資料型態

資科系
林偉川

資料型態的定義

- 資料型態是指一群個體(object)以及作用在這群個體上的運算。

資料型態的分類

- 基本資料型態
- 列舉式資料型態
- 指標資料型態

3

基本資料型態

- 常見的基本資料型態有數字、字元與布林資料型態分別介紹如下：
  - 數值：
    - 整數(integer) (-32768 – 32767)。
    - 實數(real)。
    - 字元(character)。
    - 布林值(Boolean)。

4

範例

- 範例1:
  - 請將布林資料型態的六種基本運算 And, Or, Not, Imply, Equivalence 與Exclusive Or 轉換成相對應的 if 敘述。
    And : If x then y else false
    Or : If x then true else y
    Not : If x then false else true
    Imply : If x then y else true
    Equivalence : If x then y else (Not y)
    Exclusive Or : If x then (Not y) else y

5

範例

- 範例2:
  - 請舉例說明何謂同型異構(polymorphic)？A+B，其中A為整數B為實數？
- 範例3：
  - 寬化(widening) ：將一型態轉化為另一型態其值不會改變。例如：整數轉成實數
  - 窄化(narrowing) ：將一型態轉化為另一型態其值可能會改變。例如：實數轉成整數

6

列舉式資料型態(enumerated data type)的定義

- 列舉式資料型態是將需要的資料一一定義與列舉出來。

7

列舉式資料型態的用法

- 指定敘述。
  enum student {John, Mary, Peter};
- 陣列的註標(index)。
- FOR-LOOP的控制變數。
- 布林運算式中的比較項。

8

實例說明

- 範例一：

  type seasons = (Spring, Summer, Fall, Winter);

  var P, Q, R: seasons;

  P：＝ Spring;

  Q：＝Fall;

  if P＝Q then R：＝ Summer else R：＝Winter

- 範例二：

  for i：＝ Spring to Winter do　　begin … end

9

列舉式資料型態

- Pascal語言中內建的有序型態為整數, 字元與布林值, 而使用者可自行定義的有序型態則包括了列舉式資料型態與子範圍(sub-range)。

10

指標(pointer)

- **意義**
  - 指標是參考(reference)物體的一種方式
- **指標變數的值**
  - 記憶體位址，或 nil/null
- 優點：彈性大，可以任意存取記憶體空間的值，若使用到相同資料，可僅配置一個記憶體空間給該資料，其他使用時可用指標指到該資料即可
- 缺點：安全性較差、存取速度較慢、可能造成懸置引用的問題及可靠性差

11

Pascal語言宣告指標資料型態的範例

- 範例:

  var

  ptr ： ^ integer;

  ptr代表address ➔ int *ptr; C

  ptr^代表value ➔ *ptr=4; C

  - new():配置新的記憶體空間供指標變數使用。 Malloc()➔C
  - dispose():歸還記憶體空間給系統。 Free()➔C

12

C語言的指標資料型態

- *:取值運算子。
- &:取址運算子。
  int *x, y; x=&y;

13

範例

- 範例1：
  - 請簡述C語言的指標資料型態與陣列資料型態在處理上之關連。

  int a[5], *ptr; ptr=a; v.s. ptr=&a[0];
- 範例2：
  - dangling referencing
  - de-referencing (dangling pointer)
  - dangling object

14

Dangling referencing/pointer/object

- 指標變數指到一個不存在的記憶體空間，如Pascal語言中，x是一個指標變數，當執行了dispose(x)後，若想引用x即造成dangling referencing

15

Dangling referencing/pointer/object

- 指標指到一個不存在任何有意義資訊的記憶體空間造成dangling pointer，一般為指標指到一個被呼叫程式中的區域變數，當被呼叫的程式結束後，區域變數即被清掉，而造成dangling pointer
- 一個物件仍然存放資訊的記憶體空間，可是已沒辦法可存取這個空間，即稱此空間為dangling object

16

結構性資料型態

- 將具關連性的資料結合成一個個體的方法, 即稱之為結構性資料型態。

17

陣列

- 陣列主要是由陣列的名稱, 維度(dimension), 元素型態以及陣列索引(Index)等型態組成。

18

陣列的限制

- 陣列元素必須存放在連續的記憶體空間中
- 陣列元素的型態必須完全相同。

19

重點

- 計算陣列元素個數(假設為n個元素)。作法為分別計算陣列中各個維度元素的個數再相乘即可。
- 計算陣列元素佔用之記憶體空間總量(假設陣列個別元素佔用之記憶體空間為w，則陣列元素佔用之記憶體空間總量為n ×w)。
- 計算陣列中特定元素之記憶體位址。

20

固定界限陣列

- 採用靜態儲存區配置法(static storage allocation)

21

動態界限陣列

- 採用動態儲存區配置法(dynamic storage allocation)

22

陣列描述器(array descriptor)

陣列名稱	demo
元素的型態	real
元素的長度	W
陣列的起始位址	α
維度	1
維度的上、下限	3,10

Float demo[8];

23

陣列儲存方式

- 列優先(row major ordering)：
  - 最右邊的index先加1, 再向左進位。
- 行優先(column major ordering)：
  - 最左邊的index先加1, 再向右進位。
- 註：
  - Fortran語言採用行優先，而其他高階語言則多採用列優先。

24

實例

- A(1:4,0:6)
- A為一個二維陣列，左邊的維度有四種可能值(1、2、3及4)，右邊的維度有七種可能值(0、1、2、3、4、5及6)；因此共有4 x7＝28個元素，此28個元素若分別採用列優先或行優先方式儲存，其註標值與相對應之位置如下之說明

25

實例

- 列優先：A(4,6)
- 說明：

  元素(1,0)為陣列A中之第1個元素、(1,1)為陣列A中之第2個元素、...，而元素(4,6)則陣列A中之第28個元素。

26

實例

- 行優先：A(4, 6)
- 說明：

  元素(1,0)為陣列A中之第1個元素、(1,1)為陣列A中之第5個元素、...，而元素(4,6)則陣列A中之第28個元素。

27

位址函數

- 一維陣列
- 二維陣列
- N維陣列

28

一維陣列

- A(l1..u1)
- 陣列A的之元素個數為(u1- l1+1)。
- 陣列A的計算起始位址函數如下：

  Loc(A[i])= α + w*(i-l1) , i >= l1, i <= u1

  – 說明：
  - α:陣列A在記憶體中之起始位址。
  - W:為每個元素所佔用的記憶體空間。

29

範例

- 考慮下列的宣告 var A：array [100..1000] of integer;假設一個integer佔一個memory，問A這個array佔多少memory？901個memory
- 在C語言的例子，假設一個int佔4個位元組，且起始位址為120時：

  int c[100]; 使用c[32]其位址為何？

30

二維陣列

- $A(l_1..u_1, l_2..u_2)$
  - 列優先：陣列A的計算起始位址函數如下：

    $Loc(A[i,j]) = \alpha + w[(i-l_1)*(u_2-l_2+1)+(j-l_2)]$
  - 行優先：陣列A的計算起始位址函數如下：

    $Loc(A[i,j]) = \alpha + w[(j-l_2)*(u_1-l_1+1)+(i-l_1)]$

31

二維陣列

- $A(l_1..u_1, l_2..u_2, l_3..u_3)$
  - 列優先：陣列A的計算起始位址函數如下：

    $Loc(A[i,j,k]) = \alpha + w[(i-l_1)*(u_2-l_2+1)*(u_3-l_3+1)+(j-l_2)*(u_3-l_3+1)+(k-l_3)]$
  - 行優先：陣列A的計算起始位址函數如下：

    $Loc(A[i,j]) = \alpha + w[(j-l_2)*(u_1-l_1+1)+(i-l_1)]$

32

範例

- 三維陣列(array) A[0:4;0:5;1:6]裡共有幾個元素(elements)？
- 解：

  A為一個三維陣列，最左邊的維度有5種可能值(0、1、2、3及4)，中間的維度有6種可能值(0、1、2、3、4及5)，最右邊的維度有6種可能值(1、2、3、4、5及6)；因此共有5 ×6 ×6＝180個元素。

33

範例

- 一陣列 (Array) 被以列為主的順序存放在記憶體內，每個陣列元素佔用4個位元組的記憶體，若起始位置是100，下列宣告中所　列元素的存放位置為何？

  (1) Var A：array[-100..1, 1..100]求 A[1, 12] 的位址。
  40544(列為主) 、4992 (行為主)

  (2) Var A：array[5..10, -10..20]求 A[5, -5] 的位址。
  120 (列為主) 、 220(行為主)

34

答案

- 解：(1) 40544 (2) 120
  - (1)Loc(A[1, 12])
    ＝100＋4 ×[(1-(-100)) ×(100-1+1)+(12-1)]
    ＝40544
  - (2)Loc(A[5, -5])
    ＝100＋4 ×[(5-5) ×(20-(-10)+1)+(-5-(-10))]
    ＝120

35

範例

- 一陣列(Array)每個陣列元素佔用4個位元組的記憶體，若A(3,4)的起始位置是1640，A(4,4)的起始位置是1680，A(5,5)存放位置為何？
  LOC(A(3,4))=α+4*()=1640，
  LOC(A(4,4))=α+4*()=1680，所以
  (1680-1640)/4=10➔A(3,4) — A(3,10)，
  A(4,1) — A(4,4)共10個，所以
  LOC(A(5,5))=LOC(A(4,4))+4*11=1680+11*4
  =1724

36

陣列註標(index)的語法

- 假設AAA為陣列名稱,I為註標則
  - 小括號:AAA(I) ➔ VB/VB.NET
  - 中括號:AAA[I] ➔ C/JAVA

37

陣列註標的型態

- Fortran➔ ()
- Pascal➔[]
- Ada➔()

38

記錄(record)

- 記錄是由固定數目, 但型態可以不同的元素所組成。

39

實例說明

- Pascal language

```
Type StudentData = record
                       Name : string[30];
                       Age : integer;
                       PhoneNo : integer;
                       Sex : char;
                       Address : string[50]
                   end;
struct student {
  char name[30]; int age, phoneNo; char Sex;
  char Address[50]; };
```

40

記錄引用的方式

- 函數式記號:
  - Knuth 提出。
  - ALGOL 68 採用。
    - 如 var example :
    - StudentData;
    - Name(example),
    - Age(example),
    - Phone(example),
    - Sex(example),與Address(example)

41

記錄引用的方式

- 限定名稱格式:
  - Pascal、Ada、PL/1及Modula-2 採用， 如

```
var example : StudentData;
…
example.Name,
example.Age,
example.Phone,
example.Sex, 與 example.Address
```

42

記錄的屬性(attributes)

- 欄位的名稱
- 欄位的資料型態
- 欄位的個數

43

聯合資料型態(union data type)

- 對同一個記憶體的空間而言在不同的執行時間可存放不同型態的值。

44

提供聯合資料型態的語言

- Pascal:
  - 實例:

```
type T= record
        price : real;
        case sex : (M, F) of
          M : (age : integer);
          F : (blood : (A, B, O, AB))
        end
      end;
```

45

提供聯合資料型態的語言

- Ada
- Fortran

46

範例

- 有一 PASCAL 之 type 宣告如下:

```
type T = record
        price : real;
        case sex:(M,F) of
           M:(age : integer);
          F:(blood : (A,B,O,AB));
        end;
   end;
var a : T;
```

請問 Pascal之Variant record type 有什麼不安全的地方？改age後blood亦受影響。

若sex=M，a.age:=20; 但a.blood:='B';也對➔不安全的地方

47

範例

- Ada程式語言如何避免 Pascal 語言中變異記錄的不安全性？檢查欄位之合法性
- Fortran程式語言用EQUIVALENCE來達成聯合資料型態的作用。

```
INTEGER X
REAL Y
EQUIVALENCE (X,Y)
```

48

各語言字串處理法介紹

- 由一組字元所組成之集合, 便稱之為字串。
- PL/1語言首先提供了對字串的運算處理功能。
- PL/1語言宣告字串格式的方式。
  - 宣告具固定長度。
    - 如 DCL P CHAR(50);
  - 變動長度, 但其長度有限制。
    - 如 DCL Q CHAR(50) VARYING;
  - 利用"PICTURE"指令來宣告。➔COBOL
    - 如 DCL R 'AAAA'

49

Cobol語言

- 基本資料類別有三種, 分別是:
  - Alphabetic: A
  - Alphanumeric: X
  - Numeric: 9

50

PL/1語言六個內建運算子

- |:連接運算子
- INDEX(X,Y):若字串Y是字串X的部份字串則傳回字串Y在字串X中位置的整數值,否則傳回0
- LENGTH(X):字串X的長度
- VERIFY(X,Y):若字串X的每個字元都在字串Y中出現則傳回0,否則傳回i;其中i表示字串X的第一個不在字串Y中出現字元的位置
- SUBSTR(X,i,j):取子字串。i 表示開始位置,j表示長度。
- TRANSLATE(X,Y,Z):轉換函數。將X字串中的Z字元以Y字元替代

51

Pascal語言

- ORD(C):傳回字元 C 在 ASCII CODE 中的順序。
- CHR(X):傳回 ASCII CODE 碼為 X 的字元,即傳回值是字元。

52

SNOBOL 語言

- SNOBOL 語言具有很強的字串處理能力。
  - 型態比對(Pattern match) 的格式如下:

    [標記] 主體 字樣 [:GOTO]

    [標記] 主體 字樣:個體 [:GOTO]
- **處理字串的方法:**
  - 宣告具有固定之長度。**(Pascal)**
  - 字串的長度可變動, 但有範圍的限制。**(PL/I)**
  - 字串的長度沒有任何限制。（SNOBOL)

53

集合(set)

- 集合是用來儲存沒有順序關係的資料。
- 集合資料型態是由Pascal語言所首創。
- [] ➔ 空集合
- + ➔ 聯集運算
- * ➔ 交集運算
- - ➔ 差集運算

54

可變長度資料結構

- 可變長度資料結構的意義
  - 指執行時資料結構的長度可以變動。
- 靜態資料結構與動態資料結構的比較
  - 靜態資料結構(static data structure):
    - 資料大小在編譯階段被設定，執行時不能改變
  - 動態資料結構(dynamic data structure):
    - 資料大小在編譯階段未被設定，執行時可以改變

55

常見的可變長度資料結構

- 鏈結串列(linked list)
- 佇列(queue)
- 堆疊(stack)
- 樹(tree)

56

靜態與動態型態檢驗

- 靜態型態檢驗(static type checking):
  - 變數的型態在編譯時即已決定稱之。
- 動態型態檢驗(dynamic type checking):
  - 變數的型態在執行時決定,稱之。

57

靜態與動態型態檢驗的比較

- 靜態型態檢驗:
  - 優點:產生有效率的目的碼
  - 缺點:欠缺彈性
- 動態型態檢驗:
  - 優點: 彈性較佳
  - 缺點:處理過程複雜

58

型態強制轉換

- 型態強制轉換(coercion)的意義
  - 指將某一型態的值強制轉換成為另一種型態之值
- 型態強制轉換的作法
  - 拓寬法(widening)
    - 將某一型態的值轉換為另一型態, 而值不會改變者稱之。
  - 縮窄法(narrowing)
    - 將某一型態的值轉換為另一型態, 而值會改變者稱之。

59

強制型態語言 (Strongly Typed Language)

- 在編譯時變數的型態可被決定
- 在編譯時可檢查同一運算式中，所有變數的相容性
- 若允許同一記憶體空間在不同時刻儲存不同型態之值時，必須能確保其安全性

60

型態相等性

- 型態相等性一般分為三種:
  - 名稱相等法(name equivalence)
    - 若變數用相同型態名稱宣告或一起宣告即滿足名稱相等法
  - 結構相等法(structure equivalence)
    - 變數的各項元件皆相等, 即滿足結構相等法
    - 而陣列元素則應考慮維度，若二陣列元素個數相同，但維度不同則視為不滿足結構相等法
  - 宣告相等法(declaration equivalence)
    - 變數必須一起宣告始滿足宣告相等法

61

範例

- 以下的變數宣告
  - 採用結構相等法,則變數 a, b, c, d 那幾個被宣告為相同型態？a,b,c,d
  - 採用名稱相等法呢？a,d及b,c
  - 採用宣告相等法呢？b,c

```
type
   vector = array [1..10] of integer;
var
   a: vector:
   b, c: array [1..10] of integer;
   d: vector;
```

62

範例

- 假設

  Type Q=array [1..100] of real;

  Var

  w,m:Q;

  x:array [301..400] of real;

  y,z:array [301..400] of real;

(1) 請問有那些變數為結構相等法？ x,y,z及w,m

(2) 請問有那些變數為名稱相等法？ x,y,z及w,m

(3) 請問有那些變數為宣告相等法？y,z及w,m

63

範例

- 對於一程式語言的編譯程式而言，可使用名稱相等或結構相等的方式，來決定程式中的變數型態是否相容，

請說明：

(1)何謂名稱相容？

(2)何謂結構相容？

(3)採用那一種方式，對編譯程式而言較容易處理，為什麼？名稱相容

64