Phần 2: Ngôn ngữ C++

Chương 2: Giới thiệu về ngôn ngữ lập trình C++

Các nội dung chính

1. Các đặc điểm mới của C++ so với C
2. Các khái niệm cơ bản của lập trình hướng đối tượng
3. Một số mở rộng của C++
4. Cấu trúc của một chương trình C++

1. Các đặc điểm mới của C++ so với C

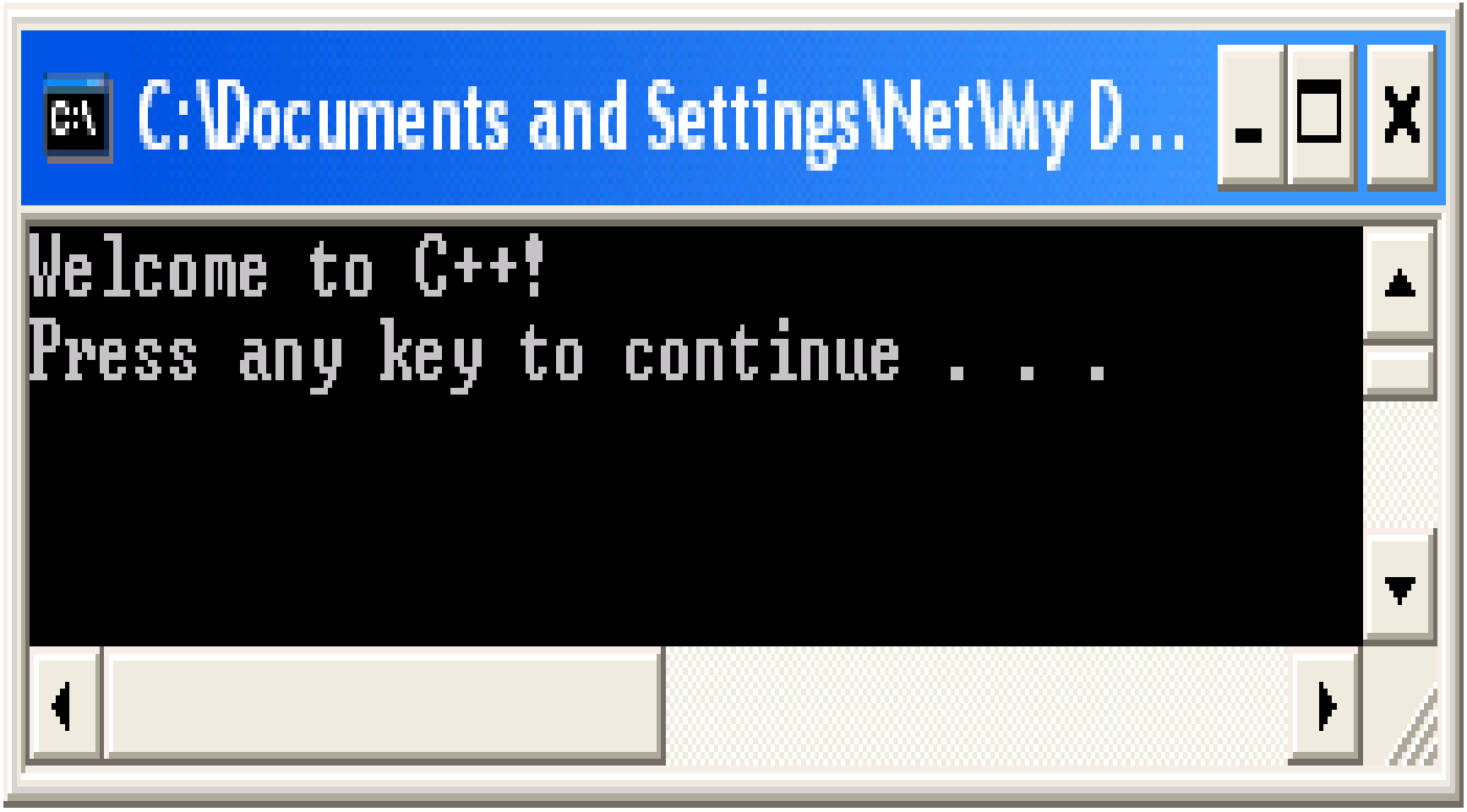
- C++ bổ sung khả năng lập trình hướng đối tượng (HĐT) với các khái niệm và các thành phần mới như: **lớp**, **đối tượng**, **sự che dấu thông tin**, **sự kế thừa**, **sự đa hình**,…Điều đó tạo cho C++ khả năng tổ chức chương trình theo cả phương pháp lập trình hướng chức năng và hướng đối tượng-một *ngôn ngữ lập trình lai*.
- Cho phép **định nghĩa chồng các hàm**, tức là các hàm có thể trùng tên với nhau trong cùng phạm vi định nghĩa và sử dụng.
- Cho phép các hàm có các **tham số nhận giá trị mặc định**
- Bổ sung loại hàm con *inline* nhằm tăng tốc độ thực hiện các hàm con.
- Bổ sung các lớp nhập/xuất mới nhằm đơn giản hoá các thao tác nhập/ xuất, tăng tính mở của các thao tác này khi phải nhập/xuất dữ liệu với các kiểu dữ liệu hay các đối tượng mới.
- Bổ sung các hàm cấp phát và giải phóng vùng nhớ động mới là *new* và *delete*.
- Bổ sung đối tượng, **tham số kiểu tham chiếu** giúp cho việc sử dụng các tham số của các hàm con được dễ dàng và hiệu quả hơn.
- Bổ sung loại chú thích mới-chú thích trên một dòng: //
- Cho phép đan xen giữa khai báo các đối tượng dữ liệu và các lệnh xử lý.

Chương trình C++ đầu tiên

- Program 2.1: In ra màn hình dòng “Welcome to C++!”

```
#include <cstdlib>
#include <iostream>   //tệp thư viện nhập/xuất chính trong C++
//#include <iostream.h>
using namespace std; //Khai báo không gian tên mặc định

int main()
{
    cout<<"Welcome to C++!"<<endl;
    system("PAUSE");
    return EXIT_SUCCESS;
}
```

Kết quả chạy Program 2.1



Chú thích về Program 2.1

- **namespace** (không gian tên): là công cụ cho phép quản lý sự xung đột về tên của các thành phần của chương trình như tên biến, tên lớp, tên hàm,v.v
- Đối tượng **cout** và toán tử **<<** nằm trong thư viện iostream, được dùng để xuất dữ liệu ra màn hình
- endl = ‘\n’: kí tự xuống dòng

2. Các khái niệm cơ bản của lập trình hướng đối tượng (object-oriented programming)

- Đối tượng và lớp (object and class)
- Thông báo và truyền thông báo (message)
- Sự che dấu các thành phần của lớp (còn gọi là sự đóng gói, encapsulation)
- Sự kế thừa (inheritance)
- Sự đa hình (polymorphism)

Đối tượng và lớp

- Đối tượng:
  - Là thành phần cơ bản nhất một chương trình theo kiểu hướng đối tượng, biểu diễn cho một đối tượng của bài toán
  - Là sự kết hợp gắn kết của các đối tượng dữ liệu và các thao tác xử lý cần thiết trên các đối tượng dữ liệu đó. Thao tác xử lý còn được gọi là phương thức (method), hay hàm thành viên

Hình tròn A
Bán kính r = 2
TínhChuVi() TínhDiệnTích()

Hình chữ nhật C
Chiều rộng a = 2 Chiều dài b = 3
TínhChuVi() TínhDiệnTích()

Một PT bậc 2
a = 3 b = 4 c = 1
TínhDelta() TínhNghiem()

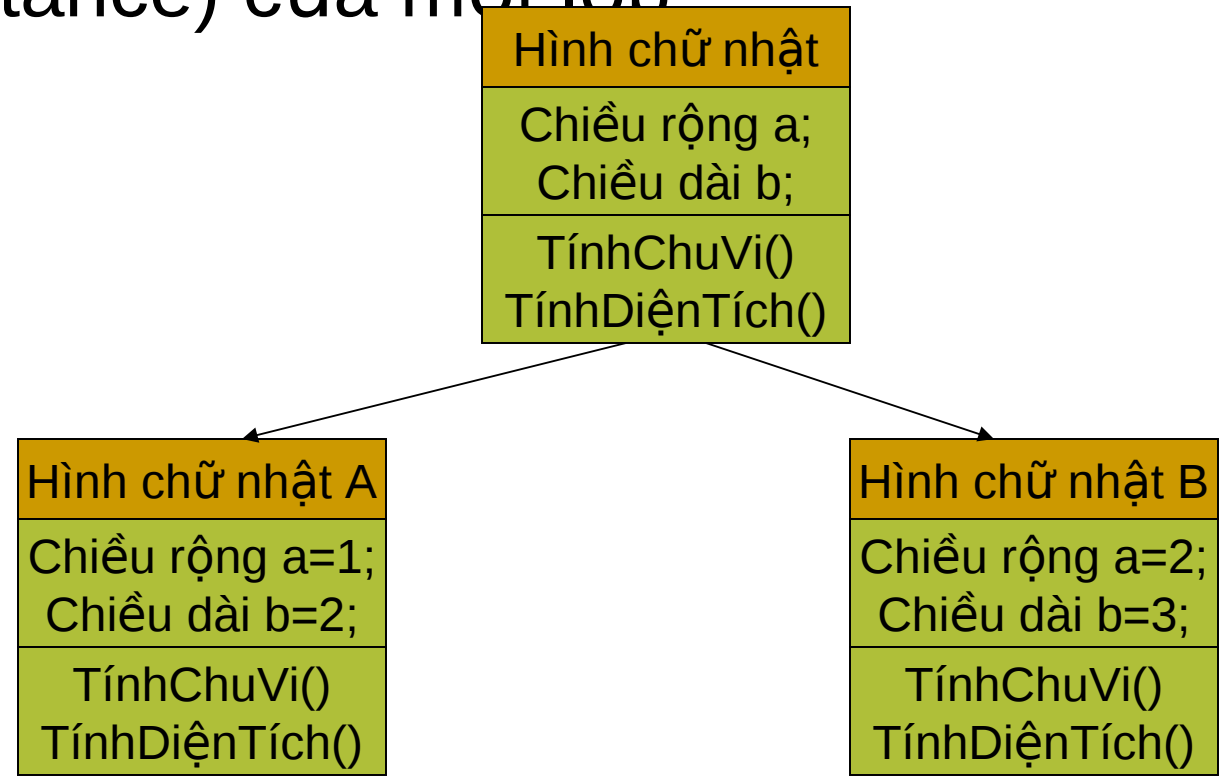
Đối tượng và lớp

- Lớp
  - Là sự khái quát hóa các đối tượng cùng loại
  - Có ý nghĩa đối với đối tượng tương tự như ý nghĩa của kiểu dữ liệu đối với các đối tượng dữ liệu, là cho phép dễ dàng tạo ra nhiều đối tượng cùng một kiểu (chung các kiểu dữ liệu và các phương thức)

Hình tròn	Hình chữ nhật	PT bậc 2
Bán kính r	Chiều rộng a; Chiều dài b;	a; b; c;
TínhChuVi() TínhDiệnTích()	TínhChuVi() TínhDiệnTích()	TínhDelta() TínhNghiem()

Quan hệ giữa đối tượng và lớp

- Đối tượng còn được gọi là *thể hiện* (instance) của một lớp



Chương trình mẫu tiếp theo

- **Program 2.2**: chương trình này sẽ có 1 đối tượng thuộc một lớp Circle, dùng để nhập vào giá trị bán kính và tính ra diện tích hình tròn này.
- Lưu ý: Phần khai báo các tệp thư viện và không gian tên của Program 2.2 giống như của Program 2.1.

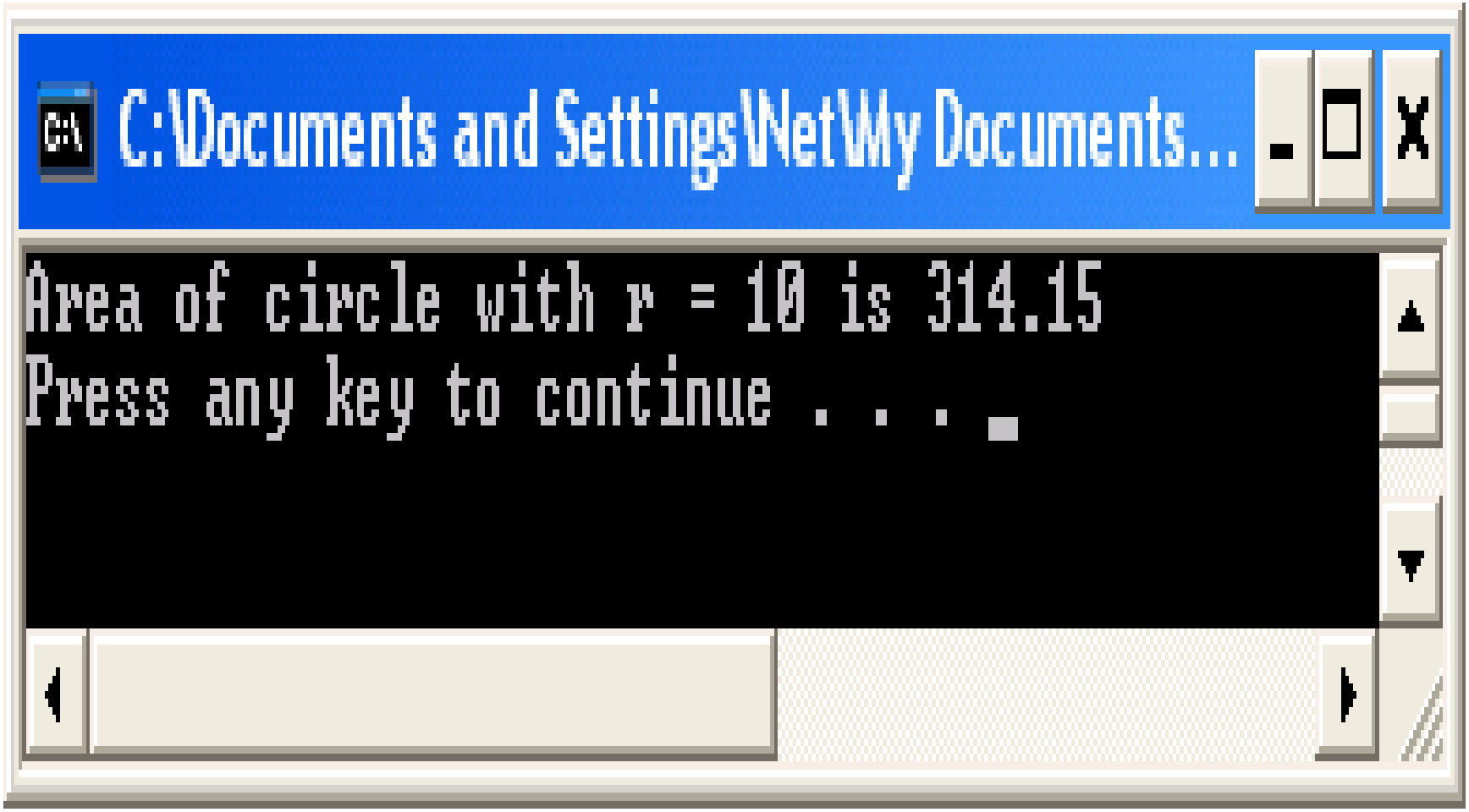
Program 2.2

```
class Circle {
    private:
        static const float PI=3.1415; //Hằng số tĩnh, hằng số của lớp
        float r;              //Bán kính, thành phần dữ liệu của từng đối tượng

    public:
        void setRadius(float re){
                r=re;
        }
        float getRadius(){
                return r;
        }
        float area(){
                return PI*r*r;
        }
};
```

Program 2.2: (tiếp và hết)

```
int main()
{
    Circle c;
    c.setRadius(10);
    cout<< “Area of circle with r = ”
    <<c.getRadius()<< " is “ <<c.area()<<endl;
    system("PAUSE");
    return EXIT_SUCCESS;
}
```

Kết quả chạy chương trình



Thông báo và truyền thông báo

- **Khái niệm**: Trong lập trình HĐT, khi một đối tượng *a* gọi một thao tác m của một đối tượng *b*, ta nói rằng a truyền thông báo m đến b. Thông báo thể hiện a muốn yêu cầu b thực hiện một công việc nào đó. Thao tác m() mà b cài đặt chính là để thực hiện yêu cầu đó

Sự che dấu các thành phần của lớp

- Khái niệm về sự che dấu: là khả năng hạn chế sự truy nhập trực tiếp vào thành phần nào đó của chương trình, mà thường là phần dữ liệu.
- Trong lập trình có cấu trúc thì thường là sự che dấu các thành phần dữ liệu cục bộ trong các hàm. Tuy nhiên khả năng che dấu của LTCT khá hạn chế, do đặc điểm là không có sự gắn kết chặt chẽ giữa dữ liệu và các thao tác xử lý. Nhờ đặc thù kết hợp dữ liệu và các thao tác xử lý vào trong đối tượng, đã cho phép lập trình HĐT tăng cường khả năng này.

Sự che dấu các thành phần của lớp

- Tại sao cần che dấu một thành phần?
  - Việc che dấu một thành phần khỏi các truy nhập không cần thiết sẽ tăng cường khả năng kiểm soát thành phần đó. Điều này giúp giảm thiểu những lỗi tiềm tàng, tăng mức an toàn của chương trình, giảm thời gian và chi phí bảo trì và nâng cấp hệ thống sau này.

Sự che dấu các thành phần của lớp

- Các mức độ che dấu trong C++
  - **private**: là mức cao nhất. Thành phần ở mức này hoàn toàn **không thể** truy nhập được từ bên ngoài lớp
  - **public**: là mức thấp nhất. Thành phần ở mức này có thể được truy nhập từ bên ngoài lớp.
  - **protected** (sẽ học sau): là mức trung bình giữa hai mức trên. Thành phần ở mức này của một lớp A sẽ không thể truy nhập được từ các đối tượng không thuộc lớp A, ngoại trừ từ những đối tượng là thuộc các lớp con của A

Ví dụ

```
class Circle {
    private:
        static const float PI=3.1415;
        float r;

    public:
        void setRadius(float re){
                r=re;
        }
        float getRadius(){
                return r;
        }
        float area(){
                return PI*r*r;
        }
};
```

```
void main() {
        Circle c;
        c.setRadius(15.5);  //OK
        c.r = 10;  //Error with private member
        cout<<“Ban binh r=”<<c.getRadius();
};
```

So sánh giữa **class** và **struct**

- Trong C++, **struct** cũng được mở rộng để cho phép bổ sung các hàm thành viên để thực hiện các xử lý trên các trường dữ liệu.
- Ngoài ra, khả năng kế thừa của **class** cũng được đưa vào trong **struct**, làm cho chức năng của struct cũng không kém gì của **class**.
- Tuy nhiên, để duy trì tính tương thích với C, các thành phần trong **struct** có mức độ che dấu mặc định là **public**.

Chương trình minh họa

- Program 2.3: về chức năng tương tự như Program 2.2, nhưng sử dụng struct thay cho class.

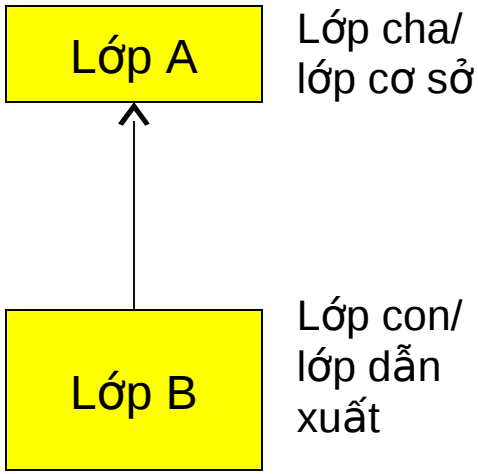
```
struct Circle {
    float r;
    static const float PI=3.14;

    float area(){
        return PI*r*r;
    }
};
```

Program 2.3 (tiếp và hết)

```
int main(){
    Circle c;
    c.r = 20;       //Truy nhập trực tiếp vào trường dữ liệu
    cout<<“Area of the circle with r=”<<c.r<<“ is ”
    <<c.area()<<endl;

    system("PAUSE");
    return EXIT_SUCCESS;
}
```

Sự kế thừa

- **Khái niệm**: kế thừa là một cách tái sử dụng mới trong C++ và các ngôn ngữ lập trình HĐT khác. Khi một lớp B kế thừa lớp A, tức là B sẽ tái sử dụng toàn bộ các thành phần trong lớp A, bao gồm cả phần dữ liệu và các phương thức

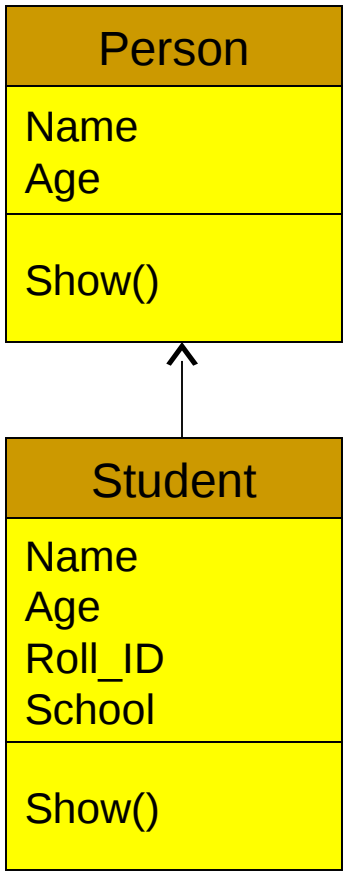


Kế thừa

- Phân loại kế thừa
  - Theo số lượng lớp cơ sở
    - Kế thừa đơn: một lớp dẫn xuất kế thừa chỉ một lớp cơ sở
    - Kế thừa bội: một lớp dẫn xuất kế thừa từ 2 lớp cơ sở trở lên
  - Theo mức độ che dấu
    - private
    - protected
    - public: kiểu kế thừa thông dụng nhất

Sự đa hình

- **Khái niệm**: đa hình xuất hiện cùng với sự kế thừa, khi trong lớp cơ sở và lớp dẫn xuất của nó có các hàm thành viên có khuôn mẫu giống nhau. Giả sử ta có một đối tượng *Obj* mà chưa biết rõ nó thuộc lớp nào, và muốn *Show* nội dung của *Obj*. Nếu *Obj* thuộc lớp *Person* thì nội dung của một *Person* sẽ được in ra. Còn nếu *Obj* thuộc lớp *Student* thì nội dung của một *Student* sẽ được in ra. Việc Obj thuộc lớp nào chỉ có thể xác định vào lúc chạy chương trình (run time), chứ không xác định được vào lúc viết và dịch chương trình (compile time). Khả năng mà một đối tượng có thể liên kết với các hàm khác nhau của các lớp khác nhau gọi là sự đa hình.



Sự đa hình

- **Sự khó khăn của đa hình**: chính là việc xác định hàm Show() của lớp nào phải được xác định vào lúc chạy, chứ không phải lúc dịch.
- **Giải pháp trong C++**:
  - Hàm ảo (virtual function)
  - Cơ chế liên kết muộn (late binding)

3. Một số mở rộng của C++

- Khả năng nhập/xuất mới
- Tham chiếu (reference)
- Tham số ngầm định trong hàm
- Các toán tử mới quản lý bộ nhớ động *new* và *delete*
- Tiện ích khai báo mọi nơi và chú thích cuối dòng
- Định nghĩa chồng hàm (overloading functions)

Khả năng nhập/xuất mới

- Nhập dữ liệu:
  - sử dụng đối tượng **cin** của lớp istream và phép toán **>>**
- Xuất dữ liệu:
  - Sử dụng đối tượng **cout** của lớp ostream và phép toán **<<**

- Lưu ý: các đối tượng và thao tác nhập xuất này đều nằm trong thư viện <iostream> (hoặc <iostream.h>)

Tham chiếu

- **Khái niệm**: tham chiếu là một tên gọi mới của một vùng nhớ được cấp phát cho một đối tượng.

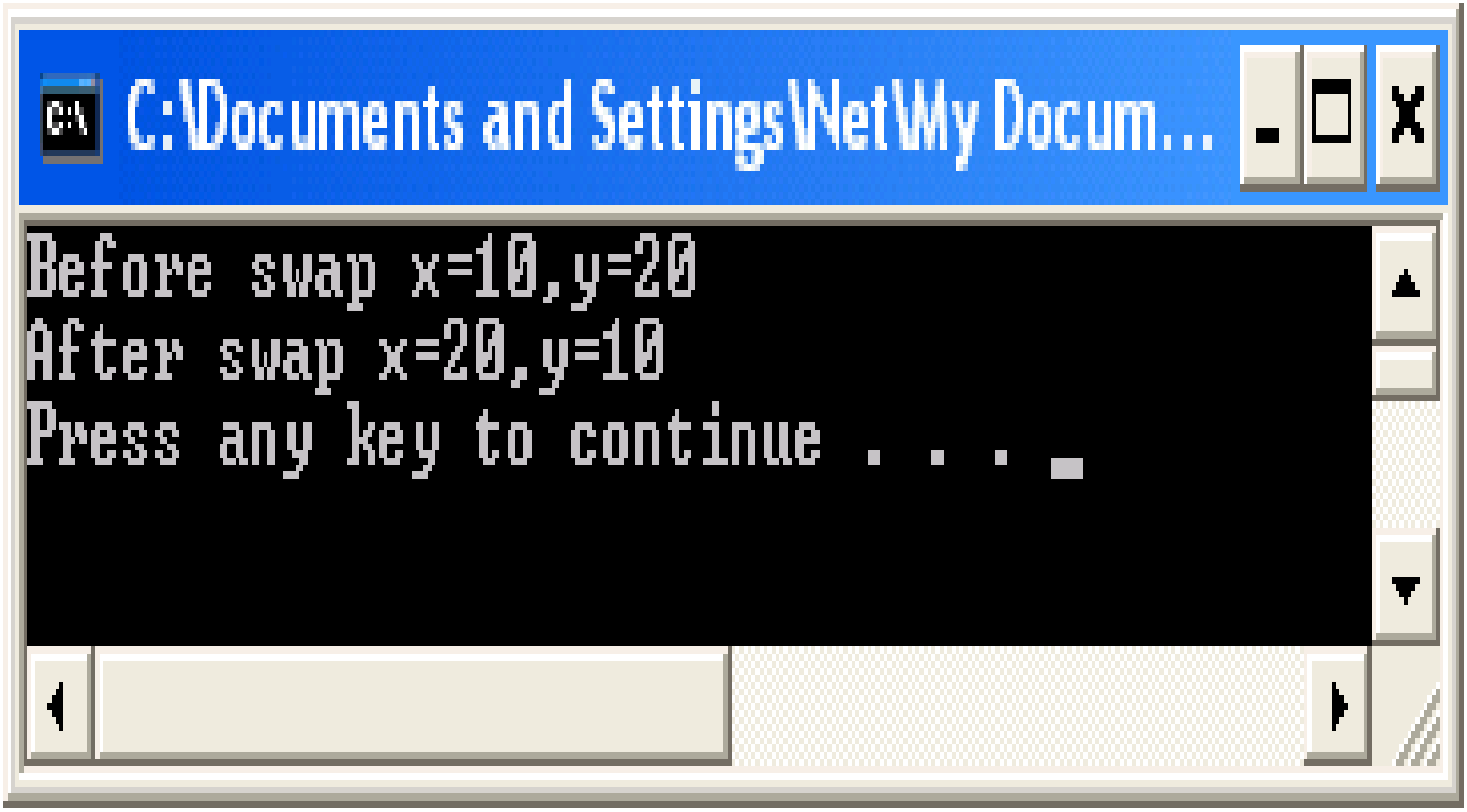
```
int  n=10;
int &m = n; //m là biến tham chiếu đến n
m = 20;      //tương đương n=20
```

Tham chiếu

- **Vai trò**: thường được sử dụng trong tham số của hàm để thực hiện truyền tham biến (trong C chỉ cho phép truyền tham trị). (Xem Program 2.4)

```
void exchange(int &a, int &b){
    int c=a;
    a=b;
    b=c;
}

int main(){
    int x=10, y=20;
    cout<<"Before swap x="<<x<<",y="<<y<<endl;

    exchange(x,y);
    cout<<"After swap x="<<x<<",y="<<y<<endl;
    system("PAUSE");
    return EXIT_SUCCESS;
}
```

Kết quả chạy Program 2.4



Hàm với tham số nhận giá trị mặc định

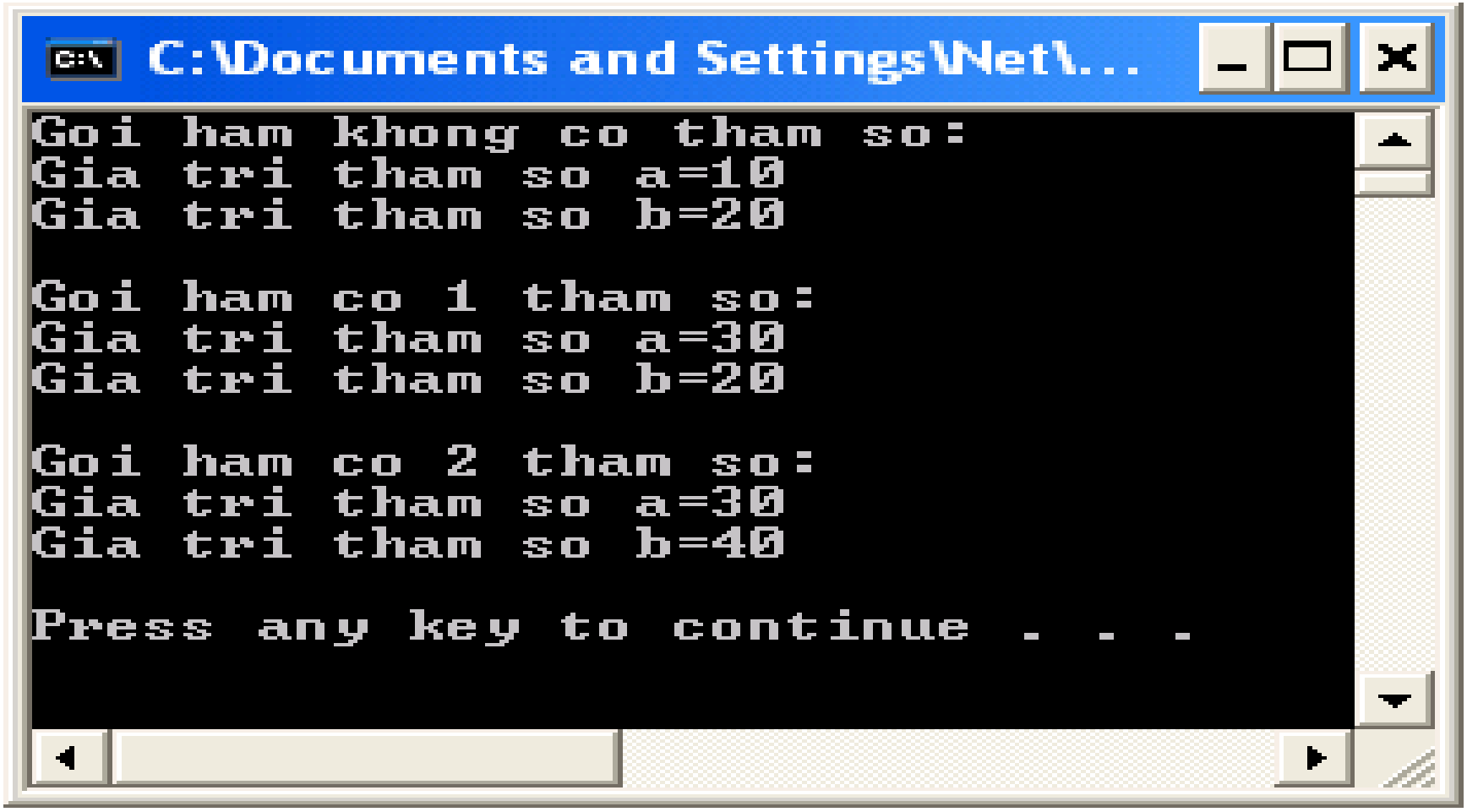
- Là mở rộng trong C++ cho phép khi gọi một hàm con, ta có thể bỏ qua một số tham số của nó, khi đó các tham số này sẽ nhận các giá trị mặc định mà đã được quy định trước đó khi khai báo hàm con này.

Chương trình minh họa (program 2.6)

```
//Khai báo hàm với giá trị ngầm định
void HamND(int a = 10, int b = 20);

int main()
{
    cout<<"Goi ham khong co tham so:"<<endl;
    HamND(); //Gọi hàm với giá trị ngầm định
    cout<<"Goi ham co 1 tham so:"<<endl;
    HamND(30);
    cout<<"Goi ham co 2 tham so:"<<endl;
    HamND(30,40);
    system("PAUSE");
    return EXIT_SUCCESS;
}
```

```
//Định nghĩa hàm với giá trị ngầm định
void HamND(int a, int b)
{
    cout<<"Gia tri tham so a="<<a<<endl;
    cout<<"Gia tri tham so b="<<b<<endl;
    cout<<endl;
}
```

Kết quả chạy chương trình



Định nghĩa chồng hàm

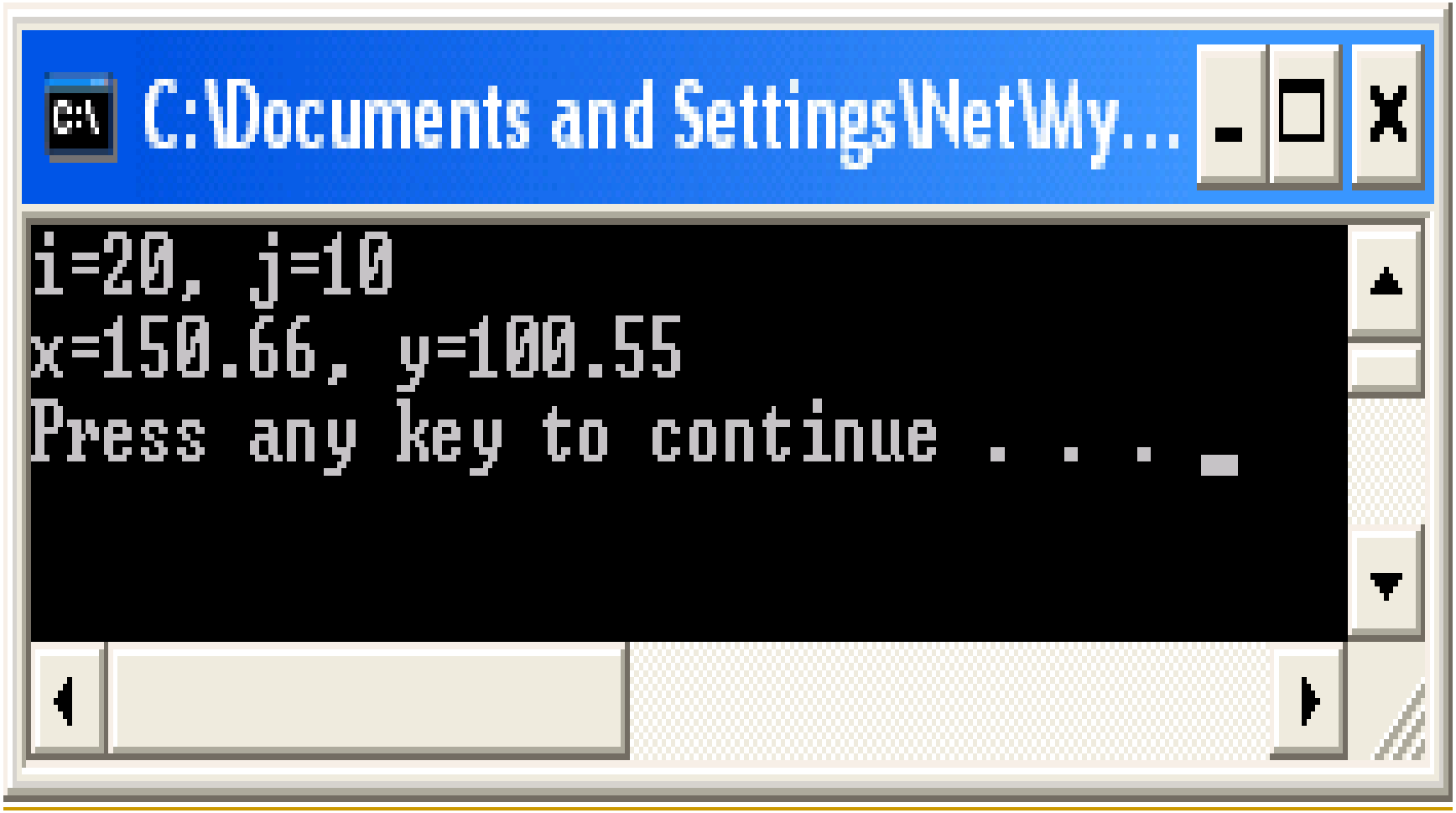
- Là khả năng cho phép định nghĩa lại một hàm nhiều lần với cùng một tên hàm, nhưng với các tham số khác nhau (có thể khác nhau về số lượng tham số và/hoặc kiểu dữ liệu của tham số)

Chương trình ví dụ (program 2.7)

```
//Định nghĩa chồng hàm swap
void swap(int &a, int &b){
        int c=a;
        a=b;
        b=c;
}
void swap(float &a, float &b){
        float c=a;
        a=b;
        b=c;
}
```

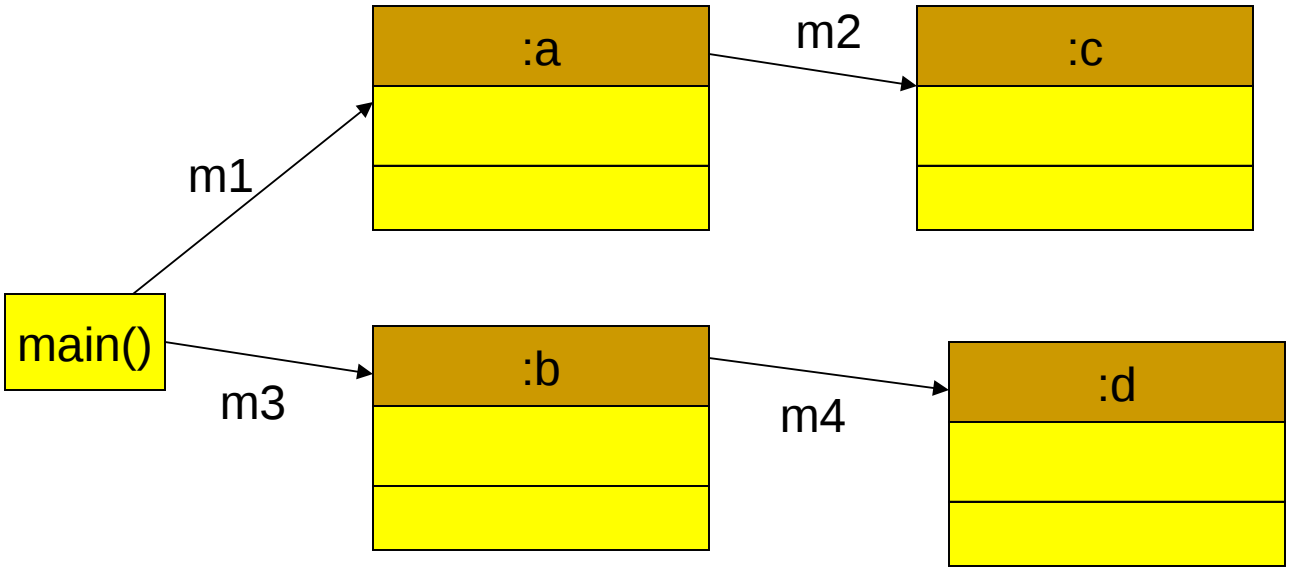
```
int main()
{
    int i=10, j=20;
    float x=100.55, y=150.66;
    swap(i,j); //Gọi hàm swap(int, int)
    cout<<"i="<<i<<", j="<<j<<endl;
    swap(x,y); //Gọi hàm swap(float, float)
    cout<<"x="<<x<<", y="<<y<<endl;

    system("PAUSE");
    return EXIT_SUCCESS;
}
```

Kết quả chạy chương trình



4. Cấu trúc của một chương trình C++

- Phần chính của một chương trình C++ (theo kiểu HĐT nói chung) bao gồm 2 phần:
  - Tập các đối tượng
  - Tập các thông báo từ hàm *main()* đến các đối tượng và được truyền giữa các đối tượng

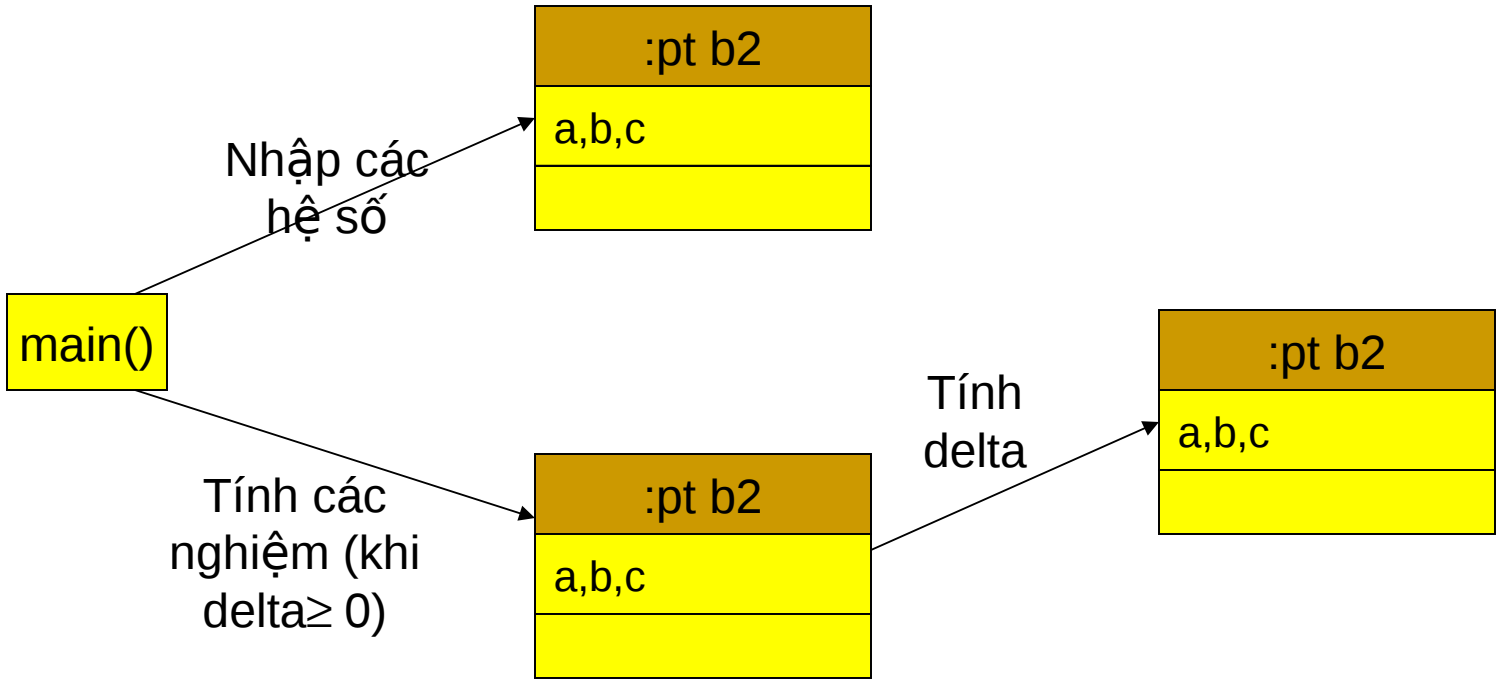


Chương trình minh họa

- **Program 2.5**: viết chương trình giải phương trình bậc 2 theo phương pháp HĐT
  - Đầu vào: 3 hệ số của 1 phương trình bậc 2
  - Đầu ra: PT có mấy nghiệm và giá trị từng nghiệm nếu có

- Phân tích: coi mỗi PT bậc 2 là một đối tượng của một lớp PT bậc 2. Khi đó cấu trúc của một chương trình sẽ như sau:

Cấu trúc Program 2.5



Program 2.5 (phần đầu)

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

using namespace std;
```

Program 2.5 (phần lớp PTB2)

```
class PTB2 {
    float a, b, c;  //Mức độ che dấu mặc định là private
    public:
        void NhapHS (float xa, float xb, float xc) {
          a = xa; b = xb; c = xc;
        }
        float  TinhDelta()  {  return (b*b - 4*a*c); }
      int  TinhNghiem(float & x1, float & x2); //Hàm trả về số nghiệm
};
```

Program 2.5 (tiếp)

```
//Đ/n hàm trả về số nghiệm
int  PTB2::TinhNghiem(float & x1, float & x2) {
        float delta = TinhDelta();
        if (delta < 0) return 0;
        else if (delta == 0) {
           x1 = -b/(2*a);
           return 1;
        } else {
           x1 = (-b-sqrt(delta))/(2*a);
           x2 = (-b+sqrt(delta))/(2*a);
           return 2;
        }
}
```

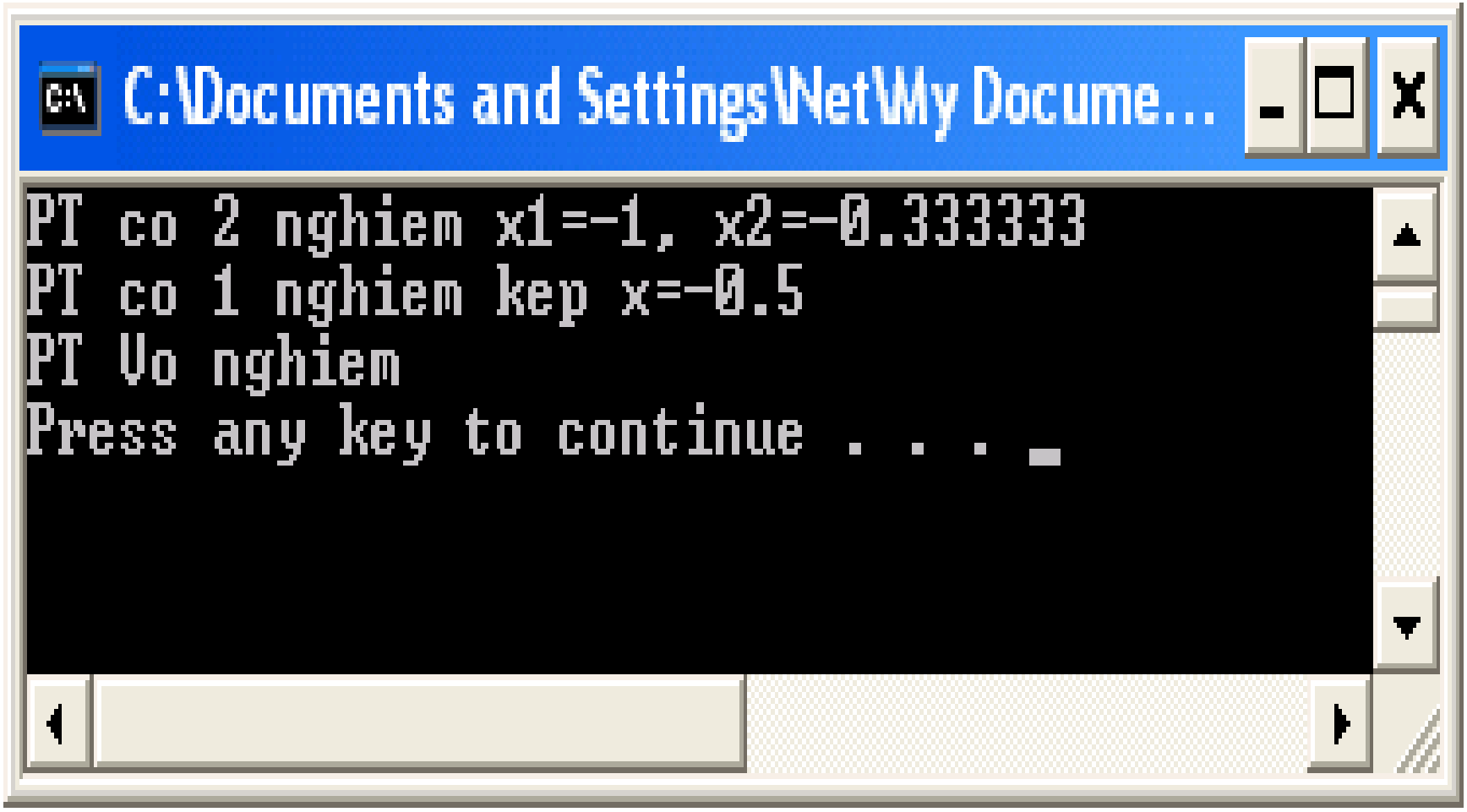
Program 2.5 (phần hàm *main*)

```
int main()
{
    PTB2 pt[3];
    pt[0].NhapHS(3,4,1);
    pt[1].NhapHS(4,4,1);
    pt[2].NhapHS(5,4,1);

    float x1,x2;
    int n;
    //tiếp trang sau
}
```

Program 2.5 (hàm *main*, phần cuối)

```
int main()
{
    …
    for (int i=0;i<3;i++){
            n = pt[i].TinhNghiem(x1,x2);
            if (n==0) cout<<"PT Vo nghiem"<<endl;
            else if (n==1){
                        cout<<"PT co 1 nghiem kep x="<<x1<<endl;
            }else cout<<"PT co 2 nghiem x1="<<x1<<",
    x2="<<x2<<endl;
    }
    return system("PAUSE"), EXIT_SUCCESS;
}
```

Kết quả chạy Program 2.5



Câu hỏi tóm tắt

- Các đặc điểm mới của C++ là gì?
- Các khái niệm cơ bản của lập trình HĐT là gì?
- Cấu trúc của một chương trình C++ như thế nào?

Bài tập

- **Bài 1**: Viết chương trình tính điện trở tương đương của 2 điện trở mắc song song theo phương pháp hướng đối tượng. Giá trị của các điện trở được nhập từ bàn phím. (Gợi ý: coi mỗi điện trở như một đối tượng)
- **Bài 2**: Mở rộng bài 1 cho việc tính điện trở của N điện trở mắc song song. Hơn nữa, khi nhập dữ liệu cho các điện trở cần kiểm tra tính hợp lệ của dữ liệu nhập vào.
- **Bài 3**: Viết chương trình trong đó có hai hàm swap chồng nhau, một hàm cho phép hoán đổi giá trị của hai kí tự, còn hàm kia cho phép hoán đổi giá trị của hai chuỗi kí tự. Chương trình sẽ thực hiện việc hoán đổi 1 cặp kí tự và 1 cặp chuỗi có giá trị nhập từ bàn phím.

Xin cảm ơn!