**ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH**

**ĐẠI HỌC CÔNG NGHỆ THÔNG TIN**

**KHOA KHOA HỌC MÁY TÍNH**



**BÁO CÁO ĐỒ ÁN**

**MÔN: NGUYÊN LÝ VÀ PHƯƠNG PHÁP LẬP TRÌNH**

**CHỦ ĐỀ: CÁC TIÊU CHÍ ĐÁNH GIÁ MỘT NGÔN NGỮ LẬP TRÌNH**

GIẢNG VIÊN HƯỚNG DẪN:

**TRỊNH QUỐC SƠN**

NHÓM**: 10**

THÀNH VIÊN NHÓM:

* **PHAN MINH NHẬT - 19521956**
* **LÊ VÕ TIẾN PHÁT – 19521993**

**MỤC LỤC**

**I. Các tiêu chí đánh giá**……………………………………………………...2

1. Khả năng đọc (Readability)………………………………………….2

1.1 Sự đơn giản tổng thể…………………………………………….2

1.2 Trực giao………………………………………………………...3

1.3 Các kiểu dữ liệu………………………………………………....3

1.4 Cú pháp………………………………………………………….4

2. Khả năng viết (Writability)…………………………………………..4

2.1 Sự đơn giản và trực giao………………………………………...4

2.2 Khả năng diễn đạt……………………………………………….5

2.3 Tính trừu tượng………………………………………………….6

3. Độ tin cậy (Reliability)………………………………………………7

3.1 Kiểm tra kiểu……………………………………………………7

3.2 Xử lí ngoại lệ……………………………………………………8

3.3 Răng cưa………………………………………………………...9

4. Chi phí (Cost)………………………………………………………..9

5. Một số tiêu chí khác (Others)………………………………………..9

5.1 Tính di động………………………………………………….....9

5.2 Tính tổng quát…………………………………………………..9

5.3 Được định nghĩa rõ ràng………………………………………..9

**II. So sánh các ngôn ngữ dựa trên các tiêu chí**…………………………...10

**\*Các nguồn tài liệu tham khảo**………………………………………........14

**I. Các tiêu chí đánh giá:**

|  |  |  |  |
| --- | --- | --- | --- |
|  | **CRITERIA** | | |
| **Characteristic** | Readability | Writability | Reliability |
| Simplicity | Checkmark with solid fill | Checkmark with solid fill | Checkmark with solid fill |
| Orthogonality | Checkmark with solid fill | Checkmark with solid fill | Checkmark with solid fill |
| Data types | Checkmark with solid fill | Checkmark with solid fill | Checkmark with solid fill |
| Syntax design | Checkmark with solid fill | Checkmark with solid fill | Checkmark with solid fill |
| Support for abstraction |  | Checkmark with solid fill | Checkmark with solid fill |
| Expressivity |  | Checkmark with solid fill | Checkmark with solid fill |
| Type checking |  |  | Checkmark with solid fill |
| Exception handling |  |  | Checkmark with solid fill |
| Restricted aliasing |  |  | Checkmark with solid fill |

**1. Readability:** Sự dễ dàng của ngôn ngữ lập trình trong việc đọc và hiểu chương trình.

**1.1. Sự đơn giản tổng thể (Overall Simplicity):**

- Một tập hợp các tính năng và cấu trúc có thể quản lý được.

- Một ngôn ngữ có quá nhiều cấu trúc cơ bản sẽ khó học hơn các ngôn ngữ khác.

- Đa tính năng (Feature Multiplicity): một phép toán sẽ có nhiều cách biểu diễn khác nhau.

*Trong C/C++: a = a + 1; a += 1; a++; ++a.*

*Trong BareBones: incr a;*

- Nạp chồng toán tử (Operator Overloading): một toán tử có nhiều hơn một nghĩa.

Ví dụ: Khi lập trình hướng đối tượng trong C++, ta có thể viết lại toán tử + để có thể cộng được đơn thức, đa thức,… thay vì cộng các số như bình thường.

Operator Overloading thường khá hữu dụng và làm ngôn ngữ trở nên đơn giản hơn, tuy nhiên nếu người dùng được cho phép tạo ra các overloading và họ làm điều đó một cách không hợp lí sẽ làm giảm khả năng đọc của ngôn ngữ.

Một số ngôn ngữ hỗ trợ: C++, C#, Ada, Python,…

\* Một ngôn ngữ quá đơn giản sẽ làm giảm khả năng đọc bởi nó thiếu các lệnh và các cấu trúc phức tạp. Ví dụ điển hình là ngôn ngữ Assembly.

**1.2. Trực giao (Orthogonality):**

- Một tập các cấu trúc cơ bản có thể được kết hợp theo một số cách để xây dựng cấu trúc điều khiển và dữ liệu của ngôn ngữ.

Ví dụ: Trong một ngôn ngữ có các kiểu dữ liệu cơ bản là integer, float, character và 2 kiểu dữ liệu là array và pointer. Ta có thể kết hợp để tạo ra mảng số nguyên, mảng kí tự, con trỏ có giá trị là số thực,...

- Mọi sự kết hợp đều được cho phép.

Ngôn ngữ có nhiều sự trực giao nhất là ALGOL 68. Có quá nhiều sự trực giao khiến cho ngôn ngữ trở nên phức tạp hơn.

**1.3. Các kiểu dữ liệu (Data types):**

- Có đầy đủ các kiểu dữ liệu và cấu trúc được định nghĩa từ trước.

- Bao gồm:

+ Basic data types (Ví dụ: integer, character, float, Boolean trong C, C++, Java,…)

+ Derived data types (Ví dụ: pointer, struct, array, union trong C; list, tuples, sets, dictionaries trong Python)

**1.4. Cú pháp (Syntax):**

- Các từ đặc biệt (ví dụ: for, while, class,…) và các phương pháp hình thành compound statements.

Trong C, dấu ngoặc nhọn dùng để xác định compound statements.

*for (int i=0, i < 10, i++)*

*{*

*printf(“%d\n”, i);*

*}*

Trong Ada, **end if** được dùng để kết thúc cấu trúc lựa chọn, **end loop** để kết thúc cấu trúc lặp.

*loop*

*Get(Current\_Character);*

*exit when Current\_Character = '\*';*

*end loop;*

- Hình thức và ý nghĩa: ngữ nghĩa hoặc ý nghĩa phải đúng theo cú pháp. Trong một số trường hợp, nguyên tắc này bị vi phạm bởi hai cấu trúc giống hệt nhau hoặc giống nhau về hình thức nhưng có ý nghĩa khác nhau,điều này tùy thuộc vào ngữ cảnh.

**2. Writability:** Sự dễ dàng của ngôn ngữ trong viêc tạo ra chương trình.

**2.1. Sự đơn giản và trực giao (Simplicity & Orthogonality):**

Chỉ nên có một số lượng nhỏ các cấu trúc cơ bản và một số luật để kết hợp chúng, nếu có quá nhiều sẽ làm cho ngôn ngữ trở nên phức tạp hơn, giảm khả năng viết.

**2.2. Khả năng diễn đạt (Expressivity):**

Ngôn ngữ có khả năng biểu diễn một phép toán một cách thuận tiện hơn.

Trong C, cách biểu diễn a++ tiện và ngắn hơn a=a+1.

Trong Java, sử dụng for để viết các vòng lặp đếm dễ hơn dùng while.

*For Example:*

*public class ForExample {*

*public static void main(String[] args){*

*for (int i=1; i<=10; i++) {*

*System.out.println(i);*

*}*

*}  
}*

*While Example:*

*public class WhileExample1 {*

*public static void main(String[] args){*

*int i=1;*

*while(i<=10)*

*{*

*System.out.println(i);*

*i++;*

*}*

*}*

*}*

**2.3. Tính trừu tượng (Abstraction):**

Khả năng định nghĩa và sử dụng các cấu trúc hoặc phép toán phức tạp cho phép bỏ qua các chi tiết.

+ Process Abstraction: có thể hiểu là các hàm hoặc là các chương trình con.

*void hoanvi (int &a, int &b)*

*{*

*int temp = a;*

*a = b;*

*b = temp;*

*}*

+ Data Abstraction

*Struct sinhvien*

*{*

*Char MSV[20]l; //ma sinh vien*

*Char hoten[30]; //ho ten sinh vien*

*Double diemTB; //diem trung binh*

*Struct ngaysinh*

*{*

*Int ngay,thang,nam;*

*} ns;*

*};*

Một số ngôn ngữ hỗ trợ: C/C++, Java, Python,…

**\* Readability và Writability là hai tiêu chí quan trọng nhất để đánh giá một ngôn ngữ lập trình.**

**3. Reliability (Độ tin cậy)**

Độ tin cậy của chương trình là xác suất của hoạt động chương trình không có lỗi đối với khoảng thời gian cụ thể trong một tập hợp các điều kiện cụ thể .

**3.1. Kiểm tra kiểu (Type checking):**

Kiểm tra lỗi kiểu (type errors) trong một chương trình, bởi trình biên dịch hoặc trong quá trình thực thi chương trình.

Có 2 cách kiểm tra kiểu:

+ ***Static type checking***: mỗi biến (variable) có một kiểu cố định liên kết với nó. Kiểu của biến được kiểm tra lúc **compiled-time** và trình biên dịch yêu cầu ta phải khai báo rõ kiểu của biến trước khi sử dụng.

*Ví dụ:*

*int a ; // the type of a is integer*

*a=3; //OK, assign 3 to a*

*a=”Hello”; // Error,cannot assign tring value to int*

Ưu điểm của Static type checking:

-Sớm phát hiện nhiều lỗi, giúp loại bỏ sự cần thiết phải lặp lại kiểm tra kiểu mỗi khi chương trình được thực thi.

- IDE hỗ trợ việc kiểm soát kiểu dữ liệu giúp viết code nhanh hơn và sai sót ít hơn.

- Mã nguồn dễ đọc hơn cho người mới.

+ ***Dynamic type checkin*g**: kiểu của biến hay nói chính xác hơn là kiểu của object liên kết với biến được kiểm tra lúc **runtime**.

*Ví dụ:*

*b=3 # variable b is bound to integer object*

*isinstance(b,int) # True*

*b= ‘hello world!’ # Now b is bound to string object*

*isinstance(b,str) # True*

*b=[‘blue,’black’,’brown’] # Now b is a list*

Ưu điểm của Dynamic type checking:

• Rất linh hoạt trong việc thiết kế chương trình.

• Không cần khai báo kiểu.

**3.2. Xử lí ngoại lệ (Exception handling):**

Chặn run-time errors và thực hiện các biện pháp sửa chữa, sau đó tiếp tục.

Ngoại lệ (Exception) là lỗi xảy ra trong quá trình thực thi một chương trình.

Ngoại lệ có thể là bất kỳ điều kiện bất thường nào trong chương trình phá vỡ luồng thực thi chương trình đó. Bất cứ khi nào một ngoại lệ xuất hiện, chương trình sẽ ngừng thực thi, chuyển qua quá trình gọi và in ra lỗi đến khi nó được xử lý.

Vd: Cặp câu lệnh try-except được sử dụng trong việc xử lý lỗi và ngoại lệ trong Python.

*Basic Syntax :*

*try :*

*//Code*

*except:*

*//Code*

**3. Răng cưa (Aliasing):**

Có hai hoặc nhiều tên riêng có thể được sử dụng để cùng truy cập vào một ô nhớ.

Vd: Hai con trỏ cùng trỏ đến một giá trị.

**\* Cả readability và writability đều ảnh hưởng đến reliability. Ngôn ngữ càng dễ viết thì khả năng viết đúng càng cao. Một ngôn ngữ khó đọc thì cũng khó để viết và chỉnh sửa.**

**4. Cost**

- Đào tạo lập trình viên sử dụng ngôn ngữ.  
- Viết chương trình.  
- Biên dịch chương trình.  
- Thực thi các chương trình.  
- Hệ thống triển khai ngôn ngữ.  
- Độ tin cậy: độ tin cậy kém dẫn đến chi phí cao.  
- Duy trì các chương trình.

**5. Others**

**5.1. Tính di động (Portability):**

Dễ dàng chuyển các chương trình từ triển khai này sang triển khai khác.

**5.2. Tính tổng quát (Generality):**

Khả năng áp dụng cho một loạt các ứng dụng.

**5.3. Được định nghĩa rõ ràng (Well-definedness):**

Tính đầy đủ và chính xác trong định nghĩa chính thức của ngôn ngữ.

**II. So sánh các ngôn ngữ dựa trên các tiêu chí:**

So sánh 3 ngôn ngữ: Pascal, C++, Python. Chúng em chọn 3 ngôn ngữ này vì 2 trong số đó là ngôn ngữ rất hot được sử dụng rất phổ biến hiện nay là Python và C++, ngôn ngữ còn lại là Pascal thì khá lạc hậu. Và điểm chung là các ngôn ngữ này là đều thường được đưa vào giảng dạy cho người mới bắt đầu.

Bọn em sẽ so sánh 3 tiêu chí chính là: Readability, Writability và Reliability.

Vì phần tiêu chí Cost bọn em không tìm thấy tài liệu nên không so sánh giữa các ngôn ngữ được.

|  |  |  |  |
| --- | --- | --- | --- |
| **Languages**  **Criterias** | **Pascal** | **C++** | **Python** |
| **Readability**  - Simplicity  - Orthogonality  - Data types  - Syntax | - Ngôn ngữ khá đơn giản, dễ tiếp cận, dễ học.  - Trực giao khá hạn chế, bởi vì có nhiều quy tắc không nhất quán.  - Có đầy đủ các kiểu basic data và derived data.  - Dễ tiếp cận nhưng đôi khi gây nhầm lẫn, khó đọc (do Pascal cặp begin-end để xác định compound statement, nên trong một đoạn code dài có nhiều compound statement thì sẽ dễ gây nhầm lẫn). | - Ngôn ngữ khá đơn giản, dễ tiếp cận, dễ học.  - Có trực giao.  - Có đầy đủ các kiểu basic data và derived data.  - Dễ tiếp cận nhưng đôi khi gây nhầm lẫn, khó đọc (do C++ cặp dấu ngoặc nhọn để xác định compound statement, nên trong một đoạn code dài có nhiều compound statement thì sẽ dễ gây nhầm lẫn). | - Ngôn ngữ khá đơn giản, dễ tiếp cận, dễ học.  - Có trực giao.  - Có đầy đủ các kiểu basic data và derived data.  - Cú pháp khá rõ ràng, dễ đọc, dễ hiểu. |
| **Wriability**  -Simplicity& Orthogonality  -Expressivity  -Abstraction | - Tính diễn đạt kém.  - Có hỗ trợ abstraction. | - Tính diễn đạt cao.  - Có hỗ trợ abstraction. | - Tính diễn đạt cao.  - Có hỗ trợ abstraction. |
| **Reliability**  - Type checking  - Exception handling  - Aliasing | - Là ngôn ngữ kiểm tra kiểu mạnh. Tuy nhiên, một số kiểm tra chỉ có thể thực hiện động.  - Các ngoại lệ có thể được xử lý bằng khối lệnh try...except.  - Có aliasing. | - Hỗ trợ rất ít trong việc kiểm tra kiểu động, phần lớn là kiểm tra kiểu tĩnh.  - Được xây dựng dựa trên 3 từ khóa: try, catch và throw.  - Có aliasing. | - Là ngôn ngữ mạnh trong kiểm tra kiểu động.  - Các ngoại lệ có thể được xử lý bằng khối lệnh try...except.  - Có aliasing. |

**Nhận xét:**

Cả 3 ngôn ngữ trên đều là các ngôn ngữ dễ học, dễ tiếp cận bởi Readability và Writability của chúng khá tốt. Không phải ngẫu nhiên mà chúng được đưa vào giảng dạy cho những người mới học lập trình . Ngoài ra cả 3 ngôn ngữ đều có độ tin cậy cao. Tuy nhiên, Pascal là ngôn ngữ lép vế hơn 2 ngôn ngữ còn lại vì ngôn ngữ này khá cũ, không hỗ trợ người dùng được nhiều hơn 2 ngôn ngữ kia, có lẽ vì vậy mà hiện tại Pascal không còn được nhiều người lập trình sử dụng nữa.

**Một số ưu điểm của ngôn ngữ Python:**

* Cấu trúc rõ ràng, cú pháp ngắn gọn.
* Tốc độ xử lý cực nhanh.
* Có trên tất cả các nền tảng hệ điều hành từ UNIX, MS – DOS, Mac OS, Windows và Linix…
* Tương thích mạnh mẽ với Unix, hardware, thirt-party software với số lượng thư viện khổng lồ (400 triệu người sử dụng).

**Một số ưu điểm của ngôn ngữ C++:**

* Tính tương thích cao, được sử dụng trên nhiều nền tảng.
* Cú pháp rất sát với suy nghĩ logic, do đó việc viết code khá nhanh chóng và đơn giản.
* Đa mô hình và có nhiều thư viện (tính năng gần giống với các plug-in).
* Có rất nhiều ứng dụng giao diện mà phía sau nó được viết bằng C++. Một số ứng dụng phổ biến có thể kể tới là: Image Ready, Adobe Premier, Photoshop và Illustrator.
* Game 3D.
* Ứng dụng tính toán và đồ họa.
* C++ là một phần không thể thiếu trong nhiều hệ điều hành phổ biến như Apple OSX và MS Windows.

**Một số nhược điểm của ngôn ngữ Pascal:**

* Ngôn ngữ lập trình Pascal thiếu sự linh hoạt.
* Có một số từ khóa dư thừa hoặc không có nhiều tác dụng.
* Hệ thống thư viện hàm được thiết kế còn khá ít,kiểu liệt kê còn chưa ổn, kiểu bản ghi cũng đã lỗi thời.
* Pascal không thống nhất về Compiler và cũng có khá ít IDE.

Vì một số nhược điểm kể trên thì Pascal chỉ giúp cho ta hiểu được cách làm việc của máy tính, cách giao tiếp để ra lệnh cho máy tính làm việc theo sự điều khiển của con người thông qua ngôn ngữ lập trình.

**Các nguồn tài liệu tham khảo:**

1. *Harrykar. ‘Language Evaluation Criteria’, posted on November 11, 2018.* [*https://progr-harrykar.blogspot.com/2018/11/language-evaluation-criteria.html*](https://progr-harrykar.blogspot.com/2018/11/language-evaluation-criteria.html)
2. *Ved Ed. ‘Pl9ch1’, post on April 08, 2019.* [*https://www.slideshare.net/VedEd/pl9ch1/*](https://www.slideshare.net/VedEd/pl9ch1/)
3. *Hoang Van Vuong. ‘Strong vs Weak, Static vs Dynamic typing là cái khỉ gì?’, posted October 26, 2016.* [*https://viblo.asia/p/strong-vs-weak-static-vs-dynamic-typing-la-cai-khi-gi-JQVkVzZokyd*](https://viblo.asia/p/strong-vs-weak-static-vs-dynamic-typing-la-cai-khi-gi-JQVkVzZokyd)
4. *Ban truyền thông ITPlus. ‘Ưu và nhược điểm của Python trong lập trình’, August 13, 2017.* [*http://itplus-academy.edu.vn/Uu-va-nhuoc-diem-cua-Python-trong-lap-trinh.html*](http://itplus-academy.edu.vn/Uu-va-nhuoc-diem-cua-Python-trong-lap-trinh.html)
5. *Admin. ‘Các lý do nên học code C++’, November 22, 2019.* [*http://visualcpp.net/cac-ly-do-nen-hoc-code-c/*](http://visualcpp.net/cac-ly-do-nen-hoc-code-c/)
6. *Duy Vinh. ‘Ưu và nhược điểm của Pascal so với ngôn ngữ lập trình khác’.* [*https://thuthuat.taimienphi.vn/uu-nhuoc-diem-cua-pascal-28880n.aspx*](https://thuthuat.taimienphi.vn/uu-nhuoc-diem-cua-pascal-28880n.aspx)