BÀI TOÁN CHI TIẾT MÁY

1. TS. Nguyễn Tấn Trần Minh Khang
2. ThS. Võ Duy Nguyên
3. ThS. Nguyễn Hoàng Ngân
4. Nguyễn Đức Anh Phúc – Source code.
5. Nguyễn Trần Phúc An – Source code

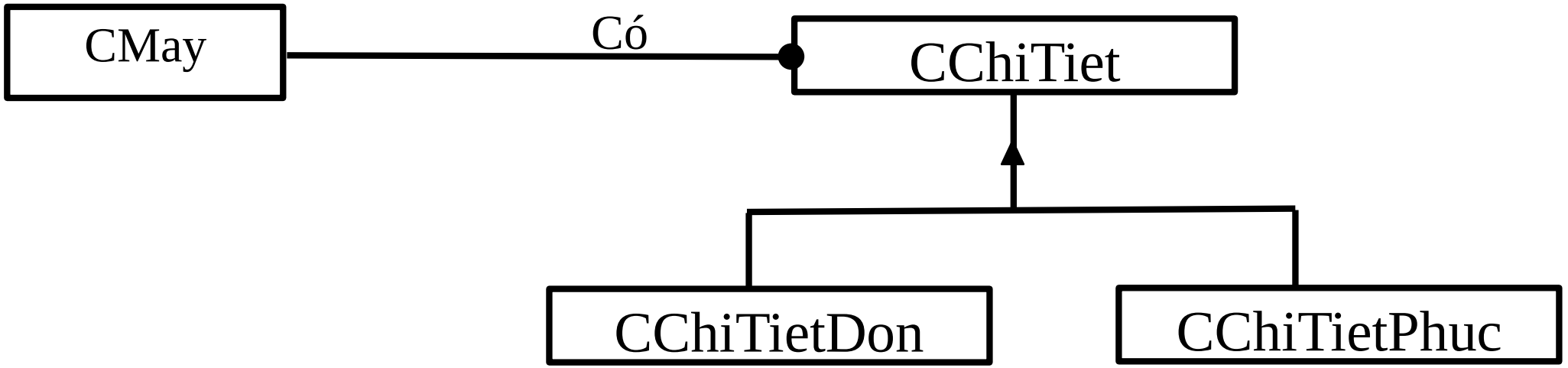
Bài toán

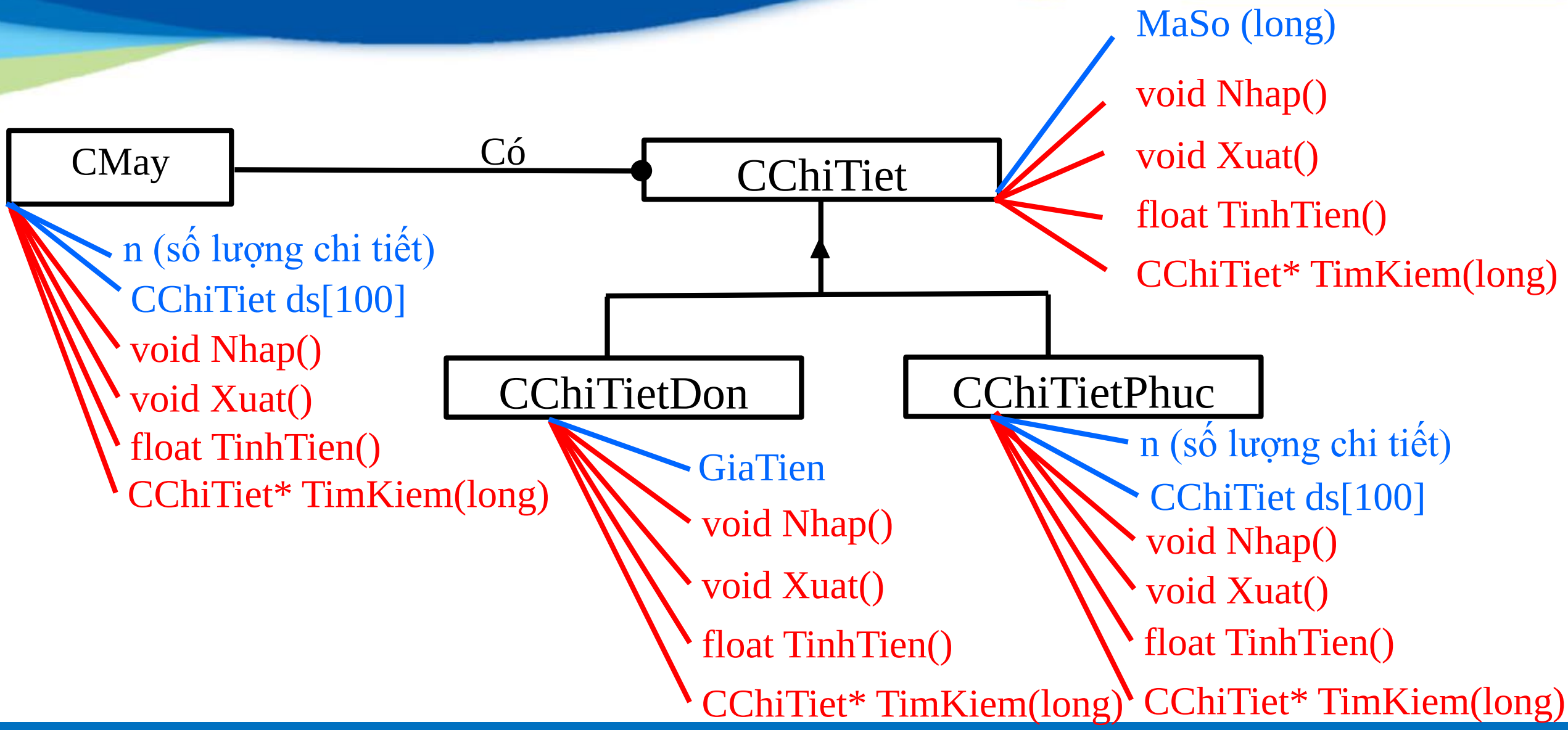
- Bài toán: Một cái máy có nhiều chi tiết. Mỗi chi tiết máy có thể là chi tiết đơn hoặc là chi tiết phức.
  - + Chi tiết đơn: là chi tiết không chứa bên trong nó chi tiết khác. Thông tin của chi tiết đơn bao gồm: mã số chi tiết, giá tiền.
  - + Chi tiết phức: là chi tiết chứa bên trong nó nhiều chi tiết thành phần, mỗi một chi tiết thành phần này có thể là chi tiết đơn hoặc là chi tiết phức. Thông tin của chi tiết phức bao gồm: mã số chi tiết, số lượng chi tiết thành phần, danh sách các chi tiết thành phần. Giá tiền của chi tiết phức bằng tổng giá tiền của các chi tiết thành phần.

Bài toán

Yêu cầu: Thiết kế các lớp thích hợp để thực hiện các yêu cầu sau:

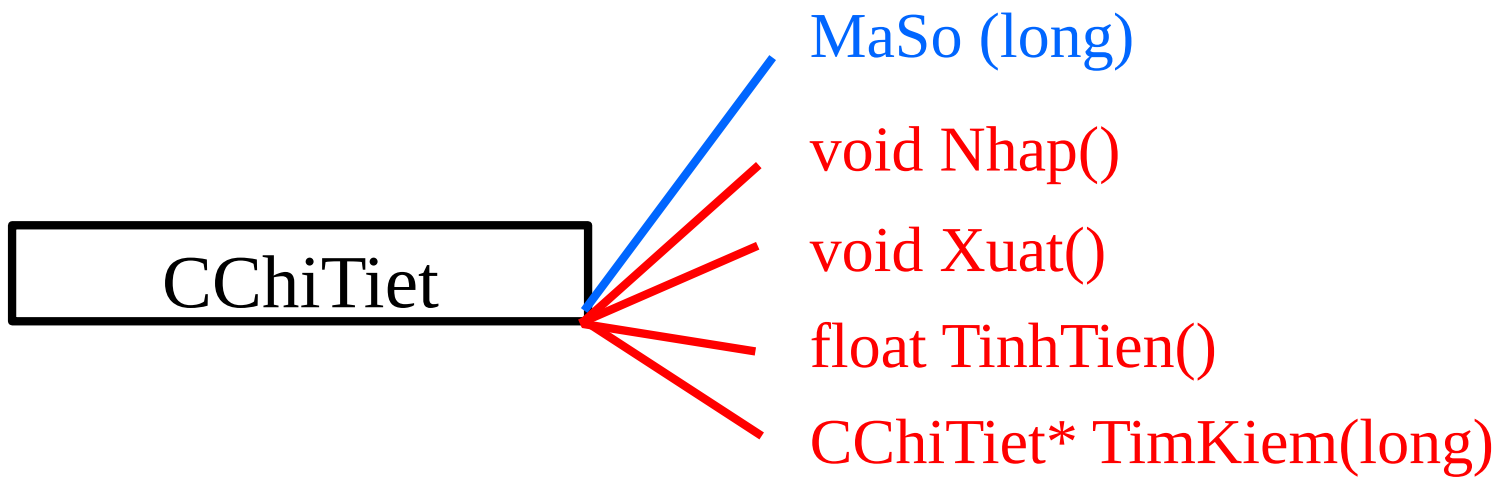
- Nhập các chi tiết cho máy.
- Xuất các chi tiết máy.
- Tính trị giá của máy.
- Tìm kiếm một chi tiết máy theo mã số.
- Đếm số lượng chi tiết đơn có trong cái máy (Bài tập về nhà).


CMay
Có
CChiTiet
CChiTietDon
CChiTietPhuc


CMay
Có
CChiTiet
MaSo (long)
void Nhap()
void Xuat()
float TinhTien()
CChiTiet* TimKiem(long)
n (số lượng chi tiết)
CChiTiet ds[100]
void Nhap()
void Xuat()
float TinhTien()
CChiTiet* TimKiem(long)
CChiTietDon
GiaTien
void Nhap()
void Xuat()
float TinhTien()
CChiTiet* TimKiem(long)
CChiTietPhuc
n (số lượng chi tiết)
CChiTiet ds[100]
void Nhap()
void Xuat()
float TinhTien()
CChiTiet* TimKiem(long)

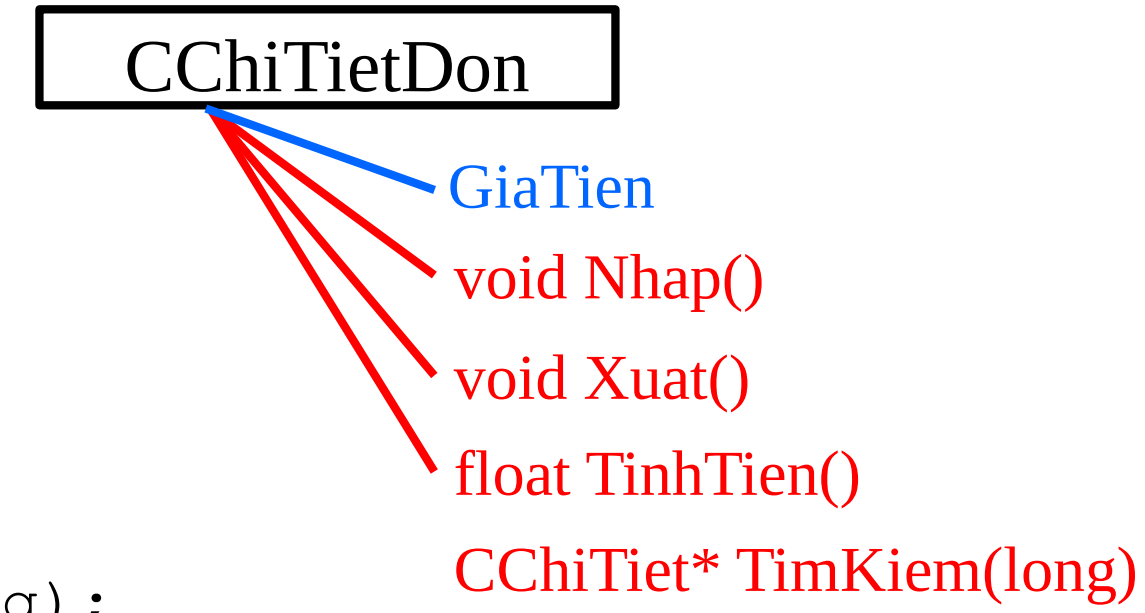
Khai báo lớp

```
11.class CChiTiet
12.{
13.    protected:
14.        long maso;
15.    public:
16.        void Nhap();
17.        void Xuat();
18.        float TinhTien();
19.        CChiTiet* TimKiem(long);
20.};
```



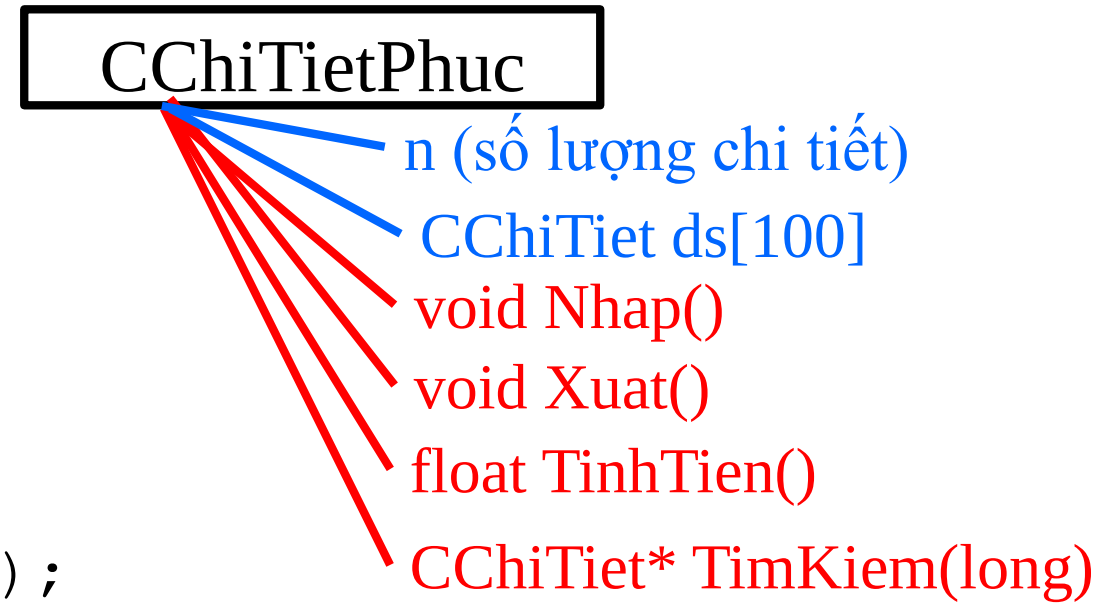
Khai báo lớp

```
11.class CChiTietDon:public CChiTiet
12.{
13.    protected:
14.        float giatien;
15.    public:
16.        void Nhap();
17.        void Xuat();
18.        float TinhTien();
19.        CChiTiet* TimKiem(long);
20.};
```



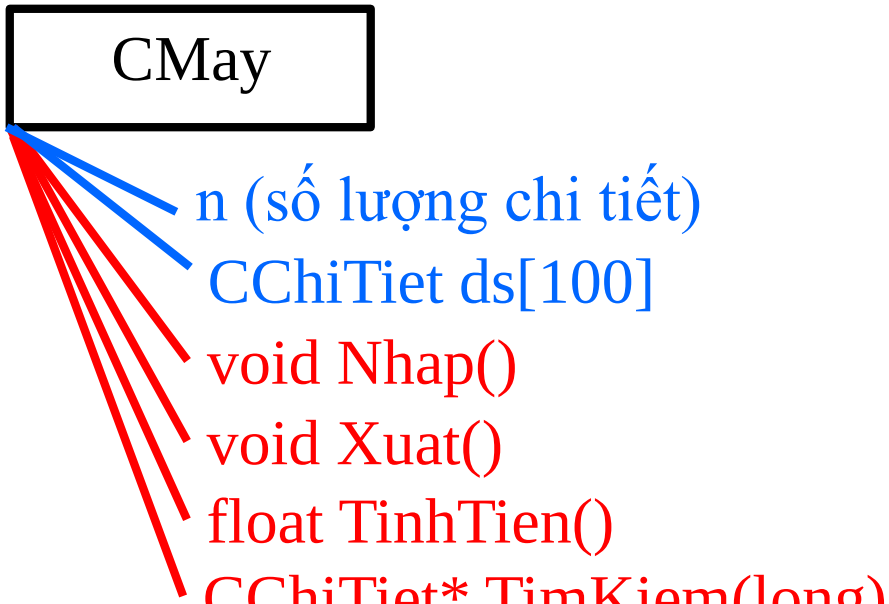
Khai báo lớp

```
11.class CChiTietPhuc:public CChiTiet
12.{
13.    protected:
14.        int n;
15.        CChiTiet ds[100];
16.    public:
17.        void Nhap();
18.        void Xuat();
19.        float TinhTien();
20.        CChiTiet* TimKiem(long);
21.};
```



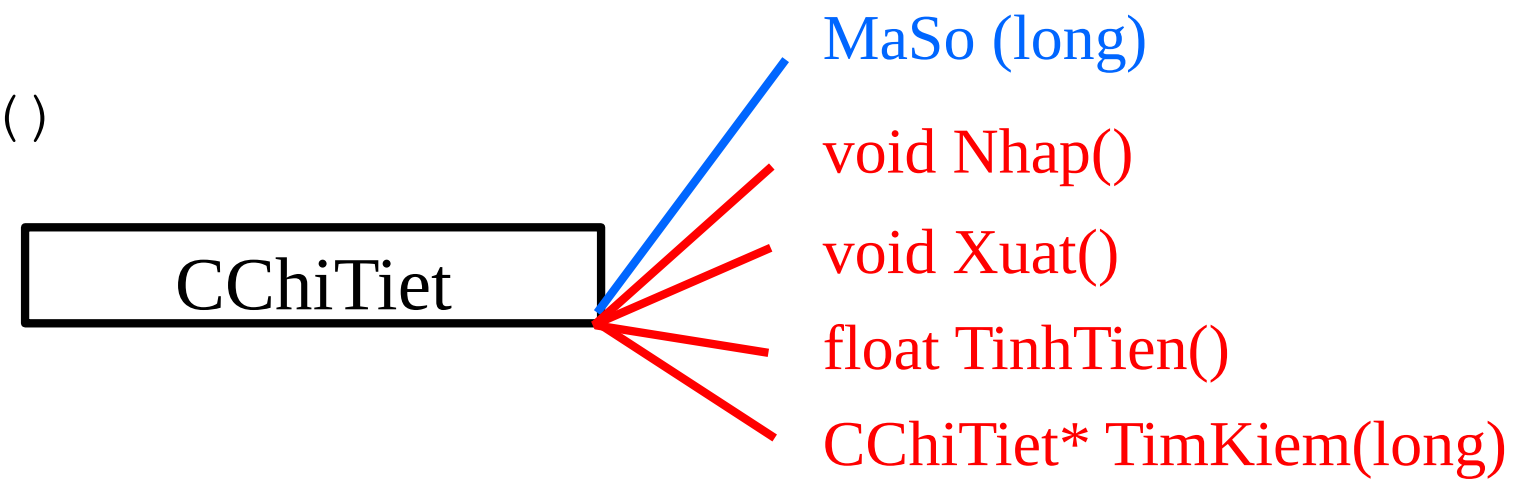
Khai báo lớp

```
11.class CMay
12.{
13.    protected:
14.        int n;
15.        CChiTiet ds[100];
16.    public:
17.        void Nhap();
18.        void Xuat();
19.        float TinhTien();
20.        CChiTiet* TimKiem(long);
21.};
```



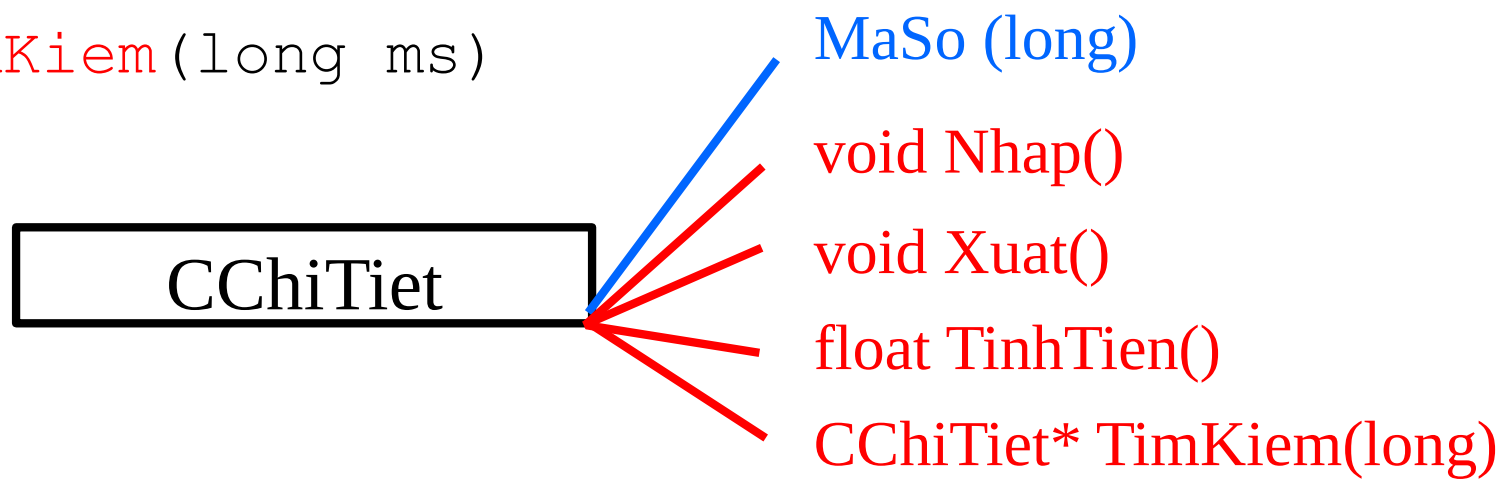
Định nghĩa các phương thức

```
11.void CChiTiet::Nhap()
12.{
13.|    return;
14.}
15.void CChiTiet::Xuat()
16.{
17.|    return;
18.}
```



Định nghĩa các phương thức

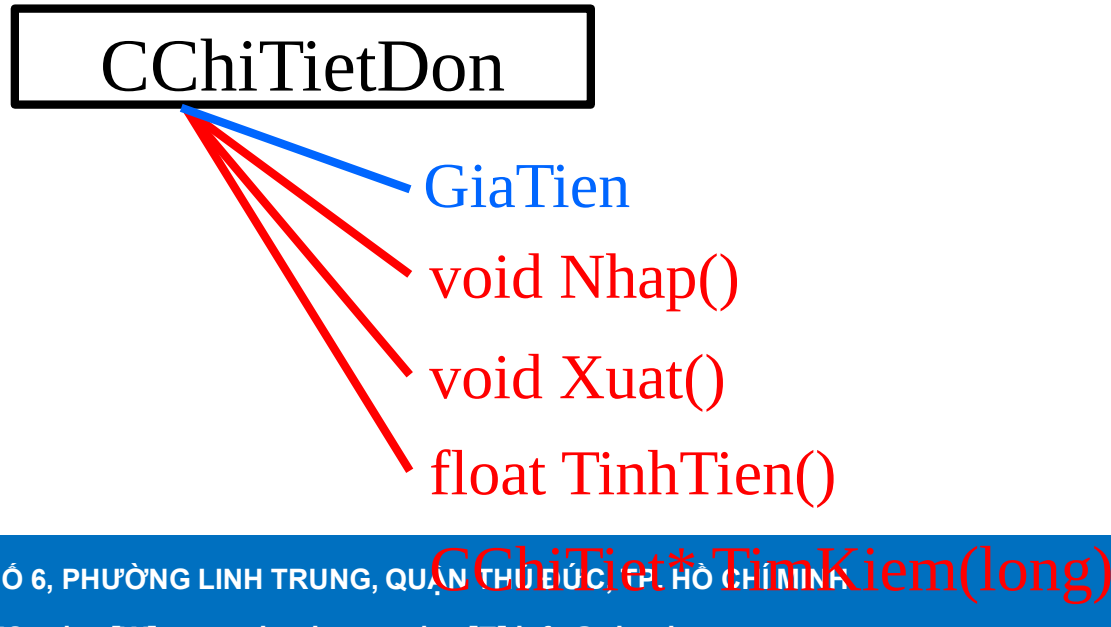
```
11.float CChiTiet::TinhTien()
12.{
13.|    return 0;
14.}
15.CChiTiet* CChiTiet::TimKiem(long ms)
16.{
17.|    if(maso==ms)
18.|        return this;
19.|    return NULL;
20.}
```



Định nghĩa các phương thức

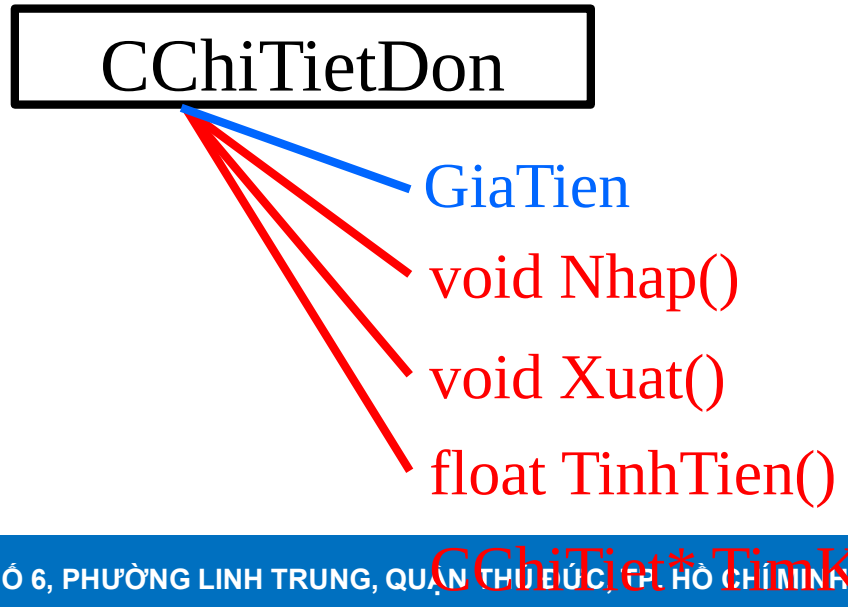
```
11.void CChiTietDon::Nhap()
12.{
13.|    cout<<"Nhap ma so:";
14.|    cin>>maso;
15.|    cout<<"Nhap gia:";
16.|    cin>>giatien;
17.}
```

```
11.void CChiTietDon::Xuat()
12.{
13.|    cout<<"Ma so:"<<maso;
14.|    cout<<"Gia:"<<giatien
15.}
```



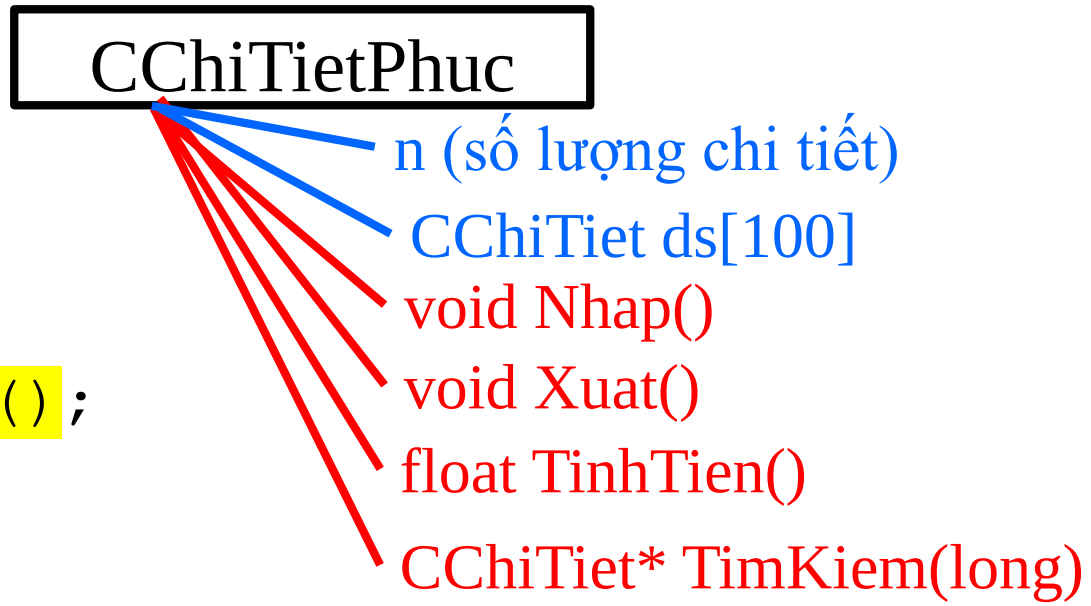
Định nghĩa các phương thức

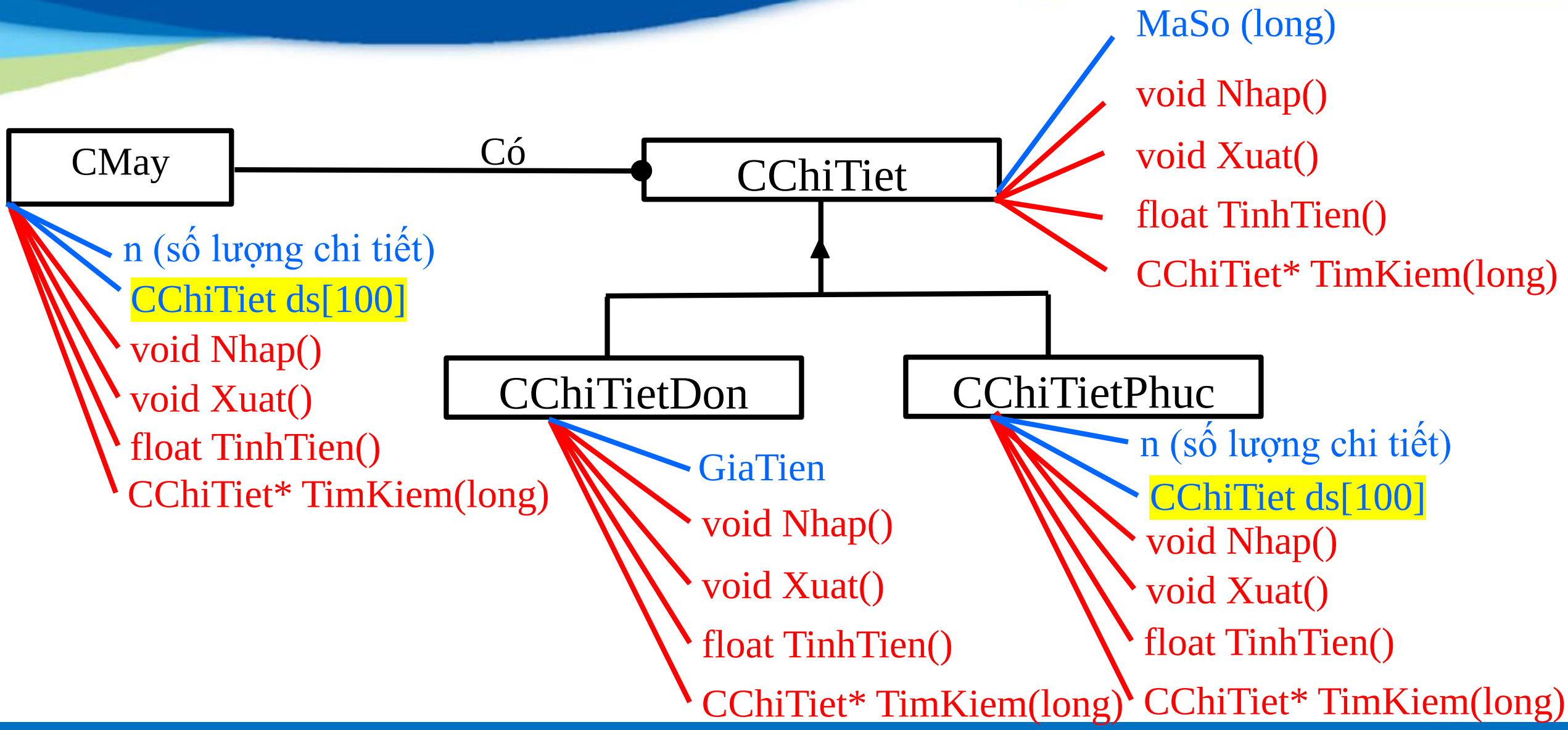
```
11.float CChiTietDon::TinhTien()
12.{
13.|    return giatien;
14.}
15.CChiTiet* CChiTietDon::TimKiem(long ms)
16.{
17.|    if(maso==ms)
18.|        return this;
19.|    return NULL;
20.}
```



Định nghĩa các phương thức

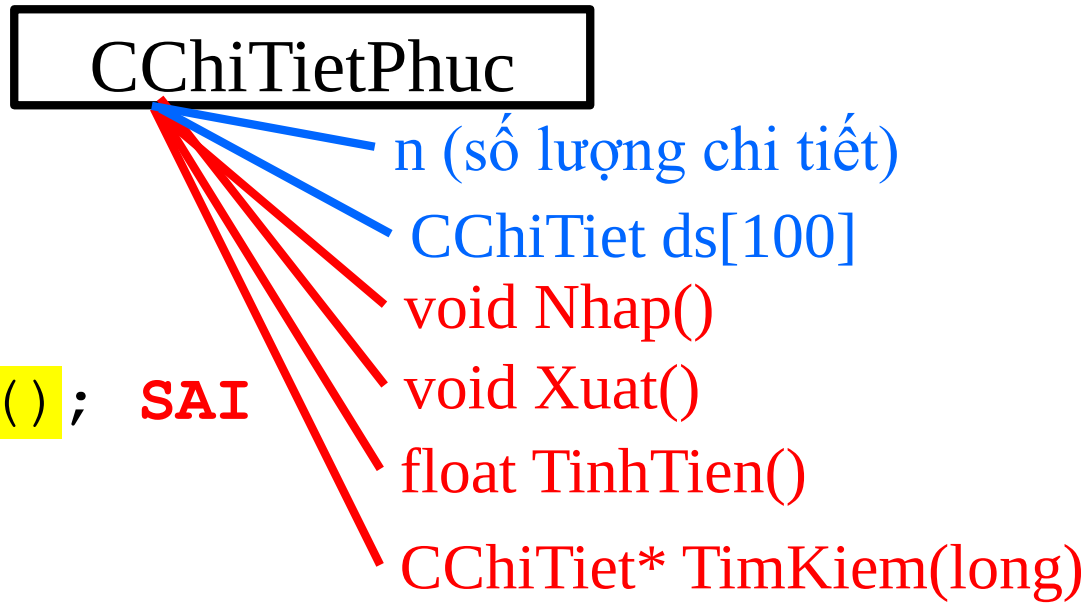
```
11.float CChiTietPhuc::TinhTien()
12.{
13.    float s = 0;
14.    for(int i=0;i<n;i++)
15.        s = s + ds[i].TinhTien();
16.    return s;
17.}
```

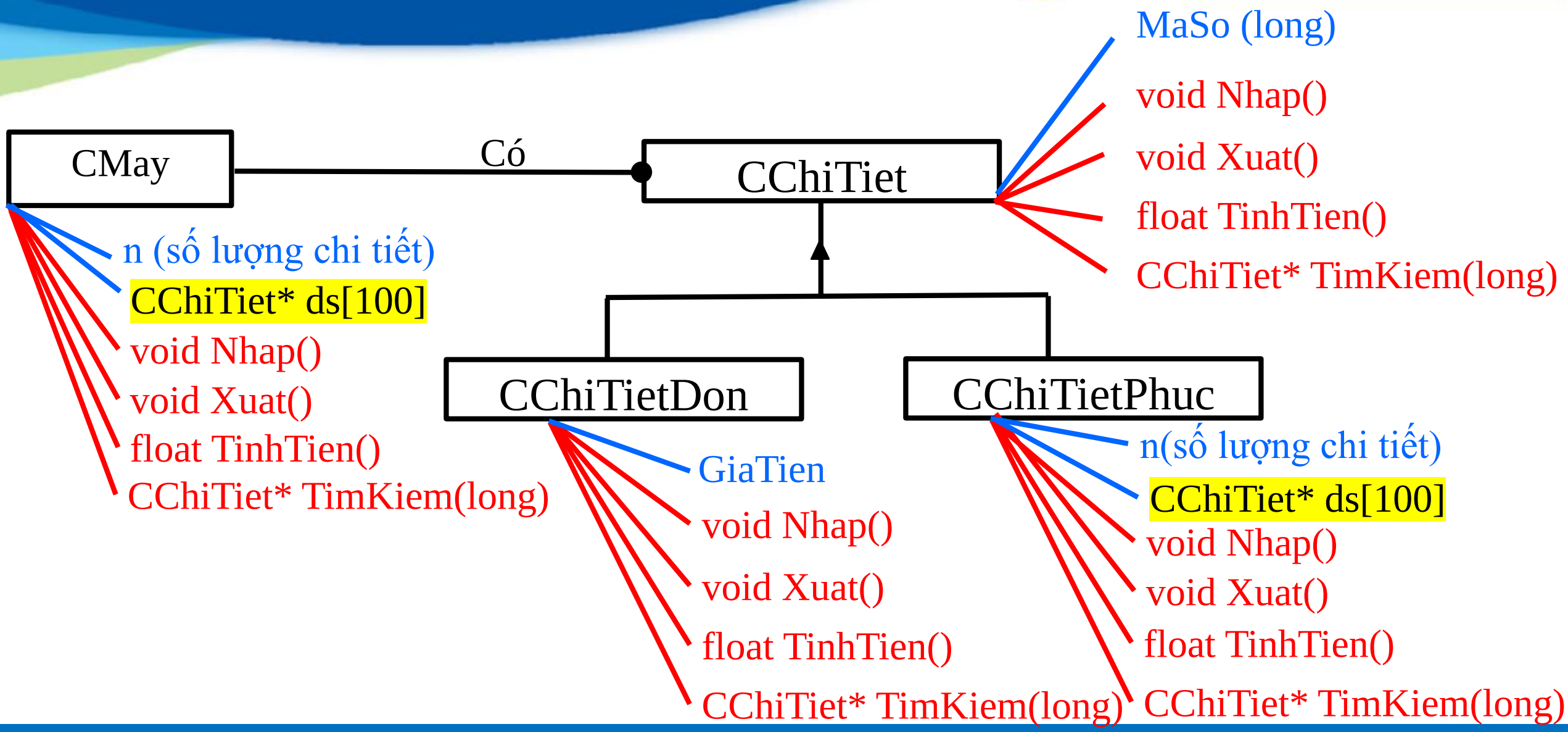



CMay
Có
CChiTiet
MaSo (long)
void Nhap()
void Xuat()
float TinhTien()
CChiTiet* TimKiem(long)
n (số lượng chi tiết)
CChiTiet ds[100]
void Nhap()
void Xuat()
float TinhTien()
CChiTiet* TimKiem(long)
CChiTietDon
GiaTien
void Nhap()
void Xuat()
float TinhTien()
CChiTiet* TimKiem(long)
CChiTietPhuc
n (số lượng chi tiết)
CChiTiet ds[100]
void Nhap()
void Xuat()
float TinhTien()
CChiTiet* TimKiem(long)

Định nghĩa các phương thức

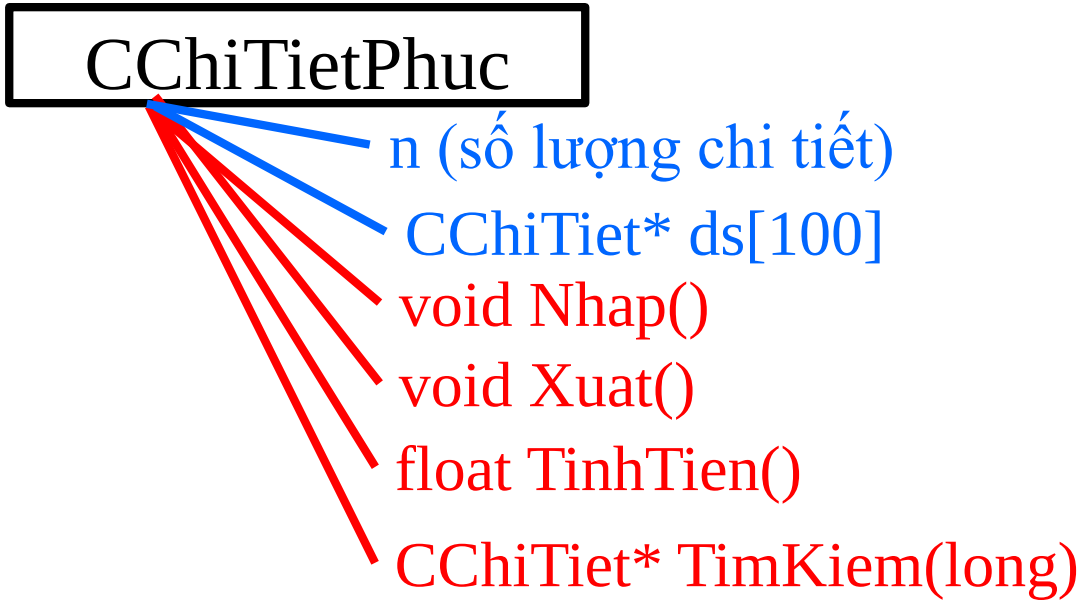
```
11.float CChiTietPhuc::TinhTien()
12.{
13.|    float s = 0;
14.|    for(int i=0;i<n;i++)
15.|        s = s + ds[i].TinhTien(); SAI
16.|    return s;
17.}
```




CMay
Có
CChiTiet
MaSo (long)
void Nhap()
void Xuat()
float TinhTien()
CChiTiet* TimKiem(long)
n (số lượng chi tiết)
CChiTiet* ds[100]
void Nhap()
void Xuat()
float TinhTien()
CChiTiet* TimKiem(long)
CChiTietDon
GiaTien
void Nhap()
void Xuat()
float TinhTien()
CChiTiet* TimKiem(long)
CChiTietPhuc
n(số lượng chi tiết)
CChiTiet* ds[100]
void Nhap()
void Xuat()
float TinhTien()
CChiTiet* TimKiem(long)

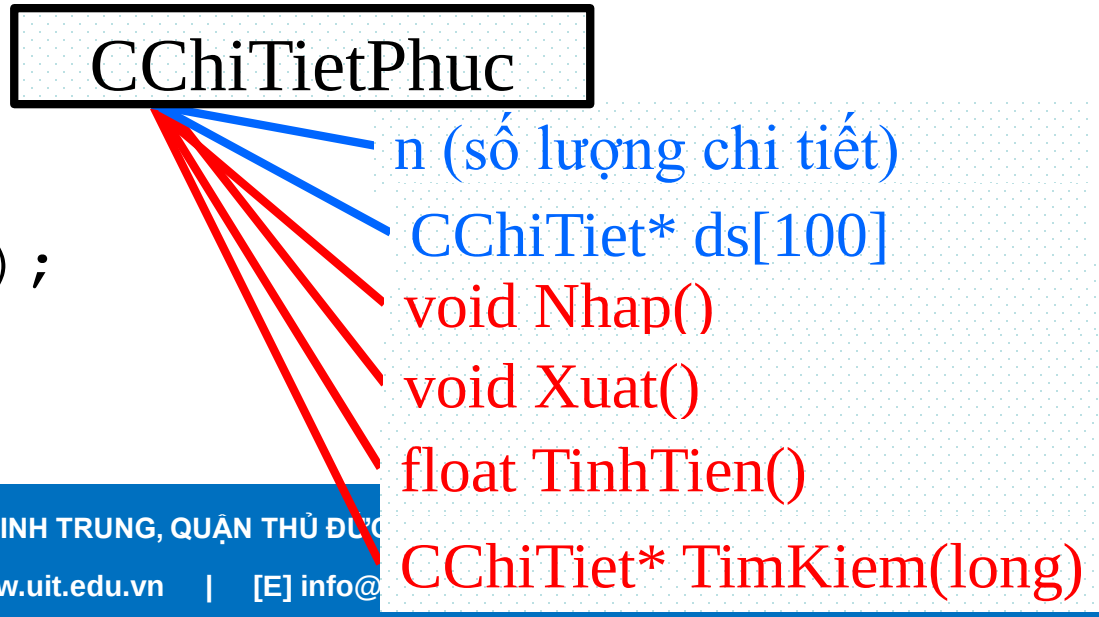
Khai báo lớp

```
11.class CChiTietPhuc:public CChiTiet
12.{
13.    protected:
14.        int n;
15.        CChiTiet* ds[100];
16.    public:
17.        void Nhap();
18.        void Xuat();
19.        float TinhTien();
20.        CChiTiet* TimKiem(long);
21.};
```



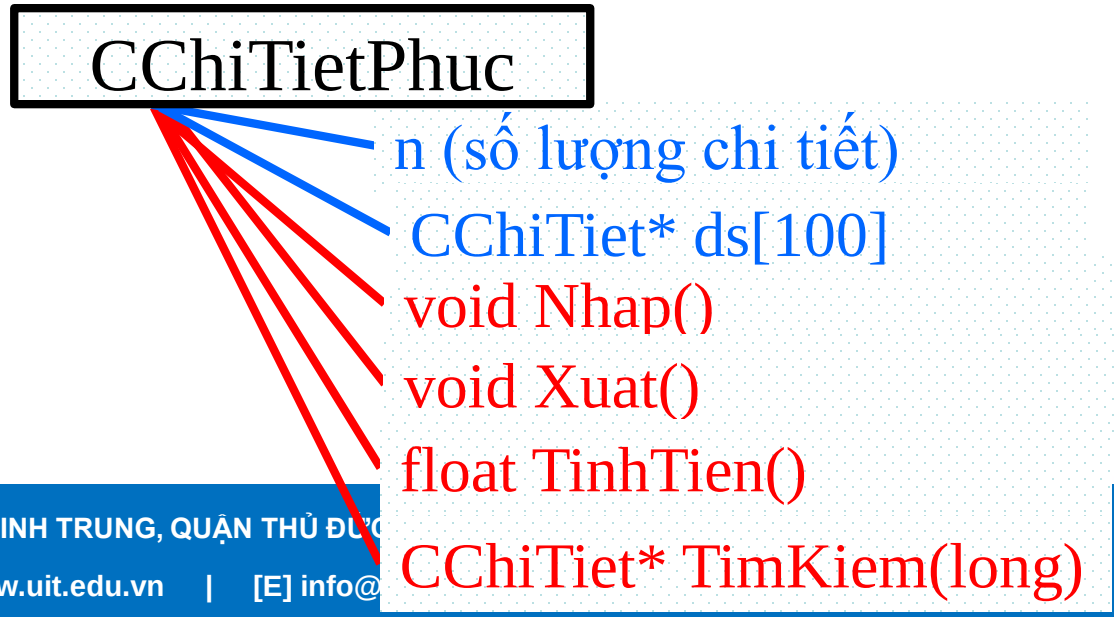
Khai báo lớp

```
11.class CMay
12.{
13.    protected:
14.        int n;
15.        CChiTiet* ds[100];
16.    public:
17.        void Nhap();
18.        void Xuat();
19.        float TinhTien();
20.        CChiTiet* TimKiem(long);
21.};
```



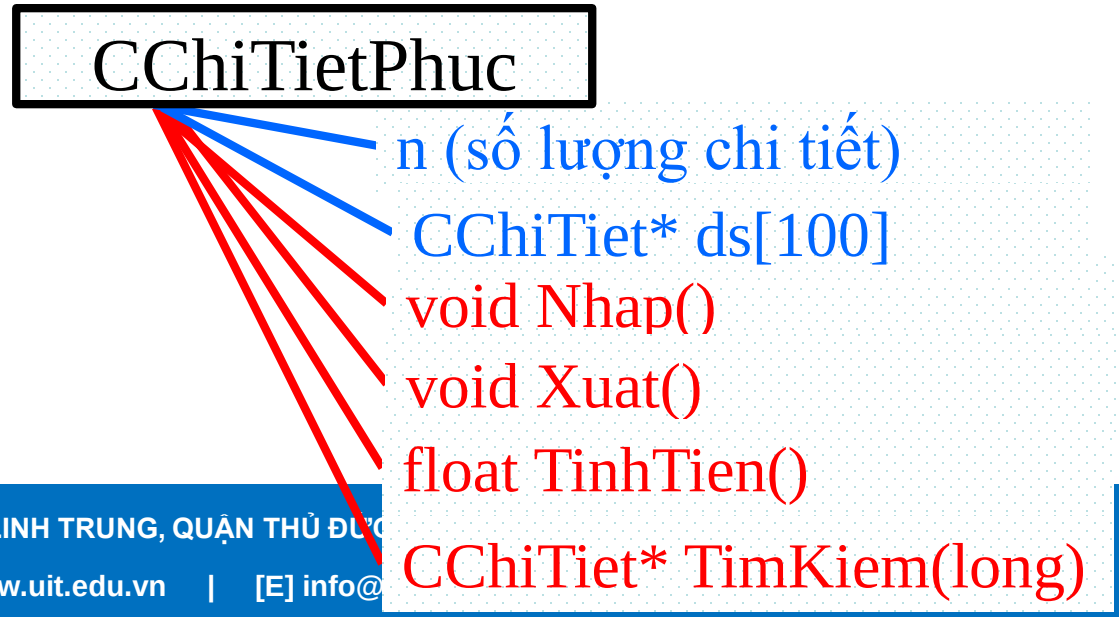
Định nghĩa các phương thức

```
11.float CChiTietPhuc::TinhTien()
12.{
13.    float s = 0;
14.    for(int i=0;i<n;i++)
15.        s = s + ds[i]->TinhTien();
16.    return s;
17.}
```



Định nghĩa các phương thức

```
11.float CChiTietPhuc::TinhTien()
12.{
13.    float s = 0;
14.    for(int i=0;i<n;i++)
15.        s = s + ds[i]->TinhTien(); SAI
16.    return s;
17.}
```

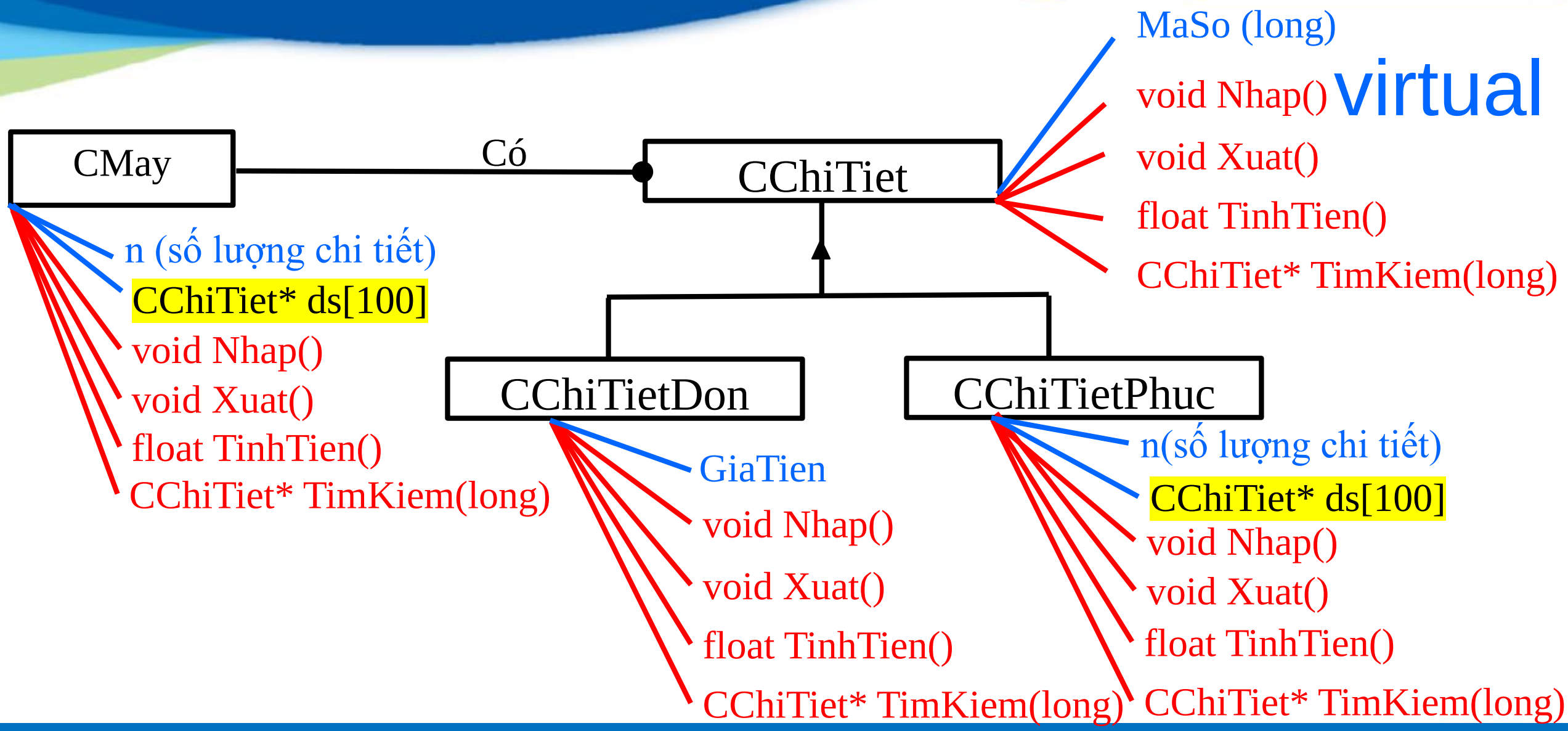


Nhắc lại lý thuyết đa xạ

- Khái niệm: Đa xạ là cơ chế **tầm vực động** (dynamic scope) cho phép "*xác định*" đúng hành vi (phương thức – method) của đối tượng (object) khi yêu cầu thực hiện.
- Việc "*xác định*" được thực hiện theo nguyên tắc tự nhiên: con trỏ đối tượng đang giữ địa chỉ của đối tượng thuộc về lớp nào thì nó sẽ gọi thực hiện phương thức của lớp đối tượng (class) đó.
- Tầm vực động (dynamic scope) là cơ chế gọi thực hiện phương thức thông qua con trỏ đối tượng.

Nhắc lại lý thuyết đa xạ

- Một phương thức được khai báo bắt đầu với từ khóa virtual thì được gọi là phương thức ảo và phương thức này được gọi thực hiện theo cơ chế đa xạ nếu lời gọi thực hiện phương thức được thông qua một con trỏ đối tượng.
- Các phương thức ở lớp dẫn xuất cùng tên và cùng danh sách tham số đầu vào thì cũng sẽ là phương thức ảo nếu ở lớp cơ sở phương thức cùng tên và cùng tham số là phương thức ảo.


CMay
Có
CChiTiet
MaSo (long)
void Nhap() virtual
void Xuat()
float TinhTien()
CChiTiet* TimKiem(long)
n (số lượng chi tiết)
CChiTiet* ds[100]
void Nhap()
void Xuat()
float TinhTien()
CChiTiet* TimKiem(long)
CChiTietDon
GiaTien
void Nhap()
void Xuat()
float TinhTien()
CChiTiet* TimKiem(long)
CChiTietPhuc
n(số lượng chi tiết)
CChiTiet* ds[100]
void Nhap()
void Xuat()
float TinhTien()
CChiTiet* TimKiem(long)

Khai báo lớp

```
11.class CChiTiet
12.{
13.    protected:
14.        long maso;
15.    public:
16.        virtual void Nhap();
17.        virtual void Xuat();
18.        virtual float TinhTien();
19.        virtual CChiTiet* TimKiem(long);
20.};
```

Khai báo lớp

```
11.class CChiTietPhuc:public CChiTiet
12.{
13.    protected:
14.        int n;
15.        CChiTiet* ds[100];
16.    public:
17.        void Nhap();
18.        void Xuat();
19.        float TinhTien();
20.        CChiTiet* TimKiem(long);
21.};
```

virtual

Các phương thức ở lớp dẫn xuất cùng tên và cùng danh sách tham số đầu vào thì cũng sẽ là phương thức ảo nếu ở lớp cơ sở phương thức cùng tên và cùng tham số là phương thức ảo.

Khai báo lớp

```
11.class CChiTietDon:public CChiTiet
12.{
13.    protected:
14.        float giatien;
15.    public:
16.        void Nhap();
17.        void Xuat();
18.        float TinhTien();
19.        CChiTiet* TimKiem(long);
20.};
```

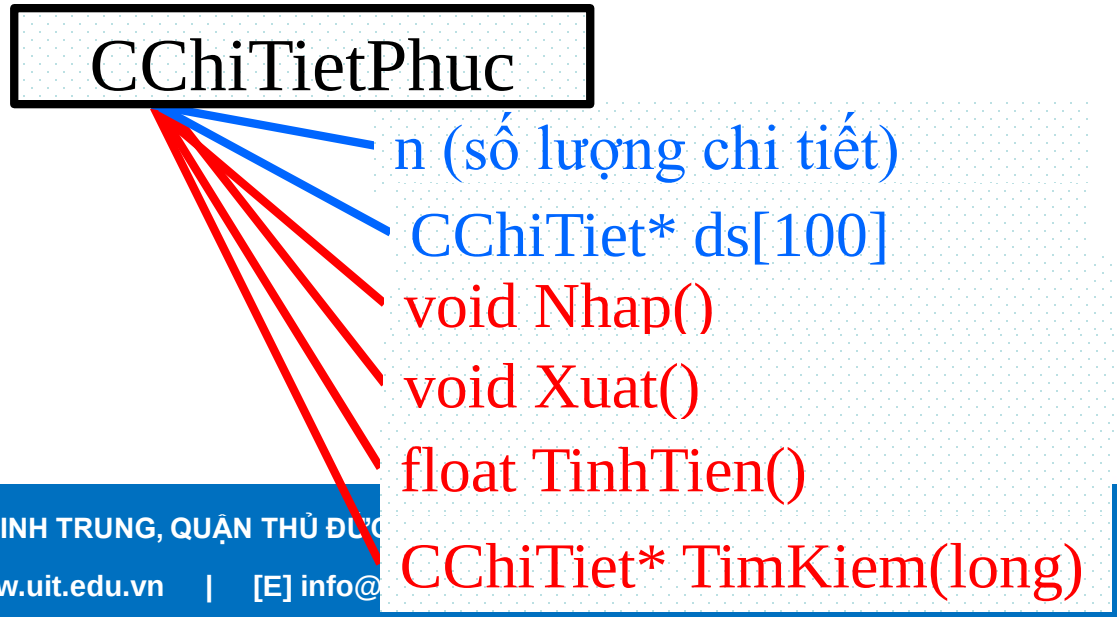
virtual

Các phương thức ở lớp dẫn xuất cùng tên và cùng danh sách tham số đầu vào thì cũng sẽ là phương thức ảo nếu ở lớp cơ sở phương thức cùng tên và cùng tham số là phương thức ảo.

Định nghĩa các phương thức

```
11.float CChiTietPhuc::TinhTien()
12.{
13.    float s = 0;
14.    for(int i=0;i<n;i++)
15.        s = s + ds[i]->TinhTien();
16.    return s;
17.}
```

ĐÚNG



Định nghĩa các phương thức

```
11.float CChiTietPhuc::TinhTien()
12.{
13.    float s = 0;
14.    for(int i=0;i<n;i++)
15.        s = s + ds[i]->TinhTien();
16.    return s;
17.}
```

ĐÚNG

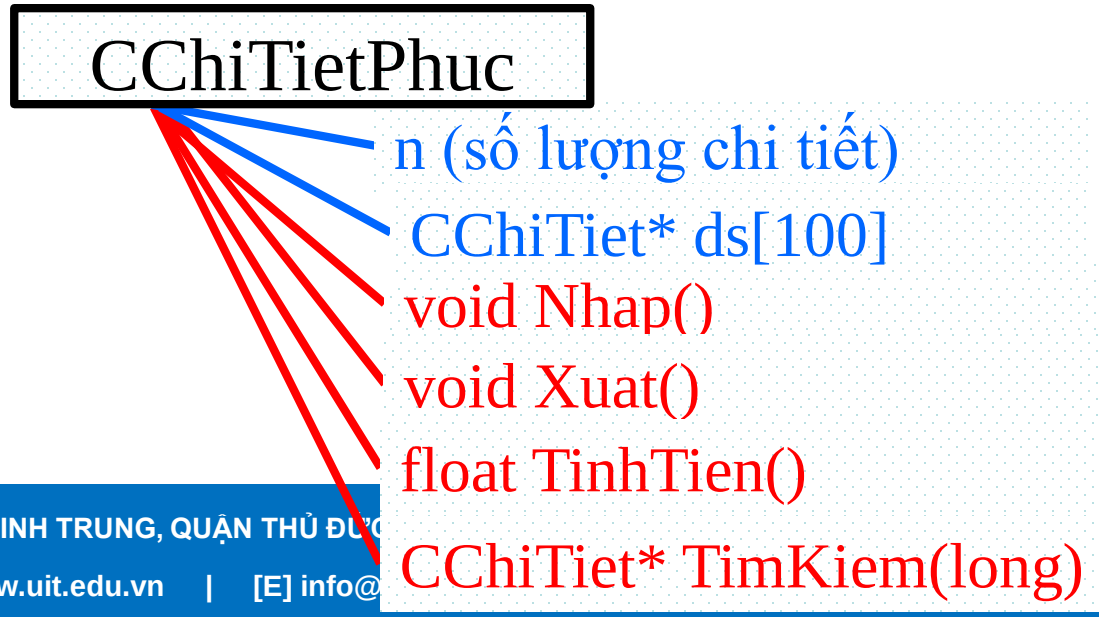
– Dòng 15 ta nói con trỏ đối tượng `ds[i]` gọi thực thực hiện phương thức `TinhTien` và đây là phương thức ảo nên việc gọi thực hiện phương thức này sẽ theo cơ chế đa xạ.

Nhắc lại lý thuyết đa xạ

- Khái niệm: Đa xạ là cơ chế **tầm vực động** (dynamic scope) cho phép "*xác định*" đúng hành vi (phương thức – method) của đối tượng (object) khi yêu cầu thực hiện.
- Việc "*xác định*" được thực hiện theo nguyên tắc tự nhiên: con trỏ đối tượng đang giữ địa chỉ của đối tượng thuộc về lớp nào thì nó sẽ gọi thực hiện phương thức của lớp đối tượng (class) đó.
- Tầm vực động (dynamic scope) là cơ chế gọi thực hiện phương thức thông qua con trỏ đối tượng.

Định nghĩa các phương thức

```
11.void CChiTietPhuc::Xuat()
12.{
13.    cout<<"\n Ma so:"<<maso;
14.    cout<<"\n So luong chi tiet thanh phan:"<<n;
15.    for(int i=0;i<n;i++)
16.        ds[i]->Xuat();
17.}
```



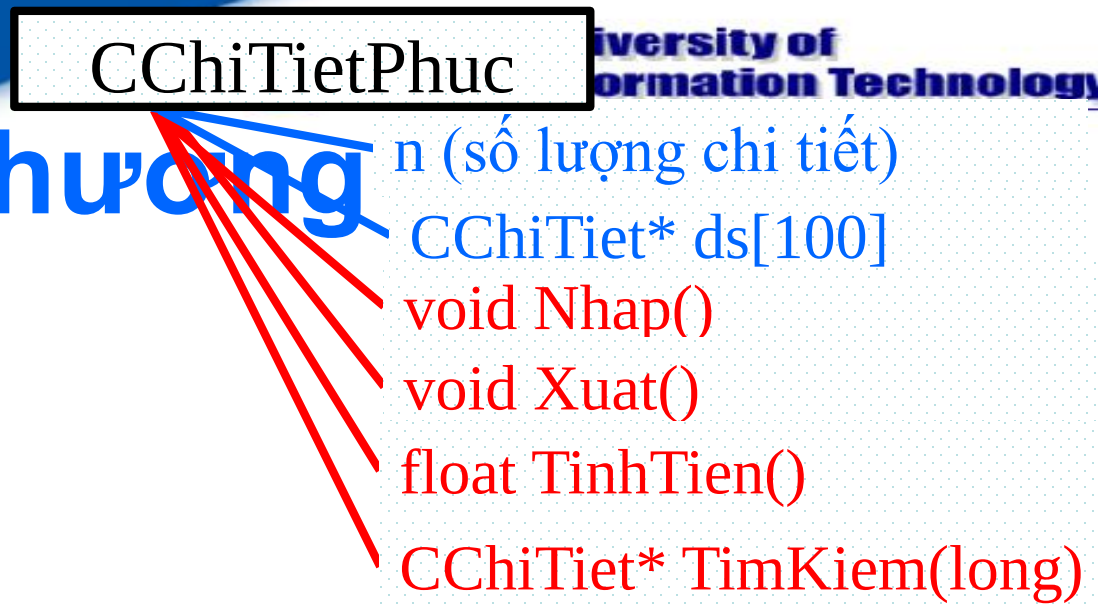
Định nghĩa các phương

CChiTietPhuc

- n (số lượng chi tiết)
- CChiTiet* ds[100]
- void Nhap()
- void Xuat()
- float TinhTien()
- CChiTiet* TimKiem(long)

```
11.CChiTiet* CChiTietPhuc::TimKiem(long ms)
12.{
13.    if(maso==ms)
14.        return this;
15.    for(int i=0;i<n;i++)
16.    {
17.        CChiTiet* kq = ds[i]->TimKiem(ms);
18.        if(kq!=NULL)
19.            return kq;
20.    }
21.    return NULL;
22.}
```

Định nghĩa các phương




```
11.void CChiTietPhuc::Nhap()
12.{
13.    cout<<"Nhap ma so:";
14.    cin>>maso;
15.    cout<<"Nhap so luong chi tiet thanh phan:";
16.    cin>>n;
17.    for(int i=0;i<n;i++)
18.    {
19.        cout<<"Nhap chi tiet a["<<i<<"]:";
20.        ds[i]->Nhap();
21.    }
22.}
```

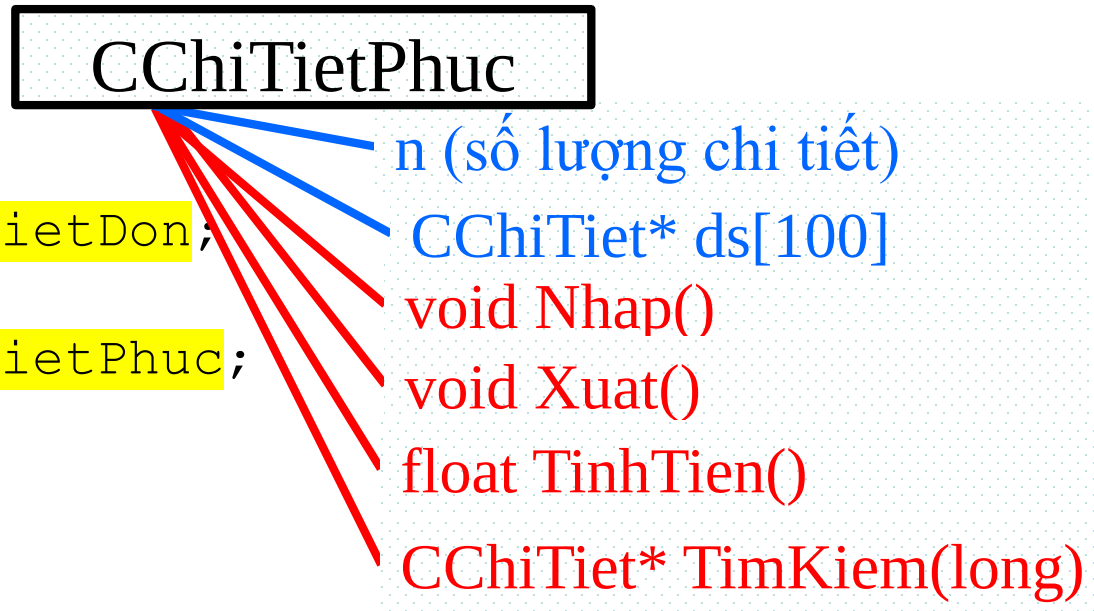
Định nghĩa các phương

CChiTietPhuc

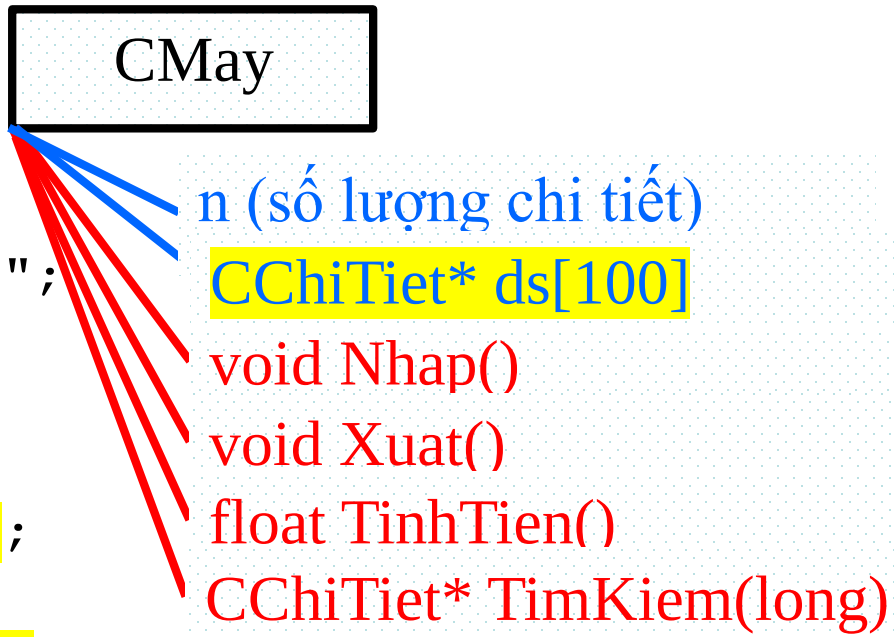
- n (số lượng chi tiết)
- CChiTiet* ds[100]
- void Nhap()
- void Xuat()
- float TinhTien()
- CChiTiet* TimKiem(long)

```
11.void CChiTietPhuc::Nhap()
12.{
13.    cout<<"Nhap ma so:";
14.    cin>>maso;
15.    cout<<"Nhap so luong chi tiet thanh phan:";
16.    cin>>n;
17.    for(int i=0;i<n;i++)
18.    {
19.        cout<<"Nhap chi tiet a["<<i<<"]:";
20.        ds[i]->Nhap(); SAI
21.    }
22.}
```

```
11.void CChiTietPhuc::Nhap()
12.{
13.    cout<<"Nhap ma so:";
14.    cin>>maso;
15.    cout<<"Nhap so luong chi tiet thanh phan:";
16.    cin>>n;
17.    for(int i=0;i<n;i++)
18.    {
19.        int loai;
20.        cout<<"Nhap loai (0. Don, 1. Phuc):";
21.        cin>>loai;
22.        switch(loai)
23.        {
24.            case 0: ds[i] = new CChiTietDon;
25.                break;
26.            case 1: ds[i] = new CChiTietPhuc;
27.                break;
28.        }
29.        ds[i]->Nhap();
30.    }
31.}
```

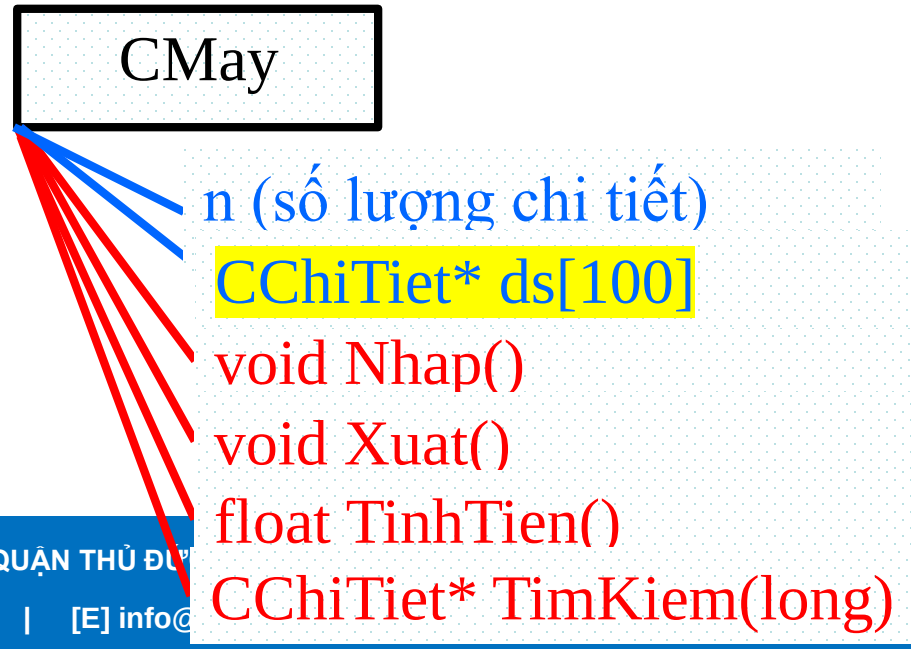


```
11.void CMay::Nhap()
12.{
13.    cout<<"Nhap so luong chi tiet thanh phan:";
14.    cin>>n;
15.    for(int i=0;i<n;i++)
16.    {
17.        int loai;
18.        cout<<"Nhap loai (0. Don, 1. Phuc):";
19.        cin>>loai;
20.        switch(loai)
21.        {
22.            case 0: ds[i] = new CChiTietDon;
23.                break;
24.            case 1: ds[i] = new CChiTietPhuc;
25.                break;
26.        }
27.        ds[i]->Nhap();
28.    }
29.}
```



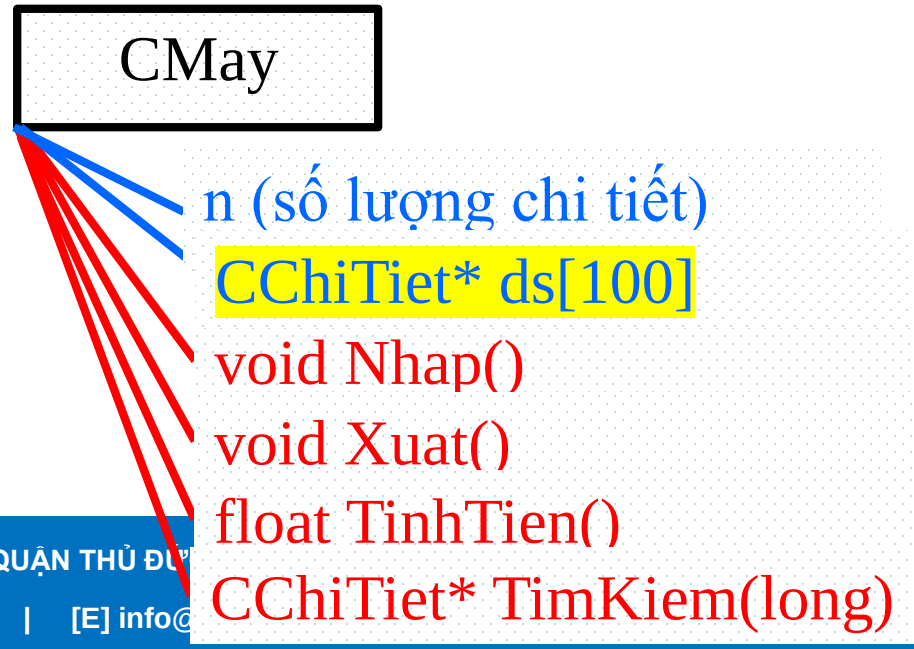
Định nghĩa các phương thức

```
11.void CMay::Xuat()
12.{
13.     cout<<"\n So luong chi tiet thanh phan:"<<n;
14.     for(int i=0;i<n;i++)
15.          ds[i]->Xuat();
16.}
```



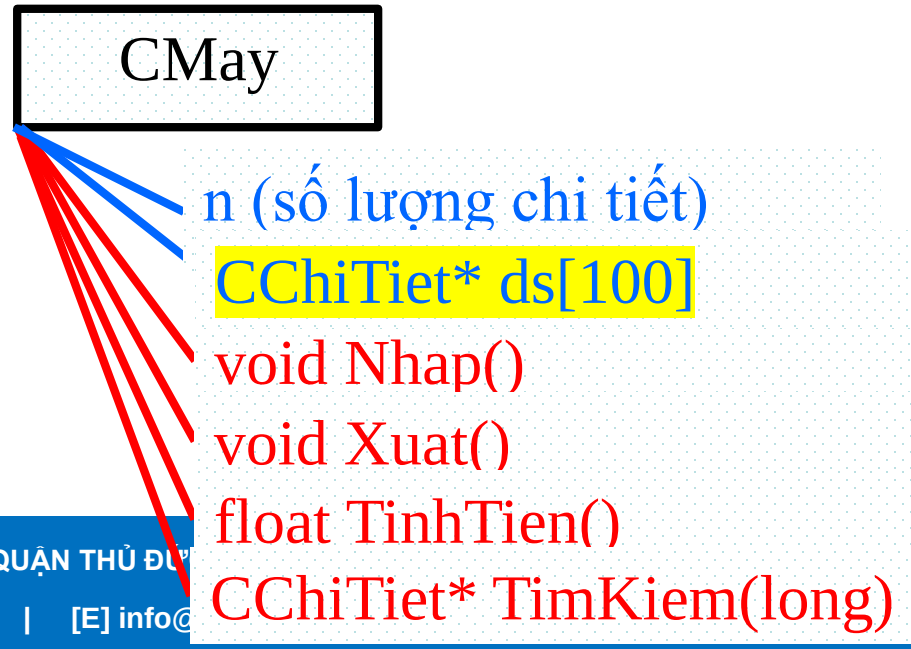
Định nghĩa các phương thức

```
11.CChiTiet* CMay::TimKiem(long ms)
12.{
13.    for(int i=0;i<n;i++)
14.    {
15.        CChiTiet* kq = ds[i]->TimKiem(ms);
16.        if(kq!=NULL)
17.            return kq;
18.    }
19.    return NULL;
20.}
```



Định nghĩa các phương thức

```
11.float CMay::TinhTien()
12.{
13.    float s = 0;
14.    for(int i=0;i<n;i++)
15.        s = s + ds[i]->TinhTien();
16.    return s;
17.}
```



```
11.class CMachDien
12.{
13.    public:
14.        virtual void Nhap();
15.        virtual float TongTro();
16.};
```

```
11.class CMachSongSong:public CMachDien
12.{
13.    protected:
14.        int n;
15.        CMachDien* ds[100];
16.    public:
17.        void Nhap();
18.        float TongTro();
19.};
```

```
11.class CMachNoiTiep:public CMachDien
12.{
13.    protected:
14.        int n;
15.        CMachDien* ds[100];
16.    public:
17.        void Nhap();
18.        float TongTro();
19.};
```

```
11.class CDienTro:public CMachDien
12.{
13.    protected:
14.        float R;
15.    public:
16.        void Nhap();
17.        float TongTro();
18.};
```

```
11.float CMachNoiTiep::TongTro()
12.{
13.     float s=0;
14.     for(int i=0;i<n;i++)
15.          s = s + ds[i]->TongTro();
16.     return s;
17.}
```

```
11.float CMachSongSong::TongTro()
12.{
13.     float s=0;
14.     for(int i=0;i<n;i++)
15.          s = s + 1/ds[i]->TongTro();
16.     return 1/s;
17.}
```

```
11.float CDienTro::TongTro()
12.{
13.     return R;
14.}
```

```
11.float CMachDien::TongTro()
12.{
13.     return 0;
14.}

15.void CMachDien::Nhap()
16.{
17.     return;
18.}
```

```
11.void CDienTro::Nhap()
12.{
13.     cout<<"Nhap R:";
14.     cin>>R;
15.}
```

```
11. void CMachNoiTiep::Nhap()
12. {
13.     cout<<"Nhap n:";
14.     cin>>n;
15.     for(int i=0;i<n;i++)
16.     {
17.         int loai;
18.         cout<<"Nhap loai (0. Noi tiep, 1. Song song):";
19.         cin>>loai;
20.         switch(loai)
21.         {
22.             case 0: ds[i] = new CMachNoiTiep;
23.                 break;
24.             case 1: ds[i] = new CMaSongSong;
25.                 break;
26.         }
27.         ds[i]->Nhap();
28.     }
29. }
```

```
11. void CMachSongSong::Nhap()
12. {
13.     cout<<"Nhap n:";
14.     cin>>n;
15.     for(int i=0;i<n;i++)
16.     {
17.         int loai;
18.         cout<<"Nhap loai (0. Noi tiep, 1. Song song):";
19.         cin>>loai;
20.         switch(loai)
21.         {
22.             case 0: ds[i] = new CMachNoiTiep;
23.                 break;
24.             case 1: ds[i] = new CMaSongSong;
25.                 break;
26.         }
27.         ds[i]->Nhap();
28.     }
29. }
```

```
11.void main()
12.{
13.     CMachDien *a;
14.     int loai;
15.     cout<<"Nhap loai (0. NT, 1. SS, 2. R:";
16.     cin>>loai;
17.     switch(loai)
18.     {
19.          case 0: a = new CMachNoiTiep;
20.               break;
21.          case 1: a = new CMaSongSong;
22.               break;
23.
24.     }
25.     a->Nhap();
26.     …
1. }
```