

**ĐẠI HỌC ĐÀ NẴNG**  
**TRƯỜNG ĐẠI HỌC KỸ THUẬT**  
**KHOA CÔNG NGHỆ THÔNG TIN - ĐIỆN TỬ VIỄN THÔNG**

**GIÁO TRÌNH MÔN HỌC**  
**LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG**

**BIÊN SOẠN: LÊ THỊ MỸ HẠNH**



**ĐÀ NẴNG, 09/2002**

# MỤC LỤC



<b>CHƯƠNG 1: GIỚI THIỆU VỀ LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG.....</b>	<b>5</b>
I. LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG (OOP) LÀ GÌ ? .....	5
I.1. Lập trình tuyến tính .....	5
I.2. Lập trình cấu trúc.....	5
I.3. Sự trừu tượng hóa dữ liệu.....	6
I.4. Lập trình hướng đối tượng .....	6
II. MỘT SỐ KHÁI NIỆM MỚI TRONG LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG.....	8
II.1. Sự đóng gói (Encapsulation) .....	8
II.2. Tính kế thừa (Inheritance) .....	9
II.3. Tính đa hình (Polymorphism) .....	10
III. CÁC NGÔN NGỮ VÀ VAI ỨNG DỤNG CỦA OOP .....	11
<b>CHƯƠNG 2: CÁC MỞ RỘNG CỦA C++ .....</b>	<b>12</b>
I. LỊCH SỬ CỦA C++ .....	12
II. CÁC MỞ RỘNG CỦA C++ .....	12
II.1. Các từ khóa mới của C++.....	12
II.2. Cách ghi chú thích .....	12
II.3. Dòng nhập/xuất chuẩn.....	13
II.4. Cách chuyển đổi kiểu dữ liệu .....	14
II.5. Vị trí khai báo biến .....	14
II.6. Các biến const.....	15
II.7. Về struct, union và enum.....	16
II.8. Toán tử định phạm vi .....	16
II.9. Toán tử new và delete.....	17
II.10. Hàm inline .....	23
II.11. Các giá trị tham số mặc định .....	24
II.12. Phép tham chiếu .....	25
II.13. Phép đa năng hóa (Overloading) .....	29
<b>CHƯƠNG 3: LỚP VÀ ĐỐI TƯỢNG .....</b>	<b>39</b>
I. DẪN NHẬP .....	39
II. CÀI ĐẶT MỘT KIỂU DO NGƯỜI DÙNG ĐỊNH NGHĨA VỚI MỘT STRUCT ..	39
III. CÀI ĐẶT MỘT KIỂU DỮ LIỆU TRỪU TƯỢNG VỚI MỘT LỚP .....	41
IV. PHẠM VI LỚP VÀ TRUY CẬP CÁC THÀNH VIÊN LỚP .....	45
V. ĐIỀU KHIỂN TRUY CẬP TỚI CÁC THÀNH VIÊN .....	47
VI. CÁC HÀM TRUY CẬP VÀ CÁC HÀM TIỆN ÍCH.....	48
VII. KHỞI ĐỘNG CÁC ĐỐI TƯỢNG CỦA LỚP : CONSTRUCTOR.....	49
VIII.SỬ DỤNG DESTRUCTOR.....	51
IX. KHI NÀO CÁC CONSTRUTOR VÀ DESTRUCTOR ĐƯỢC GỌI ? .....	53
X. SỬ DỤNG CÁC THÀNH VIÊN DỮ LIỆU VÀ CÁC HÀM THÀNH VIÊN .....	54
XI. TRẢ VỀ MỘT THAM CHIẾU TỚI MỘT THÀNH VIÊN DỮ LIỆU PRIVATE..	57
XII. PHÉP GÁN BỞI TOÁN TỬ SAO CHÉP THÀNH VIÊN MẶC ĐỊNH .....	59
XIII.CÁC ĐỐI TƯỢNG HẰNG VÀ CÁC HÀM THÀNH VIÊN CONST.....	60
XIV.LỚP NHƯ LÀ CÁC THÀNH VIÊN CỦA CÁC LỚP KHÁC .....	64
XV. CÁC HÀM VÀ CÁC LỚP FRIEND.....	67

XVI.CON TRỎ THIS .....	68
XVII.CÁC ĐỐI TƯỢNG ĐƯỢC CẤP PHÁT ĐỘNG .....	71
XVIII.CÁC THÀNH VIÊN TÌNH CỦA LỚP .....	72
<b>CHƯƠNG 4: ĐA NĂNG HÓA TOÁN TỬ .....</b>	<b>76</b>
I. DẪN NHẬP .....	76
II. CÁC NGUYÊN TẮC CƠ BẢN CỦA ĐA NĂNG HÓA TOÁN TỬ .....	76
III. CÁC GIỚI HẠN CỦA ĐA NĂNG HÓA TOÁN TỬ .....	76
IV. CÁC HÀM TOÁN TỬ CÓ THỂ LÀ CÁC THÀNH VIÊN CỦA LỚP HOẶC KHÔNG LÀ CÁC THÀNH VIÊN .....	77
V. ĐA NĂNG HOÁ CÁC TOÁN TỬ HAI NGÔI .....	80
VI. ĐA NĂNG HÓA CÁC TOÁN TỬ MỘT NGÔI .....	87
VII. ĐA NĂNG HÓA MỘT SỐ TOÁN TỬ ĐẶC BIỆT .....	90
VII.1.Toán tử [] .....	91
VII.2.Toán tử () .....	92
VIII.TOÁN TỬ CHUYỂN ĐỔI KIỂU .....	94
IX. TOÁN TỬ NEW VÀ DELETE .....	95
IX.1.Đa năng hóa toán tử new và delete toàn cục .....	96
IX.2.Đa năng hóa toán tử new và delete cho một lớp .....	97
X. ĐA NĂNG HÓA CÁC TOÁN TỬ CHÈN DÒNG << VÀ TRÍCH DÒNG >> .....	98
XI. MỘT SỐ VÍ DỤ .....	99
XI.1.Lớp String .....	99
XI.2.Lớp Date .....	103
<b>CHƯƠNG 5: TÍNH KẾ THỪA .....</b>	<b>107</b>
I. DẪN NHẬP .....	107
II. KẾ THỪA ĐƠN .....	107
II.1.Các lớp cơ sở và các lớp dẫn xuất .....	107
II.2.Các thành viên protected .....	109
II.3.Ép kiểu các con trở lớp cơ sở tới các con trở lớp dẫn xuất .....	109
II.4.Định nghĩa lại các thành viên lớp cơ sở trong một lớp dẫn xuất: .....	113
II.5.Các lớp cơ sở public, protected và private .....	113
II.6.Các constructor và destructor lớp dẫn xuất .....	113
II.7.Chuyển đổi ngầm định đối tượng lớp dẫn xuất sang đối tượng lớp cơ sở .....	116
III. ĐA KẾ THỪA (MULTIPLE INHERITANCE) .....	116
IV. CÁC LỚP CƠ SỞ ẢO (VIRTUAL BASE CLASSES) .....	119
<b>CHƯƠNG 6: TÍNH ĐA HÌNH .....</b>	<b>122</b>
I. DẪN NHẬP .....	122
II. PHƯƠNG THỨC ẢO (VIRTUAL FUNCTION) .....	122
III. LỚP TRỪU TƯỢNG (ABSTRACT CLASS) .....	125
IV. CÁC THÀNH VIÊN ẢO CỦA MỘT LỚP .....	127
IV.1.Toán tử ảo .....	127
IV.2.Có constructor và destructor ảo hay không? .....	129
<b>CHƯƠNG 7: THIẾT KẾ CHƯƠNG TRÌNH THEO HƯỚNG ĐỐI TƯỢNG .....</b>	<b>132</b>
I. DẪN NHẬP .....	132
II. CÁC GIAI ĐOẠN PHÁT TRIỂN HỆ THỐNG .....	132
III. CÁCH TÌM LỚP .....	133
IV. CÁC BƯỚC CẦN THIẾT ĐỂ THIẾT KẾ CHƯƠNG TRÌNH .....	133
V. CÁC VÍ DỤ .....	134

<b>CHƯƠNG 8: CÁC DẠNG NHẬP/XUẤT.....</b>	<b>143</b>
I. DẪN NHẬP .....	143
II. CÁC DÒNG(STREAMS) .....	143
II.1.Các file header của thư viện iostream.....	143
II.2.Các lớp và các đối tượng của dòng nhập/xuất.....	144
III. DÒNG XUẤT.....	145
III.1.Toán tử chèn dòng .....	145
III.2.Nối các toán tử chèn dòng và trích dòng.....	146
III.3.Xuất ký tự với hàm thành viên put(); Nối với nhau hàm put() .....	147
IV. DÒNG NHẬP .....	148
IV.1.Toán tử trích dòng: .....	148
IV.2.Các hàm thành viên get() và getline() .....	149
IV.3.Các hàm thành viên khác của istream .....	151
IV.4.Nhập/xuất kiểu an toàn.....	151
V. NHẬP/XUẤT KHÔNG ĐỊNH DẠNG VỚI READ(),GCOUNT() VÀ WRITE() .....	151
VI. DÒNG NHẬP/ XUẤT FILE .....	152
VI.1.Nhập/xuất file văn bản .....	154
<b>CHƯƠNG 9: HÀM VÀ LỚP TEMPLATE.....</b>	<b>159</b>
I. CÁC HÀM TEMPLATE .....	159
II. CÁC LỚP TEMPLATE.....	161

**CHƯƠNG 1****GIỚI THIỆU VỀ LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG****I. LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG (OOP) LÀ GÌ ?**

Lập trình hướng đối tượng (Object-Oriented Programming, viết tắt là OOP) là một phương pháp mới trên bước đường tiến hóa của việc lập trình máy tính, nhằm làm cho chương trình trở nên linh hoạt, tin cậy và dễ phát triển. Tuy nhiên để hiểu được OOP là gì, chúng ta hãy bắt đầu từ lịch sử của quá trình lập trình – xem xét OOP đã tiến hóa như thế nào.

**I.1. Lập trình tuyến tính**

Máy tính đầu tiên được lập trình bằng mã nhị phân, sử dụng các công tắc cơ khí để nạp chương trình. Cùng với sự xuất hiện của các thiết bị lưu trữ lớn và bộ nhớ máy tính có dung lượng lớn nên các ngôn ngữ lập trình cấp cao đầu tiên được đưa vào sử dụng. Thay vì phải suy nghĩ trên một dãy các bit và byte, lập trình viên có thể viết một loạt lệnh gần với tiếng Anh và sau đó chương trình dịch thành ngôn ngữ máy.

Các ngôn ngữ lập trình cấp cao đầu tiên được thiết kế để lập các chương trình làm các công việc tương đối đơn giản như tính toán. Các chương trình ban đầu chủ yếu liên quan đến tính toán và không đòi hỏi gì nhiều ở ngôn ngữ lập trình. Hơn nữa phần lớn các chương trình này tương đối ngắn, thường ít hơn 100 dòng.

Khi khả năng của máy tính tăng lên thì khả năng để triển khai các chương trình phức tạp hơn cũng tăng lên. Các ngôn ngữ lập trình ngày trước không còn thích hợp đối với việc lập trình đòi hỏi cao hơn. Các phương tiện cần thiết để sử dụng lại các phần mã chương trình đã viết hầu như không có trong ngôn ngữ lập trình tuyến tính. Thật ra, một đoạn lệnh thường phải được chép lặp lại mỗi khi chúng ta dùng trong nhiều chương trình do đó chương trình dài dòng, logic của chương trình khó hiểu. Chương trình được điều khiển để nhảy đến nhiều chỗ mà thường không có sự giải thích rõ ràng, làm thế nào để chương trình đến chỗ cần thiết hoặc tại sao như vậy.

Ngôn ngữ lập trình tuyến tính không có khả năng kiểm soát phạm vi nhìn thấy của các dữ liệu. Mọi dữ liệu trong chương trình đều là dữ liệu toàn cục nghĩa là chúng có thể bị sửa đổi ở bất kỳ phần nào của chương trình. Việc dò tìm các thay đổi không mong muốn đó của các phần tử dữ liệu trong một dãy mã lệnh dài và vòng vèo đã từng làm cho các lập trình viên rất mất thời gian.

**I.2. Lập trình cấu trúc**

Rõ ràng là các ngôn ngữ mới với các tính năng mới cần phải được phát triển để có thể tạo ra các ứng dụng tinh vi hơn. Vào cuối các năm trong 1960 và 1970, ngôn ngữ lập trình có cấu trúc ra đời. Các chương trình có cấu trúc được tổ chức theo các công việc mà chúng thực hiện.

Về bản chất, chương trình chia nhỏ thành các chương trình con riêng rẽ (còn gọi là hàm hay thủ tục) thực hiện các công việc rời rạc trong quá trình lớn hơn, phức tạp hơn. Các hàm này được giữ càng độc lập với nhau càng nhiều càng tốt, mỗi hàm có dữ liệu và logic riêng. Thông tin được chuyển giao giữa các hàm thông qua các tham số, các hàm có thể có các biến cục bộ mà không một ai nằm bên ngoài phạm vi của hàm lại có thể truy xuất được chúng. Như vậy, các hàm có thể được xem là các chương trình con được đặt chung với nhau để xây dựng nên một ứng dụng.

Mục tiêu là làm sao cho việc triển khai các phần mềm dễ dàng hơn đối với các lập trình viên mà vẫn cải thiện được tính tin cậy và dễ bảo quản chương trình. Một chương trình có cấu trúc được hình thành bằng cách bẻ gãy các chức năng cơ bản của chương trình thành các mảnh nhỏ mà sau đó trở thành các hàm. Bằng cách cô lập các công việc vào trong các hàm, chương trình có cấu trúc có thể làm giảm khả năng của một hàm này ảnh hưởng đến một hàm khác. Việc này cũng làm cho việc tách các vấn đề trở nên dễ dàng hơn. Sự gói gọn này cho phép chúng ta có thể viết các chương trình sáng sủa hơn và giữ được điều khiển trên từng hàm. Các biến toàn cục không còn nữa và được thay thế bằng các tham số và biến cục bộ có phạm vi nhỏ hơn và dễ kiểm soát hơn. Cách tổ chức tốt hơn này nói lên rằng chúng ta có khả năng quản lý logic của cấu trúc chương trình, làm cho việc triển khai và bảo dưỡng chương trình nhanh hơn và hữu hiệu hơn và hiệu quả hơn.