Chương 7: Ngôn ngữ lập trình hàm

Giảng viên: Ph.D Nguyễn Văn Hòa

Khoa KT-CN-MT – ĐH An Giang

Một số đặc trưng của NN mệnh lệnh

- Sử dụng nguyên lý tinh chế từng bước, hay mịn dần
- Khai báo dữ liệu để nối kết tên biến → trị
- Các kiểu dữ liệu cơ bản → kiểu dữ liệu có cấu trúc
- Cấu trúc điều khiển từng tự, rẽ nhánh, gọi chương trình con
- Hiệu ứng lề
- Lập trình cấu trúc: khối, chương trình con, module

Nội dung chính của chương

- Giới thiệu
- Hàm toán học
- Dạng hàm
- Bản chất của lập trình hàm
- Ngôn ngữ LISP

Giới thiệu

- Ngôn ngữ lập trình mệnh lệnh được xây dựa trên nguyên lý kiến trúc máy tính của *von Neumann*
  - Đơn trị làm việc trong chương trình là câu lệnh
  - Các NNLT Fortran, Pascal, Ada… sự hiệu quả quan trọng hơn là sự thích hợp để phát triển phần mềm
- Ngôn ngữ lập trình hàm (LTH) được xây dựng dựa trên các hàm toán học → ngôn ngữ không ra lệnh
  - Vì dựa trên nguyên lý hàm toán học nên LTH gần gủi với người dùng hơn, nhưng LTH thì không liên hệ chặt chẽ với kiến trúc máy tính

Hàm toán học

- Mỗi hàm toán học là một ánh xạ các phần tử của tập hợp (miền xác định) với các phần tử của tập hợp khác (miền giá trị)
- Mỗi phần tử của miền xác định tương ứng một phần tử của miền giá trị
- Mỗi định nghĩa hàm xác định miền xác định, miền giá trị và quy tắc tương tác (ánh xạ)
- Định nghĩa hàm
  - Tên hàm + danh sách tham số ≡ biểu thức
  - VD lap_phuong(x) ≡ x*x*x ;

Biểu thức *lambda*

- Đôi khi người ta dùng hàm không tên → sử dụng biểu thức *lambda*
  - *VD* `λ(x) x * x * x` thay thế lap_phuong(x) ≡ x*x*x
- Tham số biểu thức *lambda* gọi là tham số biến kết ghép
- Khi biểu thức *lambda* được định trị đối với một tham số đã cho → biểu thức được áp dụng cho tham số đó
  - `(λ(x) x * x * x)(2)` có giá trị là 8

Ngôn ngữ lập trình hàm

- Tập hợp các đối tượng dữ liệu
- Các hàm nguyên thủy
- Các dạng hàm
- Tác vụ áp dụng hàm

Đối tượng dữ liệu

- Đối tượng dữ liệu của ngôn ngữ lập trình hàm rất đơn giản
  - Nguyên tử (Atom): một chuỗi các ký tự; ABC, 123, Z34
  - Hai nguyên tử đặc biệt: T và F
  - Dãy n các đối tượng x_1, x_2, …, x_n được ký hiệu là < x_1, x_2, …, x_n>
  - NIL là ký hiệu dãy rỗng

Hàm nguyên thủy

- Là các hàm được định nghĩa sẵn trong ngôn ngữ
- Bón nhóm dạng hàm nguyên thủy
  - Nhóm hàm số học: +, -,*, /
  - Nhóm hàm vị từ: ATOM và NULL
  - Hàm đồng nhất ID:x ≡ x
  - Nhóm liên quan đến cấu trúc danh sánh (dãy)

Dạng hàm

- Dạng hàm là sự tổ hợp của các hàm: một hàm (chương trình) được xây dựng từ các hàm sẵn có bằng cách sử dụng các tác vụ tạo chương trình
- Các dạng hàm
  - Hàm hợp
  - Hàm xây dựng
  - Hàm áp dụng cho tất cả

Hàm hợp (*composition*)

- Hàm hợp là hàm dùng 2 hàm như là 2 tham số và dùng kết quả của hàm đầu tiên như là tham số thực cho hàm thứ 2
- VD hàm hợp của *h*

  `h ≡ f ° g`

  nghĩa là `h(x) ≡ f ( g ( x))`

  Nếu `f (x) ≡ x + 2` và `g (x) ≡ 3 * x`,

  `h ≡ f ° g` tương đương `(3 * x)+ 2`

Hàm xây dựng (*construction*)

- Hàm xây dựng là hàm mà các tham số của chúng cũng là hàm
- Ký hiệu hàm xây dựng: []
- Khi áp dụng vào một đối số vào hàm xây dựng: đối số được áp dụng vào hàm tham số → tập hợp kết quả
- VD
  - G(x) ≡ x*x, H(x) ≡ 2*x và I(x) ≡ x/2
  - [G,H,I](4) có kết quả là (16,8,2)

Áp dụng cho tất cả

- Là hàm lấy một hàm đơn như là một tham số
- Hàm áp dụng cho tất cả được ký hiệu là α
- Áp dụng hàm tham số vào danh sách các đối số, ta được một danh sách kết quả
- VD `h (x) ≡ x * x`

  `α( h, (2, 3, 4))` kết quả `(4, 9, 16)`

Bản chất của NN lập trình hàm

- Mục đích của việc thiết kế NN LTH là mô phỏng các hàm toán học một cách nhiều nhất có thể được
- Tiến trình tính toán trong NN LTH khác NN LT mệnh lệnh:
  - Trong LT mệnh lệnh, biểu thức được định giá và kết quả của nó được lưu trữ trong ô nhớ
  - Quản lý biến là 1 trong những nhân tố làm phức tạp NNLT mệnh lệnh
- Trong NN LTH, biến không cần thiết, như là trong trường hợp của toán học

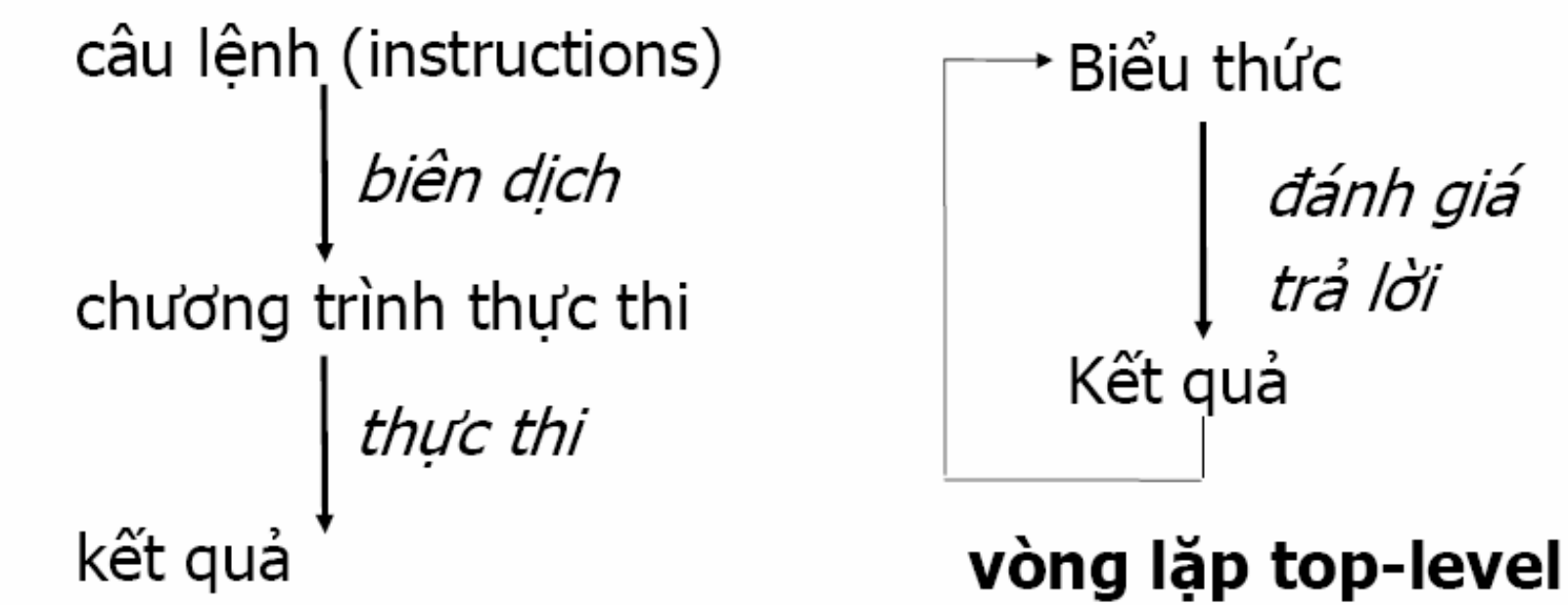
Bản chất của NN lập trình hàm (tt)

- Các lệnh lập lại sẽ được xử lý bằng đệ qui
- Chương trình là các định nghĩa hàm và các áp dụng hàm
- Sự thực hiện là việc đánh giá các áp dụng hàm
- Sự thực hiện một hàm luôn cho cùng một kết quả khi ta cho nó cùng một đối số
  - VD f(x) + f(x) và 2 * f(x) luôn luôn cùng kết quả
- Ngữ nghĩa của ngôn ngữ lập trình hàm đơn giản hơn ngữ nghĩa của ngôn ngữ lập trình mệnh lệnh

Bản chất của NN lập trình hàm (tt)

- Ngôn ngữ hàm cung cấp một tập hợp các hàm nguyên thủy và một tập các dạng hàm
- Ngoài còn cung cấp một phép toán áp dụng hàm và các cấu trúc lưu trữ dữ liệu
- Ngôn ngữ hàm được thiết kế tốt là LISP

NN biên dịch vs NN thông dịch



LISP: giới thiệu

- Được John McCarthy đề xuất vào năm 1958
- Trình biên dịch LISP đầu tiên được viết bởi Tim Hart và Mike Levin (1962) bằng chính ngôn ngữ LISP
- LISP là một trong những ngôn ngữ LT sớm nhất
- LISP đã được sử dụng rộng rãi và phát triển mạnh trong lĩnh vực trí tuệ nhân tạo trong 1980s
- Common Lisp chuẩn ra đời năm 1984

Ưu điểm của LISP

- Cú pháp đơn giản
  - Cấu trúc dữ liệu duy nhất: danh sách
  - Không có lệnh, không từ khóa
  - Tất cả các hàm đều viết ở dạng hàm
- Là một ngôn ngữ mạnh nhờ tính tương đương giữa dữ liệu và chương trình
- Mềm dẻo và dễ phát triển

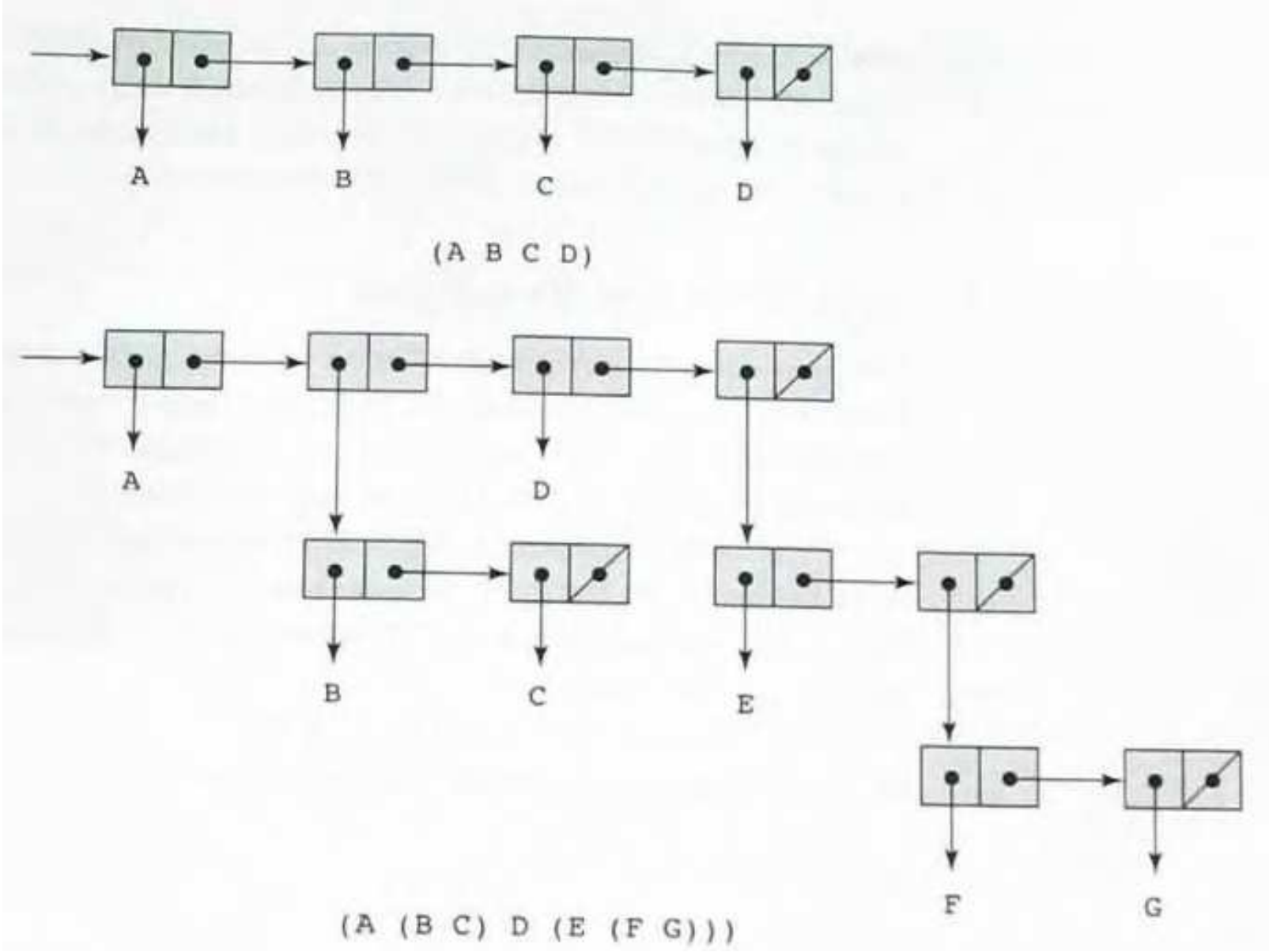
Các khái miện cơ bản

- Nguyên tử (Atom): là một đối tượng cơ bản của LISP, nguyên tử có thể là số hoặc ký hiệu
  - Số: giống như trong các NNLT khác như Pascal, C…

    VD : các hằng số 5, -20, 10.45, 18.2E+5
  - Ký hiệu (symbol): là chuỗi các ký tự (trừ các ký tự đặc biệt, dấu ngoặc & khoảng trống)
    - Các ký hiệu được bắt đầu băng dấu «‘»
    - VD ‘a, ‘anh, ‘anh_ba
    - Ký hiệu số được xem là số; VD ‘5 = 5

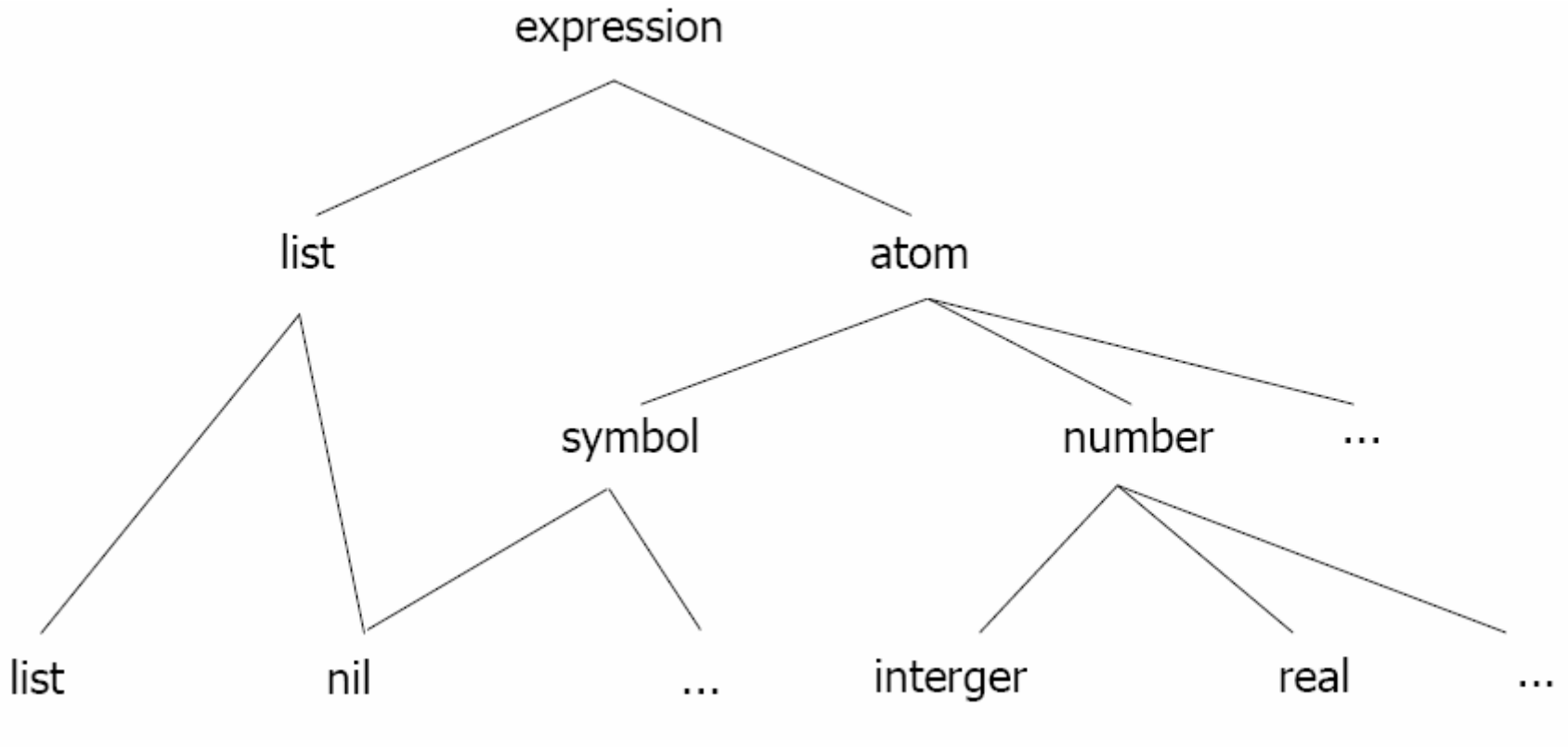
Các khái miện cơ bản (tt)

- Danh sánh: dãy các có phân biệt thứ tự của các phần tử, cách nhau ít nhất một khoảng trắng và đặt nằm trong cặp dấu ngoặc đơn ()
  - Phần tử của danh sách có thể là một nguyên tử hoặc là một danh sách
  - Các phần tử không nhất thiết có cùng kiểu
  - Hằng danh sách được mở đầu bằng dấu «‘»
  - VD
    - ‘() Danh sách rỗng, tương đương ký hiệu NIL
    - ‘(a 5 c) Danh sách gồm 3 phần tử
    - ‘(3 (b c) d (e (f g))) Danh sách gồm 4 phần tử


A
B
C
D
(A B C D)
A
D
B
C
E
F
G
(A (B C) D (E (F G)))

Các khái miện cơ bản (tt)

- Phân cấp dữ liệu



Các khái miện cơ bản (tt)

- Biểu thức: một nguyên tử hoặc một danh sách, luôn luôn có trị được định theo nguyên tắc:
  - Nếu là một số : trị là chính số đó

    VD >25, = 25
  - Nếu là ký hiệu: trị có thể là
    - Được định trước bởi LISP; VD t (TRUE), nil (NIL)
    - Giá trị dữ liệu của người sử dụng, trị gán cho biến

      VD >(setq a 3); gán 3 cho biến a
  - Nếu là danh sách có dạng (E_0, E_1,…, E_n) thì trị là
    - Phần tử đầu tiên phải là hàm đã được biết bởi LISP
    - Các phần tử E_1,…, E_n được định trị từ trái sang phải
    - VD >(+ 5 3 6) = 14; >(+ 4 (+ 3 5)) = 12

Các hàm

- Một chương trình của LISP là một hàm hoặc một hàm hợp
- Các hàm có thể được LISP định nghĩa hoặc do người dùng định nghĩa
- Hàm số học: +, -, *, /, 1+, 1-, MOD, SQRT: tác động lên biểu thức số và cho kết quả là số

  VD >(+ 5 6 2) =13

  >(- 8 3) = 5

  >(- 8 3 1) =4

  >(1+ 5) ; Tương đương (+ 5 1)

  >(1- 5) ; Tương đương (- 5 1)

  >(MOD 14 3) hoặc >(sqrt 9)

Các hàm (tt)

- Các hàm so sánh các số <, >, <=, >=, = và /=, cho kết quả là T hoặc NIL
  - VD
  - >(< 4 5) = T
  - >(> 4 (* 2 3)) = NIL
- (EQ s_1 s_2) so sánh xem hai ký hiệu s_1 và s_2 có giống nhau hay không
  - >(eq ‘tuong ‘tuong) = T
  - >(eq ‘tuong ‘duong) = NIL
  - >(eq ‘5 5) = T

Các hàm (tt)

- (EQUAL o_1 o_2) so sánh xem đối tượng bất kỳ o_1 và o_2 có giống nhau hay không
  - >(equal ‘(a b c) ‘(a b c)) = T
  - >(equal ‘(a b c) ‘(b a c)) = NIL
  - >(equal ‘a ‘a) = T

Các hàm thao tác trên danh sách

- CAR: nhận vào danh sách (DS) L, trả về phần tử đầu tiên của L
  - > (CAR '(1 2 3)) = 1
  - > (CAR 3) Error: bad argument type - 3
  - >(CAR nil) = NIL
  - > (CAR '((a b) 1 2 3)) = (A B)
- CDR: nhận DS L, trả về DS sau khi bỏ phần tử đầu tiên
  - >(cdr '(1 2 3)) = (2 3)
  - >(cdr 3) Error: bad argument type – 3
  - >(CAR (CDR ‘(a b c))) = B

Các hàm thao tác trên DS (tt)

- Viết gộp các hàm: ta có thể dùng hàm C..A/D..R để kết hợp nhiều CAR và CDR
  - VD (CADR ‘(a b c)) = B
- (CONS x L) nhận vào phần tử x và danh sách L, trả về một danh sách, có được bằng cách thêm phần tử x vào đầu danh sách L
  - >(CONS 3 '(1 2 3)) = (3 1 2 3)
  - >(CONS 3 nil) = (3)
  - >(CONS '(a b) '(1 2 3)) = ((A B) 1 2 3)

Các hàm thao tác trên DS (tt)

- (APPEND L_1 L_2 ... L_n)
  - Trả về DS là nối kết của DS L_1 L_2 ... L_n
  - (append ‘(A) ‘(B C)) = (A B C)
- (LIST E_1 E_2 ... E_n) nhận vào n biểu thức E_1, E_2, ..., E_n, trả về danh sách bao gồm n phần tử V_1, V_2, ..., V_n, trong đó E_i là giá trị của biểu thức E_i (i=1..n)
  - >(list 1 2) = (1 2)
  - >(list 'a 'b) = (A B)
  - >(list 'a 'b (+ 2 3 5)) = (A B 10)

Các hàm thao tác trên DS (tt)

- (LENGTH L)
  - Trả về tổng số phần tử trong của DS L
  - (length ‘(A B C D)) = 4
- (REVERSE L)
  - Trả về DS đảo của DS L
  - (reverse ‘(1 2 3)) = (3 2 1)
  - (reverse ‘(A B C D)) = (D C B A)
- (FIRST L)
  - Trả về phần tử đầu tiên của DS L
  - (first ‘(A B C D)) = A

Các hàm thao tác trên DS (tt)

- (REST L)
  - Trả về DS sau khi bỏ phần tử đầu tiên của DS L
  - (rest ‘(A B C D)) = ‘(B C D)
- (LAST L)
  - Trả về phần tử cuối cùng của DS L
  - (last ‘(A B C D)) = D

Các vị từ kiểm tra

- (ATOM a) xét xem a có phải là một nguyên tử
- (NUMBERP n) xét xem n có phải là một số
- (LISTP L) xét xem L có phải là một danh sách
- (SYMBOLP S) xét xem S có phải là một ký hiệu
- (NULL L) nhận vào 1 danh sách L. Nếu L rỗng thì trả về kết quả là T, ngược lại thì trả về kết quả là NIL
  - >(atom 'a) = T; >(numberp 4) = T; >(symbolp 'a) = T
  - >(listp '(1 2)) = T; >(symbolp NIL) = T;
  - >(null NIL) = T; >(null ‘(a b)) = NIL; >(null 10) = NIL

Các hàm logic AND, OR và NOT

- (AND E_1 E_2.. E_n) nhận vào n biểu thức E_1, E_2,... E_n
  - AND định trị các biểu thức E1 E2... En từ trái sang phải
  - Nếu gặp một biểu thức là NIL thì dừng và trả về kết quả là NIL. Nếu tất cả các biểu thức đều khác NIL thì trả về giá trị của biểu thức En
  - >(AND (> 3 2) (= 3 2) (+ 3 2)) = NIL
  - >(AND (> 3 2) (- 3 2) (+ 3 2)) = 5

Các hàm logic (tt)

- (OR E_1 E_2 ... E_n) nhận vào n biểu thức E_1, E_2, ... E_n
  - OR định giá các biểu thức E_1 E_2... E_n từ trái sang phải
  - Nếu gặp một biểu thức khác NIL thì dừng và trả về kết quả là giá trị của biểu thức đó
  - Nếu tất cả các biểu thứcđều là NIL thì trả về kết quả là NIL
  - >(OR (= 3 2) (+ 2 1) (list 1 2)) = 3
  - >(OR (= 2 1) (Cdr ‘(a)) (listp ‘(3))) = T

Các hàm logic (tt)

- (NOT E) nhận vào biểu thức E
  - Nếu E khác NIL thì trả về kết quả là NIL, ngược lại thì trả về kết quả là T
  - (NOT (listp '(1 2)) = NIL
  - (NOT (eq ‘tuong ‘duong)) = T

Các hàm điều khiển

- (IF E_1 E_2 E_3) nhận vào 3 biểu thức E_1, E_2 và E_3 Nếu E_1 khác NIL thì hàm trả về giá trị của E_2 ngược lại trả về giá trị của E_3
- (IF E_1 E_2) tương đương (IF E_1 E_2 NIL)
- Nếu E_2 khác NIL thì (IF E_1 E_2 E_3) tương đương (OR (AND E_1 E_2) E_3)

Các hàm điều khiển (tt)

(COND ($ĐK_1$ E_1)
($ĐK_2$ E_2)
.....
($ĐK_n$ E_n)
[(T E_{n+1})]
)

- Nếu $ĐK_1$ khác NIL thì trả về kết quả là giá trị của E_1, ngược lại sẽ xét $ĐK_2$.
- Nếu $ĐK_2$ khác NIL thì trả về kết quả là giá trị của E_2, ngược lại sẽ xét $ĐK_3$...
- Nếu $ĐK_n$ bằng NIL, thì trả về kết quả là trị của E_{n+1}

Các hàm điều khiển (tt)

- (PROGN E_1 E_2 ... E_n) nhận vào n biểu thức E_1, E_2,... E_n
  - Hàm định trị các biểu thức E_1, E_2,... E_n từ trái sang phải và trả về kết quả là giá trị của biểu thức E_n
- (PROG1 E_1 E_2 ... E_n) nhận vào n biểu thức E_1, E_2,... E_n
  - Hàm định trị các biểu thức E_1, E_2,... E_n từ trái sang phải và trả về kết quả là giá trị của biểu thức E_1

Hàm do người dùng định nghĩa

- Cú pháp định nghĩa hàm là:
  - (defun <tên hàm> <danh sách các tham số hình thức>
    <biểu thức>
)
- Ví dụ 1:
  - Định nghĩa hàm lấy bình phương của số a
  - (defun binh_phuong (a)
    (* a a)
)
  - >(binh_phuong 5) = 25

Hàm do người LT định nghĩa (tt)

- Ví dụ 2:
  - Định nghĩa hàm DIV chia số a cho số b, lấy phần nguyên
  - Trước hết ta có: a DIV b = (a – a MOD b)/b

```
(defun DIV (a b)
    (/ (- a (MOD a b)) b)
)
```

  - >(div 6 (div 4 2)) = 3

Đệ qui

- Mô tả một đệ quy bao gồm:
  - Có ít nhất một trường hợp "dừng" để kết thúc việc gọi đệ quy
  - Lời gọi đệ quy phải bao hàm yếu tố dẫn đến các trường hợp "dừng"
- Ví dụ 1: hàm giai thừa

```
(defun giai_thua (n)
    (if (= n 0) 1 (* n (giai_thua (1- n))))
)
```

  - (giai_thua 4) = 24

Đệ qui (tt)

- Ví dụ 2: hàm DIV chia a cho b lấy phần nguyên

```
(defun DIV (a b)
   (if (< a b) 0 (1+ (DIV (- a b) b)))
)
```

- Ví dụ 3: hàm truy xuất phần tử thứ i trong DS L

```
(defun phan_tu(i L)
   (cond
      ((Null L) “Khong ton tai”)
      ((= i 1) (car L));
      (T (phan_tu (1- i) (cdr L)))
   )
)
```

Các hàm nhập xuất

- (Load <Tên tập tin>)
  - Nạp một tập tin vào cho LISP và trả về T nếu việc nạp thành công, ngược lại trả về NIL
  - Tên tập tin theo quy tắc của DOS
  - Dùng để load tập tin để nạp tâp tin CT trước khi gọi lời thực hiện các hàm trong tập tin đó
  - VD >(Load “D:\btlisp\bai1.lsp”)

Các hàm nhập xuất (tt)

- (READ)
  - Đọc dữ liệu từ bàn phím cho đến khi gõ phím Enter
  - trả về kết quả là dữ liệu được nhập từ bàn phím
- (PRINT E)
  - In tra màn hình trị biểu thức E đồng thời xuống dòng trả về giá trị của E
- (PRINC E)
  - In ra màn hình giá trị của biểu thức E (không xuống dòng) và trả về giá trị của E
- (TERPRI)
  - Đưa con trỏ xuống dòng và trả về NIL

Biến toàn cục và biến cục bộ

- Biến toàn cục
  - Là biến mà phạm vi của nó là tất cả các hàm
  - Hàm (**SETQ** <tên biến> <biểu thức>)
  - VD >(setq x (* 2 3)) = 6 ↔ biến x vẫn tồn tại và có giá trị là 6
- Biến cục bộ
  - Phạm vi chỉ nằm trong hàm mà nó được tạo ra
  - Hàm (LET ((var1 E_1) (var_2 E_2) ... (var_k E_k)) E_{k+1}... E_n)
  - VD >(Let ((a 3) (b 5)) (* a b) (+ a b)) = 8

Biến toàn cục và biến cục bộ (tt)

- Biến cục bộ che biến toàn bộ
  - Khai báo biến cục bộ trong hàm LET gây khó khăn

cho việc viết chương trình hơn là sử dụng biến toàn cục

  - Giải pháp: kết hợp cả hai hàm LET và SETQ để sử dụng biến cục bộ che biến toàn cục
  - VD (LET ((var E1)…..)

    …….

    (SETQ var E2)

    ……

)

Biến toàn cục và biến cục bộ (tt)

```
(defun giai_ptb2 ()
  (let ((d 0) (e 0) (f 0))
     (print ‘’Chương trình giải phương trình bậc 2’’)
     (princ ‘’ Nhập số a: ‘’) (setq d (read))
     (princ ‘’ Nhập số b: ‘’) (setq e (read))
     (princ ‘’ Nhập số c: ‘’) (setq f (read))
     (ptb2 d e f)
   )
)
```

- Sau khi thực hiện xong chương trình này thì các biến d, e và f được giải phóng

Exercise

1. Viết hàm có 3 đối số và tính tích của hai số lớn nhất
2. Viết hàm consp kiểm tra xem đối số của nó có phải là một danh sách không rỗng
3. Viết hàm tính $\mathbf{x^n}$.
4. Viết hàm tính độ dài một chuỗi.
5. Viết hàm thực hiện phép nối hai chuỗi.

   (noi ‘(1 2 3) ‘(4 5)

   (1 2 3 4 5)
6. Viết hàm **run** để có đối số là một danh sách các số nguyên L, hàm này sẽ trả về một danh sách 2 chiều các dãy tăng (run) trong L

   >(run ‘(1 2 3 2 1 5 4 7)) = ((1 2 3) (2) (1 5) (4 7))