程式語言的演進

資科系
林偉川

章節大綱

- 程式語言的演進
- 各種語言的特性
- 高階語言的處理器
- 命令式語言與應用式語言的比較

程式語言的意義

- 由一組系統化的符號所成之集合，目的是表達某種機器解決特定問題的步驟
- 向計算機描述計算過程之工具
- 程式語言設計的目標是簡潔(simplicity)，如何以最精簡的方式來表達

3

學習程式語言的目的

- 增進對程式語言的瞭解。
- 可改進設計的程式之架構，藉以增進程式執行之效率。
- 可選擇適用的語言。
- 較易學習與設計新的語言。

4

程式語言的分類

- 第一代程式語言—機器語言(低階語言)
- 第二代程式語言—組合語言(低階語言)
- 第三代程式語言—高階語言
- 第四代程式語言—極高階語言
- 第五代程式語言—自然語言

5

第一代程式語言

- 機器語言(machine language)
  - 指令與資料均由二進碼所組成
  - 寫成的程式段不需語言處理器處理，便可直接在機器上執行

6

第二代程式語言

- 組合語言(assembly language)
  - 組合語言的指令稱為助憶碼
  - 組合語言的指令分為機器指令(machine operation)與虛擬指令(pseudo operation)二類
  - 必需經由組譯程式(assembler)的處理，才可在機器上執行
  - 與機器語言合稱為低階語言

7

第三代程式語言

- 高階語言(high level language)
  - 又稱為程序導向語言(procedure oriented language)
  - 此種語言必需經過編譯或直譯程式處理過方可執行
  - 如Pascal，C，Basic，Fortran與Cobol。

8

第四代程式語言

- 極高階語言
  - 又稱為問題導向語言(problem-oriented language)
  - 如 SQL(Structured Query Language)

9

第五代程式語言

- 自然語言 (nature language)
  - 又稱為知識庫語言(knowledge based language)
  - 語法接近人類日常生活的語言
  - 例如演算法的虛擬碼

10

Fortran

- 第一個高階語言
- 針對科學計算而設計
- 具固定格式(程式必需由第7個位置寫起)
- **Compiler 於1957/4完成，由18 人年完成**
- 變數名單不能超過6個字
- 首創輸出入格式化(I/O format，I 為整數，F 為浮點數)
- 允許隱含性變數(ijklmn開頭的變數名稱為整數)
- 提供正、零、負三種 IF 分枝結構
- 提供 Do Loop迴圈控制結構(後測迴圈)
- 後期才加入字串處理(Fortran 90)

11

範例

執行下列部份程式,求出 K 值？

```
      K=-1
      DO 400 I=1,5
      IF(K) 100,200,300
100   K=K+5
      GOTO 400
200   K=2*K+1
      GOTO 400
300   K=4*K-2
400 CONTINUE
```

SOL ：

K	-1	4	14	54	214	854
I	1	2	3	4	5	6

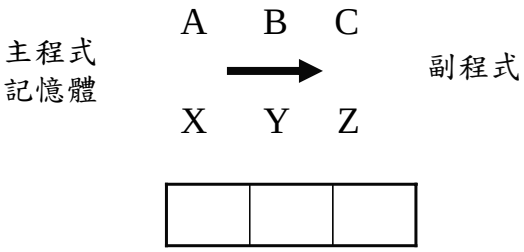
12

範例

試敘述 Fortran 語言所提供的“common area”的作用，及其處理方式？

Aliasing (別名)

Common A,B,C　　Common X,Y,Z



13

Algol 60

- ALGOrithmic Language
- 發展於1960年代, 但無較大之贊助者
- 採自由格式(free format)
- 採外顯式(explicit)宣告
- 採區塊結構(block structure)
- 允許使用遞迴副程式(recursive subroutine)
- 使用傳名呼叫法(call by name)傳遞參數

14

Algol 60

- 首創以保留字(reserved word)來定義資料型態
- 程式指令以分號結束
- 設定值符號為:=
- 動態界限陣列
- 首創結構化程式設計的概念
- 首創以B.N.F.來描述語法 (Algol 58提出)
- 結構化程式設計的概念為程式設計過程依程式的邏輯性不斷細分，直到很容易編寫程式單元為止，且應儘量避免用goto

15

結構化程式設計

- 把一個大的問題，依照邏輯上的特性，往下細分成幾個小的問題，再把這幾個小的問題，依照邏輯的特性，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
- 優點: 可以分工、可讀性高、易維護、易除錯
- 缺點: 程式碼會變長，執行時間較長

16

結構化程式設計的基本結構

- 循序結構
- 選擇結構
- 反覆結構

17

Algol W

- 使用數值結果呼叫法 (call by value result) 傳遞參數。
- 首創提供case敘述。
- 提供記錄(record)與指標(pointer) 結構。

18

Algol 68

- 參數的傳遞採用數值結果呼叫法。
- 提供記錄與指標資料型態。
- 首先提供使用者自行定義資料型態的功能。

19

範例

執行下列部份程式，求出 c 值？

主程式

```
a=5
b=4
c=aa(a-b,a+b)
Print c
```

副程式

```
Sub aa(a,b)
  c=a*b+a/b
  return c
End sub
```

SOL ：

	傳值呼叫	傳名呼叫
c	9	-2

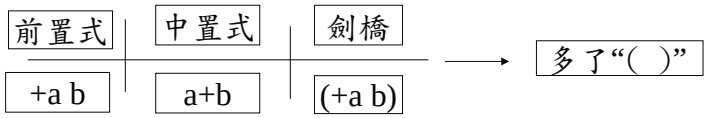
20

LISP (LISt Processing language)

(1)發展於1950 年代末期，由麻省理工學院發展。

(2)提供S 運算式(S expression)做為主要的運算式。

(3)採"劍橋波蘭式"(Cambridge polish notation)
作為算術運算式➔現今資料結構之前序運算式。



21

LISP

(4)利用垃圾收集法(garbage collection) 來管理記憶體。

(5)使用於人工智慧(Artificial Intelligence) 之應用，稱為人工智慧的低階語言。(人造智慧)

(6)大量使用遞迴觀念

22

LISP example

```
(defun equal_list(lis1 lis2)
 (COND
    ((ATOM lis1) (EQ lis1 lis2))
    ((ATOM lis2 NIL)
    ((equal_list(CAR lis1)(CAR lis2))
      (equal_list(CDR lis1)(CDR lis2)))
     (T NIL)
   ))
```

若字串相等則return true，否則return NIL

23

Lisp example

```
(defun iterative_length(lst)
  ( prog (sum)
    (SETQ sum 0)
    again (COND
      ((ATOM lst (return sum)))
    (SETQ sum (+ 1 sum))
    (SETQ lst (CDR lst))
    (GO again)
   ))
 (defun iterative_length(lst)
  (COND
 ((NULL lst) 0)
 (T (+ 1 (recursive_length( CDR lst))))
 )
)
```

24

範例

Lisp 語言記憶體的管理方式採用“Garbage collection”,試說明何謂“Garbage collection”？

Sol:

系統會主動的回收程式不使用的記憶體空間
如: Smalltalk, JAVA

25

Cobol

- COmmon Business Oriental Language
- 發展於 1960～1970 年代,由美國防部贊助。
- 主要用於商業資料處理，處理大量資料的輸入與輸出。
- 變數名稱可達30個字以 _ 連接
- 缺乏複雜的數學計算能力
- 具 IF-THEN-ELSE 敘述。
- 語法傾向自然語言(natural language)。
- 雜訊字(noise word)用來增加程式的可讀性(go to)。
- 首創與機器無關的資料描述方式(data division)。

26

Cobol語言提供的四個資料描述

- Identification Division ：程式名稱、作者名稱
- Environment Division ：環境段
- Data Division ： 資料段
- Aocedure Division ：程式段

27

APL(A Programming Language)

- 擅長數學計算。
- 採不標準字元集。
- 允許陣列整體運算。
- 提供指標(pointer) 型態。
- 首創提供集合(set) 資料型態。
- 提供記錄(record) 資料型態。
- 提供 case 結構。
- 允許使用者自訂資料型態。
- 採傳值呼叫法與傳址呼叫法傳遞參數。

28

BASIC

- Beginner's All-purpose Symbolic Instruction Code
- 1964年發展。
- 屬交談式(interactive)語言。
- 適合程式發展初期使用。
- 利用直譯器(interpreter)處理程式。

29

SNOBOL

- StriNg Oriented symBOlic Language
- 貝爾實驗室於1960年代中期發展。
- 具型樣配對(pattern matching) 能力。
- 具字串處理能力(string manipulating)。
- 型樣資料型態(pattern data type)。

30

PL/1

- Programming Language /1
- IBM 於1960年代中期設計。
- 綜合 Fortran，Cobol 與 Algol 60 之特性。
- 首創提供例外處理 (exception handling) 的能力。
- 採區塊結構 (block structure)。
- 提供遞迴呼叫 (recursive call) 的功能。
- 多重任務 (multi-tasking)。
- 首創指標(pointer)資料型態。
- 首創以維也納定義語言(Vienna Definition Language)來描述語意。

31

範例

一、 PL/1語言題融合了那三種語言的特性而成的程式語言?

sol：Cobol、Fortran、Algol。

PL/1語言有full PL/1及一般子集合PL/1(subset/G)之分，而Fortran，COBOL 等一般高階語言卻沒有。

二、既然PL/1具備的功能很強，為何無廣為流行?

sol：最大的問題是程式碼太大，記憶體無法執行。

32

範例

- 解釋名詞：

  (1) multi-tasking：一部電腦可同時處理數個工作或一個單獨的程式，可分成數個部份同等處理

  (2) multi-programming(多程式)：系統中同時有很多個程式彼此輪流執行

  (3) multi-processing：2個或2個以上的cpu

  (4) multi-processor：同multi-processing

33

SIMULA

- 發展於1960年代後期。
- **提出Coroutines的觀念**
  - **一種subprogram的變形**
- 提供資料抽象化(data abstraction)與類別(class)的觀念。

註：資料抽象化概念是SIMULA首創。

34

C 語言

- 貝爾實驗室於1970年代發展出來。
- 採區塊結構 (block structure)。
- 可攜性高，高度的移轉性➔機器獨立性。
- 適合發展系統程式。
- 具 Self-compiled 特性(C語言本身的編譯程式大部份是用C語言寫成)。
- 可呼叫組合語言。
- Unix系統絕大部份是用C語言寫成

35

C++

- C++ 語言為 Bjarne Stroustrup 設計，在貝爾實驗室中發展。
- C++ 語言設計的主要目標是希望能實現物件導向程式設計的理想，因此C++是以C語言為根本並結合了Simula 67的物件導向概念，及Algol 68的overloading特性而形成最初的C++語言。
- C++ 語言採用區塊結構（Block Structure）。
- C++ 語言非常適合發展系統程式，且利用C++ 語言製作之程式具有很高的可攜性即機器獨立性。

36

C++

- C++ 語言會區分大小寫，如main、Main、MAIN在C++語言中是被視為不同的符號。
- 輸入敘述為"cin”
  語法： cin >> "變數"
  由鍵盤將值讀入變數中。
- 輸出敘述為"cout“
  語法： cout << "資料"
  將“資料”輸出到螢幕上。
- C++語言的識別字由大小寫英文字母，數字或底線所構成，但第一個字元不得為數字。

37

PROLOG

- **PROgramming LOGic**
- Alan Colmeraure 於1970年代初期發展出來。
- 適用於人工智慧之應用，為邏輯式程式語言。又稱為人工智慧的高階語言。
- 易學但執行效率較差
- 邏輯式程式語言是程式設計師只要將與問題有關的事實與問題推演的規則描述出來即可，而不必理會該如何處理該問題
- 1982年日本宣告以 Prolog 為第五代電腦之發展語言。

38

Prolog example

- mother(joanne, jake). father(vern,joanne).
  mother(whenny, nikson). father(michael, chris).
  parent(x,y) :- father(x,y).
  parent(x,y) :- mother(x,y)
  grandparent(x,z) :- parent(x,y), parent(y,z).
- ? grandparent(vern, jake). ➔ true

39

Prolog example

- Speed(ford,100). Speed(chevy,105).
  Speed(dodge,95). Speed(volvo,80).
- Time(ford,20). Time(chevy, 21).
  Time(dodge,24). Time(volvo,24).
- Distance(x,y) :- Speed(X, sp), Time(X, ti), Y is sp*ti
- ? Distance(chevy, dd). ➔
  return dd=2205=105*21

40

PASCAL

- 紀念法國數學家Blaise Pascal。
- 於 1975 年發展完成,由 IBM 贊助。
- 採區塊結構。
- **主要設計用來教導結構話程式設計提供除了一般資料型別以外，還包括自訂型別、首創集合資料型態、指標、傳值及傳址呼叫、case結構及遞迴**

41

Pascal自訂型別

```
Type intlist=array [1.99] of integer;
…
Var
   aa : intlist;
…..
  for cc:=1 to 99 do
    aa[cc]:=aa[cc]+1;
```

42

ADA

- 紀念 Augusta ADA Byron
- 由美國防部發展,主要運用於國防需求上， 提供了資料抽象化，例外處理及平行處理 (rendezvous)等功能。

43

RPG

- **Report Program Generator**
- 由 IBM 發展，主要用作大量報表之產生。

44

GPSS

- **General Purpose Simulation System**
- 主要用於模擬(Simulation)用途。

45

PILOT

- Programmed Inquiry Learning Or Teaching
- 主要應用於電腦輔教學 (Computer Aided Instruction 即 CAI) 可幫助教師編寫教材。

46

FORTH

設計的目標是為了提供對電腦的記憶體及速度作最佳之運用。

47

MODULA-2

由 Wirth 發展出來，適用於系統軟體之開發。

48

JAVA

- 由Sun Microsystems所發展，其名稱之命名是源於突發的靈感。
- 物件導向程式語言，以 class 為基本架構。
- 提供垃圾收集法(garbage collection)來管理使用者不再使用的記憶體空間。
- 提供執行緒(Multithread)功能。
- 提供例外處理(exception) 能力。
- JAVA取消了指標(pointer)資料型態，多重繼承(multiple inheritance)，及運算子覆載(operator overloading)等性質。

49

JAVA

- 利用JAVA語言寫成的程式經由編譯器(compiler)處理後產生的碼稱為Byte Code (中間碼)，這種碼可在不同的機器平台上移植，待要執行時，再由JAVA的直譯器(interpreter)處理此Byte Code即可，因此JAVA語言比 C語言具有更高的可攜性。
- JAVA語言允許其程式段能夠透過網路系統到另一個機器平台上執行。
- 目前在 Windows、Macintosh、SUN、Linux等開發平台上已有的直譯程式可供使用。

50

高階語言的處理器

- 高階語言的處理器主要的作用即是將利用高階語言寫成的程式段翻譯成機器可接受的碼。
- 主要可分成編譯器(compiler)及直譯器(interpreter)二類，說明如下：
  - 1.編譯器：編譯器會對原始程式碼中的每一條敘述，按照先後順序做一次之處理，並產生對應的目的碼。
  - 2.直譯器：直譯器會對原始程式碼中的敘述，按照執行的先後順序做處理，並直接產生程式執行結果。

51

編譯器及直譯器之區別

	直譯器	編譯器
輸入	高階語言寫成的程式碼	高階語言寫成的程式碼
輸出	執行結果	目的碼
時間	慢	快
空間	少	多
除錯特性	佳	差
彈性	差	佳
適合階段	程式開發初期	完成
範例	Basic;Lisp;Prolog;APL;Snobol	C;C++;Pascal;Cobol;Fortran

52

命令式語言與應用式語言

•命令式(imperative)語言:

藉著改變變數之內容以做為控制程式執行的方法，主要的程式語言有 Fortran,Cobol, Basic,Algol, Pascal, PL/1, C 及C++等等。

• 應用式(applicative)語言:

語言藉著函數來表達。因此輸入將作為函數的參數而輸出則為函數的值，主要的程式語言有 Lisp、APL、snobol。

53

範例

為何命令式語言不具備平行運算(parallel computation)的能力呢？

因為受限於計算機結構的影響。

54