

ĐẠI HỌC QUỐC GIA TP. HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN



BÁO CÁO ĐỒ ÁN
MÔN HỌC CS111 - NGUYÊN LÝ VÀ PHƯƠNG PHÁP LẬP TRÌNH
ĐỀ TÀI: CÁC PHƯƠNG PHÁP ĐÁNH GIÁ
NGÔN NGỮ LẬP TRÌNH

Giáo viên hướng dẫn: Thầy Trịnh Quốc Sơn

Sinh viên thực hiện: Đặng Phước Sang

MSSV: 21521377

TP. Hồ Chí Minh, tháng 6 năm 2023

This image shows a full page of a document template designed for handwritten notes or essays. It features approximately 28 evenly spaced, horizontal black dotted lines across the entire width of the page. The background is plain white, providing a clear space for writing. There are no margins, headers, footers, or other markings present on the page.

Người nhận xét

MỤC LỤC

CHƯƠNG I: GIỚI THIỆU

- 1. Giới thiệu đề tài.....4
- 2. Đối tượng nghiên cứu.....4

CHƯƠNG II: CÁC TIÊU CHÍ ĐÁNH GIÁ NGÔN NGỮ LẬP TRÌNH

- 1. Các tiêu chí về mặt ngôn ngữ.....5
 - 1.1 Tính dễ đọc.....5
 - 1.2 Tính dễ viết.....9
 - 1.3 Độ tin cậy.....11
 - 1.4 Chi phí.....12
 - 1.5 Một số chi phí khác.....14
- 2. Các tiêu chí về mặt lập trình, ứng dụng.....14

CHƯƠNG III: ĐÁNH GIÁ, SO SÁNH CÁC NGÔN NGỮ LẬP TRÌNH HIỆN NAY

- 1. So sánh về bản chất ngôn ngữ.....15
- 2. So sánh về mặt ứng dụng.....15
 - 2.1 Ứng dụng lập trình Web.....15
 - 2.2 Ứng dụng lập trình Game.....16
 - 2.3 Ứng dụng lập trình AI.....16
- 3. Đánh giá chung.....17

CHƯƠNG IV: CÁC YẾU TỐ TÁC ĐỘNG, CHI PHỐI VIỆC THIẾT KẾ NGÔN NGỮ LẬP TRÌNH

- 1. Kiến trúc máy tính.....18
- 2. Hệ phương pháp lập trình.....19

CHƯƠNG V: KẾT LUẬN

TÀI LIỆU THAM KHẢO

CHƯƠNG I: GIỚI THIỆU

1. Giới thiệu đề tài

Ngôn ngữ lập trình là công cụ cần thiết cho các nhà phát triển trong việc tạo ra các ứng dụng phần mềm, phát triển các dịch vụ và sản phẩm kỹ thuật số, quản lý dữ liệu, nghiên cứu và triển khai các mô hình trí tuệ nhân tạo, ...

Có rất nhiều ngôn ngữ lập trình với nhiều đặc điểm khác nhau về cú pháp, hiệu năng, phong cách lập trình.

Ngoài ra, mỗi ngôn ngữ lập trình còn hỗ trợ các tính năng, thư viện đặc biệt phù hợp cho một lĩnh vực nhất định. Ví dụ: Python là ngôn ngữ phù hợp để quản lý dữ liệu và nghiên cứu về các lĩnh vực Máy học hay Trí tuệ nhân tạo nhờ có một số lượng lớn thư viện hỗ trợ. C/C++ thường được sử dụng trong lập trình nhúng nhờ vào tốc độ xử lý cao.

Với nhu cầu ngày càng cao, các ngôn ngữ lập trình ngày càng được mở rộng và phát triển. Sự phong phú và đa dạng của ngôn ngữ lập trình đòi hỏi cần có một số tiêu chí để các lập trình viên lựa chọn ngôn ngữ lập trình phù hợp với công việc.

2. Đối tượng nghiên cứu

Đối tượng nghiên cứu chủ yếu của đề án là các ngôn ngữ lập trình, nội dung tìm hiểu bao gồm:

- Các tiêu chí đánh giá một ngôn ngữ lập trình về mặt ngôn ngữ và mặt ứng dụng. (Chương 2)
- So sánh, đánh giá giữa các ngôn ngữ lập trình phổ biến hiện nay dựa trên các tiêu chí trên. (Chương 3)
- Các yếu tố tác động, chi phối việc thiết kế một ngôn ngữ lập trình. (Chương 4)

CHƯƠNG II: CÁC TIÊU CHÍ ĐÁNH GIÁ NGÔN NGỮ LẬP TRÌNH

Tiêu chí đánh giá ngôn ngữ lập trình là bộ tiêu chuẩn và nguyên tắc dùng để đánh giá chất lượng, hiệu quả của ngôn ngữ lập trình.

Các tiêu chí này được sử dụng để đánh giá các tính năng, thiết kế, cú pháp và ngữ nghĩa của ngôn ngữ lập trình nhằm xác định tính phù hợp của chúng đối với các ứng dụng và ngữ cảnh khác nhau.

1. Các tiêu chí về mặt ngôn ngữ

Tiêu chí đánh giá về mặt ngôn ngữ là bộ tiêu chuẩn và nguyên tắc đánh giá bản chất ngôn ngữ lập trình.

Các tiêu chuẩn chính bao gồm: tính dễ đọc (readability), tính dễ viết (writability), độ tin cậy (reliability) và chi phí (cost).

1.1 Tính dễ đọc (Readability)

Tính dễ đọc: là khả năng đọc và hiểu dễ dàng mã được viết bằng một ngôn ngữ cụ thể.

Một ngôn ngữ lập trình dễ đọc có cú pháp dễ hiểu, sử dụng các quy ước đặt tên thông dụng và có ý nghĩa, các đoạn mã có cấu trúc tốt. Trong khi đó, một ngôn ngữ lập trình có tính dễ đọc kém thường dẫn đến sự nhầm lẫn về ngữ nghĩa và tạo ra các lỗi gây khó khăn trong việc chỉnh sửa, quản lý và mở rộng.

Tính dễ đọc của ngôn ngữ mang lại rất nhiều lợi ích:

- Giúp nhà phát triển viết mã dễ dàng và hiệu quả hơn.
- Giảm khả năng gây ra lỗi, dễ gỡ lỗi hơn và tăng độ tin cậy của mã.
- Trong một dự án lớn, mã có thể được viết bởi nhiều người, tính dễ đọc cho phép họ hiểu dễ dàng.

1.1.1 Sự giản dị tổng thể (Overall Simplicity)

- Là tập hợp các tính năng (hay các đặc trưng) và cấu trúc có thể quản lý được. Các tính năng và cấu trúc được xây dựng cần có sự tương đồng và quen thuộc với người đọc, nhằm giúp người đọc dễ hiểu hơn.

- Cần có sự tối thiểu về đa tính năng. Đa tính năng thể hiện nhiều phương thức cho một phép tính.

Ví dụ: Trong C++, có nhiều phương thức để tăng giá trị của một biến lên 1 đơn vị:

```
count = count + 1;  
count += 1;  
count++;  
++count;
```

Khi một ngôn ngữ có quá nhiều đa tính năng có thể gây khó khăn cho người đọc, làm giảm tính dễ đọc của ngôn ngữ.

- Cần có sự tối thiểu về nạp chồng toán tử.

Ví dụ: Trong Python, toán tử + có thể dùng để cộng 2 số nguyên, cộng 2 số thực, cộng 2 chuỗi kí tự, ...

Nạp chồng toán tử giúp đơn giản hóa ngôn ngữ lập trình, giúp giảm bớt số lượng toán tử và số lượng phương thức được định nghĩa, tuy nhiên chúng có thể làm suy giảm tính dễ đọc nếu bị lạm dụng quá mức.

Tóm lại, sự giản dị cải thiện khả năng dễ đọc của ngôn ngữ lập trình, tuy vậy sự giản dị quá mức lại làm cho ngôn ngữ trở nên khó đọc.

Ví dụ: Hợp ngữ được xây dựng với các câu lệnh rất đơn giản, nhưng lại rất khó đọc vì chúng thiếu đi các câu lệnh điều khiển phức tạp, dẫn đến cấu trúc ít rõ ràng hơn. Việc đọc và hiểu hợp ngữ trở nên cực kỳ khó khăn.

1.1.2 Tính trực giao (Orthogonality)

- Là một tập hợp nhỏ các cấu trúc nguyên thủy có thể kết hợp được với nhau. Số cách kết hợp nên được tối thiểu.

- Mọi sự kết hợp đều hợp lệ và có nghĩa.

Ví dụ: Một ngôn ngữ có 4 kiểu dữ liệu nguyên thủy là int, float, double, char, và 2 cấu trúc mảng và con trỏ. Ta có thể kết hợp chúng với nhau theo nhiều cách:

```
int *p;  
int **p_to_p;  
double a[100];  
double a[100][100];  
char *k[100];
```

Việc thiếu tính trực giao có tạo ra nhiều ngoại lệ có thể ảnh hưởng tính dễ đọc của ngôn ngữ lập trình, ví dụ như: Con trỏ có thể trỏ đến bất kì loại biến hay

cấu trúc dữ liệu nào. Nếu con trỏ không thể trỏ tới mảng, sẽ có rất nhiều khả năng kết hợp bị loại bỏ.

Tính trực giao có mối quan hệ chặt chẽ với tính đơn giản. Một ngôn ngữ thiết kế có tính trực giao càng cao thì các quy tắc ngôn ngữ càng ít ngoại lệ, từ đó giúp ngôn ngữ dễ đọc và dễ hiểu hơn. Tuy nhiên, cũng giống như tính đơn giản, tính trực giao quá mức có thể khiến một ngôn ngữ trở nên phức tạp.

1.1.3 Các lệnh điều khiển (Control Statements)

- Cần có các lệnh điều khiển thông dụng (if, else, for, while, ...)
- Cần hạn chế sử dụng lệnh goto.

Các lệnh điều khiển thông dụng hiện nay cho phép chuyển đổi câu lệnh liên kế nhau, giúp chúng ta dễ dàng theo dõi cách các lệnh này hoạt động.

Các ngôn ngữ Basic và Fortran không hỗ trợ các câu lệnh điều khiển và sử dụng goto, dẫn đến việc đọc ngôn ngữ này vô cùng khó khăn.

Ví dụ: So sánh giữa 2 đoạn mã viết bằng C++ và Fortran. Ta có thể hình dung dễ dàng cách hoạt động của đoạn mã C++ do chúng nằm liền kề nhau từ trên xuống.

FORTRAN	C++
<pre>CHARACTER REPLY*4 DO 216 I=3,50,8 PRINT*, 'I=', I PRINT*, 'Wanna see I+8?' READ (*,*) REPLY <i>strings are read like any other variable.</i> IF (REPLY.NE. 'yes') GO TO 77 PRINT*, I+8 J=I/4 IF (J.GT.1.AND.J.LT.3) GOTO 216 PRINT*, 'J =', J 216 CONTINUE 77 CONTINUE</pre>	<pre>#include <strings.h> char reply[5]; for (i=3;i<51;i+=8){ cout<<"i="<<i<<endl; cout<<"Wanna see i+8?"<<endl; <i>//strings are read with cin.getline</i> cin.getline(reply,sizeof(reply)); if (strcmp(reply,"yes")) break; cout<<i+8; j=i/4; if (j>1 && j<3) continue; cout<<"j =",j; }</pre>

Ngày nay hầu hết các ngôn ngữ đều hỗ trợ các lệnh điều khiển thông dụng và không hỗ trợ goto, vì vậy yếu tố này không còn ảnh hưởng nhiều đến tính dễ đọc của ngôn ngữ lập trình.

1.1.4 Các kiểu dữ liệu và cấu trúc (Data Types and Structures)

- Cần hỗ trợ đầy đủ các kiểu dữ liệu và các cấu trúc.
- Có các phương tiện hỗ trợ xác định kiểu dữ liệu và cấu trúc.

Ví dụ: Ngôn ngữ không hỗ trợ kiểu dữ liệu Boolean cần phải dùng kiểu dữ liệu khác như số nguyên (0 cho false, 1 là true) để biểu thị phép luận lý, trong khi các ngôn ngữ hỗ trợ kiểu dữ liệu này dễ dàng diễn đạt điều đó.

1.1.5 Cân nhắc cú pháp (Syntax Considerations)

- Mã định danh (Identifiers): là tên cung cấp cho biến, hàm, nhãn. Chúng cần có độ dài linh hoạt và rõ ràng về ý nghĩa.

Ví dụ:

```
void swap(int &a, int &b);  
int num_of_images;  
class Student{};  
float mark;
```

Hầu hết các ngôn ngữ không giới hạn độ dài của định danh, phân biệt chữ hoa và chữ thường, sử dụng dấu gạch dưới để phân cách các từ. Định danh thường không được phép trùng với từ khóa.

Ngôn ngữ SQL không phân biệt chữ hoa và chữ thường.

Fortran 77 quy định độ dài định danh là 6, do đó nhiều tên biến phải viết tắt, gây khó hiểu cho người đọc.

- Từ khóa (Keyword): từ có nghĩa đặc biệt trong một ngữ cảnh nhất định.

C Keywords			
auto	double	int	struct
break	else	long	switch
case	enum	register	typedef
char	extern	return	union
continue	for	signed	void
do	if	static	while
default	goto	sizeof	volatile
const	float	short	unsigned

- Từ dành riêng (Reserved word): từ đặc biệt không thể dùng để đặt tên.

Ví dụ: if, else, for, while, class, break, ...

Mọi keywords đều là reserved words, nhưng không phải tất cả reserved words đều là keywords.

Ví dụ:

Trong Java: goto là reserved words nhưng không phải là keywords.

Trong Fortran, Algol: không có reserved words, mọi keywords đều có thể dùng để đặt tên như identifiers. Điều này có thể khiến người đọc nhầm lẫn về ý nghĩa và bản chất của chúng.

- Hình thức và ý nghĩa (Forms and meaning): Cách thức thiết kế các câu lệnh sao cho vai trò của chúng được thể hiện rõ.

Để tăng tính dễ đọc của ngôn ngữ lập trình, ta cần sử dụng các quy ước đặt tên có ý nghĩa. Tên biến, hàm nên mang tính mô tả mục đích của biến hay hàm đó.

Cấu trúc về hình thức khi viết mã cũng rất quan trọng. Nó giúp các nhà phát triển nhanh chóng định vị và hiểu rõ vai trò của các thành phần khác nhau.

1.2 Tính dễ viết (Writability)

Tính dễ viết: là khả năng viết mã dễ dàng của một ngôn ngữ lập trình.

Một ngôn ngữ lập trình dễ viết có cú pháp ngắn gọn, từ khóa và cấu trúc dễ nhớ, và sự trừu tượng mạnh mẽ giúp dễ dàng diễn đạt các ý tưởng phức tạp. Đồng thời phải hỗ trợ tốt các chức năng gỡ lỗi và kiểm tra để đảm bảo mã được viết chính xác.

Tính dễ viết của ngôn ngữ mang lại rất nhiều lợi ích:

- Giúp nhà phát triển viết mã dễ dàng và hiệu quả hơn.
- Giảm khả năng gây ra lỗi, dễ gỡ lỗi hơn và tăng độ tin cậy của mã.
- Một ngôn ngữ có tính dễ viết cao cũng sẽ có tính dễ đọc cao.

1.2.1 Tính đơn giản và trực giao (Simplicity and Orthogonality)

- Nên có ít các đặc trưng (simplicity)
- Có sự kết hợp các đặc trưng một cách nhất quán để tạo ra đặc trưng mới (orthogonality)

Ngôn ngữ có tính đơn giản và trực giao tốt vừa dễ đọc vừa dễ viết.

1.2.2 Hỗ trợ trừu tượng hóa (Support for abstraction)

- Là khả năng cho phép bỏ qua các chi tiết khi xác định và sử dụng các cấu trúc phức tạp.

- Trừu tượng hóa quá trình: phân chia chương trình thành các chương trình con. Mỗi chương trình con đảm nhiệm một tác vụ nào đó và được đặc trưng bởi một cái tên.

Trừu tượng hóa quá trình cho phép gọi chương trình con nhiều lần mà không cần cài đặt lại. Chương trình chính trở nên rõ ràng và dễ đọc.

- Trừu tượng hóa dữ liệu: tạo ra kiểu dữ liệu trừu tượng (tập hợp các đặc tính và phép toán thao tác trên dữ liệu đó).

Ví dụ: kiểu `int` do ngôn ngữ C++ định nghĩa, có phạm vi -2147483648 tới 2147483647, bộ nhớ 4 byte, có các phép toán 1 ngôi, 2 ngôi và so sánh. Kiểu do người dùng định nghĩa (dùng `struct`): `Node`, `Linked List`, `Tree`, ...

Tóm lại, trừu tượng hóa cho phép biểu diễn các cấu trúc dữ liệu phức tạp, mô tả khái quát quá trình thực thi của thuật toán, giúp mã trở nên ngắn gọn và dễ hiểu. Vì vậy, ngôn ngữ hỗ trợ trừu tượng hóa có tính dễ viết cao, tính dễ đọc vì vậy cũng cải thiện.

1.2.3 Độ biểu hiện (Expressivity)

- Một ngôn ngữ dễ viết cần có tính đặc tả tính toán tương đối thuận tiện.

Ví dụ 1: Trong C++, cách ghi `a++` thuận tiện hơn so với `a = a + 1`

Ví dụ 2: So sánh giữa C++ và Python

```
print("Hello " * 3)
```

```
#include<iostream>
using namespace std;

int main()
{
    cout << "Hello Hello Hello";
}
```

Trong ví dụ 2, giả sử cần in ra 100 từ “Hello”, trong C++ ta phải sử dụng vòng lặp, trong khi đó với Python, ta chỉ cần thay đổi thành “Hello” * 100.

1.3 Độ tin cậy (Reliability)

Độ tin cậy: là khả năng tạo ra kết quả nhất quán và có thể dự đoán được, ngay cả khi xảy ra lỗi và các trường hợp không lường trước được.

Một ngôn ngữ lập trình có độ tin cậy tốt sẽ ít có khả năng tạo ra kết quả không mong muốn hay gặp sự cố, khiến ngôn ngữ này phù hợp để xây dựng các phần mềm đáng tin cậy và mạnh mẽ.

Độ tin cậy của ngôn ngữ mang lại rất nhiều lợi ích:

- Dễ dàng bảo trì, gỡ lỗi và mở rộng mã.
- Ít xảy ra các sự cố hay lỗi không mong muốn.
- Giảm các chi phí, thiệt hại gây ra do lỗi.

1.3.1 Kiểm tra kiểu (Type Checking)

- Khả năng kiểm tra loại lỗi trong một chương trình bởi trình biên dịch hoặc trong quá trình thực hiện chương trình.

- Quá trình phát hiện lỗi có thể diễn ra trong quá trình biên dịch hoặc chạy chương trình.

- Các lỗi trong chương trình được phát hiện càng sớm thì việc sửa chữa cần thiết càng ít tốn kém.

1.3.2 Xử lý ngoại lệ (Exception Handling)

- Chặn lỗi thời gian chạy (Run-time errors) và thực hiện các biện pháp khắc phục, sau đó tiếp tục thực hiện chương trình tương ứng.

Ví dụ: Python hỗ trợ Exception Handling với try, except, else, finally.

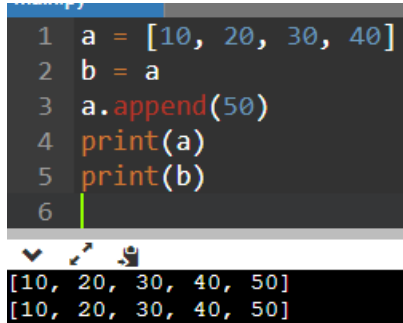
```
def divide(x,y):  
    try:  
        result = x/y  
    except ZeroDivisionError:  
        print("Số chia phải khác 0")  
    except TypeError:  
        print("Sai kiểu dữ liệu")  
    else:  
        print(f"Đáp án là: {result}")  
    finally:  
        print("Bye")
```

Xử lý ngoại lệ cho phép chương trình có thể thực thi các chương trình tương ứng hay vì dừng lại. Các biện pháp khắc phục này có thể giúp giảm bớt thời gian chạy lại.

1.3.3 Bí danh (Aliasing)

- Cho phép hai hay nhiều phương pháp tham chiếu riêng biệt cho cùng một vị trí bộ nhớ.

Ví dụ trong Python:



```
1 a = [10, 20, 30, 40]
2 b = a
3 a.append(50)
4 print(a)
5 print(b)
6
```

[10, 20, 30, 40, 50]
[10, 20, 30, 40, 50]

Đặt bí danh cho phép kiểm soát các mảng nhiều phần tử hiệu quả hơn thay vì sao chép chúng nhiều lần.

Tuy vậy chúng cũng là nguyên nhân gây ra nhiều lỗi, khó hiểu và khó tối ưu chương trình.

1.3.4 Tính dễ đọc và dễ viết (Readability and Writability)

- Một chương trình dễ đọc và dễ viết cho phép diễn đạt rõ ràng và tự nhiên các phương thức và thuật toán sẽ giúp tăng độ tin cậy của chương trình.

1.4 Chi phí (Cost)

Một số chi phí của ngôn ngữ lập trình:

- Chi phí đào tạo lập trình viên
- Chi phí viết chương trình
- Chi phí biên dịch chương trình
- Chi phí thực thi chương trình
- Chi phí triển khai hệ thống ngôn ngữ
- Chi phí do độ tin cậy thấp
- Chi phí duy trì chương trình

1.4.1 Chi phí đào tạo lập trình viên

- Là chi phí đào tạo để lập trình viên sử dụng một ngôn ngữ lập trình.

- Phụ thuộc tính dễ đọc và tính dễ viết của ngôn ngữ đó.
- Kinh nghiệm của lập trình viên cũng đóng vai trò quan trọng. Lập trình viên đã quen thuộc với một ngôn ngữ lập trình có thể học ngôn ngữ lập trình khác dễ dàng hơn.

1.4.2 Chi phí viết chương trình

- Phụ thuộc vào tính dễ viết của chương trình
- Nếu ngôn ngữ được thiết kế có mục đích gần với ứng dụng cụ thể, nó có thể giảm thời gian và chi phí viết chương trình.

1.4.3 Chi phí biên dịch chương trình

- Phụ thuộc vào chất lượng của trình biên dịch. Trình biên dịch chất lượng cao có thể giảm chi phí biên dịch.

1.4.4 Chi phí thực hiện chương trình

- Phụ thuộc vào thiết kế của ngôn ngữ lập trình
- Nếu một ngôn ngữ yêu cầu nhiều lần kiểm tra kiểu trong thời gian chạy, ngôn ngữ đó có thể làm chậm quá trình thực thi chương trình, bất kể chất lượng của trình biên dịch.

1.4.5 Chi phí triển khai hệ thống ngôn ngữ

- Nếu các hệ thống biên dịch/thông dịch miễn phí có sẵn ngay sau khi thiết kế của ngôn ngữ được phát hành, nó có thể giảm chi phí triển khai ngôn ngữ đó.

1.4.6 Chi phí do độ tin cậy thấp

- Chi phí của độ tin cậy kém có thể rất cao, đặc biệt là trong các hệ thống quan trọng như nhà máy điện hạt nhân hoặc thiết bị y tế.
- Có thể dẫn đến mất mát trong kinh doanh hoặc các vụ kiện về hệ thống phần mềm bị lỗi.

1.4.7 Chi phí duy trì chương trình

- Phụ thuộc vào tính dễ đọc của ngôn ngữ, một ngôn ngữ dễ đọc và dễ hiểu, nó có thể giảm chi phí duy trì các chương trình.

1.5 Một số tiêu chí khác

- Tính di động (Portability): các chương trình dễ dàng chuyển đổi từ triển khai này sang triển khai khác.

- Tính tổng quát (Generality): khả năng áp dụng cho nhiều ứng dụng.

- Tính xác định rõ (Well-definedness): các định nghĩa chính thức của ngôn ngữ đầy đủ và chính xác.

2. Các tiêu chí về mặt lập trình, ứng dụng

Các tiêu chí này rất đa dạng, bao gồm: khả năng bảo mật và an ninh, khả năng lập trình tính toán thuật toán phức tạp hoặc xây dựng hệ thống phân tán, hệ thống cơ sở dữ liệu, ...

Một số tiêu chí phổ biến:

- Tính bảo mật của ngôn ngữ lập trình: cần có định dạng kiểu dữ liệu rõ ràng, khả năng kiểm soát bộ nhớ, có cơ chế tự động dọn rác, ...

- Khả năng lập trình Web: có khả năng hỗ trợ xây dựng các ứng dụng web, bao gồm frontend và backend.

- Khả năng lập trình Game: cần có khả năng tính toán và xử lý tốt, tốc độ thực thi cao kết hợp với khả năng xử lý đồ họa.

- Khả năng lập trình AI: cần có khả năng tính toán và xử lý tốt, hỗ trợ các tính năng và thư viện phục vụ cho các thuật toán máy học, có thể biểu diễn các cấu trúc phức tạp như mạng neuron nhân tạo.

CHƯƠNG III: SO SÁNH, ĐÁNH GIÁ CÁC NGÔN NGỮ LẬP TRÌNH HIỆN NAY

1. So sánh về bản chất ngôn ngữ

Hầu hết các ngôn ngữ bậc cao đều thỏa mãn đầy đủ 4 tiêu chí: dễ đọc, dễ viết, tin cậy và chi phí thấp so với các ngôn ngữ bậc thấp.

Ví dụ: So sánh giữa C++ và Python

- Python có tính dễ đọc và dễ viết cao hơn so với C++, do ngôn ngữ này có cú pháp ngắn gọn, hỗ trợ đầy đủ các tính năng. Python dễ tiếp cận hơn cho người mới học, trong khi C++ yêu cầu lập trình viên cần tay nghề cao.

- C++ có tốc độ xử lý nhanh hơn, do C++ được trình biên dịch chuyển trực tiếp sang mã máy trong khi Python chọn cách diễn giải. Ngoài ra, C++ chạy trực tiếp trên ổ cứng thay vì chạy trên một môi trường máy ảo như Python.

- Python có thể được triển khai trên mọi hệ điều hành, trong khi C++ cần có trình biên dịch phù hợp cho mỗi hệ điều hành.

- C++ có cơ chế định dạng kiểu dữ liệu mạnh có thể đảm bảo độ bảo mật cao hơn, trong khi đó Python cung cấp kiểu dữ liệu động không đảm bảo độ bảo mật.

- Python có khả năng quản lý bộ nhớ tốt với Garbage Collectors, trong khi với C++, các cơ chế cấp phát và thu hồi bộ nhớ cần được lập trình chính xác để tránh các lỗi phát sinh.

2. So sánh về mặt ứng dụng

2.1 Ứng dụng lập trình Web

Các ngôn ngữ phổ biến như Java, Ruby, Python, JavaScript được ưa chuộng hơn, vì chúng cung cấp các tính năng cần thiết và phù hợp với lập trình web.

Ví dụ:

- Ruby cung cấp framework Ruby on Rails, dễ dàng kết nối với các cơ sở dữ liệu như MySQL, Oracle, ..., cung cấp cơ chế bảo mật tốt, có tính hướng đối tượng cao và có một hệ thống cộng đồng lớn.

- Python cung cấp rất nhiều framework cho lập trình web, bao gồm Django, Flask, Pyramid, ..., dễ dàng kết nối với các cơ sở dữ liệu và sở hữu cộng đồng lớn.

2.2 Ứng dụng lập trình Game

Phổ biến nhất là C#, C/C++, Java, Python

Ví dụ:

- C# là ngôn ngữ hướng đối tượng mạnh mẽ với tốc độ xử lý, tính toán nhanh, được ưa chuộng trong lập trình các ứng dụng trò chơi. Unity Game Engine cung cấp rất nhiều công cụ hữu ích để xây dựng và phát triển sản phẩm game bằng C# nhanh chóng.

- JavaScript có thể lập trình được game chạy ngay trên trình duyệt. Các games chạy trên các trang web nổi tiếng như y8.com, game24h.vn đa số được viết bằng ngôn ngữ này.

- Python cung cấp các thư viện như Pygame, Kivy, Pyglet cho phép dễ dàng tạo ra các sản phẩm game đơn giản, tuy nhiên do tốc độ xử lý chậm, ngôn ngữ này ít được dùng để tạo các game lớn có đồ họa cao.

2.3 Ứng dụng lập trình AI

Phổ biến nhất là Python, C++, Java, ...

Ví dụ:

- Python là ngôn ngữ phổ biến nhất để lập trình AI với số lượng thư viện khổng lồ:

- + Tính toán: Numpy, Scipy, ...
- + Xử lý dữ liệu: Pandas, Matplotlib, Seaborn, ...
- + Framework: Tensorflow, Keras, Pytorch, Sklearn, ...
- + Dữ liệu lớn: Vaex, Hadoop, Kafka, PySpark, ...

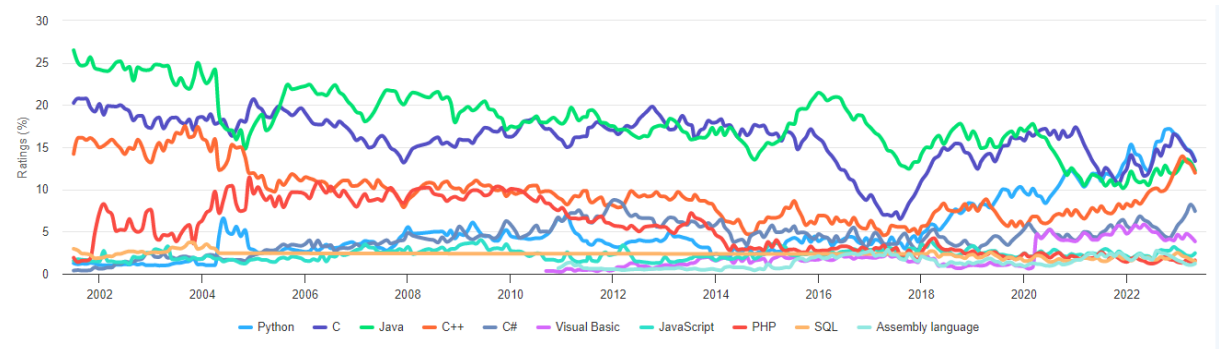
- C++ được ưa chuộng khi triển khai các mô hình AI. Các thư viện lớn về Deep Learning của Python thường được xây dựng bằng C++, ví dụ như Tensorflow (62.7%), PyTorch (52.4%).

- Java chủ yếu phát triển các ứng dụng AI chạy trên nền Server. Java có nhiều thư viện hỗ trợ để lập trình AI, Machine Learning, Deep Learning như TensorFlow, DL4J, MAHOUT.

3. Đánh giá chung

Các tiêu chí về bản chất ngôn ngữ thường ảnh hưởng đến việc lựa chọn cho việc học tập, nghiên cứu. Tùy vào mục tiêu kiến thức, ta lựa chọn ngôn ngữ phù hợp, đây cũng là nền tảng quan trọng để tìm hiểu về tính ứng dụng của ngôn ngữ lập trình.

Các ngôn ngữ lập trình hiện đại ngày nay thường được so sánh và cân nhắc lựa chọn dựa vào tính ứng dụng của chúng. Để lựa chọn ngôn ngữ lập trình phù hợp, cần tìm hiểu kỹ về mặt ứng dụng của các ngôn ngữ và yêu cầu của dự án.



Đồ thị trên so sánh 10 ngôn ngữ lập trình phổ biến nhất của Chỉ số Cộng đồng lập trình TIOBE (Tính đến tháng 5/2023). Xếp hạng này dựa trên số lượng kỹ sư toàn thế giới, các khóa học và tính toán trên các công cụ tìm kiếm Google, Bing, Wikipedia, Amazon, Youtube, Baidu, ...

CHƯƠNG IV: CÁC YẾU TỐ TÁC ĐỘNG, CHI PHỐI VIỆC THIẾT KẾ NGÔN NGỮ LẬP TRÌNH

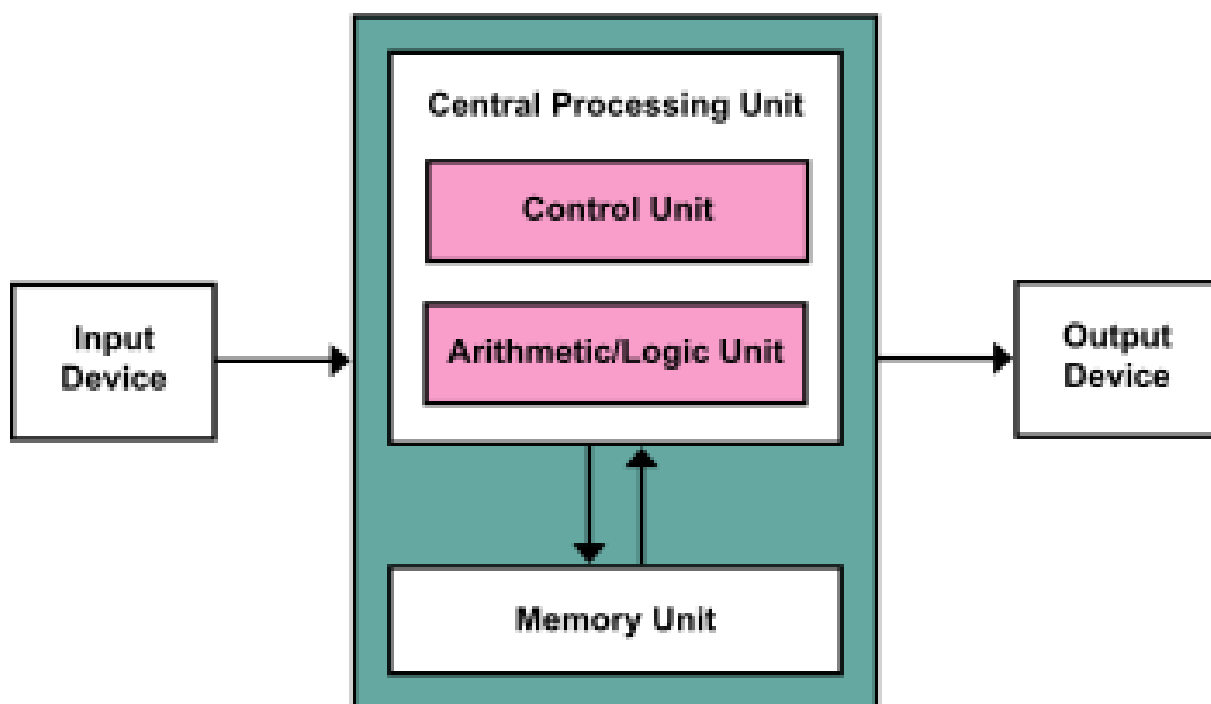
1. Kiến trúc máy tính (Computer Architecture)

- Các ngôn ngữ phát triển xung quanh kiến trúc máy tính phổ biến, hay còn gọi là kiến trúc von Neumann. Đây là kiến trúc máy tính dựa trên mô tả năm 1945 của nhà toán học và vật lý John von Neumann và những người khác trong Bản thảo đầu tiên của Báo cáo về EDVAC.

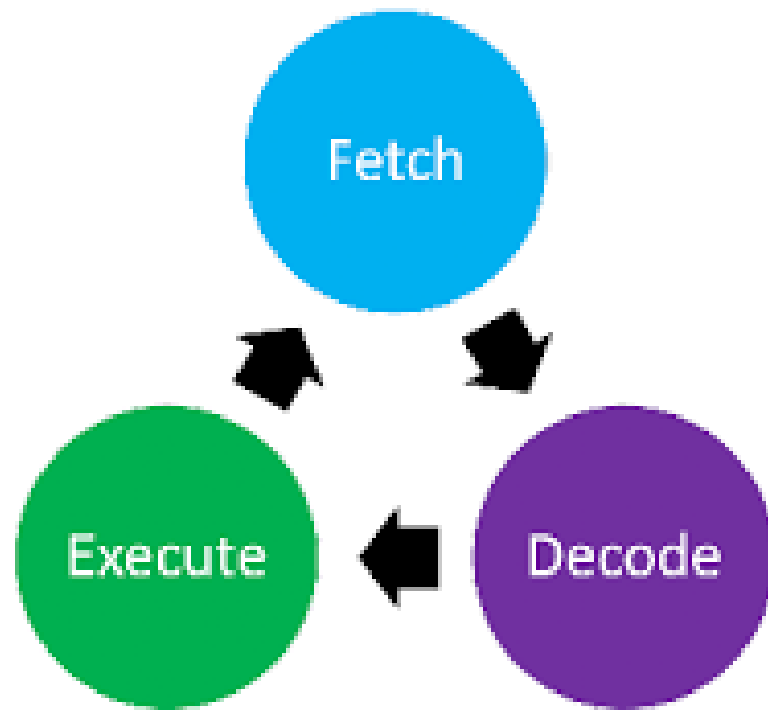
- Những ngôn ngữ thiết kế theo kiến trúc này còn gọi là ngôn ngữ mệnh lệnh (imperative language)

- Đặc điểm

- + Dữ liệu và chương trình được lưu trữ trong cùng bộ nhớ
- + Bộ nhớ độc lập với CPU
- + Các hướng dẫn (lệnh) và dữ liệu được dẫn từ bộ nhớ đến CPU
- + Kết quả hoạt động của CPU được trả về bộ nhớ



- Chu trình lệnh trong CPU: Tìm nạp, giải mã, thực thi. (Fetch-execute cycle)



+ Tìm nạp: Trong đó lệnh được bắt từ RAM và được sao chép vào bên trong bộ xử lý.

+ Giải mã: Trong đó lệnh đã bắt trước đó được giải mã và gửi đến các đơn vị thực thi.

+ Thực thi: Nơi lệnh được giải quyết và kết quả được ghi trong thanh ghi bên trong của bộ xử lý hoặc trong địa chỉ bộ nhớ của RAM.

2. Hệ phương pháp lập trình (Programming Methodologies)

- Các phương pháp phát triển phần mềm mới (ví dụ: phát triển phần mềm hướng đối tượng) đã dẫn đến các mô hình lập trình mới, hay rộng hơn là ngôn ngữ lập trình mới.

- Từ 1950 đến đầu những năm 1960: Ứng dụng đơn giản, lo lắng về hiệu suất máy tính.

- 1970: Chi phí phần cứng giảm, chi phí lập trình viên tăng. Các vấn đề lớn hơn đang được máy tính giải quyết.

+ Lập trình có cấu trúc (structured programming)

+ Thiết kế từ trên xuống (top-down design) và sàng lọc từng bước (step-wise refinement)

+ Kiểm tra kiểu không đầy đủ

+ Các câu lệnh kiểm soát không đầy đủ (cần sử dụng goto)

- Cuối những năm 1970: Chuyển từ hướng thủ tục (procedure-oriented) sang hướng dữ liệu (data-oriented). Nhấn mạnh vào thiết kế dữ liệu, tập trung vào việc sử dụng các kiểu dữ liệu trừu tượng để giải quyết vấn đề. Hầu hết các ngôn ngữ được thiết kế trong thời điểm này đều hỗ trợ trừu tượng hóa dữ liệu.

- Giữa những năm 1980: Lập trình hướng đối tượng (Object-oriented programming)

+ Dữ liệu trừu tượng: quá trình xử lý được đóng gói, quyền truy cập được kiểm soát.

+ Kế thừa: một đối tượng có thể kế thừa các đặc điểm từ một đối tượng khác.

+ Đa hình: một hành động hay phương thức có thể làm những công việc khác nhau dựa trên đối tượng nó đang hành động.

- Các ngôn ngữ mệnh lệnh phổ biến hiện nay đều hỗ trợ OOP (C++, Java, Ada 95, ...)

- Các khái niệm hướng đối tượng cũng xuất hiện trong

+ Lập trình hàm (CLOS)

+ Lập trình logic (Prolog++)

CHƯƠNG V: KẾT LUẬN

Qua đồ án này, em đã tìm hiểu được các tiêu chí đánh giá ngôn ngữ lập trình cả về mặt ngôn ngữ và về mặt ứng dụng. Về mặt ngôn ngữ, có 4 tính chất quan trọng là tính dễ đọc, tính dễ viết, độ tin cậy và chi phí. Các tính chất này thường quyết định mục đích sử dụng của ngôn ngữ, cũng như là cơ sở để lựa chọn ngôn ngữ phục vụ học tập, nghiên cứu.

Các ngôn ngữ lập trình hiện nay được so sánh chủ yếu dựa vào mặt ứng dụng khác nhau, điều này cũng phụ thuộc vào mục đích của người sử dụng cũng như yêu cầu của sản phẩm, dự án hay môn học.

Em đã tiến hành tìm hiểu về tính ứng dụng của một số ngôn ngữ phổ biến hiện nay như Python, C++, JavaScript, ... phân tích được những ưu và nhược điểm của chúng đối với từng mặt ứng dụng, từ đó đưa ra cách nhìn tổng quát cho mỗi ngôn ngữ lập trình.

Cuối cùng, em đã tìm hiểu về hai yếu tố cơ bản ảnh hưởng đến việc thiết kế một ngôn ngữ lập trình, đó là kiến trúc máy tính và hệ các phương pháp lập trình.

TÀI LIỆU THAM KHẢO

- [1] “TIOBE Index for May 2023” <https://www.tiobe.com/tiobe-index/> (accessed May 3rd, 2023)
- [2] “Language Evaluation Criteria” <https://www.geeksforgeeks.org/language-evaluation-criteria/> (accessed May 3rd, 2023)
- [3] Robert W. Sebesta, “Concepts of Programming Languages 9th”

