控制結構

資科系
林偉川

結構化程式設計

- 定義
  - 從事程式設計的過程中，依照程式的邏輯特性將程式細分成幾個較小的問題，再將這些較小的問題同樣依照程式的邏輯特性再往下細分成更小的問題，依此類推直到很容易編寫程式的單元時為止。
- 注意事項
  - 當採用結構化程式設計法來設計程式時，應當儘量避免使用GOTO命令，以避免破壞程式的可讀性及結構性。
- 優點
  - 程式可分工、容易除錯、可讀性較高及較易維護
- 缺點
  - 程式碼較長及執行時間較久

結構化程式設計的基本結構

- 基本結構共有三類
  - 循序結構(sequential structure)
    - 敘述會按照先後順序來執行
  - 選擇結構(selection structure)
    - 單選擇
    - 雙重選擇
    - 多重選擇
  - 反覆結構(iteration structure)
    - 前測迴路
    - 後測迴路

3

循序結構

- 循序執行的程式段即為循序結構
- 範例

  敘述-1;
  敘述-2;
  …
  敘述-n;

  以上n條敘述按照敘述-1、敘述-2、…、敘述-n的順序執行

4

選擇結構

- 選擇結構可分為
  - 單選擇
  - 雙重選擇
  - 多重選擇

5

單選擇結構

- 意義
  - 條件成立時有相對應的敘述必須被處理，但是當條件不成立時則沒有相對應的敘述要處理，因此稱為單路選擇
- 範例
  - Pascal語言
    - if 條件 then 敘述
  - C/C++/Java語言
    - if (條件) 敘述
  - Basic語言
    - if 條件 then 行號或敘述
  - Algol 60語言
    - if 條件 then敘述
- 以上的單路選擇敘述均代表條件成立時執行對應之敘述，但若條件不成立則直接執行if的下一條敘述

6

雙重選擇結構

- 意義
  - 條件成立時有相對應的敘述必須被處理，而且當條件不成立時也有相對應的敘述要處理，因此稱為雙重選擇
- 範例
  - Pascal
    - if 條件 then 敘述1 else 敘述2
  - C/C++/Java
    - if (條件) 敘述1；else 敘述2
  - BASIC
    - if 條件 then 行號或敘述 else 行號或敘述
  - Algol 60
    - if 條件 then 敘述1 else敘述2

7

範例

若A=-5，B=5，則執行完下列程式後，A、B變成多少？(A)5,5 (B) -5,-5 (C) 5,-5 (D) 15,5

```
if (A<0)
      if (B<0)  A=B; else B=A;
else A=B+10;
```

8

範例

(1)執行下列程式所印出之值為何？2

(2)若將S的值改為85，則印出之值又將為何？4

(3)又若將S的值改為65，則結果為何？6

```
S=30;
G=2;
IF S> =60 THEN IF S> =70 THEN
   G=4
ELSE G=6;
WRITELN(G);
```

9

範例

- 執行右側的Java程式，可能得到幾種不同的V值，分別為何？使V值為1的條件為何？(將條件盡量簡化，其中C,L,S,Y皆為boolean) !S &&(!L) 或 (!C)&&(!S)

```
V=1;
  if (C && L)
    V=2;
  else if (! C) {
     if (S && Y)
        V=3;
     else if (S && (! Y))
        V=4; }
```

10

範例

底下的C程式會讓變數X之值為何？3

```
if (((3>2) && (2<2)) | | (5==6) | | ((5>4) && 3))  x=3; else  x=4;
```

底下的Fortran程式，變數New,Nand之值為何？8,8

```
   New=0
   Nand=28
5  if(New .GE. Nand) GOTO 20
   New=New+2
  Nand=Nand-New
 GOTO 5
20 PRINT *, New, Nand
```

11

多重選擇結構

- 多重選擇結構一般是指case結構
- Algol W首創case結構

12

多重選擇結構 -- Algol W

- Algol W的case結構語法如下：

```
case <整數運算式> of
begin
    exp 1;
    exp 2;
     ...
    exp n;
end;
```

其中case的選擇項一定必須是整數運算式，此點大大限制了case敘述的使用彈性。當選擇項的值為1時執行exp 1，為2時執行exp 2,...,為n時則執行exp n

13

多重選擇結構 -- Pascal

- Pascal語言提供的case敘述語法如下：

```
case <選擇運算式> of
  <選擇標記1>: 敘述1;
  <選擇標記2>: 敘述2;
.....
  <選擇標記n-1>: 敘述(n-1)
  〔else 敘述n〕
end;
```

- 以上case敘述代表當<選擇運算式>＝<選擇標記1>時執行敘述-1，若<選擇運算式>＝<選擇標記2>時執行敘述-2，….，若<選擇運算式>＝<選擇標記n-1>時執行敘述-(n-1)

14

多重選擇結構 -- Pascal

- 但若<選擇運算式>與所有的<選擇標記>皆不符合時則執行時執行敘述-n(假如“else”部份存在)
- Pascal語言的case敘述屬於內隱分歧(implicit branch)結構
- 運算式的型態：整數，布林值，字元、列舉式資料型態或子範圍皆可

15

多重選擇結構 -- Pascal

```
case number of
1,3,5 : begin
  odd:=odd+1
  oddsum=oddsum+1
        end;
2,4,6 : begin
  even:=even+1
  evensum=evensum+1
        end;
else writeln('error input');
end;
```

16

多重選擇結構 – C/C++/Java

- C/C++/Java 語言提供的多重選擇結構語法

```
switch (<選擇運算式>){
case (選擇標記1):   敘述1;   break;
case (選擇標記2):   敘述2;   break;
 .....
case (選擇標記n-1): 敘述(n-1);  break;
[default: 敘述n;]
}
```

- 以上switch敘述代表，當<選擇運算式>＝<選擇標記1>時執行敘述1，若<選擇運算式>＝<選擇標記2>時執行敘述2，….，若<選擇運算式>＝<選擇標記n-1>時執行敘述(n-1)

17

多重選擇結構 – C/C++/Java

- 但若<選擇運算式>與所有的<選擇標記>皆不符合時則執行時執行敘述n(假如default部份存在)。
- switch敘述屬於外顯分歧(explicit branch)結構
- switch結構中之敘述為單一或複合敘述皆可

18

多重選擇結構 -- C/C++/Java

```
switch (number) {
case (1,3,5) : odd:=odd+1; oddsum=oddsum+1;
        break;
case (2,4,6) : even:=even+1; evensum=evensum+1;
        break;
default: printf('error input');
}
```

19

多重選擇結構 – ADA

- Ada語言的case結構之語法結構

  case 運算式 is
  {when 選擇 { | 選擇}敘述}
  end case
- "運算式"可為整數或列舉式資料的型態
- 可利用"others"來減少未定義之情況

20

多重選擇結構 – Fortran

- Fortran語言的if結構之語法結構

語法一：if 條件 L1, L2

語法二：if (數學運算式) L1, L2, L3

- 語法一為條件成立會GOTO到L1，否則將GOTO到L2
- 語法二為數學運算式<0會GOTO到L1，數學運算式=0會GOTO到L2，數學運算式>0會GOTO到L2
- 可讀性不高，因使用GOTO

21

多重選擇結構設計應注意的事項

- 選擇運算式允許的型態為何？
- 是否允許由CASE標記內, 跳出到外部結構？
- 是否允許外部結構跳入CASE標記內？
- 標記間是否必須彼此互斥？
- 是否提供“default”或“else”等未定義情況？

22

反覆結構

- 定義
  - 反覆結構(iteration structure)是指讓一個或一群敘述能夠反覆的執行之敘述群
- 種類
  - 前測迴路
  - 後測迴路

23

前測迴路

- 特性
  - 迴圈執行的最少次數是0次
  - 最多次數是無限多次
- 一般程式語言常見的前測迴路可分為二類
  - while-loop
  - for-loop

24

Pascal 的 while-loop

- 語法

```
while <條件> do
begin
   迴圈敘述
end;
```

- 執行while loop時會先測試<條件>是否成立，當<條件>成立時，才會進入迴圈敘述執行，否則跳出迴圈結構，所以while loop可能一次也不執行

25

C/C++/Java 的 while-loop

- 語法

```
while (<條件>){
   迴圈敘述;\
}
```

- 執行while loop時會先測試<條件>是否成立，當<條件>成立時，才會進入迴圈敘述執行，否則跳出迴圈結構，所以while loop可能一次也不執行

26

Basic 的 while-loop

- 語法

  while <條件>
  　迴圈敘述
  wend

- 執行while loop時會先測試<條件>是否成立，當<條件>成立時，才會進入迴圈敘述執行，否則跳出迴圈結構，所以while loop可能一次也不執行
- while loop 重要特性
  - 迴圈執行之次數事先可不知

27

範例

- 右側程式段執行完畢後，S=？

- ```
  S = 0;
  I = 7;
  while (I <= 127) {
     S := S+I;
     I := I+3;
  }
  ```

28
  ```

Pascal 的 for-loop

第一種

```
for 控制變數 := <初值> to <終值> do
  begin
        迴圈敘述
  end;
```

第二種

```
for 控制變數 := <初值> downto <終值> do
  begin
        迴圈敘述
  end;
```

29

範例

- Pascal程式片段如右，試計算出當程式執行後，何數將被印出？

```
s:= 0;
for i:=7 to 13 do
  s := s+i
writeln(s);

s:= 0;
for i:=13 downto 7 do
  s := s+i
writeln(s);
```

30

C/C++/Java 的 for-loop

- 語法

```
for (exp1;exp2;exp3) {
    迴圈敘述
}
```

exp1：設定控制變數之初值

exp2：設定迴圈執行時，控制變數之範圍

exp3：設定控制變數變化的情況

31

範例

- C/C++/Java程式片段如右，試計算出當程式執行後，何數將被印出？

```
s= 0;
for (i =7; i<=127; i+=3)
  s = s+i
next
printf("%d", S);
```

32

Basic 的 for-loop

- 語法

  FOR 控制變數 = 初值 TO 終值 STEP 遞增量
  迴圈敘述
  NEXT 控制變數

  當遞增量>=0時，控制變數<=終值，執行迴圈敘述

  當遞增量<0時，控制變數>=終值，執行迴圈敘述

33

範例

- BASIC程式片段如右，試計算出當程式執行後，何數將被印出？

```
10 S = 0
20 FOR I = 11 TO 123 STEP 7
30   S = S+I
40 NEXT I
50 PRINT S
```

34

範例

- 右列BASIC程式中K=K+I＊J將被執行多少次？

```
10 FOR I=1 TO 6 STEP 3
20   FOR J=5 TO 0 STEP-2
30     K=K+I*J
40   NEXT J
50 NEXT I
```

35

範例

```
10  S=0
20  FOR I=7  TO 127 STEP 3
30    S=S+I
40  NEXT I
50  PRINT S
60  END
```

請問S=？

36

Fortran 77 的 DO-loop

- Fortran 77“DO-loop”即一般高階語言的“for-loop”
- 語法

  Do label 控制變數=初值,終值,增值
  　　迴圈敘述
  label continue
- DO loop的特性為控制變數的型態可以是整數、單精確度型態
- 控制變數在Do loop中可變更，僅能由Do敘述進入Do loop中(即“單一入口”之規定)
- Fortran 77的Do loop結構中的迴圈敘述可能一次也未能執行，也就是說Fortran 77的DO loop是前測迴路

37

Fortran 77範例

```
k=0
Do 10 I=1,6
  IF(k)1,2,3
1  k=k+4
     GOTO 10
2  k=k-3
     GOTO 10
3  k=3*k-1
10  CONTINUE
```

38

Algol 60 的 for-loop

- 語法

for敘述::=for控制變數:= <list-element> {,<list-element>}do <stmt>
<list-element>::= <exp>｜<exp> step <exp> until <exp>｜<exp>
while <Boolean-exp>

- Algol 60“for loop”的特性為前測迴路
- 控制變數的型態可以是整數或實數
- 控制變數在Do loop中能改變
- 可利用GOTO敘述跳躍至迴圈敘述內

39

Ada 的 for-loop

- 語法

for 控制變數 in〔reverse〕範圍
loop
　迴圈敘述
end loop

- 控制變數的型態必須是整數或列舉式資料型態
- 控制變數在Do loop中不能改變
- 不可利用GOTO敘述跳躍進入迴圈敘述

40

Ada範例

```
for I in 1..100
loop
    sum:=sum+I;
end loop

for I in reverse 100..1
loop
    sum:=sum+I;
end loop
```

41

後測迴路 (post-test loop)

- 特性
  - 迴圈執行的最少次數是1次
  - 最多次數是無限多次

42

Pascal 的後測迴路

- 語法

```
repeat
    迴圈敘述
until <條件>
```

- 先執行迴圈敘述再檢查<條件>，<條件>不成立時執行迴圈敘述，當<條件>成立時將離開迴圈結構

43

C/C++/Java 的後測迴路

- 語法

```
do {
    迴圈敘述
}
while (<條件>)
```

- 先執行迴圈敘述再檢查<條件>，<條件>成立時執行迴圈敘述，當<條件>不成立時將離開迴圈結構

44

Fortran IV 的 DO-loop

- 語法

  Do label控制變數=初值,終值,增值
  　迴圈敘述
  label continue
- 控制變數的型態必須是整數
- 控制變數在Do loop中不能改變
- 可利用GOTO敘述離開迴圈敘述
- 最特別的是Fortran IV的Do loop結構中的迴圈敘述至少會執行一次，也就是說**Fortran IV**的**DO loop**其實是後測迴路，這種作法違背了程式設計師的習慣，因此在Fortran 77便已將“DO loop”改為前測迴路

45

流程圖(flow chart)

- 主要的功用是發展程式時可利用流程圖來做為分析的工具，而且讓維護工作較容易且容易除錯

46

常見的流程圖符號

- 開始/結束符號

- 敘述符號

- 副程式

- 條件判斷符號

- 輸入/輸出符號

- 連接符號

- 流向符號

47

精選習題

- 請簡述GOTO敘述的分類
- GOTO結構有何利弊，舉出四種可以取代GOTO結構的流程控制敘述指令
- 試舉一例說明何謂無窮迴路(infinite loop)？
- 說明FORTRAN IV和FORTRAN 77 之DO迴圈的不同之處。
- 試問C語言和Pascal語言的for loop有何不同？二種語言不同設計的主要理由為何？
- 試說明C語言的switch結構和Pascal語言的case結構的相異之處。

48