ĐẠI HỌC THÁI NGUYÊN
TRƯỜNG ĐẠI HỌC KỸ THUẬT CÔNG NGHIỆP

BÁO CÁO TỔNG KẾT
ĐỀ TÀI KHOA HỌC VÀ CÔNG NGHỆ CẤP TRƯỜNG

XÂY DỰNG VIDEO BÀI GIẢNG MÔN LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG

Mã số: T2022-VD38

Chủ nhiệm đề tài: ThS. Trần Thị Ngọc Linh

Thái Nguyên, 10/2023

ĐẠI HỌC THÁI NGUYÊN

TRƯỜNG ĐẠI HỌC KỸ THUẬT CÔNG NGHIỆP

BÁO CÁO TỔNG KẾT

ĐỀ TÀI KHOA HỌC VÀ CÔNG NGHỆ CẤP TRƯỜNG

XÂY DỰNG VIDEO BÀI GIẢNG MÔN LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG

Mã số: T2022-VD38

Xác nhận của tổ chức chủ trì	**Chủ nhiệm đề tài**
KT. HIỆU TRƯỞNG **PHÓ HIỆU TRƯỞNG**	*(Ký tên)*
PGS.TS. Vũ Ngọc Pi	**ThS. Trần Thị Ngọc Linh**

Thái Nguyên, 10/2023

TRƯỜNG ĐẠI HỌC

KỸ THUẬT CÔNG NGHIỆP

Đơn vị: Khoa Điện tử

THÔNG TIN KẾT QUẢ NGHIÊN CỨU

1. Thông tin chung:

- Tên đề tài: Xây dựng video bài giảng môn Lập trình hướng đối tượng
- Mã số: T2022-VD38
- Chủ nhiệm: Trần Thị Ngọc Linh
- Cơ quan chủ trì: Trường ĐH Kỹ thuật Công nghiệp
- Thời gian thực hiện: tháng 04 năm 2022 đến tháng 10 năm 2023

2. Mục tiêu: Xây dựng video bài giảng tương ứng với 45 tiết học. Mỗi tiết học được cô đọng giảng dạy trong 15 phút video để sinh viên có thể nắm được tổng thể nội dung của bài học và chuẩn bị bài trước khi đến lớp.

3. Kết quả nghiên cứu: Xây dựng 24 video bài giảng được bố trí theo các chủ đề chính trong môn học.

4. Sản phẩm:

- Sản phẩm đào tạo: các video bài giảng phục vụ cho môn học Lập trình hướng đối tượng
- Sản phẩm khoa học:
- Sản phẩm ứng dụng:

5. Hiệu quả: Sản phẩm của đề tài được sử dụng cho môn học Lập trình hướng đối tượng, thuộc chương trình đào tạo ngành Kỹ thuật máy tính của Khoa Điện tử. Sinh viên có thể sử dụng các Video này như một nguồn tài liệu chính thức của môn học để nâng cao lượng kiến thức ngoài các tiết học trên lớp. Qua các video này, sinh viên có thể chủ động hơn trong phương pháp tiếp cận kiến thức của môn học bằng cách xem lại các nội dung chưa nắm vững.

6. Khả năng áp dụng và phương thức chuyển giao kết quả nghiên cứu:

- Video bài giảng môn Lập trình hướng đối tượng được áp dụng làm bài giảng, tài liệu tham khảo cho sinh viên chuyên ngành Kỹ thuật máy tính khoa Điện tử Trường Đại học kỹ thuật công nghiệp. Nhằm nâng cao chất lượng đào tạo, đáp ứng nhu cầu của người học trong xu thế phát triển của công nghệ số.

Ngày 10 tháng 10 năm 2023

Cơ quan chủ trì	**Chủ nhiệm đề tài**
KT.HIỆU TRƯỞNG	
PHÓ HIỆU TRƯỞNG	
BỘ GIÁO DỤC VÀ ĐÀO TẠO TRƯỜNG ĐẠI HỌC KỸ THUẬT CÔNG NGHIỆP ĐẠI HỌC THÁI NGUYÊN	
PGS.TS. Vũ Ngọc Pi	**ThS. Trần Thị Ngọc Linh**

1. Đặt vấn đề

1.1. Tổng quan

Cùng với sự phát triển về khoa học công nghệ, nhu cầu của người học để phù hợp với sự phát triển trong giáo dục nhằm tiếp cận và khai thác tốt các nền tảng công nghệ. Bên cạnh hình thức giảng dạy trực tiếp cho người học, các trường cần sử dụng nhiều hơn các hình thức khác như đào tạo online, thiết kế môi trường ảo để người học và người dạy có thể tương tác lẫn nhau và truyền đạt thông tin, tổ chức thực hành tại các phòng thí nghiệm hay phòng mô phỏng ảo nhằm tạo ra tính trực quan, sinh động, đa dạng trong hoạt động dạy và học.

Việc đưa hệ thống E-learning vào hoạt động tại trường Đại học, tạo ra một kênh học tập khác góp phần nâng cao chất lượng đào tạo. Hiện nay việc sử dụng hệ thống E-learnin đã trở thành tự giác đối với hầu hết giảng viên và sinh viên trong Trường vì những lợi ích thiết thực mà hệ thống mang lại.

Video là một phương tiện truyền thông phong phú và mạnh mẽ được sử dụng trong
elearning. Nó có thể trình bày thông tin một cách hấp dẫn và nhất quán. Để sử dụng video hiệu quả như một công cụ được tăng cường trong giáo dục cần xem xét ba yếu tố: cách quản lý nội dung và hình thức của video; làm thế nào để video thu hút tối đa sự tham gia của sinh viên; làm thế nào để thúc đẩy học tập tích cực từ video.

Thực tế cho thấy việc sử dụng video trong quá trình dạy và học đã mang lại những hiệu quả đáng ghi nhận, nó cho thấy sự phù hợp với xu thế phát triển của xã hội, nhu cầu của người học và hiệu quả kinh tế nó mang lại cho nền giáo dục trong thời đại công nghệ số. Nó giúp cho người học có thể tiếp cận kiến thức một cách chủ động, linh hoạt, và tạo ra sự hấp dẫn mới mẻ với người học.

Việc sử dụng các bài giảng video ngày càng phổ biến, các bài giảng thiết kế không chỉ giúp người học tiếp cận thông tin một cách trực quan mà còn có sự tương tác trao đổi hiệu quả. Ngoài ra việc thiết kế bài giảng vô cùng quan trọng vì nó có vai trò tạo động lực, truyền thêm cảm hứng học tập đến người học. Trách nhiệm của người làm công việc thiết kế là phải sáng tạo ra những bài giảng có hình thức thú vị, truyền cảm hứng giúp người học có nhiều động lực hơn.

Ngày nay, việc sử dụng video để tự học, tra cứu nguồn tài liệu,... là nhu cầu cần thiết và thường xuyên. Việc xây dựng video rất có ý nghĩa trong thực tiễn, đáp ứng được nhu cầu cấp bách của người học trong thời đại 4.0. Cách sử dụng nguồn thông tin là bài giảng video làm tài liệu học tập và nghiên cứu này đã mang lại hiệu quả lớn, giúp người học rút ngắn được thời gian học tập, nghiên cứu nhờ sự trực quan và những lợi ích của cách thức này mang lại, góp phần thúc đẩy sự phát triển, nâng cao chất lượng giáo dục

1.2. Mục tiêu xây dựng

Việc mô phỏng các kiến thức trực, quan sinh động là một đặc điểm nổi bật của các bài giảng E-learning và hỗ trợ rất nhiều trong việc ghi nhớ thông tin, giúp ích cho việc học

Lập trình hướng đối tượng là môn học cơ sở ngành bao gồm nhiều kiến thức nền tảng liên quan đến lập trình, là một môn học xương sống của ngành nên đòi hỏi sinh viên phải có được một nền tảng kiến thức tốt để dễ dàng tiếp cận được các môn học có liên quan cũng như vận dụng vào thực tế viết phần mềm sau này. Việc xây dựng 1 bài giảng đáp ứng được các tiêu chí sau là vô cùng cần thiết để người học tiếp thu dễ dàng hơn, có thể nhanh chóng đạt được mục tiêu của môn học và giúp học viên ghi nhớ kiến thức dễ hơn và lâu dài hơn

- Bài giảng được thiết kế logic giúp người học ghi nhớ tốt hơn
- Bài giảng có nội dung kiến thức ngắn gọn, cô đọng dễ hiểu
- Bài giảng có minh họa cụ thể cho từng nội dung
- Bài giảng được thiết kế thú vị giúp cuốn hút người học hơn
- Bài giảng được thiết kế tối ưu giúp tạo sự tương tác nhiều hơn
- Bài giảng được thiết kế tương tác nhiều khuyến khích người học chủ động hơn

2. Thiết kế bài giảng video

2.1. Cơ sở lý thuyết

- Lập trình hướng đối tượng C++ là một ngôn ngữ lập trình mạnh được sử dụng rộng rãi trên thế giới trong nhiều lĩnh vực khác nhau như: Trí tuệ nhân tạo, cơ sở dữ liệu, hệ thống thời gian thực, siêu văn bản đa phương tiên, lập trình song song,... Lập trình hướng đối tượng C++ là môn học quan trọng của cơ sở ngành, là tiền đề cho nhiều môn chuyên ngành quan trọng. Giúp sinh viên tiếp cận với cách thức lập trình hiện nay và có thể vận dụng vào việc phát triển phần mềm và chuyển giao công nghệ.
- Xây dựng bài giảng video môn Lập trình hướng đối tượng C++ dựa trên đề cương chi tiết của môn học và việc thiết kế hợp lý nội dung của từng video nhằm đáp ứng các tiêu chí chuẩn tạo được sự hấp dẫn và dễ hiểu đối với người học từ đó người học dễ dàng vận dụng vào việc xây dựng và phát triển các phần mềm hiện nay

2.2. Mục tiêu thiết kế

- Các video được thiết kế nhằm cung cấp cho người học những kiến thức cơ bản, ngắn gọn, cốt lõi của môn học đảm bảo các tiêu chí: Dễ học, dễ hiểu, dễ vận dụng trong các bài toán thực tế. Với tiêu chí như vậy bất kỳ đối tượng nào cũng có thể thực hiện code một cách dễ dàng
- Mỗi video tương đương với một chương là một chủ đề xuyên suốt của môn học, ngoài giới thiệu ngắn gọn cô đọng về cơ sở lý thuyết, ở mỗi vấn đề quan trọng của mỗi chương đều minh họa bằng các video code trực tiếp trên nền tảng phần mềm DEV C++ giúp ngời học có thể quan sát một cách trực quan nhất quá trình thực

hiện code các câu lệnh cũng như cách giải quyết vấn đề từ đó khắc sâu kiến thức về môn học và có thể vận dụng ngay đối với các bài toán thực tế

2.3. Đề cương chi tiết môn học Lập trình hướng đối tượng

Tuần	Nội dung
	Chương 1: Tổng quan về lập trình hướng đối tượng
	Các nội dung giảng dạy chính trên lớp (4 tiết)
	Nội dung lý thuyết
1	- Tổng quan về lập trình hướng đối tượng - Kỹ thuật lập trình hướng đối tượng - Lớp – Đối tượng - Con trỏ đối tượng
	Chương 2: Lớp
2-4	*Các nội dung giảng dạy chính trên lớp (12 tiết)* *Nội dung lý thuyết* - Lớp - Hàm bạn - Hàm tạo - Hàm hủy
	Chương 3. Toán tử tải bội
5-6	*Các nội dung giảng dạy chính trên lớp (8 tiết)* *Nội dung lý thuyết* - Toán tử tải bội - Nạp chồng toán tử
	Chương 4. Kế thừa
7-9	*Các nội dung giảng dạy chính trên lớp (10 tiết)* *Nội dung lý thuyết* - Kế thừa - Hàm ảo - Lớp cơ sở ảo - Đa hình
	Chương 5. Khuân hình

10	*Các nội dung giảng dạy chính trên lớp (3 tiết)* *Nội dung lý thuyết* - Khuân hình hàm
	- Khuân hinh lớp

2.4. Nội dung chi tiết

Chương 1: **Tổng quan về lập trình hướng đối tượng**

- Các Video cung cấp những kiến thức cơ bản, nền tảng, các ứng dụng,...của lập trình hướng đối tượng C++ từ đó có cái nhìn tổng quát về ngôn ngữ lập trình mạnh mẽ hiện nay

Chương 2: **Lớp**

- Các Video cung cấp kiến thức nền tảng quan trọng về Lớp, đối tượng,.... cách thức xây dựng dữ liệu, cách thức tổ chức bài toán từ đó xây dựng phần mềm theo hướng đối tượng

Chương 3: **Toán tử tải bội**

- Các Video cung cấp công cụ, cách thức xử lý bài toán theo cách đặc thù hướng đối tượng. Thể hiện được sức mạnh, ưu điểm của lập trình hướng đối tượng

Chương 4: **Kế thừa**

- Các Video trình bày nội dung kiến thức đặc trưng của lập trình hướng đối tượng, cách thức giải quyết bài toán một cách tối ưu của phương pháp lập trình này. Qua đây người học sẽ nhận thấy ưu điểm rõ nét của việc tái sử dụng code của kế thừa,... Đây chính là khả năng ưu việt của lập trình hướng đối tượng so với các phương pháp lập trình khác. Từ đó thấy được sự cần thiết của lập trình hướng đối tượng trong xu thế phát triển phần mềm hiện nay.

Chương 5: **Khuân hình**

- Các Video trình bày sức mạnh vượt trội của lập trình hướng đối tượng khi sử dụng khuân hình trong việc xây dựng bài toán, tổ chức dữ liệu và tính hiệu quả của việc sử dụng khuân hình nhằm đơn giản nhưng tối ưu của loại hình lập trình hiện đại giúp nâng cao chất lượng phần mềm cũng như lợi ích vượt trội của nó

3. Tài liệu tham khảo

[1]. Phạm Văn Ất; C++ và lập trình hướng đối tượng, NXB Khoa học và kỹ thuật; 2000

[2]. Lê Đăng Hưng, Tạ Tuấn Anh, Nguyễn Hữu Đức ; Nguyễn Thanh Thuỷ (ch.b.); Lập trình hướng đối tượng với C++; NXB Khoa học kỹ thuật, 2003

[3]. Tạ Tuấn Anh, Nguyễn Quang Huy, Nguyễn Hữu Đức; Bài tập lập trình hướng đối tượng với C++, NXB Khoa học kỹ thuật, 2001

TRƯỜNG ĐẠI HỌC KỸ THUẬT CÔNG NGHIỆP
KHOA ĐIỆN TỬ

LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG C++

THS. TRẦN THỊ NGỌC LINH

MỤC TIÊU HỌC TẬP

- Định hướng phương pháp xây dựng phần mềm cho một hệ thống cơ bản:
  - Hướng dẫn cách lập trình theo hướng đối tượng
  - Cung cấp kiến thức cơ bản về cách thức và các kỹ thuật nâng cao khi lập trình theo hướng đối tượng
  - Áp dụng để xây dựng được phần mềm đơn giản

NỘI DUNG

1. TỔNG QUAN VỀ LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG
2. LỚP
3. TOÁN TỬ TẢI BỘI
4. KẾ THỪA
5. KHUÂN HÌNH

CHƯƠNG 1:

TỔNG QUAN VỀ LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG

NỘI DUNG

1. Tổng quan về lập trình hướng đối tượng
2. Kỹ thuật lập trình hướng đối tượng
3. Lớp – Đối tượng
4. Con trỏ đối tượng

1.1. LẬP TRÌNH TRUYỀN THỐNG

1. Lập trình tuyến tính: Là phương pháp lập trình không sử dụng hàm, dữ liệu đều dùng chung và tất cả các biến đều ở dạng toàn cục.

- **Ưu điểm:** Dễ viết rất nhanh, vì không phải thông qua giai đoạn gọi hàm, gọi đối tượng.
- **Nhược điểm:**
  - Khó bảo trì và quản lý dòng code.
  - Không sử dụng được lại mã lệnh
  - Chỉ viết được các chương trình nhỏ.

1.1. LẬP TRÌNH TRUYỀN THỐNG

2. Lập trình cấu trúc: Lấy các hàm làm nền tảng cơ bản để xây dựng chương trình, chương trình sẽ được phân nhỏ thành các hàm và mỗi hàm sẽ có chức năng riêng biệt.

- **Ưu điểm:**
  - Chương trình được tổ chức khoa học hơn nên dễ quản lý và bảo trì
  - Có thể thực hiện được nhiều chương trình lớn hơn
- **Nhược điểm:**
  - Cách tiếp cận đôi khi không phù hợp với thực tế
  - Khó mô tả được các hoạt động của thế giới thực
  - Bảo mật kém

1.1. LẬP TRÌNH TRUYỀN THỐNG

3. Lập trình theo modul: lấy ý tưởng đóng hộp, các hàm có chức năng giống nhau sẽ được gom lại thành một modun độc lập, khi cần sử dụng module nào sẽ gọi tới module đó nên một chương trình có thể có nhiều module chứ không riêng lẻ độc lập.

- **Ưu điểm:**
  - Xây dựng được các chương trình lớn.
  - Code rõ ràng, dễ quản lý, bảo trì và nâng cấp.
  - Phân theo khối nên mạch lạc
- **Nhược điểm:**
  - Dữ liệu không có sự gắn kết với nhau
  - Dữ liệu khởi tạo không bị hủy sau khi gọi hàm

LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG LÀ GÌ?

- Lập trình hướng đối tượng (OOP) là một phương pháp lập trình sử dụng các đối tượng tương tác để giải quyết những nhiệm vụ phức tạp trong lập trình. Mỗi đối tượng sẽ có những thuộc tính và hành vi khác nhau, giúp cho việc giải quyết bài toán trở nên hiệu quả hơn(dễ phát triển, dễ bảo trì,..). Phương pháp này cho phép tạo ra các đối tượng và trừu tượng hóa chúng. Để sử dụng phương pháp này cần hiểu rõ 4 tính chất:
  - **Tính đóng gói**
  - **Tính kế thừa**
  - **Tính đa hình**
  - **Tính trừu tượng.**

9

NHƯ VẬY:

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

10

1.2. KỸ THUẬT LẬP TRÌNH OOP

1.2.1 Đối tượng

1.2.2 Thuộc tính & Phương thức

1.2.3 Lớp & Lớp con

1.2.4 Lớp trừu tượng

1.2.5 Truyền thông điệp

1.2.6 Tính trừu tượng

1.2.7 Tính đóng gói

1.2.8 Tính Kế thừa

1.2.9 Tính đa hình

11

1.3.1. ĐỐI TƯỢNG

Là sự thể hiện của một lớp. Một đối tượng là sự đóng gói 2 thành phần:

- Thuộc tính (dữ liệu)
- Hành vi, thao tác(phương thức, hàm)

12

1.2.1. ĐỐI TƯỢNG

Ví dụ:

- bàn, ghế, con chó, con mèo
- Quyển sách, cái bút,
- ....

13

1.2.2 THUỘC TÍNH & PHƯƠNG THỨC

- Thuộc tính bao gồm: Hằng, biến, Tham số nội tại
- Kiểu thuộc tính:
  - Kiểu cổ điển
  - Kiểu do người dùng định nghĩa
- Phương thức(hàm thành viên)
  - Các hàm nội tại của đối tượng
  - Có kiểu trả về

Ví dụ: SINHVIEN (MSSV, họ tên, địa chỉ, giới tính, nhập, xuất thông tin)

14

1.2.3 LỚP (CLASS) & LỚP CON (SUBCLASS)

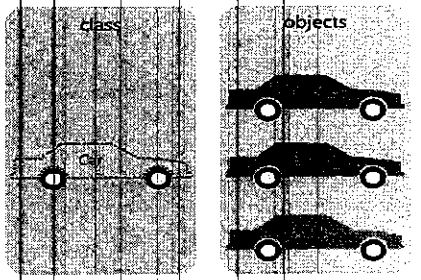
- **Lớp:**
  - Là tập hợp các đối tượng có cùng thuộc tính và hành vi
  - Là bản bản mẫu mô tả một cấu trúc dữ liệu gồm:
    - Các thành phần dữ liệu(thuộc tính)
    - Các phương thức(hàm)
- **Lớp con:** Là lớp thông thường, có thêm tính chất kế thừa đặc tính của lớp khác

15

VÍ DỤ

class Car

- Thuộc tính: tên xe, loại xe, hãng sản xuất, màu sắc
- Phương thức(hàm): đi, dừng,...



16

VÍ DỤ

- class Người(họ tên, địa chỉ, giới tính, nhập thông tin, xuất thông tin)
- ➢ 2 lớp Giảng viên, sinhvien được kế thừa từ lớp người và có thêm các thuộc tính
- class Giangvien(MSGV, khoa, tuổi, tính lương)
- class sinhvien(MSSV, lớp, khoa, quê quán, TínhĐTB)
- ➢ Khi đó: Lớp Giangvien, sinhvien được gọi là **lớp con** của lớp Người.

  Lớp: Người được gọi là lớp cha

17

1.2.4 LỚP TRỪU TƯỢNG

- Là lớp mà nó không thể trở thành một lớp thực tế nào
- Được thiết kế nhằm tạo ra lớp có đặc tính tổng quát
- Bản thân nó chưa có ý nghĩa nên chưa thể viết mã cho đối tượng

Ví dụ:

- Lớp hình phẳng
- Lớp động vật

18

VÍ DỤ

- Để xây dựng 1 ứng dụng cho bệnh viện ta cần nhập thông tin của bệnh nhân: Họ tên, địa chỉ, giới tính, email, điện thoại, chiều cao, cân nặng, màu da, màu tóc, tình trạng bệnh,….

➢ Tuy nhiên không phải bất kỳ thuộc tính nào cũng cần thiết và hữu ích cho việc xây dựng và phát triển vì vậy ta cần lựa chọn và sử dụng những thuộc tính hữu ích, liên quan cho việc xây dựng và phát triển ứng dụng đó. Quá trình này được gọi là tạo 1 lớp trừu tượng

19

1.2.5 TRUYỀN THÔNG ĐIỆP

- **Thông điệp:** Là phương tiện để đối tượng này chuyển yêu cầu tới đối tượng khác.
- Một thông điệp bao gồm:
  - Handle của đối tượng đích (đối tượng chủ)
  - Tên phương thức cần thực hiện
  - Các thông tin cần thiết khác (tham số)
- Hệ thống yêu cầu đối tượng thực hiện phương thức:
  - Gửi thông báo và tham số cho đối tượng
  - Kiểm tra tính hợp lệ của thông báo
  - Gọi thực hiện hàm tương ứng phương thức

20

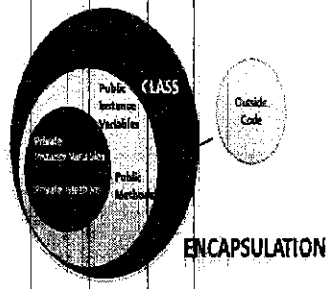
1.2.6 TÍNH TRỪU TƯỢNG

- Trừu tượng có nghĩa là tổng quát hóa một cái gì đó, không cần chú ý chi tiết bên trong mà người ta vẫn hiểu mỗi khi nghe về nó.
- Ví dụ: Khi chạy xe tay ga có hành động: tăng ga để tăng tốc, thì chức năng tăng ga đại diện cho trừu tượng. Ta chỉ cần biết là tăng ga thì xe tăng tốc, không cần biết bên trong nó làm thế nào.
- Trong lập trình OOP, tính trừu tượng nghĩa là chọn ra các thuộc tính, phương thức của đối tượng cần cho việc giải quyết bài toán. Vì một đối tượng có rất nhiều thuộc tính phương thức, nhưng với bài toán cụ thể không nhất thiết phải chọn tất cả.

21

1.3.7 TÍNH ĐÓNG GÓI

- Việc ràng buộc giữa code và data với nhau tạo thành một khối duy nhất được gọi là đóng gói.
- Đóng gói cũng được sử dụng để bảo vệ trạng thái bên trong của một đối tượng. Bởi việc ẩn giấu các biến biểu diễn trạng thái của đối tượng. Việc chỉnh sửa đối tượng được thực hiện, xác nhận thông qua các phương thức. Hơn nữa, việc ẩn giấu các biến thì các lớp sẽ không chia sẻ thông tin với nhau được. Điều này làm giảm số lượng khớp nối có thể có trong một ứng dụng.
- Ví dụ: viên thuốc con nhộng được đóng gói với nhiều loại thuốc bên trong.



22

1.2.7 TÍNH ĐÓNG GÓI

- Là kỹ thuật giúp che giấu đi những thông tin bên trong đối tượng bằng cách sử dụng phạm vi truy cập private cho các thuộc tính. Muốn giao tiếp hay lấy các thông tin của đối tượng thì phải thông qua các phạm vi truy cập là public. Bằng cách hạn chế quyền truy cập, đóng gói sẽ hạn chế được các lỗi khi phát triển chương trình.
- Ví dụ: không thể thấy được các bản chất của một người (tính cách, sở thích,...), những thứ ta biết đều thông qua các hành động của người đó thể hiện ra ngoài, nhưng các thông tin này chưa chắc đã là thật

23

CÁC LỢI ÍCH:

- Hạn chế được các truy xuất không hợp lệ tới các thuộc tính của đối tượng.
- Giúp cho trạng thái của các đối tượng luôn đúng.
  - Giúp ẩn đi những thông tin không cần thiết về đối tượng.
  - Cho phép thay đổi cấu trúc bên trong lớp mà không ảnh hưởng tới lớp khác.

➢ **Lưu ý**: mục đích chính của tính đóng gói là để hạn chế các lỗi khi phát triển chương trình chứ không phải là bảo mật hay che giấu thông tin

24

1.2.8 KẾ THỪA

- Kế thừa cho phép xây dựng một lớp mới(lớp con) dựa trên các định nghĩa của lớp đã có(lớp cha). Nghĩa là lớp cha có thể chia sẻ dữ liệu và phương thức cho các lớp con. Các lớp con không phải định nghĩa lại, ngoài ra lớp con có thể mở rộng các thành phần kế thừa và bổ sung thêm các thành phần mới.
- Ví dụ:
  - Lớp điện thoại: Lớp cha
  - Lớp Android, lớp Iphone: Lớp con

25

VÍ DỤ

- **Ví dụ:** Lớp Người, Giảng viên, Sinh viên
  - Lớp Người(họ tên, giới tính, ngày sinh)
  - Lớp Giảng viên(họ tên, giới tính, ngày sinh, MGV, Khoa)
  - Lớp Sinh viên(họ tên, giới tính, ngày sinh, MSV, Điểm)
  - Khi đó lớp Người: lớp cha
  - Lớp Giảng viên, Sinh viên: lớp con
  - Quan hệ kế thừa: Giảng viên: Kế thừa lớp Người, Sinh viên: Kế thừa lớp Người

26

1.2.9 TÍNH ĐA HÌNH (POLYMORPHISM)

- Tính đa hình là một hành động có thể được thực hiện bằng nhiều cách khác nhau. Đây là một tính chất có thể nói là chứa đựng hầu hết sức mạnh của lập trình hướng đối tượng.
- Hiểu một cách đơn giản hơn: Đa hình là khái niệm mà hai hoặc nhiều lớp có những phương thức giống nhau nhưng có thể thực thi theo những cách thức khác nhau.

27

VÍ DỤ

- Có 2 con vật: chó, mèo. Cả 2 con vật này đều thuộc lớp động vật. Nhưng khi ta bảo cả 2 con vật này kêu thì con chó sẽ kêu gâu gâu, con mèo sẽ kêu meo meo. Vậy trong ví dụ này chó, mèo là các đối tượng. 2 con vật cùng hiểu hành động là kêu nhưng mỗi con sẽ kêu theo các cách khác nhau.
- Trong chiến đấu có: Lục quân, không quân, hải quân. Khi cần cho 3 đội quân cùng đánh ta sẽ phải ra lệnh: Lục quân(): đánh, Hải quân(): đánh, Không quân(): đánh

Nếu dung đa hình:

- Lục quân(): đánh
- Hải quân(): đánh
- Không quân(): đánh

28

TẠI SAO SỬ DỤNG TÍNH ĐA HÌNH

- Trong ví dụ trên: Việc tạo ra các tên khác nhau cho một hàm có chức năng giống nhau thì sẽ làm code không nhất quán, rườm rà, cồng kềnh.
- Tính đa hình cho phép chúng ta tạo ra một hàm duy nhất nhưng sẽ thực thi khác nhau với từng lớp khác nhau.

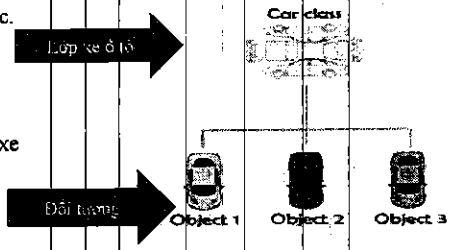
➢ Tính đa hình (polymorphism) trong cho phép chúng ta tạo ra những mã code nhất quán, chương trình cũng sẽ hoạt động tốt

29

1.3. LỚP – ĐỐI TƯỢNG

1.3.1. Lớp

- Để quản lý một hoặc nhiều đối tượng, trong lập trình hướng đối tượng người ta tạo ra một khung gọi là lớp. Trong lớp sẽ có các biến ta gọi là các thuộc tính, và có thể chứa các hàm ta gọi nó là phương thức.
- Ví dụ: Lớp xe ô tô
- Thuộc tính:
  - Tốc độ xe, Nhãn hiệu xe, Giá xe
- Phương thức:
  - Đi, chạy, dừng, đỗ,



30

CÁC THÀNH PHẦN CỦA LỚP

- Thuộc tính:
  - Một thuộc tính của một lớp là một trạng thái chung được đặt tên của lớp đó. Ví dụ: Lớp Ôtô có các thuộc tính: màu sắc, vận tốc, hãng sản xuất,…
  - Mỗi đối tượng của lớp có các giá trị của thuộc tính khác nhau. Ví dụ: một chiếc Ôtô bạn đang sử dụng có thể có màu đen, vận tốc 60 km/h.
- Phương thức:
  - Xác định các hoạt động chung mà tất cả các đối tượng của lớp thực hiện được.
  - Ví dụ: Lớp Ôtô có các phương thức: tăng tốc độ, giảm tốc độ,…

31

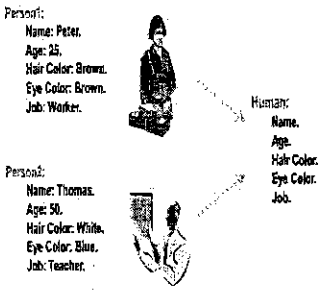
NHƯ VẬY:

- Lớp là một mô tả trừu tượng của nhóm các đối tượng có cùng bản chất, là một khuân mẫu mà từ đó các đối tượng được tạo ra. Mỗi đối tượng sẽ là thể hiện cụ thể của những mô tả trừu tượng đó
- Một Lớp sẽ gồm: dữ liệu (**thuộc tính**) và phương thức (**hàm thành phần**).
- Lớp là một kiểu dữ kiểu do người lập trình tự định nghĩa

32

1.3.2 ĐỐI TƯỢNG (OBJECT)

- Đối tượng được xem là một thực thể trừu tượng của một sự vật trong thế giới thực. Thực thể này sẽ được con người mô hình hóa, lưu giữ những đặc điểm, hành vi, từ đó tạo ra các thuộc tính (**Attribute**) và phương thức (**Method**) của đối tượng.



33

1.3.2 ĐỐI TƯỢNG (OBJECT)

- Trong C++, Đối tượng (Object) là một thực thể trong thế giới thực,

Ví dụ, ghế, xe hơi, bút, điện thoại, máy tính xách tay, v.v.

- Nói cách khác, đối tượng là một thực thể có trạng thái và hành vi. Ở đây, trạng thái có nghĩa là dữ liệu và hành vi có nghĩa là chức năng.
- Đối tượng là một thể hiện của một lớp. Tất cả các thành viên của lớp có thể được truy cập thông qua đối tượng.

34

1.3.2 ĐỐI TƯỢNG (OBJECT)

- Là khái niệm trừu tượng phản ánh các thực thể trong thế giới thực
  - Có thể là một thực thể vật lý
  - Có thể là một khái niệm trừu tượng
- Là sự thể hiện của một lớp
- Một đối tượng là sự đóng gói 2 thành phần:
  - Thuộc tính hay dữ liệu
  - Các hành vi, thao tác, hay phương thức
  - Ví dụ: Điện thoại, Bàn, ghế,...

35

1.4. CON TRỎ ĐỐI TƯỢNG

- **Con trỏ đối tượng (pointer to object)** dùng để chứa địa chỉ của biến đối tượng
- Con trỏ đối tượng là con trỏ trỏ đến địa chỉ của một đối tượng có kiểu lớp. Các thao tác liên quan đến con trỏ đối tượng bao gồm:
  - Khai báo con trỏ đối tượng
  - Cấp phát bộ nhớ cho con trỏ đối tượng
  - Sử dụng con trỏ đối tượng
  - Giải phóng bộ nhớ cho con trỏ đối tượng

36

1.4. CON TRỎ ĐỐI TƯỢNG

A. Khai báo

- **Cú pháp:**

  - **<Tên lớp> *<Tên con trỏ đối tượng>;**

- Ví dụ: Khai báo một con trỏ đối tượng có kiểu của lớp Car, ta khai báo như sau:

  Car *myCar;

- Khi đó, myCar là một con trỏ đối tượng có kiểu lớp Car

B. CẤP PHÁT BỘ NHỚ CHO CON TRỎ ĐỐI TƯỢNG

- Con trỏ đối tượng cũng cần phải cấp phát bộ nhớ hoặc trỏ vào một địa chỉ của một đối tượng lớp, xác định trước khi được sử dụng. Cấp phát bộ nhớ cho con trỏ đối tượng cũng bằng thao tác new:

  <Tên con trỏ đối tượng> = new <Tên lớp>([<Các đối số>]);

Ví dụ, nếu lớp Car có hai hàm khởi tạo như sau:

- class Car{ public:
- Car(); Car(int, string, float); };

thì ta có thể cấp phát bộ nhớ theo hai cách, tương ứng với hai hàm khởi tạo của lớp

B. CẤP PHÁT BỘ NHỚ CHO CON TRỎ ĐỐI TƯỢNG

myCar = new Car(); // Khởi tạo không tham số

myCar = new Car(100, "Ford", 3000); // Khởi tạo đủ tham số

Lưu ý:

- Các đối số truyền phải tương ứng với ít nhất một trong các hàm khởi tạo của lớp.
- Khi sử dụng hàm khởi tạo không có tham số, ta vẫn phải sử dụng cặp ngoặc đơn "()" trong thao tác new.
- Khi lớp không có một hàm khởi tạo tường minh nào, sẽ dùng hàm khởi tạo ngầm định của C++ và cú pháp tương tự như sử dụng hàm khởi tạo tường minh không có tham số

C. SỬ DỤNG CON TRỎ ĐỐI TƯỢNG

Con trỏ đối tượng được sử dụng qua các thao tác:

- Trỏ đến địa chỉ của một đối tượng cùng lớp
- Truy nhập đến các phương thức của lớp

D. GIẢI PHÓNG BỘ NHỚ CHO CON TRỎ ĐỐI TƯỢNG

- Con trỏ đối tượng được giải phóng thông qua thao tác delete:

delete <Tên con trỏ đối tượng>;

```
Car *ptrCar = new Car();// Khai báo và cấp phát bộ nhớ
... // Sử dụng con trỏ ptrCar
delete ptrCar; // Giải phóng bộ nhớ.
```

41

1.5. CÁC BƯỚC THIẾT KẾ CHƯƠNG TRÌNH OOP

❖Các bước chính:

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

42

1.6. ƯU ĐIỂM

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

43

1.7. ỨNG DỤNG CỦA OOP

- **Lĩnh vực chính:**
  - Cơ sở dữ liệu hướng đối tượng
  - Hệ thống thời gian thực
  - Phát triển phần mềm
  - Hệ siêu văn bản, đa phương tiện
  - Trí tuệ nhân tạo
  - Lập trình song song, mạng nơron …

44

Chương 2: Lớp

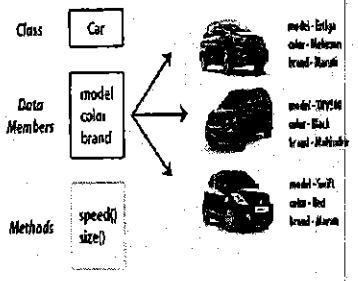
NỘI DUNG

1. Lớp
2. Hàm bạn
3. Hàm tạo
4. Hàm hủy

2.1. LỚP

Lớp (class) là kiểu người dùng tự định nghĩa (user-defined).

Một lớp được sử dụng để xác định form của một đối tượng và nó kết nối sự biểu diễn dữ liệu và các phương thức để thao tác các dữ liệu đó vào trong một package. Dữ liệu và hàm bên trong một lớp được gọi là các thành viên của lớp đó.



2.1. LỚP

2.1. Khái niệm

Lớp:

- Là tập hợp các đối tượng có cùng thuộc tính và hành vi
- Là bản bản mẫu mô tả một cấu trúc dữ liệu gồm:
  - Các thành phần dữ liệu(thuộc tính)
  - Các phương thức(hàm)

KHAI BÁO

```
class <tên_lớp>
{     private:      [Khai báo các thuộc tính]
                    [Định nghĩa các hàm thành phần (phương thức)]
      public :      [Khai báo các thuộc tính]
                    [Định nghĩa các hàm thành phần (phương thức)]
      protected: [Khai báo các thuộc tính]
                    [Định nghĩa các hàm thành phần (phương thức)]
};
```

53

TRONG ĐÓ

- Class: từ khóa
- <tên_lớp>: do người sử dụng tự đặt và tuân theo quy tắc đặt tên
- Thuộc tính: các thuộc tính mô tả đối tượng
- Phương thức: các hàm thành phần
- Thuộc tính, phương thức: là các thành phần của lớp được quy định phạm vi sử dụng (private, public, protecdted). Ngầm định là private

54

TRONG ĐÓ

- Private: phạm vi dùng riêng. Các đối tượng của các lớp khác không truy nhập được các thành phần này.
- protected: phạm vi được bảo vệ. Qui định loại đối tượng nào được truy nhập đến các thành phần được bảo vệ
- public: phạm vi dung chung. Các đối tượng của các lớp khác đều có thể truy nhập vào các thành phần này

55

THUỘC TÍNH CỦA LỚP

Thuộc tính của lớp là thành phần chứa dữ liệu, đặc trưng cho các tính chất của lớp.
Thuộc tính của lớp được khai báo theo cú pháp sau:

<Kiểu dữ liệu> <Tên thuộc tính>;

Trong đó:

- Kiểu dữ liệu: có thể là các kiểu dữ liệu cơ bản của C++, cũng có thể là các kiểu dữ liệu phức tạp do người dùng tự định nghĩa
- Tên thuộc tính: là tên thuộc tính của lớp, có tính chất như một biến thông thường. Tên thuộc tính phải tuân theo quy tắc đặt tên biến của C++.

56

KHAI BÁO

- **Lưu ý:**
  - ✓Không được khởi tạo giá trị ban đầu cho các thuộc tính ngay trong lớp. Vì các thuộc tính chỉ có giá trị khi nó gắn với một đối tượng cụ thể, là một thể hiện (biến) của lớp.

```
class Car{
private: int speed; // đúng
int weight = 500; // lỗi };
```

57

PHƯƠNG THỨC CỦA LỚP

<Kiểu trả về> <Tên phương thức>([<Các tham số>]);

- **Trong đó:**
- Kiểu trả về: Có thể là các kiểu dữ liệu cơ bản, cũng có thể là kiểu do người dùng định nghĩa
- Tên phương thức: do người dùng tự đặt, tuân theo quy tắc đặt tên biến
- Các tham số: Các tham số đầu vào của phương thức, được biểu diễn bằng kiểu dữ liệu tương ứng. Các tham số được phân cách bởi dấu phẩy

58

LƯU Ý:

- ✓ Khả năng truy nhập phương thức từ bên ngoài phụ thuộc vào phương thức được khai báo trong phạm vi của từ khóa nào: private, protected hay public.
- ✓Nếu phương thức được cài đặt ngay trong lớp thì các tham số phải tường minh, nghĩa là mỗi tham số phải được biểu diễn bằng một cặp <Kiểu dữ liệu> <Tên tham số> như khi cài đặt chi tiết một hàm tự do trong chương trình.
- ✓Thông thường, chỉ các phương thức ngắn (trên một dòng) là nên cài đặt ngay trong lớp. Còn lại nên cài đặt các phương thức bên ngoài lớp để chương trình được sáng sủa, rõ ràng và dễ theo dõi.

59

VÍ DỤ

✓Xây dựng cấu trúc mô tả lớp Giảng viên gồm các thuộc tính: MSGV, tên GV, Giới tính, đơn vị, năm sinh, chức vụ, lương. Để lập danh sách chào mừng ngày 8.3 Hãy in thông tin nữ Giảng viên khoa điện tử

→ Lớp Giangvien

Thuộc tính: MSGV, tên GV, Giới tính, đơn vị, năm sinh, chức vụ, lương

Phương thức: nhập, in

Trong đó: MSGV, tên GV, Giới tính, đơn vị, chức vụ: **kiểu xâu ký tự**

năm sinh, lương: kiểu số thực

60

KHAI BÁO

```
class Giangvien {
private:
        string MSGV, tên GV, Giới tính, đơn vị, chức vụ;
        float  năm sinh, lương ;
        public:
        void nhap();
        void in();
        };
```

61

VÍ DỤ

→ Lớp XEOTO

Thuộc tính: Loại xe, hãng sản xuất, màu sắc, tốc độ, động cơ

Phương thức: dừng, di chuyển, tăng tốc

LỚP XE OTO	
THUỘC TÍNH	Loại xe Hãng sản xuất Màu sắc Tốc độ Động cơ
PHƯƠNG THỨC	Dừng Di chuyển Tăng tốc

62

VÍ DỤ

Xây dựng cấu trúc dữ liệu mô tả sinh viên:

- Dữ liệu: MSSV, họ tên, gt, diachi, lop, khoa, diem1, diem 2, diem3, dtb;
- Phương thức: nhập, tính đtb, in

→ Lớp sinh viên

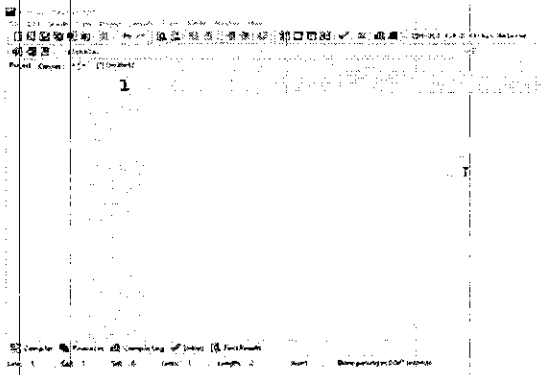
63

VÍ DỤ

```
class sinhvien {
private:
      string mssv, hoten, gioitinh, diachi, lop, khoa;
      float diem1, diem2, diem3, dtb;
      public:
      void nhap();
      void in();
      void tinhdiem();
      };
```

64

CHƯƠNG TRÌNH



VÍ DỤ

Viết chương trình xây dựng một **lớp Vector** để mô tả các đối tượng vector trong kh... gian **n chiều** và thực hiện các công việc sau: **nhập tọa độ, xuất tọa độ, cộng, trừ hai vector, nhân vô hướng hai vector**.

Cơ sở lý thuyết:

Cho 2 vector a có n chiều, **a = (a1, a2, a3, a4, a5, …, an)**

và vector b có n chiều: **b = (b1, b2, b3, b4, b5, …, bn)**

Các phép tính:

KQ a + b sẽ là *một vector c* = **(a1 + b1, a2 + b2, a3 + b3, a4 + b4, a5 + b5, …, an + bn)**

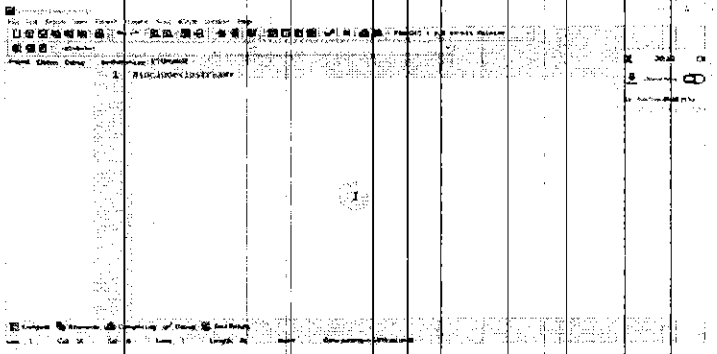
KQ: a - b sẽ là *một vector c* = **(a1 - b1, a2 - b2, a3 - b3, a4 - b4, a5 + b5, …, an - bn)**

KQ: a.b sẽ là *một số cụ thể* = **a1*b1 + a2*b2 + a3*b3 + a4*b4 + a5*b5 + … + an*bn)**

PHÂN TÍCH BÀI TOÁN

```
class Vector
    private:
        int sochieu;
        int toado[100];
    public:
        void Input()
        void Output()
        Vector Add(Vector B)
        Vector Subtr(Vector B)
        int Multi(Vector B) //Nhan vo huong
```


CHƯƠNG TRÌNH



CÁC LOẠI PHẠM VI TRUY CẬP

Phạm vi truy cập là cách mà người lập trình quy định về quyền được truy xuất đến các thành phần của lớp. Trong C++ có 3 loại phạm vi chính là: **private, protected, public**

Phạm Vi Truy Cập	Ý Nghĩa
public	Không hạn chế. Thành phần có thuộc tính này có thể được truy cập ở bất kì vị trí nào.
private	Thành phần có thuộc tính này sẽ chỉ được truy cập từ bên trong lớp. Bên ngoài lớp hay trong lớp dẫn xuất sẽ không thể truy cập được.
protected	Mở rộng hơn private một chút. Thành phần có thuộc tính này sẽ có thể truy cập ở trong nội bộ lớp và trong lớp dẫn xuất (lớp dẫn xuất sẽ được trình bày trong bài Tính Kế Thừa)

HÀM THÀNH VIÊN TRONG LỚP

Trong C++ có 2 cách để xác định hàm thành viên:

•Xác định bên trong lớp

•Xác định bên ngoài lớp

(Để xác định một hàm thành viên bên ngoài lớp ta phải sử dụng toán tử scope ::kèm theo đó là tên lớp và tên phương thức.)

70

CÁC PHƯƠNG THỨC(HÀM)

Các phương thức khởi tạo: Constructor

- Các phương thức truy vấn: Queries
- Các phương thức cập nhập: Updates
- Các phương thức hủy: Destructor

71

VÍ DỤ

- Xây dựng 1 class có tên sinh viên bao gồm các thuộc tính: MSSV, họ tên, giới tính, địa chỉ, lớp, khoa, điểm 1,điểm 2, điểm 3, dtb.
- Yêu cầu:
  - Nhập thông tin cho danh sách các sinh viên
  - Tính điểm trung bình cho danh sách các sinh viên

72

VÍ DỤ

- class sinh viên
- Thuộc tính:
  - string: MSSV, họ tên, giới tính, địa chỉ, lớp, khoa;
  - float: điểm 1,điểm 2, điểm 3, dtb;
- Phương thức:
- Nhập(), in(), tinhdtb();
- Sinhvien a[100];

73

CHƯƠNG TRÌNH

```
24
25      }
26      void output(){
27          cout<<" \t + mssv: "<<mssv<<endl;
28          cout<<" \t + hoten: "<<hoten<<endl;
29          cout<<" \t + gt: "<<gt<<endl;
30          cout<<" \t + lop: "<<lop<<endl;
31          cout<<" \t + khoa: "<<khoa<<endl;
32          cout<<" \t + diem 1: "<<diem1<<endl;
33          cout<<" \t + diem 2: "<<diem2<<endl;
34          cout<<" \t + diem3: "<<diem3<<endl;
35          cout<<" \t + DTB: "<<dtb;
36              }
37
38  };
```

74

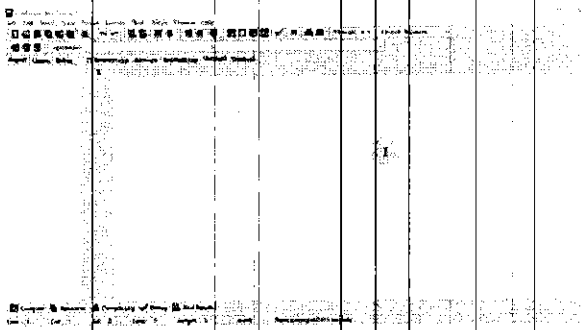
VÍ DỤ

Viết chương trình xây dựng một lớp có tên là SVTNUT bao gồm các thuộc tính: MSSV, Ho_Ten, Dia_Chi, Mail, Gioi_Tinh, Nam_Sinh, Lop, Khoa;

Hãy nhập thông tin của danh sách sinh viên và in ra màn hình thông tin của danh sách sinh viên vừa nhập

75

VÍ DỤ



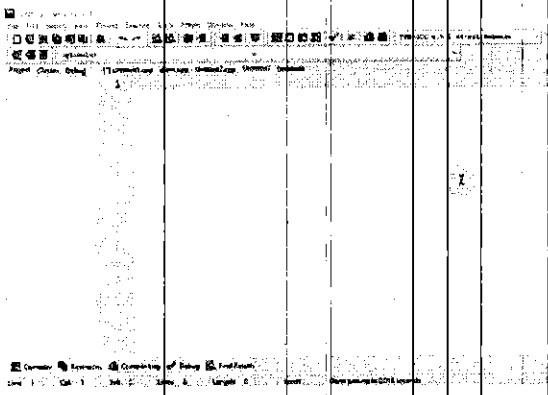
76

PHÂN TÍCH BÀI TOÁN

```
class SVTNUT
private:
        string MSSV, Ho_Ten, Dia_Chi, Mail, Gioi_Tinh, Nam_Sinh, Lop, Khoa;
public:
                void Nhapttsv();
                void Xuatttsv();
```

77

CHƯƠNG TRÌNH



78

2.2. HÀM BẠN(FRIEND FUNCTION)

- Một trong những tính chất quan trọng của OOP là che giấu dữ liệu, nghĩa là hàm không phải thành viên sẽ không thể truy xuất dữ liệu private hoặc protected của một đối tượng.
- Nhưng, đôi khi sự giới hạn này sẽ khiến lập trình viên phải viết đoạn mã nguồn dài và phức tạp. Vì thế, có một cơ chế dựng sẵn trong lập trình C++ cho phép truy xuất dữ liệu private hoặc protected từ các hàm không phải thành viên.
- Điều này được thực hiện thông qua hàm bạn hoặc lớp bạn.

79

2.2. HÀM BẠN(FRIEND FUNCTION)

- Một hàm được định nghĩa là hàm bạn sẽ có thể truy xuất dữ liệu private và protected của một lớp.
- Trình biên dịch sẽ biết một hàm là hàm bạn thông qua từ khóa **friend**.
- Để truy xuất dữ liệu, ta khai báo hàm bạn: hàm bạn được đặt bên trong thân của lớp (có thể nằm ở bất cứ đâu bên trong lớp) bắt đầu với từ khóa friend.

80

VÍ DỤ

```
using namespace std;
 class TamGiac{
     private: int a,b,c;
     public:
     TamGiac() { a = 1; b = 1; c = 1; }
     friend bool kiemTraTG(TamGiac tg){…….}; };
     int main() {
     TamGiac t(3,4,5);
     cout<<kiemTraTG(t); cout<<t; return 0; }
```

85

TÍNH CHẤT CỦA HÀM FRIEND

- Hàm không nằm trong phạm vi của lớp mà nó đã được khai báo là friend.
- Không thể được gọi bằng cách sử dụng đối tượng vì nó không nằm trong phạm vi của lớp đó.
- Có thể được gọi như một hàm bình thường mà không cần sử dụng đối tượng.
- Không thể truy cập trực tiếp vào tên thành viên mà phải sử dụng tên đối tượng và dấu chấm toán tử với tên thành viên.
- Có thể được khai báo trong phần private hoặc public.

86

VÍ DỤ

```
#include <iostream>
 using namespace std;
 class Box {
private:
   int length;
 public:
   Box() : length(0) {}
   friend int printLength(Box);
//ham friend
};
int printLength(Box b) {
   b.length += 10;
   return b.length;
}

int main() {
   Box b;
   cout << "Chieu dai cua box: " <<
printLength(b) << endl;
   return 0;
}
```

87

VÍ DỤ

Tính tổng 2 phân số và in kết quả ra màn hình:

Class phân số

private: int tu, mau

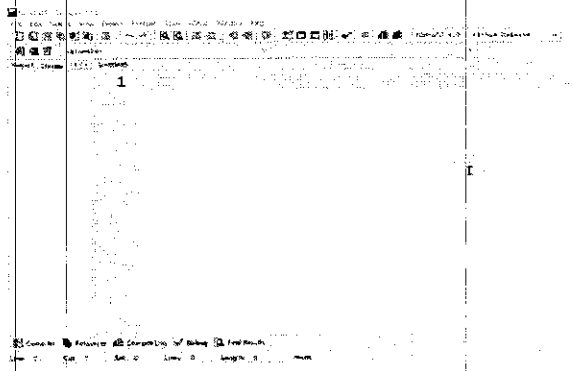
Hàm thành viên:

Public: nhập(), tìm UCLN(), in()

hàm friend: cộng 2 phân số

88

CHƯƠNG TRÌNH



2.3. HÀM TẠO (CONSTRUCTOR)

- Trong C++, **hàm tạo** (constructor) là một phương thức đặc biệt được gọi tự động tại thời điểm đối tượng được tạo. Mục đích của hàm xây dựng là để khởi tạo các thành viên dữ liệu của đối tượng. Hàm xây dựng phải cùng tên với tên lớp và không có bất cứ kiểu gì trả về kể cả kiểu void

2.3.HÀM TẠO (CONSTRUCTOR)

- Là một kiểu hàm thành viên đặc biệt, được gọi tự động khi một đối tượng được tạo. Trong C ++, hàm tạo có tên trùng với tên của lớp và nó không có kiểu trả về.
- **Cú pháp:**

```
class <Tên lớp>{
  public:
   <Tên lớp>([<Các tham số>]); // Khai báo hàm khởi tạo
};
```


VÍ DỤ

```
class HocSinh {
  int mshs;
  string ten;
  public:
    HocSinh();
    HocSinh(int m) {
      mshs = m;      }
    HocSinh(string t) {
      ten = t;
    }
    HocSinh(int m, string t) {
      mshs = m;
      ten = t;
    } };
```

VÍ DỤ

```
class sinh_vien {
  public:
    sinh_vien() // Hàm tạo
{     // Đoạn mã   } };
```

93

2.3.1. ĐẶC ĐIỂM CỦA HÀM TẠO

- Tên hàm khởi tạo trùng với tên của lớp.
- Không có kiểu dữ liệu trả về (kể cả void).
- Phải được khai báo với phạm vi truy cập là public.
- Hàm khởi tạo có thể có đối số hoặc không có đối số.
- Trong một lớp có thể có nhiều hàm khởi tạo (cùng tên nhưng khác đối số).

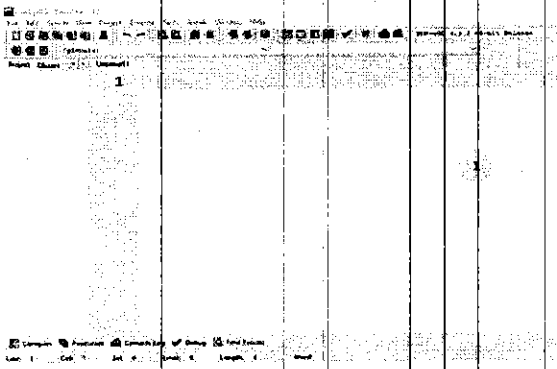
94

2.3.2. HÀM TẠO MẶC ĐỊNH

- Là hàm xây dựng không có tham số.
- Nó sẽ tự động được gọi tại thời điểm đối tượng được tạo.
- Nếu lớp không có hàm tạo nào thì mặc nhiên chương trình sẽ tạo cho lớp đó một hàm tạo mặc định.

95

VÍ DỤ: HÀM TẠO MẶC ĐỊNH



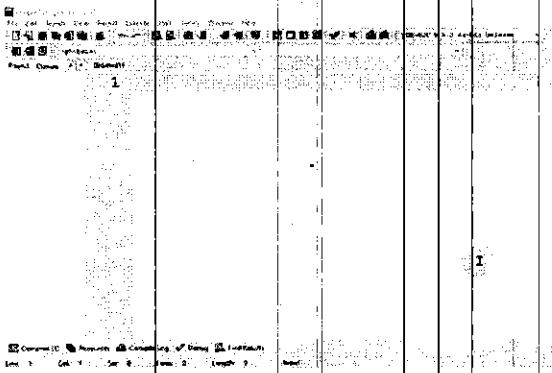
96

2.3.3. HÀM TẠO CÓ THAM SỐ

- Hàm tạo có tham số được gọi là một phương thức khởi tạo chứa tham số (Thường dùng để khởi tạo các dữ liệu thành viên)
- Ví dụ: Ta có hàm khởi tạo sinh_vien(int a). Giá trị tham số được dung để khởi tạo giá trị ID cho biến ID_sv. Khi tạo 1 đối tượng của lớp sinh_vien, ta sẽ truyền giá trị cho hàm tạo.
- Ví dụ: sinh_vien sv(1);
- Với hàm khởi tạo gán giá trị cho biến ID_sv, ta có thể in ra được giá trị đã gán.

97

CHƯƠNG TRÌNH VÍ DỤ



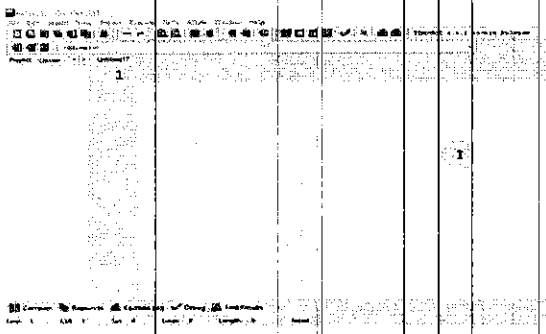
98

2.3.4. HÀM TẠO SAO CHÉP

- Hàm tạo sao chép (*Copy Constructor*) trong C++ là một hàm tạo được sử dụng để khai báo và khởi tạo một đối tượng từ một đối tượng khác.
- Hàm tạo sao chép mặc nhiên: Nếu chúng ta không định nghĩa hàm tạo sao chép thì mặc nhiên trình biên dịch sẽ tự động tạo cho chúng ta một hàm tạo sao chép mặc nhiên.
- Hàm tạo sao chép do người dùng định nghĩa: Đây là hàm tạo do người dùng tự định nghĩa để sao chép từ một đối tượng đã có sẵn.

99

CHƯƠNG TRÌNH VÍ DỤ HÀM TẠO SAO CHÉP



100

2.4. HÀM HỦY(DESTRUCTOR)

- Là một loại hàm thành viên đặc biệt khác của class, được thực thi khi một đối tượng của class đó bị hủy. Nếu các hàm constructors (hàm khởi tạo) được thiết kế để khởi tạo một class, thì các hàm destructors (hàm hủy) được thiết kế để hỗ trợ việc dọn dẹp bộ nhớ.
- ***Cú pháp: ~ tên lớp();***

101

ĐẶC ĐIỂM

- Trong 1 lớp chỉ có một hàm hủy
- Không có các tham số
- Không có kiểu dữ liệu trả về.
- Nếu ta không định nghĩa hàm hủy thì trình biên dịch sẽ tự tạo ra một hàm hủy mặc định và thực hiện nó khi kết thúc chương trình.
- Không được phép khai báo là static
- Được khai báo trong phạm vi public của một lớp
- Không có tính kế thừa

102

Chương 3: TOÁN TỬ TẢI BỘI

103

NỘI DUNG

1. Toán tử tải bội
2. Nạp chồng toán tử

104

KHI ĐÓ TA PHẢI VIẾT 2 HÀM

```
int max1(int a, int b)
 { if (a > b)
return a;
 else return b; }
float max2(float a, float b)
{ if (a > b)
 return a;
else return b; }
```

```
int main() {
 cout << "int max = " <<
max1(4, 5) << endl;
cout << "float max = " <<
max2(4.4, 5.5) << endl; return
0; }
```


3.2. NẠP CHỒNG TOÁN TỬ

- Để tính tổng 2 số nguyên a, b ta thường làm như sau:

  int a = 5;

  int b = 4;

  int c = a + b; // = 9

  Nếu để cộng 2 phân số?

  PhanSo ps1(1, 2); s

  PhanSo ps2(2, 3);

  // Làm sao để có thể cộng hai phân số?

  PhanSo ketQua = ps1 + ps2;

  sử dụng nạp chồng toán tử

CƠ CHẾ HOẠT ĐỘNG

Về bản chất, việc thực hiện các toán tử cũng tương đương với việc gọi hàm

ví dụ

PhanSo a(1, 2);

PhanSo b(2, 3);

PhanSo ketQua = a + b;

//Tương đương với

//PhanSo ketQua = a.cong(b);

3.2. NẠP CHỒNG TOÁN TỬ

- Như vậy, ta sẽ có nhiều hàm với các tên gọi khác nhau. Việc sử dụng tên như vậy sẽ gây bất lợi cho người lập trình khi gọi hàm.

-> Nạp chồng hàm ra đời để giải quyết vấn đề trên.

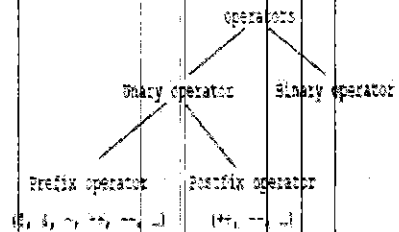
- Kỹ thuật này cho phép sử dụng cùng một tên gọi cho các hàm "giống nhau" (có cùng mục đích). Nhưng khác nhau về kiểu dữ liệu tham số hoặc số lượng tham số.

3.2. NẠP CHỒNG TOÁN TỬ

- Nạp chồng toán tử (Operator Overloading) được dùng để định nghĩa toán tử có sẵn trong c++ phục vụ cho dữ liệu riêng. Giả sử có lớp PhanSo và có các phương thức tính toán như Cong, Tru, Nhan, Chia.
- Nếu gặp một biểu thức phức tạp, số lượng phép tính nhiều thì việc sử dụng các phương thức trên khá khó khăn và có thể gây rối cho người lập trình. Vì thế ta sẽ nạp chồng lại các toán tử để có thể tạo một cái nhìn trực quan vào code, giảm thiểu các lỗi sai không đáng có.

CÁC LOẠI TOÁN TỬ

- C++ chỉ cho phép người dùng overloading lại các toán tử có sẵn trong c++
- Một toán tử có thể được định nghĩa cho nhiều kiểu dữ liệu khác nhau.



TOÁN TỬ ĐƠN(TOÁN TỬ 1 NGÔI)

- Toán tử đơn là toán tử một ngôi (unary operator), có thể được dùng làm toán tử trước (prefix operator) và toán tử sau (postfix operator). Ví dụ phép tăng (++) hay phép giảm (--)
- **Ví dụ:**
  - prefix operator: ++i;
  - postfix operator: i++;

TOÁN TỬ ĐÔI(TOÁN TỬ HAI NGÔI)

Toán tử đôi là toán tử có 2 ngôi (binary operator).

Ví dụ:

A+B,

A*B,

hay toán tử chỉ mục […] cũng là toán tử đôi.

CÚ PHÁP

<Kiểu_trả_về> operator <toán tử> (danh_sách_đối_số)

{ //Định nghĩa toán tử }

Trong đó:

– Kiểu_trả_về là kiểu dữ liệu của hàm nạp chồng toán tử.

– <toán tử> là tên toán tử cần nạp chồng (+, -, *, /,…)

– operator <toán tử> <danh_sách_đối_số) gọi là hàm nạp chồng toán tử. Nó có thể là hàm thành phần hoặc là hàm bạn của lớp (class) nhưng không thể là hàm tĩnh.

LƯU Ý:

- Nếu toán tử tài bội là hàm thành phần thì: không có tham số cho toán tử một ngôi và một tham số cho toán tử hai ngôi. Cũng giống như hàm thành phần thông thường, hàm thành phần toán tử có đối đầu tiên (không tường minh) là con trỏ this
- Nếu toán tử tài bội là hàm bạn thì: có một tham số cho toán tử một ngôi và hai tham số cho toán tử hai ngôi.

HẠN CHẾ CỦA NẠP CHỒNG TOÁN TỬ

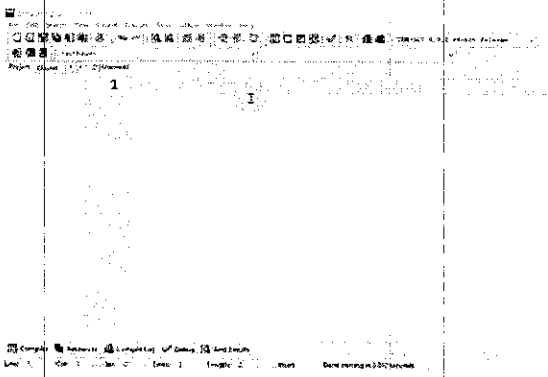
- Không thể tạo toán tử mới
- Không thể kết hợp các toán tử theo cách mà trước đó không được định nghĩa
- Không thay đổi được thứ tự ưu tiên toán tử
- Không thể tạo cú pháp mới cho toán tử
- Không thể định nghĩa lại một đinh nghĩa đã có của một toán tử

VÍ DỤ

Viết chương trình thực hiện nạp chồng toán tử +, - của 2 số phức

- **Phân tích bài toán:**
- Xây dựng class sp
  - Thuộc tính: thực, ảo
  - Phương thức: sử dụng hàm tạo để khởi gán giá trị cho số phức, hàm output để hiển thị số phức kết quả, 2 hàm bạn kết hợp nạp chồng toán tử đối với phép + và – hai số phức

CHƯƠNG TRÌNH



121

VÍ DỤ

Viết chương trình C++ thực hiện các yêu cầu:

- Xây dựng Class PS gồm các thuộc tính:
  + Tử số
  + Mẫu số
- Thực hiện: 4 phép tính đơn giản (+, -, *, /) của 2 phân số có kiểu class PS và in kết quả lên màn hình

122

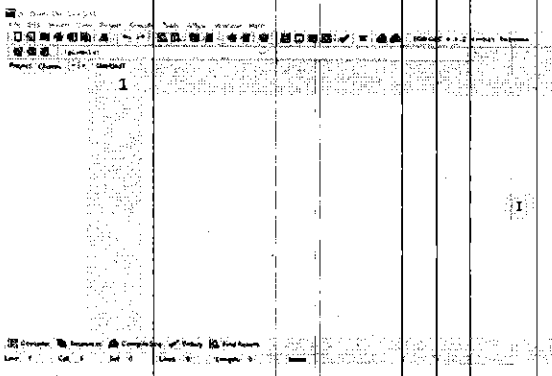
PHÂN TÍCH BÀI TOÁN

Xây dựng class PS

- Thuộc tính: tuso, mauso
- Phương thức: input(), GCD() tìm UCLN, ouput(), operator +, -,*,/

123

CHƯƠNG TRÌNH



124

CHƯƠNG 4: KẾ THỪA

NỘI DUNG

1.Khái niệm

2.Đơn kế thừa

3.Đa kế thừa

4.Hàm ảo

5.Lớp cơ sở ảo

6.Đa hình

KẾ THỪA LÀ GÌ?

- Kế thừa là một trong các tính chất đặc trưng của lập trình hướng đối tượng, bên cạnh tính đóng gói (encapsulation), che giấu thông tin (hiding information), tính đa hình (polymorphism) và tính trừu tượng (abstraction)
- Kế thừa (inheritance) là một tính chất đặc trưng của lập trình hướng đối tượng. Nó có nghĩa là một class thừa hưởng lại tất cả các thuộc tính, phương thức của class mà nó kế thừa.
- Class kế thừa từ một class khác gọi là lớp con (child class hay subclass) hay lớp dẫn xuất (derived class). Class được lớp khác kế thừa được gọi là lớp cha (parent class hay superclass) hay lớp cơ sở (base class).

4.1. KẾ THỪA LÀ GÌ?

- Ví dụ như bạn có một class con người, có các thuộc tính cơ bản như họ tên, ngày sinh, quê quán, class sinh viên kế thừa từ class con người. Vậy, class sinh viên sẽ có các thuộc tính họ tên, ngày sinh, quê quán từ class con người mà không cần phải khai báo. Class con người sẽ là lớp cha và class sinh viên là lớp con.
- Ngoài các thuộc tính của class cha, class con còn có thể có thêm các thuộc tính, phương thức của riêng mình. Ví dụ như sinh viên thì có thêm các thuộc tính như MSSV, tên trường, chuyên ngành…

KẾ THỪA

- Đơn kế thừa
- Đa kế thừa

4.2. ĐƠN KẾ THỪA

- Là một lớp chỉ được kế thừa từ đúng một lớp khác. Hay nói cách khác, lớp con chỉ có duy nhất một lớp cha.

Cú pháp:

class <tên lớp con>:[phạm vi dữ liệu]<tên lớp cha>

{// nội dung của lớp con }

Trong đó:

- Class: từ khóa
- <tên lớp con>, <tên lớp cha>: tên do người sử dụng tự đặt, tuân theo quy tắc đặt tên trong C++
- Phạm vi dữ liệu: public, protected

VÍ DỤ

- xây dựng class CHUHO gồm các thuộc tính: mã chủ hộ, tên chủ hộ, năm sinh, giới tính, địa chỉ, nghề nghiệp
- class NHANKHAU kế thừa class CHU HO có thêm các thuộc tính: tên nhân khẩu, giới tính, năm sinh, mối quan hệ

Yêu cầu:

- Nhập thông tin cho danh sách nhân khẩu trên
- In ra danh sách những chủ hộ có nhân khẩu là sinh viên
- Tính tổng số nhân khẩu của phường phan đình phùng
- In ra danh sách cácchủ hộ là nam >60t
- Cho biết chủ hộ nào có tuổi lớn nhất

PHÂN TÍCH BÀI TOÁN

Xây dựng 2 lớp

1. Class ChuHo

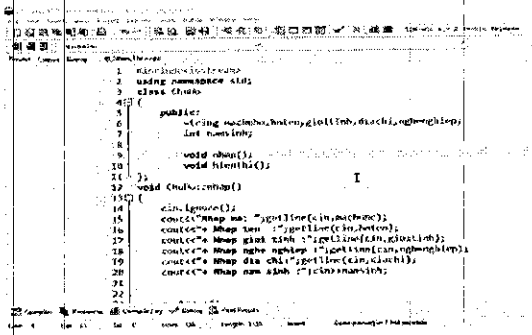
- Thuộc tính: mach, hoten, gioitinh, diachi, nghenghiep, namsinh;
- Phương thức: void nhap(); void hienthi();

2. class NhanKhau:public ChuHo

- Thuộc tính: ChuHo sv[100]; char moiquanhe[50];
- Phương thức: void nhap(); void hienthi(); void nnsinhvien(); void tong(); void nam60(); void lonnhat();

3. Thiết kế menu để thực hiện các thao tác tương ứng với yêu cầu của bài toán

CHƯƠNG TRÌNH



VÍ DỤ

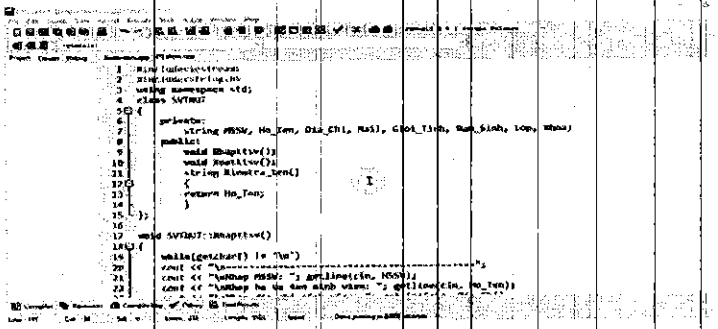
- xây dựng lớp "SVTNUT" gồm các thuộc tính: Mã sinh viên, Họ tên, Địa chỉ, Mail, Giới tính, Năm sinh, Lớp, Khoa
- Xây dựng lớp"DIEM" kế thừa của lớp "SVTNUT" và có thêm các thuộc tính: Diem1, Diem2, Diem3, DiemTB.
- Yêu cầu:

1. Nhập vào danh sách sinh viên
2. Sắp xếp danh sách sinh viên theo thứ tự từ điển(a-z)
3. Sắp xếp danh sách sinh viên theo thự từ điểmTB giảm dần
4. In ra màn hình danh sách sinh viên có điểm TB cao nhất
5. In ra màn hình danh sách sinh viên phải học lại. Biết sinh viên phải học lại là sinh viên có điểm của 1 môn bất kỳ dưới 4.0

PHÂN TÍCH BÀI TOÁN

```
class SVTNUT
private: string MSSV, Ho_Ten, Dia_Chi, Mail, Gioi_Tinh, Nam_Sinh, Lop, Khoa;
        public:
                void Nhapttsv();
                void Xuatttsv();
                string Kiemtra_ten()
class diemsvtnut: public SVTNUT
        private:        float Diem1,Diem2,Diem3, DiemTB;
        public: void Nhap();
                float TinhdiemTB();
                void Xuat();
                void Sap_xep_ten();
                void Sap_xep_diem();
```

CHƯƠNG TRÌNH



VÍ DỤ

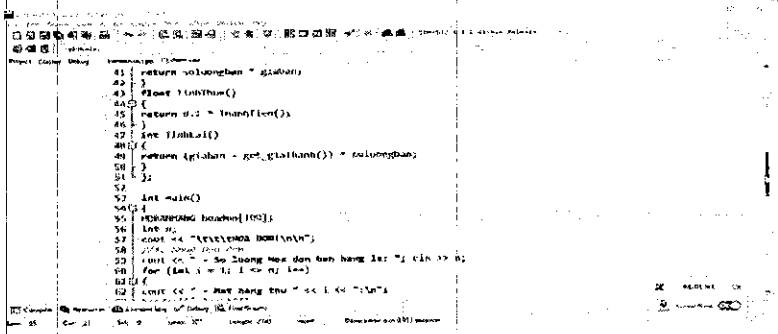
Viết chương trình thực hiện các công việc sau: Xây dựng lớp cơ sở MATHANG gồm các thuộc tính: Tên hàng, năm sản xuất, giá thành. Phương thức: Nhập, xuất thông tin

- Xây dựng lớp HDBANHANG ***kế thừa*** từ lớp MATHANG có thêm các thuộc tính: Số lượng bán, giá bán. Phương thức: Nhập, xuất thông tin, tính thành tiền (=số lượng * giá bán), tính thuế (=10% thành tiền), tính lãi (chênh lệch giá * số lượng bán)
- Chương trình chính thực hiện các yêu cầu sau:
  - Nhập danh sách N hoá đơn bán hàng
  - ***Sắp xếp*** danh sách các mặt hàng có tiền lãi giảm dần
  - Hiện ra màn hình danh sách gồm: số thứ tự, tên hàng, giá thành, số lượng bán, giá bán, thành tiền, thuế và tiền lãi.
  - Tính tổng tiền của các hoá đơn bán hàng

PHÂN TÍCH BÀI TOÁN

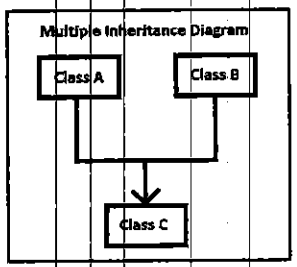
```
class MATHANG
    private:
        string tenhang;
        int NSX;
        float   giathanh;
    public:     void Input(); void Output();
    class HDBANHANG: public MATHANG
    private:
        int SLB,GB;
    public:     void Input(); void Output();  float thanhtien();  float tinhthue();
float tinhlai();
```


VÍ DỤ



4.3. ĐA KẾ THỪA

- Là một lớp có thể kế thừa từ nhiều hơn một lớp khác. Nghĩa là một lớp con được kế thừa từ nhiều hơn một lớp cơ sở.



CÚ PHÁP

class <tên lớp con>: [phạm vi dữ liệu] <tên lớp cha1>, [phạm vi dữ liệu] <tên lớp cha2>,..

{// nội dung của lớp con }

Trong đó:

- Class: là từ khóa
- tên lớp con, tên lớp cha1, tên lớp cha 2,..: là tên do người sử dụng tự đặt
- Phạm vi dữ liệu: private, public, protected

145

4.3. ĐA KẾ THỪA

Một lớp con sẽ kế thừa tất cả các phương thức của lớp cơ sở, ngoại trừ:

- Constructor, destructor và copy constructor của lớp cơ sở.
- Overloaded operator (toán tử nạp chồng) của lớp cơ sở.
- Hàm friend của lớp cơ sở.

146

TRUY NHẬP CÁC THÀNH PHẦN TRONG LỚP DẪN XUẤT

- Các thành phần của lớp dẫn xuất (derived class):
  - Các thành phần khai báo trong lớp dẫn xuất
  - Các thành phần kế thừa được từ lớp cơ sở.
- Các đối tượng của lớp dẫn xuất có thể truy cập các thành phần của nó thông qua toán tử chấm "."

147

VÍ DỤ

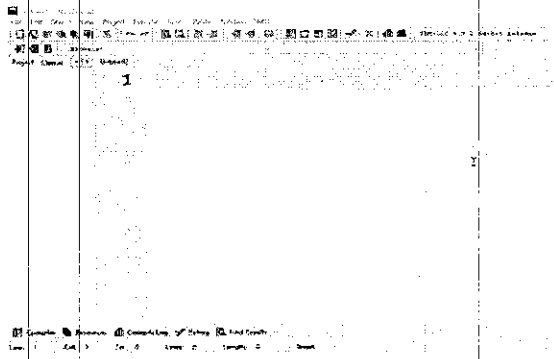
Sử dụng đa kế thừa viết chương trình tính chi phí phải trả sau khi sơn diện tích hình chữ nhật. Biết đơn giá là 300.000/ m*m;

Phân tích bài toán: ta xây dựng 3 class

1. Class Hinh: (lớp cơ sở)
   - Thuộc tính: chiều rộng, chiều dài
   - Phương thức: set_chieurong(), set_chieudai()
2. Class Chiphi gồm phương thức tính chi phí phải trả(lớp cơ sở)
3. Class HCN: kế thừa 2 class Hinh và Chiphi gồm phương thức tính dientich

148

CHƯƠNG TRÌNH



4.4. HÀM ẢO

- Hàm ảo giống như một hàm tải bội, nhưng kiểu dữ liệu, số lượng và kiểu dữ liệu của các tham số của hàm ảo ở cả hai lớp cơ sở và dẫn xuất đều như nhau.
- Hàm ảo là một hàm thành viên trong lớp cơ sở mà lớp dẫn xuất khi kế thừa cần **phải định nghĩa lại**(nghĩa là thành phần của một lớp, được khai báo trong lớp cơ sở rồi nhưng lại được khai báo lại trong lớp dẫn xuất với từ khoá "virtual" ở đằng trước.).
- Hàm ảo là một loại hàm đặc biệt, khi được gọi, sẽ tự động hiểu và chọn đúng đối tượng gốc để gọi đúng hàm của đối tượng đó giữa lớp cơ sở và lớp dẫn xuất. Khả năng này được gọi là đa hình.

TẠI SAO PHẢI DÙNG HÀM ẢO

- Hàm ảo là cơ chế của C++ cho phép cài đặt đa cấu hình động, và được khai báo với từ khóa virtual.
- Nếu có nhiều hàm thành viên có cùng tên trong lớp cơ sở và lớp kế thừa, thì các hàm ảo giúp lập trình viên khả năng để gọi hàm thành viên của lớp khác với cùng lời gọi hàm phụ thuộc vào ngữ cảnh khác nhau. Đặc điểm này trong lập trình C++ còn được biết đến với tên gọi là tính đa hình.
- Nếu một lớp cơ sở và một lớp kế thừa có cùng hàm và nếu ta viết code để truy cập hàm đó bởi sử dụng con trỏ của lớp cơ sở, thì hàm trong lớp cơ sở được thực thi ngay cả khi đối tượng của lớp kế thừa được tham chiếu bởi biến con trỏ đó.

TẠI SAO PHẢI DÙNG HÀM ẢO

Để hiểu **tại sao phải sử dụng hàm ảo**, chúng ta cùng xem xét một ví dụ:

```
class Base{
public:        void print(){
                   cout<<"Base class";
       }};
class Derived : public Base{
public:
       void print(){
                   cout<<"Derived class";
       }
};
```

TA THẤY

Lớp **Derived** kế thừa **public** từ lớp **Base**. Ta tạo một **con trỏ (pointer)** có kiểu là lớp **Base** trỏ đến một đối tượng của lớp **Derived**. Sau đó, gọi hàm **print()** thì bản chất là gọi hàm **print()** của lớp **Base**. Mà điều ta muốn là phải gọi đến hàm **print()** của lớp **Derived** bởi con trỏ đang trỏ đến đối tượng của lớp **Derived**;


```
void main(){
        Derived derived1;
        Base* base1 = &derived1;
        //calls function of Base class
        base1->print();
        system("pause");
}
```

Để tránh trường hợp này, chúng ta khai báo hàm **print()** trong lớp **Base** là một **hàm ảo**. Lúc này, lớp dẫn xuất phải định nghĩa lại hàm ảo đó. Khi gọi hàm **print()** thông qua con trỏ thì sẽ gọi đến hàm được định nghĩa lại trong lớp dẫn xuất.

4.4. HÀM ẢO

Hàm ảo là một phần không thể thiếu để thể hiện tính đa hình trong kế thừa được hỗ trợ bởi ngôn ngữ C++.

Lưu ý: Con trỏ của lớp cơ sở có thể chứa địa chỉ của đối tượng thuộc lớp dẫn xuất, nhưng ngược lại thì không được

CÚ PHÁP

class <Tên lớp> {

protected: **virtual** <Kiểu trả về> <tên hàm>([<tham số>]) {} };

Ví dụ:

```
class NhanVien
{ public:
      virtual void nhap()
            {       // than hàm   }
      virtual void xuat()
  {       // than hàm   } };
```

ĐỊNH NGHĨA HÀM ẢO (VIRTUAL FUNCTION)

Sử dụng từ khóa **virtual** để định nghĩa một hàm ảo trong lớp cơ sở. Lớp dẫn xuất kế thừa từ lớp cơ sở **phải định nghĩa lại hàm ảo**

```
#include iostream>
using namespace std;
class Base
public:
virtual void print()//virtual function
                cout<<"Base class";
class Derived : public Base
public:
        void print()
                cout<<"Derived class";
void main()
        Derived derived1;
        Base* base1 = &derived1;
        //calls function of Derived class
        base1->print();
        system("pause");
```


ĐỊNH NGHĨA HÀM ẢO (VIRTUAL FUNCTION)

```
class Base {
    public:
        virtual void print() {
            // code
        }
};
class Derived : public Base {
    public:
        void print() {
            // code
        }
};
int main() {
    Derived derived1;
    Base* base1 = &derived1;
    base1->print();
    return 0;
}
```

print() of Derived class is called because print() in Base class is virtual

Hàm ảo chỉ khác hàm thành phần thông thường khi được gọi từ **một con trỏ**. Sử dụng hàm ảo khi muốn con trỏ đang trỏ tới đối tượng của lớp nào thì hàm thành phần của lớp đó sẽ được gọi mà không xem xét đến kiểu của con trỏ.

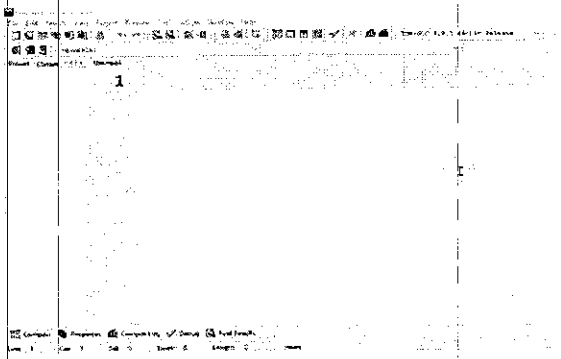
ĐẶC ĐIỂM

- Hàm ảo là cơ chế của C++ cho phép cài đặt đa cấu hình động.
- Nếu có nhiều hàm thành viên có cùng tên trong lớp cơ sở và lớp dẫn xuất, thì hàm ảo dùng để gọi hàm thành viên của lớp khác với cùng lời gọi hàm phụ thuộc vào ngữ cảnh khác nhau(tính đa hình)
- **Lưu ý**: Con trỏ của lớp cơ sở có thể chứa địa chỉ của đối tượng thuộc lớp dẫn xuất, nhưng ngược lại thì không được.

NHƯ VẬY

- Trong C ++, chúng ta có thể không ghi đè các hàm nếu chúng ta sử dụng một con trỏ của lớp cơ sở để trỏ đến một đối tượng của lớp dẫn xuất. Việc sử dụng các hàm ảo trong lớp cơ sở đảm bảo rằng hàm có thể được ghi đè trong các trường hợp này

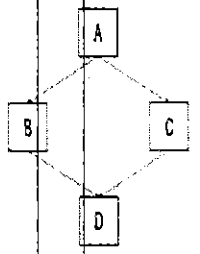
CHƯƠNG TRÌNH VÍ DỤ



1

4.5. LỚP CƠ SỞ ẢO

- Xét tình huống các lớp kế thừa theo sơ đồ như sau:
- Lớp B, C kế thừa từ lớp A. Lớp D kế thừa từ hai lớp B, C. Như vậy, lớp A được kế thừa hai lần bởi lớp D. Lần thứ nhất được kế thừa thông qua lớp B, lần thứ hai được kế thừa thông qua lớp C.
- Lúc này, nếu đối tượng của lớp D gọi đến một hàm được kế thừa từ lớp A thì sẽ gây ra một sự mơ hồ. Không biết hàm đó được kế thừa gián tiếp từ lớp B hay lớp C.
- Để giải quyết sự không rõ ràng này, C++ có một cơ chế mà nhờ đó chỉ có một bản sao của lớp A ở trong lớp D -> sử dụng lớp cơ sở ảo (virtual base class).



Khai báo lớp A là lớp cơ sở ảo trong các lớp B và C theo cú pháp sau:

```
class A{ //Định nghĩa lớp };
class B :  virtual public A{ //Định nghĩa lớp };
class C : virtual public A{ //Định nghĩa lớp };
class D : public B, public C{ //Định nghĩa lớp };
```

Việc chỉ định A là lớp cơ sở ảo trong các lớp B và C, nghĩa là A sẽ chỉ xuất hiện một lần trong lớp D. Khai báo này không ảnh hưởng đến các lớp B và C.

4.6. ĐA HÌNH

- Trong C++ sự kế thừa cho phép có sự tương ứng giữa lớp cơ sở và các lớp dẫn xuất trong sơ đồ thừa kế:
- Một con trỏ có kiểu lớp cơ sở luôn có thể trỏ đến địa chỉ của một đối tượng của lớp dẫn xuất.
- Tuy nhiên, khi thực hiện lời gọi một phương thức của lớp, trình biên dịch sẽ quan tâm đến kiểu của con trỏ chứ không phải đối tượng mà con trỏ đang trỏ tới: phương thức của lớp mà con trỏ có kiểu được gọi chứ không phải phương thức của đối tượng mà con trỏ đang trỏ tới được gọi.

Ví dụ: Lớp mayAcer kế thừa từ lớp Mayvitinh, cả hai lớp này đều định nghĩa phương thức show()

```
class Mayvitinh{
        public: void show(){
                cout << "mayvitinh" << endl; } };
class mayAcer:
        public Mayvitinh{
        public: void show(){
                cout << "mayAcer" << endl; } };
```

khi đó, nếu ta khai báo một con trỏ lớp mayAcer, nhưng lại trỏ vào địa chỉ của một đối tượng lớp Mayvitinh:

```
mayAcer may1; Mayvitinh *tenmay = &may1; tenmay->show();
```

165

4.6. ĐA HÌNH(POLYMORPHISM)

- Có thể thấy chương trình sau khi chạy sẽ gọi đến phương thức show() của lớp Mayvitinh, mà không gọi tới phương thức show() của lớp mayAcer.

-> Để giải quyết vấn đề này, ta phải sử dụng hàm ảo (cần sử dụng đến **tính đa hình**)

166

4.6. ĐA HÌNH(POLYMORPHISM)

Để chương trình gọi tới phương thức show() của lớp mayAcer ta sử dụng hàm ảo virtual như sau:

```
#include <iostream>
using namespace std;
class Mayvitinh{
        public:
                virtual void show(){
                        cout << "mayvitinh" << endl; } };
class mayAcer: public Mayvitinh{
        public:
                void show(){ cout << "mayAcer" << endl; } };
int main(){
        mayAcer may1;
        Mayvitinh *tenmay = &may1;
        tenmay->show(); }
```

167

4.6. ĐA HÌNH(POLYMORPHISM)

- Trong ví dụ trên ta sử dụng thêm từ khóa virtual vào hàm show() trong lớp cơ sở Mayvitinh.
- Từ khóa virtual này dùng để khai báo một hàm là hàm ảo.
- Khi khai báo hàm ảo với từ khóa virtual nghĩa là hàm này sẽ được gọi theo loại đối tượng được trỏ (hoặc tham chiếu), chứ không phải theo loại của con trỏ (hoặc tham chiếu). Điều này dẫn tới kết quả khác nhau:

➢ Nếu không khai báo hàm ảo virtual trình biên dịch sẽ gọi hàm tại lớp cở sở

➢ Nếu dùng hàm ảo virtual trình biên dịch sẽ gọi hàm tại lớp dẫn xuất

168

4.6. ĐA HÌNH(POLYMORPHISM)

Đa hình (polymorphism) nghĩa là có nhiều hình thái khác nhau. Tiêu biểu là, **đa hình** xuất hiện khi có một cấu trúc cấp bậc của các lớp và chúng liên quan với nhau bởi tính kế thừa

Nói cách khác:

Là hiện tượng mà những đối tượng thuộc các class khác nhau có thể biểu diễn cùng một thông thiệp theo các cách khác nhau.

Đa hình trong C++ nghĩa là một lời gọi tới một hàm thành viên sẽ làm cho một hàm khác để được thực thi phụ thuộc vào kiểu của đối tượng mà triệu hồi hàm đó.

169

VÍ DỤ

- **Trong quân đội gồm có: lục quân, hải quân, không quân. Bình thường nếu chỉ huy lệnh:**
- Hải quân! Đánh thì chỉ có hải quân sẽ chiến đấu
- Không quân! Đánh thì chỉ có không quân sẽ chiến đấu
- Lục quân! Đánh thì chỉ có lục quân sẽ chiến đấu
- Nếu muốn cả 3 đội quân cùng chiến đấu. Thay vì lệnh: Hải quân đánh, không quân đánh, lục quân đánh. Thì ta chỉ cần lệnh Đánh đây chính là sức mạnh của đa hình

170

CÓ 2 LOẠI ĐA HÌNH

- Compile time Polymorphism.
- Runtime Polymorphism.

1. Compile time Polymorphism:

- Tính đa hình này được sử dụng bằng cách nạp chồng hàm hoặc nạp chồng toán tử.

2. Runtime Polymorphism:

- Tính đa hình được thể hiện ở cách nạp chồng toán tử trong kế thừa

171

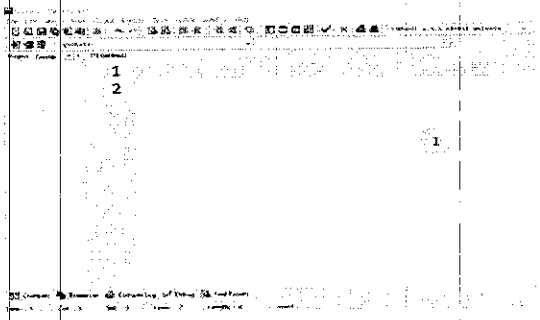
NHƯ VẬY

- Đa hình trong C++ chỉ đơn giản có nghĩa là nhiều hơn một dạng. Nghĩa là, một thực thể (chức năng hoặc toán tử) hoạt động khác nhau trong các tình huống khác nhau.
- Chúng ta có thể triển khai tính đa hình trong C ++ bằng các cách sau:
    - Nạp chồng hàm
    - Nạp chồng toán tử
    - Ghi đè hàm
    - Hàm ảo

172

ĐA HÌNH TRONG NẠP CHỒNG HÀM

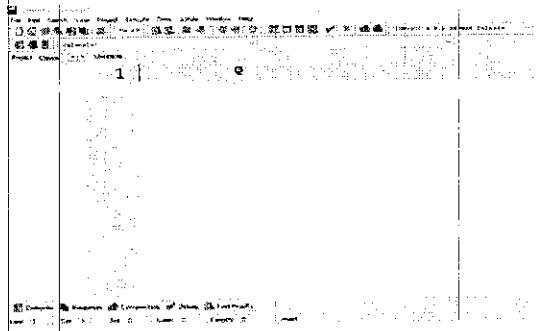
VD: Tính tổng 2 số a,b(nguyên, thực), tổng 3 số nguyên a,b,c



ĐA HÌNH TRONG NẠP CHỒNG TOÁN TỬ

Trong C ++, chúng ta có thể nạp chồng một toán tử. Chúng ta không thể sử dụng nạp chồng toán tử cho các kiểu dữ liệu cơ bản như int, double, float. Nạp chồng toán tử về cơ bản là nạp chồng hàm, trong đó các hàm toán tử khác nhau có cùng ký hiệu nhưng các toán hạng khác nhau. Tùy thuộc vào toán hạng, các hàm toán tử khác nhau được thực thi

VÍ DỤ: NẠP CHỒNG ĐỐI VỚI PHÉP TOÁN ++



ĐA HÌNH TRONG GHI ĐÈ HÀM

Trong tính kế thừa C ++, chúng ta có thể có cùng một hàm trong lớp cơ sở cũng như các lớp dẫn xuất của nó. Khi chúng ta gọi hàm bằng cách sử dụng một đối tượng của lớp dẫn xuất, thì hàm của lớp dẫn xuất được thực thi thay vì hàm trong lớp cơ sở. Vì vậy, các hàm khác nhau sẽ được thực thi tùy thuộc vào đối tượng gọi hàm.

NỘI DUNG

1. **Khuân hình hàm**
2. **Khuân hình lớp**

5.1. KHUÔN HÌNH HÀM

- Đặt vấn đề: xây dựng hàm tìm max của 2 số, 3 số, .. n số (số nguyên, số thực)

-> Giải quyết: Nạp chồng hàm max

Nhược điểm:

- Những hàm nạp chồng xử lý giống nhau, chỉ khác về tham số vi phạm nguyên tắc trùng lặp code, khó quản lý, bảo trì với những hàm phức tạp.
- Việc nạp chồng liên tục các hàm nhiều lần mất nhiều thời và bộ nhớ của máy tính,

5.1. KHUÔN HÌNH HÀM

- Giải pháp: Để tối ưu việc nạp chồng chúng ta sẽ xử lí chúng bằng cách viết một hàm khuôn mẫu
- **Khuôn mẫu hàm** là một cái khuôn dùng để tạo ta nhiều hàm có định nghĩa giống nhau. Nó có thể được tái sử dụng nhiều lần.
- Khi hàm khuôn mẫu được gọi, **compiler sẽ tạo ra một bản sao** của hàm đó, và thay thế kiểu dữ liệu tương ứng của các tham số khuôn mẫu.

KHÁI NIỆM

- Khuân hình hàm(khuân mẫu hàm): Là mẫu của hàm có tham số là kiểu của đối số. Với mỗi giá trị hợp lệ của đối số sẽ phát sinh một hàm cụ thể gọi là hàm thể hiện

CÚ PHÁP

template <class T1, class T2....> <kiểu tham chiếu> <tên khuôn hình hàm>([ds tham số])

{ //thân khuôn hình hàm}

Trong đó:

- class: kiểu hoặc sự phân lớp không phải là class chúng ta hay thấy trong lập trình hướng đối tượng.
- class có thể thay thế bằng **typename,** tuy nhiên nên sử dụng từ khóa class trong mọi trường hợp.
- T1,T2: Tên riêng thể hiện cho 1 kiểu dữ liệu tổng quát

185

VÍ DỤ

```
template <class T>
    T max(T a, T b){
     if(a>b) return a ;
     else return b;}
```

186

CÁCH SỬ DỤNG

- Sử dụng khuân hình hàm giống hệt 1 hàm thông thường
- Gọi hàm từ khuôn hình hàm:

<tên hàm>(đối số)

- Tên hàm trùng tên khuôn hình hàm

Ví dụ:

max(a,b) ;

hoặc cout<<max(a,b);

187

VÍ DỤ: TÌM GIÁ TRỊ MAX CỦA 1 MẢNG BẤT KỲ

```
1 #include<iostream>
2 using namespace std;
3
4
```

188

ƯU ĐIỂM

- Tiết kiệm rất nhiều thời gian
- Giảm code để bảo trì
- Có thể an toàn hơn,

Nhược điểm(rất nhỏ)

- Một số trình biên dịch cũ hơn không có hỗ trợ
- Thường tạo ra các thông báo lỗi trông có vẻ khó giải mã hơn so với các hàm thông thường
- Có thể tăng thời gian biên dịch và kích thước code

189

5.2. KHUÂN HÌNH LỚP

Ví dụ: xây dựng lớp MT1 gồm:

- Mảng các phần tử kiểu: int, char, float, long ...
- Các phương thức: nhập, in, cộng, trừ,..
- **Nhận xét:**
- Với mỗi kiểu dữ liệu của mảng sẽ có 1 lớp, các lớp này có chung các thao tác, khác nhau về kiểu dữ liệu

→ C++ cho phép xây dựng một mẫu của lớp, mẫu này có tham số để ứng với mỗi giá trị của tham số sẽ phát sinh một lớp

190

KHÁI NIỆM

- Là một mẫu của lớp có các tham số là các kiểu dữ liệu (tham số kiểu) . Với mỗi giá trị của tham số kiểu sẽ phát sinh ra một thể hiện là một lớp cụ thể (lớp khuôn hình)
- Cú pháp:

template <class kieu_du_lieu> class ten_lop { . . . }

- **Trong đó:**

- Kieu_du_lieu là tên kiểu(vd T1, T2) sẽ được xác định khi một lớp được khai báo.

191

VÍ DỤ

```
template <class T>
    class MT1{
    int spt;
    T d[10];
    public:
    void nhap();
      void in();
    ...}
```

192

SỬ DỤNG KHUÂN HÌNH LỚP

- Với mỗi giá trị của tham số kiểu, chương trình dịch sẽ phát sinh ra một lớp cụ thể
- Cú pháp: **<tên khuôn hình><kiểu>**
- Khai báo đối tượng: **<tên khuôn hình><kiểu> <tên biến>**

Ví dụ:

MT<int> a;

MT<float> b; MT<long> c;

193

CÁC THAM SỐ TRONG KHUÂN HÌNH LỚP

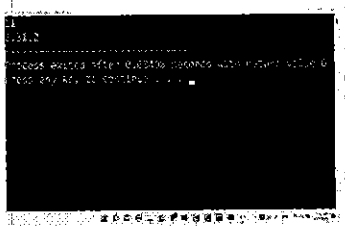
- Hoàn toàn giống như khuôn hình hàm, các khuôn hình lớp có thể có các tham số kiểu và tham số biểu thức.
- Một lớp thể hiện, được khai báo bằng cách liệt kê đằng sau tên khuôn hình lớp các tham số thực, là tên kiểu dữ liệu, với số lượng bằng các tham số trong danh sách của khuôn hình lớp (template<...>)

194

VÍ DỤ

Lập chương trình xây dựng khuôn hình lớp cho lớp điểm trong không gian 2 chiều. Yêu cầu xây dựng constructor để tạo đối tượng và phương thức xuất điểm ra màn hình.

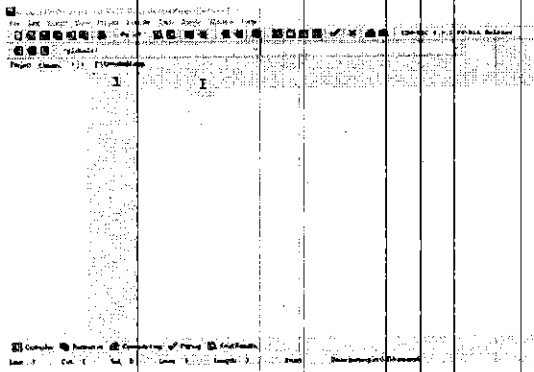
```
#include<iostream>
using namespace std;
template <class t>
class diem
{t x,y;
        public: diem ( t x1=0 ,t y1=0)
        {               x=x1;           y=y1;   }
void xuat()
        {       cout <<x<<y;   } };
main()
        {
        diem <int> a(2,1);
        a.xuat();
        diem <float> b(2.3,1.2);
        b.xuat();       }
```



195

VÍ DỤ:

Xây dựng khuôn hình lớp tam giác có thuộc tính là độ dài 3 cạnh, sử dụng hàm tạo để khởi tạo giá trị các cạnh và hiển thị thông tin các cạnh ra màn hình



196

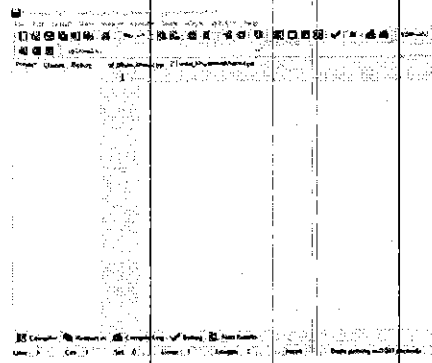
VÍ DỤ

Viết chương trình sử dụng khuôn hình hàm để sắp xếp mảng có kiểu dữ liệu bất kỳ theo chiều giảm dần.

-> Tiến hành sắp xếp mảng số thực, số nguyên theo thứ tự giảm dần -> sử dụng khuân hình hàm

197

CHƯƠNG TRÌNH



198

ĐẠI HỌC THÁI NGUYÊN
TRƯỜNG ĐẠI HỌC KỸ THUẬT CÔNG NGHIỆP

THUYẾT MINH
ĐỀ TÀI KHOA HỌC VÀ CÔNG NGHỆ CẤP TRƯỜNG
NĂM 2022

TÊN ĐỀ TÀI
XÂY DỰNG VIDEO BÀI GIẢNG MÔN LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG

Mã số: T2022-VD38

Chủ nhiệm đề tài: ThS. Trần Thị Ngọc Linh

THÁI NGUYÊN, NĂM 2023

THUYẾT MINH ĐỀ TÀI
KHOA HỌC VÀ CÔNG NGHỆ CẤP TRƯỜNG NĂM 2022

1. TÊN ĐỀ TÀI: XÂY DỰNG VIDEO BÀI GIẢNG MÔN LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG	2. MÃ SỐ: T2022-VD38

3. LĨNH VỰC NGHIÊN CỨU

Khoa học Tự nhiên	☐	Khoa học Kỹ thuật và Công nghệ	X
Khoa học Y, dược	☐	Khoa học Nông nghiệp	☐
Khoa học Xã hội	☐	Khoa học Nhân văn	☐

4. LOẠI HÌNH NGHIÊN CỨU

Cơ bản	Ứng dụng	Triển khai
☐	X	☐

5. THỜI GIAN THỰC HIỆN DỰ KIẾN: .12 tháng

Từ tháng 03 năm 2022 đến tháng 03 năm 2023

6. CHỦ NHIỆM ĐỀ TÀI

Họ và tên: Trần Thị Ngọc Linh Học vị: Thạc Sỹ

Chức danh khoa học: Năm sinh: 1981

Địa chỉ cơ quan: Khoa Điện tử - ĐH KTCN Điện thoại di động: 0945855155

Fax:

Điện thoại cơ quan:

E-mail: tranngoclinh@tnut.edu.vn

7. NHỮNG THÀNH VIÊN THAM GIA NGHIÊN CỨU ĐỀ TÀI

TT	Họ và tên	Đơn vị công tác và lĩnh vực chuyên môn	Nội dung nghiên cứu cụ thể được giao	Chữ ký
1				
2				
....				

8. ĐƠN VỊ PHỐI HỢP CHÍNH

Tên đơn vị trong và ngoài nước	Nội dung phối hợp nghiên cứu	Họ và tên người đại diện đơn vị

9. TỔNG QUAN TÌNH HÌNH NGHIÊN CỨU THUỘC LĨNH VỰC CỦA ĐỀ TÀI Ở TRONG VÀ NGOÀI NƯỚC

9.1. Trong nước *(phân tích, đánh giá tình hình nghiên cứu thuộc lĩnh vực của đề tài ở Việt Nam, liệt kê danh mục các công trình nghiên cứu, tài liệu có liên quan đến đề tài được trích dẫn khi đánh giá tổng quan)*

9.2. Ngoài nước *(phân tích, đánh giá tình hình nghiên cứu thuộc lĩnh vực của đề tài trên thế giới, liệt kê danh mục các công trình nghiên cứu, tài liệu có liên quan đến đề tài được trích dẫn khi đánh giá tổng quan)*

9.3. Danh mục các công trình đã công bố thuộc lĩnh vực của đề tài của chủ nhiệm và những thành viên tham gia nghiên cứu (*họ và tên tác giả; bài báo; ấn phẩm; các yếu tố về xuất bản*)

a) Của chủ nhiệm đề tài

b) Của các thành viên tham gia nghiên cứu

(Những công trình được công bố trong 5 năm gần nhất)

10. TÍNH CẤP THIẾT CỦA ĐỀ TÀI:

- Do dịch covid nên sinh viên phải học online nhiều, việc xây dựng video bài giảng môn Lập trình hướng đối tượng nhằm giúp cho sinh viên qua video bài giảng sẽ chủ động, đạt hiệu quả cao hơn trong quá trình học tập và tiếp thu kiến thức của môn học
- Học phần Lập trình hướng đối tượng là học phần cơ sở ngành được giảng dạy cho sinh viên ngành Kỹ thuật máy tính. Đây là học phần sinh viên cần học tập, làm bài tập, xây dựng phần mềm trên máy tính vì vậy việc xây dựng video trực quan, sinh động, giúp sinh viên hiểu bài là rất cần thiết.

11. MỤC TIÊU ĐỀ TÀI:

Xây dựng video bài giảng môn Lập trình hướng đối tượng để nâng cao chất lượng dạy và học online. Giúp sinh viên nắm được nội dung bài học và chuẩn bị bài trước khi đến lớp

12. ĐỐI TƯỢNG, PHẠM VI NGHIÊN CỨU

12.1. Đối tượng nghiên cứu: Xây dựng video bài giảng cho môn Lập trình hướng đối tượng

12.2. Phạm vi nghiên cứu: 45 video phục vụ cho các tiết học của môn học. Mỗi video 15 phút

13. CÁCH TIẾP CẬN, PHƯƠNG PHÁP NGHIÊN CỨU

13.1. Cách tiếp cận: Video sẽ bám sát vào đề cương, bài giảng học phần và chạy mô phỏng trực tiếp trên phần mềm DEV.

13.2. Phương pháp nghiên cứu: Quay video và giảng trên phần mềm Power Point và viết mã lệnh trên phần mềm DEV.

14. NỘI DUNG NGHIÊN CỨU VÀ TIẾN ĐỘ THỰC HIỆN

14.1. Nội dung nghiên cứu (***Mô tả chi tiết những nội dung nghiên cứu của đề tài***)

Chương 1: Lớp – Đối tượng

Chương 2: Hàm bạn

Chương 3: Toán tử tải bội

Chương 4: Kế thừa

Chương 5: Khuân Hình

14.2. Tiến độ thực hiện

STT	Các nội dung, công việc thực hiện	Sản phẩm	Thời gian (bắt đầu-kết thúc)	Người thực hiện
1	Chương 1: Lớp – Đối tượng	Các video bài giảng liên quan đến chương 1	3/2022 – 4/2022	Trần Thị Ngọc Linh
2	Chương 2: Hàm Bạn	Các video bài giảng liên quan đến chương 2	4/2022 – 5/2022	Trần Thị Ngọc Linh
3	Chương 3: Toán tử tải bội	Các video bài giảng liên quan đến chương 3	6/2022 – 8/2022	Trần Thị Ngọc Linh
4	Chương 4: Kế thừa	Các video bài giảng liên quan đến chương 4	8/2022 – 11/2022	Trần Thị Ngọc Linh
5	Chương 5: Khuân Hình	Các video bài giảng liên quan đến chương 5	11/2022-12/2022	Trần Thị Ngọc Linh
6	Hậu kỳ chỉnh sửa các video	Các video hoàn thiện	02/2023	Trần Thị Ngọc Linh
7	Viết báo cáo tổng kết	Quyển báo cáo tổng kết đề tài	03/2023	Trần Thị Ngọc Linh

15. SẢN PHẨM

Stt	Tên sản phẩm	Số lượng	Yêu cầu chất lượng sản phẩm *(mô tả chi tiết chất lượng sản phẩm đạt được như nội dung, hình thức, các chỉ tiêu, thông số kỹ thuật,...)*
I	Sản phẩm khoa học (Các công trình khoa học sẽ được công bố: sách, bài báo khoa học, ..)		
1.1			
...			
II	Sản phẩm đào tạo (cử nhân, thạc sĩ, tiến sĩ,...)		
2.1	Video bài giảng môn Lập trình hướng đối tượng	45 video	Video có hình ảnh rõ nét, tiếng to, rõ ràng, nội dung bám sát đề cương môn học, được dùng làm tài liệu giảng dạy cho sinh viên
2.2			
...			

III	Sản phẩm ứng dụng

16. PHƯƠNG THỨC CHUYỂN GIAO KẾT QUẢ NGHIÊN CỨU VÀ ĐỊA CHỈ ỨNG DỤNG

16.1. Phương thức chuyển giao: Bộ video bài giảng sẽ được chuyển giao cho Bộ môn Tin học Công Nghiệp, Khoa Điện tử, trường Đại học Kỹ thuật Công nghiệp.

16.2. Địa chỉ ứng dụng: Trường Đại học Kỹ thuật Công nghiệp.

17. TÁC ĐỘNG VÀ LỢI ÍCH MANG LẠI CỦA KẾT QUẢ NGHIÊN CỨU

17.1. Đối với lĩnh vực giáo dục và đào tạo:

- Có thể sử dụng như các bài giảng trong Đào tạo trực tuyến, từ xa

- Sinh viên có thể học trên các video theo thời gian chủ động và không giới hạn số lần. Do đó sinh viên có thể xem trước bài giảng và đến giờ học trao đổi, thảo luận sâu hơn với giáo viên.

17.2. Đối với lĩnh vực khoa học và công nghệ có liên quan

17.3. Đối với phát triển kinh tế-xã hội

17.4. Đối với tổ chức chủ trì và các cơ sở ứng dụng kết quả nghiên cứu:

- Góp phần xây dựng, hoàn thiện bộ học liệu phục vụ giảng dạy, học tập trong Nhà trường.

- Đảm bảo và nâng cao chất lượng đào tạo trong Nhà trường trong bối cảnh dịch bệnh Covid-19.

- Có thể khai thác, sử dụng phục vụ cho công tác Đào tạo từ xa.

18. KINH PHÍ THỰC HIỆN ĐỀ TÀI

Tổng kinh phí: 5.400.000 VNĐ

Bằng chữ: Năm triệu bốn trăm ngàn đồng chẵn./.

(Dự toán chi tiết các mục chi đính kèm có xác nhận của các đơn vị liên quan.)

Ngày 10 tháng 03 năm 2022

Chủ nhiệm đề tài	**PHÒNG KHCN&HTQT**
ThS. Trần Thị Ngọc Linh	
HỘI ĐỒNG KHOA KHOA ĐIỆN TỬ	**KT. HIỆU TRƯỞNG** **PHÓ HIỆU TRƯỞNG**
PGS.TS. Đào Huy Du	**PGS.TS. Vũ Ngọc Pi**

BỘ GIÁO DỤC VÀ ĐÀO TẠO
TRƯỜNG ĐẠI HỌC KỸ THUẬT CÔNG NGHIỆP
ĐẠI HỌC THÁI NGUYÊN

DỰ TOÁN KINH PHÍ ĐỀ TÀI KH&CN CẤP TRƯỜNG NĂM 2022

Tên đề tài: Xây dựng video bài giảng môn Lập trình hướng đối tượng
Chủ nhiệm đề tài: Trần Thị Ngọc Linh
Thành viên chính:
Thành viên:

ĐVT: VNĐ

STT	Nội dung	Dự toán			
1	***Mục chi tiền công lao động tham gia trực tiếp (1)***	**Người thực hiện**	**Số ngày công**	**Hệ số tiền công theo ngày (2)***	**Thành tiền**
1.1	Xây dựng thuyết minh đề tài	Trần Thị Ngọc Linh	0,5	0,45	335.250
1.2	Xây dựng các file bài giảng trình chiếu Powerpoint cho môn học	Trần Thị Ngọc Linh	1	0,45	670.500
1.3	Ghi hình các bài giảng chương 1	Trần Thị Ngọc Linh	1	0,45	670.500
1.4	Ghi hình các bài giảng chương 2	Trần Thị Ngọc Linh	1	0,45	670.500
1.5	Ghi hình các bài giảng chương 3	Trần Thị Ngọc Linh	1	0,45	670.500
1.6	Ghi hình các bài giảng chương 4	Trần Thị Ngọc Linh	1	0,45	670.500
1.7	Ghi hình các bài giảng chương 5	Trần Thị Ngọc Linh	1	0,45	670.500
1.8	Hậu kỳ chỉnh sửa các video	Trần Thị Ngọc Linh	1	0,45	670.500
1.9	Viết báo cáo tổng kết	Trần Thị Ngọc Linh	0,5	0,45	335.250
	Tổng 1		**8**		**5.364.000**
2	***Mục chi khác***				
	Phô tô, in ấn				36.000
	Tổng 2				**36.000**
	Tổng (1+2)				**5.400.000**

** 0,45 là hệ số của chủ nhiệm đề tài; 0,3 là hệ số của thành viên chính; 0,15 là hệ số của thành viên*

Cơ quan chủ trì
KT. HIỆU TRƯỞNG
PHÓ HIỆU TRƯỞNG

BỘ GIÁO DỤC ĐÀO TẠO
TRƯỜNG ĐẠI HỌC KỸ THUẬT CÔNG NGHIỆP
ĐẠI HỌC THÁI NGUYÊN

PGS.TS. Vũ Ngọc Pi

TRƯỞNG PHÒNG KHCN&HTQT

CHỦ NHIỆM ĐỀ TÀI

Trần Thị Ngọc Linh

TRƯỞNG PHÒNG KH-TC