Bài 3. Cơ bản về lớp trong C++

I. Lập trình hướng thủ tục và hướng đối tượng

- Cả hai cách tiếp cận đều thực hiện theo phương pháp tinh chỉnh từng bước (stepwise refinement)
- Tiếp cận hướng thủ tục (function oriented):
  - Tập chung vào các hàm và việc phân rã các hàm
  - Các cấu trúc dữ liệu (ở mức ngôn ngữ lập trình) được định nghĩa sớm.
  - Các cấu trúc dữ liệu khó có thể thay đổi
- Tiếp cận hướng đối tượng (Object Oriented)
  - Tập chung vào các đối tượng trừu tượng
  - Các cấu trúc dữ liệu trừu tượng được định nghĩa sớm
  - Cấu trúc dữ liệu chi tiết mức ngôn ngữ chưa được định nghĩa
  - Cấu trúc dữ liệu dễ thay đổi hơn

Ví dụ

- Bài toán: Lập chương trình nhập vào tọa độ các đỉnh của 1 tam giác bất kỳ trong mặt phẳng. Tính diện tích và chu vi của tam giác đó. In kết quả lên màn hình

Tiếp cận hướng thủ tục

- Xây dựng các hàm
  - Định nghĩa cấu trúc dữ liệu biểu diễn một tam giác
  - Nhập dữ liệu
  - Tính diện tích
  - Tính chu vi
  - Xây dựng hàm main() sử dụng các hàm ở trên

Định nghĩa cấu trúc dữ liệu và các hàm

```
typedef struct Tamgiac
{
    float xA, yA, xB,yB, xC, yC;
}
void Nhap(Tamgiac &t)
{
    cout<<“Nhap toa do dinh thu nhat:”;
    cin>>t.xA>>t.yA;
    cout<<“Nhap toa do dinh thu hai:”;
    cin>>t.xB>>t.yB;
    cout<<“Nhap toa do dinh thu ba:”;
    cin>>t.xC>>t.yC;
}
```

Tiếp cận hướng đối tượng

- **Xây dựng lớp tam giác**

```
class Tamgiac
{
   private:
         float xA, yA, xB,yB, xC, yC;
   public:
    void Nhap();
    float Dientich();
    float Chuvi();
};
```

II. Khái niệm lớp - Khai báo lớp

- Lớp là một khái niệm mở rộng của cấu trúc dữ liệu, nó có thể chứa đựng cả dữ liệu và các hàm
- Đối tượng (object) là một thể hiện của lớp. Trong lập trình lớp được xem như là một kiểu dữ liệu, đối tượng là các biến

```
class class_name  {
 access_specifier_1:
            member1;
access_specifier_2:
            member2;
 ...
} ;
```

- class_name : Tên lớp cần tạo
- access_specifier : là các đặc tả truy nhập (private, protected, public)

- member : khai báo các thành phần của lớp (có thể là thuộc tính hoặc các hàm thành viên)

Ví dụ: Khai báo lớp biểu diễn các hình chữ nhật phương thức đặt giá trị cho các thuộc tính và phương thức tính diện tích

```
class CRectangle
{
        int width, height;
        public:
            void set_values (int,int);
            int area (void);
 } ;
```

Ví dụ: Khai báo lớp biểu diễn các ma trận với các phương thức đặt số hàng, số cột, nhập các phần tử và in các phần tử

```
class CMatrix{
    private:
        int rows, cols;
        float *element;
    public:
        void setColRow(int,int)
        void printMatrix();
        void inputMatrix();
    };
```

III. Cài đặt các phương thức

- Ta có thể cài đặt các phương thức bên trong lớp hoặc bên ngoài lớp.

Cài đặt phương thức bên ngoài lớp

```
DataType class_Name::Func_Name([Argument_list]){
        Các câu lệnh;
 }
```

Ví dụ

```
class CRectangle {
    int width, height;
    public:
     void  set_values (int a,int b);
    int area () {
            return (width*height);
    }
};
```

```
void CRectangle:: set_values (int a, int b)
{
            width = a;
            height = b;
}
```

Chương trình hoàn thiện

```
#include <iostream.h>
#include <conio.h>
class CRectangle{
 int width, height;
public:
   void set_values (int,int);
   int area () {return width*height);}
 };
 void CRectangle::set_values (int a, int b) {width = a; height = b; }
 int main () {
 CRectangle rect;
 rect.set_values (3, 4);
 cout << "area: " << rect.area();
 getch();   return 0;
}
```

Ví dụ: Xây dựng phương thức nhập ma trận

```
void CMatrix:: inputMatrix(){
      int i,j;
      if(element != NULL) delete []element;
         element = new float[rows*cols];
          for(i=0; i<rows; i++)
              for(j=0; j<cols; j++){
                 cout<<“element[“<<i<<“][“<<j<<“]=“;
                 cin>>element[i*cols+j];
          }
}
```

IV. Truy cập đến các thành phần của lớp

- **Biến đối tượng**
  - Khai báo: classname objname;
  - Truy nhập:
    - objname.Property //Truy nhập thuộc tính của lớp
    - objname.Method([arg]) // Truy nhập các phương thức
  - Ví dụ: CRectangle rect;

    rect.width

    rect.set_values (3, 4);

Chú ý: Chỉ được phép truy nhập các thành phần trong vùng private trong các phương thức của lớp

- **Con trỏ đối tượng**
  - Khai báo: classname *pointername;
  - Trước khi sử dụng con trỏ để lưu trữ dữ liệu ta cần gán địa chỉ của một đối tượng đã có cho nó hoặc cấp phát bộ nhớ cho nó.
  - Truy nhập:
    - pointername→properties
    - pointername→method([arg])
  - Ví dụ:

```
CRectangle *rect;
rect = new CRectangle(); //cấp bộ nhớ
rect→width
rect→set_values (3, 4);
```

V. Cấu tử - Hủy tử

- Các đối tượng khi được tạo ra thì cần được gán giá trị cho các thuộc tính của nó để tránh gặp phải những giá trị không mong muốn trong quá trình xử lý.
- Trong ví dụ trên nếu ta không gọi rect.set_values (3,4); mà gọi ngay rect.area(); thì diện tích của hình chữ nhật là bao nhiêu?
- Để tránh được điều đó trong lớp cần xây dựng các hàm đặc biệt để khởi tạo giá trị cho các thuộc tính của đối tượng khi tạo ra các đối tượng - Ta gọi các hàm đó là các cấu tử hay hàm khởi tạo (constructor)
- Trong một lớp có thể nạp chồng nhiều cấu tử.

• Hủy tử là các hàm đặc biệt trong lớp nó được tự động gọi tới khi cần hủy bỏ đối tượng khỏi bộ nhớ

• Trong lớp chỉ xây dựng hủy tử nếu nó có thuộc tính được cấp phát bộ nhớ động (có thuộc tính con trỏ)

•Trong một lớp chỉ xây dựng 1 hủy tử

Tạo các cấu tử và hủy tử

```
class class_Name{
        private:
                khai báo các thuộc tính, phương thức riêng;
        public:
                class_Name(); //cấu tử không đối
                class_Name(arg_list); //cấu tử có đối
                 ~class_Name();  //hủy tử
                khai báo các thuộc tính và phương thức công khai
};
```

❖ **Cài đặt các cấu tử:** Các câu lệnh trong các cấu tử thực hiện khởi gán giá trị, cấp phát bộ nhớ cho các thuộc tính của lớp.

❖**Cài đặt hủy tử:** Trong thân của hủy tử ta thực hiện các lệnh xóa bỏ các thuộc tính con trỏ.

Ví dụ: xây dựng lớp hình chữ nhật

```
#include <iostream.h>
#include <conio.h>
class CRectangle {
int width, height;
public:
CRectangle();
CRectangle (int,int);
 int area () { return (width*height);}
};

CRectangle::CRectangle ()
 {  width =0;   height = 0;  }

CRectangle::CRectangle (int a, int b)
 {  width = a;   height = b;  }
```

```
int main () {
   CRectangle r;
   CRectangle rect (3,4);
   CRectangle rectb (5, 6);
   cout << "rect area: " << rect.area();
   cout << "rectb area: " << rectb.area() ;
   getch();
    return 0;
}
```

Cấu tử không đối

Cấu tử có đối đầy đủ

Ví dụ: Xây dựng lớp ma trận

```
#include <iostream.h>
#include <conio.h>
class CMatrix{
    private:
        int rows, cols;
        float *element;
    public:
        CMatrix();
        CMatrix(int, int);
        ~CMatrix();
        void setColRow(int,int)
        void printMatrix();
        void inputMatrix();
 };
Cmatrix::CMatrix()
{
    rows = 0;   cols = 0;
    element = NULL;
}
Cmatrix::CMatrix(int row, int col)
{
    rows = row;   cols = col;
    element = new float [rows*cols];
}
Cmatrix::~Cmatrix(){
   delete  [ ] element;
}
void CMatrix:: inputMatrix(){
    int i,j;
    if(element != NULL) delete element;
    element = new float[rows*cols];
    for(i=0; i<rows; i++)
       for(j=0; j<cols; j++){
          cout<<"element["<<i<<"]["<<j<<"]=";
          cin>>element[i*cols+j];
       }
}
```

```
void CMatrix:: printMatrix(){
      int i, j;
      for(i=0; i<rows; i++)
      {
          cout<<"\n";
         for(j=0; j<cols; j++){
             cout<<element[i*cols+j]<<"   " ;
       }
 }
void CMatrix:: setColRow(int r,int c)
{
    rows = r;
    cols  = c;
}
```

```
void main(){
      CMatrix  m(3, 3);
      m.inputMatrix();
      m.printMatrix();
      getch();
}
```

Tạo ma trân vuông 3*3

Bài tập

1. Xây dựng lớp biểu diễn các điểm trong mặt phẳng với một cấu tử không đối, một cấu tử có đối đầy đủ, hai phương thức nhập và in tọa độ của điểm lên màn hình.
2. Xây dựng lớp biểu diễn một đoạn thẳng (biết đoạn thẳng được xác định bởi tọa độ điểm đầu và điểm cuối). Với các cấu tử không đối, có đối đầy đủ, phương thức nhập, in tọa độ hai đầu mút, tính độ dài đoạn thẳng.
3. Xây dựng lớp biểu diễn các thí sinh, biết mỗi thi sinh bao gồm các thông tin: Số báo danh, Họ tên, năm sinh, giới tính, điểm toán, điểm lý, điểm hóa. Lớp có các cấu tử, các phương thức nhập, in, lấy tổng điểm, lấy điểm từng môn
4. Xây dựng lớp biểu diễn đối tượng thời gian (time). Với các hàm tạo, các phương thức nhập in, phương thức lấy các thuộc tính, phương thức đặt giá trị cho từng thuộc tính

Bài tập (tt)

4. Xây dựng lớp biểu diễn các đối tượng dãy số với các phương thức hàm tạo, hàm in, hàm thêm một phần tử vào dãy, hàm xóa một phần tử của dãy, hàm tìm kiếm một phần tử có trong dãy không nếu có trả lại vị trí của phần tử đó trong dãy.
5. Xây dựng lớp biểu diễn các đối tượng là các sinh viên (các thuộc tính, phương thức do sv tự xác định)

VI. Lớp mẫu (template class)

- Khi định nghĩa các lớp mà kiểu dữ liệu của một số thuộc tính chưa được xác định thì khi đó ta định nghĩa các lớp này là các lớp mẫu, các thuộc tính chưa được xác định kiểu có kiểu mẫu.
- Ví dụ: định nghĩa một lớp mô tả quan hệ của hai số bất kỳ với các phương thức hàm tạo, phương thức lấy giá trị lớn nhất của chúng.

```
#include"conio.h"
#include <iostream>
template <class T, class Y>
class CPair {
        T a;
        Y b;
    public:
    CPair(){ }
    CPair (T a1, Y b1){
        a=a1; b=b1;}
    T getmax ();
};
```

```
template <class T, class Y >
T CPair<T,Y>::getmax () {
    T retval;
     retval = a>b? a : b;
     return retval;
}

int main () {
    CPair<float, int> p1 (100, 75);
    CPair<float,float> p2(10.2,10.5) ;
    cout << p1.getmax();
    cout<<"\n"<<p2.getmax();
    getch();
    return 0;
}
```

Ví dụ lớp mẫu biểu diễn các ma trận

```
template< class T>
class CMatrix{
    private:
        int rows, cols;
         T *element;
    public:
        void setColRow(int,int)
        void printMatrix();
        void inputMatrix();
    };
```

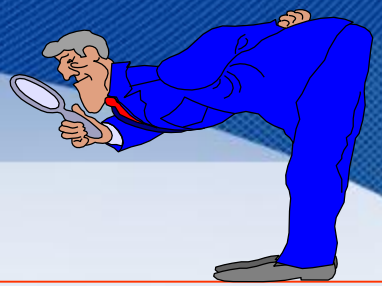
```
void main(){
 CMatrix<float> a;
}
```

Break

Cơ bản về lớp trong C++ (tiếp theo)

Con trỏ this, phương thức của lớp là các toán tử,

I. Con trỏ this



Xét lại ví dụ Tam giác

```
void Nhap(Tamgiac &t)
{
   cout<<“Nhap toa do dinh thu nhat:”;
   cin>>t.xA>>t.yA;
   cout<<“Nhap toa do dinh thu hai:”;
   cin>>t.xB>>t.yB;
   cout<<“Nhap toa do dinh thu ba:”;
   cin>>t.xC>>t.yC;
}
```

```
void main(){
     Tamgiac t, t1;
      Nhap(t);
      Nhap(t1);
}
```

```
void Tamgiac::Nhap()
{
  cout<<“Nhap toa do dinh thu nhat:”;
   cin>>xA>>yA;
   cout<<“Nhap toa do dinh thu hai:”;
   cin>>xB>>yB;
   cout<<“Nhap toa do dinh thu ba:”;
   cin>>xC>>yC;
}
```

```
void main() {
     Tamgiac t, t1;
      t.Nhap();
      t1.Nhap();
}
```

- Tất cả các phương thức của lớp đều có một đối ẩn là một con trỏ (this) có kiểu là kiểu lớp chứa phương thức đó.

Ví dụ:

```
class A {
    private:
        …
    public:
        DataType method();
}
```

Chú ý: Con trỏ this là đối mặc định do đó người lập trình không cần khai báo

Truy nhập đến các thành phần của lớp từ con trỏ this

cách 1: this->property, this->method([arg]);

cách 2: property, method([arg]);

Ví dụ

```
void Tamgiac::Nhap()
{
  cout<<“Nhap toa do dinh thu nhat:”;
   cin>>xA>>yA;
   cout<<“Nhap toa do dinh thu hai:”;
   cin>>xB>>yB;
   cout<<“Nhap toa do dinh thu ba:”;
   cin>>xC>>yC;
}
```

```
void Tamgiac::Nhap()
{
  cout<<“Nhap toa do dinh thu nhat:”;
   cin>>this->xA>>this->yA;
   cout<<“Nhap toa do dinh thu hai:”;
   cin>> this-> xB>> this-> yB;
   cout<<“Nhap toa do dinh thu ba:”;
   cin>> this-> xC>> this->yC;
}
```

```
void main(){
     Tamgiac t, t1;
      t.Nhap();
      t1.Nhap();
}
```

```
void main() {
     Tamgiac t, t1;
      t.Nhap();
      t1.Nhap();
}
```

II. Phương thức của lớp là các toán tử

- Các nhóm toán tử:
  - Toán tử một ngôi: --, ++, -
  - Toán tử hai ngôi: +, -, *, /, …..
  - Toán tử so sánh: >, >=, <, <=, !, !=, …
  - Toán tử nhập xuất (>>, <<)
  - Toán tử new

II. Cài đặt các phương thức toán tử

- Toán tử một ngôi

```
class classname{
    private:
        …
    public:
        DataType operator sign();
        DataType operator sign();
        …
};
```

sign: là dấu toán (ví dụ: ++, --)

- Toán tử hai ngôi

```
class classname{
    private:
        …
    public:
        DataType operator sign(DataType1 argN);
        DataType operator sign(DataType1 argN);
        …
};
```

DataType là kiểu trả lại của toán tử

DataType1 là kiểu của đối tượng mà toán tử tác động

sign: là dấu toán (ví dụ: +, -, *, /, …)

Toán tử nhập – xuất

- **Toán tử nhập**

```
istream& operator >>(istream &is, classname &obj)
{
    cout<<“Thông báo:”;
    is>>obj.property;
   …
    return is;
}
```

- **Toán tử xuất**

```
ostream& operator <<(ostream &os, classname &obj)
{
    os<< “Thông báo:” << obj.property;
   …
    return os;
}
```

Ví dụ:Xây dựng lớp phân số

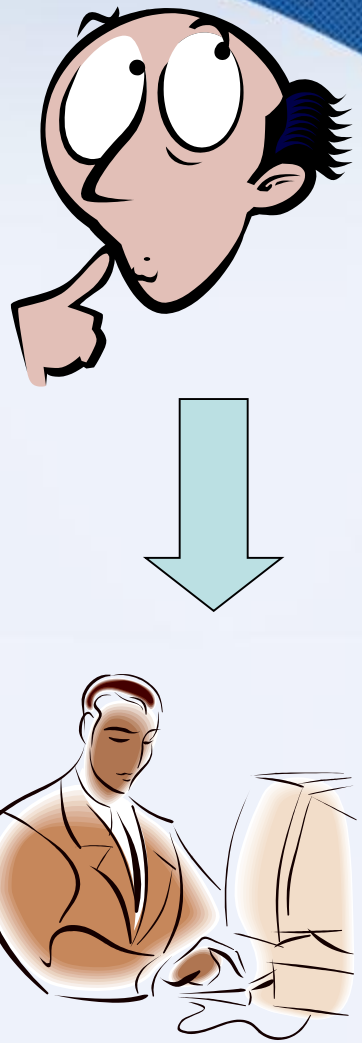
- **Phân tích bài toán**
  - Xây dựng hàm tìm uscln của hai số nguyên dương bất kỳ
  - Xây dựng lớp
    - Thuộc tính:
      - Tử, mẫu
    - Phương thức
      - Hàm tạo
      - Rút gọn phân số
      - Phương thức toán tử: >>, <<, +, -, *, /, -- (giảm), ++ (tăng), - (đổi dấu)
      - Phép toán so sánh
- **Lập trình**

Lớp phân số

```
class PS{
        private:
                long tu, mau;
                PS rutgon();
        public:
                friend ostream & operator<<(ostream &os, PS p);
                friend istream & operator>>(istream &is, PS &p);
                //Toan tu mot ngoi
                PS operator ++();
                PS operator --();
                PS operator -();
                //Toan tu hai ngoi
                PS operator+(PS p);
                PS operator -(PS p);
                PS operator*(PS p);
                PS operator /(PS p);
                //Cac phuong thuc so sanh
                int operator ==(PS p);
                int operator >(PS p);
};
```

Bài tập

1. Xây dựng lớp biểu diễn các vector trong không gian n chiều có các phương thức toán tử: +, - hai vector, * tích vô hướng hai véc tơ,-(đổi dấu) >>, <<.
2. Xây dựng lớp biểu diễn các đa thức với các phương thức toán tử: +, -, * hai đa thức, tính giá trị đa thức, >>, <<.
3. Xây dựng lớp biểu diễn các ma trận có các phương thức toán tử: +, - ,* hai ma trận, >>, <<.



Dẫn xuất và Thừa kế

- Về nhà tự nghiên cứu
  - Lớp dẫn xuất
  - Lớp cơ sở
  - Phương thức ảo
  - Phương thức thuần ảo
  - Lớp cơ sở ảo
  - Phương thức friend