CHƯƠNG 3
DANH SÁCH LIÊN KẾT

GV. Ngô Công Thắng
Bộ môn Công nghệ phần mềm
Khoa Công nghệ thông tin
Website: dse.vnua.edu.vn/ncthang
Email: ncthang@vnua.edu.vn

CHƯƠNG 3
DANH SÁCH LIÊN KẾT

1. Giới thiệu về danh sách liên kết

- Sử dụng con trỏ để tổ chức lưu trữ danh sách tuyến tính sẽ tạo ra cấu trúc dữ liệu danh sách liên kết.
- Mỗi phần tử của danh sách được lưu trữ trong một phần tử nhớ mà ta gọi là nút (node). Mỗi nút bao gồm một số ô nhớ liền kề nhau và có thể nằm ở bất kỳ chỗ nào trong bộ nhớ. Trong mỗi nút ngoài thông tin ứng với phần tử, còn có địa chỉ của nút liền kề nó.

1. Giới thiệu về danh sách liên kết (*tiếp*)

- Danh sách liên kết được chia thành danh sách liên kết đơn, danh sách liên kết vòng và danh sách liên kết kép.
- Danh sách liên kết đơn có thể dùng làm cấu trúc lưu trữ cho cấu trúc ngăn xếp và hàng đợi.

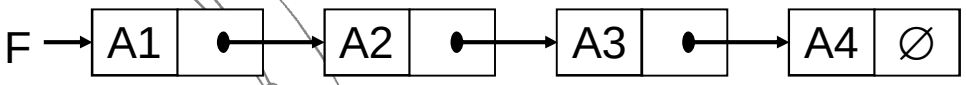
2. Danh sách liên kết đơn

2.1. Quy tắc tổ chức danh sách liên kết đơn

- Mỗi nút trong danh sách có hai trường, trường INFOR chứa thông tin của phần tử và trường LINK chứa địa chỉ của nút đứng sau (đây chính là địa chỉ liên kết).

2.1. Quy tắc tổ chức danh sách liên kết đơn (*tiếp*)

- Nút cuối cùng trong danh sách không có nút đứng sau nên trường địa chỉ là rỗng, không chứa địa chỉ, ta ký hiệu là ∅.
- Để truy nhập vào tất cả nút trong danh sách thì phải có địa chỉ nút đầu tiên, do đó cần phải có con trỏ F trỏ tới nút đầu tiên.
- Nếu danh sách rỗng thì qui ước F = ∅



2.1. Quy tắc tổ chức danh sách liên kết đơn (*tiếp*)

- Để tổ chức lưu trữ một danh sách liên kết thì phải có:
  - Phải có phương tiện chia bộ nhớ ra thành các nút và ở mỗi nút có thể truy nhập vào từng trường.
  - Phải có cơ chế để xác định một nút đang được sử dụng hoặc chưa được sử dụng (nút trống).
  - Phải có cơ chế cung cấp các nút trống khi có yêu cầu sử dụng và thu hồi lại các nút khi không cần dùng nữa.
- Ta ký hiệu:
  - P ⇐ AVAIL là phép lấy ra một nút trống có địa chỉ là P (cấp phát một nút)
  - P ⇒ AVAIL là phép thu hồi một nút có địa chỉ là P

2.2. Một số phép toán trên danh sách liên kết đơn

- Ký hiệu: Một nút có địa chỉ là p (được trỏ bởi p) thì INFOR(p) và LINK(p) tương ứng chỉ trường INFOR và LINK của nút đó.

a) Bổ sung một nút mới vào danh sách

Cho danh sách liên kết đơn F, M là con trỏ trỏ tới một nút trong danh sách. Viết thủ tục bổ sung phần tử dữ liệu X **vào sau** nút M.

2.2. Một số phép toán trên danh sách liên kết đơn (*tiếp*)

a) Bổ sung một nút mới vào danh sách:

- Vào: F, M, X
- Ra: Không có

{Thủ tục này bổ sung phần tử X vào sau nút trỏ bởi M trong danh sách liên kết đơn F}

```
Procedure INSERT(F,M,X)
  1. {Tạo nút mới}
  new ⇐ AVAIL
  INFOR(new):=X; LINK(new):= ∅;
```


2.2. Một số phép toán trên danh sách liên kết đơn (*tiếp*)

```
2. {Thực hiện bổ sung: Nếu danh sách rỗng thì bổ
   sung nút mới vào thành nút đầu tiên. Nếu danh
   sách không rỗng thì bổ sung nút mới vào sau
   nút M}
   If F=∅ then F:=new
  Else begin
            LINK(new):=LINK(M);
            LINK(M):=new;
       end;
Return
```


2.2. Một số phép toán trên danh sách liên kết đơn (*tiếp*)

b) Loại bỏ một nút khỏi danh sách

- Vào: F, M
- Ra: Không

{Cho danh sách liên kết đơn F, loại bỏ nút trỏ bởi M khỏi danh sách.}

```
 Procedure DELETE(F,M)
  1. { Trường hợp danh sách rỗng}
  If F=∅  then begin
      Write('danh sách rỗng')
      Return
       end
```


2.2. Một số phép toán trên danh sách liên kết đơn (*tiếp*)

b) Loại bỏ một nút khỏi danh sách

```
2. {Nút trỏ bởi M là nút đầu tiên của danh
   sách }
   If M=F then begin
      F:=LINK(F)
      M ⇒ AVAIL
      Return
   end
```

2.2. Một số phép toán trên danh sách liên kết đơn (*tiếp*)

3. {Tìm đến nút đứng trước nút M }
   P:=F;
   While LINK(P) # M do P:=LINK(P)
4. {Loại bỏ nút trỏ bởi M}
   LINK(P):=LINK(M)
5. {Hủy nút M}
   M ⇒ AVAIL

Return

2.2. Một số phép toán trên danh sách liên kết đơn (*tiếp*)

c) Ghép hai danh sách liên kết đơn

Cho 2 danh sách liên kết đơn lần lượt trỏ bởi p và q, ghép 2 danh sách trở thành một danh sách và cho p trỏ tới. Thuật toán có các bước sau:

Procedure Ghep(p,q)

1. {Danh sách trỏ bởi q rỗng}
   If q = ∅ then Return
2. {Trường hợp danh sách trỏ bởi p rỗng}
   If p = ∅ then begin
      p:=q
      return
   end

2.2. Một số phép toán trên danh sách liên kết đơn (*tiếp*)

c) Ghép hai danh sách liên kết đơn

3. {Tìm đến nút cuối danh sách p}
   p1:= p
   While link(p1) # ∅ do p1:=link(p1);
4. {Ghép}
   Link(p1):=q;

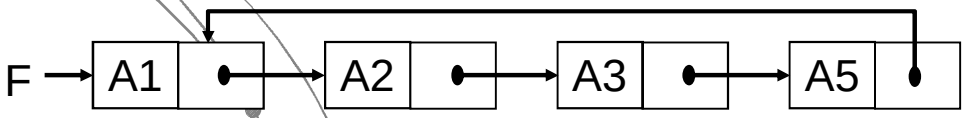
Return

Ưu nhược điểm của danh sách liên kết đơn

- Với danh sách tuyến tính động, trong quá trình xử lý luôn có bổ sung, loại bỏ thì tổ chức danh sách liên kết là hợp lý, tận dụng được các vùng nhớ nằm rải rác trong bộ nhớ.
- Chỉ có phần tử đầu tiên là truy nhập trực tiếp, các phần tử khác phải truy nhập qua phần tử đứng trước nó.
- Tốn bộ nhớ do phải lưu cả 2 trường infor và link ở mỗi nút.

3. Danh sách liên kết vòng

- Danh sách liên kết vòng (Circularly Linked List) là một dạng cải tiến của danh sách liên kết đơn.
- Trong danh sách liên kết vòng, trường địa chỉ của nút cuối cùng không phải là rỗng mà lại chứa địa chỉ của nút đầu tiên của danh sách.

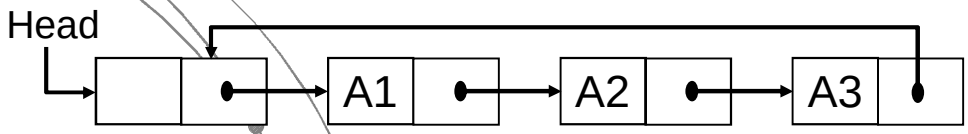


3. Danh sách liên kết vòng (*tiếp*)

- Ưu nhược điểm của danh sách nối vòng:
  - Danh sách nối vòng làm cho việc truy nhập vào các nút trong danh sách linh hoạt hơn. Ta có thể truy nhập vào danh sách bắt đầu từ một nút nào cũng được, không nhất thiết phải từ nút đầu tiên. Nút nào cũng có thể là nút đầu tiên và con trỏ F trỏ vào nút nào cũng được.
  - Nhược điểm của danh sách nối vòng là trong xử lý nếu không cẩn thận sẽ dẫn tới một chu trình không kết thúc.

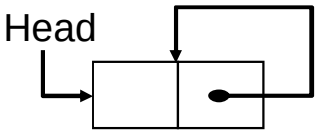
3. Danh sách liên kết vòng (*tiếp*)

- Để khắc phục nhược điểm của danh sách nối vòng ta đưa thêm vào một nút đặc biệt gọi là “nút đầu danh sách” (list head node). Trường Infor của nút này không chứa dữ liệu, con trỏ HEAD trỏ tới nút đầu danh sách này cho phép ta truy nhập vào danh sách.



3. Danh sách liên kết vòng (*tiếp*)

- Việc dùng thêm nút đầu danh sách đã làm cho danh sách luôn có ít nhất 1 nút nên không bao giờ rỗng. Danh sách có 1 nút HEAD có LINK(Head)= Head.



- Các phép toán bổ sung và loại bỏ nút trong danh sách liên kết vòng tương tự danh sách liên kết đơn .

4. Danh sách liên kết kép

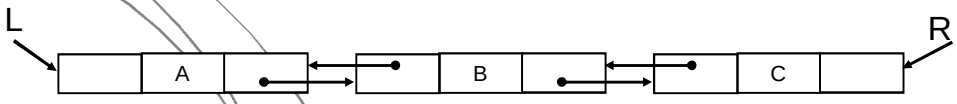
4.1. Giới thiệu

- Danh sách liên kết kép (double linked list) là danh sách mà ở mỗi nút có 2 con trỏ, một trỏ tới nút đứng trước và một trỏ tới nút đứng sau nó.
- Cấu trúc một nút như sau:

LPTR: Con trỏ trỏ tới nút đứng trước
RPTR: Con trỏ trỏ tới nút đứng sau
INFOR: Trường thông tin.

4.1. Giới thiệu (tiếp)

- LPTR của nút cực trái và RPTR của nút cực phải có giá trị là ∅.
- Để truy nhập vào danh sách cả 2 chiều ta phải dùng 2 con trỏ: Con trỏ L trỏ vào nút cực trái, con trỏ R trỏ vào nút cực phải.
- Khi danh sách rỗng thì L = R = ∅.



4.2. Các phép toán trên danh sách liên kết kép

a) Chèn thêm một nút vào danh sách

- Chèn phần tử dữ liệu X **vào trước** nút M trong danh sách liên kết kép L, R.

4.2. Các phép toán trên danh sách liên kết kép

- Vào: L,R,M,X
- Ra: Không có

{Thủ tục này bổ sung phần tử X **vào trước nút M** trong DSLK kép L, R}

Procedure Insert(L,R,M,X)

1. {Tạo nút mới}

new ⇐ AVAIL

IF new= ∅ then begin write('Hết bộ nhớ'); return; end;

INFOR(new) := X

LPTR(new):=RPTR(new):= ∅

a) Chèn thêm một nút vào danh sách

2. {Trường hợp danh sách rỗng}
   If L=R=∅ then begin
      L:=R:=new ;
      Return;
   end
3. {M trỏ tới nút cực trái}
   If M=L then begin
      RPTR(new):= L;
      LPTR(L):= new;
      L:=new;
      Return;
   end

a) Chèn thêm một nút vào danh sách

4. {Bổ sung vào giữa DS trước M}
   RPTR(LPTR(M)):=new
   LPTR(new):=LPTR(M)
   LPTR(M):=new
   RPTR(new):=M
5. {Kết thúc}
   Return

b) Loại bỏ một nút ra khỏi danh sách liên kết kép

- Cho danh sách liên kết kép L, R. M là con trỏ trỏ tới một nút trong danh sách cần loại bỏ.
- Vào: (L,R), M
- Ra: Không có

{Thủ tục này loại bỏ nút trỏ bởi M trong DSLK kép L, R}
   Procedure Delete(L, R, M)
1. {Trường hợp danh sách rỗng }
   If L=R=∅ then begin
      Write(' danh sach rong ')
      Return
   end

b) Loại bỏ một nút ra khỏi danh sách liên kết kép

2. {Loại bỏ}
   Case
      L= R: Begin {Danh sach chỉ còn 1 nút M}
            L:= ∅
            R:= ∅
         end
      M=L: Begin { Nút cực trái bị loại }
            L:=RPTR(L)
            LPTR(L):= ∅
         end

b) Loại bỏ một nút ra khỏi danh sách liên kết kép

```
M=R:  Begin  { Nút cực phải bị loại }
        R:=LPTR(R)
        RPTR(R):= ∅
      end
ELSE
      RPTR(LPTR(M)):=RPTR(M)
      LPTR(RPTR(M)):=LPTR(M)
End Case
M ⇒ AVAIL; {Hủy nút M}
3. {Kết thúc}
Return
```

5. Cài đặt ngăn xếp và hàng đợi bằng danh sách liên kết đơn

5.1. Cài đặt stack bằng danh sách lk đơn

Dùng danh sách liên kết đơn trỏ bởi F để cài đặt Stack thì F là đỉnh Stack. Các phép bổ sung và loại bỏ đều được thực hiện ở đỉnh Stack, tức là đều được thực hiện ở đầu danh sách. Thực hiện các phép toán này tương tự như danh sách liên kết đơn.

5. Cài đặt ngăn xếp và hàng đợi bằng danh sách liên kết đơn

- Vào: T, X
- Ra: Không có

{Thủ tục này bổ sung phần tử X vào ngăn xếp T lưu trữ bằng DSLK đơn}

```
Procedure Push(Var T,X)
   1) {Tạo nút mới}
   new <= AVAIL; Infor(new):=X; Link(new):=∅;
   2) {Bổ sung vào ngăn xếp}
   Link(new):=T: T:=new;
Return
```


5. Cài đặt ngăn xếp và hàng đợi bằng danh sách liên kết đơn

- Vào: T
- Ra: Phần tử dữ liệu loại bỏ

{Hàm này loại bỏ phần tử đỉnh ngăn xếp T lưu trữ bằng DSLK đơn và trả về phần tử này}

```
Function Pop(Var T)
   1) {Trường hợp ngăn xếp rỗng}
   If T= ∅ then begin write('Ngan xep da rong'); return; end;
   2) {Loại bỏ phần tử đỉnh của ngăn xếp}
   Tg:=Infor(T); P:=T; T:=Link(T); P => AVAIL;
   Pop:=Tg;
Return
```

5. Cài đặt ngăn xếp và hàng đợi bằng danh sách liên kết đơn

- Vào: T
- Ra: TRUE nếu ngăn xếp rỗng, FALSE nếu không rỗng

{Hàm kiểm tra ngăn xếp T lưu trữ bằng DSLK đơn, trả về TRUE nếu n.xếp rỗng và FALSE nếu chưa rỗng}

```
Function Empty(T)
    If T = ∅ then Empty:=TRUE;
    Else Empty:=FALSE;
Return
```


5.2. Cài đặt Queue bằng danh sách liên kết đơn

- Cài đặt Queue bằng danh sách liên kết đơn trỏ bởi F thì F là lối trước (F).
- Khi loại bỏ một phần tử khỏi Queue thì loại bỏ ở lối trước, do đó F phải trỏ tới nút tiếp theo.
- Khi bổ sung một phần tử vào Queue thì bổ sung ở lối sau, do đó phải tìm đến nút cuối cùng rồi thêm một nút vào sau nút cuối cùng. Thực hiện các phép toán này tương tự như danh sách liên kết đơn.

5.2. Cài đặt Queue bằng danh sách liên kết đơn

- Vào: (F, R), X
- Ra: Không có

{Thủ tục này bổ sung phần tử X vào lối sau của hàng đợi (F, R) lưu trữ bằng DSLK đơn}

```
Procedure Qinsert(F,R,X)
        1) {Tạo nút mới}
        new <= AVAIL;
        Infor(new) := X; Link(new) := ∅;
        2) {Bổ sung nút mới vào hàng đợi}
        If F=R=∅ Then F:=R:=new
        Else begin Link(R) := new; R := new; end;
Return
```


Bài tập

1. Thế nào là danh sách nối vòng. Nêu ưu nhược điểm của nó.
2. Để khắc phục hạn chế của danh sách nối vòng người ta làm thế nào.
3. Thế nào là danh sách nối kép? Qui ước biểu diễn một nút của danh sách nối kép.
4. Nêu ưu nhược điểm của danh sách nối kép.
5. Cài đặt Stack bằng danh sach nối đơn như thế nào. Cần chú ý gì khi thực hiện các phép bổ sung, loại bỏ phần tử.
6. Cài đặt Queue bằng danh sach nối đơn như thế nào. Cần chú ý gì khi thực hiện các phép bổ sung, loại bỏ phần tử.