Bài 3. Cơ bản về lập trình hướng đối tượng trong C++

I. Lập trình hướng chức năng và hướng đối tượng

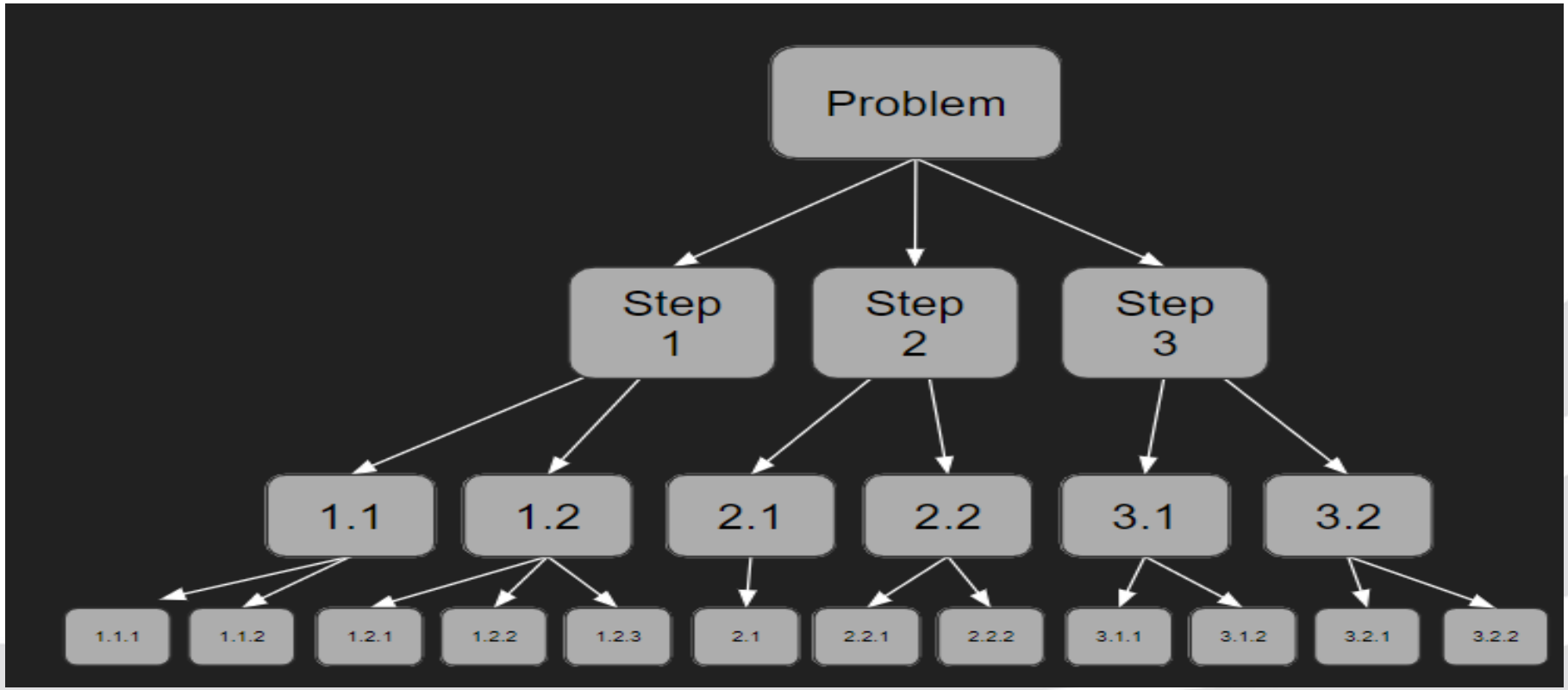
I. Lập trình hướng chức năng và hướng đối tượng
II. Khái niệm lớp, đối tượng
III. Cài đặt các phương thức của lớp
IV. Truy cập đến các thành phần của lớp
V. Cấu tử (constructor), Hủy tử (destructor)
VI. Lớp mẫu (template class)
VII. Bài tập

Một số phương pháp lập trình chính hiện nay

1. Lập trình tuyến tính
2. Lập trình logic
3. Lập trình hàm
4. Lập trình hướng cấu trúc
5. Lập trình hướng đối tượng

I. Lập trình hướng chức năng và hướng đối tượng (tt)

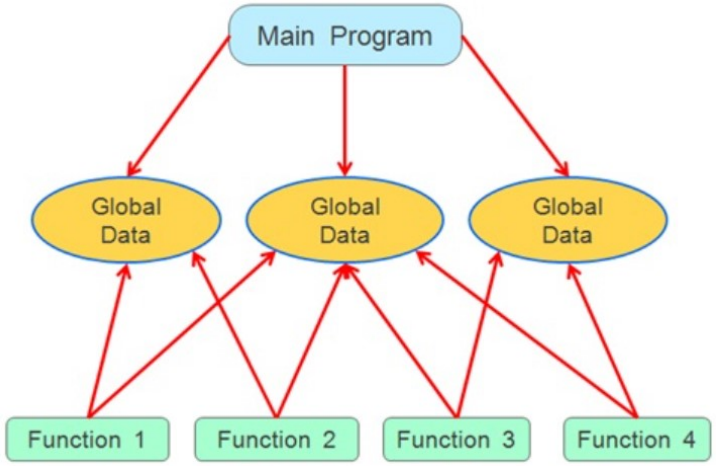
- Cả hai cách tiếp cận đều thực hiện theo phương pháp tinh chỉnh từng bước (stepwise refinement)



I. Lập trình hướng chức năng và hướng đối tượng (tt)

❑ Tiếp cận hướng chức năng (functional oriented):

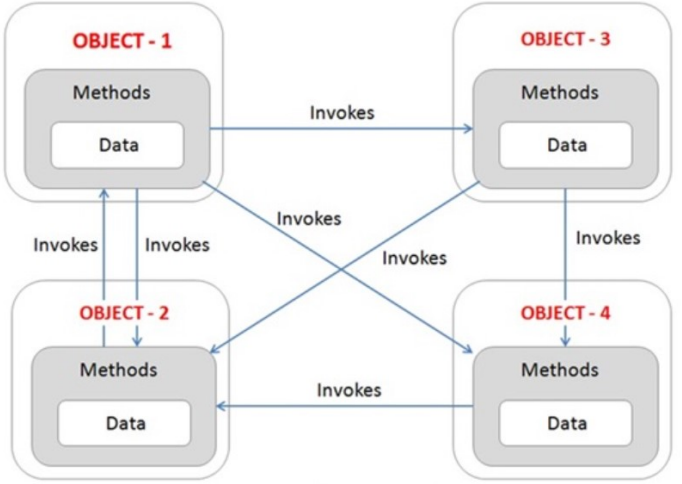
- Tập chung vào việc phân rã các hàm
- Định nghĩa các cấu trúc dữ liệu



- Ưu điểm: Chương trình dễ hiểu, dễ theo dõi, tư duy giải thuật rõ ràng.
- Nhược điểm: Các cấu trúc dữ liệu khó có thể thay đổi, dữ liệu trao đổi thông qua biến toàn cục, khó sử dụng lại được code

❑ Tiếp cận hướng đối tượng (Object Oriented)

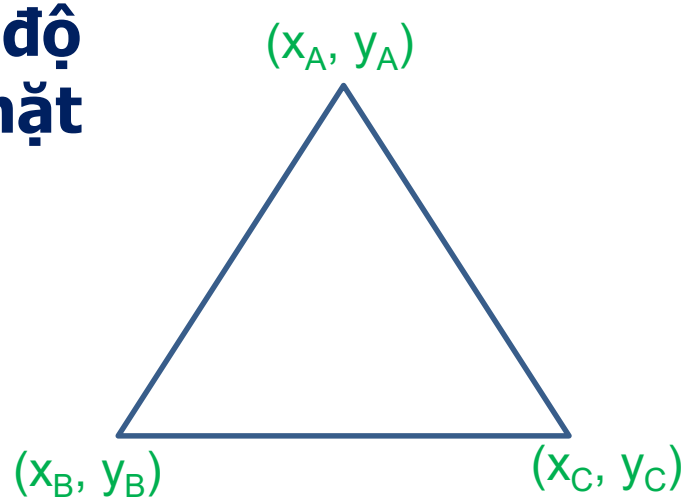
- Tập chung vào các đối tượng cần xử lý
- Xây dựng các lớp mô tả các đối tượng



- Ưu điểm: Khả năng thừa kế và sử dụng lại, dễ dàng sửa đổi cấu trúc dữ liệu mà không ảnh hưởng đến cấu trúc trước đó
- Nhược điểm: Chương trình khó hiểu hơn, đòi hỏi tư duy trừu tượng cao

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 của tam giác
  - Tính chu vi của tam giác
  - In kết quả lên màn hình



Tiếp cận hướng chức năng

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

```
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;
}
```

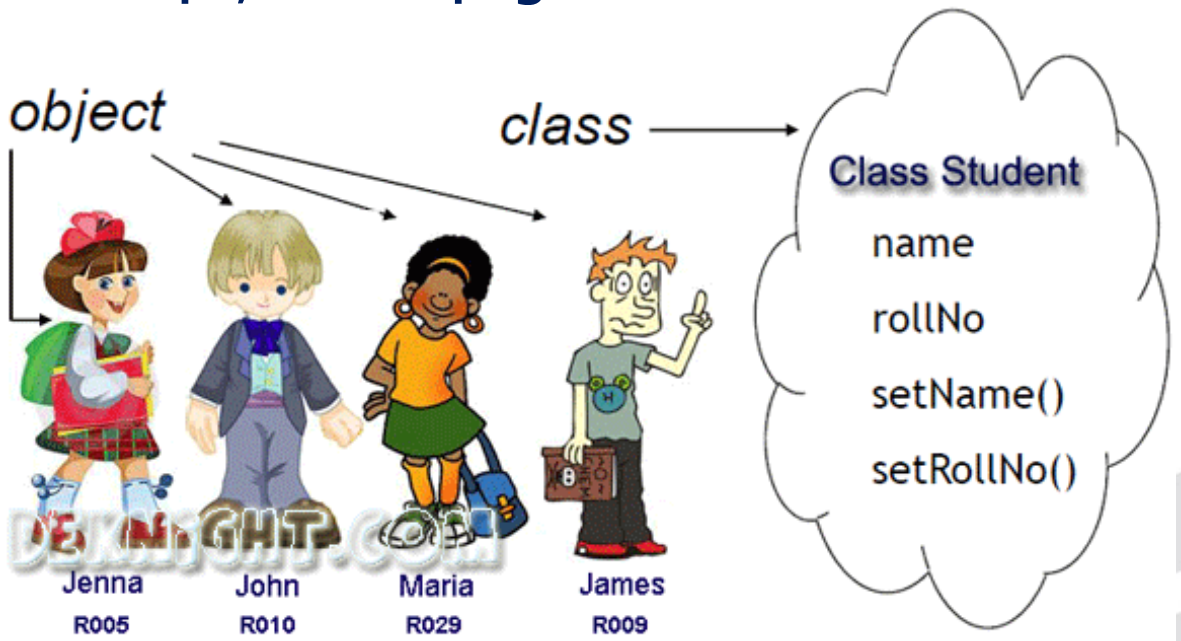
Tiếp cận hướng đối tượng

— **Định nghĩa lớp biểu diễn các đối tượng tam giác**

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

II. Khái niệm lớp, đối tượng

- **Lớp (class)** 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 cụ thể của lớp. Trong lập trình lớp được xem như là một kiểu dữ liệu, đối tượng là các biến
- **Ví dụ:**



II. Khái niệm lớp, đối tượng

- **Thuộc tính (property/attribute)** là các thành phần dữ liệu mô tả các đối tượng

  DataType Tên_thuộc_tính;

- **Phương thức (method)** là các hàm mô tả các hành động của đối tượng

  DataType Tên_phương_thức(Danh sách đối);

- **Ví dụ:**

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

II. Khái niệm lớp, đối tượng (tt)

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
{
        private:
            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 *a;
    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.**
- **Nếu cài đặt phương thức bên ngoài lớp thì viết như sau:**

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

Ví dụ:

```
class CRectangle {
    private:
        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 <bits/stdc++.h>
using namespace std;
class CRectangle{
private:
    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;
}
```

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);

❑ Con trỏ đối tượng

- Khai báo:

  classname *pointername;

- **Lưu ý:** Trước khi sử dụng con trỏ để lưu trữ DL cần cấp phát bộ nhớ cho con trỏ.
- Truy nhập:
  - Pointername->properties
  - Pointername->method([arg])
- Ví dụ:

```
CRectangle *rect;
rect = new CRectangle(); //Cấp bộ nhớ
int x = rect->width;
rect->set_values (3, 4);
```

V. Cấu tử (constructor), Hủy tử (destructor)

- **Cấu tử (constructor)** là các phương thức đặc biệt để thực hiện các lệnh khi đối tượng được sinh ra
- Trong một lớp có thể nạp chồng nhiều cấu tử.

- Hủy tử (destructor) là các hàm đặc biệt để thực thi các lệnh trước khi đối tượng bị hủy bỏ khỏi bộ nhớ
- Trong một lớp chỉ xây dựng 1 hủy tử

- **Ví dụ**

```
#include<bits/stdc++.h>
using namespace std;
class Cat{
    private:
        string name;
        int age;
        int weight;
    public:
        Cat(){
            cout<<"\nmeo meo";
        }
        ~Cat(){
            cout<<"\nI die!";
        }
};
int main(){
    Cat x;
}
```

```
meo meo
I die!
--------------------------------
Process exited after 0.01986 seconds with return value 0
Press any key to continue . . .
```

Khai báo 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: Tạo các cấu tử và hủy tử

```
class CMatrix{
    private:
        int rows, cols;
        float *a;
    public:
        CMatrix();
        CMatrix(int, int);
        ~CMatrix();
        void setColRow(int,int)
        void printMatrix();
        void inputMatrix();
    };
Cmatrix::CMatrix() {
    rows = 0;
    cols = 0;
     a = NULL;
}
```

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

VI. Lớp mẫu (template class)

- **Lớp mẫu** là lớp sử dụng kiểu mẫu để khai báo các biến và phương thức
- **Ví dụ:** XD lớp QL một dãy các phần tử bất kỳ có 2 phương thức nhập dãy, in dãy.

```
template <class T>
class CArray{
        int n;
        T *a;
    public:
        void InputArr();
        void PrintArr();
};
template <class T >
void CArray<T>::PrintArr() {
    for(int i=0; i<n; i++){
       cout<<a[i]<<"\t";
}
```

```
template <class T >
void CArray<T>::InputArray() {
    cout<<"Nhap so phan tu:";
    cin>>n;
    if (a != NULL) delete a;
    a = new  T[n];
    for(int i=0; i<n; i++){
        cout<<"a["<<i<<"]=";
        cin >> a[i];
    }
}
```

```
int main () {
    CArray<float> x;
     x.InputArr();
     cout<<"Day vua
nhap:";
     x.PrintArr();
     return 0;
}
```

VII. 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 thí 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
5. 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.
6. Xây dựng lớp biểu diễn các đối tượng là các sinh viên (các thuộc tính, phương thức do bạn tự xác định)