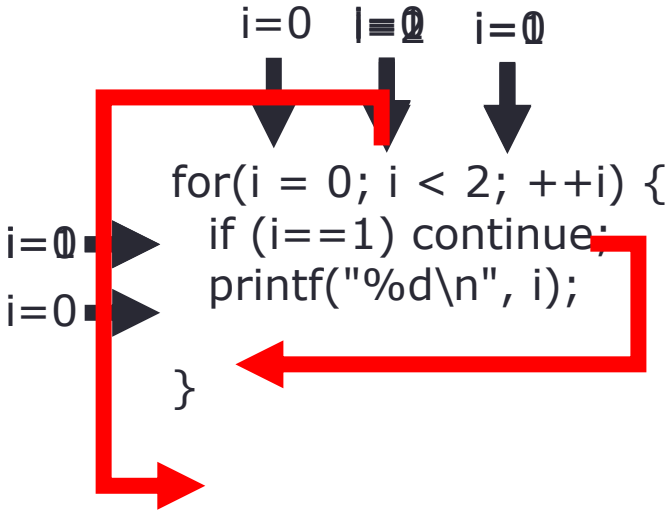


Control of the loop flow (cont'd)

- continue省略之後的指令，跳至迴圈最後一行，然後繼續執行。
  - 適用於while-loop，do-while-loop，for-loop。

```
n=0;
num=0;
while (n < 10) {
   scanf("%d",&i);
   if (i == 0) continue;
   n++;
   sum+=i;

}
```



Control of the loop flow (cont'd)

- goto則是可以直接跳到某個標籤(label)。
  - label宣告語法
    - Identifier: statement
    - Identifier:;
  - goto identifier;
- goto很方便但**不要濫用**，因為會讓程式的結構與流程變得混亂。
  - 其實可以完全不用goto。

```
while(expr1) {
  …
  while (expr2) {
    goto done;
  }
  …
}
done:;
```

```
for(d=2; d<n; ++d) {
  if(n %d == 0) goto done;
}
done:
if (d < n)
  printf("%d is divisible by %d\n",n,d);
else
  printf("%d is prime\n",n);
```

4.10 邏輯運算子(logical operators)

- 邏輯運算子(logical operators)
  - ! 否
  - && 且
  - || 或
- 透過邏輯運算子可以結合邏輯運算式
  - (i != 0) && (j / i >0)

&&	T	F
T	T	F
F	F	F

||	T	F
T	T	T
F	T	F

!	T	F
	F	T

Short-circuit nature of && and ||

- 例子: exp1 && exp2

  當exp1為false，exp2的值不管是什麼整個式子必為false。因此當exp1為False ，C根本不去算exp2的值。
- 例子: exp1 || exp2

  當exp1為true，exp2的值不管是什麼整個式子必為true。因此當exp1為true ，C根本不去算exp2的值。
- 注意下面short-circuit所造成的現象

  例子:

```
int i, j;
i = 0;
j = 2;

(1) (i<1) && (++j >2)
(2) (i<0) && (++j >2)
```

4.11 Confusing Equality (==) and Assignment (=) Operators

- Dangerous error
  - 可以產生值的運算式都可以用在條件式裡
    - 若運算式的值是非零就被當成true，值是零就被當成**false**

```
if (1+2) {
   printf("true\n");
}
if (a*x*x+b*x+c) {
}
```

  - Example using **==**:

```
if ( payCode == 4 )
   printf( "You get a bonus!\n" );
```

    - Checks **paycode**, if it is **4** then a bonus is awarded

4.11 Confusing Equality (==) and Assignment (=) Operators

- Example, replacing == with =

```
if ( payCode = 4 )
   printf( "You get a bonus!\n" );
```

  - This sets **paycode** to **4**
  - **4是非零因此payCode=4這個運算式為true**。
- Logic error, not a syntax error

```
if (4== payCode)
   printf( "You get a bonus!\n" );
```

Correct syntax

rvalue

```
if (4= payCode)
   printf( "You get a bonus!\n" );
```

Incorrect syntax

4.11 Confusing Equality (==) and Assignment (=) Operators

- **lvalues**
  - 可以出現在=(assignment)左邊的東西，它們的值可以改變，例如變數
    - `x = 4;`
- **rvalues**
  - 只可以出現在=(assignment)右邊的東西，如常數
    - Cannot write `4 = x;`
    - Must write `x = 4;`
- **lvalues can be used as rvalues, but not vice versa**
    - `y = x;`

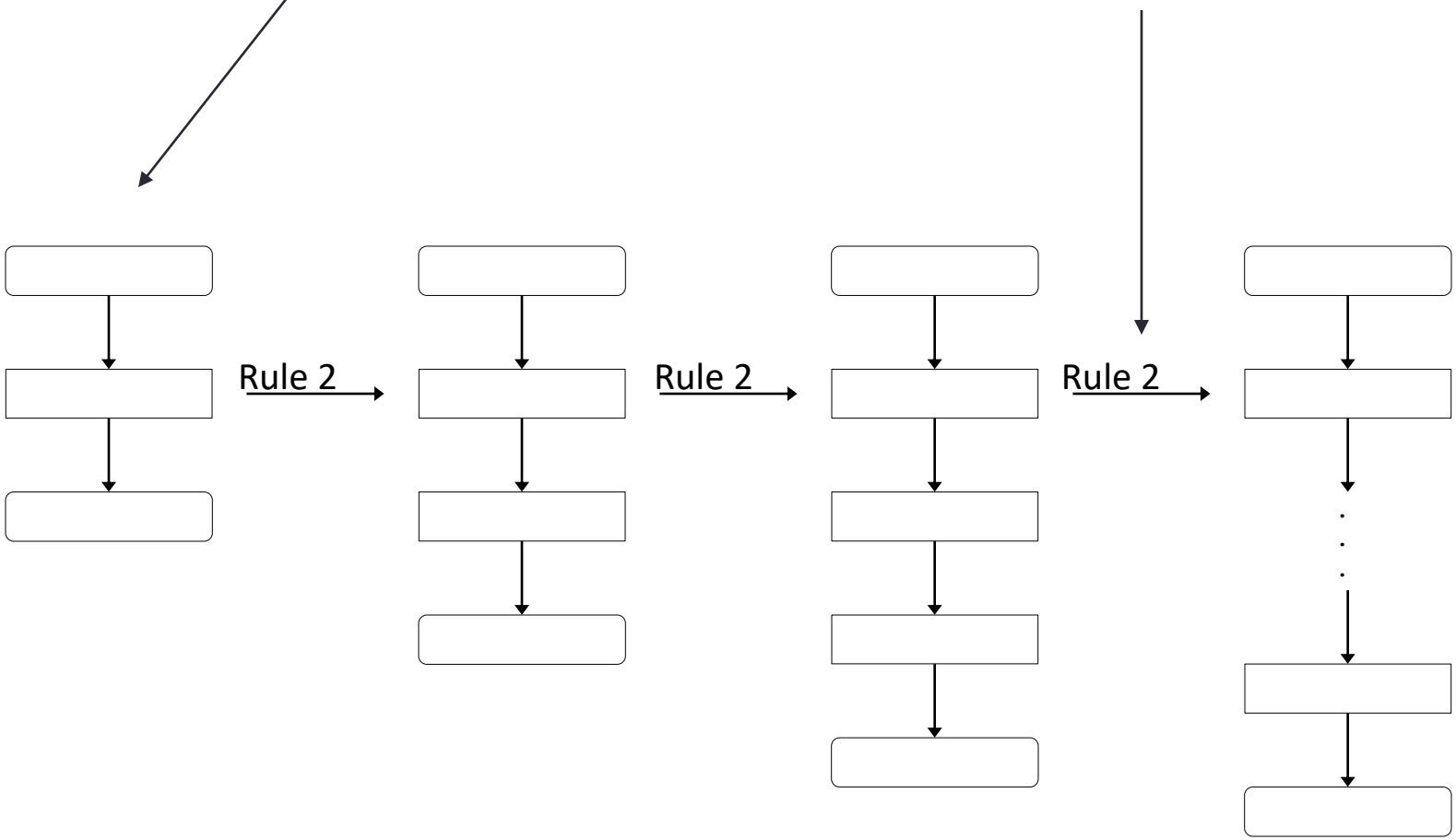
4.12 Structured-Programming Summary

- 結構化程式設計(Structured programming)
  - Easier than unstructured programs to understand, test, debug and, modify programs
- 結構化程式設計原則
  - 考慮單一出/入點的結構
  - 原則
    1. 從問題最簡化的形式開始著手
    2. 將每一個大步驟再細分成若干小步驟
    3. 每一個小步驟再用各種控制結構把它實作出來 (如sequence, **if**, **if/else**, **switch**, **while**, **do/while** or **for**)
    4. 可以重複應用規則2與3

4.12 Structured-Programming Summary

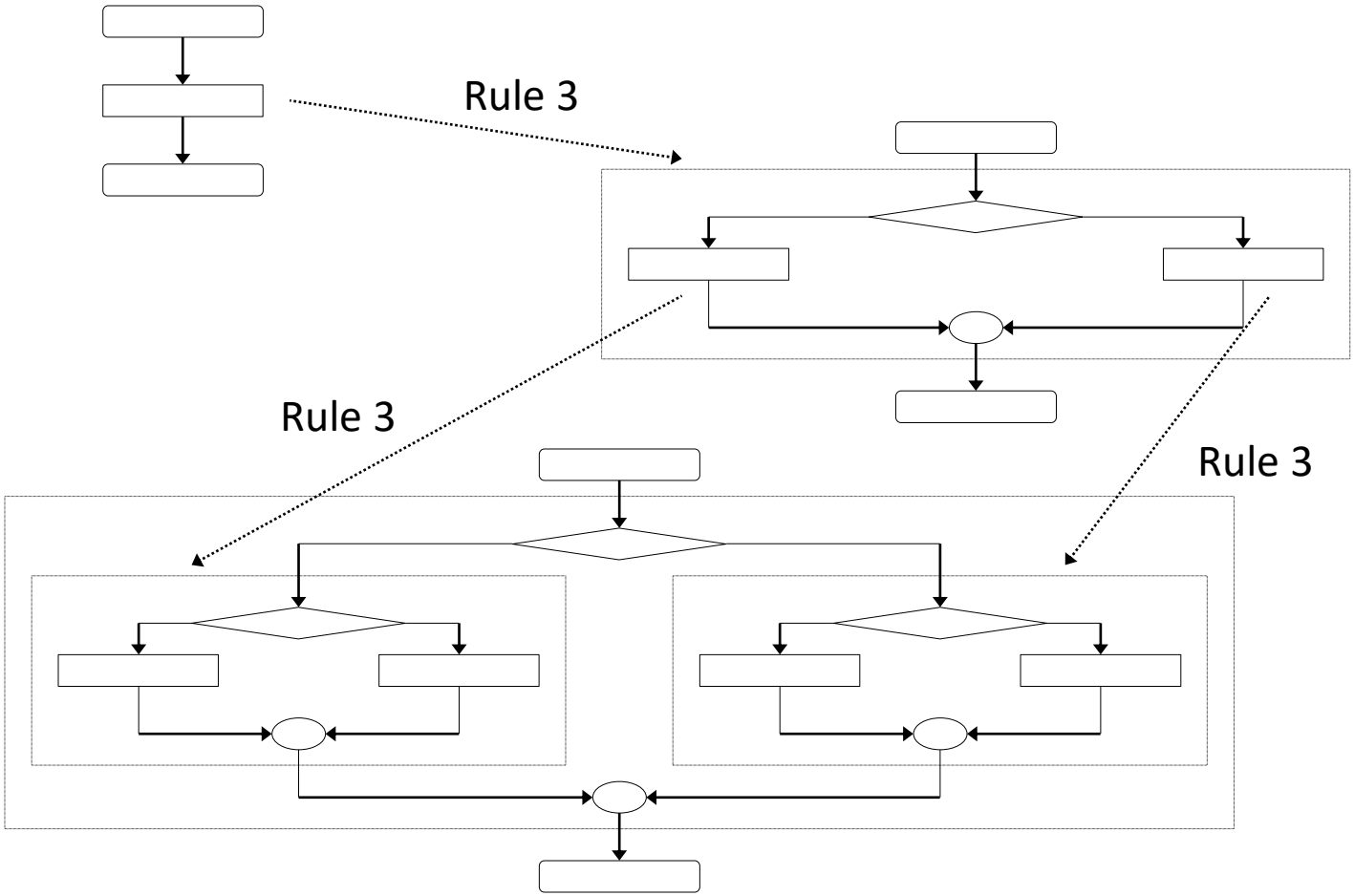
Rule 1 -從問題最簡化的形式開始著手

Rule 2 -將每一個大步驟再細分成若干小步驟



4.12 Structured-Programming Summary

Rule 3 -每一個小步驟再用各種控制結構把它實作出來
(如sequence, **if**, **if/else**, **switch**, **while**, **do/while** or **for**)



4.12 Structured-Programming Summary

- 所有的程式皆可由下面這三種控制結構組成
  - 循序(Sequence) – handled automatically by compiler
  - 選擇(Selection) – `if`, `if/else` or `switch`
  - 反覆(Repetition) – `while`, `do/while` or `for`
    - Can only be combined in two ways
      - Nesting (rule 3)
      - Stacking (rule 2)
  - 任何的選擇結構都可以用if指令改寫，任何的反覆結構都可以用while指令改寫。

例子:計算GCD (greatest common divisor)

- **分析**: $m = ak$, $n = bk$ where k is the gcd of m and n, and $m > n$.

  $a=pb+q \Rightarrow m =(pb+q)k = pbk + qk \Rightarrow qk = m \% n$,
- **設計算式**:(1) 若$m \% n$ ==0, n 是gcd

  (2) 否則t = $m \% n$; $m = n$; $n = t$;再反覆相同步驟。
- **設計測試例子**:用來驗證算式/程式是否正確

  Iteration 1: m:15 n:10

  q:5 (15 % 10)

  Iteration 2: m:10 n:5

  q:0 (10 % 5)

  ⇒ 5是15與10的 gcd
- **寫程式**

```
int q;
if (m < n) { //處理m < n的狀況
      q = n;
      n = m;
      m = q;
}
while ((q=m % n) != 0) {
     m = n;
     n = q;
 }
 printf("GCD:%d\n", n);
```

例子:質因數分解

```
// 限制: n > 1
while (n % 2 == 0) {
    printf("2,");
    n/=2;
}
for(i=3;i*i<=n; i+= 2) {
    while(n%i==0) {
        printf("%d,",i);
        n/=i;
    }
}
if (n>1) {
    printf("%d\n",n);
} else {
    printf("\n");
}
```

先從n分解出質數2

i必定是質數(why?)

$$n = a_1^{d_1} \times a_2^{d_2} \times \cdots \times a_k^{d_k}$$

$a_1 < a_2 < \cdots < a_k$,為n的質因數

範例: $n = 2^3 \times 3^2 \times 11^1$

i	2	2	2	3	3	5
n	$2^3 \times 3^2 \times 11^1$	$2^2 \times 3^2 \times 11^1$	$2^1 \times 3^2 \times 11^1$	$3^2 \times 11^1$	$3^1 \times 11^1$	11^1

奇數質數由小到大從n分解出來。

離開for-迴圈時表示，此時n絕對無法由兩個以上>= i的質數所組成。所以再不需要分解n了。

安全的使用scanf

- scanf回傳值表示成功讀入幾個資料。
- 可以由scanf的回傳值判斷是否成功讀入資料

```
int x = scanf("%d", &grade);
```

  - 如果使用者輸入一個整數，scanf會傳回1，x的內容會是1。
  - 如果使用者輸入hello，scanf會傳回0，x的內容會是0。

```
int x = scanf("%d%d",&num1,&num2);
```

  - 如果使用者輸入兩個整數，scanf會傳回2，x的內容會是2。

字元型態

- 字元型態
  - char
  - unsigned char

  範例

```
char c;
c = 'A';  // c is 'A' now
c = 65;  // c is also 'A'
c++;      // c is now 'B'

if (c >= 'a' && c <= 'z') {
   c = c - 'a' + 'A';              // convert a lower-case letter to a upper-case letter
}
```

- 除了當字元外，C其實就是將字元型態看成小整數
  - signed char -128~127
  - unsigned char 0~255

  - char本身是signed或unsigned視編譯器如何處理。

Escape sequence

Description	Escape sequence
Bell	\a
Backspace	\b
Form feed	\f
New line	\n
Carriage return	\r
Horizontal tab	\t
Vertical tab	\v
Backslash	\\
Question mark	\?
Single quote	\'
Double quote	\"
Octal	\oct
Hexadecimal	\x

```
#define ESC '\x1b'
```

Character-handling function

- #include<ctype.h>
- 常用的函數
  - toupper
  - tolower

格式化輸出/入字元

- 使用scanf其轉換指引為%c

```
char ch;
scanf("%c", &ch);
printf("%c", ch);

scanf(" %c", &ch); //若要濾掉空白

//若要濾掉new-line character
do {
   scanf("%c",&ch);
} while (ch=='\n');
```

```
printf("enter an integer:");
scanf("%d",&u);
printf("Enter a command:");
scanf("%c",&ch);
 //ch會讀到讀u後接下的那一個字元
```

- putchar輸出一個字元
  - putchar(ch);

- getchar輸入一個字元

```
//read until '\n' is encountered
while((ch=getchar())!= '\n') {
   statements
}

//skip the rest of line
while(getchar()!= '\n');

//skip blanks
while(getchar()== ' ');
```

Program: Determine the length of a message

```
#include <stdio.h>

int main(void)
{
  char ch;
  int len = 0;

  printf("Enter a message: ");
  ch = getchar();
  while (ch != '\n') {
    len++;
    ch = getchar();
  }
  printf("Your message was %d character(s)
    long.\n", len);

  return 0;
}
```

```
#include <stdio.h>

int main(void)
{
  int len = 0;

  printf("Enter a message: ");
  while (getchar() != '\n') {
    len++;
  }
  printf("Your message was %d character(s)
    long.\n", len);

  return 0;
}
```