Chương 1

LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG

PHƯƠNG PHÁP GIẢI QUYẾT BÀI TOÁN MỚI

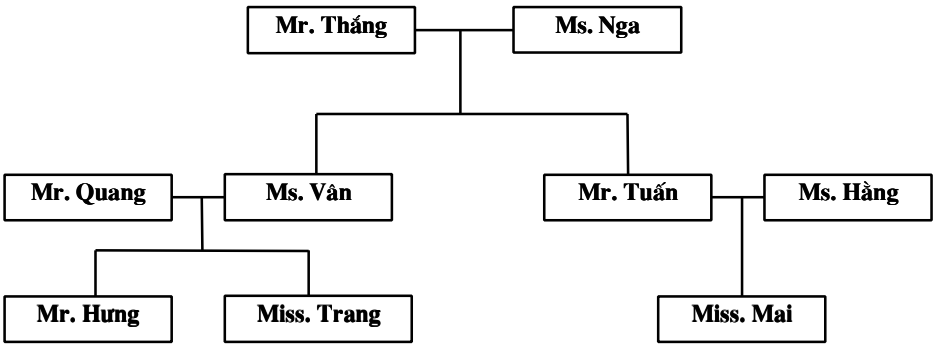
1. PHƯƠNG PHÁP LẬP TRÌNH

Từ nhiều năm nay chúng ta đã nghe nhiều đến thuật ngữ "Lập trình hướng đối tượng" (OOP - Object Oriented Programming). Vậy thực chất nó là gì? Để hiểu được vấn đề này chúng ta bắt đầu nhìn lại một chút lịch sử phát triển các phương pháp lập trình. Vào những ngày đầu phát triển của máy tính, khi các phần mềm còn rất đơn giản chỉ cỡ vài chục dòng lệnh, chương trình được viết tuần tự với các câu lệnh thực hiện từ đầu đến cuối. Cách viết chương trình như thế này gọi là phương pháp **lập trình tuyến tính**. Khoa học máy tính ngày càng phát triển, các phần mềm đòi hỏi ngày càng phức tạp và lớn hơn rất nhiều. Đến lúc này phương pháp lập trình tuyến tính tỏ ra kém hiệu quả và có những trường hợp người lập trình không thể kiểm soát được chương trình. Thế là phương pháp **lập trình cấu trúc** (LTCT) ra đời. Theo cách tiếp cận này, chương trình được tổ chức thành các chương trình con. Mỗi chương trình con đảm nhận xử lý một công việc nhỏ trong toàn bộ hệ thống. Mỗi chương trình con này lại có thể chia nhỏ thành các chương trình con nhỏ hơn. Quá trình phân chia như vậy tiếp tục diễn ra cho đến các chương trình con nhỏ nhận được đủ đơn giản. Người ta gọi đó là quá trình làm mịn dần. Các chương trình con tương đối độc lập với nhau, do đó có thể phân công cho từng nhóm đảm nhận viết các chương trình con khác nhau. Ngôn ngữ lập trình thể hiện rõ nét nhất phương pháp lập trình cấu trúc chính là Pascal. Tuy nhiên, khi sử dụng phương pháp lập trình này vẫn còn gặp một khó khăn lớn là tổ chức dữ liệu của hệ thống như thế nào trong máy tính. Bởi vì theo quan điểm của LTCT thì *Chương trình = Cấu trúc dữ liệu + Giải thuật*. Để làm được việc này đòi hỏi người lập trình phải có kiến rất vững về cấu trúc dữ liệu. Một khó khăn nữa gặp phải là giải thuật của chương trình phụ thuộc rất chặt chẽ vào cấu trúc dữ liệu, do vậy chỉ cần một sự thay đổi nhỏ ở cấu trúc dữ liệu cũng có thể làm thay đổi giải thuật và như vậy phải viết lại chương trình. Điều này rõ ràng không thể thích hợp khi phải xây dựng một dự án phần mềm rất lớn. Một phương pháp lập trình mới ra đời để khắc phục nhược điểm này và đó chính là phương pháp **lập trình hướng đối tượng** (LTHĐT). Điểm căn bản của phương pháp này là thiết kế chương trình xoay quanh dữ liệu của hệ thống. Nghĩa là lúc này các thao tác xử lý của hệ thống được gắn liền với dữ liệu và như vậy một sự thay đổi nhỏ của dữ liệu chỉ ảnh hưởng đến các một số nhỏ các hàm xử lý liên quan. Sự gắn kết giữa dữ liệu và các hàm xử lý trên chúng tạo ra đối tượng. Một ưu điểm nữa có ở phương pháp LTHĐT là cách tiếp cận bài toán trở nên gần gũi với thực tế hơn. Để hiểu rõ hơn về phương pháp lập trình này, không gì tốt hơn là chúng ta đi vào một bài toán cụ thể, chẳng hạn bài toán quan hệ gia đình. ở đây yêu

cầu làm thế nào để thể hiện được các mối quan hệ giữa các thành viên trong một gia đình trên máy tính và có thể trả lời được câu hỏi dạng khá tổng quát: "A và B có quan hệ như thế nào trong gia đình ?" với A và B là hai cá thể bất kỳ. Chúng ta sẽ phân tích xem cách giải quyết bài toán này như thế nào.

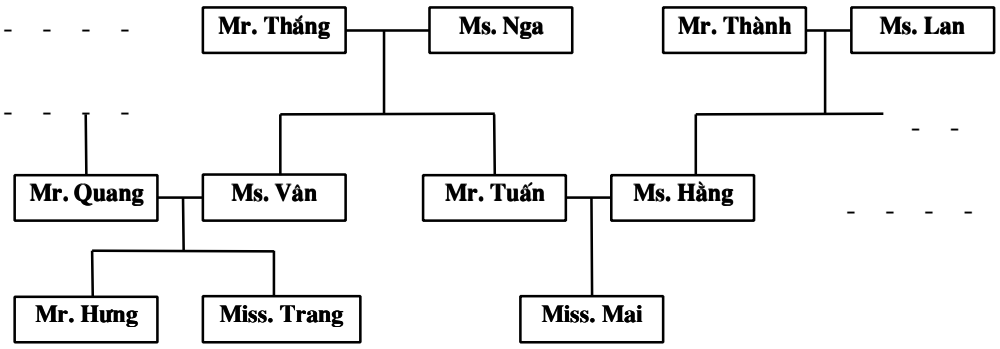
2. BÀI TOÁN QUAN HỆ GIA ĐÌNH

Trong xã hội, mỗi người đều có một gia đình, trong đó tồn tại nhiều mối quan hệ gia đình khá phức tạp như ông, bà, cha, mẹ, cô, chú, bác, v.v. Thông thường, để thể hiện các mối quan hệ này người ta biểu diễn bằng một sơ đồ cây quan hệ. Dưới đây là một ví dụ biểu diễn một gia đình ba thế hệ bằng hình 1.1.



Hình 1.1 Cây quan hệ trong một gia đình

Để giải quyết bài toán này theo phương pháp LTCT, công việc đầu tiên là phải xây dựng một cấu trúc dữ liệu thể hiện được cây quan hệ trên. Trông qua có vẻ là đơn giản nhưng nếu thử làm xem sẽ thấy không đơn giản chút nào, thậm chí còn khó. Bởi vì nó đòi hỏi người lập trình phải rất thành thạo sử dụng con trỏ, phải xây dựng được giải thuật cập nhật thông tin trên cây quan hệ. Các giải thuật này tương đối phức tạp đối với một cấu trúc dữ liệu như trong bài toán. Yêu cầu của bài toán là trả lời được câu hỏi dạng như "Hưng và Mai có quan hệ như thế nào ?". Câu trả lời của chương trình phải là "Hưng là anh họ của Mai". Để có thể thực hiện được như vậy, rõ ràng chúng ta phải xây dựng được giải thuật tìm được mối quan hệ giữa hai nút trên cây quan hệ. Một vấn đề phức tạp và tế nhị hơn là tên gọi cho các mối quan hệ gia đình ở Việt nam rất phong phú! Một khó khăn là phải vét cạn hết các mối quan hệ có thể có trên một cây quan hệ. Một khó khăn nữa gặp phải là khi cần phát triển, chương trình phải quản lý được nhiều gia đình cùng một lúc và các gia đình này có mối quan hệ thông gia với nhau. Hình 1.2 là sơ đồ quan hệ được phát triển từ sơ đồ ví dụ trên minh hoạ cho vấn đề này.



Hình 1.2 Mở rộng quan hệ giữa các gia đình

Một câu hỏi đặt ra: "Liệu với cấu trúc dữ liệu cũ có đảm bảo giải quyết được vấn đề này không ?". Rõ ràng câu trả lời là không. Sơ đồ quan hệ trên hình vẽ sẽ phải mô tả quan hệ của một gia đình. Chỉ với chút ít sự thay đổi về cấu trúc dữ liệu cũng dẫn đến một loạt vấn đề đòi hỏi phải viết lại các giải thuật của chương trình. Phương pháp lập trình mới hướng đối tượng cho phép chúng ta khắc phục được các vấn đề đã nêu ra. Trong suốt các trình bày của cuốn sách này sẽ cố gắng nêu bật được cách giải quyết vấn đề nhờ LTHĐT.

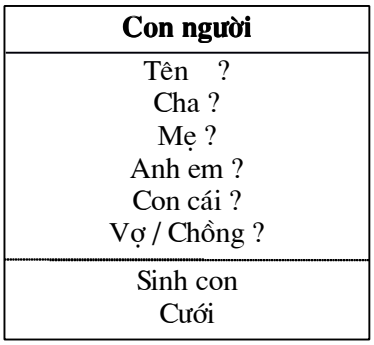
Theo cách tiếp cận LTHĐT, bài toán quan hệ gia đình được xem xét dưới góc độ quản lý tập các đối tượng **Con người**. Để biết mối quan hệ gia đình của mỗi cá thể, cần thể hiện một số quan hệ cơ bản như cha, mẹ, anh em, con cái, vợ chồng của cá thể đó. Như vậy, mỗi đối tượng con người của bài toán có các thuộc tính riêng, nói lên rằng cha mẹ, anh em, v.v.. của họ là ai. Ngoài ra cũng cần có một thuộc tính nữa cho biết tên cá thể là gì. Có thể mô tả một lớp các đối tượng con người như hình 1.3.

Con người
Tên ?
Cha ?
Mẹ ?
Anh em ?
Con cái ?
Vợ / Chồng ?

Hình 1.3 Mô tả một lớp các đối tượng con người

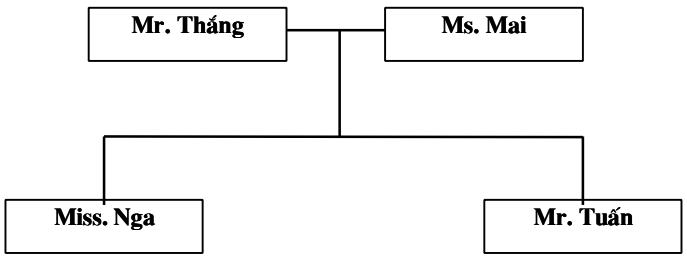
Nếu chỉ có như vậy thì chẳng khác gì một cấu trúc hay bản ghi trong cấu trúc dữ diệu được sử dụng ở phương pháp LTCT. Vấn đề ở đây là phương pháp LTHĐT xem các mối quan hệ trong gia đình được hình thành một cách tự nhiên do các sự kiện cụ thể trong cuộc sống tạo nên. Ví dụ, khi người phụ nữ sinh con, đứa con cô

ta sinh ra sẽ có mẹ là cô ta và cha là chồng cô ta, đồng thời anh chồng phải được cập nhật để có thêm đứa con này. Những đứa con trước của cô ta sẽ có thêm đứa em này và đứa bé có thêm những người anh hoặc người chị đó. Dễ dàng thấy rằng có hai sự kiện chính tác động đến mối quan hệ gia đình là sự sinh con của người phụ nữ và hôn nhân giữa hai cá thể khác giới trong xã hội. Các sự kiện này gắn liền với từng con người trong bài toán. Điều này có nghĩa là khi nói đến một sự kiện nào thì phải chỉ ra nó được phát sinh bởi người nào. Ví dụ, khi nói sự kiện sinh con thì phải biết người nào sinh. Khi một sự kiện của một con người nào đó xảy ra (ví dụ như sinh con) thì các thuộc tính của chính anh ta sẽ bị thay đổi, đồng thời thuộc tính của một số đối tượng liên quan cũng có thể thay đổi theo. Quá trình đóng gói giữa các sự kiện và thuộc tính sẽ tạo ra **Đối tượng,** khái niệm cơ bản của phương pháp LTHĐT. Một mô tả chung cho các đối tượng con người của bài toán được gọi là một **Lớp**. Hình 1.4 minh hoạ một lớp Con người có thêm các sự kiện của bài toán.



Hình 1.4 Các sự kiện bổ sung gắn với con người.

Sau khi đã gắn kết các sự kiện vào đối tượng như trên, vấn đề là tạo một sơ đồ quan hệ gia đình như thế nào. Dưới đây là một ví dụ minh hoạ việc tạo ra một quan hệ gia đình dựa trên các sự kiện cuộc sống. Giả thiết là đã có hai đối tượng là ông Thắng và bà Mai.



Các sự kiện để tạo ra cây quan hệ trên có thể viết theo trật tự như sau:

Thắng.Cưới (Mai)

Mai.Sinh con (gái, Nga)

Đối tượng tạo sự kiện . Sự kiện (thông số kèm theo sự kiện)

Các sự kiện viết theo cú pháp:

Như vậy các bạn đã thấy rằng chúng ta không cần phải quan tâm đến cách tạo một cấu trúc cây quan hệ như thế nào bên trong dữ liệu của chương trình mà vẫn có thể cung cấp dữ liệu bài toán cho chương trình thông qua các sự kiện như trên. Chúng ta quay lại vấn đề chính của bài toán là trả lời các câu hỏi về mối quan hệ gia đình như thế nào khi tiếp cận bài toán theo phương pháp này. Để trả lời được câu hỏi tổng quát "X và Y có quan hệ gia đình như thế nào ?" ta cần phải trả lời các câu hỏi nhỏ như "X có phải là anh của Y không ?", "X có phải là ông nội của Y không ?", v.v.. Câu hỏi có thể nhìn từ góc độ đối tượng X như : "Đối tượng có phải là anh của Y không ?", "có phải là ông nội của Y không ?", v.v.. Như vậy câu hỏi lúc này đã giao về cho đối tượng để trả lời. Các đối tượng lúc này cần phải có các phương thức để trả lời các câu hỏi như vậy. Và bây giờ một lớp đối tượng Con người được minh hoạ như hình 1.5.

Con người
Tên ? Cha ? Mẹ ? Anh em ? Con cái ? Vợ / Chồng ?
Sinh con Cưới Là anh Là ông nội .....

Hình 1.5 Thêm các phương thức trả lời câu hỏi

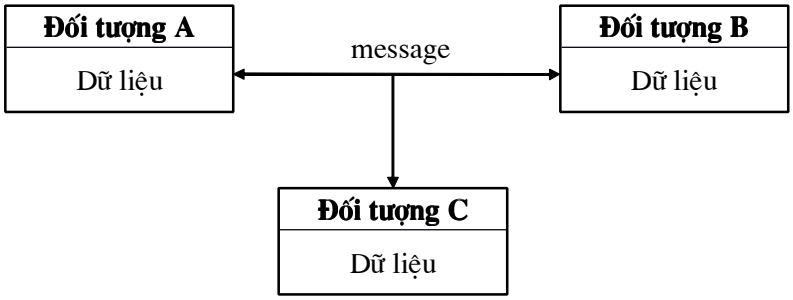
Ta xem xét các đối tượng trả lời các câu hỏi như thế nào? Chẳng hạn X trả lời câu hỏi "Đối tượng có phải là anh của Y không ?" hoàn toàn đơn giản. Nó chỉ cần kiểm tra xem Y có phải là anh em mà trong thuộc tính của nó lưu giữ không. Hoàn

toàn tương tự đối với các câu hỏi quan hệ gần như là em, là chị, là bố, là mẹ,... Còn câu hỏi như "Đối tượng có phải là ông nội của Y không ?" phức tạp hơn chút ít. Để trả lời được các câu hỏi có quan hệ xa như thế ta phải dựa vào kết quả trả lời của các câu hỏi về các quan hệ gần gũi hơn. Để biết được X đúng là ông nội của Y thì phải chỉ ra một người Z nào đó mà X là bố của Z và Z là bố của Y. Nếu không chỉ ra được Z thì X không phải là ông nội của Y. Việc tìm kiếm Z hoàn toàn đơn giản bởi vì chương trình quản lý tập các đối tượng con người. Hãy tìm Z trong tập đối tượng Con người. Có thể thấy câu hỏi ban đầu đã được phân chia thành hai câu hỏi đơn giản với chúng mà đã có cách trả lời. Tóm lại, các vấn đề của bài toán đã được giải quyết khi tiếp cận theo phương pháp LTHĐT. Một lợi điểm có thể thấy ngay là bài toán được phân tích rất gần với thực tế và tự nhiên.

Trên đây mới chỉ là sự phân tích sơ khai bài toán dựa theo phương pháp LTHĐT. Để làm hoàn chỉnh được bài toán còn cần một số kĩ thuật của LTHĐT như tính kế thừa, tính đa hình, ... Chúng tôi hy vọng rằng qua sự phân tích một bài toán nhỏ trên đã chứng tỏ được lợi ích của phương pháp LTHĐT. Trong mục tiếp theo chúng tôi sẽ tóm tắt và đưa ra tổng quan sơ bộ về LTHĐT.

3. LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG

Lập trình hướng đối tượng đặt trọng tâm vào đối tượng, yếu tố quan trọng trong quá trình phát triển chương trình và không cho phép dữ liệu biến động tự do trong hệ thống. Dữ liệu được gắn chặt với các hàm thành các vùng riêng mà chỉ có các hàm đó tác động lên và cấm các hàm bên ngoài truy nhập tới một cách tuỳ tiện. LTHĐT cho phép chúng ta phân tích bài toán thành các thực thể được gọi là các đối tượng và sau đó xây dựng các dữ liệu cùng các hàm xung quanh các đối tượng đó. Các đối tượng có thể tác động, trao đổi thông tin với nhau thông qua cơ chế thông báo (message). Tổ chức một chương trình hướng đối tượng có thể mô tả như trong hình 1.6.



Hình 1.6 Các đối tượng trao đổi qua thông báo

LTHĐT có các đặc tính chủ yếu sau:

1. Tập trung vào dữ liệu thay cho các hàm

2. Chương trình được chia thành các đối tượng.
3. Các cấu trúc dữ liệu được thiết kế sao cho đặc tả được đối tượng.
4. Các hàm thao tác trên các vùng dữ liệu của đối tượng được gắn với cấu trúc dữ liệu đó.
5. Dữ liệu được đóng gói lại, được che giấu và không cho phép các hàm ngoại lai truy nhập tự do.
6. Các đối tượng tác động và trao đổi thông tin với nhau qua các hàm
7. Có thể dễ dàng bổ sung dữ liệu và các hàm mới vào đối tượng nào đó khi cần thiết
8. Chương trình được thiết kế theo cách tiếp cận từ dưới lên (bottom-up).

Sau đây là một số khái niệm được sử dụng trong LTHĐT.

3.1 Một số khái niệm

Đối tượng (object)

Đối tượng là sự kết hợp giữa dữ liệu và thủ tục (hay còn gọi là các phương thức - method) thao tác trên dữ liệu đó. Có thể đưa ra công thức phản ánh bản chất kỹ thuật của LTHĐT như sau:

Đối tượng = Dữ liệu + Phương thức

Lớp (class)

Lớp là một khái niệm mới trong LTHĐT so với các kỹ thuật lập trình khác. Đó là một tập các đối tượng có cấu trúc dữ liệu và các phương thức giống nhau (hay nói cách khác là một tập các đối tượng cùng loại). Như vậy khi có một lớp thì chúng ta sẽ biết được một mô tả cấu trúc dữ liệu và phương thức của các đối tượng thuộc lớp đó. Mỗi đối tượng sẽ là một thể hiện cụ thể (instance) của lớp đó. Trong lập trình, chúng ta có thể coi một lớp như là một kiểu, còn các đối tượng sẽ là các biến có kiểu của lớp.

Nguyên tắc đóng gói dữ liệu

Trong LTCT ta đã thấy là các hàm hay thủ tục được sử dụng mà không cần biết đến nội dung cụ thể của nó. Người sử dụng chỉ cần biết chức năng của hàm cũng như các tham số cần truyền vào để gọi hàm chạy mà không cần quan tâm đến những lệnh cụ thể bên trong nó. Người ta gọi đó là sự đóng gói về chức năng.

Trong LTHĐT, không những các chức năng được đóng gói mà cả dữ liệu cũng như vậy. Với mỗi đối tượng người ta không thể truy nhập trực tiếp vào các thành phần dữ liệu cảu nó mà phải thông qua các thành phần chức năng (các phương thức) để làm việc đó.

Chúng ta sẽ thấy sự đóng gói thực sự về dữ liệu chỉ có trong một ngôn ngữ LTHĐT "thuần khiết" (pure) theo nghĩa các ngôn ngữ được thiết kế ngay từ đầu chỉ

cho LTHĐT. Còn đối với các ngôn ngữ "lai" (hybrid) được xây dựng trên các ngôn ngữ khác ban đầu chưa phải là HĐT như C++ được nói đến trong cuốn sách này, vẫn có những ngoại lệ nhất định vi phạm nguyên tắc đóng gói dữ liệu.

Tính kế thừa (inheritance)

Một khái niệm quan trọng của LTHĐT là sự kế thừa. Sự kế thừa cho phép chúng ta định nghĩa một lớp mới trên cơ sở các lớp đã tồn tại, tất nhiên có bổ sung những phương thức hay các thành phần dữ liệu mới. Khả năng kế thừa cho phép chúng ta sử dụng lại một cách dễ dàng các module chương trình mà không cần một thay đổi các module đó. Rõ ràng đây là một điểm mạnh của LTHĐT so với LTCT.

Tính đa hình (polymorphime)

Tính đa hình xuất hiện khi có khái niệm kế thừa. Giả sử chúng ta có một kế thừa lớp hình tứ giác và lớp hình tam giác kế thừa từ lớp hình đa giác (hình tam giác và tứ giác sẽ có đầy đủ các thuộc tính và tính chất của một hình đa giác). Lúc này một đối tượng thuộc lớp hình tam giác hay tứ giác đều có thể hiểu rằng nó là một hình đa giác. Mặt khác với mỗi đa giác ta có thể tính diện tích của nó. Như vậy làm thế nào mà một đa giác có thể sử dụng đúng công thức để tính diện tích phù hợp với nó là hình tam giác hay tứ giác. Ta gọi đó là tính đa hình.

3.2 Các ưu điểm của LTHĐT

LTHĐT đem lại một số lợi thế cho người thiết kế lẫn người lập trình. Cách tiếp cận hướng đối tượng giải quyết được nhiều vấn đề tồn tại trong quá trình phát triển phần mềm và tạo ra được những phần mềm có độ phức tạp và chất lượng cao. Phương pháp này mở ra một triển vọng to lớn cho người lập trình. Những ưu điểm chính của LTHĐT là:

1. Thông qua nguyên lý kế thừa, chúng ta có thể loại bỏ được những đoạn chương trình lặp lại trong quá trình mô tả các lớp và có thể mở rộng khả năng sử dụng của các lớp đã xây dựng mà không cần phải viết lại.
2. Chương trình được xây dựng từ những đơn thể (đối tượng) trao đổi với nhau nên việc thiết kế và lập trình sẽ được thực hiện theo quy trình nhất định chứ không phải dựa vào kinh nghiệm và kỹ thuật như trước nữa. Điều này đảm bảo rút ngắn được thời gian xây dựng hệ thống và tăng năng suất lao động.
3. Nguyên lý đóng gói hay che giấu thông tin giúp người lập trình tạo ra được những chương trình an toàn không bị thay đổi bởi những đoạn chương trình khác.
4. Có thể xây dựng được ánh xạ các đối tượng của bài toán vào đối tượng chương trình.
5. Cách tiếp cận thiết kế đặt trọng tâm vào dữ liệu, giúp chúng ta xây dựng được mô hình chi tiết và dễ dàng cài đặt hơn.

6. Các hệ thống hướng đối tượng dễ mở rộng, nâng cấp thành những hệ lớn hơn.
7. Kỹ thuật truyền thông báo trong việc trao đổi thông tin giữa các đối tượng làm cho việc mô tả giao diện với các hệ thống bên ngoài trở nên đơn giản hơn.
8. Có thể quản lý được độ phức tạp của những sản phẩm phần mềm.

3.3 Những ứng dụng của LTHĐT

LTHĐT là một trong những thuật ngữ được nhắc đến nhiều nhất hiện nay trong công nghệ phần mềm và nó được ứng dụng để phát triển phần mềm trong nhiều lĩnh vực khác nhau. Trong số đó, ứng dụng quan trọng và nổi tiếng nhất hiện nay là thiết kế giao diện với người sử dụng, kiểu như Windows. Các hệ thông tin quản lý trong thực tế thường rất phức tạp, chứa nhiều đối tượng với các thuộc tính và hàm phức tạp. Để giải quyết những hệ thông tin phức tạp như thế, LTHĐT tỏ ra rất hiệu quả. Các lĩnh vực ứng dụng phù hợp với kỹ thuật LTHĐT có thể liệt kê như dưới đây:

*Các hệ thống làm việc theo thời gian thực.

*Các hệ mô hình hoá hoặc mô phỏng các quá trình.

*Các hệ cơ sở dữ liệu hướng đối tượng.

*Các hệ siêu văn bản (hypertext), đa phương tiện (multimedia).

*Các hệ thống trí tuệ nhân tạo và các hệ chuyên gia.

*Các hệ thống song song và mạng nơ-ron.

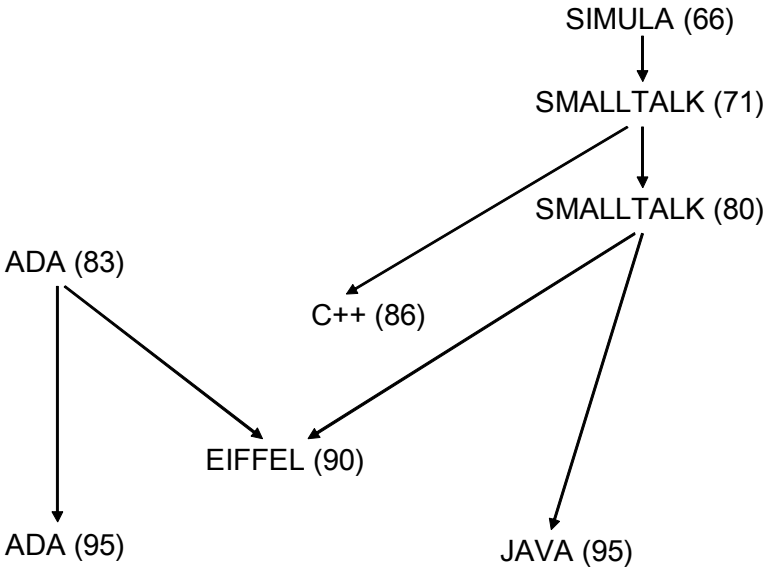
*Các hệ tự động hoá văn phòng hoặc trợ giúp quyết định.

*Các hệ CAD/CAM.

Với nhiều đặc tính phong phú của LTHĐT nói riêng, của phương pháp phân tích thiết kế và phát triển hướng đối tượng nói chung chúng ta hy vọng công nghiệp phần mềm sẽ có những cải tiến nhảy vọt không những về chất lượng, mà còn gia tăng nhanh về số lượng trong tương lai.

4. CÁC NGÔN NGỮ LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG

LTHĐT không phải là đặc quyền của một ngôn ngữ đặc biệt nào. Cũng giống như kỹ thuật lập trình có cấu trúc, các khái niệm trong LTHĐT được thể hiện trong nhiều ngôn ngữ lập trình khác nhau. Những ngôn ngữ cung cấp được những khả năng LTHĐT được gọi là ngôn ngữ lập trình hướng đối tượng. Tuy vẫn có những ngôn ngữ chỉ cung cấp khả năng tạo lớp và đối tượng mà không cho phép kế thừa, do đó hạn chế khả năng LTHĐT. Hình 1.7 cho chúng ta một cái nhìn tổng quan về sự phát triển các ngôn ngữ LTHĐT.



Hình 1.7 Sự phát triển của các ngôn ngữ LTHĐT

Các ngôn ngữ SIMULA, SMALLTALK, JAVA thuộc họ ngôn ngữ LTHĐT thuần khiết, nghĩa là nó không cho phép phát triển các chương trình cấu trúc trên các ngôn ngữ loại này. Còn ngôn ngữ C++ thuộc loại ngôn ngữ "lai" bởi vì nó được phát triển từ ngôn ngữ C. Do đó trên C++ vẫn có thể sử dụng tính cấu trúc và đối tượng của chương trình. Điều này tỏ ra rất phù hợp khi chúng ta mới bắt đầu học một ngôn ngữ lập trình. Đó chính là lý do mà chúng tôi sử dụng ngôn ngữ C++ để giới thiệu phương pháp LTHĐT trong cuốn sách này. Một lý do khác nữa là C++ sử dụng cú pháp của ngôn ngữ C là ngôn ngữ rất thông dụng trong lập trình chuyên nghiệp.

5. NGÔN NGỮ LẬP TRÌNH C++

Vào năm 1983, giáo sư Bjarne Stroustrap bắt đầu nghiên cứu và phát triển việc cài đặt khả năng LTHĐT vào ngôn ngữ C tạo ra một ngôn ngữ mới gọi là C++. Tên gọi này có thể phân tích ý nghĩa rằng nó là ngôn ngữ C mà có hai đặc điểm mới tương ứng với hai dấu cộng. Đặc điểm thứ nhất là một số khả năng mở rộng so với C như tham chiếu, chồng hàm, tham số mặc định... Đặc điểm thứ hai chính là khả năng LTHĐT. Hiện nay C++ chưa phải là một ngôn ngữ hoàn toàn ổn định. Kể từ khi phiên bản đầu tiên ra đời vào năm 1986 đã có rất nhiều thay đổi trong các phiên bản C++ khác nhau: bản 1.1 ra đời vào năm 1986, 2.0 vào năm 1989 và 3.0 vào năm 1991. Phiên bản 3.0 này được sử dụng để làm cơ sở cho việc định nghĩa một ngôn ngữ C++ chuẩn (kiểu như Ansi C).

Trên thực tế hiện nay tất cả các chương trình dịch C++ đều tương thích với phiên bản 3.0. Vì vậy C++ hầu như không gây bất kỳ một khó khăn nào khi chuyển đổi từ một môi trường này sang môi trường khác, như chúng ta đã biết C++ như là một sự bổ sung khả năng LTHĐT vào ngôn ngữ C. Sẽ có nhiều người nghĩ rằng ngôn ngữ C nói ở đây là C theo chuẩn ANSI. Thực ra không phải hoàn toàn như vậy. Tên thực tế vẫn tồn tại một vài điểm không tương thích giữa ANSI C và C++.

Mặt khác cũng cần thấy rằng những mở rộng có trong C++ so với Ansi C không chỉ là để phục vụ cho mục đích tạo cho ngôn ngữ khả năng LTHĐT. Có những thay đổi chỉ với mục đích đơn thuần là tăng sức mạnh cho ngôn ngữ C hiện thời.

Ngoài ra có một vài thay đổi nhỏ ở C++ so với ANSI C như sau:

* Định nghĩa các hàm: khai báo, truyền tham số và giá trị trả lại.
* Sự tương thích giữa các con trỏ.
* Tính linh hoạt của các hằng (const).

Các đặc điểm mở rộng trong C++

Như đã đề cập ở trên C++ chứa cả những mở rộng so với C mà không liên quan đến kỹ thuật hướng đối tượng. Những mở rộng này sẽ được mô tả cụ thể trong chương sau, ở đây chúng ta chỉ tóm tắt lại một vài điểm chính.

* Khả năng viết các dòng chú thích mới.
* Khả năng khai báo linh hoạt hơn.
* Khả năng định nghĩa lại các hàm: các hàm cùng tên có thể thực hiện theo những thao tác khác nhau. Các lời gọi hàm sẽ dùng kiểu và số tham số để xác định đúng hàm nào cần thực hiện.
* Có thêm các toán tử định nghĩa bộ nhớ động mới: new và delete.
* Khả năng định nghĩa các hàm inline để tăng tốc độ thực hiện chương trình.
* Tạo các biến tham chiếu đến các biến khác.

LTHĐT trong C++

C++ chứa đựng khái niệm lớp. Một lớp bao gồm các thành phần dữ liệu hay là thuộc tính và các phương thức hay là hàm thành phần. Từ một lớp ta có thể tạo ra các đối tượng hoặc bằng cách khai báo thông thường một biến có kiểu là lớp đó hoặc bằng cách cấp phát bộ nhớ động nhờ sử dụng toán tử new. C++ cho phép chúng ta đóng gói dữ liệu nhưng nó không bắt buộc chúng ta thực hiện điều đó. Đây là một nhược điểm của C++. Tuy nhiên cũng cần thấy rằng bản thân C++ chỉ là sự mở rộng của C nên nó không thể là một ngôn ngữ LTHĐT thuần khiết được.

C++ cho phép ta định nghĩa các hàm thiết lập (constructor) cho một lớp. Hàm thiết lập là một phương thức đặc biệt được gọi đến tại thời điểm một đối tượng của lớp được tạo ra. hàm thiết lập có nhiệm vụ khởi tạo một đối tượng: cấp phát bộ nhớ, gán các giá trị cho các thành phần dữ liệu cũng như việc chuẩn bị chỗ cho các đối tượng mới. Một lớp có thể có một hay nhiều hàm thiết lập. Để xác định hàm thiết lập nào cần gọi đến, chương trình biên dịch sẽ so sánh các đối số với các tham số truyền vào. Tương tự như hàm thiết lập, một lớp có thể có một hàm huỷ bỏ (destructor), một phương thức đặc biệt được gọi đến khi đối tượng được giải phóng khỏi bộ nhớ.

Lớp trong C++ thực chất là một kiểu dữ liệu do người sử dụng định nghĩa. Khái niệm định nghĩa chồng toán tử cho phép định nghĩa các phép toán trên một lớp giống như các kiểu dữ liệu chuẩn của C. Ví dụ ta có thể định nghĩa một lớp số phức với các phép toán cộng, trừ, nhân, chia.

Cũng giống như C, C++ có khả năng chuyển đổi kiểu. Không những thế, C++ còn cho phép mở rộng sự chuyển đổi này sang các kiểu do người sử dụng tự định nghĩa (các lớp). Ví dụ, ta có thể chuyển đổi từ kiểu chuẩn int của C sang kiểu số phức mà ta định nghĩa chẳng hạn.

C++ cho phép thực hiện kế thừa các lớp đã xây dựng. Từ phiên bản 2.0 trở đi, C++ còn cho phép một lớp kế thừa cùng một lúc từ nhiều nhiều lớp khác nhau (gọi là sự đa kế thừa).

Cuối cùng C++ cung cấp những thao tác vào ra mới dựa trên cơ sở khái niệm luồng dữ liệu (flow). Sự ưu việt của các thao tác này ở chỗ:

* Sử dụng đơn giản.
* Kích thước bộ nhớ được rút gọn.
* Khả năng áp dụng trên các kiểu do người sử dụng định nghĩa bằng cách sử dụng cơ chế định nghĩa chồng toán tử.