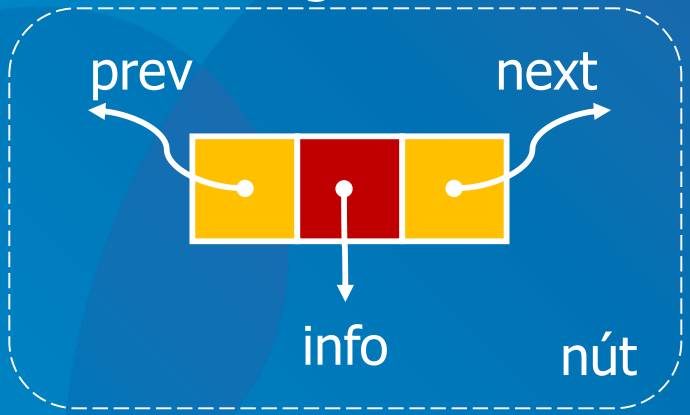
Chương 2.4

Danh sách liên kết đôi

Danh sách liên kết đôi

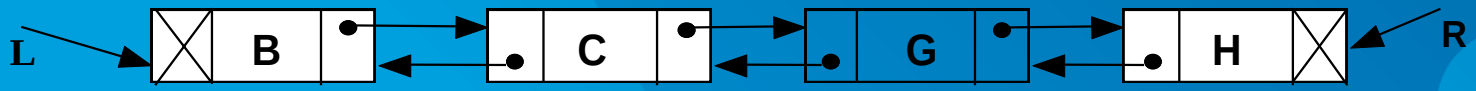
- Qui cách của nút trong danh sách liên kết đôi



Trường PREV của nút đầu tiên và trường NEXT của nút cuối cùng đều có giá trị NULL

Cần nắm được hai con trỏ, con trỏ L trỏ tới nút cực trái, con trỏ R trỏ tới nút cực phải của danh sách

Với danh sách rỗng , L = R = NULL



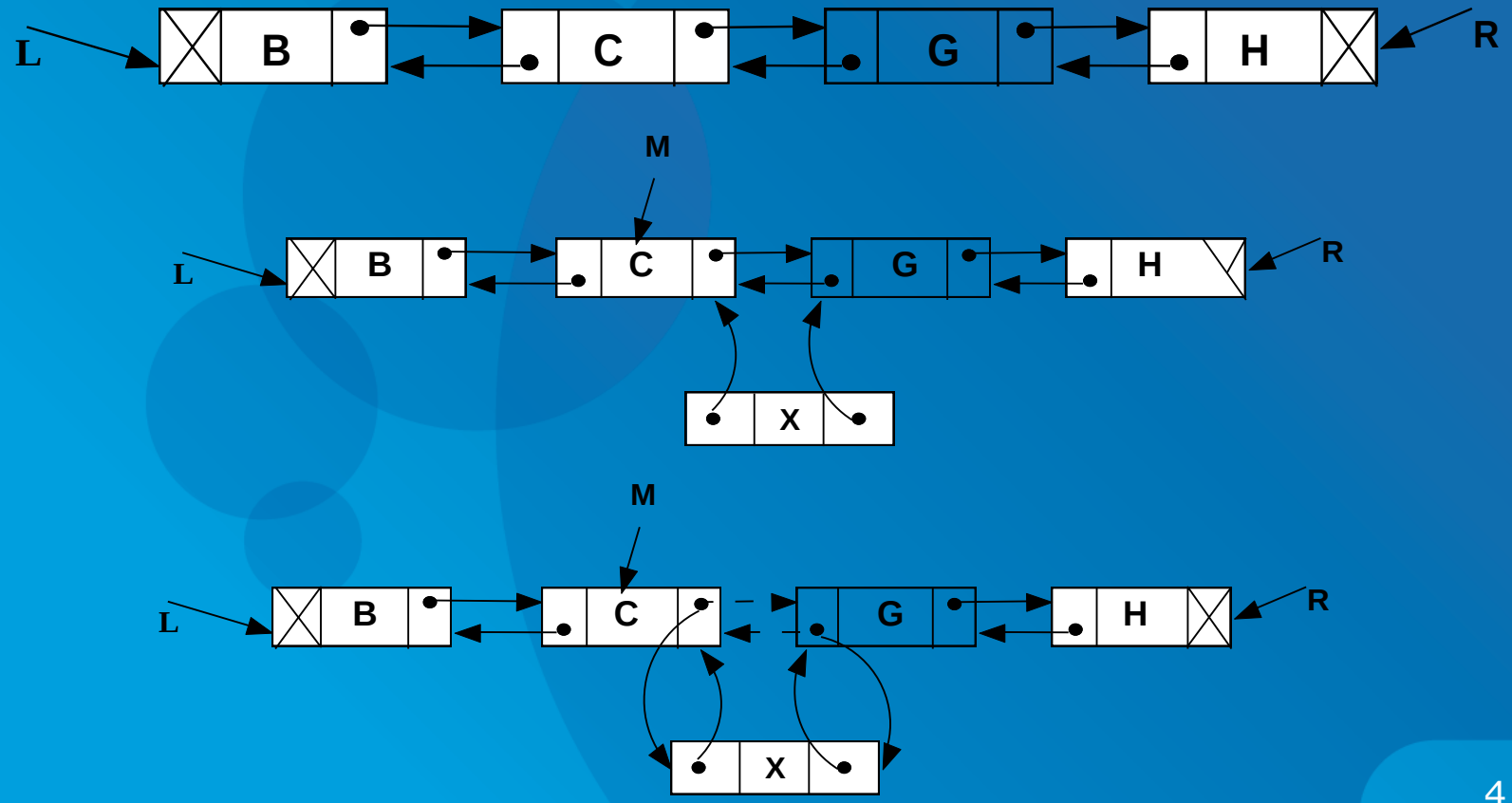
Danh sách liên kết đôi

- Khai báo danh sách liên kết đôi trong C

```
struct dlnode{
        int  info; //type of item
        struct dlnode *next;
        struct dlnode *prev;
} ;
typedef struct dlnode DLNODE;
typedef DLNODE *DLNODEPTR;
DLNODEPTR left, right;
```

Các thao tác trên danh sách liên kết đôi

Bổ sung một phần tử vào sau một nút được trỏ bởi con trỏ M biết trước



Các thao tác trên danh sách liên kết đôi

Giải thuật bổ sung một phần tử mới vào danh sách liên kết đôi

```
Procedure INSERT-DOUBLE (L, R, M, X)
{Bổ sung một phần tử chứa dữ liệu X vào sau phần tử trỏ bởi
   M}
1. {Tạo lập nút mới}
   call New(p) ; {xin cấp phát một nút mới có địa chỉ là p}
   INFO(p) := X;
2.  {Danh sách rỗng}
   if L = R= NULL then begin
                              PREV(p):= NEXT(p) := NULL;
                              L:= R:=p;
                              return;
                        end;
(Còn tiếp)
```

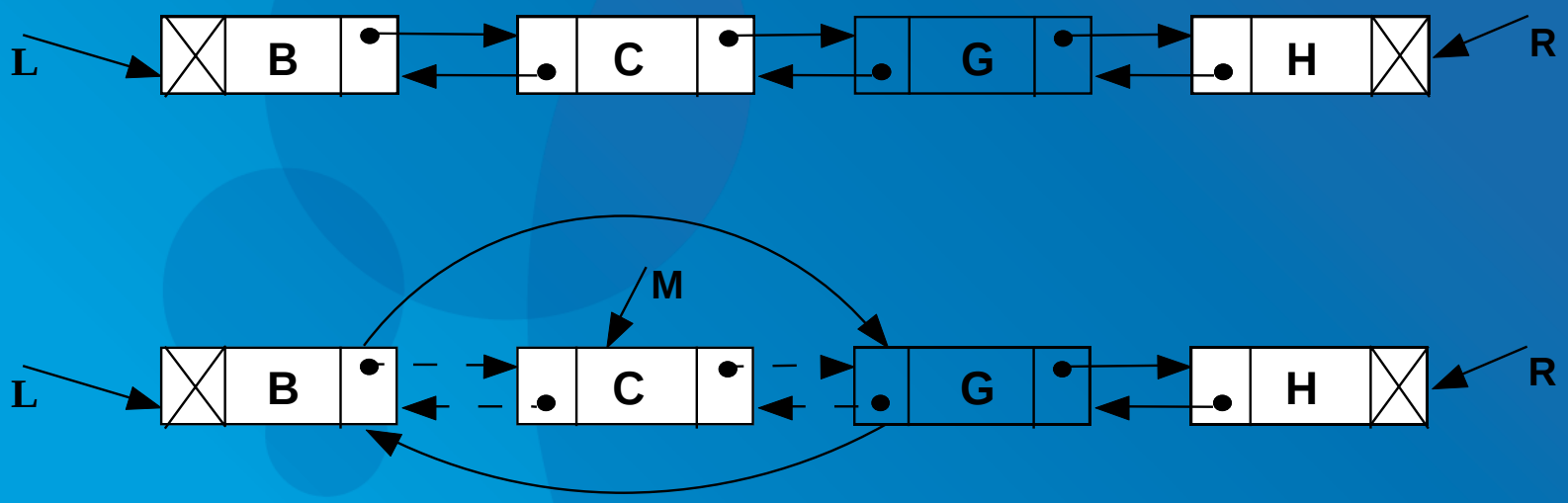
Các thao tác trên danh sách liên kết đôi

Bổ sung vào danh sách liên kết đôi (tiếp)

```
3. {Trường hợp M là nút cực phải}
if M = R then begin
   NEXT(p) := NULL; PREV(p) := M; NEXT(M) := p;
   R:= p;
end;
4. { Bổ sung vào giữa}
PREV(p) := M; NEXT(p) := NEXT(M);
PREV(NEXT(M)) := p;
NEXT(M) := p;
5. return.
```

Các thao tác trên danh sách liên kết đôi

Loại bỏ một phần tử



Các thao tác trên danh sách liên kết đôi

Giải thuật loại bỏ một phần tử khỏi danh sách liên kết đôi

```
Procedure DELETE-DOUBLE (L, R, M)
{Loại bỏ phần tử trỏ bởi M }
1. {Danh sách rỗng}
   if L= R= NULL  then return;
2.  {Loại bỏ}
   if L= R and L = M then L:=R:= NULL;
   else if M = L then begin L:= NEXT(L); PREV(L) := NULL; NEXT(M):=NULL; end;
   else if M = R then begin R:= PREV(R); NEXT(R) := NULL; PREV(M):=NULL; end;
        else begin NEXT(PREV(M)) :=NEXT(M); PREV(NEXT(M)) := PREV(M);
             PREV(M):=NULL; NEXT(M):=NULL;
        end;
   call Dispose(M);
3. return.
```

Biểu diễn đa thức sử dụng danh sách

- Bài toán cộng hai đa thức
  - Dạng tổng quát của một đa thức

$$P(x) = a_n x^n + a_{n-1} x^{n-1} + ... + a_1 x + a_0$$

$$A(x) = 2x^8 - 5x^7 + 3x^2 + 4x - 7$$

$$B(x) = 6x^8 + 5x^7 - 2x^6 + x^4 - 8x^2$$

  - Viết giải thuật tìm tổng 2 đa thức trên

Cách tiếp cận sử dụng danh sách kế tiếp

- Biểu diễn đa thức sử dụng danh sách lưu trữ kế tiếp
  - Mỗi số hạng của đa thức ứng với một phần tử của vector lưu trữ
  - Một vector có kích thước n có các phần tử đánh số từ 1 đến n thì lưu trữ được một đa thức có số mũ tối đa là n-1
  - Phần hệ số a_i của một số hạng được lưu trong chính phần tử của vector lưu trữ
  - Phần số mũ i của một số hạng thì ẩn trong thứ tự của phần tử lưu trữ
  - Phần tử thứ i trong vector lưu trữ lưu thông tin về số hạng $a_{i-1}x^{i-1}$
    - » Phần tử thứ 1 lưu trữ thông tin a_0
    - » Phần tử thứ 2 lưu trữ thông tin về a_1
    - » …

Cách tiếp cận sử dụng lưu trữ kế tiếp

Ví dụ:

$$A(x) = 2x^8 - 5x^7 + 3x^2 + 4x - 7$$

$$B(x) = 6x^8 + 5x^7 - 2x^6 + x^4 - 8x^2$$

A[1]	A[2]	A[3]	A[4]	A[5]	A[6]	A[7]	A[8]	A[9]
-7	4	3	0	0	0	0	-5	2

B[1]	B[2]	B[3]	B[4]	B[5]	B[6]	B[7]	B[8]	B[9]
0	0	-8	0	1	0	-2	5	6

Cách tiếp cận sử dụng lưu trữ kế tiếp

Giải thuật cộng hai đa thức lưu trữ trên vector

```
Procedure ADD-POLY1(A,m, B, n, C)
Begin
{A, B là hai vector lưu trữ hai đa thức đã cho;
m,n lần lượt là kích thước của A,B, giả sử m <= n ;
C là vector lưu trữ kết quả}
for i:= 1 to n do begin
      if i<= m then
            C[i] := A[i] + B[i] ;
      else
            C[i] := B[i] ;
   end.
End
```

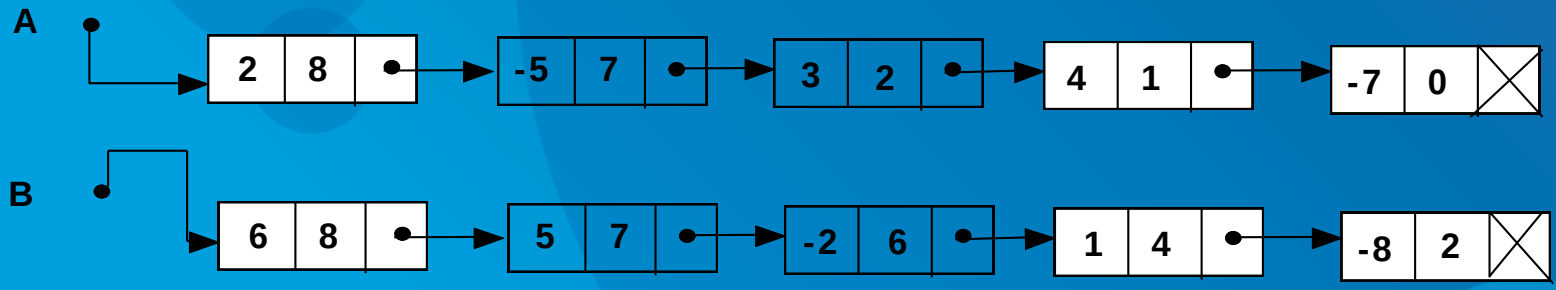
Cách tiếp cận sử dụng danh sách liên kết

- Biểu diễn đa thức sử dụng danh sách liên kết
  - Một đa thức được biểu diễn dưới dạng danh sách liên kết đơn
  - Quy cách của 1 nút

COEF	EXP	LINK

Ví dụ: $A(x) = 2x^8 - 5x^7 + 3x^2 + 4x - 7$

$B(x) = 6x^8 + 5x^7 - 2x^6 + x^4 - 8x^2$



Cách tiếp cận sử dụng danh sách liên kết

```
Procedure ADD-POLY2(A, B,  C)
Begin
1. p:= A; q:=B;
2. call New(C) ; d:= C; {d trỏ vào nút cuối cùng của C}
3. while p <> NULL and q <> NULL do
   case
        EXP(p) = EXP(q):        x := COEF(p) + COEF(q) ;
                        if x<>0 then call ATTACH(x, EXP(p), d) ;
                        p:= LINK(p) ; q:= LINK(q);
        EXP(p) > EXP(q):        call  ATTACH(COEF(p), EXP(p),d);
                        p:= LINK(p);
        EXP(p) < EXP(q):        call  ATTACH(COEF(q), EXP(q),d);
                        q:= LINK(q);
   end case;  {Còn tiếp}
```

Cách tiếp cận sử dụng danh sách liên kết

```
4. {Trường hợp A kết thúc trước, A ngắn hơn}
   while q <> NULL do begin
        call ATTACH(COEF(q), EXP(q),d); q:= LINK(q);
   end ;
5. {Trường hợp B kết thúc trước}
   while p <> NULL do begin
        call ATTACH(COEF(p), EXP(p), d) ; p := LINK(p);
   end ;
6. {Kết thúc danh sách tổng} LINK(d) := NULL;
7. {Cho con trỏ C trỏ tới danh sách tổng}
   t:= C; C:= LINK(t); call dispose(t);
8. return.
```

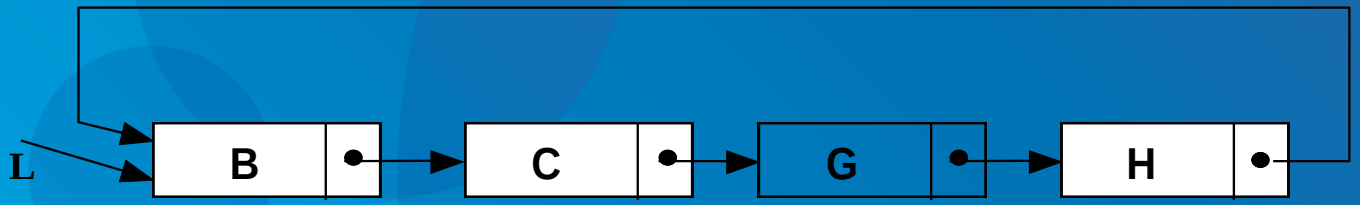
Cách tiếp cận sử dụng danh sách liên kết

Giải thuật gắn một nút mới vào đuôi một danh sách

```
Procedure ATTACH(c, e, d)
Begin
{c, e lần lượt là hệ số và số mũ của nút mới;
d là con trỏ trỏ tới nút đuôi của danh sách }
1. {Khởi tạo nút mới}
   call New(p); COEF(p) := c; EXP(p) := e;
2. {Gắn vào danh sách tổng, biến nó thành nút đuôi mới }
   LINK(d) := p;
   d:=p;
End
```

Các dạng danh sách liên kết khác

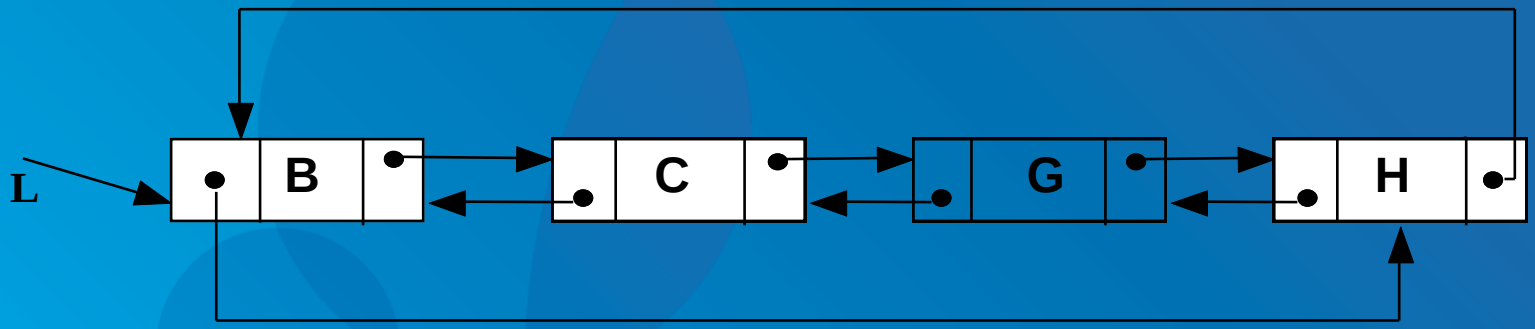
- Danh sách nối vòng (Circularly Linked-List)
  - Trường LINK của nút cuối cùng của danh sách chứa địa chỉ của nút đầu tiên trong danh sách




```
struct clnode{
        int  info;
        struct clnode *next;
} ;
```

Các dạng danh sách liên kết khác

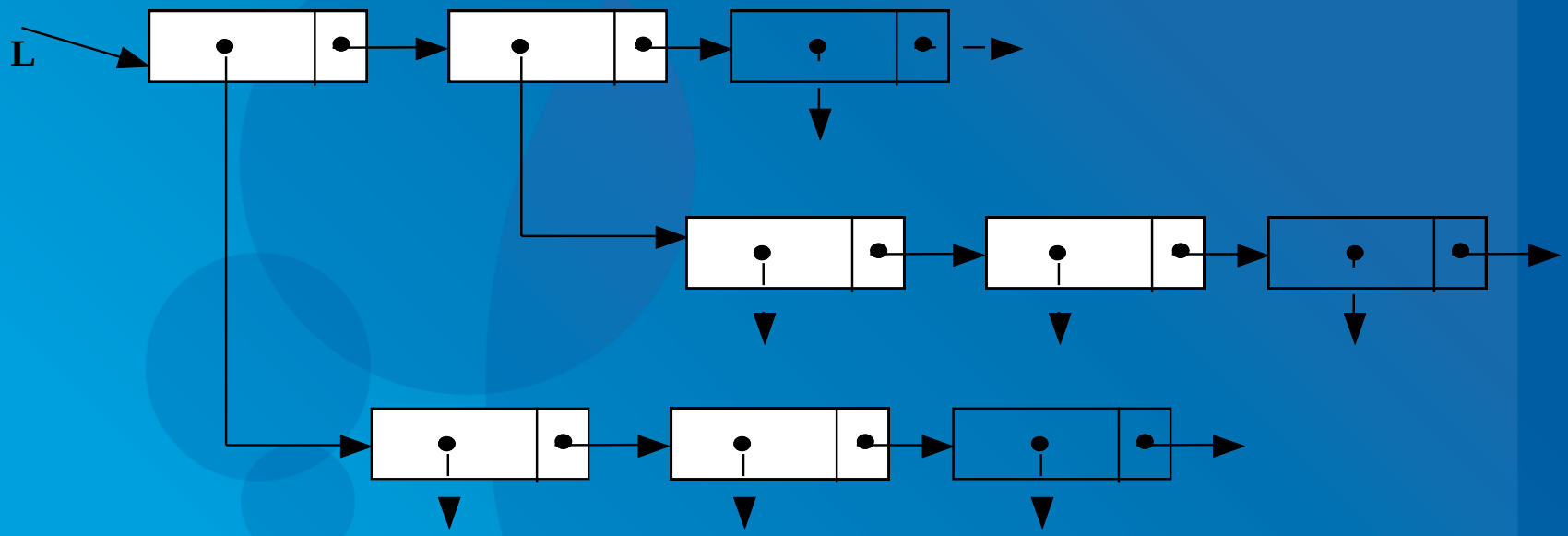
- Danh sách nối vòng kép




```
struct cdlnode{
        int  info;
        struct cdlnode *next;
        struct cdlnode *prev;
} ;
```

Các dạng danh sách liên kết khác

- Danh sách của danh sách




```
struct node{
        struct node*  info;
        struct node *next;
} ;
```