Danh sách nối đơn (Singlely Linked List)~

Bởi:

Khoa CNTT ĐHSP KT Hưng Yên

KHÁI NIỆM DANH SÁCH NỐI ĐƠN

Mỗi phần tử của danh sách đơn là một cấu trúc chứa 2 thông tin :

- *Thành phần dữ liệu:* lưu trữ các thông tin về bản thân phần tử .

- *Thành phần mối liờn kết:* lưu trữ địa chỉ của phần tử kế tiếp trong danh sách, hoặc lưu trữ giá trị NULL nếu là phần tử cuối danh s ách.

Ta có định nghĩa tổng quát

typedef struct tagNode

{ Data Info; // Data là kiểu đó định nghĩa trước

struct tagNode* pNext; // con trỏ chỉ đến cấu trúc node

}NODE;

Ví dụ: Định nghĩa danh sách đơn lưu trữ hồ sơ sinh viên:

typedef struct SinhVien

{ char Ten[30];

int MaSV;

}SV;

typedef struct SinhvienNode

{ SV Info;

struct SinhvienNode* pNext;

}SVNode;

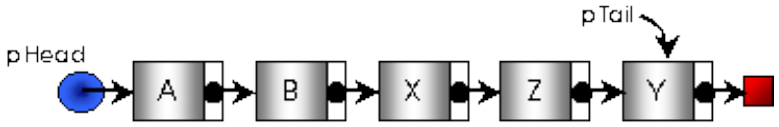
- Một phần tử trong danh sách đơn là một biến động sẽ được yêu cầu cấp phát khi cần. Và danh sách đơn chính là sự liên kết các biến động này với nhau do vậy đạt được sự linh động khi thay đổi số lượng các phần tử
- Nếu biết được địa chỉ của phần tử đầu tiên trong danh sách đơn thỡ cú thể dựa vào thụng tin pNext của nú để truy xuất đến phần tử thứ 2 trong xâu, và lại dựa vào thông tin Next của phần tử thứ 2 để truy xuất đến phần tử thứ 3...Nghĩa là để quản lý một xâu đơn chỉ cần biết địa chỉ phần tử đầu xâu. Thường một con trỏ Head sẽ được dùng để lưu trữ địa chỉ phần tử đầu xâu, ta gọi Head là đầu xâu. Ta có khai báo:

NODE *pHead;

- Tuy về nguyên tắc chỉ cần quản lý xâu thông qua đầu xâu pHead, nhưng thực tế có nhiều trường hợp cần làm việc với phần tử cuối xâu, khi đó mỗi lần muốn xác định phần tử cuối xâu lại phải duyệt từ đầu xâu. Để tiện lợi, cú thể sử dụng thờm một con trỏ pTail giữ địa chỉ phần tử cuối xâu. Khai báo pTail như sau :

NODE *pTail;

Lúc này có xâu đơn:



MỘT SỐ PHÉP TOÁN TRÊN DANH SÁCH NỐI ĐƠN

Giả sử có các định nghĩa:

typedef struct tagNode

{

Data Info;

struct tagNode* pNext;

}NODE;

Danh sách nối đơn (Singlely Linked List)~

typedef struct tagList

{

NODE* pHead;

NODE* pTail;

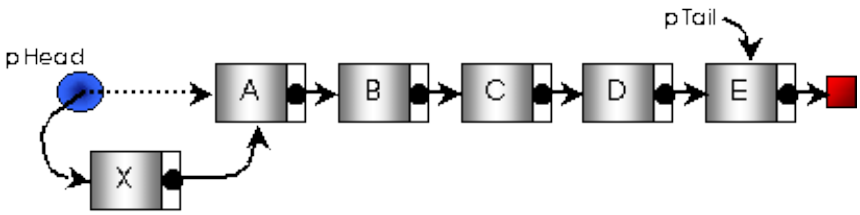
}LIST;

NODE *new_ele // giữ địa chỉ của một phần tử mới được tạo

Data x; // lưu thông tin về một phần tử sẽ được tạo

Và đã xây dựng thủ tục GetNode để tạo ra một phần tử cho danh sách với thông tin chứa trong x:

Chèn một phần tử vào danh sách:



Có 3 loại thao tác chèn new_ele vào xâu:

Cách 1: Chèn vào đầu danh sách

- Thuật toán :

Bắt đầu:

Nếu Danh sách rỗng Thì

B11 : Head = new_elelment;

B12 : Tail = Head;

Ngược lại

B21 : new_ele ->pNext = Head;

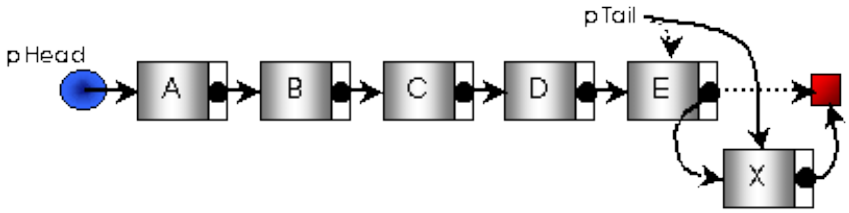
B22 : Head = new_ele ;

Danh sách nối đơn (Singlely Linked List)~

- Cài đặt :

```
void AddFirst(LIST &l, NODE* new_ele)
{
if (l.pHead==NULL) //Xâu rỗng
{
l.pHead = new_ele; l.pTail = l.pHead;
}
else
{
new_ele->pNext = l.pHead;
l.pHead = new_ele;
}
}
NODE* InsertHead(LIST &l, Data x)
{
NODE* new_ele = GetNode(x);

if (new_ele ==NULL) return NULL;
if (l.pHead==NULL)
{
l.pHead = new_ele; l.pTail = l.pHead;
}
```

```
else
{
new_ele->pNext = l.pHead;
l.pHead = new_ele;
}
return new_ele;
}
```



Cách 2: Chèn vào cuối danh sách

- Thuật toán :

Bắt đầu :

Nếu Danh sách rỗng Thì

B11 : Head = new_elelment;

B12 : Tail = Head;

Ngược lại

B21 : Tail ->pNext = new_ele;

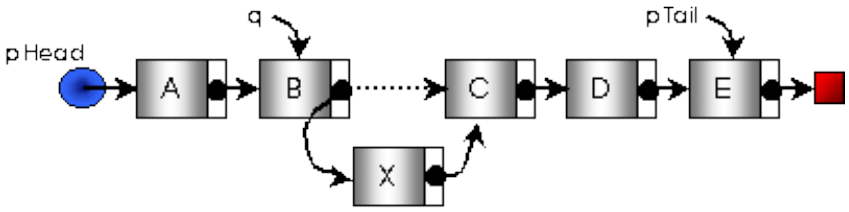
B22 : Tail = new_ele ;

- Cài đặt :

```
void AddTail(LIST &l, NODE *new_ele)
{
```

```
if (l.pHead==NULL)
{
l.pHead = new_ele; l.pTail = l.pHead;
}
else
{
l.pTail->Next = new_ele;
l.pTail = new_ele;
}
}

NODE* InsertTail(LIST &l, Data x)
{
NODE* new_ele = GetNode(x);

if (new_ele ==NULL) return NULL;
if (l.pHead==NULL)
{
l.pHead = new_ele; l.pTail = l.pHead;
}
else
{
```

```
l.pTail->Next = new_ele;

l.pTail = new_ele;

}

return new_ele;

}
```



Cách 3 : Chèn vào danh sách sau một phần tử q

- Thuật toán :

Bắt đầu :

Nếu (q != NULL) thì

B1 : new_ele -> pNext = q->pNext;

B2 : q->pNext = new_ele ;

- Cài đặt :

```
void AddAfter(LIST &l,NODE *q, NODE* new_ele)

{

if ( q!=NULL)

{

new_ele->pNext = q->pNext;

q->pNext = new_ele;

if(q == l.pTail)
```

```
l.pTail = new_ele;
}
else //chèn vào đầu danh sách
AddFirst(l, new_ele);
}
void InsertAfter(LIST &l,NODE *q, Data x)
{
NODE* new_ele = GetNode(x);

if (new_ele ==NULL) return NULL;

if ( q!=NULL)
{
new_ele->pNext = q->pNext;
q->pNext = new_ele;
if(q == l.pTail)
l.pTail = new_ele;
}
else //chèn vào đầu danh sách
AddFirst(l, new_ele);
}
```

Tìm một phần tử trong danh sách đơn

- Thuật toán :

Xâu đơn đòi hỏi truy xuất tuần tự, do đó chỉ có thể áp dụng thuật toán tìm tuyến tính để xác định phần tử trong xâu có khoá k. Sử dụng một con trỏ phụ trợ p để lần lượt trỏ đến các phần tử trong xâu. Thuật toán được thể hiện như sau :

Bước 1:

p = Head; //Cho p trỏ đến phần tử đầu danh sách

Bước 2:

Trong khi (p != NULL) và (p->pNext != k) thực hiện:

B21 : p:=p->Next;// Cho p trỏ tới phần tử kế

Bước 3:

Nếu p != NULL thì p trỏ tới phần tử cần tìm

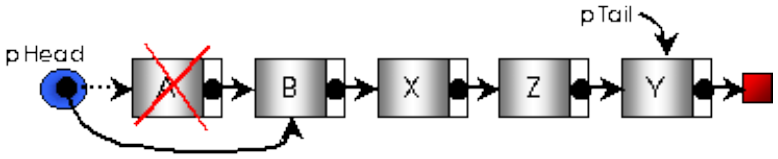
Ngược lại: không có phần tử cần tìm.

- Cài đặt :

```
NODE *Search(LIST l, Data k)
{ NODE *p;
p = l.pHead;
while((p!= NULL)&&(p->Info != x))
p = p->pNext;
return p;
}
```

Hủy một phần tử khỏi danh sách

Có 3 loại thao tác thông dụng hủy một phần tử ra khỏi xâu. Chúng ta sẽ lần lượt khảo sát chúng. Lưu ý là khi cấp phát bộ nhớ, chúng ta đã dùng hàm new. Vì vậy khi giải phóng bộ nhớ ta phải dùng hàm delete.



Hủy phần tử đầu xâu:

- Thuật toán :

Bắt đầu:

Nếu (Head != NULL) thì

B1: p = Head; // p là phần tử cần hủy

B2:

B21 : Head = Head →pNext; // tách p ra khỏi xâu

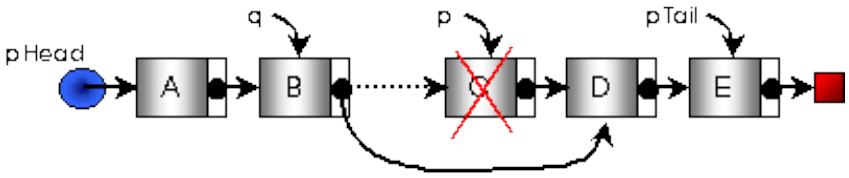
B22 : free(p); // Hủy biến động do p trỏ đến

B3: Nếu Head=NULL thì Tail = NULL; //Xâu rỗng

- Cài đặt :

```
Data RemoveHead(LIST &l)

{ NODE *p;

Data x = NULLDATA;



if ( l.pHead != NULL)

{

p = l.pHead; x = p→Info;
```

```
l.pHead = l.pHead→pNext;
delete p;
if(l.pHead == NULL) l.pTail = NULL;
}
return x;
}
```



Hủy một phần tử đứng sau phần tử q

- Thuật toán :

Bắt đầu:

Nếu (q!= NULL) thì

B1: p = q->Next; // p là phần tử cần hủy

B2: Nếu (p != NULL) thì // q không phải là cuối xâu

B21 : q->Next = p->Next; // tách p ra khỏi xâu

B22 : free(p); // Hủy biến động do p trỏ đến

- Cài đặt :

```
void RemoveAfter (LIST &l, NODE *q)
{ NODE *p;

if ( q != NULL)
{
```

```
p = q ->pNext ;
if ( p != NULL)
{
if(p == l.pTail) l.pTail = q;
q->pNext = p->pNext;
delete p;
}
}
else
RemoveHead(l);
}
```

Hủy 1 phần tử có khoá k

- Thuật toán :

Bước 1:

Tìm phần tử p có khóa k và phần tử q đứng trước nó

Bước 2:

Nếu (p!= NULL) thì // tìm thấy k

Hủy p ra khỏi xâu tương tự hủy phần tử sau q;

Ngược lại

Báo không có k;

- Cài đặt :

```
int RemoveNode(LIST &l, Data k)
```

```
{ NODE *p = l.pHead;
NODE *q = NULL;

while( p != NULL)
{
if(p->Info == k) break;
q = p; p = p->pNext;
}
if(p == NULL) return 0; //Không tìm thấy k
if(q != NULL)
{
if(p == l.pTail)
l.pTail = q;
q->pNext = p->pNext;
delete p;
}
else //p là phần tử đầu xâu
{
l.pHead = p->pNext;
if(l.pHead == NULL)
l.pTail = NULL;
}
```

return 1;

}

Duyệt danh sách

Duyệt danh sách là thao tác thường được thực hiện khi có nhu cầu xử lý các phần tử của danh sách theo cùng một cách thức hoặc khi cần lấy thông tin tổng hợp từ các phần tử của danh sách như:

- Đếm các phần tử của danh sách,
- Tìm tất cả các phần tử thoả điều kiện,
- Huỷ toàn bộ danh sách (và giải phóng bộ nhớ)

Để duyệt danh sách (và xử lý từng phần tử) ta thực hiện các thao tác sau:

- Thuật toán :

Bước 1:

p = Head; //Cho p trỏ đến phần tử đầu danh sách

Bước 2:

Trong khi (Danh sách chưa hết) thực hiện

B21 : Xử lý phần tử p;

B22 : p:=p->pNext; // Cho p trỏ tới phần tử kế

- Cài đặt :

```
void ProcessList (LIST &l)
{ NODE *p;

p = l.pHead;
while (p!= NULL)
```

{

ProcessNode(p); // xử lý cụ thể tùy ứng dụng

p = p->pNext;

}

}

LƯU Ý :

Để huỷ toàn bộ danh sách, ta có một chút thay đổi trong thủ tục duyệt (xử lý) danh sách trên (ở đây, thao tác xử lý bao gồm hành động giải phóng một phần tử, do vậy phải cập nhật các liên kết liên quan) :

- Thuật toán :

Bước 1:

Trong khi (Danh sách chưa hết) thực hiện

B11:

p = Head;

Head:=Head ->pNext; // Cho p trỏ tới phần tử kế

B12:

Hủy p;

Bước 2:

Tail = NULL; //Bảo đảm tính nhất quán khi xâu rỗng

- Cài đặt :

void ReamoveList(LIST &l)

{ NODE *p;

while (l.pHead!= NULL)

```
{
p = l.pHead;
l.pHead = p->pNext;
delete p;
}
l.pTail = NULL;
}
```