3.3. Truy cập tới các thành phần của lớp

- Đối với đối tượng thông thường:

  Tên đối tượng. Tên thuộc tính

  Tên đối tượng. Tên phương thức(…)
- Đối với phần tử mảng đối tượng:

  phần tử mảng. Tên thuộc tính

  phần tử mảng. Tên phương thức(…)
- Đối với con trỏ đối tượng:

  tên con trỏ -> Tên thuộc tính

  tên con trỏ -> Tên phương thức(…)

Ví dụ 3.3. Chương trình sau định nghĩa lớp Diem để mô tả các đối tượng là điểm trong mặt phẳng tọa độ Oxy, đặc trưng bởi hoành độ và tung độ. Các phương thức nhập, xuất dữ liệu và minh họa chương trình trên hai đối tượng:

```
class Diem
    {
            private:  int x,y ;
            public :
               void nhap();
               void xuat();
   };
void Diem::nhap()
             {
                cout<< "\n Nhap hoanh do: ";      cin>>x;
                cout<< "\n Nhap tung do: ";       cin>>y;
             }
void Diem:: xuat()
            {
                 cout<<"\n x = " <<x
                        <<" y = "<<y;
            }
```

```
int main()
  {
    Diem d1,d2;
    d1.nhap();
    d2.nhap();

    d1.xuat();
    d2.xuat();

}
```

```
//doan ma lenh LOI
int main()
  {
    Diem d1,d2;
    d1.x =3;  //loi truy cap tu doi tuong d1 vao vung private
   d1.y = 5; //loi truy cap tu doi tuong d1 vao vung private

   d2.x = 4; //loi truy cap tu doi tuong d1 vao vung private

   d2.y = 6; //loi truy cap tu doi tuong d1 vao vung private



    d1.xuat();
    d2.xuat();
}
```

Chú ý:

1. Trong hàm main(), nếu đưa vào các câu lệnh :

d1.x = 5 ;

d1.y = 10 ;

Trình biên dịch sẽ báo báo lỗi đối các lệnh này, bởi vì các thuộc tính x, y thuộc vùng private nên không cho phép đối tượng d1 truy cập đến.

2. Các thuộc tính private chỉ có thể được xử lý bởi các phương thức của lớp. Các lớp thường cung cấp các phương thức public để cho phép thiết lập (set) (nghĩa là "ghi") hoặc lấy (get) (nghĩa là "đọc") các giá trị của các thuộc tính private. Các phương thức này thường được gọi là "set" hay "get". Ta xem các ví dụ sau:

Ví dụ 3.4. Chương trình sau minh họa việc nhập, xuất một mảng gồm không quá 50 đối tượng của lớp Sinhvien và hiển thị sinh viên có điểm trung bình lớn nhất.

//sinhvien.cpp

```
#include <iostream>
#include <iomanip>
using namespace std;

class Sinhvien
 {
   private:
            char masv[10],ht[30];
           float dtb;
   public:
            void nhap();
           void xuat();
           float get_dtb();  //Lấy ra dtb của vùng riêng
 };
```

```
void Sinhvien::nhap()
  {
    cout<<"\n Nhap ma sv :";
    cin.get(masv,10);
    cin.ignore();
    cout<<"\n Nhap ho ten sv :";
    cin.get(ht,30);
    cout<<"\n Diem trung binh :";
    cin>>dtb;
    cin.ignore();
  }
void Sinhvien::xuat()
  {
            cout<<setw(10)<<masv
                <<setw(30)<<ht
                <<setw(5)<<dtb<<endl;
  }
float Sinhvien::get_dtb()  //lấy điểm trung bình
 {    return dtb;  }
```

```
int main()
{
  Sinhvien sv[50];
  int i,n;
  cout<<"\n Nhap so sinh vien :";
  cin>>n;
  cin.ignore();
  for (i = 0;i<n;i++)
     sv[i].nhap();
  //Tim Max cua dtb
  float max = sv[0].get_dtb();
  for (i=1; i<n ; i++)
    if (max < sv[i].get_dtb())
              max = sv[i].get_dtb();
```

```
 //In sinh vien  co dtb dat max
  cout<<"\nDanh sach sinh vien co dtb "
        <<" cao nhat: "<<endl;
  cout<<setw(10)<<" Ma sv"
        <<setw(30)<<" Ho va ten"
        <<setw(20)<<" Diem trung binh"<<endl;
  for (i=0; i<n ; i++)
         if (sv[i].get_dtb() == max)
            sv[i].xuat();
  return 0;
}
```

Ví dụ 3.5. Xét lớp Sinhvien, chương trình sau minh họa hiển thị danh sách sinh viên có sắp xếp theo thứ tự của họ tên.

```
//sinhvien2.cpp
class Sinhvien
 {
   private:
            char masv[10],hoten[30];
       float dtb;
   public:
        void nhap();
        void xuat();
        char * get_hoten();  //lấy ra hoten
 };
```

```
void Sinhvien::nhap()
  {
    //
  }

void Sinhvien::xuat()
  {
     //…
  }
char * Sinhvien::get_hoten()
 {
    return hoten;
 }
```

```
int main()
{ Sinhvien sv[50];
  int i,n;
  cout<<"\n Nhap so sinh vien: ";
  cin>>n;
  for (i = 0;i<n;i++)
     sv[i].nhap();
  //sap xep theo masv
  for (int i=0; i<n-1 ; i++)
   for (int j = i+1; j<n ; j++)
      if (strcmp(sv[i].get_hoten(),sv[j].get_hoten())>0)
        {
          Sinhvien tam = sv[i];   sv[i]=sv[j];  sv[j] = tam;
        }
  cout<<"\nDanh sach sinh vien co sap xep theo masv: ";
  for (i=0; i<n ; i++)
          sv[i].xuat();
  return 0;
}
```

Thảo luận: Khai báo lại thuộc tính masv, hoten là các đối tượng của lớp **string**. Viết lại chương trình in danh sách sinh viên có sắp xếp theo thứ tự tăng của họ tên.

3.4. Con trỏ đối tượng

Con trỏ đối tượng dùng để chứa địa chỉ của biến đối tượng, được khai báo như sau :

Tên lớp * Tên con trỏ;

Chẳng hạn, với lớp Diem, ta có thể khai báo:

```
Diem *p1,*p2,*p3;  // Khai báo 3 con trỏ p1,p2,p3
Diem D1,D2;        // Khai báo hai đối tượng D1,D2
Diem d[20] ;       // Khai báo mảng đối tượng
```

Có thể thực hiện câu lệnh :

```
p1 = &D2 ; // p1 chứa địa chỉ của D2, p1 trỏ tới D2
p2 = d ;             // p2 trỏ tới đầu mảng d
p3 = new Diem    // cấp phát bộ nhớ cho p3
```

3.5. Con trỏ this

Ta hãy xem lại hàm nhap() của lớp Diem trong ví dụ trên:

```
void Diem::nhap()
{  cout<<"\n Nhap hoanh do:";
   cin>>x;
  cout<<"\n Nhap tung do:";
  cin>>y;
}
```

Trong hàm này ta sử dụng tên các thuộc tính x, y một cách đơn độc. Điều này dường như mâu thuẩn với quy tắc sử dụng thuộc tính. Tuy nhiên điều này được lý giải như sau:

- C++ sử dụng một con trỏ đặc biệt có tên là **this, nó là tham số đầu tiên của** các phương thức. Khi một đối tượng gọi đến phương thức thì tham số truyền cho con trỏ this chính là địa chỉ của đối tượng đó, vì vậy con trỏ this sẽ trỏ tới đối tượng đó. Ví dụ: d1.nhap(); thì địa chỉ của d1 sẽ truyền cho con trỏ this của hàm nhap().
- Các thuộc tính viết trong phương thức được hiểu là thuộc một đối tượng do con trỏ *this* trỏ tới. Như vậy, với lời gọi trên thì các thuộc tính x, y trong hàm nhập chính là: this->x, this->y

Cụ thể:

Với lời gọi tới hàm nhap() từ đối tượng d1:

d1.nhap();

Trong trường hợp này thì this =&d1. Do đó this->x chính là d1.x và this->y chính là d1.y

Lúc đó hàm nhập sẽ được viết lại:

```
void Diem::nhap()
{
  cout<<"\n Nhap hoanh do:";
   cin>>this->x;
  cout<<"\n Nhap tung do:";
  cin>>this->y;
}
```

Như vậy: Trong mọi phương thức của lớp, this là tham số con trỏ đầu tiên của phương thức nhưng ngầm định không viết ra. Các thuộc tính cũng không cần ghi thêm tên con trỏ this.

Ví dụ: Xét lớp Diem, hãy định nghĩa thêm các phương thức: Tính tổng toạ độ của 2 điểm, tìm khoảng cách giữa 2 điểm, tìm điểm đối xứng qua gốc toạ độ. Viết chương trình minh hoạ.

Chú ý:

Công thức tính khoảng cách A (x1,y1) và B (x2,y2):

Kc(A,B) = $\sqrt{(x1-x2)^2+(y1-y2)^2}$

```
#include <iostream>
#include <iomanip>
#include <math.h>
```

d1(x,y)

.

d2(x,y)

.

```
using namespace std;

class Diem
{
        private:
            int x,y ;
        public :
            void nhap();
            void xuat();
            Diem tong(Diem d2);  //tinh tong 2 doi tuong: dt tro boi this và dt d2
            float kc(Diem d2);      //tinh kc tu dt tro boi this den dt d2
            Diem doixung();       //doi xung voi doi tuong ma this tro toi
};
```

```
void Diem::nhap()
{
  cout<<"\n Nhap hoanh do:";
   cin>>this->x;
  cout<<"\n Nhap tung do:";
  cin>>this->y;
}
void Diem::xuat()
  { cout<<"\n x = " <<x<<" y = " <<y<<endl;
  }

Diem Diem::tong(Diem d2)
   { Diem d;
     d.x = x + d2.x;
     d.y = y + d2.y;
     return d;
   }
float Diem::kc(Diem d2)
 {
            return sqrt(pow(x-d2.x,2)+pow(y-d2.y,2));
 }
```

```
Diem Diem::doixung()
{
    Diem d;
    d.x = -x;
    d.y = -y;
    return d;
}
int main()
  { Diem d1,d2,d3,d4;
    d1.nhap(); d2.nhap();
    d3 = d1.tong(d2);  //tong cua d1 va d2
    cout<<"\nTong toa do la : ";
    d3.xuat();  //khong the viet cout<<d3
    cout<<fixed<<setprecision(2);
   cout<<"\n Khoang cach giua d1 va d2 la: “
         << d1.kc(d2);   //this tro toi d1, tinh khoang cach tu d1 den d2
    cout<<"\n Khoang cach giua d2 va d1 la: “
          <<d2.kc(d1);   //this tro toi d2, khoang cach tu d2 den d1
    d4=d1.doixung();  //this tro toi d1, lay doi xung cua d1
   cout<<"\n Doi xung cua d1 la”; d4.xuat();
   return 0;
  }
```

Thảo luận: Có nhận xét gì khi viết hàm tính khoảng cách như sau:

```
float Diem::kc(Diem d1, Diem d2)
  {
    return sqrt(pow(d2.x - d1.x,2) + pow(d2.y -d1.y,2));
   }
```

Cách viết này đúng nhưng bị "thừa" 1 tham số.
Giả sử có khai báo Diem d,d1,d2 và đã nhập dữ liệu cho d1, d2:
Lời gọi hàm:

```
 k = d.kc(d1,d2);
```

Tính được kc từ d1 đến d2
Chú ý: Không nên viết như thế này.

3.6. PHÉP GÁN TRÊN ĐỐI TƯỢNG

Phép gán = có thể được sử dụng để gán một đối tượng cho một đối tượng khác của cùng một kiểu. Toán tử gán như thế bình thường được thực hiện bởi toán tử sao chép thành phần – mỗi thành phần của một đối tượng được sao chép riêng tới cùng thành phần ở đối tượng khác.

Việc sao chép thành phần có thể phát sinh lỗi khi sử dụng với một lớp mà thành phần dữ liệu chứa vùng nhớ cấp phát động. Trong trường hợp này, ta cần phải quá tải toán tử gán =, điều này sẽ xét đến trong Chương 4.

Các ví dụ

Ví dụ 3.6: Lớp phân số //lop phanso.cpp

Ví dụ 3.7: Lớp số phức

Ví dụ 3.8: Lớp thời gian

Ví dụ 3.6

```
//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();
            phanso cong(phanso p2);
            phanso tich(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 phanso::cong(phanso p2)
   {
    phanso p;
    p.t = t*p2.m + m*p2.t;    //  t/m + p2.t/p2.m
    p.m = m*p2.m;
    return p;  //tra ve doi tuong p
   }
```

$\frac{t}{m} + \frac{p2.t}{p2.m}$

```
//tuong tu cho hieu 2 phan so

phanso phanso::tich(phanso p2)
  {
    phanso p;
    p.t = t*p2.t;
    p.m = m*p2.m;
    return p;
  }
//tuong tu cho chia 2 phan so
int main()
  {
   phanso p1,p2,p3,p4;

   cout<<"\nNhap phan so thu nhat:";
   p1.nhap();  //con tro this tro vao p1

   cout<<"\nNhap phan so thu hai:";
   p2.nhap();  //con tro this tro vao p2

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

$\frac{t}{m}$ X $\frac{p2.t}{p2.m}$

```
p3 = p1.cong(p2); //cong p1 voi p2, con tro this tro vao p1
cout<<"\nTong hai phan so:";
p3.xuat();

p4 = p1.tich(p2); //tich p1 voi p2, con tro this tro vao p1
cout<<"\nTich hai phan so p1 va p2:";
p4.xuat();
return 0;
}
```

Thảo luận:

1. **Hãy tối giản kết quả của các phương thức cong, tich.**
2. **Hãy viết chương trình để thực hiện được biểu thức sau:**

p2 = k + p1;

p3 = p1 + k;

Trong đó k là số nguyên và p1,p2,p3 là các đối tượng phân số.

Nhắc lại: Để tìm ước chung lớn nhất của 2 số nguyên dương a, b ta thường dùng thuật toán Euclide:

$$\text{UCLN(a,b)} = \begin{cases} a \text{ nếu } a = b \\ UCLN(a-b, b) \text{nếu } a > b \\ UCLN(a, b-a) \text{nếu } a < b \end{cases}$$

Để tối giản một phân số: Chia cả tử và mẫu cho UCLN của tử và mẫu.

//Ví dụ 3.7

```
#include <iostream>
#include <math.h>
using namespace std;
class Sophuc
{
 private:
   float a,b;  //a la phan thuc va b la phan ao cua so phuc a + bi, i goi la so ao,
               //dinh nghia i^2 = -1
 public:
   void nhap();
   void xuat();
   Sophuc tong(Sophuc c2);
   Sophuc tich(Sophuc c2);
   float  modun();
};
void Sophuc::nhap()
{
      cout <<"\n a = "; cin >> a;
      cout <<"\n b = "; cin >> b;
    }
```

```
void Sophuc::xuat()
  { if (b>=0)
        cout<<a<<" + "<<b<<"i"<<endl;
    else
        cout<<a<<b<<"i"<<endl;
  }
Sophuc Sophuc::tong(Sophuc c2)
  {
    Sophuc c;                    //(a+bi) + (c2.a+c2.bi) = a+c2.a + (b+c2.b)i
    c.a= a + c2.a ;
    c.b= b + c2.b ;
    return c;
  }
Sophuc Sophuc::tich(Sophuc c2)
  {
    Sophuc c;
    c.a= a*c2.a - b*c2.b ;  //(a+bi)*(c2.a+c2.bi) = (a*c2.a- b*c2.b) +
                            //(a*c2.b + b*c2.a)i
    c.b= a*c2.b + b*c2.a ;
   return c;
 }
```

```
float  Sophuc::modun()
    { return sqrt(a*a + b*b);   //mo dun cua so phuc a + bi
}
int main()
{
  Sophuc c1, c2, c3, c4;
  c1.nhap();
  c2.nhap();
  c3 = c1.tong(c2);  //tong cua c1 và c2 va gan cho c3
  cout<<"c1 = ";
  c1.xuat();
  cout<<"c2 = ";
  c2.xuat();
  cout<<"\nTong cua c1 va c2:"<<endl;
  cout<<"c3 = ";
  c3.xuat();
  cout<<"mo dun cua c1 = "<<c1.modun()<<endl;
  cout<<"mo dun cua c2 = "<<c2.modun()<<endl;
  c4=c1.tich(c2);  //tich cua c1 va c2 va gan cho c4
  cout<<"Tich cua c1 va c2 = ";
  c4.xuat();
  return 0;
  }
```

Thảo luận: Viết thêm các phương thức tìm hiệu và thương của 2 số phức.