CẤU TRÚC DỮ LIỆU VÀ GIẢI THUẬT

Data Structures & Algorithms

DANH SÁCH LIÊN KẾT

LIST

Nội dung

1. Kiểu danh sách
2. Danh sách liên kết đơn.
3. Stack
4. Queue

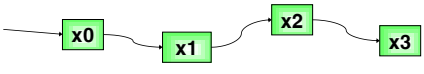
KIỂU DANH SÁCH

- **Danh sách** = { **các phần tử có cùng kiểu**}
- Danh sách là một ***kiểu dữ liệu tuyến tính*** :
  - Mỗi phần tử ***có nhiều nhất 1 phần tử đứng trước***
  - Mỗi phần tử **có *nhiều nhất 1 phần tử đứng sau***
- Là kiểu dữ liệu quen thuộc trong thực tế :
  - Danh sách học sinh
  - Danh mục sách trong thư viện
  - Danh bạ điện thoại
  - Danh sách các nhân viên trong công ty
  - ...

Các hình thức tổ chức danh sách

- CTDL cho **mỗi phần tử** ?
- Thể hiện **liên kết của các phần tử** ?
- *Hai hình thức cơ bản* :
  - Liên kết **ngầm** : Mảng (array)

  - Liên kết **tường minh** :Danh sách liên kết (list)



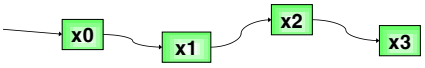
Mảng – Danh sách liên kết ngầm

- Mỗi **liên hệ giữa các phần tử được thể hiện ngầm**:
  - xi : phần tử thứ i trong danh sách
  - xi , xi_{+1} là kế cận trong danh sách
- Phải **lưu trữ liên tiếp các phần tử trong bộ nhớ**
  - công thức xác định địa chỉ phần tử thứ i:
    address(i) = address(1) + (i-1)*sizeof(T)
- **Ưu điểm** : **Truy xuất trực tiếp, nhanh chóng**
- **Nhược điểm**:

  - **Sử dụng bộ nhớ kém hiệu quả**
  - Kích thước **cố định**
  - Các **thao tác thêm vào , loại bỏ không hiệu quả**

Danh sách liên kết tường minh

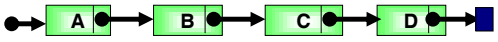
- CTDL cho một phần tử:
  - **Thông tin bản thân**
  - **Địa chỉ của phần tử kế** trong danh sách



- **Mỗi phần tử là một biến động**
- **Ưu điểm**
  - + **Sử dụng hiệu quả bộ nhớ**
  - + **Linh động về số lượng phần tử**

Các loại danh sách liên kết

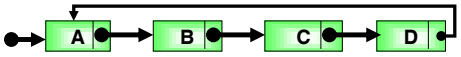
- **Danh sách liên kết đơn**: Mỗi phần tử liên kết với phần tử đứng sau nó trong danh sách



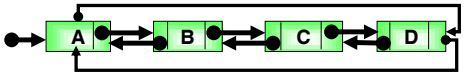
➢ **Danh sách liên kết kép**: Mỗi phần tử liên kết với phần tử đứng trước và sau nó trong danh sách

➢ **Danh sách liên Vòng**: Phần tử cuối danh sách liên với phần tử đầu danh sách

- **Danh sách liên Vòng:** Phần tử cuối danh sách liên với phần tử đầu danh sách
  - Danh sách liên kết đơn vòng

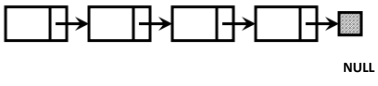


- Danh sách liên kết đôi vòng



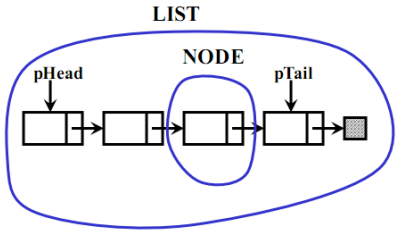
Danh sách liên kết đơn - LIST

Hình ảnh:

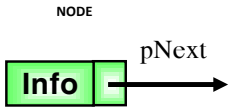


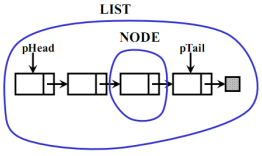
Danh sách liên kết đơn - LIST

Hình ảnh:



Danh sách liên kết đơn - LIST

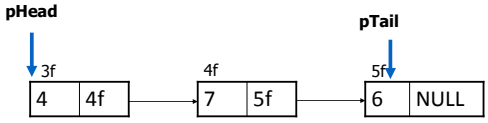





```
11.struct node
12.{
13.    KDL info;
14.    struct node*pNext;
15.};
16.typedef struct node NODE;
```

```
17.struct list
18.{
19.    NODE*pHead;
20.    NODE*pTail;
21.};
22.typedef struct list LIST;
```


Danh sách liên kết đơn - LIST



Trong ví dụ trên thành phần dữ liệu là 1 số nguyên

Danh sách liên kết đơn - LIST

➢Cấu trúc dữ liệu của 1 nút trong List đơn

```
struct  node
{
    KDL   Info;   // Lưu thông tin dữ liệu có KDL
    struct  node *pNext; //Lưu địa chỉ của Node đứng sau
};
```

➢Cấu trúc dữ liệu của DSLK đơn

```
struct  list
{
    NODE *pHead;//Lưu địa chỉ Node đầu tiên trong List
    NODE *pTail; //Lưu địa chỉ của Node cuối cùng trong
List
};
```


Danh sách liên kết đơn - LIST

Ví dụ 1: Hãy khai báo CTDL cho DSLK đơn các **số nguyên**

```
10.struct node
11.{
12.    int info;
13.    struct node*pNext;
14.};
15.typedef struct node NODE;
```

```
16.struct list
17.{
18.    NODE*pHead;
19.    NODE*pTail;
20.};
21.typedef struct list LIST;
```


Danh sách liên kết đơn - LIST

Ví dụ 2: Hãy khai báo CTDL cho DSLK đơn các **số thực**

```
10.struct node
11.{
12.    float info;
13.    struct node*pNext;
14.};
15.typedef struct node NODE;
```

```
16.struct list
17.{
18.    NODE*pHead;
19.    NODE*pTail;
20.};
21.typedef struct list LIST;
```


Danh sách liên kết đơn - LIST

Ví dụ 3: Hãy khai báo CTDL cho DSLK đơn các **phân số**

```
10.struct phanso
11.{
12.    int tu;
13.    int mau;
14.};
15.typedef struct phanso PHANSO;
```

```
16.struct node
17.{
18.    PHANSO info;
19.    struct node*pNext;
20.};
21.typedef struct node NODE;
```

```
22.struct list
23.{
24.    NODE*pHead;
25.    NODE*pTail;
26.};
27.typedef struct list LIST;
```


Danh sách liên kết đơn - LIST

Ví dụ 4: Hãy khai báo CTDL cho DSLK đơn **tọa độ** các điểm trong mặt **phẳng oxy**

```
10.struct diem
11.{
12.    float x;
13.    float y;
14.};
15.typedef struct diem DIEM;
```

```
16.struct node
17.{
18.    DIEM info;
19.    struct node*pNext;
20.};
21.typedef struct node NODE;
```

```
22.struct list
23.{
24.    NODE*pHead;
25.    NODE*pTail;
26.};
27.typedef struct list LIST;
```


Các thao tác trên LIST

➢***Khởi tạo DSLK đơn*** - Tạo 1 DSLK đơn rỗng

➢**Tạo** 1 node có trường bằng x

➢Thêm một **node có khóa x** vào **danh sách**

➢ **Duyệt danh sách**

➢**Hủy một phần tử** trong danh sách

➢**Sắp xếp** danh sách liên kết đơn

Khởi tạo DSLK đơn

- Khái niệm: Khởi tạo danh sách liên kết đơn là tạo ra danh sách rỗng không chứa node nào hết.
  - Định nghĩa hàm

```
1. void Init(LIST &ℓ)
2. {
3.     ℓ.pHead = NULL;
4.     ℓ.pTail = NULL;
5. }
```

19

Kiểm tra DSLK đơn rỗng

- Khái niệm: Kiểm tra danh sách liên kết đơn rỗng là hàm trả về giá trị 1 khi danh sách rỗng. Trong tình huống danh sách không rỗng thì hàm sẽ trả về giá trị 0.
  - Định nghĩa hàm

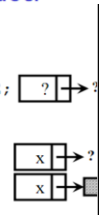
```
1. int IsEmpty(LIST ℓ)
2. {
3.     if (ℓ.pHead==NULL)
4.         return 1;
5.     return 0;
6. }
```

20

Tạo NODE cho DSLK đơn

- **Khái niệm: Tạo node cho danh sách liên kết đơn là xin cấp phát bộ nhớ có kích thước bằng với kích thước của kiểu dữ liệu NODE để chứa thông tin đã được biết trước.**
  - Định nghĩa hàm trừu tượng

```
1. NODE* GetNode(KDL x)
2. {
3.     NODE *p=new NODE;
4.     if (p==NULL)
5.         return NULL;
6.     p->info = x;
7.     p->pNext = NULL;
8.     return p;
9. }
```



21

Tạo NODE cho DSLK đơn

Ví dụ 1: Định nghĩa hàm tạo một node cho DSLK đơn các số nguyên để chứa thông tin đã được biết trước

```
1. NODE* GetNode(int x)
2. {
3.     NODE *p = new NODE;
4.     if (p==NULL)
5.         return NULL;
6.     p->info = x;
7.     p->pNext = NULL;
8.     return p;
```

22

Tạo NODE cho DSLK đơn

Ví dụ 2: Định nghĩa hàm tạo một node cho DSLK đơn các số thực để chứa thông tin đã được biết trước

```
1. NODE* GetNode(float x)
2. {
3.     NODE *p = new NODE;
4.     if (p==NULL)
5.         return NULL;
6.     p->info = x;
7.     p->pNext = NULL;
8.     return p;
9. }
```

23

Tạo NODE cho DSLK đơn

Ví dụ 3: Định nghĩa hàm tạo một node cho DSLK đơn các phân số chứa thông tin đã được biết trước

```
1. NODE* GetNode(PHANSO x)
2. {
3.     NODE *p = new NODE;
4.     if (p==NULL)
5.         return NULL;
6.     p->info = x;
7.     p->pNext = NULL;
8.     return p;
9. }
```

24

Tạo NODE cho DSLK đơn

Ví dụ 4: Định nghĩa hàm tạo một node cho DSLK đơn các điểm trong hệ tọa độ oxy chứa thông tin đã được biết trước

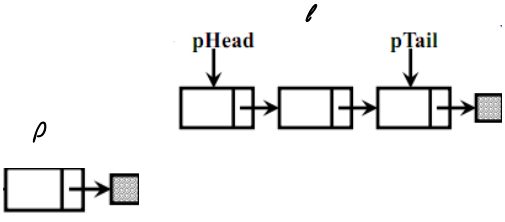
```
1. NODE* GetNode(DIEM P)
2. {
3.     NODE *p = new NODE;
4.     if (p==NULL)
5.         return NULL;
6.     p->info = P;
7.     p->pNext = NULL;
8.     return p;
9. }
```

25

Thêm một phần tử vào List đơn

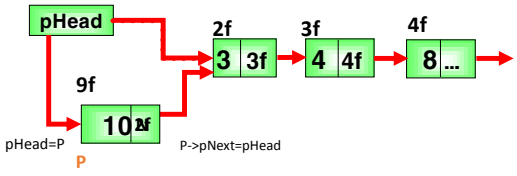
- **Nguyên tắc thêm:** Khi thêm 1 phần tử vào List thì **có làm cho pHead, pTail thay đổi**?
- **Các vị trí cần thêm 1 phần tử vào List:**
  - Thêm vào **đầu List đơn**
  - Thêm vào **cuối List đơn**
  - Thêm vào **sau 1 phần tử q** trong list

Thêm một NODE vào đầu List đơn



27

Thêm một NODE vào đầu List đơn



28

Thêm một NODE vào đầu List đơn

```
void AddHead(LIST &l, Node* p)
{
    if (l.pHead==NULL)
    {
        l.pHead = p;
        l.pTail = l.pHead;
    }
    else
    {
        p->pNext = l.pHead;
        l.pHead = p;
    }
}
```

Nhập List đơn từ bàn phím

Nhập list đơn từ bàn phím là lần lượt nhập các thông tin của từng node trong danh sách.

- Yêu cầu người dùng nhập vào **số node của list**.
- **Nhập vào từng node** trong n của list
  - **Tạo ra** một node
  - **Nhập giá trị info** vào node vừa tạo (**GetNode()**).
  - Thêm node vào list bằng cách thêm vào đầu (**addHead()**).

30

Nhập List đơn từ bàn phím

Nhập list đơn từ bàn phím là lần lượt nhập các thông tin của từng node trong danh sách.

```
void input(List &l)
{
    int n;
    cout<<"Nhap vao so phan tu cua list";
    cin>>n;
    Init(l);
    for(int i=1;i<=n;i++)
    {
        KDL x;
        Nhap(x);
        NODE* p = GetNode(x);
        if(p!=NULL)
            AddHead(l,p)

    }
}
```


Nhập List đơn từ bàn phím

VD 1: Nhập List các **số nguyên**

```
struct node
{
    int info;
    struct node*pNext;
};
typedef struct node NODE;
struct list
{
    NODE*pHead;
    NODE*pTail;
};
typedef struct list LIST;
```


Nhập List đơn từ bàn phím

VD 1: Nhập List các **số nguyên**

```
void Init(LIST&ℓ)
{
    ℓ.pHead = NULL;
    ℓ.pTail = NULL;
}
NODE* GetNode(int x)
{
    NODE *p = new NODE;
    if (p==NULL)
        return NULL;
    p->info = x;
    p->pNext = NULL;
    return p;
}
```


Nhập List đơn từ bàn phím

VD 1: Nhập List các **số nguyên**

```
void AddHead(LIST&ℓ, NODE*p)
{
    if(ℓ.pHead==NULL)
        ℓ.pHead = ℓ.pTail = p;
    else
    {
        p->pNext = ℓ.pHead;
        ℓ.pHead = p;
    }
}
```


Nhập List đơn từ bàn phím

VD 1: Nhập List các số nguyên

```
void input(List &l)
{
    int n;
    cout<<"Nhap vao so phan tu cua list";
    cin>>n;
    Init(l);
    for(int i=1;i<=n;i++)
    {
        int x;
        cin>>x;
        NODE* p = GetNode(x);
        if(p!=NULL)
            AddHead(l,p)

    }
}
```


Nhập List đơn từ bàn phím

VD 2: Nhập List các phân số

```
struct phanso
{
    int tu;
    int mau;
};
typedef struct phanso PHANSO;
struct node
{
    PHANSO info;
    struct node*pNext;
};
typedef struct node NODE;
struct list
{
    NODE*pHead;
    NODE*pTail;
};
typedef struct list LIST;
```

Nhập List đơn từ bàn phím

VD 2: Nhập List các phân số

```
1. void Init(LIST&ℓ)
2. {
3.     ℓ.pHead = NULL;
4.     ℓ.pTail = NULL;
5. }
6. NODE* GetNode(PHANSO x)
7. {
8.     NODE *p = new NODE;
9.     if (p==NULL)
10.        return NULL;
11.    p->info = x;
12.    p->pNext = NULL;
13.    return p;
14.}
```

37

Nhập List đơn từ bàn phím

VD 2: Nhập List các phân số

```
void nhap(PHANSO &x)
{
    cout<<"Nhap tu";
    cin>>x.tu;
    cout<<"\n Nhap mau";
    cin>>x.mau;
}
```

38

Nhập List đơn từ bàn phím

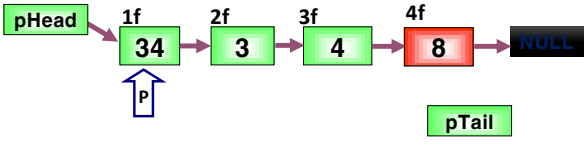
VD 2: Nhập List các phân số

```
void input(List &l)
{
    int n;
    cout<<"Nhap vao so phan tu cua list";
    cin>>n;
    Init(l);
    for(int i=1;i<=n;i++)
    {
        PHANSO x;
        Nhap(x);
        NODE* p = GetNode(x);
        if(p!=NULL)
            AddHead(l,p)
    }
}
```

39

Duyệt tuần tự list đơn

- Khái niệm: duyệt danh sách liên kết đơn là thăm qua tất cả các node mỗi node một lần.



Tới cuối DSLK thì dừng

40

Duyệt tuần tự list đơn

- Định nghĩa hàm trừu tượng

```
11. KDL <Tên Hàm>(LIST ℓ)
12.{
13.    ...
14.    NODE*p = ℓ.pHead;
15.    while(p!=NULL)
16.    {
17.        ...
18.        p = p->pNext;
19.    }
20.    ...
21.}
```

41

Các thao tác duyệt list đơn

- **In ra** danh sách liên kết đơn – In ra giá trị Info của DSLK đơn.
- **Tìm kiếm** node có **trường Info** thỏa mãn điều kiện.

42

Các thao tác duyệt list đơn

Ví dụ 1: **In ra** danh sách liên kết các **số nguyên**

```
10.struct node
11.{
12.    int info;
13.    struct node*pNext;
14.};
15.typedef struct node NODE;
16.struct list
17.{
18.    NODE*pHead;
19.    NODE*pTail;
20.};
21.typedef struct list LIST;
```


Các thao tác duyệt list đơn

Ví dụ 1: **In ra** danh sách liên kết các **số nguyên**

```
40  void XUAT(List l)
41  {
42      NODE* p= l.pHead;
43      while(p!=NULL)
44      {
45          cout<<p->info<<endl;
46          p=p->pNext;
47      }
48  }
```


Các thao tác duyệt list đơn

Ví dụ 2: Định nghĩa hàm tính **tổng các số lẻ** trong dslk đơn các số nguyên

```
- Định nghĩa hàm
10.int TongLe(LIST l)
11.{
12.    int s = 0;
13.    NODE*p = l.pHead;
14.    while(p!=NULL)
15.    {
16.        if(p->info%2!=0)
17.            s = s + p->info;
18.        p = p->pNext;
19.    }
20.    return s;
21.}
```


Chương trình đầu tiên với List đơn

Bài toán: Viết chương trình thực hiện các yêu cầu sau:

- **Nhập dslk đơn các số nguyên.**
- **Tính tổng các giá trị trong dslk đơn.**
- **Xuất dslk đơn.**

Chương trình đầu tiên với List đơn

- Tọa cấu trúc node và list tương ứng (**struct node, struct list**)
- Tạo ra một list rỗng (**Init()**).
- Yêu cầu người dùng nhập vào số node của list. Nhập vào từng node trong n của list (**inPut()**)
  - Tạo ra một node
  - Nhập giá trị info vào node vừa tạo (**GetNode()**).
  - Thêm node vào list bằng cách thêm vào đầu (**addHead()**).
- In ra dslk đơn vừa tạo (**xuat()**).