3.7. Hàm bạn

Trong thực tế thường xảy ra trường hợp có một số lớp cần sử dụng chung một hàm. C++ giải quyết vấn đề này bằng cách dùng hàm bạn. Để một hàm trở thành bạn của một lớp:

Dùng từ khóa ***friend*** để khai báo hàm trong lớp và xây dựng hàm bên ngoài như các hàm thông thường (không dùng từ khóa friend). Mẫu viết như sau :

```
class  A
        {
                private : ...
                public : ...
                ...
                // Khai báo các hàm bạn của lớp A
                friend  void      f1 (...) ;
                friend  double   f2 (...) ;
                ...
        } ;
// Xây dựng các hàm f1,f2
void f1 (...)  //không có tên lớp
        { …}
double f2 (...)
        { …}
```

Hàm bạn có những tính chất sau:

- Hàm bạn không phải là phương thức của lớp.
- Việc truy cập tới hàm bạn được thực hiện như hàm thông thường.
- Trong thân hàm bạn của một lớp có thể truy cập tới các thuộc tính của đối tượng thuộc lớp này. Đây là sự khác nhau duy nhất giữa hàm bạn và hàm thông thường.
- Một hàm có thể là bạn của nhiều lớp. Lúc đó nó có quyền truy cập tới tất cả các thuộc tính của các đối tượng trong các lớp này.
- Hàm bạn được định nghĩa trong vùng public

Chú ý: Do hàm bạn không phải là phương thức của lớp nên tham số của hàm bạn không có con trỏ this

Để làm cho hàm f trở thành bạn của các lớp A, B, giả sử khi định nghĩa lớp A có dùng lớp B, lúc đó ta cần khai báo trước lớp B như sau :

```
class B ; // Khai báo trước lớp B
// Định nghĩa  lớp A
class A
        {       // Khai báo f là bạn của A
                friend void f(... ); //có dùng lớp B
        } ;
// Định nghĩa lớp B
class B
        {       // Khai báo f là bạn của B
                friend void f(...); }
        } ;
// Xây dựng hàm bạn f cua cac lop A va B
void f(...)
  { ... }
```

Ví dụ 3.11. Chương trình sau minh họa cách khai báo trước của một lớp. Giả sử các lớp A và B đều mô tả các đối tượng là số nguyên. Hàm bạn của hai lớp A và B để hoán đổi dữ liệu của đối tượng lớp A và dữ liệu của đối tượng lớp B.

```
#include <iostream>
using namespace std;
class B;   //khai bao truoc lớp B
class A
{
  private:
      int m;
  public:
      void Nhap();
      void Xuat();
      friend void traodoi(A &ob1, B &ob2);
  };
```

```
class B
  {
      int n;
    public:
      void Nhap();
      void Xuat();
      friend void traodoi(A &ob1, B &ob2);
  };
void A::Nhap()
 {
   cout<<"\n Nhap m =";
   cin>>m;
 }
void A::Xuat()
 {
   cout<<"m = "<<m;
 }
```

```
void B::Nhap()
 {
   cout<<"\n Nhap n =";
   cin>>n;
 }
void B::Xuat()
 {
   cout<<" n = "<<n;
 }
void traodoi(A &x, B &y)  //không có tên lớp
 {
   int t = x.m;
   x.m = y.n;
   y.n = t;
 }
```

```
int main()
{
    A ob1;
    B ob2;
    ob1.Nhap();
    ob2.Nhap();
    cout<< "Gia tri ban dau :" <<"\n";
    ob1.Xuat();
    ob2.Xuat();
    traodoi(ob1, ob2); //goi ham ban nhu ham thong thuong
    cout<<"\nGia tri sau khi thay doi:"<<"\n";
    ob1.Xuat();
    ob2.Xuat();
    return 0;
  }
```

Thảo luận: Viết lại chương trình trên nhưng không dùng hàm bạn

Ví dụ 3.9: Hàm bạn lớp phanso

Ví dụ 3.10: Hàm bạn lớp số phức (các hàm bạn: tính tổng, tích 2 số phức, mo đun của số phức)

Ví dụ 3.9

```
//minh hoa dung ham ban de tinh tong va tich 2 phan so
#include<iostream>
using namespace std;

class phanso
 {
   private:
              int t,m; //tu so va mau so
   public:
          void nhap();
          void xuat();
          friend phanso tong(phanso p1, phanso p2);
          friend phanso tich(phanso p1, phanso p2);
};
 void phanso::nhap()
                {
                        cout<<"\nNhap tu so:";cin>>t;
                        do
                         {
                           cout<<"\nNhap mau so:";cin>>m;
                         } while(m==0);  //mau so bang 0 thi nhap lai
                 }
```

```
void phanso::xuat()
                {
                        cout<<t<<"/"<<m<<endl;
                }

phanso tong(phanso p1, phanso p2)
  {
    phanso p;  //p chua co du lieu
    p.t = p1.t*p2.m + p1.m*p2.t;   //   p1.t/p1.m + p2.t/p2.m
    p.m = p1.m*p2.m;
    return p;
  }

phanso tich(phanso p1, phanso p2)
  {
    phanso p;
    p.t = p1.t*p2.t;     //   p1.t/p1.m + p2.t/p2.m
    p.m = p1.m*p2.m;
    return p;
  }
```

```
int main()
  {
   phanso p1,p2,p3,p4;

   cout<<"\nNhap phan so thu nhat:";
   p1.nhap();

   cout<<"\nNhap phan so thu hai:";
   p2.nhap();

   cout<<"\nPhan so thu nhat:";p1.xuat();
   cout<<"\nPhan so thu hai:";p2.xuat();

  p3 = tong(p1,p2);  // goi ham ban tong
   cout<<"\nTong hai phan so:";
  p3.xuat();

   p4 = tich(p1,p2);  // goi ham ban tong
   cout<<"\nTich hai phan so:";
   p4.xuat();

   return 0;
  }
```

Thảo luận:

1. Phân biệt hàm bạn và hàm thành phần (phương thức)
2. Viết chương trình cho ví dụ 3.10
3. Xét lớp phanso, hãy viết chương trình để có thể thực hiện biểu thức sau:

p2=k+p1;

Trong đó k là một số nguyên, p1,p2 là các đối tượng của lớp phanso.

3.8. Thuộc tính tĩnh và phương thức tĩnh

Xem giáo trình!

3.9. Hàm tạo (constructor)

Hàm tạo là một hàm thành phần đặc biệt của lớp làm nhiệm vụ tạo lập một đối tượng mới. Chương trình dịch sẽ cấp phát bộ nhớ cho đối tượng, sau đó sẽ gọi đến hàm tạo. Hàm tạo sẽ khởi gán giá trị cho các thuộc tính của đối tượng và có thể thực hiện một số công việc khác nhằm chuẩn bị cho đối tượng mới.

Khi xây dựng hàm tạo cần lưu ý những đặc tính sau của hàm tạo:

- Tên hàm tạo trùng với tên của lớp.
- Hàm tạo không có kiểu trả về.
- Hàm tạo phải được khai báo trong vùng public.
- Hàm tạo có thể được xây dựng bên trong hoặc bên ngoài định nghĩa lớp.
- Hàm tạo có thể có đối số hoặc không có đối số.
- Trong một lớp có thể có nhiều hàm tạo (cùng tên nhưng khác các đối số).

Chú ý: Trong một lớp, nếu ta không định nghĩa hàm tạo thì hàm tạo mặc định được tự động sinh ra và không thực hiện việc khởi tạo giá trị thuộc tính của đối tượng.

Ví dụ:

```
class DIEM
{
  private:
     int x,y;
  public:
     DIEM()                    //Ham tao khong tham so
         {
           x = y = 0;
      }
     DIEM(int x1, int y1)  //Ham tao co tham so
      {
                    x = x1;y=y1;
      }
     //Cac thanh phan khac
};
```

Chú ý :

- Nếu lớp không có hàm tạo, chương trình dịch sẽ cung cấp một hàm tạo mặc định không đối, hàm này thực chất không làm gì cả. Như vậy một đối tượng tạo ra chỉ được cấp phát bộ nhớ, còn các thuộc tính của nó chưa được xác định.
- Khi một đối tượng được khai báo thì hàm tạo của lớp tương ứng sẽ tự động thực hiện và khởi gán giá trị cho các thuộc tính của đối tượng. Dựa vào các tham số trong khai báo mà chương trình dịch sẽ biết cần gọi đến hàm tạo nào.
- Nếu trong lớp đã có ít nhất một hàm tạo, thì hàm tạo mặc định sẽ không được phát sinh nữa. Khi đó mọi câu lệnh xây dựng đối tượng mới đều sẽ gọi đến một hàm tạo của lớp. Nếu không tìm thấy hàm tạo cần gọi thì chương trình dịch sẽ báo lỗi.

```
class DIEM
    {
      private:
          int x,y;
       public:
          DIEM()                    //Ham tao khong tham so
              {
                 x = y = 0;
           }
          DIEM(int x1, int y1)  //Ham tao co 2 tham so
           {
                          x = x1;y=y1;
           }
          //Cac thanh phan khac
    };
```

Với khai báo:

Diem ob1, ob2(5),ob3(4,9);

Thì dữ liệu của ob1 là x=y=0;

Dữ liệu của ob2 : Lỗi do chương trình không có hàm tạo 1 tham số.

Dữ liệu của ob3: x=4, y =9

Các dữ liệu này sau này có thể thay đổi.

Thảo luận: Hoàn chỉnh chương trình ở trên, xét các trường hợp có hàm tạo mặc định và không có hàm tạo mặc định.

3.10. Hàm hủy (destructor)

Hàm huỷ là một hàm thành phần của lớp, có chức năng ngược với hàm tạo. Hàm huỷ được gọi để giải phóng một đối tượng trước khi đối tượng được hủy bỏ.

Nếu trong lớp không định nghĩa hàm huỷ thì một hàm huỷ mặc định không làm gì cả được phát sinh. Đối với nhiều lớp thì hàm huỷ mặc định là đủ, không cần đưa vào một hàm huỷ mới.

Trường hợp cần xây dựng hàm hủy thì tuân theo quy tắc sau:

- Mỗi lớp chỉ có một hàm hủy.
- Hàm hủy không có kiểu, không có giá trị trả về và không có đối số.
- Tên hàm hàm huỷ có một dấu ngã (~) ngay trước tên lớp.
- Ví dụ: ~DIEM()
  { }

Ví dụ 3.22. Để quản lý các học phần của một trường đại học, ta dùng một danh sách liên kết đơn, mỗi nút của danh sách biểu diễn một đối tượng học phần và có các thông tin: mã học phần, tên học phần, số tín chỉ. Kiểu lớp dùng để mô tả đối tượng học phần được cho như sau:

```
class Hocphan
{
      private:
            char *mahp    // mã học phần
            char *tenhp;  // tên học phần
            int sotc;             // số tín chỉ
      public:
            Hocphan *next;
            // Các phương thức để khởi tạo giá trị
            // cho mahp, tenhp, sotc
      // Các phương thức để trả về giá trị
      // của mahp, tenhp, sotc
};
```

Cho khai báo kiểu lớp `List` sau đây, trong đó mỗi đối tượng của lớp `List` là một danh sách liên kết chứa các phần tử kiểu `Hocphan`:

```
class List
{
     private:
          Hocphan * first;
     public:
          //các phương thức thể hiện yêu cầu
};
```

Chương trình sau đây sẽ thực hiện các yêu cầu:

1. Định nghĩa các phương thức để khởi tạo giá trị cho `mahp`, `tenhp`, `sotc`; Các phương thức lấy giá trị của `mahp`, `tenhp`, `sotc` của lớp `Hocphan`.

2. Định nghĩa các phương thức của lớp `List`:

- Nhập dữ liệu cho đối tượng của lớp `List`, quá trình nhập dữ liệu kết thúc khi mã học phần nhập vào là rỗng.

- Sắp xếp các học phần theo thứ tự tăng của mã học phần.

- Tìm kiếm theo mã học phần.

- Xóa học phần theo mã học phần.

- Hiển thị thông tin về các học phần.

3. Hàm `main()` minh họa.

Bài tập

- Sinh viên làm hết các bài tập cuối chương 3 trong giáo trình!