Lời cảm ơn

Lời đầu tiên em xin được cảm ơn Ths. Phạm Trọng Nghĩa đã giúp đỡ em trong những bước đầu của làm một bản báo cáo phù hợp và chuyên nghiệp cho môi trường đại học. Tiếp theo là cảm ơn gia đình đã luôn bên cạnh cổ vũ tinh thần giúp em hoàn thành sản phẩm. Cuối cùng, em xin cảm ơn thầy Lê Minh Hoàng đã truyền lửa cho em học về Pascal thay vì hắt hủi nó.

Mục Lục

Danh Mục Hình

Danh Mục Bảng

Phần 1: Những Ngôn Ngữ Lập Trình Đầu Tiên

Khi mà những phần mềm được viết ra càng ngày càng phức tạp, lỗi con người khi phải trực tiếp viết những mã nhị phân trở nên càng ngày càng phổ biến. Do đó, những kỹ sư máy tính cần một công cụ cho phép biến một bản thảo có ngôn ngữ gần gũi với con người thành mã máy. Và do đó, ngôn ngữ lập trình đầu tiên được ra đời.

1.1. Assembly

Ra đời vào những năm 1940, *Assembly* — tiếng Việt là *hợp ngữ* — có thể được xem là ngôn ngữ lập trình bậc thấp nhất khi mà các câu lệnh của nó đều rất gần gũi với lệnh máy tính. Mỗi câu lệnh của *Assembly* thường chỉ thực hiện một lệnh máy tính duy nhất. Nhưng vì các lệnh của *Assembly* phụ thuộc mật thiết vào lệnh máy tính như vậy, có thể nói mỗi nền tảng khác nhau sẽ có một ngôn ngữ *Assembly* rất riêng.

Ví dụ, đây là chương trình xuất "Hello World" ra màn hình khi viết trên nền tảng `Linux x86_64` của tác giả:

```
section .data
  text db "Hello World", 10

section .text
global _start
_start:
  mov rax, 1
  mov rdi, 1
  mov rsi, text
  mov rdx, 14
  syscall

  mov rax, 60
  mov rdi, 0
  syscall
```

Còn đây là khi viết trên `MOS 6502` *(code của u/Proxy_PlayerHD)* [1]:

```
START:
    LDX #0
    .LOOP:
        LDA STRING,X
        BEQ .EXIT
        STA TERMINAL
        INC X
    BNE .LOOP
    .EXIT:
STP

STRING:
#d "Hello World!\n\0"
```

Từ 2 đoạn code trên có thể thấy: *Assembly* cho phép người dùng trực tiếp điều khiển CPU, bộ nhớ và thiết bị đầu ra của máy tính. Đồng thời, có những câu lệnh chỉ tồn tại trong nền tảng này, nhưng lại không tồn tại trong nền tảng kia.

Dù phức tạp, *Assembly* vẫn rất thông dụng cho đến tận ngày hôm nay. Chương trình được viết bởi

Assembly chạy rất nhanh và hiệu quả. Có rất nhiều ngôn ngữ bậc cao thông dụng hiện nay phải biên dịch chính nó về *Assembly*. Ngoài ra, ta vẫn bắt gặp những chương trình được viết bởi ngôn ngữ này trong các thiết bị như Raspberry Pi và các thiết bị nhúng.

1.2. Fortran

Sau sự ra đời của *Assembly*, nhiều ngôn ngữ lập trình đã được tạo ra với mục đích biên dịch chính nó xuống *Assembly*. Điều này giúp việc lập trình ít phụ thuộc nền tảng hơn và dễ đọc viết hơn, khi mà cú pháp và câu lệnh được ghi trên những ngôn ngữ này không phụ thuộc mật thiết với mã máy tính như *Assembly* nữa. Trong số các ngôn ngữ ấy, *Fortran* (1954) là ngôn ngữ đầu tiên được hoàn thiện chỉnh chu. Điều đó khiến cho nó được công nhận là ngôn ngữ lập trình bậc cao đầu tiên trên thế giới.

Là ngôn ngữ lập trình bậc cao đời đầu, *Fortran* không khỏi bị dính phải những chỉ trích về hiệu năng và độ tin cậy. Tuy nhiên, so với *Assembly*, *Fortran* mang lại hiệu quả lao động cao hơn nhiều cho lập trình viên. Nó giúp họ lập trình nhanh, chính xác, và không đòi hỏi quá nhiều kiến thức phần cứng để sử dụng.

Lấy ví dụ, đây là phần mềm xuất "Hello World" ra màn hình được viết bằng Fortran. Nó có thể được biên dịch để chạy trên nhiều nền tảng khác nhau, và nó dễ đọc, dễ hiểu và dễ viết hơn rất nhiều so với *Assembly*:

```
program f
  print *, 'Hello, World!'
end program f
```

Tuy độ phổ biến của *Fortran* ngày nay đã sụt giảm đáng kể, nó vẫn được tin dùng trong những chương trình hiệu năng cao và vẫn đang ngày càng được cải tiến.

1.3. LISP

LISP (1958) là một ngôn ngữ lập trình hết sức thú vị. LISP nổi tiếng với việc tiên phong tạo ra những kỹ thuật lập trình mà tính đến nay vẫn còn rất thông dụng, được sử dụng rộng rãi trong các ngôn ngữ lập trình đời mới.

Vào thời điểm LISP được tạo ra, nó là ngôn ngữ lập trình bậc cao thứ 2 (sau Fortran). Nhưng khác với Fortran, LISP là một ngôn ngữ *thông dịch*: Một trình thông dịch sẽ đọc từng dòng trong mã nguồn của LISP, lập tức chạy dòng đó rồi mới đọc dòng kế tiếp.

Ngoài ra, LISP là ngôn ngữ tiên phong cho những tính năng như:

- **Dynamic Typing**: Trình biên dịch của LISP sẽ tự động gán kiểu dữ liệu cho các biến trong runtime. Người dùng không cần phải chỉ định cụ thể kiểu dữ liệu cho biến
- **Garbage Collector**: Người dùng không cần phải quản lý bộ nhớ một cách thủ công

Về kỹ thuật, LISP là ngôn ngữ đầu tiên cho phép đệ quy, có nghĩa là một hàm có thể tự gọi chính nó. Lấy ví dụ, ta có số thứ n trong dãy số Fibonacci được định nghĩa như sau:

$$fibonacci(n) = \begin{cases} f(n-1) + f(n-2), & \text{nếu } n \geq 2 \\ 1, & \text{nếu } n = 1 \\ 0, & \text{nếu } n = 0 \end{cases}$$

Ta có thể biểu diễn hàm này trong LISP như sau [2]:

```
(defun fibonacci (n &optional (a 0) (b 1) (acc ()))
  (if (zerop n)
      (nreverse acc)
      (fibonacci (1 - n) b (+ a b) (cons a acc))))
```

1.4. Cobol & BASIC

COBOL (1959) và BASIC (1964) là 3 ngôn ngữ rất thông dụng cho đến gần hết thế kỷ 20. Trong đó, nổi bật nhất với công chúng có lẽ là BASIC, khi vào năm 1991, Microsoft đã tung ra hậu bản của BASIC: Visual Basic — một công cụ rất thân thuộc với những người viết macro cho Microsoft Excel.

Ngoài ra, COBOL tuy đã không còn được sử dụng rộng rãi, nó vẫn được update thường xuyên và rất được tin dùng trong các hệ thống đặc thù.

Tiểu chương: Pascal

Được phát hành vào năm 1970, trong hơn 20 năm sau đó, Pascal đã từng là ông vua của mọi ngôn ngữ lập trình.

Pascal cho phép người dùng thực hiện rất nhiều kỹ thuật mà trước đây chưa từng có một ngôn ngữ nào cho phép, từ đó khiến cho Pascal trở thành sự lựa chọn mặc định khi người ta viết sách vở về thuật toán, học thuật, cấu trúc dữ liệu, vâng vâng… Ngoài ra, Pascal cũng được dạy rộng rãi trong các chương trình đại học khắp toàn cầu do cú pháp của nó rất rõ nghĩa.

Nhưng tuổi thọ của Pascal không dài lâu. Trong suốt một khoảng thời gian dài sau đó, Pascal không được chuẩn hóa với kỹ thuật lập trình hiện đại.

Theo thời gian, Pascal gặp phải những vấn đề như: Cú pháp lỗi thời, không được mở rộng, thiếu thốn thư viện, cộng đồng phát triển Free Pascal lại tung ra những bản update đầy lỗi, đến cả biên dịch cũng sai. Đường lối phát triển của Pascal cũng gặp nhiều vấn đề khi bộ phận phát triển đầu tư quá nhiều vào các công cụ thiếu chất lượng, trong khi Object Pascal — một ngôn ngữ mở rộng khá hay và hợp thời của Pascal — thì lại bị làm sơ sài.

Đến đầu năm 1990, vai trò về học thuật — điều mà Pascal làm tốt nhất — dần bị C và C++ thay thế. Tính đến nay, Pascal đã không còn thực sự dẫn đầu trong một lĩnh vực nào nữa.

Phần 2: Những Ngôn Ngữ Lập Trình Thông Dụng

2.1. The C Programming Language & The C Family

Năm 1972, *C* lần đầu được công bố, nhưng đến tận năm 1983 mới được đón nhận nồng nhiệt nhờ vào sự ra đời của trình biên dịch *GCC* do *Richard Stallman* viết ra. Về sau, với sự lớn mạnh của *Unix-based operating system* và sự ra đời của hệ điều hành *Linux*, *C* và *C++* (ban đầu là một extension của *C*) nhanh chóng trở thành ngôn ngữ tiêu chuẩn và phổ biến nhất trong nhiều lĩnh vực, trong đó bao gồm phát triển OS, drivers, giao thức, và phát triển các ứng dụng hiệu năng cao.

Đặc biệt, *C* và *C++* được sử dụng rộng rãi trong các siêu máy tính, thiết bị IOT, hệ thống nhúng, và các thiết bị vi điện tử khác.

Về học thuật, *C++* có vai trò giống như *Pascal* ngày xưa: Được sử dụng để viết sách vở về thuật toán và cấu trúc dữ liệu, đồng thời được giảng dạy ở rất nhiều giáo trình. Riêng với bộ môn lập trình thi đấu, *C++* luôn là ngôn ngữ yêu thích của các competitive programmers. Dù là thi đấu offline (ICPC, IOI) hay thi đấu online (codeforces.com, atcoder.jp, oj.vnoi.info, freecontest.net) thì *C++* luôn là sự lựa chọn của đa số các đấu thủ.

Về cú pháp, *C* và *C++* tôn trọng sự tự do của người dùng đến mức tối đa. Lấy ví dụ, để truy cập phần tử thứ `i` trong mảng `a`, ta có thể sử dụng bất kỳ cách nào sau đây:

```
a[i]
i[a]
```

```
*(a+i)
*(i+a)
```

Cả 4 dòng trên đều rất có logic nếu người đọc thật sự am hiểu cách *C* và *C++* hoạt động. Tuy nhiên, sự tự do quá mức của gia đình *C* đôi lúc cũng sinh ra những cục code vô cùng tối nghĩa như thế này:

```
++(**p)***--q
```

Cũng may là trong thực tế cũng không có ai thực sự code như vậy cả.

Ngoài ra, *GCC* còn là trung tâm của cả một hệ tư tưởng Free Software, và *C* bản thân nó cũng là một nét văn hóa. Ảnh chế, trò đùa, và các cuộc hội đàm liên quan đến *C* đã luôn là món ăn tinh thần vô cùng sảng khoái của cộng đồng coder trên mạng từ thời kỳ đầu của internet đến nay.

2.2. Java

Java (1995) nổi tiếng với cấu trúc file mã nguồn vô cùng chặt chẽ. Nó cũng được biết đến là ngôn ngữ đã đưa kỹ thuật lập trình hướng đối tượng (Object-oriented Programming) lên đỉnh cao.

Sự chặt chẽ của *Java* và cách nó triển khai *OOP* khiến codebase của *Java* vô cùng dễ mở rộng và chỉnh sửa. Do đó, *Java* được ưa chuộng trong những dự án doanh nghiệp quy mô lớn. Sự thuận tiện trong việc mở rộng của Java còn được thể hiện rõ qua video games, khi các tựa games được viết bằng ngôn ngữ này đều rất dễ được cộng đồng làm mod, mà nổi tiếng nhất có lẽ là Minecraft.

Là một ngôn ngữ thông dịch, *Java* có thể dễ dàng chạy trên vô số các thiết bị khác nhau. Nhưng cũng chính vì điều này mà hiệu năng của *Java* thực sự là một vấn đề đáng kể. Lấy ví dụ, tựa game Minecraft phiên bản *Java* có hiệu năng tệ hơn rất nhiều lần so với bản ngôn ngữ biên dịch — Bedrock Edition.

2.3. C#

Tuy mang chữ *C*, *C#* lại không liên quan gì đến gia đình *C*, mà nó lại giống với *Java* hơn. *C#* cũng là ngôn ngữ hướng đối tượng nghiêm ngặt như *Java*. Cấu trúc files và các keywords đều rất giống *Java*, nên từ khi nó ra mắt cho đến nay, nhiều người vẫn hay đùa rằng *C#* là "Microsoft Java".

Tuy nhiên, *C#* lại khắc phục được rất nhiều vấn đề của *Java*, bao gồm cả vấn đề về hiệu năng runtime. Cộng với sự chống lưng từ ông lớn Microsoft, *C#* có thể được sử dụng để phát triển ứng dụng chạy native trên bất kỳ một nền tảng phổ biến nào, kể cả là các thiết bị di động hay trên web.

Đặc biệt, *C#* cũng là ngôn ngữ phổ biến nhất để làm video games.

2.4. Python, Lua & Haskell

Điểm chung của *Python*, *Lua* và *Haskell* chính là việc phát triển phần mềm với chúng đều được thực hiện nhanh chóng và tốn rất ít công sức. Những người sử dụng Linux chắc hẳn đều đã quá quen thuộc với cả 3 ngôn ngữ này, khi chúng được dùng để thiết kế và tủy chỉnh vô số các chương trình phần mềm và hệ thống.

Python là một trong những ngôn ngữ thông dụng nhất hiện nay. Ngoài phát triển phần mềm, Python còn được sử dụng rộng rãi trong nghiên cứu.

Lua là ngôn ngữ dùng để viết config và plugins phổ biến nhất trên Linux.

Còn *Haskell* — ngôn ngữ đã mang kỹ thuật functional programming lên đỉnh cao — cũng rất được ưa chuộng vì nó vừa dễ học, dễ viết, nhưng lại cho hiệu quả runtime vô cùng tuyệt vời. Tiện thể, compiler từ latex sang pdf mà tác giả đang dùng được viết hoàn toàn bằng *Haskell*.

2.5. Javascript

Sự phát triển của internet đã đặt ra một yêu cầu: Các browsers (trình duyệt) cần một ngôn ngữ nhanh, nhẹ, và không cần biên dịch để biến những trang web khô khan trở nên sống động. Và thế là *Javascript* ra đời.

Javascript — không giống như cái tên, nó không có liên quan gì đến Java — có sứ mệnh biến những trang web tĩnh thành trang web động. Có nghĩa là: Khi một người ghé thăm 1 website, *Javascript* sẽ giúp cho thông tin của website đó được cập nhật và thay đổi liên tục, qua đó khiến không gian internet trở nên sống động và thu hút hơn.

Javascript cũng là ngôn ngữ được phát triển năng nổ nhất, khi mà mỗi tuần trôi qua, tuần nào cũng có thêm một hai framework/library *Javascript* chào đời.

Tuy nhiên, tính chất nhanh, nhẹ của *Javascript* đã mang lại một hệ quả khó tránh: Suốt mấy thập kỷ được sử dụng rộng rãi, nó vẫn toàn lỗi là lỗi. Một ví dụ điển hình:

```
>> typeof(10)
"number"
>> [1, 4, 2, 10, 3].sort()
Array(5) [ 1, 10, 2, 3, 4 ]
```

Và một ví dụ điển hình khác:

```
>> parseInt(0.00005)
0
>> parseInt(0.000005)
0
>> parseInt(0.0000005)
5
```

2.6. Query Languages

Cơ sở dữ liệu luôn là một phần quan trọng trong khoa học máy tính. Đi kèm với sự phát triển của các cơ sở dữ liệu là các ngôn ngữ truy vấn, trong đó nổi bật nhất chính là *SQL* — Structured Query Language. Về khái niệm, "ngôn ngữ truy vấn" là bất kỳ ngôn ngữ lập trình nào cho phép thu thập và chỉnh sửa thông tin trên cơ sở dữ liệu.

Các ngôn ngữ truy vấn thường chỉ hoạt động trên một số hệ quản trị cơ sở dữ liệu nhất định. Một số hệ quản trị cơ sở dữ liệu phổ biến ngày nay có thể kể đến như: MySQL, MS SQL Server, Mongo DB, PostgreSQL, Redis,…

2.7. Shell Languages

Không phải ngôn ngữ lập trình nào cũng được tạo ra để viết phần mềm, một số trong số chúng được sử dụng làm công cụ để người dùng giao tiếp với máy tính.

Tiêu biểu trong số đó là *Bash*, *Fish* và *ZSH* của hệ điều hành Linux, và *Powershell* của hệ điều hành Windows. Riêng *ZSH* được sử dụng trên cả MacOS.

Đây đều là những công cụ rất đắc lực trong các hệ thống tự động. Kể cả trong nhu cầu sử dụng máy tính thông thường, một khi thuần thục, người dùng có thể đạt được hiệu suất công việc cao hơn rất nhiều so với khi sử dụng giao diện người dùng trong vô số các trường hợp khác nhau.

Phần 3: Lập Trình trong Tương Lai

3.1. Những Ngôn Ngữ Lập Trình Mới

Rust (2010) và Go (2009) là hai ngôn ngữ lập trình tuy mới nhưng lại vô cùng được đón nhận. Riêng Rust, người ta vẫn hay truyền tai nhau câu đùa rằng "Let's rewrite the whole universe in Rust" đủ để thấy sự yêu mến của cộng đồng cho ngôn ngữ trẻ này.

Điểm đặc biệt của 2 ngôn ngữ này là chúng có nét giống C, viết nhanh như Python và chạy nhanh như C. Vừa dễ phát triển, vừa chạy hiệu quả... Liệu đây có phải là xu hướng thiết kế ngôn ngữ lập trình trong tương lai?

3.2. Những Ngôn Ngữ Lập Trình Sẽ Còn

Xuyên suốt bề dày lịch sử phát triển, không ít những ngôn ngữ lập trình được tạo ra đã phải chìm vào quên lãng. Tuy nhiên, vẫn có không ít các ngôn ngữ trở nên trường tồn theo thời gian và trở thành tiêu chuẩn trong lập trình.

Lấy ví dụ, *Assembly* tính đến nay vẫn không có cạnh tranh. Còn *C*, sao bao nhiêu lần người ta nỗ lực thay thế nó bằng *Go lang* hay *Carbon* thì vị trí của nó vẫn vững vàng cho đến hiện tại, và có lẽ trong tương lai cũng vậy.

3.3. Những Công Cụ Mới

Ứng dụng trí tuệ nhân tạo, Microsoft đã tạo ra Github Copilot — một con bot viết code theo ý của người sử dụng dựa trên những gì mà nó học được từ code của người khác. Tuy khả năng của Copilot ở thời điểm hiện tại là rất hạn chế vì nó chỉ giải quyết được những vấn đề rất dễ và rất lan man, nhưng Copilot cũng mở ra một tương lai thú vị về lập trình.

Ngoài ra, nhiều người còn tin rằng chúng ta sẽ code bằng những hình thức khác với bây giờ, tức là sẽ không cần đến chuột hay bàn phím truyền thống. Họ nghĩ đến Visual Programming (1990s), nghĩ đến lập trình bằng giọng nói, những màn hình, bàn phím ảo lơ lửng... Tuy nhiên tính đến nay vẫn không có gì thay thế được bàn phím. Một giây rời bàn phím là một giây lãng phí, điển hình có thể quan sát hiệu quả làm việc của các lập trình viên sử dụng Tiling Manager (chủ yếu trên Linux).

3.4. Những Kỹ Thuật Mới

Trong xuyên suốt quá trình hình thành và phát triển của ngôn ngữ lập trình, các kỹ thuật mới liên tục được hình thành và áp dụng. Trong tương lai cũng vậy.

Ví dụ hiện tại, khi bàn về cơ sở dữ liệu và lưu trữ thông tin, ta hay nghĩ đến dạng bảng ghi, hay đơn giản hơn là dạng plain text. Nhưng với sự hiệu quả mà *GraphGL* đang mang lại, biết đâu sẽ có một ngày thông tin sẽ được lưu trữ dưới dạng đồ thị?

3.5. Những Công Nghệ Mới

Máy tính lượng tử đang là một trong những công trình công nghệ được nghiên cứu sôi nổi hiện nay. Nếu như những chiếc máy này được hoàn thiện và thương mại hóa, chúng chắc chắn sẽ mang lại một cuộc cách mạng như máy tính của chúng ta đã làm được năm xưa.

Đến lúc đó, mọi kỹ thuật lập trình trên những cỗ máy lượng tử sẽ khác đi rất nhiều so với máy tính nhị phân. Đó có thể được xem là tương lai thú vị nhất, đột phá nhất của ngành lập trình.

Tài Liệu Tham Khảo

[1] Proxy Player HD, ”Haha just another naive beginner”, reddit, https://www.reddit.com/r/ProgrammerHumor/comments/p9blzg/comment/h9ww0iq/?context=3, Oct-24-2022

[2] Sylwester, ”Generating Fibonacci series in Lisp using recursion”, stackoverflow, https://stackoverflow.com/questions/23065846/generating-fibonacci-series-in-lisp-using-recursion, Oct-24-2022