20

**Các lớp cha**

Không nhân lên các đối tượng mà không cần thiết.

– William Occam

• Giới thiệu

• Các lớp có nguồn gốc

Chức năng Thành viên; Người xây dựng và Người phá hủy

• Phân cấp lớp

Nhập trường; Chức năng ảo; Trình độ chuyên môn rõ ràng; Kiểm soát ghi đè; sử dụng cơ sở thành viên, hàm trả về giảm bớt

• Các lớp trừu tượng

• Kiểm soát truy cập

Thành viên được bảo vệ; Quyền truy cập vào các lớp cơ sở; sử dụng-Khai báo và Kiểm soát truy cập

• Con trỏ đến thành viên

Con trỏ đến các thành viên chức năng; Con trỏ tới thành viên dữ liệu; Thành viên gốc và thành viên có nguồn gốc

• Lời khuyên

**20.1 Giới thiệu**

Từ Simula, C ++ đã mượn ý tưởng của các lớp và cấu trúc phân cấp lớp. Ngoài ra, nó còn mượn ý tưởng thiết kế rằng các lớp nên được sử dụng để mô hình hóa các khái niệm trong ứng dụng của lập trình viên và thế giới. C ++ cung cấp các cấu trúc ngôn ngữ hỗ trợ trực tiếp các khái niệm thiết kế này. Ngược lại, sử dụng các tính năng ngôn ngữ hỗ trợ ý tưởng thiết kế giúp phân biệt việc sử dụng C ++ hiệu quả. Sử dụng cấu trúc ngôn ngữ chỉ là đạo cụ ký hiệu cho các kiểu lập trình truyền thống là bỏ sót điểm mạnh của C ++.

Một khái niệm (ý tưởng, khái niệm, v.v.) không tồn tại một cách cô lập. Nó cùng tồn tại với các khái niệm liên quan và phần lớn sức mạnh của nó có được từ các mối quan hệ với các khái niệm khác. Ví dụ, cố gắng giải thích những gì về chiếc xe hơi. Bạn sẽ sớm giới thiệu các khái niệm về bánh xe, động cơ, người lái xe, người đi bộ, xe tải, xe cứu thương, đường xá, dầu nhớt, vé chạy quá tốc độ, nhà nghỉ, v.v. Vì chúng tôi sử dụng các lớp để đại diện cho các khái niệm, vấn đề trở thành cách thể hiện mối quan hệ giữa các khái niệm. Tuy nhiên, chúng tôi không thể diễn đạt các mối quan hệ tùy ý trực tiếp bằng ngôn ngữ lập trình. Ngay cả khi chúng tôi có thể, chúng tôi sẽ không muốn. Để trở nên hữu ích, các lớp của chúng ta nên được định nghĩa hẹp hơn so với các khái niệm hàng ngày của chúng ta - và chính xác hơn.

Khái niệm về một lớp cha và các cơ chế ngôn ngữ liên quan của nó được cung cấp để thể hiện các mối quan hệ thứ bậc, nghĩa là, để thể hiện tính chung giữa các lớp. Ví dụ, các khái niệm về hình tròn và hình tam giác có liên quan với nhau ở chỗ chúng đều là hình dạng; nghĩa là họ có chung khái niệm về một hình dạng. Do đó, chúng ta định nghĩa rõ ràng lớp hình tròn và lớp Tam giác để có chung lớp Hình dạng. Trong trường hợp đó, lớp chung, ở đây là hình dạng, được gọi là lớp cơ sở hoặc siêu lớp và các lớp cha từ ​​lớp đó, ở đây hình tròn và tam giác, được gọi là lớp cha hoặc lớp con. Biểu diễn hình tròn và hình tam giác trong một chương trình mà không liên quan đến khái niệm về hình dạng sẽ là thiếu một thứ thiết yếu. Chương này là một khám phá về ý nghĩa của ý tưởng đơn giản này, là cơ sở cho cái thường được gọi là lập trình hướng đối tượng. Các tính năng ngôn ngữ hỗ trợ xây dựng các lớp mới từ những lớp hiện có:

• Kế thừa triển khai: để tiết kiệm nỗ lực triển khai bằng cách chia sẻ các cơ sở được cung cấp bởi một lớp cơ sở

• Kế thừa giao diện: cho phép các lớp cha khác nhau được sử dụng thay thế cho nhau thông qua giao diện được cung cấp bởi một lớp cơ sở chung

Kế thừa giao diện thường được gọi là đa hình thời gian chạy (hoặc đa hình động). Ngược lại, việc sử dụng thống nhất các lớp không liên quan đến tính kế thừa được cung cấp bởi các mẫu (§3.4, Chương 23) thường được gọi là đa hình thời gian biên dịch (hoặc đa hình tĩnh).

Cuộc thảo luận về phân cấp lớp được tổ chức thành ba chương:

• Các lớp bắt nguồn (Chương 20): Chương này giới thiệu các tính năng cơ bản của ngôn ngữ hỗ trợ lập trình hướng đối tượng. Các lớp cơ sở và cha, các hàm ảo và kiểm soát truy cập được bao phủ.

• Phân cấp lớp (Chương 21): Chương này tập trung vào việc sử dụng các lớp cơ sở và lớp cha để tổ chức hiệu quả mã xung quanh khái niệm phân cấp lớp. Phần lớn chương này được dành để thảo luận về các kỹ thuật lập trình, nhưng các khía cạnh kỹ thuật của đa kế thừa (các lớp có nhiều hơn một lớp cơ sở) cũng được đề cập.

• Nhận dạng kiểu thời gian chạy (Chương 22): Chương này mô tả các kỹ thuật để điều hướng rõ ràng các cấu trúc phân cấp lớp. Cụ thể, các hoạt động chuyển đổi kiểu động và ép kiểu tĩnh được trình bày, cũng như hoạt động xác định kiểu của một đối tượng cho một trong các lớp cơ sở của nó (typeid).

Có thể tìm thấy phần giới thiệu ngắn gọn về ý tưởng cơ bản về tổ chức phân cấp của các kiểu trong Chương 3: các lớp cơ sở và cha (§3.2.2) và các hàm ảo (§3.2.3). Các chương này sẽ xem xét các tính năng cơ bản này và các kỹ thuật thiết kế và lập trình liên quan của chúng một cách chi tiết hơn.

**20.2 Các lớp có nguồn gốc**

Cân nhắc xây dựng một chương trình giao dịch với những người được một công ty tuyển dụng. Một chương trình như vậy có thể có cấu trúc dữ liệu như sau:

struct NhanVien {

string first\_name , family\_name;

char middle\_initial;

Date hiring\_date;

short department;

// ...

};

Tiếp theo, chúng tôi có thể cố gắng xác định một người quản lý:

struct QuanLi {

NhanVien empty; // hồ sơ nhân viên của người quản lý

DanhSach <NhanVien∗> group; // người quản lý

short level;

// ...

};

Một nhà quản lý cũng là một nhân viên; dữ liệu Nhân viên được lưu trữ trong thành viên empty của một đối tượng Người quản lý. Điều này có thể rõ ràng đối với một người đọc con người - đặc biệt là một người đọc cẩn thận - nhưng không có gì nói với trình biên dịch và các công cụ khác rằng Người quản lý cũng là một Nhân viên. Người quản lý∗ không phải là Nhân viên∗, vì vậy người ta không thể đơn giản sử dụng cái này khi người kia được yêu cầu. Đặc biệt, người ta không thể đưa Người quản lý vào danh sách Nhân viên mà không viết mã đặc biệt. Chúng tôi có thể sử dụng chuyển đổi loại rõ ràng trên Người quản lý∗ hoặc đưa địa chỉ của thành viên empty vào danh sách nhân viên. Tuy nhiên, cả hai giải pháp đều không phù hợp và có thể khá mù mờ. Cách tiếp cận đúng là tuyên bố rõ ràng rằng Người quản lý là Nhân viên, với một vài thông tin được thêm vào:

Struct QuanLi : public NhanVien {

DanhSach<NhanVien∗> group;

short level;

// ...

};

Người quản lý có nguồn gốc từ Nhân viên và ngược lại, Nhân viên là một lớp cơ sở của Người quản lý. Lớp người quản lý có các thành viên của lớp Nhân viên (tên\_người, phòng ban, v.v.) ngoài các thành viên của chính nó (nhóm, cấp, v.v.). Khởi tạo thường được biểu diễn bằng đồ thị bởi một con trỏ từ lớp cha đến lớp cơ sở của nó cho biết rằng lớp cha tham chiếu đến cơ sở của nó (thay vì ngược lại):

NhanVien



QuanLi