ĐẠI HỌC BÁCH KHOA HÀ NỘI

BÁO CÁO

# Tìm hiểu về các mô thức lập trình (imperative paradigm languages)

Môn học: **Kỹ Thuật Lập Trình**

Giảng viên: **Vũ Đức Vượng**

Mã lớp học: **128715**

Sinh viên: **Chu Văn Hiếu**

Mã số sinh viên: **20194046**

Email: **Hieu.cv194046@sis.hust.edu.vn**

# I. Tổng quan về một số mô thức lập trình phổ biến

## 1. Visual paradigm

Trong máy tính , ngôn ngữ lập trình trực quan ( VPL ) là bất kỳ ngôn ngữ lập trình nào cho phép người dùng tạo chương trình bằng cách thao tác các phần tử chương trình bằng đồ thị thay vì chỉ định chúng bằng văn bản .

VPL cho phép lập trình với các biểu thức trực quan, sắp xếp không gian của văn bản và ký hiệu đồ họa, được sử dụng như các thành phần của cú pháp hoặc ký hiệu phụ . Ví dụ, nhiều VPL (được gọi là luồng dữ liệu hoặc lập trình theo sơ đồ ) dựa trên ý tưởng về "hộp và mũi tên", trong đó các hộp hoặc các đối tượng màn hình khác được coi như các thực thể, được kết nối bằng các mũi tên, đường hoặc cung biểu thị các mối quan hệ.

Các VPL có thể được phân loại thêm, theo loại và mức độ biểu đạt hình ảnh được sử dụng, thành các ngôn ngữ dựa trên biểu tượng, ngôn ngữ dựa trên biểu mẫu và ngôn ngữ biểu đồ. Môi trường lập trình trực quan cung cấp các yếu tố đồ họa hoặc biểu tượng mà người dùng có thể thao tác theo cách tương tác theo một số ngữ pháp không gian cụ thể để xây dựng chương trình.

Mục tiêu chung của VPL là làm cho việc lập trình dễ tiếp cận hơn với người mới và hỗ trợ lập trình viên ở ba cấp độ khác nhau:

- Cú pháp
- Ngữ nghĩa
- Ngữ dụng

Hiện nay các ngôn ngữ lập trình, hệ quản trị cơ sở dữ liệu theo hướng trực quan thường dùng như : Visual Basic, Visual Foxpro, Visual C, Delphi...

## 2. Parallel paradigm

Ngôn ngữ lập trình song song là ngôn ngữ được thiết kế để lập trình các thuật toán và ứng dụng trên các máy tính song song. Xử lý song song là cơ hội lớn để phát triển các hệ thống hiệu suất cao và giải quyết các vấn đề lớn trong nhiều lĩnh vực ứng dụng. Ngôn ngữ lập trình song song (còn được gọi là ngôn ngữ đồng thời ) cho phép thiết kế các thuật toán song song như một tập hợp các hành động đồng thời được ánh xạ trên các phần tử tính toán khác nhau.

"Số lượng transistor trên mỗi đơn vị inch vuông sẽ tăng lên gấp đôi sau mỗi 24 tháng."Gordon Moore – sáng lập viên của tập đoàn intel.

Cùng với định luật Moore, các máy tính hiện nay đã được trang bị các bộ vi xử lý đa nhân mạnh mẽ. Tuy nhiên để tận dụng được sức mạnh đó đòi hỏi lập trình viên phải tận dụng được hết các nhân trong bộ vi xử lý. Vì vậy, thay vì lập trình tuần tự như trước đây chỉ sử dụng một nhân của bộ vi xử lý thì người lập trình viên

ngày nay phải dùng kỹ thuật lập trinh song song để tận dụng hiệu suất của bộ vi xử lý đa nhân.

Bộ vi xử lý có nhiều nhân sẽ tăng tốc chương trình song song tuy nhiên không có nghĩa là nó sẽ tăng lên 100% trên một nhân (core) được thêm vào. Thậm chí chương trình song song không hề tăng hiệu suất lên trong một số trường hợp. Vì vậy lập trình viên phải biết quyết định khi nào sử dụng lập trình song song bằng cách sử dụng các công cụ đo lường để xác định được tốc độ thực thi của chương trình.

Lập trình song song là một công việc rất phức tạp so với lập trình tuần tự thông thường, người phát triển phải thực hiện một quá trình "song song hóa", biến đổi các chương trình tuần tự thành chương trình song song có khả năng tận dụng tối đa sức mạnh của hệ thống. Hiện nay, các phân loại của mô hình lập trình song song có thể được chia rộng rãi thành hai lĩnh vực: tương tác quá trình và phân rã vấn đề.

### 3. Concurrent paradigm

Khái niệm tính toán đồng thời thường bị nhầm lẫn với khái niệm liên quan nhưng khác biệt của tính toán song song , mặc dù cả hai đều có thể được mô tả là "nhiều quá trình thực thi trong cùng một khoảng thời gian ". Trong tính toán song song, việc thực thi xảy ra tại cùng một thời điểm vật lý.

ví dụ: trên các bộ xử lý riêng biệt của máy nhiều bộ xử lý , với mục tiêu tăng tốc độ tính toán — tính toán song song là không thể trên bộ xử lý đơn ( một lõi ), vì chỉ một bộ xử lý tính toán có thể xảy ra bất kỳ lúc nào (trong bất kỳ chu kỳ đồng hồ đơn lẻ nào). Ngược lại, tính toán đồng thời bao gồm các vòng đời của quy trình chồng chéo, nhưng việc thực thi không nhất thiết phải xảy ra cùng một lúc. Mục tiêu ở đây là mô hình hóa các quy trình trong thế giới bên ngoài xảy ra đồng thời, chẳng hạn như nhiều máy khách truy cập vào một máy chủ cùng một lúc. Cấu trúc hệ thống phần mềm bao gồm nhiều phần giao tiếp đồng thời có thể hữu ích để giải quyết sự phức tạp, bất kể các phần có thể được thực thi song song hay không.

- Những lợi ích của tính toán đồng thời :

  - Thông lượng chương trình tăng lên — việc thực hiện song song một chương trình đồng thời cho phép số tác vụ được hoàn thành trong một

thời gian nhất định tăng tương ứng với số bộ xử lý theo định luật Gustafson

- Khả năng đáp ứng cao đối với đầu vào / đầu ra — các chương trình chuyên sâu về đầu vào / đầu ra chủ yếu đợi các hoạt động đầu vào hoặc đầu ra hoàn thành. Lập trình đồng thời cho phép thời gian chờ đợi được sử dụng cho một nhiệm vụ khác.
- Cấu trúc chương trình thích hợp hơn — một số vấn đề và miền vấn đề rất phù hợp để biểu diễn dưới dạng các nhiệm vụ hoặc quy trình đồng thời.

### 4. Distributed paradigm

Thông thường mã lệnh của một chương trình khi thực thi được tập trung trên cùng một máy, đây là cách lập trình truyền thống. Sự phát triển như vũ bão của mạng máy tính đặc biệt là mạng Internet toàn cầu, đã khiến các chương trình truyền thống này không còn đáp ứng được yêu cầu nữa. Các chương trình bây giờ yêu cầu phải có sự hợp tác xử lý, tức là mã lệnh của nó đã không tập trung trên một máy mà được phân tán trên nhiều máy. Khi một ứng dụng có mã lệnh thực thi được phân tán trên nhiều máy thì chương trình đó được gọi là chương trình phân tán và việc lập trình để tạo ra các chương trình này được gọi là lập trình phân tán. Có rất nhiều công nghệ lập trình phân tán như: DCOM, CORBA, RMI, EJB.. trong đó RMI là công nghệ thuần Java và dễ lập trình nhất.

- Có các mô hình lập trình phân tán như :
  - Mô hình client/server .
  - Mô hình tác tử di động .
  - Mô hình dựa trên thành phần , mô hình đối tượng phân tán .

Những hệ thống phân tán lan rộng đầu tiên đó là những mạng local – area networks như Ethernet được phát minh vào khoảng năm 1970 . Nghiên cứu về tính toán phân tán đã trở thành một bộ phận của nganh khoa học máy tính vào cuối những năm 1970 và đầu những năm 1980 .

Có hai lý do để chúng ta sử dụng hệ thống phân tán và tính toán phân tán :

- Tính tự nhiên của ưng dụng có thể yêu cầu sử dụng mọt mạng thong tin mà nối một số máy tính .
- Có lợi vì lý do thực tế . Ví dụ có thể tăng hiệu quả , giảm chi phí với việc sử dụng nhiều máy tinh cấp thấp so với việc sử dụng một máy tính cấp cao duy nhất .

### 5. Service-oriented paradig

Lập trình hướng dịch vụ (SOP) là một mô hình lập trình sử dụng "dịch vụ" làm đơn vị hoạt động của máy tính, để thiết kế và triển khai các ứng dụng kinh doanh tích hợp và các chương trình phần mềm quan trọng . Các dịch vụ có thể đại diện cho các bước của quy trình kinh doanh và do đó một trong những ứng dụng chính của mô hình này là phân phối hiệu quả chi phí các ứng dụng kinh doanh độc lập hoặc tổng hợp có thể "tích hợp từ trong ra ngoài".

Các công cụ thiết kế ngữ nghĩa và nền tảng tự động hóa thời gian chạy có thể được xây dựng để hỗ trợ các khái niệm cơ bản của SOP. Ví dụ, một máy ảo dịch vụ (SVM) tự động tạo các đối tượng dịch vụ như các đơn vị công việc và quản lý ngữ cảnh của chúng có thể được thiết kế để chạy dựa trên siêu dữ liệu chương trình SOP được lưu trữ trong XML và được tạo bởi một công cụ tự động hóa thời gian thiết kế. Theo thuật ngữ SOA, SVM vừa là nhà sản xuất dịch vụ vừa là người tiêu dùng dịch vụ.

- Các khái niệm cơ bản:
  - Đóng gói
  - Lớp
  - Kế thừa
  - Trừu tượng
  - Đa hình

## II. Imperative paradigm

### 1. Giới thiệu

#### 1.1.Lập trình mệnh lệnh (imperative paradigm)

**first do this and next do that**

Lập trình mệnh lệnh (từ tiếng Latinh imperare = command) là mô hình lập trình lâu đời nhất. Một chương trình dựa trên mô hình này được tạo thành từ một chuỗi hướng dẫn được xác định rõ ràng cho máy tính.

Do đó, mã nguồn cho các ngôn ngữ mệnh lệnh là một chuỗi các lệnh, chỉ định những gì máy tính phải làm - và khi nào - để đạt được kết quả mong muốn. Giá trị được sử dụng trong các biến được thay đổi trong thời gian chạy chương trình. Để

điều khiển các lệnh, các cấu trúc điều khiển như vòng lặp hoặc các nhánh được tích hợp vào mã . Lập trình mệnh lệnh đòi hỏi sự hiểu biết về các chức năng cần thiết để giải quyết một vấn đề, thay vì phụ thuộc vào các mô hình có thể giải quyết vấn đề đó. Trọng tâm của lập trình mệnh lệnh là cách giải quyết vấn đề, điều này cần có hướng dẫn chi tiết từng bước. Bởi vì mã được viết thực hiện các chức năng thay vì các mô hình, lập trình viên phải viết mã từng bước.

Các ngôn ngữ lập trình mệnh lệnh rất cụ thể và hoạt động là hướng hệ thống. Một mặt, mã dễ hiểu; mặt khác, nhiều dòng văn bản nguồn được yêu cầu để mô tả những gì có thể đạt được với một phần nhỏ các lệnh sử dụng ngôn ngữ lập trình khai báo. Các ngôn ngữ lập trình hướng đối tượng và thủ tục ( OOP ) thuộc loại lập trình mệnh lệnh, một số ngôn ngữ lập trình phổ biến hiện nay:

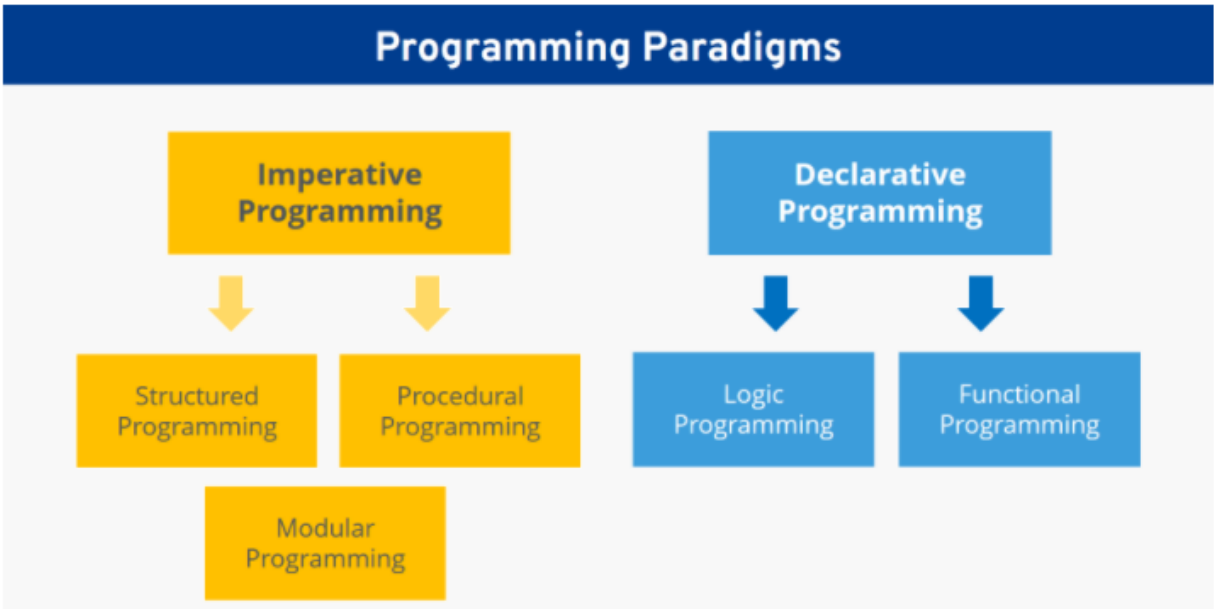
- FORTRAN
- Java
- C, C++
- Pascal
- Python
- C#, Assembler,…

Các ngôn ngữ lập trình mệnh lệnh khác nhau có thể được chỉ định cho ba kiểu lập trình cấp dưới khác - có cấu trúc, thủ tục và mô-đun (structured, procedural, and modular).

+ Các kiểu lập trình có cấu trúc mở rộng nguyên tắc bắt buộc cơ bản với cụ thể cấu trúc điều khiển: chuỗi, lựa chọn, và lặp đi lặp lại. Cách tiếp cận này dựa trên mong muốn hạn chế hoặc tránh hoàn toàn các câu lệnh nhảy làm cho mã được thiết kế theo thứ bậc trở nên phức tạp không cần thiết.

+ Cách tiếp cận thủ tục chia nhiệm vụ mà một chương trình phải thực hiện thành các nhiệm vụ con nhỏ hơn, được mô tả riêng trong mã. Điều này dẫn đến các mô-đun lập trình cũng có thể được sử dụng trong các chương trình khác.

+ Các mô hình lập trình mô-đun đi thêm một bước nữa bằng cách thiết kế, phát triển và thử nghiệm các thành phần chương trình cá nhân độc lập với nhau. Các mô-đun riêng lẻ sau đó được kết hợp để tạo ra phần mềm thực tế.




Lập trình mệnh lệnh là một trong hai mô hình lập trình cơ bản trong khoa học máy tính.

### 1.2.Lịch sử ra đời và phát triển

Các ngôn ngữ mệnh lệnh sớm nhất là ngôn ngữ máy của máy tính nguyên thủy. Trong các ngôn ngữ này, các hướng dẫn rất đơn giản, giúp việc triển khai phần cứng dễ dàng hơn nhưng lại cản trở việc tạo ra các chương trình phức tạp. FORTRAN , được phát triển bởi John Backus tại International Business Machines (IBM) bắt đầu từ năm 1954, là ngôn ngữ lập trình chính đầu tiên loại bỏ những trở ngại của mã máy trong việc tạo ra các chương trình phức tạp. FORTRAN là một ngôn ngữ được biên dịch cho phép các biến được đặt tên, các biểu thức phức tạp, chương trình con và nhiều tính năng khác hiện phổ biến trong các ngôn ngữ mệnh lệnh.

Hai thập kỷ tiếp theo chứng kiến sự phát triển của nhiều ngôn ngữ lập trình mệnh lệnh cấp cao khác. Vào cuối những năm 1950 và 1960,ALGOL được phát triển để cho phép các thuật toán toán học được thể hiện dễ dàng hơn và thậm chí được dùng làm ngôn ngữ đích của hệ điều hành cho một số máy tính.

MUMPS (1966) đã đưa mô hình mệnh lệnh đến một cực điểm logic, bằng cách hoàn toàn không có bất kỳ câu lệnh nào, hoàn toàn dựa vào các lệnh, thậm chí đến mức làm cho các lệnh IF và ELSE độc lập với nhau, chỉ được kết nối bởi một biến bên trong có tên là $TEST. COBOL (1960) và BASIC (1964) đều cố gắng làm cho cú pháp lập trình giống tiếng Anh hơn.

Trong những năm 1970, Pascal được phát triển bởi Niklaus Wirth , và C được tạo ra bởi Dennis Ritchietrong khi ông ấy đang làm việc tại Phòng thí nghiệm Bell . Wirth tiếp tục thiết kế Modula-2 và Oberon . Đối với nhu cầu của Bộ Quốc phòng Hoa Kỳ , Jean Ichbiah và một nhóm tại Honeywell bắt đầu thiết kế Ada vào năm 1978, sau một dự án kéo dài 4 năm để xác định các yêu cầu đối với ngôn ngữ này. Đặc điểm kỹ thuật được xuất bản lần đầu tiên vào năm 1983, với các bản sửa đổi vào các năm 1995, 2005 và 2012.

Những năm 1980 chứng kiến sự phát triển nhanh chóng trong quan tâm đến lập trình hướng đối tượng . Những ngôn ngữ này là bắt buộc về kiểu dáng, nhưng đã thêm các tính năng để hỗ trợ các đối tượng . Hai thập kỷ cuối của thế kỷ 20 chứng kiến sự phát triển của nhiều ngôn ngữ như vậy. Smalltalk -80, ban đầu được Alan Kay hình thành vào năm 1969, được phát hành vào năm 1980, bởi Trung tâm Nghiên cứu Xerox Palo Alto ( PARC ). Rút ra từ các khái niệm trong một ngôn ngữ hướng đối tượng khác— Simula (được coi là ngôn ngữ lập trình hướng đối tượng đầu tiên trên thế giới , được phát triển vào những năm 1960).

Bjarne Stroustrup đã thiết kế C ++ , một ngôn ngữ hướng đối tượng dựa trên C. Thiết kế của C ++ bắt đầu vào năm 1979 và việc triển khai đầu tiên được hoàn thành vào năm 1983. Vào cuối những năm 1980 và 1990, các ngôn ngữ mệnh lệnh đáng chú ý dựa trên các khái niệm hướng đối tượng là Perl , được phát hành bởi Larry Wall vào năm 1987; Python , do Guido van Rossum phát hành năm 1990; Visual Basic và Visual C ++ (bao gồm Microsoft Foundation Class Library (MFC) 2.0), được Microsoft phát hành lần lượt vào năm 1991 và 1993; PHP , được phát hành bởi Rasmus Lerdorf vào năm 1994; Java , của James Gosling ( Sun Microsystems) vào năm 1995, JavaScript , của Brendan Eich ( Netscape ) và Ruby , của Yukihiro "Matz" Matsumoto, đều được phát hành vào năm 1995. .NET Framework (2002) của Microsoft là cốt lõi của nó, cũng như các ngôn ngữ đích chính của nó, VB. NET và C # chạy trên nó; tuy nhiên F # của Microsoft , một ngôn ngữ chức năng, cũng chạy trên nó.

### 1.3. Một số ví dụ về ngôn lập trình mệnh lệnh

Các ngôn ngữ lập trình mệnh lệnh được đặc trưng bởi bản chất hướng dẫn của chúng và do đó, yêu cầu nhiều dòng mã hơn đáng kể để diễn đạt những gì có thể được mô tả chỉ với một vài hướng dẫn trong kiểu khai báo. Trong ví dụ sau, mục đích là xuất ra một danh sách các tên:

Lập trình mệnh lệnh (PHP)

```
1        $participantlist = [1 => 'Peter', 2 => 'Henry', 3 => 'Sarah'];
2        $firstnames= [];
3        foreach ($participantlist as $id => $name) {
4           $firstnames[] = $name;
5        }
```

- Ngôn ngữ C

```
// average of five number in C

int marks[5] = { 12, 32, 45, 13, 19 } int sum = 0;
float average = 0.0;
for (int i = 0; i < 5; i++) {
    sum = sum + marks[i];
}
average = sum / 5;
```

- Mô hình lập trình theo thủ tục (C++)

```
#include <iostream>
using namespace std;
int main()
{
    int i, fact = 1, num;
    cout << "Enter any Number: ";
    cin >> number;
    for (i = 1; i <= num; i++) {
        fact = fact * i;
    }
    cout << "Factorial of " << num << " is: " << fact << endl;
    return 0;
}
```

- Lập trình hướng đối tượng (Java)

```
import java.io.*;

class GFG {
    public static void main(String[] args)
    {
        System.out.println("GfG!");
        Signup s1 = new Signup();
        s1.create(22, "riya", "riya2@gmail.com", 'F', 89002);
    }
}

class Signup {
    int userid;
    String name;
    String emailid;
    char sex;
    long mob;

    public void create(int userid, String name,
                       String emailid, char sex, long mob)
    {
        System.out.println("Welcome to
                 GeeksforGeeks\nLets create your account\n");
        this.userid = 132;
        this.name = "Radha";
        this.emailid = "radha.89@gmail.com";
        this.sex = 'F';
        this.mob = 900558981;
        System.out.println("your account has been created");
    }
}
```

- Cách tiếp cận xử lý song song

Xử lý song song là xử lý các lệnh chương trình bằng cách chia chúng cho nhiều bộ xử lý. Một hệ thống xử lý song song có nhiều bộ xử lý với mục tiêu chạy một chương trình trong thời gian ngắn hơn bằng cách chia chúng. Cách tiếp cận này có vẻ giống như phân chia và chinh phục.

## 2. Cơ sở lý luận và cơ sở của lập trình mệnh lệnh

Việc triển khai phần cứng của hầu hết tất cả các máy tính là bắt buộc. Gần như tất cả phần cứng máy tính được thiết kế để thực thi mã máy , mã này có nguồn gốc từ máy tính và được viết theo kiểu mệnh lệnh.

Từ quan điểm cấp thấp này, trạng thái chương trình được xác định bởi nội dung của bộ nhớ và các câu lệnh là các lệnh bằng ngôn ngữ máy gốc của máy tính. Các ngôn ngữ mệnh lệnh cấp cao hơn sử dụng các biến và các câu lệnh phức tạp hơn, nhưng vẫn tuân theo cùng một mô hình. Công thức nấu ăn và danh sách kiểm tra quy trình , không phải chương trình máy tính, cũng là những khái niệm quen thuộc có phong cách tương tự như lập trình mệnh lệnh; mỗi bước là một chỉ dẫn, và thế giới vật chất giữ trạng thái. Vì các ý tưởng cơ bản của lập trình mệnh lệnh đều quen thuộc về mặt khái niệm và được thể hiện trực tiếp trong phần cứng, nên hầu hết các ngôn ngữ máy tính đều theo phong cách mệnh lệnh.

Các câu lệnh gán , trong mô hình mệnh lệnh, thực hiện một thao tác trên thông tin nằm trong bộ nhớ và lưu trữ kết quả trong bộ nhớ để sử dụng sau này. Ngoài ra, các ngôn ngữ mệnh lệnh cấp cao cho phép đánh giá các biểu thức phức tạp , có thể bao gồm sự kết hợp của các phép toán số học và đánh giá hàm , và việc gán giá trị kết quả cho bộ nhớ. Câu lệnh lặp (như trong vòng lặp while , vòng lặp do while và vòng lặp for) cho phép một chuỗi các câu lệnh được thực thi nhiều lần. Các vòng lặp có thể thực hiện các câu lệnh mà chúng chứa một số lần được xác định trước hoặc chúng có thể thực thi chúng lặp lại cho đến khi một số điều kiện thay đổi.

Các câu lệnh rẽ nhánh có điều kiện cho phép một chuỗi các câu lệnh chỉ được thực hiện nếu một số điều kiện được đáp ứng. Nếu không, các câu lệnh bị bỏ qua và trình tự thực thi tiếp tục từ câu lệnh theo sau chúng. Các câu lệnh rẽ nhánh không điều kiện cho phép một chuỗi thực thi được chuyển sang một phần khác của chương trình. Chúng bao gồm bước nhảy (được gọi là goto trong nhiều ngôn ngữ), chuyển đổi và chương trình con, chương trình con hoặc lệnh gọi thủ tục (thường quay trở lại câu lệnh tiếp theo sau cuộc gọi).

Trong giai đoạn đầu của sự phát triển của các ngôn ngữ lập trình cấp cao , sự ra đời của khối cho phép xây dựng các chương trình trong đó một nhóm các câu lệnh và khai báo có thể được coi như thể chúng là một câu lệnh. Điều này, cùng với sự ra đời của các chương trình con , cho phép các cấu trúc phức tạp được thể hiện bằng cách phân rã phân cấp thành các cấu trúc thủ tục đơn giản hơn.

Nhiều ngôn ngữ lập trình mệnh lệnh (chẳng hạn như Fortran , BASIC và C ) là sự trừu tượng của hợp ngữ .

### 3. Lập trình mệnh lệnh và lập trình khai báo

Các ngôn ngữ lập trình mệnh lệnh khác với các ngôn ngữ khai báo ở một điểm cơ bản: lập trình mệnh lệnh tập trung vào "cách thức", lập trình khai báo về "cái gì".

Có một câu nói về 2 khái niệm này: Lập trình mệnh lệnh **(Imperative programming)** nhấn mạnh tới việc chúng ta làm **như thế nào**, còn lập trình khai báo **(Declarative programming)** nhấn mạnh việc chúng ta làm **cái gì**.

Có thể lấy ví dụ thực tế như sau, một cặp vợ chồng đến nhà hàng, người bồi bàn muốn hỏi họ ngồi ở đâu, họ sẽ trả lời:

- Hướng mệnh lệnh **(Imperative, như thế nào):** tôi thấy bàn kia còn trống và sát cửa sổ, nơi có view nhìn đẹp, nên tôi và chồng tôi sẽ tới đó ngồi.
- Hướng khai báo **(Declarative, cái gì):** cho tôi bàn 2 người.

Chúng ta có thể để ý thấy là khi đưa ra phép khai báo, thông thường đã có phép mệnh lệnh ẩn bên trong. Ví dụ như khi gọi bàn 2 người, thì cặp vợ chồng này mặc nhiên cho rằng bồi bàn biết được nên chọn bàn nào và hướng dẫn khách của mình tới bàn đó như thế nào.

Ví dụ SQL và HTML

SQL

```sql
SELECT * FROM Users WHERE Country='Vietnam';
```

HTML

```html
<article>
    <header>
        <h1>Declarative Programming</h1>
    </header>
</article>
```

Chúng ta có thể thấy, trong cả 2 ngôn ngữ trên thì chúng ta quan tâm nhiều hơn đến kết quả sẽ cho ra **cái gì,** chứ không quan tâm đến việc chúng **làm thế nào** để ra kết quả đó.

Trong các ngôn ngữ khai báo, mã nguồn vẫn rất trừu tượng về mặt thủ tục cụ thể. Để đi đến giải pháp, một thuật toán được sử dụng để tự động xác định và áp dụng các phương pháp thích hợp. Cách tiếp cận này có nhiều ưu điểm: Các chương trình có thể được **viết nhanh hơn nhiều** và các ứng dụng cũng rất dễ tối ưu hóa. Nếu một phương pháp mới được phát triển trong tương lai, các hướng dẫn trừu tượng trong mã nguồn có nghĩa là thuật toán có thể dễ dàng sử dụng phương pháp mới hơn.

### 4. Ưu điểm, nhược điểm của lập trình mệnh lệnh

Nhiều ngôn ngữ lập trình dựa trên mô hình lập trình mệnh lệnh đang được sử dụng ngày nay.

Một mặt, điều này là do cách tiếp cận là **hình thức lập trình ban đầu** . Mặt khác, bất chấp sự tồn tại của các mô hình thay thế, mô hình mệnh lệnh vẫn có một số ưu điểm thực tế.

Các ngôn ngữ này **tương đối dễ học** , vì mã có thể được đọc giống như hướng dẫn từng bước. Do đó, các lập trình viên thường học một ngôn ngữ bắt buộc đầu tiên như một phần trong quá trình đào tạo của họ.

**Tính dễ đọc** là một yếu tố quan trọng trong hoạt động hàng ngày. Cuối cùng, việc bảo trì và tối ưu hóa các ứng dụng không nên được liên kết với một người cụ thể; các nhân viên khác nhau sẽ có thể làm điều đó mà không gặp quá nhiều khó khăn ngay cả khi họ chưa tự viết mã từ đầu.

Một nhược điểm của lập trình thủ tục là đối với các vấn đề phức tạp hơn cần được giải quyết, số lượng mã nhanh chóng bắt đầu tăng lên. Nó vẫn dễ đọc nhưng **trở nên khó hiểu do khối lượng của nó** .

Việc thực thi không được phân định rõ ràng khỏi chương trình vì nó ở trong kiểu khai báo, do đó, các can thiệp tiếp theo có thể tạo ra các lỗi không mong muốn. Các phần mở rộng cũng khó triển khai hơn trong mã mệnh lệnh thuần túy -

không giống như trong mô hình khai báo, nơi có các phương thức có thể được sử dụng để thêm chúng một cách riêng biệt.

❖ Ưu điểm

+ Dễ học, dễ gỡ lỗi

+ Mô hình khái niệm (đường dẫn giải pháp) rất dễ hiểu cho người mới bắt đầu

+ Có thể tính đến các đặc điểm của các ứng dụng cụ thể

+ Mọi người thường được dạy hầu hết về lập trình mệnh lệnh ở trường học và hầu hết các nơi làm việc đều sử dụng nó, vì vậy mức độ quen thuộc cao.

❖ Nhược điểm

+ Mã nhanh chóng trở nên rất rộng rãi và do đó gây nhầm lẫn

+ Rủi ro lỗi cao hơn khi chỉnh sửa

+ Lập trình hướng hệ thống có nghĩa là bảo trì ngăn chặn việc phát triển ứng dụng

+ Tối ưu hóa và mở rộng khó hơn

Trong thực tế, các dạng hỗn hợp của mô thức lập trình thường được sử dụng ngày nay vì cả hai kiểu lập trình mệnh lệnh và khai báo đều có những ưu điểm và nhược điểm của chúng. Tuy nhiên, phong cách lập trình khai báo ngày càng chiếm ưu thế và được bổ sung bởi các phương pháp lập trình mệnh lệnh.

## 5. Tài liệu tham khảo

https://whatis.techtarget.com/definition/imperative-programming

https://phocode.com/haskell/haskell-lap-trinh-menh-lenh-vs-lap-trinh-khai-bao/

https://www.geeksforgeeks.org/introduction-of-programming-paradigms/

https://www.ionos.com/digitalguide/websites/web-development/imperative-programming/

https://www.quora.com/What-are-the-pros-and-cons-of-imperative-programming

https://en.wikipedia.org/wiki/Imperative_programming#Rationale_and_foundations_of_imperative_programming