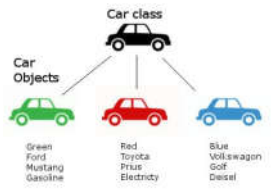



The diagram shows four student objects (Jenna, John, Maria, James) with their roll numbers (R005, R010, R025, R005) and a 'Class Student' box containing attributes (name, rollNo) and methods (setName(), setRollNo()). Arrows point from the objects to the class box.



The diagram shows a 'Car class' box with an arrow pointing to it from a group of three car objects: a green Ford Mustang (Gasoline), a red Toyota Prius (Electricity), and a blue Volkswagen Golf (Diesel).

## KỸ THUẬT LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG C++



Abstraction, Encapsulation, Modularity, Identity, Typing, Concurrency, Polymorphism, Persistence, Hierarchy, Messaging, object oriented programming, Scott Salsman.

**Tạ Quang Chiếu**  
[quangchieu.ta@gmail.com](mailto:quangchieu.ta@gmail.com)  
**FIT - HUMG**  
 05-02-2017

1

## GIỚI THIỆU CHUNG

### ❖ Mục đích:

- Giới thiệu phương pháp lập trình hướng đối tượng
- Tìm hiểu các khái niệm cơ bản của lập trình hướng đối tượng (OOP)
- Lập trình hướng đối tượng với C++,

### ❖ Thời lượng: 45t = 30t lý thuyết + 15t bài tập

### ❖ Yêu cầu:

- Nắm được lý thuyết, giải các bài tập và lập trình hướng đối tượng bằng ngôn ngữ C++ trên máy tính

### ❖ Phần mềm: TURBO C 3.0, C Free, Dev C++, CodeBlocks, Visual Studio 2008, 2010, ...

### ❖ Cách học:

- Làm các bài tập theo từng phần bằng cách viết các chương trình C++ trên máy tính,
- Bài tập lớn.

### ❖ Tài liệu tham khảo:

- Lập trình hướng đối tượng với C++, Nguyễn Thanh Thủy (chủ biên), NXB KHK, 1999.
- Giáo trình C++ và lập trình hướng đối tượng, GS Phạm Văn Ất, NXB Hồng Đức, 2009.
- Giáo trình Kỹ thuật lập trình hướng đối tượng bằng C++, Nguyễn Tuấn Anh, NXB Giáo dục Việt Nam
- Tham khảo tài liệu trên internet

## CÁCH ĐÁNH GIÁ HỌC PHẦN

Điểm học phần được đánh giá theo quy chế hiện hành:

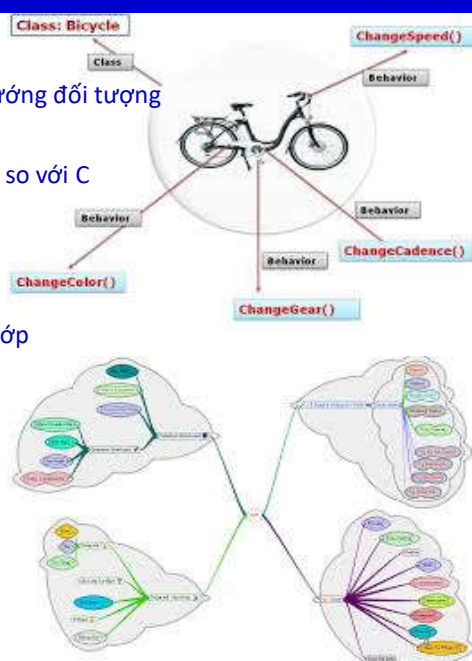
$$\text{Điểm đánh giá học phần} = 0,6A + 0,3B + 0,1C$$

TT	Điểm thành phần	Quy định	Trọng số	
1	Điểm chuyên cần	Số tiết tham dự học/tổng số tiết	10%	C
2	Điểm bài tập nhóm	- Bài tập lớn	30%	B
3	Điểm kiểm tra giữa kỳ	- Thi viết (60 phút)		
4	Điểm thảo luận bài tập	Bài tập giao về nhà hoặc thảo luận trên lớp		
5	Điểm thi kết thúc học phần	- Thi viết (60 phút). - Tham dự đủ 80% tiết lý thuyết - Dự thi kết thúc học phần.	60%	A

3

## NỘI DUNG

- ❖ Chương 1. Giới thiệu về lập trình hướng đối tượng
- ❖ Chương 2. Một số mở rộng của C++ so với C
- ❖ Chương 3. Lớp và đối tượng
- ❖ Chương 4. Định nghĩa toán tử trên lớp
- ❖ Chương 5. Kỹ thuật thừa kế
- ❖ Chương 6. Tính đa hình
- ❖ Chương 7. Khuôn hình



## CHƯƠNG 1. GIỚI THIỆU VỀ LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG

### Bài tập chương 1 – Ôn tập ngôn ngữ lập trình C

1. Lập trình tìm Min, Max của một dãy có n số thực ( $n > 0$ ).
2. Lập trình tính tổng  $S = 1 + 1/2 + \dots + 1/n$  ( $n > 0$ ).
3. Lập trình sắp xếp tăng dần dãy n số thực ( $n > 0$ ).
4. Lập chương trình tính diện tích hình tam giác, cho trước 3 cạnh a,b,c (giả thiết a, b, c tạo thành tam giác), theo công thức sau:

$$S = \sqrt{p(p-a)(p-b)(p-c)} \quad , \quad p = \frac{a+b+c}{2}$$

5. Viết hàm tính tổng  $S = 1+2+\dots+n$ , với n là tham số truyền vào và hàm trả về tổng S.
6. Cho một danh sách hàng hóa gồm tên hàng, số lượng, đơn giá, thành tiền (thành tiền = số lượng x đơn giá). Viết chương trình thực hiện các việc sau:
  - Nhập dữ liệu hàng hóa từ bàn phím, kết thúc nhập khi tên hàng là "\*" (số hàng hóa không quá 200).
  - Đưa ra màn hình tên hàng và số lượng các hàng hóa có số lượng nhỏ nhất.
  - Đọc thêm một hàng hóa mới (tên hàng, số lượng, đơn giá), chèn vào vị trí thứ k ( $0 < k < \text{số hàng hóa}$ ), với k được đọc từ bàn phím.
  - Tính tổng trị giá hàng hóa trong danh sách.
7. Lập trình quản lý danh sách sinh viên, thông tin gồm: họ tên, điểm toán, điểm lý, điểm hóa, điểm trung bình. Viết các hàm nhập danh sách, in danh sách, sắp xếp theo điểm trung bình giảm dần.

5

## NỘI DUNG

### ❖ Chương 1. Giới thiệu về lập trình hướng đối tượng

### ❖ Chương 2. Một số mở rộng của C++ so với C

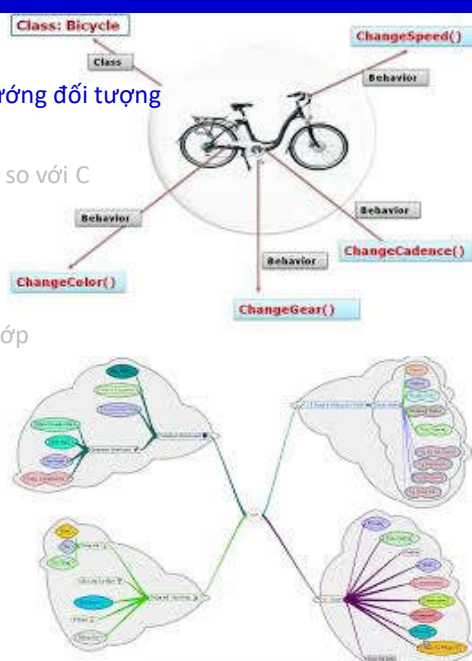
### ❖ Chương 3. Lớp và đối tượng

### ❖ Chương 4. Định nghĩa toán tử trên lớp

### ❖ Chương 5. Kỹ thuật thừa kế

### ❖ Chương 6. Tính đa hình

### ❖ Chương 7. Khuôn hình



## CHƯƠNG 1. GIỚI THIỆU VỀ LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG

### 1. Tổng quan

- ❖ Lập trình hướng đối tượng
  - Lập trình định hướng đối tượng
  - Object Oriented Programming (OOP)
- ❖ Được xem là:
  - Cách tiếp cận mới, hiệu quả hơn
  - Giúp tăng năng suất
  - Dễ dàng bảo trì, sửa đổi, nâng cấp
- ❖ Mục đích:
  - Giảm bớt thao tác viết mã
  - Mô tả chân thực thế giới thực

#### Quá trình phát triển các kỹ thuật lập trình

- ❖ Lập trình tuyến tính (phi cấu trúc) → Lập trình cấu trúc (thủ tục/hàm) → Lập trình Modul → Lập trình hướng đối tượng

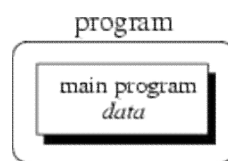
7

## CHƯƠNG 1. GIỚI THIỆU VỀ LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG

### 2. Tổng quan về các kỹ thuật lập trình

- ❖ **Lập trình tuyến tính**
  - Còn gọi là lập trình phi cấu trúc
  - Giải quyết các bài toán tương nhỏ, đối đơn giản
- ✓ **Đặc điểm:**
  - Chỉ gồm một chương trình chính,
  - Gồm một dãy tuần tự các câu lệnh,
  - **Toàn bộ chương trình** được viết trong hàm `main()` duy nhất,
  - Chương trình ngắn, ít hơn 100 dòng lệnh.
- ✓ **Nhược điểm:**
  - Không sử dụng lại được các đoạn mã,
  - Không có khả năng kiểm soát phạm vi truy xuất dữ liệu,
  - Mọi dữ liệu trong chương trình là toàn cục,
  - Dữ liệu có thể bị sửa đổi ở bất cứ vị trí nào trong chương trình,

→ Không đáp ứng được việc triển khai phần mềm



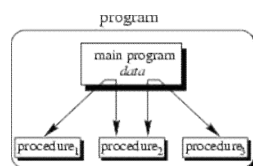
8

## CHƯƠNG 1. GIỚI THIỆU VỀ LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG

### 2. Tổng quan về các kỹ thuật lập trình

#### ❖ Lập trình cấu trúc

- ✓ Ra đời vào những năm 70:
- ✓ Chương trình được chia nhỏ thành chương trình con:
  - Thủ tục (Procedure)
  - Hàm (Function)
- ✓ Các chương trình con:
  - Độc lập với nhau và có dữ liệu riêng
  - Trao đổi qua: tham số và biến toàn cục
- ✓ Xuất hiện khái niệm trừu tượng hoá
  - Là khả năng quan sát sự vật mà:
    - Không quan tâm tới các chi tiết không quan trọng bên trong
    - Không quan tâm tới việc thực hiện như thế nào
  - Trừu tượng hoá dữ liệu
  - Trừu tượng hoá thao tác
- ✓ Ngôn ngữ lập trình cấu trúc: C, Pascal...



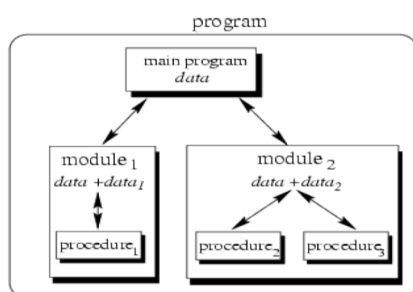
9

## CHƯƠNG 1. GIỚI THIỆU VỀ LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG

### 2. Tổng quan về các kỹ thuật lập trình

#### ❖ Với lập trình mô đun:

- Các thủ tục có chung một chức năng được nhóm lại với nhau
- Chương trình được chia thành nhiều phần nhỏ
- Các phần tương tác thông qua việc gọi thủ tục
- Mỗi mô đun có dữ liệu của riêng nó



10

## CHƯƠNG 1. GIỚI THIỆU VỀ LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG

### 2. Tổng quan về các kỹ thuật lập trình

#### ❖ **Nhược điểm của lập trình truyền thống:**

- Chương trình khó kiểm soát
- Khó khăn trong việc bổ sung, nâng cấp chương trình
- Khi thay đổi, bổ sung dữ liệu dùng chung thì phải thay đổi gần như tất cả thủ tục/hàm liên quan
- Khả năng sử dụng lại các đoạn mã chưa nhiều
- Không mô tả đầy đủ, trung thực hệ thống trong thực tế

11

## CHƯƠNG 1. GIỚI THIỆU VỀ LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG

### 2. Tổng quan về các kỹ thuật lập trình

#### ❖ **Lập trình hướng đối tượng**

##### ✓ **Là phương pháp lập trình:**

- Mô tả chính xác các đối tượng trong thế giới
- Lấy đối tượng làm nền tảng xây dựng thuật toán
- Thiết kế xoay quanh dữ liệu của hệ thống
- Chương trình được chia thành các lớp đối tượng
- Dữ liệu được đóng gói, che dấu và bảo vệ
- Đối tượng làm việc với nhau qua thông báo
- Chương trình được thiết kế theo cách từ dưới lên (bottom-up)



12

## CHƯƠNG 1. GIỚI THIỆU VỀ LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG

### 3. Một số khái niệm cơ bản

❖ **Hệ thống hướng đối tượng:** Là hệ thống có đặc điểm sau:

- ✓ **Gồm tập hợp các đối tượng**
  - Sự đóng gói của 2 thành phần:
    - Dữ liệu (thuộc tính của đối tượng)
    - Các thao tác trên dữ liệu
- ✓ **Các đối tượng có thể kế thừa các đặc tính của đối tượng khác**
- ✓ **Hoạt động thông qua sự tương tác giữa các đối tượng nhờ cơ chế truyền thông điệp**
  - Thông báo
  - Gửi & nhận thông báo

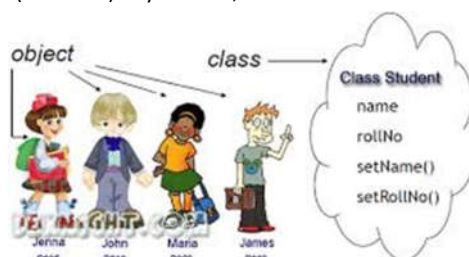
13

## CHƯƠNG 1. GIỚI THIỆU VỀ LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG

### 3. Một số khái niệm cơ bản

❖ **Đối tượng:**

- ✓ **Là khái niệm trừu tượng phản ánh các thực thể trong thế giới thực**
  - Có thể là một thực thể vật lý
  - Có thể là một khái niệm trừu tượng
- ✓ **Được định nghĩa là sự thể hiện của một lớp**
- ✓ **Chính là các thực thể trong hệ thống hướng đối tượng**
- ✓ **Một đối tượng là sự đóng gói 2 thành phần:**
  - Trạng thái (state) hay dữ liệu
  - Các ứng xử (behavior) hay hành vi, thao tác



14

## CHƯƠNG 1. GIỚI THIỆU VỀ LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG

### 3. Một số khái niệm cơ bản

#### ❖ Thuộc tính & phương thức

##### ✓ Thuộc tính bao gồm:

- Xác định trạng thái của một đối tượng, đặc điểm của đối tượng
- Hằng, biến
- Tham số nội tại

##### ✓ Thuộc tính được xác định kiểu, gồm:

- Kiểu cố định
- Kiểu do người dùng định nghĩa

##### ✓ Phương thức: Là các hành vi của đối tượng, được sử dụng để thực hiện các thao tác trên các thuộc tính

- Các hàm nội tại của đối tượng
- Có kiểu trả về
- Tên gọi khác: hàm thành viên

15

## CHƯƠNG 1. GIỚI THIỆU VỀ LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG

### 3. Một số khái niệm cơ bản

#### ❖ Lớp (Class)

- Là tập hợp các đối tượng có cùng thuộc tính và hành vi
- Là bản thiết kế hoặc bản mẫu mô tả một cấu trúc dữ liệu gồm:
  - Các thành phần dữ liệu
  - Các phương thức
- Lớp được sử dụng như kiểu dữ liệu người dùng định nghĩa

#### ❖ 1.2.3. Sự đóng gói (Encapsulation)

- Sự đóng gói là cơ chế ràng buộc dữ liệu và các thao tác trên dữ liệu thành thể thống nhất.
- Sự đóng gói gồm:
  - Bao gói: người dùng giao tiếp với hệ thống qua giao diện
  - Che dấu: ngăn chặn các thao tác không được phép từ bên ngoài
- Ưu điểm:
  - Quản lý sự thay đổi
  - Bảo vệ dữ liệu

16



## CHƯƠNG 1. GIỚI THIỆU VỀ LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG

## 3. Một số khái niệm cơ bản

## ❖ Sự đóng gói (Encapsulation)

```
class Box
{
public:
    double getVolume(void)
    {
        return length * breadth * height;
    }
private:
    double length;    // Length of a box
    double breadth;   // Breadth of a box
    double height;    // Height of a box
};
```

- Các biến **length**, **breadth**, và **height** là **private**.
- Chúng chỉ có thể được truy cập bởi các thành viên khác của lớp **Box**, và không thể bởi bất kỳ phần khác trong chương trình.
- Đây là một cách thực hiện tính đóng gói trong C++.

17

## CHƯƠNG 1. GIỚI THIỆU VỀ LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG

## 3. Một số khái niệm cơ bản

## ❖ Sự kế thừa - Inheritance

- Là khả năng cho phép xây dựng lớp mới được thừa hưởng các thuộc tính của lớp đã có
- Khi tạo một lớp, thay vì viết toàn bộ các thành viên dữ liệu và các hàm thành viên mới, lập trình viên có thể nên kế thừa các thành viên của một lớp đang tồn tại.
- Lớp đang tồn tại này được gọi là **Base Class - lớp cơ sở**, và lớp mới được xem như là **Derived Class – lớp thừa kế**.

## ✓ Đặc điểm:

- Lớp nhận được có thể bổ sung thêm các thành phần
- Hoặc định nghĩa lại các thành phần của lớp cha

## ✓ Các loại kế thừa:

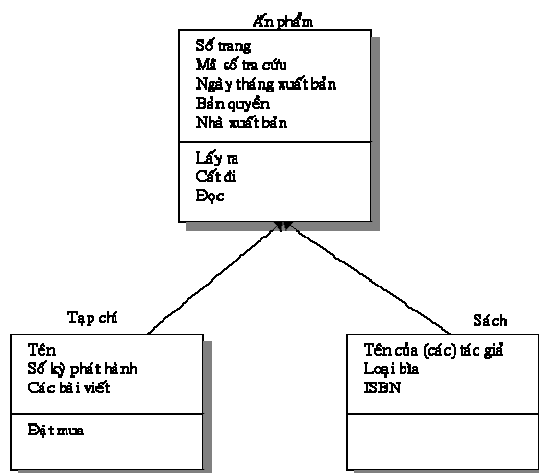
- Đơn kế thừa: là một lớp con chỉ kế thừa từ một lớp cha. Và tất nhiên một lớp cha có thể có nhiều lớp con kế thừa từ nó và lớp con đó có thể là lớp cha của lớp khác.
- Đa kế thừa: là một lớp con có thể kế thừa từ hai hoặc nhiều lớp cha (lớp cơ sở)
- Trong kế thừa nếu Class B kế thừa từ Class A và Class C kế thừa từ Class B. Thì Class C được thừa hưởng các thuộc tính phương thức (cho phép) của A và B.

18

## CHƯƠNG 1. GIỚI THIỆU VỀ LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG

### 3. Một số khái niệm cơ bản

#### ❖ Sự kế thừa - Inheritance

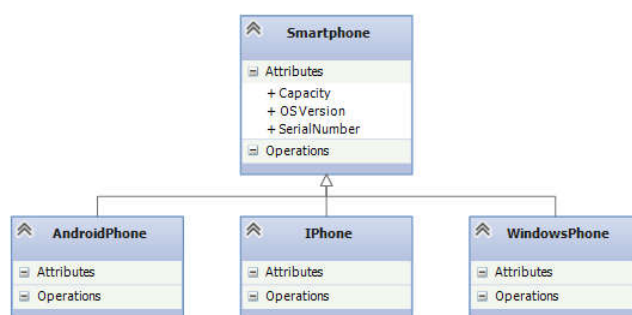


19

## CHƯƠNG 1. GIỚI THIỆU VỀ LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG

### 3. Một số khái niệm cơ bản

#### ❖ Sự kế thừa - Inheritance



20

## CHƯƠNG 1. GIỚI THIỆU VỀ LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG

## 3. Một số khái niệm cơ bản

## ❖ Sự kế thừa - Inheritance

```

1 #include <iostream>
2 using namespace std;
3 // lop co so: Hinh
4 class Hinh
5 {
6 public:
7     void setChieuRong(int rong)
8     {
9         chieurong = rong;
10    }
11    void setChieuCao(int cao)
12    {
13        chieucao = cao;
14    }
15 protected:
16     int chieurong;
17     int chieucao;
18 };
19 // day la lop ke thua: HinhChuNhat
20 class HinhChuNhat: public Hinh
21 {
22 public:
23     int tinhDienTich()
24     {
25         return chieurong * chieucao;
26     }
27 };
28 //=====
29 int main(void)
30 {
31     HinhChuNhat Hcn;
32     Hcn.setChieuRong(14);
33     Hcn.setChieuCao(30);
34     // in dien tich cua doi tuong.
35     cout << "Tong dien tich la: " << Hcn.tinhDienTich() << endl;
36     return 0;
37 }

```

Tong dien tich la: 420

21

## CHƯƠNG 1. GIỚI THIỆU VỀ LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG

## 3. Một số khái niệm cơ bản

## ❖ 1.2.5. Sự đa hình - Polymorphism

- Tính đa hình xuất hiện khi có khái niệm kế thừa, đó là khả năng thực hiện một phương thức có cùng tên trong các lớp con.
- Thực hiện bởi:
  - Định nghĩa lại
  - Nạp chồng hàm

22

## CHƯƠNG 1. GIỚI THIỆU VỀ LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG

### 4. Các bước thiết kế chương trình OOP

#### ❖ Các bước chính:

- Xác định các dạng đối tượng (lớp)
- Tìm dữ liệu dùng chung, chia sẻ
- Xác định lớp cơ sở dựa vào dữ liệu dùng chung
- Xây dựng lớp dẫn xuất từ lớp cơ sở

#### ❖ 1.4. Ưu điểm của lập trình hướng đối tượng

- Loại bỏ các đoạn mã lặp lại
- Tạo ra các chương trình an toàn, bảo mật
- Dễ dàng mở rộng và nâng cấp
- Rút ngắn thời gian xây dựng hệ thống
- Tăng năng suất và hiệu quả hơn
- Chương trình được thiết kế theo đúng qui trình

23

## CHƯƠNG 1. GIỚI THIỆU VỀ LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG

### 4. Các bước thiết kế chương trình OOP

#### ❖ Một số ngôn ngữ lập trình hướng đối tượng

- Có thể chia thành 2 loại:
  - Ngôn ngữ hỗ trợ hướng đối tượng: Pascal, C++, VB...
  - Ngôn ngữ hướng đối tượng: SmallTalk, JAVA...
- Một số ngôn ngữ lập trình hướng đối tượng hiện nay: Visual C++, VB.NET, C#, JAVA...

#### ❖ Ứng dụng của lập trình hướng đối tượng

- Dùng để phát triển phần mềm trong nhiều lĩnh vực khác nhau, ví dụ: hệ điều hành Windows...
- Lĩnh vực chính:
  - Hệ thống thời gian thực
  - Cơ sở dữ liệu hướng đối tượng
  - Hệ siêu văn bản, đa phương tiện
  - Trí tuệ nhân tạo
  - Lập trình song song, mạng nơron ...

24

## CHƯƠNG 2 - MỘT SỐ MỞ RỘNG CỦA C++ SO VỚI C

### 2.1. Lịch sử của C++

- C++ được xây dựng trên nền của C
- C++ được đưa ra bởi Bjarne Stroustrup, phiên bản đầu tiên ra mắt năm 1980, với tên "C with class". Phiên bản thương mại đầu tiên vào năm 1986.
- Ưu điểm của C++: Hỗ trợ lập trình hướng đối tượng, có nhiều thư viện mẫu chuẩn.

### 2.2. Một số mở rộng của C++ so với C

#### ❖ Lời chú thích

- Có hai cách chú thích:
- Cách 1: chú thích trên nhiều dòng

```
/*
    Nội dung ghi chú
*/
```

- Ví dụ:

```
1  #include <iostream>
2  using namespace std;
3
4
5  /*
6  Đây là hàm main của chương trình
7  Hàm này sẽ chạy đầu tiên trong chương trình
8  Cho dù bạn đặt nó ở vị trí nào
9  */
10 void main()
11 {
12     // Code
13 }
```

25

## CHƯƠNG 2 - MỘT SỐ MỞ RỘNG CỦA C++ SO VỚI C

- Cách 2: chú thích trên 1 dòng

Dùng //

- Ví dụ:

```
1  #include <iostream>
2  using namespace std;
3
4  void main()
5  {
6      int tuoi = 20; // Khai báo và gán giá trị cho biến tuoi
7
8      cout << tuoi; // In giá trị biến tuoi ra màn hình
9  }
```

26

## CHƯƠNG 2 - MỘT SỐ MỞ RỘNG CỦA C++ SO VỚI C

### ❖ Từ khóa mới

- Một số từ khóa mới của C++:

<b>asm</b>	<b>catch</b>	<b>class</b>
<b>delete</b>	<b>friend</b>	<b>inline</b>
<b>new</b>	<b>operator</b>	<b>private</b>
<b>protected</b>	<b>public</b>	<b>template</b>
<b>this</b>	<b>throw</b>	<b>try</b>
<b>virtual</b>		

27

## CHƯƠNG 2 - MỘT SỐ MỞ RỘNG CỦA C++ SO VỚI C

### ❖ Khai báo biến

- C++ cho phép khai báo biến:
  - Tại bất cứ đâu
  - Trước khi sử dụng
- Có hiệu lực trong phạm vi chương trình kể từ vị trí nó xuất hiện
- Ví dụ:

```
for(int i=0; i < n-1; i++)
    for(int j = i+1; j<n; j++)
        if(x[i] > x[j])
        {
            float tg = x[i] ;
            x[i] = x[j] ;
            x[j] = tg;
        }
```

28

## CHƯƠNG 2 - MỘT SỐ MỞ RỘNG CỦA C++ SO VỚI C

## ❖ Chuyển đổi và ép kiểu

- C++ cho phép chuyển kiểu rộng rãi:
  - Khi gán giá trị số vào biến kiểu khác
  - Các kiểu số khác nhau trong cùng 1 biểu thức
- Ép kiểu:
  - **Ép kiểu theo kiểu cũ:**  
`(kiểu) biểu thức`
  - Ví dụ:  
`myInt = (int) myFloat`
  - **Ép kiểu theo kiểu mới:**  
`kiểu (biểu thức)`
  - Ví dụ:  
`myInt = int (myFloat)`
  - Ví dụ:  
`for(i=1; i<=n; i++) s += float(1)/i;`

29

## CHƯƠNG 2 - MỘT SỐ MỞ RỘNG CỦA C++ SO VỚI C

## ❖ Vào ra trong C++

- Cú pháp:
  - **Xuất dữ liệu:**  
`cout << bt1 << ... << btn;`
  - **Nhập dữ liệu:**  
`cin >> biến1 >> biến2 >> ... >> biếnn;`
- Chú ý:
  - Khi dùng `cout`, `cin`, phải có khai báo `#include <iostream.h>`
  - Dùng `cin.ignore(1)` để bỏ kí tự '\n' khi nhập chuỗi ký tự.
  - **Nhập một chuỗi ký tự** (kể cả dấu cách) có độ dài không quá `dmax`, ta dùng:  
`cin.getline(biến, dmax);`
- Ví dụ:
 

```
char hoten[30];
...
cout << "Ho va ten: ";
cin.ignore(1);
cin.getline(hoten, 30);
```

30

## CHƯƠNG 2 - MỘT SỐ MỞ RỘNG CỦA C++ SO VỚI C

- Để quy định **số thực** được in ra có đúng **p** chữ số sau dấu chấm thập phân, ta dùng:

```
cout << setiosflags(ios::showpoint) << setprecision(p);
```

- Ví dụ:

```
cout << setiosflags(ios::showpoint) << setprecision(3);
cout << 1.23456;
```

sẽ đưa ra số 1.235 (có đúng 3 chữ số lẻ).

- Để quy định độ rộng tối thiểu là **w** vị trí cho giá trị (nguyên, thực, chuỗi), ta dùng hàm **setw(w)** (trong thư viện "iomanip.h"), hàm này cần được đặt sau toán tử xuất và nó chỉ có hiệu lực cho 1 giá trị được in gần nhất

- Ví dụ:

```
cout << "\n" << setw(3) << stt << setw(25) << hoten <<
        setw(10) << ngaysinh << setw(5) << diem;
```

31

## CHƯƠNG 2 - MỘT SỐ MỞ RỘNG CỦA C++ SO VỚI C

❖ **Cấp phát và giải phóng bộ nhớ**

- Vẫn có thể dùng malloc(), calloc(), free() như đã biết trong C.
- Ngoài ra, C++ sử dụng thêm hai toán tử **new** và **delete**:
- **new**: để cấp phát bộ nhớ
- **Dạng 1**: cấp phát bộ nhớ cho 1 biến

```
new <tên_kiểu_dữ_liệu>
```

- Ví dụ:

```
int *p;
p = new int;
```

sẽ cấp phát một vùng nhớ cho một phần tử có kiểu int và gán cho p địa chỉ tương ứng.

- **Dạng 2**: cấp phát bộ nhớ cho n phần tử.

```
new <tên_kiểu_dữ_liệu> [ n ] :
```

Trong đó n là một biểu thức nguyên không âm, khi đó toán tử **new** xin cấp phát một vùng nhớ để chứa n thành phần có kiểu dữ liệu <tên\_kiểu\_dữ\_liệu> và trả lại con trỏ đến đầu vùng nhớ đó nếu cấp phát thành công.

32



## CHƯƠNG 2 - MỘT SỐ MỞ RỘNG CỦA C++ SO VỚI C

## ▪ Ví dụ:

```
int *p;
p = new int [100];
```

sẽ cấp phát vùng nhớ cho một mảng chứa 100 phần tử kiểu int và đặt địa chỉ đầu của vùng nhớ cho biến p.

▪ **delete**: để giải phóng bộ nhớ đã được cấp phát bởi new

**Cú pháp:** `delete <con_trỏ>`

## ▪ Ví dụ:

```
delete p;
```

33

## CHƯƠNG 2 - MỘT SỐ MỞ RỘNG CỦA C++ SO VỚI C

▪ **Ví dụ 2.1:** Cấp phát bộ nhớ động cho mảng 1 chiều:

```
#include <iostream>
#include <stdlib.h>
#include <conio.h>
using namespace std;
int main()
{
    int n;
    do
    {
        cout<<"Nhập vào số phần tử của mảng:";
        cin>>n;
    } while(n <= 0);
    int *p = new int[n];
    if (p == NULL)
    {
        cout<<"Không còn bộ nhớ để cấp phát\n";
        return 0;
    }
}
```

34

## CHƯƠNG 2 - MỘT SỐ MỞ RỘNG CỦA C++ SO VỚI C

```

for(int i=0; i<n; i++)
    p[i] = rand()%100; //Tao cac so ngau nhien tu 0 den 99
cout<<"Mang truoc khi sap xep\n";
for(i=0; i<n; i++) cout<<p[i]<<" ";
for(i=0; i<n-1; i++)
    for(int j=i+1; j<n; j++)
        if (p[i] > p[j])
        {
            int temp = p[i];
            p[i] = p[j];
            p[j] = temp;
        }
cout<<"\nMang sau khi sap xep tang dan\n";
for(i=0; i<n; i++) cout << p[i] << " ";
delete p;
system("pause"); return 0;
}

```

35

## CHƯƠNG 2 - MỘT SỐ MỞ RỘNG CỦA C++ SO VỚI C

▪ **Ví dụ 2.2:** Cấp phát bộ nhớ động cho mảng 2 chiều:

```

#include <iostream>
#include <iomanip>
#include <conio.h>
using namespace std;
int main()
{ int m, n, i, j; float **p;
  do
  { cout<<"Nhap vao so hang cua ma tran:"; cin >> m;
    cout<<"Nhap vao so cot cua ma tran:"; cin >> n;
  } while (m <= 0 || n <= 0);
  //Cap phat bo nho dong
  p = new float *[m];
  if (p == NULL)
  { cout<<"Khong con bo nho de cap phat\n";
    return 0; }
  for(i = 0; i < m; i++) p[i] = new float [n];
}

```

36

## CHƯƠNG 2 - MỘT SỐ MỞ RỘNG CỦA C++ SO VỚI C

```

//Nhap ma tran
for(i=0; i<m; i++)
    for(j = 0; j<n; j++)
        { cout << "Nhap phan tu  [" << i << "][" << j << "] = ";
          cin >> p[i][j]; }
//In ma tran
for(i=0; i<m; i++)
    { for(j = 0; j<n; j++)
      cout << setw(8) << p[i][j];
      cout << "\n"; }
//Giai phong vung nho
for(i=0; i<m; i++) delete p[i];
delete p;
system("pause");
return 0;
}

```

37

## CHƯƠNG 2 - MỘT SỐ MỞ RỘNG CỦA C++ SO VỚI C

## ❖ Tham chiếu

- Tham chiếu giống như một bí danh của biến.
- Cú pháp:

**<kiểu dữ liệu> &<biến tham chiếu> = <biến>;**

- Ví dụ:

```

int    a,    &x = a;
x = 1;           // a=1
cout << x;       //in ra a
x++;            //a=2
a++;            //a=3

```

38

## CHƯƠNG 2 - MỘT SỐ MỞ RỘNG CỦA C++ SO VỚI C

- Tham chiếu cho phép hàm thao tác trực tiếp trên biến được truyền vào.

- Ví dụ 2.3:**

```
#include <iostream>
#include <conio.h>
using namespace std;
void Hoan_vi(int &X, int &Y)
{
    int Temp=X;
    X=Y;
    Y=Temp;
}
//-----
void main()
{
    int X = 10, Y = 5;
    cout<<"Truoc khi hoan vi: X = "<<X<<",Y = "<<Y<<endl;
    Hoan_vi(X,Y);
    cout<<"Sau khi hoan vi: X = "<<X<<",Y = "<<Y<<endl;
    getch(); }
```

39

## CHƯƠNG 2 - MỘT SỐ MỞ RỘNG CỦA C++ SO VỚI C

- Giá trị trả về của hàm là một tham chiếu

- Dạng hàm:**

```
<kiểu dữ liệu> &<hàm> ( <danh sách tham số hình thức> )
{ ...
    return <biến có phạm vi toàn cục>;
}
```

- Ví dụ 2.4:**

```
#include <iostream.h>
#include <conio.h>
int a[5];
int &f(int *d, int i) //Hàm trả về một tham chiếu
{
    return d[i];
}
```

40

## CHƯƠNG 2 - MỘT SỐ MỞ RỘNG CỦA C++ SO VỚI C

```

void main( )
{
    for( int i = 0; i < 5; i++)
    {
        cout << "a[" << i+1 << "] = " ;
        cin >> f(a,i);
    }
    cout << "Mang sau khi nhap vao\n";
    for(i = 0; i < 5; i++) cout << a[i] << " ";
    getch( );
}

```

41

## CHƯƠNG 2 - MỘT SỐ MỞ RỘNG CỦA C++ SO VỚI C

## ❖ Hằng tham chiếu

## ▪ Cú pháp:

**const <kiểu dữ liệu> &<hằng tham chiếu> = <Biến/hằng>;**

## ▪ Ví dụ:

```

int n = 10;
const int &m = n;

```

## ❖ Toán tử phạm vi

- Trong trường hợp biến toàn cục và biến cục bộ của các hàm cùng tên với nhau, chúng ta muốn truy cập biến cần thao tác thì cần xác định biến nào là biến toàn cục, biến nào là biến cục bộ. C++ thêm toán tử phạm vi "::" vào trước tên biến, chương trình dịch sẽ xác định biến đó là biến toàn cục

42

## CHƯƠNG 2 - MỘT SỐ MỞ RỘNG CỦA C++ SO VỚI C

### ▪ Ví dụ 2.5:

```
#include <iostream.h>
int    x = 15;
main()
{
    int    x =20;           // biến cục bộ
    cout<<"x = "<<x;
    cout<<"\nx = "<< ::x ; // biến toàn cục
}
```

Sẽ in ra là:

```
x = 20
x = 15
```

43

## CHƯƠNG 2 - MỘT SỐ MỞ RỘNG CỦA C++ SO VỚI C

### ❖ **Nạp chồng hàm**

▪ Nạp chồng hàm là các hàm có cùng tên nhưng có các đối số khác nhau. Khi gặp hàm này, trình biên dịch gọi hàm dựa vào:

- Kiểu trả về của hàm
- Số lượng đối số
- Kiểu của đối số

44

## CHƯƠNG 2 - MỘT SỐ MỞ RỘNG CỦA C++ SO VỚI C

## ▪ Ví dụ 2.6:

Nạp chồng hàm tìm max của 2 số nguyên, của 3 số nguyên, của 2 số thực, của mảng các số nguyên:

```

1 #include <iostream>
2 #include <conio.h>
3 using namespace std;
4 int max(int, int);
5 int max(int, int, int);
6 float max(float, float);
7 int max(int, int*);
8 int main()
9 {
10     int x = 10, y = 5, z = -2;
11     float a = 2.5, b = 3.5;
12     int m[5] = {3, 6, 11, 7, 25};
13     cout << "max(x,y) : " << max(x,y) << "\n";
14     cout << "max(x,y,z) : " << max(x,y,z) << "\n";
15     cout << "max(a,b) : " << max(a,b) << "\n";
16     cout << "max(m) : " << max(5,m) << "\n";
17     getch();
18     system("pause");
19     return 0;
20 }

21 int max(int m, int n)
22 {
23     return (m > n ? m : n); }
24 //-----
25 int max(int m, int n, int p)
26 {
27     return max(max(m,n),p); }
28 //-----
29 float max(float m, float n)
30 {
31     return (m > n ? m : n); }
32 //-----
33 int max(int m, int *n)
34 {
35     int tg = n[0];
36     for(int i = 1; i < m; i++) tg = max(tg, n[i]);
37     return tg;
38 }
39

```

45

## CHƯƠNG 2 - MỘT SỐ MỞ RỘNG CỦA C++ SO VỚI C

## ❖ Định nghĩa chồng các toán tử

- a/ Tên hàm toán tử:
  - gồm từ khóa **operator** và tên phép toán.
- Ví dụ:
  - operator + (định nghĩa chồng phép +)
  - operator \* (định nghĩa chồng phép \*)
- b/ Các đối của hàm toán tử:
  - Với các phép toán có 2 toán hạng: Hàm toán tử cần có 2 đối.

46

## CHƯƠNG 2 - MỘT SỐ MỞ RỘNG CỦA C++ SO VỚI C

- Ví dụ 2.7: nạp chồng toán tử + hai phân số:

```

1 #include <iostream>
2 #include <conio.h>
3 using namespace std;
4 struct phanso
5 {
6     int ts, ms;
7 };
8 phanso operator + (phanso a, phanso b)
9 {
10     phanso c;
11     c.ts = a.ts * b.ms + b.ts * a.ms;
12     c.ms = a.ms * b.ms;
13     return c;
14 }
15 void inphanso (phanso a)
16 {
17     cout << a.ts << "/" << a.ms;
18 }
19 //-----
19 //-----
20 int main()
21 {
22     phanso x = {4, 3}, y = {2, 5}, z;
23     z = x + y; //tương đương với z = operator + (x, y);
24     inphanso(x); cout << " + "; inphanso(y); cout << " = "; inphanso(z);
25     //getch();
26     system("pause");
27     return 0;
28 }

```

47

## CHƯƠNG 2 - MỘT SỐ MỞ RỘNG CỦA C++ SO VỚI C

- Với các phép toán có 1 toán hạng: Hàm toán tử cần có 1 đối.
- Ví dụ 2.8: Hàm toán tử đổi dấu tất cả các phần tử của ma trận

```

#include <iostream.h>
#include <conio.h>
#include <iomanip.h>
struct MT
{
    float a[20][20];
    int m, n;
};
MT operator - (MT x)
{
    MT y;
    y.m = x.m;
    y.n = x.n;
    for(int i=0; i < x.m; i++)
        for(int j = 0; j < x.n; j++)
            y.a[i][j] = -x.a[i][j];
    return y;
}

```

48



## CHƯƠNG 2 - MỘT SỐ MỞ RỘNG CỦA C++ SO VỚI C

```

void main( )
{
    MT    a, b;
    do
    {
        cout<<"Nhap so hang ma tran: "; cin>>a.m;
        cout<<"Nhap so cot ma tran: ";  cin>>a.n;
    } while (a.m <= 0 || a.n <= 0);
    for(int i=0; i<a.m; i++)
        for(int j=0; j<a.n; j++)
        {
            cout<<"Nhap phan tu: ";
            cin>>a.a[i][j];
        }
    b = - a;    //tương đương với    b = operator - (a);
    cout<<"\nMa tran vao:\n";
    for(i=0; i<a.m; i++)
    {
        for(j=0; j<a.n; j++) cout<<setw(5)<<a.a[i][j];
        cout<<endl;
    }
}

```

49

## CHƯƠNG 2 - MỘT SỐ MỞ RỘNG CỦA C++ SO VỚI C

```

cout<<"\nMa tran doi dau:\n";
for(i=0; i<b.m; i++)
{
    for(j=0; j<b.n; j++) cout<<setw(5)<<b.a[i][j];
    cout<<endl;
}
getch( );
}

```

50

## CHƯƠNG 2 - MỘT SỐ MỞ RỘNG CỦA C++ SO VỚI C

## ❖ Tham số ngầm định trong lời gọi hàm

- Ngôn ngữ C++ có khả năng định nghĩa các giá trị tham số ngầm định cho các hàm. Bình thường khi gọi một hàm, chúng ta cần gửi một giá trị cho một tham số đã được định nghĩa trong hàm đó. Tuy nhiên, trong nhiều trường hợp chúng ta có thể dùng giá trị ngầm định cho tham số.

## ▪ Ví dụ 2.9:

```

1 #include <iostream>
2 #include <conio.h>
3 using namespace std;
4 void ham(int m, int n)
5 {
6     cout << "\nTham so thu nhât: " << m;
7     cout << "\nTham so thu hai: " << n;
8 }
9 int main( )
10 {
11     int x = 10, y = 20;
12     void ham(int = 0, int = 12); // đặt giá trị ngầm định cho tham số
13     ham(x,y);
14     ham(x);
15     ham( );
16     system("pause");
17     return 0;
18 }

```

## Kết quả:

Tham so thu nhât: 10  
 Tham so thu hai: 20  
 Tham so thu nhât: 10  
 Tham so thu hai: 12  
 Tham so thu nhât: 0  
 Tham so thu hai: 12

51

## CHƯƠNG 2 - MỘT SỐ MỞ RỘNG CỦA C++ SO VỚI C

## ❖ Hàm inline

- Khi gặp hàm inline, trình biên dịch sẽ không biên dịch hàm này thành một chương trình con riêng biệt, mà chèn thẳng vào các chỗ mà hàm này được gọi, như vậy chạy chương trình sẽ nhanh hơn.
- Chú ý:
  - Sử dụng hàm **inline** sẽ làm cho chương trình lớn lên vì trình biên dịch chèn đoạn chương trình vào các chỗ mà hàm này được gọi, do đó các hàm inline thường là các hàm nhỏ, ít phức tạp.
  - Các hàm **inline** phải được định nghĩa trước khi sử dụng.

52

## CHƯƠNG 2 - MỘT SỐ MỞ RỘNG CỦA C++ SO VỚI C

### ■ Ví dụ 2.10:

```

1 #include <iostream>
2 #include <conio.h>
3 using namespace std;
4 inline float mu3(float s)
5 {
6     return s * s * s;
7 }
8 //-----
9 void main()
10 {
11     cout<<"Nhập chiều dài cạnh hình lập phương:";
12     float d;
13     cin >> d;
14     cout << "The tích hình lập phương = " << mu3(d);
15     //getch();
16     system("pause");
17     return 0;
18 }
19
20

```

53

## CHƯƠNG 2 - MỘT SỐ MỞ RỘNG CỦA C++ SO VỚI C

### Bài tập chương 2

1. Làm lại các bài tập ở chương 1 với cout và cin
2. Cấp phát bộ nhớ động, nhập ma trận thực cấp  $m \times n$ :
  - Tìm phần tử lớn nhất
  - Sắp xếp tăng dần theo hướng từ trái sang phải, từ trên xuống dưới.
  - In ma trận sau khi đã sắp xếp
3. Xây dựng chương trình thao tác với phân số: nhập, in, tối giản, cộng, trừ, nhân, chia hai phân số (sử dụng chồng toán tử +, -, \*, /).
4. Xây dựng chương trình thao tác với vec tơ:
 

Nhập vec tơ, in vec tơ, tính tổng, tích vô hướng hai vectơ (sử dụng chồng toán tử +, \*)
5. Xây dựng chương trình thao tác với số phức: nhập, in số phức; cộng, trừ 2 số phức; cộng, trừ số phức với số thực (sử dụng chồng toán tử +, -)

54

## CHƯƠNG 3 - LỚP VÀ ĐỐI TƯỢNG

**3.1. Lớp****□ a. Định nghĩa lớp**

- Lớp được định nghĩa theo mẫu sau:

```
class <tên_lớp>
{
    [quyền truy nhập:]
    <khai báo các thành phần dữ liệu của lớp>
    [quyền truy nhập:]
    <khai báo các thành phần hàm của lớp>
};
```

**Trong đó:**

- <tên\_lớp>**: Do người dùng đặt, tuân theo các qui tắc về tên
- Ví dụ**: SinhVien, NGUOI, Hoa\_Don, phanso, Ma\_Tran...

55

## CHƯƠNG 3 - LỚP VÀ ĐỐI TƯỢNG

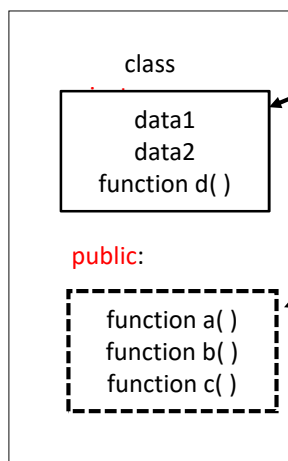
**[quyền truy nhập:]**

- Là khả năng truy nhập thành phần dữ liệu, có thể là **private**, hoặc **public**, hoặc **protected**, ngầm định là **private**:
- private**: Các thành phần private chỉ được sử dụng bên trong lớp (**trong thân các phương thức của lớp**).
- public**: Các thành phần public được sử dụng ở cả bên trong lẫn bên ngoài lớp.
- protected**: Các thành phần protected được sử dụng trong **lớp đó và các lớp con kế thừa**.

56

## CHƯƠNG 3 - LỚP VÀ ĐỐI TƯỢNG

## quyền truy nhập



Phần được khai báo với từ khóa **private** chỉ được truy nhập bởi các phương thức của cùng class

Phần được khai báo với từ khóa **public** có thể được truy nhập tại bất kỳ nơi nào trong chương trình

Thông thường, thành phần dữ liệu (member data) là **private**, và các hàm thành phần (member functions) là **public**

57

## CHƯƠNG 3 - LỚP VÀ ĐỐI TƯỢNG

- **Thành phần của lớp:** có thể là
  - Thành phần dữ liệu (member data)
  - Phương thức (hoặc hàm thành phần – member function).

- **Khai báo thành phần dữ liệu:**

**<kiểu dữ liệu > <tên\_thành\_phần>;**

- **Chú ý:** không được khởi tạo giá trị ban đầu

- **Ví dụ:**

```
char hoten[30];
int namsinh;
float diem;
```

58

### CHƯƠNG 3 - LỚP VÀ ĐỐI TƯỢNG

- Khai báo hàm thành phần:

- **Cách 1:** Khai báo bên trong lớp và định nghĩa ở bên ngoài lớp

```
<kiểu trả về > <tên lớp>::<tên_hàm>([đối số])
{
    // <thân hàm>
}
```

- **Cách 2:** định nghĩa ngay ở bên trong lớp.

- **Ví dụ 3.1:** Xây dựng lớp điểm trên mặt phẳng có tọa độ (x,y), các phương thức nhập điểm, in ra điểm, di chuyển điểm.

59

### CHƯƠNG 3 - LỚP VÀ ĐỐI TƯỢNG

```
#include <iostream.h>
#include <conio.h>
class diem
{
    private:
        float    x, y;
    public:
        void    nhap_diem();
        void    in_diem()    //Định nghĩa ở bên trong lớp
        {
            cout<<"Diem (" << x << ", " << y << ")";
        }
        void    di_chuyen(float    dx, float    dy);
};
//Định nghĩa các hàm thành phần ở bên ngoài lớp
void diem :: nhap_diem()
{
    cout << "Nhap hoành do, tung do cua diem: ";
    cin >> x >> y;
}
//-----
void diem :: di_chuyen(float    dx, float    dy)
{
    x += dx;
    y += dy;
}
```

60

## CHƯƠNG 3 - LỚP VÀ ĐỐI TƯỢNG

```

main()
{
    diem a;
    a.nhap_diem();
    a.in_diem();
    float dx, dy;
    cout<<"\ndx = "; cin>>dx;
    cout<<"dy = "; cin>>dy;
    a.di_chuyen(dx, dy);
    a.in_diem();
}

```

61

## CHƯƠNG 3 - LỚP VÀ ĐỐI TƯỢNG

▪ **Ví dụ 3.2:** Xây dựng lớp phân số có:

- Dữ liệu: tử số, mẫu số.
- Phương thức: nhập phân số, tối giản phân số, in phân số.

```

#include <iostream.h>
#include <conio.h>
#include <math.h>
class Phanso
{
    private:
        int ts, ms;
    public:
        voidNhapPS();
        voidToigian();
        voidInPS();
};

```

62

### CHƯƠNG 3 - LỚP VÀ ĐỐI TƯỢNG

```
//Định nghĩa hàm InPS ở bên
ngoài lớp
void Phanso :: InPS()
{
    cout << ts << "/" << ms;
}
// Định nghĩa hàm NhapPS ở bên
ngoài lớp
void Phanso::NhapPS()
{
    cout<<"Tu so: "; cin>>ts;
    do
    {
        cout<<"Mau so: "; cin>>ms;
    } while (ms == 0);
}

//Định nghĩa hàm Toigian
void
Phanso::Toigian_PS()
{
    int a, b;
    if(ts!=0)
    {
        a = abs(ts);
        b = abs(ms);
    } while (a != b)
    if (a > b)    a -= b;
    else    b -= a;
        ts = ts/a;
        ms = ms/a;
    }
```

63

### CHƯƠNG 3 - LỚP VÀ ĐỐI TƯỢNG

```
// Ham main
main()
{
    Phanso p;
    p.NhapPS();
    p.InPS();
    p.Toigian();
    p.InPS();
}
```

64



## CHƯƠNG 3 - LỚP VÀ ĐỐI TƯỢNG

## ❑ b. Đối tượng

- Cú pháp khai báo đối tượng:

`<tên_lớp> <tên_đối_tượng>;`

- Ví dụ: Phanso a, b;

## ❑ c. Truy xuất thành phần

- Truy xuất thành phần dữ liệu:

`<tên_đối_tượng>.<tên_tp_dữ_liệu>;`

- Nếu là con trỏ:

`<tên_con_trỏ>-><tên_tp_dữ_liệu>;`

- Truy xuất thành phần hàm:

`<tên_đối_tượng>.<tên_hàm>([ds đối số]);`

- Ví dụ: a.NhapPS(); a.InPS();

- Với con trỏ: <tên\_con\_trỏ> -> <tên\_hàm>([đối số]);

- Ví dụ:

```
Phanso *p;
p -> NhapPS(); p -> InPS();
```

65

## CHƯƠNG 3 - LỚP VÀ ĐỐI TƯỢNG

## 3.2. Các phương thức

- ❑ Một đối tượng thường có 4 kiểu phương thức cơ bản:

- Các phương thức khởi tạo (Constructor)
- Các phương thức truy vấn (Queries)
- Các phương thức cập nhập (Updates)
- Các phương thức hủy (Destructor)

- ❑ a. Hàm khởi tạo (constructor)

- Khai báo hàm tạo:

`<tên_lớp>([ds tham số]);`

- Định nghĩa hàm tạo ở ngoài lớp:

```
<tên_lớp>::<tên_lớp>([ds tham số])
{
    //thân hàm
}
```

66

## CHƯƠNG 3 - LỚP VÀ ĐỐI TƯỢNG

## ▪ Ví dụ: Hàm tạo:

```
diem(float tx, float ty);
```

## ▪ Định nghĩa ở bên ngoài lớp:

```
diem::diem(float tx, float ty)
{
    x = tx;
    y = ty;
}
```

## ▪ Như vậy hàm khởi tạo:

- Có với mọi lớp
- Tên hàm giống tên lớp
- Không có kiểu nên không cần khai báo kiểu trả về
- Không có giá trị trả về
- Nếu không xây dựng thì chương trình tự động sinh hàm khởi tạo mặc định
- Được gọi tự động khi khai báo thể hiện của lớp

67

## CHƯƠNG 3 - LỚP VÀ ĐỐI TƯỢNG

## ▪ Một số hàm khởi tạo:

- Hàm khởi tạo mặc định (default constructor): Hàm tạo mặc định do chương trình dịch cung cấp khi trong khai báo lớp không có định nghĩa hàm tạo nào.
- Hàm tạo có các giá trị ngầm định cho các tham số.

## ▪ Ví dụ:

```
diem(float tx = 0, float ty = 0)
{
    x = tx; y = ty;
};
```

## ▪ Hàm khởi tạo sao chép (copy constructor)

## Khai báo:

`<tên_lớp> (const <tên_lớp> &<tên_tham_số>);`

Đối tượng mới sẽ là bản sao của đối tượng đã có.

## ▪ Ví dụ:

```
diem (const diem &p)
{
    x = p.x ; y = p.y;
}
```

68

## CHƯƠNG 3 - LỚP VÀ ĐỐI TƯỢNG

### ❑ b. Hàm hủy - Destructor

#### ▪ Khai báo:

~ <tên\_lớp>( );

#### ▪ Chức năng: Hủy bỏ, giải phóng các đối tượng khi nó hết phạm vi tồn tại

#### ▪ Như vậy hàm hủy:

- Không có đối số
- Không có giá trị trả về
- Không định nghĩa lại
- Trùng tên với lớp và có dấu ~ ở trước
- Thực hiện một số công việc trước khi hệ thống giải phóng bộ nhớ
- Chương trình dịch tự động sinh hàm hủy mặc định

69

## CHƯƠNG 3 - LỚP VÀ ĐỐI TƯỢNG

### 3.3. Mảng và con trỏ của đối tượng

#### ❑ Khai báo mảng các đối tượng:

<tên\_lớp> <tên\_mảng>[số phần tử];

Ví dụ: SinhVien sv[50]; Phanso p[8];

#### ❑ Khai báo con trỏ đối tượng:

<tên\_lớp> \* <tên\_con\_trỏ>;

- Ví dụ: SinhVien \*sv1 ; Phanso \*p1;

70

## CHƯƠNG 3 - LỚP VÀ ĐỐI TƯỢNG

## 3.4. Hàm bạn và lớp bạn

## ❑ a. Hàm bạn

Hàm bạn của một lớp là hàm không phải là thành phần của lớp, nhưng có khả năng truy xuất đến mọi thành phần của đối tượng

## ❑ Cú pháp khai báo hàm bạn:

`friend <kiểu trả về> <tên hàm>(<tham số>);`

Sau đó định nghĩa hàm ở ngoài lớp như các hàm tự do khác.

- **Ví dụ 3.3:** Xây dựng hàm tự do kiểm tra xem hai điểm có trùng nhau, là bạn của lớp **diem**:

71

## CHƯƠNG 3 - LỚP VÀ ĐỐI TƯỢNG

```

1 #include <iostream>
2 #include <conio.h>
3 using namespace std;
4 class diem
5 {
6     private:
7         float x, y;
8     public:
9         diem(float tx = 0, float ty = 0)
10        {
11            x = tx; y = ty;
12        }
13        friend int trung(diem, diem);
14    };
15 int trung(diem p, diem q)
16 {
17     if(p.x == q.x && p.y == q.y)
18         return 1;
19     else return 0;
20 }
21 //-----
22 void main( )
23 {
24     diem a(1,0), b(1), c;
25     if(trung(a,b)) cout<<"a trung với b";
26     else cout<<"na không trung với b";
27     if(trung(a,c))cout<<"na trung với c";
28     else cout<<"na không trung với c";
29     system("pause");
30 }

```

Kết quả??

72

## CHƯƠNG 3 - LỚP VÀ ĐỐI TƯỢNG

▪ **Nhận xét:**

- Hàm bạn không phải là hàm thành viên nên không bị ảnh hưởng của từ khoá truy xuất
- Không hạn chế số lượng hàm bạn
- Hàm bạn của một lớp có thể là hàm tự do
- Hàm bạn của một lớp có thể là hàm thành phần của một lớp khác
- Một hàm có thể là bạn của nhiều lớp khác nhau.

73

## CHƯƠNG 3 - LỚP VÀ ĐỐI TƯỢNG

□ **b. Lớp bạn:**

- Lớp A là lớp bạn của lớp B nếu trong B có chứa khai báo:  
**friend class A;**
- Nếu A là lớp bạn của B thì mọi hàm thành phần của A sẽ trở thành hàm bạn của B.
- **Ví dụ 3.4:** Chương trình xây dựng lớp ma trận vuông, lớp vector là bạn (friend) của nhau, tính tích vô hướng của 2 vector, tích của 2 ma trận, tích ma trận với vector, tích của vector với ma trận:

```
#include <iostream.h>
#include <stdio.h>
#include <conio.h>
#define MAX 50
```

74

## CHƯƠNG 3 - LỚP VÀ ĐỐI TƯỢNG

```

class ma_tran;
class vec_to;
class ma_tran
{
    int n;
    float a[MAX][MAX];
public:
    friend class vec_to;
    ma_tran() { n=0; }
    ma_tran(int m);
    ~ma_tran() { };
    void nhap_mt();
    void in_mt();
    ma_tran tich(const
ma_tran &);
    vec_to tich(const
vec_to &);
};

class vec_to
{
    int n;
    float x[MAX];
public:
    friend class ma_tran;
    vec_to() { n=0; }
    vec_to(int m);
    ~vec_to() { };
    void nhap_vt();
    void in_vt();
    vec_to tich(const ma_tran
&);
    float tich(const vec_to
&);
};

```

75

## CHƯƠNG 3 - LỚP VÀ ĐỐI TƯỢNG

```

ma_tran::ma_tran(int m)
{
    n = m;
    for(int i=0; i<n; i++)
        for(int j=0; j<n; j++)
            a[i][j] = 0;
}
//-----
void ma_tran::nhap_mt()
{
    for(int i=0; i<n; i++)
        for(int j=0; j<n; j++)
        {
            cout<<"Nhap phan tu hang "<<i+1<<" cot
"<<j+1<<": ";
            cin>>a[i][j];
        }
}

```

76

## CHƯƠNG 3 - LỚP VÀ ĐỐI TƯỢNG

```

vec_to::vec_to(int m)
{
    n = m;
    for(int i=0; i<n; i++) x[i] = 0;
}
//-----
void vec_to::nhap_vt()
{
    for(int i=0; i<n; i++)
    {
        cout<<"Nhap phan tu thu "<<i+1<<" : ";
        cin>>x[i];
    }
}

```

77

## CHƯƠNG 3 - LỚP VÀ ĐỐI TƯỢNG

```

void ma_tran::in_mt()
{
    for(int i=0; i<n; i++)
    {
        for(int j=0; j<n; j++) printf("%8.2f",a[i][j]);
        printf("\n");
    }
}
//-----
void vec_to::in_vt()
{
    for(int i=0; i<n; i++) printf("%8.2f",x[i]);
}

```

78

## CHƯƠNG 3 - LỚP VÀ ĐỐI TƯỢNG

```

ma_tran ma_tran::tich(const ma_tran &b)
{
    ma_tran c;
    c.n = n;
    for(int i=0; i<n; i++)
        for(int j=0; j<n; j++)
        {
            c.a[i][j] = 0;
            for(int k=0; k<n; k++)
                c.a[i][j] += a[i][k]*b.a[k][j];
        }
    return c;
}

```

79

## CHƯƠNG 3 - LỚP VÀ ĐỐI TƯỢNG

```

vec_to ma_tran::tich(const vec_to &y)
{
    vec_to z;
    int i,j;
    z.n = n;
    for(i=0; i<n; i++)
    {
        z.x[i] = 0;
        for(j=0; j<n; j++)
            z.x[i] += a[i][j]*y.x[j];
    }
    return z;
}

```

80



## CHƯƠNG 3 - LỚP VÀ ĐỐI TƯỢNG

```

vec_to  vec_to::tich(const  ma_tran  &b)
{
    vec_to  z;
    int i,j;
    for(j=0; j<n; j++)
    {
        z.x[j] = 0;
        for(i=0; i<n; i++)
            z.x[j] += b.a[i][j]*x[i];
    }
    z.n = n;
    return  z;
}

```

81

## CHƯƠNG 3 - LỚP VÀ ĐỐI TƯỢNG

```

float  vec_to::tich(const
vec_to  &y)
{
    float  tg = 0;
    for(int i=0; i<n; i++)
        tg += x[i] * y.x[i];
    return  tg;
}

```

82

## CHƯƠNG 3 - LỚP VÀ ĐỐI TƯỢNG

```

void main()
{   int  n;
    clrscr();          //hoặc system("cls") với #include
    <stdlib.h>
    do
    {   cout<<"Nhập cap ma tran va vecto: ";
    cin >> n;
        } while (n <= 0 || n>MAX);
    ma_tran  a(n), b(n), c(n);
    vec_to   x(n), y(n);
    cout<<"Nhập ma tran A:\n"; a.nhap_mt();
    cout<<"\nNhập ma tran B:\n"; b.nhap_mt();
    cout<<"\nNhập ma tran C:\n"; c.nhap_mt();
    cout<<"\nNhập vecto X:\n"; x.nhap_vt();
    cout<<"\nNhập vecto Y:\n"; y.nhap_vt();

```

83

## CHƯƠNG 3 - LỚP VÀ ĐỐI TƯỢNG

```

ma_tran  d = a.tich(b);
vec_to   u = a.tich(x);
vec_to   v = y.tich(c);
float     s = x.tich(y);
clrscr();
cout<<"\nMa tran A:\n"; a.in_mt();
cout<<"\nMa tran B:\n"; b.in_mt();
cout<<"\nMa tran C:\n"; c.in_mt();
cout<<"\nVecto X:\n"; x.in_vt();
cout<<"\n\nVecto Y:\n"; y.in_vt();
cout<<"\n\nMa tran tich D = A*B:\n"; d.in_mt();
cout<<"\nVecto tich U = A*X:\n"; u.in_vt();
cout<<"\n\nVecto tich V = Y*C:\n"; v.in_vt();
cout<<"\n\nTich vo huong X*Y = : "<< s;
    getch(); // hoặc system("pause") với #include
    <stdlib.h>
}

```

84

## CHƯƠNG 3 - LỚP VÀ ĐỐI TƯỢNG

```

ma_tran d = a.tich(b);
vec_to u = a.tich(x);
vec_to v = y.tich(c);
float s = x.tich(y);
clrscr();
cout<<"\nMa tran A:\n"; a.in_mt();
cout<<"\nMa tran B:\n"; b.in_mt();
cout<<"\nMa tran C:\n"; c.in_mt();
cout<<"\nVecto X:\n"; x.in_vt();
cout<<"\n\nVecto Y:\n"; y.in_vt();
cout<<"\n\nMa tran tich D = A*B:\n"; d.in_mt();
cout<<"\nVecto tich U = A*X:\n"; u.in_vt();
cout<<"\n\nVecto tich V = Y*C:\n"; v.in_vt();
cout<<"\n\nTich vo huong X*Y = : "<< s;
getch(); // hoặc system("pause") với #include
<stdlib.h>
}

```

85

## CHƯƠNG 3 - LỚP VÀ ĐỐI TƯỢNG

**3.5. Con trỏ this**

- Con trỏ **this** được dùng để xác định địa chỉ của đối tượng dùng làm tham số ngầm định cho hàm thành phần. Như vậy có thể truy cập đến các thành phần của đối tượng gọi hàm thành phần gián tiếp thông qua **this**.

- Ví dụ:**

```

void diem :: nhap_diem( )
{
    cout << "Nhap hoành đo va tung đo cua diem: "
         << endl;
    cin >> x >> y;
}

```

- Các thuộc tính viết trong phương thức trên được hiểu là thuộc một đối tượng do con trỏ **this** trỏ tới.

86

### CHƯƠNG 3 - LỚP VÀ ĐỐI TƯỢNG

- Như vậy phương thức **nhap\_diem( )** có thể viết một cách tường minh như sau:

```
void diem::nhap_diem()
{
    cout << "Nhap hoành do va tung do cua diem: "
    cin >> this -> x >> this -> y;
}
```

87

### CHƯƠNG 3 - LỚP VÀ ĐỐI TƯỢNG

#### 3.5. Thành phần tĩnh

##### a. Dữ liệu tĩnh

- Là thành phần dữ liệu của lớp nhưng không gắn cụ thể với đối tượng nào
- Dùng chung cho toàn bộ lớp
- Các đối tượng của lớp đều dùng chung thành phần tĩnh này

##### ❑ Khai báo:

**static <kiểu dữ liệu> <tên thành phần>;**

##### ❑ Ví dụ 3.5:

Tạo thành phần dữ liệu đếm các phân số trong lớp Phanso.

```
#include <iostream.h>
#include <conio.h>
class Phanso
{
    int ts, ms;
    static int dem;
```

88

## CHƯƠNG 3 - LỚP VÀ ĐỐI TƯỢNG

```

public:
    Phanso(int    m = 0, int    n = 1)
    {
        ts = m;
        ms = n;
        dem++;
    }
    static int    So_PS()
    { return    dem; }
};
//-----
int    Phanso :: dem = 0;    //Khởi tạo giá trị

```

89

## CHƯƠNG 3 - LỚP VÀ ĐỐI TƯỢNG

```

void    main()
{
    Phanso    p(1,2), q(3,4), r;
    clrscr();
    cout<<"So cac phan so la: " <<Phanso::So_PS();
    cout<<"So cac phan so la: " <<r.So_PS();
    getch();
}

```

**☐ Truy xuất thành phần dữ liệu tĩnh:**

- Theo đối tượng, VD: r.So\_PS();
- Theo phương thức, VD: Phanso :: So\_PS();

90

## CHƯƠNG 3 - LỚP VÀ ĐỐI TƯỢNG

❑ **Chú ý:**

- Thành phần dữ liệu tĩnh tồn tại ngay khi chưa có đối tượng nào.
- Phải được khởi tạo trước khi đối tượng phát sinh
- Phải khởi tạo ngoài mọi hàm theo cú pháp:

**<kiểu dl> <tên lớp>::<tên t/phần d/liệu> = <giá trị>;**

- Ví dụ: `int Phanso :: dem = 0;`

**b. Phương thức tĩnh**

- Là hàm thành phần của lớp nhưng không gắn với đối tượng cụ thể nào
- Dùng để thao tác chung cho lớp
- Trong thân hàm không có đối tượng ẩn

❑ **Khai báo:**

**static <kiểu d/liệu trả về> <tên hàm> (tham số);**

91

## CHƯƠNG 3 - LỚP VÀ ĐỐI TƯỢNG

❑ **Ví dụ 3.6:**

Xây dựng lớp sinh viên có thành phần tĩnh là số sinh viên của lớp, phương thức tĩnh in ra số sinh viên hiện có.

```
#include <iostream.h>
class lop_sv
{
    static int so_sv;
public:
    lop_sv() {so_sv++;}           //Hàm tạo
    ~lop_sv() {so_sv--;}         //Hàm hủy
    static void in_so_sv()
    { cout<<"\nSố sinh viên hiện tại là " << so_sv; }
};
int lop_sv::so_sv = 0;
```

92

## CHƯƠNG 3 - LỚP VÀ ĐỐI TƯỢNG

```

main()
{
    lop_sv a[5], b, c;
    a[0].in_so_sv();
    lop_sv *d = new lop_sv ;
    a[1].in_so_sv(); d -> in_so_sv();
    b.~lop_sv(); c.~lop_sv(); a[4].~lop_sv();
    a[2].in_so_sv();
}

```

The screenshot shows a command prompt window with the title "E:\Bai\_Giang\_LTHDT\_C++\TP\_static\Lop\_". The output of the program is as follows:

```

So sinh vien hien tai la 7
So sinh vien hien tai la 8
So sinh vien hien tai la 8
So sinh vien hien tai la 5Press any

```

93

## CHƯƠNG 3 - LỚP VÀ ĐỐI TƯỢNG

**3.6. Đối tượng hằng****a. Đối tượng hằng:**

Một đối tượng có thể được khai báo là hằng bằng cách dùng từ khóa const.

**❑ Ví dụ:**

```
const diem d = diem(0, 0);
```

**b. Phương thức hằng:**

Là hàm thành phần của lớp nhưng không có khả năng thay đổi thành phần dữ liệu trong đối tượng.

**❑ Khai báo:**

**<kiểu d/liệu trả về> <tên hàm> (tham số) const ;**

**❑ Định nghĩa:**

**<kiểu d/liệu trả về> <tên lớp>::<tên hàm> (tham số) const  
{ //thân hàm }**

**❑ VD:**

```

void Phanso::InPS() const
{
    cout << ts << "/" << ms; }

```

94

## CHƯƠNG 3 - LỚP VÀ ĐỐI TƯỢNG

**3.7. Thành phần đối tượng**

Thành phần đối tượng là thành phần dữ liệu của lớp có kiểu là một lớp khác.

❑ **Khai báo:**

**<tên lớp> <tên thành phần dữ liệu> ;**

❑ **Ví dụ:**

Thành phần *ngayvt* của lớp *hoadon* là đối tượng của lớp *ngay*.

```
class ngay
{
private:
    int ng, th, nm;
public:
    void nhap_ngay();
    void in_ngay();
};
```

95

## CHƯƠNG 3 - LỚP VÀ ĐỐI TƯỢNG

```
class hoadon
{
private:
    char mavt[5];
    char tenvt[30];
    int loai;
    ngay ngayvt;
    float soluong, dongia, thanhtien;
public:
    void nhap_hd();
    void in_hd();
};
```

96



## CHƯƠNG 3 - LỚP VÀ ĐỐI TƯỢNG

**Bài tập chương 3****Bài 1.** Xây dựng lớp phân số gồm các thành phần:

- Dữ liệu: tử số, mẫu số
- Phương thức: nhập, in, tối giản, so sánh nhỏ hơn giữa 2 phân số

Hàm main:

- Nhập mảng có n phân số ( $n \leq 10$ )
- Sắp xếp mảng phân số theo thứ tự giảm dần
- In mảng sau khi xếp

**Bài 2.** Xây dựng lớp số phức gồm các thành phần:

- Dữ liệu: phần thực, phần ảo
- Phương thức: nhập, in, tính trị tuyệt đối của số phức, tổng, hiệu 2 số phức.

Hàm main:

- Nhập 2 số phức
- Tính và in tổng, hiệu hai số phức

97

## CHƯƠNG 3 - LỚP VÀ ĐỐI TƯỢNG

**Bài tập chương 3****Bài 3.** Xây dựng lớp vectơ gồm các thành phần:

- Dữ liệu: số phần tử, mảng các phần tử
- Phương thức: nhập, in, tổng 2 vectơ, tích vô hướng 2 vectơ

Hàm main:

- Nhập 2 vectơ a, b
- Tính và in  $a+b$ ,  $a*b$

**Bài 4.** Xây dựng lớp ma trận vuông gồm các thành phần:

- Dữ liệu: số hàng ma trận vuông, mảng các phần tử ma trận.
- Phương thức: nhập, in, kiểm tra ma trận có là đơn vị không.

Hàm main:

- Nhập ma trận vuông
- Thông báo có là ma trận đơn vị không
- In ma trận vuông

98

**CHƯƠNG 3 - LỚP VÀ ĐỐI TƯỢNG****Bài tập chương 3**

**Bài 5.** Xây dựng lớp sinh viên gồm các thành phần:

- Dữ liệu: họ tên, ngày sinh, giới tính, lớp, điểm toán, lý, hóa, đtb
- Phương thức: nhập, in, tính điểm trung bình

Hàm main:

- Nhập danh sách sinh viên
- Sắp xếp theo điểm trung bình giảm dần
- In danh sách sau khi xếp

**Bài 6.** Xây dựng lớp hóa đơn gồm các thành phần:

- Dữ liệu: mã vật tư, tên vật tư, loại phiếu (nhập/xuất), ngày lập, khối lượng, đơn giá, thành tiền
- Phương thức: nhập, in hóa đơn.

Hàm main:

- Nhập danh sách hóa đơn
- Tính thành tiền cho các hóa đơn và in tổng thành tiền
- In danh sách hóa đơn sau khi sắp xếp theo số tiền giảm dần.