# Lời nói đầu

Lập trình cấu trúc là phương pháp tổ chức, phân chia chương trình thành các hàm, thủ tục, chúng được dùng để xử lý dữ liệu nhưng lại tách rời các cấu trúc dữ liệu. Thông qua các ngôn ngữ Foxpro, Pascal, C đa số những người làm Tin học đã khá quen biết với phương pháp lập trình này.

Lập trình hướng đối tượng dựa trên việc tổ chức chương trình thành các lớp. Khác với hàm và thủ tục, lớp là một đơn vị bao gồm cả dữ liệu và các phương thức xử lý. Vì vậy lớp có thể mô tả các thực thể một cách chân thực, đầy đủ cả phần dữ liệu và yêu cầu quản lý. Tư tưởng lập trình hướng đối tượng được áp dụng cho hầu hết các ngôn ngữ mới chạy trên môi trường Windows như Microsoft Access, Visual Basic, Visual C. Vì vậy việc nghiên cứu phương pháp lập trình mới này là rất cần thiết đối với tất cả những người quan tâm, yêu thích Tin học.

C ra đời năm 1973 với mục đích ban đầu là để viết hệ điều hành Unix trên máy tính mini PDP. Sau đó C đã được sử dụng rộng rãi trên nhiều loại máy tính khác nhau và đã trở thành một ngôn ngữ lập trình cấu trúc rất được ưa chuộng.

Để đưa C vào thế giới hướng hướng đối tượng, năm 1980 nhà khoa học người Mỹ B. Stroustrup đã cho ra đời một ngôn ngữ C mới có tên ban đầu là “C có lớp”, sau đó đến năm 1983 thì gọi là C++. Ngôn ngữ C++ là một sự phát triển mạnh mẽ của C. Trong C++ chẳng những đưa vào tất cả các khái niệm, công cụ của lập trình hướng đối tượng mà còn đưa vào nhiều khả năng mới mẻ cho hàm. Như vậy C++ là một ngôn ngữ lai cho phép tổ chức chương trình theo các lớp và các hàm. Có thể nói C++ đã thúc đẩy ngôn ngữ C vốn đã rất thuyết phục đi vào thế giới lập trình hướng đối tượng và C++ đã trở thành ngôn ngữ hướng đối tượng nổi bật trong những năm 90.

Cuốn sách này sẽ trình bầy một cách hệ thống các khái niệm của lập trình hướng đối tượng được cài đặt trong C++ như lớp, đối tượng, sự thừa kế, tính tương ứng bội và các khả năng mới trong xây dựng, sử dụng hàm như: đối tham chiếu, đối mặc định, hàm trùng tên, hàm toán tử. Có một số vấn đề còn ít được biết đến như cách xây dựng hàm với số đối bất định trong C cũng sẽ được giới thiệu. Các chương từ 1 đến 10 với cách giải thích tỉ mỉ và với gần 100 chương trình minh hoạ sẽ cung cấp cho bạn đọc các khái niệm, phương pháp và kinh nghiệm lập trình hướng đối tượng trên C++. Mục lục cuối sách sẽ hệ thống ngắn gọn phương pháp phân tích, thiết kế và lập trình hướng đối tượng trên bình diện chung.

**Cuốn sách gồm 10 chương và 6 phụ lục:**

* **Chương 1** hướng dẫn cách làm việc với phần mềm TC++ 3.0 để thử nghiệm các chương trình, trình bầy sơ lược về các phương pháp lập trình và giới thiệu một số mở rộng đơn giản của C++ .
* **Chương 2** trình bầy các khả năng mới trong việc xây dựng và sử dụng hàm trong C++ như biến tham chiếu, đối có kiểu tham chiếu, đối có giá trị mặc định, hàm trực tuyến, hàm trùng tên, hàm toán tử.
* **Chương 3** nói về một khái niệm trung tâm của lập trình hướng đối tượng là lớp gồm: Định nghĩa lớp, khai báo các biến, mảng đối tượng (kiểu lớp), phương thức, dùng con trỏ this trong phương thức, phạm vi truy xuất của các thành phần, các phương thức toán tử.
* **Chương 4** trình bầy các vấn đề tạo dựng, sao chép, huỷ bỏ các đối tượng và các vấn đề khác có liên quan như: Hàm tạo, hàm tạo sao chép, hàm huỷ, toán tử gán, cấp phát bộ nhớ cho đối tượng, hàm bạn, lớp bạn.
* **Chương 5** trình bầy một khái niệm quan trọng tạo nên khả năng mạnh của lập trình hướng đối tượng trong việc phát triển, mở rộng phần mềm, đó là khả năng thừa kế của các lớp.
* **Chương 6** trình bầy một khái niệm quan trọng khác cho phép xử lý các vấn đề khác nhau, các thực thể khác nhau, các thuật toán khác nhau theo cùng một lược đồ thống nhất, đó là tính tương ứng bội và phương thức ảo. Các công cụ này cho phép dễ dàng tổ chức chương trình quản lý nhiều dạng đối tượng khác nhau.
* **Chương 7** nói về việc tổ chức vào - ra trong C++. C++ đưa vào một khái niệm mới gọi là các dòng tin (Stream). Các thao tác vào - ra sẽ thực hiện trao đổi dữ liệu giữa bộ nhớ với dòng tin: Vào là chuyển dữ liệu từ dòng nhập vào bộ nhớ, ra là chuyển dữ liệu từ bộ nhớ lên dòng xuất. Để nhập xuất dữ liệu trên một thiết bị cụ thể nào, ta chỉ cần gắn dòng nhập xuất với thiết bị đó. Việc tổ chức vào ra theo cách như vậy là rất khoa học và tiện lợi vì nó có tính độc lập thiết bị.
* **Chương 8** trình bầy các hàm đồ hoạ sử dụng trong C và C++. Các hàm này được sử dụng rải rác trong toàn bộ cuốn sách để xây dựng các đối tượng đồ hoạ.
* **Chương 9** trình bầy các hàm truy xuất trực tiếp vào bộ nhớ của máy tính, trong đó có bộ nhớ màn hình. Các hàm này sẽ được sử dụng trong chương 10 để xây dựng các lớp menu và cửa sổ .
* **Chương 10** giới thiệu 5 chương trình tương đối hoàn chỉnh nhằm minh hoạ thêm khả năng và kỹ thuật lập trình hướng đối tượng trên C++
* **Phụ lục 1** trình bầy các phép toán trong C++ và thứ tự ưu của chúng.
* **Phụ lục 2** liệt kê một danh sách các từ khoá của C++.
* **Phụ lục 3** trình bầy bảng mã ASCII và mã quét của các ký tự.
* **Phụ lục 4** trình bầy một vấn đề quan trọng nhưng còn ít được nói đến trong các tài liệu, đó là cách sử dụng con trỏ void để xây dựng các hàm với số đối không cố định giống như các hàm printf và scanf của C.

Vì trong C++ vẫn sử dụng các hàm của C, nên trong **phụ lục 5** sẽ giới thiệu tóm tắt hơn 200 hàm để bạn đọc tiện việc tra cứu.

Cuối cùng, **phụ lục 6** trình bầy một cách ngắn gọn phương pháp phân tích, thiết kế và lập trình hướng đối tượng trên bình diện chung.

Khi viết chúng tôi đã hết sức cố gắng để cuốn sách được hoàn chỉnh, song chắc chắn không tránh khỏi thiếu sót, vì vậy rất mong nhận được sự góp ý của độc giả.

Nhân dịp này chúng tôi xin chân thành cám ơn cử nhân Nguyễn Văn Phác đã tận tình giúp đỡ trong việc hiệu đính và biên tập cuốn sách này.

# Chương 1: C++ và lập trình hướng đối tượng

Trong chương này trình bầy các vấn đề sau:

* Cách sử dụng phần mềm TC++ 3.0
* Những sửa đổi cần thiết một chương trình C để biến nó thành một chương trình C++ (chạy được trong môi trường C++)
* Tóm lược về các phương pháp lập trình cấu trúc và lập trình hướng đối tượng
* Những mở rộng của C++ so với C

## Bài 1. Làm việc với TC++ 3.0

Các ví dụ trong cuốn sách này sẽ viết và thực hiện trên môi trường TC++ 3.0. Bộ cài đặt TC++

3.0 gồm 5 đĩa. Sau khi cài đặt (giả sử vào thư mục C:\TC) thì trong thư mục TC sẽ gồm các thư mục con sau:

C:\TC\BGI chứa các tệp đuôi BGI và CHR

C:\TC\BIN chứa các tệp chương trình (đuôi EXE) như TC, TCC, TLIB, TLINK C:\TC\INCLUDE chứa các tệp tiêu đề đuôi H

C:\TC\LIB chứa các tệp đuôi LIB, OBJ

Để vào môi trường của TC++ chỉ cần thực hiện tệp chương trình TC trong thư mục C:\TC\BIN . Kết quả nhận được hệ menu chính của TC++ với mầu nền xanh gần giống như hệ menu quen thuộc của TC (Turbo C). Hệ menu của TC++ gồm các menu: File, Edit, Search, Run, Compile, Debug, Project, Options, Window, Help.

Cách soạn thảo, biên dịch và chạy chương trình trong TC++ cũngg giống như trong TC, ngoại trừ điểm sau: Tệp chương trình trong hệ soạn thảo của TC++ có đuôi mặc định là CPP cũng trong TC thì tệp chương trình luôn có đuôi C.

Trong TC++ có thể thực hiện cả chương trình C và C++. Để thực hiện chương trình C cần dựng đuôi C để đặt tên cho tệp chương trình, để thực hiện chương trình C++ cần dựng đuôi CPP để đặt tên cho tệp chương trình.

## Bài 2. C và C++

* Có thể nói C++ là sự mở rộng (đáng kể) của C. Điều đó có nghĩa là mọi khả năng, mọi khái niệm trong C đều dùng được trong C++.
* Vì trong C++ sử dụng gần như toàn bộ các khái niệm, định nghĩa, các kiểu dữ liệu, các cấu trúc lệnh, các hàm và các công cụ khác của C, nên yêu cầu bắt buộc đối với các đọc giả C++ là phải biết sử dụng tương đối thành thạo ngôn ngữ C.
* Vì C++ là sự mở rộng của C, nên bản thân một chương trình C đó là chương trình C++ (chỉ cần thay đuôi C bằng đuôi CPP). Tuy nhiên Trình biên dịch TC++ yêu cầu mọi hàm chuẩn dùng trong chương trình đều phải khai báo nguyên mẫu bằng một câu lệnh #include, trong khi điều này không bắt buộc đối với Trình biên dịch của TC
* Trong C có thể dùng một hàm chuẩn mà bỏ qua câu lệnh #include để khai báo nguyên mẫu của hàm được dùng. Điều này không báo lỗi khi biên dịch, nhưng có thể dẫn đến kết quả sai khi chạy chương trình.

**Ví dụ** khi biên dịch chương trình sau trong môi trường C sẽ không gặp các dòng cảnh báo (Warning) và thông báo lỗi (error). Nhưng khi chạy sẽ nhận được kết quả sai.

*#include <stdio.h>*

*void main()*

*{*

*float a,b,c,p,s;*

*printf("\nNhap a, b, c ");*

*scanf("%f%f%f",&a,&b,&c);*

*p=(a+b+c)/2;*

*s= sqrt(p\*(p-a)\*(p-b)\*(p-c));*

*printf("\nDien tich = %0.2f",s);*

*getch();*

*}*

Nếu biên dịch chương trình này trong TC++ sẽ nhận được các thông báo lỗi sau:

Eror: Funtion ‘sqrt’ should have a prototype

Eror: Funtion ‘getch’ should have a prototype

Để biến chương trình trên thành một chương trình C++ cần:

+ Đặt tên chương chường với đuôi CPP

+ Thêm 2 câu lệnh #include để khai báo nguyên mẫu cho các hàm sqrt, getch: **#include <math.h>**

**#include <conio.h>**

## Bài 3. Lập trình cấu trúc và lập trình hướng đối tượng

* 1. **Phương pháp lập trình cấu trúc**
* Tư tưởng chính của lập trình cấu trúc là tổ chức chương trình thành các chương trình con. Trong PASCAL có 2 kiểu chương trình con là thủ tục và hàm. Trong C chỉ có một loại chương trình con là hàm.
* Hàm là một đơn vị chương trình độc lập dùng để thực hiện một phần việc nào đó như: Nhập số liệu, in kết quả hay thực hiện một số tính toán. Hàm cần có đối và các biến, mảng cục bộ dùng riêng cho hàm.
* Việc trao đổi dữ liệu giữa các hàm thực hiện thông qua các đối và các biến toàn bộ.
* Các ngôn ngữ như C, PASCAL, FOXPRO là các ngôn ngữ cho phép triển khai phương pháp lập trình cấu trúc.
* Một chương trình cấu trúc gồm các cấu trúc dữ liệu (như biến, mảng, bản ghi) và các hàm, thủ tục.
* Nhiệm vụ chính của việc tổ chức thiết kế chương trình cấu trúc là tổ chức chương trình thành các hàm, thủ tục: Chương trình sẽ bao gồm các hàm, thủ tục nào.

**Ví dụ** xét yêu cầu sau: Viết chương trình nhập toạ độ (x,y) của một dẫy điểm, sau đó tìm một cặp điểm cách xa nhau nhất.

Trên tư tưởng của lập trình cấu trúc có thể tổ chức chương trình như sau:

+ Sử dụng 2 mảng thực toàn bộ x và y để chứa toạ độ dẫy điẻm

+ Xây dựng 2 hàm:

Hàm nhapsl dùng để nhập toạ độ n điểm, hàm này có một đối là biến nguyên n và được khai báo như sau:

**void nhapsl(int n);**

Hàm do\_dai dùng để tính độ dài đoạn thẳng đi qua 2 điểm có chỉ số là i và j , nó được khai báo như sau:

**float do\_dai(int i, int j);**

Chương trình C cho bài toán trên được viết như sau:

*#include <stdio.h>*

*#include <conio.h>*

*#include <math.h>*

*float x[100],y[100];*

*float do\_dai(int i, int j)*

*{*

*return sqrt(pow(x[i]-x[j],2)+pow(y[i]-y[j],2));*

*}*

*void nhapsl(int n)*

*{*

*int i;*

*for (i=1;i<=n;++i)*

*{*

*printf("\nNhap toa do x, y cua diem thu %d : ",i); scanf("%f%f",&x[i],&y[i]);*

*}*

*}*

*void main()*

*{*

*int n,i,j,imax,jmax;*

*float d,dmax;*

*printf("\nSo diem N= "); scanf("%d",&n);*

*nhapsl(n);*

*dmax=do\_dai(1,2);*

*imax=1;jmax=2;*

*for (i=1;i<=n-1;++i)*

*for (j=i+1;j<=n;++j)*

*{*

*d=do\_dai(i,j);*

*if (d>dmax)*

*{*

*dmax=d; imax=i; jmax=j;*

*}*

*}*

*printf("\nDoan thang lon nhat co do dai bang: %0.2f",dmax);*

*printf("\n Di qua 2 diem co chi so la %d va %d",imax,jmax);*

*getch();*

*}*

* 1. **Phương pháp lập trình hướng đối tượng**

+ Khỏi niệm trung tâm của lập trình hướng đối tượng là lớp (class). Có thể xem lớp là sự kết hợp các thành phần dữ liệu và các hàm. Cũng có thể xem lớp là sự mở rộng của cấu trúc trong C (struct) bằng cách đưa thêm vào các phương thức (method) hay cũng gọi là hàm thành viên (member function). Một lớp được định nghĩa như sau:

*Class Tên\_Lớp*

*{*

*// Khai báo các thành phần dữ liệu*

*// Khai báo các phương thức*

*};*

+ Các phương thức có thể được viết (xây dựng) bên trong hoặc bên ngoài (phía dưới) phần định nghĩa lớp. Cấu trúc (cách viết) phương thức tương tự như hàm ngoại trừ quy tắc sau: Khi xây dựng một phương thức bên ngoài định nghĩa lớp thì trong dòng đầu tiên cần dùng tên lớp và 2 dấu : đặt trước tên phương thức để chỉ rừ phương thức thuộc lớp nào (xem ví dụ bên dưới).

+ Sử dụng các thành phần dữ liệu trong phương thức: Vì phương thức và các thành phần dữ liệu thuộc cùng một lớp và vì phương thức được lập lên cốt để xử lý các thành phần dữ liệu, nên trong thân của phương thức có quyền truy nhập đến các thành phần dữ liệu (của cùng lớp).

+ Biến lớp: Sau khi định nghĩa một lớp, có thể dùng tên lớp để khai báo các biến kiểu lớp hay cũng gọi là đối tượng. Mỗi đối tượng sẽ có các thành phần dữ liệu và các phương thức. Lời gọi một phương thức cần chứa tên đối tượng để xác định phương thức thực hiện từ đối tượng nào.

+ Một chương trình hướng đối tượng sẽ bao gồm các lớp có quan hệ với nhau.

+ Việc phân tích, thiết kế chương trình theo phương pháp hướng đối tượng nhằm thiết kế, xây dựng các lớp.

+ Từ khái niệm lớp nẩy sinh hàng loạt khái niệm khác như: Thành phần dữ liệu, phương thức, phạm vi, sự đóng gói, hàm tạo, hàm huỷ, sự thừa kế, lớp cơ sử, lớp dẫn xuất, tương ứng bội, phương thức ảo, ...

+ Ưu điểm của việc thiết kế hướng đối tượng là tập trung xác định các lớp để mô tả các thực thể của bài toán. Mỗi lớp đưa vào các thành phần dữ liệu của thực thể và xây dựng luôn các phương thức để xử lý dữ liệu. Như vậy việc thiết kế chương trình xuất phát từ các nội dụng, các vấn đề của bài toán.

+ Các ngụn ngữ thuần tuý hướng đối tượng (như Smalltalk) chỉ hỗ trợ các khái niệm về lớp, không có các khái niệm hàm.

+ C++ là ngôn ngữ lai , nó cho phép sử dụng cả các công cụ của lớp và hàm.

Để minh hoạ các khái niệm vừa nêu về lập trình hướng đối tượng ta trở lại xét bài toán tìm độ dài lớn nhất đi qua 2 điểm. Trong bài toán này ta gặp một thực thể là dẫy điểm. Các thành phần dữ liệu của lớp dẫy điểm gồm:

* Biến nguyên n là số điểm của dẫy
* Con trỏ x kiểu thực trỏ đến vùng nhớ chứa dẫy hoành độ
* Con trỏ y kiểu thực trỏ đến vùng nhớ chứa dẫy tung độ Các phương thức cần đưa vào theo yêu cầu bài toán gồm:
* Nhập toạ độ một điểm
* Tính độ dài đoạn thẳng đi qua 2 điểm

Dưới đây là chương trình viết theo thiết kế hướng đối tượng. Để thực hiện chương trình này nhớ đặt tên tệp có đuôi CPP. Xem chương trình ta thấy thờm một điều mới trong C++ là:

Các khai báo biến, mảng có thể viết bất kỳ chỗ nào trong chương trình (tất nhiên phải trước khi sử dụng biến, mảng).

*#include <stdio.h>*

*#include <conio.h>*

*#include <math.h>*

*#include <alloc.h>*

*class daydiem*

*{*

*public:*

*int n;*

*float \*x,\*y;*

*float do\_dai(int i, int j)*

*{*

*return sqrt(pow(x[i]-x[j],2)+pow(y[i]-y[j],2));*

*}*

*void nhapsl(void);*

*};*

*void daydiem::nhapsl(void)*

*{*

*int i;*

*printf("\nSo diem N= "); scanf("%d",&n); x=(float\*)malloc((n+1)\*sizeof(float));*

*y=(float\*)malloc((n+1)\*sizeof(float));*

*for (i=1;i<=n;++i)*

*{*

*printf("\nNhap toa do x, y cua diem thu %d : ",i); scanf("%f%f",&x[i],&y[i]);*

*}*

*}*

*void main()*

*{*

*daydiem p;*

*p.nhapsl();*

*int n,i,j,imax,jmax;*

*float d,dmax;*

*n=p.n;*

*dmax=p.do\_dai(1,2); imax=1;jmax=2;*

*for (i=1;i<=n-1;++i)*

*{*

*for (j=i+1;j<=n;++j)*

*{*

*d=p.do\_dai(i,j);*

*if (d>dmax)*

*{*

*dmax=d; imax=i; jmax=j;*

*}*

*}*

*}*

*printf("\nDoan thang lon nhat co do dai bang: %0.2f",dmax);*

*printf("\n Di qua 2 diem co chi so la %d va %d",imax,jmax);*

*getch();*

*}*

## Bài 4. Một số mở rộng đơn giản của C++ so với C

Trong mục này trình bầy một số mở rộng của C++ , tuy đơn giản, ngắn gọn nhưng đem lại rất nhiều tiện lợi.

* 1. **Viết các dòng ghi chú**

Trong C++ vẫn có thể viết các dòng ghi chú trong các dấu /\* và \*/ như trong C. Cách này cho phép viết các ghi chú trên nhiều dòng hoặc trên một dòng. Ngoài ra trong C++ cũng cho phép viết ghi chú trên một dòng sau 2 dấu gạch chéo, ví dụ:

**int x, y;** // Khai báo 2 biến thực

### Khai báo linh hoạt

Trong C tất cả các câu lệnh khai báo biến, mảng cục bộ phải đặt tại đầu khối. Do vậy nhiều khi, vị trí khai báo và vị trí sử dụng của biến khá xa nhau, gây khó khăn trong việc kiểm soát chương trình. C++ đó khắc phục nhược điểm này bằng cách cho phép các lệnh khai báo biến, mảng có thể đặt bất kỳ chỗ nào trong chương trình trước khi các biến, mảng được sử dụng. Ví dụ chương trình nhập một dẫy số thực rồi sắp xếp theo thứ tự tăng dần có thể viết trong C++ như sau:

*#include <stdio.h>*

*#include <conio.h>*

*#include <alloc.h>*

*void main()*

*{*

*int n;*

*printf("\n So phan tu cua day N= ");*

*scanf("%d",&n);*

*float \*x= (float\*)malloc((n+1)\*sizeof(float));*

*for (int i=1;i<=n;++i)*

*{*

*printf("\nX[%d]= ",i);*

*scanf("%f",x+i);*

*}*

*for (i=1;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;*

*}*

*}*

*}*

*printf("\nDay sau khi sap xep\n");*

*for (i=1;i<=n;++i)*

*{*

*printf("%0.2f ",x[i]);*

*}*

*getch();*

*}*

* 1. **Toán tử ép kiểu**

Toán tử này được viết trong C như sau: (Kiểu) biểu thức

Trong C++ vẫn có thể dùng cách viết này. Ngoài ra C++ cho phép viết một cách khác tiện lợi hơn như sau:

Kiểu(biểu thức)

Ví dụ chương trình tính công thức

S = 2/1 + 3/2 + ... + (n+1)/n

với n là một số nguyên dương nhập từ bàn phím, có thể viết như sau:

*#include <stdio.h>*

*#include <conio.h>*

*void main()*

*{*

*int n;*

*printf("\n So phan tu cua day N= ");*

*scanf("%d",&n);*

*float s=0.0;*

*for (int i=1;i<=n;++i)*

*{*

*s += float(i+1)/float(i) ; // Ep kieu theo C++*

*printf("S= %0.2f ",s);*

*}*

*getch();*

*}*

* 1. **Hằng có kiểu**

Để tạo ra một hằng có kiểu, ta sử dụng từ khoá const đặt trước một khai báo có khởi gán giá trị.

Sau đây là một số ví dụ.

+ Hằng nguyên:

**const int maxsize = 1000;**

**int a[maxsize] ;**

+ Cấu trúc hằng:

*typedef struct*

*{*

*int x, y ; // Toạ độ của điểm*

*int mau ; // Mó mầu của điểm*

*} DIEM ;*

*const DIEM d = {320, 240, 15};*

Chương trình dưới đây minh hoạ cách dùng hằng có kiểu. Chương trình tạo một cấu trúc hằng (kiểu DIEM) mụ tả điểm giữa màn hình đồ hoạ với mầu trắng. Điểm này được hiển thị trên màn hình đồ hoạ.

*#include <stdio.h>*

*#include <conio.h>*

*#include <graphics.h>*

*#include <stdlib.h>*

*typedef struct*

*{*

*int x,y; int mau;*

*} DIEM;*

*void main()*

*{*

*int mh=0,mode=0;*

*initgraph(&mh,&mode,"");*

*int loi=graphresult();*

*if (loi)*

*{*

*printf("\nLoi do hoa: %s",grapherrormsg(loi));*

*getch(); exit(0);*

*}*

*const DIEM gmh = {getmaxx()/2,getmaxy()/2,WHITE};*

*putpixel(gmh.x,gmh.y,gmh.mau);*

*getch();*

*closegraph();*

*}*

**Chú ý:**

* + 1. Có thể dùng các hàm để gán giá trị cho các hằng có kiểu (trong chương trình trên dựng các hàm getmax và getmaxy).
    2. Mọi câu lệnh nhằm thay đổi giá trị hằng có kiểu đều bị báo lỗi khi biên dịch chương trình. Ví dụ nếu trong chương trình đưa vào câu lệnh:

**gmh.x=200;**

thì khi dịch chương trình sẽ nhận được thông báo lỗi như sau: **Cannot modify a const object**

* 1. **Các kiểu char và int**

Trong C một hằng ký tự được xem là nguyên do đó nó có kích thước 2 byte, ví dụ trong C:

**sizeof(‘A’) = sizeof(int) = 2**

Cũng trong C++ một hằng ký tự được xem là giá trị kiểu char và có kích thước một byte. Như vậy trong C++ thì:

**sizeof(‘A’) = sizeof(char) = 1**

* 1. **Lấy địa chỉ các phần tử mảng thực 2 chiều**

Trong Turbo C 2.0 không cho phép dùng phép & để lấy địa chỉ các phần tử mảng thực 2 chiều. Vì vậy khi nhập một ma trân thực (dùng scanf) ta phải nhập qua một biến trung gian sau đó mới gán cho các phần tử mảng.

Trong TC ++ 3.0 cho phép lấy địa chỉ các phần tử mảng thực 2 chiều, do đó có thể dùng scanf để nhập trực tiếp vào các phần tử mảng.

Chương trình C++ dưới đây sẽ minh hoạ điều này. Chương trình nhập một ma trận thực cấp mxn và xỏc định phần tử có giá trị lớn nhất.

*#include <conio.h>*

*#include <stdio.h>*

*void main()*

*{*

*float a[20][20], smax;*

*int m,n,i,j, imax, jmax;*

*clrscr();*

*puts( "Cho biet so hang va so cot cua ma tran: ") ;*

*scanf("%d%d",&m,&n) ;*

*for (i=1;i<=m;++i) for (j=1;j<=n;++j)*

*{*

*printf("\na[%d][%d]= ",i,j);*

*scanf("%f",&a[i][j]); // Lấy địa chỉ phần tử mảng thực 2 chiều*

*}*

*smax = a[1][1]; imax=1; jmax=1;*

*for (i=1;i<=m;++i)*

*{*

*for (j=1;j<=n;++j)*

*{*

*if (smax<a[i][j])*

*{*

*smax = a[i][j];*

*imax=i ;*

*jmax = j;*

*}*

*}*

*}*

*puts( "\n\n Ma tran") ;*

*for (i=1;i<=m;++i)*

*{*

*for (j=1;j<=n;++j)*

*{*

*if (j==1) puts("");*

*printf("%6.1f", a[i][j]);*

*}*

*}*

*puts( "\n\nPhan tu max:" );*

*printf("\nco gia tri = %6.1f", smax);*

*printf("\nTai hang %d cot %d " ,imax, jmax) ;*

*getch();*

*}*

## Bài 5. Vào ra trong C++

* 1. **Các toán tử và phương thức xuất nhập**

Để in dữ liệu ra màn hình và nhập dữ liệu từ bàn phớm , trong C++ vẫn có thể dựng các hàm printf và scanf (như chỉ ra trong các chương trình C++ ở các mục trên).

Ngoài ra trong C++ cũng dựng toán tử xuất:

**cout << biểu thức << ... << biểu thức ;**

để đưa giá trị các biểu thức ra màn hình, dựng toán tử nhập:

**cin >> biến >> ... >> biến**

để nhập các giá trị số (nguyên thực) từ bàn phím và gán cho các biến.

Để nhập một dẫy không quá n ký tự và chứa vào mảng h (kiểu char) có thể dùng phương thức cin.get như sau:

**cin.get(h,n);**

***Chú ý 1:*** Toán tử nhập cin >> sẽ để lại ký tự chuyển dòng ‘\n’ trong bộ đệm, ký tự này có thể làm trôi phương thức cin.get. Để khắc phục tình trạng trên cần dựng phương thức cin.ignore để bỏ qua một ký tự chuyển dòng như sau:

**cin.ignore(1);**

***Chú ý 2:*** Để sử dụng các toán tử và phương thức nói trên cần khai báo tệp tiêu đề:

**#include <iostream.h>**

Chương trình sau minh hoạ việc sử dụng các cụng cụ vào ra mới của C++ để nhập một danh sách n thí sinh. Dữ liệu mỗi thí sinh gồm họ tên, các điểm toán, lý, hoá. Sau đó in danh sách thí sinh theo thứ tự giảm của tổng điểm.

*#include <iostream.h>*

*#include <conio.h>*

*void main()*

*{*

*struct*

*{*

*char ht[25];*

*float t,l,h,td;*

*} ts[50],tg;*

*int n,i,j;*

*clrscr();*

*cout << " So thi sinh: " ; cin >> n ;*

*for (i=1;i<=n;++i)*

*{*

*cout << "\n Thi sinh " << i ;*

*cout << "\n Ho ten: " ;*

*cin.ignore(1);*

*cin.get(ts[i].ht,25) ;*

*cout << "Cac diem toan, ly, hoa: ";*

*cin >> ts[i].t >> ts[i].l >> ts[i].h ;*

*ts[i].td = ts[i].t + ts[i].l + ts[i].h ;*

*}*

*for (i=1;i<=n-1;++i)*

*{*

*for (j=i+1;j<=n;++j)*

*{*

*if (ts[i].td < ts[j].td )*

*{*

*tg=ts[i];*

*ts[i]=ts[j];*

*ts[j]=tg;*

*}*

*}*

*}*

*cout << "\nDanh sach thi sinh sau khi sap xep " ;*

*for (i=1;i<=n;++i)*

*{*

*cout << "\n Ho ten: " << ts[i].ht;*

*cout << " Tong diem: " << ts[i].td;*

*}*

*getch();*

*}*

* 1. **Định dạng khi in ra màn hình**

+ Để quy định số thực (float, double) được in ra có đúng p chữ số sau dấu chấm thập phân, ta sử dụng đồng thời các hàm sau:

**setiosflags(ios::showpoint); // Bật cờ hiệu showpoint**

**setprecision(p);**

Các hàm này cần đặt trong toán tử xuất như sau:

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

Câu lệnh trên sẽ có hiệu lực đối với tất cả các toán tử xuất tiếp theo cho đến khi gặp một câu lệnh định dạng mới.

+ Để quy định độ rộng tối thiểu là w vị trí cho giá trị (nguyên, thực, chuỗi) được in trong các toán tử xuất, ta dùng hàm

**setw(w)**

Hàm này cần đặt trong toán tử xuất và nó chỉ có hiệu lực cho một giá trị được in gần nhất. Các giá trị in ra tiếp theo sẽ có độ rộng tối thiểu mặc định là 0. Như vậy câu lệnh:

**cout << setw(3) << “AB” << “CD”**

Sẽ in ra 5 ký tự là: một dấu cách và 4 chữ cỏi A, B, C và D.

***Chú ý:*** Muốn sử dụng các hàm trên cần đưa vào câu lệnh #include sau:

**#include <iomanip.h>**

Trở lại chương trình trên ta thấy danh sỏch thớ sinh in ra sẽ không thẳng cột. Để khắc phục điều này cần viết lại đoạn chương trình in như sau:

*cout << "\nDanh sach thi sinh sau khi sap xep " ;*

*cout << setiosflags(ios::showpoint) << setprecision(1) ;*

*for(i=1;i<=n;++i)*

*{*

*cout << "\n Ho ten: " << setw(25) << ts[i].ht;*

*cout << " Tong diem: " << setw(5)<< ts[i].td;*

*}*

*getch();*

Chương trình dưới đây là một minh hoạ khác về việc sử dụng các toán tử nhập xuất và cách định dạng trong C++ . Chương trình nhập một ma trận thực cấp mxn. Sau đó in ma trận dưới dạng bảng và tìm một phần tử lớn nhất.

*#include <iostream.h>*

*#include <iomanip.h>*

*#include <conio.h>*

*void main()*

*{*

*float a[20][20], smax;*

*int m,n,i,j, imax, jmax;*

*clrscr();*

*cout << " Cho biet so hang va so cot cua ma tran: " ;*

*cin >> m >> n ;*

*for (i=1;i<=m;++i)*

*{*

*for (j=1;j<=n;++j)*

*{*

*cout << "a[" << i << "," << j << "]= " ;*

*cin >> a[i][j] ;*

*}*

*}*

*smax = a[1][1]; imax=1; jmax=1;*

*for (i=1;i<=m;++i)*

*{*

*for (j=1;j<=n;++j)*

*{*

*if (smax<a[i][j])*

*{*

*smax = a[i][j];*

*imax=i ;*

*jmax = j;*

*}*

*}*

*}*

*cout << "\n\n Ma tran" ;*

*cout << setiosflags(ios::showpoint) << setprecision(1) ;*

*for (i=1;i<=m;++i)*

*{*

*for (j=1;j<=n;++j)*

*{*

*if (j==1)*

*{*

*cout << '\n' ;*

*cout << setw(6) << a[i][j];*

*}*

*}*

*}*

*cout << "\n\n" << "Phan tu max:" << '\n' ;*

*cout << "co gia tri = " << setw(6) << smax;*

*cout << "\nTai hang " << imax << " cot " << jmax ;*

*getch();*

*}*

## Bài 6. Cấu trúc, hợp và kiểu liệt kê

### Tên sau từ khoá struct được xem như tên kiểu cấu trúc

Trong C++ một kiểu cấu trúc cũngg được định nghĩa như C theo mẫu:

**struct Tên\_kiểu\_ct**

**{**

**// Khai báo các thành phần của cấu trúc**

**} ;**

Sau đó để khai báo các biến, mảng cấu trúc, trong C dùng mẫu sau: struct Tên\_kiểu\_ct danh sách biến, mảng cấu trúc ;

Như vậy trong C, tên viết sau từ khoá struct chưa phải là tên kiểu và chưa có thể dùng để khai báo.

Trong C++ xem tên viết sau từ khoá struct là tên kiểu cấu trúc và có thể dùng nó để khai báo.

Như vậy để khai báo các biến, mảng cấu trúc trong C++ , ta có thể dùng mẫu sau: Tên\_kiểu\_ct danh sách biến, mảng cấu trúc ;

**Ví dụ** sau sẽ: Định nghĩa kiểu cấu trúc TS (thí sinh) gồm các thành phần : ht (họ tên), sobd (số báo danh), dt (điểm toán), dl (điểm lý), dh (điểm hoá) và td (tổng điểm), sau đó khai báo biến cấu trúc h và mảng cấu trúc ts.

*struct TS*

*{*

*char ht [25];*

*long sobd;*

*float dt, dl, dh, td;*

*} ;*

*TS h, ts[1000] ;*

* 1. **Tên sau từ khoá union được xem như tên kiểu hợp**

Trong C++ một kiểu hợp (union) cũngg được định nghĩa như C theo mẫu:

**union Tên\_kiểu\_hợp**

**{**

**// Khai báo các thành phần của hợp**

**} ;**

Sau đó để khai báo các biến, mảng kiểu hợp , trong C dùng mẫu sau:

**union Tên\_kiểu\_hợp danh sách biến, mảng kiểu hợp ;**

Như vậy trong C, tên viết sau từ khoá union chưa phải là tên kiểu và chưa có thể dùng để khai báo.

Trong C++ xem tên viết sau từ khoá union là tên kiểu hợp và có thể dùng nó để khai báo. Như vậy để khai báo các biến, mảng kiểu hợp, trong C++ có thể dùng mẫu sau:

Tên\_kiểu\_hợp danh sách biến, mảng kiểu hợp ;

* 1. **Các union không tên**

Trong C++ cho phép dùng các union không tên dạng:

**union**

**{**

**// Khai báo các thành phần**

**} ;**

Khi đó các thành phần (khai báo trong union) sẽ dùng chung một vùng nhớ. Điều này cho phép tiết kiệm bộ nhớ và cho phép dễ dàng tách các byte của một vùng nhớ.

**Ví dụ** nếu các biến nguyên i , biến ký tự ch và biến thực x không đồng thời sử dụng thì có thể khai báo chúng trong một union không tên như sau:

*union*

*{*

*int i ;*

*char ch ;*

*float x ;*

*} ;*

Khi đó các biến i , ch và f sử dụng chung một vùng nhớ 4 byte.

Xét ví dụ khác, để tách các byte của một biến unsigned long ta dùng union không tên sau:

*union*

*{*

*unsigned long u ;*

*unsigned char b[4] ;*

*};*

Khi đó nếu gán

u = 0xDDCCBBAA; // Số hệ 16

thì :

b[0] = 0xAA

b[1] = 0xBB

b[2] = 0xCC

b[3] = 0xDD

* 1. **Kiểu liệt kê (enum)**

+ Cũng giống như cấu trúc và hợp, tên viết sau từ khoá enum được xem là kiểu liệt kê và có thể dùng để khai báo, ví dụ:

**enum MAU { xanh, do, tim, vang } ; // Định nghĩa kiểu MAU**

MAU m, dsm[10] ; // Khai báo các biến, mảng kiểu MAU

+ Các giá trị kiểu liệt kê (enum) là các số nguyên. Do đó có thể thực hiện các phép tính trên các giá trị enum, có thể in các giá trị enum, có thể gán giá trị enum cho biến nguyên.

**Ví dụ:**

MAU m1 , m2;

int n1, n2;

m1 = tim;

m2 = vàng;

n1 = m1; // n1 = 2

n2 = m1 + m2; // n2 = 5

printf (“\n %d”, m2); // in ra số 3

+ Không thể gán trực tiếp một giá trị nguyên cho một biến enum mà phải dùng phép ép kiểu

**Ví dụ:**

m1 = 2 ; // lỗi

m1 = MAU(2) ; // đúng

## Bài 7. Cấp phát bộ nhớ

* 1. **Trong C++** có thể sử dụng các hàm cấp phát bộ nhớ động của C như: hàm malloc để cấp phát bộ nhớ, hàm free để giải phóng bộ nhớ được cấp phát.
  2. **Ngoài ra trong C++** cũng đưa thêm toán tử new để cấp phát bộ nhớ và toán tử delete để giải phóng bộ nhớ được cấp phát bởi new
  3. **Cách dùng toán tử new để cấp phát bộ nhớ như sau:**

+ Trước hết cần khai báo một con trỏ để chứa địa chỉ vùng nhớ sẽ được cấp phát:

**Kiểu \*p;**

ở đây Kiểu có thể là:

* Các kiểu dữ liệu chuẩn của C++ như int , long, float , double, char , ...
* các kiểu do lập trình viên định nghĩa như: mảng, hợp, cấu trúc, lớp, ...

+ Sau đó dùng toán tử new theo mẫu:

**p = new Kiểu ; // Cấp phát bộ nhớ cho một biến (một phần tử)**

**p = new Kiểu[n] ; //Cấp phát bộ nhớ cho n phần tử**

**Ví dụ** để cấp phát bộ nhớ cho một biến thực ta dùng câu lệnh sau: float \*px = new float ;

Để cấp phát bộ nhớ cho 100 phần tử nguyên ta dùng các câu lệnh:

*int \*pn = new int[100] ;*

*for (int i=0 ; i < 100 ; ++i )*

*pn[i] = 20\*i ; // Gán cho phần tử thứ i*

* 1. **Hai cách kiểm tra sự thành công của new**

Khi dùng câu lệnh:

**Kiểu \*p = new Kiểu[n] ;**

hoặc câu lệnh:

**Kiểu \*p = new Kiểu ;**

để cấp phát bộ nhớ sẽ xuất hiện một trong 2 trường hợp: thành công hoặc không thành công:

* Nếu thành công thì p sẽ chứa địa chỉ đầu vùng nhớ được cấp phát.
* Nếu không thành cụng thì p = NULL.

Đoạn chương trình sau minh hoạ cách kiểm tra lỗi cấp phát bộ nhớ:

*double \*pd ; int n ;*

*cout << “\n Số phần tử : “ ;*

*cin >> n ;*

*pd = new double[n] ;*

*if (pd==NULL)*

*{*

*cout << “ Lỗi cấp phát bộ nhớ “ exit (0) ;*

*}*

Cách thứ 2 để kiểm tra sự thành công của toán tử new là dùng con trỏ hàm:

**\_new\_handler**

được định nghĩa trong tệp “new.h”. Khi gặp lỗi trong toán tử new (cấp phát không thành công) thì chương trình sữ thực hiện một hàm nào đó do con trỏ \_new*\_*handler trỏ tới. Cách dùng con trỏ này như sau:

+ Xây dựng một hàm dùng để kiểm tra sự thành công của new

+ Gán tên hàm này cho con trỏ \_new\_handler

Như vậy hàm kiểm tra sẽ được gọi mỗi khi có lỗi xẩy ra trong toán tử new.

Đoạn chương trình kiểm tra theo cách thứ nhất có thể viết theo cách thứ hai như sau:

*void kiem\_tra\_new(void) // Lập hàm kiểm tra*

*{*

*cout << “ Lỗi cấp phát bộ nhớ “ exit (0) ;*

*}*

*\_new\_handler = kiem\_tra\_new // Gán tên hàm cho con trỏ double \*pd ;*

*int n ;*

*cout << “\n Số phần tử : “ ; cin >> n ;*

*pd = new double[n] ; // Khi xẩy ra lỗi sẽ gọi hàm kiểm\_tra\_new*

***Chú ý:*** Có thể dùng lệnh gán để gán tên hàm xử lý lỗi cho con trỏ \_new*\_*handler như trong đoạn chương trình trên, hoặc dựng hàm:

**set\_new\_handler(Tên hàm) ;**

(xem các chương trình minh hoạ bên dưới)

* 1. **Toán tử delete dùng để giải phóng vùng nhớ được cấp phát bởi new**

Cách dùng như sau:

**delete p ;** // p là con trỏ dùng trong new

Ví dụ:

*float \*px ;*

*px = new float[2000] ; // Cấp phát bộ nhớ cho 2000 phần tử thực*

*// Sử dụng bộ nhớ được cấp phát*

*delete px ; // giải phóng bộ nhớ*

* 1. **Hai chương trình minh hoạ**

Chương trình thứ nhất minh hoạ cách dựng new để cấp phát bộ nhớ chứa n thí sinh. Mỗi thí sinh là một cấu trúc gồm các trường ht (họ tên), sobd (số báo danh) và td (tổng điểm). Chương trình sẽ nhập n, cấp phát bộ nhớ chứa n thớ sinh, kiểm tra lỗi cấp phát bộ nhớ (dựng cách 1), nhập n thớ sinh, sắp xếp thớ sinh theo thứ tự giảm của tổng điểm, in danh sách thí sinh sau khi sắp xếp, và cuối cùng là giải phóng bộ nhớ đó cấp phát.

*#include <iomanip.h>*

*#include <iostream.h>*

*#include <stdlib.h>*

*#include <conio.h>*

*struct TS*

*{*

*char ht[20];*

*long sobd;*

*float td;*

*};*

*void main(void)*

*{*

*TS\*ts ;*

*int n;*

*cout << "\n So thi sinh n = " ;*

*cin >> n;*

*ts = new TS[n+1];*

*if(ts==NULL)*

*{*

*cout << "\nLoi cap phat bo nho " ;*

*getch();*

*exit(0);*

*}*

*for (int i=1;i<=n;++i)*

*{*

*cout <<"\nThi sinh thu " << i;*

*cout << "\nHo ten: " ;*

*cin.ignore(1) ;*

*cin.get(ts[i].ht,20);*

*cout << "So bao danh: " ;*

*cin >> ts[i].sobd ;*

*cout << "Tong diem: " ; cin >> ts[i].td ;*

*}*

*for (i=1;i<=n-1;++i)*

*{*

*for (int j=i+1;j<=n;++j)*

*{*

*if (ts[i].td < ts[j].td)*

*{*

*TS tg=ts[i];*

*ts[i]=ts[j];*

*ts[j]=tg;*

*}*

*}*

*}*

*cout << setiosflags(ios::showpoint) << setprecision(1) ;*

*for (i=1;i<=n;++i)*

*{*

*cout << "\n" << setw(20) << ts[i].ht <<*

*setw(6)<< ts[i].sobd <<setw(6)<< ts[i].td;*

*}*

*delete ts;*

*getch();*

*}*

Chương trình thứ hai minh hoạ cách dựng con trỏ \_new\_handler để kiểm tra sự thành công của toán tử new. Chương trình sẽ cấp phát bộ nhớ cho một mảng con trỏ và sẽ theo rồi khi nào thì không đủ bộ nhớ để cấp phát.

*#include <new.h>*

*#include <iostream.h>*

*#include <stdlib.h>*

*#include <conio.h>*

*int k;*

*void loi\_bo\_nho(void)*

*{*

*cout << "\nLoi bo nho khi cap phat bo nho cho q[" << k << "]"; getch();*

*exit(0);*

*}*

*void main()*

*{*

*double \*q[100] ; long n;*

*clrscr();*

*set\_new\_handler(loi\_bo\_nho); // \_new\_handler=loi\_bo\_nho;*

*n=10000;*

*for ( k=0;k<100;++k)*

*{*

*q[k] = new double[n];*

*}*

*cout << "Khong loi";*

*getch();*

*}*

## Bài 8. Các hàm trong C++

Trong C++ có rất nhiều mở rộng, cải tiến về hàm làm cho việc xây dựng và sử dụng hàm rất tiện lợi. Điều này sẽ trình bầy kỹ trong chương sau. Trong mục này chỉ thống kê một số điểm mới về hàm mà C++ đưa vào.

* 1. **Đối kiểu tham chiếu**

Trong C, để nhận kết quả của hàm cần dùng đối con trỏ, làm cho việc xây dựng cũngg như sử dụng hàm khá phiền phức. Trong C++ đưa vào đối kiểu tham chiếu (giống như PASCAL) dùng để chứa kết quả của hàm, khiến cho việc tạo lập cũng như sử dụng hàm đơn giản hơn.

* 1. **Đối tham chiếu const**

Đối tham chiếu có đặc điểm là các câu lệnh trong thân hàm có thể truy nhập tới và dễ dàng làm cho giá trị của nó thay đổi. Nhiều khi ta muốn dùng đối kiểu tham chiếu chỉ để tăng tốc độ trao đổi dữ liệu giữa các hàm, không muốn dùng nó để chứa kết quả của hàm. Khi đó có thể dùng đối tham chiếu const để bảo toàn giá trị của đối trong thân hàm.

* 1. **Đối có giá trị mặc định**

Trong nhiều trương hợp người dùng viết một lời gọi hàm nhưng cũng chưa biết nên chọn giá trị nào cho các đối . Để khắc phục khó khăn này, C++ đưa ra giải pháp đối có giá trị mặc định. Khi xây dựng hàm, ta gán giá trị mặc định cho một số đối. Người dùng nếu không cung cấp giá trị cho các đối này, thì hàm sẽ dựng giỏ trị mặc định.

* 1. **Hàm on line**

Đối với một đoạn chương trình nhỏ (số lệnh không lớn) thì việc thay các đoạn chương trình này bằng các lời gọi hàm sẽ làm cho chương trình gọn nhẹ đôi chút nhưng làm tăng thời gian máy. Trong các trường hợp này có thể dùng hàm trực tuyến (on line) vừa giảm kích thước chương trình nguồn, vừa không làm tăng thời gian chạy máy.

* 1. **Các hàm trùng tên (định nghĩa chồng các hàm)**

Để lấy giá trị tuyệt đối của một số, trong C cần lập ra nhiều hàm với tên khác nhau, ví dụ abs cho số nguyên, fabs cho số thực, labs cho số nguyên dài, cabs cho số phức. Điều này rừ ràng gây phiền toái cho người sử dụng. Trong C++ cho phép xây dựng các hàm trùng tên nhưng khác nhau về kiểu đối. Như vậy chỉ cần lập một hàm để lấy giá trị tuyệt đối cho nhiều kiểu dữ liệu khác nhau.

* 1. **Định nghĩa chồng toán tử**

Việc dựng các phộp toán thay cho một lời gọi hàm rừ ràng làm cho chương trình ngắn gọn, sáng sủa hơn nhiều. Ví dụ để thực hiện phép cộng 2 ma trận nếu dùng phép cộng và viết:

C = A + B ;

thì rất gần với toán học. Trong C++ cho phép dựng các phép toán chuẩn để đặt tên cho các hàm (gọi là định nghĩa nạp chồng toán tử). Sau đó có thể thay lời gọi hàm bằng các phép toán như nói ở trên. Như vậy một phép toán mang nhiều ý nghĩa, ví dụ phép + có thể hiểu là cộng 2 số nguyên, 2 số thực hoặc 2 ma trận. C++ sẽ căn cứ vào kiểu của các số hạng mà quyết định chọn phép cộng cụ thể.

# Chương 2: Hàm trong C++

Chương này trình bày những khả năng mới của C++ trong việc xây dựng và sử dụng hàm. Đó là:

+ Kiểu tham chiếu và việc truyền dữ liệu cho hàm bằng tham chiếu.

+ Đối tham chiếu hằng (const)

+ Đối có giá trị mặc định

+ Hàm trực tuyến

+ Việc định nghĩa chồng các hàm

+ Việc định nghĩa chồng các toán tử

## Biến tham chiếu (Reference variable)

* 1. **Hai loại biến dùng trong C**

Trước khi nói đến biến tham chiếu, chúng ta nhắc lại 2 loại biến gặp trong C là:

* Biến giá trị dùng để chứa dữ liệu (nguyên, thực, ký tự, ... )
* Biến con trỏ dùng để chứa địa chỉ

Các biến này đều được cung cấp bộ nhớ và có địa chỉ. Ví dụ câu lệnh khai báo:

**double x , \*px;**

sẽ tạo ra biến giá trị kiểu double x và biến con trỏ kiểu double px. Biến x có vùng nhớ 8 byte, biến px có vùng nhớ 4 byte (nếu dùng mô hình Large). Biến x dùng để chứa giá trị kiểu double, ví dụ lệnh gán:

**x = 3.14;**

sẽ chứa giá trị 3.14 vào biễn x. Biến px dùng để chứa địa chỉ của một biến thực, ví dụ câu lệnh:

**px = &x ;**

sẽ lưu trữ địa chỉ của biễn x vào con trỏ px.

* 1. **Biến tham chiếu**

Trong C++ cho phép sử dụng loại biến thứ ba là biến tham chiếu. So với 2 loại biến quen biết nói trên, thì biến này có những đặc điểm sau:

+ Biến tham chiếu không được cấp phát bộ nhớ, không có địa chỉ riêng.

+ Nó dùng làm bí danh cho một biến (kiểu giá trị) nào đó và nó sử dụng vùng nhớ của biến này.

Ví dụ câu lệnh:

***float u, v, &r = u ;***

tạo ra các biến thực u, v và biến tham chiếu thực r. Biến r không được cấp phát bộ nhớ, nó là một tên khác (bí danh) của u và nó dùng chung vùng nhớ của biến u.

**Thuật ngữ:** Khi r là bí danh (alias) của u thì ta nói r tham chiếu đến biến u. Như vậy 2 thuật ngữ trên được hiểu như nhau.

**ý nghĩa:** Khi r là bí danh của u thì r dùng chung vùng nhớ của u, dó đó :

+ Trong mọi câu lệnh, viết u hay viết r đều có ý nghĩa như nhau, vì đều truy nhập đến cùng một vùng nhớ.

+ Có thể dùng biến tham chiếu để truy nhập đến một biến kiểu giá trị.

**Ví dụ:**

*int u, v, &r = u;*

*r = 10 ; // u=10*

*cout << u ; // in ra số 10*

*r++ ; // u = 11*

*++ u ; // r = 12*

*cout << r ; // in ra số 12*

*v = r ; // v=12*

*& r ; // Cho địa chỉ của u*

**Công dụng:** Biến tham chiếu thường được sử dụng làm đối của hàm để cho phép hàm truy nhập đến các tham số biến trong lời gọi hàm.

**Vài chú ý về biến tham chiếu:**

* + 1. Vì biến tham chiếu không có địa chỉ riêng, nó chỉ là bí danh của một biến kiểu giá trị nên trong khai báo phải chỉ rõ nó tham chiếu đến biến nào. Ví dụ nếu khai báo:

**double &x ;**

thì Trình biên dịch sẽ báo lỗi:

Reference variable ‘x’ must be initialized

* + 1. Biến tham chiếu có thể tham chiếu đến một phần tử mảng, ví dụ:

**int a[10] , &r = a[5];**

**r = 25 ; // a[5] = 25**

* + 1. Không cho phép khai báo mảng tham chiếu
    2. Biến tham chiếu có thể tham chiếu đến một hằng. Khi đó nó sẽ sử dụng vùng nhớ của hằng và nó có thể làm thay đổi giá trị chứa trong vùng nhớ này.

**Ví dụ** nếu khai báo:

**int &s = 23 ;**

thì Trình biên dịch đưa ra cảnh báo (warning): Temporary used to initialize 's'

Tuy nhiên chương trình vẫn làm việc. Các câu lệnh dưới đây vẫn thực hiện và cho kết quả như sau:

s++;

cout << "\ns= " << s; // In ra s=24

Chương trình dưới đây minh hoạ cách dùng biến tham chiếu đến một phần tử mảng cấu trúc để nhập dữ liệu và thực hiện các phép tính trên các trường của phần tử mảng cấu trúc.

*#include <iostream.h>*

*#include <conio.h>*

*struct TS*

*{*

*char ht[25];*

*float t,l,h,td;*

*};*

*void main()*

*{*

*TS ts[10],&h=ts[1]; // h tham chiếu đến ts[1]*

*cout << "\n Ho ten: ";*

*cin.get(h.ht,25);*

*cout << "Cac diem toan, ly, hoa: ";*

*cin >> h.t >> h.l >> h.h;*

*h.td = h.t + h.l + h.h;*

*cout << "\n Ho ten: " << ts[1].ht;*

*cout << "\n Tong diem: " << ts[1].td;*

*getch();*

*}*

* 1. **Hằng tham chiếu (const)**

Hằng tham chiếu được khai báo theo mẫu:

int n = 10 ;

const int &r = n;

Cũng giống như biến tham chiếu, hằng tham chiếu có thể tham chiếu đến một biến hoặc một hằng.

Ví dụ:

int n = 10 ;

const int &r = n ; // Hằng tham chiếu r tham chiếu đến biến n

const int &s=123 ; //Hằng tham chiếu s tham chiếu đến hằng 123

Sự khác nhau giữa biến và hằng tham chiếu ở chỗ: Không cho phép dùng hằng tham chiếu để làm thay đổi giá trị của vùng nhớ mà nó tham chiếu.

**Ví dụ:**

*int y = 12, z ;*

*const int &py=y; // Hằng tham chiếu py tham chiếu đến biến y*

*y++; // Đúng*

*z = 2\*py ; // Đúng z = 26*

*cout << y <<" "<< py; // In ra: 13 13*

*py=py+1; // Sai, Trình biên dịch thông báo lỗi:*

*// Cannot modify a const object*

**Cách dùng:** Hằng tham chiếu cho phép sử dụng giá trị chứa trong một vùng nhớ, nhưng không cho phép thay đổi giá trị này.

Hằng tham chiếu thường được sử dụng làm đối của hàm để cho phép hàm sử dụng giá trị của các tham số trong lời gọi hàm, nhưng tránh không làm thay đổi giá trị của các tham số.

## Truyền giá trị cho hàm theo tham chiếu

* 1. **Hàm trong C**

Trong C chỉ có một cách truyền dữ liệu cho hàm theo giá trị :

+ Cấp phát vùng nhớ cho các đối.

+ Gán giá trị các tham số trong lời gọi hàm cho các đối sau đó hàm làm việc trên vùng nhớ của các đối chứ không liên quan gì đến các tham số.

Như vây chương trình sẽ tạo ra các bản sao (các đối) của các tham số và hàm sẽ thao tác trên các bản sao này, chứ không làm việc trực tiếp với các tham số. Phương pháp này có 2 nhược điểm chính:

Tốn kém về thời gian và bộ nhớ vì phải tạo ra các bản sao. Không thao tác trực tiếp trên các tham số, vì vậy không làm thay đổi được giá trị các tham số.

* 1. **Truyền giá trị cho hàm theo tham chiếu**

Trong C++ cung cấp thêm cách truyền dữ liệu cho hàm theo tham chiếu bằng cách dùng đối là biến tham chiếu hoặc đối là hằng tham chiếu. Cách này có ưu điểm:

Không cần tạo ra các bản sao của các tham số, do đó tiết kiệm bộ nhớ và thời gian chạy máy.

Hàm sẽ thao tác trực tiếp trên vùng nhớ của các tham số, do đó dễ dàng thay đổi giá trị các tham số khi cần.

* 1. **Mối quan hệ giữa đối và tham số trong lời gọi hàm**

Nếu đối là biến hoặc hằng tham chiếu kiểu K thì tham số (trong lời gọi hàm) phải là biến hoặc phần tử mảng kiểu K. Ví dụ:

+ Đối là biến hoặc hằng tham chiếu kiểu double, thì tham số là biến hoặc phần tử mảng kiểu double

+ Đối là biến hoặc hằng tham chiếu kiểu cấu trúc, thì tham số là biến hoặc phần tử mảng kiểu cấu trúc

* 1. **Các chương trình minh hoạ**

/\*

Chương trình sau được tổ chức thành 3 hàm: Nhập dãy số double

Hoán vị 2 biến double

Sắp xếp dẫy số double theo thứ tự tăng dần

Chương trình sẽ nhập một dẫy số và in dẫy sau khi sắp xếp

\*/

*#include <iostream.h>*

*#include <conio.h>*

*#include <stdio.h>*

*void nhapds(double \*a, int n)*

*{*

*for (int i=1; i<= n ; ++i)*

*{*

*cout << "\nPhan tu thu " << i << " : " ;*

*cin >> a[i] ;*

*}*

*}*

*void hoanvi(double &x, double &y)*

*{*

*double tg=x;*

*x=y;*

*y= tg;*

*}*

*void sapxep(double \* a, int n)*

*{*

*for (int i=1; i <= n-1 ;++i)*

*{*

*for (int j=i+1 ; j<=n ;++j)*

*{*

*if (a[i] > a[j])*

*{*

*hoanvi(a[i],a[j]);*

*}*

*}*

*}*

*}*

*void main()*

*{*

*double x[100];*

*int i, n;*

*cout <<"\n N= "; cin >> n;*

*nhapds(x,n);*

*sapxep(x,n);*

*for (i=1;i<=n;++i)*

*{*

*printf("\n%0.1lf",x[i]);*

*}*

*getch();*

*}*

*/\**

Chương trình sau gồm các hàm:

* Nhập dẫy cấu trúc (mỗi cấu trúc chứa dữ liệu một thí sinh)
* Hoán vị 2 biến cấu trúc
* Sắp xếp dẫy thí sinh theo thứ tự giảm của tổng điểm
* In một cấu trúc (in họ tên và tổng điểm)

Chương trình sẽ nhập dữ liệu một danh sách thí sinh, nhập điểm chuẩn và in danh sách thí sinh trúng tuyển

*\*/*

*#include <iostream.h>*

*#include <iomanip.h>*

*#include <conio.h>*

*struct TS*

*{*

*char ht[20];*

*float t,l,h,td;*

*};*

*void ints(const TS &ts)*

*{*

*cout << setiosflags(ios::showpoint) << setprecision(1) ;*

*cout << "\nHo ten: " << setw(20) << ts.ht << setw(6) << ts.td ;*

*}*

*void nhapsl(TS \*ts,int n)*

*{*

*for (int i=1;i<=n;++i)*

*{*

*cout << "\n Thi sinh " << i ;*

*cout << "\n Ho ten: " ;*

*cin.ignore(1);*

*cin.get(ts[i].ht,25) ;*

*cout << "Cac diem toan, ly, hoa: ";*

*cin >> ts[i].t >> ts[i].l >> ts[i].h ;*

*ts[i].td = ts[i].t + ts[i].l + ts[i].h ;*

*}*

*}*

*void hvts(TS &ts1, TS &ts2)*

*{*

*TS tg=ts1;*

*ts1=ts2;*

*ts2=tg;*

*}*

*void sapxep(TS \*ts,int n)*

*{*

*for (int i=1;i<=n-1;++i)*

*{*

*for (int j=i+1;j<=n;++j)*

*{*

*if (ts[i].td < ts[j].td)*

*{*

*hvts(ts[i],ts[j]);*

*}*

*}*

*}*

*}*

*void main()*

*{*

*TS ts[100];*

*int n,i;*

*clrscr();*

*cout << " So thi sinh: " ; cin >> n ;*

*nhapsl(ts,n);*

*sapxep(ts,n) ;*

*float dc;*

*cout << " Diem chuan: " ; cin >> dc;*

*cout << "\n\nDanh sach trung tuyen\n" ;*

*for (i=1;i<=n;++i)*

*{*

*if (ts[i].td >= dc)*

*{*

*ints(ts[i]);*

*}*

*else*

*{*

*break;*

*}*

*}*

*getch();*

*}*

*/\**

Chương trình sau gồm các hàm:

Nhập một ma trận thực cấp mxn

In một ma trận thực dưới dạng bảng

Tìm phần tử lớn nhất và phần tử nhỏ nhất của dẫy số thưc;

Chương trình sẽ nhập một ma trận, in ma trận vừa nhập và in các phần tử lớn nhất và nhỏ nhất trên mỗi hàng của ma trận

*\*/*

*#include <iostream.h>*

*#include <iomanip.h>*

*#include <conio.h>*

*#include <stdio.h>*

*void nhapmt(float a[20][20], int m, int n)*

*{*

*for (int i=1 ; i<= m ; ++i)*

*{*

*for (int j=1; j<= n ; ++j)*

*{*

*cout << "\na[" << i << "," << j << "]= " ;*

*cin >> a[i][j] ;*

*}*

*}*

*}*

*void inmt(float a[20][20], int m, int n)*

*{*

*cout << setiosflags(ios::showpoint) << setprecision(1);*

*for (int i=1 ; i<= m ; ++i)*

*{*

*for (int j=1; j<= n ; ++j)*

*{*

*if (j==1)*

*{*

*cout << "\n" ;*

*cout << setw(6) << a[i][j] ;*

*}*

*}*

*}*

*}*

*void maxminds(float \*x, int n,int &vtmax, int &vtmin)*

*{*

*vtmax = vtmin = 1 ;*

*for (int i=2; i<=n ; ++i)*

*{*

*if (x[i] > x[vtmax]) vtmax = i;*

*if (x[i] < x[vtmin]) vtmin = i;*

*}*

*}*

*void main()*

*{*

*float a[20][20];*

*int m, n;*

*cout <<"\n So hamg va so cot ma tran: "; cin >> m >> n;*

*nhapmt(a,m,n);*

*clrscr();*

*inmt(a,m,n);*

*float \*p = (float\*)a;*

*int vtmax, vtmin;*

*for (int i=1;i<=m;++i)*

*{*

*p = ((float\*)a) + i\*20 ;*

*maxminds(p , n, vtmax, vtmin) ;*

printf("\nHang %d Phan tu max= %6.1f tai cot

%d",i,p[vtmax],vtmax);

*printf("\n Phan tu min= %6.1f tai cot %d", p[vtmin],vtmin);*

*}*

*getch();*

*}*

## Hàm trả về các tham chiếu

Hàm có thể có kiểu tham chiếu và trả về giá trị tham chiếu. Khi đó có thể dùng hàm để truy nhập đến một biến hoặc một phần tử mảng nào đó. Dưới đây là một số ví dụ.

**Ví dụ 1** trình bầy một hàm trả về một tham chiếu đến một biến toàn bộ. Do đó có thể dùng hàm để truy nhập đến biến này.

*#include <iostream.h>*

*#include <conio.h>*

*int z ;*

*int &f() // Hàm trả về một bí danh của biến toàn bộ z*

*{*

*return z;*

*}*

*void main(void)*

*{*

*f()=50; // z = 50*

*cout <<"\nz= " << z;*

*getch();*

*}*

**Ví dụ 2** trình bày một hàm trả về bí danh của một biến cấu trúc toàn bộ. Khác với ví dụ trên, ở đây không dùng hàm một cách trực tiếp mà gán hàm cho một biến tham chiếu, sau đó dùng biến tham chiếu này để truy nhập đến biến cấu trúc toàn bộ.

*#include <iostream.h>*

*#include <conio.h>*

*struct TS*

*{*

*char ht[25];*

*float t,l,h,td;*

*};*

*TS ts;*

*TS &f()*

*{*

*return ts;*

*}*

*void main()*

*{*

*TS &h=f(); // h tham chiếu đến biến ts*

*cout << "\n Ho ten: " ;*

*cin.get(h.ht,25) ;*

*cout << "Cac diem toan, ly, hoa: ";*

*cin >> h.t >> h.l >> h.h ;*

*h.td = h.t + h.l + h.h ;*

*cout << "\n Ho ten: " << ts.ht;*

*cout << "\n Tong diem: " << ts.td;*

*getch();*

*}*

**Ví dụ 3** trình bày một hàm trả về bí danh của một phần tử mảng cấu trúc toàn bộ.

Hàm sẽ kiểm tra xem chỉ số mảng có vượt ra ngoài miền quy định hay không. Sau đó dùng hàm này để truy nhập đến các phần tử mảng cấu trúc.

*#include <iostream.h>*

*#include <conio.h>*

*#include <stdlib.h>*

*struct TS*

*{*

*char ht[25];*

*float t,l,h,td;*

*};*

*TS \*ts;*

*void cap\_phat\_bo\_nho\_nhapsl(int n)*

*{*

*ts = new TS[n+1] ;*

*if (ts==NULL)*

*{*

*cout << "Loi cap phat bo nho " ; exit(1);*

*}*

*for (int i=1;i<=n;++i)*

*{*

*TS &h=ts[i];*

*cout << "\nThi sinh thu " << i ;*

*cout << "\n Ho ten: " ;*

*cin.ignore(1);*

*cin.get(h.ht,25) ;*

*cout << "Cac diem toan, ly, hoa: ";*

*cin >> h.t >> h.l >> h.h ;*

*h.td = h.t + h.l + h.h ;*

*}*

*}*

*TS &f(int i, int n) // Cho bi danh ts[i]*

*{*

*if (i<1 || i>n)*

*{*

*cout << "Chi so mang khong hop le " ;*

*exit(1);*

*}*

*return ts[i];*

*}*

*void main()*

*{*

*int n, i ;*

*cout << "\n So thi sinh : "; cin >> n;*

*cap\_phat\_bo\_nho\_nhapsl(n);*

*while (1)*

*{*

*cout << "\nCan xem thi sinh thu may: " ;*

*cout << "\nChon so tu 1 den " << n << "(bam sai ket thuc CT) ";*

*cin >> i;*

*TS &h=f(i,n);*

*cout << "\n Ho ten: " << h.ht;*

*cout << "\n Tong diem: " << h.td;*

*}*

*}*

## Đối có giá trị mặc định

* 1. **Thế nào là đối mặc định**

Một trong các khả năng mạnh của C++ là nó cho phép xây dựng hàm với các đối có giá trị mặc định. Thông thường số tham số trong lời gọi hàm phải bằng số đối của hàm. Mỗi đối sẽ được khởi gán giá trị theo tham số tương ứng của nó. Trong C++ cho phép tạo giá trị mặc định cho các đối. Các đối này có thể có hoặc không có tham số tương ứng trong lời gọi hàm. Khi không có tham số tương ứng, đối được khởi gán bởi giá trị mặc định.

**Ví dụ:** hàm delay với đối số mặc định được viết theo một trong 2 cách sau:

**Cách 1**(Không khai báo nguyên mẫu):

*void delay(int n=1000)*

*{*

*for (int i=0 ; i<n ; ++i)*

*{*

*…;*

*}*

*}*

**Cách 2**(Có khai báo nguyên mẫu):

*void delay(int n=1000) ;*

*void delay(int n)*

*{*

*for (int i=0 ; i<n ; ++i)*

*{*

*…;*

*}*

*}*

**Cách dùng:**

+ Cung cấp giá trị cho đối n (Có tham số trong lời gọi hàm)

delay(5000) ; // Đối n = 5000

+ Sử dụng giá trị mặc định của đối (Không có tham số trong lời gọi)

delay() ; // Đối n = 1000

* 1. **Quy tắc xây dựng hàm với đối mặc định**

+ Các đối mặc định cần phải là các đối cuối cùng tính từ trái sang phải. Giả sử có 5 đối theo thứ tự từ trái sang phải là

d1, d2, d3, d4, d5 Khi đó:

nếu một đối mặc định thì phải là d5

nếu hai đối mặc định thì phải là d4, d5 nếu ba đối mặc định thì phải là d3, d4, d5

...

Các ví dụ sai:

d3 và d5 mặc định (khi đó d4 cũng phải mặc định)

d3 và d4 mặc định (khi đó d5 cũng phải mặc định)

+ Khi xây dựng hàm, nếu sử dụng khai báo nguyên mẫu, thì các đối mặc định cần được khởi gán trong nguyên mẫu, ví dụ:

// Khởi gán giá trị cho 3 đối mặc định d3, d4 và d5)

*void f(int d1, float d2, char \*d3=”HA NOI”, int d4 = 100, double d5=3.14);*

*void f(int d1, float d2, char \*d3, int d4, double d5)*

*{*

*// Các câu lệnh trong thân hàm*

*}*

+ Không được khởi gán lại cho các đối mặc định trong dòng đầu của định nghĩa hàm. Nếu vi phạm điều này thì Chương trình dịch sẽ thông báo lỗi.

+ Khi xây dựng hàm, nếu không khai báo nguyên mẫu, thì các đối mặc định được khởi gán trong dòng đầu của định nghĩa hàm, ví dụ:

// Khởi gán giá trị cho 3 đối mặc định d3, d4 và d5)

*void f(int d1, float d2, char \*d3=”HA NOI”, int d4 = 100, double d5=3.14)*

*{*

*// Các câu lệnh trong thân hàm*

*}*

+ Giá trị dùng để khởi gán cho đối mặc định

Có thể dùng các hằng, các biến toàn bộ, các hàm để khởi gán cho đối mặc định, ví dụ:

*int MAX = 10000;*

*void f(int n, int m = MAX, int xmax = getmaxx(), int ymax = getmaxy());*

* 1. **Cách sử dụng hàm có đối mặc định**

Lời gọi hàm cần viết theo quy định sau:

Các tham số thiếu vắng trong lời gọi hàm phải tương ứng với các đối mặc định cuối cùng (tính từ trái sang phải).

Nói cách khác: Đã dùng giá trị mặc định cho một đối (tất nhiên phải là đối mặc định) thì cũng phải sử dụng giá trị mặc định cho các đối còn lại.

Ví dụ với hàm có 3 đối mặc định:

*void f(int d1, float d2, char \*d3=”HA NOI”, int d4 = 100, double d5=3.14) ;*

Thì các lời gọi sau là đúng:

*f(3, 3.4, ”ABC”, 10, 1.0) ; // Đầy đủ tham số*

*f(3, 3.4, ”ABC”) ; // Thiếu 2 tham số cuối*

*f(3, 3.4) ; // Thiếu 3 tham số cuối*

Các lời gọi sau là sai:

*f(3) ; // Thiếu tham số cho đối không mặc định d2*

*f(3, 3.4, , 10) ; // Đã dùng giá trị mặc định cho d3,*

*// thì cũng phải dùng giá trị mặc định cho d4 và d5*

* 1. **Các ví dụ**

Hàm ht (bên dưới) dùng để hiển thị chuỗi ký tự dc trên n dòng màn hình. Các đối dc và n đều có giá trị mặc định.

*#include <conio.h>*

*#include <iostream.h>*

*void ht(char \*dc="HA NOI",int n=10) ;*

*void ht(char \*dc , int n )*

*{*

*for (int i=0; i<n; ++i) cout << "\n" << dc;*

*}*

*void main()*

*{*

*ht(); // In dòng chữ “HA NOI” trên 10 dòng*

*ht("ABC",3); // In dòng chữ “ABC” trên 3 dòng*

*ht("DEF"); // In dòng chữ “DEF” trên 10 dòng*

*getch();*

*}*

**Ví dụ** dưới đây trình bày hàm hiển thị một chuỗi str trên màn hình đồ hoạ, tại vị trí (x,y) và có mầu m. Các đối x, y và m là mặc định. Dùng các hàm getmaxx() và getmaxy() để khởi gán cho x, y. Dùng hằng RED gán cho m.

*#include <conio.h>*

*#include <graphics.h>*

*void hiendc(char \*str, int x=getmaxx()/2,*

*int y = getmaxy()/2, int m=RED);*

*void hiendc(char \*str, int x,int y, int m)*

*{*

*int mau\_ht = getcolor(); // Luu mau hien tai setcolor(m);*

*outtextxy(x,y,str) ;*

*setcolor(mau\_ht); // Khoi phuc mau hien tai*

*}*

*void main()*

*{*

*int mh=0, mode=0;*

*initgraph(&mh,&mode,"");*

*setbkcolor(BLUE);*

*hiendc("HELLO"); // HELLO mầu đỏ giữa màn hình*

*hiendc("CHUC MUNG",1,1); // CHUC MUNG mầu đỏ tại vị trí (1,1)*

*hiendc("CHAO",1,400,YELLOW); // CHAO mầu vàng tại vị trí (1,400)*

*getch();*

*}*

**Ví dụ** dưới đây trình bầy hàm tính tích phân xác định gồm 3 đối: f là hàm cần tính tích phân, a và b là các cận dưới và trên (a<b). Cả 3 đối f, a và b đều mặc định. Giá trị mặc định của con trỏ hàm f là địa chỉ của hàm bp (bình phương), của a bằng 0, của b bằng 1.

*#include <conio.h>*

*#include <iostream.h>*

*#include <iomanip.h>*

*#include <math.h>*

*double bp(double x);*

*double tp(double (\*f)(double)=bp, double a=0.0, double b=1.0);*

*double bp(double x)*

*{*

*return x\*x;*

*}*

*double tp(double (\*f)(double), double a, double b)*

*{*

*int n=1000;*

*double s=0.0, h=(b-a)/n;*

*for (int i=0; i<n ; ++i)*

*{*

*s+= f(a+i\*h + h) + f(a+i\*h );*

*return s\*h/2;*

*}*

*}*

*void main()*

*{*

*clrscr();*

*cout << setiosflags(ios::showpoint) << setprecision(2);*

*cout << "\nTich phan tu 0 den 1 cua x\*x= " << tp() ;*

*cout << "\nTich phan tu 0 den 1 cua exp(x)= " << tp(exp);*

*cout << "\nTich phan tu 0 den PI/2 cua sin(x) " << tp(sin,0,3.14/2);*

*getch();*

*}*

## Các hàm trực tuyến (inline)

* 1. **Ưu, nhược điểm của hàm**

Việc tổ chức chương trình thành các hàm có 2 ưu điểm rõ rệt :

* **Thứ nhất** là chia chương trình thành các đơn vị độc lập, làm cho chương trình được tổ chức một cách khoa học dễ kiểm soát dễ phát hiện lỗi, dễ phát triển, mở rộng.
* **Thứ hai** là giảm được kích thước chương trình, vì mỗi đoạn chương trình thực hiện nhiệm vụ của hàm được thay bằng một lời gọi hàm.

Tuy nhiên hàm cũng có nhược điểm:

* Làm chậm tốc độ chương trình do phải thực hiện một số thao tác có tính thủ tục mỗi khi gọi hàm như:

+ Cấp phát vùng nhớ cho các đối và biến cục bộ.

+ Truyền dữ liệu của các tham số cho các đối.

+ Giải phóng vùng nhớ trước khi thoát khỏi hàm.

Các hàm trực tuyến trong C++ cho khả năng khắc phục được nhược điểm nói trên.

* 1. **Các hàm trực tuyến**

Để biến một hàm thành trực tuyến ta viết thêm từ khoá inline vào trước khai báo nguyên mẫu hàm. Nếu không dùng nguyên mẫu thì viết từ khoá này trước dòng đầu tiên của định nghĩa hàm.

Ví dụ:

*inline float f(int n, float x);*

*float f(int n, float x)*

*{*

*// Các câu lệnh trong thân hàm*

*}*

hoặc

*inline float f(int n, float x)*

*{*

*// Các câu lệnh trong thân hàm*

*}*

***Chú ý:*** Trong mọi trường hợp, từ khoá inline phải xuất hiện trước các lời gọi hàm thì Trình biên dịch mới biết cần xử lý hàm theo kiểu inline.

**Ví dụ:** hàm f trong chương trình sau sẽ không phải là hàm trực tuyến vì từ khoá “inline” viết sau lời gọi hàm:

*#include <conio.h>*

*#include <iostream.h>*

*void main()*

*{*

*int s;*

*s = f(5,6);*

*cout << s;*

*getch();*

*}*

*inline int f(int a, int b)*

*{*

*return a\*b;*

*}*

***Chú ý:*** Trong C++ , nếu hàm được xây dựng sau lời gọi hàm thì bắt buộc phải khai báo nguyên mẫu hàm trước lời gọi. Trong ví dụ trên, Trình biên dịch C++ sẽ bắt lỗi vì thiếu khai báo nguyên mẫu hàm f .

* 1. **Cách biên dịch hàm trực tuyến**

Chương trình dịch xử lý các hàm inline như các macro (được định nghĩa trong lệnh #define), nghĩa là nó sẽ thay mỗi lời gọi hàm bằng một đoạn chương trình thực hiện nhiệm vụ của hàm. Cách này làm cho chương trình dài ra, nhưng tốc độ chương trình tăng lên do không phải thực hiện các thao tác có tính thủ tục khi gọi hàm.

* 1. **So sánh macro và hàm trực tuyến**

Dùng macro và hàm trực tuyến đều dẫn đến hiệu quả tương tự, tuy nhiên người ta thích dùng hàm trực tuyến hơn, vì cách này đảm bảo tính cấu trúc của chương trình, dễ sử dụng và tránh được các sai sót lặt vặt thường gặp khi dùng #define (như thiếu các dấu ngoặc, dấu chấm phẩy)

Khi nào thì nên dùng hàm trực tuyến

Phương án dùng hàm trực tuyến rút ngắn được thời gian chạy máy nhưng lại làm tăng khối lượng bộ nhớ chương trình (nhất là đối với các hàm trực tuyến có nhiều câu lệnh). Vì vậy chỉ nên dùng phương án trực tuyến đối với các hàm nhỏ.

* 1. **Sự hạn chế của Trình biên dịch**

Không phải khi gặp từ khoá inline là Trình biên dịch nhất thiết phải xử lý hàm theo kiểu trực tuyến.

***Chú ý*** rằng từ khoá inline chỉ là một sự gợi ý cho Trình biên dịch chứ không phải là một mệnh lệnh bắt buộc.

Có một số hàm mà các Trình biên dịch thường không xử lý theo cách inline như các hàm chứa biến static, hàm chứa các lệnh chu trình hoặc lệnh goto hoặc lệnh switch, hàm đệ quy. Trong trường hợp này từ khoá inline lẽ dĩ nhiên bị bỏ qua.

Thậm chí từ khoá inline vẫn bị bỏ qua ngay cả đối với các hàm không có những hạn chế nêu trên nếu như Trình biên dịch thấy cần thiết (ví dụ đã có quá nhiều hàm inline làm cho bộ nhớ chương trình quá lớn)

**Ví dụ:** Chương trình sau sử dụng hàm inline tính chu vi và diện tích của hình chữ nhật:

**Phương án 1:**Không khai báo nguyên mẫu. Khi đó hàm dtcvhcn phải đặt trên hàm main.

*#include <conio.h>*

*#include <iostream.h>*

*inline void dtcvhcn(int a, int b, int &dt, int &cv)*

*{*

*dt=a\*b;*

*cv=2\*(a+b);*

*}*

*void main()*

*{*

*int a[20], b[20], cv[20], dt[20], n;*

*cout << "\n So hinh chu nhat: ";*

*cin >> n;*

*for (int i=1; i<=n; ++i)*

*{*

*cout << "\nNhap 2 canh cua hinh chu nhat thu" << i << ":";*

*cin >> a[i] >> b[i];*

*dtcvhcn(a[i], b[i], dt[i], cv[i]);*

*}*

*clrscr();*

*for (i=1;i<=n;++i)*

*{*

*cout << "\n Hinh chu nhat thu " << i << " : ";*

*cout << "\nDo dai 2 canh= " << a[i] << " va " << b[i];*

*cout << "\nDien tich= " << dt[i];*

*cout << "\nChu vi= " << cv[i];*

*}*

*getch();*

*}*

**Phương án 2:**Sử dụng khai báo nguyên mẫu. Khi đó từ khoá inline đặt trước nguyên mẫu.

***Chú ý:*** Không được đặt inline trước định nghĩa hàm. Trong chương trình dưới đây nếu đặt inline trước định nghĩa hàm thì hậu quả như sau: Chương trình vẫn dịch thông, nhưng khi chạy thì chương trình bị quẩn, không thoát được.

*#include <conio.h>*

*#include <iostream.h>*

*inline void dtcvhcn(int a, int b, int &dt, int &cv);*

*void main()*

*{*

*int a[20],b[20],cv[20],dt[20],n;*

*cout << "\n So hinh chu hat: " ;*

*cin >> n;*

*for (int i=1;i<=n;++i)*

*{*

*cout << "\nNhap 2 canh cua hinh chu nhat thu " <<i<< ": ";*

*cin >> a[i] >> b[i] ;*

*dtcvhcn(a[i],b[i],dt[i],cv[i]);*

*}*

*clrscr();*

*for (i=1;i<=n;++i)*

*{*

*cout << "\n Hinh chu nhat thu " << i << " : ";*

*cout << "\nDo dai 2 canh= " << a[i] << " va " << b[i] ;*

*cout << "\nDien tich= " << dt[i] ;*

*cout << "\nChu vi= " << cv[i] ;*

*}*

*getch();*

*}*

*void dtcvhcn(int a, int b, int &dt, int &cv)*

*{*

*dt=a\*b; cv=2\*(a+b);*

*}*

## Định nghĩa chồng các hàm (Overloading)

* 1. **Khái niệm về định nghĩa chồng**
* Định nghĩa chồng (hay còn gọi sự tải bội) các hàm là dùng cùng một tên để định nghĩa các hàm khác nhau. Đây là một mở rộng rất có ý nghĩa của C++.
* Như đã biết, trong C và các ngôn ngữ khác (như PASCAL, FOXPRO,...) mỗi hàm đều phải có một tên phân biệt. Đôi khi đây là một sự hạn chế lớn, vì phải dùng nhiều hàm khác nhau để thực hiện cùng một công việc. Ví dụ để lấy giá trị tuyệt đối trong C cần dùng tới 3 hàm khác nhau:

**int abs(int i); // Lấy giá trị tuyệt đối giá trị kiểu int**

**longt labs(longt l); // Lấy giá trị tuyệt đối giá trị kiểu long**

**double fabs(double d); // Lấy giá trị tuyệt đối giá trị kiểu double**

* Nhờ khả năng định nghĩa chồng, trong C++ có thể dùng chung một tên cho cả 3 hàm trên như sau:

**int abs(int i) ; // Lấy giá trị tuyệt đối giá trị kiểu int**

**longt abs(longt l) ; // Lấy giá trị tuyệt đối giá trị kiểu long**

**double abs(double d) ; // Lấy giá trị tuyệt đối giá trị kiểu double**

* 1. **Yêu cầu về các hàm định nghĩa chồng**
* Khi dùng cùng một tên để định nghĩa nhiều hàm, Trình biên dịch C++ sẽ dựa vào sự khác nhau về tập đối của các hàm này để đổi tên các hàm. Như vậy, sau khi biên dịch mỗi hàm sẽ có một tên khác nhau.
* Từ đó cho thấy: các hàm được định nghĩa trùng tên phải có tập đối khác nhau (về số lượng hoặc kiểu).
* Nếu 2 hàm hoàn toàn trùng tên và trùng đối 🡪 Trình biên dịch sẽ không có cách nào phân biệt được. Ngay cả khi 2 hàm này có kiểu khác nhau thì Trình biên dịch vẫn báo lỗi.

**Ví dụ**: Xây dựng 2 hàm:

* Cùng có tên là f và cùng có một đối nguyên a
* Kiểu hàm khác nhau. Hàm thứ nhất kiểu nguyên (trả về a\*a), hàm thứ hai kiểu void (in giá trị a).

🡪Chương trình sẽ bị thông báo lỗi khi biên dịch (bạn hãy thử xem sao)

*#include <conio.h>*

*#include <iostream.h>*

*int f(int a)*

*{*

*return a\*a;*

*}*

*void f(int a)*

*{*

*cout << "\n " << a;*

*}*

*void main()*

*{*

*int b=f(5); f(b);*

*getch();*

*}*

* 1. **Sử dụng các hàm định nghĩa chồng**

Khi gặp một lời gọi, Trình biên dịch sẽ căn cứ vào số lượng và kiểu của các tham số để gọi hàm có đúng tên và đúng bộ đối số tương ứng.

**Ví dụ:**

**abs(123); // Tham số kiểu int, gọi hàm int abs(int i) ;**

**abs(123L); // Tham số kiểu long, gọi hàm long abs(long l);**

**abs(3.14); //Tham số kiểu double, gọi hàm double abs(double d);**

Khi không có hàm nào có bộ đối cùng kiểu với bộ tham số (trong lời gọi) 🡪 Trình biên dịch sẽ chọn hàm nào có bộ đối gần kiểu nhất (phép chuyển kiểu dễ dàng nhất).

**Ví dụ:**

**abs(‘A’) ; // Tham số kiểu char, gọi hàm int abs(int i) ;**

**abs(3.14F); // Tham số kiểu float, gọi hàm double abs(double d);**

* 1. **Nên sử dụng phép định nghĩa chồng các hàm như thế nào**
* Như đã nói ở trên, khi xây dựng cũng như sử dụng các hàm trùng tên, Trình biên dịch C++ đã phải suy đoán và giải quyết nhiều trường hợp khá nhập nhằng.
* Vì vậy không nên lạm dụng quá đáng khả năng định nghĩa chồng, vì điều đó làm cho chương trình khó kiểm soát và dễ dẫn đến sai sót. Việc định nghĩa chồng sẽ hiệu quả hơn nếu được sử dụng theo các lời khuyên sau:
* Chỉ nên định nghĩa chồng các hàm thực hiện những công việc như nhau nhưng trên các đối tượng có kiểu khác nhau.

**Ví dụ:** trong chương trình cần xây dựng các hàm:

cộng 2 ma trận vuông kiểu double

cộng 2 ma trận vuông kiểu int

cộng 2 ma trân chữ nhật kiểu double

cộng 2 ma trận chữ nhật kiểu int

thì 4 hàm trên nên định nghĩa chồng (đặt cùng tên).

* Nên dùng các phép chuyển kiểu (nếu cần) để bộ tham số trong lời gọi hoàn toàn trùng kiểu với bộ đối số của một hàm được định nghĩa chồng. Vì như thế mới tránh được sự nhập nhằng cho Trình biên dịch và Trình biên dịch sẽ chọn đúng hàm cần gọi.
  1. **Lấy địa chỉ các hàm trùng tên**

Giả sử có 4 hàm đều có tên là tinh\_max được khai báo như sau:

*int tinh\_max(int a, int b, int c) ; // Max của 3 số nguyên*

*double tinh\_max(double a, double b, double c); // Max của 3 số // thực*

*int tinh\_max(int \*a, int n) ; // Max của một dẫy số nguyên*

*double tinh\_max(double \*a, int n) ; //Max của một dẫy số thực*

Vấn đề đặt ra là làm thế nào lấy được địa chỉ của mỗi hàm.

**Câu trả lời như sau:**

Để lấy địa chỉ của một hàm, ta khai báo một con trỏ hàm có kiểu và bộ đối như hàm cần lấy địa chỉ. Sau đó gán tên hàm cho con trỏ hàm. Ví dụ:

*int (\*f1)(int , int, int );*

*f1 = tinh\_max ;//Lấy địa chỉ của hàm thứ nhất double (\*f2)(double , double, double);*

*f2 = tinh\_max ; // Lấy địa chỉ của hàm thứ hai int (\*f3)(int \*, int );*

*f3 = tinh\_max ; // Lấy địa chỉ của hàm thứ ba double (\*f4)(double \*, int );*

*f4 = tinh\_max ; // Lấy địa chỉ của hàm thứ tư*

## Các ví dụ

**Ví dụ 1:** Chương trình giải bài toán tìm max của một dẫy số nguyên và max của một dẫy số thực. Trong chươmg trình có 6 hàm. Hai hàm dùng để nhập dẫy số nguyên và dẫy số thực có tên chung là nhapds. Bốn hàm: tính max 2 số nguyên, tính max 2 số thực, tính max của dẫy số nguyên, tính max của dẫy số thực được đặt chung một tên là max.

#include <conio.h> #include <iostream.h> #include <iomanip.h> void nhapds(int \*x, int n);

void nhapds(double \*x, int n); int max(int x, int y);

double max(double x, double y); int max(int \*x, int n);

double max(double \*x, int n); void nhapds(int \*x, int n)

{

for (int i=1;i<=n;++i)

{

cout << "Phan tu " << i << " = " ; cin >> x[i] ;

}

}

void nhapds(double \*x, int n)

{

for (int i=1;i<=n;++i)

{

cout << "Phan tu " << i << " = " ; cin >> x[i] ;

}

}

int max(int x, int y)

{

return x>y?x:y ;

}

double max(double x, double y)

{

return x>y?x:y ;

}

int max(int \*x, int n)

{

int s=x[1];

for (int i=2;i<=n;++i) s = max(s,x[i]);

return s;

}

double max(double \*x, int n)

{

double s=x[1];

for (int i=2;i<=n;++i) s = max(s,x[i]);

return s;

}

void main()

{

int a[20] , n , ni, nd, maxi ; double x[20] , maxd ; clrscr();

cout << "\nSo phan tu nguyen ni = " ; cin >> ni ;

cout << "Nhap day so nguyen\n " ; nhapds(a,ni);

cout << "\nSo phan tu thuc nd = " ; cin >> nd ;

cout << "Nhap day so thuc\n " ; nhapds(x,nd);

maxi = max(a,ni); maxd = max(x,nd);

cout << "\nMax cua day nguyen = " << maxi ; cout << "\nMax cua day thuc = " << maxd ; getch();

}

## Ví dụ 2:

Chương trình sau thực hiện phép nhân ma trận:

D = A\*B\*C

trong đó A, B là các ma trận vuông, C là ma trận chữ nhật. Trong chương trình có 3 cặp hàm trùng tên để thực hiện 3 nhiệm vụ (nhưng trên 2 đối tượng khác nhau là ma trận vuông và chữ nhật): Nhập ma trận, nhân 2 ma trận và in ma trân.

#include <conio.h> #include <iostream.h> #include <iomanip.h> typedef int MT[20][20];

void nhapmt(MT a,char \*ten, int m, int n); void inmt(MT a,char \*ten, int m, int n);

void nhanmt(MT a,MT b, MT c, int m, int n, int p); void nhapmt(MT a,char \*ten, int n);

void inmt(MT a,char \*ten, int n);

void nhanmt(MT a,MT b, MT c, int n); void nhapmt(MT a, char \*ten, int m, int n)

{

for (int i=1;i<=m;++i) for (int j=1;j<=n;++j)

{

cout << "\n" << ten <<"[" << i << "," << j << "]= " ;

cin >> a[i][j];

}

}

void nhapmt(MT a,char \*ten, int n)

{

nhapmt(a,ten,n,n) ;

}

void inmt(MT a,char \*ten, int m, int n)

{

cout << "\nMa tran: " << ten; for (int i=1;i<=m;++i)

{

cout << "\n" ;

for (int j=1;j<=n;++j)

cout << setw(6) << a[i][j];

}

}

void inmt(MT a,char \*ten, int n)

{

inmt(a,ten,n,n) ;

}

void nhanmt(MT a,MT b, MT c, int m, int n, int p)

{

for (int i=1;i<=m;++i) for (int j=1;j<=p;++j)

{

c[i][j]=0;

for (int k=1;k<=n;++k) c[i][j] += a[i][k] \* b[k][j];

}

}

void nhanmt(MT a,MT b, MT c, int n)

{

nhanmt(a,b,c,n,n, n) ;

}

void main()

{

MT a,b,c,d; // d= abc MT u;

clrscr(); nhapmt(a,"A",2);

nhapmt(b,"B",2);

nhapmt(c,"C",2,3);

nhanmt(a,b,u,2);

nhanmt(u,c,d,2,2,3);

inmt(a,"A",2);

inmt(b,"B",2);

inmt(u,"U = A\*B",2);

inmt(c,"C",2,3);

inmt(d,"D = U\*C",2,3); getch();

}

## Định nghĩa chồng các toán tử

* 1. **Các phép toán trong C và C++**

Trong C và C++ có khá nhiều các phép toán dùng để thực hiện các thao tác trên các kiểu dữ liệu chuẩn. Ví dụ các phép số học: + - \* / áp dụng cho các kiểu dữ liệu nguyên, thực. Phép lấy phần dư % áp dụng đối với kiểu nguyên.

## Thực hiện các phép toán trên các kiểu dữ liệu không chuẩn trong C

Việc thực hiện các phép toán trên các đối tượng tự định nghĩa (như mảng, cấu trúc) là nhu cầu bắt buộc của thực tế. Chẳng hạn cần thực hiện các phép số học trên số phức, trên phân số, trên đa

thức, trên véc tơ, trên ma trận. Để đáp ứng yêu cầu này, ta sử dụng các hàm trong C. Ví dụ sau đây là một chương trình C gồm các hàm nhập phân số, in phân số và thực hiện các phép cộng trừ nhân chia phân số. Chương trình sẽ nhập 5 phân số: p, q, z, u, v và tính phân số s theo công thức:

s = (p – q\*z)/(u + v) #include <conio.h> #include <stdio.h> #include <math.h> typedef struct

{

int a,b;

} PS;

void nhap(PS \*p); void in(PS p);

int uscln(int x, int y);

PS rutgon(PS p);

PS cong(PS p1, PS p2); PS tru(PS p1, PS p2);

PS nhan(PS p1, PS p2); PS chia(PS p1, PS p2); void nhap(PS \*p)

{

int t, m;

printf("\nTu va mau: "); scanf("%d%d", &t, &m); p->a = t; p->b = m;

}

void in(PS p)

{

printf(" %d/%d",p.a,p.b);

}

int uscln(int x, int y)

{

x=abs(x); y=abs(y); if (x\*y==0) return 1; while (x!=y)

if (x>y) x-=y; else y-=x;

return x;

}

PS rutgon(PS p)

{

PS q;

int x; x=uscln(p.a,p.b);

q.a = p.a / x ;

q.b = p.b / x ; return q;

}

PS cong(PS p1, PS p2)

{

PS q;

q.a = p1.a\*p2.b + p2.a\*p1.b;

q.b = p1.b \* p2.b ; return rutgon(q);

}

PS tru(PS p1, PS p2)

{

PS q;

q.a = p1.a\*p2.b - p2.a\*p1.b;

q.b = p1.b \* p2.b ; return rutgon(q);

}

PS nhan(PS p1, PS p2)

{

PS q;

q.a = p1.a \* p2.a ;

q.b = p1.b \* p2.b ; return rutgon(q);

}

PS chia(PS p1, PS p2)

{

PS q;

q.a = p1.a \* p2.b ;

q.b = p1.b \* p2.a ; return rutgon(q);

}

void main()

{

PS p, q, z, u, v ; PS tu,mau, s;

printf("\n Nhap phan so p: "); nhap(&p); printf("\n Nhap phan so q: ");nhap(&q); printf("\n Nhap phan so z: ");nhap(&z); printf("\n Nhap phan so u: ");nhap(&u); printf("\n Nhap phan so v: ");nhap(&v); tu = nhan(q,z);

tu = tru(p,tu) ; mau = cong(u,v) ; s = chia(tu,mau);

printf(“\n Phan so s = “); in(s); getch();

}

**Nhận xét:** Việc sử dụng các hàm để thực hiện các phép tính không được tự nhiên và tỏ ra dài dòng. Ví dụ để thực hiện một công thức

s = (p - q\*z)/(u + v)

phải dùng 2 biến trung gian và 4 lời gọi hàm. Câu hỏi đặt ra là có cách nào để chỉ cần viết đúng công thức toán học, mà vẫn nhận được kết quả mong muốn hay không?

Trong C++ có thể đáp ứng được mong muốn này bằng cách sử dụng các phép toán chuẩn của nó cho các kiểu dữ liệu tự định nghĩa (mảng, cấu trúc, ...). Nói cách khác C++ cho phép dùng các phép toán để định nghĩa các hàm, mà ta thường gọi là định nghĩa chồng các toán tử (hay còn gọi: Sự tải bội các toán tử).

## Cách định nghĩa chồng các toán tử

* + 1. **Tên hàm toán tử**: Gồm từ khoá operator và tên phép toán, ví dụ: operator+ (định nghĩa chồng phép +)

operator- (định nghĩa chồng phép -)

## Các đối của hàm toán tử:

1. ***Với các phép toán có 2 toán hạng****,* thì hàm toán tử cần có 2 đối. Đối thứ nhất ứng với toán hạng thứ nhất, đối thứ hai ứng với toán hạng thứ hai. Do vậy, với các phép toán không giao hoán (như phép-) thì thứ tự đối là rất quan trọng.

**Ví dụ** các hàm toán tử cộng , trừ phân số được khai báo như sau: struct PS

{

int a; // Tử số int b; // Mẫu số

} ;

|  |  |  |
| --- | --- | --- |
| PS | operator+(PS p1, PS p2); | // p1 + p2 |
| PS | operator-(PS p1, PS p2); | // p1 - p2 |
| PS | operator\*(PS p1, PS p2); | // p1 \* p2 |
| PS | operator/(PS p1, PS p2); | // p1 / p2 |

1. ***Với các phép toán có một toán hạng,*** thì hàm toán tử có một đối. Ví dụ hàm toán tử đổi dấu ma trận (đổi dấu tất cả các phần tử của ma trận) được khai báo như sau:

struct MT

{

double a[20][20] ; // Mảng chứa các phần tử ma trận int m ; // Số hàng ma trận

int n ; // Số cột ma trân

} ;

MT operator-(MT x) ;

* + 1. **Thân của hàm toán tử:** Viết như thân của hàm thông thường. Ví dụ hàm đổi dấu ma trận có thể được định nghĩa như sau:

struct MT

{

double a[20][20] ; // Mảng chứa các phần tử ma trận int m ; // Số hàng ma trận

int n ; // Số cột ma trân

} ;

MT operator-(MT x)

{

MT y;

for (int i=1; i<= m ;++i) for (int j=1; j<= n ;++j)

y[i][j] = - x[i][j] ; return y;

}

## Cách dùng hàm toán tử

Có 2 cách dùng:

*Cách 1:* Dùng như một hàm thông thường bằng cách viết lời gọi.

## Ví dụ:

PS p, q, u, v ;

u = operator+(p, q) ; // u = p + q v = operator-(p, q) ; // v = p - q

*Cách 2:* Dùng như phép toán của C++ .

## Ví dụ:

PS p, q, u, v ;

u = p + q ; // u = p + q v = p - q ; // v = p - q

***Chú ý:*** Khi dùng các hàm toán tử như phép toán của C++ ta có thể kết hợp nhiều phép toán để viết các công thức phức tạp. Cũng cho phép dùng dấu ngoặc tròn để quy định thứ tự thực hiện các phép tính. Thứ tự ưu tiên của các phép tính vẫn tuân theo các quy tắc ban đầu của C++ . Chẳng hạn các phép \* và / có thứ ưu tiên cao hơn so với các phép + và -

## Ví dụ:

PS p, q, u, v, s1, s2 ;

s1 = p\*q - u/v ; // s1 = (p\*q)

s2 = (p - q)/(u + v) ; // s2 = (p - q)/(u + v)

## § 8. Các ví dụ về định nghĩa chồng toán tử

**Ví dụ 1:** Trong ví dụ này ngoài việc sử dụng các hàm toán tử để thực hiện 4 phép tính trên phân số, còn định nghĩa chồng các phép toán << và >> để xuất và nhập phân số (xem chi tiết trong chương 7).

Hàm operator<< có 2 đối kiểu ostream& và PS (Phân số). Hàm trả về giá trị kiểu ostream&.

Hàm được khai báo như sau:

ostream& operator<< (ostream& os, PS p); Tượng tự hàm operator>> được khai báo như sau:

istream& operator>> (istream& is,PS &p);

Dưới đây sẽ chỉ ra cách xây dựng và sử dụng các hàm toán tử. Chúng ta cũng sẽ thấy việc sử dụng các hàm toán tử rất tự nhiên, ngắn gọn và tiện lợi.

Chương trình dưới đây có nội dung như chương trình trong §6.2, nhưng thay các hàm bằng các hàm toán tử.

#include <conio.h> #include <iostream.h> #include <math.h> typedef struct

{

int a,b;

} PS;

ostream& operator<< (ostream& os, PS p); istream& operator>> (istream& is,PS &p); int uscln(int x, int y);

PS rutgon(PS p);

PS operator+(PS p1, PS p2); PS operator-(PS p1, PS p2); PS operator\*(PS p1, PS p2); PS operator/(PS p1, PS p2);

ostream& operator<< (ostream& os, PS p)

{

os << p.a << '/' << p.b ; return os;

}

istream& operator>> (istream& is,PS &p)

{

cout << "Nhap tu va mau: " ; is >> p.a >> p.b ;

return is;

}

int uscln(int x, int y)

{

x=abs(x); y=abs(y); if (x\*y==0) return 1; while (x!=y)

if (x>y) x-=y; else y-=x;

return x;

}

PS rutgon(PS p)

{

PS q;

int x; x=uscln(p.a,p.b);

q.a = p.a / x ;

q.b = p.b / x ; return q;

}

PS operator+(PS p1, PS p2)

{

PS q;

q.a = p1.a\*p2.b + p2.a\*p1.b;

q.b = p1.b \* p2.b ; return rutgon(q);

}

PS operator-(PS p1, PS p2)

{

PS q;

q.a = p1.a\*p2.b - p2.a\*p1.b;

q.b = p1.b \* p2.b ; return rutgon(q);

}

PS operator\*(PS p1, PS p2)

{

PS q;

q.a = p1.a \* p2.a ;

q.b = p1.b \* p2.b ; return rutgon(q);

}

PS operator/(PS p1, PS p2)

{

PS q;

q.a = p1.a \* p2.b ;

q.b = p1.b \* p2.a ; return rutgon(q);

}

void main()

{

PS p, q, z, u, v ; PS s;

cout <<"\nNhap cac PS p, q, z, u, v:\n " ; cin >> p >> q >> z >> u >> v ;

s = (p - q\*z) / (u + v) ;

cout << "\n Phan so s = " << s; getch();

}

**Ví dụ 2:** Chương trình đưa vào các hàm toán tử: operator- có một đối dùng để đảo dấu một đa thức operator+ có 2 đối dùng để cộng 2 đa thức operator- có 2 đối dùng để trừ 2 đa thức

operator\* có 2 đối dùng để nhân 2 đa thức operator^ có 2 đối dùng để tính giá đa thức tại x operator<< có 2 đối dùng để in đa thức operator>> có 2 đối dùng để nhập đa thức

Chương trình sẽ nhập 4 đa thức: p, q, r, s. Sau đó tính đa thức: f = -(p+q)\*(r-s)

Cuối cùng tính giá trị f(x), với x là một số thực nhập từ bàn phím. #include <conio.h>

#include <iostream.h> #include <math.h> struct DT

{

double a[20]; // Mang chua cac he so da thuc a0, a1,... int n ; // Bac da thuc

} ;

ostream& operator<< (ostream& os, DT d); istream& operator>> (istream& is,DT &d); DT operator-(const DT& d);

DT operator+(DT d1, DT d2); DT operator-(DT d1, DT d2); DT operator\*(DT d1, DT d2);

double operator^(DT d, double x); // Tinh gia tri da thuc

ostream& operator<< (ostream& os, DT d)

{

os << " - Cac he so (tu ao): " ; for (int i=0 ; i<= d.n ; ++i)

os << d.a[i] <<" " ; return os;

}

istream& operator>> (istream& is, DT &d)

{

cout << " - Bac da thuc: " ; cin >> d.n;

cout << "Nhap cac he so da thuc:\n" ; for (int i=0 ; i<= d.n ; ++i)

{

cout << "He so bac " << i << " = " ; is >> d.a[i] ;

}

return is;

}

DT operator-(const DT& d)

{

DT p;

p.n = d.n;

for (int i=0 ; i<=d.n ; ++i) p.a[i] = -d.a[i];

return p;

}

DT operator+(DT d1, DT d2)

{

DT d;

int k,i;

k = d1.n > d2.n ? d1.n : d2.n ; for (i=0; i<=k ; ++i)

if (i<=d1.n && i<=d2.n) d.a[i] = d1.a[i] + d2.a[i];

else if (i<=d1.n) d.a[i] = d1.a[i];

else

d.a[i] = d2.a[i]; i=k;

while (i>0 && d.a[i]==0.0) --i;

d.n = i; return d ;

}

DT operator-(DT d1, DT d2)

{

return (d1 + (-d2));

}

DT operator\*(DT d1, DT d2)

{

DT d;

int k, i, j;

k = d.n = d1.n + d2.n ;

for (i=0; i<=k; ++i) d.a[i] = 0; for (i=0 ; i<= d1.n ; ++i)

for (j=0 ; j<= d2.n ; ++j) d.a[i+j] += d1.a[i]\*d2.a[j] ;

return d;

}

double operator^(DT d, double x)

{

double s=0.0 , t=1.0;

for (int i=0 ; i<= d.n ; ++i)

{

s += d.a[i]\*t; t \*= x;

}

return s;

}

void main()

{

DT p,q,r,s,f; double x,g; clrscr();

cout <<"\nNhap da thuc P " ; cin >> p; cout <<"\nNhap da thuc Q " ; cin >> q; cout <<"\nNhap da thuc R " ; cin >> r; cout <<"\nNhap da thuc S " ; cin >> s; cout << "\nNhap so thuc x: " ; cin >> x; f = -(p+q)\*(r-s);

g = f^x;

cout << "\nDa thuc f " << f ;

cout << "\n x = " << x; cout << "\nf(x) = " << g; getch();

}

## § 9. Các bài toán về ma trận và véc tơ

Trong mục này sẽ xét các ma trận thực vuông cấp n và các véc tơ thực cấp n. Chúng được biểu diễn thông qua các kiểu cấu trúc MT và VT:

struct MT

{

double a[20][20] ; // Mang a chứa các phần tử ma trận int n ; // Cấp ma trận

} ;

struct VT

{

double b[20]; // Mang chua cac phan tu cua vec to int n ; // Cap vec to

} ;

Để xử lý ma trận và véc tơ, chúng ta xây dựng 9 hàm toán tử:

ostream& operator<< (ostream& os, const MT& x); // In ma trận ostream& operator<< (ostream& os, const VT& v); // In véc tơ istream& operator>> (istream& is,MT& x); // Nhập ma trận istream& operator>> (istream& is, VT &v); // Nhập véc tơ MT operator+(const MT& x1, const MT& x2); // Cộng 2 ma trận MT operator-(const MT& x1, const MT& x2); // Trừ 2 ma trận MT operator\*(const MT& x1, const MT& x2); // Nhân 2 ma trận VT operator\*(const MT& x, const VT& v); // Nhân ma trận véc tơ MT operator!(MT x); // Nghịch đảo ma trận

Thuật toán cho 8 hàm toán tử đầu tương đối quen thuộc không có gì phải bàn. Để nghịch đảo ma trận có nhiều cách, ở đây chúng ta dùng phương pháp Jordance như sau. Giả sử cần nghịch đảo ma trận x cấp n. Ta dùng thêm ma trận đơn vị y. Sau đó thực hiện đồng thời các phép tính trên cả x và y sao cho x trở thành đơn vị. Kết quả y chính là nghịch đảo của x. Thuật toán được tiến hành trên n bước. Nội dung của bước k (k = 1,...,n) như sau:

Tìm chỉ số r ( k <= r <= n) sao cho

abs(x[r,k]) = max { abs(x[i,k] với i = k,...,n }

Nếu abs(x[r,k]) = 0 thì ma trận không có nghịch đảo và thuật toán kết thúc giữa chừng. Hoán vị hàng k với hàng r trong cả 2 ma trận x và y.

Chia hàng k của cả x và y cho tg = x[k,k] (mục đích làm cho x[k,k] = 1).

Biến đổi để cột k của x trơ thành véc tơ đơn vị bằng cách làm cho các phần tử x[i,k] = 0 (với i khác k). Muốn vậy ta thực hiện các phép tính sau trên cả x và y:

(hàng i) = (hàng i) - x[i,k]\*(hàng k) , với mọi i khác k

Nội dung chương trình là nhập 4 ma trận X, Y, R, S và véc tơ u. Sau đó tính véc tơ v theo công thức:

-1

v = ((X + Y)\*(R - S)) u

Như sẽ thấy trong hàm main() dưới đây, nhờ các hàm toán tử mà câu lệnh tính v được viết gần giống như công thức toán học nêu trên.

/\* Chương trình \*/ #include <conio.h> #include <iostream.h> #include <iomanip.h> #include <math.h> struct MT

{

double a[20][20]; // Mang chua cac phan tu ma tran int n ; // Cap ma tran

} ;

struct VT

{

double b[20]; // Mang chua cac phan tu cua vec to int n ; // Cap vec to

} ;

ostream& operator<< (ostream& os, const MT& x); ostream& operator<< (ostream& os, const VT& v); istream& operator>> (istream& is,MT& x); istream& operator>> (istream& is, VT &v);

MT operator+(const MT& x1, const MT& x2); MT operator-(const MT& x1, const MT& x2); MT operator\*(const MT& x1, const MT& x2); VT operator\*(const MT& x, const VT& v);

MT operator!(MT x); // Tinh ma tran nghich dao ostream& operator<< (ostream& os, const MT& x)

{

os << setprecision(2) << setiosflags(ios::showpoint); for (int i=1 ; i<= x.n ; ++i)

{

os << "\n" ;

for (int j=1; j<=x.n; ++j)

os << setw(6) << x.a[i][j] ;

}

os << "\n" ; return os;

}

ostream& operator<< (ostream& os, const VT& v)

{

os << setprecision(2) << setiosflags(ios::showpoint); for (int i=1 ; i<= v.n ; ++i)

os << setw(6) << v.b[i] ; os << "\n" ;

return os;

}

istream& operator>> (istream& is, MT& x)

{

cout << " - Cap ma tran: " ; is >> x.n;

cout << "Nhap cac phan tu :\n" ; for (int i=1 ; i<= x.n ; ++i)

for (int j=1; j<=x.n; ++j)

{

cout << "PT hang " << i << " cot " << j << " = " ; is >> x.a[i][j] ;

}

return is;

}

istream& operator>> (istream& is, VT& v)

{

cout << " - Cap vec to: " ; is >> v.n;

cout << "Nhap cac phan tu :\n" ; for (int i=1 ; i<= v.n ; ++i)

{

cout << "Phan tu thu " << i << " = " ; is >> v.b[i] ;

}

return is;

}

MT operator+(const MT& x1, const MT& x2)

{

if (x1.n!=x2.n)

{

cout << "\nKhong thuc hien duoc phep cong vi 2 MT khong cung cap"; getch();

return x1;

}

else

{

MT x;

int i, j, n;

n = x.n = x1.n ;

for (i=1; i<=n; ++i) for (j=1; j<=n ;++j)

x.a[i][j] = x1.a[i][j] + x2.a[i][j] ; return x;

}

}

MT operator-(const MT& x1, const MT& x2)

{

if (x1.n!=x2.n)

{

cout << "\nKhong thuc hien duoc phep tru vi 2 MT khong cung cap"; getch();

return x1;

}

else

{

MT x;

int i, j, n;

n = x.n = x1.n;

for (i=1; i<=n; ++i) for (j=1; j<=n ;++j)

x.a[i][j] = x1.a[i][j] - x2.a[i][j] ; return x;

}

}

MT operator\*(const MT& x1, const MT& x2)

{

if (x1.n!=x2.n)

{

cout << "\nKhong thuc hien duoc phep nhan vi 2 MT khong cung cap"; getch();

return x1;

}

else

{

MT x;

int n, i, j,k;

n = x.n = x1.n;

for (i=1; i<=n; ++i) for (j=1; j<=n ;++j)

{

x.a[i][j] = 0.0 ;

for (k=1 ; k<=n; ++k)

x.a[i][j] += x1.a[i][k]\*x2.a[k][j] ;

}

return x;

}

}

VT operator\*(const MT& x, const VT& v)

{

if (x.n != v.n)

{

cout << "\n Cap ma tran khac cap vec to, phep nhan vo nghia"; getch();

return v;

}

else

{

VT u; int n; n = u.n = v.n ;

for (int i=1; i <=n ; ++i)

{

u.b[i] = 0;

for (int j=1; j<=n; ++j) u.b[i] += x.a[i][j]\*v.b[j];

}

return u;

}

}

MT operator!(MT x)

{

MT y;

int i,j,k,r,n; double tg;

n = y.n = x.n ;

for (i=1 ; i<=n ; ++i)

for (j=1 ; j<=n ; ++j) if (i==j) y.a[i][j] = 1;

else y.a[i][j] = 0; for (k=1; k<=n; ++k)

{

r=k;

for (i=k+1; i<=n; ++i)

if (abs(x.a[i][k]) > abs(x.a[r][k]) ) r = i; if (abs(x.a[r][k]) < 1.0E-8)

{

cout << "\n Ma tran suy bien, khong co nghich dao" ; getch();

return x;

}

/\* Hoan vi hang r va hang k \*/ for (j=1 ; j<=n ; ++j)

{

tg = x.a[k][j]; x.a[k][j] = x.a[r][j]; x.a[r][j] = tg;

tg = y.a[k][j]; y.a[k][j] = y.a[r][j]; y.a[r][j] = tg;

}

/\* Chia hang k cho a[k,k] \*/ tg = x.a[k][k] ;

for (j=1 ; j<=n ; ++j)

{

x.a[k][j] /= tg;

y.a[k][j] /= tg;

}

/\* Khu cot k : lam cho a[i,k] = 0 voi i != k \*/ for (int i=1; i<= n ; ++i)

if (i != k)

{

tg = x.a[i][k] ;

for (j=1 ; j<=n ; ++j)

{

x.a[i][j] -= tg\*x.a[k][j] ;

y.a[i][j] -= tg\*y.a[k][j] ;

}

}

}

return y;

}

void main()

{

MT x,y,r,s; VT u,v; clrscr();

cout <<"\nNhap ma tran X " ; cin >> x; cout <<"\nNhap ma tran Y " ; cin >> y; cout <<"\nNhap ma tran R " ; cin >> r; cout <<"\nNhap ma tran S " ; cin >> s; cout <<"\nNhap vec to u " ; cin >> u;

v = !((x+y)\*(r-s))\*u ;

cout << "\nVec to v = xu " << v ; getch();

}

**Chương 3**

# Khái niệm về lớp

Như đã nói ở trên, lớp là khái niệm trung tâm của lập trình hướng đối tượng, nó là sự mở rộng của các khái niệm cấu trúc (struct) của C và bản ghi (record) của PASCAL. Ngoài các thành phần dữ liệu (như cấu trúc), lớp còn chứa các thành phần hàm , còn gọi là phương thức (method) hay hàm thành viên (member function). Cũng giống như cấu trúc, lớp có thể xem như một kiểu dữ liệu. Vì vậy lớp còn gọi là kiểu đối tượng và lớp được dùng để khai báo các biến, mảng đối tượng (như thể dùng kiểu int để khai báo các biến mảng nguyên). Như vậy từ một lớp có thể tạo ra (bằng cách khai báo) nhiều đối tượng (biến, mảng) khác nhau. Mỗi đối tượng có vùng nhớ riêng của mình. Vì vậy cũng có thể quan niệm lớp là tập hợp các đối tượng cùng kiểu.

Chương này sẽ trình bầy cách định nghĩa lớp, cách xây dựng phương thức, giải thích về phạm vi truy nhập, sư dụng các thành phần của lớp, cách khai báo biến, mảng cấu trúc, lời gọi tới các phương thức.

## Bài 1. Định nghĩa lớp

1. Lớp được định nghĩa theo mẫu:

## class tên\_lớp

**{**

## // Khai báo các thành phần dữ liệu (thuộc tính)

**// Khai báo các phương thức**

## } ;

**// Định nghĩa (xây dựng) các phương thức**

### Chú ý:

Thuộc tính của lớp có thể là các biến, mảng, con trỏ có kiểu chuẩn (int, float, char, char\*, long,...) hoặc kiểu ngoài chuẩn đã định nghĩa trước (cấu trúc, hợp, lớp, ...) . Thuộc tính của lớp không thể có kiểu của chính lớp đó, nhưng có thể là kiểu con trỏ lớp này, ví dụ:

class A

{

A x ; // Không cho phép, vì x có kiểu lớp A A \*p ; // Cho phép , vì p là con trỏ kiểu lớp A

...

} ;

1. Khi báo các thành phần của lớp (thuộc tính và phương thức) có thể dùng các từ khoá private và public để quy định phạm vi sử dụng của các thành phần. Nếu không quy định cụ thể (không dùng các từ khoá private và public) thì C++ hiểu đó là private.

Các thành phần private (riêng) chỉ được sử dụng bên trong lớp (trong thân của các phương thức của lớp). Các hàm không phải là phương thức của lớp không được phép sử dụng các thành phần này.

Các thành phần public (công cộng) được phép sử dụng ở cả bên trong và bên ngoài lớp.

1. Các thành phần dữ liệu thường (nhưng không bắt buộc) khai báo là private để bảo đảm tính giấu kín, bảo vệ an toàn dữ liệu của lớp, không cho phép các hàm bên ngoài xâm nhập vào dữ liệu của lớp.
2. Các phương thức thường khai báo là public để chúng có thể được gọi tới (sử dụng) từ các hàm khác trong chương trình.
3. Các phương thức có thể được xây dựng bên ngoài hoặc bên trong định nghĩa lớp. Thông thường, các phương thức ngắn được viết bên trong định nghĩa lớp, còn các phương thức dài thì viết bên ngoài định nghĩa lớp.
4. Trong thân phương thức của một lớp (giả sử lớp A) có thể sử dụng:

+ Các thuộc tính của lớp A

+ Các phương thức của lớp A

+ Các hàm tự lập trong chương trình. Vì phạm vi sử dụng của hàm là toàn chương trình.

1. Giá trị trả về của phương thức có thể có kiểu bất kỳ (chuẩn và ngoài chuẩn)

**Ví dụ** sau sẽ minh hoạ các điều nói trên. Chúng ta sẽ định nghĩa lớp để mô tả và xử lý các điểm trên màn hình đồ hoạ. Lớp được đăt tên là DIEM.

+ Các thuộc tính của lớp gồm: int x ; // hoành độ (cột) int y ; // tung độ (hàng) int m ; // mầu

+ Các phương thức:

Nhập dữ liệu một điểm Hiển thị một điểm

ẩn một điểm

Lớp điểm được xây dựng như sau:

*class DIEM*

*{*

*private:*

*int x, y, m ; public:*

*void nhapsl() ; void hien() ; void an()*

*{*

*putpixel(x, y, getbkcolor());*

*}*

*} ;*

*void DIEM::nhap()*

*{*

*cout << “\nNhập hoành độ (cột) và tung độ (hàng) của điểm: “ cin >> x >> y ;*

*cout << “\nNhập mã mầu của điểm: “ cin >> m ;*

*}*

*void DIEM::hien()*

*{*

*int mau\_ht ;*

*mau\_ht = getcolor(); putpixel(x, y, m); setcolor(mau\_ht);*

*}*

Qua ví dụ trên có thể rút ra một số điều cần nhớ sau:

+ Trong cả 3 phương thức (dù viết trong hay viết ngoài định nghĩa lớp) đều được phép truy nhập đến các thuộc tính x, y và m của lớp.

+ Các phương thức viết bên trong định nghĩa lớp (như phương thức an() ) được viết như một hàm thông thường.

+ Khi xây dựng các phương thức bên ngoài lớp, cần dùng thêm tên lớp và toán tử phạm vi :: đặt ngay trước tên phương phức để quy định rõ đây là phương thức của lớp nào.

## Bài 2. Biến, mảng đối tượng

Như đã nói ở trên, một lớp (sau khi định nghĩa) có thể xem như một kiểu đối tượng và có thể dùng để khai báo các biến, mảng đối tượng. Cách khai báo biến, mảng đối tượng cũng giống như khai báo biến, mảng các kiểu khác (như int, float, cấu trúc, hợp, ...), theo mẫu sau:

Tên\_lớp danh sách đối ; Tên\_lớp danh sách mảng ;

**Ví dụ** sử dụng lớp DIEM ở §1, có thể khai báo các biến, mảng DIEM như sau: DIEM d1, d2, d3 ; // Khai báo 3 biến đối tượng d1, d2, d3

DIEM d[20] ; // Khai báo mảng đối tượng d gồm 20 phần tử

Mỗi đối tượng sau khi khai báo sẽ được cấp phát một vùng nhớ riêng để chứa các thuộc tính của chúng. Chú ý rằng sẽ không có vùng nhớ riêng để chứa các phương thức cho mỗi đối tượng. Các phương thức sẽ được sử dụng chung cho tất cả các đối tượng cùng lớp. Như vậy về bộ nhớ được cấp phát thì đối tượng giống cấu trúc. Trong trương hợp này:

sizeof(d1) = sizeof(d2) = sizeof(d3) = 3\*sizeof(int) = 6 sizeof(d) = 20\*6 = 120

## Thuộc tính của đối tượng:

Trong ví dụ trên, mỗi đối tượng d1, d2, d3 và mỗi phần tử d[i] đều có 3 thuộc tính là x, y, m. Chú ý là mỗi thuộc đều thuộc về một đối tượng, vì vậy không thể viết tên thuộc một cách riêng rẽ mà bao giờ cũng phải có tên đối tượng đi kèm, giống như cách viết trong cấu trúc của C hay bản ghi của PASCAL. Nói cách khác, cách viết thuộc tính của đối tượng như sau:

tên\_đối\_tượng.Tên\_thuộc\_tính

Với các đối tượng d1, d2, d3 và mảng d, có thể viết như sau: d1.x // Thuộc tính x của đối tượng d1

d2.x // Thuộc tính x của đối tượng d2 d3.y // Thuộc tính y của đối tượng d3 d[2].m // Thuộc tính m của phần tử d[2] d1.x = 100 ; // Gán 100 cho d1.x

d2.y = d1.x; // Gán d1.x cho d2.y

## Sử dụng các phương thức

Cũng giống như hàm, một phương thức được sử dụng thông qua lời gọi. Tuy nhiên trong lời gọi phương thức bao giờ cũng phải có tên đối tượng để chỉ rõ phương thức thực hiện trên các thuộc tính của đối tượng nào. Ví dụ lời gọi:

d1.nhapsl();

sẽ thực hiện nhập số liệu vào các thành phần d1.x, d1.y và d1.m Câu lệnh

d[3].nhapsl() ;

sẽ thực hiện nhập số liệu vào các thành phần d[3].x, d[3].y và d[3].m

Chúng ta sẽ minh hoạ các điều nói trên bằng một chương trình đơn giản sử dụng lớp DIEM để nhập 3 điểm, hiện rồi ẩn các điểm vừa nhập. Trong chương trình đưa vào hàm kd\_do\_hoa() dùng để khởi động hệ đồ hoạ.

*#include <conio.h> #include <iostream.h> #include <graphics.h> class DIEM*

*{*

*private:*

*int x, y, m ; public:*

*void nhapsl(); void an()*

*{*

*putpixel(x,y,getbkcolor());*

*}*

*void hien();*

*};*

*void DIEM::nhapsl()*

*{*

*cout << "\nNhap hoanh do (cot) va tung do (hang) cua diem: " ; cin >> x >> y ;*

*cout << " \nNhap ma mau cua diem: " ; cin >> m ;*

*}*

*void DIEM::hien()*

*{*

*int mau\_ht;*

*mau\_ht = getcolor() ; putpixel(x,y,m); setcolor(mau\_ht);*

*}*

*void kd\_do\_hoa()*

*{*

*int mh, mode ; mh=mode=0; initgraph(&mh, &mode, "");*

*}*

*void main()*

*{*

*DIEM d1, d2, d3 ;*

*d1.nhapsl();*

*d2.nhapsl();*

*d3.nhapsl(); kd\_do\_hoa(); setbkcolor(BLACK); d1.hien();*

*d2.hien();*

*d3.hien();*

*getch();*

*d1.an();*

*d2.an();*

*d3.an();*

*getch(); closegraph();*

*}*

## Bài 3. Con trỏ đối tượng

Con trỏ đối tượng dùng để chứa địa chỉ của biến, mảng đối tượng. Nó được khai báo như sau: Tên\_lớp \*con trỏ ;

**Ví dụ** dùng lớp DIEM có thể khai báo:

DIEM \*p1 , \*p2, \*p3 ; // khai báo 3 con trỏ p1, p2, p3 DIEM d1, d2 ; // Khai báo 2 đối tượng d1, d2

DIEM d[20] ; // Khai báo mảng đối tượng và có thể thực hiện các câu lệnh:

p1 = &d2 ; // p1 chứa địa chỉ của d2 , hay p1 trỏ tới d2 p2 = d ; // p2 trỏ tới đầu mảng d

p3 = new DIEM // Tạo một đối tượng và chứa địa chỉ của nó

// vào p3

Để sử dụng thuộc tính của đối tượng thông qua con trỏ, ta viết như sau: Tên\_con\_trỏ->Tên\_thuộc\_tính

***Chú ý:*** Nếu con trỏ chứa địa chỉ đầu của mảng, có thể dùng con trỏ như tên mảng.

Như vậy sau khi thực hiện các câu lệnh trên thì: p1->x và d2.x là như nhau

p2[i].y và d[i].y là như nhau

## Tóm lại ta có quy tắc sau

**Quy tắc sử dụng thuộc tính:** Để sử dụng một thuộc tính của đối tượng ta phải dùng phép . hoặc phép -> . Trong chương trình, không cho phép viết tên thuộc tính một cách đơn độc mà phải đi kèm tên đối tượng hoặc tên con trỏ theo các mẫu sau:

Tên\_đối\_tượng.Tên\_thuộc\_tính Tên\_con\_trỏ->Tên\_thuộc\_tính

Tên\_mảng\_đối\_tượng[chỉ\_số].Tên\_thuộc\_tính Tên\_con\_trỏ[chỉ\_số].Tên\_thuộc\_tính

Chương trình dưới đây cũng sử dụng lớp DIEM (trong §1) để nhập một dẫy điểm, hiển thị và ẩn các điểm vừa nhập. Chương trình dùng một con trỏ kiểu DIEM và dùng toán tử new để tạo ra một dẫy đối tượng.

*#include <conio.h> #include <iostream.h> #include <graphics.h> class DIEM*

*{*

*private:*

*int x, y, m ; public:*

*void nhapsl(); void an()*

*{*

*putpixel(x,y,getbkcolor());*

*}*

*void hien();*

*};*

*void DIEM::nhapsl()*

*{*

*cout <<"\nNhap hoanh do (cot) va tung do (hang) cua diem:" ; cin >> x >> y ;*

*cout << " \nNhap ma mau cua diem: " ; cin >> m ;*

*}*

*void DIEM::hien()*

*{*

*int mau\_ht;*

*mau\_ht = getcolor() ; putpixel(x,y,m);*

*setcolor(mau\_ht);*

*}*

*void kd\_do\_hoa()*

*{*

*int mh, mode ; mh=mode=0; initgraph(&mh, &mode, "");*

*}*

*void main()*

*{*

*DIEM \*p;*

*int i, n;*

*cout << "So diem: " ; cin >> n;*

*p = new DIEM[n+1]; for (i=1; i<=n; ++i)*

*p[i].nhapsl(); kd\_do\_hoa();*

*for (i=1; i<=n; ++i) p[i].hien();*

*getch();*

*for (i=1; i<=n; ++i) p[i].an();*

*getch(); closegraph();*

*}*

## Bài 4. Đối của phương thức, con trỏ this

* 1. **Con trỏ this là đối thứ nhất của phương thức**

Chúng ta hãy xem lại phương thức nhapsl của lớp DIEM

*void DIEM::nhapsl()*

*{*

*cout <<"\nNhap hoanh do (cot) va tung do (hang) cua diem:" ; cin >> x >> y ;*

*cout << " \nNhap ma mau cua diem: " ; cin >> m ;*

*}*

Rõ ràng trong phương thức này chúng ta sử dụng tên các thuộc tính x, y và m một cách đơn độc. Điều này có vẻ như mâu thuẫn với quy tắc sử dụng thuộc tính nêu trong mục trước. Song sự thể như sau:

C++ sử dụng con trỏ đặc biệt **this** trong các phương thức. Các thuộc tính viết trong phương thức được hiểu là thuộc một đối tượng do con trỏ this trỏ tới. Như vậy phương thức nhapsl() có thể viết một cách tường minh như sau:

*void DIEM::nhapsl()*

*{*

*cout << "\nNhap hoanh do (cot) va tung do (hang) cua diem:" ; cin >> this->x >> this->y ;*

*cout << " \nNhap ma mau cua diem: " ; cin >> this->m ;*

*}*

Từ góc độ hàm số có thể kết luận rằng: Phương thức bao giờ cũng có ít nhất một đối là con trỏ this và nó luôn luôn là đối đầu tiên của phương thức.

## Tham số ứng với đối con trỏ this

Xét một lời gọi tới phương thức nhapsl() : DIEM d1;

d1.nhapsl() ;

Trong trường hợp này tham số truyền cho con trỏ this chính là địa chỉ của d1: this = &d1

Do đó:

this->x chính là d1.x this->y chính là d1.y this->m chính là d1.m

Như vậy câu lệnh d1.nhapsl() ;

sẽ nhập dữ liệu cho các thuộc tính của đối tượng d1. Từ đó có thể rút ra kết luận sau:

Tham số truyền cho đối con trỏ this chính là địa chỉ của đối tượng đi kèm với phương thức trong lời gọi phương thức.

## Các đối khác của phương thức

Ngoài đối đặc biệt this (đối này không xuất hiện một cách tường minh), phương thức còn có các đối khác được khai báo như trong các hàm. Đối của phương thức có thể có kiểu bất kỳ (chuẩn và ngoài chuẩn).

**Ví dụ** để xây dựng phương thức vẽ đường thẳng qua 2 điểm ta cần đưa vào 3 đối: Hai đối là 2 biến kiểu DIEM, đối thứ ba kiểu nguyên xác định mã mầu. Vì đã có đối ngầm định this là đối thứ nhất, nên chỉ cần khai báo thêm 2 đối. Phương thức có thể viết như sau:

*void DIEM::doan\_thang(DIEM d2, int mau)*

*{*

*int mau\_ht;*

*mau\_ht = getcolor(); setcolor(mau);*

*line(this->x,this->y,d2.x,d2.y);*

*setcolor(mau\_ht);*

*}*

Chương trình sau minh hoạ các phương thức có nhiều đối. Ta vẫn dùng lớp DIEM nhưng có một số thay đổi:

+ Bỏ thuộc tính m (mầu)

+ Bỏ các phương thức hien và an

+Đưa vào 4 phương thức mới:

ve\_ doan\_thang (Vẽ đoạn thẳng qua 2 điểm) ve\_tam\_giac (Vẽ tam giác qua 3 điểm)

do\_dai (Tính độ dài của đoạn thẳng qua 2 điểm) chu\_vi (Tính chu vi tam giác qua 3 điểm)

Chương trình còn minh hoạ:

+ Việc phương thức này sử dụng phương thức khác (phương thức ve\_tam\_giac sử dụng phương thức ve\_doan\_thang, phương thức chu\_vi sử dụng phương thức do\_dai)

+ Sử dụng con trỏ this trong thân các phương thức ve\_tam\_giac và chu\_vi

Nội dung chương trình là nhập 3 điểm, vẽ tam giác có đỉnh là 3 điểm vừa nhập sau đó tính chu vi tam giác.

*#include <conio.h> #include <iostream.h> #include <graphics.h> #include <math.h> #include <stdio.h> class DIEM*

*{*

*private:*

*int x, y ; public:*

*void nhapsl();*

*void ve\_doan\_thang(DIEM d2, int mau) ;*

*void ve\_tam\_giac(DIEM d2, DIEM d3,int mau) ; double do\_dai(DIEM d2)*

*{*

*DIEM d1 = \*this ;*

*return sqrt( pow(d1.x - d2.x,2) + pow(d1.y - d2.y,2) ) ;*

*}*

*double chu\_vi(DIEM d2, DIEM d3);*

*};*

*void DIEM::nhapsl()*

*{*

*cout <<" \nNhap hoanh do (cot) va tung do (hang) cua diem:" ; cin >> x >> y ;*

*}*

*void kd\_do\_hoa()*

*{*

*int mh, mode ; mh=mode=0; initgraph(&mh, &mode, "");*

*}*

*void DIEM::ve\_doan\_thang(DIEM d2, int mau)*

*{*

*setcolor(mau);*

*line(this->x,this->y,d2.x,d2.y);*

*}*

*void DIEM::ve\_tam\_giac(DIEM d2, DIEM d3,int mau)*

*{*

*(\*this).ve\_doan\_thang(d2,mau); d2.ve\_doan\_thang(d3,mau); d3.ve\_doan\_thang(\*this,mau);*

*}*

*double DIEM::chu\_vi(DIEM d2, DIEM d3)*

*{*

*double s;*

*s= (\*this).do\_dai(d2) + d2.do\_dai(d3) + d3.do\_dai(\*this) ; return s;*

*}*

*void main()*

*{*

*DIEM d1, d2, d3;*

*char tb\_cv[20] ; d1.nhapsl();*

*d2.nhapsl();*

*d3.nhapsl(); kd\_do\_hoa();*

*d1.ve\_tam\_giac(d2,d3,15); double s = d1.chu\_vi(d2,d3);*

*sprintf(tb\_cv,"Chu vi = %0.2f", s); outtextxy(10,10,tb\_cv);*

*getch(); closegraph();*

*}*

## Một số nhận xét về đối của phương thức và lời gọi phương thức

*+* Quan sát nguyên mẫu phương thức:

void ve\_doan\_thang(DIEM d2, int mau) ; sẽ thấy phương thức có 3 đối:

Đối thứ nhât là một đối tượng DIEM do this trỏ tới Đối thứ hai là đối tượng DIEM d2

Đối thứ ba là biến nguyên mau

Nội dung phương thức là vẽ một đoạn thẳng đi qua các điểm \*this và d2 theo mã mầu mau. Xem thân của phương sẽ thấy được nội dung này:

*void DIEM::ve\_doan\_thang(DIEM d2, int mau)*

*{*

*setcolor(mau);*

*line(this->x,this->y,d2.x,d2.y);*

*}*

Tuy nhiên trong trương hợp này, vai trò của this không cao lắm, vì nó được đưa vào chỉ cốt làm rõ đối thứ nhất. Trong thân phương thức có thể bỏ từ khoá this vẫn được.

+ Vai trò của this trở nên quan trọng trong phương thức ve\_tam\_giac: void ve\_tam\_giac(DIEM d2, DIEM d3,int mau) ;

Phương thức này có 4 đối là:

this trỏ tới một đối tượng kiểu DIEM d2 một đối tượng kiểu DIEM

d3 một đối tượng kiểu DIEM mau một biến nguyên

Nội dung phương thức là vẽ 3 cạnh: cạnh 1 đi qua \*this và d2 cạnh 2 đi qua d2 và d3

cạnh 3 đi qua d3 và \*this

Các cạnh trên được vẽ nhờ sử dụng phương thức ve\_doan\_thang: Vẽ cạnh 1 dùng lệnh: (\*this).ve\_doan\_thang(d2,mau) ;

Vẽ cạnh 2 dùng lệnh: d2.ve\_doan\_thang(d3,mau); Vẽ cạnh 3 dùng lệnh: d3.ve\_doan\_thang(\*this,mau);

Trong trường này rõ ràng vai trò của this rất quan trọng. Nếu không dùng nó thì công việc trơ nên khó khăn, dài dòng và khó hiểu hơn. Chúng ta hãy so sánh 2 phương án:

Phương án dùng this trong phương thức ve\_tam\_giac:

*void DIEM::ve\_tam\_giac(DIEM d2, DIEM d3,int mau)*

*{*

*(\*this).ve\_doan\_thang(d2,mau); d2.ve\_doan\_thang(d3,mau); d3.ve\_doan\_thang(\*this,mau);*

*}*

Phương án không dùng this trong phương thức ve\_tam\_giac:

*void DIEM::ve\_tam\_giac(DIEM d2, DIEM d3,int mau)*

*{*

*DIEM d1;*

*d1.x = x; d1.y = y;*

*d1.ve\_doan\_thang(d2,mau); d2.ve\_doan\_thang(d3,mau); d3.ve\_doan\_thang(d1,mau);*

*}*

## Bài 5. Nói thêm về kiểu phương thức và kiểu đối của phương thức

* 1. **Kiểu phương thức**

Phương thức có thể không có giá trị trả về (kiểu void) hoặc có thể trả về một giá trị có kiểu bất kỳ, kể cả giá trị kiểu đối tượng, con trỏ đối tượng, tham chiếu đối tượng.

## Đối của phương thức

Đối của phương thức (cũng giống như đối của hàm) có thể có kiểu bất kỳ:

+ Kiểu dữ liệu chuẩn như int, float, char,... . Con trỏ hoặc tham chiếu đến kiểu dữ liệu chuẩn như int\*, float\*, char\*, int&, float&, char&,...

+ Các kiểu ngoài chuẩn đã định nghĩa trước như đối tượng, cấu trúc, hợp, enum,... . Con trỏ hoặc tham chiếu đến các kiểu ngoài chuẩn này.

+ Kiểu đối tượng của chính phương thức, con trỏ hoặc tham chiếu đến kiểu đối tượng này.

## Các ví dụ

**Ví dụ 1 minh hoạ:**

+ Thuộc tính (thành phần dữ liệu) của lớp có thể là đối tượng của lớp khác đã định nghĩa bên

trên.

+ Phương thức có giá trị trả về kiểu đối tượng và con trỏ đối tượng.

Nội dung chương trình là nhập một dẫy hình chữ nhật, sau đó tìm hình chữ nhật có max diện tích và hình chữ nhật có max chu vi.

Chương trình được tổ chức thành 2 lớp:

+ Lớp HINH\_CN gồm:

* Các thuộc tính: d và r (chiều dài và chiều rộng)
* Các phương thức

void nhapsl() ; // Nhập chiều dài, rộng int dien\_tich(); // Tính diện tích

int chu\_vi() ; // Tính chu vi

+ Lớp DAY\_HINH\_CN gồm

* Các thuộc tính:

int n ; //số hình chữ nhật của dẫy

HINH\_CN \*h; //Con trỏ tới dẫy đối tượng của lớp HINH\_CN

* Các phương thức

void nhapsl(); // Nhập một dẫy hình chữ nhật HINH\_CN hinh\_dt\_max() ; //Trả về hình chữ nhật có

// diện tích max

HINH\_CN \*hinh\_cv\_max() ; // Trả về con trỏ tới HCN có

// chu vi max

*#include <conio.h> #include <iostream.h> class HINH\_CN*

*{*

*private:*

*int d, r; // chieu dai va chieu rong public:*

*void nhapsl()*

*{*

*cout << " \nNhap chieu dai va chieu rong: " ; cin >> d >> r ;*

*}*

*void in()*

*{*

*cout << "\nchieu dai = " << d ; cout << " chieu rong= " << r;*

*}*

*int dien\_tich()*

*{*

*return d\*r;*

*}*

*int chu\_vi()*

*{*

*return 2\*(d+r);*

*}*

*} ;*

*class DAY\_HINH\_CN*

*{*

*private:*

*int n; // So hinh ch nhat HINH\_CN \*h;*

*public:*

*void nhapsl();*

*HINH\_CN hinh\_dt\_max() ; HINH\_CN \*hinh\_cv\_max() ;*

*} ;*

*void DAY\_HINH\_CN::nhapsl()*

*{*

*cout << "So hinh CN = " ;*

*cin >> n;*

*h = new HINH\_CN[n+1]; for (int i=1;i<=n;++i)*

*h[i].nhapsl();*

*}*

*HINH\_CN DAY\_HINH\_CN::hinh\_dt\_max()*

*{*

*HINH\_CN hdtmax; hdtmax = h[1];*

*for (int i=2; i<=n; ++i)*

*if (h[i].dien\_tich() > hdtmax.dien\_tich() ) hdtmax = h[i];*

*return hdtmax;*

*}*

*HINH\_CN \*DAY\_HINH\_CN::hinh\_cv\_max()*

*{*

*int imax = 1;*

*for (int i=2; i<=n; ++i)*

*if (h[i].chu\_vi() > h[imax].chu\_vi() ) imax = i ;*

*return (h+imax);*

*}*

*void main()*

*{*

*DAY\_HINH\_CN d;*

*HINH\_CN hdtmax; d.nhapsl();*

*hdtmax = d.hinh\_dt\_max(); hdtmax.in() ;*

*HINH\_CN \*hcvmax=d.hinh\_cv\_max(); hcvmax->in() ;*

*getch();*

*}*

## Ví dụ 2 minh hoạ:

+ Thuộc tính (thành phần dữ liệu) của lớp có thể là đối tượng của lớp khác đã định nghĩa bên

trên.

+ Phương thức có giá trị trả về kiểu đối tượng

+ Vai trò của con trỏ this (xem phương thức maxdt của lớp TAM\_GIAC)

+ Phương thức tĩnh (xem phương thức tao\_tg của lớp TAM\_GIAC)

Nội dung chương trình là nhập một dẫy các điểm, sau đó tìm tam giác lớn nhất (về diện tích) có đỉnh là các điểm vừa nhập.

Chương trình được tổ chức thành 2 lớp:

+ Lớp DIEM gồm:

* Các thuộc tính: x và y (toạ độ của điểm)
* Các phương thức

void nhapsl() ; // Nhập x, y void in() ; // In toạ độ

double do\_dai(DIEM d2) ; // Tính độ dài đoạn thẳng qua

// 2 điểm (điểm ẩn xác định bởi this và điểm d2)

+ Lớp TAM\_GIAC gồm:

* Các thuộc tính:

DIEM d1,d2,d3; // 3 đỉnh của tam giác

* Các phương thức:

void nhapsl(); // Nhập toạ độ 3 đỉnh void in(); // In toạ độ 3 đỉnh

// Tạo một đối tượng TAM\_GIAC từ 3 đối tượng DIEM static TAM\_GIAC tao\_tg(DIEM e1, DIEM e2, DIEM e3) double dien\_tich() ; // Tính diện tích

// Tìm tam giác có diện tích max trong 2 tam giác \*this và t2 TAM\_GIAC maxdt(TAM\_GIAC t2);

+ Các vấn đề đáng chú ý trong chương trình là:

* Phương thưc tĩnh tao\_tg (sẽ giải thích bên dưới)
* Phương thưc maxdt

+ Thuật toán là:

* Duyệt qua các tổ hợp 3 điểm.
* Dùng phương thức tao\_tg để lập tam giác từ 3 điểm
* Dùng phương thức maxdt để chọn tam giác có diện tích lớn hơn trong 2 tam giác: tam giác vừa tạo và tam giác có diện tích max (trong số các tam giác đã tạo)

*#include <conio.h> #include <iostream.h> #include <math.h> class DIEM*

*{*

*private:*

*double x,y; // Toa do cua diem public:*

*void nhapsl()*

*{*

*cout << " Toa do x, y: " ;*

*cin >> x >> y ;*

*}*

*void in()*

*{*

*cout << " x = " << x << " y = " << y;*

*}*

*double do\_dai(DIEM d2)*

*{*

*return sqrt(pow(x-d2.x,2) + pow(y-d2.y,2) );*

*}*

*} ;*

*class TAM\_GIAC*

*{*

*private:*

*DIEM d1,d2,d3; // 3 dinh tam giac public:*

*void nhapsl(); void in();*

*static TAM\_GIAC tao\_tg(DIEM e1, DIEM e2, DIEM e3)*

*{*

*TAM\_GIAC t;*

*t.d1=e1; t.d2 = e2; t.d3=e3; return t;*

*}*

*double dien\_tich() ;*

*TAM\_GIAC maxdt(TAM\_GIAC t2);*

*} ;*

*void TAM\_GIAC::nhapsl()*

*{*

*cout << "\nDinh 1 - " ; d1.nhapsl();*

*cout << "\nDinh 2 - " ; d2.nhapsl();*

*cout << "\nDinh 3 - " ; d3.nhapsl();*

*}*

*void TAM\_GIAC::in()*

*{*

*cout << "\nDinh 1: " ; d1.in();*

*cout << "\nDinh 2: " ; d2.in();*

*cout << "\nDinh 3: " ; d3.in();*

*}*

*double TAM\_GIAC::dien\_tich()*

*{*

*double a,b,c,p,s; a=d1.do\_dai(d2); b=d2.do\_dai(d3); c=d3.do\_dai(d1); p=(a+b+c)/2;*

*return sqrt(p\*(p-a)\*(p-b)\*(p-c));*

*}*

*TAM\_GIAC TAM\_GIAC::maxdt(TAM\_GIAC t2)*

*{*

*if (this->dien\_tich() > t2.dien\_tich()) return \*this ;*

*else*

*return t2;*

*}*

*void main()*

*{*

*DIEM d[50];*

*int n, i ; clrscr();*

*cout << "\n So diem= "; cin >> n;*

*for (i=1; i<=n; ++i)*

*{*

*cout << "\nNhap diem " << i << " - " ; d[i].nhapsl();*

*}*

*int j, k ; TAM\_GIAC tmax, t;*

*tmax = TAM\_GIAC::tao\_tg(d[1],d[2],d[3]); for (i=1;i<=n-2;++i)*

*for (j=i+1;j<=n-1;++j) for (k=j+1;k<=n;++k)*

*{*

*t=TAM\_GIAC::tao\_tg(d[i],d[j],d[k]); tmax = tmax.maxdt(t);*

*}*

*cout << "\n\nTam giac co dien tich lon nhat: " ; tmax.in();*

*cout << "\nDien tich = " << tmax.dien\_tich(); getch();*

*}*

***Chú ý 1:*** Để tạo một đối tượng TAM\_GIAC từ 3 đối tượng DIEM ta đã dùng phương thức tĩnh:

*static TAM\_GIAC tao\_tg(DIEM e1, DIEM e2, DIEM e3)*

*{*

*TAM\_GIAC t;*

*t.d1=e1; t.d2 = e2; t.d3=e3; return t;*

*}*

Phương thức tĩnh (sẽ nói thêm trong các mục bên dưới) có các đặc điểm sau:

+ Nó giống phương thức thông thường ở chỗ: Trong thân của nó có thể truy nhập tới các thành phần của lớp (cụ thể là lớp TAM\_GIAC).

+ Nó khác phương thức thông thường ở chỗ:

* Không có đối ngầm định xác định bởi con trỏ this (như phương thức thông thường). Như vậy phương thức tao\_tg có đúng 3 đối.
* Nó không gắn với một đối tượng cụ thể nào của lớp, nên trong lời gọi tới phương thức ảo có thể dùng tên lớp, ví dụ (xem hàm main):

t=TAM\_GIAC::tao\_tg(d[i],d[j],d[k]);

***Chú ý 2:*** Không thể thay phương thức tĩnh tao\_tg bằng hàm, vì trong thân hàm không được truy xuất đến các thuộc tính của lớp TAM\_GIAC. Tuy nhiên có một giải pháp khác là dùng khái niệm hàm bạn (friend). Hàm bạn của một lớp có quyền truy nhập đến các thuộc tính của lớp. Trong ví dụ 3 dưới đây ta sẽ xây dựng hàm tao\_tg như một hàm bạn của lớp TAM\_GIAC.

***Chú ý 3:*** còn một giải pháp nữa là dùng hàm tạo (constructor) sẽ trình bầy trong các chương sau:

Chương trình dưới đây có nội dung giống như ví dụ 2, nhưng thay phương thức tĩnh tao\_tg bằng hàm bạn tao\_tg.

**Ví dụ 3:** Minh hoạ cách dùng hàm bạn. Nội dung chương trình giống như trong ví dụ 2.

*#include <conio.h> #include <iostream.h> #include <math.h> class DIEM*

*{*

*private:*

*double x,y; // Toa do cua diem public:*

*void nhapsl()*

*{*

*cout << " Toa do x, y: " ; cin >> x >> y ;*

*}*

*void in()*

*{*

*cout << " x = " << x << " y = " << y;*

*}*

*double do\_dai(DIEM d2)*

*{*

*return sqrt(pow(x-d2.x,2) + pow(y-d2.y,2) );*

*}*

*} ;*

*class TAM\_GIAC*

*{*

*private:*

*DIEM d1,d2,d3; // 3 dinh tam giac public:*

*void nhapsl(); void in();*

*friend TAM\_GIAC tao\_tg(DIEM e1, DIEM e2, DIEM e3)*

*{*

*TAM\_GIAC t;*

*t.d1=e1; t.d2 = e2; t.d3=e3; return t;*

*}*

*double dien\_tich() ;*

*TAM\_GIAC maxdt(TAM\_GIAC t2);*

*} ;*

*void TAM\_GIAC::nhapsl()*

*{*

*cout << "\nDinh 1 - " ; d1.nhapsl();*

*cout << "\nDinh 2 - " ; d2.nhapsl();*

*cout << "\nDinh 3 - " ; d3.nhapsl();*

*}*

*void TAM\_GIAC::in()*

*{*

*cout << "\nDinh 1: " ; d1.in();*

*cout << "\nDinh 2: " ; d2.in();*

*cout << "\nDinh 3: " ; d3.in();*

*}*

*double TAM\_GIAC::dien\_tich()*

*{*

*double a,b,c,p,s; a=d1.do\_dai(d2); b=d2.do\_dai(d3); c=d3.do\_dai(d1); p=(a+b+c)/2;*

*return sqrt(p\*(p-a)\*(p-b)\*(p-c));*

*}*

*TAM\_GIAC TAM\_GIAC::maxdt(TAM\_GIAC t2)*

*{*

*if (this->dien\_tich() > t2.dien\_tich()) return \*this ;*

*else*

*return t2;*

*}*

*void main()*

*{*

*DIEM d[50];*

*int n, i ; clrscr();*

*cout << "\n So diem= "; cin >> n;*

*for (i=1; i<=n; ++i)*

*{*

*cout << "\nNhap diem " << i << " - " ; d[i].nhapsl();*

*}*

*int j, k ; TAM\_GIAC tmax, t;*

*tmax = tao\_tg(d[1],d[2],d[3]); for (i=1;i<=n-2;++i)*

*for (j=i+1;j<=n-1;++j) for (k=j+1;k<=n;++k)*

*{*

*t=tao\_tg(d[i],d[j],d[k]); tmax = tmax.maxdt(t);*

*}*

*cout << "\n\nTam giac co dien tich lon nhat: " ; tmax.in();*

*cout << "\nDien tich = " << tmax.dien\_tich(); getch();*

*}*

***Chú ý:*** Hàm bạn có thể xây dựng bên trong định nghĩa lớp (như chương trình trên) hoặc có thể khai báo bên trong và xây dựng bên ngoài định nghĩa lớp như sau:

*class TAM\_GIAC*

*{*

*private:*

*DIEM d1,d2,d3; // 3 dinh tam giac public:*

*void nhapsl(); void in();*

*friend TAM\_GIAC tao\_tg(DIEM e1,DIEM e2,DIEM e3); double dien\_tich() ;*

*TAM\_GIAC maxdt(TAM\_GIAC t2);*

*} ;*

*TAM\_GIAC tao\_tg(DIEM e1, DIEM e2, DIEM e3)*

*{*

*TAM\_GIAC t;*

*t.d1=e1; t.d2 = e2; t.d3=e3; return t;*

*}*

***Nhận xét:*** *Không cho phép dùng từ khoá friend khi xây dựng hàm (bên ngoài lớp)*

## Bài 6. Hàm, hàm bạn

* 1. **Hàm có các tính chất sau:**

+ Phạm vi của hàm là toàn bộ chương trình, vì vậy hàm có thể được gọi tới từ bất kỳ chỗ nào.

Như vây trong các phương thức có thể sử dụng hàm.

+ Đối của hàm có thể là các đối tượng, tuy nhiên có một hạn chế là trong thân hàm không cho phép truy nhập tới thuộc tính của các đối này. Ví dụ giả sử đã định nghĩa lớp:

*class DIEM*

*{*

*private:*

*double x,y; // Toa do cua diem public:*

*void nhapsl()*

*{*

*cout << " Toa do x, y: " ; cin >> x >> y ;*

*}*

*void in()*

*{*

*cout << " x = " << x << " y = " << y;*

*}*

*};*

Dùng lớp DIEM, ta xây dựng hàm tính độ dài của đoạn thẳng đi qua 2 điểm như sau:

*double do\_dai(DIEM d1, DIEM d2)*

*{*

*return sqrt(pow(d1.x-d2.x,2) + pow(d1.y-d2.y,2));*

*}*

Hàm này sẽ bị báo lỗi khi dịch, vì trong thân hàm không cho phép sử dụng các thuộc tính d1.x, d1.y, d2.x, d2.y của các đối tượng d1 và d2 thuộc lớp DIEM.

+ Phạm vi sử dụng của các phương thức (public) là toàn chương trình, vì vậy trong thân hàm có thể gọi tới các phương thức. Ví dụ giả sử đã định nghĩa lớp:

*class DIEM*

*{*

*private:*

*double x,y; // Toa do cua diem public:*

*void nhapsl()*

*{*

*cout << " Toa do x, y: " ; cin >> x >> y ;*

*}*

*void in()*

*{*

*cout << " x = " << x << " y = " << y;*

*}*

*double do\_dai(DIEM d2)*

*{*

*return sqrt(pow(x-d2.x,2) + pow(y-d2.y,2) );*

*}*

*} ;*

Khi đó bằng cách dùng phương thức do\_dai, ta có thể viết hàm tính diện tích tam giác có đỉnh là các đối tượng d1, d2, d3 của lớp DIEM như sau:

*double dt\_tg(DIEM d1, DIEM d2, DIEM d3)*

*{*

*double a,b,c,p,s;*

*a=d1.do\_dai(d2); b=d2.do\_dai(d3); c=d3.do\_dai(d1); p=(a+b+c)/2;*

*return sqrt(p\*(p-a)\*(p-b)\*(p-c));*

*}*

Bằng cách dùng hàm dt\_tg, có thể tổ chức lại chương trình tìm tam giác có diện tích lớn nhất (ở mục trên) một cách đơn giản hơn( bỏ đi lớp TAM\_GIAC) như ví dụ sau.

## Ví dụ 1:

*#include <conio.h> #include <iostream.h> #include <math.h> class DIEM*

*{*

*private:*

*double x,y; // Toa do cua diem public:*

*void nhapsl()*

*{*

*cout << " Toa do x, y: " ; cin >> x >> y ;*

*}*

*void in()*

*{*

*cout << " x = " << x << " y = " << y;*

*}*

*double do\_dai(DIEM d2)*

*{*

*return sqrt(pow(x-d2.x,2) + pow(y-d2.y,2) );*

*}*

*} ;*

*double dt\_tg(DIEM d1, DIEM d2, DIEM d3)*

*{*

*double a,b,c,p,s; a=d1.do\_dai(d2); b=d2.do\_dai(d3); c=d3.do\_dai(d1); p=(a+b+c)/2;*

*return sqrt(p\*(p-a)\*(p-b)\*(p-c));*

*}*

*void main()*

*{*

*DIEM d[50];*

*int n, i,j,k,imax,jmax,kmax ; clrscr();*

*cout << "\n So diem= "; cin >> n;*

*for (i=1; i<=n; ++i)*

*{*

*cout << "\nNhap diem " << i << " - " ; d[i].nhapsl();*

*}*

*imax=1; jmax=2; kmax=3; for (i=1;i<=n-2;++i)*

*for (j=i+1;j<=n-1;++j) for (k=j+1;k<=n;++k)*

*if (dt\_tg(d[i],d[j],d[k]) > dt\_tg(d[imax],d[jmax],d[kmax]))*

*{*

*imax = i ; jmax = j; kmax = k;*

*}*

*cout << "\n\nTam giac co dien tich lon nhat: " ; cout << "\nDinh 1 - "; d[imax].in();*

*cout << "\nDinh 2 - "; d[jmax].in();*

*cout << "\nDinh 3 - "; d[kmax].in();*

*cout << "\nDien tich = " << dt\_tg(d[imax],d[jmax],d[kmax]) ; getch();*

*}*

**Nhận xét:** Chương trình trên làm việc trên mảng d kiểu DIEM. Bây giờ nếu ta dùng mảng ngoài thì từ số thứ tự sẽ suy ra phần tử của mảng. Như vây hàm

double dt\_tg(DIEM d1, DIEM d2, DIEM d3);

có 3 đối kiểu DIEM có thể thay bằng hàm có 3 đối nguyên: double dt\_tg(int i, int j, int k);

để tính diện tích tam giác có đỉnh là d[i], d[j] và d[k] . ý tưởng này được thể hiện trong ví dụ sau.

**Ví dụ 2:** Chương trình dùng mảng đối tượng ngoài.

***Chú ý:*** Khai báo mảng đối tượng phải đặt sau định nghĩa kiểu đối tượng (định nghĩa lớp).

*#include <conio.h> #include <iostream.h> #include <math.h>*

*double dt\_tg(int i, int j, int k); // Khai báo hàm dt\_tg class DIEM*

*{*

*private:*

*double x,y; // Toa do cua diem public:*

*void nhapsl(); void in();*

*double do\_dai(DIEM d2);*

*} ;*

*// Chú ý: Khai báo mảng kiểu DIEM phải đặt sau định nghĩa*

*// lớp DIEM DIEM d[50];*

*void DIEM::nhapsl()*

*{*

*cout << " Toa do x, y: " ; cin >> x >> y ;*

*}*

*void DIEM::in()*

*{*

*cout << " x = " << x << " y = " << y;*

*}*

*double DIEM::do\_dai(DIEM d2)*

*{*

*return sqrt(pow(x-d2.x,2) + pow(y-d2.y,2) );*

*}*

*double dt\_tg(int i, int j, int k)*

*{*

*double a,b,c,p,s; a=d[i].do\_dai(d[j]);*

*b=d[j].do\_dai(d[k]);*

*c=d[k].do\_dai(d[i]); p=(a+b+c)/2;*

*return sqrt(p\*(p-a)\*(p-b)\*(p-c));*

*}*

*void main()*

*{*

*int n, i,j,k,imax,jmax,kmax ; clrscr();*

*cout << "\n So diem= "; cin >> n;*

*for (i=1; i<=n; ++i)*

*{*

*cout << "\nNhap diem " << i << " - " ; d[i].nhapsl();*

*}*

*imax=1; jmax=2; kmax=3; for (i=1;i<=n-2;++i)*

*for (j=i+1;j<=n-1;++j) for (k=j+1;k<=n;++k)*

*if (dt\_tg(i,j,k) > dt\_tg(imax,jmax,kmax))*

*{*

*imax = i ; jmax = j; kmax = k;*

*}*

*cout << "\n\nTam giac co dien tich lon nhat: " ; cout << "\nDinh 1 - "; d[imax].in();*

*cout << "\nDinh 2 - "; d[jmax].in();*

*cout << "\nDinh 3 - "; d[kmax].in();*

*cout << "\nDien tich = " << dt\_tg(imax,jmax,kmax); getch();*

*}*

## Hàm bạn (friend function)

* + 1. **Để một hàm trở thành bạn của một lớp, có 2 cách viết:**

*Cách 1:* Dùng từ khoá friend để khai báo hàm trong lớp và xây dựng hàm bên ngoài như các hàm thông thường (không dùng từ khoá friend). Mẫu viết như sau:

*class A*

*{*

*private:*

*// Khai báo các thuộc tính public:*

*...*

*// Khai báo các hàm bạn của lớp A friend void f1(...);*

*friend double f2(...); friend A f3(...) ;*

*...*

*} ;*

*// Xây dựng các hàm f1, f2, f3 void f1(...)*

*{*

*...*

*}*

*double f2(...)*

*{*

*...*

*}*

*A f3(...)*

*{*

*...*

*}*

*Cách 2:* Dùng từ khoá friend để xây dựng hàm trong định nghĩa lớp. Mẫu viết như sau:

*class A*

*{*

*private:*

*// Khai báo các thuộc tính public:*

*...*

*// Xây dựng các hàm bạn của lớp A void f1(...)*

*{*

*...*

*}*

*double f2(...)*

*{*

*...*

*}*

*A f3(...)*

*{*

*...*

*}*

*...*

*} ;*

## Tính chất của hàm bạn

Trong thân hàm bạn của một lớp có thể truy nhập tới các thuộc tính của các đối tượng thuộc lớp này. Đây là sự khác nhau duy nhất giữa hàm bạn và hàm thông thường. Chú ý rằng hàm bạn không phải là phương thức của lớp. Phương thức có một đối ẩn (ứng với con trỏ this) và lời gọi của phương thức phải gắn với một đối tượng nào đó (địa chỉ đối tượng này được truyền cho con trỏ this). Lời gọi của hàm bạn giống như lời gọi của hàm thông thường.

Ví dụ sau sẽ so sánh phương thức, hàm bạn và hàm tự do (hàm thông thường). Xét lớp SP (số phức). Hãy so sánh 3 phương án để thực hiện việc cộng 2 số phức:

*Phương án 1:* Dùng phương thức

*class SP*

*{*

*private:*

*double a; // Phần thực double b; // Phần ảo*

*public:*

*SP cong(SP u2)*

*{*

*SP u:*

*u.a = this->a + u2.a ;*

*u.b = this->b + u2.b ; return u;*

*}*

*} ;*

Cách dùng

SP u, u1, u2;

u = u1.cong(u2);

*Phương án 2:* Dùng hàm bạn

*class SP*

*{*

*private:*

*double a; // Phần thực double b; // Phần ảo*

*public:*

*friend SP cong(SP u1, SP u2)*

*{*

*SP u:*

*u.a = u1.a + u2.a ;*

*u.b = u1.b + u2.b ; return u;*

*}*

*};*

Cách dùng

SP u, u1, u2;

u = cong(u1, u2);

*Phương án 3:* Dùng hàm tự do

*class SP*

*{*

*private:*

*double a; // Phần thực double b; // Phần ảo*

*public:*

*...*

*} ;*

*SP cong(SP u1, SP u2)*

*{*

*SP u:*

*u.a = u1.a + u2.a ;*

*u.b = u1.b + u2.b ; return u;*

*}*

Phương án này không được chấp nhận, Trình biên dịch sẽ báo lỗi vì trong thân hàm không được quyền truy xuất đến các thuộc tính riêng (private) a, b của các đối tượng u, u1 và u2 thuộc lớp SP.

* + 1. **Một hàm có thể là bạn của nhiều lớp được không?** Câu trả lời là được. Khi một hàm là bạn của nhiều lớp, thì nó có quyền truy nhập tới tất cả các thuộc tính của các đối tượng trong các lớp này. Để làm cho hàm f trở thành bạn của các lớp A, B và C ta sử dụng mẫu viết sau:

class B; // Khai báo trước lớp A class B; // Khai báo trước lớp B class C; // Khai báo trước lớp C

// Định nghĩa lớp A class A

{

// Khai báo f là bạn của A friend void f(...) ;

} ;

// Định nghĩa lớp B class B

{

// Khai báo f là bạn của B friend void f(...) ;

} ;

// Định nghĩa lớp C class C

{

// Khai báo f là bạn của C friend void f(...) ;

} ;

// Xây dụng hàm f

void f(...)

{

...

}

Chương trình sau đây minh hoạ cách dùng hàm bạn (bạn của một lớp và bạn của nhiều lớp). Chương trình đưa vào 2 lớp VT (véc tơ), MT (ma trận) và 3 hàm bạn để thực hiện các thao tác trên 2 lớp này:

// Hàm bạn với lớp VT dùng để in một véc tơ friend void in(const VT &x);

// Hàm bạn với lớp MT dùng để in một ma trận friend void in(const MT &a);

// Hàm bạn với cả 2 lớp MT và VT dùng để nhân ma trận với véc tơ friend VT tich(const MT &a,const VT &x);

Nội dung chương trình là nhập một ma trận vuông cấp n và một véc tơ cấp n, sau đó thực hiện phép nhân ma trận với véc tơ vừa nhập.

// Chương trình CT3\_09.CPP

*#include <conio.h> #include <iostream.h> #include <math.h> class VT;*

*class MT ; class VT*

*{*

*private:*

*int n;*

*double x[20]; // Toa do cua diem public:*

*void nhapsl();*

*friend void in(const VT &x);*

*friend VT tich(const MT &a,const VT &x) ;*

*} ;*

*class MT*

*{*

*private:*

*int n;*

*double a[20][20]; public:*

*friend VT tich(const MT &a,const VT &x); friend void in(const MT &a);*

*void nhapsl();*

*} ;*

*void VT::nhapsl()*

*{*

*cout << "\n Cap vec to = "; cin >> n ;*

*for (int i=1; i<=n ; ++i)*

*{*

*cout << "\nPhan tu thu " << i << " = " ; cin >> x[i];*

*}*

*}*

*void MT::nhapsl()*

*{*

*cout << "\n Cap ma tran = "; cin >> n ;*

*for (int i=1; i<=n ; ++i) for (int j=1; j<=n; ++j)*

*{*

*cout << "\nPhan tu thu hang "<< i << " cot " << j << "=" ; cin >> a[i][j];*

*}*

*}*

*VT tich(const MT &a,const VT &x)*

*{*

*VT y;*

*int n=a.n; if (n!=x.n) return x;*

*y.n = n;*

*for (int i=1; i<=n; ++i)*

*{*

*y.x[i]=0;*

*for (int j=1; j<=n; ++j) y.x[i] += a.a[i][j]\*x.x[j];*

*}*

*return y;*

*}*

*void in(const VT &x)*

*{*

*cout << "\n";*

*for (int i=1; i<=x.n; ++i) cout << x.x[i] << " ";*

*}*

*void in(const MT &a)*

*{*

*for (int i=1; i<=a.n; ++i)*

*{*

*cout << "\n" ;*

*for (int j=1; j<=a.n; ++j) cout << a.a[i][j] << " ";*

*}*

*}*

*void main()*

*{*

*MT a; VT x,y; clrscr();*

*a.nhapsl();*

*x.nhapsl();*

*y=tich(a,x);*

*clrscr();*

*cout << "\nMa tran A:"; in(a);*

*cout << "\n\nVec to x: " ; in(x);*

*cout << "\n\nVec y = Ax: " ; in(y);*

*getch();*

*}*

## Bài 7. Phạm vi truy xuất

* 1. **Các từ khoá private và public**

Các thành phần (thuộc tính và phương thức) của lớp có thể khai báo là private hoặc public theo mẫu:

private:

// Khai báo các thành phần riêng của lớp public:

// Khai báo các thành phần chung (công cộng)

***Chú ý:*** Các thành phần khai báo mặc định (không dùng các từ khoá private và public) được xem là các thành phần private.

* 1. **Các thành phần riêng của lớp** chỉ được sử dụng trong phạm vi của lớp (trong thân các phương thức của lớp). Chúng không thể đem ra sử dụng bên ngoài lớp.

+ Một thuộc tính private: Thuộc tính này (của một đối tượng nào đó) chỉ có thể được sử dụng trong thân của các phương thức cùng lớp.

+ Một phương thức private: Chỉ được sử dụng trong thân của các phương thức cùng lớp.

Ví dụ sau minh hoạ cách dùng phương thức private. Xét lớp PS (phân số) với 2 thuộc tính nguyên là t (tử) và m (mẫu). Giả sử cần xây dựng các phương thức để thực hiện các phép toán cộng trừ, nhân, chia phân số. Do các phép toán này cần dùng trong toàn bộ chương trình, nên các phương thức thực hiện các phép toán cần khai báo là public. Để thực hiện các phép tính trên phân số cần dùng đến phép rút gọn phân số. Ta có thể dùng một phương thức private để làm điều này vì việc rút gọn chỉ dùng trong nội bộ lớp.

* 1. **Các thành phần công cộng của lớp** có phạm vi sử dụng trong toàn chương trình. Như vậy nếu một thuộc tính được khai báo là public, thì nó có thể được truy nhập trong thân của bất kỳ hàm nào trong chương trình.

Ví dụ trong §6 đã chỉ ra phương án dùng một hàm (tự do) để thực hiện phép cộng 2 số phức như sau là sai:

*Phương án 3:* Dùng hàm tự do class SP

{

private:

double a; // Phần thực double b; // Phần ảo

public:

...

} ;

SP cong(SP u1, SP u2)

{

SP u:

u.a = u1.a + u2.a ;

u.b = u1.b + u2.b ; return u;

}

Tuy nhiên nếu sửa chữa bằng cách khai báo các thuộc tính a và b là public thì lại được.

**Nhận xét:** Các thuộc tính thường khai báo là private để đảm bảo tính dấu kín, an toàn dữ liệu của lớp.

## Bài 8. Các phương thức toán tử

* 1. **Cách đặt tên**

Các phương thức toán tử được xây dựng như các phương thức thông thường, chỉ có khác cách đặt tên. Tên các phương thức toán tử (cũng giống như hàm toán tử) được tạo bằng cách ghép từ khoá operator với một phép toán, ví dụ:

operator+ operator<< operator>>

## Con trỏ this

Cũng giống như phương thức thông thường, phương thức toán tử có đối đầu tiên (đối không tường minh) là con trỏ this.

## Toán tử một toán hạng

Các phương thức toán tử một toán hạng: Dùng ngay con trỏ this để biểu thị toán hạng duy nhất này, nên trong phương thức sẽ không có đối tường minh. Ví dụ phương thức toán tử - (đổi dấu) một đối tượng kiểu SP (số phức) có thể viết như sau:

class SP

{

private:

double a; // Phần thực double b; // Phần ảo

public:

SP operator-();

} ;

SP SP:: operator-()

{

SP u ;

u.a = - this->a ;

u.b = - this->b ; return u;

}

Cách dùng:

SP u, v;

u = -v;

## Toán tử hai toán hạng

Các phương thức toán tử hai toán hạng: Con trỏ this ứng với toán hạng thứ nhất, nên trong phương thức chỉ cần dùng một đối tường minh để biểu thị toán hạng thứ hai. Ví dụ phương thức toán tử + (cộng) hai đối tượng kiểu SP (số phức) có thể viết như sau:

class SP

{

private:

double a; // Phần thực double b; // Phần ảo

public:

SP operator+(SP u2);

} ;

SP SP:: operator+(SP u2)

{

SP u ;

u.a = this->a + u2.a ;

u.b = this->b + u2.b ; return u;

}

Cách dùng:

SP p, p, r; r = p + q ;

## Lớp DT (Đa thức)

Chương trình sau sẽ định nghĩa lớp DT và đưa vào các phương thức, hàm:

+ Các thuộc tính:

int n ; // bậc đa thức

double \*a ; // trỏ tới vùng nhớ chứa các hệ số đa thức

+ Các phương thức operator+, operator- dùng để đổi dấu các hệ số đa thức operator+ dùng để cộng 2 đa thức

operator- dùng để trừ 2 đa thức operator\* dùng để nhân 2 đa thức operator^ dùng để tính giá trị đa thức

operator[] dùng để cho biết bậc và hệ số của đa thức

+ Các hàm bạn:

operator<< dùng để in các hệ số đa thức operator>> dùng để nhập các hệ số đa thức

+ Hàm (tự do)

double F(DT p, double x) dùng để tính p(x)-giá trị đa thức tại x

+ Nói thêm về phương thức chỉ số và hàm tự do F

* Nếu p là đối tượng của lớp DT, thì hàm chỉ số cho biết: p[-1] = double(n)

p[i] = a[i] , i=0, 1, ..., n

* Hàm tự do F sẽ dùng phương thức chỉ số để xác định n , các hệ số đa thức và dùng chúng để tính giá trị đa thức.

+ Trong chương trình sử dụng hàm new để cấp phát vùng nhớ chứa hệ số đa thức.

+ Nội dung chương trình gồm:

* Nhập, in các đa thức p, q, r, s
* Tính đa thức: f = -(p + q)\*(r - s)
* Nhập các số thực x1 và x2
* Tính f(x1) (bằng cách dùng phương thức operator^)
* Tính f(x2) (bằng cách dùng hàm F)

// Chương trình CT3\_10.CPP

*#include <conio.h> #include <iostream.h> #include <math.h> class DT*

*{*

*private:*

*int n; // Bac da thuc*

*double \*a; // Tro toi vung nho chua cac he so da thuc*

*// a0, a1,...*

*public:*

*friend ostream& operator<< (ostream& os,const DT &d); friend istream& operator>> (istream& is,DT &d);*

*DT operator-();*

*DT operator+(const DT &d2); DT operator-(DT d2);*

*DT operator\*(const DT &d2);*

*double operator^(const double &x); // Tinh gia tri da thuc double operator[](int i)*

*{*

*if(i<0)*

*return double(n); else*

*return a[i];*

*}*

*} ;*

*// Ham tinh gia tri da thuc double F(DT d,double x)*

*{*

*double s=0.0 , t=1.0; int n;*

*n = int(d[-1]);*

*for (int i=0; i<=n; ++i)*

*{*

*s += d[i]\*t; t \*= x;*

*}*

*return s;*

*}*

*ostream& operator<< (ostream& os,const DT &d)*

*{*

*os << " - Cac he so (tu ao): " ; for (int i=0 ; i<= d.n ; ++i)*

*os << d.a[i] <<" " ; return os;*

*}*

*istream& operator>> (istream& is,DT &d)*

*{*

*cout << " - Bac da thuc: " ; cin >> d.n;*

*d.a = new double[d.n+1];*

*cout << "Nhap cac he so da thuc:\n" ; for (int i=0 ; i<= d.n ; ++i)*

*{*

*cout << "He so bac " << i << " = " ; is >> d.a[i] ;*

*}*

*return is;*

*}*

*DT DT::operator-()*

*{*

*DT p;*

*p.n = n;*

*p.a = new double[n+1]; for (int i=0 ; i<=n ; ++i)*

*p.a[i] = -a[i]; return p;*

*}*

*DT DT::operator+(const DT &d2)*

*{*

*DT d;*

*int k,i;*

*k = n > d2.n ? n : d2.n ;*

*d.a = new double[k+1]; for (i=0; i<=k ; ++i)*

*if (i<=n && i<=d2.n) d.a[i] = a[i] + d2.a[i];*

*else if (i<=n) d.a[i] = a[i];*

*else*

*d.a[i] = d2.a[i];*

*i=k;*

*while(i>0 && d.a[i]==0.0) --i;*

*d.n = i; return d ;*

*}*

*DT DT::operator-(DT d2)*

*{*

*return (\*this + (-d2));*

*}*

*DT DT::operator\*(const DT &d2)*

*{*

*DT d;*

*int k, i, j;*

*k = d.n = n + d2.n ;*

*d.a = new double[k+1];*

*for (i=0; i<=k; ++i) d.a[i] = 0; for (i=0 ; i<= n ; ++i)*

*for (j=0 ; j<= d2.n ; ++j) d.a[i+j] += a[i]\*d2.a[j] ;*

*return d;*

*}*

*double DT::operator^(const double &x)*

*{*

*double s=0.0 , t=1.0;*

*for (int i=0 ; i<= n ; ++i)*

*{*

*s += a[i]\*t; t \*= x;*

*}*

*return s;*

*}*

*void main()*

*{*

*DT p,q,r,s,f;*

*double x1,x2,g1,g2; clrscr();*

*cout <<"\nNhap da thuc P " ; cin >> p; cout << "\nDa thuc p " << p ;*

*cout <<"\nNhap da thuc Q " ; cin >> q;*

*cout << "\nDa thuc q " << q ;*

*cout <<"\nNhap da thuc R " ; cin >> r; cout << "\nDa thuc r " << r ;*

*cout <<"\nNhap da thuc S " ; cin >> s; cout << "\nDa thuc s " << s ;*

*f = -(p+q)\*(r-s);*

*cout << "\nNhap so thuc x1: " ; cin >> x1; cout << "\nNhap so thuc x2: " ; cin >> x2; g1 = f^x1;*

*g2 = F(f,x2);*

*cout << "\nDa thuc f " << f ;*

*cout << "\n f("<<x1<<") = " << g1; cout << "\n f("<<x2<<") = " << g2; getch();*

*}*

**Chương 4**

# Hàm tạo, hàm huỷ và các vấn đề liên quan

Chương này trình bầy một số vấn đề có tính chuyên sâu hơn về lớp như:

+ Hàm tạo (constructor)

+ Hàm huỷ (destructor)

+ Toán tử gán và hàm tạo sao chép

+ Mối liên quan giữa hàm tạo và đối tượng thành phần

+ Các thành phần tĩnh

+ Lớp bạn, hàm bạn

+ Đối tượng hằng

+ Phương thức inline

## Bài 1. Hàm tạo (constructor)

* 1. **Công dụng**

Hàm tạo cũng là một phương thức của lớp (nhưng khá đặc biệt) dùng để tạo dựng một đối tượng mới. Chương trình dịch sẽ cấp phát bộ nhớ cho đối tượng sau đó sẽ gọi đến hàm tạo. Hàm tạo sẽ khởi gán giá trị cho các thuộc tính của đối tượng và có thể thực hiện một số công việc khác nhằm chuẩn bị cho đối tượng mới.

## Cách viết hàm tạo

* + 1. **Điểm khác của hàm tạo và các phương thức thông thường**

Khi viết hàm tạo cần để ý 3 sự khác biệt của hàm tạo so với các phương thức khác như sau:

+ Tên của hàm tạo: Tên của hàm tạo bắt buộc phải trùng với tên của lớp.

+ Không khai báo kiểu cho hàm tạo.

+ Hàm tạo không có kết quả trả về.

## Sự giống nhau của hàm tạo và các phương thức thông thường

Ngoài 3 điểm khác biệt trên, hàm tạo được viết như các phương thức khác:

+ Hàm tạo có thể được xây dựng bên trong hoặc bên ngoài định nghĩa lớp.

+ Hàm tạo có thể có đối hoặc không có đối.

+ Trong một lớp có thể có nhiều hàm tạo (cùng tên nhưng khác bộ đối).

**Ví dụ** sau định nghĩa lớp DIEM\_DH (Điểm đồ hoạ) có 3 thuộc tính: int x; // hoành độ (cột) của điểm

int y; // tung độ (hàng) của điểm int m; // mầu của điểm

và đưa vào 2 hàm tạo để khởi gán cho các thuộc tính của lớp:

// Hàm tạo không đối: Dùng các giá trị cố định để khởi gán cho

// x, y, m DIEM\_DH() ;

// Hàm tạo có đối: Dùng các đối x1, y1, m1 để khởi gán cho

// x, y, m

// Đối m1 có giá trị mặc định 15 (mầu trắng) DIEM\_DH(int x1, int y1, int m1=15) ;

class DIEM\_DH

{

private:

int x, y, m ; public:

//Hàm tạo không đối: khởi gán cho x=0, y=0, m=1

// Hàm này viết bên trong định nghĩa lớp DIEM\_DH()

{

x=y=0; m=1;

}

// Hàm tạo này xây dựng bên ngoài định nghĩa lớp DIEM\_DH(int x1, int y1, int m1=15) ;

// Các phương thức khác

} ;

// Xây dựng hàm tạo bên ngoài định nghĩa lớp DIEM\_DH:: DIEM\_DH(int x1, int y1, int m1)

{

x=x1; y=y1; m=m1;

}

## Dùng hàm tạo trong khai báo

+ Khi đã xây dựng các hàm tạo, ta có thể dùng chúng trong khai báo để tạo ra một đối tượng đồng thời khởi gán cho các thuộc tính của đối tượng được tạo. Dựa vào các tham số trong khai báo mà Trình biên dịch sẽ biết cần gọi đến hàm tạo nào.

+ Khi khai báo một biến đối tượng có thể sử dụng các tham số để khởi gán cho các thuộc tính của biến đối tượng.

+ Khi khai báo mảng đối tượng không cho phép dùng các tham số để khởi gán.

+ Câu lệnh khai báo một biến đối tượng sẽ gọi tới hàm tạo 1 lần

+ Câu lệnh khai báo một mảng n đối tượng sẽ gọi tới hàm tạo n lần.

## Ví dụ:

DIEM\_DH d; // Gọi tới hàm tạo không đối.

// Kết quả d.x=0, d.y=0, d.m=1 DIEM\_DH u(200,100,4); // Gọi tới hàm tạo có đối.

// Kết quả u.x=200, u.y=100, d.m=4 DIEM\_DH v(300,250); // Gọi tới hàm tạo có đối.

// Kết quả v.x=300, v.y=250, d.m=15 DIEM\_DH p[10] ; // Gọi tới hàm tạo không đối 10 lần

***Chú ý:*** Với các hàm có đối kiểu lớp, thì đối chỉ xem là các tham số hình thức, vì vậy khai báo đối (trong dòng đầu của hàm) sẽ không tạo ra đối tượng mới và do đó không gọi tới các hàm tạo.

## Dùng hàm tạo trong cấp phát bộ nhớ

+ Khi cấp phát bộ nhớ cho một đối tượng có thể dùng các tham số để khởi gán cho các thuộc tính của đối tượng, ví dụ:

DIEM\_DH \*q =new DIEM\_DH(50,40,6);//Gọi tới hàm tạo có đối

// Kết quả q->x=50, q->y=40, q->m=6 DIEM\_DH \*r = new DIEM\_DH ; // Gọi tới hàm tạo không đối

// Kết quả r->x=0, r->y= 0, r->m=1

+ Khi cấp phát bộ nhớ cho một dẫy đối tượng không cho phép dùng tham số để khởi gán, ví dụ: int n=20;

DIEM\_DH \*s = new DIEM\_DH[n] ; // Gọi tới hàm tạo không

// đối 20 lần.

## Dùng hàm tạo để biểu diễn các đối tượng hằng

+ Như đã biết, sau khi định nghĩa lớp DIEM\_DH thì có thể xem lớp này như một kiểu dữ liệu như int, double, char, ...

Với kiểu int chúng ta có các hằng int, như 356.

Với kiểu double chúng ta có các hằng double, như 98.75

Khái niệm hằng kiểu int, hằng kiểu double có thể mở rộng cho hằng kiểu DIEM\_DH

+ Để biểu diễn một hằng đối tượng (hay còn gọi: Đối tượng hằng) chúng ta phải dùng tới hàm tạo. Mẫu viết như sau:

Tên\_lớp(danh sách tham số) ;

**Ví dụ** đối với lớp DIEM\_DH nói trên, có thể viết như sau:

DIEM\_DH(345,123,8) // Biểu thị một đối tượng kiểu DIEM\_DH

// có các thuộc tính x=345, y=123, m=8

***Chú ý:*** Có thể sử dụng một hằng đối tượng như một đối tượng. Nói cách khác, có thể dùng hằng đối tượng để thực hiện một phương thức, ví dụ nếu viết:

DIEM\_DH(345,123,8).in();

thì có nghĩa là thực hiện phương thức in() đối với hằng đối tượng.

## Ví dụ minh hoạ

Chương trình sau đây minh hoạ cách xây dựng hàm tạo và cách sử dùng hàm tạo trong khai báo, trong cấp phát bộ nhớ và trong việc biểu diễn các hằng đối tượng.

*//CT4\_02.CPP*

*#include <conio.h> #include <iostream.h> #include <iomanip.h> class DIEM\_DH*

*{*

*private:*

*int x,y,m;*

*public:*

*// Hàm bạn dùng để in đối tượng DIEM\_DH friend void in(DIEM\_DH d)*

*{*

*cout <<"\n " << d.x << " "<< d.y<<" " << d.m ;*

*}*

*// Phương thức dùng để in đối tượng DIEM\_DH void in()*

*{*

*cout <<"\n " << x << " "<< y<<" " << m ;*

*}*

*//Hàm tạo không đối DIEM\_DH()*

*{*

*x=y=0; m=1;*

*}*

*//Hàm tạo có đối, đối m1 có giá trị mặc định là 15 (mầu trắng) DIEM\_DH(int x1,int y1,int m1=15);*

*};*

*//Xây dựng hàm tạo DIEM\_DH::DIEM\_DH(int x1,int y1,int m1)*

*{*

*x=x1; y=y1; m=m1;*

*}*

*void main()*

*{*

*DIEM\_DH d1; // Gọi tới hàm tạo không đối DIEM\_DH d2(200,200,10); // Gọi tới hàm tạo có đối DIEM\_DH \*d;*

*d= new DIEM\_DH(300,300); // Gọi tới hàm tạo có đối clrscr();*

*in(d1); //Gọi hàm bạn in() d2.in();//Gọi phương thức in() in(\*d); //Gọi hàm bạn in()*

*DIEM\_DH(2,2,2).in();//Gọi phương thức in() DIEM\_DH t[3]; // 3 lần gọi hàm tạo không đối DIEM\_DH \*q; // Gọi hàm tạo không đối*

*int n;*

*cout << "\nN= "; cin >> n;*

*q=new DIEM\_DH[n+1]; // (n+1) lần gọi hàm tạo không đối for (int i=0;i<=n;++i)*

*q[i]=DIEM\_DH(300+i,200+i,8);//(n+1) lần gọi hàm tạo có đối for (i=0;i<=n;++i)*

*q[i].in(); // Gọi phương thức in() for (i=0;i<=n;++i)*

*DIEM\_DH(300+i,200+i,8).in();// Gọi phương thức in() getch();*

*}*

## Bài 2. Lớp không có hàm tạo và hàm tạo mặc định

Các chương trình nêu trong chương 3 đều không có hàm tạo. Vậy khi đó các đối tượng được hình thành như thế nào ?

* 1. **Nếu lớp không có hàm tạo**, Chương trình dịch sẽ cung cấp một hàm tạo mặc định không đối (default). Hàm này thực chất không làm gì cả. Như vậy một đối tượng tạo ra chỉ được cấp phát bộ nhớ, còn các thuộc tính của nó chưa được xác định. Chúng ta có thể kiểm chứng điều này, bằng cách chạy chương trình sau:

*//CT4\_03.CPP*

*// Hàm tạo mặc định #include <conio.h> #include <iostream.h> class DIEM\_DH*

*{*

*private:*

*int x,y,m; public:*

*// Phuong thuc void in()*

*{*

*cout <<"\n " << x << " "<< y<<" " << m ;*

*}*

*};*

*void main()*

*{*

*DIEM\_DH d;*

*d.in();*

*DIEM\_DH \*p;*

*p= new DIEM\_DH[10];*

*clrscr();*

*d.in();*

*for (int i=0;i<10;++i)*

*(p+i)->in(); getch();*

*}*

* 1. **Nếu trong lớp đã có ít nhất một hàm tạo,** thì hàm tạo mặc định sẽ không được phát sinh nữa. Khi đó mọi câu lệnh xây dựng đối tượng mới đều sẽ gọi đến một hàm tạo của lớp. Nếu không tìm thấy hàm tạo cần gọi thì Chương trình dịch sẽ báo lỗi. Điều này thường xẩy ra khi chúng ta không xây dựng hàm tạo không đối, nhưng lại sử dụng các khai báo không tham số như ví dụ sau:

*#include <conio.h> #include <iostream.h> class DIEM\_DH*

*{*

*private:*

*int x,y,m; public:*

*// Phương thức dùng để in đối tượng DIEM\_DH void in()*

*{*

*cout <<"\n " << x << " "<< y<<" " << m ;*

*}*

*//Hàm tạo có đối DIEM\_DH::DIEM\_DH(int x1,int y1,int m1)*

*{*

*x=x1; y=y1; m=m1;*

*}*

*};*

*void main()*

*{*

*DIEM\_DH d1(200,200,10); // Gọi tới hàm tạo có đối DIEM\_DH d2; // Gọi tới hàm tạo không đối*

*d2= DIEM\_DH(300,300,8); // Gọi tới hàm tạo có đối d1.in();*

*d2.in();*

*getch();*

*}*

Trong các câu lệnh trên, chỉ có câu lệnh thứ 2 trong hàm main() là bị báo lỗi. Câu lệnh này sẽ gọi tới hàm tạo không đối, mà hàm này chưa được xây dựng.

*Giải pháp:* Có thể chọn một trong 2 giải pháp sau:

* Xây dựng thêm hàm tạo không đối.
* Gán giá trị mặc định cho tất cả các đối x1, y1 và m1 của hàm tạo đã xây dựng ở trên. Theo phương án 2, chương trình có thể sửa như sau:

*#include <conio.h>*

*#include <iostream.h> class DIEM\_DH*

*{*

*private:*

*int x,y,m; public:*

*// Phương thức dùng để in đối tượng DIEM\_DH void in()*

*{*

*cout <<"\n " << x << " "<< y<<" " << m ;*

*}*

*//Hàm tạo có đối , tất cả các đối đều có giá trị mặc định DIEM\_DH::DIEM\_DH(int x1=0,int y1=0,int m1=15)*

*{*

*x=x1; y=y1; m=m1;*

*}*

*};*

*void main()*

*{*

*DIEM\_DH d1(200,200,10); // Gọi tới hàm tạo, không dùng*

*// tham số mặc định*

*DIEM\_DH d2; // Gọi tới hàm tạo , dùng 3 tham số mặc định d2= DIEM\_DH(300,300); // Gọi tới hàm tạo, dùng 1 tham số*

*// mặc định*

*d1.in();*

*d2.in();*

*getch();*

*}*

## Bài 3. Lớp đa thức

Chương trình dưới đây là sự cải tiến chương trình trong mục 8.5 của chương 3 bằng cách đưa vào 2 hàm tạo:

*//Hàm tạo không đối DT()*

*{*

*this->n=0; this->a=NULL;*

*}*

*//Hàm tạo có đối DT(int n1)*

*{*

*this->n=n1 ;*

*this->a = new double[n1+1];*

*}*

Hàm tạo có đối sẽ tạo một đối tượng mới (kiểu DT) gồm 2 thuộc tính là biến nguyên n và con trỏ

a. Ngoài ra còn cấp phát bộ vùng nhớ (cho a) để chứa các hệ số của đa thức.

Nếu không xây dựng hàm tạo, mà sử dụng hàm tạo mặc định thì các đối tượng (kiểu DT) tạo ra bởi các lệnh khai báo sẽ chưa có bộ nhớ để chứa đa thức. Như vậy đối tượng tạo ra chưa hoàn chỉnh và chưa dùng được. Để có một đối tượng hoàn chỉnh phải qua 2 bước:

+ Dùng khai báo để tạo các đối tượng, ví dụ: DT d;

+ Cấp phát vùng nhớ (cho đối tượng) để chứa đa thức, ví dụ:

d.n = m;

d.a = new double[m+1] ;

Quy trình này được áp dụng trong các phương thức toán tử của chương trình trong mục 8.5 chương 3. Rõ ràng quy trình này vừa dài vừa không tiện lợi, lại hay mắc lỗi, vì người lập trình hay quên không cấp phát bộ nhớ.

Việc dùng các hàm tạo để sản sinh ra các đối tượng hoàn chỉnh tỏ ra tiện lợi hơn, vì tránh được các thao tác phụ (như cấp phát bộ nhớ) nằm bên ngoài khai báo. Phương án dùng hàm tạo sẽ được sử dụng trong các phương thức toán tử của chương trình dưới đây:

+ Nội dung chương trình gồm:

* Nhập, in các đa thức p, q, r, s
* Tính đa thức: f = -(p + q)\*(r - s)
* Nhập các số thực x1 và x2
* Tính f(x1) (bằng cách dùng phương thức operator^)
* Tính f(x2) (bằng cách dùng hàm F)

*// CT4\_05.CPP*

*#include <conio.h> #include <iostream.h> #include <math.h> class DT*

*{*

*private:*

*int n; // Bac da thuc*

*double \*a; // Tro toi vung nho chua cac he so da thuc*

*// a0, a1,...*

*public:*

*DT()*

*{*

*this->n=0; this->a=NULL;*

*}*

*DT(int n1)*

*{*

*this->n=n1 ;*

*this->a = new double[n1+1];*

*}*

*friend ostream& operator<< (ostream& os,const DT &d);*

*friend istream& operator>> (istream& is,DT &d); DT operator-();*

*DT operator+(const DT &d2); DT operator-(DT d2);*

*DT operator\*(const DT &d2);*

*double operator^(const double &x); // Tinh gia tri da thuc double operator[](int i)*

*{*

*if (i<0)*

*return double(n); else*

*return a[i];*

*}*

*} ;*

*// Ham tinh gia tri da thuc double F(DT d,double x)*

*{*

*double s=0.0 , t=1.0; int n;*

*n = int(d[-1]);*

*for (int i=0; i<=n; ++i)*

*{*

*s += d[i]\*t; t \*= x;*

*}*

*return s;*

*}*

*ostream& operator<< (ostream& os,const DT &d)*

*{*

*os << " - Cac he so (tu ao): " ; for (int i=0 ; i<= d.n ; ++i)*

*os << d.a[i] <<" " ; return os;*

*}*

*istream& operator>> (istream& is,DT &d)*

*{*

*if (d.a!=NULL) delete d.a; cout << " - Bac da thuc: " ; cin >> d.n;*

*d.a = new double[d.n+1];*

*cout << "Nhap cac he so da thuc:\n" ; for (int i=0 ; i<= d.n ; ++i)*

*{*

*cout << "He so bac " << i << " = " ; is >> d.a[i] ;*

*}*

*return is;*

*}*

*DT DT::operator-()*

*{*

*DT p(this->n);*

*for (int i=0 ; i<=n ; ++i) p.a[i] = -a[i];*

*return p;*

*}*

*DT DT::operator+(const DT &d2)*

*{*

*int k,i;*

*k = n > d2.n ? n : d2.n ; DT d(k);*

*for (i=0; i<=k ; ++i)*

*if (i<=n && i<=d2.n) d.a[i] = a[i] + d2.a[i];*

*else if (i<=n) d.a[i] = a[i];*

*else*

*d.a[i] = d2.a[i]; i=k;*

*while(i>0 && d.a[i]==0.0) --i;*

*d.n = i; return d ;*

*}*

*DT DT::operator-(DT d2)*

*{*

*return (\*this + (-d2));*

*}*

*DT DT::operator\*(const DT &d2)*

*{*

*int k, i, j;*

*k = n + d2.n ; DT d(k);*

*for (i=0; i<=k; ++i) d.a[i] = 0; for (i=0 ; i<= n ; ++i)*

*for (j=0 ; j<= d2.n ; ++j) d.a[i+j] += a[i]\*d2.a[j] ;*

*return d;*

*}*

*double DT::operator^(const double &x)*

*{*

*double s=0.0 , t=1.0;*

*for (int i=0 ; i<= n ; ++i)*

*{*

*s += a[i]\*t; t \*= x;*

*}*

*return s;*

*}*

*void main()*

*{*

*DT p,q,r,s,f;*

*double x1,x2,g1,g2; clrscr();*

*cout <<"\nNhap da thuc P " ; cin >> p; cout << "\nDa thuc p " << p ;*

*cout <<"\nNhap da thuc Q " ; cin >> q; cout << "\nDa thuc q " << q ;*

*cout <<"\nNhap da thuc R " ; cin >> r; cout << "\nDa thuc r " << r ;*

*cout <<"\nNhap da thuc S " ; cin >> s; cout << "\nDa thuc s " << s ;*

*f = -(p+q)\*(r-s);*

*cout << "\nNhap so thuc x1: " ; cin >> x1; cout << "\nNhap so thuc x2: " ; cin >> x2; g1 = f^x1;*

*g2 = F(f,x2);*

*cout << "\nDa thuc f " << f ;*

*cout << "\n f("<<x1<<") = " << g1; cout << "\n f("<<x2<<") = " << g2; getch();*

*}*

## Bài 4. Hàm tạo sao chép (copy constructor)

* 1. **Hàm tạo sao chép mặc định**

Giả sử đã định nghĩa một lớp nào đó, ví dụ lớp PS (phân số). Khi đó:

+ Ta có thể dùng câu lệnh khai báo hoặc cấp phát bộ nhớ để tạo các đối tượng mới, ví dụ: PS p1, p2 ;

PS \*p = new PS ;

+ Ta cũng có thể dùng lệnh khai báo để tạo một đối tượng mới từ một đối tượng đã tồn tại, ví dụ: PS u;

PS v(u) ; // Tạo v theo u

ý nghĩa của câu lệnh này như sau:

- Nếu trong lớp PS chưa xây dựng hàm tạo sao chép, thì câu lệnh này sẽ gọi tới một hàm tạo sao chép mặc định (của C++). Hàm này sẽ sao chép nội dung từng bit của u vào các bit tương ứng của

v. Như vậy các vùng nhớ của u và v sẽ có nội dung như nhau. Rõ ràng trong đa số các trường hợp, nếu lớp không có các thuộc tính kiểu con trỏ hay tham chiếu, thì việc dùng các hàm tạo sao chép mặc định (để tạo ra một đối tượng mới có nội dung như một đối tượng cho trước) là đủ và không cần xây dựng một hàm tạo sao chép mới.

- Nếu trong lớp PS đã có hàm tạo sao chép (cách viết sẽ nói sau) thì câu lệnh: PS v(u) ;

sẽ tạo ra đối tượng mới v, sau đó gọi tới hàm tạo sao chép để khởi gán v theo u.

**Ví dụ** sau minh hoạ cách dùng hàm tạo sao chép mặc định:

Trong chương trình đưa vào lớp PS (phân số):

+ Các thuộc tính gồm: t (tử số) và m (mẫu).

+ Trong lớp không có phương thức nào cả mà chỉ có 2 hàm bạn là các hàm toán tử nhập (>>) và xuất (<<).

+ Nội dung chương trình là: Dùng lệnh khai báo để tạo một đối tương u (kiểu PS) có nội dung như đối tượng đã có d.

*//CT4\_06.CPP*

*// Ham tao sao chep mac dinh #include <conio.h>*

*#include <iostream.h> class PS*

*{*

*private:*

*int t,m ; public:*

*friend ostream& operator<< (ostream& os,const PS &p)*

*{*

*os << " = " << p.t << "/" << p.m; return os;*

*}*

*friend istream& operator>> (istream& is, PS &p)*

*{*

*cout << " - Nhap tu va mau: " ; is >> p.t >> p.m ;*

*return is;*

*}*

*};*

*void main()*

*{*

*PS d;*

*cout << "\n Nhap PS d"; cin >> d; cout << "\n PS d " << d;*

*PS u(d);*

*cout << "\n PS u " << u; getch();*

*}*

## Cách xây dựng hàm tạo sao chép

+ Hàm tạo sao chép sử dụng một đối **kiểu tham chiếu đối tượng** để khởi gán cho đối tượng mới. Hàm tạo sao chép được viết theo mẫu:

Tên\_lớp (const Tên\_lớp & dt)

{

// Các câu lệnh dùng các thuộc tính của đối tượng dt

// để khởi gán cho các thuộc tính của đối tượng mới

}

+ **Ví dụ** có thể xây dựng hàm tạo sao chép cho lớp PS như sau:

*class PS*

*{*

*private:*

*int t,m ; public:*

*PS (const PS &p)*

*{*

*this->t = p.t ; this->m = p.m ;*

*}*

*...*

*} ;*

## Khi nào cần xây dựng hàm tạo sao chép

**+ Nhận xét:** Hàm tạo sao chép trong ví dụ trên không khác gì hàm tạo sao chép mặc định.

+ Khi lớp không có các thuộc tính kiểu con trỏ hoặc tham chiếu, thì dùng hàm tạo sao chép mặc định là đủ.

+ Khi lớp có các thuộc tính con trỏ hoặc tham chiếu, thì hàm tạo sao chép mặc định chưa đáp ứng được yêu cầu. Ví dụ lớp DT (đa thức) trong §3:

*class DT*

*{*

*private:*

*int n; // Bac da thuc*

*double \*a; // Tro toi vung nho chua cac he so da thuc*

*// a0, a1,...*

*public:*

*DT()*

*{*

*this->n=0; this->a=NULL;*

*}*

*DT(int n1)*

*{*

*this->n=n1 ;*

*this->a = new double[n1+1];*

*}*

*friend ostream& operator<< (ostream& os,const DT &d); friend istream& operator>> (istream& is,DT &d);*

*....*

*} ;*

Bây giờ chúng ta hãy theo rõi xem việc dùng hàm tạo mặc định trong đoạn chương trình sau sẽ dẫn đến sai lầm như thế nào:

DT d ;

// Tạo đối tượng d kiểu DT cin >> d ;

/\* Nhập đối tượng d , gồm: nhập một số nguyên dương và gán cho d.n, cấp phát vùng nhớ cho d.a, nhập các hệ số của đa thức và chứa vào vùng nhớ được cấp phát

\*/

DT u(d) ;

/\* Dùng hàm tạo mặc định để xây dựng đối tượng u theo d Kết quả: u.n = d.n và u.a = d.a. Như vậy 2 con trỏ u.a và

d.a cùng trỏ đến một vùng nhớ.

\*/

**Nhận xét:** Mục đích của ta là tạo ra một đối tượng u giống như d, nhưng độc lập với d. Nghĩa là khi d thay đổi thì u không bị ảnh hưởng gì. Thế nhưng mục tiêu này không đạt được, vì u và d có

chung một vùng nhớ chứa hệ số của đa thức, nên khi sửa đổi các hệ số của đa thức trong d thì các hệ số của đa thức trong u cũng thay đổi theo. Còn một trường hợp nữa cũng dẫn đến lỗi là khi một trong 2 đối tượng u và d bị giải phóng (thu hồi vùng nhớ chứa đa thức) thì đối tượng còn lại cũng sẽ không còn vùng nhớ nữa.

Ví dụ sau sẽ minh hoạ nhận xét trên: Khi d thay đổi thì u cũng thay đổi và ngược lại khi u thay đổi thì d cũng thay đổi theo.

*//CT4\_07.CPP*

*#include <conio.h> #include <iostream.h> #include <math.h> class DT*

*{*

*private:*

*int n; // Bac da thuc*

*double \*a; // Tro toi vung nho chua cac he so da thuc*

*// a0, a1,...*

*public:*

*DT()*

*{*

*this->n=0; this->a=NULL;*

*}*

*DT(int n1)*

*{*

*this->n=n1 ;*

*this->a = new double[n1+1];*

*}*

*friend ostream& operator<< (ostream& os,const DT &d); friend istream& operator>> (istream& is,DT &d);*

*} ;*

*ostream& operator<< (ostream& os,const DT &d)*

*{*

*os << " - Cac he so (tu ao): " ; for (int i=0 ; i<= d.n ; ++i)*

*os << d.a[i] <<" " ; return os;*

*}*

*istream& operator>> (istream& is,DT &d)*

*{*

*if (d.a!=NULL) delete d.a; cout << " - Bac da thuc: " ; cin >> d.n;*

*d.a = new double[d.n+1];*

*cout << "Nhap cac he so da thuc:\n" ; for (int i=0 ; i<= d.n ; ++i)*

*{*

*cout << "He so bac " << i << " = " ; is >> d.a[i] ;*

*}*

*return is;*

*}*

*void main()*

*{*

*DT d;*

*clrscr();*

*cout <<"\nNhap da thuc d " ; cin >> d; DT u(d);*

*cout << "\nDa thuc d " << d ; cout << "\nDa thuc u " << u ;*

*cout <<"\nNhap da thuc d " ; cin >> d; cout << "\nDa thuc d " << d ;*

*cout << "\nDa thuc u " << u ;*

*cout <<"\nNhap da thuc u " ; cin >> u; cout << "\nDa thuc d " << d ;*

*cout << "\nDa thuc u " << u ; getch();*

*}*

## Ví dụ về hàm tạo sao chép

Trong chương trình trên đã chỉ rõ: Hàm tạo sao chép mặc định là chưa thoả mãn đối với lớp DT. Vì vậy cần viết hàm tạo sao chép để xây dựng đối tượng mới ( ví dụ u) từ một đối tượng đang tồn tại (ví dụ d) theo các yêu cầu sau:

+ Gán d.n cho u.n

+ Cấp phát một vùng nhớ cho u.a để có thể chứa được (d.n + 1) hệ số.

+ Gán các hệ số chứa trong vùng nhớ của d.a sang vùng nhớ của u.a

Như vây chúng ta sẽ tạo được đối tượng u có nội dung ban đầu giống như d, nhưng độc lập với

d.

Để đáp ứng các yêu cầu nêu trên, hàm tạo sao chép cần được xây dựng như sau: DT::DT(const DT &d)

{

this->n = d.n;

this->a = new double[d.n+1]; for (int i=0;i<=d.n;++i)

this->a[i] = d.a[i];

}

Chương trình sau sẽ minh hoạ điều này: Sự thay đổi của d không làm ảnh hưởng đến u và ngược lại sự thay đổi của u không làm ảnh hưởng đến d.

//CT4\_08.CPP

// Viết hàm tạo sao chép cho lớp DT

*#include <conio.h> #include <iostream.h> #include <math.h> class DT*

*{*

*private:*

*int n; // Bac da thuc*

*double \*a; // Tro toi vung nho chua cac he so da thuc*

*// a0, a1,...*

*public:*

*DT()*

*{*

*this->n=0; this->a=NULL;*

*}*

*DT(int n1)*

*{*

*this->n=n1 ;*

*this->a = new double[n1+1];*

*}*

*DT(const DT &d);*

*friend ostream& operator<< (ostream& os,const DT &d); friend istream& operator>> (istream& is,DT &d);*

*} ;*

*DT::DT(const DT &d)*

*{*

*this->n = d.n;*

*this->a = new double[d.n+1]; for (int i=0;i<=d.n;++i)*

*this->a[i] = d.a[i];*

*}*

*ostream& operator<< (ostream& os,const DT &d)*

*{*

*os << " - Cac he so (tu ao): " ; for (int i=0 ; i<= d.n ; ++i)*

*os << d.a[i] <<" " ; return os;*

*}*

*istream& operator>> (istream& is,DT &d)*

*{*

*if (d.a!=NULL) delete d.a; cout << " - Bac da thuc: " ; cin >> d.n;*

*d.a = new double[d.n+1];*

*cout << "Nhap cac he so da thuc:\n" ; for (int i=0 ; i<= d.n ; ++i)*

*{*

*cout << "He so bac " << i << " = " ; is >> d.a[i] ;*

*}*

*return is;*

*}*

*void main()*

*{*

*DT d;*

*clrscr();*

*cout <<"\nNhap da thuc d " ; cin >> d; DT u(d);*

*cout << "\nDa thuc d " << d ; cout << "\nDa thuc u " << u ;*

*cout <<"\nNhap da thuc d " ; cin >> d; cout << "\nDa thuc d " << d ;*

*cout << "\nDa thuc u " << u ;*

*cout <<"\nNhap da thuc u " ; cin >> u; cout << "\nDa thuc d " << d ;*

*cout << "\nDa thuc u " << u ; getch();*

*}*

## Bài 5. Hàm huỷ (Destructor)

* 1. **Công dụng của hàm huỷ**

Hàm huỷ là một hàm thành viên của lớp (phương thức) có chức năng ngược với hàm tạo. Hàm huỷ được gọi trước khi giải phóng (xoá bỏ) một đối tượng để thực hiện một số công việc có tính “dọn dẹp” trước khi đối tượng được huỷ bỏ, ví dụ như giải phóng một vùng nhớ mà đối tượng đang quản lý, xoá đối tượng khỏi màn hình nếu như nó đang hiển thị, ...

Việc huỷ bỏ một đối tượng thường xẩy ra trong 2 trường hợp sau:

+ Trong các toán tử và các hàm giải phóng bộ nhớ, như delete, free,...

+ Giải phóng các biến, mảng cục bộ khi thoát khỏi hàm, phương thức.

## Hàm huỷ mặc định

Nếu trong lớp không định nghĩa hàm huỷ, thì một hàm huỷ mặc định không làm gì cả được phát sinh. Đối với nhiều lớp thì hàm huỷ mặc định là đủ, và không cần đưa vào một hàm huỷ mới.

## Quy tắc viết hàm huỷ

Mỗi lớp chỉ có một hàm huỷ viết theo các quy tắc sau:

+ Kiểu của hàm: Hàm huỷ cũng giống như hàm tạo là hàm không có kiểu, không có giá trị trả về.

+ Tên hàm: Tên của hàm huỷ gồm một dẫu ngã (đứng trước) và tên lớp:

~Tên\_lớp

+ Đối: Hàm huỷ không có đối

Ví dụ có thể xây dựng hàm huỷ cho lớp DT (đa thức) ở §3 như sau:

*class DT*

*{*

*private:*

*int n; // Bac da thuc*

*double \*a; // Tro toi vung nho chua cac he so da thuc*

*// a0, a1,...*

*public:*

*~DT()*

*{*

*this->n=0; delete this->a;*

*}*

*...*

*} ;*

## Vai trò của hàm huỷ trong lớp DT

* + 1. **Khiếm khuyết của chương trình trong bài 3**

Chương trình trong §3 định nghĩa lớp DT (đa thức) khá đầy đủ gồm:

+ Các hàm tạo

+ Các hàm toán tử nhập >>, xuất <<

+ Các hàm toán tử thực hiện các phép tính + - \*

Tuy nhiên vẫn còn thiếu hàm huỷ để giải phóng vùng nhớ mà đối tượng kiểu DT (cần huỷ) đang quản lý.

Chúng ta hãy phân tích các khiếm khuyết của chương trình này:

+ Khi chương trình gọi tới một phương thức toán tử để thực hiện các phép tính cộng, trừ, nhân đa thức, thì một đối tượng trung gian được tạo ra. Một vùng nhớ được cấp phát và giao cho nó (đối tượng trung gian) quản lý.

+ Khi thực hiện xong phép tính sẽ ra khỏi phương thức. Đối tượng trung gian bị xoá, tuy nhiên chỉ vùng nhớ của các thuộc tính của đối tượng này được giải phóng. Còn vùng nhớ (chứa các hệ số của đa thức) mà đối tượng trung gian đang quản lý thì không hề bị giải phóng. Như vậy số vùng nhớ bị chiếm dụng vô ích sẽ tăng lên.

## Cách khắc phục

Nhược điểm trên dễ dàng khắc phục bằng cách đưa vào lớp DT hàm huỷ viết trong 5.3 (mục trên).

## Lớp hình tròn đồ hoạ

Chương trình dưới đây gồm:

*Lớp HT (hình tròn) với các thuộc tính:*

int r; // Bán kính

int m ; // Mầu hình tròn

int xhien,yhien; // Vị trí hiển thị hình tròn trên màn hình char \*pht; // Con trỏ trỏ tới vùng nhớ chứa ảnh hình tròn int hienmh; // Trạng thái hiện (hienmh=1), ẩn (hienmh=0)

*Các phương thức:*

+ Hàm tạo không đối HT();

thực hiện việc gán giá trị bằng 0 cho các thuộc tính của lớp.

+ Hàm tạo có đối

HT(int r1,int m1=15); thực hiện các việc:

* Gán r1 cho r, m1 cho m
* Cấp phát bộ nhớ cho pht
* Vẽ hình tròn và lưu ảnh hình tròn vào vùng nhớ của pht

+ Hàm huỷ

~HT();

thực hiện các việc:

* Xoá hình tròn khỏi màn hình (nếu đang hiển thị)
* Giải phóng bộ nhớ đã cấp cho pht

+ Phương thức

void hien(int x, int y);

có nhiệm vụ hiển thị hình tròn tại (x,y)

+ Phương thức

void an();

có nhiệm vụ làm ẩn hình tròn

*Các hàm độc lập:*

void ktdh(); //Khởi tạo đồ hoạ

void ve\_bau\_troi(); // Vẽ bầu trời đầy sao

void ht\_di\_dong\_xuong(); // Vẽ một cặp 2 hình tròn di

// chuyển xuống

void ht\_di\_dong\_len();// Vẽ một cặp 2 hình tròn di

// chuyển lên trên

Nội dung chương trình là tạo ra các chuyển động xuống và lên của các hình tròn.

//CT4\_09.CPP

// Lop do hoa

// Ham huy

// Trong ham huy co the goi PT khac

*#include <conio.h> #include <iostream.h> #include <math.h> #include <stdlib.h> #include <graphics.h> #include <dos.h>*

*void ktdh();*

*void ve\_bau\_troi();*

*void ht\_di\_dong\_xuong(); void ht\_di\_dong\_len(); int xmax,ymax;*

*class HT*

*{*

*private:*

*int r,m ;*

*int xhien,yhien; char \*pht;*

*int hienmh; public:*

*HT();*

*HT(int r1,int m1=15);*

*~HT();*

*void hien(int x, int y); void an();*

*};*

*HT:: HT()*

*{*

*r=m=hienmh=0; xhien=yhien=0; pht=NULL;*

*}*

*HT::HT(int r1,int m1)*

*{*

*r=r1; m=m1; hienmh=0; xhien=yhien=0;*

*if (r<0) r=0; if (r==0)*

*{*

*pht=NULL;*

*}*

*else*

*{*

*int size; char \*pmh;*

*size = imagesize(0,0,r+r,r+r); pmh = new char[size]; getimage(0,0,r+r,r+r,pmh); setcolor(m);*

*circle(r,r,r); setfillstyle(1,m); floodfill(r,r,m);*

*pht = new char[size]; getimage(0,0,r+r,r+r,pht); putimage(0,0,pmh,COPY\_PUT); delete pmh;*

*pmh=NULL;*

*}*

*}*

*void HT::hien(int x, int y)*

*{*

*if (pht!=NULL && !hienmh) // chua hien*

*{*

*hienmh=1; xhien=x; yhien=y;*

*putimage(x,y,pht,XOR\_PUT);*

*}*

*}*

*void HT::an()*

*{*

*if (hienmh) // dang hien*

*{*

*hienmh=0; putimage(xhien,yhien,pht,XOR\_PUT);*

*}*

*}*

*HT::~HT()*

*{*

*an();*

*if (pht!=NULL)*

*{*

*delete pht; pht=NULL;*

*}*

*}*

*void ktdh()*

*{*

*int mh=0,mode=0; initgraph(&mh,&mode,""); xmax = getmaxx();*

*ymax = getmaxy();*

*}*

*void ve\_bau\_troi()*

*{*

*for (int i=0;i<2000;++i)*

*putpixel(random(xmax), random(ymax), 1+random(15));*

*}*

*void ht\_di\_dong\_xuong()*

*{*

*HT h(50,4);*

*HT u(60,15);*

*h.hien(0,0);*

*u.hien(40,0);*

*for (int x=0;x<=340;x+=10)*

*{*

*h.an();*

*u.an();*

*h.hien(x,x); delay(200); u.hien(x+40,x); delay(200);*

*}*

*}*

*void ht\_di\_dong\_len()*

*{*

*HT h(50,4);*

*HT u(60,15);*

*h.hien(340,340);*

*u.hien(380,340);*

*for (int x=340;x>=0;x-=10)*

*{*

*h.an();*

*u.an();*

*h.hien(x,x); delay(200); u.hien(x+40,x); delay(200);*

*}*

*}*

*void main()*

*{*

*ktdh(); ve\_bau\_troi(); ht\_di\_dong\_xuong(); ht\_di\_dong\_len(); getch(); closegraph();*

*}*

## Các nhận xét:

1. Trong thân hàm huỷ gọi tới phương thức an().
2. Điều gì xẩy ra khi bỏ đi hàm huỷ:

+ Khi gọi hàm ht\_di\_dong\_xuong() thì có 2 đối tượng kiểu HT được tạo ra. Trong thân hàm sử dụng các đối tượng này để vẽ các hình tròn di chuyển xuống. Khi thoát khỏi hàm thì 2 đối tượng (tạo ra ở trên) được giải phóng. Vùng nhớ của các thuộc tính của chúng bị thu hồi, nhưng vùng nhớ cấp phát cho thuộc tính pht chưa được giải phóng và ảnh của 2 hình tròn (ở phía dưới màn hình) vẫn không được cất đi.

+ Điều tương tự xẩy ra sau khi ra khỏi hàm ht\_di\_dong\_len() : vùng nhớ cấp phát cho thuộc tính pht chưa được giải phóng và ảnh của 2 hình tròn (ở phía trên màn hình) vẫn không được thu dọn.

## Bài 6. Toán tử gán

* 1. **Toán tử gán mặc định**

Toán tử gán (cho lớp) là một trường hợp đặc biệt so với các toán tử khác. Nếu trong lớp chưa định nghĩa một phương thức toán tử gán thì Trình biên dịch sẽ phát sinh một toán tử gán mặc định để thực hiện câu lệnh gán 2 đối tượng của lớp, ví du:

HT h1, h2(100,6);

h1 = h2 ; // Gán h2 cho h1

Toán tử gán mặc định sẽ sẽ sao chép đối tượng nguồn (h2) vào đối tượng đích (h1) theo từng bit một.

Trong đa số các trường hợp khi lớp không có các thành phần con trỏ hay tham chiếu thì toán tử gán mặc định là đủ dùng và không cần định nghĩa một phương thức toán tử gán cho lớp. Nhưng đối với các lớp có thuộc tính con trỏ như lớp DT (đa thức), lớp HT (hình tròn) thì toán tử gán mặc định không thích hợp và việc xây dựng toán tử gán là cần thiết.

## Cách viết toán tử gán

Cũng giống như các phương thức khác, phương thức toán tử gán dùng đối con trỏ this để biểu thị đối tượng đích và dùng một đối tường minh để biểu thị đối tượng nguồn. Vì trong thân của toán tử gán không nên làm việc với bản sao của đối tượng nguồn, mà phải làm việc trực tiếp với đối tượng nguồn, nên kiểu đối tường minh nhất thiết phải là kiểu tham chiếu đối tượng.

Phương thức toán tử gán có thể có hoặc không có giá trị trả về. Nếu không có giá trị trả về (kiểu void), thì khi viết chương trình không được phép viết câu lệnh gán liên tiếp nhiều đối tượng, như:

u = v = k = h ;

Nếu phương thức toán tử gán trả về tham chiếu của đối tượng nguồn, thì có thể dùng toán tử gán thể thực hiện các phép gán liên tiếp nhiều đối tượng.

**Ví dụ** đối với lớp HT (trong mục trước), có thể xây dựng toán tử gán như sau:

*void HT::operator=(const HT &h)*

*{*

*r = h.r ; m = h.m ; xhien = yhien = 0; hienmh = 0 ;*

*if (h.pht==NULL) pht = NULL;*

*else*

*{*

*int size;*

*size = imagesize(0,0,r+r,r+r); pht = new char[size]; memcpy(pht,h.pht,size);*

*}*

*}*

Với toán tử gán này, chỉ cho phép gán đối tượng nguồn cho một đối tượng đích. Như vậy câu lệnh sau là sai:

HT u, v, h ; u = v = h ;

Bây giờ ta sửa lại toán gán để nó trả về tham chiếu đối tượng nguồn như sau:

*const HT & HT::operator=(const HT &h)*

*{*

*r = h.r ; m = h.m ; xhien = yhien = 0; hienmh = 0 ;*

*if (h.pht==NULL)*

*pht = NULL; else*

*{*

*int size;*

*size = imagesize(0,0,r+r,r+r); pht = new char[size]; memcpy(pht,h.pht,size);*

*}*

*return h ;*

*}*

Với toán tử gán mới này, ta có thể viết câu lệnh để gán đối tượng nguồn cho nhiều đối tượng đích. Như vậy các câu lệnh sau là được:

HT u, v, h ; u = v = h ;

## Toán tử gán và hàm tạo sao chép

+ Toán tử gán không tạo ra đối tượng mới, chỉ thực hiện phép gán giữa 2 đối tượng đã tồn tại.

+ Hàm tạo sao chép được dùng để tạo một đối tượng mới và gán nội dung của một đối tượng đã tồn tại cho đối tượng mới vừa tạo.

+ Nếu đã xây dựng toán tử gán mà lại dùng hàm tạo sao chép mặc định thì chưa đủ, vì việc khởi gán trong câu lệnh khai báo sẽ không gọi tới toán tử gán mà lại gọi tới hàm tạo sao chép.

+ Như vậy đối với lớp có thuộc tính con trỏ, thì ngoài hàm tạo, cần xây dựng thêm:

* Hàm huỷ
* Hàm tạo sao chép
* Phương thức toán tử gán

***Chú ý:*** Không phải mọi câu lệnh chứa có dấu = đều gọi đến toán tử gán. Cần phân biệt 3 trường hợp:

* + 1. Câu lệnh new (chứa dấu =) sẽ gọi đến hàm tạo, ví dụ: HT \*h= new HT(50,6); // gọi đến hàm tạo có đối
    2. Câu lệnh khai báo và khởi gán (dùng dấu =) sẽ gọi đến hàm tạo sao chép, ví dụ: HT k=\*h; // gọi đến hàm tạo sao chep
    3. Câu lệnh gán sẽ gọi đến toán tử gán, ví dụ: HT u;

u=\*h; // gọi đến phương thức toán tử gán

## Ví dụ minh hoạ

Chương trình dưới đây định nghĩa lớp HT (hình tròn) và minh hoạ:

+ Hàm tạo và hàm huỷ

+ Phương thức toán tử gán có kiểu tham chiếu

+ Hàm tạo sao chép

+ Cách dùng con trỏ this trong hàm tạo sao chép

+ Cách dùng con trỏ \_new\_handler để kiểm tra việc cấp phát bộ nhớ.

*//CT4\_10.CPP*

*// Lop do hoa*

*// Ham huy*

*// toan tu gan - tra ve tham chieu*

*// Ham tao sao chep*

*// Trong ham huy co the goi PT khac #include <conio.h>*

*#include <iostream.h> #include <stdlib.h> #include <graphics.h> #include <new.h> #include <mem.h>*

*static void kiem\_tra\_bo\_nho() ; void ktdh();*

*int xmax,ymax;*

*void kiem\_tra\_bo\_nho()*

*{*

*outtextxy(1,1,"LOI BO NHO"); getch();*

*closegraph(); exit(1);*

*}*

*class HT*

*{*

*private:*

*int r,m ;*

*int xhien,yhien; char \*pht;*

*int hienmh; public:*

*HT();*

*HT(int r1,int m1=15);*

*HT(const HT &h);*

*~HT();*

*void hien(int x, int y); void an();*

*const HT &operator=(const HT &h);*

*};*

*const HT & HT::operator=(const HT &h)*

*{*

*// outtextxy(1,1,"Gan"); getch(); r = h.r ; m = h.m ;*

*xhien = yhien = 0; hienmh = 0 ;*

*if (h.pht==NULL) pht = NULL;*

*else*

*{*

*int size;*

*size = imagesize(0,0,r+r,r+r); pht = new char[size]; memcpy(pht,h.pht,size);*

*}*

*return h;*

*}*

*HT::HT(const HT &h)*

*{*

*//outtextxy(300,1,"constructor sao chep"); getch();*

*\*this = h;*

*}*

*HT:: HT()*

*{*

*r=m=hienmh=0; xhien=yhien=0; pht=NULL;*

*}*

*HT::HT(int r1,int m1)*

*{*

*r=r1; m=m1; hienmh=0; xhien=yhien=0;*

*if (r<0) r=0; if (r==0)*

*{*

*pht=NULL;*

*}*

*else*

*{*

*int size; char \*pmh;*

*size = imagesize(0,0,r+r,r+r); pmh = new char[size];*

*getimage(0,0,r+r,r+r,pmh); setcolor(m);*

*circle(r,r,r); setfillstyle(1,m); floodfill(r,r,m);*

*pht = new char[size]; getimage(0,0,r+r,r+r,pht); putimage(0,0,pmh,COPY\_PUT); delete pmh;*

*pmh=NULL;*

*}*

*}*

*void HT::hien(int x, int y)*

*{*

*if (pht!=NULL && !hienmh) // chua hien*

*{*

*hienmh=1; xhien=x; yhien=y;*

*putimage(x,y,pht,XOR\_PUT);*

*}*

*}*

*void HT::an()*

*{*

*if (hienmh) // dang hien*

*{*

*hienmh=0; putimage(xhien,yhien,pht,XOR\_PUT);*

*}*

*}*

*HT::~HT()*

*{*

*an();*

*if (pht!=NULL)*

*{*

*delete pht; pht=NULL;*

*}*

*}*

*void ktdh()*

*{*

*int mh=0,mode=0; initgraph(&mh,&mode,""); xmax = getmaxx();*

*ymax = getmaxy();*

*}*

*void main()*

*{*

*\_new\_handler = kiem\_tra\_bo\_nho ; ktdh();*

*HT \*h= new HT(50,6); // gọi hàm tạo có đối h->hien(100,200);*

*HT k=\*h; // gọi hàm tạo sao chép k.hien(200,200);*

*HT t,v,u;*

*t = v = u = \*h; // gọi toán tử gán u.hien(300,200);*

*v.hien(400,200);*

*t.hien(500,200); getch(); closegraph();*

*}*

## Vai trò của phương thức toán tử gán

Chương trình trên sẽ vẽ 5 hình tròn trên màn hình. Điều gì sẽ xẩy ra nếu bỏ đi phương thức toán tử gán và hàm tạo sao chép?

+ Nếu bỏ cả hai, thì chỉ xuất hiên một hình tròn tại vị trí (100,200).

+ Nếu bỏ toán tử gán (giữ hàm tạo sao chép) thì chỉ xuất hiện 2 hình tròn tại các vị trí (100,200) và (200,200).

+ Nếu bỏ hàm tạo sao chép (giữ toán tử gán) thì xuất hiện 4 hình tròn.

## Bài 7. Phân loại phương thức, phương thức inline

* 1. **Phân loại các phương thức**

Có thể chia phương thức thành các nhóm:

* + 1. Các phương thức thông thường
    2. Các phương thức dùng để xây dựng và huỷ bỏ đối tượng gồm:

+ Hàm tạo không đối,

+ Hàm tạo có đối

+ Hàm tạo sao chép

+ Hàm huỷ

* + 1. Các phương thức toán tử

## Con trỏ this

Mọi phương thức đều dùng con trỏ this như đối thứ nhất (đối ẩn). Ngoài ra trong phương thức có thể đưa vào các đối tường minh được khai báo như đối của hàm.

+ Với các phương thức thông thường, thì đối ẩn biểu thị đối tượng chủ thể trong lời gọi phương thức.

+ Với các hàm tạo, thì đối ẩn biểu thị đối tượng mới được hình thành.

+ Với các hàm huỷ, thì đối ẩn biểu thị đối tượng sắp bị huỷ bỏ.

+ Với các phương thức toán tử, thì đối ẩn biểu thị toán hạng đối tượng thứ nhất.

## Phương thức inline.

Có 2 cách để biên dịch phương thức theo kiểu inline:

*Cách 1:* Xây dựng phương thức bên trong định nghĩa lớp.

*Cách 2:* Thêm từ khoá inline vào định nghĩa phương thức (bên ngoài định nghĩa lớp).

***Chú ý*** là chỉ các phương thức ngắn không chứa các câu lệnh phức tạp (như chu trình, goto, switch, đệ quy) mới có thể trơ thành inline. Nếu có ý định biên dịch theo kiểu inline các phương thức chứa các câu lệnh phức tạp nói trên, thì Trình biên dịch sẽ báo lỗi.

Trong chương trình dưới đây, tất cả các phương thức của lớp PS (phân số) đều là phương thức inline

*//CT4\_11.CPP*

*// Lop PS*

*// Inline*

*#include <conio.h> #include <iostream.h> class PS*

*{*

*private:*

*int t,m ; public:*

*PS()*

*{*

*t=0;m=1;*

*}*

*PS(int t1, int m1); void nhap();*

*void in();*

*PS operator\*=(PS p2)*

*{*

*t\*=p2.t; m\*=p2.m; return \*this;*

*}*

*};*

*inline PS::PS(int t1, int m1)*

*{*

*t=t1; m=m1;*

*}*

*inline void PS::nhap()*

*{*

*cout << "\nNhap tu va mau: " ; cin >> t >> m;*

*}*

*inline void PS::in()*

*{*

*cout << "\nPS = " << t << "/" << m ;*

*}*

*void main()*

*{*

*PS q,p,s(3,5);*

*cout << "\n Nhap PS p"; p.nhap();*

*s.in();*

*p.in();*

*q = p\*=s; p.in();*

*q.in();*

*getch();*

*}*

## Bài 8. Hàm tạo và đối tượng thành phần

* 1. **Lớp bao, lớp thành phần**

Một lớp có thuộc tính là đối tượng của lớp khác gọi là lớp bao, ví dụ:

*class A*

*{*

*private:*

*int a, b;*

*...*

*} ;*

*class B*

*{*

*private:*

*double x, y, z;*

*...*

*} ;*

*class C*

*{*

*private:*

*int m, n;*

*A u;*

*B p, q;*

*...*

*} ;*

Trong ví dụ trên thì:

C là lớp bao

A, B là các lớp thành phần (của C)

## Hàm tạo của lớp bao

+ Chú ý là trong các phương thức của lớp bao không cho phép truy nhập trực tiếp đến các thuộc tính của các đối tượng của các lớp thành phần.

+ Vì vậy, khi xây dựng hàm tạo của lớp bao, phải sư dụng các hàm tạo của lớp thành phần để khởi gán cho các đối tượng thành phần của lớp bao.

**Ví dụ** khi xây dựng hàm tạo của lớp C, cần dùng các hàm tạo của lớp A để khởi gán cho đối tượng thành phần u và dùng các hàm tạo của lớp B để khởi gán cho các đối tượng thành phần p, q.

## Cách dùng hàm tạo của lớp thành phần để xây dựng hàm tạo của lớp bao

+ Để dùng hàm tạo (của lớp thành phần) khởi gán cho đối tưọng thành phần của lớp bao, ta sử dụng mẫu:

tên\_đối\_tượng(danh sách giá trị)

+ Các mẫu trên cần viết bên ngoài thân hàm tạo, ngay sau dòng đầu tiên. Nói một cách cụ thể hơn, hàm tạo sẽ có dạng:

tên\_lớp(danh sách đối) : tên\_đối\_tượng( danh sách giá trị),

...

tên\_đối\_tượng( danh sách giá trị)

{

// Các câu lệnh trong thân hàm tạo

}

***Chú ý*** là các dấu ngoặc sau tên đối tượng luôn luôn phải có, ngay cả khi danh sách giá trị bên trong là rỗng.

+ Danh sách giá trị lấy từ danh sách đối. Dựa vào danh sách giá trị, Trình biên dịch sẽ biết cần dùng hàm tạo nào để khởi gán cho đối tượng. Nếu danh sách giá trị là rỗng thì hàm tạo không đối sẽ được sử dụng.

+ Các đối tượng muốn khởi gán bằng hàm tạo không đối có thể bỏ qua, không cần phải liệt kê trong hàm tạo. Nói cách khác: Các đối tượng không được liệt kê trên dòng đầu hàm tạo của lớp bao, đều được khởi gán bằng hàm tạo không đối của lớp thành phần.

## Ví dụ:

*class A*

*{*

*private:*

*int a, b; public:*

*A()*

*{*

*a=b=0;*

*}*

*A(int a1, int b1)*

*{*

*} ;*

*class B*

*{*

*}*

*...*

*a = a1; b = b1;*

*private:*

*double x, y, z; public:*

*B()*

*{*

*x = y = z = 0.0 ;*

*}*

*B(double x1, double y1)*

*{*

*x = x1; y = y1; z = 0.0 ;*

*}*

*B(double x1, double y1, double z1)*

*{*

*} ;*

*class C*

*}*

*...*

*x = x1; y = y1; z = z1 ;*

*{*

*private:*

*int m, n;*

*A u, v;*

*B p, q, r; public:*

*C(int m1, int n1,int a1, int b1, double x1, double y1, double x2, double y2, double z2) : u(), v(a1,b1), q(x1,y1), r(x2,y2,z2)*

*{*

*m = m1 ; n = n1;*

*}*

*} ;*

Trong hàm tạo nói trên của lớp C, thì các đối tượng thành phần được khởi gán như sau: u được khởi gán bằng hàm tạo không đối của lớp A

v được khởi gán bằng hàm tạo 2 đối của lớp A q được khởi gán bằng hàm tạo 2 đối của lớp B r được khởi gán bằng hàm tạo 3 đối của lớp B

p (không có mặt) được khởi gán bằng hàm tạo không đối của lớp B

## Sử dụng các phương thức của lớp thành phần

Mặc dù lớp bao có các thành phần đối tượng, nhưng trong lớp bao lại không được phép truy nhập đến các thuộc tính của các đối tượng này. Vì vậy giải pháp thông thường là:

+ Trong các lớp thành phần, xây dựng sẵn các phương thức để có thể lấy ra các thuộc tính của lớp.

+ Trong lớp bao dùng các phương thức của lớp thành phần để nhận các thuộc tính của các đối tượng thành viên cần dùng đến.

## Các ví dụ

Hai chương trình dưới đây minh hoạ các điều đã nói trong các mục trên.

## Ví dụ 1:

Trong ví dụ này xét 2 lớp:

DIEM (Điểm) và DT (Đoạn thẳng) Lớp DIEM là lớp thành phần của lớp DT

*//CT4\_12.CPP*

*// Thuoc tinh doi tuong #include <conio.h> #include <iostream.h> class DIEM*

*{*

*private:*

*int x,y ; public:*

*DIEM()*

*{*

*x=y=0;*

*}*

*DIEM(int x1, int y1)*

*{*

*x= x1; y=y1;*

*}*

*void in()*

*{*

*cout << "(" << x << "," << y << ")" ;*

*}*

*} ;*

*class DT*

*{*

*private:*

*DIEM d1, d2;*

*int m; public:*

*DT() : d1(), d2()*

*{*

*m=0;*

*}*

*DT(int m1,int x1, int y1, int x2, int y2) : d1(x1,y1), d2(x2,y2)*

*{*

*m=m1;*

*}*

*DT(int m1,DIEM t1, DIEM t2)*

*{*

*m=m1; d1 = t1; d2 = t2;*

*}*

*void in()*

*{*

*cout << "\n Diem dau : "; d1.in(); cout << "\n Diem cuoi: "; d2.in(); cout << "\n Mau : " << m;*

*}*

*};*

*void main()*

*{*

*DT u, v(1,100,100,200,200), s(2,DIEM(300,300), DIEM(400,400)) ;*

*clrscr();*

*u.in();*

*v.in();*

*s.in();*

*getch();*

*}*

## Ví dụ 2:

Trong ví dụ này xét 3 lớp: Diem (Điểm)

DTron (Đường tròn) HTron (Hình tròn)

Lớp DTron có một lớp thành phần là lớp Diem.

Lớp HTron có 2 lớp thành phần là lớp DTron và lớp Diem. Trong lớp DTron đưa vào phương thức vẽ đường tròn.

Trong lớp HTron đưa vào phương thức vẽ và tô mầu hình tròn.

Khi xây dựng phương thức của lớp bao cần sử dụng các phương thức của lớp thành phần.

*//CT4\_13.CPP*

*// Thuoc tinh doi tuong #include <conio.h> #include <iostream.h> #include <graphics.h> class Diem*

*{*

*private:*

*int x,y ; public:*

*Diem()*

*{*

*x=y=0;*

*}*

*Diem(int x1, int y1)*

*{*

*x= x1; y=y1;*

*}*

*int getx()*

*{*

*return x;*

*}*

*int gety()*

*{*

*return y;*

*}*

*} ;*

*class DTron // Duong tron*

*{*

*private:*

*Diem t ; // tam int r ;*

*int m; public:*

*DTron()*

*{*

*r=m=0;*

*}*

*DTron(int x1,int y1,int r1,int m1): t(x1,y1)*

*{*

*m=m1; r=r1;*

*}*

*int mau()*

*{*

*return m;*

*}*

*void ve()*

*{*

*setcolor(m); circle(t.getx(),t.gety(),r);*

*}*

*};*

*class HTron*

*{*

*private:*

*DTron dt; Diem d; int m;*

*public:*

*HTron()*

*{*

*m=0;*

*}*

*HTron(int x1, int y1, int r1, int m1,*

*int x, int y, int mt): dt(x1,y1,r1,m1), d(x,y)*

*{*

*m = mt;*

*}*

*void ve()*

*{*

*dt.ve(); setfillstyle(1,m);*

*floodfill(d.getx(),d.gety(),dt.mau());*

*}*

*} ;*

*void main()*

*{*

*int mh=0, mode=0; initgraph(&mh,&mode,""); setbkcolor(1);*

*DTron dt(100,100,80,6);*

*HTron ht(300,300,150,15,300,300,4);*

*dt.ve();*

*ht.ve();*

*getch(); closegraph();*

*}*

## Bài 9. Các thành phần tĩnh

* 1. **Thành phần dữ liệu tĩnh**

+ Thành phần dữ liệu được khai báo bằng từ khoá static gọi là tĩnh, ví dụ:

*class A*

*{*

*private:*

*static int ts ; // Thành phần tĩnh int x;*

*....*

*} ;*

+ Thành phần tĩnh được cấp phát một vùng nhớ cố định. Nó tồn tại ngay cả khi lớp chưa có một đối tượng nào cả.

+ Thành phần tĩnh là chung cho cả lớp, nó không phải là riêng của mỗi đối tượng. Ví dụ xét 2 đối tượng:

A u,v ; // Khai báo 2 đối tượng

thì giữa các thành phần x và ts có sự khác nhau như sau:

u.x và v.x có 2 vùng nhớ khác nhau

u.ts và v.ts chỉ là một, chúng cùng biểu thị một vùng nhớ thành phần ts tồn tại ngay khi u và v chưa khai báo

+ Để biểu thị thành phần tĩnh, ta có thể dùng tên lớp, ví du: Đối với ts thì 3 cách viết sau là tương đương:

A::ts u.ts v.ts

+ Khai báo và khởi gán giá trị cho thành phần tĩnh

Thành phần tĩnh sẽ được cấp phát bộ nhớ và khởi gán giá trị ban đầu bằng một câu lệnh khai báo đặt sau định nghĩa lớp (bên ngoài các hàm, kể cả hàm main), theo các mẫu:

int A::ts ; // Khởi gán cho ts giá trị 0

int A::ts = 1234; // Khởi gán cho ts giá trị 1234

***Chú ý:*** Khi chưa khai báo thì thành phần tĩnh chưa tồn tại. Ví dụ xét chương trình sau:

*#include <conio.h> #include <iostream.h>*

*class HDBH // Hoá đơn bán hàng*

*{*

*private:*

*char \*tenhang ; // Tên hàng double tienban ; // Tiền bán*

*static int tshd ; // Tổng số hoá đơn*

*static double tstienban ; // Tổng số tiền bán public:*

*static void in()*

*{*

*cout <<"\n" << tshd;*

*cout <<"\n" << tstienban;*

*}*

*} ;*

*void main()*

*{*

*HDBH::in();*

*getch();*

*}*

Các thành phần tĩnh tshd và tstienban chưa khai báo, nên chưa tồn tại. Vì vậy các câu lệnh in giá trị các thành phần này trong phương thức in là không logic. Khi dịch chương trình, sẽ nhận được các thông báo lỗi (tại phương thức in) như sau:

Undefined symbol HDBH::tshd in module ... Undefined symbol HDBH::tstienban in module ...

Có thể sửa chương trình trên bằng cách đưa vào các lệnh khai báo các thành phần tĩnh tshd và tstienban như sau:

*//CT4\_14.CPP*

*// thanh phan tinh*

*// Lop HDBH (hoa don ban hang) #include <conio.h>*

*#include <iostream.h> class HDBH*

*{*

*private:*

*int shd ;*

*char \*tenhang ; double tienban ; static int tshd ;*

*static double tstienban ; public:*

*static void in()*

*{*

*cout <<"\n" << tshd;*

*cout <<"\n" << tstienban;*

*}*

*} ;*

*int HDBH::tshd=5;*

*double HDBH::tstienban=20000.0; void main()*

*{*

*HDBH::in();*

*getch();*

*}*

## Khi nào cần sử dụng các thành phần dữ liệu tĩnh

Xét một ví dụ về quản lý các hoá đơn bán hàng. Mỗi hoá đơn có: Tên hàng, số tiền bán. Rõ ràng các thuộc tính nói trên là riêng của mỗi hoá đơn. Mặt khác nếu chúng ta quan tâm đến tổng số hoá đơn đã bán, tổng số tiền đã bán, thì các thông tin này là chung. Vì vậy khi thiết kế lớp HDBH (hoá đơn bán hàng) , thì ta có thể đưa vào 4 thành phần dữ liệu là:

tenhang (tên hàng) tienban (tiền bán)

tshd (tổng số hoá đơn) tstienban (tổng số tiền bán)

Các thuộc tính tenhang và tienban là riêng của mỗi hoá đơn, nên chúng được chọn là các thuộc tính thông thường. Còn các thuộc tính tshd và tstienban là chung cho cả lớp nên chúng được chọn là các thuộc tính tĩnh.

## Phương thức tĩnh

+ Có 2 cách viết phương thức tĩnh:

*Cách 1:* Dùng từ khoá static đặt trước định nghĩa phương thức viết bên trong định nghĩa lớp (như phương thưc in() ví dụ cuối của mục 9.1).

*Cách 2:* Nếu phương thức xây dựng bên ngoài định nghĩa lớp, thì dùng từ khoá static đặt trước khai báo phương thức bên trong định nghĩa lớp. Chú ý không cho phép dùng từ khoá static đặt trước định nghĩa phương thức viết bên ngoài định nghĩa lóp.

+ Phương thức tĩnh là chung cho cả lớp, nó không lệ thuộc vào một đối tượng cụ thể, nó tồn tại ngay khi lớp chưa có đối tượng nào (xem ví dụ trong mục 9.1).

+ Lời gọi phương thức tĩnh:

Có thể xuất phát từ một đối tượng nào đó (như vẫn dùng khi gọi các phương thức khác) Có thể dùng tên lớp

Ví dụ xét lớp HDBH trong mục 9.1 và xét các câu lênh: HDBH u, v;

Khi đó để gọi phương thức tĩnh in() có thể dùng một trong các lệnh sau: u.in();

v.in();

HDBH::in();

+ Vì phương thức tĩnh là độc lập với các đối tượng, nên không thể dùng phương thức tĩnh để xử lý dữ liệu của các đối tượng chủ thể trong lời gọi phương thức tĩnh. Nói cách khác không cho phép truy nhập tới các thuộc tính (trư thuộc tính tĩnh) trong thân phương thức tĩnh. Điều đó cũng đồng nghĩa với việc không cho phép dùng con trỏ this trong phương thức tĩnh.

Ví dụ nếu lập phương thức tĩnh in() để in các thuộc tính của lớp HDBH như sau:

*class HDBH*

*{*

*private:*

*int shd ;*

*char \*tenhang ; double tienban ; static int tshd ;*

*static double tstienban ; public:*

*static void in()*

*{*

*cout <<"\n" << tshd;*

*cout <<"\n" << tstienban; cout <<"\n" << tenhang; cout <<"\n" << tienban;*

*}*

*} ;*

thì sẽ bị lỗi, vì trong thân phương thức tĩnh không cho phép truy nhập đến các thuộc tính tenhang và tienban.

## Ví dụ minh hoạ việc dùng phương thức tĩnh

Xét bài toán quản lý hoá đơn bán hàng. Mỗi hoá đơn có 2 dữ liêu là tên hàng và tiền bán. Sử dụng hàm tạo để tạo ra các hoá đơn, dùng hàm huỷ để bỏ đi (loại đi) các hoá đơn không cần lưu trữ, dùng một phương thức để sửa chữa nội dung hoá đơn (nhập lại tiền bán). Vấn đề đặt ra là sau một số thao tác: Tạo, sửa và huỷ hoá đơn thì tổng số hoá đơn còn lại là bao nhiêu và tổng số tiền trên các hoá đơn còn lại là bao nhiêu?

Chương trình dưới đây nhằm đáp ứng yêu cầu đặt ra.

*//CT4\_14.CPP*

*// thanh phan tinh*

*// Lop HDBH (hoa don ban hang) #include <conio.h>*

*#include <iostream.h> class HDBH*

*{*

*private:*

*char \*tenhang ; double tienban ; static int tshd ;*

*static double tstienban ; public:*

*HDBH(char \*tenhang1=NULL,double tienban1=0.0 )*

*{*

*tienban=tienban1; tenhang=tenhang1;*

*++tshd;*

*tstienban += tienban;*

*}*

*~HDBH()*

*{*

*--tshd;*

*tstienban -= tienban;*

*}*

*void sua(); static void in();*

*} ;*

*int HDBH::tshd=0;*

*double HDBH::tstienban=0; void HDBH::in()*

*{*

*cout <<"\n\nTong so hoa don: " << tshd; cout <<"\nTong so tien: " << tstienban;*

*}*

*void HDBH::sua()*

*{*

*cout << "\n\nTen hang: " << tenhang; cout << "\nTien ban : " << tienban; tstienban -= tienban;*

*cout << "\nSua tien ban thanh : " ; cin >> tienban;*

*tstienban += tienban;*

*}*

*void main()*

*{*

*HDBH \*h1 = new HDBH("Xi mang",2000); HDBH \*h2 = new HDBH("Sat thep",3000); HDBH \*h3 = new HDBH("Ti vi",4000);*

*clrscr(); HDBH::in();*

*getch(); delete h1; HDBH::in();*

*getch();*

*h2->sua();*

*HDBH::in();*

*getch(); delete h3; HDBH::in();*

*getch();*

*}*

## Bài 10. Mảng đối tượng

* 1. **Khai báo**

Có thể dùng tên lớp để khai báo mảng đối tượng (giống như khai báo mảng int, float, char, ...) theo mẫu:

Tên\_lớp tên\_mảng[kích\_cỡ] ;

**Ví dụ** giả sử đã định nghĩa lớp DIEM (Điểm), khi đó có thể khai báo các mảng đối tượng DIEM như sau:

DIEM a[10], b[20] ;

ý nghĩa: a là mảng kiểu DIEM gồm 10 phần tử

b là mảng kiểu DIEM gồm 20 phần tử

Câu lệnh khai báo mảng sẽ gọi tới hàm tạo không đối để tạo các phần tử mảng. Trong ví dụ trên, hàm tạo được gọi 30 lần để tạo 30 phần tử mảng đối tượng.

## Khai báo và khởi gán

Để khai báo mảng và khởi gán giá trị cho các phần tử mảng đối tượng, cần dùng các hàm tạo có đối theo mẫu sau:

Tên\_lớp tên\_mảng[kích\_cớ] = { Tên\_lớp(các tham số), ...,

Tên\_lớp(các tham số) } ;

**Ví dụ** giả sử lớp DIEM đã định nghĩa:

*class DIEM*

*{*

*private:*

*int x, y ; public:*

*DIEM()*

*{*

*x=y=0;*

*}*

*DIEM(int x1, int y1)*

*{*

*x=x1; y=y1;*

*}*

*void nhapsl();*

*void ve\_doan\_thang(DIEM d2, int mau) ;*

*};*

Khi đó các câu lệnh khai báo dưới đây đều đúng: DIEM d[5] = {DIEM(1,1),DIEM(200,200)};

DIEM u[] = {DIEM(1,1),DIEM(200,200)};

ý nghĩa của các lệnh này như sau:

Câu lệnh đầu gọi tới hàm tạo 2 lần để khởi gán cho d[1], d[2] và gọi tới hàm tạo không đối 3 lần để tạo các phần tử d[3], d[4] và d[5].

Câu lệnh sau gọi tới hàm tạo 2 lần để khởi gán cho u[1], u[2]. Mảng u sẽ gồm 2 phần tử.

## Biểu thị thành phần của phần tử mảng

Để biểu thị thuộc tính của phần tử mảng đối tượng, ta viết như sau: Tên\_mảng[chỉ số] . Tên\_thuộc\_tính

Để thực hiện phương thức đối với phần tử mảng ta viết như sau: Tên\_mảng[chỉ số] . Tên\_phương\_thức(danh sách tham số) ;

**Ví dụ** để vẽ đoạn thẳng nối điểm d[1] với d[2] theo mầu đỏ, ta có thể dùng phương thức ve\_doan\_thang như sau:

d[1].ve\_doan\_thang(d[2], 4);// Thực hiện phương thức đối với d[1]

## Ví dụ

Chương trình dưới đây đưa vào lớp TS (thí sinh) và xét bài toán: Nhập một danh sách thí sinh, sắp xếp danh sách theo thứ tự giảm của tổng điểm. Chương trình minh hoạ:

+ Cách dùng mảng đối tượng.

+ Vai trò con trỏ this (trong phương thức hv(hoán vị)) .

+ Các hàm tạo, hàm huỷ.

+ Vai trò của toán tử gán (nếu sử dụng phép gán mặc định chương trình sẽ cho kết quả sai).

*//CT4\_15.CPP*

*// mang doi tuong*

*// Lop TS (thi sinh)*

*// Chu y vai tro cua toan tu gan #include <conio.h>*

*#include <iostream.h> #include <string.h> class TS*

*{*

*private:*

*char \*ht; double td;*

*public:*

*TS()*

*{*

*ht = new char[20]; td = 0;*

*}*

*~TS()*

*{*

*delete ht;*

*}*

*const TS &operator=(const TS &ts2)*

*{*

*this->td = ts2.td; strcpy(this->ht,ts2.ht); return ts2;*

*}*

*void nhap(int i); void in(); double gettd()*

*{*

*return td;*

*}*

*void hv(TS &ts2)*

*{*

*TS tg;*

*tg = \*this ;*

*\*this = ts2 ; ts2 = tg;*

*}*

*} ;*

*void TS::in()*

*{*

*cout << "\nHo ten: " << ht << " Tong diem: " << td;*

*}*

*void TS::nhap(int i)*

*{*

*cout << "\nNhap thi sinh " << i ; cout << "\nHo ten: " ; cin >> ht; cout << "Tong diem: " ; cin >> td;*

*}*

*void main()*

*{*

*TS ts[100];*

*int n, i, j;*

*clrscr();*

*cout << "\n So thi sinh: " ; cin >> n;*

*for (i=1; i<= n; ++i) ts[i].nhap(i);*

*cout <<"\n Danh sach nhap vao:"; for (i=1; i<= n; ++i)*

*ts[i].in();*

*for (i=1; i<n ; ++i)*

*for (j=i+1 ; j<=n; ++j)*

*if (ts[i].gettd() < ts[j].gettd()) ts[i].hv(ts[j]);*

*cout <<"\n\n Danh sach sau khi sap xep:"; for (i=1; i<= n; ++i)*

*ts[i].in(); getch();*

*}*

## Bài 11. cấp phát bộ nhớ cho đối tượng

* 1. **Cách cấp phát bộ nhớ cho đối tượng**

Có thể dùng new và tên lớp để cấp phát một vùng nhớ cho một hoặc một dẫy các đối tượng. Bộ nhớ cấp phát được quản lý bởi một con trỏ kiểu đối tượng. Ví dụ sau khi đã định nghĩa lớp DIEM như trong mục trên, ta có thể thực hiện các lệnh cấp phát bộ nhớ như sau:

int n = 10;

DIEM \*p, \*q, \*r ;

p = new DIEM ; // Cấp phát bộ nhớ cho một đối tượng q = new DIEM[n] ; //Cấp phát bộ nhớ cho n đối tượng

r = new DIEM(200,100); // Cấp phát bộ nhớ và khởi gán cho

// một đối tượng

## Làm việc với đối tượng thông qua con trỏ

+ Giả sử con trỏ p trỏ tới vùng nhớ của một đối tượng nào đó. Khi đó:

* + - Để biểu thị một thành phần (thuộc tính hoặc phương thức) của đối tượng, ta dùng mẫu viết sau:

p -> tên\_thành\_phần

* + - Để biểu thị đối tượng, ta dùng mẫu viết sau:

\*p

+ Giả sử con trỏ q trỏ tới địa chỉ đầu vùng nhớ của một dẫy đối tượng. Khi đó:

* + - Để biểu thị một thành phần (thuộc tính hoặc phương thức) của đối tượng thứ i, ta dùng một trong các mẫu viết sau:

q[i].tên\_thành\_phần (q+i)-> tên\_thành\_phần

* + - Để biểu thị đối tượng thứ i, ta dùng một trong các mẫu viết sau: q[i]

\*(q+i)

## Bài toán sắp xếp thí sinh

Trong mục 10.4. đã sử dụng mảng đối tượng để giải quyết bài toán: Nhập một danh sách thí sinh, sắp xếp danh sách theo thứ tự giảm của tổng điểm. Dưới đây sẽ đưa ra phương án mới bằng cách dùng con trỏ và cấp phát bộ nhớ cho các đối tượng. Chương trình chỉ thay đổi hàm main() như sau:

*void main()*

*{*

*TS \*ts; int n, i, j;*

*clrscr();*

*cout << "\n So thi sinh: " ; cin >> n;*

*ts = new TS[n+1]; for (i=1; i<= n; ++i)*

*ts[i].nhap(i);*

*cout <<"\n Danh sach nhap vao:"; for (i=1; i<= n; ++i)*

*ts[i].in();*

*for (i=1; i<n ; ++i)*

*for (j=i+1 ; j<=n; ++j)*

*if (ts[i].gettd() < ts[j].gettd()) ts[i].hv(ts[j]);*

*cout <<"\n\n Danh sach sau khi sap xep:"; for (i=1; i<= n; ++i)*

*ts[i].in(); getch();*

*}*

**Nhận xét:** Sự khác biệt giữa hàm main mới và hàm main trong 10.4 là rất ít.

## Danh sách móc nối

Chương trình dưới đây định nghĩa lớp tự trỏ TS (lớp có thuộc tính kiểu \*TS). Lớp này được dùng để tổ chức danh sách móc nối. Chương trình nhập một danh sách thí sinh và chứa trong một danh sách móc nối. Sau đó duyệt trên danh sách này để in các thí sinh trúng tuyển. So với lớp TS nêu trong mục 10.4, lớp TS ở đây có một số điểm khác như sau:

+ Thêm thuộc tính:

TS \*dc; // Dùng để chứa địa chỉ của một đối tượng kiểu TS

+ Thêm các phương thức:

void setdc(TS \*dc1) ; // Gán dc1 cho thuộc tính dc TS \*getdc() ; // Nhận giá trị của dc

+ Phương thức nhap trong chương trình trước có kiểu void nay sửa là: int nhap(int i);

Phương thức trả về 1 nếu họ tên nhập vào khác trống, trả về 0 nếu trái lại.

+ Bỏ đi các phương thức không dùng đến như: Toán tử gán, hoán vị.

*//CT4\_16.CPP*

*// Danh sách móc nối*

*// Lop TS (thi sinh) #include <conio.h> #include <iostream.h> #include <string.h> #include <ctype.h>*

*#include <stdio.h> class TS*

*{*

*private:*

*char \*ht; double td; TS \*dc;*

*public:*

*TS()*

*{*

*ht = new char[20]; td = 0;*

*dc=NULL;*

*}*

*~TS()*

*{*

*delete ht; dc=NULL ;*

*}*

*int nhap(int i); void in(); double gettd()*

*{*

*return td;*

*}*

*void setdc(TS \*dc1)*

*{*

*dc=dc1;*

*}*

*TS \*getdc()*

*{*

*return dc;*

*}*

*} ;*

*void TS::in()*

*{*

*cout << "\nHo ten: " << ht << " Tong diem: " << td;*

*}*

*int TS::nhap(int i)*

*{*

*cout << "\nNhap thi sinh " << i ;*

*cout << "\nHo ten (Bấm Enter để kết thúc nhập): " ; fflush(stdin);*

*gets(ht);*

*if (ht[0]==0) return 0;*

*cout << "Tong diem: " ; cin >> td; dc=NULL;*

*return 1;*

*}*

*void main()*

*{*

*int i=0;*

*TS \*pdau,\*p,\*q; pdau=NULL; clrscr(); while(1)*

*{*

*q=new TS;*

*++i;*

*if (q->nhap(i)==0)*

*{*

*delete q; break;*

*}*

*if (pdau==NULL) pdau = p = q;*

*else*

*{*

*p->setdc(q) ; p = q;*

*}*

*}*

*/\* In \*/*

*double diemchuan;*

*cout << "\nDiem chuan: " ; cin >> diemchuan;*

*cout <<"\nDanh sach trung tuyen:" ; p=pdau;*

*while (p!=NULL)*

*{*

*if (p->gettd()>=diemchuan) p->in();*

*p = p->getdc();*

*}*

*getch();*

*}*

## Bài 12. Đối tượng hằng, phương thức hằng

+ Cũng giống như các phần tử dữ liệu khác, một đối tượng có thể được khai báo là hằng bằng cách dùng từ khoá const. Ví dụ:

*class DIEM*

*{*

*private:*

*int x, y;*

*int m; public:*

*DIEM()*

*{*

*x = y = m = 0;*

*}*

*DIEM(int x1, int y1, int m1=15)*

*{*

*x= x1; y= y1; m= m1;*

*}*

*...*

*} ;*

const DIEM d = DIEM(200,100); // Khai báo đối tượng hằng

+ Khi khai báo cần sử dụng các hàm tạo để khởi gán giá trị cho đối tượng hằng. Giá trị khởi tạo có thể là các hằng, các biến, các biểu thức và các hàm, ví dụ:

int x0=100, y0=50; m0 =4;

const DIEM d5 = DIEM(x0 + getmaxx()/2, y0 + getmaxy()/2, m0);

+ Các phương thức có thể sử dụng cho các đối tượng hằng là hàm tạo và hàm huỷ. Về lý thuyết các đối tượng hằng không thể bị thay đổi, mà chỉ được tạo ra hoặc huỷ bỏ đi.

Khi dùng một phương thức cho đối tượng hằng, thì CTBD (Chương trình biên dich) sẽ cảnh báo (warning):

Non-const function called for const object

Tuy nhiên, chương trình EXE vẫn được tạo và khi thực hiện chương trình, thì nội dung các đối tượng hằng vẫn bị thay đổi. Chương trình dưới đây sẽ minh hoạ điều này. Chương trình đưa vào lớp PS (phân số). Phương thức toán tử ++ vẫn có thể làm thay đổi đối tượng hằng (mặc dù khi biên dịch có 3 cảnh báo).

*//CT4\_19.CPP*

*// doi tuong const*

*// Lop PS (phan so) #include <conio.h> #include <iostream.h> #include <string.h> #include <math.h> class PS*

*{*

*private:*

*int t,m; public:*

*PS()*

*{*

*t = m = 0;*

*}*

*PS(int t1, int m1)*

*{*

*t = t1; m = m1;*

*}*

*PS operator++()*

*{*

*t += m ; return \*this ;*

*}*

*void in()*

*{*

*cout << "\nPS= " << t << "/" << m;*

*}*

*void nhap()*

*{*

*cout << "\n Nhap tu va mau: " ; cin >> t >> m;*

*}*

*} ;*

*void main()*

*{*

*int t1=-3, m1=5;*

*const PS p = PS(abs(t1)+2,m1+2); // Khai báo đối tượng hằng clrscr();*

*p.in();*

*++p;*

*p.in();*

*getch();*

*}*

## + Phương thức const

Để biến một phương thức thành const ta chỉ việc viết thêm từ khoá const vào sau dòng đầu của phương thức.

***Chú ý:*** Nếu phương thức được khai báo bên trong và định nghĩa bên ngoài lớp, thì từ khoá const cần được bổ sung cả trong khai báo và định nghĩa phương thức.

Trong thân phương thức const không cho phép làm thay đổi các thuộc tính của lớp. Vị vậy việc dùng phương thức const cho các đối tượng hằng sẽ đảm bảo giữ nguyên nội dung của các đối tượng hằng.

Đương nhiên các phương thức const vẫn dùng được cho các đối tượng khác. Ví dụ sau về lớp PS (phân số) minh hoạ việc dùng phương thức const.

*// Đối tượng const*

*// Phương thức const*

*// Lop PS (phan so) #include <conio.h> #include <iostream.h> #include <string.h> #include <math.h> class PS*

*{*

*private:*

*int t,m; public:*

*PS()*

*{*

*t = m = 0;*

*}*

*PS(int t1, int m1)*

*{*

*t = t1; m = m1;*

*}*

*PS operator++()*

*{*

*t += m ; return \*this ;*

*}*

*void in() const ; void nhap()*

*{*

*cout << "\n Nhap tu va mau: " ; cin >> t >> m;*

*}*

*} ;*

*void PS::in() const*

*{*

*cout << "\nPS= " << t << "/" << m;*

*}*

*void main()*

*{*

*int t1=-3, m1=5;*

*const PS p = PS(abs(t1)+2,m1+2); PS q;*

*clrscr();*

*q.nhap();*

*p.in();*

*q.in();*

*getch();*

*}*

## Bài 13. Hàm bạn, lớp bạn

* 1. **Hàm bạn** (xem mục bài 6, chương 3) của một lớp, tuy không phải là phương thức của lớp, nhưng có thể truy nhập đến các thành phần riêng (private) của lớp. Một hàm có thể là bạn của nhiều lớp.
  2. **Nếu lớp A được khai báo là bạn của lớp B** thì tất cả các phương thức của A đều có thể truy nhập đến các thành phần riêng của lớp B. Một lớp có thể là bạn của nhiều lớp khác. Cũng có thể khai báo A là bạn của B và B là bạn của A.

## Cách khai báo lớp bạn

Giả sử có 3 lớp A, B và C. Để khai báo lớp này là bạn của lớp kia, ta viết theo mẫu sau:

// Khai báo trước các lớp class A;

class B ; class C;

// Định nghĩa các lớp class A

{

...

friend class B ; // Lớp B là bạn của A friend class C ; // Lớp C là bạn của A

...

};

class B

{

...

friend class A ; // Lớp A là bạn của B friend class C ; // Lớp C là bạn của B

...

};

class C

{

...

friend class B ; // Lớp B là bạn của C

...

};

## Ví dụ

Chương trình dưới đây có 2 lớp:

MT (ma trận vuông) VT (véc tơ)

Lớp MT là bạn của VT và lớp VT là bạn của MT. Trong chương trình sử dụng các phương thức trùng tên:

2 phương thức nhap(): nhập ma trận

nhập véc tơ

2 phương thức in(): in ma trận

in véc tơ

4 phương thức tich():

tích ma trận với ma trận, kết quả là ma trận tích ma trận với véc tơ, kết quả là véc tơ tích véc tơ với ma trận, kết quả là véc tơ tích véc tơ với véc tơ, kết quả là số thực

Nội dung chương trình là:

+ Nhập các ma trận A, B, C

+ Nhập các véc tơ

+ Tính tích D = AB

+ Tính tích u = Dy

+ Tính tích v = xC

+ Tính tích s = vu

*//CT4\_17.CPP*

*// Lop ban*

*// Lop MT , lop VT #include <conio.h> #include <iostream.h> class MT;*

*class VT; class MT*

*{*

*private:*

*double a[10][10]; int n;*

*public:*

*friend class VT; MT()*

*{*

*n=0;*

*}*

*void nhap(); void in();*

*VT tich(const VT &y); MT tich(const MT &b) ;*

*} ;*

*class VT*

*{*

*private:*

*double x[10]; int n;*

*public:*

*friend class MT; VT()*

*{*

*n=0;*

*}*

*void nhap(); void in();*

*VT tich(const MT &b); double tich(const VT &y) ;*

*} ;*

*void MT::nhap()*

*{*

*cout << "\n Cap ma tran: " ; cin >> n;*

*for (int i=1; i<=n; ++i) for (int j=1; j<=n; ++j)*

*{*

*cout << "\nPhan tu hang " << i << " cot " << j << " = " ; cin >> a[i][j];*

*}*

*}*

*void MT::in()*

*{*

*for (int i=1; i<=n; ++i)*

*{*

*cout << "\n" ;*

*for (int j=1; j<=n; ++j) cout << a[i][j] << " " ;*

*}*

*}*

*void VT::nhap()*

*{*

*cout << "\n Cap vec to: " ; cin >> n;*

*for (int i=1; i<=n; ++i)*

*{*

*cout << "\nPhan tu thu " << i << " = " ; cin >> x[i];*

*}*

*}*

*void VT::in()*

*{*

*for (int i=1; i<=n; ++i) cout << x[i] << " " ;*

*}*

*VT MT::tich(const VT &y)*

*{*

*VT z;*

*int i,j;*

*for (i=1; i<=n; ++i)*

*{*

*z.x[i] = 0.0 ;*

*for (j=1; j<=n; ++j) z.x[i] += a[i][j]\*y.x[j];*

*}*

*z.n = n; return z;*

*}*

*MT MT::tich(const MT &b)*

*{*

*MT c;*

*int i,j,k;*

*for (i=1; i<=n; ++i) for (j=1; j<=n; ++j)*

*{*

*c.a[i][j] = 0.0 ;*

*for (k=1; k<=n; ++k)*

*c.a[i][j] += a[i][k]\*b.a[k][j];*

*}*

*c.n = n; return c;*

*}*

*VT VT::tich(const MT &b)*

*{*

*VT z;*

*int i,j;*

*for (j=1; j<=n; ++j)*

*{*

*z.x[j] = 0.0 ;*

*for (i=1; i<=n; ++i) z.x[j] += b.a[i][j]\*x[i];*

*}*

*z.n = n; return z;*

*}*

*double VT::tich(const VT &y)*

*{*

*double tg=0.0;*

*for (int i=1; i<=n; ++i) tg += x[i]\*y.x[i];*

*return tg;*

*}*

*void main()*

*{*

*MT a,b,c; VT x,y; clrscr();*

*cout << "\nMa tran A"; a.nhap();*

*cout << "\nMa tran B"; b.nhap();*

*cout << "\nMa tran C"; c.nhap();*

*cout << "\nvec to X"; x.nhap();*

*cout << "\nvec to Y"; y.nhap();*

*MT d= a.tich(b); VT u = d.tich(y); VT v = x.tich(c);*

*double s = v.tich(u);*

*cout << "\n\nVec to v\n";*

*v.in();*

*cout << "\n\nMa tran D"; d.in();*

*cout << "\n\nVec to y\n"; y.in();*

*cout << "\n\nS= vDy = " << s; getch();*

*}*

**chương 5**

A

# Dẫn xuất và thừa kế

Có 2 khái niệm rất quan trọng đã làm nên toàn bộ thế mạnh của phương pháp lập trình hướng đối tượng đó là tính kế thừa (inheritance) và tính tương ứng bội (polymorphism). Tính kế thừa cho phép các lớp được xây dựng trên các lớp đã có. Trong chương này sẽ nói về sự thừa kế của các lớp.

## Bài 1. Sự dẫn xuất và tính thừa kế

* 1. **Lớp cơ sở và lớp dẫn xuất**

Một lớp được xây dựng thừa kế một lớp khác gọi là lớp dẫn xuất. Lớp dùng để xây dựng lớp dẫn xuất gọi là lớp cơ sở.

Lớp nào cũng có thể là một lớp cơ sở. Hơn thế nữa, một lớp có thể là cơ sở cho nhiều lớp dẫn xuất khác nhau. Đến lượt mình, lớp dẫn xuất lại có thể dùng làm cơ sở để xây dựng các lớp dân xuất khác. Ngoài ra một lớp có thể dẫn xuất từ nhiều lớp cơ sở.

Dưới đây là một số sơ đồ về quan hệ dẫn xuất của các lớp:

**Sơ đồ 1:** Lớp B dẫn xuất từ lớp A, lớp C dẫn xuất từ lớp B

A

B

C

**Sơ đồ 2:** Lớp A là cơ sở của các lớp B, C và D

D

C

B

A

**Sơ đồ 3:** Lớp D dẫn xuất từ 3 lớp A, B, C

D

C

B

**Sơ đồ 4:** Lược đồ dẫn xuất tổng quát

E

D

C

B

A

F

G

H

**Tính thừa kế:** Một lớp dẫn xuất ngoài các thành phần của riêng nó, nó còn được thừa kế tất cả các thành phần của các lớp cơ sở có liên quan. Ví dụ trong sơ đồ 1 thì lớp C được thừa kế các thành phần của các lớp B và A. Trong sơ đồ 3 thì lớp D được thừa kế các thành phần của các lớp A, B và

C. Trong sơ đồ 4 thì lớp G được thừa kế các thành phần của các lớp D, E, A, B và C.

## Cách xây dựng lớp dân xuất

Giả sử đã định nghĩa các lớp A và B. Để xây dựng lớp C dân xuất từ A và B, ta viết như sau:

*class C : public A, public B*

*{*

*private:*

*// Khai báo các thuộc tính public:*

*// Các phương thức*

*} ;*

## Thừa kế private và public

Trong ví dụ trên, lớp C thừa kế public các lớp A và B. Nếu thay từ khoá public bằng private, thì sự thừa kế là private.

***Chú ý:*** Nếu bỏ qua không dùng từ khoá thì hiểu là private, ví dụ nếu định nghĩa:

*class C : public A, B*

*{*

*private:*

*// Khai báo các thuộc tính public:*

*// Các phương thức*

*} ;*

thì A là lớp cơ sở public của C , còn B là lớp cơ sở private của C.

Theo kiểu thừa kế public thì tất cả các thành phần public của lớp cơ sở cũng là các thành phần public của lớp dẫn xuất.

Theo kiểu thừa kế private thì tất cả các thành phần public của lớp cơ sở sẽ trơ thành các thành phần private của lớp dẫn xuất.

## Thừa kế các thành phần dữ liệu (thuộc tính)

Các thuộc tính của lớp cơ sở được thừa kế trong lớp dẫn xuất. Như vậy tập thuộc tính của lớp dẫn xuất sẽ gồm: các thuộc tính mới khai báo trong định nghĩa lớp dẫn xuất và các thuộc tính của lớp cơ sở.

Tuy vậy trong lớp dẫn xuất không cho phép truy nhập đến các thuộc tính private của lớp cơ sở.

***Chú ý:*** Cho phép đặt trùng tên thuộc tính trong các lớp cơ sở và lớp dẫn xuất.

## Ví dụ:

*class A*

*{*

*private:*

*int a, b, c; public:*

*...*

*};*

*class B*

*{*

*private:*

*double a, b, x; public:*

*...*

*};*

*class C : public A, B*

*{*

*private:*

*char \*a , \*x ; int b ;*

*public:*

*...*

*};*

Khi đó lớp C sẽ có các thuộc tính:

A::a , A::b, A::c (kiểu int) - thừa kế từ A

B::a , B::b, B::x (kiểu double) - thừa kế từ B

a, x (kiểu char\*) và b (kiểu int) - khai báo trong C

Trong các phương thức của C chỉ cho phép truy nhập trực tiếp tới các thuộc tính khai báo trong

C.

## Thừa kế phương thức

Trừ:

+ Hàm tạo

+ Hàm huỷ

+ Toán tử gán

các phương thức (public) khác của lớp cơ sở được thừa kế trong lớp dẫn xuất.

**Ví dụ:** Trong chương trình dưới đây:

+ Đầu tiên định nghĩa lớp DIEM có: Các thuộc tính x, y

Hai hàm tạo Phương thức in()

+ Sau đó xây dựng lớp HINH\_TRON dẫn xuất từ lớp DIEM, đưa thêm: Thuộc tính r

Hai hàm tạo Phương thức getR

***Chú ý*** cách dùng hàm tạo của lớp cơ sở (lớp DIEM) để xây dựng hàm tạo của lớp dẫn xuất.

+ Trong hàm main:

Khai báo đối tượng h kiểu HINH\_TRON

Sử dụng phương thức in() đối với h (sự thừa kế) Sử dụng phương thức getR đối với h

*//CT5-01*

*// Lop co so #include <conio.h>*

*#include <iostream.h> class DIEM*

*{*

*private:*

*double x, y; public:*

*DIEM()*

*{*

*x = y =0.0;*

*}*

*DIEM(double x1, double y1)*

*{*

*x = x1; y = y1;*

*}*

*void in()*

*{*

*cout << "\nx= " << x << " y= " << y;*

*}*

*};*

*class HINH\_TRON : public DIEM*

*{*

*private:*

*double r; public:*

*HINH\_TRON()*

*{*

*r = 0.0;*

*}*

*HINH\_TRON(double x1, double y1,*

*double r1): DIEM(x1,y1)*

*{*

*r = r1;*

*}*

*double getR()*

*{*

*return r;*

*}*

*};*

*void main()*

*{*

*HINH\_TRON h(2.5,3.5,8);*

*clrscr();*

*cout << "\nHinh tron co tam: "; h.in();*

*cout << "\nCo ban kinh= " << h.getR(); getch();*

*}*

## Lớp cơ sở và đối tượng thành phần

Lớp cơ sở thường được xử lý giống như một thành phần kiểu đối tượng của lớp dẫn xuất. Ví dụ chương trình trong 1.5 có thể thay bằng một chương trình khác trong đó thay việc dùng lớp cơ sở DIEM bằng một thành phần kiểu DIEM trong lớp HINH\_TRON. Chương trình mới có thể viết như sau:

*//CT5-02*

*// Lop co doi tuong thanh phan #include <conio.h>*

*#include <iostream.h> class DIEM*

*{*

*private:*

*double x, y; public:*

*DIEM()*

*{*

*x = y =0.0;*

*}*

*DIEM (double x1, double y1)*

*{*

*x = x1; y = y1;*

*}*

*void in()*

*{*

*cout << "\nx= " << x << " y= " << y;*

*}*

*} ;*

*class HINH\_TRON*

*{*

*private:*

*DIEM d;*

*double r; public:*

*HINH\_TRON() : d()*

*{*

*r = 0.0;*

*}*

*HINH\_TRON(double x1, double y1, double r1): d(x1,y1)*

*{*

*r = r1;*

*}*

*void in()*

*{*

*d.in();*

*}*

*double getR()*

*{*

*return r;*

*}*

*};*

*void main()*

*{*

*HINH\_TRON h(2.5,3.5,8);*

*clrscr();*

*cout << "\nHinh tron co tam: "; h.in();*

*cout << "\nCo ban kinh= " << h.getR(); getch();*

*}*

## Bài 2. Hàm tạo, hàm huỷ đối với tính thừa kế

* 1. **Lớp dẫn xuất không thừa kế các hàm tạo, hàm huỷ, toán tử gán của các lớp cơ sở**

## Cách xây dựng hàm tạo của lớp dẫn xuất

+ Hàm tạo cần có các đối để khởi gán cho các thuộc tính (thành phần dữ liệu) của lớp.

+ Có thể phân thuộc tính làm 3 loại ứng với 3 cách khởi gán khác nhau:

* + 1. Các thuộc tính mới khai báo trong lớp dẫn xuất. Trong các phương thức của lớp dẫn xuất có thể truy xuất đến các thuộc tính này. Vì vậy chúng thường được khởi gán bằng các câu lệnh gán viết trong thân hàm tạo.
    2. Các thành phần kiểu đối tượng. Trong lớp dẫn xuất không cho phép truy nhập đến các thuộc tính của các đối tượng này. Vì vậy để khởi gán cho các đối tượng thành phần cần dùng hàm tạo của lớp tương ứng. Điều này đã trình bầy trong mục 8 chương 4.
    3. Các thuộc tính thừa kế từ các lớp cở sở. Trong lớp dẫn xuất không được phép truy nhập đến các thuộc tính này. Vì vậy để khởi gán cho các thuộc tính nói trên, cần sử dụng hàm tạo của lớp cơ sở. Cách thức cũng giống như khởi gán cho các đối tượng thành phần, chỉ khác nhau ở chỗ: Để khởi gán cho các đối tượng thành phần ta dùng tên đối tượng thành phần, còn để khởi gán cho các thuộc tính thừa kế từ các lớp cơ sở ta dùng tên lớp cơ sở:

Tên\_đối\_tượng\_thành\_phần(danh sách giá trị) Tên\_lớp\_cơ\_sở(danh sách giá trị)

Danh sách giá trị lấy từ các đối của hàm tạo của lớp dẫn xuất đang xây dựng (xem ví dụ mục 2.4 và bài 6, ví dụ 1)

## Hàm huỷ

Khi một đối tượng của lớp dẫn xuất được giải phóng (bị huỷ), thì các đối tượng thành phần và các đối tượng thừa kế từ các lớp cơ sở cũng bị giải phóng theo. Do đó các hàm huỷ tương ứng sẽ được gọi đến.

Như vậy khi xây dựng hàm huỷ của lớp dẫn xuất, chúng ta chỉ cần quan tâm đến các thuộc tính (không phải là đối tượng) khai báo thêm trong lớp dẫn xuất mà thôi. Ta không cần để ý đến các đối tượng thành phần và các thuộc tính thừa kế từ các lớp cơ sở. (xem ví dụ mục 2.4 và bài 6, ví dụ 2)

## Ví dụ xét các lớp

+ Lớp NGUOI gồm:

* Các thuộc tính

char \*ht ; // Họ tên int ns ;

* Hai hàm tạo, phương thức in() và hàm huỷ

+ Lớp MON\_HOC gồm:

* Các thuộc tính

char \*monhoc ; // Tên môn học int st ; // Số tiết

* Hai hàm tạo, phương thức in() và hàm huỷ

+ Lớp GIAO\_VIEN :

* Kế thừa từ lớp NGUOI
* Đưa thêm các thuộc tính

char \*bomon ; // Bộ môn công tác MON\_HOC mh ; // Môn học đang dậy

* Hai hàm tạo , phương thức in() và hàm huỷ

Hãy để ý cách xây dựng các hàm tạo, hàm huỷ của lớp dẫn xuất GIAO\_VIEN. Trong lớp GIAO\_VIEN có thể gọi tới 2 phương thức in():

GIAO\_VIEN::in() // Được xây dựng trong lớp GIAO\_VIEN NGUOI::in() // Thừa kế từ lớp NGUOI

Hãy chú ý cách gọi tới 2 phương thức in() trong chương trình dưới đây.

*//CT5-03*

*// Ham tao cua lop dan suat #include <conio.h> #include <iostream.h> #include <string.h>*

*class MON\_HOC*

*{*

*private:*

*char \*monhoc; int st;*

*public:*

*MON\_HOC()*

*{*

*monhoc=NULL; st=0;*

*}*

*MON\_HOC(char \*monhoc1, int st1)*

*{*

*int n = strlen(monhoc1); monhoc = new char[n+1]; strcpy(monhoc,monhoc1); st=st1;*

*}*

*~ MON\_HOC()*

*{*

*if (monhoc!=NULL)*

*{*

*delete monhoc; st=0;*

*}*

*}*

*void in()*

*{*

*cout << "\nTen mon: " << monhoc << " so tiet: " << st;*

*}*

*} ;*

*class NGUOI*

*{*

*private:*

*char \*ht; int ns;*

*public:*

*NGUOI()*

*{*

*ht=NULL;*

*ns=0;*

*}*

*NGUOI(char \*ht1, int ns1)*

*{*

*int n = strlen(ht1); ht = new char[n+1]; strcpy(ht,ht1); ns=ns1;*

*}*

*~NGUOI()*

*{*

*if (ht!=NULL)*

*{*

*delete ht; ns=0;*

*}*

*}*

*void in()*

*{*

*cout << "\nHo ten : " << ht << " nam sinh: " << ns;*

*}*

*} ;*

*class GIAO\_VIEN : public NGUOI*

*{*

*private:*

*char \*bomon;*

*MON\_HOC mh;*

*public:*

*GIAO\_VIEN():mh(),NGUOI()//Su dung ham tao khong doi*

*{*

*bomon=NULL;*

*}*

*GIAO\_VIEN(char \*ht1, int ns1, char \*monhoc1,int st1, char \*bomon1 ): NGUOI(ht1,ns1),mh(monhoc1, st1)*

*{*

*int n = strlen(bomon1); bomon = new char[n+1]; strcpy(bomon,bomon1);*

*}*

*~GIAO\_VIEN()*

*{*

*if (bomon!=NULL) delete bomon;*

*}*

*void in()*

*{*

*// Su dung phuong thuc in NGUOI::in();*

*cout << "\n Cong tac tai bo mon: " << bomon; mh.in();*

*}*

*};*

*void main()*

*{*

*clrscr();*

*GIAO\_VIEN g1; // Goi toi cac ham tao khong doi GIAO\_VIEN \*g2;*

*//Goi toi cac ham tao co doi*

*g2 = new GIAO\_VIEN("PHAM VAN AT", 1945, "CNPM",*

*60, "TIN HOC");*

*g2->in();*

*/\**

*co the viet*

*g2->GIAO\_VIEN::in();*

*\*/*

*g2->NGUOI::in();*

*getch();*

*delete g2; // Goi toi cac ham huy getch();*

*}*

## Bài 3. Phạm vi truy nhập đến các thành phần của lớp cơ sở

* 1. **Các từ khoá quy định phạm vi truy nhập của lớp cơ sở**

+ Mặc dù lớp dẫn xuất được thừa kế tất cả các thành phần của lớp cơ sở, nhưng trong lớp dẫn xuất không thể truy nhập tới tất cả các thành phần này. Giải pháp thường dùng là sử dụng các phương thức của lớp cở sở để truy nhập đến các thuộc tính của chính lớp cơ sở đó. Cũng có thể sử dụng các giải pháp khác dưới đây.

+ Các thành phần private của lớp cở sở không cho phép truy nhập trong lớp dẫn xuất.

+ Các thành phần public của lớp cơ sở có thể truy nhập bất kỳ chỗ nào trong chương trình. Như vậy trong các lớp dẫn xuất có thể truy nhập được tới các thành phần này.

+ Các thành phần khai báo là protected có phạm vi truy nhập rộng hơn so với các thành phần private, nhưng hẹp hơn so với các thành phần public. Các thành phần protected của một lớp chỉ được mở rộng phạm vi truy nhập cho các lớp dẫn xuất trực tiếp từ lớp này.

## Hai kiểu dẫn xuất

Có 2 kiểu dẫn xuất là private và public, chúng cho các phạm vi truy nhập khác nhau tới lớp cơ sở. Cụ thể như sau:

+ Các thành phần public và protected của lớp cơ sở sẽ trở thành các thành phần public và protected của lớp dẫn xuất theo kiểu public.

+ Các thành phần public và protected của lớp cơ sở sẽ trở thành các thành phần private của lớp dẫn xuất theo kiểu private.

## Ví dụ :

Giả sử lớp A có:

thuộc tính public a1 thuộc tính protected a2

và lớp B dẫn xuất public từ A, thì A::a1 trở thành public trong B, A::a2 trở thành protected trong B.

Do đó nếu dùng B làm lớp cở để xây dựng lớp C. Thì trong C có thể truy nhập tới A::a1 và A::a2.

Thế nhưng nếu sửa đổi để B dẫn xuất private từ A, thì cả A::a1 và A::a2 trơ thành private trong B, và khi đó trong C không được phép truy nhập tới các thuộc tính A::a1 và A::a2.

Để biết tường tận hơn, chúng ta hãy biên dịch chương trình:

*//CT5-04*

*// Pham vi truy nhap #include <conio.h> #include <iostream.h> #include <string.h> class A*

*{*

*protected: int a1;*

*public:*

*int a2;*

*A()*

*{*

*a1=a2=0;*

*}*

*A(int t1, int t2)*

*{*

*a1=t1; a2= t2;*

*}*

*void in()*

*{*

*cout << a1 <<" " << a2 ;*

*}*

*} ;*

*class B: private A*

*{*

*protected: int b1;*

*public:*

*int b2;*

*B()*

*{*

*b1=b2=0;*

*}*

*B(int t1, int t2, int u1, int u2)*

*{*

*a1=t1; a2=t2; b1=u1;b2=u2;*

*}*

*void in()*

*{*

*cout << a1 <<" " << a2 << " " << b1 << " " << b2;*

*}*

*} ;*

*class C : public B*

*{*

*public:*

*C()*

*{*

*b1=b2=0;*

*}*

*C(int t1, int t2, int u1,int u2)*

*{*

*a1=t1; a2=t2; b1=u1;b2=u2;*

*}*

*void in()*

*{*

*cout << a1;*

*cout <<" " << a2 << " " << b1 << " " << b2;*

*}*

*};*

*void main()*

*{*

*C c(1,2,3,4);*

*c.in();*

*getch();*

*}*

Chúng ta sẽ nhận được 4 thông báo lỗi sau trong lớp C (tại hàm tạo có đối và phương thức in): A::a1 is not accessible

A::a2 is not accessible A::a1 is not accessible A::a2 is not accessible

Bây giờ nếu sửa đổi để lớp B dẫn xuất public từ A thì chương trình sẽ không có lỗi và làm việc tốt.

## Bài 4. Thừa kế nhiều mức và sự trùng tên

* 1. **Sơ đồ xây dựng các lớp dẫn xuất theo nhiều mức**

Như đã biết:

+ Khi đã định nghĩa một lớp (ví dụ lớp A), ta có thể dùng nó làm cơ sở để xây dựng lớp dẫn xuất (ví dụ B).

+ Đến lượt mình, B có thể dùng làm cơ sở để xây dựng lớp dẫn xuất mới (ví dụ C).

+ Tiếp đó lại có thể dùng C làm cơ sở để xây dựng lớp dẫn xuất mới.

+ Sự tiếp tục theo cách trên là không hạn chế.

Sơ đồ trên chính là sự thừa kế nhiều mức. Ngoài ra chúng ta cũng đã biết:

+ Một lớp có thể được dẫn xuất từ nhiều lớp cơ sở.

+ Một lớp có thể dùng làm cơ sở cho nhiều lớp.

Hình vẽ dưới đây là một ví dụ về sơ đồ thừa kế khá tổng quát, thể hiện được các điều nói trên:

B

A

C

G

F

D

H

E

Diễn giải:

Lớp D dẫn xuất từ A và B Lớp E dẫn xuất từ C

Lớp F dẫn xuất từ D

Lớp G dẫn xuất từ D và E Lớp H dẫn xuất từ E

## Sự thừa kế nhiều mức.

+ Như đã biết: Lớp dẫn xuất thừa kế tất cả các thành phần (thuộc tính và phương thức) của lớp cở sở, kể cả các thành phần mà lớp cơ sở được thừa kế.

+ Hãy áp dụng nguyên lý trên để xét lớp G:

* + - Lớp G thừa kế các thành phần của các lớp D và E
    - Lớp D thừa kế các thành phần của lớp A và B
    - Lớp E thừa kế các thành phần của lớp C

Như vậy các thành phần có thể sử trong lớp G gồm:

* + - Các thành phần khai báo trong G (của riêng G)
    - Các thành phần khai báo trong các lớp D, E, A, B, C (được thừa kế).

## Sự trùng tên

Như đã nói trong 4.2: Trong lớp G có thể sử dụng (trực tiép hay gián tiếp) các thành phần của riêng G và các thành phần mà nó được thừa kế từ các lớp D, E, A, B, C. Yêu cầu về cách đặt tên ở đây là:

+ Tên các lớp không được trùng lặp

+ Tên các thành phần trong một lớp cũng không được trùng lặp

+ Tên các thành phần trong các lớp khác nhau có quyền được trùng lặp.

Để phân biệt các thành phần trùng tên trong lớp dẫn xuất, cần sử dụng thêm tên lớp (xem ví dụ trong 4.4).

## Sử dụng các thành phần trong lớp dẫn xuất

Như đã nói ở trên: Thành phần của lớp dẫn xuất gồm:

+ Các thành phần khai báo trong lớp dẫn xuất

+ Các thành phần mà lớp dẫn xuất thừa kế từ các lớp cơ sở Quy tắc sử dụng các thành phần trong lớp dẫn xuất:

*Cách 1:* Dùng tên lớp và tên thành phần. Khi đó Chương trình dịch C++ dễ dàng phân biệt thành phần thuộc lớp nào. Ví dụ:

D h; // h là đối tượng của lớp D dẫn xuất từ A và B h.D::n là thuộc tính n khai báo trong D

h.A::n là thuộc tính n thừa kế từ A (khai báo trong A) h.D::nhap() là phương thức nhap() định nghĩa trong D h.A::nhap() là phương thức nhap() định nghĩa trong A

*Cách 2:* Không dùng tên lớp, chỉ dùng tên thành phần. Khi đó Chương trình dịch C++ phải tự phán đoán để biết thành phần đó thuộc lớp nào. Cách phán đoán như sau: Trước tiên xem thành phần đang xét có trùng tên với một thành phần nào của lớp dẫn xuất không? Nếu trùng thì đó là thành phần của lớp dẫn xuất. Nếu không trùng thì tiếp tục xét các lớp cơ sở theo thứ tự: Các lớp có quan hệ gần với lớp dẫn xuất xét trước, các lớp quan hệ xa xét sau. Hãy chú ý trường hợp sau:

Thành phần đang xét có mặt đồng thời trong 2 lớp cơ sở có cùng một đẳng cấp quan hệ với lớp dẫn xuất. Gặp trường hợp này Chương trình dịch C++ không thể quyết định được thành phần này thừa kế từ lớp nào và buộc phải đưa ra một thông báo lỗi (xem ví dụ dưới đây). Cách khắc phục: Trường hợp này phải sử dụng thêm tên lớp như trình bầy trong cách 1.

**Ví dụ** xét lớp dẫn xuất D. Lớp D có 2 cơ sở là các lớp A và B. Giả sử các lớp A, B và D được định nghĩa:

*class A*

*{*

*private:*

*int n;*

*float a[20]; public:*

*void nhap(); void xuat():*

*} ;*

*class B*

*{*

*private:*

*int m,n;*

*float a[20][20]; public:*

*void nhap(); void xuat():*

*} ;*

*class D : public A, public B*

*{*

*private:*

*int k; public:*

*void nhap(); void xuat():*

*} ;*

### Hãy chú ý các điểm sau:

1. Dùng các phương thức của các lớp A, B để xây dựng các phương thức của D

// Xây dựng phương thức nhap() void D::nhap()

{

cout << “\n Nhap k : “ ;

cin >> k ; // k là thuộc tính của D

A::nhap(); // Nhập các thuộc tính mà D thừa kế từ A

B::nhap(); // Nhập các thuộc tính mà D thừa kế từ B

}

// Xây dựng phương thức xuat() void D::xuat()

{

cout << “\n k = “ << k ;

A::xuat(); // Xuất các thuộc tính mà D thừa kế từ A B::xuat(); // Xuất các thuộc tính mà D thừa kế từ B

}

1. Làm việc với các đối tượng của lớp dẫn xuất D h ; // Khai báo h là đối tượng của lớp D h.nhap() ; // tương tương với h.D::nhap();

h.A::xuat(); // In giá trị các thuộc tính h.A::n và h.A::a h.B::xuat(); // In giá trị các thuộc tính h.B::m, h.B::n và h.B::a h.D::xuat() ; // In giá trị tất cả các thuộc tính của h

h.xuat() ; // tương đương với h.D::xuat() ;

## Bài 5. Các lớp cơ sở ảo

* 1. **Một lớp cơ sở xuất hiện nhiều lần trong lớp dẫn xuất**

Một điều hiển nhiên là không thể khai báo 2 lần cùng một lớp trong danh sách của các lớp cơ sở cho một lớp dẫn xuất. Chẳng hạn ví dụ sau là không cho phép:

class B : public A, public A

{

} ;

Tuy nhiên vẫn có thể có trường hợp cùng một lớp cơ sở được đề cập nhiều hơn một lần trong các lớp cơ sở trung gian của một lớp dẫn xuất. Ví dụ:

*#include <iostream.h> class A*

*{*

*public:*

*int a;*

*} ;*

*class B : public A*

*{*

*public:*

*int b;*

*} ;*

*class C : public A*

*{*

*public:*

*int c;*

*} ;*

*class D : public B , public C*

*{*

*public:*

*int d;*

*} ;*

*void main()*

*{*

*D h ;*

*h.d = 4 ; // tốt*

*h.c = 3 ; // tốt*

*h.b = 2 ; // tốt*

*h.a = 1 ; // lỗi*

*}*

Trong ví dụ này A là cơ sở cho cả 2 lớp cơ sở trực tiếp của D là B và C. Nói cách khác có 2 lớp cơ sở A cho lớp D. Vì vậy trong câu lệnh:

h.a = 1 ;

thì Chương trình dịch C++ không thể nhận biết được thuộc tính a thừa kế thông qua B hay thông qua C và nó sẽ đưa ra thông báo lỗi sau:

Member is ambiguous: ‘A::a’ and ‘A::a’

## Các lớp cơ sở ảo

Giải pháp cho vấn đề nói trên là khai báo A như một lớp cơ sở kiểu virtual cho cả B và C. Khi đó B và C được định nghĩa như sau:

*class B : virtual public A*

*{*

*public:*

*int b;*

*} ;*

*class C : virtual public A*

*{*

*public:*

*int c;*

*} ;*

Các lớp cơ sở ảo (virtual) sẽ được kết hợp để tạo một lớp cơ sở duy nhất cho bất kỳ lớp nào dẫn xuất từ chúng. Trong ví dụ trên, hai lớp cơ sở A ( A là cơ sở của B và A là cơ sở của C) sẽ kết hợp lại để trở thành một lớp cơ sở A duy nhất cho bất kỳ lớp dẫn xuất nào từ B và C. Như vậy bây giờ D sẽ chỉ có một lớp cơ sở A duy nhất, do đó phép gán:

h.a = 1 ;

sẽ gán 1 cho thuộc tính a của lớp cơ sở A duy nhất mà D kế thừa.

## Bài 6. Một số ví dụ về hàm tạo, hàm huỷ trong thừa kế nhiều mức

**Ví dụ 1.** Ví dụ này minh hoạ cách xây dựng hàm tạo trong các lớp dẫn xuất. Ngoài ra còn minh hoạ cách dùng các phương thức của các lớp cơ sở trong lớp dẫn xuất và cách xử lý các đối tượng thành phần.

Xét 4 lớp A, B, C và D. Lớp C dẫn xuất từ B, lớp D dẫn xuất từ C và có thành phần là đối tượng kiểu A.

*//CT5-06*

*// Thua ke nhieu muc*

*// Ham tao*

*#include <conio.h> #include <iostream.h> #include <string.h> class A*

*{*

*private:*

*int a; char \*str ;*

*public:*

*A()*

*{*

*a=0; str=NULL;*

*}*

*A(int a1,char \*str1)*

*{*

*a=a1; str=strdup(str1);*

*}*

*void xuat()*

*{*

*} ;*

*class B*

*{*

*cout << "\n" << "So nguyen lop A= " << a*

*<< " Chuoi lop A: " << str ;*

*}*

*private:*

*int b; char \*str ;*

*public:*

*B()*

*{*

*b=0; str=NULL;*

*}*

*B(int b1,char \*str1)*

*{*

*b=b1; str=strdup(str1);*

*}*

*void xuat()*

*{*

*cout << "\n" << "So nguyen lop B = " << b*

*<< " Chuoi lop B: " << str ;*

*}*

*} ;*

*class C : public B*

*{*

*private:*

*int c;*

*char \*str ; public:*

*C():B()*

*{*

*c=0; str=NULL;*

*}*

*C(int b1,char \*strb,int c1, char \*strc) : B(b1,strb)*

*{*

*c=c1; str=strdup(strc);*

*}*

*void xuat()*

*{*

*B::xuat();*

*cout << "\n" << "So nguyen lop C = " << c*

*<< " Chuoi lop C: " << str ;*

*}*

*} ;*

*class D : public C*

*{*

*private:*

*int d; char \*str ; A u;*

*public:*

*D():C(),u()*

*{*

*d=0; str=NULL;*

*}*

*D(int a1, char \*stra,int b1,char \*strb,int c1, char \*strc,*

*int d1, char \*strd) : u(a1,stra), C(b1,strb,c1,strc)*

*{*

*d=d1; str=strdup(strd);*

*}*

*void xuat()*

*{*

*u.xuat();*

*C::xuat();*

*cout << "\n" << "So nguyen lop D = " << d*

*<< " Chuoi lop D: " << str ;*

*}*

*} ;*

*void main()*

*{*

*D h(1,"AA",2,"BB",3,"CC",4,"DD");*

*clrscr();*

*cout << "\n\n Cac thuoc tinh cua h thua ke B: " ; h.B::xuat();*

*cout << "\n\n Cac thuoc tinh cua h thua ke B va C: " ; h.C::xuat();*

*cout << "\n\n Cac thuoc tinh cua h thua ke B,C va khai bao trong D:" ; h.xuat();*

*getch();*

*}*

**Ví dụ 2.** Ví dụ này minh hoạ cách xây dựng hàm huỷ trong lớp dẫn xuất. Chương trình trong ví dụ này lấy từ chương trình của ví dụ 1, sau đó đưa thêm vào các hàm huỷ.

*//CT5-07*

*// Thua ke nhieu muc*

*// Ham tao*

*// Ham huy #include <conio.h>*

*#include <iostream.h> #include <string.h> class A*

*{*

*private:*

*int a; char \*str ;*

*public:*

*A()*

*{*

*a=0; str=NULL;*

*}*

*A(int a1,char \*str1)*

*{*

*a=a1; str=strdup(str1);*

*}*

*~A()*

*{*

*cout <<"\n Huy A"; getch(); a=0;*

*if (str!=NULL) delete str;*

*}*

*void xuat()*

*{*

*} ;*

*class B*

*{*

*cout << "\n" << "So nguyen lop A= " << a*

*<< " Chuoi lop A: " << str ;*

*}*

*private:*

*int b; char \*str ;*

*public:*

*B()*

*{*

*b=0; str=NULL;*

*}*

*B(int b1,char \*str1)*

*{*

*b=b1; str=strdup(str1);*

*}*

*~B()*

*{*

*cout <<"\n Huy B"; getch(); b=0;*

*if (str!=NULL) delete str;*

*}*

*void xuat()*

*{*

*cout << "\n" << "So nguyen lop B = " << b*

*<< " Chuoi lop B: " << str ;*

*}*

*} ;*

*class C : public B*

*{*

*private:*

*int c;*

*char \*str ; public:*

*C():B()*

*{*

*c=0; str=NULL;*

*}*

*C(int b1,char \*strb,int c1, char \*strc) : B(b1,strb)*

*{*

*c=c1; str=strdup(strc);*

*}*

*~C()*

*{*

*cout <<"\n Huy C"; getch(); c=0;*

*if (str!=NULL) delete str;*

*}*

*void xuat()*

*{*

*B::xuat();*

*cout << "\n" << "So nguyen lop C = " << c*

*<< " Chuoi lop C: " << str ;*

*}*

*} ;*

*class D : public C*

*{*

*private:*

*int d; char \*str ; A u;*

*public:*

*D():C(),u()*

*{*

*d=0; str=NULL;*

*}*

*D(int a1, char \*stra,int b1,char \*strb,int c1, char \*strc, int d1, char \*strd) : u(a1,stra), C(b1,strb,c1,strc)*

*{*

*d=d1; str=strdup(strd);*

*}*

*~D()*

*{*

*cout <<"\n Huy D"; getch(); d=0;*

*if (str!=NULL) delete str;*

*}*

*void xuat()*

*{*

*u.xuat();*

*C::xuat();*

*cout << "\n" << "So nguyen lop D = " << d*

*<< " Chuoi lop D: " << str ;*

*}*

*} ;*

*void main()*

*{*

*D \*h;*

*h = new D(1,"AA",2,"BB",3,"CC",4,"DD");*

*clrscr();*

*cout << "\n\n Cac thuoc tinh cua h thua ke B: " ; h->B::xuat();*

*cout << "\n\n Cac thuoc tinh cua h thua ke B va C: " ; h->C::xuat();*

*cout << "\n\n Cac thuoc tinh cua h thua ke B,C va khai bao trong D:" ; h->xuat();*

*delete h; // Lan luot goi toi cac ham huy cua cac lop D, A, C, B getch();*

*}*

## Bài 7. Toán tử gán của lớp dẫn xuất

* 1. **Khi nào cần xây dựng toán tử gán:** Khi lớp dẫn xuất có các thuộc tính (kể cả thuộc tính thừa kế từ các lớp cơ sở) là con trỏ, thì nhất thiết không được dùng toán tử gán mặc định, mà phải xây dựng cho lớp dẫn xuất một toán tử gán.

## Cách xây dựng toán tử gán cho lớp dẫn xuất

+ Trước hết cần xây dựng toán tử gán cho các lớp cơ sở

+ Vấn đề mấu chốt là: Khi xây dựng toán tử gán cho lớp dẫn xuất thì làm thế nào để sử dụng được các toán tử gán của lớp cơ sở. Cách giải quyết như sau:

- Xây dựng các phương thức (trong các lớp cơ sở) để nhận được địa chỉ của đối tượng ẩn của lớp. Phương thức này được viết đơn giản theo mẫu:

Tên\_lớp \* get\_DT ( )

{

return this ;

}

- Trong thân của toán tử gán (cho lớp dẫn xuất), sẽ dùng phương thức trên để nhận địa chỉ đối tượng của lớp cơ sở mà lớp dẫn xuất thừa kế. Sau đó thực hiện phép gán trên các đối tượng này.

Phương pháp nêu trên có thể minh hoạ một cách hình thức như sau: Giả sử lớp B dân xuất từ A. Để xây dựng toán tử gán cho B, thì:

* + 1. Trong lớp A cần xây dựng toán tử gán và phương thức cho địa chỉ của đối tượng ẩn. Cụ thể A cần được định nghĩa như sau:

class A

{

...

A & operator=(A& h)

{

//các câu lệnh để thực hiện gán trong A

}

// Phương thức nhận địa chỉ đối tượng ẩn của A A\* get\_A()

{

return this;

}

...

} ;

* + 1. Toán tử gán trong lớp B cần như sau: class B : public A

{

...

B & operator=(B& h)

{

A \*u1, \*u2;

u1 = this->get\_A(); u2 = h.get\_A();

\*u1 = \*u2 ; // Sử dụng phép gán trong A để gán các

// thuộc tính mà B kế thừa từ A

//Các câu lệnh thực hiện gán các thuộc tính riêng của B

}

...

} ;

## Ví dụ

Chương trình dưới đây minh hoạ cách xây dựng toán tử gán cho lớp D có 2 lớp cơ sở là C và B (C là lớp cơ sở trực tiếp, còn B là cơ sở của C) . Ngoài ra D còn có một thuộc tính là đối tượng của lớp A.

*//CT5-08*

*// Thua ke nhieu muc*

*// gan*

*#include <conio.h> #include <stdio.h> #include <iostream.h> #include <string.h> class A*

*{*

*private:*

*int a; char \*str ;*

*public:*

*A()*

*{*

*a=0; str=NULL;*

*}*

*A& operator=(A& h)*

*{*

*this->a = h.a;*

*if (this->str!=NULL) delete this->str; this->str = strdup(h.str);*

*return h;*

*}*

*void nhap()*

*{*

*cout << "\nNhap so nguyen lop A: " ; cin >> a ; if (str!=NULL) delete str;*

*cout << "\nNhap chuoi lop A: " ; char tg[30];*

*fflush(stdin); gets(tg); str = strdup(tg);*

*}*

*void xuat()*

*{*

*cout << "\n" << "So nguyen lop A= " << a*

*<< " Chuoi lop A: " << str ;*

*}*

*} ;*

*class B*

*{*

*private:*

*int b; char \*str ;*

*public:*

*B()*

*{*

*b=0; str=NULL;*

*}*

*B\* getB()*

*{*

*return this;*

*}*

*B& operator=(B& h)*

*{*

*this->b = h.b;*

*if (this->str!=NULL) delete this->str; this->str = strdup(h.str);*

*return h;*

*}*

*void nhap()*

*{*

*cout << "\nNhap so nguyen lop B: " ; cin >> b ; if (str!=NULL) delete str;*

*cout << "\nNhap chuoi lop B: " ; char tg[30];*

*fflush(stdin); gets(tg); str = strdup(tg);*

*}*

*void xuat()*

*{*

*cout << "\n" << "So nguyen lop B = " << b*

*<< " Chuoi lop B: " << str ;*

*}*

*} ;*

*class C : public B*

*{*

*private:*

*int c;*

*char \*str ; public:*

*C():B()*

*{*

*c=0; str=NULL;*

*}*

*C\* getC()*

*{*

*return this;*

*}*

*C& operator=(C& h)*

*{*

*B \*b1, \*b2;*

*b1= this->getB(); b2= h.getB();*

*\*b1 = \*b2; this->c = h.c;*

*if (this->str!=NULL) delete this->str; this->str = strdup(h.str);*

*return h;*

*}*

*void nhap()*

*{*

*B::nhap();*

*cout << "\nNhap so nguyen lop C: " ; cin >> c ; if (str!=NULL) delete str;*

*cout << "\nNhap chuoi lop C: " ; char tg[30];*

*fflush(stdin); gets(tg); str = strdup(tg);*

*}*

*void xuat()*

*{*

*B::xuat();*

*cout << "\n" << "So nguyen lop C = " << c*

*<< " Chuoi lop C: " << str ;*

*}*

*} ;*

*class D : public C*

*{*

*private:*

*int d; char \*str ; A u;*

*public:*

*D():C(),u()*

*{*

*d=0; str=NULL;*

*}*

*D& operator=(D& h)*

*{*

*this->u = h.u;*

*C \*c1,\*c2;*

*c1 = this->getC(); c2 = h.getC();*

*\*c1 = \*c2; this->d = h.d;*

*if (this->str!=NULL) delete this->str; this->str = strdup(h.str);*

*return h;*

*}*

*void nhap()*

*{*

*u.nhap();*

*C::nhap();*

*cout << "\nNhap so nguyen lop D: " ; cin >> d ; if (str!=NULL) delete str;*

*cout << "\nNhap chuoi lop D: " ; char tg[30];*

*fflush(stdin); gets(tg); str = strdup(tg);*

*}*

*void xuat()*

*{*

*u.xuat();*

*C::xuat();*

*cout << "\n" << "So nguyen lop D = " << d*

*<< " Chuoi lop D: " << str ;*

*}*

*} ;*

*void main()*

*{*

*D h1,h2,h3; clrscr();*

*h1.nhap(); h3=h2=h1; cout<<"\n\nH2:"; h2.xuat(); cout<<"\n\nH3:"; h3.xuat();*

*h1.nhap(); cout<<"\n\nH2:"; h2.xuat(); cout<<"\n\nH3:"; h3.xuat(); cout<<"\n\nH1:"; h1.xuat();*

*getch();*

*}*

## Bài 8. Hàm tạo sao chép của lớp dẫn xuất

* 1. **Khi nào cần xây dựng hàm tạo sao chép:** Khi lớp dẫn xuất có các thuộc tính (kể cả thuộc tính thừa kế từ các lớp cơ sở) là con trỏ, thì nhất thiết không được dùng hàm tạo sao chép mặc định, mà phải xây dựng cho lớp dẫn xuất một hàm tạo sao chép.

## Cách xây dựng hàm tạo sao chép cho lớp dẫn xuất

+ Trước hết cần xây dựng toán tử gán cho lớp dẫn xuất (xem §7).

+ Sau đó xây dựng hàm tạo sao chép cho lớp dẫn xuất theo mẫu: Tên\_lớp\_dẫn\_xuất (Tên\_lớp\_dẫn\_xuất &h )

{

\*this = h ;

}

## Ví dụ

Chương trình dưới đây minh hoạ cách xây dựng hàm tạo sao chép cho lớp D có 2 lớp cơ sở là C và B (C là lớp cơ sở trực tiếp, còn B là cơ sở của C) . Ngoài ra D còn có một thuộc tính là đối tượng của lớp A. Chương trình này dựa trên chương trình trong mục 7.3 với 2 thay đổi:

+ Xây dựng thêm hàm tạo sao chép cho lớp D.

+ Thay đổi một số câu lệnh trong hàm main để sử dùng hàm tạo sao chép.

Để thấy rõ vai trò của hàm tạo sao chép chúng ta hãy so sánh kết quả nhận được trong 2 trường hợp: Có hàm tạo sao chép và bỏ đi hàm này.

*//CT5-09*

*// Thua ke nhieu muc*

*// Ham tao sao chep #include <conio.h> #include <stdio.h> #include <iostream.h> #include <string.h> class A*

*{*

*private:*

*int a; char \*str ;*

*public:*

*A()*

*{*

*a=0; str=NULL;*

*}*

*A& operator=(A& h)*

*{*

*this->a = h.a;*

*if (this->str!=NULL) delete this->str; this->str = strdup(h.str);*

*return h;*

*}*

*void nhap()*

*{*

*cout << "\nNhap so nguyen lop A: " ; cin >> a ; if (str!=NULL) delete str;*

*cout << "\nNhap chuoi lop A: " ; char tg[30];*

*fflush(stdin); gets(tg); str = strdup(tg);*

*}*

*void xuat()*

*{*

*cout << "\n" << "So nguyen lop A= " << a*

*<< " Chuoi lop A: " << str ;*

*}*

*} ;*

*class B*

*{*

*private:*

*int b; char \*str ;*

*public:*

*B()*

*{*

*b=0; str=NULL;*

*}*

*B\* getB()*

*{*

*return this;*

*}*

*B& operator=(B& h)*

*{*

*this->b = h.b;*

*if (this->str!=NULL) delete this->str; this->str = strdup(h.str);*

*return h;*

*}*

*void nhap()*

*{*

*cout << "\nNhap so nguyen lop B: " ; cin >> b ; if (str!=NULL) delete str;*

*cout << "\nNhap chuoi lop B: " ; char tg[30];*

*fflush(stdin); gets(tg); str = strdup(tg);*

*}*

*void xuat()*

*{*

*cout << "\n" << "So nguyen lop B = " << b*

*<< " Chuoi lop B: " << str ;*

*}*

*} ;*

*class C : public B*

*{*

*private:*

*int c;*

*char \*str ; public:*

*C():B()*

*{*

*c=0; str=NULL;*

*}*

*C\* getC()*

*{*

*return this;*

*}*

*C& operator=(C& h)*

*{*

*B \*b1, \*b2;*

*b1= this->getB(); b2= h.getB();*

*\*b1 = \*b2; this->c = h.c;*

*if (this->str!=NULL) delete this->str; this->str = strdup(h.str);*

*return h;*

*}*

*void nhap()*

*{*

*B::nhap();*

*cout << "\nNhap so nguyen lop C: " ; cin >> c ; if (str!=NULL) delete str;*

*cout << "\nNhap chuoi lop C: " ; char tg[30];*

*fflush(stdin); gets(tg); str = strdup(tg);*

*}*

*void xuat()*

*{*

*B::xuat();*

*cout << "\n" << "So nguyen lop C = " << c*

*<< " Chuoi lop C: " << str ;*

*}*

*} ;*

*class D : public C*

*{*

*private:*

*int d; char \*str ; A u;*

*public:*

*D():C(),u()*

*{*

*d=0; str=NULL;*

*}*

*D(D& h) // Ham tao sao chep*

*{*

*\*this=h;*

*}*

*D& operator=(D& h)*

*{*

*this->u = h.u;*

*C \*c1,\*c2;*

*c1 = this->getC(); c2 = h.getC();*

*\*c1 = \*c2; this->d = h.d;*

*if (this->str!=NULL) delete this->str; this->str = strdup(h.str);*

*return h;*

*}*

*void nhap()*

*{*

*u.nhap();*

*C::nhap();*

*cout << "\nNhap so nguyen lop D: " ; cin >> d ; if (str!=NULL) delete str;*

*cout << "\nNhap chuoi lop D: " ; char tg[30];*

*fflush(stdin); gets(tg); str = strdup(tg);*

*}*

*void xuat()*

*{*

*u.xuat();*

*C::xuat();*

*cout << "\n" << "So nguyen lop D = " << d*

*<< " Chuoi lop D: " << str ;*

*}*

*} ;*

*void main()*

*{*

*D h1;*

*clrscr();*

*h1.nhap();*

*D h2(h1);*

*cout<<"\n\nH2:"; h2.xuat();*

*h1.nhap(); cout<<"\n\nH2:"; h2.xuat(); cout<<"\n\nH1:"; h1.xuat();*

*getch();*

*}*

## Bài 9. Phát triển, hoàn thiện chương trình

Có thể dùng tính thừa kế để phát triển khả năng của chương trình.

* 1. **ý tưởng của việc phát triển chương trình như sau:** Sau khi đã xây dựng được một lớp, ta sẽ phát triển lớp này bằng cách xây một lớp dân xuất trong đó đưa thêm các thuộc tính và phương thức mới. Quá trình trên lại tiếp tục với lớp vừa nhận được. Ta cũng có thể xây dựng các lớp mới có thuộc tính là đối tượng của các lớp cũ. Bằng cách này, sẽ nhận được một dẫy các lớp càng ngày càng hoàn thiện và có nhiều khả năng hơn.

## Ví dụ về việc phát triển chương trình

Giả sử cần xây dựng chương trình vẽ một số hình phẳng. Chúng ta có thể phát triển chương trình này như sau:

Đầu tiên định nghĩa lớp DIEM (Điểm) gồm 2 thuộc tính x, y. Từ lớp DIEM xây dựng lớp DUONG\_TRON (Đương tròn) bằng cách bổ sung 2 biến nguyên là r (bán kính) và md (mầu đường). Từ lớp DUONG\_TRON xây dựng lớp HINH\_TRON bằng cách thêm vào biến nguyêm mt (mầu tô). Đi theo một nhánh khác: Xây dựng lớp DOAN\_THANG (Đoạn thẳng) gồm 2 đối tượng kiểu DIEM, và lớp TAM\_GIAC gồm 3 đối tượng DIEM.

Chương trình dưới đây cho phép vẽ các đường tròn, hình tròn, đoạn thẳng và hình tam giác.

Chương trình còn minh hoạ cách dùng con trỏ **this** trong lớp dẫn xuất để thực hiện các phương thức của lớp co sở. Ngoài ra còn minh hoạ cách dùng toán tử chỉ số [] để nhận các toạ độ x, y từ các đối tượng của lớp DIEM.

*//CT5-10*

*// Phat trien chuong trinh #include <conio.h> #include <stdio.h> #include <iostream.h> #include <string.h> #include <graphics.h> class DIEM*

*{*

*private:*

*int x,y; public:*

*DIEM()*

*{*

*x=y=0;*

*}*

*DIEM(int x1, int y1)*

*{*

*x=x1; y=y1;*

*}*

*DIEM(DIEM &d)*

*{*

*this->x= d.x; this->y= d.y;*

*}*

*int operator[](int i)*

*{*

*if (i==1) return x; else return y;*

*}*

*};*

*class DUONG\_TRON : public DIEM*

*{*

*private:*

*int r,md; public:*

*DUONG\_TRON() : DIEM()*

*{*

*r=md=0;*

*}*

*DUONG\_TRON(DIEM d, int r1, int md1) : DIEM(d)*

*{*

*r=r1; md=md1;*

*}*

*void ve()*

*{*

*setcolor(md);*

*circle ( (\*this)[1],(\*this)[2],r);*

*}*

*int getmd()*

*{*

*return md;*

*}*

*} ;*

*class HINH\_TRON : public DUONG\_TRON*

*{*

*private:*

*int mt; public:*

*HINH\_TRON() : DUONG\_TRON()*

*{*

*mt=0;*

*}*

*HINH\_TRON(DIEM d, int r1, int md1, int mt1) :*

*DUONG\_TRON(d,r1,md1)*

*{*

*mt=mt1;*

*}*

*void ve()*

*{*

*DUONG\_TRON::ve();*

*setfillstyle(1,mt); floodfill((\*this)[1],(\*this)[2],this->getmd());*

*}*

*} ;*

*class DOAN\_THANG*

*{*

*private:*

*DIEM d1, d2;*

*int md; public:*

*DOAN\_THANG() : d1(), d2()*

*{*

*md=0;*

*}*

*DOAN\_THANG(DIEM t1, DIEM t2, int md1)*

*{*

*d1=t1; d2 = t2; md=md1;*

*}*

*void ve()*

*{*

*setcolor(md);*

*line(d1[1],d1[2] ,d2[1],d2[2]);*

*}*

*} ;*

*class TAM\_GIAC*

*{*

*private:*

*DIEM d1,d2,d3;*

*int md, mt; public:*

*TAM\_GIAC(): d1(), d2(), d3()*

*{*

*md=mt=0;*

*}*

*TAM\_GIAC(DIEM t1,DIEM t2,DIEM t3,int md1,int mt1)*

*{*

*d1=t1; d2=t2; d3 = t3; md=md1;mt=mt1;*

*}*

*void ve()*

*{*

*DOAN\_THANG(d1,d2,md).ve();*

*DOAN\_THANG(d1,d3,md).ve();*

*DOAN\_THANG(d2,d3,md).ve();*

*setfillstyle(1,mt); floodfill((d1[1]+d2[1]+d3[1])/3,(d1[2]+d2[2]+d3[2])/3,md);*

*}*

*} ;*

*void ktdh()*

*{*

*int mh=0,mode=0; initgraph(&mh,&mode,"");*

*}*

*void main()*

*{*

*ktdh();*

*DUONG\_TRON dt(DIEM(100,100),80,MAGENTA); HINH\_TRON ht(DIEM(400,100),80,RED,YELLOW); DOAN\_THANG t(DIEM(100,100),DIEM(400,100),BLUE); TAM\_GIAC tg(DIEM(250,150), DIEM(100,400),*

*DIEM(400,400), CYAN, CYAN);*

*dt.ve();*

*ht.ve();*

*t.ve();*

*tg.ve();*

*getch(); closegraph();*

*}*

## Bài 10. Bổ sung, nâng cấp chương trình

Có thể dùng tính thừa kế để sửa đổi, bổ sung, nâng cấp chương trình.

* 1. **ý tưởng của việc nâng cấp chương trình như sau:** Giả sử đã có một chương trình hoạt động tốt. Bây giờ cần có một số bổ sung thay đổi không nhiều lắm. Có 2 giải pháp: Hoặc sửa chữa các lớp đang hoạt động ổn định, hoặc xây dựng một lớp dẫn xuất để thực hiện các bổ sung, sửa chữa trên lớp này. Rõ ràng giải pháp thứ 2 tỏ ra hợp lý hơn .

## Ví dụ

Giả sử đã có chương trình quản lý giáo viên gồm 3 lớp MON\_HOC (Môn học), GV (Giáo viên) và BM (Bộ môn) :

class MON\_HOC

*{*

*private:*

*char tenmh[20] ; // Tên môn học int sotiet ; // Số tiết*

*public:*

*MON\_HOC() ; // Hàm tạo*

*const MON\_HOC& operator=(const MON\_HOC& m) ;*

*// Gán*

*void nhap() ; // Nhập void xuat() ; // Xuất*

*} ;*

*class GV*

*{*

*private:*

*char ht[25]; // Ho ten int ns; // Nam sinh*

*int sm; // So mon hoc co the day MON\_HOC \*mh ; //Danh sach cac mon hoc*

*public:*

*GV() ; // Hàm tạo*

*~GV() ; //Hàm huỷ*

*int getsm() ; // Cho biết số môn (dùng trong BM::sapxep) const GV& operator=(const GV& g); // Gán (dùng trong*

*// BM::sapxep)*

*void nhap(); // Nhập void xuat(); // Xuất*

*} ;*

*class BM // Bo mon*

*{*

*private:*

*char tenbm[20]; // Tên bộ môn int n; // So giao vien*

*GV \*gv; // Danh sach giao vien public:*

*BM() // Hàm tạo void nhap(); // Nhập void xuat(); // Xuất*

*void sapxep(); // Sắp xếp*

*} ;*

Chương trình cho phép:

* + 1. Nhập danh sách giáo viên của bộ môn.
    2. Sắp xếp danh sách giáo viên theo thứ tự giảm của số môn mà mỗi giáo viên có thể giảng dậy.
    3. In danh sách giáo viên.

Nội dung chương trình như sau:

*//CT5-11*

*// Nang cap chuong trinh*

*// CT ban dau #include <conio.h> #include <stdio.h>*

*#include <iostream.h> #include <string.h> class MON\_HOC*

*{*

*private:*

*char tenmh[20]; int sotiet;*

*public:*

*MON\_HOC()*

*{*

*tenmh[0]=sotiet=0;*

*}*

*const MON\_HOC& operator=(const MON\_HOC& m)*

*{*

*strcpy(this->tenmh,m.tenmh); this->sotiet = m.sotiet; return m;*

*}*

*void nhap()*

*{*

*cout << "\nTen mon hoc:"; fflush(stdin); gets(tenmh); cout << "So tiet: " ;*

*cin >> sotiet;*

*}*

*void xuat()*

*{*

*cout << "\nTen mon hoc:" << tenmh*

*<< " so tiet: " << sotiet;*

*}*

*} ;*

*class GV*

*{*

*private:*

*char ht[25]; // Ho ten int ns; // Nam sinh*

*int sm; // So mon hoc co the day MON\_HOC \*mh ; //Danh sach cac mon hoc*

*public:*

*GV()*

*{*

*ht[0]= ns= sm= 0 ; mh = NULL;*

*}*

*~GV()*

*{*

*ht[0]= ns= sm= 0 ; if (mh) delete mh;*

*}*

*int getsm()*

*{*

*return sm;*

*}*

*const GV& operator=(const GV& g); void nhap();*

*void xuat();*

*} ;*

*const GV& GV::operator=(const GV& g)*

*{*

*strcpy(this->ht,g.ht); this->ns=g.ns;*

*int n = g.sm; this->sm = n;*

*if (this->mh) delete this->mh; if (n)*

*{*

*this->mh = new MON\_HOC[n+1]; for (int i=1; i<=n; ++i)*

*this->mh[i] = g.mh[i];*

*}*

*return g;*

*}*

*void GV::nhap()*

*{*

*cout << "\nHo ten: " ; fflush(stdin); gets(ht); cout << "Nam sinh: " ; cin >> ns;*

*cout << "So mon co the giang day: " ; cin >> sm;*

*if (this->mh) delete this->mh; if(sm)*

*{*

*this->mh = new MON\_HOC[sm+1];*

*for (int i=1; i<=sm; ++i) this->mh[i].nhap();*

*}*

*}*

*void GV::xuat()*

*{*

*cout << "\nHo ten: " << ht ; cout << "\nNam sinh: " << ns ;*

*cout << "\nSo mon co the giang day: " << sm; if (sm)*

*{*

*cout << "\n Do la: ";*

*for (int i=1; i<=sm; ++i) this->mh[i].xuat();*

*}*

*}*

*class BM // Bo mon*

*{*

*private:*

*char tenbm[20];*

*int n; // So giao vien*

*GV \*gv; // Danh sach giao vien public:*

*BM()*

*{*

*tenbm[0] = n = 0; gv = NULL;*

*}*

*void nhap(); void xuat(); void sapxep();*

*} ;*

*void BM::nhap()*

*{*

*cout << "\n\nTen bo mon: " ; fflush(stdin); gets(tenbm); cout << "So giao vien: ";*

*cin >> n;*

*if (gv) delete gv; if (n)*

*{*

*gv = new GV[n+1];*

*for (int i=1; i<=n; ++i) gv[i].nhap();*

*}*

*}*

*void BM::xuat()*

*{*

*cout << "\nBo mon: " << tenbm; cout << "\nSo giao vien: " << n; if (n)*

*{*

*cout << "\n Danh sach giao vien cua bo mon:"; for (int i=1; i<=n; ++i)*

*gv[i].xuat();*

*}*

*}*

*void BM::sapxep()*

*{*

*GV tg; int i,j; if (n)*

*for (i=1;i<n;++i)*

*for (j=i+1;j<=n;++j)*

*if (gv[i].getsm()<gv[j].getsm())*

*{*

*tg=gv[i]; gv[i]=gv[j]; gv[j]=tg;*

*}*

*}*

*void main()*

*{*

*BM b;*

*b.nhap();*

*b.sapxep();*

*b.xuat();*

*getch();*

*}*

Vấn đề đặt ra là: Hiện nay các giáo viên đã bắt đầu hướng dẫn luận văn tốt nghiệp cho sinh viên. Vì vậy cần bổ sung thông tin này vào phần dữ liệu giáo viên. Để nâng cấp chương trình chúng ta sẽ định nghĩa lớp GV2 dẫn xuất từ lớp GV, sau đó trong lớp BM sẽ thay GV bằng GV2. Có 2 chỗ cần bổ sung và một chỗ cần sửa đổi như sau:

1. Bổ sung trong lớp GV phương thức:

## GV\* getGV()

**{**

## return this;

**}**

Phương thức này dùng để xây dựng toán tử gán cho lớp GV2.

1. Trong lớp BM thay GV bằng GV2. Điều này có thể đạt được bằng sửa chữa trực tiếp hoặc bằng một câu lệnh #define :

## #define GV GV2

1. Định nghĩa thêm 2 lớp: LV (Luận văn) và GV2 (Lớp GV2 dẫn xuất từ lớp GV) như sau:

*class LV // Luan van*

*{*

*private:*

*char tenlv[30]; // Ten luan van char tensv[25]; // Ten sinh vien*

*int nambv; // Nam bao ve luan van public:*

*LV() ; // Hàm tạo*

*const LV& operator=(const LV& l) ; // Gán void nhap() ; // Nhập*

*void xuat() ;*

*} ;*

*class GV2 : public GV*

*{*

*private:*

*int solv; // Số luận văn đã hướng dẫn LV \*lv; // Danh sách luận văn*

*public:*

*GV2(); // Hàm tạo*

*~GV2() ; // Hàm huỷ*

*GV2& operator=(GV2& g); // Gán void nhap(); // Nhập*

*void xuat(); // Xuất*

*} ;*

Chương trình nâng cấp như sau:

*//CT5-12B*

*// Nang cap chuong trinh*

*// CT nang cap #include <conio.h> #include <stdio.h> #include <iostream.h> #include <string.h> class MON\_HOC*

*{*

*private:*

*char tenmh[20]; int sotiet;*

*public:*

*MON\_HOC()*

*{*

*tenmh[0]=sotiet=0;*

*}*

*const MON\_HOC& operator=(const MON\_HOC& m)*

*{*

*strcpy(this->tenmh,m.tenmh); this->sotiet = m.sotiet; return m;*

*}*

*void nhap()*

*{*

*cout << "\nTen mon hoc:"; fflush(stdin); gets(tenmh); cout << "So tiet: " ;*

*cin >> sotiet;*

*}*

*void xuat()*

*{*

*cout << "\nTen mon hoc:" << tenmh*

*<< " so tiet: " << sotiet;*

*}*

*} ;*

*// Bo sung phuong thuc getGV cho lop GV*

*// dung de xay dung toan tu gan cho lop GV2 class GV*

*{*

*private:*

*char ht[25]; // Ho ten int ns; // Nam sinh*

*int sm; // So mon hoc co the day MON\_HOC \*mh ; //Danh sach cac mon hoc*

*public:*

*GV()*

*{*

*ht[0]= ns= sm= 0 ;*

*mh = NULL;*

*}*

*~GV()*

*{*

*ht[0]= ns= sm= 0 ; if (mh) delete mh;*

*}*

*// Bo sung phuong thuc getGV GV\* getGV()*

*{*

*return this;*

*}*

*int getsm()*

*{*

*return sm;*

*}*

*const GV& operator=(const GV& g); void nhap();*

*void xuat();*

*} ;*

*const GV& GV::operator=(const GV& g)*

*{*

*strcpy(this->ht,g.ht); this->ns=g.ns;*

*int n = g.sm; this->sm = n;*

*if (this->mh) delete this->mh; if (n)*

*{*

*this->mh = new MON\_HOC[n+1]; for (int i=1; i<=n; ++i)*

*this->mh[i] = g.mh[i];*

*}*

*return g;*

*}*

*void GV::nhap()*

*{*

*cout << "\nHo ten: " ; fflush(stdin); gets(ht); cout << "Nam sinh: " ;*

*cin >> ns;*

*cout << "So mon co the giang day: " ; cin >> sm;*

*if (this->mh) delete this->mh; if (sm)*

*{*

*this->mh = new MON\_HOC[sm+1]; for (int i=1; i<=sm; ++i)*

*this->mh[i].nhap();*

*}*

*}*

*void GV::xuat()*

*{*

*cout << "\nHo ten: " << ht ; cout << "\nNam sinh: " << ns ;*

*cout << "\nSo mon co the giang day: " << sm; if (sm)*

*{*

*cout << "\n Do la: ";*

*for (int i=1; i<=sm; ++i) this->mh[i].xuat();*

*}*

*}*

*// Bo sung cac lop LV va GV2 class LV // Luan van*

*{*

*private:*

*char tenlv[30]; // Ten luan van char tensv[25]; // Ten sinh vien*

*int nambv; // Nam bao ve luan van public:*

*LV()*

*{*

*tenlv[0]=tensv[0] = nambv = 0;*

*}*

*const LV& operator=(const LV& l)*

*{*

*strcpy(this->tenlv,l.tenlv); strcpy(this->tensv,l.tensv);*

*this->nambv = l.nambv ; return l;*

*}*

*void nhap()*

*{*

*cout << "\nTen luan van:"; fflush(stdin); gets(tenlv); cout << "Ten sinh vien:"; fflush(stdin); gets(tensv); cout << "Nam bao ve: " ; cin >> nambv ;*

*}*

*void xuat()*

*{*

*cout << "\nTen lan van:" << tenlv*

*<< " Sinh vien: " << tensv*

*<< " Nam bao ve: " << nambv;*

*}*

*} ;*

*class GV2 : public GV*

*{*

*private:*

*int solv;*

*LV \*lv;*

*public:*

*GV2():GV()*

*{*

*solv = 0 ; lv = NULL;*

*}*

*~GV2()*

*{*

*if (solv) delete lv;*

*}*

*GV2& operator=(GV2& g); void nhap();*

*void xuat();*

*} ;*

*GV2& GV2::operator=(GV2& g)*

*{*

*GV \*g1, \*g2;*

*g1 = this->getGV(); g2 = g.getGV();*

*\*g1 = \*g2; int n = g.solv;*

*this->solv = n;*

*if (this->lv) delete this->lv; if (n)*

*{*

*this->lv = new LV[n+1]; for (int i=1; i<=n; ++i) this->lv[i] = g.lv[i];*

*}*

*return g;*

*}*

*void GV2::nhap()*

*{*

*GV::nhap();*

*cout << "So luan van da huong dan: " ; cin >> solv;*

*if (this->lv) delete this->lv; if (solv)*

*{*

*this->lv = new LV[solv+1]; for (int i=1; i<=solv; ++i)*

*this->lv[i].nhap();*

*}*

*}*

*void GV2::xuat()*

*{*

*GV::xuat();*

*cout << "\nSo luan van da huong dan: " << solv; if (solv)*

*{*

*cout << "\n Do la: ";*

*for (int i=1; i<=solv; ++i) this->lv[i].xuat();*

*}*

*}*

*// Sua lop BM: thay GV bang GV2 #define GV GV2*

*class BM // Bo mon*

*{*

*private:*

*char tenbm[20];*

*int n; // So giao vien*

*GV \*gv; // Danh sach giao vien public:*

*BM()*

*{*

*tenbm[0] = n = 0; gv = NULL;*

*}*

*void nhap(); void xuat(); void sapxep();*

*} ;*

*void BM::nhap()*

*{*

*cout << "\n\nTen bo mon: " ; fflush(stdin); gets(tenbm); cout << "So giao vien: ";*

*cin >> n;*

*if (gv) delete gv; if (n)*

*{*

*gv = new GV[n+1];*

*for (int i=1; i<=n; ++i) gv[i].nhap();*

*}*

*}*

*void BM::xuat()*

*{*

*cout << "\nBo mon: " << tenbm; cout << "\nSo giao vien: " << n; if (n)*

*{*

*cout << "\n Danh sach giao vien cua bo mon:"; for (int i=1; i<=n; ++i)*

*gv[i].xuat();*

*}*

*}*

*void BM::sapxep()*

*{*

*GV tg; int i,j; if (n)*

*for (i=1;i<n;++i)*

*for (j=i+1;j<=n;++j)*

*if (gv[i].getsm()<gv[j].getsm())*

*{*

*tg=gv[i]; gv[i]=gv[j]; gv[j]=tg;*

*}*

*}*

*#undef GV void main()*

*{*

*BM b;*

*b.nhap();*

*b.sapxep();*

*b.xuat();*

*getch();*

*}*

## Bài 11. Từ khái quát đến cụ thể

Tính thừa kế cũng thường dùng để thiết kế các bài toán theo hướng từ khái quát đến cụ thể, từ chung đến riêng. Đầu tiên đưa ra các lớp để mô tả những đối tượng chung, sau đó dẫn xuất tới các đối tượng ngày một cụ thể hơn.

Một trường hợp khác cũng thường gặp là: Quản lý nhiều thực thể có những phần dữ liệu chung. Khi đó ta có thể xây dựng một lớp cơ sở gồm các phần dữ liệu chung. Mỗi thực thể sẽ được mô tả bằng một lớp dẫn xuất từ lớp cơ sở này.

Sau đây là một số ví dụ minh hoạ:

**Ví dụ 1** (minh hoạ tư tưởng đi từ khái quát đến cụ thể) : Giả sử cần quản lý sinh viên của một trường đại học. Khi đó ta có thể bắt đầu từ lớp SINH\_VIEN (Sinh viên). Sau đó dùng nó làm cơ sở để dẫn xuất tới các lớp mô tả các đối tượng sinh viên cụ thể hơn, ví dụ: SV Tin, SV Toán, SV Luật, SV Du lịch, ...

Các bài toán kiểu như vậy rất thường gặp trong thực tế.

**Ví dụ 2** (minh hoạ phần chung của nhiều thực thể). Giả sử cần xây dựng phần mềm để thực hiện các phép tính về ma trân vuông và véc tơ cấp n. Ta có nhận xét là n chung cho cả véc tơ và ma trận. Hơn nữa nó còn chung cho tất cả các ma trận và véc tơ cùng xét trong bài toán. Vì vậy có thể định

nghĩa một lớp cơ sở chỉ có một thuộc tính tĩnh (static) n. Các lớp ma trận, véc tơ dẫn xuất từ lớp này và sử dụng chung cùng một giá trị n.

Dưới đây là chương trình thực hiện các phép toán ma trận, véc tơ. Chương trình được tổ chức thành 3 lớp:

Lớp CAP (Cấp ma trận, véc tơ) gồm một thành phần tĩnh n và phương thức nhập n.

Lớp VT (Véc tơ) có một thuộc tính là mảng một chiều (chứa các phần tử của véc tơ) và các phương thức nhập, xuất.

Lớp MT (Ma trận) có một thuộc tính là mảng 2 chiều (chứa các phần tử của ma trận) , các phương thức nhập, xuất và nhân. Lớp MT là bạn của lớp VT.

Chương trình sẽ nhập một ma trận, nhập một véc tơ và tính tích của chúng.

*//CT5-13*

*// ma tran vec to*

*// Dùng thuộc tính static #include <conio.h> #include <stdio.h> #include <iostream.h> #include <ctype.h> class CAP;*

*class MT; class VT; class CAP*

*{*

*private:*

*static int n; public:*

*void nhap()*

*{*

*int ch;*

*if (n==0)*

*{*

*cout << "\nN= "; cin >> n;*

*}*

*else*

*{*

*cout <<"\n Hien n = " << n;*

*cout << "\n Co thay doi n? - C/K"; ch=toupper(getch());*

*if (ch=='C')*

*{*

*cout << "\nN= "; cin >> n;*

*}*

*}*

*}*

*int getN()*

*{*

*return n;*

*}*

*} ;*

*int CAP::n=0;*

*class MT : public CAP*

*{*

*private:*

*double a[20][20]; public:*

*void nhap(); void xuat();*

*VT operator\*(VT x);*

*};*

*class VT : public CAP*

*{*

*private:*

*double x[20]; public:*

*friend class MT; void nhap(); void xuat();*

*};*

*void MT::nhap()*

*{*

*int n,i,j;*

*n = this->getN(); if (n==0)*

*{*

*this->CAP::nhap(); n = this->getN();*

*}*

*for (i=1; i<=n; ++i) for (j=1; j<=n; ++j)*

*{*

*cout << " PT hang " << i << " cot " << j << " = "; cin >> a[i][j];*

*}*

*}*

*void MT::xuat()*

*{*

*int n,i,j;*

*n = this->getN(); if (n)*

*for (int i=1; i<=n; ++i)*

*{*

*cout << "\n" ;*

*for (int j=1; j<=n; ++j) cout << a[i][j] << " ";*

*}*

*}*

*VT MT::operator\*(VT x)*

*{*

*VT y;*

*int n,i,j;*

*n = this->getN();*

*for (i=1; i<=n; ++i)*

*{*

*y.x[i]=0;*

*for (j=1; j<=n; ++j) y.x[i] += a[i][j]\*x.x[j];*

*}*

*return y;*

*}*

*void VT::nhap()*

*{*

*int n,i;*

*n = this->getN(); if (n==0)*

*{*

*this->CAP::nhap(); n = this->getN();*

*}*

*for (i=1; i<=n; ++i)*

*{*

*cout << " PT thu " << i << " = "; cin >> x[i];*

*}*

*}*

*void VT::xuat()*

*{*

*int n,i;*

*n = this->getN(); if (n)*

*{*

*cout << "\n";*

*for (int i=1; i<=n; ++i)*

*{*

*cout << x[i] << " ";*

*}*

*}*

*}*

*void main()*

*{*

*MT a; VT x,y; clrscr();*

*cout<<"\nNhap ma tran A:"; a.nhap();*

*cout<<"\n\nNhap Vec to X:\n"; x.nhap();*

*y = a\*x; cout<<"\n\nMa tran A"; a.xuat(); cout<<"\n\nVec to X"; x.xuat();*

*cout<<"\n\nVec to Y=AX"; y.xuat();*

*getch();*

*}*

## Bài 12. Toàn thể và bộ phận

Thông thường khi xem xét, giải quyết một bài toán, ta thường chia nó thành các bài toán nhỏ hơn. Nói cách khác: Một bài toán lớn bao gồm nhiều bài toán bộ phận. Khi đó ta có thể định nghĩa các lớp cho các bài toán bộ phận. Lớp cho bài toán chung được dẫn xuất từ các lớp nói trên.

Xét một thí dụ đơn giản là bài toán quản lý thư viện. Nó gồm 2 bài toán bộ phận là quản lý sách và quản lý đọc giả. Chúng ta sẽ xây dựng lớp SACH và lớp DOC\_GIA. Sau đó dùng các lớp này làm cơ sở để xây dựng lớp THU\_VIEN

**Chương 6**

# Tương ứng bội và phương thức ảo

Tương ứng bội và phương thức ảo là công cụ mạnh của C++ cho phép tổ chức quản lý các đối tượng khác nhau theo cùng một lược đồ. Một khái niệm khác liên quan là: lớp cơ sở trừu tượng. Chương này sẽ trình bầy cách sử dụng các công cụ trên để xây dựng chương trình quản lý nhiều đối tượng khác nhau theo một lược đồ thống nhất.

## Bài 1. Phương thức tĩnh

* 1. **Lời gọi tới phương thức tĩnh**

Như đã biết một lớp dẫn xuất được thừa kế các phương thức của các lớp cơ sở tiền bối của nó. Ví dụ lớp A là cơ sở của B, lớp B lại là cơ sở của C, thì C có 2 lớp cơ sở tiền bối là B và A. Lớp C được thừa kế các phương thức của A và B. Các phương thức mà chúng ta vẫn nói là các phương thức tĩnh. Để tìm hiểu thêm về cách gọi tới các phương thức tĩnh, ta xét ví dụ về các lớp A, B và C như sau:

*class A*

*{*

*public:*

*void xuat()*

*{*

*cout << "\n Lop A " ;*

*}*

*};*

*class B:public A*

*{*

*public:*

*void xuat()*

*{*

*cout << "\n Lop B " ;*

*}*

*};*

*class C:public B*

*{*

*public:*

*void xuat()*

*{*

*cout << "\n Lop C " ;*

*}*

*};*

Lớp C có 2 lớp cơ sở tiền bối là A , B và C kế thừa các phương thức của A và B. Do đó một đối tượng của C sẽ có tới 3 phương thức xuat. Hãy theo rõi các câu lệnh sau:

C h ; // h là đối tượng kiểu C

h.xuat() ; // Gọi tới phương thức h.D::xuat()

h.B::xuat() ; // Gọi tới phương thức h.B::xuat() h.A::xuat() ; // Gọi tới phương thức h.A::xuat()

Các lời gọi phương thức trong ví dụ trên đều xuất phát từ đối tượng h và mọi lời gọi đều xác định rõ phương thức cần gọi.

Bây giờ chúng ta hãy xét các lời gọi không phải từ một biến đối tượng mà từ một con trỏ. Xét các câu lệnh:

A \*p, \*q, \*r; // p, q, r là con trỏ kiểu A A a; // a là đối tượng kiểu A

B b; // b là đối tượng kiểu B C c; // c là đối tượng kiểu c

**Chúng ta hãy ghi nhớ mệnh đề sau** về con trỏ của các lớp dẫn xuất và cơ sở:

**Phép gán con trỏ:** Con trỏ của lớp cơ sở có thể dùng để chứa địa chỉ các đối tượng của lớp dẫn xuất.

Như vậy cả 3 phép gán sau đều hợp lệ: p = &a ;

q = &b ; r = &c ;

Chúng ta tiếp tục xét các lời gọi phương thức từ các con trỏ p, q, r: p->xuat();

1. >xuat();
2. >xuat();

và hãy lý giải xem phương thức nào (trong các phương thức A::xuat, B::xuat và C::xuat) được gọi. Câu trả lời như sau:

Cả 3 câu lệnh trên đều gọi tới phương thức A::xuat() , vì các con trỏ p, q và r đều có kiểu A. Như vậy có thể tóm lược cách thức gọi các phương thức tĩnh như sau:

**Quy tắc gọi phương thức tĩnh:** Lời gọi tới phương thức tĩnh bao giờ cũng xác định rõ phương thức nào (trong số các phương thức trùng tên của các lớp có quan hệ thừa kế) được gọi:

* + 1. Nếu lời gọi xuất phát từ một đối tượng của lớp nào, thì phương thức của lớp đó sẽ được gọi.
    2. Nếu lời gọi xuất phát từ một con trỏ kiểu lớp nào, thì phương thức của lớp đó sẽ được gọi bất kể con trỏ chứa địa chỉ của đối tượng nào.

## Ví dụ

Xét 4 lớp A, B, C và D. Lớp B và C có chung lớp cơ sở A. Lớp D dẫn xuất từ C. Cả 4 lớp đều có phương thức xuat(). Xét hàm:

void hien(A \*p)

{

p->xuat();

}

Không cần biết tới địa chỉ của đối tượng nào sẽ truyền cho đối con trỏ p, lời gọi trong hàm luôn luôn gọi tới phương thức A::xuat() vì con trỏ p kiểu A. Như vậy bốn câu lệnh:

hien(&a);

hien(&b);

hien(&c);

hien(&d);

trong hàm main (của chương trình dưới đây) đều gọi tới A::xuat().

*//CT6-01*

*// Phuong thuc tinh #include <conio.h> #include <stdio.h> #include <iostream.h> #include <ctype.h> class A*

*{*

*private:*

*int n; public:*

*A()*

*{*

*n=0;*

*}*

*A(int n1)*

*{*

*n=n1;*

*}*

*void xuat()*

*{*

*cout << "\nLop A: "<< n;*

*}*

*int getN()*

*{*

*return n;*

*}*

*};*

*class B:public A*

*{*

*public:*

*B():A()*

*{*

*}*

*B(int n1):A(n1)*

*{*

*}*

*void xuat()*

*{*

*cout << "\nLop B: "<<getN();*

*}*

*};*

*class C:public A*

*{*

*public:*

*C():A()*

*{*

*}*

*C(int n1):A(n1)*

*{*

*}*

*void xuat()*

*{*

*cout << "\nLop C: "<<getN();*

*}*

*};*

*class D:public C*

*{*

*public:*

*D():C()*

*{*

*}*

*D(int n1):C(n1)*

*{*

*}*

*void xuat()*

*{*

*cout << "\nLop D: "<<getN();*

*}*

*};*

*void hien(A \*p)*

*{*

*p->xuat();*

*}*

*void main()*

*{*

*A a(1);*

*B b(2);*

*C c(3);*

*D d(4);*

*clrscr();*

*hien(&a);*

*hien(&b);*

*hien(&c);*

*hien(&d);*

*getch();*

*}*

## Bài 2. Sự hạn chế của phương thức tĩnh

Ví dụ sau cho thấy sự hạn chế của phương thức tĩnh trong việc sử dụng tính thừa kế để phát triển chương trình.

Giả sử cần xây dựng chương trình quản lý thí sinh. Mỗi thí sinh đưa vào ba thuộc tính: Họ tên, số báo danh và tổng điểm. Chương trình gồm ba chức năng: Nhập dữ liệu thí sinh, in dữ liệu thí sinh ra máy in và xem - in (in họ tên ra màn hình, sau đó lựa chọn hoặc in hoặc không). Chương trình dưới đây sử dụng lớp TS (Thí sinh) đáp ứng được yêu cầu đặt ra.

*//CT6-02*

*// Han che phuong thuc tinh*

*// Lop TS*

*#include <conio.h> #include <stdio.h> #include <iostream.h> #include <ctype.h> class TS*

*{*

*private:*

*char ht[25]; int sobd; float td;*

*public:*

*void nhap()*

*{*

*cout << "\nHo ten: " ; fflush(stdin); gets(ht); cout << "So bao danh: " ; cin >> sobd;*

*cout << "Tong diem: " ; cin >> td;*

*}*

*void in()*

*{*

*fprintf(stdprn,"\n\nHo ten: %s", ht); fprintf(stdprn,"\nSo bao danh: %d", sobd); fprintf(stdprn,"\nTong diem: %0.1f", td);*

*}*

*void xem\_in()*

*{*

*int ch;*

*cout << "\nHo ten: " << ht ; cout << "\nCo in khong? - C/K" ; ch = toupper(getch());*

*if (ch=='C')*

*this->in();*

*}*

*} ;*

*void main()*

*{*

*TS t[100];*

*int i, n;*

*cout << "\nSo thi sinh: "; cin >> n;*

*for (i=1; i<=n; ++i) t[i].nhap();*

*for (i=1; i<=n; ++i) t[i].xem\_in();*

*getch();*

*}*

Giả sử Nhà trường muốn quản lý thêm địa chỉ của thí sinh. Vì sự thay đổi ở đây là không nhiều, nên chúng ta không đả động đến lớp TS mà xây dựng lớp mới TS2 dẫn xuất từ lớp TS. Trong lớp TS2 đưa thêm thuộc tính dc (địa chỉ) và các phương thức nhap, in. Cụ thể lớp TS2 được định nghĩa như sau:

*class TS2:public TS*

*{*

*private:*

*char dc[30] ; // Dia chi public:*

*void nhap()*

*{*

*TS::nhap();*

*cout << "Dia chi: " ; fflush(stdin); gets(dc);*

*}*

*void in()*

*{*

*TS::in();*

*fprintf(stdprn,"\nDia chi: %s", dc);*

*}*

*};*

Trong lớp TS2 không xây dựng lại phương thức xem\_in, mà sẽ dùng phương thức xem\_in của lớp TS. Chương trình mới như sau:

*//CT6-03*

*// Han che phuong thuc tinh*

*// Lop TS TS2 #include <conio.h> #include <stdio.h>*

*#include <iostream.h> #include <ctype.h> class TS*

*{*

*private:*

*char ht[25]; int sobd; float td;*

*public:*

*void nhap()*

*{*

*cout << "\nHo ten: " ; fflush(stdin); gets(ht); cout << "So bao danh: " ; cin >> sobd;*

*cout << "Tong diem: " ; cin >> td;*

*}*

*void in()*

*{*

*fprintf(stdprn,"\n\nHo ten: %s", ht); fprintf(stdprn,"\nSo bao danh: %d", sobd); fprintf(stdprn,"\nTong diem: %0.1f", td);*

*}*

*void xem\_in()*

*{*

*int ch;*

*cout << "\nHo ten: " << ht ; cout << "\nCo in khong? - C/K" ; ch = toupper(getch());*

*if (ch=='C')*

*this->in(); //Goi den TS::in() (Vi this la con tro*

*//kieu TS)*

*}*

*} ;*

*class TS2:public TS*

*{*

*private:*

*char dc[30] ; // Dia chi public:*

*void nhap()*

*{*

*TS::nhap();*

*cout << "Dia chi: " ; fflush(stdin); gets(dc);*

*}*

*void in()*

*{*

*TS::in();*

*fprintf(stdprn,"\nDia chi: %s", dc);*

*}*

*};*

*void main()*

*{*

*TS2 t[100];*

*int i, n;*

*cout << "\nSo thi sinh: "; cin >> n;*

*for (i=1; i<=n; ++i) t[i].nhap();*

*for (i=1; i<=n; ++i) t[i].xem\_in();*

*getch();*

*}*

Khi thực hiện chương trình này, chúng ta nhận thấy: Dữ liệu in ra vẫn không có địa chỉ. Điều này có thể giải thích như sau:

Xét câu lệnh (thứ 2 từ dưới lên trong hàm main):

t[i].xem\_in() ;

Câu lệnh này gọi tới phương thức xem\_in của lớp TS2 (vì t[i] là đối tượng của lớp TS2). Nhưng lớp TS2 không định nghĩa phương thức xem\_in, nên phương thức TS::xem\_in() sẽ được gọi tới. Hãy theo rõi phương thức này:

*void xem\_in()*

*{*

*int ch;*

*cout << "\nHo ten: " << ht ; cout << "\nCo in khong? - C/K" ; ch = toupper(getch()); if(ch=='C')*

*this->in(); //Goi den TS::in() (Vi* ***this*** *la con tro kieu TS)*

*}*

Các lệnh đầu của phương thức sẽ in họ tên thí sinh. Nếu chọn có (bấm phím C), thì câu lệnh: this->in() ;

sẽ được thực hiện. Mặc dù địa chỉ của t[i] (là đối tượng của lớp TS2) được truyền cho con trỏ this, thế nhưng câu lệnh này luôn luôn gọi tới phương thức TS::in(), vì con trỏ this ở đây có kiểu TS và vì in() là phương thức tĩnh. Kết quả là không in được địa chỉ của thí sinh.

Như vậy việc sử dụng các phương thức tĩnh in() (trong các lớp TS và TS2) đã không đáp ứng được yêu cầu phát triển chương trình. Có một giải pháp rất đơn giản là: Định nghĩa các phương thức in() trong các lớp TS và TS2 như các phương thức ảo (virtual).

## Bài 3. Phương thức ảo và tương ứng bội

* 1. **Cách định nghĩa phương thức ảo**

Giả sử A là lớp cơ sở, các lớp B, C, D dẫn xuất (trực tiếp hoặc dán tiếp) từ A. Giả sử trong 4 lớp trên đều có các phương thức trùng dòng tiêu đề (trùng kiểu, trùng tên, trùng các đối). Để định nghĩa các phương thức này là các phương thức ảo, ta chỉ cần:

+ Hoặc thêm từ khoá virtual vào dòng tiêu đề của phương thức bên trong định nghĩa lớp cơ sở A.

+ Hoặc thêm từ khoá virtual vào dòng tiêu đề bên trong định nghĩa của tất cả các lớp A, B, C và

D.

**Ví dụ:** *Cách 1: class A*

*{*

*...*

*virtual void hien\_thi()*

*{*

*cout << “\n Đây là lớp A” ;*

*};*

*} ;*

*class B : public A*

*{*

*...*

*void hien\_thi()*

*{*

*cout << “\n Đây là lớp B” ;*

*};*

*} ;*

*class C : public B*

*{*

*...*

*void hien\_thi()*

*{*

*cout << “\n Đây là lớp C” ;*

*};*

*} ;*

*class D : public A*

*{*

*...*

*void hien\_thi()*

*{*

*cout << “\n Đây là lớp D” ;*

*};*

*} ;*

*Cách 2: class A*

*{*

*...*

*virtual void hien\_thi()*

*{*

*cout << “\n Đây là lớp A” ;*

*};*

*} ;*

*class B : public A*

*{*

*...*

*virtual void hien\_thi()*

*{*

*cout << “\n Đây là lớp B” ;*

*};*

*} ;*

*class C : public B*

*{*

*...*

*virtual void hien\_thi()*

*{*

*cout << “\n Đây là lớp C” ;*

*};*

*} ;*

*class D : public A*

*{*

*...*

*virtual void hien\_thi()*

*{*

*cout << “\n Đây là lớp D” ;*

*};*

*} ;*

***Chú ý:*** Từ khoá virtual không được đặt bên ngoài định nghĩa lớp. Ví dụ nếu viết như sau là sai (CTBD sẽ báo lỗi).

*class A*

*{*

*...*

*virtual void hien\_thi() ;*

*} ;*

*virtual void hien\_thi() // Sai*

*{*

*cout << “\n Đây là lớp A” ;*

*};*

*Cần sửa lại như sau: class A*

*{*

*...*

*virtual void hien\_thi() ;*

*} ;*

*void hien\_thi() // Đúng*

*{*

*cout << “\n Đây là lớp A” ;*

*};*

## Quy tắc gọi phương thức ảo

Để có sự so sánh với phương thức tĩnh, ta nhắc lại quy tắc gọi phương thức tĩnh nêu trong §1.

## Quy tắc gọi phương thức tĩnh

Lời gọi tới phương thức tĩnh bao giờ cũng xác định rõ phương thức nào (trong số các phương thức trùng tên của các lớp có quan hệ thừa kế) được gọi:

1. Nếu lời gọi xuất phát từ một đối tượng của lớp nào, thì phương thức của lớp đó sẽ được gọi.
2. Nếu lời gọi xuất phát từ một con trỏ kiểu lớp nào, thì phương thức của lớp đó sẽ được gọi bất kể con trỏ chứa địa chỉ của đối tượng nào.

## Quy tắc gọi phương thức ảo

Phương thức ảo chỉ khác phương thức tĩnh khi được gọi từ một con trỏ (trường hợp 2 nêu trong mục 3.2.1). Lời gọi tới phương thức ảo từ một con trỏ chưa cho biết rõ phương thức nào (trong số các phương thức ảo trùng tên của các lớp có quan hệ thừa kế) sẽ được gọi. Điều này phụ thuộc vào đối tượng cụ thể mà con trỏ đang trỏ tới: Con trỏ đang trỏ tới đối tượng của lớp nào thì phương thức của lớp đó sẽ được gọi.

**Ví dụ** A, B, C và D là các lớp đã định nghĩa trong 3.1. Ta khai báo một con trỏ kiểu A và 4 đối tượng:

A \*p ; // p là con trỏ kiểu A

A a ; // a là biến đối tượng kiểu A B b ; // b là biến đối tượng kiểu B C c ; // c là biến đối tượng kiểu C D d ; // d là biến đối tượng kiểu D

Xét lời gọi tới các phương thức ảo hien\_thi sau:

p = &a; // p trỏ tới đối tượng a của lớp A p->hien\_thi() ; // Gọi tới A::hien\_thi()

p = &b; // p trỏ tới đối tượng b của lớp B p->hien\_thi() ; // Gọi tới B::hien\_thi()

p = &c; // p trỏ tới đối tượng c của lớp C p->hien\_thi() ; // Gọi tới C::hien\_thi()

p = &d; // p trỏ tới đối tượng d của lớp D p->hien\_thi() ; // Gọi tới D::hien\_thi()

## Tương ứng bội

Chúng ta nhận thấy cùng một câu lệnh p->hien\_thi();

tương ứng với nhiều phương thức khác nhau. Đây chính là tương ứng bội. Khả năng này rõ ràng cho phép xử lý nhiều đối tượng khác nhau, nhiều công việc, thậm chí nhiều thuật toán khác nhau theo cùng một cách thức, cùng một lược đồ. Điều này sẽ được minh hoạ trong các mục tiếp theo.

## Liên kết động

Có thể so sánh sự khác nhau giữ phương thức tĩnh và phương thức ảo trên khía cạnh liên kết một lời gọi với một phương thức. Trở lại ví dụ trong 3.2:

A \*p ; // p là con trỏ kiểu A

A a ; // a là biến đối tượng kiểu A B b ; // b là biến đối tượng kiểu B C c ; // c là biến đối tượng kiểu C D d ; // d là biến đối tượng kiểu D

Nếu hien\_thi() là các phương thức tĩnh, thì dù p chứa địa chỉ của các đối tượng a, b, c hay d, thì lời gọi:

p->hien\_thi() ;

luôn luôn gọi tới phương thức A::hien\_thi()

Như vậy một lời gọi (xuất phát từ con trỏ) tới phương thức tĩnh luôn luôn liên kết với một phương thức cố định và sự liên kết này xác định trong quá trình biên dịch chương trình.

Cũng với lời gọi: p->hien\_thi() ;

như trên, nhưng nếu hien\_thi() là các phương thức ảo, thì lời gọi này không liên kết cứng với một phương thức cụ thể nào. Phương thức mà nó liên kết (gọi tới) còn chưa xác định trong giai đoạn dịch chương trình. Lời gọi này sẽ:

+ liên kết với A::hien\_thi() , nếu p chứa địa chỉ đối tượng lớp A

+ liên kết với B::hien\_thi() , nếu p chứa địa chỉ đối tượng lớp B

+ liên kết với C::hien\_thi() , nếu p chứa địa chỉ đối tượng lớp C

+ liên kết với D::hien\_thi() , nếu p chứa địa chỉ đối tượng lớp D

Như vậy một lời gọi (xuất phát từ con trỏ) tới phương thức ảo không liên kết với một phương thức cố định, mà tuỳ thuộc vào nội dung con trỏ. Đó là sự liên kết động và phương thức được liên kết (được gọi) thay đổi mỗi khi có sự thay đổi nội dung con trỏ trong quá trình chạy chương trình.

## Quy tắc gán địa chỉ đối tượng cho con trỏ lớp cơ sở

+ Như đã nói trong §1, C++ cho phép gán địa chỉ đối tượng của một lớp dẫn xuất cho con trỏ của lớp cơ sở. Như vậy các phép gán sau (xem 3.2) là đúng:

A \*p ; // p là con trỏ kiểu A

1. a ; // a là biến đối tượng kiểu A B b ; // b là biến đối tượng kiểu B C c ; // c là biến đối tượng kiểu C D d ; // d là biến đối tượng kiểu D p = &a; // p và a cùng lớp A

p = &b; // p là con trỏ lớp cơ sở, b là đối tượng lớp dẫn xuất p = &c; // p là con trỏ lớp cơ sở, c là đối tượng lớp dẫn xuất p = &d; // p là con trỏ lớp cơ sở, d là đối tượng lớp dẫn xuất

**+ *Tuy nhiên cần chú ý là:*** Không cho phép gán địa chỉ đối tượng của lớp cở sở cho con trỏ của lớp dẫn xuất. Như vậy ví dụ sau là sai:

1. \*q ; A a ;

q = &a;

Sai vì: Gán địa chỉ đối tượng của lớp cơ sở A cho con trỏ của lớp dẫn xuất B

## Ví dụ

Ta sửa chương trình trong bài 1 bằng cách định nghĩa các phương thức xuat() là ảo. Khi đó bốn câu lệnh:

hien(&a);

hien(&b);

hien(&c);

hien(&d);

trong hàm main (của chương trình dưới đây) sẽ lần lượt gọi tới 4 phương thức khác nhau: A::xuat()

B::xuat()

C::xuat()

D::xuat()

//CT6-01B

// Phuong thuc ảo và tương ứng bội

*#include <conio.h> #include <stdio.h> #include <iostream.h> #include <ctype.h> class A*

*{*

*private:*

*int n; public:*

*A()*

*{*

*n=0;*

*}*

*A(int n1)*

*{*

*n=n1;*

*}*

*virtual void xuat()*

*{*

*cout << "\nLop A: "<< n;*

*}*

*int getN()*

*{*

*return n;*

*}*

*};*

*class B:public A*

*{*

*public:*

*B():A()*

*{*

*}*

*B(int n1):A(n1)*

*{*

*}*

*void xuat()*

*{*

*cout << "\nLop B: "<<getN();*

*}*

*};*

*class C:public A*

*{*

*public:*

*C():A()*

*{*

*}*

*C(int n1):A(n1)*

*{*

*}*

*void xuat()*

*{*

*cout << "\nLop C: "<<getN();*

*}*

*};*

*class D:public C*

*{*

*public:*

*D():C()*

*{*

*}*

*D(int n1):C(n1)*

*{*

*}*

*void xuat()*

*{*

*cout << "\nLop D: "<<getN();*

*}*

*};*

*void hien(A \*p)*

*{*

*p->xuat();*

*}*

*void main()*

*{*

*A a(1);*

*B b(2);*

*C c(3);*

*D d(4);*

*clrscr();*

*hien(&a);*

*hien(&b);*

*hien(&c);*

*hien(&d);*

*getch();*

*}*

## 3.5. Sự thừa kế của các phương thức ảo

Cũng giống như các phương thức thông thường khác, phương thức ảo cũng có tính thừa kế.

Chẳng hạn trong chương trình trên (mục 3.4) ta bỏ đi phương thức xuat() của lớp D, thì câu lệnh: hien(&d) ;

(câu lệnh gần cuối trong hàm main) sẽ gọi tới C::xuat() , phương thức này được kế thừa trong lớp D (vì D dẫn xuất từ C).

## Bài 4. Sự linh hoạt của phương thức ảo trong phát triển nâng cấp chương trình

Ví dụ về các lớp TS và TS2 trong §2 đã chỉ ra sự hạn chế của phương thức tĩnh trong việc sử dụng tính thừa kế để nâng cấp, phát triển chương trình. Trong §2 cũng đã chỉ ra lớp TS2 chưa đáp ứng được yêu cầu nêu ra là in địa chỉ của thí sinh. Giải pháp cho vấn đề này rất đơn giản: Thay các phương thức tĩnh in() bằng cách dùng chúng như các phương thức ảo. Chương trình khi đó sẽ như sau:

*//CT6-03B*

*// Sự linh hoạt của phương thức ảo*

*// Lop TS TS2 #include <conio.h> #include <stdio.h>*

*#include <iostream.h> #include <ctype.h> class TS*

*{*

*private:*

*char ht[25]; int sobd; float td;*

*public:*

*void nhap()*

*{*

*cout << "\nHo ten: " ; fflush(stdin); gets(ht); cout << "So bao danh: " ; cin >> sobd;*

*cout << "Tong diem: " ; cin >> td;*

*}*

*virtual void in()*

*{*

*fprintf(stdprn,"\n\nHo ten: %s", ht); fprintf(stdprn,"\nSo bao danh: %d", sobd); fprintf(stdprn,"\nTong diem: %0.1f", td);*

*}*

*void xem\_in()*

*{*

*int ch;*

*cout << "\nHo ten: " << ht ; cout << "\nCo in khong? - C/K" ; ch = toupper(getch());*

*if (ch=='C')*

*this->in(); // Vì in() là phương thức ảo nên*

*//có thể gọi đến TS::in() hoặc TS2::in()*

*}*

*} ;*

*class TS2:public TS*

*{*

*private:*

*char dc[30] ; // Dia chi public:*

*void nhap()*

*{*

*TS::nhap();*

*cout << "Dia chi: " ; fflush(stdin); gets(dc);*

*}*

*void in()*

*{*

*TS::in();*

*fprintf(stdprn,"\nDia chi: %s", dc);*

*}*

*};*

*void main()*

*{*

*TS2 t[100];*

*int i, n;*

*cout << "\nSo thi sinh: "; cin >> n;*

*for (i=1; i<=n; ++i)*

*t[i].nhap();*

*for (i=1; i<=n; ++i) t[i].xem\_in(); getch();*

*}*

Khi thực hiện chương trình này, chúng ta nhận thấy: Dữ liệu thí sinh in ra đã có địa chỉ. Điều này có thể giải thích như sau:

Xét câu lệnh (thứ 2 từ dưới lên trong hàm main): t[i].xem\_in() ;

Câu lệnh này gọi tới phương thức xem\_in của lớp TS2 (vì t[i] là đối tượng của lớp TS2). Nhưng lớp TS2 không định nghĩa phương thức xem\_in, nên phương thức TS::xem\_in() sẽ được gọi tới. Hãy theo rõi phương thức này:

*void xem\_in()*

*{*

*int ch;*

*cout << "\nHo ten: " << ht ; cout << "\nCo in khong? - C/K" ; ch = toupper(getch());*

*this->in(); // Vì in() là phương thức ảo nên*

*//có thể gọi đến TS::in() hoặc TS2::in()*

*}*

Các lệnh đầu của phương thức sẽ in họ tên thí sinh. Nếu chọn Có (bấm phím C), thì câu lệnh: this->in() ;

sẽ được thực hiện. Địa chỉ của t[i] (là đối tượng của lớp TS2) được truyền cho con trỏ this (của lớp cơ sở TS). Vì in() là phương thức ảo và vì this đang trỏ tới đối tượng t[i] của lớp TS2, nên câu lệnh này gọi tới phương thức TS2::in(). Trong phương thức TS2::in() có in địa chỉ của thí sinh.

Như vậy việc sử dụng các phương thức tĩnh in() (trong các lớp TS và TS2) đã không đáp ứng được yêu cầu phát triển chương trình. Có một giải pháp rất đơn giản là: Định nghĩa các phương thức in() trong các lớp TS và TS2 như các phương thức ảo (virtual).

## Bài 5. Lớp cơ sở trừu tượng

* 1. **Lớp cơ sở trừu tượng**

Một lớp cơ sở trừu tượng là một lớp chỉ được dùng làm cơ sở cho các lớp khác. Không hề có đối tượng nào của một lớp trừu tượng được tạo ra cả, bởi vì nó chỉ được dùng để định nghĩa một số khái niệm tổng quát, chung cho các lớp khác. Một ví dụ về lớp trừu tượng là lớp CON\_VAT (con vật), nó sẽ dùng làm cơ sở để xây dựng các lớp con vật cụ thể như lớp CON\_CHO (con chó), CON\_MEO (con mèo),... (xem ví dụ bên dưới)

Trong C++ , thuật ngữ “Lớp trừu tượng” đặc biệt áp dụng cho các lớp có chứa các phương thức ảo thuần tuý. Phương thức ảo thuần tuý là một phương thức ảo mà nội dung của nó không có gì. Cách thức định nghĩa một phương thức ảo thuần tuý như sau:

virtual void tên\_phương\_thức() = 0 ;

## Ví dụ:

*class A*

*{*

*public:*

*virtual void nhap() = 0 ; virtual void xuat() = 0 ; void chuong();*

*} ;*

Trong ví dụ trên, thì A là lớp cơ sở trừu tượng. Các phương thức nhap và xuat được khai báo là các lớp ảo thuần tuý (bằng cách gán số 0 cho chúng thay cho việc cài đặt các phương thức này). Phương thức chuong() là một phương thức bình thường và sẽ phải có một định nghĩa ở đâu đó cho phương thức này.

Không có đối tượng nào của một lớp trừu tượng lại có thể được phát sinh. Tuy nhiên các con trỏ và các biến tham chiếu đến các đối tượng của lớp trừu tượng thì vẫn hợp lệ. Bất kỳ lớp nào dẫn xuất từ một lớp cớ sở trừu tượng phải định nghĩa lại tất cả các phương thức thuần ảo mà nó thừa hưởng, hoặc bằng các phương thức ảo thuần tuý, hoặc bằng những định nghĩa thực sự. Ví dụ:

*class B : public A*

*{*

*public:*

*virtual void nhap() = 0 ; virtual void xuat()*

*{*

*// Các câu lệnh*

*}*

*} ;*

Theo ý nghĩa về hướng đối tượng, ta vẫn có thể có một lớp trừu tượng mà không nhất thiết phải chứa đựng những phương thức thuần tuý ảo.

Một cách tổng quát mà nói thì bất kỳ lớp nào mà nó chỉ được dùng làm cơ sở cho những lớp khác đều có thể được gọi là “lớp trừu tượng”. Một cách dễ dàng để nhận biết một lớp trừu tượng là xem có dùng lớp đó để khai báo các đối tượng hay không? . Nếu không thì đó là lớp cơ sở trừu tượng.

## Ví dụ

Giả sử có 20 ô, mỗi ô có thể nuôi một con chó hoặc một con mèo. Yêu cầu xây dựng chương trình gồm các chức năng:

+ Nhập một con vật mới mua (hoặc chó, hoặc mèo) vào ô rỗng đầu tiên.

+ Xuất (đem bán) một con vật (hoặc chó, hoặc mèo).

+ Thống kê các con vật đang nuôi trong 20 ô. Chương trình được tổ chức như sau:

+ Trước tiên định nghĩa lớp CON\_VAT là lớp cơ sở ảo. Lớp này có một thuộc tính là tên con vật và một phương thức ảo dùng để xưng tên.

+ Hai lớp là CON\_MEO và CON\_CHO được dẫn xuất từ lớp CON\_VAT

+ Cuối cùng là lớp DS\_CON\_VAT (Danh sách con vật) dùng để quản lý chung cả mèo và chó. Lớp này có 3 thuộc tính là: số con vật cực đại (chính bằng số ô), số con vật đang nuôi và một mảng con trỏ kiểu CON\_VAT. Mỗi phần tử mảng sẽ chứa địa chỉ của một đối tượng kiểu CON\_MEO hoặc CON\_CHO.

Lớp sẽ có 3 phương thức để thực hiện 3 chức năng nêu trên của chương trình. Nội dung chương trình như sau:

*//CT6-04*

*// Lop co so truu tuong*

*// Lop CON\_VAT #include <conio.h> #include <stdio.h> #include <iostream.h> #include <ctype.h> #include <string.h> class CON\_VAT*

*{*

*protected: char \*ten;*

*public:*

*CON\_VAT()*

*{*

*ten = NULL;*

*}*

*CON\_VAT(char \*ten1)*

*{*

*ten = strdup(ten1);*

*}*

*virtual void xung\_ten()*

*{*

*}*

*} ;*

*class CON\_MEO:public CON\_VAT*

*{*

*public:*

*CON\_MEO() : CON\_VAT()*

*{*

*}*

*CON\_MEO(char \*ten1) : CON\_VAT(ten1)*

*{*

*}*

*virtual void xung\_ten()*

*{*

*cout << "\nToi la chu meo: " << ten ;*

*}*

*};*

*class CON\_CHO:public CON\_VAT*

*{*

*public:*

*CON\_CHO() : CON\_VAT()*

*{*

*}*

*CON\_CHO(char \*ten1) : CON\_VAT(ten1)*

*{*

*}*

*virtual void xung\_ten()*

*{*

*cout << "\nToi la chu cho: " << ten ;*

*}*

*};*

*class DS\_CON\_VAT // Danh sach con vat*

*{*

*private:*

*int max\_so\_con\_vat; int so\_con\_vat; CON\_VAT \*\*h ;*

*public:*

*DS\_CON\_VAT(int max);*

*~DS\_CON\_VAT();*

*int nhap(CON\_VAT \*c); CON\_VAT\* xuat(int n); void thong\_ke();*

*} ;*

*DS\_CON\_VAT::DS\_CON\_VAT(int max)*

*{*

*max\_so\_con\_vat = max; so\_con\_vat = 0;*

*h = new CON\_VAT\*[max]; for (int i=0; i<max; ++i) h[i] = NULL;*

*}*

*DS\_CON\_VAT::~DS\_CON\_VAT()*

*{*

*max\_so\_con\_vat = 0;*

*so\_con\_vat = 0; delete h;*

*}*

*int DS\_CON\_VAT::nhap(CON\_VAT \*c)*

*{*

*if (so\_con\_vat==max\_so\_con\_vat) return 0;*

*int i=0;*

*while (h[i]!=NULL) ++i; h[i]=c;*

*so\_con\_vat++ ; return (i+1);*

*}*

*CON\_VAT\* DS\_CON\_VAT::xuat(int n)*

*{*

*if (n<1 || n > max\_so\_con\_vat) return NULL ;*

*--n ;*

*if (h[n])*

*{*

*CON\_VAT \*c = h[n]; h[n]=NULL;*

*so\_con\_vat-- ; return c;*

*}*

*else*

*return NULL;*

*}*

*void DS\_CON\_VAT::thong\_ke()*

*{*

*if (so\_con\_vat)*

*{*

*cout << "\n" ;*

*for (int i=0; i<max\_so\_con\_vat; ++i) if (h[i])*

*h[i]->xung\_ten();*

*}*

*}*

*CON\_CHO c1("MUC"); CON\_CHO c2("VEN"); CON\_CHO c3("LAI"); CON\_CHO c4("NHAT"); CON\_CHO c5("BONG");*

*CON\_MEO m1("MUOP"); CON\_MEO m2("DEN"); CON\_MEO m3("TRANG"); CON\_MEO m4("TAM THE"); CON\_MEO m5("VANG");*

*void main()*

*{*

*DS\_CON\_VAT d(20);*

*clrscr();*

*d.nhap(&c1);*

*int im2 = d.nhap(&m2); d.nhap(&c3);*

*d.nhap(&m1);*

*int ic4 = d.nhap(&c4); d.nhap(&c5);*

*d.nhap(&m5);*

*d.nhap(&c2);*

*d.nhap(&m3);*

*d.thong\_ke();*

*d.xuat(im2);*

*d.xuat(ic4);*

*d.thong\_ke();*

*getch();*

*}*

***Chú ý:*** Theo quan điểm chung về cách thức sử dụng, thì lớp CON\_VAT là lớp cơ sở trừu tượng. Tuy nhiên theo quan điểm của C++ thì lớp này chưa phải là lớp cơ sở trừu tượng, vì trong lớp không có các phương thức thuần tuý ảo. Phương thức xung\_ten:

virtual void xung\_ten()

{

}

là phương thức ảo, được định nghĩa đầy đủ , mặc dù thân của nó là rỗng.

Do vậy khai báo:

CON\_VAT cv(“Con vat chung”); vẫn được C++ chấp nhận.

Bây giờ nếu định nghĩa lại phương thức xung\_ten như sau: virtual void xung\_ten() = 0 ;

thì nó trở thành phương thức thuần ảo và C++ sẽ quan niệm lớp CON\_VAT là lớp trừu tượng. Khi đó câu lệnh khai báo:

CON\_VAT cv(“Con vat chung”); sẽ bị C++ bắt lỗi với thông báo:

Cannot create instance of abstruct class ‘CON\_VAT’

## Bài 6. Sử dụng tương ứng bội và phương thức ảo

* 1. **Chiến lược sử dụng tương ứng bội**

Tương ứng bội cho phép xét các vấn đề khác nhau, các đối tượng khác nhau, các phương pháp khác nhau, các cách giải quyết khác nhau theo cùng một lược đồ chung.

Các bước áp dụng tương ứng bội có thể tổng kết lại như sau:

* + 1. Xây dựng lớp cơ sở trừu tượng bao gồm những thuộc tính chung nhất của các thực thể cần quản lý. Đưa vào các phương thức ảo hay thuần ảo dùng để xây dựng các nhóm phương thức ảo cho các lớp dẫn xuất sau này. Mỗi nhóm phương thức ảo sẽ thực hiện một chức năng nào đó trên các lớp.
    2. Xây dựng các lớp dẫn xuất bắt đầu từ lớp cơ sở ảo. Số mức dẫn xuất là không hạn chế. Các lớp dẫn xuất sẽ mô tả các đối tượng cụ thể cần quản lý.
    3. Xây dựng các phương thức ảo trong các dẫn xuất. Các phương thức này tạo thành các nhóm phương thức ảo trong sơ đồ các lớp có quan hệ thừa kế.
    4. Xây dựng lớp quản lý các đối tượng. Dữ liệu của lớp này là một dẫy con trỏ của lớp cơ sở trừu tượng ban đầu. Các con trỏ này có thể chứa địa chỉ đối tượng của các lớp dẫn xuất. Do vậy có thể dùng các con trỏ này để thực hiện các thao tác trên các đối tượng của bất kỳ lớp dẫn xuất nào.

## Ví dụ

Chương trình quản lý các con vật trong §5 là một ví dụ về cách sử dụng tương ứng bội. Dưới đây là một ví dụ khác. Giả sử có 4 hình vẽ: Đoạn thẳng, hình tròn, hình chữ nhật và hình vuông. Bốn hình cho hiện thẳng hàng trên màn hình tạo thành một bức tranh. Nếu thay đổi thứ tự các hình sẽ nhận được các bức tranh khác nhau. Chương trình dưới đây sẽ cho hiện tất cả các bức tranh khác nhau. Chương trình được tổ chức theo các bước nêu trong 6.1:

+ Lớp cơ sở trừu tượng là lớp HINH (hình) gồm một thuộc tính mau (mầu) và một phương thức ảo thuần tuý:

virtual void draw(int x, int y) = 0 ;

+ Các lớp dẫn xuất trực tiếp từ lớp hình là : DTHANG , HTRON và CHUNHAT.

+ Lớp VUONG dẫn xuất từ lớp CHUNHAT.

+ Lớp quản lý chung là lớp picture có thuộc tính là một mảng con trỏ kiểu HINH gồm 4 phần tử dùng để chứa địa chỉ 4 đối tượng: DTHANG, HTRON, CHUNHAT và VUONG. Sử dụng phương thức draw gọi từ 4 phần tử mảng nói trên sẽ nhận được một bức tranh. Bằng cách hoán vị các phần tử này, sẽ nhận được tất cả các bức tranh khác nhau.

*//CT6-05*

*// Lop co so truu tuong*

*// Lop hinh hoc #include <conio.h> #include <graphics.h> class HINH*

*{*

*private:*

*int mau; public:*

*HINH(int m)*

*{*

*mau = m;*

*}*

*getmau()*

*{*

*return mau;*

*}*

*virtual void draw(int x, int y) = 0;*

*};*

*class DTHANG : public HINH*

*{*

*private:*

*int dodai; public:*

*DTHANG(int d, int m):HINH(m)*

*{*

*dodai = d ;*

*}*

*virtual void draw(int x, int y)*

*{*

*setcolor(getmau()) ; line(x,y,x+dodai,y);*

*}*

*};*

*class CHUNHAT: public HINH*

*{*

*private:*

*int rong, cao; public:*

*CHUNHAT(int r, int c, int m):HINH(m)*

*{*

*rong = r; cao = c;*

*}*

*virtual void draw(int x, int y )*

*{*

*setcolor(getmau()) ; rectangle(x,y,x+rong,y+cao); setfillstyle(1,getmau()); floodfill(x+rong/2,y+cao/2, getmau() );*

*}*

*};*

*class VUONG : public CHUNHAT*

*{*

*public:*

*VUONG(int a, int m): CHUNHAT(a,a,m)*

*{*

*}*

*};*

*class HTRON: public HINH*

*{*

*private:*

*int bk; //Ban kinh public:*

*HTRON(int bk1, int m):HINH(m)*

*{*

*bk = bk1;*

*}*

*virtual void draw(int x, int y)*

*{*

*setcolor(getmau()) ; circle(x+bk,y+bk,bk); setfillstyle(1,getmau());*

*floodfill(x + bk, y + bk,getmau());*

*}*

*};*

*class picture*

*{*

*private:*

*HINH \*h[4];*

*public:*

*picture(HINH \*h0,HINH \*h1,HINH \*h2,HINH \*h3)*

*{*

*h[0]=h0;*

*h[1]=h1;*

*h[2]=h2;*

*h[3]=h3;*

*}*

*void paint(int \*k); void listpaint();*

*} ;*

*void picture::paint(int \*k)*

*{*

*for (int i=0; i<4; ++i)*

*h[k[i]]->draw(10+i\*150, 200);*

*}*

*void picture::listpaint()*

*{*

*int k[4],i1,i2,i3,i4; for (i1=0;i1<4;++i1)*

*for (i2=0;i2<4;++i2) if (i2!=i1)*

*for (i3=0;i3<4;++i3) if (i3!=i2 && i3!=i1)*

*for (i4=0;i4<4;++i4)*

*if (i4!=i3 && i4!=i2 && i4!=i1)*

*{*

*k[0]=i1;k[1]=i2;*

*k[2]=i3;k[3]=i4;*

*paint(k);*

*getch(); cleardevice();*

*}*

*}*

*DTHANG dt(120,14);*

*HTRON ht(60,RED);*

*CHUNHAT cn(120,100,MAGENTA); VUONG v(120,CYAN);*

*} ;*

*void main()*

*{*

*int mh=0,mode=0; initgraph(&mh,&mode,""); picture pic(&dt,&ht,&cn,&v); pic.listpaint();*

*getch(); closegraph();*

*}*

## Bài 7. Xử lý các thuật toán khác nhau

Có thể sử dụng tương ứng bội để tổ chức thực hiện các thuật toán khác nhau trên cùng một bài toán như sau:

+ Lớp cơ sở trừu tượng sẽ chứa dữ liệu bài toán và một phương thức ảo.

+ Mỗi lớp dẫn xuất ứng với một thuật toán cụ thể. Phương thức ảo của lớp dẫn xuất sẽ thực hiện một thuật toán cụ thể.

+ Sử dụng một mảng con trỏ của lớp cơ sở và gán cho mỗi phần tử mảng địa chỉ của một đối tượng của lớp dẫn xuất. Sau đó dùng các phần tử mảng con trỏ để gọi tới các phương thức ảo. Bằng cách đó sẽ thực hiện cùng một bài toán theo các thuật toán khác nhau và dễ dàng so sánh hiêụ quả của các thuật toán.

Ví dụ sau minh hoạ việc thực hiện bài toán sắp xếp dẫy số nguyên theo thứ tự tăng bằng cách dùng đồng thời 3 thuật toán: Thuật toán lựa chọn (Select\_Sort), thuật toán sắp xếp nhanh (Quick\_Sort) và thuật toán vun đống (Heap\_Sort). Chương trình gồm 4 lớp:

+ Lớp cơ sở trừu tượng: class sort

{

protected: int \*a;

void hoan\_vi(long i, long j) ; public:

virtual void sapxep(int \*a1, long n) ;

} ;

Lớp này gồm:

* Một thành phần dữ liệu là con trỏ a trỏ tới một vùng nhớ chứa dẫy số nguyên cần sắp xếp.
* Phương thức hoan\_vi(i,j) dùng để hoán vị các phần tử a[i] và a[j]. Phương thức này được dùng trong 3 lớp dẫn xuất bên dưới.
* Phương thức ảo sapxep(a1,n) dùng để sắp xếp dẫy n số nguyên chứa trong mảng a1.

+ Ba lớp dẫn xuất là: SELECT\_SORT, QUICK\_SORT và HEAP\_SORT. Mỗi lớp đều có phương thức ảo:

virtual void sapxep(int \*a1, long n) ;

để thực hiện hiện việc sắp xếp theo theo một thuật toán cụ thể.

+ Trong hàm main() sẽ tạo ra một dẫy 30000 số nguyên một cách ngẫu nhiên, sau đó lần lượt sử dụng 3 thuật toán sắp xếp để so sánh. Kết quả như sau:

Thời gian sắp xếp theo thuật toán Select sort là: 19.20 giây Thời gian sắp xếp theo thuật toán Quick sort là: 0.11 giây Thời gian sắp xếp theo thuật toán Heap sort là: 0.44 giây Nội dung chương trình như sau:

*//CT6-06*

*// Lop co so truu tuong*

*// Lop sort*

*#include <conio.h> #include <stdio.h> #include <time.h> #include <stdlib.h> #include <iostream.h>*

*#include <dos.h> class sort*

*{*

*protected:*

*int \*a;*

*void hoan\_vi(long i, long j)*

*{*

*int tg = a[i]; a[i] = a[j]; a[j] = tg;*

*}*

*public:*

*virtual void sapxep(int \*a1, long n)*

*{*

*a = a1;*

*}*

*} ;*

*class select\_sort : public sort*

*{*

*public:*

*virtual void sapxep(int \*a1, long n) ;*

*} ;*

*void select\_sort::sapxep(int \*a1, long n)*

*{*

*long i,j,r; sort::sapxep(a1,n); for (i=1; i<n; ++i)*

*{*

*r=i;*

*for (j=i+1; j<=n; ++j) if(a[j] < a[r]) r = j;*

*if(r!=i) hoan\_vi(i,r);*

*}*

*}*

*class quick\_sort : public sort*

*{*

*private:*

*void q\_sort(long l, long r); public:*

*virtual void sapxep(int \*a1, long n) ;*

*} ;*

*void quick\_sort::q\_sort(long l, long r)*

*{*

*int x; long i,j;*

*if (l < r)*

*{*

*x = a[l]; i = l; j = r+1; do*

*{*

*++i; --j;*

*while (i<r && a[i] < x) ++i ; while (a[j] > x) --j ;*

*if (i<j) hoan\_vi(i,j);*

*} while (i<j);*

*hoan\_vi(l,j);*

*q\_sort(l,j-1); q\_sort(j+1,r);*

*}*

*}*

*void quick\_sort::sapxep(int \*a1, long n)*

*{*

*sort::sapxep(a1,n); q\_sort(1,n);*

*}*

*class heap\_sort : public sort*

*{*

*private:*

*void shift(long i, long n); public:*

*virtual void sapxep(int \*a1, long n) ;*

*} ;*

*void heap\_sort::shift(long i, long n)*

*{*

*long l,r,k;*

*l = 2\*i; r = l+1; if (l>n) return; if (l==n)*

*{*

*if (a[i]<a[l]) hoan\_vi(i,l); return;*

*}*

*if (a[l] > a[r]) k = l;*

*else*

*k = r;*

*if (a[i]>=a[k]) return;*

*else*

*{*

*hoan\_vi(i,k);*

*shift(k,n);*

*}*

*}*

*void heap\_sort::sapxep(int \*a1, long n)*

*{*

*long i; sort::sapxep(a1,n);*

*/\* Tao dong \*/*

*for (i=n/2 ; i>=1; --i) shift(i,n);*

*/\* Lap \*/*

*for (i=n ; i>=2; --i)*

*{*

*hoan\_vi(1,i);*

*shift(1,i-1);*

*}*

*}*

*void main()*

*{*

*long i,n;*

*struct time t1,t2;*

*int \*a, k, tg, sec, hund; n=30000;*

*a=(int\*) malloc((n+1)\*sizeof(int)); if (a==NULL)*

*{*

*puts("\nLoi BN"); getch();*

*exit(0);*

*}*

*sort \*s[3]; select\_sort ss; quick\_sort qs; heap\_sort hs;*

*s[0]=&ss; s[1]=&qs; s[2]=&hs; clrscr();*

*for (k=0; k<3; ++k)*

*{*

*srand(5000);*

*for (i=1;i<=n;++i) a[i]=rand();*

*gettime(&t1);*

*s[k]->sapxep(a,n); gettime(&t2);*

*tg = (t2.ti\_sec - t1.ti\_sec)\*100 + t2.ti\_hund - t1.ti\_hund ; sec = tg / 100;*

*hund = tg % 100;*

*printf("\n Sap xep %d %d %d %d %d",k+1, t2.ti\_sec,t2.ti\_hund,t1.ti\_sec,t1.ti\_hund);*

*printf("\n Sap xep %d Thoi gian %d sec %d hund",*

*k+1,sec,hund);*

*}*

*getch();*

*}*

**Chương 7**

# Các dòng tập tin (Stream)

C đã cung cấp một thư viện các hàm nhập xuất như printf, scanf, gets, getch(), puts, puch(), fprintf, fscanf, fopen, fwite, fread,. Các hàm này làm việc khá hiệu quả nhưng không thích ứng

với cách tổ chức chương trình hướng đối tượng.

C++ sử dụng khái niệm dòng tin (stream) và đưa ra các lớp dòng tin để tổ chức việc nhập xuất. Dòng tin có thể xem như một dẫy các byte. Thao tác nhập là lấy (đọc) các byte từ dòng tin (khi đó gọi là dòng nhập - input) vào bộ nhớ. Thao tác xuất là đưa các byte từ bộ nhớ ra dòng tin (khi đó gọi là dong xuất - output). Các thao tác này là độc lập thiết bị. Để thực hiện việc nhập, xuất lên một thiết bị cụ thể, chúng ta chỉ cần gắn dòng tin với thiết bị này.

## Bài 1. các lớp stream

Có 4 lớp quan trọng cần nhớ là:

+ Lớp cơ sở ios

+ Từ lớp ios dẫn xuất đến 2 lớp istream và ostream

+ Hai lớp istream và ostream lại dẫn xuất tới lớp iostream Sơ đồ kế thừa giữa các lớp như sau:

ios

iostream

ostream

## Lớp ios

istream

+ Thuộc tính của lớp: Trong lớp ios định nghĩa các thuộc tính được sử dụng làm các cờ định dạng cho việc nhập xuất và các cờ kiểm tra lỗi (xem bên dưới).

+ Các phương thức: Lớp ios cung cấp một số phương thức phục vụ việc định dạng dữ liệu nhập xuất, kiểm tra lỗi (xem bên dưới).

## Lớp istream

Lớp này cung cấp toán tử nhập >> và nhiều phương thức nhập khác (xem bên dưới) như các phương thức: get, getline, read, ignore, peek, seekg, tellg,...

## Lớp ostream

Lớp này cung cấp toán tử xuất << và nhiều phương thức xuất khác (xem bên dưới) như các phương thức: put, write, flush, seekp, tellp,...

## Lớp iostream

Lớp này thừa kế các phương thức nhập xuất của các lớp istream và ostream.

## Bài 2. Dòng cin và toán tử nhập

Dòng cin là một đối tượng kiểu istream đã định nghĩa trong C++ . Đó là dòng vào (input) chuẩn gắn với bàn phím (tương tự như stdin của C). Các thao tác nhập trên dòng cin đồng nghĩa với nhập dữ liệu từ bàn phím.

Do cin là một đối tượng của lớp istream nên với cin chung ta có thể sử dụng toán tử nhập >> và các phương thức nhập của các lớp ios và istream.

Cách dùng toán tử nhập để đọc dữ liệu từ dòng cin như sau:

## cin >> Tham\_số ;

Trong đó Tham\_số có thể là:

* + Biến hoặc phần tử mảng nguyên để nhận một số nguyên
  + Biến hoặc phần tử mảng thực để nhận một số thực
  + Biến hoặc phần tử mảng ký tự để nhận một ký tự
  + Con trỏ ký tự để nhận một dẫy các ký tự khác trống

***Chú ý:*** Các toán tử nhập có thể viết nối đuôi để nhập nhiều giá trị trên một dòng lệnh như sau:

## cin >> Tham\_số\_1 >> Tham\_số\_2 >> ... >> Tham\_số\_k ;

**Cách thức nhập như sau:** Bỏ qua các ký tự trắng (dấu cách, dấu tab, dấu chuyển dòng) đứng trước nếu có và sau đó đọc vào các ký tự tương ứng với kiểu yêu cầu. Cụ thể đối với từng kiểu như sau:

Khi nhập số nguyên sẽ bỏ qua các ký tự trắng đứng trước nếu có, sau đó bắt đầu nhận các ký tự biểu thị số nguyên. Việc nhập kết thúc khi gặp một ký tự trắng hoặc một ký tự không thể hiểu là thành phần của số nguyên. Ví dụ nếu trên dòng vào (gõ từ bàn phím) chứa các ký tự

<space><space>123X2 và Tham\_số (bên phải cin) là biến nguyên n thì n sẽ nhận giá trị 123. Con trỏ nhập sẽ dừng tại ký tự X.

Phép nhập một số thực cũng tiến hành tương tự: Bỏ qua các khoảng trắng đứng trước nếu có, sau đó bắt đầu nhận các ký tự biểu thị số Thực. Việc nhập kết thúc khi gặp một ký tự trắng hoặc một ký tự không thể hiểu là thành phần của số thực.

Phép nhập một ký tự cũng vậy: Bỏ qua các khoảng trắng đứng trước nếu có, sau đó nhận một ký tự khác ký tự trắng. Ví dụ nếu gõ <space><space>XY thì ký tự X được nhận và con trỏ nhập dừng tại ký tự Y.

Phép nhập một dẫy ký tự: Bỏ qua các khoảng trắng đứng trước nếu có, sau đó bắt đầu nhận từ một ký tự khác ký tự trắng. Việc nhập kết thúc khi gặp một ký tự trắng.

**Ví dụ 1:** Xét đoạn chương trình:

*char ten[10], que[12]; char ch;*

*int n; float x;*

*cin >> n >> x >> ch >> ten >> que ;*

Nếu gõ các ký tự: 123<s>3.14<s><s>ZHONG<s>HAI<s>PHONG<Enter>

(để cho gọn sẽ ký hiệu <s> là <space>) thì kết quả nhập như sau:

n=123 x=3.14

ch=’Z’ ten=”HONG” que = “HAI”

Con trỏ nhập sẽ dừng tại ký tự <space> trước từ PHONG. Các ký tự còn lại sẽ được nhận trong các câu lệnh nhập tiếp theo.

**Ví dụ 2:** Xét đoạn chương trình:

*int m; float y;*

*cin >> m >> y;*

*Nếu gõ:*

*<s><s>456<s><s>4.5<Enter> thì kết quả nhập là:*

*m = 456*

*y = 4.5*

Ký tự <Enter> vẫn còn lại trên dòng nhập.

## Bài 3. Nhập ký tự và chuỗi ký tự từ bàn phím

Chúng ta nhận thấy toán tử nhập >> chỉ tiện lợi khi dùng để nhập các giá trị số (nguyên, thực).

Để nhập ký tự và chuỗi ký tự nên dùng các phương thức sau (định nghĩa trong lớp istream):

## cin.get cin.getline cin.ignore

* 1. **Phương thức get có 3 dạng** (thực chất có 3 phương thức cùng có tên get):

### Dạng 1:

int cin.get() ;

dùng để đọc một ký tự (kể cả khoảng trắng). Cách thức đọc của cin.get có thể minh hoạ qua ví dụ sau: Xét các câu lệnh

char ch;

ch = cin.get()

+ Nếu gõ

ABC<Enter>

thì biến ch nhận mã ký tự A, các ký tự BC<Enter> còn lại trên dòng vào.

+ Nếu gõ

A<Enter>

thì biến ch nhận mã ký tự A, ký tự <Enter> còn lại trên dòng vào.

+ Nếu gõ

<Enter>

thì biến ch nhận mã ký tự <Enter> (bằng 10) và dòng vào rỗng.

### Dạng 2:

istream& cin.get(char &ch) ;

dùng để đọc một ký tự (kể cả khoảng trắng) và đặt vào một biến kiểu char được tham chiếu bởi ch.

### Chú ý:

+ Cách thức đọc của cin.get dạng 2 cũng giống như dạng 1

+ Do cin.get() dạng 2 trả về tham chiếu tới cin, nên có thể sử dụng các phương thức get() dạng 2 nối đuôi nhau. Ví dụ 2 nếu khai báo

char ch1, ch2; thì 2 câu lệnh:

cin.get(ch1);

cin.get(ch2);

có thể viết chung trên một câu lệnh sau: cin.get(ch1).get(ch2);

### Dạng 3:

istream& cin.get(char \*str, int n, char delim = ‘\n’);

dùng để đọc một dẫy ký tự (kể cả khoảng trắng) và đưa vào vùng nhớ do str trỏ tới. Quá trình đọc kết thúc khi xẩy ra một trong 2 tình huống sau:

+ Gặp ký tự giới hạn (cho trong delim). Ký tự giới hạn mặc định là ‘\n’ (Enter)

+ Đã nhận đủ (n-1) ký tự

### Chú ý:

+ Ký tự kết thúc chuỗi ‘\0’ được bổ sung vào dẫy ký tự nhận được

+ ký tự giới hạn vẫn còn lại trên dòng nhập để dành cho các lệnh nhập tiếp theo.

### Chú ý:

+ Cũng giống như get() dạng 2, có thể viết các phương thức get() dạng 3 nối đuôi nhau trên một dòng lệnh.

+ Ký tự <Enter> còn lại trên dòng nhập có thể làm trôi phương thức get() dạng 3. Ví dụ xét đoạn chương trình:

char ht[25], qq[20], cq[30]; cout << “\nHọ tên: “ ; cin.get(ht,25);

cout << “\nQuê quán: “ ; cin.get(qq,20);

cout << “\nCơ quan: “ ; cin.get(cq,30);

cout <<”\n” <<ht<<” “<<qq<<” “<<cq

Đoạn chương trình dùng để nhập họ tên, quê quán và cơ quan. Nếu gõ: Pham Thu Huong<Enter>

thì câu lệnh get đầu tiên sẽ nhận được chuỗi “Pham Thu Huong” cất vào mảng ht. Ký tự <Enter> còn lại sẽ làm trôi 2 câu lệnh get tiếp theo. Do đó câu lệnh cuối cùng sẽ chỉ in ra Pham Thu Huong.

Để khắc phục tình trạng trên, có thể dùng một trong các cách sau:

+ Dùng phương thức get() dạng 1 hoặc dạng 2 để lấy ra ký tự <Enter> trên dòng nhập trước khi dùng get (dạng 3).

+ Dùng phương thức ignore để lấy ra một số ký tự không cần thiết trên dòng nhập trước khi dùng get dạng 3. Phương thức này viết như sau:

cin.ignore(n) ; // Lấy ra (loại ra hay bỏ qua) n ký tự trên

// dòng nhập.

Như vậy để có thể nhập được cả quê quán và cơ quan, cần sửa lại đoạn chương trình trên như sau:

char ht[25], qq[20], cq[30]; cout << “\nHọ tên: “ ; cin.get(ht,25);

cin.get(); // Nhận <Enter> cout << “\nQuê quán: “ ; cin.get(qq,20);

ignore(1); // Bỏ qua <Enter> cout << “\nCơ quan: “ ; cin.get(cq,30);

cout <<”\n” <<ht<<” “<<qq<<” “<<cq

## Phương thức getline

Tương tự như get dạng 3, có thể dùng getline để nhập một dẫy ký tự từ bàn phím. Phương thức này được mô tả như sau:

istream& cin.getline(char \*str, int n, char delim = ‘\n’);

Phương thức đầu tiên làm việc như get dạng 3, sau đó nó loại <Enter> ra khỏi dòng nhập (ký tự

<Enter> không đưa vào dẫy ký tự nhận được). Như vậy có thể dùng getline để nhập nhiều chuối ký tự (mà không lo ngại các câu lệnh nhập tiếp theo bị trôi).

**Ví dụ** đoạn chương trình nhập họ tên, quê quán và cơ quan bên trên có thể viết như sau (bằng cách dùng getline):

char ht[25], qq[20], cq[30]; cout << “\nHọ tên: “ ; cin.getline(ht,25);

cout << “\nQuê quán: “ ; cin.getline(qq,20);

cout << “\nCơ quan: “ ; cin.get(cq,30);

cout <<”\n” <<ht<<” “<<qq<<” “<<cq

***Chú ý:*** Cũng giống như get() dạng 2 và get() dạng 3, có thể viết các phương thức getline() nối đuôi nhau trên một dòng lệnh. Ví dụ đoạn chương trình trên có thể viết lại như sau:

char ht[25], qq[20], cq[30];

cout << “\nHọ tên, Quê quán và Cơ quan: “ ; cin.getline(ht,25).getline(qq,20).get(cq,30); cout <<”\n” <<ht<<” “<<qq<<” “<<cq

## Phương thức ignore

Phương thức ignore dùng để bỏ qua (loại bỏ) một số ký tự trên dòng nhập. Trong nhiều trường hợp, đây là việc làm cần thiết để không làm ảnh hưởng đến các phép nhập tiếp theo.

Phương thức ignore được mô tả như sau: istream& cin.ignore(int n=1);

Phương thức sẽ bỏ qua (loại bỏ) n ký tự trên dòng nhập.

## Nhập đồng thời giá trị số và ký tự

Như đã nói trong §2, toán tử nhập >> bao giờ cũng để lại ký tự <Enter> trên dòng nhập. Ký tự

<Enter> này sẽ làm trôi các lệnh nhập ký tự hoặc chuỗi ký tự bên dưới. Do vậy cần dùng:

hoặc ignore() hoặc get() dạng 1 hoặc get() dạng 2

để loại bỏ ký tự <Enter> còn sót lại ra khỏi dòng nhập trước khi thực hiện việc nhập ký tự hoặc chuỗi ký tự.

* 1. **Ví dụ:** Chương trình dưới đây sử dụng lớp TSINH (Thí sinh) với 2 phương thức xuat và nhap.

*//CT7\_04.CPP*

*// Nhập dữ liêu số và ký tự #include <iostream.h> #include <conio.h>*

*struct TS*

*{*

*int sobd; char ht[25];*

*float dt,dl,dh,td;*

*} ;*

*class TSINH*

*{*

*private:*

*TS \*ts; int sots;*

*public:*

*TSINH()*

*{*

*ts=NULL;*

*sots=0;*

*}*

*TSINH(int n)*

*{*

*ts=new TS[n+1]; sots=n;*

*}*

*~TSINH()*

*{*

*if (sots)*

*{*

*sots=0;*

*ts = NULL;*

*}*

*}*

*void nhap(); void xuat();*

*} ;*

*void TSINH::nhap()*

*{*

*if (sots)*

*for (int i=1; i<=sots; ++i)*

*{*

*cout << "\nThi sinh "<< i << ": " ; cout << "\nSo bao danh: " ;*

*cin >> ts[i].sobd; cin.ignore();*

*cout << "Ho ten: " ; cin.get(ts[i].ht,25);*

*cout << "Diem toan, ly , hoa: " ;*

*cin >> ts[i].dt >> ts[i].dl >> ts[i].dh; ts[i].td = ts[i].dt + ts[i].dl + ts[i].dh;*

*}*

*}*

*void TSINH::xuat()*

*{*

*if (sots)*

*{*

*cout << "\nDanh sach thi sinh:" ; for (int i=1; i<=sots; ++i)*

*cout << "\nHo ten: " << ts[i].ht << " So BD: "<< ts[i].sobd*

*<<" Tong diem: "<< ts[i].td;*

*}*

*}*

*void main()*

*{*

*int n; clrscr();*

*cout << "\nSo thi sinh: "; cin>>n;*

*TSINH \*t = new TSINH(n); t->nhap() ;*

*t->xuat();*

*getch(); delete t;*

*}*

## Bài 4. Dòng cout và toán tử xuất

* 1. **Dòng cout**

Dòng cout là một đối tượng kiểu ostream đã định nghĩa trong C++. Đó là dòng xuất (output) chuẩn gắn với màn hình (tương tự như stdout của C). Các thao tác xuất trên dòng cout đồng nghĩa với xuất dữ liệu ra màn hình.

Do cout là một đối tượng của lớp ostream nên với cout chung ta có thể sử dụng toán tử xuất << và các phương thức xuất của các lớp ios và ostream.

## Toán tử xuất

C++ định nghĩa chồng toán tử dịch trái << để gửi các ký tự ra dòng xuất. Cách dùng toán tử xuất để xuất dữ liệu từ bộ nhớ ra dòng cout như sau:

## cout << Tham\_số ;

Trong đó Tham\_số biểu thị một giá trị cần xuất ra màn hình. Giá trị sẽ được biến đổi thành một dẫy ký tự trước khi đưa ra dòng xuất. Kiểu của Tham\_số có thể như sau:

* Nguyên (xuất giá trị nguyên)
* Thực (xuất giá trị thực)
* ký tự - char (xuất một ký tự)
* con trỏ ký tự - char\* (xuất chuỗi ký tự)

***Chú ý:*** Các toán tử xuất có thể viết nối đuôi nhau (để xuất nhiều giá trị) trên một dòng lệnh như sau:

## cout << Tham\_số\_1 << Tham\_số\_2 << ... << Tham\_số\_k ;

***Chú ý:*** Toán tử xuất được định nghĩa chồng (trùng tên) với toán tử dịch trái và nó cùng có mức độ ưu tiên như toán tử dịch trái. Xem phụ lục 1 chúng ta thấy toán tử xuất có thứ tự ưu tiên lớn hơn các toán tử trong biểu thức điều kiện. Vì vậy nếu dùng toán tử xuất để in một biểu thức điều kiện như sau:

int a=5, b=10;

cout << “\nMax= “ << a>b?a:b ;

thì Trình biên dịch sẽ báo lỗi. Để tránh lỗi cần dùng các dấu ngoặc tròn để bao biểu thức điều kiện như sau:

int a=5, b=10;

cout << “\nMax= “ << (a>b?a:b) ;

**Tóm lại:** Nên bao các biểu thức trong 2 dấu ngoặc tròn.

## Định dạng (tạo khuôn dạng cho) dữ liệu xuất

Việc định dạng dữ liệu xuất hay tạo khuôn dạng cho dữ liệu xuất là một việc cần thiết. Ví dụ cần in các giá trị thực trên 10 vị trí trong đó có 2 vị trí dành cho phần phân.

Bản thân toán tử xuất chưa có khả năng định dạng, mà cần sử dụng các công cụ sau:

+ Các phương thức định dạng

+ Các các cờ định dạng

+ Các hàm và bộ phận định dạng

Mục sau sẽ trình bầy cách định dạng giá trị xuất.

## Bài 5. Các phương thức định dạng

* 1. **Nội dung định dạng giá trị xuất**

Nội dung định dạng là xác định các thông số:

* + - Độ rộng quy định
    - Độ chính xác
    - Ký tự độn
    - Và các thông số khác

**+ Độ rộng thực tế của giá trị xuất:** Như đã nói ở trên, C++ sẽ biến đổi giá trị cần xuất thành một chuỗi ký tự rồi đưa chuỗi này ra màn hình. Ta sẽ gọi số ký tự của chuỗi này là độ rộng thực tế của giá trị xuất. Ví dụ với các câu lệnh:

int n=4567, m=-23 ; float x = -3.1416 ;

char ht[] = “Tran Van Thong” ;

thì:

Độ rộng thực tế của n là 4, của m là 3, của x là 7, của ht là 14.

**+ Độ rộng quy đinh** là số vị trí tối thiểu trên màn hình dành để in giá trị. Theo mặc định, độ rộng quy định bằng 0. Chúng ta có thể dùng phương thức cout.width() để thiết lập rộng này. Ví dụ câu lệnh:

cout.width(8);

sẽ thiết lập độ rộng quy định là 8.

## + Mối quan hệ giữa độ rộng thực tế và độ rộng quy định

* Nếu độ rộng thực tế lớn hơn hoặc bằng độ rộng quy định thì số vị trí trên màn hình chứa giá trị xuất sẽ bằng độ rộng thực tế.
* Nếu độ rộng thực tế nhỏ hơn độ rộng quy định thì số vị trí trên màn hình chứa giá trị xuất sẽ bằng độ rộng quy định. Khi đó sẽ có một số vị trí dư thừa. Các vị trí dư thừa sẽ được độn (lấp đầy) bằng khoảng trống.

**+ Xác định ký tự độn:** Ký tự độn mặc định là dấu cách (khoảng trống). Tuy nhiên có thể dùng phương thức cout.fill() để chọn một ký tự độn khác. Ví dụ với các câu lệnh sau:

int n=123; // Độ rộng thực tế là 3 cout.fill(‘\*’); // Ký tự độn là \* cout.width(5); // Độ rộng quy định là 5 cout << n ;

thì kết quả in ra là:

\*\*123

**+ Độ chính xác** là số vị trí dành cho phần phân (khi in số thực). Độ chính xác mặc định là 6. Tuy nhiên có thể dùng phương thức cout.precision() để chọn độ chính xác. Ví dụ với các câu lệnh:

float x = 34.455 ; // Độ rộng thực tế 6 cout.precision(2) ; // Độ chính xác 2

cout.width(8); // Độ rộng quy ước 8 cout.fill(‘0’) ; // Ký tự độn là số 0 cout << x ;

thì kết quả in ra là: 0034.46

## Các phương thức định dạng

1. Phương thức

## int cout.width()

cho biết độ rộng quy định hiện tại.

1. Phương thức

## int cout.width(int n)

Thiết lập độ rộng quy định mới là n và trả về độ rộng quy định trước đó.

***Chú ý:*** Độ rộng quy định n chỉ có tác dụng cho một giá trị xuất. Sau đó C++ lại áp dụng độ rộng quy định bằng 0.

**Ví dụ** với các câu lệnh: int m=1234, n=56; cout << “\nAB” cout.width(6);

cout << m ; cout << n ;

thì kết quả in ra là:

AB 123456

(giữa B và số 1 có 2 dấu cách).

1. Phương thức

## int cout.precision()

Cho biết độ chính xác hiện tại (đang áp dụng để xuất các giá trị thức).

1. Phương thức

## int cout.precision(int n)

Thiết lập độ chính xác sẽ áp dụng là n và cho biết độ chính xác trước đó. Độ chính xác được thiết lập sẽ có hiệu lực cho tới khi gặp một câu lệnh thiết lập độ chính xác mới.

1. Phương thức

## char cout.fill()

Cho biết ký tự độn hiện tại đang được áp dụng.

1. Phương thức

## char cout.fill(char ch)

Quy định ký tự độn mới sẽ được dùng là ch và cho biết ký tự độn đang dùng trước đó. Ký tự độn được thiết lập sẽ có hiệu lực cho tới khi gặp một câu lệnh chọn ký tự độn mới.

**Ví dụ** xét chương trình:

*//CT7\_06.CPP*

*// Cac phuong thuc dinh dang #include <iostream.h> #include <conio.h>*

*void main()*

*{*

*clrscr();*

*float x=-3.1551, y=-23.45421;*

*cout.precision(2); cout.fill('\*');*

*cout << "\n" ; cout.width(8); cout << x; cout << "\n" ; cout.width(8); cout << y; getch();*

*}*

*Sau khi thực hiện, chương trình in ra màn hình 2 dòng sau:*

*\*\*\*-3.16*

*\*\*-23.45*

## Bài 6. Cờ định dạng

* 1. **Khái niệm chung về cờ**

Mỗi cờ chứa trong một bit. Cờ có 2 trạng thái: Bật (on) - có giá trị 1

Tắt (off) - có giá trị 0

(Trong 6.3 sẽ trình bầy các phương thức dùng để bật, tắt các cờ)

Các cờ có thể chứa trong một biến kiểu long. Trong tệp <iostream.h> đã định nghĩa các cờ sau:

|  |  |  |
| --- | --- | --- |
| ios::left | ios::right | ios::internal |
| ios::dec | ios::oct | ios::hex |
| ios::fixed | ios::scientific | ios::showpos |
| ios::uppercase | ios::showpoint | ios::showbase |

## Công dụng của các cờ

Có thể chia các cờ thành các nhóm:

## Nhóm 1 gồm các cờ định vị (căn lề) :

ios::left ios::right ios::internal

**Cờ ios::left**: Khi bật cờ ios:left thì giá trị in ra nằm bên trái vùng quy định, các ký tự độn nằm sau, ví dụ:

35\*\*\*

-89\*\*

**Cờ ios::right**: Khi bật cờ ios:right thì giá trị in ra nằm bên phải vùng quy định, các ký tự độn nằm trước, ví dụ:

\*\*\*35

\*\*-89

***Chú ý:*** Mặc định cờ ios::right bật.

**Cờ ios::internal**: Cờ ios:internal có tác dụng giống như cờ ios::right chỉ khác là dấu (nếu có) in đầu tiên, ví dụ:

\*\*\*35

-\*\*89

Chương trình sau minh hoạ cách dùng các cờ định vị:

*//CT7\_06.CPP*

*// Cac phuong thuc dinh dang*

*// Co dinh vi*

*#include <iostream.h> #include <conio.h> void main()*

*{*

*clrscr();*

*float x=-87.1551, y=23.45421;*

*cout.precision(2); cout.fill('\*');*

*cout.setf(ios::left); // Bật cờ ios::left cout << "\n" ;*

*cout.width(8); cout << x; cout << "\n" ; cout.width(8); cout << y;*

*cout.setf(ios::right); // Bật cờ ios::right cout << "\n" ;*

*cout.width(8); cout << x; cout << "\n" ; cout.width(8); cout << y;*

*cout.setf(ios::internal); // // Bật cờ ios::internal cout << "\n" ;*

*cout.width(8); cout << x; cout << "\n" ; cout.width(8); cout << y;*

*getch();*

*}*

Sau khi thực hiện chương trình in ra 6 dòng như sau:

-87.16\*\*

23.45\*\*\*

\*\*-87.16

\*\*\*23.45

-\*\*87.16

\*\*\*23.45

## Nhóm 2 gồm các cờ định dạng số nguyên:

ios::dec ios::oct ios::hex

+ Khi ios::dec bật (mặc định): Số nguyên được in dưới dạng cơ số 10

+ Khi ios::oct bật : Số nguyên được in dưới dạng cơ số 8

+ Khi ios::hex bật : Số nguyên được in dưới dạng cơ số 16

**Nhóm 3 gồm các cờ định dạng số thực**: ios::fĩxed ios::scientific ios::showpoint

Mặc định: Cờ ios::fixed bật (on) và cờ ios::showpoint tắt (off).

+ Khi ios::fixed bật và cờ ios::showpoint tắt thì số thực in ra dưới dạng thập phân, số chữ số phần phân (sau dấu chấm) được tính bằng độ chính xác n nhưng khi in thì bỏ đi các chữ số 0 ở cuối.

**Ví dụ** nếu độ chính xác n = 4 thì:

Số thực -87.1500 được in: -87.15

Số thực 23.45425 được in: 23.4543

Số thực 678.0 được in: 678

+ Khi ios::fixed bật và cờ ios::showpoint bật thì số thực in ra dưới dạng thập phân, số chữ số phần phân (sau dấu chấm) được in ra đúng bằng độ chính xác n.

**Ví dụ** nếu độ chính xác n = 4 thì:

Số thực -87.1500 được in: -87.1500

Số thực 23.45425 được in: 23.4543

Số thực 678.0 được in: 678.0000

+ Khi ios::scientific bật và cờ ios::showpoint tắt thì số thực in ra dưới dạng mũ (dạng khoa học). Số chữ số phần phân (sau dấu chấm) của phần định trị được tính bằng độ chính xác n nhưng khi in thì bỏ đi các chữ số 0 ở cuối.

**Ví dụ** nếu độ chính xác n = 4 thì:

Số thực -87.1500 được in: -8.715e+01 Số thực 23.45425 được in: 2.3454e+01 Số thực 678.0 được in: 6.78e+02

+ Khi ios::scientific bật và cờ ios::showpoint bật thì số thực in ra dưới dạng mũ. Số chữ số phần phân (sau dấu chấm) của phần định trị được in đúng bằng độ chính xác n.

**Ví dụ** nếu độ chính xác n = 4 thì:

Số thực -87.1500 được in: -8.7150e+01 Số thực 23.45425 được in: 2.3454e+01 Số thực 678.0 được in: 6.7800e+01

## Nhóm 4 gồm các hiển thị:

ios::showpos ios::showbase ios::uppercase

## Cờ ios::showpos

+ Nếu cờ ios::showpos tắt (mặc định) thì dấu cộng không được in trước số dương.

+ Nếu cờ ios::showpos bật thì dấu cộng được in trước số dương.

## Cờ ios::showbase

+ Nếu cờ ios::showbase bật thì số nguyên hệ 8 được in bắt đầu bằng ký tự 0 và số nguyên hệ 16 được bắt đầu bằng các ký tự 0x. Ví dụ nếu a = 40 thì:

dạng in hệ 8 là: 050 dạng in hệ 16 là 0x28

+ Nếu cờ ios::showbase tắt (mặc định) thì không in 0 trước số nguyên hệ 8 và không in 0x trước số nguyên hệ 16. Ví dụ nếu a = 40 thì:

dạng in hệ 8 là: 50

dạng in hệ 16 là 28

## Cờ ios::uppercase

+ Nếu cờ ios::uppercase bật thì các chữ số hệ 16 (như A, B, C, ...) được in dưới dạng chữ hoa.

+ Nếu cờ ios::uppercase tắt (mặc định) thì các chữ số hệ 16 (như A, B, C, ...) được in dưới dạng chữ thường.

## Các phương thức bật tắt cờ

Các phương thức này định nghĩa trong lớp ios.

+ Phương thức

## long cout.setf(long f) ;

sẽ bật các cờ liệt kê trong f và trả về một giá trị long biểu thị các cờ đang bật. Thông thường giá trị f được xác định bằng cách tổ hợp các cờ trình bầy trong mục 6.1.

**Ví dụ** câu lệnh:

cout.setf(ios::showpoint | ios::scientific) ; sẽ bật các cờ ios::showpoint và ios::scientific.

+ Phương thức

## long cout.unsetf(long f) ;

sẽ tắt các cờ liệt kê trong f và trả về một giá trị long biểu thị các cờ đang bật. Thông thường giá trị f được xác định bằng cách tổ hợp các cờ trình bầy trong mục 6.1.

**Ví dụ** câu lệnh:

cout.unsetf(ios::showpoint | ios::scientific) ; sẽ tắt các cờ ios::showpoint và ios::scientific.

+ Phương thức

## long cout.flags(long f) ;

có tác dụng giống như cout.setf(long). Ví dụ câu lệnh:

cout.flags(ios::showpoint | ios::scientific) ; sẽ bật các cờ ios::showpoint và ios::scientific.

+ Phương thức

## long cout.flags() ;

sẽ trả về một giá trị long biểu thị các cờ đang bật.

## Bài 7. Các bộ phận định dạng và các hàm định dạng

* 1. **Các bộ phận định dạng (định nghĩa trong <iostream.h>)**

Các bộ phận định dạng gồm: dec // như cờ ios::dec oct // như cờ ios::oct hex // như cờ ios::hex

endl // xuất ký tự ‘\n’ (chuyển dòng) flush // đẩy dữ liệu ra thiết bị xuất

Chúng có tác dụng như cờ định dạng nhưng được viết nối đuôi trong toán tử xuất nên tiện sử dụng hơn.

**Ví dụ** xét chương trình đơn giản sau:

*//CT7\_08.CPP*

*// Bo phan dinh dang #include <iostream.h> #include <conio.h> void main()*

*{*

*clrscr(); cout.setf(ios::showbase)*

*cout << "ABC" << endl << hex << 40 << " " << 41; getch();*

*}*

Chương trình sẽ in 2 dòng sau ra màn hình: ABC

0x28 0x29

## Các hàm định dạng (định nghĩa trong <iomanip.h>)

Các hàm định dạng gồm:

setw(int n) // như cout.width(int n) setpecision(int n) // như cout.pecision(int n) setfill(char ch) // như cout. fill(char ch) setiosflags(long l) // như cout.setf(long f) resetiosflags(long l) // như cout.unsetf(long f)

Các hàm định dạng có tác dụng như các phương thức định dạng nhưng được viết nối đuôi trong toán tử xuất nên tiện sử dụng hơn.

***Chú ý 1:*** Các hàm định dạng (cũng như các bộ phận định dạng) cần viết trong các toán tử xuất.

Một hàm định dạng đứng một mình như một câu lệnh sẽ không có tác dụng định dạng.

***Chú ý 2:*** Muốn sử dụng các hàm định dạng cần bổ sung vào đầu chương trình câu lệnh:

## #include <iomanip.h>

**Ví dụ** có thể thay phương thức cout.setf(ios::showbase) ;

trong chương trình của mục 7.1 bằng hàm cout << setiosflags(ios::showbase); (chú ý hàm phải viết trong toán tử xuất)

Như vậy chương trình trong 7.1 có thể viết lại theo các phương án sau:

***Phương án 1:*** *#include <iostream.h> #include <iomanip.h> #include <conio.h> void main()*

*{*

*clrscr();*

*cout << setiosflags(ios::showbase) ;*

*cout << "ABC" << endl << hex << 40 << " " << 41; getch();*

*}*

***Phương án 2:*** *#include <iostream.h> #include <iomanip.h> #include <conio.h> void main()*

*{*

*clrscr();*

*cout << "ABC" << endl << setiosflags(ios::showbase)*

*<< hex << 40 << " " << 41;*

*getch();*

*}*

Dưới đây là ví dụ khác về việc dùng các hàm và bộ phận định dạng. Các câu lệnh: int i = 23;

cout << i << endl << setiosflags(ios::showbase)

<< hex << i << dec << setfill(‘\*’)

<< endl << setw(4) << i << setfill(‘0’)

<< endl << setw(5) << i ; sẽ in ra màn hình như sau:

23

0x17

\*\*23 00023

* 1. **Ví dụ:** Chương trình dưới đây minh hoạ cách dùng các hàm định dạng và phương thức định dạng để in danh sách thí sinh dưới dạng bảng với các yêu cầu sau: Số báo danh in 4 ký tự (chèn thêm số 0 vào trước ví dụ 0003), tổng điểm in với đúng một chữ số phần phân.

*//CT7\_08.CPP*

*// Bo phan dinh dang*

*// Ham dinh dang*

*// Co dinh dang #include <iostream.h> #include <iomanip.h> #include <conio.h> struct TS*

*{*

*int sobd; char ht[25];*

*float dt,dl,dh,td;*

*};*

*class TSINH*

*{*

*private:*

*TS \*ts; int sots;*

*public:*

*TSINH()*

*{*

*ts=NULL;*

*sots=0;*

*}*

*TSINH(int n)*

*{*

*ts=new TS[n+1]; sots=n;*

*}*

*~TSINH()*

*{*

*if (sots)*

*{*

*sots=0;*

*ts = NULL;*

*}*

*}*

*void nhap(); void sapxep(); void xuat();*

*} ;*

*void TSINH::nhap()*

*{*

*if (sots)*

*for (int i=1; i<=sots; ++i)*

*{*

*cout << "\nThi sinh "<< i << ": " ; cout << "\nSo bao danh: " ;*

*cin >> ts[i].sobd; cin.ignore();*

*cout << "Ho ten: " ; cin.get(ts[i].ht,25);*

*cout << "Diem toan, ly , hoa: " ;*

*cin >> ts[i].dt >> ts[i].dl >> ts[i].dh; ts[i].td = ts[i].dt + ts[i].dl + ts[i].dh;*

*}*

*}*

*void TSINH::sapxep()*

*{*

*int i,j;*

*for (i=1; i< sots; ++i)*

*for (j=i+1; j<= sots; ++j) if (ts[i].td < ts[j].td)*

*{*

*TS tg; tg=ts[i]; ts[i]=ts[j]; ts[j]=tg;*

*}*

*}*

*void TSINH::xuat()*

*{*

*if (sots)*

*{*

*cout << "\nDanh sach thi sinh:" ; cout.precision(1);*

*cout << setiosflags(ios::left);*

*cout << "\n" << setw(20) << "Ho ten" << setw(8)*

*<< "So BD" << setw(10) << "Tong diem"; for (int i=1; i<=sots; ++i)*

*cout << "\n" << setw(20)<<setiosflags(ios::left) << ts[i].ht*

*<< setw(4) << setfill('0') << setiosflags(ios::right)*

*<< ts[i].sobd << " " << setfill(32)*

*<< setiosflags(ios::left|ios::showpoint)*

*<< setw(10) << ts[i].td;*

*}*

*}*

*void main()*

*{*

*int n; clrscr();*

*cout << "\nSo thi sinh: "; cin>>n;*

*TSINH \*t = new TSINH(n); t->nhap() ;*

*t->sapxep();*

*t->xuat();*

*getch(); delete t;*

*}*

## Bài 8. Các dòng tin chuẩn

Có 4 dòng tin (đối tượng của các lớp Stream) đã định nghĩa trước, được cài đặt khi chương trình khởi động. Hai trong số đó đã nói ở trên là:

cin dòng input chuẩn gắn với bàn phím, giống như stdin của C. cout dòng output chuẩn gắn với màn hình, giống như stdout của C.

Hai dòng tin chuẩn khác là:

cerr dòng output lỗi chuẩn gắn với màn hình, giống như stderr của C. clog giống cerr nhưng có thêm bộ đệm.

***Chú ý 1:*** Có thể dùng các dòng cerr và clog để xuất ra màn hình như đã dùng đối với cout.

***Chú ý 2:*** Vì clog có thêm bộ đệm, nên dữ liệu được đưa vào bộ đệm. Khi đầy bộ đệm thì đưa dữ liệu từ bộ đệm ra dòng clog. Vì vậy trước khi kết thúc xuất cần dùng phương thức:

clog.flush();

để đẩy dữ liệu từ bộ đệm ra clog.

Chương trình sau minh hoạ cách dùng dòng clog. Chúng ta nhận thấy, nếu bỏ câu lệnh clog.flush() thì sẽ không nhìn thấy kết quả xuất ra màn hình khi chương trình tạm dừng bởi câu lệnh getch().

// Dùng clog và flush *#include <iostream.h> #include <conio.h>*

*void main()*

*{*

*clrscr();*

*float x=-87.1500, y=23.45425,z=678.0;*

*clog.setf(ios::scientific); clog.precision(4); clog.fill('\*');*

*clog << "\n"; clog.width(10); clog << x; clog << "\n"; clog.width(10); clog << y; clog << "\n"; clog.width(10); clog << z; clog.flush();*

*getch();*

*}*

## Bài 9. Xuất ra máy in

Trong số 4 dòng tin chuẩn không dòng nào gắn với máy in. Như vậy không thể dùng các dòng này để xuất dữ liệu ra máy in. Để xuất dữ liệu ra máy in (cũng như nhập, xuất trên tệp) cần tạo ra các dòng tin mới và cho nó gắn với thiết bị cụ thể. C++ cung cấp 3 lớp stream để làm điều này, đó là các lớp:

**ifstream** dùng để tạo dòng nhập

**ofstream** dùng để tạo dòng xuất

**fstream** dùng để tạo dòng nhập, dòng xuất hoặc dòng nhập-xuất

Mỗi lớp có 4 hàm tạo dùng để khai báo các dòng tin (đối tượng dòng tin). Trong mục sau sẽ nói thêm về các hàm tạo này.

Để tạo một dòng xuất và gắn nó với máy in ta có thể dùng một trong các hàm tạo sau: ofstream Tên\_dòng\_tin(int fd) ;

ofstream Tên\_dòng\_tin(int fd, char \*buf, int n) ; Trong đó:

+ Tên\_dòng\_tin là tên biến đối tượng kiểu ofstream hay gọi là tên dòng xuất do chúng ta tự đặt.

+ fd (file descriptor) là chỉ số tập tin. Chỉ số tập tin định sẵn đối với stdprn (máy in chuẩn) là 4.

+ Các tham số buf và n xác định một vùng nhớ n byte do buf trỏ tới. Vùng nhớ sẽ được dùng làm bộ đệm cho dòng xuất.

**Ví dụ 1** câu lệnh: ofstream prn(4) ;

sẽ tạo dòng tin xuất prn và gắn nó với máy in chuẩn. Dòng prn sẽ có bộ đệm mặc định. Dữ liệu trước hết chuyển vào bộ đệm, khi đầy bộ đệm thì dữ liệu sẽ được đẩy từ bộ đệm ra dòng prn. Để chủ động yêu cầu đẩy dữ liệu từ bộ đệm ra dòng prn có thể sử dụng phương thức flush hoặc bộ phận định dạng flush. Cách viết như sau:

prn.flush(); // Phương thức

prn << flush ; // Bộ phận định dạng

Các câu lệnh sau sẽ xuất dữ liệu ra prn (máy in) và ý nghĩa của chúng như sau: prn << “\nTong = “ << (4+9) ; // Đưa một dòng vào bộ đệm

prn << “\nTich =“ << (4\*9); // Đưa tiếp dòng thứ 2 vào bộ đệm prn.flush(); // Đẩy dữ liệu từ bộ đệm ra máy in (in 2 dòng)

Các câu lệnh dưới đây cũng xuất dữ liệu ra máy in nhưng sẽ in từng dòng một: prn << “\nTong = “ << (4+9) << flush ; // In một dòng

prn << “\nTich = “ << (4\*9) ; << flush // In dòng thứ hai

**Ví dụ 2:** Các câu lệnh char buf[1000] ;

ofstream prn(4,buf,1000) ;

sẽ tạo dòng tin xuất prn và gắn nó với máy in chuẩn. Dòng xuất prn sử dụng 1000 byte của mảng buf làm bộ đệm. Các câu lệnh dưới đây cũng xuất dữ liệu ra máy in:

prn << “\nTong = “ << (4+9) ; // Đưa dữ liệu vào bộ đệm prn << “\nTich = “ << (4\*9) ; // Đưa dữ liệu vào bộ đệm prn.flush() ; // Xuất 2 dòng (ở bộ đệm) ra máy in

***Chú ý:*** Trước khi kết thúc chương trình, dữ liệu từ bộ đệm sẽ được tự động đẩy ra máy in.

**Chương trinh minh hoạ:** Chương trình dưới đây tương tự như chương trình trong mục 7.3 (chỉ sửa đổi phương thức xuất) nhưng thay việc xuất ra màn hình bằng xuất ra máy in.

*//CT7\_08B.CPP*

*// Xuat ra may in*

*// Bo phan dinh dang*

*// Ham dinh dang #include <iostream.h> #include <iomanip.h> #include <conio.h> struct TS*

*{*

*int sobd; char ht[25];*

*float dt,dl,dh,td;*

*} ;*

*class TSINH*

*{*

*private:*

*TS \*ts;*

*int sots; public:*

*TSINH()*

*{*

*ts=NULL;*

*sots=0;*

*}*

*TSINH(int n)*

*{*

*ts=new TS[n+1]; sots=n;*

*}*

*~TSINH()*

*{*

*if (sots)*

*{*

*sots=0;*

*ts = NULL;*

*}*

*}*

*void nhap(); void sapxep(); void xuat();*

*} ;*

*void TSINH::nhap()*

*{*

*if (sots)*

*for (int i=1; i<=sots; ++i)*

*{*

*cout << "\nThi sinh "<< i << ": " ; cout << "\nSo bao danh: " ;*

*cin >> ts[i].sobd; cin.ignore();*

*cout << "Ho ten: " ; cin.get(ts[i].ht,25);*

*cout << "Diem toan, ly , hoa: " ;*

*cin >> ts[i].dt >> ts[i].dl >> ts[i].dh; ts[i].td = ts[i].dt + ts[i].dl + ts[i].dh;*

*}*

*}*

*void TSINH::sapxep()*

*{*

*int i,j;*

*for (i=1; i< sots; ++i)*

*for (j=i+1; j<= sots; ++j) if (ts[i].td < ts[j].td)*

*{*

*TS tg; tg=ts[i]; ts[i]=ts[j]; ts[j]=tg;*

*}*

*}*

*void TSINH::xuat()*

*{*

*ostream prn(4); if (sots)*

*{*

*prn << "\nDanh sach thi sinh:" ; prn.precision(1);*

*prn << setiosflags(ios::left);*

*prn << "\n" << setw(20) <<"Ho ten" << setw(8)*

*<< "So BD"<< setw(10) << "Tong diem"; for (int i=1; i<=sots; ++i)*

*prn << "\n" << setw(20)<<setiosflags(ios::left) <<ts[i].ht << setw(4) << setfill('0')<<setiosflags(ios::right)<< ts[i].sobd*

*<< " " << setfill(32) <<setiosflags(ios::left|ios::showpoint)*

*<<setw(10)<< ts[i].td;*

*}*

*}*

*void main()*

*{*

*int n; clrscr();*

*cout << "\nSo thi sinh: "; cin>>n;*

*TSINH \*t = new TSINH(n); t->nhap() ;*

*t->sapxep();*

*t->xuat();*

*getch(); delete t;*

*}*

## Bài 10. Làm việc với tệp

* 1. **Các lớp dùng để nhập, xuất dữ liệu lên tệp**

Như đã nói ở trên, C++ cung cấp 4 dòng tin chuẩn để làm việc với bàn phím và màn hình. Muốn nhập xuất lên tệp chúng ta cần tạo các dòng tin mới (khai báo các đối tượng Stream) và gắn chúng với một tệp cụ thể. C++ cung cấp 3 lớp stream để làm điều này, đó là các lớp:

ofstream dùng để tạo các dòng xuất (ghi tệp) ifstream dùng để tạo các dòng nhập (đọc tệp)

fstream dùng để tạo các dòng nhập, dòng xuất hoặc dòng nhập-xuất Sơ đồ dẫn xuất các lớp như sau:

istream

fstreambase

ios

ifstream

ofstream

ostream

fstream

## Ghi dữ liệu lên tệp

Thủ tục ghi dữ liệu lên tệp như sau:

* + 1. Dùng lớp ofstream để tạo ra một dòng xuất và gắn nó với một tệp cụ thể. Khi đó việc xuất dữ liệu ra dòng này đồng nghĩa với việc ghi dữ liệu lên tệp.
    2. Thực hiện xuất dữ liệu ra dòng xuất vừa tạo như thể xuất dữ liệu ra dòng xuất chuẩn cout.

## Đọc dữ liệu từ tệp

Thủ tục đọc dữ liệu từ tệp như sau:

* + 1. Dùng lớp ifstream để tạo ra một dòng nhập và gắn nó với một tệp cụ thể. Khi đó việc nhập dữ liệu từ dòng này đồng nghĩa với việc đọc dữ liệu từ tệp.
    2. Thực hiện nhập dữ liệu từ dòng nhập vừa tạo như thể nhập dữ liệu từ dòng nhập chuẩn cin.

## Đọc - ghi dữ liệu đồng thời trên tệp

Thủ tục đọc-ghi dữ liệu đồng thời trên tệp như sau:

* + 1. Dùng lớp fstream để tạo ra một dòng nhập-xuất và gắn nó với một tệp cụ thể.
    2. Thực hiện nhập dữ liệu từ dòng nhập-xuất vừa tạo như thể nhập dữ liệu từ dòng nhập chuẩn cin.
    3. Thực hiện xuất dữ liệu ra dòng nhập-xuất vừa tạo như thể xuất dữ liệu ra dòng xuất chuẩn cout.

**Nhận xét:** Như vậy:

1. Việc xuất dữ liệu ra máy in hoặc lên tệp được thực hiện hoàn toàn giống như xuất dữ liệu ra dòng xuất chuẩn cout (màn hình).
2. Việc đọc dữ liệu từ tệp được thực hiện hoàn toàn giống như nhập dữ liệu từ dòng nhập chuẩn cin (bàn phím).

## Bài 11. Ghi dữ liệu lên tệp

* 1. **Lớp ofstream**

Để ghi dữ liệu lên tệp chúng ta sử dụng lớp ofstream. Lớp ofstream thừa kế các phương thức của các lớp ios và ostream. Nó cũng thừa kế phương thức:

## close

của lớp fstreambase. Ngoài ra lớp ofstream có thêm các hàm tạo và các phương thức sau:

* + 1. Hàm tạo:

**ofstream() ;** // Không đối

dùng để tạo một đối tượng ofstream (dòng xuất), chưa gắn với tệp.

* + 1. Hàm tạo:

## ofstream(const char \*fn, int mode = ios::out,

**int prot = filebuf::openprot);**dùng để tạo một đối tượng ofstream, mở tệp có tên fn để ghi và gắn đối tượng vừa tạo với tệp được mở.

+ Tham số fn cho biết tên tệp.

+ Tham số mode có giá trị mặc định là ios::out (mở để ghi). Tham số này có thể là một hợp của các giá trị sau:

ios::binary ghi theo kiểu nhị phân (mặc định theo kiểu văn bản) ios::out ghi tệp, nếu tệp đã có thì nó bị xoá

ios::app ghi bổ sung vào cuối tệp

ios::ate chuyển con trỏ tệp tới cuối tệp sau khi mở tệp ios::trunc xoá nội dung của tệp nếu nó tồn tại ios::nocreate nếu tệp chưa có thì không làm gì (bỏ qua) ios::noreplace nếu tệp đã có thì không làm gì (bỏ qua)

+ Tham số thứ ba prot quy định cấp bảo vệ của dòng tin, tham số này có thể bỏ qua vì nó đã được gán một giá trị mặc định.

* + 1. Hàm tạo:

## ofstream(int fd);

dùng để tạo một đối tượng ofstream và gắn nó với một tệp có chỉ số fd đang mở.

(Để mở và lấy chỉ số (số hiệu) tệp có thể dùng hàm \_open, xem cuốn Kỹ thuật Lập trình C của tác giả).

* + 1. Hàm tạo:

## ofstream(int fd, char \*buf, int n);

dùng để tạo một đối tượng ofstream , gắn nó với một tệp có chỉ số fd đang mở và sử dùng một vùng nhớ n byte do buf trỏ tới làm bộ đệm.

* + 1. Phương thức:

## void open(const char \*fn, int mode = ios::out,

**int prot = filebuf::openprot);**dùng để mở tệp có tên fn để ghi và gắn nó với đối tượng ofstream. Các tham số của phương thức có cùng ý nghĩa như trong hàm tạo thứ 2.

## Các cách ghi tệp

Có 2 cách chính sau:

+ *Cách 1:* Dùng hàm tạo 2 để xây dựng một dòng xuất, mở một tệp để ghi và gắn tệp với dòng xuất. Sau đó dùng toán tử xuất << và các phương thức để xuất dữ liệu ra dòng xuất vừa tạo như thể xuất dữ liệu ra cout (xem các mục trên).

+ *Cách 2:* Dùng hàm tạo 1 để xây dựng một dòng xuất. Sau đó dùng phương thức open để mở một tệp cụ thể và cho gắn với dòng xuất vừa xây dựng. Khi không cần làm việc với tệp này nữa, chúng ta có thể dùng phương thức close để chấm dứt mọi ràng buộc giữa dòng xuất và tệp. Sau đó có thể gắn dòng xuất với tệp khác. Theo cách này, có thể dùng một dòng xuất (đối tượng ofstream) để xuất dữ liệu lên nhiều tệp khác nhau.

## Ví dụ

**Chương trình 1:** Chương trình dưới đây sẽ nhập danh sách n thí sinh. Thông tin thí sinh gồm: Họ tên, tỉnh hoặc thành phố cư trú, số báo danh, các điểm toán lý hoá. Dữ liệu thí sinh được ghi trên 2 tệp: Tệp DS1.DL ghi thí sinh theo thứ tự nhập từ bàn phím, tệp DS2.DL ghi thí sinh theo thứ tự giảm của tổng điểm. Cấu trúc của 2 tệp như sau:

Dòng đầu ghi một số nguyên bằng số thí sinh.

Các dòng tiếp theo ghi dữ liệu của thí sinh. Mỗi thí sinh ghi trên 2 dòng, dòng 1 ghi họ tên trên 24 vị trí và tên tỉnh trên 20 vị trí. Dòng 2 ghi số báo danh (6 vị trí), các điểm toán, lý , hoá và tổng điểm (mỗi điểm ghi trên 6 vị trí trong đó một vị trí chứa phần phân). Chương trình sử dụng lớp TS (Thí sinh) có 3 phương thức: Nhập, sắp xếp và ghi tệp. Cách ghi tệp sử dụng ở đây là cách 1: Dùng hàm tạo dạng 2 của lớp ofstream.

Chương trình 2 ngay bên dưới cũng giải quyết cùng bài toán nêu trên nhưng sử dụng cách ghi tệp thứ 2 (dùng hàm tạo 1 và phương thức open)

Một điều đáng nói ở đây là việc nhập một chuỗi ký tự (như họ tên và tên tỉnh) bằng các phương thức get hoặc getline chưa được thuận tiện, vì 2 lý do sau: thứ nhất là các phương thức này có thể bị ký tự chuyển dòng (còn sót trên cin) làm trôi. Thứ hai là các phương thức này có thể để lại một số ký tự trên dòng cin (nếu số ký tự gõ nhiều hơn so với quy định) và các ký tự này sẽ gây ảnh hưởng đến các phép nhập tiếp theo. Để khắc phục các nhược điểm trên, chúng ta đưa vào 2 chương trình trên hàm getstr để nhập chuỗi ký tự từ bàn phím.

*//CT7\_10.CPP*

*// Ghi Tep*

*#include <iostream.h> #include <iomanip.h> #include <fstream.h> #include <conio.h> #include <stdlib.h> #include <ctype.h>*

*void getstr(char \*str,int n)*

*{*

*char tg[21];*

*while(1) // Bỏ qua Enter và nhập tối đa n-1 ký tự*

*{*

*cin.get(str,n);*

*if (str[0]) break;*

*else*

*cin.ignore();*

*}*

*while(1) // Loại các ký tự còn lại ra khỏi dòng nhập cin*

*{*

*cin.get(tg,20);*

*if (tg[0]==0)*

*{*

*cin.ignore(); break;*

*}*

*}*

*}*

*struct TSINH*

*{*

*char ht[25]; char ttinh[21]; int sobd;*

*float dt,dl,dh,td;*

*} ;*

*class TS*

*{*

*private:*

*int sots;*

*TSINH \*ts;*

*public:*

*TS()*

*{*

*sots=0;*

*ts = NULL;*

*}*

*void nhap(); void sapxep();*

*void ghitep(char \*ttep);*

*};*

*void TS::nhap()*

*{*

*cout << "\n So thi sinh: " ; cin >> sots ;*

*int n=sots;*

*ts = new TSINH[n+1]; for (int i=1; i<=n; ++i)*

*{*

*cout << "\n Nhap thi sinh thu: " << i << endl; cout << "Ho ten: " ;*

*getstr(ts[i].ht,25);*

*cout << "Tinh hoac thanh pho: " ; getstr(ts[i].ttinh,21);*

*cout << "So bao danh: " ; cin >> ts[i].sobd ;*

*cout << "Cac diem toan, ly, hoa: " ; cin >> ts[i].dt >> ts[i].dl >> ts[i].dh ; ts[i].td =ts[i].dt + ts[i].dl + ts[i].dh ;*

*}*

*}*

*void TS::sapxep()*

*{*

*int n = sots;*

*for (int i=1; i< n; ++i)*

*for (int j=i+1; j<= n; ++j) if (ts[i].td < ts[j].td)*

*{*

*TSINH tg = ts[i]; ts[i] = ts[j];*

*ts[j] = tg;*

*}*

*}*

*void TS::ghitep(char \*ttep)*

*{*

*ofstream f(ttep); f << sots ;*

*f << setprecision(1) << setiosflags(ios::showpoint); for (int i=1; i<=sots; ++i)*

*{*

*f << endl << setw(24) << ts[i].ht << setw(20) << ts[i].ttinh ;*

*f << endl << setw(6) << ts[i].sobd*

*<< setw(6) << ts[i].dt*

*<< setw(6) << ts[i].dl*

*<< setw(6) << ts[i].dh*

*<< setw(6) << ts[i].td ;*

*}*

*f.close();*

*}*

*void main()*

*{*

*clrscr(); TS t;*

*t.nhap(); t.ghitep("DS1.DL"); t.sapxep(); t.ghitep("DS2.DL");*

*cout << "\n Hoan thanh"; getch();*

*}*

## Chương trình 2:

*//CT7\_11.CPP*

*// Ghi Tep*

*#include <iostream.h> #include <iomanip.h> #include <fstream.h> #include <conio.h> #include <stdlib.h> #include <ctype.h>*

*void getstr(char \*str,int n)*

*{*

*char tg[21]; while(1)*

*{*

*cin.get(str,n);*

*if (str[0]) break;*

*else*

*cin.ignore();*

*}*

*while(1)*

*{*

*cin.get(tg,20);*

*if (tg[0]==0)*

*{*

*cin.ignore(); break;*

*}*

*}*

*}*

*struct TSINH*

*{*

*char ht[25]; char ttinh[21]; int sobd;*

*float dt,dl,dh,td;*

*} ;*

*class TS*

*{*

*private:*

*int sots;*

*TSINH \*ts;*

*public:*

*TS()*

*{*

*sots=0;*

*ts = NULL;*

*}*

*void nhap(); void sapxep();*

*void ghitep(char \*ttep);*

*};*

*void TS::nhap()*

*{*

*cout << "\n So thi sinh: " ; cin >> sots ;*

*int n=sots;*

*ts = new TSINH[n+1]; for (int i=1; i<=n; ++i)*

*{*

*cout << "\n Nhap thi sinh thu: " << i << endl; cout << "Ho ten: " ;*

*getstr(ts[i].ht,25);*

*cout << "Tinh hoac thanh pho: " ; getstr(ts[i].ttinh,21);*

*cout << "So bao danh: " ; cin >> ts[i].sobd ;*

*cout << "Cac diem toan, ly, hoa: " ; cin >> ts[i].dt >> ts[i].dl >> ts[i].dh ; ts[i].td =ts[i].dt + ts[i].dl + ts[i].dh ;*

*}*

*}*

*void TS::sapxep()*

*{*

*int n = sots;*

*for (int i=1; i< n; ++i)*

*for (int j=i+1; j<= n; ++j) if (ts[i].td < ts[j].td)*

*{*

*TSINH tg = ts[i]; ts[i] = ts[j];*

*ts[j] = tg;*

*}*

*}*

*void TS::ghitep(char \*ttep)*

*{*

*ofstream f; f.open(ttep,ios::out|ios::noreplace); if (f.bad())*

*{*

*cout << "\nTep " << ttep << " da ton tai"; cout << "\nCo ghi de? - C/K";*

*int ch=getch();*

*if (toupper(ch)=='C')*

*{*

*f.close();*

*f.open(ttep) ;*

*}*

*else*

*exit(1);*

*}*

*f << sots ;*

*f << setprecision(1) << setiosflags(ios::showpoint); for (int i=1; i<=sots; ++i)*

*{*

*f << endl << setw(24) << ts[i].ht << setw(20) << ts[i].ttinh ; f << endl << setw(6) << ts[i].sobd*

*<< setw(6) << ts[i].dt*

*<< setw(6) << ts[i].dl*

*<< setw(6) << ts[i].dh*

*<< setw(6) << ts[i].td ;*

*}*

*f.close();*

*}*

*void main()*

*{*

*clrscr(); TS t;*

*t.nhap(); t.ghitep("DS1.DL"); t.sapxep(); t.ghitep("DS2.DL");*

*cout << "\n Hoan thanh"; getch();*

*}*

## Bài 12. Đọc dữ liệu từ tệp

* 1. **Lớp ifstream**

Để đọc dữ liệu từ tệp chúng ta sử dụng lớp ifstream. Lớp ifstream thừa kế các phương thức của các lớp ios và istream. Nó cũng thừa kế phương thức:

## close

của lớp fstreambase. Ngoài ra lớp ifstream có thêm các hàm tạo và các phương thức sau:

* + 1. Hàm tạo:

**ifstream() ;** // Không đối

dùng để tạo một đối tượng ifstream (dòng nhập), chưa gắn với tệp.

* + 1. Hàm tạo:

## ifstream(const char \*fn, int mode = ios::in, int prot = filebuf::openprot);

dùng để tạo một đối tượng ifstream, mở tệp có tên fn để đọc và gắn đối tượng vừa tạo với tệp được mở.

+ Tham số fn cho biết tên tệp.

+ Tham số mode có giá trị mặc định là ios::in (mở để đọc). Tham số này có thể là một hợp của các giá trị sau:

ios::binary đọc theo kiểu nhị phân (mặc định theo kiểu văn bản) ios::ate chuyển con trỏ tệp tới cuối tệp sau khi mở tệp

+ Tham số thứ ba prot quy định cấp bảo vệ của dòng tin, tham số này có thể bỏ qua vì nó đã được gán một giá trị mặc định.

* + 1. Hàm tạo:

## ifstream(int fd);

dùng để tạo một đối tượng ifstream và gắn nó với một tệp có chỉ số fd đang mở.

(Để mở và lấy chỉ số (số hiệu) tệp có thể dùng hàm \_open, xem cuốn Kỹ thuật Lập trình C của tác giả)

* + 1. Hàm tạo:

## ifstream(int fd, char \*buf, int n);

dùng để tạo một đối tượng ifstream , gắn nó với một tệp có chỉ số fd đang mở và sử dùng một vùng nhớ n byte do buf trỏ tới làm bộ đệm.

* + 1. Phương thức:

## void open(const char \*fn, int mode = ios::in, int prot = filebuf::openprot);

dùng để mở tệp có tên fn để đọc và gắn nó với đối tượng ifstream. Các tham số của phương thức có cùng ý nghĩa như trong hàm tạo thứ 2.

## Các cách đọc tệp

Có 2 cách chính sau:

*+ Cách 1:* Dùng hàm tạo 2 để xây dựng một dòng nhập, mở một tệp để đọc và gắn tệp với dòng nhập. Sau đó dùng toán tử nhập >> và các phương thức để nhập dữ liệu từ dòng nhập vừa tạo như thể nhập dữ liệu từ cin (xem các mục trên)

+ *Cách 2:* Dùng hàm tạo 1 để xây dựng một dòng nhập. Sau đó dùng phương thức open để mở một tệp cụ thể và cho gắn với dòng nhập vừa xây dựng. Khi không cần làm việc với tệp này nữa, chúng ta có thể dùng phương thức close để chấm dứt mọi ràng buộc giữa dòng nhập và tệp. Sau đó có thể gắn dòng nhập với tệp khác. Theo cách này, có thể dùng một dòng nhập (đối tượng ifstream) để nhập dữ liệu từ nhiều tệp khác nhau.

## Kiểm tra sự tồn tại của tệp, kiểm tra cuối tệp

+ Khi mở một tệp để đọc mà tệp không tồn tại thì sẽ phát sinh lỗi, khi đó phương thức bad trả về giá trị khác không. Ví dụ để kiểm tra xem tệp DSTS (Danh sách thí sinh) đã tồn tại hay không có thể dùng đoạn chương trình:

ifstream fin(“DSTS”); if (fin.bad())

{

cout << “\nTep DSTS không tồn tai”; exit(1);

}

+ Trong quá trình đọc, con trỏ tệp sẽ chuyển dần về cuối tệp. Khi con trỏ tệp đã ở cuối tệp (hết dữ liệu) mà vẫn thực hiện một lệnh đọc thì phương thức eof sẽ cho giá trị khác không. Chương trình

dưới đây dùng phương thức eof để xác định độ dài (số byte) của tệp TC.EXE (chú ý cần dùng kiểu đọc nhị phân):

*//CT7\_14.CPP*

*// Do dai tep*

*#include <iostream.h> #include <fstream.h> #include <conio.h> #include <stdlib.h> void main()*

*{*

*clrscr();*

*long dd=0; char ch;*

*ifstream f("TC.EXE",ios::in | ios::binary); if (f.bad())*

*{*

*cout << "\nTep TC.EXE khong ton tai"; getch();*

*exit(1);*

*}*

*while(f.get(ch),!f.eof()) ++dd;*

*cout << "\n Do dai TC.EXE: " << dd; getch();*

*}*

## Ví dụ

Chương trình dưới đây sẽ:

+ Đọc danh sách thí sinh từ tệp DS1.DL do chương trình trong muc bài 11 tạo ra.

+ In danh sách thí sinh vừa đọc.

+ Sắp xếp dẫy thí sinh (vừa nhập từ tệp) theo thứ tự giảm của tổng điểm.

+ Ghi danh sách thí sinh sau khi sắp xếp lên tệp DS3.DL

+ Đọc danh sách thí sinh từ tệp DS3.DL

+ In danh sách thí sinh đọc từ tệp DS3.DL

Chương trình sử dụng lớp TS (Thí sinh) có 4 phương thức: void xuat();

void sapxep();

void ghitep(char \*ttep); void doctep(char \*ttep);

*//CT7\_12.CPP*

*// Doc tep*

*#include <iostream.h> #include <iomanip.h>*

*#include <fstream.h> #include <conio.h> #include <stdlib.h> #include <ctype.h> struct TSINH*

*{*

*char ht[25]; char ttinh[21]; int sobd;*

*float dt,dl,dh,td;*

*} ;*

*class TS*

*{*

*private:*

*int sots;*

*TSINH \*ts;*

*public:*

*TS()*

*{*

*sots=0;*

*ts = NULL;*

*}*

*void xuat(); void sapxep();*

*void ghitep(char \*ttep); void doctep(char \*ttep);*

*};*

*void TS::xuat()*

*{*

*cout << "\n\nSo thi sinh: " << sots;*

*cout << setprecision(1) << setiosflags(ios::showpoint); for (int i=1; i<=sots; ++i)*

*{*

*cout << "\nThi sinh thu: " << i ; cout << "\nHo ten: " << ts[i].ht ;*

*cout << "\nTinh - thanh pho: " << ts[i].ttinh ; cout << "\nSo bao danh: " << ts[i].sobd ; cout << "\nCac diem toan, ly, hoa: "*

*<< setw(5) << ts[i].dt*

*<< setw(5) << ts[i].dl*

*<< setw(5) << ts[i].dh ;*

*cout << "\nTong diem: " << ts[i].td ;*

*}*

*}*

*void TS::sapxep()*

*{*

*int n = sots;*

*for (int i=1; i< n; ++i)*

*for (int j=i+1; j<= n; ++j) if (ts[i].td < ts[j].td)*

*{*

*TSINH tg = ts[i]; ts[i] = ts[j];*

*ts[j] = tg;*

*}*

*}*

*void TS::ghitep(char \*ttep)*

*{*

*ofstream f; f.open(ttep,ios::out|ios::noreplace); if (f.bad())*

*{*

*cout << "\nTep " << ttep << " da ton tai"; cout << "\nCo ghi de? - C/K";*

*int ch=getch();*

*if (toupper(ch)=='C')*

*{*

*f.close();*

*f.open(ttep) ;*

*}*

*else*

*exit(1);*

*}*

*f << sots ;*

*f << setprecision(1) << setiosflags(ios::showpoint); for (int i=1; i<=sots; ++i)*

*{*

*f << endl << setw(24) << ts[i].ht << setw(20) << ts[i].ttinh ; f << endl << setw(6) << ts[i].sobd*

*<< setw(6) << ts[i].dt*

*<< setw(6) << ts[i].dl*

*<< setw(6) << ts[i].dh*

*<< setw(6) << ts[i].td ;*

*}*

*f.close();*

*}*

*void TS::doctep(char \*ttep)*

*{*

*ifstream f; f.open(ttep);*

*if (f.bad())*

*{*

*cout << "\nTep " << ttep << " khong ton tai"; getch();*

*exit(1);*

*}*

*f >> sots ; f.ignore();*

*if (ts!=NULL) delete ts; ts = new TSINH[sots+1];*

*for (int i=1; i<=sots; ++i)*

*{*

*f.get(ts[i].ht,25).get(ts[i].ttinh,21); ;*

*f >> ts[i].sobd >> ts[i].dt >> ts[i].dl*

*>> ts[i].dh >> ts[i].td ; f.ignore();*

*}*

*f.close();*

*}*

*void main()*

*{*

*clrscr(); TS t;*

*t.doctep("DS1.DL"); t.xuat();*

*t.sapxep(); t.ghitep("DS3.DL");*

*t.doctep("DS3.DL"); t.xuat();*

*cout << "\n Hoan thanh"; getch();*

*}*

## Bài 13. Đọc ghi đồng thời trên tệp

* 1. **Lớp fstream**

Để đọc ghi đồng thời trên tệp, chúng ta sử dụng lớp fstream. Lớp fstream thừa kế các phương thức của các lớp ofstream và ifstream. Ngoài ra lớp fstream có các hàm tạo và phương thức sau:

* + 1. Hàm tạo:

**fstream() ;** // Không đối

dùng để tạo một đối tượng fstream (dòng nhập-xuất), chưa gắn với tệp.

* + 1. Hàm tạo:

## fstream(const char \*fn, int mode,

**int prot = filebuf::openprot);**

dùng để tạo một đối tượng fstream, mở tệp có tên fn và gắn đối tượng vừa tạo với tệp được mở.

+ Tham số fn cho biết tên tệp.

+ Tham số mode quy định các kiểu truy nhập và có thể là tổ hợp của các giá trị sau: ios::binary đọc-ghi theo kiểu nhị phân (mặc định theo kiểu văn bản).

ios::out ghi tệp, nếu tệp đã có thì nó bị xoá

ios::in đọc tệp

ios::app ghi bổ sung vào cuối tệp

ios::ate chuyển con trỏ tệp về cuối sau khi mở ios::trunc xoá nội dung của tệp nếu nó tồn tạI ios::nocreate nếu tệp chưa có thì không làm gì (bỏ qua) ios::noreplace nếu tệp đã có thì không làm gì (bỏ qua)

### Chú ý:

+ Tham số mode không có giá trị mặc định.

+ Tham số thứ ba prot quy định cấp bảo vệ của dòng tin, tham số này có thể bỏ qua vì nó đã được gán một giá trị mặc định.

* + 1. Hàm tạo:

## fstream(int fd);

dùng để tạo một đối tượng fstream và gắn nó với một tệp có chỉ số fd đang mở.

(Để mở và lấy chỉ số (số hiệu) tệp có thể dùng hàm \_open, xem cuốn Kỹ thuật Lập trình C của tác giả)

* + 1. Hàm tạo:

## fstream(int fd, char \*buf, int n);

dùng để tạo một đối tượng fstream , gắn nó với một tệp có chỉ số fd đang mở và sử dùng một vùng nhớ n byte do buf trỏ tới làm bộ đệm.

* + 1. Phương thức:

## void open(const char \*fn, int mode,

**int prot = filebuf::openprot);**

dùng để mở tệp có tên fn và gắn nó với đối tượng fstream. Các tham số của phương thức có cùng ý nghĩa như trong hàm tạo thứ 2.

***Chú ý:*** Tham số mode không có giá trị mặc định.

## Các cách đọc-ghi đồng thời trên tệp

Có 2 cách chính sau:

*+ Cách 1:* Dùng hàm tạo 2 để xây dựng một dòng nhập-xuất, mở một tệp để đọc-ghi và gắn tệp với dòng nhập-xuất. Sau đó dùng toán tử nhập >> , toán tử xuất >> và các phương thức nhập, xuất để nhập, xuất dữ liệu ra dùng nhập-xuất vừa tạo (như đối với các dòng chuẩn cin và cout). Ví dụ:

fstream f(“DU\_LIEU”, ios::in | ios::out) ;

+ *Cách 2:* Dùng hàm tạo 1 để xây dựng một dòng nhập-xuất. Sau đó dùng phương thức open để mở một tệp cụ thể (để đọc và ghi) và cho gắn với dòng nhập-xuất vừa xây dựng. Khi không cần làm việc với tệp này nữa, chúng ta có thể dùng phương thức close để chấm dứt mọi ràng buộc giữa dòng nhập-xuất và tệp. Sau đó có thể gắn dòng nhập-xuất với tệp khác. Theo cách này, có thể dùng một dòng nhập-xuất (đối tượng fstream) để đọc-ghi dữ liệu từ nhiều tệp khác nhau.

## Ví dụ:

fstream f;

f.open(“DU\_LIEU”, ios::in | ios::out) ;

## Di chuyển con trỏ tệp

* + 1. **Để di chuyển con trỏ tệp trên dòng xuất,** chúng ta sử dụng các phương thức sau (của lớp ostream) :

1. Phương thức

## ostream& seekp(long n) ;

sẽ chuyển con trỏ tệp tới vị trí (byte) thứ n (số thứ tự các byte tính từ 0).

1. Phương thức

## ostream& seekp(long offset, seek\_dir dir) ;

sẽ chuyển con trỏ tệp tới vị trí **offset** kể từ vị trí xuất phát **dir**. Giá trị của **offset** có thể âm, còn **dir**

có thể nhận một trong các giá trị sau: ios::beg xuất phát từ đầu tệp ios::end xuất phát từ cuối tệp

ios::cur xuất phát từ vị trí hiện tại của con trỏ tệp

1. Phương thức

## long teelp() ;

cho biết vị trí hiện tại của con trỏ tệp.

* + 1. **Để di chuyển con trỏ tệp trên dòng nhập,** chúng ta sử dụng các phương thức sau (của lớp istream):

1. Phương thức

## istream& seekg(long n) ;

sẽ chuyển con trỏ tệp tới vị trí (byte) thứ n (số thứ tự các byte tính từ 0)

1. Phương thức

## istream& seekg(long offset, seek\_dir dir) ;

sẽ chuyển con trỏ tệp tới vị trí **offset** kể từ vị trí xuất phát **dir**. Giá trị của **offset** có thể âm, còn **dir**

có thể nhận một trong các giá trị sau:

ios::beg xuất phát từ đầu tệp ios::end xuất phát từ cuối tệp

ios::cur xuất phát vị trí hiện tại của con trỏ tệp

1. Phương thức

## long teelg() ;

cho biết vị trí hiện tại của con trỏ tệp.

* + 1. **Để di chuyển con trỏ tệp trên dòng nhập-xuất,** chúng ta có thể sử dụng cả 6 phương thức nêu trên.

## Ví dụ

**Ví dụ 1.** Trong §12 đã viết chương trình xác định độ dài của tệp TC.EXE. Dưới đây là một phương án khác đơn giản hơn:

fstream f(“TC.EXE”);

f.seekg(0,ios::end); do\_dai = f.teelg();

**Ví dụ 2.** Chương trình dưới đây sẽ nhập danh sách n thí sinh từ bàn phím và ghi lên tệp. Sau đó đưa con trỏ tệp về đầu tệp và bắt đầu đọc dữ liệu thí sinh từ tệp để in ra màn hình. Thông tin thí sinh gồm: Họ tên, tỉnh hoặc thành phố cư trú, số báo danh, các điểm toán lý hoá.

*//CT7\_13.CPP*

*// ghi - đọc đồng thời #include <iostream.h> #include <iomanip.h> #include <fstream.h> #include <conio.h> #include <stdlib.h> #include <ctype.h> #include <stdio.h> void main()*

*{*

*char ht[25], ttinh[21], ttep[40]; int sobd,stt ;*

*float dt, dl, dh, td; fstream f;*

*cout << "\nTen tep: " ; cin >> ttep;*

*f.open(ttep,ios::out|ios::in|ios::noreplace); if (f.bad())*

*{*

*cout << "\nTep " << ttep << " da ton tai"; cout << "\nCo ghi de? - C/K";*

*int ch=getch();*

*if (toupper(ch)=='C')*

*{*

*f.close(); f.open(ttep,ios::out|ios::in|ios::trunc) ;*

*}*

*else*

*exit(1);*

*}*

*stt=0 ;*

*f << setprecision(1) << setiosflags(ios::showpoint); while(1)*

*{*

*++stt;*

*cout << "\nNhap thi sinh thu: " << stt ;*

*cout << "\nHo ten (neu rong thi ket thuc nhap) : "; cin.ignore();*

*cin.getline(ht,25);*

*if (ht[0]==0) break;*

*cout << "Tinh - thanh pho: "; cin.getline(ttinh,21);*

*cout << "SoBD, diem toan, diem ly, diem hoa: " ; cin >> sobd >> dt>> dl >> dh ;*

*td = dt + dl + dh ; if (stt>1) f << endl;*

*f << setw(24) << ht << setw(20) << ttinh ; f << endl << setw(6) << sobd*

*<< setw(6) << dt*

*<< setw(6) << dl*

*<< setw(6) << dh*

*<< setw(6) << td ;*

*}*

*f.seekg(0); stt=0; clrscr();*

*cout << "\nDanh sach thi sinh\n"; cout << setprecision(1) << setiosflags(ios::showpoint); while(1)*

*{*

*f.getline(ht,25).getline(ttinh,21); if (f.eof()) break;*

*++stt;*

*f >> sobd >> dt >> dl >> dh >> td; f.ignore();*

*cout << "\nThi sinh thu: " << stt ; cout << "\nHo ten: "<< ht;*

*cout << " \nTinh - thanh pho: " << ttinh; cout << "\nSo bao danh: " << sobd;*

*cout << "\nDiem toan, ly, hoa va tong diem: "*

*<<setw(6)<< dt << setw(6) <<dl << setw(6) << dh*

*<< setw(6) << td ;*

*}*

*f.close();*

*cout << "\n Hoan thanh"; getch();*

*}*

## Bài 14. Xử lý lỗi

Khi làm việc với tệp không phải mọi việc đều trôi chẩy mà thường xẩy ra nhiều điều trục trặc, chẳng hạn:

* + 1. Mở một tệp để đọc nhưng tệp không tồn tại.
    2. Đọc dữ liệu nhưng con trỏ tệp đã ở cuối tệp
    3. Ghi dữ liệu nhưng hết không gian đĩa (đĩa đầy).
    4. Tạo tệp nhưng đia hỏng, hoặc đĩa cấm ghi hoặc đĩa đầy.
    5. Dùng tên tệp không hợp lệ
    6. Định thực hiện một thao tác nhưng tệp lại không được mở ở mode phù hợp để thực hiện thao tác đó.

Tóm lại khi làm việc với tệp thường gặp nhiều lỗi khác nhau, nếu không biết cách phát hiện xử lý thì chương trình sẽ dẫn đến rối loạn hoặc cho kết quả sai. Trong lớp ios của C++ có nhiều phương thức cho phép phát hiện lỗi khi làm việc với tệp. Đó là:

1. Phương thức

## int eof() ;

Nếu con trỏ tệp đã ở cuối tệp mà lại thực hiện một lệnh đọc dữ liệu thì phương thức eof() trả về giá trị khác không, trái lại phương thức có giá trị bằng 0.

1. Phương thức

## int fail() ;

Phương thức fail() trả về giá trị khác không nếu lần nhập xuất cuối cùng có lỗi, trái lại phương thức có giá trị bằng 0.

1. Phương thức

## int bad() ;

Phương thức bad() trả về giá trị khác không khi một phép nhập xuất không hợp lệ hoặc có lỗi mà chưa phát hiện được, trái lại phương thức có giá trị bằng 0.

1. Phương thức

## int good() ;

Phương thức good() trả về giá trị khác không nếu mọi việc đều tốt đẹp ( không có lỗi nào xẩy ra). Khi có một lỗi nào đó thì phương thức có giá trị bằng 0.

1. Phương thức

## void clear() ;

dùng để tắt tất cả các bit lỗi.

**Ví dụ 1.** Khi mở tệp để đọc cần kiểm tra xem tệp có tồn tại không. Để làm điều đó, chúng ta có thể dùng đoạn chương sau:

char ten\_tep[40] ;

cout << “\n Cho biết tên tệp: “ ; cin >> ten\_tep ;

ifstream f(ten\_tep); if (f.bad())

{

cout << “\n Tệp << ten\_tep << “không tồn tại” ; exit(1) ;

}

**Ví dụ 2.** Khi tạo tệp mới để ghi cần kiểm tra xem có tạo được tệp hay không. Để làm điều đó, chúng ta có thể dùng đoạn chương sau:

char ten\_tep[40] ;

cout << “\n Cho biết tên tệp: “ ; cin >> ten\_tep ;

ofstream f(ten\_tep); if (f.bad())

{

cout << “\n Không tạo được tệp << ten\_tep ; exit(1) ;

}

**Ví dụ 3.** Để xác định độ dài của tệp, có thể dùng phương thức eof() và thuật toán sau:

+ Đọc một byte (chú ý phải đọc theo kiểu nhị phân)

+ Nếu việc đọc thành công ( eof()=0 ) thì cộng thêm một vào bộ đếm. Nếu việc đọc không thành ( eof() != 0 ) thì kết thúc vùng lặp.

Thuật toán trên được thể hiện bằng đoạn chương trình sau: ifstream f(ten\_tep, ios::in | ios::binary) ;

long dem = 0 ; char ch; while (1)

{

f.get(ch) ;

if (!eof()) dem++ ; else break;

}

## Bài 15. Nhập xuất nhị phân

* 1. **Chọn kiểu nhập xuất nhị phân**

Kiểu nhập xuất mặc định là văn bản. Để chọn kiểu nhập xuất nhị phân, thì trong tham số mode (của hàm tạo dạng 2 và phương thức open) cần chứa giá trị:

ios::binary

**Ví dụ** muốn mở tệp “DSTS.DL” để đọc-ghi theo kiểu nhị phân và gắn tệp với dòng nhập-xuất fs

, ta dùng câu lệnh sau:

fstream fs(“DSTS.DL” , ios::in | ios::out | ios::binary) ;

## Đọc, ghi ký tự

+ Để ghi một ký tự lên tệp có thể dùng phương thức: ostream & put(char) ;

+ Để đọc một ký tự từ tệp có thể dùng phương thức: istream & get(char &) ;

***Cần chú ý rằng*:** Cách đọc ghi ký tự theo kiểu văn bản khác với cách đọc ghi ký tự theo kiểu nhị phân (xem chương 10, cuốn Kỹ thuật lập trình C của tác giả)

**Ví dụ** để sao tệp có thể dùng thuật toán đơn giản sau:

+ Đọc một ký tự từ tệp nguồn

+ Nếu đọc thành công ( phương thức eof() = 0) thì ghi lên tệp đích và lại tiếp tục đọc ký tự tiếp theo.

+ Nếu đọc không thành công ( phương thức eof() != 0) thì kết thúc.

***Chú ý*** là phải dùng kiểu nhập xuất nhị phân thì thuật toán mới cho kết quả chính xác. Chương trình sao tệp dưới đây viết theo thuật toán trên.

*//CT7\_15.CPP*

*// Sao tep*

*#include <iostream.h> #include <fstream.h> #include <conio.h> #include <stdlib.h> void main()*

*{*

*clrscr();*

*char tep\_nguon[40], tep\_dich[40] ; char ch;*

*fstream fnguon, fdich;*

*cout << "\nTen tep nguon: " ; cin >> tep\_nguon; cout << "\nTen tep dich: " ; cin >> tep\_dich; fnguon.open(tep\_nguon,ios::in | ios::binary); fdich.open(tep\_dich,ios::out | ios::binary);*

*if (fnguon.bad() || fdich.bad() )*

*{*

*cout << "\n Loi mo tep nguon hoac dich " ; getch();*

*exit(1);*

*}*

*while(fnguon.get(ch),!fnguon.eof()) fdich.put(ch) ;*

*fnguon.close(); fdich.close();*

*cout << "\nHoan thanh" ; getch();*

*}*

## Đọc, ghi một dẫy ký tự theo kiểu nhị phân

+ Phương thức:

ostream & write(char \*buf, int n) ;

sẽ xuất n ký tự (byte) chứa trong buf ra dòng xuất.

+ Phương thức:

istream & read(char \*buf, int n) ;

sẽ nhập n ký tự (byte) từ dòng nhập và chứa vào buf.

+ Phương thức int gcount

cho biết số ký tự thực sự đọc được trong phương thức read.

***Chú ý:*** Các phương thức write, read chỉ làm việc một cách chính xác trong kiểu nhập-xuất nhị phân.

Dưới đây là chương trình sao tệp sử dụng các phương thức write, read và gcount.

*//CT7\_16.CPP*

*// Sao tep dung write, read va gcount #include <iostream.h>*

*#include <fstream.h> #include <conio.h> #include <stdlib.h> void main()*

*{*

*clrscr();*

*char tep\_nguon[40], tep\_dich[40] ; char buf[5000];*

*int n;*

*fstream fnguon, fdich;*

*cout << "\nTen tep nguon: " ; cin >> tep\_nguon; cout << "\nTen tep dich: " ; cin >> tep\_dich; fnguon.open(tep\_nguon,ios::in | ios::binary);*

*fdich.open(tep\_dich,ios::out | ios::binary); if (fnguon.bad() || fdich.bad() )*

*{*

*cout << "\n Loi mo tep nguon hoac dich " ; getch();*

*exit(1);*

*}*

*while(fnguon.read(buf,5000),(n=fnguon.gcount())) fdich.write(buf,n) ;*

*fnguon.close(); fdich.close();*

*cout << "\nHoan thanh" ; getch();*

*}*

## Bài 16. Đọc ghi đồng thời theo kiểu nhị phân

Chương trình dưới đây minh hoạ cách đọc ghi đồng thời trên tệp theo kiểu nhị phân. Chương trình sử dụng các phương thức write, read, các phương thức di chuyển con trỏ tệp và các phương thức kiểm tra lỗi. Chương trình gồm 3 chức năng:

* + 1. Nhập một danh sách thí sinh mới và ghi vào tệp TS.DL
    2. Bổ sung thí sinh vào tệp TS.DL
    3. Xem sửa thí sinh trên tệp TS.DL

*//CT7\_18.CPP*

*// Doc tep*

*#include <iostream.h> #include <iomanip.h> #include <fstream.h> #include <conio.h> #include <stdlib.h> #include <ctype.h> #include <string.h> #include <stdio.h> struct TSINH*

*{*

*char ht[25]; int sobd; float td;*

*};*

*class TS*

*{*

*private:*

*TSINH ts;*

*char ten\_tep[40]; int sots;*

*static int size; public:*

*TS(char \*ttep); void tao\_ds(); void bo\_sung(); void xem\_sua();*

*};*

*int TS::size = sizeof(TSINH); TS::TS(char \*ttep)*

*{*

*strcpy(ten\_tep,ttep); fstream f;*

*f.open(ten\_tep,ios::binary|ios::in|ios::ate); if (!f.good())*

*sots = 0 ; else*

*{*

*sots=f.tellg()/size ;*

*}*

*}*

*void TS::tao\_ds()*

*{*

*fstream f; f.open(ten\_tep,ios::binary|ios::out|ios::noreplace); if (!f.good())*

*{*

*cout << "\nDanh sach da ton tai" ; cout << "\nCo tao lai khong? - C/K" ; char ch=getch();*

*if (toupper(ch) != 'C') return;*

*else*

*{*

*f.close(); f.open(ten\_tep,ios::binary|ios::out|ios::trunc);*

*}*

*}*

*sots=0; while(1)*

*{*

*cout << "\nThi sinh thu: " << (sots+1) ;*

*cout << "\nHo ten (Bam Enter de ket thuc): "; fflush(stdin);*

*gets(ts.ht);*

*if (ts.ht[0]==0) break; cout << "\nSo bao danh: "; cin >> ts.sobd;*

*cout << "\nTong diem: "; cin >> ts.td; f.write((char\*)(&ts),size) ; sots++ ;*

*}*

*f.close();*

*}*

*void TS::bo\_sung()*

*{*

*fstream f; f.open(ten\_tep,ios::binary|ios::app|ios::nocreate); if (!f.good())*

*{*

*cout << "\nDanh sach chua tao" ;*

*cout << "\nCo tao moi khong? - C/K" ; char ch=getch();*

*if (toupper(ch) != 'C') return;*

*else*

*{*

*f.close(); f.open(ten\_tep,ios::binary|ios::out);*

*}*

*}*

*int stt=0; while(1)*

*{*

*cout << "\nBo sung thi sinh thu: " << (stt+1); cout << "\nHo ten (Bam Enter de ket thuc): "; fflush(stdin);*

*gets(ts.ht);*

*if (ts.ht[0]==0) break; cout << "\nSo bao danh: ";*

*cin >> ts.sobd;*

*cout << "\nTong diem: "; cin >> ts.td; f.write((char\*)(&ts),size) ;*

*++stt;*

*}*

*sots += stt ; f.close();*

*}*

*void TS::xem\_sua()*

*{*

*fstream f; int ch; f.open(ten\_tep,ios::binary|ios::out|ios::in|ios::nocreate); if (!f.good())*

*{*

*cout << "\nDanh sach chua tao" ; getch();*

*return ;*

*}*

*cout << "\nDanh sach gom: " << sots << "thi sinh" ; int stt;*

*while(1)*

*{*

*cout << "\nCan xem-sua thi sinh thu (Bam 0 de ket thuc): " ; cin >> stt ;*

*if (stt<1 || stt > sots) break; f.seekg((stt-1)\*size,ios::beg);*

*f.read((char\*)(&ts),size);*

*cout << "\nHo ten : " << ts.ht;*

*cout << "\nSo ba danh: " << ts.sobd ; cout << "\nTong diem: " << ts.td ; cout << "\nCo sua khong? - C/K" ; ch=getch();*

*if (toupper(ch)=='C')*

*{*

*f.seekg(-size,ios::cur) ; cout << "\nHo ten: "; fflush(stdin); gets(ts.ht);*

*cout << "\nSo bao danh: "; cin >> ts.sobd;*

*cout << "\nTong diem: "; cin >> ts.td; f.write((char\*)(&ts),size) ;*

*}*

*}*

*f.close();*

*}*

*void main()*

*{*

*int chon; clrscr();*

*TS t("TS.DL");*

*while(1)*

*{*

*clrscr();*

*cout << "\n1. Tao danh sach thi sinh moi" ; cout << "\n2. Bo sung danh sach thi sinh" ; cout << "\n3. Xem-sua danh sach thi sinh" ; cout << "\n4. Ket thuc chuong trinh " ; chon = getch();*

*chon = chon - 48; clrscr();*

*if (chon==1) t.tao\_ds();*

*else if(chon==2) t.bo\_sung(); else if(chon==3) t.xem\_sua(); else break;*

*}*

*clrscr();*

*cout << "\n Hoan thanh"; getch();*

*}*

## Bài 17. Xây dựng toán tử nhâp xuất đối tượng trên tệp

Trong các mục trên đã trình bầy cách dùng các toán tử nhập >> và xuất << để ghi dữ liệu kiểu chuẩn (nguyên, thực, ký tự, chuỗi ký tự) trên tệp. Mục này trình bầy cách xây dựng các toán tử dùng để đọc ghi các đối tượng của một lớp bất kỳ do người dùng định nghĩa.

Giả sử chúng ta muốn sử dụng các toán tử nhập xuất để đọc ghi các đối tượng của lớp TS. Khi đó ta đưa vào các hàm bạn toán tử nhập xuất như sau:

class TS

{

private:

// Khai báo các thuộc tính

public:

friend fstream& operator<<(fstream& fs,const TS &t); friend fstream& operator>>(fstream& fs,TS &t);

...

} ;

**Về kiểu ghi:** Có thể xây dựng các toán tử để thực hiện các phép đọc ghi theo kiểu văn bản cũng như nhị phân.

**Ví dụ 1:** Ghi theo kiểu văn bản

Chương trình dưới đây minh hoạ cách xây dựng và sử dụng các toán tử nhập xuất đối tượng trên màn hình, bàn phím và tệp. Chương trình đưa vào lớp TS (Thí sinh) và các hàm toán tử cho phép nhập xuất các đối tượng TS trên màn hình, bàn phím và tệp. Chương trình gồm các nội dung sau:

+ Tạo tệp TS.DL dùng để đọc và ghi theo kiểu văn bản.

+ Nhập 3 thí sinh từ bàn phím và chứa vào 3 biến đối tượng t1, t2, t3.

+ Ghi nội dung của 3 biến đối tượng t1, t2, t3 lên tệp TS.DL

+ Đọc các đối tượng từ tệp TS.DL và chứa vào 3 biến t4, t5, t6

+ In các biến đối tượng t4, t5, t6 ra màn hình

+ Chuyển con trỏ về đầu tệp, dùng chu trình while để lần lượt đọc các đối tượng từ tệp và in ra màn hình. Dùng phương thức eof để kiểm tra xem đã đọc hết dữ liệu hay chưa.

*//CT7\_17.CPP*

*// Cac toan tu doc ghi doi tuong tren Tep #include <iostream.h>*

*#include <iomanip.h> #include <fstream.h> #include <conio.h> #include <stdlib.h> #include <ctype.h> class TS*

*{*

*private:*

*char ht[25]; float td;*

*public:*

*friend ostream& operator<<(ostream& os,const TS &t); friend istream& operator>>(istream& is,TS &t);*

*friend fstream& operator<<(fstream& fs,const TS &t); friend fstream& operator>>(fstream& fs,TS &t);*

*};*

*fstream& operator>>(fstream& fs,TS &t)*

*{*

*fs.getline(t.ht,25); fs >> t.td; fs.ignore();*

*return fs;*

*}*

*ostream& operator<<(ostream& os,const TS &t)*

*{*

*os << "\nHo ten: " << t.ht ;*

*os << "\nTong diem: " << t.td; return os;*

*}*

*fstream& operator<<(fstream& fs,const TS &t)*

*{*

*fs << t.ht << endl; fs << t.td << endl; return fs;*

*}*

*istream& operator>>(istream& is,TS &t)*

*{*

*cout << "\nHo ten: " ; is.get(t.ht,25);*

*cout << "Tong diem: " ; is >> t.td ;*

*is.ignore(); return is;*

*}*

*void main()*

*{*

*clrscr();*

*fstream f("TS.DL",ios::out | ios::in | ios::trunc); TS t1,t2,t3,t4,t5,t6,t;*

*cin >> t1 >> t2 >> t3; f << t1 << t2 <<t3;*

*f.seekg(0); f>>t4>>t5>>t6;*

*cout << t4 << t5 << t6; f.seekg(0);*

*while (f>>t ,!f.eof()) cout << t;*

*f.close();*

*cout << "\n Xong"; getch();*

*}*

**Ví dụ 2** : Ghi theo kiểu nhị phân

Chương trình dưới đây cũng có các chức năng như chương trình trong ví dụ 1 bên trên, nhưng cách ghi đọc tệp theo kiểu nhị phân.

*//CT7\_19.CPP*

*// Cac toan tu doc ghi doi tuong tren Tep*

*// Kieu nhi phan #include <iostream.h> #include <iomanip.h> #include <fstream.h> #include <conio.h> #include <stdlib.h> #include <ctype.h> class TS*

*{*

*private:*

*char ht[25]; float td; static int size;*

*public:*

*friend ostream& operator<<(ostream& os,const TS &t); friend istream& operator>>(istream& is,TS &t);*

*friend fstream& operator<<(fstream& fs,const TS &t); friend fstream& operator>>(fstream& fs,TS &t);*

*};*

*int TS::size= sizeof(TS);*

*fstream& operator>>(fstream& fs,TS &t)*

*{*

*fs.read( (char\*)(&t) , t.size); return fs;*

*}*

*fstream& operator<<(fstream& fs,const TS &t)*

*{*

*fs.write( (char\*)(&t) , t.size); return fs;*

*}*

*ostream& operator<<(ostream& os,const TS &t)*

*{*

*os << t.ht << endl; os << t.td << endl;*

*return os;*

301

*}*

*istream& operator>>(istream& is,TS &t)*

*{*

*cout << "\nHo ten: " ; is.get(t.ht,25);*

*cout << "Tong diem: " ; is >> t.td ;*

*is.ignore(); return is;*

*}*

*void main()*

*{*

*clrscr();*

*fstream f("THU.DL",ios::binary | ios::out|ios::in|ios::trunc); TS t1,t2,t3,t4,t5,t6,t;*

*cin >> t1 >> t2 >> t3; f << t1 << t2 <<t3;*

*f.seekg(0); f>>t4>>t5>>t6;*

*cout << t4 << t5 << t6; f.seekg(0);*

*while( f>>t ,!f.eof() ) cout << t;*

*f.close();*

*cout << "\n Xong"; getch();*

*}*

## Bài 18. Hệ thống các lớp stream

Mục này hệ thống lại các lớp stream mà chúng ta đã sử dụng bên trên để tổ chức xuất nhập trên màn hình, bàn phím, máy in và tệp

## Sơ đồ quan hệ giữa các lớp

istream

ostream

fstreambase

ios

GS: Phạm Văn Ất

[tenshi3003@gmail.com](mailto:tenshi3003@gmail.com)

ifstream

ofstream

fstream

## Các phương thức của lớp ios

* + 1. int bad()
    2. void clear(int=0)
    3. int eof()
    4. int fail()
    5. int fill()
    6. int fill(char)
    7. long flags()
    8. long flags(long)
    9. int good()
    10. int precision()
    11. int precision(int)
    12. long setf(long)
    13. long setf(long setbits, long field)
    14. long unsetf(long)
    15. int width()
    16. int width(int)

## Các phương thức của lớp istream

* + 1. operator>>
    2. int gcount()
    3. int get()
    4. istream& get(char\*, int, char = ‘\n’)
    5. istream& get(char&)
    6. istream& getline(char\*, int, char = ‘\n’)
    7. istream& ignore(int n = 1, int delim = EOF)
    8. int peek()
    9. istream& putback(char)
    10. istream& read(char\*, int)
    11. istream& seekg(long)
    12. istream& seekg(long, seek\_dir)
    13. long tellg()

## Các phương thức của lớp ostream

* + 1. operator<<
    2. ostream& flush()
    3. ostream& put(char)
    4. ostream& seekp(long)
    5. ostream& seekp(long, seek\_dir)
    6. long tellp()
    7. ostream& write(char\*, int)

## Các phương thức của lớp fstreambase

void close()

## Các phương thức của lớp ifstream

* + 1. ifstream()
    2. ifstream(const char\*, int = ios::in, int = filebuf::openprot)
    3. ifstream(int )
    4. ifstream(int , char\*, int)
    5. void open(const char\*, int = ios::in, int = filebuf::openprot)

## Các phương thức của lớp ofstream

* + 1. ofstream()
    2. ofstream(const char\*, int = ios::out, int = filebuf::openprot)
    3. ofstream(int )
    4. ofstream(int , char\*, int)
    5. void open(const char\*, int = ios::out, int = filebuf::openprot)

## Các phương thức của lớp fstream

* + 1. fstream()
    2. fstream(const char\*, int, int = filebuf::openprot)
    3. fstream(int )
    4. fstream(int , char\*, int)
    5. void open(const char\*, int, int = filebuf::openprot)

**Chương 8**

# Đồ họa

Trong chương này sẽ giới thiệu các hàm để vẽ các đường và hình cơ bản như đường tròn, cung elip, hình quạt, đường gẫy khúc, hình đa giác, đường thẳng, đường chữ nhật, hình chữ nhật, hình hộp chữ nhật, ... Ngoài ra còn đề cập tới các vấn đề rất lý thú khác như: xử lý văn bản trên màn hình đồ họa, cửa sổ và kỹ thuật tạo ảnh di động. Các hàm đồ họa được khai báo trong tệp graphics.h.

## Bài 1. Khái niệm đồ họa

Để hiểu kỹ thuật lập trình đồ họa, đầu tiên phải hiểu các yếu tố cơ bản của đồ họa. Từ trước đến nay chúng ta chủ yếu làm việc với kiểu văn bản. Nghĩa là màn hình được thiết lập để hiển thị 25 dòng, mỗi dòng có thể chứa 80 ký tự. Trong kiểu văn bản, các ký tự hiển thị trên màn hình đã được phần cứng của máy PC ấn định trước và ta không thể nào thay đổi được kích thước, kiểu chữ.

ở màn hình đồ họa, ta có thể xử lý đến từng chấm điểm (pixel) trên màn hình và do vậy muốn vẽ bất kỳ thứ gì cũng được. Sự bài trí và số pixel trên màn hình được gọi là độ phân giải (resolution). Do mỗi kiểu màn hình đồ họa có một cách xử lý đồ họa riêng nên TURBO C cung cấp một tệp tin điều khiển riêng cho từng kiểu đồ họa. Bảng 8-1 cho thấy các kiểu đồ họa và các tệp tin điều khiển chúng.

Ngoài các tệp có đuôi BGI chứa chương trình điều khiển đồ họa, TURBO C còn cung cấp các tệp tin đuôi CHR chứa các Font chữ để vẽ các kiểu chữ khác nhau trên màn hình đồ họa. Đó là các tệp:

GOTH.CHR LITT.CHR SANS.CHR TRIP.CHR

***Bảng 8-1.*** Các tệp tin điều khiển đồ họa của TURBO C++

|  |  |
| --- | --- |
| **Tên tệp tin** | **Kiểu màn hình đồ họa** |
| ATT.BGI | ATT & T6300 (400 dòng) |
| CGA.BGI | IBMCGA, MCGA và các máy tương thích |
| EGAVGA.BGI | IBM EGA, VGA và các máy tương thích |
| HERC.BGI | Hercules monochrome và các máy tương thích |
| IBM8514.BGI | IBM 8514 và các máy tương thích |
| PC3270.BGI | IBM 3270 PC |

Màn hình đồ họa gồm nhiều điểm ảnh được sắp xếp trên các đường thẳng ngang và dọc. Điều này đúng cho tất cả các kiểu màn hình đồ họa của máy tính. Khác biệt chủ yếu giữa chúng là kích thước và số các điểm ảnh. Trong kiểu CGA (độ phân giải thấp), điểm ảnh có kích thước lớn, chiều ngang có 320 điểm ảnh, còn theo chiều dọc có 200 điểm ảnh. Màn hình VGA có độ phân giải cao hơn: điểm ảnh nhỏ hơn, trên mỗi hàng có 640 điểm ảnh và trên mỗi cột có 480 điểm ảnh. Điểm ảnh càng nhỏ thì số điểm ảnh trên màn hình càng nhiều và chất lượng đồ họa càng cao.

Mỗi kiểu đồ họa dùng một hệ tọa độ riêng. Hệ tọa độ cho màn hình VGA là 640 x 480 như sau :

(0,0) (639,0)

(0,479) (639,479)

***Hình 8.1.*** Hệ tọa độ VGA

Nhờ hệ tọa độ này, ta có thể tác động hay tham chiếu đến bất kỳ điểm ảnh nào trên màn hình đồ họa.

Nếu dùng màn hình CGA thì góc dưới phải có tọa độ (319, 199). Độc lập với kiểu đồ họa đang sử dụng, các hàm getmaxx và getmaxy bao giờ cũng cho tọa độ x và y lớn nhất trong kiểu đồ họa đang dùng.

Một chương trình đồ họa thường gồm các phần sau:

* Khởi động hệ thống đồ họa.
* Xác định mầu nền (mầu màn hình), mầu đường vẽ, mầu tô và kiểu (mẫu) tô.
* Vẽ, tô mầu các hình mà ta mong muốn.
* Các thao tác đồ họa khác như cho hiện các dòng chữ...
* Đóng hệ thống đồ họa để trở về mode văn bản.

## Bài 2. Khởi động hệ đồ họa

Mục đích của việc khởi động hệ thống đồ họa là xác định thiết bị đồ họa (màn hình) và mốt đồ họa sẽ sử dụng trong chương trình. Để làm điều này ta dùng hàm:

void initgraph(int \*graphdriver, int \*graphmode,char \*driverpath);

trong đó: driverpath là đường dẫn của thư mục chứa các tệp tin điều khiển đồ họa, graphdriver, graphmode cho biết màn hình và mốt đồ họa sẽ sử dụng trong chương trình. Bảng 8-2 cho thấy các giá trị khả dĩ của graphdriver và graphmode.

**Ví dụ 1.** Giả sử máy tính của ta có màn hình EGA, các tệp tin đồ họa chứa trong thư mục C:

\TC, khi đó ta có thể khởi động hệ thống đồ họa như sau:

*#include "graphics.h" main()*

*{*

*int mh=EGA, mode= EGALO; initgraph(&mh, &mode, "C:\TC");*

*. . .*

*}*

***Bảng 8-2.*** Các giá trị khả dĩ của graphdriver, graphmode

|  |  |  |
| --- | --- | --- |
| **graphdriver** | **graphmode** | **Độ phân giải** |
| Detect (0) |  |  |
| CGA (1) | CGAC0 (0) | 320 x 200 |
|  | CGAC1 (1) | 320 x 200 |
|  | CGAC2 (2) | 320 x 200 |
|  | CGAC3 (3) | 320 x 200 |
|  | CGAHi (4) | 640 x 200 |
| MCGA (2) | MCGA0 (0) | 320 x 200 |
|  | MCGA1 (1) | 320 x 200 |
|  | MCGA2 (2) | 320 x 200 |
|  | MCGA3 (3) | 320 x 200 |
|  | MCGAMed (4) | 640 x 200 |
|  | MCGAHi (5) | 640 x 480 |
| EGA (3) | EGALO (0) | 640 x 200 |
|  | EGAHi (1) | 640 x 350 |
| EGA64 (4) | EGA64LO (0) | 640 x 200 |
|  | EGA64Hi (1) | 640 x 350 |
| EGAMONO (5) | EGAMONOHi (0) | 640 x 350 |
| VGA (9) | VGALO (0) | 640 x 200 |
|  | VGAMED (1) | 640 x 350 |
|  | VGAHI (2) | 640 x 480 |
| HERCMONO (7) | HERCMONOHI | 720 x 348 |
| ATT400 (8) | ATT400C0 (0) | 320 x 200 |
|  | ATT400C1 (1) | 320 x 200 |
|  | ATT400C2 (2) | 320 x 200 |
|  | ATT400C3 (3) | 320 x 200 |
|  | ATT400MED (4) | 640 x 400 |
|  | ATT400HI (5) | 640 x 400 |
| PC3270 (10) | PC3270HI (0) | 720 x 350 |
| IBM8514 (6) | IBM8514LO (0) | 640 x 480, 256 mầu |
|  | IBM8514HI (1) | 1024 x 768, 256 mầu |

***Chú ý 1.*** Bảng 8-2 cho các tên hằng và giá trị của chúng mà các biến graphdriver, graphmode có thể nhận. Chẳng hạn hằng DETECT có giá trị 0, hằng VGA có giá trị 9, hằng VGALO có giá trị 0... Khi lập trình ta có thể dùng tên hằng hoặc giá trị tương ứng của chúng. Chẳng hạn các phép gán trong ví dụ 1 có thể viết theo một cách khác tương đương như sau:

mh=3; mode=0;

***Chú ý 2.*** Bảng 8.2 cho thấy độ phân giải phụ thuộc cả vào màn hình và mode. Ví dụ trong màn hình EGA nếu dùng mode EGALO thì độ phân giải là 640 x 200, hàm getmaxx cho giá trị 639, hàm getmaxy cho giá trị 199 . Nếu cũng màn hình EGA mà dùng mode EGAHI thì độ phân giải là 640x 350, hàm getmaxx cho giá trị 639, hàm getmaxy cho giá trị 349.

***Chú ý 3.*** Nếu không biết chính xác kiểu màn hình đang sử dụng thì ta gán cho biến graphdriver hằng DETECT hay giá trị 0. Khi đó kết quả của hàm initgraph sẽ là:

* Kiểu của màn hình đang sử dụng được phát hiện, giá trị số của nó được gán cho biến graphdriver.
* Mode đồ họa ở độ phân giải cao nhất ứng với màn hình đang sử dụng cũng được phát hiện và giá trị số của nó được gán cho biến graphmode.

Như vậy việc dùng hằng số DETECT chẳng những có thể khởi động được hệ thống đồ họa của màn hình hiện có theo mode có độ phân giải cao nhất, mà còn giúp ta xác định chính xác kiểu màn hình đang sử dụng.

**Ví dụ 2.** Chương trình dưới đây xác định kiểu màn hình đang sử dụng:

*#include "graphics.h" #include "stdio.h" main()*

*{*

*int mh=0, mode= 0; initgraph(&mh, &mode, "");*

*printf("\n Giá trị số của màn hình là: %d", mh); closegraph();*

*}*

Nếu chương trình cho kết quả:

Giá trị số của màn hình là: 3

thì ta có thể khẳng định loại màn hình đang dùng là EGA.

***Chú ý 4.*** Nếu chuỗi dùng để xác định driverpath là một chuỗi rỗng (như trong ví dụ 2) thì chương trình dịch sẽ tìm các tệp điều khiển đồ họa trên thư mục chủ.

## Bài 3. Lỗi đồ họa

Khi khởi động hệ thống đồ họa nếu máy không tìm thấy các chương trình điều khiển đồ họa thì sẽ phát sinh lỗi đồ họa và việc khởi động coi như không thành. Lỗi đồ họa còn phát sinh khi dùng các hàm đồ hoạ. Trong mọi trường hợp, hàm graphresult cho biết có lỗi hay không lỗi và đó là lỗi gì. Bảng 8-3 cho các mã lỗi mà hàm này phát hiện được. Ta có thể dùng hàm grapherrormsg với mã lỗi do hàm graphresult trả về để biết được đó là lỗi gì, ví dụ:

int maloi;

maloi = graphresult();

printf("\nLỗi đồ họa là: %d", grapherrormsg(maloi));

***Bảng 8-3.*** Các mã lỗi của Graphresult

|  |  |  |
| --- | --- | --- |
| **Hằng** | **Trị** | **Lỗi phát hiện** |
| grOk | 0 | Không có lỗi |
| grNoInitGraph | -1 | Chưa khởi động hệ đồ họa |
| grNotDetected | -2 | Không có phần cứng đồ họa |
| grFileNotFound | -3 | Không tìm thấy trình điều khiển đồ họa |
| grInvalidDriver | -4 | Trình điều khiển không hợp lệ |
| grNoLoadMem | -5 | Không đủ RAM cho đồ họa |
| grNoScanMem | -6 | Vượt vùng RAM trong Scan fill |
| grNoFloodMem | -7 | Vượt vùng RAM trong flood fill |
| grFontNoFound | -8 | Không tìm thấy tập tin Font |
| grNoFontMem | -9 | Không đủ RAM để nạp Font |

|  |  |  |
| --- | --- | --- |
| grInvalidMode | -10 | Kiểu đồ họa không hợp lệ cho trình điều khiển |
| grError | -11 | Lỗi đồ họa tổng quát |
| grIOerror | -12 | Lỗi đồ họa vào ra |
| grInvalidFont | -13 | Tập tin Font không hợp lệ |
| grInvalidFontNum | -14 | Số hiệu Font không hợp lệ |

## Bài 4. Mầu và mẫu

1. **Để chọn mầu nền ta sử dụng hàm**

void setbkcolor(int color);

## Để chọn mầu đường vẽ ta dùng hàm

void setcolor(int color);

## Để chọn mẫu (kiểu) tô và mầu tô ta dùng hàm

void setfillstyle(int pattern, int color);

Trong cả 3 trường hợp color xác định mã của mầu. Các giá trị khả dĩ của color cho trong bảng 8- 4, pattern xác định mã của mẫu tô (xem bảng 8-5).

Mẫu tô và mầu tô sẽ được sử dụng trong các hàm pieslice, fillpoly, bar, bar3d và floodfill (xem

§5 dưới đây).

## Chọn giải mầu

Để thay đổi giải mầu đã được định nghĩa trong bảng 8.4 ta dùng hàm void setpalette(int colornum, int color);

**Ví dụ** câu lệnh setpalette(0, Lightcyan);

biến mầu đầu tiên trong bảng mầu thành xanh lơ nhạt. Các mầu khác không bị ảnh hưởng.

***Bảng 8-4.*** Các giá trị khả dĩ của color

|  |  |  |
| --- | --- | --- |
| **Tên hằng** | **Giá trị số** | **Mầu hiển thị** |
| BLACK | 0 | Đen |
| BLUE | 1 | Xanh da trời |
| GREEN | 2 | Xanh lá cây |
| CYAN | 3 | Xanh lơ |
| RED | 4 | Đỏ |
| MAGENTA | 5 | Tím |
| BROWN | 6 | Nâu |
| LIHGTGRAY | 7 | Xám nhạt |
| DARKGRAY | 8 | Xám sẫm |
| LIGHTBLUE | 9 | Xanh da trời nhạt |
| LIGHTGREEN | 10 | Xanh lá cây nhạt |
| LIGHTCYAN | 11 | Xanh lơ nhạt |
| LIGHTRED | 12 | Đỏ nhạt |
| LIGHTMAGENTA | 13 | Tím nhạt |
| YELLOW | 14 | Vàng |

|  |  |  |
| --- | --- | --- |
| WHITE | 15 | Trắng |

## Để nhận giải mầu hiện hành ta dùng hàm

void getpalette (struct palettetype \*palette);

ở đây palettetype là kiểu đã định nghĩa trước như sau:

*#define MAXCOLORS 15 struct palettetype*

*{*

*unsigned char size;*

*unsigned char colors[MAXCOLORS+1];*

*};*

ở đây: size là số lượng mầu trong palette, colors là mảng chứa mầu với chỉ số mảng chạy từ 0 đến size - 1

***Bảng 8-5.*** Các giá trị khả dĩ của pattern

|  |  |  |
| --- | --- | --- |
| **Tên hằng** | **Giá trị số** | **Mô tả kiểu tô** |
| EMPTY\_FILL | 0 | Tô bằng mầu nền |
| SOLID\_FILL | 1 | Tô bằng đường nét liền |
| LINE\_FILL | 2 | Tô bằng - - - |
| LTSLASH\_FILL | 3 | Tô bằng /// |
| SLASH\_FILL | 4 | Tô bằng /// in đậm |
| BKSLASH\_FILL | 5 | Tô bằng \\\ in đậm |
| LTBKSLASH\_FILL | 6 | Tô bằng \\\ |
| HATCH\_FILL | 7 | Tô bằng đường gạch bóng nhạt |
| XHATCH\_FILL | 8 | Tô bằng đường gạch bóng chữ thập |
| INTERLEAVE\_FILL | 9 | Tô bằng đường đứt quãng |
| WIDE\_DOT\_FILL | 10 | Tô bằng dấu chấm thưa |
| CLOSE\_DOT\_FILL | 11 | Tô bằng dấu chấm mau |

1. **Hàm getcolor** trả về mầu đã xác định trước đó bằng hàm setcolor.
2. **Hàm getbkcolor** trả về mầu đã xác định trước đó bằng hàm setbkcolor.
3. **Hàm getmaxcolor** trả về mã mầu cực đại thuộc giải mầu hiện đang có hiệu lực. Trên 256 K EGA, hàm getmaxcolor luôn cho giá trị 15.

## Bài 5. Vẽ và tô mầu

Có thể chia các đường và hình thành bốn nhóm chính:

* + Đường tròn và ellipse
  + Đường gấp khúc và hình đa giác
  + Đường thẳng
  + Hình chữ nhật

454

## Đường tròn và hình tròn

Nhóm này gồm cung tròn, đường tròn, cung ellipse và hình quạt.

* 1. **Cung tròn.** Để vẽ một cung tròn ta dùng hàm void arc(int x, int y, int gd, int gc, int r);

ở đây:

(x, y) là tọa độ của tâm cung tròn, r là bán kính

gd là góc đầu gc là góc cuối

***Chú ý:*** Trong tất cả các hàm dưới đây, góc tính theo độ và có giá trị từ 0 đến 360.

* 1. **Đường tròn.** Để vẽ một đường tròn ta dùng hàm void circle(int x, int y, int r);

ở đây:

(x, y) là tọa độ của tâm; r là bán kính đường tròn.

* 1. **Cung ellipse.** Để vẽ một cung Ellipse ta dùng hàm void ellipse(int x,int y,int gd,int gc,int xr,int yr);

ở đây:

(x, y) là tọa độ của tâm cung Ellipse gd là góc đầu

gc là góc cuối

xr là bán trục ngang yr là bán trục đứng.

* 1. **Hình quạt.** Để vẽ và tô màu một hình quạt ta dùng hàm void pieslice(int x,int y,int gd,int gc,int r);

ở đây:

(x,y) là tọa độ tâm hình quạt gd là góc đầu

gc là góc cuối r là bán kính

**Ví dụ 1.** Chương trình dưới đây sẽ vẽ: một cung tròn ở góc phần tư thứ nhất, một cung ellipse ở góc phần tư thứ ba, một đường tròn và một hình quạt quét từ 90 đến 360 độ.

*#include <graphics.h> main()*

*{*

*int mh, mode;*

*// Khởi động đồ họa, màn hình EGA, mode EGALO mh=EGA;*

*mode=EGALO; initgraph(&mh, &mode,"");*

*// Mầu nền Green, mầu đường vẽ*

*//White, mầu tô Red, kiểu tô SlashFill setbkcolor (GREEN);*

*setcolor (WHITE);*

*setfillstyle (SLASH\_FILL, RED);*

*// Vẽ: một cung tròn ở góc phần tư thứ nhất,*

*// một cung Ellipse ở góc phần tư thứ ba,*

*// một đường tròn, một quạt tròn arc(160, 50, 0, 90, 45);*

*ellipse(480, 50, 180, 270, 150, 45);*

*circle(160, 150, 45);*

*pieslice(480, 150, 90, 360, 45);*

*// Kết thúc chế độ đồ họa closegraph();*

*}*

## Đường gấp khúc và đa giác

1. **Muốn vẽ một đường gấp khúc đi qua n điểm**: (x1,y1), ... , (xn,yn) thì trước hết ta phải đưa các tọa độ vào một mảng a nào đó kiểu int. Nói một cách chính xác hơn, cần gán x1 cho a[0], y1 cho a[1], x2 cho a[2], y2 cho a[3],... Sau đó ta viết lời gọi hàm:

drawpoly(n, a);

Khi điểm cuối (xn, yn) trùng với điểm đầu (x1, y1) ta nhận được một đường gấp khúc khép kín.

## Giả sử a là mảng đã nói trong điểm 5, khi đó lời gọi hàm

fillpoly(n, a);

sẽ vẽ và tô mầu một đa giác có đỉnh là các điểm (x1, y1), ... ,(xn, yn).

**Ví dụ 2.** Chương trình dưới đây sẽ vẽ một đường gấp khúc và hai hình tam giác.

*#include <graphics.h>*

*// Xây dựng các mảng chứa tọa độ các đỉnh int poly1[]={5,200,190,5,100,300};*

*int poly2[]={205,200,390,5,300,300};*

*int poly3[]={405,200,590,5,500,300,405,200};*

*main()*

*{*

*int mh=0, mode=0; initgraph(&mh, &mode, "");*

*// Mầu nền CYAN, mầu đường vẽ*

*// YELLOW, mầu tô MAGENTA, mẫu tô SolidFill setbkcolor (CYAN); Setcolor (YELLOW); setfillstyle (SOLID\_FILL, MAGENTA);*

*drawpoly (3, poly1); // Đường gấp khúc fillpoly (3, poly2); // Hình đa giác fillpoly(4, poly3); // Hình đa giác closegraph();*

*}*

## Đường thẳng

1. **Hàm**

void line(int x1,int y1,int x2,int y2);

vẽ đường thẳng nối hai điểm (x1, y1) và (x2, y2) nhưng không làm thay đổi vị trí con chạy.

## Hàm

void lineto(int x,int y);

vẽ đường thẳng từ điểm hiện tại tới điểm (x, y) và chuyển con chạy đến điểm (x, y).

## Hàm

void linerel(int dx,int dy);

vẽ một đường thẳng từ vị trí hiện tại (x, y) của con chạy đến điểm (x + dx,y + dy). Con chạy được di chuyển đến vị trí mới.

## Hàm

void moveto(int x,int y);

sẽ di chuyển con chạy tới vị trí (x, y).

**Ví dụ 3.** Chương trình dưới đây tạo lên một đường gấp khúc bằng các đoạn thẳng. Đường gấp khúc đi qua các đỉnh: (20, 20), (620, 20), (620, 180), (20, 180) và (320, 100).

*#include <graphics.h> main()*

*{*

*int mh=0, mode=0; initgraph(&mh, &mode, ""); setbkcolor(GREEN); setcolor(YELLOW); moveto(320,100); line(20,20,620,20);*

*linerel(-300,80); lineto(620,180); lineto(620,20); closegraph();*

*}*

## Hình chữ nhật

1. **Hàm**

void rectangle(int x1,int y1,int x2,int y2);

sẽ vẽ một đường chữ nhật có các cạnh song song với các cạnh của màn hình. Tọa độ đỉnh trên bên trái của hình chữ nhật là (x1,y1) và điểm dưới bên phải là (x2,y2).

## Hàm

void bar(int x1,int y1,int x2,int y2);

sẽ vẽ và tô mầu một hình chữ nhật. Các giá trị x1,y1,x2 và y2 có ý nghĩa như đã nói trong điểm 11.

## Hàm

void bar3d(int x1,int y1,int x2,int y2,int depth,int top);

sẽ vẽ một khối hộp chữ nhật, mặt ngoài của nó là hình chữ nhật xác định bởi các tọa độ x1,y1,x2,y2 (như đã nói trong điểm 12). Hình chữ nhật này được tô mầu. Tham số depth ấn định số điểm ảnh trên bề sâu của khối 3 chiều. Tham số top có thể nhận trị 1 (TOPON) hay 0 (TOPOFF) và khối 3 chiều sẽ có nắp hay không nắp (xem hình vẽ).

TOPON TOPOFF

**Ví dụ 4.** Chương trình dưới đây sẽ vẽ một đường chữ nhật, một hình chữ nhật và một khối hộp chữ nhật có nắp.

*#include <graphics.h> main()*

*{*

*int mh=0, mode=0; initgraph(&mh, &mode, ""); setbkcolor(GREEN); setcolor(RED);*

*setfillstyle(CLOSE\_DOT\_FILL,YELLOW); rectangle(5,5,300,160); bar(5,175,300,340); bar3d(320,100,500,340,100,1);*

*closegraph();*

*}*

## Bài 6. Chọn kiểu đường

1. **Hàm**

void setlinestyle(int linestyle,int pattern,int thickness);

tác động đến nét vẽ của các thủ tục line, lineto, rectange, drawpoly, circle,... Hàm này cho phép ta ấn định 3 yếu tố của đường thẳng là dạng, bề dầy và mẫu tự tạo.

+ Dạng đường do tham số linestyle khống chế. Sau đây là các giá trị khả dĩ của linestyle và dạng đường thẳng tương ứng.

SOLID\_LINE = 0 Nét liền DOTTED\_LINE = 1 Nét chấm

CENTER\_LINE = 2 Nét chấm gạch DASHED\_LINE = 3 Nét gạch USERBIT\_LINE = 4 Mẫu tự tạo

+ Bề dầy do tham số thickness khống chế. Giá trị này có thể là: NORM\_WIDTH = 1 Bề dầy bình thường THICK\_WIDTH = 3 Bề dầy gấp ba

+ Mẫu tự tạo: Nếu tham số thứ nhất là USERBIT\_LINE thì ta có thể tạo ra mẫu đường thẳng bằng tham số pattern. Ví dụ xét đoạn chương trình:

int pattern= 0x1010;

setlinestyle(USERBIT\_LINE, pattern, NORM\_WIDTH); line(0,0,100,200);

Giá trị của pattern trong hệ 16 là 0x1010 hay trong hệ 2 là 0001 0000 0001 0000

Chỗ nào có bit 1 điểm ảnh sẽ sáng, bit 0 làm tắt điểm ảnh.

## Để nhận các giá trị hiện hành của 3 yếu tố trên ta dùng hàm:

void getlinesettings(struct linesettingstype \*lineinfo); với kiểu linesettingstype đã được định nghĩa trước như sau:

struct linesettingstype

{

int linestyle; unsigned int upattern; int thickness;

};

**Ví dụ 1.** Chương trình dưới đây minh họa cách dùng các hàm setlinestyle và getlinesettings để vẽ đường thẳng.

*// kiểu đường*

*#include <graphics.h> #include <conio.h> main()*

*{*

*struct linesettingstype kieu\_cu; int mh=0, mode=0; initgraph(&mh, &mode, "");*

*if (graphresult!= grOk) exit(1); setbkcolor(GREEN); setcolor(RED); line(0,0,100,0);*

*// Lưu lại kiểu hiện tại getlinesettings(kieu\_cu);*

*// Thiết lập kiểu mới setlinestyle(DOTTED\_LINE,0,THICK\_WIDTH); line(0,0,100,10);*

*// Phục hồi kiểu cũ setlinestyle(kieu\_cu.linestyle, kieu\_cu.upattern, kieu\_cu.thickness); Line(0,20,100,20);*

*getch(); closegraph();*

*}*

## Hàm

void setwritemode( int writemode);

sẽ thiết lập kiểu thể hiện đường thẳng cho các hàm line, drawpoly, linerel, lineto, rectangle. Kiểu thể hiện do tham số writemode khống chế:

* + Nếu writemode bằng COPY\_PUT = 0, thì đường thẳng được viết đè lên dòng đang có trên màn hình.
  + Nếu writemode bằng XOR\_PUT = 1, thì mầu của đường thẳng định vẽ sẽ kết hợp với mầu của từng chấm điểm của đường hiện có trên màn hình theo phép toán XOR (chương 3, §3) để tạo lên một đường thẳng mới.

Một ứng dụng của XOR\_PUT là: Khi thiết lập kiểu writemode bằng XOR\_PUT rồi vẽ lại đường thẳng cùng mầu thì sẽ xóa đường thẳng cũ và khôi phục trạng thái của màn hình.

Chương trình dưới đây minh họa cách dùng hàm setwritemode. Khi thực hiện ta sẽ thấy hình chữ nhật thu nhỏ dần vào tâm màn hình.

## Ví dụ 2:

*// Thu hình;*

*#include <graphics.h> #include <conio.h> main()*

*{*

*struct linesettingstype kieu\_cu; int mh=0, mode=0, x1, y1, x2, y2; initgraph(&mh, &mode, "");*

*if (graphresult!= grOk) exit(1); setbkcolor(GREEN); setcolor(RED);*

*setfillstyle(CLOSE\_DOT\_FILL, YELLOW); x1=0; y1=0;*

*x2=getmaxx(); y2=getmaxy(); setwritemode(XOR\_PUT);*

*tt: rectangle(x1,y1,x2,y2); // Vẽ hình chữ nhật if ( (x1+1)<(x2-1) && (y1+1)<(y2-1) )*

*{*

*rectangle(x1,y1,x2,y2); // xóa hình chữ nhật x1=x1+1; y1=y1+1; co hình chữ nhật x2=x2-1; y2=y2-1;*

*goto tt;*

*}*

*setwritemode(COPY\_PUT); // Trở về overwrite mode closegraph();*

*}*

## Bài 7. Cửa sổ (Viewport)

1. **Viewport** là một vùng chữ nhật trên màn hình đồ họa tựa như window trong textmode. Để thiết lập viewport ta dùng hàm

void setviewport(int x1,int y1,int x2,int y2,int clip);

trong đó (x1,y1) là tọa độ góc trên bên trái và (x2,y2) là tọa độ góc dưới bên phải. Bốn giá trị này phải thỏa mãn:

0 <= x1 <= x2

0 <= y1 <= y2

Tham số clip có thể nhận một trong hai giá trị: clip = 1 không cho phép vẽ ra ngoài viewport clip = 0 Cho phép vẽ ra ngoài viewport.

**Ví dụ** câu lệnh setviewport(100,50,200,150, 1);

sẽ thiết lập viewport. Sau khi lập viewport ta có hệ tọa độ mới mà góc trên bên trái của viewport sẽ có tọa độ (0,0).

## Để nhận viewport hiện hành ta dùng hàm

void getviewsettings(struct viewporttype \*vp); ở đây kiểu viewporttype đã được định nghĩa như sau:

struct viewporttype

{

int left, top, right, bottom; int clip;

};

## Để xóa viewport ta dùng hàm

void clearviewport(void);

## Để xóa màn hình và đưa con chạy về tọa độ (0,0) của màn hình ta dùng hàm

void cleardevice(void);

***Chú ý:*** Câu lệnh này sẽ xóa mọi thứ trên màn hình.

464

**5. Tọa độ âm dương**

Nhờ sử dụng Viewport có thể viết các chương trình đồ họa theo tọa độ âm dương. Muốn vậy ta thiết lập viewport sao cho tâm tuyệt đối của màn hình là góc trên bên trái của viewport và cho clip = 0 để có thể vẽ ra ngoài giới hạn của viewport. Sau đây là đoạn chương trình thực hiện công việc trên

int xc, yc;

xc= getmaxx()/2; yc= getmaxy()/2;

setviewport(xc, yc, getmaxx(), getmaxy(), 0);

Như thế màn hình sẽ được chia làm 4 phần với tọa độ âm dương như sau: Phần tư trái trên: x âm, y âm

Phần tư trái dưới: x âm, y dương Phần tư phải trên: x dương, y âm Phần tư phải dưới: x dương, y dương

Chương trình dưới đây vẽ đồ thị hàm sin(x) trong hệ trục tọa độ âm dương. Hoành độ x lấy các giá trị từ -4\*PI đến 4\*PI. Trong chương trình có dùng hai hàm mới là: outtextxy và putpixel (xem các mục sau).

## Ví dụ 1:

*// đồ thị hàm sin #include <graphics.h> #include <conio.h> #include <math.h> #define SCALEX 20*

*#define SCALEY 60 main()*

*{*

*int mh=0, mode=0, x, y, i; initgraph(&mh, &mode, "");*

*if (graphresult!= grOk) exit(1); setviewport(getmaxx()/,getmaxy()/2, getmaxx(),getmaxy(), 0);*

*// Kẻ hệ trục tọa độ setcolor(BLUE);*

*line(-(getmaxx()/2),0,getmaxx()/2,0);*

*line(0,-(getmaxy()/2),0,getmaxy()/2); settextjustify(1,1); setcolor(RED); outtextxy(0,0,"(0,0)");*

*for (i=-400;i<=400;++i)*

*{*

*x=round(2\*M\_PI\*i\*SCALEX/200); y=round(sin(2\*M\_PI\*i/200)\*SCALEY); putpixel(x,y,YELLOW);*

*}*

*getch();*

*}*

**Ví dụ 1** tạo lên một đồ thị từ các chấm điểm. Bây giờ ta sửa ví dụ 1 đôi chút: giữ nguyên từ đầu đến outtextxy, thay phần cuối bởi đoạn chương trình dưới đây. Ta sẽ được đồ thị từ các đoạn thẳng rất ngắn ghép lại.

## Ví dụ 2:

*// Phần đầu giống ví dụ 1 setcolor(YELLOW);*

*for (i=-400;i<=400;++i)*

*{*

*x=round(2\*M\_PI\*i\*SCALEX/200); y=round(sin(2\*M\_PI\*i/200)\*SCALEY);*

*if(i= -400) moveto(x,y); else lineto(x,y);*

*}*

*getch();*

*}*

## Hàm

**Bài 8. Tô điểm, tô miền**

void putpixel(int x, int y, int color);

sẽ tô điểm (x,y) theo mầu xác định bởi color.

## Hàm

unsigned getpixel(int x, int y);

sẽ trả về số hiệu mầu của điểm ảnh ở vị trí (x,y). Chú ý: nếu điểm này chưa được tô mầu bởi các hàm vẽ hoặc putpixel (mà chỉ mới được tạo mầu nền bởi setbkcolor) thì hàm cho giá trị bằng 0. Vì vậy có thể dùng hàm này theo mẫu dưới đây để xác định các nét vẽ trên màn hình đồ hoạ và vẽ ra giấy.

if (getpixel(x,y)!=0)

{

// Điểm (x,y) được vẽ , đặt một chấm điểm ra giấy

}

## Tô miền

Để tô mầu cho một miền nào đó trên màn hình ta dùng hàm void floodfill(int x, int y, int border);

ở đây:

(x,y) là tọa độ của một điểm nào đó gọi là điểm gieo. tham số border chứa mã của một mầu.

Sự hoạt động của hàm floodfill phụ thuộc vào giá trị của x,y, border và trạng thái màn hình.

* 1. Khi trên màn hình có một đường (cong hoặc gấp khúc) khép kín mà mã mầu của nó bằng giá trị của border thì:

+ Miền giới hạn bởi đường kín sẽ được tô mầu nếu điểm gieo (x,y) nằm bên trong miền này.

+ Nếu điểm gieo (x,y) nằm bên ngoài thì phần màn hình bên ngoài miền đóng nói trên được tô màu.

* 1. Khi trên màn hình không có một đường nào như vậy, thì cả màn hình được tô màu.

**Ví dụ 1.** Chương trình dưới đây sẽ vẽ một đường tròn đỏ trên màn hình xanh. Tọa độ (x,y) của điểm gieo được nạp vào từ bàn phím. Tùy thuộc vào giá trị cụ thể của x,y, chương trình sẽ tô mầu vàng cho hình tròn hoặc phần màn hình bên ngoài hình tròn.

*#include <graphics.h> #include <stdio.h> main()*

*{*

*int mh=0, mode=0, x, y; initgraph(&mh, &mode, "");*

*if (graphresult!= grOk) exit(1); setbkcolor(GREEN); setcolor(RED); setfillstyle(11,YELLOW); circle(320,100,50); moveto(1,150);*

*outtext(" Toa do diem gieo x,y "); scanf("%d%d",&x,&y); flooddfill(x,y,RED);*

*}*

**Ví dụ 2.** Minh họa cách dùng hàm Putpixel và hàm getpixel để vẽ các điểm ảnh và sau đó xóa các điểm ảnh. Muốn kết thúc chương trình bấm ESC.

*#include <conio.h> #include <graphics.h> #include <stdio.h> #include <stdlib.h>*

*int seed = 1962; // Nhân cho bộ tạo số ngẫu nhiên int numpts = 2000; // Vẽ 2000 chấm điểm*

*int ESC = 27;*

*void putpixelplay(void); main()*

*{*

*int mh=0, mode=0; initgraph(&mh, &mode, ""); if (graphresult()!= grOk)*

*exit(1); putpixelplay(); closegraph();*

*}*

*void putpixelplay(void)*

*{*

*int i,x,y,color,xmax,ymax,maxcolor,ch;*

*struct viewporttype v; getviewsettings(&v);*

*xmax=(v.right - v.left -1); ymax=(v.bottom - v.top -1); maxcolor=getmaxcolor();*

*while (!kbhit())*

*{*

*//Vẽ các chấm điểm một cách ngẫu nhiên srand(seed);*

*i=0;*

*while(i<=numpts)*

*{*

*++i;*

*x=random(xmax)+1;y=random(ymax)+1; color=random(maxcolor); putpixel(x,y,color);*

*}*

*// Xóa các điểm ảnh srand(seed);*

*i=0;*

*while(i<=numpts)*

*{*

*++i;*

*x= random(xmax) + 1; y= random(ymax) + 1; color=random(maxcolor);*

*putpixel(x,y,0);*

*}*

*if(kbhit())*

*{*

*ch=getch();*

*if (ch==ESC) break;*

*}*

*}*

*} // Kết thúc hàm putpixelplay*

## Bài 9. Xử lý văn bản trên màn hình đồ hoạ

1. **Hiển thị văn bản trên màn hình đồ hoạ**

Hàm

void outtext (char \*s);

sẽ hiện chuỗi ký tự (do s trỏ tới) tại vị trí hiện tại của con trỏ.

Hàm

void outtextxy(int x,int y,char \*s);

sẽ hiện chuỗi ký tự (do s trỏ tới) tại ví trí (x,y).

**Ví dụ 1:** Hai cách sau đây sẽ cho cùng kết quả outtextxy (100,100," chao ban ");

và

moveto (100,100); outtext (" chao ban ");

***Chú ý:*** Trong mốt đồ họa vẫn cho phép dùng hàm nhập dữ liệu scanf và các hàm bắt phím getch,

kbhit.

## Fonts

Như đã nói ở trên: Các Fonts nằm trong các tệp tin .CHR trên đĩa. Các Font này cho các kích thước và kiểu chữ khác nhau sẽ hiện thị trên màn hình đồ hoạ bằng outtext hay outtextxy. Để chọn và nạp Font chúng ta dùng hàm:

void settextstyle(int font,int direction,int charsize);

***(Chú ý:*** hàm chỉ có tác dụng nếu tồn tại các tệp .CHR) Với direction là một trong hai hằng số:

HORIZ\_DIR = 0

VERT\_DIR = 1

Nếu direction là HORIZ\_DIR, văn bản sẽ hiển thị theo hướng nằm ngang từ trái sang phải. Nếu direction là VERT\_DIR, văn bản sẽ hiển thị theo chiều đứng từ dưới lên trên.

Đối charsize là hệ số phóng to ký tự và có giá trị trong khoảng từ 1 đến 10.

* Nếu charsize = 1, ký tự được thể hiện trong hình chữ nhật 8\*8 pixel.
* Nếu charsize = 2, ký tự được thể hiện trong hình chữ nhật 16\*16 pixel.

. . .

* Nếu charsize = 10, ký tự được thể hiện trong hình chữ nhật 80\*80 pixel. Cuối cùng là tham số font để chọn kiểu chữ và nhận một trong các hằng sau:

DEFAULT\_FONT = 0

TRIPLEX\_FONT = 1

SMALL\_FONT = 2

SANS\_SERIF\_FONT = 3

GOTHIC\_FONT = 4

Các giá trị do settextstyle thiết lập sẽ dữ nguyên cho đến khi gọi một settextstyle mới.

## Ví dụ 2:

settextstyle (3,VERT\_DIR,2); outtextxy (50,50," HELLO ");

## Vị trí hiển thị

Hàm settextjustify cho phép ấn định nơi hiển thị văn bản của outtext theo quan hệ với vị trí hiện tại của con chạy hay của outtextxy theo quan hệ với toạ độ (x,y).

Hàm này có dạng

void settextjustify(int horiz, int vert);

Tham số horiz có thể là một trong các hằng số sau: LEFT\_TEXT = 0 (Văn bản xuất hiện bên phải con chạy)

CENTER\_TEXT = 1 (Chỉnh tâm văn bản theo vị trí con chạy) RIGHT\_TEXT = 2 (Văn bản xuất hiện bên trái con chạy)

Tham số Vert có thể là một trong các hằng số sau: BOTTOM\_TEXT = 0 (Văn bản xuất hiện phía trên con chạy) CENTER\_TEXT = 1 (Chỉnh tâm văn bản theo vị trí con chạy) TOP\_TEXT = 2 (Văn bản xuất hiện phía dưới con chạy)

## Ví dụ 3:

settextjustify(1,1); outtextxy(100,100,"ABC");

Kết quả là điểm (100,100) sẽ nằm giữa chữ B.

## Bề rộng và bề cao của văn bản

Hàm

void textheight (char \*s);

trả về chiều cao (theo pixel) của chuỗi do s trỏ tới. Ví dụ nếu ký tự có kích thước 8\*8 thì textheight ("H") = 8

**Ví dụ 4:** Đoạn chương trình dưới đây sẽ cho hiện 5 dòng chữ.

*#include <graphics.h> main()*

*{*

*int mh=0,mode=0,y,size; initgraph(&mh,&mode,""); y=10;*

*settextjustify(0,0);*

*for (size=1; size<=5; ++size)*

*{*

*settextstyle(0,0,size); outtextxy(0,y,"GRAPHICS");*

*y += textheight("GRAPHICS") + 10;*

*}*

*getch(); closegraph();*

*}*

Hàm

void textwidth(char \*s);

sẽ dựa vào chiều dài của chuỗi, kích thước Font chữ, hệ số khuyếch đại chữ để trả về bề rộng (theo pixel) của chuỗi do s trỏ tới.

**Ví dụ 5:** Trong chương trình dưới đây sẽ lập các hàm vào ra trên màn hình đồ hoạ.

*#include <graphics.h> #include <conio.h> #define Enter 13*

*#define Lmargin 10*

*void text\_write(int \*x,int \*y,char \*s); void text\_writeln(int \*x,int \*y,char \*s); void text\_read(int \*x,int \*y,char \*s); void text\_write(int \*x,int \*y,char \*s)*

*{*

*outtextxy(\*x,\*y,s); \*x += textwidth(s);*

*}*

*void text\_writeln(int \*x,int \*y,char \*s)*

*{*

*outtextxy(\*x,\*y,s);*

*\*x=Lmargin;*

*\*y += textheight(s)+5;*

*}*

*void text\_read(int \*x,int \*y,char \*s)*

*{*

*int i=0; char ch[2]; ch[1]=0;*

*while(1)*

*{*

*ch[0]=getch(); if(ch[0]==Enter) break; text\_write(x,y,ch); s[i]=ch[0]; ++i;*

*}*

*s[i]=0;*

*}*

*main()*

*{*

*int mh=0,mode=0,x,y,xmax,ymax; char name[25]; initgraph(&mh,&mode,"");*

*settextstyle(TRIPLEX\_FONT,HORIZ\_DIR,3); x=Lmargin; y=100;*

*text\_write (&x,&y,"cho ten cua ban: ");*

*text\_read (&x,&y,name); text\_writeln (&x,&y,"" ); text\_write(&x,&y,"chao ban "); text\_write(&x,&y,name); getch();*

*closegraph();*

*}*

## Bài 10. Cắt hình, Dán hình và Tạo ảnh chuyển động

1. **Hàm**

unsigned imagesize(int x1,int y1,int x2,int y2)

trả về số byte cần thiết để lưu trữ ảnh trong phạm vi hình chữ nhật (x1,y1,x2,y2).

## Hàm

#include <alloc.h>

void \*malloc(unsigned n);

trả về con trỏ trỏ tới một vùng nhớ n byte mới được cấp phát.

## Hàm

void getimage(int x1,int y1,int x2,int y2,void \*bitmap);

sẽ chép các điểm ảnh của hình chữ nhật (x1,y1,x2,y2) và các thông tin về bề rộng, cao của hình chữ nhật vào vùng nhớ do bitmap trỏ tới. Vùng nhớ và biến bitmap cho bởi hàm malloc. Độ lớn của vùng nhớ được xác định bằng hàm imagesize.

## Hàm

void putimage(int x,int y,void \*bitmap,int copymode);

dùng để sao ảnh lưu trong vùng nhớ bitmap ra màn hình tại vị trí (x,y). Tham số copymode xác định kiểu sao chép ảnh, nó có thể nhận các giá trị sau:

COPY\_PUT = 0 Sao chép nguyên xi.

XOR\_PUT = 1 Các điểm ảnh trong bitmap kết hợp với các

điểm ảnh trên màn hình bằng phép XOR

OR\_PUT = 2 Các điểm ảnh trong bitmap kết hợp với các

điểm ảnh trên màn hình bằng phép OR

AND\_PUT = 3 Các điểm ảnh trong bitmap kết hợp với các

điểm ảnh trên màn hình bằng phép AND

NOT\_PUT = 4 ảnh xuất hiện trên màn hình theo dạng đảo

ngược (phép NOT) với ảnh trong bitmap.

*Nhận xét:* Nếu dùng mode XOR\_PUT để chép hình, rồi lặp lại đúng câu lệnh đó thì hình sẽ bị xoá và màn hình trở lại như cũ. Kỹ thuật này dùng để tạo lên các hình ảnh chuyển động.

**Ví dụ 1:** Chương trình dưới đây minh hoạ cách dùng imagesize, malloc, getimage và putimage.

*#include <alloc.h> #include <graphics.h> main()*

*{*

*int mh=0,mode=0; char \*p;*

*unsigend size;*

*initgraph (&mh,&mode,""); bar(0,0,getmaxx(),getmaxy()); size = imagesize(10,20,30,40);*

*p=(char\*)malloc(size); // p trỏ tới vùng nhớ size byte*

*// mới được cấp phát getimage (10,20,30,40,p);*

*getch(); cleardevice();*

*putimage (100,100,p,COPY\_PUT); getch();*

*closegraph();*

*}*

## Tảo ảnh di động

Nguyên tắc tạo ảnh di động giống như phim hoạt hình:

* + Vẽ một hình (trong chuỗi hình mô tả chuyển động)
  + Delay
  + Xoá hình đó
  + Vẽ hình kế theo
  + Delay

. . .

## Vẽ hình

*Cách 1:* Vẽ lại một ảnh nhưng tại các vị trí khác nhau.

*Cách 2:* Lưu ảnh vào một vùng nhớ rối đưa ảnh ra màn hình tại các vị trí khác nhau.

## Xóa ảnh

*Cách 1:* Dùng hàm cleardevice

*Cách 2:* Dùng hàm putimage (mode XOR\_PUT) để xếp chồng lên ảnh cần xoá.

*Cách 3:* Lưu trạng thái màn hình vào một chỗ nào đó. Vẽ một hình ảnh. Đưa trạng thái cũ màn hình ra xếp đè lên ảnh vừa vẽ.

Kỹ thuật tạo ảnh di động được minh hoạ trong các chương trình của §11.

## Bài 11. Một số chương trình đồ hoạ

**Chương trình 1:** Đầu tiên vẽ bầu trời đầu sao. Sau đó từng chùm pháo hoa được bắn lên bầu trời. Khi bấm phím Enter thì việc bắn pháo hoa kết thúc, ta nhận lại bầu trời đầy sao. Bấm tiếp Enter thì kết thúc chương trình.

*// Bắn pháo hoa trên bầu trời đầy sao*

*#include <graphics.h> #include <conio.h> #include <stdlib.h> #include <alloc.h> main()*

*{*

*int x[101],y[101];*

*int mh=0,mode=0,i,n; char \*p[101];*

*initgraph(&mh,&mode,""); if (graphresult()!=0) exit(1); setcolor(RED);*

*// Vẽ bầu trời đầy sao for (i=1;i<=1000;++i)*

*{*

*putpixel(random(getmaxx()), random(getmaxy()),random(getmaxcolor()));*

*}*

*// Lưu hiện trạng 100 hình chữ nhật trên màn hình để khôi phục for (i=1;i<=100;++i)*

*{*

*x[i]=random(getmaxx())-10;*

*y[i]=random(getmaxy())-10;*

*if (x[i]<0) x[i]=0;*

*if (y[i]<0) y[i]=0; n=imagesize(x[i],y[i],x[i]+10,y[i]+10); p[i]=(char\*)malloc(n); getimage(x[i],y[i],x[i]+10,y[i]+10,p[i]);*

*}*

*// Chu trình bắn pháo hoa do*

*{*

*// Đưa 100 quả pháo lên màn hình tại các vị trí quy định for (i=1;i<=100;++i)*

*{*

*setfillstyle(SOLID\_FILL,i%15+1); pieslice(x[i]+5,y[i]+5,0,360,5);*

*}*

*delay(500);*

*//Xoá chùm pháo hoa vừa bắn bằng cách khôi phục màn hình*

*for (i=100;i>=1;--i) putimage(x[i],y[i],p[i],COPY\_PUT);*

*delay(500);*

*} while(!kbhit()); getch();*

*getch(); closegraph();*

*}*

**Chương trình 2:** Vẽ đồng hồ có 3 kim giờ, phút và giây. Đồng hồ chạy đúng theo giờ hệ thống.

Muốn kết thúc chương trình bấm Enter.

*// Đồng hồ*

*#include <graphics.h> #include <conio.h> #include <math.h> #include <dos.h>*

*// Hàm kẻ đoạn thẳng từ tâm đồng hồ theo độ, chiều dài,*

*// độ dầy và mầu*

*void ke(int ddo, unsigned dai, unsigned day,unsigned mau);*

*// Kẻ kim giây khi biết số giây void ke\_giay(unsigned giay);*

*// Kẻ kim phút khi biết số phút void ke\_phut(unsigned phut);*

*// Kẻ kim giờ khi biết số giờ*

*void ke\_gio(unsigned gio, unsigned phut);*

*void chay\_kim\_giay(void); void chay\_kim\_phut(void); void chay\_kim\_gio(void);*

*int x0,y0,rgio,rphut,rgiay,mgio,mphut,mgiay; unsigned phutgioht,gioht,phutht,giayht;*

*void ke(int ddo, unsigned dai, unsigned day,unsigned mau)*

*{*

*unsigned x,y; float goc;*

*while (ddo>=360) ddo=ddo-360; goc=(M\_PI/180)\*ddo;*

*x=x0+ (int)(dai\*cos(goc)+0.5); y=y0- (int)(dai\*sin(goc)+0.5); setcolor(mau); setlinestyle(0,0,day); line(x0,y0,x,y);*

*}*

*// Hàm ke kim giay*

*void ke\_giay(unsigned giay)*

*{*

*int ddo;*

*ddo = (90 - 6\*giay); ke(ddo,rgiay,1,mgiay);*

*}*

*// Hàm ke kim phut*

*void ke\_phut(unsigned phut)*

*{*

*int ddo;*

*ddo= (90-6\*phut); ke(ddo,rphut,3,mphut);*

*}*

*// Hàm ke kim gio*

*void ke\_gio(unsigned gio, unsigned phut)*

*{*

*int ddo;*

*ddo = 360 + 90 - 30\*(gio%12) - (phut+1)/2; ke(ddo,rgio,3,mgio);*

*}*

*// Hàm chỉnh giây hiện tại và làm chuyển động kim giây void chay\_kim\_giay(void)*

*{*

*unsigned giay; struct time t; gettime(&t);*

*giay=t.ti\_sec;*

*if (giay!=giayht)*

*{*

*ke\_giay(giayht); giayht=giay; ke\_giay(giayht);*

*}*

*}*

*// Hàm chỉnh phút hiện tại và làm chuyển động kim phút void chay\_kim\_phut(void)*

*{*

*unsigned phut; struct time t; gettime(&t); phut=t.ti\_min;*

*if (phut!=phutht)*

*{*

*ke\_phut(phutht); phutht=phut; ke\_phut(phutht);*

*}*

*}*

*// Hàm chỉnh giờ phút hiện tại và làm chuyển động kim giờ void chay\_kim\_gio(void)*

*{*

*unsigned h,gio,phut,sophut,sophutht; struct time t;*

*gettime(&t);*

*gio=t.ti\_hour; phut=t.ti\_min; sophut = gio\*60+phut; sophutht = gioht\*60+phutgioht;*

*if ( sophut<sophutht) sophut=sophut+ 12\*60; h=sophut-sophutht;*

*if (h>=12)*

*{*

*ke\_gio(gioht,phutgioht); phutgioht=phut; gioht=gio; ke\_gio(gioht,phutgioht);*

*}*

*}*

*main()*

*{*

*struct time t;*

*char \*dso[]={"", "12", "1", "2", "3", "4", "5", "6", "7", "8", "9", "10", "11"};*

*int i,mh=0,mode=0,r,x,y; float goc; initgraph(&mh,&mode,"");*

*x0=(getmaxx()/2)-1; y0=(getmaxy()/2)-1; r=y0-2;*

*rgiay = r-10; rphut=r-50; rgio=r-90;*

*mgiay= BROWN; mphut=RED; // mgio:=magenta; mgio=YELLOW;*

*// Vẽ chu vi đồng hồ*

*setcolor(BLUE); setlinestyle(0,0,3); circle(x0,y0,r); setfillstyle(1,YELLOW);*

*floodfill(0,0,BLUE);*

*setfillstyle(1,WHITE); floodfill(x0,y0,BLUE); setlinestyle(0,0,1);*

*circle(x0,y0,10);*

*setfillstyle(1,GREEN); floodfill(x0,y0,BLUE); settextjustify(1,1); setcolor(MAGENTA); outtextxy(x0,y0+120,"IBM-JIMIKO");*

*// Ghi chữ số*

*settextstyle(3,0,3); settextjustify(1,1); setcolor(BLUE); for (i=1;i<=12;++i)*

*{*

*goc=(2\*M\_PI+M\_PI/2) - (i-1)\*(M\_PI/6); x = x0+ (int)(rphut\*cos(goc)+0.5);*

*y = y0- (int)(rphut\*sin(goc)+0.5); outtextxy(x,y,dso[i]);*

*}*

*// Xác định thời điểm đầu gettime(&t);*

*gioht=t.ti\_hour; phutht=t.ti\_min; giayht=t.ti\_sec; phutgioht=phutht;*

*setwritemode(XOR\_PUT);*

*// Ve kim gio,phut,giay ke\_gio(gioht,phutgioht); ke\_phut(phutht); ke\_giay(giayht);*

*// Làm chuyển động các kim do*

*{*

*chay\_kim\_giay(); chay\_kim\_phut(); chay\_kim\_gio();*

*}*

*while(!kbhit()); closegraph();*

*}*

**Chương trình 3:** Vẽ một con tầu vũ trụ bay trên bầu trời đầy sao theo quỹ đạo ellipse. Trong khi tầu chuyển động thì các ngôi sao thay nhau nhấp nháy

// Tầu vũ trụ chuyển động trên bầu trời đầy sao nhấp nháy

*#include <graphics.h> #include <stdio.h> #include <conio.h> #include <alloc.h>*

*#include <stdlib.h> #include <math.h>*

*// Khai báo các hàm trong chương trình void tau\_cd(void); // tầu chuyển động*

*void nhap\_nhay\_bt(void); // nhấp nháy bầu trời void main(void); // hàm main*

*// Khai báo các biến mảng ngoài int a,b,x,y,x0,y0,mh=0,mode=0,n,i; float goc,xt,yt;*

*char \*p;*

*int xx[1001],yy[1001];*

*// Hàm main void main(void)*

*{*

*initgraph(&mh,&mode,""); if (graphresult()!=0) exit(1);*

*// Vẽ tầu vũ trụ setcolor(RED); ellipse(100,50,0,360,20,8); ellipse (100,46,190,357,20,6); line(107,44,110,38);*

*circle(110,38,2); line(93,44,90,38);*

*circle(90,38,2); setfillstyle(SOLID\_FILL,BLUE); floodfill(101,54,RED); setfillstyle(SOLID\_FILL,MAGENTA); floodfill(94,45,RED);*

*// Lưu ảnh của tầu vũ trụ vào bộ nhớ n=imagesize(79,36,121,59); p=(char\*)malloc(n); getimage(79,36,121,59,p);*

*// Vẽ bầu trời đầy sao và lưu vị trí của chúng*

*// vào các mảng xx, yy để phục vụ hàm nhap\_nhay\_bt cleardevice();*

*for (i=1;i<=1000;++i)*

*{*

*xx[i]=random(getmaxx()); yy[i]=random(getmaxy()); putpixel(xx[i],yy[i],random(getmaxcolor()));*

*}*

*// Xác định giá trị ban đầu cho các biến*

*// dùng để điều khiển chuyển động tầu goc= 2\*M\_PI + M\_PI/2;*

*x0= (getmaxx() - 42)/2; y0= (getmaxy() - 25)/2; a=x0; b=y0;*

*// chu trình tầu vũ trụ chuyển động và các ngôi sao nhấp nháy do*

*{*

*tau\_cd(); nhap\_nhay\_bt();*

*} while(!kbhit()); getch(); closegraph();*

*}*

*void tau\_cd(void)*

*{*

*xt=a\*cos(goc)+x0; yt=-b\*sin(goc)+y0;*

*x=(int)(xt+0.5); y=(int)(yt+0.5);*

*// Đặt tầu vũ trụ lên màn hình putimage(x,y,p,XOR\_PUT); delay(500);*

*// Xóa putimage(x,y,p,XOR\_PUT);*

*// Thay đổi góc để làm cho tầu chuyển động goc -= M\_PI/30;*

*if (goc<M\_PI/2) goc=2\*M\_PI+M\_PI/2;*

*}*

*void nhap\_nhay\_bt(void)*

*{*

*static i=1; // Lệnh này thực hiện một lần khi dịch int j;*

*// Cho nhấp nháy bằng cách đổi mầu 50 ngôi sao for (j=1;j<=50;++j)*

*{*

*putpixel(xx[i],yy[i],random(getmaxcolor()));*

*++i;*

*if (i>1000) i=1;*

*}*

*}*

## Bài 12. In ảnh từ màn hình đồ hoạ

Hàm in\_anh dưới đây sẽ in ảnh trong miền chữ nhật (xt, yt, xd, yd) của màn hình đồ hoạ ra giấy trên các máy in LQ1070, LQ1170 và FX1050.

void in\_anh(int dd,int xt,int yt,int xd,int yd);

Tham số dd là độ đậm của nét in. Thực chất dd là số lần in lại. Bình thường chon dd=1. Nếu muốn in rõ hơn ta chọn dd bằng 2 hay 3.

Trong hàm in\_anh có dùng hàm tao\_mau, nó được mô tả như sau: int tao\_mau(int k,int x,int y);

Hàm này sẽ dò trên k chấm điểm theo chiều dọc bắt đầu từ toạ độ (x,y) trên màn hình để biết xem chấm điểm nào đã tô mầu. Hàm sẽ trả về một giá trị nguyên tạo bởi các bit 1 (ứng với điểm đã tô mầu) và 0 (ứng với điểm chưa tô mầu).

Hàm in\_anh sẽ dùng hàm tao\_mau để duyệt trên miền chữ nhật (xt,yt,xd,yd). Mỗi lần duyệt sẽ nhận được một mẫu các chấm điểm (giá trị nguyên) và mẫu này được in ra giấy.

Dưới đây là nội dung của 2 hàm nói trên.

*// in ảnh*

*#include "stdio.h" #include "graphics.h"*

*int tao\_mau(int k,int x,int y);*

*void in\_anh(int dd,int xt,int yt,int xd,int yd); int tao\_mau(int k,int x,int y)*

*{*

*int c=0,i;*

*for (i=0;i<k;++i)*

*if (getpixel(x,y+i)) c =c|(128>>i); return c;*

*}*

*void in\_anh(int dd,int xt,int yt,int xd,int yd)*

*{*

*//dd - so lan in lai mot dong char c,ch1;*

*int scot,m,mm,k,dong,cot,i,j,n1,n2; dong=(yd-yt+1)/6; mm=(yd-yt+1) % 6; cot=xd-xt+1;*

*for (i=0;i<=dong;++i)*

*{*

*if (i<dong) m=6; else m=mm; if (m>0)*

*{*

*scot=0;*

*for (j=0;j < cot;++j)*

*if (tao\_mau(m,xt+j,yt+i\*6)) scot=j+1; if (scot)*

*{*

*n1=scot % 256; n2= scot/256; for (k=0;k<dd;++k)*

*{*

*fprintf(stdprn,"%c%c%c%c%c%c",13,27,'\*',*

*0,n1,n2); //LQ*

*for (j=0;j < scot;++j)*

*{*

*if (kbhit())//bat phim*

*{*

*if ((ch1=getch())==0) getch(); if (ch1==27) goto ket;*

*}*

*c=tao\_mau(m,xt+j,yt+i\*6); fprintf(stdprn,"%c",c);*

*}*

*}*

*}*

*fprintf(stdprn,"%c%c%c",27,'A',m); fprintf(stdprn,"\n");*

*}*

*}*

*ket: fprintf(stdprn,"%c%c",27,'@');*

*}*

**Chương 9**

# Truy nhập trực tiếp vào bộ nhớ

Trong chương này trình bầy các vấn đề:

+ Hai kiểu địa chỉ: Địa chỉ phân đoạn và địa chỉ thực

+ Truy nhập tới địa chỉ phân đoạn

+ Đổi từ địa chỉ phân đoạn sang địa chỉ thực

+ Bộ nhớ màn hình, truy nhập trực tiếp vào bộ nhớ màn hình

+ Dựng con trỏ để lấy dữ liệu từ bộ nhớ phân đoạn

+ Dựng con trỏ hàm để thực hiện các thủ tục của DOS

## Bài 1. Các hàm truy nhập theo địa chỉ phân đoạn

* 1. **Hàm pokeb**: Gửi một ký tự vào bộ nhớ.

+ Nguyên mẫu trong dos.h như sau:

## void pokeb(unsigned seg, unsigned off, char value);

+ Cụng dụng: Gửi giá trị ký tự value vào bộ nhớ tại địa chỉ phân đoạn seg:off

* 1. **Hàm peekb:** Nhận một ký tự từ bộ nhớ.

+ Nguyên mẫu trong dos.h như sau:

## char peekb(unsigned seg, unsigned off);

+ Cụng dụng: Nhận một byte tại địa chỉ phân đoạn seg:off

* 1. **Hàm poke:** Gửi một số nguyên vào bộ nhớ.

+ Nguyên mẫu trong dos.h như sau:

## void poke(unsigned seg, unsigned off, int value);

+ Cụng dụng: Gửi giá trị nguyên value vào bộ nhớ tại địa chỉ phân đoạn seg:off

* 1. **Hàm peek:** Nhận một số nguyên từ bộ nhớ.

+ Nguyên mẫu trong dos.h như sau:

## int peek(unsigned seg, unsigned off);

+ Cụng dụng: Nhận một word tại địa chỉ phân đoạn seg:off

* 1. **Hàm movedata:** Sao các byte.

+ Nguyên mẫu trong mem.h như sau:

## void movedata(unsigned seg\_gui, unsigned off\_gui, unsigned seg\_nhan, unsigned off\_nhan, int n);

+ Cụng dụng: Sao n byte từ seg\_gui:off\_gui đến seg\_nhan:off\_nhan

## Bài 2. Bộ nhớ màn hình văn bản

* 1. **Cách biểu diễn ký tự trong bộ nhớ màn hình**

Bộ nhớ màn hình văn bản bắt đầu từ địa chỉ : (0xb800:0x0000)

Khi đưa một ký tự vào vựng nhớ màn hình, thì nú sẽ hiện lờn màn hình. Mỗi ký tự trên màn hình chiếm 2 byte trong bộ nhớ màn hình: byte đầu chứa mó ASCII, byte thứ hai biểu diễn mầu hiển thị gọi là byte thuộc tính. Các bit của byte thuộc tính:

B7B6B5B4B3B2B1B0

được chia làm 3 nhóm:

+ Nhóm 1 gồm bit B7 biểu thị sự nhấp nháy. Nếu B7=0 thì ký tự khụng nhấp nháy, nếu B7=1 thì ký tự sẽ nhấp nháy.

+ Nhóm 2 gồm các bit B6, B5 và B4. Các bit này chứa được một số nguyên từ 0 đến 7 và biểu thị 8 mầu nền của ký tự.

+ Nhóm 3 gồm các bit B3, B2, B1 và B0. Các bit này chứa được một số nguyên từ 0 đến 15 và biểu thị 16 mầu của ký tự.

## Trang màn hình

Mỗi trang màn hình gồm 80x25 ký tự, do đó cần 80x25x2=4000 byte bộ nhớ. Thực tế mỗi trang màn hình được phân bố 4096 = 0x1000 byte. Như vậy 4 trang màn hình được phân bố như sau:

+ Trang màn hình thứ 0 bắt đầu từ địa chỉ 0xB800:0x0000

+ Trang màn hình thứ 1 bắt đầu từ địa chỉ 0xB800:0x1000

+ Trang màn hình thứ 2 bắt đầu từ địa chỉ 0xB800:0x2000

+ Trang màn hình thứ 3 bắt đầu từ địa chỉ 0xB800:0x3000

## Chọn trang hiển thị

Tại mỗi thời điểm chỉ có thể hiển thị được một trong 4 trang màn hình. Để hiển thị trang màn hình thứ t (t=0,1,2,3) chúng ta sử dụng chức năng 5 của ngắt 0x10 theo mẫu sau:

union REGS v,r;

v.h.ah = 5 ; // Chức năng 5

v.h.al = t ; // Số hiệu trang màn hình cần hiển thị int86(0x10, &v, &r); // Thực hiện ngắt 0x10

## Ví dụ minh hoạ

Ví dụ sau dựng hàm pokeb để đưa các ký tự vào các trang của bộ nhớ màn hình, sau đó dựng chức năng 5 của ngắt 0x10 để chọn trang hiển thị.

*//CT9\_03.CPP*

*#include <dos.h> #include <conio.h>*

*char d1[]={'C',1\*16+14,'H',1\*16+14,'U',1\*16+14,'C',1\*16+14};*

*char d2[]={'M',2\*16+15,'U',2\*16+15,'N',2\*16+15,'G',2\*16+15};*

*void main()*

*{*

*union REGS v,r; clrscr();*

*//Mặc định hiển thị trang 0 for (int i=0;i<8;++i) pokeb(0xb800,i,d1[i]); getch();*

*//Hien thi trang 1 v.h.ah = 5 ; v.h.al = 1 ; int86(0x10,&v,&r);*

*for (i=0;i<8;++i) pokeb(0xb800,0x1000+i,d2[i]); getch();*

*//Hien thi trang 0 v.h.ah = 5 ; v.h.al = 0 ; int86(0x10,&v,&r); getch();*

*//Hien thi trang 1 v.h.ah = 5 ; v.h.al = 1 ; int86(0x10,&v,&r); getch();*

*}*

## Bài 3. Chuyển đổi địa chỉ

* 1. **Để chuyển từ địa chỉ thực sang địa chỉ phân đoạn ta dựng các macro:**

unsigned FP\_SEG(địa\_chỉ\_thực) unsigned FP\_OFF(địa\_chỉ\_thực)

## Để chuyển từ địa chỉ phân đoạn sang địa chỉ thực ta dựng macro:

void far \*MK\_FP(seg,off)

**Ví dụ 1**. Sau khi thực hiện các cõu lệnh: char buf[100];

unsigned ds,dx;

ds = FP\_SEG(buf); dx = FP\_OFF(buf); thì ds:dx chứa địa chỉ của mảng buf.

**Ví dụ 2.** Sau khi thực hiện các cõu lệnh: char far \*pchar;

pchar = (char far\*)MK\_FP(0xb800:0);

thì pchar trỏ tới địa chỉ đầu của bộ nhớ màn hình. Khi đó ta có thể sử dụng các lệnh gán để truy nhập trực tiếp tới bộ nhớ màn hình.

## Bài 4. các ví dụ minh hoạ

**Chương trình 1.** Chương trình minh hoạ cách truy nhập trực tiếp vào bộ nhớ màn hình có địa chỉ đầu là 0xB800:0. Chương trình gồm hàm main() và hai hàm sau:

* + 1. Hàm cuaso

void cuaso(int dongt,int cott,int dongd,int cotd,int maucs);

thiết lập một cửa sổ mầu có toạ độ góc trên-trái là (dongt, cott) và góc dưới-phải là (dongd,cotd). Mầu cho bởi tham số maucs. ở đây sử dụng hàm pokeb và địa chỉ phân đoạn.

* + 1. Hàm duarmh

void duarmh(char \*day, int dong, int cotd, int cotc,int m\_nen, int m\_chu);

sẽ đưa ra màn hình một dẫy ký tự (chứa trong dãy) tại dòng dong, từ cột cotd đến cotc. Mầu nền cho bởi m\_nen, mầu chữ cho bởi m\_chữ. ở đây sử dụng toán tử gán trên địa chỉ thực.

Trong hàm main() sẽ sử dụng các hàm cuaso và duarmh để tạo hai cửa sổ và viết hai dòng chữ trên trang màn hình thứ hai (từ dòng 26 đến dòng 50).

*/\**

*chương trình minh hoạ cách truy nhập trực tiếp vào bộ nhớ của màn hình*

*\*/*

*#include "dos.h" #include "conio.h"*

*void duarmh(char \*day, int dong,I nt cotd, int cotc,I nt m\_nen, int m\_chu);*

*void cuaso(int dongt,int cott,int dongd,int cotd,int maucs); main()*

*{*

495 496

*cuaso(26,1,50,80,BLUE);*

*duarmh("Chuc mung nam moi", 28, 30, 50, MAGENTA, WHITE); cuaso(30,20,46,60,RED);*

*duarmh("Chuc mung nam moi", 40, 30, 50, MAGENTA, YELLOW); getch();*

*}*

*void cuaso(int dongt,int cott,int dongd,int cotd,int maucs)*

*/\* Dung dia phan doan \*/*

*{*

*int i, j, p, t, dt, dd, mau; union REGS v, r;*

*/\* Xac dinh thuoc tinh mau \*/ mau = (maucs << 4)+maucs;*

*/\**

*Xac dinh trang man hinh t*

*va cac chi so dong tren dt, dong duoi dd trong trang t*

*\*/*

*t=(dongt-1)/25;*

*dt=(dongt-1)-t\*25; dd=(dongd-1)-t\*25;*

*/\* Chon t la trang hien thi \*/ v.h.ah=5;v.h.al=t; int86(0x10,&v,&r);*

*/\**

*Dua cac khoang trong (ma 32) va thuoc tinh mau vao cac vi tri thich hop cua bo nho man hinh*

*\*/*

*for (i=dt;i<=dd;++i)*

*{*

*p=t\*4096+i\*160+(cott-1)\*2; for (j=0;j<=cotd-cott;++j)*

*{*

*pokeb(0xb800,p+2\*j,32); pokeb(0xb800,p+2\*j+1,mau);*

*}*

*}*

*}*

*void duarmh(char \*day, int dong, int cotd, int cotc, int m\_nen, int m\_chu)*

*/\* Dung dia chi thuc \*/*

*{*

*int i,p,t,d,kt,mau; char far \*buf; union REGS v,r;*

*/\* Lay dia chi thuc cua bo nho man hinh \*/ buf=(char far\*)MK\_FP(0xb800,0);*

*/\* Xac dinh thuoc tinh mau \*/ mau = (m\_nen << 4)+m\_chu;*

*/\**

*Xac dinh trang man hinh t*

*va cac chi so dong d trong trang t*

*\*/*

*t=(dong-1)/25; d=dong-1-t\*25;*

*/\* Chon t la trang hien thi \*/ v.h.ah=5;v.h.al=t; int86(0x10,&v,&r); p=t\*4096+d\*160+(cotd-1)\*2;*

*/\**

*Dua cac ky tu va thuoc tinh mau*

*vao cac vi tri thich hop cua bo nho man hinh*

*\*/*

*for (i=0;i<=cotc-cotd;++i)*

*{*

*if ((kt=day[i])==0) break; buf[p+2\*i]=kt; buf[p+2\*i+1]=mau;*

*}*

*}*

**Chương trình 2.** Biết địa chỉ của các thủ tục xử lý ngắt được lưu trữ trong bộ nhớ từ địa chỉ 0000:0000 đến 0000:0x0400. Chương trình sẽ cho biết địa chỉ của thủ tục xử lý ngắt n (giá trị n nhập vào từ bàn phím). Số hiệu của ngắt được tính từ 0, nhưng n được đánh số từ 1.

*/\**

*Xac dinh dia chi cac thu tuc ngat \*/ #include "dos.h"*

*#include "conio.h" #include "stdio.h" main()*

*{*

*unsigned char far \*p; /\*p se tro toi bang vecto ngat\*/ int n; /\* n - so hieu ngat, n=1,2,... \*/*

*int k; /\* vi tri cua ngat n trong bang vecto ngat \*/ unsigned seg,off;*

*/\* p tro toi bang vecto ngat \*/ p=(unsigned char far\*)MK\_FP(0,0); clrscr();*

*while(1)*

*{*

*printf("\n So hieu ngat (Bam 0 - Ket thuc): "); scanf("%d",&n); if(n==0) break;*

*k=(n-1)\*4;*

*off=p[k]+256\*p[k+1]; seg=p[k+2]+256\*p[k+3]; printf("\nDia chi %x:%x",seg,off);*

*}*

*}*

**Chương trình 3.** Chương trình minh hoạ cách dựng con trỏ hàm để thực hiện thủ tục khởi động lại máy của DOS, biết địa chỉ đầu của thủ tục này là 0xFFFF:0000 . Chương trình yêu cầu nhập mật khẩu. Nếu chọn đỳng (bấm ABCD và Enter) thì chương trình tiếp tục làm việc, nếu vào sai thì sẽ khởi động lại máy.

*#include <dos.h> #include <conio.h> #include <iostream.h> #include <ctype.h>*

*typedef void far (\*HAM)(void); void khoi\_dong\_may(void)*

*{*

*HAM f;*

*f = (HAM)MK\_FP(0xFFFF,0); f();*

*}*

*char mat\_khau[]= {'A','B','C','D'};*

*int n = sizeof(mat\_khau)/sizeof(char); void main()*

*{*

*char i, ch, sai\_mat\_khau; clrscr();*

*i=0;*

*sai\_mat\_khau=0;*

*cout << "\nMat khau: "; while(1)*

*{*

*ch=getch();*

*if (ch==13) break; cout << '\*' ;*

*if (i<n)*

*{*

*if (toupper(ch)!=mat\_khau[i]) sai\_mat\_khau=1;*

*}*

*else*

*sai\_mat\_khau=1;*

*++i ;*

*}*

*if (sai\_mat\_khau)*

*{*

*cout << "\nSai mat khau, Khoi dong lai may"; getch();*

*khoi\_dong\_may();*

*}*

*else*

*{*

*cout << "\nDung mat khau, tiep tuc chuong trinh"; getch();*

*}*

*}*

**Chương trình 4.** Chương trình minh hoạ cách dựng biến con trỏ để lấy dữ liệu về thời gian hệ thống chứa trong 4 byte bắt đầu từ địa chỉ 0:0x46C . Chương trình cũng minh hoạ cách truy nhập trực tiếp bộ nhớ màn hình văn bản (địa chỉ đầu là 0xB800:0) và cách bắt phím tổng quát. Chương trình sẽ in ra màn hình các chữ cái một cách ngẫu nhiên. Khi bấm phím F1 chương trình tạm dừng để thông báo thời gian. Để kết thúc chương trình bấm phím ESC.

*#include <dos.h>*

*#include <conio.h> #include <stdlib.h>*

*#define VT 132 // vi tri thong bao*

*//Dia chi vung nho man hinh*

*char far \*p\_mh = (char far\*)MK\_FP(0xB800,0) ;*

*//Dia chi 4 byte chua thoi gian*

*unsigned long far \*t\_time=(unsigned long far\*)MK\_FP(0,0x46C); char buf\_time[]={'T',47,'I',47,'M',47,'E',47,':', 47, 32, 47, 32, 47,*

*32, 47, 32, 47, 32, 47, 32, 47, 32, 47, 32, 47, 32, 47};*

*char buf\_luu[28];*

*void thong\_bao\_thoi\_gian()*

*{*

*//Luu trang thai man hinh for (int i=0; i<28; ++i) buf\_luu[i]=p\_mh[i];*

*// Xac dinh gio, phut, giay*

*int gio = (int)(\*t\_time/65543) ; unsigned long du = \*t\_time%65543 ; int phut = (int)(du/1092);*

*du = du%1092;*

*int giay = (int)(du/18);*

*//Doi ra ky tu dua vao mang buf\_time buf\_time[12]=gio/10 + 48;*

*buf\_time[14]=gio%10 + 48;*

*buf\_time[18]=phut/10 + 48;*

*buf\_time[20]=phut%10 + 48;*

*buf\_time[24]=giay/10 + 48;*

*buf\_time[26]=giay%10 + 48;*

*//Dua thong bao goi ra man hinh for (i=0; i<28; ++i)*

*p\_mh[i] = buf\_time[i]; getch();*

*//Khoi phuc man hinh for (i=0; i<28; ++i) p\_mh[i] = buf\_luu[i];*

*}*

*void main()*

*{*

*int ch1, ch2; clrscr();*

*while(1)*

*{*

*if (kbhit())*

*{*

*ch1=getch();*

*if (ch1==0) ch2=getch(); if (ch1==27) //ESC*

*break;*

*if (ch1==0 && ch2==59) // Bam F1 thong\_bao\_thoi\_gian();*

*}*

*//In cac chu cai mot cach ngau nhien gotoxy(random(80)+1,random(25)+1); putch(random(26)+65);*

*delay(400);*

*}*

*}*

**Chương 10**

# Một số chương trình hướng đối tượng trên C++

Chương này trình bầy thêm một số chương trình hướng đối tượng trên C++. Đây là các chương trình tương đối phức tạp, hữu ích và sử dụng các công cụ mạnh của C++ như: Cách truy nhập trực tiếp bộ nhớ màn hình, kỹ thuật đồ hoạ, con trỏ void, tính kế thừa, lớp cơ sở trừu tượng, tương ứng bội, phương thức ảo.

## Bài 1. Lớp cửa sổ

Chương trình gồm lớp cua\_so và lớp stack

## + Lớp cửa sổ Thuộc tính gồm:

char \*noidung; // Trỏ đến vùng nhớ chứa nội dung

// soạn thảo trên cửa sổ

int cao,rong ; // Chiều cao và chiều rộng cửa sổ int mau; // mau = 16\*mau\_nen + mau\_chu

int ra\_mh; // Cho biết cửa sổ đã được đưa ra màn hình chưa? int posx,posy; // Vị trí trên trái của cửa sổ trên màn hình word \*pluu; // Trỏ đến vùng nhớ chứa nội dung

// phần màn hình bị cửa sổ đè lên

## Phương thức gồm:

cua\_so();

cua\_so(int c,int r,byte mau\_nen, byte mau\_chu);

int push(int x,int y); // Đưa cửa sổ ra màn hình tại (x,y)

// cho phép soạn thảo trên cửa sổ

// Bấm F6 chuyển sang cửa sổ khác

// Bấm ESC kết thúc

void pop(); // Tháo gỡ cửa sổ và khôi phục màn hình int get\_ra\_mh();

## + Lớp stack (dùng để quản lý một dẫy cửa sổ) Thuộc tính gồm:

int max; //Số cửa sổ cực đại có thể quản lý

int num; //Số cửa sổ hiện có trong stack cua\_so \*\*pcs; //Con trỏ trỏ đến vùng nhớ chứa

//địa chỉ của các đối tượng cua\_so

## Phương thức gồm:

stack();

stack(int max\_cs);

int accept(cua\_so \*cs,int x,int y); //Đưa một cửa sổ

//vào stack, nó sẽ hiện lên màn hình void del(); // Loại cửa sổ khỏi stack, nó sẽ bị xoá

// khỏi màn hình

## Nội dung chương trình:

+ Đầu tiên hiện cửa sổ thứ nhất nền GREEN chữa WHITE. Có thể soạn thảo trên đó.

+ Nếu bấm ESC kết thúc chương trình, nếu bấm F6 thì hiện thêm cửa sổ thứ hai nền CYAN chữ MAGENTA. Có thể soạn thảo trên đó.

+ Nếu bấm ESC kết thúc chương trình, nếu bấm F6 thì hiện thêm cửa sổ thứ ba nền RED chữ YELLOW. Có thể soạn thảo trên đó.

+ Đang ở một cửa sổ, nếu bấm ESC thì kết thúc chương trình, nếu bấm F6 thì hiện cửa sổ tiếp theo (theo thứ tự vòng quanh: 1 -> 2 -> 3 -> 1).

**Chương trình** sử dụng phương pháp truy nhập trực tiếp bộ nhớ màn hình trình bầy trong chương 9.

*// CT10\_01.CPP*

*// lop cua\_so #include <stdio.h> #include <conio.h> #include <dos.h> #include <alloc.h>*

*typedef unsigned int word; typedef unsigned char byte; struct kt\_word*

*{*

*word kt;*

*};*

*struct kt\_byte*

*{*

*byte ma, mau;*

*};*

*union ky\_tu*

*{*

*struct kt\_byte h; struct kt\_word x;*

*};*

*typedef union ky\_tu far \*VP;*

*VP vptr=(VP)MK\_FP(0xb800,0);*

*// Vi tri x,y tren man hinh*

*#define VPOS(x,y) (VP)(vptr + ((y)-1)\*80+(x)-1) class cua\_so*

*{*

*private:*

*char \*noidung; int cao, rong;*

*int mau; // mau = 16\*mau\_nen + mau\_chu*

*int ra\_mh; int posx,posy; word \*pluu;*

*public:*

*cua\_so();*

*cua\_so(int c,int r,byte mau\_nen, byte mau\_chu); int push(int x,int y);*

*void pop();*

*int get\_ra\_mh();*

*};*

*cua\_so::cua\_so()*

*{*

*cao=rong=mau=ra\_mh=posx=posy=0; noidung=NULL; pluu=NULL;*

*}*

*cua\_so::cua\_so(int c,int r,byte mau\_nen, byte mau\_chu)*

*{*

*cao=c; rong=r;*

*mau= 16\*mau\_nen+mau\_chu; ra\_mh=posx=posy=0;*

*noidung = (char\*)malloc(cao\*rong); for (int i=0;i<cao\*rong;++i) noidung[i]=32;*

*pluu= (word\*)malloc(2\*cao\*rong);*

*}*

*int cua\_so::push(int x,int y)*

*{*

*word \*p= pluu; char \*pnd=noidung; VP ptr;*

*int i,j;*

*// Luu man hinh if (ra\_mh==0)*

*{*

*ra\_mh=1; posx=x;posy=y;*

*for (i=posx;i<=posx+rong-1;++i) for (j=posy;j<=posy+cao-1;++j)*

*{*

*ptr=VPOS(i,j); \*p=ptr->x.kt; ++p;*

*}*

*}*

*// Hien noi dung dang soan thao tren cua so*

*for (i=posx;i<=posx+rong-1;++i) for (j=posy;j<=posy+cao-1;++j)*

*{*

*ptr=VPOS(i,j); ptr->h.mau=mau;*

*ptr->h.ma=\*pnd; ++pnd;*

*}*

*// Soan thao*

*int xx=posx,yy=posy,ch1,ch2; while (1)*

*{*

*gotoxy(xx,yy);*

*if ((ch1=getch())==0) ch2=getch();*

*if (ch1==27)break; // ESC Ket Thuc Soan Thao else if (ch1==0&&ch2==64)break; //F6*

*else if (ch1==13)*

*{*

*++yy; xx=posx; if(yy>=posy+cao) break;*

*}*

*else if (ch1!=0)*

*{*

*ptr=VPOS(xx,yy); ptr->h.ma=ch1;*

*++xx;*

*if (xx>=posx+rong) {++yy; xx=posx;} if (yy>=posy+cao) break;*

*}*

*else if (ch2==72||ch2==80||ch2==75||ch2==77)*

*{*

*if (ch2==72) yy--;*

*else if (ch2==80) ++yy; else if (ch2==75) --xx; else ++xx;*

*if (xx<posx) xx=posx;*

*if (xx>=posx+rong) {++yy; xx=posx;} if (yy<posy) yy=posy;*

*if (yy>=posy+cao) break;*

*}*

*}*

*// Luu ket qua soan thao pnd=noidung;*

*for (i=posx;i<=posx+rong-1;++i)*

*for (j=posy;j<=posy+cao-1;++j)*

*{*

*ptr=VPOS(i,j);*

*\*pnd=ptr->h.ma; ++pnd;*

*}*

*if (ch1==0&&ch2==64) return 0; //F6 else return 1;*

*}*

*void cua\_so::pop() // Khoi phuc vung nho bi cua so chiem*

*{*

*if (ra\_mh==0) return; ra\_mh=0;*

*word \*p=pluu;*

*VP ptr;*

*int i,j;*

*for (i=posx;i<=posx+rong-1;++i) for (j=posy;j<=posy+cao-1;++j)*

*{*

*ptr=VPOS(i,j); ptr->x.kt=\*p; ++p;*

*}*

*}*

*int cua\_so::get\_ra\_mh()*

*{*

*return ra\_mh;*

*}*

*//class stack class stack*

*{*

*private:*

*int max,num; cua\_so \*\*pcs;*

*public:*

*stack();*

*stack(int max\_cs);*

*int accept(cua\_so \*cs,int x,int y); void del();*

*};*

*stack::stack()*

*{*

*max=num=0; pcs=NULL;*

*}*

*stack::stack(int max\_cs)*

*{*

*max=max\_cs; num=0; pcs=(cua\_so\*\*)malloc(max\*sizeof(cua\_so\*)); for (int i=0;i<max;++i) pcs[i]=NULL;*

*}*

*int stack::accept(cua\_so \*cs,int x,int y)*

*{*

*int gt;*

*if (num==max)return 0; if (!cs->get\_ra\_mh())*

*{*

*pcs[num]=cs; ++num;*

*}*

*gt=cs->push(x,y); return gt;*

*}*

*void stack::del()*

*{*

*if (num==0) return;*

*--num;*

*pcs[num]->pop(); pcs[num]=NULL;*

*}*

*main()*

*{*

*int ch;*

*cua\_so w1(10,40,GREEN,WHITE), w2(12,42,CYAN,MAGENTA), w3(14,44,RED,YELLOW);*

*stack s(4); clrscr(); while(1)*

*{*

*ch=s.accept(&w1,5,5); if(ch==1)break; ch=s.accept(&w2,8,8); if(ch==1)break; ch=s.accept(&w3,11,11);*

*if(ch==1)break;*

*}*

*s.del(); s.del(); s.del();*

*}*

## Bài 2. Lớp menu

Lớp cmenu có 2 phương thức để tạo lập và sử dụng menu:

## Hàm tạo

cmenu(int so\_cn\_menu,char \*\*nd\_menu);

dùng để tạo một menu (đối tượng kiểu cmenu). Hàm tạo chứa 2 đối là:

+ Biến so\_cn\_menu chứa số chức năng của menu

+ Con trỏ nd\_menu trỏ tới một vùng nhớ chứa địa chỉ các chuỗi ký tự dùng làm tiêu đề menu và tiêu đề các chức năng menu.

**Ví dụ** các câu lệnh:

char \*nd[]={"Quản lý vật tư", "Nhập số liệu",

"Tìm kiếm","Kết thúc"}; cmenu mc(3,nd);

sẽ tạo một menu mc gồm 3 chức năng: Nhập số liệu, Tìm kiếm và Kết thúc. Menu có tiêu đề là: Quản lý vật tư

## Phương thức

int menu(int x,int y,int mau\_nen,int mau\_chon); thực hiện các việc sau:

+ Hiển thị menu tại vị trí (x,y) trên màn hình. Menu có mầu nền xác định bởi đối mau\_nen và mầu chức năng định chọn (hộp sáng) xác định bởi đối mau\_chon.

+ Cho phép sử dụng các phím mũi tên lên, xuống để di chuyển hộp sáng và dùng phím Enter để thoát khỏi phương thức.

+ Sau khi thoát khỏi, phương thức trả về giá trị bằng số thứ tự (tính từ 1) của chức năng được chọn.

Chương trình dưới đây xây dựng lớp cmenu và minh hoạ cách sử dụng lớp này.

*/\**

*CT10\_02.CPP*

*menu.cpp lop cmenu*

*\*/*

*#include <stdio.h> #include <conio.h> #include <dos.h> #include <alloc.h> #include <string.h> typedef unsigned int word;*

*typedef unsigned char byte;*

*struct kt\_word*

*{*

*word kt;*

*};*

*struct kt\_byte*

*{*

*byte ma, mau;*

*};*

*union ky\_tu*

*{*

*struct kt\_byte h; struct kt\_word x;*

*};*

*typedef union ky\_tu far \*VP;*

*VP vptr=(VP)MK\_FP(0xb800,0);*

*// Vi tri x,y tren man hinh*

*#define VPOS(x,y) (VP)(vptr + ((y)-1)\*80+(x)-1) class cmenu*

*{*

*private:*

*int so\_cn,cao,rong,posx,posy; int chon;*

*char \*\*nd; private:*

*void hiendc(char \*dc,int x,int y, int mau);*

*void hien\_menu(int x,int y,int mau\_nen,int mau\_chon); public:*

*cmenu(int so\_cn\_menu,char \*\*nd\_menu);*

*int menu(int x,int y,int mau\_nen,int mau\_chon);*

*};*

*cmenu::cmenu(int so\_cn\_menu,char \*\*nd\_menu)*

*{*

*cao=so\_cn=so\_cn\_menu; nd=nd\_menu; rong=0;*

*chon=1; int d;*

*for(int i=0;i<=so\_cn;++i)*

*if( (d=strlen(nd[i])) > rong) rong=d;*

*}*

*void cmenu::hiendc(char \*dc,int x,int y, int mau)*

*{*

*VP ptr; int i;*

*byte m=16\*mau+15; //chu trang for(i=0;i<rong;++i)*

*{*

*ptr=VPOS(x+i,y); ptr->h.mau=m ; ptr->h.ma=32;*

*}*

*for(i=0;i<rong;++i)*

*{*

*ptr=VPOS(x+i,y); if(dc[i]==0)break; ptr->h.ma=dc[i];*

*}*

*}*

*void cmenu::hien\_menu(int x,int y,int mau\_nen,int mau\_chon)*

*{*

*for(int i=0;i<=so\_cn;++i) hiendc(nd[i],x,y+i,mau\_nen);*

*hiendc(nd[chon],x,y+chon,mau\_chon);*

*}*

*int cmenu::menu(int x,int y,int mau\_nen,int mau\_chon)*

*{*

*int ch1,ch2,chonluu;*

*//Trinh bay hien\_menu(x,y,mau\_nen,mau\_chon);*

*//Bat phim while(1)*

*{*

*if( (ch1=getch())==0 ) ch2=getch(); if(ch1==13) //chon chuc nang*

*return (chon);*

*else if( (ch1==0)&&(ch2==80||ch2==72))*

*{*

*//Di chuyen hop sang chonluu=chon; if(ch2==80) ++chon; else --chon; if(chon<1) chon=cao;*

*else if(chon>cao) chon=1;*

*if(chon!=chonluu)*

*{*

*hiendc(nd[chonluu],x,y+chonluu,mau\_nen); hiendc(nd[chon],x,y+chon,mau\_chon);*

*}*

*}*

*}*

*}*

*char \*nd[]={"TINH DIEN TICH", "Tam giac","Hinh tron", "Chu nhat", "Hinh vuong", "Ket thuc chuong trinh"};*

*void main()*

*{*

*cmenu mc(5,nd); int chon; clrscr();*

*while(1)*

*{*

*chon=mc.menu(5,5,BLUE,MAGENTA); if(chon==1)*

*{*

*clrscr();*

*puts("TAM GIAC");*

*getch(); clrscr();*

*}*

*else if(chon==2)*

*{*

*clrscr();*

*puts("HINH TRON");*

*getch();clrscr();*

*}*

*else if(chon==3)*

*{*

*clrscr();*

*puts("CHU NHAT");*

*getch();clrscr();*

*}*

*else if(chon==4)*

*{*

*clrscr();*

*puts("HINH VUONG");*

*getch(); clrscr();*

*}*

*else break;*

*}*

*}*

## Bài 3. Lớp hình học

Chương trình dưới đây gồm:

+ Lớp “hinh” là lớp cơ sở trừu tượng

+ Và 3 lớp dẫn suất từ lớp “hình” là:

* Lớp “khoihop” biểu thị các khối hộp lập phương
* Lớp “duong” biểu thị các đoạn thẳng qua 2 điểm
* Lớp “tron” biểu thị các đường tròn

Chương trình minh hoạ cách dùng tượng ứng bội và phương thức ảo. Nội dung chương trình như sau:

+ Khi chạy chương trình sẽ thấy xuất hiện một khối hộp lập phương.

+ Có thể di chuyển khối hộp bằng các phím mũi tên.

+ Bấm phím Q sẽ xuất hiện một đoạn thẳng.

+ Có thể di chuyển đoạn thẳng bằng các phím mũi tên.

+ Bấm phím Q sẽ xuất hiện một đường tròn.

+ Có thể di chuyển đường tròn bằng các phím mũi tên.

+ Bấm phím Q sẽ kết thúc chương trình.

/\*

CT10\_03.CPP

LOP hinh hoc

Minh hoa cach dung:

+ lop co so truu tuong

+ Tuong ung boi va phuong thuc ao

*\*/*

*#include <graphics.h> #include <process.h> #include <stdio.h> #include <conio.h>*

*char getkey(int &dx,int &dy); class hinh*

*{*

*protected: int mau;*

*public:*

*hinh(void)*

*{*

*mau=0;*

*}*

*hinh(int m)*

*{*

*mau=m;*

*}*

*virtual void dchuyen(int b)=0;*

*};*

*class khoihop : public hinh*

*{*

*private:*

*int x,y; int a ;*

*public:*

*khoihop(void):hinh()*

*{*

*x=y=a=0;*

*}*

*khoihop(int m,int x1,int y1, int a1):hinh(m)*

*{*

*x=x1; y=y1; a=a1;*

*}*

*virtual void dchuyen(int b); void hien(void)*

*{*

*setfillstyle(1,mau); bar3d(x,y,x+a,y+a,a/2,1);*

*}*

*void an(void)*

*{*

*setfillstyle(1,getbkcolor()); bar(x,y-a/2,x+a+a/2,y+a+a/2);*

*}*

*};*

*class duong:public hinh*

*{*

*private:*

*int x1,y1,x2,y2; public:*

*duong(void):hinh()*

*{*

*x1=x2=y1=y1=0;*

*}*

*duong(int m,int a,int b,int c,int d):hinh(m)*

*{*

*x1=a;y1=b;x2=c;y2=d;*

*}*

*virtual void dchuyen(int b); void hien(void)*

*{*

*setcolor(mau); line(x1,y1,x2,y2);*

*}*

*void an(void)*

*{*

*setcolor(getbkcolor()); line(x1,y1,x2,y2);*

*}*

*};*

*class tron:public hinh*

*{*

*private:*

*int x,y,r; public:*

*tron(void):hinh()*

*{*

*x=y=r=0;*

*}*

*tron(int m,int a,int b,int d):hinh(m)*

*{*

*x=a; y=b; r=d;*

*}*

*virtual void dchuyen(int b); void hien(void)*

*{*

*setcolor(mau); circle(x,y,r);*

*}*

*void an(void)*

*{*

*setcolor(getbkcolor()); circle(x,y,r);*

*}*

*};*

*char getkey(int &dx,int &dy)*

*{*

*int ch1,ch2; dx=dy=0; while (1)*

*{*

*ch1=getch(); if (ch1==0)*

*ch2=getch();*

*if (ch1=='q'||ch1=='Q') return('q');*

*if ((ch1==0&&(ch2==80||ch2==72||ch2==75||ch2==77)))*

*{*

*if (ch2==80) dy=1;*

*else if (ch2==72) dy=-1; else if (ch2==77) dx=1; else dx=-1;*

*return(0);*

*}*

*}*

*}*

*void khoihop::dchuyen(int b)*

*{*

*int dx,dy; while (1)*

*{*

*hien();*

*if (getkey(dx,dy)=='q') break; an();*

*x+=b\*dx; y+=b\*dy;*

*}*

*}*

*void duong::dchuyen(int b)*

*{*

*int dx,dy; while (1)*

*{*

*hien();*

*if (getkey(dx,dy)=='q') break; an();*

*x1+=b\*dx; x2+=b\*dx; y1+=b\*dy; y2+=b\*dy;*

*}*

*}*

*void tron::dchuyen(int b)*

*{*

*int dx,dy; while (1)*

*{*

*hien();*

*if (getkey(dx,dy)=='q') break; an();*

*x+=b\*dx; y+=b\*dy;*

*}*

*}*

*void main()*

*{*

*int mh=0,mode=0; initgraph(&mh,&mode,""); if (graphresult())*

*{*

*printf("\n LOI"); getch();*

*exit(0);*

*}*

*setbkcolor(0);*

*// setwritemode(0); hinh \*h[3];*

*khoihop M(4,300,200,15); duong D(10,10,10,60,60); tron T(14,200,200,50);*

*h[0]=&M; h[1]=&D;h[2]=&T;*

*for(int i=0;i<3;++i) h[i]->dchuyen(10); closegraph();*

*}*

## Bài 4. Các lớp ngăn xếp và hàng đợi

Chương trình tổ chức thành 4 lớp chính:

1. **Lớp container** (thùng chứa) gồm 2 thuộc tính: unsigned long count; //Số phần tử trong thùng chứa void (\*errhandler)(); //Con trỏ tới hàm xử lý lỗi
2. **Lớp s\_list thừa kế từ lớp container**, có thêm 2 thuộc tính các con trỏ kiểu cấu trúc listnode: struct listnode

{

void \*dataptr; listnode \*next;

};

listnode \*head; // Trỏ tới đầu danh sách listnode \*tail; // Trỏ tới cuối danh sách

Các phần tử được chứa trong lớp s\_list dưới dạng một danh sách móc nối đơn. Mỗi nút chứa địa chỉ của một phần tử. Do ở đây dùng kiểu con trỏ void nên có thể đưa vào lớp s\_list các phần tử có kiểu bất kỳ.

## Lớp stack thừa kế từ lớp s\_list

1. **Lớp queue thừa kế từ lớp stack**

Các lớp stack và queue không có các thuộc tính riêng. Hai phương thức quan trọng của các lớp này là:

virtual int store(void \*item) ; // Cất vào một phần tử virtual void \*retrieve () ; // Lấy ra một phần tử

***Chú ý là:*** Lớp stack hoạt động theo nguyên tắc LIFO (vào sau ra trước) còn lớp queue hoạt động theo nguyên tắc FIFO (vào trước ra trước) .

Chương trình sau minh hoạ cách dùng liên kết bội, phương thức ảo và con trỏ kiểu void để quản lý các kiểu dữ liệu khác nhau.

*Hoạt động của chương trình như sau:*

+ Trước tiên lần lượt đưa địa chỉ của biến đối tượng ts1, chuỗi “HA NOI”, biến nguyên a, biến đối tượng ts2 và biến thực x vào ngăn xếp s1 và hàng đợi q1.

+ Thực hiện phép gán các biến đối tượng: s2 = s1 ;

q2 = q1 ;

+ Lấy các phần tử trong ngăn xếp s2 theo trình tự ngược với lúc đưa vào.

+ Lấy các phần tử trong hàng đợi q2 theo trình tự như lúc đưa vào.

/\*

CT10\_05.CPP

Lop vat chua (container) Lop danh sach moc noi Lop ngan xep

Lop hang doi Chu y:

* 1. constructor sao chep cua lop dan suat
  2. toan tu gan cua lop dan suat
  3. co the dung cac phuong thuc khac de viet constructor va destructor
  4. Dung con tro this

*\*/*

*#include <stdio.h> #include <iostream.h> #include <iomanip.h> #include <conio.h> #include <alloc.h> #include <dos.h>*

*//Lop container class container*

*{*

*protected:*

*unsigned long count; //so pt trong thung chua void (\*errhandler)();*

*public:*

*container();*

*container(const container &c); // Ham tao sao chep void operator=(const container &c); // Gan unsigned long getcount(); // Cho biet so phan tu*

*// Dinh ham xl loi*

*void seterrorhandler(void (\*userhandler)());*

*// 4 phuong thuc thuan ao*

*virtual int store(void \*item)=0;//Cat mot phan tu vao thung virtual void \*examine()=0; // Xem gia tri mot phan tu virtual void \*retrieve ()=0; // Lay mot pt ra*

*virtual void empty()=0; // Lam cho thung tro nen rong*

*};*

*// Cai dat*

*// Ham xl loi mac dinh void defaulthandler(); void defaulthandler()*

*{*

*puts("\nContainer error: memory allocation failure");*

*}*

*container::container ()*

*{*

*count=0; errhandler= defaulthandler;*

*}*

*container::container(const container &c)*

*{*

*count=c.count; errhandler=c.errhandler;*

*}*

*// Gan*

*void container::operator=(const container &c)*

*{*

*count=c.count; errhandler=c.errhandler;*

*}*

*// Cho biet so pt*

*unsigned long container::getcount()*

*{*

*return count;*

*}*

*// Dinh ham xl loi*

*void container::seterrorhandler(void (\*userhandler)())*

*{*

*errhandler=userhandler;*

*}*

*// Lop danh sach moc noi don class s\_list:public container*

*{*

*protected:*

*//Cau truc mot nut trong ds struct listnode*

*{*

*void \*dataptr; listnode \*next;*

*};*

*listnode \*head; listnode \*tail;*

*private:*

*// phuong thuc sao chep void copy(const s\_list &s1);*

*public:*

*s\_list();*

*s\_list(const s\_list &s1);*

*~s\_list();*

*void operator=(const s\_list &s1);*

*// 4 phuong thuc ao*

*virtual int store(void \*item)=0; // Cat mot phan tu vao*

*// thung*

*virtual void \*examine()=0; // Xem gia tri mot phan tu virtual void \*retrieve ()=0; // Lay mot pt ra*

*virtual void empty(); // Lam cho thung tro nen rong*

*};*

*//Cai dat*

*void s\_list::copy(const s\_list &s1)*

*{*

*head=NULL; tail=NULL; listnode \*temp = s1.head; while(temp!=NULL)*

*{*

*if(head==NULL)*

*{*

*head= new listnode; if(head==NULL) errhandler(); tail=head;*

*}*

*else*

*{*

*tail->next = new listnode;*

*if(tail->next == NULL) errhandler(); tail = tail->next;*

*}*

*tail->dataptr= temp->dataptr; tail->next=NULL;*

*temp = temp->next;*

*}*

*}*

*// constructor s\_list::s\_list() : container()*

*{*

*head=NULL; tail=NULL;*

*}*

*s\_list::s\_list(const s\_list &s1):container(s1)*

*{*

*copy(s1);*

*}*

*s\_list::~s\_list()*

*{*

*this->empty();*

*}*

*void s\_list::operator=(const s\_list &s1)*

*{*

*this->empty(); count=s1.count; copy(s1);*

*}*

*void s\_list::empty()*

*{*

*listnode \*q,\*p;*

*p = head; head=NULL; tail=NULL; while (p!=NULL)*

*{*

*q=p; p=p->next; delete q;*

*}*

*}*

*// Lop stack*

*class stack:public s\_list*

*{*

*public:*

*stack();*

*stack(const stack &st);*

*void operator=(const stack &st);*

*virtual int store(void \*item); // Cat mot phan tu vao thung virtual void \*examine(); // Xem gia tri mot phan tu virtual void \*retrieve(); // Lay mot pt ra*

*};*

*stack::stack():s\_list()*

*{*

*}*

*stack::stack(const stack &st):s\_list(st)*

*{*

*}*

*void stack::operator=(const stack &st)*

*{*

*this->s\_list::operator=(st); //Dung toan tu gan cua s\_list*

*}*

*int stack::store(void \*item) // Cat mot phan tu vao thung*

*{*

*//Dua vao dau danh sach listnode \*p;*

*p= new listnode ; if(p==NULL) return 1; count++;*

1. *>dataptr=item; p->next=head; head=p; return 0;*

*}*

*void \*stack::examine() // Xem gia tri mot phan tu*

*{*

*if(count==0) return NULL; else*

*return head->dataptr;*

*}*

*void \*stack::retrieve() // Lay mot pt ra*

*{*

*if(count==NULL) return NULL; else*

*{*

*listnode \*p; void \*value; value = head->dataptr; p=head;*

*head = p->next; delete p;*

*count--; return value;*

*}*

*}*

*// Lop queue*

*class queue:public stack*

*{*

*public:*

*queue();*

*queue(const queue &q);*

*void operator=(const queue &q);*

*virtual int store(void \*item); // Cat mot phan tu vao thung*

*};*

*queue::queue(): stack()*

*{*

*}*

*queue::queue(const queue &q):stack(q)*

*{*

*}*

*void queue::operator=(const queue &q)*

*{*

*this->stack::operator=(q); //Dung toan tu gan cua stack*

*}*

*int queue::store(void \*item)*

*{*

*// Dat vao cuoi listnode \*q; q=new listnode;*

*if(q==NULL)return 1;*

*// Bo sung*

1. *>next=NULL; q->dataptr=item; if(count==0)*

*{*

*head=q; tail=q;*

*}*

*else*

*{*

*tail->next=q; tail=q;*

*}*

*count++; return 0;*

*}*

*class TS*

*{*

*private:*

*char ht[25]; int sobd; float td;*

*public:*

*void nhap()*

*{*

*cout << "\nHo ten: " ; fflush(stdin);*

*gets(ht);*

*cout << "So bao danh: " ;*

*cin >> sobd;*

*cout << "Tong diem: " ; cin >> td;*

*}*

*void xuat()*

*{*

*cout << "\nHo ten: " << ht;*

*cout << "\nSo bao danh: " << sobd;*

*cout << "\nTong diem: " << setiosflags(ios::showpoint)*

*<< setprecision(1)<<setw(5)<< td;*

*}*

*};*

*// Ham main void main()*

*{*

*stack s1,s2; queue q1,q2; TS ts1,ts2,ts;*

*int a=123,b; float x=3.14,y; char \*str; clrscr();*

*ts1.nhap();*

*ts2.nhap();*

*//Gui vao*

*s1.store(&ts1); q1.store(&ts1); s1.store("HA NOI"); q1.store("HA NOI"); s1.store(&a); q1.store(&a); s1.store(&ts2); q1.store(&ts2); s1.store(&x); q1.store(&x);*

*//Lay ra tu ngan xep theo nguyen tac LIFO cout <<"\n\nLay ra tu ngan xep:" ; s2=s1;*

*y = \*((float\*)s2.retrieve());*

*cout << "\nSo thuc = " <<setiosflags(ios::showpoint)*

*<< setprecision(2)<< y; ts = \*((TS\*)s2.retrieve()); ts.xuat();*

*b = \*((int\*)s2.retrieve());*

*cout << "\nSo nguyen = " << b; str = (char\*)s2.retrieve();*

*cout << "\nChuoi ky tu: " << str; ts = \*((TS\*)s2.retrieve()); ts.xuat();*

*//Lay ra tu hang doi theo nguyen tac FIFO cout <<"\n\nLay ra tu hang doi:" ; q2=q1;*

*ts = \*((TS\*)q2.retrieve()); ts.xuat();*

*str = (char\*)q2.retrieve();*

*cout << "\nChuoi ky tu: " << str; b = \*((int\*)q2.retrieve());*

*cout << "\nSo nguyen = " << b; ts = \*((TS\*)q2.retrieve()); ts.xuat();*

*y = \*((float\*)q2.retrieve());*

*cout << "\nSo thuc = " << setiosflags(ios::showpoint)*

*<< setprecision(2)<< y; getch();*

*}*

## Bài 5. Các lớp sắp xếp

Trong tệp C\_SORT.H dưới đây sẽ chứa 4 lớp sắp xếp: sort, select\_sort, quick\_sort và heap\_sort. tổng quát hơn. So với các lớp sắp xếp trong mục §7 chương 6 thì các lớp ở đây tổng quát hơn ở chỗ:

+ Các lớp trong mục §7 chương 6 chỉ cho phép sắp xếp một dẫy số nguyên theo thứ tự tăng dần.

+ Các lớp dưới đây cho phép sắp xếp một dẫy phần tử có kiểu bất kỳ (nguyên, thực, cấu trúc, lớp, ...) và theo một tiêu chuẩn sắp xếp bất kỳ.

1. **Lớp sort** là lớp cơ sở trừu tượng

## + Các thuộc tính:

protected:

void \*a ; // Trỏ tới vùng nhớ chứa dẫy

// phần tử cần sắp xếp

int size ; // Độ lớn tính theo byte của phần tử

int (\*nho\_hon)(void\* pt1, void\* pt2); // Con trỏ hàm

// định nghĩa pt1 nhỏ hơn pt2

## + Các phương thức:

protected:

void hoan\_vi(int i, int j) ; // Hoán vị các phần tử thứ i và j

void \* dia\_chi (int m); // Cho địa chỉ của phần tử thứ m public:

virtual void sapxep(void \*a1,int n,int itemsize,

int (\*ss\_nho\_hon)(void\* ,void\* )) ; // Sắp xếp dẫy

// n phần tử chứa trong vùng nhớ a1, mỗi phần tử

// có độ dài itemsize, thứ tự tăng được quy định

// bởi hàm ss\_nho\_hon

1. **Lớp select\_sort** dẫn xuất từ lớp sort. Lớp này sẽ thực hiện việc sắp xếp theo phương pháp chon (xem mục §7 chương 6).

## + Các phương thức:

public:

virtual void sapxep(void \*a1,int n,int itemsize,

int (\*ss\_nho\_hon)(void\* ,void\* )) ; // thực hiện

// sắp xếp theo phương pháp chọn

1. **Lớp quick\_sort** dẫn xuất từ lớp sort. Lớp này sẽ thực hiện việc sắp xếp theo phương pháp quick sort (xem mục §7 chương 6)

## + Các phương thức:

private:

void q\_sort(int l, int r); public:

virtual void sapxep(void \*a1,int n,int itemsize,

int (\*ss\_nho\_hon)(void\* ,void\* )) ; // thực hiện

// sắp xếp theo phương pháp quick sort

1. **Lớp heap\_sort** dẫn xuất từ lớp sort. Lớp này sẽ thực hiện việc sắp xếp theo phương pháp heap sort (xem mục §7 chương 6).

## + Các phương thức:

private:

void shift(int i, int n); public:

virtual void sapxep(void \*a1,int n,int itemsize,

int (\*ss\_nho\_hon)(void\* ,void\* )) ; // thực hiện

// sắp xếp theo phương pháp heap sort Dưới đây là nội dung tệp C\_SORT.H

*//C\_SORT.H*

*// Lop co so truu tuong*

*// Lop sort*

*#include <conio.h>*

*#include <stdio.h> #include <stdlib.h> #include <iostream.h> #include <iomanip.h> #include <mem.h> class sort*

*{*

*protected: void \*a; int size;*

*int (\*nho\_hon)(void\*,void\*); void\* dia\_chi(int m)*

*{*

*return (void\*) ((char\*)a + size\*(m-1));*

*}*

*void hoan\_vi(int i, int j)*

*{*

*void \*tg, \*di, \*dj; di= dia\_chi(i); dj= dia\_chi(j);*

*tg = new char[size]; memcpy(tg,di,size); memcpy(di,dj,size); memcpy(dj,tg,size);*

*}*

*public:*

*virtual void sapxep(void \*a1,int n,int itemsize,*

*int (\*ss\_nho\_hon)(void\*,void\*))*

*{*

*a=a1;*

*size=n; // Cho C++ hai long size=itemsize;*

*nho\_hon= ss\_nho\_hon;*

*}*

*} ;*

*class select\_sort : public sort*

*{*

*public:*

*virtual void sapxep(void \*a1,int n,int itemsize,*

*int (\*ss\_nho\_hon)(void\*,void\*)) ;*

*} ;*

*void select\_sort::sapxep(void \*a1,int n,int itemsize,*

*int (\*ss\_nho\_hon)(void\*,void\*))*

*{*

*int i,j,r;*

*sort::sapxep(a1,n,itemsize,ss\_nho\_hon); for(i=1; i<n; ++i)*

*{*

*r=i;*

*for(j=i+1; j<=n; ++j) if(nho\_hon(dia\_chi(j),dia\_chi(r))) r = j; if(r!=i) hoan\_vi(i,r);*

*}*

*}*

*class quick\_sort : public sort*

*{*

*private:*

*void q\_sort(int l, int r); public:*

*virtual void sapxep(void \*a1,int n,int itemsize,*

*int (\*ss\_nho\_hon)(void\*,void\*)) ;*

*} ;*

*void quick\_sort::q\_sort(int l, int r)*

*{*

*void \*x; int i,j;*

*x = new char[size]; if(l < r)*

*{*

*memcpy(x, dia\_chi(l), size); i = l; j = r+1;*

*do*

*{*

*++i; --j;*

*while(i<r && nho\_hon(dia\_chi(i),x)) ++i ; while(nho\_hon(x,dia\_chi(j)) ) --j ;*

*if(i<j) hoan\_vi(i,j);*

*} while (i<j);*

*hoan\_vi(l,j);*

*q\_sort(l,j-1); q\_sort(j+1,r);*

*}*

*}*

*void quick\_sort::sapxep(void \*a1,int n,int itemsize,*

*int (\*ss\_nho\_hon)(void\*,void\*))*

*{*

*sort::sapxep(a1,n,itemsize,ss\_nho\_hon); q\_sort(1,n);*

*}*

*class heap\_sort : public sort*

*{*

*private:*

*void shift(int i, int n); public:*

*virtual void sapxep(void \*a1,int n,int itemsize,*

*int (\*ss\_nho\_hon)(void\*,void\*));*

*} ;*

*void heap\_sort::shift(int i, int n)*

*{*

*int l,r,k;*

*l = 2\*i; r = l+1; if(l>n) return; if(l==n)*

*{*

*if (nho\_hon(dia\_chi(i), dia\_chi(l))) hoan\_vi(i,l);*

*return;*

*}*

*if(nho\_hon(dia\_chi(r), dia\_chi(l))) k = l;*

*else*

*k = r;*

*if (!nho\_hon(dia\_chi(i), dia\_chi(k))) return;*

*else*

*{*

*hoan\_vi(i,k);*

*shift(k,n);*

*}*

*}*

*void heap\_sort::sapxep(void \*a1,int n,int itemsize,*

*int (\*ss\_nho\_hon)(void\*,void\*))*

*{*

*long i;*

*sort::sapxep(a1,n,itemsize,ss\_nho\_hon);*

*// Tao dong*

*for(i=n/2 ; i>=1; --i) shift(i,n);*

*// Lap*

*for(i=n ; i>=2; --i)*

*{*

*hoan\_vi(1,i);*

*shift(1,i-1);*

*}*

*}*

## Bài 6. Ví dụ về Các lớp sắp xếp

Trong mục này trình bầy 2 chương trình minh hoạ cách dùng các lớp nói trên. Chương trình thứ nhất minh hoạ cách sử dụng các lớp trong tệp C\_SORT.H để sắp xếp một dẫy thí sinh theo thứ tự giảm và thứ tự tăng của tổng điểm. Chương trình thứ hai minh hoạ cách dùng các lớp trong C\_SORT.H để sắp xếp một dẫy số nguyên theo chiều tăng và chiều giảm.

## Chương trình 1

*//CT10-08*

*// Lop co so truu tuong*

*// Lop sort #include "c\_sort.h" class TS*

*{*

*private:*

*char ht[25]; int sobd; float td;*

*public:*

*float get\_td()*

*{*

*return td;*

*}*

*void nhap()*

*{*

*cout << "\nHo ten: " ; fflush(stdin);*

*gets(ht);*

*cout << "So bao danh: " ; cin >> sobd;*

*cout << "Tong diem: " ; cin >> td;*

*}*

*void xuat()*

*{*

*cout << "\nHo ten: " << ht;*

*cout << "\nSo bao danh: " << sobd;*

*cout << "\nTong diem: " << setiosflags(ios::showpoint)*

*<< setprecision(1)<<setw(5)<< td;*

*}*

*};*

*int ss\_tong\_diem\_giam(void \*ts1, void \*ts2)*

*{*

*return ( ((TS\*)ts1)->get\_td() > ((TS\*)ts2)->get\_td()) ;*

*}*

*int ss\_tong\_diem\_tang(void \*ts1, void \*ts2)*

*{*

*return ( ((TS\*)ts1)->get\_td() < ((TS\*)ts2)->get\_td()) ;*

*}*

*void main()*

*{*

*TS t[100];*

*sort \*sa; int n,i; clrscr();*

*cout << "\nSo thi sinh: "; cin >> n;*

*for(i=1; i<=n; ++i) t[i].nhap();*

*for(i=1; i<=n; ++i) t[i].xuat(); getch();*

*cout << "\n\nSap xep giam theo tong diem - PP Select Sort" ; sa= new select\_sort;*

*sa->sapxep( t+1,n,sizeof(TS),ss\_tong\_diem\_giam); for(i=1; i<=n; ++i) t[i].xuat();*

*delete sa; getch();*

*cout << "\n\nSap xep tang theo tong diem - PP Select Sort"; sa= new select\_sort;*

*sa->sapxep( t+1,n,sizeof(TS),ss\_tong\_diem\_tang); for(i=1; i<=n; ++i) t[i].xuat();*

*delete sa; getch();*

*cout << "\n\nSap xep giam theo tong diem - PP Quick Sort" ;*

*sa= new quick\_sort;*

*sa->sapxep( t+1,n,sizeof(TS),ss\_tong\_diem\_giam); for(i=1; i<=n; ++i) t[i].xuat();*

*delete sa; getch();*

*cout << "\n\nSap xep tang theo tong diem - PP Quick Sort" ; sa= new quick\_sort;*

*sa->sapxep( t+1,n,sizeof(TS),ss\_tong\_diem\_tang); for(i=1; i<=n; ++i) t[i].xuat();*

*delete sa; getch();*

*cout << "\n\nSap xep giam theo tong diem - PP Heap Sort" ; sa= new heap\_sort;*

*sa->sapxep( t+1,n,sizeof(TS),ss\_tong\_diem\_giam); for(i=1; i<=n; ++i) t[i].xuat();*

*delete sa; getch();*

*cout << "\n\nSap xep tang theo tong diem - PP Heap Sort" ; sa= new heap\_sort;*

*sa->sapxep( t+1,n,sizeof(TS),ss\_tong\_diem\_tang); for(i=1; i<=n; ++i) t[i].xuat();*

*delete sa; getch();*

*}*

## Chương trình 2

*//CT10-09*

*// Lop co so truu tuong*

*// Lop sort #include "c\_sort.h"*

*int ss\_tang(void \*i1,void \*i2)*

*{*

*return \*((int\*)i1) < \*((int\*)i2);*

*}*

*int ss\_giam(void \*i1,void \*i2)*

*{*

*return \*((int\*)i1) > \*((int\*)i2);*

*}*

*void main()*

*{*

*int i,n;*

*struct time t1,t2;*

*int b[20],a[20], k, tg, sec, hund; n=10;*

*sort \*s[3]; select\_sort ss; quick\_sort qs; heap\_sort hs;*

*s[0]=&ss; s[1]=&qs; s[2]=&hs; clrscr();*

*srand(5000); for(i=1;i<=n;++i) b[i]=rand(); cout<<"\nDay ban dau\n ";*

*for(i=1;i<=n;++i) cout <<b[i]<<" "; cout<<"\n\nCac day tang sap xep theo ";*

*cout << "select\_sort, quick\_sort, heap\_sort\n"; for(k=0; k<3; ++k)*

*{*

*for(i=1;i<=n;++i) a[i]=b[i];*

*s[k]->sapxep (a+1,n,sizeof(int),ss\_tang);*

*//In*

*for(i=1;i<=n;++i) cout <<a[i]<<" "; cout<<"\n";*

*}*

*cout<<"\n\nCac day giam sap xep theo ";*

*cout << "select\_sort, quick\_sort, heap\_sort\n"; for(k=0; k<3; ++k)*

*{*

*for(i=1;i<=n;++i) a[i]=b[i];*

*s[k]->sapxep (a+1,n,sizeof(int),ss\_giam);*

*//In*

*for(i=1;i<=n;++i) cout <<a[i]<<" "; cout << "\n";*

*}*

*getch();*

*}*

**Phụ lục 1**

# Thứ tự ưu tiên của các phép toán

Các phép toán được chia thành 16 nhóm. Các phép toán trong cùng nhóm có mực độ ưu tiên như nhau.

Về trình tự kết hợp thì:

+ Các phép tính của nhóm 2, nhóm 14 và toán tử gán (nhóm 15) kết hợp từ phải sang trái.

+ Các phép toán còn lại kết hợp từ trái qua phải.

## Nhóm một

() Gọi hoàm (Function call)

[] Chỉ số mảng (Array subscript)

-> Chọn gián tiếp một thành phần (indirect component selector)

:: Xác định phạm vi truy nhập (scope access/resolution)

. Chọn trực tiếp một thành phần (direct component selector)

## Nhóm hai

() Gọi hoàm (Function call)

! Phủ định logic (Logical negation -NOT)

~ Lấy phần bù theo bit (Bitwise (1's) complement)

+ Dấu cộng (Unary plus)

- Dấu trừ (Unary minus)

++ Phép tăng một (Preincrement or postincrement)

-- Phép giảm một (Predecrement or postdecrement) & Phép lấy địa chỉ (Address)

* + Truy nhập gián tiếp (Indirection)

sizeof Cho kích thước của toán hạng (returns size of operand, in bytes) new Cấp phát bộ nhớ động (dynamically allocates C++ storage)

delete Giải phóng bộ nhớ (dynamically deallocates C++ storage)

## Nhóm ba

* + Nhân ( Multiply)

/ Chia (Divide)

% Lấy phần dư (Remainder - modulus)

## Nhóm bốn

.\* Gọi gián tiếp tới thành phần từ một biến đối tượng

->\* Gọi gián tiếp tới thành phần từ một con trỏ đối tượng

## Nhóm năm

+ Cộng (Binary plus)

- Trừ (Binary minus)

## Nhóm sáu

<< Dịch trái (Shift left)

>> Dịch phải (Shift right)

## Nhóm bẩy

< Nhỏ hơn (Less than)

<= Nhỏ hơn hoặc bằng (Less than or equal to)

> Lớn hơn (Greater than)

>= Lớn hơn hoặc bằng (Greater than or equal to)

## Nhóm tám

== Bằng (Equal to)

!= Không bằng (Not equal to)

## Nhóm chín

& Phép và theo bit (Bitwise AND)

## Nhóm mười

^ Phép hoặc loại trừ theo bit (Bitwise XOR)

## Nhóm mười một

| Phép hoặc theo bit (Bitwise OR)

## Nhóm mười hai

&& Phép và logic (Logical AND)

## Nhóm mười ba

&& Phép hoặc logic (Logical OR)

## Nhóm mười bốn

?: Toán tử điều kiện (a ? x : y means "if a then x, else y")

## Nhóm mười năm

= Phép gán đơn giản (Simple assignment)

\*= Phép gán sau khi nhân (Assign product)

/= Phép gán sau khi chia (Assign quotient)

%= Phép gán sau khi lấy phần dư (Assign remainder)

+= Phép gán sau khi cộng (Assign sum)

-= Phép gán sau khi trừ (Assign difference)

&= Phép gán sau khi AND theo bit (Assign bitwise AND)

^= Phép gán sau khi XOR theo bit (Assign bitwise XOR)

|= Phép gán sau khi OR theo bit (Assign bitwise OR)

<<= Phép gán sau khi dịch trái (Assign left shift)

>>= Phép gán sau khi dịch phải (Assign right shift)

## Nhóm mười sáu

, Toán tử phẩy dùng để phân cách các phần tử

Tất cả các toán tử nói trên đều có thể định nghĩa chồng trừ các toán tử sau:

. Chọn trực tiếp một thành phần

.\* Gọi gián tiếp tới thành phần từ một biến đối tượng

:: Toán tử xác định phạm vi truy nhập

?: Toán tử điều kiện

**Phụ lục 2**

# Các từ khóa của c++

|  |  |  |  |
| --- | --- | --- | --- |
| asm | double | new | switch |
| auto | else | operator | template |
| break | enum | private | this |
| case | extern | protected | throw |
| catch | float | public | try |
| char | for | register | typedef |
| class | friend | return | union |
| const | goto | short | unsigned |
| continue | if | signed | virtual |
| default | inline | sizeof | void |
| delete | int | static | volatile |
| do | long | struct | while |

## Bảng mã ascii

**Phụ lục 3**

# Bảng mã asskey và giải quyết

Bộ ký tự ASCII gồm 256 ký tự được phân bố như sau:

+ 32 ký tự đầu tiên là các ký tự điều khiển không in được như ký tự Enter (mã 13), ký tự ESC (mã 27).

+ Các mã ASCII 32-47, 58-64, 91-96 và 123-127 là các ký tự đặc biệt như dấu chấm, dấu phẩy, dấu cách, dấu ngoặc, dấu móc, dấu hỏi,...

+ Các mã ASCII 48-57 là 10 chữ số

+ Các mã ASCII 65-90 là các chữ cái hoa từ A đến Z

+ Các mã ASCII 97-122 là các chữ cái thường từ a đến z

***Lưu ý:*** Chữ thường có mã ASCII lớn hơn 32 so với chữ hoa tương ứng. Ví dụ mã ASCII của a là 97 còn mã ASCII của A là 65.

+ Các mã ASCII 128-255 là các ký tự đồ hoạ.

Bảng sau cho mã ASCII của 128 ký tự đầu tiên. Để nhận được các ký tự đồ hoạ (có mã từ 128 đến 255) có thể dùng chương trình sau:

// In các ký tự đồ hoạ lên màn hình #include <stdio.h>

#include <conio.h> main()

{

int i; clrscr();

for (i=128; i<=255; ++i)

printf("%6d%2c",i,i);

}

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| mã | ký tự | mã | ký tự | **Bảng mã ASCII**  mã | ký tự |
| (Số TT) |  | (Số TT) |  | (Số TT) |  |
| 0 | NUL | 26 | SUB | 52 | 4 |
| 1 | SOH | 27 | ESC | 53 | 5 |
| 2 | STX | 28 | FS | 54 | 6 |
| 3 | ETX | 29 | GS | 55 | 7 |
| 4 | EOT | 30 | RS | 56 | 8 |
| 5 | ENQ | 31 | US | 57 | 9 |
| 6 | ACK | 32 | Space | 58 | : |
| 7 | BEL | 33 | ! | 59 | ; |
| 8 | BS | 34 | “ | 60 | < |
| 9 | HT | 35 | # | 61 | = |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 10 | LF | 36 | $ | 62 | > |
| 11 | VT | 37 | % | 63 | ? |
| 12 | FF | 38 | & | 64 | @ |
| 13 | CR | 39 | ‘ | 65 | A |
| 14 | SO | 40 | ( | 66 | B |
| 15 | SI | 41 | ) | 67 | C |
| 16 | DLE | 42 | \* | 68 | D |
| 17 | DC1 | 43 | + | 69 | E |
| 18 | DC2 | 44 | , | 70 | F |
| 19 | DC3 | 45 | - | 71 | G |
| 20 | DC4 | 46 | . | 72 | H |
| 21 | NAK | 47 | / | 73 | I |
| 22 | SYN | 48 | 0 | 74 | J |
| 23 | ETB | 49 | 1 | 75 | K |
| 24 | CAN | 50 | 2 | 76 | L |
| 25 | EM | 51 | 3 | 77 | M |
| 78 | N | 95 | \_ | 112 | p |
| 79 | O | 96 | \* | 113 | q |
| 80 | P | 97 | a | 114 | r |
| 81 | Q | 98 | b | 115 | s |
| 82 | R | 99 | c | 116 | t |
| 83 | S | 100 | d | 117 | u |
| 84 | t | 101 | e | 118 | v |
| 85 | U | 102 | f | 119 | w |
| 86 | V | 103 | g | 120 | x |
| 87 | W | 104 | h | 121 | y |
| 88 | X | 105 | i | 122 | z |
| 89 | Y | 106 | J | 123 | { |
| 90 | Z | 107 | k | 124 | | |
| 91 | [ | 108 | l | 125 | } |
| 92 | \ | 109 | m | 126 | ~ |
| 93 | ] | 110 | n | 127 | DEL |
| 94 | ^ | 111 | o |  |  |

## Bảng mã scan từ bàn phím

Mỗi phím trên bàn phím của IBM PC đều được gán một con số, gọi là mã scan, từ 1 đến 83. IBM PC AT đùng một nhóm mã khác, từ 1 đến 108 các mã này bắt đầu bằng các phím số, các phím chữ, rồi đến các phím chức năng và cuối cùng là các phím cho con trỏ, khi một phím được nhấn thì bộ xử lý của bàn phím gửi cho CPU mã scan tương ứng, khi nó được nhả thì mã scan cộng thêm 80 hex sẽ được gửi tiếp cho CPU.

|  |  |  |  |
| --- | --- | --- | --- |
| **Hex**  1 | **Thập phân**  1 | **Phím của PC**  ESC | **Phím của PC-AT**  Tilde |
| 2-B | 2-11 | 1-9,0 | 1-9,0 |
| C | 12 | trừ, gạch dưới | trừ, gạch dưới |
| D | 13 | =, + | =, + |
| E | 14 | Backspace | \,thanh đứng |
| F | 15 | Tab | Backspace |
| 10 | 16 | Q | Tab |
| 11 | 17 | W | Q |
| 12 | 18 | E | W |
| 13 | 19 | R | E |
| 14 | 20 | T | R |
| 15 | 21 | Y | T |
| 16 | 22 | U | Y |
| 17 | 23 | I | U |
| 18 | 24 | O | I |
| 19 | 25 | P | O |
| 1A | 26 | [ | P |
| 1B | 27 | ] | [ |
| 1C | 28 | Enter | ] |
| 1D | 29 | Ctrl |  |
| 1E | 30 | A | Ctrl |
| 1F | 31 | S | A |
| 20 | 32 | D | S |
| 21 | 33 | F | D |
| 22 | 34 | G | F |
| 23 | 35 | H | G |
| 24 | 36 | J | H |
| 25 | 37 | K | J |
| 26 | 38 | L | K |
| 27 | 39 | Chấm phẩy, : | L |
| 28 | 40 | Nháy | Chấm phẩy,: |
| 29 | 41 | Tidle | Nháy |
| 2A | 42 | Shift | trái |
| 2B | 43 | \, thanh đứng | Enter |
| 2C | 44 | Z | Shift trái |
| 2D | 45 | X |  |
| 2E | 46 | C | Z |
| 2F | 47 | V | X |
| 30 | 48 | B | C |

|  |  |  |  |
| --- | --- | --- | --- |
| 31 | 49 | N | V |
| 32 | 50 | M | B |
| 33 | 51 | Phảy | N |
| 34 | 52 | Chấm | M |
| 35 | 53 | /,? | Phảy |
| 36 | 54 | Shift phải | Chấm |
| 37 | 55 | \*, PrtScr | /, ? |
| 38 | 56 | Alt |  |
| 39 | 57 | Space bar | Shift phải |
| 3A | 58 | Caps Lock | Alt |
| 3B | 59 | F1 |  |
| 3C | 60 | F2 |  |
| 3D | 61 | F3 | Space bar |
| 3E | 62 | F4 |  |
| 3F | 63 | F5 |  |
| 40 | 64 | F6 | Caps Lock |
| 41 | 65 | F7 | F2 |
| 42 | 66 | F8 | F4 |
| 43 | 67 | F9 | F6 |
| 44 | 68 | F10 | F8 |
| 45 | 69 | Num Lock | F10 |
| 46 | 70 | Scroll Lock,Break | F1 |
| 47 | 71 | Home | F3 |
| 48 | 72 | mũi tên lên | F5 |
| 49 | 73 | PgUp | F7 |
| 4A | 74 | Dấu trừ bàn tính | F9 |
| 4B | 75 | Mũi tên trái |  |
| 4C | 76 | 5 | của bàn tính |
| 4D | 77 | Mũi tên phải |  |
| 4F | 79 | End |  |
| 50 | 80 | Mũi tên xuống |  |
| 51 | 81 | PgDn |  |
| 52 | 82 | Ins |  |
| 53 | 83 | Del |  |
| 5A | 90 |  | ESC |
| 5B | 91 |  | Home |
| 5C | 92 |  | Mũi tên trái |
| 5D | 93 |  | End |
| 5F | 95 |  | Num Lock |
| 60 | 96 |  | Mũi tên lên |

|  |  |  |
| --- | --- | --- |
| 61 | 97 | 5 của bàn tính |
| 62 | 98 | Mũi tên xuống |
| 63 | 99 | Ins |
| 64 | 100 | Scroll Lock |
| 65 | 101 | PgUp |
| 66 | 102 | Mũi tên phải |
| 67 | 103 | PgDn |
| 68 | 104 | Del |
| 69 | 105 | Sys |
| 6A | 106 | \*, PrtScr |
| 6B | 107 | - |
| 6C | 108 | + |

**Phụ lục 4**

# Hàm với đối số bất định trong C

Trong các giáo trình C thường chỉ hướng dẫn cách xây dựng hàm với các đối cố định. Mỗi đối cần có một tham số (cùng kiểu với nó) trong lời gọi hàm. Tuy nhiên một vài hàm chuẩn của C lại không như vậy, mà linh hoạt hơn, chẳng khi dùng hàm printf hay scanf thì số tham số mà ta cung cấp cho hàm là không cố định cả về số lượng lẫn kiểu cách. Ví dụ trong câu lệnh:

printf(“\n Tổng = %d “ , 3+4+5) ; có 2 tham số, nhưng trong câu lệnh:

printf(“\n Hà Nội“ ) ; chỉ có một tham số.

Như vậy cần phân biệt các khái niệm sau:

* Đối số cố định được khai báo trong dòng đầu của hàm, nó có tên và kiểu
* Tham số ứng với đối số cố định gọi là tham số cố định
* Đối bất định được khai báo bởi ba dấu chấm: bất định cả về số lượng và kiểu
* Tham số bất định (ứng với đối bất định) là một danh sách giá trị với số lượng và kiểu tuỳ ý (không xác định)

Trong phụ lục này sẽ trình bầy cách xây dựng các hàm với đối số bất định. Công cụ chủ yếu được dùng là con trỏ và danh sách.

## Biến con trỏ

Biến con trỏ (hay con trỏ) dùng để chứa địa chỉ của biến, mảng, hàm, ... Có nhiều kiểu địa chỉ, vì vậy cũng có nhiều kiểu con trỏ. Biến con trỏ được khai báo theo mẫu:

Kiểu \*Tên\_biến\_con\_trỏ ;

## Ví dụ:

float px ; **//** px là con trỏ thực

Các phép toán quan trọng trên con trỏ gồm:

+ Gán địa chỉ một vùng nhớ cho con trỏ (dùng toán tử gán, phép lấy địa chỉ, các hàm cấp phát bộ nhớ)

+ Truy nhập vào vùng nhớ thông qua con trỏ, dùng phép toán:

\*Tên\_con\_trỏ

(Để ý ở đây có 2 vùng nhớ: vùng nhớ của biến con trỏ và vùng nhớ mà địa chỉ đầu của nó chứa trong biến con trỏ)

+ Cộng địa chỉ để con trỏ chứa địa chỉ của phần tử tiếp theo, dùng phép toán:

++ Tên\_con\_trỏ hoặc Tên\_con\_trỏ ++

***Chú ý*** rằng các phép toán trên chỉ có thể thực hiện đối với con trỏ có kiểu.

## Danh sách không cùng kiểu

Dùng con trỏ có kiểu chỉ quản lý được một danh sách giá trị cùng kiểu, ví dụ dẫy số thực, dẫy số nguyên, dẫy các cấu trúc,....

Khi cần quản lý một danh sách các giá trị không cùng kiểu ta phải dùng con trỏ không kiểu (void) khai báo như sau:

void \* Tên\_con\_trỏ ;

Con trỏ void có thể chứa các địa chỉ có kiểu bất kỳ, và dùng để trỏ đến vùng nhớ chứa danh sách cần quản lý. Một chú ý quan trọng là mỗi khi gửi vào hay lấy ra một giá trị từ vùng nhớ, thì tuỳ theo kiểu giá trị mà ta phải dùng phép chuyển kiểu thích hợp đối với con trỏ. Ví dụ sau minh hoạ cách lập một danh sách gồm một số nguyên, một số thực và một chuỗi ký tự. Chúng ta cần một bộ nhớ để chứa số nguyên, số thực và địa chỉ chuỗi và dùng các con trỏ void để quản lý vùng nhớ này.

void \*list **,** \*p **; //** Con trỏ list trỏ tới đầu danh sách

// p dùng để duyệt qua các phần tử của danh sách list=malloc(sizeof(int) + sizeof(float)+ sizeof(char\*) ); p=list;

\*((int\*)p) = 12; // Đưa số nguyên 12 vào danh sách ((int\*)p)++ ; // Chuyển sang phần tử tiếp theo

\*((float\*)p) = 3.14; // Đưa số thực 3.14 vào danh sách ((float\*)p)++ ; // Chuyển sang phần tử tiếp theo

\*((char\*\*)p) = “HA NOI”; // Đưa địa chỉ chuỗi “HA NOI”

// vào danh sách

// Nhận các phần tử trong danh sách p=list; // Về đầu danh sách

int a = \*((int\*)p); // Nhận phần tử thứ nhất ((int\*)p)++ ; // Chuyển sang phần tử tiếp theo float x= \*((float\*)p); // Nhận phần tử thứ hai ((float\*)p)++ ; // Chuyển sang phần tử tiếp theo char \*str = \*((char\*\*)p) ; // Nhận phần tử thứ ba

## Hàm với đối số bất định

+ Đối bất định bao giờ cũng đặt sau cùng và được khai báo bằng dấu ba chấm. Ví dụ ví dụ hàm void f(int n, char \*s, ...) ;

có 2 đối cố định là n, s và đối bất định.

+ Để nhận được các tham số bất định trong lời gọi hàm ta cần lưu ý các điểm sau:

* Các tham số bất định chứa trong một danh sách. Để nhận được địa chỉ đầu danh sách ta dùng một con trỏ void và phép gán sau:

void \*list ; list = ... ;

* Dùng một tham số cố định kiểu chuỗi để quy định số lượng và kiểu của mỗi tham số trong danh sách, ví dụ:

“3i” hiểu là : tham số bất định gồm 3 giá trị int “3f” hiểu là : tham số bất định gồm 3 giá trị float

“fiss” hiểu là có 4 tham số bất định có kiểu lần lượt là float, int, char\*, char\*

Một khi đã biết được địa chỉ đầu danh sách, biết được số lượng và kiểu của mỗi tham số , thì dễ dàng nhận được giá trị các tham số để sử dụng trong thân hàm.

bất định. Hàm dùng để in c

**Ví dụ** sau đây minh hoạ cách xây dựng các hàm với tham số

563 564

ác giá trị

kiểu int, float và char. Hàm có một tham số cố định để cho biết có bao nhiêu giá trị và kiểu các giá trị cần in. Kiểu quy định như sau: i là int, f là float, s là char\*. Tham số có 2 cách viết: lặp (gồm một hằng số nguyên và một chữ cái định kiểu) và liệt kê (một dẫy các chữ cái định kiểu). Ví dụ:

“4s” có nghĩa in 4 chuỗi

“siif” có nghĩa in một chuỗi, 2 giá trị nguyên và một giá trị thực: #include <stdio.h>

#include <ctype.h> #include <string.h> #include <conio.h> #include <stdlib.h> #include <stdarg.h>

void InDanhSachGiaTri(char \*st,...)

{

void \*list ; int gt\_int ; float gt\_float; char \*gt\_str; int n,i ;

char kieu; int lap;

list = ... ; // list tro toi vung nho chua danh sach dia chi cac

// tham so lap = isdigit(st[0]) ; if (lap)

n=st[0] - '0' ;

else

n=strlen(st);

printf("\n n= %d lap = %d",n,lap); getch(); for(i=0;i<n;++i)

{

if(lap) kieu=st[1];

else

kieu = st[i];

printf("\nKieu= %c",kieu); getch(); switch(kieu)

{

case 'i' :

gt\_int = \*((int\*)list); if(!lap)

((int\*)list)++ ;

printf("\nGia tri %d = %d",i,gt\_int); break;

case 'f' :

gt\_float = (float) (\*((double\*)list)); if(!lap)

((double\*)list)++ ;

printf("\nGia tri %d = %0.2f",i,gt\_float); break;

case 's' :

gt\_str = \*((char\*\*)list) ; if(!lap)

((char\*\*)list)++ ;

printf("\nGia tri %d = %s",i,gt\_str);

}

}

}

void main()

{

float x=3.14; int a=123;

char \*tp="HAI PHONG"; InDanhSachGiaTri("4i",a); InDanhSachGiaTri("4s","HA NOI");

InDanhSachGiaTri("ifsssffii", a, x, tp, tp,"QUY NHON",

x, 6.28, a, 246);

InDanhSachGiaTri("4f",6.28); getch();

}

## Hàm không đối và hàm với đối bất định

Nhiều người nghĩ hàm khai báo như sau void f();

là hàm không đối trong C. Trong C++ thì hiểu như thế là đúng, còn trong C thì đó là hàm có đối bất định (hàm không đối trong C khai báo như sau: f(void) ). Do không có đối cố định nào cho biết về số lượng và kiểu của các tham số bất định, nên giải pháp ở đây là dùng các biến toàn bộ. Rõ ràng giải pháp này không không thuận tiện cho người dùng vì phải khai báo đúng tên biến toàn bộ và phải khởi gán giá trị cho nó trước khi gọi hàm. Ví dụ trình bầy một hàm chỉ có đối bất định dùng để tính max và min của các giá trị thực. Các tham số bất định được đưa vào theo trình tự sau: Địa chỉ chứa max, địa chỉ chứa min, các giá trị nguyên cần tính max, min. Chương trình dùng biến toàn bộ N để cho biết số giá trị nguyên cần tính max, min.

int N;

void maxmin()

{

void \*lt = ... ;

float \*max, \*min , tg; int i;

max = \*((float\*\*)lt)++; min = \*((float\*\*)lt)++;

\*max = \*min = (float) \*((double\*)lt)++; for(i=1;i<N;++i)

{

tg= (float) \*((double\*)lt)++; if(tg > \*max) \*max = tg; if(tg < \*min) \*min = tg;

}

}

**phụ lục 5**

# Tóm tắt các hàm của Turbo C theo thứ tự ABC

* 1. \_chmod <io.h>
  2. \_close <io.h>
  3. \_creat <io.h>
  4. \_open <io.h>
  5. abort <process.h>
  6. abs <stdlib.h>
  7. acos <math.h>
  8. arc <graphics.h>
  9. asin <math.h>
  10. atan <math.h>
  11. atan2 <math.h>
  12. atof <ctype.h>
  13. atoi <ctype.h>
  14. atol <ctype.h>
  15. bar <graphics.h>
  16. bar3d <graphics.h>
  17. cabs <math.h>
  18. calloc <alloc.h>
  19. ceil <math.h>
  20. chdir <dir.h>
  21. chmod <io.h>
  22. circle <graphics.h>
  23. cleardevive <graphics.h>
  24. clearviewport <graphics.h>
  25. close <io.h>
  26. clreol <conio.h>
  27. clrscr <conio.h>
  28. coreleft <alloc.h>
  29. cos <math.h>
  30. cosh <math.h>
  31. cprintf <conio.h>
  32. creat <io.h>
  33. cscanf <conio.h>
  34. delay <dos.h>
  35. delline <conio.h>
  36. disable <dos.h>
  37. drawpoly <graphics.h>
  38. ecvt <ctype.h>
  39. ellipse <graphics.h>
  40. enable <dos.h>
  41. exit <process.h>
  42. exp <math.h>
  43. fabs <math.h>
  44. fclose <stdio.h>
  45. fcloseall <stdio.h>
  46. fcvt <ctype.h>
  47. feof <stdio.h>
  48. ferror <stdio.h>
  49. fflush <stdio.h>
  50. fflushall <stdio.h>
  51. fgetc <stdio.h>
  52. fgets <stdio.h>
  53. fillpopy <graphics.h>
  54. findfirst <dir.h>
  55. findnext <dir.h>
  56. floodfill <graphics.h>
  57. floor <math.h>
  58. fmode <math.h>
  59. fopen <stdio.h>
  60. FP\_OFF <dos.h>
  61. FP\_SEG <dos.h>
  62. fprintf <stdio.h>
  63. fprintf <stdio.h>
  64. fputc <stdio.h>
  65. fputs <stdio.h>
  66. fread <stdio.h>
  67. free <alloc.h>
  68. fscanf <stdio.h>
  69. fseek <stdio.h>
  70. fteel <stdio.h>
  71. fwrite <stdio.h>
  72. gcvt <ctype.h>
  73. geninterrupt <dos.h>
  74. getbkcolor <graphics.h>
  75. getc <stdio.h>
  76. getch <conio.h>
  77. getchar <stdio.h>
  78. getche <conio.h>
  79. getcolor <graphics.h>
  80. getcwd <dir.h>
  81. getdate <time.h>
  82. getimage <graphics.h>
  83. getlinesettings <graphics.h>
  84. getmaxcolor <graphics.h>
  85. getmaxx <graphics.h>
  86. getmaxy <graphics.h>
  87. getpalette <graphics.h>
  88. getpixel <graphics.h>
  89. gets <stdio.h>
  90. gettextinfo <conio.h>
  91. gettime <dos.h>
  92. gettime <time.h>
  93. getvect <dos.h>
  94. getviewport <graphics.h>
  95. getw <stdio.h>
  96. gotoxy <conio.h>
  97. gotoxy <conio.h>
  98. grapherrormsg <graphics.h>
  99. graphresult <graphics.h>
  100. imagesize <graphics.h>
  101. initgraph <graphics.h>
  102. int86 <dos.h>
  103. int86x <dos.h>
  104. intdos <dos.h>
  105. intdosx <dos.h>
  106. intr <dos.h>
  107. inxdigit <ctype.h>
  108. isalnum <ctype.h>
  109. isalpha <ctype.h>
  110. iscntrl <ctype.h>
  111. isdigit <ctype.h>
  112. isgraph <ctype.h>
  113. islower <ctype.h>
  114. isprint <ctype.h>
  115. ispunct <ctype.h>
  116. isspace <ctype.h>
  117. isupper <ctype.h>
  118. itoa <ctype.h>
  119. kbhit <conio.h>
  120. keep <dos.h>
  121. labs <stdlib.h>
  122. line <graphics.h>
  123. linerel <graphics.h>
  124. lineto <graphics.h>
  125. log <math.h>
  126. log10 <math.h>
  127. lseek <io.h>
  128. ltoa <ctype.h>
  129. malloc <alloc.h>
  130. memccpy <memory.h> hoặc <string.h>
  131. memchr <memory.h> hoặc <string.h>
  132. memcmp <memory.h> hoặc <string.h>
  133. memcpy <memory.h> hoặc <string.h>
  134. memicmp <memory.h> hoặc <string.h>
  135. memset <memory.h> hoặc <string.h>
  136. MK\_FP <dos.h>
  137. mkdir <dir.h>
  138. movedata <mem.h>
  139. movedata <memory.h> hoặc <string.h>
  140. moveto <graphics.h>
  141. nosound <dos.h>
  142. open <io.h>
  143. outtext <graphics.h>
  144. outtextxy <graphics.h>
  145. peek <dos.h>
  146. peekb <dos.h>
  147. perror <stdio.h>
  148. pieslice <graphics.h>
  149. poke <dos.h>
  150. pokeb <dos.h>
  151. pow <math.h>
  152. printf <stdio.h>
  153. putc <stdio.h>
  154. putch <conio.h>
  155. putchar <stdio.h>
  156. putimage <graphics.h>
  157. putpixel <graphics.h>
  158. puts <stdio.h>
  159. putw <stdio.h>
  160. rand <stdlib.h>
  161. random <stdlib.h>
  162. randomize <stdlib.h> và <time.h>
  163. read <io.h>
  164. realloc <alloc.h>
  165. rectangle <graphics.h>
  166. remove <stdio.h>
  167. rewind <stdio.h>
  168. rmdir <dir.h>
  169. scanf <stdio.h>
  170. segread <dos.h>
  171. setbkcolor <graphics.h>
  172. setcolor <graphics.h>
  173. setdate <time.h>
  174. setfillstyle <graphics.h>
  175. setlinestyle <graphics.h>
  176. setpalette <graphics.h>
  177. settextjustify <graphics.h>
  178. settextstyle <graphics.h>
  179. settime <time.h>
  180. setvect <dos.h>
  181. setviewport <graphics.h>
  182. setwritemode <graphics.h>
  183. sin <math.h>
  184. sinh <math.h>
  185. sleep <dos.h>
  186. sound <dos.h>
  187. sprintf <stdio.h>
  188. sqrt <math.h>
  189. srand <stdlib.h>
  190. strcat <string.h>
  191. strchr <string.h>
  192. strcmp <string.h>
  193. strcmpi <string.h>
  194. strcpy <string.h>
  195. strcspn <string.h>
  196. strdup <string.h>
  197. stricmp <string.h>
  198. strlen <string.h>
  199. strlwr <string.h>
  200. strncat <string.h>
  201. strncmp <string.h>
  202. strncpy <string.h>
  203. strnicmp <string.h>
  204. strnset <string.h>
  205. strpbrk <string.h>
  206. strrchr <string.h>
  207. strrev <string.h>
  208. strset <string.h>
  209. strspn <string.h>
  210. strstr <string.h>
  211. strupr <string.h>
  212. system <process.h>
  213. tan <math.h>
  214. tanh <math.h>
  215. textbackground <conio.h>
  216. textcolor <conio.h>
  217. textheight <graphics.h>
  218. textmode <conio.h>
  219. textwidth <graphics.h>
  220. time <time.h>
  221. tolower <ctype.h>
  222. toupper <ctype.h>
  223. ultoa <ctype.h>
  224. unlink <stdio.h>
  225. wherex <conio.h>
  226. wherey <conio.h>
  227. window <conio.h>
  228. write <io.h>