**Ngôn Ngữ C++**

**1.C++ Là Gì**

C++ là một loại ngôn ngữ lập trình. Đây là một dạng ngôn ngữ đa mẫu hình tự do có kiểu tĩnh và hỗ trợ lập trình thủ tục, dữ liệu trừu trượng, lập trình hướng đối tượng, và lập trình đa hình. Từ thập niên 1990, C++ đã trở thành một trong những ngôn ngữ thương mại phổ biến nhất trong khi đó.

Bjarne Stroustrup của Bell Labs đã phát triển C++ (mà tên nguyên thủy là "C với các lớp" trong suốt thập niên 1980 như là một bản nâng cao của ngôn ngữ C. Những bổ sung nâng cao bắt đầu với sự thêm vào của khái niệm lớp, tiếp theo đó là các khái niệm hàm ảo, chồng toán tử, đa kế thừa, tiêu bản, và xử lý ngoại lệ. Tiêu chuẩn của ngôn ngữ C++ đã được thông qua trong năm 1998 như là ISO/IEC 14882:1998. Phiên bản hiện đang lưu hành là phiên bản C++14, ISO/IEC 14882:2014[1]. Hiện tại tiêu chuẩn mới nhất của ngôn ngữ C++ là C++14, phát hành ngày 15/12/2014 (ghi theo định dạng ISO 8601 là 2014-12-15). Và C++ còn là ngôn ngữ được cải tiến từ C

**2.Tổng quan về kỹ thuật**

Trong tiêu chuẩn 1998 của C++ có hai phần chính: phần ngôn ngữ cốt lõi và phần Thư viện chuẩn C++(STL - Standard Template Library). Phần thư viện này lại bao gồm hầu hết thư viện tiêu bản chuẩn và phiên bản có điều chỉnh chút ít của thư viện chuẩn C. Nhiều thư viện C++ hiện hữu không thuộc về tiêu chuẩn như là thư viện Boost. Thêm vào đó, nhiều thư viện không theo tiêu chuẩn được viết trong C một cách tổng quát đều có thể sử dụng trong các chương trình C++.

**3.Chức năng dẫn nhập trong C++**

So với C, C++ tăng cường thêm nhiều tính năng, bao gồm: khai báo như mệnh đề, chuyển kiểu giống như hàm, new/delete, bool, các kiểu tham chiếu, const, các hàm nội tuyến (inline), các đối số mặc định, nạp chồng hàm, không gian tên (namespace), các lớp (bao gồm tất cả các chức năng liên quan tới lớp như kế thừa, hàm thành viên (phương pháp), hàm ảo, lớp trừu tượng, và cấu tử), nạp chồng toán tử, tiêu bản, toán tử phạm vi ::, xử lý ngoại lệ, và sự nhận dạng kiểu trong thời gian thi hành.

C++ còn tiến hành nhiều phép kiểm tra kiểu hơn C trong nhiều trường hợp.

Câu lệnh chú giải bắt đầu với // nguyên là một phần của BCPL được tái sử dụng trong C++.

Một số thành phần của C++ sau này đã được thêm vào C, bao gồm const, inline, khai báo biến trong vòng lặp for và chú giải kiểu C++ (sử dụng ký hiệu //). Tuy nhiên, C99 cũng bổ sung thêm một số tính năng không có trong C++, ví dụ như macro với số đối số động.

Vì được phát triển từ C, trong C++, thuật ngữ đối tượng có nghĩa là vùng nhớ như được dùng trong C, chứ không phải là một phiên bản của lớp như được hiểu trong phần lớn ngôn ngữ lập trình hướng đối tượng khác.

**4.Thư viện C++**

Thư viện chuẩn C++ dùng lại thư viện chuẩn C với một số điều chỉnh nhỏ để giúp nó hoạt động tốt hơn với ngôn ngữ C++. Một bộ phận lớn khác của thư viện C++ dựa trên Thư viện tiêu bản chuẩn (hay còn gọi là STL - viết tắt từ chữ Standard Template Library). Thư viện này có nhiều công cụ hữu dụng như là các thùng chứa (ví dụ như vector, danh sách liên kết và biến lặp (tổng quát hóa từ khái niệm con trỏ) để cung cấp những thùng chứa này sự truy cập giống như là truy cập mãng. Xa hơn nữa, bảng (đa) ánh xạ (mảng kết hợp) và (đa) tập, tất cả được cung cấp để có thể xuất ra các giao diện tương thích. Do đó, có thể dùng tiêu bản để viết các thuật toán tổng quát mà chúng làm việc được với bất kì thùng chứa nào hay với bất kì dãy nào được định nghĩa bởi biến lặp. Giống như C, các tính năng của thư viện này thì được truy cập bởi việc sử dụng lệnh dẫn hướng #include để bao gồm một tập tin tiêu đề chuẩn. C++ cung ứng 69 tiêu đề chuẩn, trong đó có 19 tiêu đề không còn hiệu lực nữa.

Vì thư viện chuẩn được thiết kế bởi những chuyện gia hàng đầu và đã được chứng minh trong toàn bộ lịch sử kĩ nghệ, các thành phần của thư viện này được khuyến cáo sử dụng thay vì dùng những phần viết tay bên ngoài hay những phương tiện cấp thấp khác. Ví dụ, dùng std:vector hay std::string thay vì dùng kiểu mảng đơn thuần sẽ không những là cho "đời sống dễ thở hơn", mà còn là một cách hữu hiệu để viết phần mềm được an toàn và linh hoạt hơn.

STL nguyên là một thư viện của hãng HP và sau đó là của SGI, trước khi nó được nhận vào thành chuẩn C++. Tiêu chuẩn thì không tham chiếu nó bằng cái tên "STL", khi đa phần nó chỉ là bộ phận tiêu chuẩn. Tuy vậy, nhiều người vẩn dùng khái niệm "STL" này để phân biệt nó với phần còn lại của thư viện C++ như là IOstream, quốc tế hóa (ký tự và ngôn ngữ trình bày), chẩn đoán, thư viện C, v.v..

Một đề án mang tên STLPort, dựa cơ sở trên SGI STL, bảo trì các thiết lập mới của STL, IOStream và string. Các đề án khác cũng có những xây dựng đặc thù riêng của thư viện chuẩn với các mục tiêu thiết kế khác nhau. Mỗi nơi sản xuất hay phổ biến nhà trình dịch C++ đều bao gồm một sự thiết lập của thư viện, vì đây là phần quan trọng của tiêu chuẩn và lại là kỳ vọng của người lập trình.

**5.C++ Các chức năng hướng đối tượng**

C++ dẫn nhập thêm một số chức năng hướng đối tượng (OO) lên C. Nó cung cấp các lớp mà có 4 chức năng thông dụng trong các ngôn ngữ OO: tính trừu tượng, tính bao đóng, tính đa hình, và tính kế thừa.

Lưu ý: trong phần này các từ "hàm nội tại", "phương pháp", hay "hàm" đều có cùng một nghĩa là "phương thức thuộc về một lớp".

**a.Tính đóng gói**

C++ xây dựng tính đóng bằng cách cho phép mọi thành viên của một lớp có thể được khai báo bằng các từ khoá public, private, hay protected. (xem thêm các khái niệm cơ bản trong ngôn ngữ OOP). Một thành viên private chỉ có thể được truy cập từ các phương pháp (hàm nội tại) là thành viên của chính lớp đó hay được truy cập từ các hàm và các lớp được đặc biệt cho phép sử dụng bằng cách dùng từ khóa friend. Một thành viên protected của một lớp sẽ có thể truy cập được từ các thành viên (nào đó) của các lớp có tính kế thừa của nó hay cũng có thể truy cập được từ các thành viện của chính lớp đó và của mọi thành viên friend.

Nguyên lý của OOP là mọi và chỉ có các hàm là có thể truy cập được đến các giá trị nội tại của cùng lớp thì nên có tính đóng. C++ có hỗ trợ đặc tính này (qua các hàm thành viên và các hàm friend), nhưng C++ lại không là yêu cầu bắt buộc: người lập trình có thể khai báo các phần hay tất cả các giá trị nội tại là công cộng (public), và cũng cho phép làm cho toàn bộ lớp trở thành công cộng. Lý do là vì C++ hỗ trợ không chỉ lập trình hướng đối tượng mà còn hỗ trợ các mẫu hình yếu hơn như là lập trình mô-đun.

Một thói quen tốt cần có trong thực hành là khai báo mọi dữ liệu đều là riêng tư (private), hay ít nhất ở dạng bảo tồn, và sau đó, tạo ra một giao diện nhỏ (thông qua các phương pháp) cho người dùng của lớp này dấu đi các chi tiết thiết lập bên trong.

**b.Tính đa hình**

Khái niệm đa hình được dùng khá rộng rãi và là khái niệm bị lạm dụng cũng như không được định nghĩa rõ ràng.

Nói chung tính đa hình trong lập trình hướng muốn nói đến 1 đoạn code nhưng trong 2 trường hợp khác nhau có thể xuất ra 2 kết quả khác nhau. Vì tính chất ra nhiều kết quả khác nhau này nên nó được gọi là đa hình.

Trong trường hợp của C++, khái niệm này thường được nối kết với các tên của các hàm thành viên. Các hàm thành viên này có cùng tên, sự khác nhau chỉ có thể được dựa vào một hay cả hai yếu tố sau:

* Số lượng và kiểu của các đối số (tức là nguyên mẫu của hàm) -- Tính chất này gọi là đa hình tĩnh (static polymorphism)
* Kiểu lớp mà thực thể thực sự thuộc vào. Tính chất này được dùng khi hàm thành viên được định nghĩa là hàm ảo qua từ khóa virtual—tính chất này gọi là đa hình động (dynamic polymorphism)

Khi được gọi thì chương trình sẽ tùy theo hai yếu tố trên để xác định chính xác hàm nào phải được thực thi trong số các hàm cùng tên.

Trong C, thì đa hình (động) có thể đạt tới bằng cách dùng từ khóa switch hay dùng [con trỏ hàm](https://vi.wikipedia.org/w/index.php?title=Con_tr%E1%BB%8F_h%C3%A0m&action=edit&redlink=1).

C++ còn cung cấp hai tính năng độc đáo cho đa hình là:

* Sự tải bội (overloading): Cho phép một toán tử hay một hàm có những ứng xử khác nhau phụ thuộc vào kiểu của các toán hạng hay các tham số tại thời điểm toán tử hay hàm được triệu gọi.

Ví dụ, ta có thể định nghĩa hai hàm trùng tên như sau: float Demo(float a, float b) {return a + b;} và int Demo(int a, int b) {return a - b;} Ta cũng có thể tải bội phép cộng cho lớp MATRIX để có thể viết được C = A + B khi A, B và C có kiểu MATRIX.

* Tính ảo (virtual): Cho phép một phương thức (hàm thành viên hoặc toán tử) của lớp có ứng xử khác nhau phụ thuộc vào sự kế thừa của lớp con cháu (Xem phương thức Chao() trong ví dụ trên.

Hai tính năng trên cho phép chương trình định ra nhiều sự thiết lập khác nhau của một hàm để sử dụng ứng với các kiểu (khác nhau) của các đối tượng.

Việc *quá tải hàm* cho phép các chương trình khai báo nhiều hàm có chung một tên (ngay cả việc các hàm này thuộc cùng một lớp). Các hàm này phân biệt được bởi số lượng và kiểu của các [tham số](https://vi.wikipedia.org/wiki/Tham_s%E1%BB%91_(khoa_h%E1%BB%8Dc_m%C3%A1y_t%C3%ADnh)).

**c.Tính kế thừa**

Kế thừa từ một lớp cơ sở có thể được khai báo thông qua các đặc tính công cộng, bảo tồn, hay riêng tư. Những đặc tính này cho phép xác định khi nào các lớp dẫn xuất hay không liên quan có thể sử dụng các thành viên công cộng, bảo tồn, hay riêng tư của lớp cơ sở. Tuy nhiên, chỉ có sự kế thừa dạng công cộng là hoàn toàn theo đúng ý nghĩa của việc "kế thừa". Hai dạng khác thì ít được dùng hơn. Nếu các đặc tả này không được khai báo thì việc kế thừa được gán mặc định là dạng riêng tư cho lớp cơ sở và dạng công cộng cho một cấu trúc cơ sở.

Các lớp cơ sở có thể được khai báo là ảo (thông qua từ khóa virtual). Kế thừa ảo bảo đảm rằng chỉ có một thực thể của lớp cơ sở tồn tại trong đồ thị kế thừa, tránh được một số vấn đề mơ hồ của việc đa kế thừa.

Đa kế thừa cũng là một tính năng có nhiều tranh cãi trong C++. Tính đa kế thừa cho phép một lớp được dẫn xuất từ nhiều hơn một lớp cơ sở; điều này có thể dẫn tới một đồ thị phức tạp của các quan hệ kế thừa. Ví dụ, lớp "Buổi học" có thể kế thừa từ hai lớp "thời gian" và "bộ môn". Một số ngôn ngữ khác như Java, tiến hành cách thức tương tự bằng cách cho phép kế thừa của nhiều giao diện trong khi giới hạn số lượng của các lớp cơ sở (kế thừa) chỉ còn là một lớp. (giao diện, không như lớp, không cho phép thiết lập nội dung của các thành viên và do đó không thể có thực thể)