SỰ PHÁT TRIỂN CỦA CÁC NGÔN NGỮ LẬP TRÌNH TRONG 20 NĂM TRỞ LẠI ĐÂY

Kiều Tiến Bình[11], Chung Vinh Hiển[12]

Tóm tắt: *Sự phát triển nhanh chóng của Internet vào giữa những năm 1990 là một động lực lớn cho thế giới ngôn ngữ lập trình tiếp tục phát triển trong cả ngành công nghiệp và nghiên cứu, khi các hệ thống và ứng dụng thay đổi. Ngày nay có rất nhiều ngôn ngữ lập trình với các ngôn ngữ, cú pháp và tính năng khác nhau. Các nhà phát triển hiện có thể sử dụng một ngôn ngữ dựa trên sở thích của khách hàng hoặc của riêng họ. Bài viết sau sẽ đi sâu vào lịch sử, sự phát triển và thịnh hành của các ngôn ngữ lập trình trong 20 năm qua. Ngoài việc phác thảo lịch sử của các ngôn ngữ, bài viết còn cung cấp thông tin về đặc trưng và ưu điểm của từng ngôn ngữ.*

Từ khóa: *Lập trình, Ngôn ngữ lập trình, Ngôn ngữ lập trình bậc cao, công nghệ thông tin.*

Abstract: *The rapid growth of the Internet in the mid 1990s was a major driver for the world of programming language to grow continuously in both industry and research while systems and applications have been changing. There are many programming languages with different languages, syntax and features today. Developers can now use a language based on customer preferences or their own. The following article will discuss deeply the history, development and prevalence of programming languages over the past 20 years. In addition to outlining the history , the article also provides information about the features and advantages of each language.*

Keywords: *Programming, Programming language, High-level programming language, Information technology.*

1. Giới thiệu

Lập trình là một việc làm mà ở đó lập trình viên sử dụng các ngôn ngữ lập trình, các đoạn mã lệnh (code) và các tiện ích có sẵn. Từ đó họ xây dựng nên các chương trình, các ứng dụng, các webside, các phần mềm và trò chơi,... để người dùng có thể giao tiếp với máy tính, hoặc tương tác qua lại với nhau. (Karen M, 2018)

Khi một người có thể lập trình xây dựng, tạo ra các chương thì được gọi là lập trình viên. Lập trình chỉ là một phần trong ngành công nghệ thông tin, chứ nó không phải là công nghệ thông tin. Nhiều lĩnh vực hay nhiều ngành khách nhau như y học, thương mại điện tử... là các ứng dụng thành tựu của công nghệ thông tin. (Dương Hoài Thương, 2020)

[11] Giảng viên Khoa Kỹ thuật - Công nghệ, Trường Đại học Nam Cần Thơ

[12] Sinh viên Khoa Kỹ thuật - Công nghệ, Trường Đại học Nam Cần Thơ

Ngôn ngữ lập trình là một loại ngôn ngữ của máy tính mang một hệ thống các quy tắc riêng đã được chuẩn hóa, sao cho cả con người và các thiết bị điện tử đều hiểu được thì lập trình viên có nhiệm vụ mô tả các chương trình làm việc dành cho thiết bị điện tử đó. (Nguyễn Tịnh, 2019)

Ngày nay trên thế giới thì có rất nhiều ngôn ngữ lập trình. Nhưng với tốc độ phát triển công nghệ thông tin được ví như vũ bão thì cứ mỗi năm có thể ra đời hàng chục ngôn ngữ lập trình. Hơn 700 ngôn ngữ lập trình đã cho ra đời. Tuy nhiên, không dừng lại ở con số đó bởi vì theo thống kê hàng năm liên tục có những ngôn ngữ mới được sinh ra. Do đó khó có thể xác định chính xác là bao nhiêu. (Nguyễn Linh Trang, 2018)

Với mỗi một ngôn ngữ lập trình luôn có những ứng dụng đặc trưng riêng. Thông thường mỗi một lập trình viên chỉ chọn một hoặc một vài ngôn ngữ lập trình nhất định để làm việc. Để trở thành một lập trình viên chuyên nghiệp, các lập trình viên phải nắm rõ và làm được bao gồm cả viết đoạn mã lệnh, cả thiết kế; xây dựng, bảo trì, sửa lỗi, nâng cấp các hệ thống. (Julie Luong, 2020[a])

Có rất nhiều ngông ngữ lập trình trên thế giới, cứ mỗi năm thống kê sẽ có một số ngôn ngữ mới ra đời. Vậy thì điểm chung của các ngôn ngữ đó là gì?. Dù là các ngôn ngữ khác nhau nhưng nhìn chung ngôn ngữ lập trình thường có 3 thành phần cơ bản đó là

- Bảng chữ cái: Để viết chương trình người lập trình phải dùng đúng các tập hợp ký tự quy định trong bảng chữ cái: 26 chữ cái thường(a, b, c, ..., z), 26 chữ cái in hoa (A, B, C, ..., Z), 10 chữ số thập phân (0, 1, 2, 3, ..., 9) và các kí tự đặc biệt.

- Cú pháp: là bộ quy tắc chung của một ngôn ngữ dùng để viết chương trình. Trong khi chương trình được dịch, lỗi cú pháp sẽ được phát hiện, chương trình được dịch xong cho đến khi không còn lỗi cú pháp nào.

- Ngữ nghĩa: Dựa vào ngử cảnh, các câu lệnh được viết có ý nghĩa, tính chất và thuộc tính khác nhau. Có thể phát hiện lỗi ngữ nghĩa khi chương trình được thực thi với những công việc hay dữ liệu cụ thể. Tức là ngữ nghĩa có nhiệm vụ xác định ý nghĩa của các tổ hợp kí tự được viết trong chương trình còn cú pháp kiểm tra tính hợp lệ của một chương trình. Lỗi cú pháp được chương trình dịch phát hiện còn lỗi ngữ nghĩa chỉ được phát hiện khi thực hiện chương trình trên dữ liệu cụ thể. (Anh Loan, 2019[a])

Bố cục của bài viết gồm 4 phần: Giới thiệu, từ năm 1988 đến năm 2000, từ năm 2000 đến nay và kết luận. Ở phần giới thiệu đưa ra các khái niệm cơ bản và giới thiệu tổng quát về ngôn ngữ lập tình, phân loại và sự hình thành của chúng, đồng thời nêu lên được các đặc điểm chung của chúng. Nội dung của phần 2 và 3 là sự phát triển của một số ngôn ngữ điển hình từ năm đầu tiên ra đời của máy tính cho đến nay và được cấu trúc gồm 3 phần. Đầu tiên tìm hiểu một số thông tin chung, kế tiếp là lịch sử và đặc trưng riêng, từ đó nêu ra ưu điểm của từng loại ngôn ngữ. Sau đó bài viết sẽ được đúc kết trong phần cuối.

1.1. Lịch sử

Trong giai đình 1842-1849, Ada Lovelace đã dịch cuốn hồi ký của nhà toán học người Ý Luigi Menabrea về cỗ máy đề xuất mới nhất của Charles Babbage: Động cơ phân tích; bà đã bổ sung hồi ký với các ghi chú được chỉ định chi tiết một phương pháp để tính toán số Bernoulli với động cơ, được hầu hết các nhà sử học công nhận là chương trình máy tính được xuất bản đầu tiên trên thế giới. (J. Fuegi and J. Francis, 2003)

Các mã máy tính đầu tiên được chuyên biệt cho các ứng dụng của họ: ví dụ, Alonzo Church đã có thể thể hiện tính toán lambda một cách có cấu trúc và máy Turing là một trừu tượng hóa hoạt động của một máy đánh chữ.

Jacquard Looms và Charles Babbage's Difference Engine đều có ngôn ngữ đơn giản để mô tả các hành động mà các máy này nên thực hiện do đó họ là những người tạo ra ngôn ngữ lập trình đầu tiên. (Vũ An, 2018)

1.2. Phân loại

Hàng nghìn ngôn ngữ lập trình khác nhau đã được tạo ra và nhiều ngôn ngữ lập trình khác đang được tạo ra hàng năm. Tuy nhiên, các ngôn ngữ lập trình có thể được phân thành 3 loại:

Ngôn ngữ máy - mã máy (machine language): là ngôn ngữ chính, quan trong và nền tảng của bộ vi xử lý. Đặc điểm chung của tất cả các loại ngôn ngữ khác nhau khi viết chương trình thì cuối cùng đều được chuyển thành ngôn ngữ máy trước khi chương trình đó được thi hành. Bộ vi xử lí có thể nhận biết và thực hiện một cách trực tiếp các chỉ thị trong ngôn ngữ máy được biểu diễn dưới dạng mã nhị phân. (Ann Dang, 2020)

Hợp ngữ (assembly language): là một ngôn ngữ lập trình bậc thấp, để viết chương trình nó dùng các từ viết tắt trong tiếng Anh. Hợp ngữ đã từng được sử dụng phổ biến và rộng rãi, nhưng ngày nay nó chỉ được dùng trong một số lĩnh vực nhất định, chủ yếu là để xử lý các vấn đề liên quan đến tốc độ cao hoặc giao tiếp trực tiếp với phần cứng. Ví dụ như các chương trình điều khiển các thiết bị, các hệ thống nhúng cấp thấp và các ứng dụng thời gian thực. Với nhược điểm là chương trình phức tạp, khó khăn trong việc ghi nhớ và phụ thuộc vào từng loại thiết bị. Đặc biệt, cần phải có công cụ hợp dịch để dịch từ hợp ngữ ra ngôn ngữ máy để thiết bị điện tử hiểu và thực thi được chương trình. (Ngọc Lam, 2020)

Ngôn ngữ lập trình bậc cao (High-level programming language): là một trong những ngôn ngữ gần với ngôn ngữ tự nhiên của con người, có tính độc lập cao, ít phụ thuộc vào phần cứng và các trình dịch. Một số ngôn ngữ lập trình bậc cao phổ biến hiện nay như: ngôn ngữ lập trình C, C++, C#, Java, PHP, Python. (Jordan Trần, 2019)

1.3. Đặc điểm chung

Mỗi ngôn ngữ lập trình có thể được xem như là một tập hợp của các chi tiết kỹ thuật chú trọng đến cú pháp, từ vựng, và ý nghĩa của ngôn ngữ. Những chi tiết kỹ thuật này thường bao gồm:

Kiểu dữ liệu: Hệ thống kiểu của ngôn ngữ lập trình là một hệ thống đặc thù với các dữ liệu được tổ chức sắp xếp trong một chương trình. Các kiểu dữ liệu thông dụng của nhiều ngôn ngữ lập trình được định nghĩa sẵn như: integer (biểu diễn các số nguyên), char (biểu diễn các ký tự đơn lẻ), string (biểu diễn chuỗi các ký tự, hay còn gọi là chuỗi, để tạo thành câu hay cụm từ). (Dương Phàm Thiên, 2019)

Cấu trúc dữ liệu: Để lắp đặt các cấu trúc dữ liệu phức tạp từ các kiểu có sẵn và để liên kết các tên với các kiểu mới kết hợp (dùng các kiểu mảng, danh sách, hàng đợi, ngăn xếp hay tập tin) thì đa số các ngôn ngữ đều cung cấp các cách thức. Các lập tình viên có thể định nghĩa các kiểu dữ liệu mới có riêng các hàm và các biến (các phương thức và các thuộc tính) được gọi là đối tượng trong ngôn ngữ hướng đối tượng. các đối tượng trong Một chương trình có thể được thực thi như là các chương trình con độc lập và có thể được thiết kế các tương tác để mô hình hóa và mô phỏng các ngôn ngữ hướng đối tượng để có riêng những tính năng hoạt động và tương tác với thế giới bên ngoài. Ngoài ra, một ưu thế trong việc dùng ngôn ngữ hướng đối tượng để mô tả các đối tượng là thừa kế và đa hình. (Nguyen Minh Trung Gia Dinh, 2015)

Các mệnh lệnh và dòng điều khiển: Sau khi xác định dữ liệu, cần chỉ cho máy tính cách thực hiện thao tác. Các câu cơ bản có thể được xây dựng bằng cách sử dụng các từ khóa (được định nghĩa trong ngôn ngữ lập trình), hoặc chúng có thể được kết hợp bằng cách sử dụng các cấu trúc ngữ pháp hoặc cú pháp hiện có. Điều này đã được xác định. Những nguyên tắc cơ bản này được gọi là câu lệnh. Có thể kết hợp các câu theo một thứ tự cụ thể tùy thuộc vào ngôn ngữ. Điều này cho phép chương trình được cấu hình để thực hiện nhiều chức năng. Ngoài các lệnh điều chỉnh và sửa đổi dữ liệu, còn có các lệnh để điều khiển luồng xử lý máy tính, chẳng hạn như các nhánh được xác định bởi nhiều phiên bản, vòng lặp hoặc các hàm kết hợp. Đây là những khối xây dựng cơ bản của một ngôn ngữ lập trình. *(Julie Luong, 2020[b])*

Các tên và tham số: Muốn cho chương trình thi hành được thì phải có phương pháp xác định được các vùng trống của bộ nhớ để làm kho chứa dữ liệu. Phương pháp được biết nhiều nhất là thông qua tên của các biến. Tùy theo ngôn ngữ, các vùng trống gián tiếp có thể bao gồm các tham chiếu, mà thật ra, chúng là các con trỏ (pointer) chỉ đến những vùng chứa khác của bộ nhớ, được cài đặt trong các biến hay nhóm các biến. Phương pháp này gọi là đặt tên kho nhớ. Tương tự với phương pháp đặt tên kho nhớ, là phương pháp đặt tên những nhóm của các chỉ thị. Trong hầu hết các ngôn ngữ lập trình, đều có cho phép gọi đến các macro hay các chương trình con như là các câu lệnh để thi hành nội dung mô tả trong các macro hay chương trình con này thông qua tên. Việc dùng tên như thế này cho phép các chương trình đạt tới một sự linh hoạt cao và có giá trị lớn trong việc tái sử dụng mã nguồn (vì người viết mã không cần phải lặp lại những đoạn mã giống nhau mà chỉ việc định nghĩa các macro hay các chương trình con). Các tham chiếu gián tiếp đến các chương trình khả dụng hay các bộ phận dữ liệu đã được xác định từ trước cho phép nhiều ngôn ngữ định hướng ứng dụng tích hợp được các thao tác khác nhau. (Anh Loan, 2019[b])

Cơ chế tham khảo và việc sử dụng mã nguồn: Để chạy một chương trình, cần có cách xác định dung lượng bộ nhớ có sẵn để lưu trữ dữ liệu. Phương pháp nổi tiếng nhất là sử dụng một tên

biến. Trong một số ngôn ngữ, khoảng trắng gián tiếp có thể thực sự chứa một con trỏ đến một vùng nhớ khác và một tham chiếu được triển khai trong một biến hoặc một nhóm các biến. Phương pháp này được gọi là đặt tên cho cửa hàng. Giống như cửa hàng được đặt tên, đó là một cách đặt tên cho một tập hợp các hướng dẫn. Trong hầu hết các ngôn ngữ lập trình, macro hoặc chương trình con có thể được gọi dưới dạng hướng dẫn để thực thi các mô tả của macro hoặc chương trình con đó theo tên. Việc sử dụng một tên như vậy cung cấp rất nhiều tính linh hoạt của chương trình và rất quan trọng khi sử dụng lại mã (người viết không cần phải lặp lại cùng một đoạn mã, họ chỉ cần xác định macro hoặc chương trình con). Nhiều mã có thể được thực hiện bằng cách gián tiếp tham chiếu đến các chương trình có sẵn hoặc các mục dữ liệu được xác định trước. Ngôn ngữ ứng dụng để tích hợp các hoạt động khác nhau. (Nguyễn Tịnh. 2019)

2. Từ năm 1952 đến năm 2000

2.1. Ngôn ngữ lập trình đầu tiên

Vào những năm 1940, các máy tính chạy bằng điện hiện đại đầu tiên được tạo ra. Tốc độ hạn chế và dung lượng bộ nhớ buộc các lập trình viên phải viết các chương trình ngôn ngữ lắp ráp thủ công. Cuối cùng người ta nhận ra rằng lập trình bằng ngôn ngữ lắp ráp đòi hỏi rất nhiều nỗ lực trí tuệ.

Một đề xuất ban đầu cho một ngôn ngữ lập trình cấp cao là Plankalkül, được phát triển bởi Konrad Zuse cho máy tính Z1 của ông từ năm 1943 đến năm 1945 nhưng không được thực hiện vào thời điểm đó. (Rojas, Raúl. 2000)

Các ngôn ngữ lập trình hoạt động đầu tiên được thiết kế để truyền đạt các hướng dẫn cho máy tính được viết vào đầu những năm 1950. Mã ngắn (Short Code) của John Mauchly, được đề xuất vào năm 1949, là một trong những ngôn ngữ cấp cao đầu tiên từng được phát triển cho một máy tính điện tử. Không giống như mã máy, các câu báo cáo Mã ngắn đại diện cho các biểu thức toán học ở dạng dễ hiểu. Tuy nhiên, chương trình đã phải được dịch sang mã máy mỗi khi nó chạy, làm cho quá trình chậm hơn nhiều so với chạy mã máy tương đương. (iTEK News, 2018)

Vào đầu những năm 1950, Alick Glennie đã phát triển Autocode, có thể là ngôn ngữ lập trình biên soạn đầu tiên, tại Đại học Manchester. Năm 1954, một lần lặp thứ hai của ngôn ngữ, được gọi là "Mark 1 Autocode", được phát triển cho Mark 1 bởi R. A. Brooker. Brooker cũng đã phát triển một mã tự động cho Ferranti Mercury vào những năm 1950 kết hợp với Đại học Manchester. Phiên bản cho EDSAC 2 được phát minh bởi Douglas Hartree của Phòng thí nghiệm Toán học Đại học Cambridge vào năm 1961. Được gọi là EDSAC 2 Autocode, nó là một sự phát triển thẳng từ Mercury Autocode thích nghi với hoàn cảnh thực tế, tối ưu hóa mã đối tượng và chẩn đoán ngôn ngữ nguồn được nâng cao. Một chủ đề phát triển hiện đại nhưng riêng biệt, Atlas Autocode được phát triển bởi Đại học Manchester Atlas 1.(Sebesta W.S, 2006)

Năm 1954, FORTRAN được phát minh bởi IBM do một nhóm dẫn đầu bởi John Backus; nó là ngôn ngữ lập trình mục đích chung cấp cao đầu tiên được sử dụng rộng rãi để có một thực hiện chức năng, trái ngược với một thiết kế trên giấy. Khi FORTRAN lần đầu tiên được giới

thiệu, nó được xem với sự hoài nghi do lỗi, sự chậm trễ trong phát triển và hiệu quả so sánh của các chương trình "mã hóa bằng tay" được viết trong hội đồng. Tuy nhiên, trong một thị trường phần cứng đang phát triển nhanh chóng; ngôn ngữ cuối cùng trở nên nổi tiếng với hiệu quả của nó. Nó vẫn là một ngôn ngữ phổ biến cho máy tính hiệu suất cao và được sử dụng cho các chương trình có điểm chuẩn và xếp hạng các siêu máy tính nhanh nhất thế giới. (Padua David, 2000)

Một ngôn ngữ lập trình ban đầu khác được phát minh bởi Grace Hopper ở Mỹ, được gọi là FLOW-MATIC. Nó được phát triển cho UNIVAC I tại Remington Rand trong giai đoạn từ năm 1955 đến năm 1959. Hopper phát hiện ra rằng khách hàng xử lý dữ liệu kinh doanh không thoải mái với ký hiệu toán học, và vào đầu năm 1955, bà và nhóm của mình đã viết một đặc điểm kỹ thuật cho một ngôn ngữ lập trình tiếng Anh và thực hiện một nguyên mẫu. Trình biên dịch FLOW-MATIC đã được công khai vào đầu năm 1958 và đã hoàn thành đáng kể vào năm 1959. Flow-Matic là một ảnh hưởng lớn trong việc thiết kế COBOL, vì chỉ có nó và hậu duệ trực tiếp aimaco của nó được sử dụng thực tế vào thời điểm đó. (Sammet Jean E. 1972)

2.2. Giai đoạn từ năm 1960 đến năm 1970

Giai đoạn từ cuối những năm 1960 đến cuối những năm 1970 đã mang lại một bước phát triển lớn của ngôn ngữ lập trình. Hầu hết các mô hình ngôn ngữ chính hiện đang được sử dụng đã được phát minh trong giai đoạn này:

- Speakeasy, được phát triển vào năm 1964 tại Phòng thí nghiệm Quốc gia Argonne (ANL) bởi Stanley Cohen, là một gói lập trình OOPS (hướng đối tượng, giống như gói số MATLAB, IDL và Mathematica) sau này. Speakeasy có cú pháp nền tảng Fortran rõ ràng. Nó lần đầu tiên đề cập đến tính toán vật lý hiệu quả trong nội bộ tại ANL, đã được sửa đổi để sử dụng nghiên cứu (như "Modeleasy") cho Hội đồng Dự trữ Liên bang vào đầu những năm 1970 và sau đó đã được cung cấp thương mại; Speakeasy và Modeleasy hiện vẫn đang được sử dụng. (Cohen Stanley, 1971)

- Simula, được phát minh vào cuối những năm 1960 bởi Nygaard và Dahl như một siêu tập hợp của Algol 60, là ngôn ngữ đầu tiên được thiết kế để hỗ trợ lập trình hướng đối tượng.(Nygaard Kristen, 1978)

- C- một ngôn ngữ lập trình hệ thống ban đầu, được phát triển bởi Dennis Ritchie và Ken Thompson tại Bell Labs từ năm 1969 đến 1973. (Fruderica. 2020)

- Smalltalk (giữa những năm 1970) cung cấp một thiết kế hoàn chỉnh của một ngôn ngữ hướng đối tượng. (Kay Alan, 2017)

- Prolog, được thiết kế vào năm 1972 bởi Colmerauer, Roussel, và Kowalski, là ngôn ngữ lập trình logic đầu tiên. (Bratko Ivan, 2012)

- ML đã xây dựng một hệ thống loại đa hình (được phát minh bởi Robin Milner vào năm 1973) trên đầu trang của Lisp, tiên phong đánh máy ngôn ngữ lập trình chức năng tĩnh. (Robin Milner, 1978)

Mỗi ngôn ngữ này sinh ra cả một gia đình hậu duệ, và hầu hết các ngôn ngữ hiện đại đều được biết đến ít nhất trong tổ tiên của họ. Những năm 1960 và 1970 cũng chứng kiến cuộc tranh luận đáng kể về giá trị của "chương trình có cấu trúc", về cơ bản có nghĩa là lập trình mà không cần sử dụng "goto". Một phần đáng kể các lập trình viên tin rằng, ngay cả trong các ngôn ngữ cung cấp "goto", đó là phong cách lập trình xấu để sử dụng nó ngoại trừ trong những trường hợp hiếm hoi. Cuộc tranh luận này có liên quan chặt chẽ đến thiết kế ngôn ngữ: một số ngôn ngữ hoàn toàn không bao gồm "goto", buộc phải lập trình có cấu trúc trên lập trình viên.

Để cung cấp thời gian biên dịch nhanh hơn, một số ngôn ngữ được cấu trúc cho "bỏ qua trình biên dịch"

2.3. Giai đoạn những năm 80 của thế kỷ 20

Những năm 1980 là những năm củng cố tương đối bằng các ngôn ngữ bắt buộc. Thay vì phát minh ra các mô hình mới, tất cả các phong trào này được xây dựng dựa trên những ý tưởng được phát minh trong thập kỷ trước. C++ kết hợp lập trình hướng đối tượng và hệ thống. Chính phủ Hoa Kỳ đã tiêu chuẩn hóa Ada- một ngôn ngữ lập trình hệ thống được thiết kế để sử dụng bởi các nhà thầu quốc phòng. Ở Nhật Bản và các nơi khác, một khoản tiền lớn đã được chi cho việc điều tra cái gọi là ngôn ngữ lập trình thế hệ thứ năm kết hợp các cấu trúc lập trình logic. Cộng đồng ngôn ngữ chức năng chuyển sang chuẩn hóa ML và Lisp. Nghiên cứu ở Miranda- một ngôn ngữ chức năng với đánh giá lười biếng , bắt đầu nắm giữ trong thập kỷ này.

Một xu hướng mới quan trọng trong thiết kế ngôn ngữ là tăng cường tập trung vào lập trình cho các hệ thống quy mô lớn thông qua việc sử dụng các mô-đun, hoặc các đơn vị tổ chức quy mô lớn của mã. Modula,Ada và ML đều phát triển các hệ thống mô-đun đáng chú ý trong những năm 1980. Các hệ thống mô-đun thường được kết hôn với các cấu trúc lập trình chung chung công nghệ, về bản chất, các mô-đun parameterized.

Mặc dù các mô hình mới ra đời cho các ngôn ngữ lập trình bắt buộc không xuất hiện, nhiều nhà nghiên cứu đã mở rộng ý tưởng về các ngôn ngữ trước đó và điều chỉnh chúng theo bối cảnh mới. Ví dụ, các ngôn ngữ của các hệ thống Argus và Emerald thích nghi lập trình hướng đối tượng cho các hệ thống phân tán.

Những năm 1980 cũng mang lại những tiến bộ trong việc thực hiện ngôn ngữ lập trình. Phong trào RISC trong kiến trúc máy tính giả định rằng phần cứng nên được thiết kế cho trình biên dịch hơn là cho các lập trình viên. Được hỗ trợ bởi những cải tiến tốc độ bộ xử lý cho phép các kỹ thuật biên dịch ngày càng tích cực, phong trào RISC đã gây ra sự quan tâm lớn hơn đến công nghệ biên dịch cho các ngôn ngữ cấp cao.

2.4. Giai đoạn những năm 90 của thế kỷ 20

Sự phát triển nhanh chóng của Internet vào giữa những năm 1990 là sự kiện lịch sử lớn tiếp theo trong ngôn ngữ lập trình. Bằng cách mở ra một nền tảng hoàn toàn mới cho các hệ thống máy tính, Internet đã tạo ra một cơ hội cho các ngôn ngữ mới được áp dụng. Đặc biệt, ngôn ngữ lập trình JavaScript trở nên phổ biến vì được tích hợp sớm với trình duyệt web Netscape Navigator. Nhiều ngôn ngữ đã được sử dụng rộng rãi trong việc phát triển các ứng

dụng tùy chỉnh cho các máy chủ web như PHP. Những năm 1990 không thấy sự mới lạ cơ bản trong các ngôn ngữ bắt buộc , nhưng nhiều sự tái tổ hợp và trưởng thành của những ý tưởng cũ. Thời đại này bắt đầu sự lây lan của các ngôn ngữ chức năng. Một triết lý thúc đẩy lớn là năng suất lập trình viên. Nhiều ngôn ngữ "phát triển ứng dụng nhanh chóng" (RAD) xuất hiện, thường đi kèm với một IDE, thu gom rác thải , và là hậu duệ của các ngôn ngữ cũ. Tất cả các ngôn ngữ như vậy đều hướng đối tượng. Chúng bao gồm Object Pascal, Visual Basic và Java. Java nói riêng nhận được nhiều sự chú ý.

Cấp tiến và sáng tạo hơn so với các ngôn ngữ RAD là các ngôn ngữ kịch bản mới. Chúng không trực tiếp đi xuống từ các ngôn ngữ khác và có các cú pháp mới và kết hợp các tính năng tự do hơn. Nhiều người xem xét các ngôn ngữ kịch bản được năng suất hơn ngay cả các ngôn ngữ RAD, nhưng thường vì sự lựa chọn mà làm cho các chương trình nhỏ đơn giản hơn nhưng các chương trình lớn khó khăn hơn để viết và duy trì. Tuy nhiên, ngôn ngữ kịch bản đã trở thành những ngôn ngữ nổi bật nhất được sử dụng liên quan đến Web.

3. Từ năm 2000 đến nay

3.1. C# năm 2001

3.1.1. Tổng quan

C # (hay C Sharp) là một trong những ngôn ngữ lập trình đơn giản được phát triển bởi một nhóm kỹ sư Microsoft vào năm 2000. C # là một ngôn ngữ lập trình hướng đối tượng hiện đại dựa trên hai trong số các ngôn ngữ mạnh mẽ nhất. C ++ và Java. (Thiên Phú, 2019)

Nó là một ngôn ngữ lập trình chạy trên nền tảng Microsoft và yêu cầu máy tính. Có thể coi .NET Framework là sự kết hợp của ngôn ngữ C và C ++. Từ đó, tận dụng tối đa lợi ích của hai ngôn ngữ này. (Hoàng Nam, 2014)

Vì những khả năng đặc biệt của nó, C # có thể được sử dụng để lập trình bất kỳ ứng dụng nào. Tuy nhiên, ưu điểm lớn nhất của nó là lập trình và phát triển ứng dụng web, ngày càng trở nên phổ biến trên nền tảng di động. (Nguyễn Hải, 2020)

C # được thiết kế cho Cơ sở hạ tầng ngôn ngữ chung (CLI), chứa mã thực thi và môi trường thời gian chạy, đồng thời có thể sử dụng các ngôn ngữ cấp cao khác nhau trên các nền tảng và kiến trúc máy tính khác nhau. (VietAds, 2021)

3.1.2. Lịch sử

Trong quá trình phát triển .NET Framework, các thư viện lớp ban đầu được viết bằng cách sử dụng một hệ thống trình biên dịch mã được quản lý được gọi là "Simple Managed C" (SMC). Vào tháng 1 năm 1999, Anders Hejlsberg thành lập một nhóm để xây dựng một ngôn ngữ mới vào thời điểm đó được gọi là Cool, viết tiền cho "Ngôn ngữ hướng đối tượng giống như C". Microsoft đã xem xét việc giữ tên "Cool" là tên cuối cùng của ngôn ngữ, nhưng đã chọn không làm như vậy vì lý do thương hiệu. Vào thời điểm dự án .NET được công bố công khai tại Hội nghị các nhà phát triển chuyên nghiệp tháng 7 năm 2000, ngôn ngữ đã được đổi tên thành C #, và các thư viện lớp và thời gian chạy ASP.NET đã được chuyển sang C #. (Hamilton Naomi, 2008)

Hejlsberg là nhà thiết kế chính của C #và kiến trúc sư trưởng tại Microsoft, và trước đây đã tham gia vào việc thiết kế Turbo Pascal, Embarcadero Delphi (trước đây là CodeGear Delphi, Inprise Delphi và Borland Delphi), và Visual J ++. Trong các cuộc phỏng vấn và các bài báo kỹ thuật, ông đã nói rằng những sai sót trong hầu hết các ngôn ngữ lập trình chính (ví dụ: C ++, Java, Delphi và Smalltalk) đã thúc đẩy các nguyên tắc cơ bản của Common Language Runtime (CLR), từ đó, thúc đẩy thiết kế của chính ngôn ngữ C#. (Wylie Wong, 2002)

3.1.3. Đặc trưng

C # là một ngôn ngữ đơn giản: C # loại bỏ một số phức tạp và nhầm lẫn từ các ngôn ngữ như Java và C ++ như macro, đã kế thừa và loại bỏ các lớp cơ sở ảo. Các giao diện, cú pháp, biểu thức, toán tử, v.v. đều dựa trên C và C ++, nhưng dễ lập trình và sử dụng. Loại bỏ sự trùng lặp không cần thiết và cải thiện cú pháp để sửa đổi. (Nguyễn Anh Tuấn, 2018)

C # là một ngôn ngữ hiện đại. Nó là một ngôn ngữ đơn giản nhưng rất hiện đại với tất cả các chức năng cần thiết. Tất cả các khái niệm lập trình mơ hồ mới mà đã học, chẳng hạn như xử lý ngoại lệ, kiểu mở rộng dữ liệu và bảo mật mã nguồn, đều được tích hợp vào C Sharp.

C # là một ngôn ngữ lập trình hướng đối tượng. Đối với lập trình hướng đối tượng, đây là kỹ thuật phải thỏa mãn đầy đủ 4 thuộc tính trừu tượng, đóng gói, đa hình (polymorphism) và kế thừa.

C # là một ngôn ngữ không có từ khóa. Sự ra đời của C # là sự bác bỏ quan điểm cho rằng ngôn ngữ càng có nhiều từ khóa thì nó càng có sức mạnh. Bằng chứng cho thấy không phải như vậy. C Sharp là ngôn ngữ giới hạn từ khóa trong khoảng 80 từ khóa và hơn 10 kiểu dữ liệu tích hợp, nhưng các chức năng mà nó cung cấp rất mạnh mẽ và đủ mạnh để thực hiện bất kỳ tác vụ nào. (Ngọc Muội, 2020)

3.1.4. Ưu điểm

Giao diện C rõ ràng tạo ra các cấu trúc ngôn ngữ giống như nhiều ngôn ngữ thông thường. Như vậy, người dùng không chỉ dễ dàng tiếp cận mà còn nhanh chóng làm quen với ngôn ngữ lập trình mới. (Hoàng Nam, 2014)

C Sharp dùng để biên dịch trên nhiều nền máy tính khác nhau. Được thừa hưởng nhiều ưu điểm từ các nền tảng như C, C ++, Java, v.v.. (Hoàng Nam, 2014)

Là một phần trong các sản phẩm của Microsoft và .NET Framework, phần mềm nhanh chóng nhận được sự tin tưởng của người dùng và được sử dụng rộng rãi trên khắp thế giới. Ngoài ra, C Sharp còn hỗ trợ Visual Studio IDE và nhiều plugin rất mạnh mẽ. (Hoàng Nam, 2014)

3.2. Groovy năm 2003

3.2.1. Tổng quan

Groovy là một ngôn ngữ lập trình hướng đối tượng dựa trên Java. Nó là một ngôn ngữ lập trình động với các tính năng tương tự như Python, Ruby, Perl và Smalltalk. Nó cũng có thể được sử dụng như một ngôn ngữ kịch bản chạy trên nền tảng Máy ảo Java. (Paul King, 2020)

Groovy sử dụng cú pháp tương tự như Java, nhưng không có dấu chấm phẩy ở cuối mỗi dòng, nó sẽ được tự động chuyển đổi thành bytecode và thực thi trong JVM (Java Virtual Machine). (Ly Nam, 2018)

3.2.2. Lịch sử

James Strachan lần đầu tiên nói về sự phát triển của Groovy trên blog của mình vào tháng 8 năm 2003. Vào tháng 3 năm 2004, Groovy đã được đệ trình lên JCP với tên JSR 241 và được chấp nhận bằng lá phiếu. Một số phiên bản đã được phát hành từ năm 2004 đến 2006. Sau khi nỗ lực chuẩn hóa Quy trình Cộng đồng Java (JCP) bắt đầu, việc đánh số phiên bản đã thay đổi và một phiên bản có tên "1.0" được phát hành vào ngày 2 tháng 1 năm 2007. Sau nhiều phiên bản beta và các ứng cử viên phát hành được đánh số 1.1, vào ngày 7 tháng 12 năm 2007, Groovy 1.1 Final đã được phát hành và ngay lập tức được đánh số lại là Groovy 1.5 để phản ánh nhiều thay đổi được thực hiện. (James Strachan, 2003)

3.2.3. Đặc trưng

Hầu hết các tệp Java hợp lệ cũng là các tệp Groovy hợp lệ. Mặc dù hai ngôn ngữ tương tự nhau, mã Groovy có thể nhỏ gọn hơn, bởi vì nó không cần tất cả các yếu tố mà Java cần. Điều này giúp các lập trình viên Java có thể học groovy dần dần bằng cách bắt đầu với cú pháp Java quen thuộc trước khi có được nhiều thành ngữ lập trình Groovy hơn. (Wayback Machine, 2010)

Các tính năng groovy không có sẵn trong Java bao gồm cả nhập tĩnh và động (với từ khóa), quá tải toán tử, cú pháp gốc cho danh sách và mảng kết hợp (bản đồ), hỗ trợ gốc cho biểu thức chính quy, lặp đa hình, nội suy chuỗi, thêm phương pháp trợ giúp và toán tử điều hướng an toàn để tự động kiểm tra con trỏ null. (Wayback machine, 2004)

Vì phiên bản 2 Groovy cũng hỗ trợ mô-đun (chỉ có thể vận chuyển các lọ cần thiết theo nhu cầu dự án, do đó làm giảm kích thước của thư viện groovy), kiểm tra loại, biên dịch tĩnh, cải tiến cú pháp Project Coin, khối multicast và cải tiến hiệu suất liên tục bằng cách sử dụng hướng dẫn được giới thiệu trong Java 7.

Groovy cung cấp hỗ trợ gốc cho các ngôn ngữ đánh dấu khác nhau như XML và HTML, được thực hiện thông qua cú pháp Mô hình Đối tượng Tài liệu nội tuyến (DOM). Tính năng này cho phép định nghĩa và thao tác của nhiều loại tài sản dữ liệu không đồng nhất với cú pháp và phương pháp lập trình thống nhất và súc tích.

Không giống như Java, một tập tin mã nguồn Groovy có thể được thực hiện như là một kịch bản (không biên dịch), nếu nó chứa mã bên ngoài bất kỳ định nghĩa lớp, nếu nó là một lớp với một phương pháp chính, hoặc nếu nó là một Runnable hoặc Groovy Test Case. Một kịch bản Groovy được phân tích đầy đủ, biên dịch, và tạo ra trước khi thực hiện (tương tự như Python và Ruby) và phiên bản biên dịch không được lưu như là một artifact của quá trình.

3.2.4. Ưu điểm

Groovy được tạo ra để khắc phục nhiều hạn chế của Java và giúp lập trình dễ dàng hơn. Groovy hỗ trợ tới 99% cú pháp Java, vì vậy có thể dễ dàng sao chép và dán mã Java vào Groovy

để thực thi. Ngoài ra, so với Java, Groovy có cú pháp riêng rất ngắn gọn giúp viết mã nhanh hơn. (Nhà sách tin học, 2020)

Groovy tương tác với các thư viện Java, vì vậy có thể khai báo và sử dụng tất cả các thư viện Java trong Groovy. Các lập trình viên Java sẽ thấy rất dễ dàng và thú vị khi chuyển sang mã Groovy. (Dương Minh Kiệt. 2020)

3.3. Scala năm 2003

3.3.1. Tổng quan

Scala loại bỏ một số cấu trúc phức tạp hơn khỏi khuôn khổ Java hoặc .NET và thêm chúng vào một số tính năng nâng cao hơn. (Odersky Martin, 2006)

Scala ngắn gọn, súc tích, dễ đọc và dễ học. Cấu trúc nhẹ và ngắn gọn của Scala cho phép các lập trình viên giảm kích thước mã của họ ít nhất từ hai đến ba lần so với Java. Do đó, mã nhanh hơn và dễ bảo trì hơn. (Odersky Martin, 2008)

Nó rất chính xác vì nó được trang bị hệ thống phát hiện và tránh nhiều lỗi ứng dụng tại thời điểm biên dịch.Ngôn ngữ này có tính mở rộng. Nó cung cấp một ngôn ngữ máy độc đáo, dễ dàng thêm thư viện, hỗ trợ việc lập trình dựa trên các thư viện hàm có sẵn.

3.3.2. Lịch sử

Thiết kế của Scala bắt đầu vào năm 2001 tại École Polytechnique Fédérale de Lausanne (EPFL) (ở Lausanne, Thụy Sĩ) bởi Martin Odersky . Nó được tiếp nối từ công việc trên Funnel, một ngôn ngữ lập trình kết hợp ý tưởng từ lập trình chức năng và lưới Petri. Odersky trước đây đã làm việc trên Generic Java và javac, trình biên dịch Java của Sun. (Martin Odersky, 2006)

Sau khi phát hành nội bộ vào cuối năm 2003, Scala được phát hành công khai vào đầu năm 2004 trên nền tảng Java. Một phiên bản thứ hai (v2.0) tiếp theo vào tháng 3 năm 2006.

3.3.3. Đặc trưng

Scala có mô hình biên dịch giống như Java và C #, cụ thể là biên dịch riêng biệt và tải lớp động, để mã Scala có thể gọi các thư viện Java.

Đặc điểm hoạt động của Scala cũng giống như Java. Trình biên dịch Scala tạo ra mã byte gần giống với mã được tạo bởi trình biên dịch Java. Trên thực tế, mã Scala có thể được dịch ngược thành mã Java có thể đọc được, ngoại trừ một số hoạt động của hàm tạo nhất định. Đối với máy ảo Java (JVM), không thể phân biệt được mã Scala và mã Java. Sự khác biệt duy nhất là một thư viện thời gian chạy bổ sung scala-library.jar. (Odersky M và Rompf T, 2014)

Scala bổ sung một số lượng lớn các tính năng so với Java và có một số khác biệt cơ bản trong mô hình biểu thức và kiểu cơ bản của nó, điều này làm cho ngôn ngữ này sạch hơn về mặt lý thuyết và loại bỏ một số trường hợp góc trong Java. (David Pollak, 2008)

3.3.4. Ưu điểm

Scala ngắn gọn, súc tích, dễ đọc, dễ học và dễ hiểu. Cấu trúc của Scala nhẹ và ngắn gọn đến mức các nhà phát triển có thể giảm kích thước mã nguồn của nó ít nhất từ hai đến ba lần so với Java. Do đó, mã nhanh hơn và dễ bảo trì hơn. (Bell, 2020)

Scala rất chính xác vì nó có thể phát hiện và tránh nhiều lỗi ứng dụng tại thời điểm biên dịch. Scala rất lớn. Nó cung cấp một ngôn ngữ máy duy nhất giúp dễ dàng thêm các cấu trúc ngôn ngữ mới làm thư viện, do đó giúp hỗ trợ lập trình dựa trên các thư viện hàm hiện có. (Trần Thị Thùy Dương, 2015) (Mỹ Phượng, 2020)

3.4. Go năm 2009

3.4.1. Tổng quan

Go là một ngôn ngữ lập trình được biên dịch, được nhập tĩnh được thiết kế tại Google bởi Robert Griesemer, Rob Pike và Ken Thompson.Về mặt cú pháp, Go tương tự như C, nhưng với tính năng an toàn bộ nhớ, thu gom rác, nhập cấu trúc, và đồng thời kiểu CSP. Ngôn ngữ này thường được gọi là Golang vì tên miền của nó, golang.org, nhưng tên riêng là Go. (Kincaid Jason, 2009)

3.4.2. Lịch sử

Go được thiết kế tại Google trong năm 2007 để cải thiện chương trình năng suất trong một kỷ nguyên đa lõi, mạng máy và lớn codebases. Các nhà thiết kế muốn giải quyết những lời chỉ trích về các ngôn ngữ khác đang được sử dụng tại Google, nhưng vẫn giữ các đặc điểm hữu ích của chúng: nhập tĩnh và hiệu quả thời gian chạy (như C), khả năng đọc và khả năng sử dụng (như Python hoặc JavaScript), mạng hiệu suất cao và đa xử lý. (Pike Rob, 2010)

3.4.3. Đặc trưng

Mặc dù các tính năng đồng thời của Go không chủ yếu nhắm vào xử lý song song, chúng có thể được sử dụng để lập trình các máy đa xử lý bộ nhớ dùng chung. Nhiều nghiên cứu khác nhau đã được thực hiện về hiệu quả của phương pháp này. Một trong những nghiên cứu này đã so sánh kích thước (tính bằng dòng mã) và tốc độ của các chương trình được viết bởi một lập trình viên dày dạn không quen với ngôn ngữ và việc sửa các chương trình này bởi một chuyên gia cờ vây (từ nhóm phát triển của Google), thực hiện tương tự đối với Chapel, Click và Intel TBB. Nghiên cứu phát hiện ra rằng những người không phải là chuyên gia có xu hướng viết các thuật toán chia để trị với một đi câu lệnh mỗi lần đệ quy, trong khi chuyên gia đã viết các chương trình đồng bộ hóa công việc phân phối bằng cách sử dụng một quy trình trên mỗi bộ xử lý. Các chương trình của chuyên gia thường nhanh hơn, nhưng cũng lâu hơn. (Tang Peiyi, 2010).

3.4.4. Ưu điểm

Kết quả nhanh chóng: Trình biên dịch nhanh làm cho nó hoạt động như một ngôn ngữ thông dịch. Không biết rằng nó sẽ biên dịch. Có thể nghĩ rằng đang sử dụng một ngôn ngữ dịch như Ruby.

Bảo mật: gõ tĩnh mạnh mẽ và thu thập rác. Một phím bấm mạnh có nghĩa là có thể chuyển bất kỳ dữ liệu nào ở bất kỳ đâu. Phải rõ ràng. Gõ tĩnh có nghĩa là trình biên dịch biết các kiểu của tất cả các biến. Go không có chuyển đổi loại theo mặc định. Ví dụ: uint8 và uint16 là các loại khác nhau (ngoại trừ một số trường hợp).

Dễ dàng làm việc với: Nó súc tích, rõ ràng và dễ đọc.

Hiện đại: Hỗ trợ tích hợp sẵn trong ngôn ngữ cho các ứng dụng phân tán nối mạng đa lõi và hơn thế nữa. (Đỗ Hoàng, 2019)

3.5. Swift năm 2014

3.5.1. Tổng quan

Swift là một ngôn ngữ lập trình đa mục đích, đa mô hình, được biên dịch bởi Apple Inc. và cộng đồng mã nguồn mở, được phát hành lần đầu vào năm 2014. Swift được phát triển để thay thế cho ngôn ngữ lập trình Objective-C trước đó của Apple, với tên gọi Objective-C hầu như không thay đổi kể từ đầu những năm 1980 và thiếu các đặc điểm ngôn ngữ hiện đại. Swift hoạt động với các khung Cocoa và Cocoa Touch của Apple, và một khía cạnh quan trọng trong thiết kế của Swift là khả năng tương tác với phần lớn mã Objective-C hiện có được phát triển cho các sản phẩm của Apple trong những thập kỷ trước. Nó được xây dựng với khung biên dịch LLVM mã nguồn mở và đã được đưa vào Xcode kể từ phiên bản 6, phát hành vào năm 2014. Trên nền tảng Apple, nó sử dụng thư viện thời gian chạy Objective-C cho phép C, Objective-C, C ++ và Swift mã để chạy trong một chương trình. (Timmer John, 2014)

3.5.2. Lịch sử

Việc phát triển Swift bắt đầu vào tháng 7 năm 2010 bởi Chris Lattner, với sự hợp tác cuối cùng của nhiều lập trình viên khác tại Apple. Swift lấy các ý tưởng ngôn ngữ "từ Objective-C, Rust, Haskell, Ruby, Python, C #, CLU và quá nhiều thứ khác để liệt kê". Vào ngày 2 tháng 6 năm 2014, ứng dụng Apple Worldwide Developers Conference (WWDC) đã trở thành ứng dụng phát hành công khai đầu tiên được viết bằng Swift. Một phiên bản beta của ngôn ngữ lập trình đã được phát hành cho các nhà phát triển đã đăng ký của Apple tại hội nghị, nhưng công ty không hứa rằng phiên bản Swift cuối cùng sẽ là mã nguồn tương thích với phiên bản thử nghiệm. Apple đã lên kế hoạch cung cấp các bộ chuyển đổi mã nguồn nếu cần cho bản phát hành đầy đủ. (Lattner Chris. 2014)

3.5.3. Đặc trưng

Swift là một sự thay thế cho ngôn ngữ Objective-C sử dụng các khái niệm lý thuyết ngôn ngữ lập trình hiện đại và cố gắng trình bày một cú pháp đơn giản hơn. Trong thời gian giới thiệu, nó được mô tả đơn giản là "Objective-C không có hành lý của C". (Metz Rachel, 2014). Theo mặc định, Swift không tiếp xúc với con trỏ và khác không an toàn accessors, trái ngược với Objective-C, trong đó sử dụng con trỏ pervasively để đề cập đến trường hợp đối tượng. Ngoài ra, việc sử dụng cú pháp giống Smalltalk của Objective-C để thực hiện các lệnh gọi phương thức đã được thay thế bằng kiểu ký hiệu dấu chấm và hệ thống không gian tên quen

thuộc hơn với các lập trình viên từ các ngôn ngữ hướng đối tượng (OO) phổ biến khác như Java hoặc C #. Swift giới thiệu các tham số được đặt tên đúng và giữ lại các khái niệm Objective-C chính, bao gồm các giao thức, bao đóng và danh mục, thường thay thế cú pháp cũ bằng các phiên bản rõ ràng hơn và cho phép các khái niệm này được áp dụng cho các cấu trúc ngôn ngữ khác, như kiểu liệt kê (enums). (Weber Harrison, 2014)

3.5.4. Ưu điểm

Phát triển ứng dụng di động nhanh chóng và dễ dàng. Thay vì thường xuyên chạy các trình biên dịch và chương trình thử nghiệm, các lập trình viên có thể tập trung vào các tích hợp phức tạp hơn. Nó cũng làm giảm mức tiêu thụ điện năng của phần cứng và chi phí cho các nhà phát triển. (Mỹ Phượng, 2020)

Mã của Swift ngắn gọn, súc tích và dễ đọc. Cụ thể, có thể viết 3-5 dòng mã Obj-C trong một dòng mã Swift. Các lớp đối tượng Swift đã được đơn giản hóa và các dòng mã được tổ chức hợp lý và logic hơn. Điều này không chỉ giúp lập trình viên tiết kiệm thời gian hoàn thành dự án mà còn đơn giản hóa quá trình bảo trì và sửa chữa các lỗi trong tương lai. Swift kế thừa và mở rộng tất cả các chức năng của Objective-C cũ, cung cấp cho các lập trình viên một thời gian chạy tốt, quen thuộc, dễ quản lý và phát triển. (Bell, 2020)

Swift hoạt động liên tục với Objective-C, vì vậy có thể viết ứng dụng bằng hai ngôn ngữ. Các ứng dụng do Swift xây dựng đều tuân thủ obj-C và tiêu tốn ít phần cứng hơn, mang đến cho người dùng trải nghiệm chơi game tuyệt vời trên thiết bị iOS. (Bell, 2020)

4. Kết luận

Lịch sử của các ngôn ngữ lập trình thật hấp dẫn. Ai có thể nghĩ rằng một thuật toán từ giữa thế kỷ 19 sẽ mở đường cho xã hội phát triển theo hướng công nghệ mà chúng ta đang sống ngày nay. Từ những mã máy ban đầu cho đến mã phức tạp mà con người có thể đọc được, cung cấp năng lượng cho các công nghệ yêu thích của chúng ta ngày nay, ngôn ngữ lập trình đã đi một chặng đường dài. Điều chắc chắn là, lập trình máy tính sẽ tiếp tục phát triển như nó đã làm trong 150 năm qua và thật thú vị khi thấy những gì tương lai mang lại.

TÀI LIỆU THAM KHẢO

[1] Karen McCandless. 2018, *What is Computer Programming?*, Codecademy, truy cập ngày 18 tháng 01 năm 2021, <https://news.codecademy.com/what-is-computer-programming/>.

[2] Dương Hoài Thương. 2020, "Lập trình là gì? Bạn hiểu như thế nào về lập trình" blog, ngày 13 tháng 11, *Code Gym*, truy cập ngày 18 tháng 01 năm 2021, <https://blog.codegym.vn/2020/11/13/lap-trinh-la-gi-ban-hieu-nhu-the-nao-ve-lap-trinh/>.

[3] Nguyễn Tịnh. 2019, *Ngôn ngữ lập trình là gì?*, Smartline and AZDIGI hosting, truy cập ngày 18 tháng 01 năm 2021, <https://hocban.vn/ngon-ngu-lap-trinh-la-gi>.

[4] Nguyễn Linh Trang. 2018, *Có bao nhiêu ngôn ngữ lập trình*, Niit-Ict Hà Nội, truy cập ngày 18 tháng 01 năm 2021, <https://niithanoi.vn/chi-tiet-tin/2028/co-bao-nhieu-ngon-ngu-lap-trinh.html>.

[5] Julie Luong. 2020, *Học lập trình là gì? Có khó không? Cần học từ đâu, học những gì?*, Magenest JSC, truy cập ngày 18 tháng 01 năm 2021, <https://magenest.com/vi/hoc-lap-trinh/>.

[6] Anh Loan. 2019, "Các thành phần của ngôn ngữ lập trình bạn cần biết" blog, ngày 19 tháng 11, *MindX Technology School*, truy cập ngày 18 tháng 01 năm 2021, <https://mindx.edu.vn/blog/post/cac-thanh-phan-cua-ngon-ngu-lap-trinh.>.

[7] J. Fuegi and J. Francis. 2003, "Lovelace & Babbage and the creation of the 1843 'notes'", Annals of the History of Computing, 25 (4): 16-26, "Lovelace & Babbage and the creation of the 1843 'notes'", Annals of the History of Computing, 25 (4): 16-26.

[8] Vũ An. 2018, *Đâu là ngôn ngữ lập trình đầu tiên trên thế giới?*, Công ty Cổ phần Mạng Trực tuyến Metam, truy cập ngày 18 tháng 01 năm 2021, <https://quantrimang.com/dau-la-ngon-ngu-lap-trinh-dau-tien-tren-the-gioi-147589>.

[9] iTEK News. 2018, *Đâu là ngôn ngữ lập trình đầu tiên trên thế giới?*, Kinh nghiệm tin học, truy cập ngày 18 tháng 01 năm 2021, <https://kinhnghiemtinhoc.com/dau-la-ngon-ngu-lap-trinh-dau-tien-tren-the-gioi/>.

[10] Ann Dang. 2020, *Ngôn ngữ máy là gì?*, Công ty Cổ phần SOC, truy cập ngày 18 tháng 01 năm 2021, <https://tuhoclaptrinh.edu.vn/bai-viet/ngon-ngu-may-la-gi-117.html>.

[11] Ngọc Lam. 2020, *Hợp ngữ là gì? Ứng dụng của hợp ngữ - Assembly Language,* Công ty Cổ phần Dịch vụ và Truyền thông Hưng Việt, truy cập ngày 18 tháng 01 năm 2021, <https://timviec365.com.vn/hop-ngu-la-gi-b527.html>.

[12] Jordan Trần. 2019, *Ngôn ngữ bậc cao và ngôn ngữ bậc thấp*, Howkteam, truy cập ngày 18 tháng 01 năm 2021, < https://www.howkteam.vn/course/goc-lap-trinh-vien/ngon-ngu-bac-cao-va-ngon-ngu-bac-thap-3936>.

[13] Dương Phàm Thiên. 2019, "Ngôn ngữ lập trình là gì?" blog, ngày 26 tháng 02, *Dương Thiên Expert*, truy cập ngày 18 tháng 01 năm 2021, <http://phamthienzblog.blogspot.com/2019/02/ngon-ngu-lap-trinh-la-gi.html.>.

[14] Nguyen Minh Trung Gia Dinh. 2015, Bài 5: Ngôn ngữ lập trình, Wordpress, truy cập ngày 18 tháng 01 năm 2021, <https://nguyenminhtrunggiadinh.wordpress.com/category/uncategorized/>.

[15] Julie Luong. 2020, *Ngôn ngữ lập trình là gì? 10 Ngôn ngữ phổ biến và 5 công dụng của chúng*, Magenest JSC, truy cập ngày 18 tháng 01 năm 2021, <https://magenest.com/vi/ngon-ngu-lap-trinh/>.

[16] Anh Loan. 2019, "Phân loại các ngôn ngữ lập trình cơ bản" blog, ngày 25 tháng 11, *MindX Technology School*, truy cập ngày 18 tháng 01 năm 2021, <https://mindx.edu.vn/blog/post/ngon-ngu-lap-trinh-la-gi>.

[17] Nguyễn Tịnh. 2019, *Phân loại ngôn ngữ lập trình*, Smartline and AZDIGI hosting, truy cập ngày 18 tháng 01 năm 2021, <https://hocban.vn/ngon-ngu-lap-trinh-la-gi>.

[18] Rojas Raúl.2000, "Plankalkül: The First High-Level Programming Language and its Implementation". Institut frame Informatik, Freie Universität Berlin, Technical Report B-3/2000.

[19] Sebesta, W.S. 2006. Concepts of Programming Languages. p. 44. ISBN 978-0-321-33025-3.

[20] Padua David. 2000, "The FORTRAN I Compiler" (PDF). Computing in Science and Engineering. 2 (1): 70-75.

[21] Sammet Jean E. 1972, "Programming Languages: History and Future". Communications of the ACM. 15 (7): 601-610.

[22] Thiên Phú. 2019, *C# là gì ? Tổng quan về C#*, KING Vietnam, truy cập ngày 18 tháng 01 năm 2021, <https://eduking.edu.vn/c-la-gi-tong-quan-ve-c/>.

[23] Hamilton Naomi. 2008, *The A-Z of Programming Languages: C#*. Computerworld, truy cập ngày 18 tháng 01 năm 2021, <https://www.computerworld.com/au/>.

[24] Ly Nam. 2018, *Giới thiệu về Groovy, nó có gì hay ho?* , SuSuDev, truy cập ngày 18 tháng 01 năm 2021, <https://susudev.com/gioi-thieu-ve-groovy-mot-ngon-ngu-lap-trinh-thu-vi.html>.

[25] James Strachan. 2003, *Groovy - the birth of a new dynamic language for the Java platform*, truy cập ngày 18 tháng 01 năm 2021, <https://web.archive.org/web/20030901064404/http://radio.weblogs.com/0112098/2003/08/29.html>.

[26] Timmer John. 2014*, A fast look at Swift, Apple's new programming language*, Ars Technica, Condé Nast, truy cập ngày 18 tháng 01 năm 2021, <https://arstechnica.com/gadgets/2014/06/a-fast-look-at-swift-apples-new-programming-language/>.

[27] Lattner Chris. 2014, *Chris Lattner's Homepage*. Chris Lattner., truy cập ngày 18 tháng 01 năm 2021, < http://nondot.org/sabre/>.

[28] Metz Rachel. 2014, *Apple Seeks a Swift Way to Lure More Developers,* Technology Review, truy cập ngày 18 tháng 01 năm 2021, <https://www.technologyreview.com/news/527821/apple-seeks-a-swift-way-to-lure-more-developers>.

[29] Weber Harrison. 2014, *Apple announces 'Swift,' a new programming language for macOS & iOS*. VentureBeat, truy cập ngày 18 tháng 01 năm 2021, <https://venturebeat.com/2014/06/02/apple-introduces-a-new-programming-language-swift-objective-c-without-the-c/>.

[30] Mỹ Phượng. 2020, *Những lý do nên sử dụng ngôn ngữ lập trình Swift*, truy cập ngày 18 tháng 01 năm 2021, <https://source.vn/ngon-ngu-lap-trinh-swift/>.

[31] Bell. 2020, *Ngôn ngữ lập trình Swift là gì?*, SOC, truy cập ngày 18 tháng 01 năm 2021, <https://tuhoclaptrinh.edu.vn/bai-viet/ngon-ngu-lap-trinh-swift-la-gi-178.html>.

[32] Nguyễn Hải. 2020, *Ngôn ngữ lập trình C sharp là gì? Ưu nhược điểm C#*, chiasekinang.com, truy cập ngày 18 tháng 01 năm 2021, <https://chiasekinang.com/ngon-ngu-c-sharp/>.

[33] VietAds. 2021, *C# Là Gì? Tìm Hiểu Về C# Là Gì?*, VietAds, truy cập ngày 18 tháng 01 năm 2021, <https://vietadsgroup.vn/c-la-gi-tim-hieu-ve-c-la-gi-.html>.

[34] Wylie Wong. 2002, *Why Microsoft's C# isn't*, CNET, truy cập ngày 18 tháng 01 năm 2021, <https://www.cnet.com/news/why-microsofts-c-isnt/>.

[35] Nguyễn Anh Tuấn. 2018, Giới thiệu ngôn ngữ C#, nguyenanhtuanweb.wordpress, truy cập ngày 18 tháng 01 năm 2021, <https://nguyenanhtuanweb.wordpress.com/2018/02/27/gioi-thieu-ngon-ngu-c/>.

[36] Ngọc Muội. 2020, *C sharp là gì? Những điều cần biết về C Sharp*, my webside, truy cập ngày 18 tháng 01 năm 2021, <https://ntna009.mywebsite.vn/c-sharp-la-gi-nhung-dieu-can-biet-ve-c-sharp.html>.

[37] Paul King. 2020, *A history of the Groovy programming language*, ACM, Inc, truy cập ngày 18 tháng 01 năm 2021, <https://dl.acm.org/doi/10.1145/3386326>.

[38] Wayback Machine. 2010, *Groovy style and language feature guidelines for Java developers*, truy cập ngày 18 tháng 01 năm 2021, <https://web.archive.org/web/20150117214709/http://groovy.codehaus.org/Groovy+style+and+language+feature+guidelines+for+Java+developers>.

[39] Wayback Machine. 2010, *Groovy - Differences from Java,* Groovy.codehaus.org, truy cập ngày 18 tháng 01 năm 2021, <https://web.archive.org/web/20090317025737/http://groovy.codehaus.org/Differences+from+Java>.

[40] Guillaume LaForge. 2012, *What's new in Groovy 2.0?*, truy cập ngày 18 tháng 01 năm 2021, <https://www.infoq.com/articles/new-groovy-20/>.

[41] Nhà sách tin học. 2020, "Chia Sẻ Khóa Học Dành Cho Nhà Phát Triển Apache Groovy Toàn Tập" blog, ngày 7 tháng 5, *Nhà sách tin học*, truy cập ngày 18 tháng 01 năm 2021, <http://nhasachtinhoc.blogspot.com/2020/05/chia-se-khoa-hoc-danh-cho-nha-phat-trien-groovy.html>.

[42] Dương Minh Kiệt. 2020, *Vì sao ngôn ngữ lập trình Java được sử dụng nhiều ở các doanh nghiệp*, truy cập ngày 18 tháng 01 năm 2021, <https://cpc.vn/vi-vn/Tin-tuc-su-kien/Tin-tuc-chi-tiet/articleId/37071.

[43] Odersky Martin. 2006, "An Overview of the Scala Programming Language" (PDF) (2nd ed.). École Polytechnique Fédérale de Lausanne (EPFL), truy cập ngày 18 tháng 01 năm 2021, <https://www.scala-lang.org/docu/files/ScalaOverview.pdf>.

[44] Odersky, Martin (2008). *Programming in Scala. Mountain View*, California: Artima. p. 3, truy cập ngày 18 tháng 01 năm 2021, <https://books.google.com.vn/books?id=MFjNhTjeQKkC&q=scala+is+pronounced+skah-la%20is%20pronounced%20skah-lah&f=false>.

[45] Martin Odersky. 2006, "A Brief History of Scala" blog, ngày 9 tháng 6, *Artima.com weblogs*, truy cập ngày 18 tháng 01 năm 2021, <https://www.artima.com/weblogs/viewpost.jsp?thread=163733>.

[46] Odersky M và Rompf T. 2014, "Unifying functional and object-oriented programming with Scala", truy cập ngày 18 tháng 01 năm 2021, <https://dl.acm.org/doi/10.1145/2591013>.

[47] David Pollak, 2008, "For all you know, it's just another Java library" blog, *wayback machine*, truy cập ngày 18 tháng 01 năm 2021, https://web.archive.org/web/20100831041226/http://blog.lostlake.org/index.php?%2Farchives%2F73-For-all-you-know%2C-its-just-another-Java-library.html.

[48] Trần Thị Thùy Dương. 2015, *Sơ lược về ngôn ngữ lập trình Scala*, Viblo, truy cập ngày 18 tháng 01 năm 2021, <https://viblo.asia/p/so-luoc-ve-ngon-ngu-lap-trinh-scala-1l0rvmx0GyqA>.

[49] Kincaid, Jason. 2009, *Google's Go: A New Programming Language That's Python Meets C++*. TechCrunch. truy cập ngày 18 tháng 01 năm 2021, <https://techcrunch.com/2009/11/10/google-go-language/>.

[50] Pike Rob. 2010, *Another Go at Language Design*, Stanford EE Computer Systems Colloquium, truy cập ngày 18 tháng 01 năm 2021, <https://web.stanford.edu/class/ee380/Abstracts/100428.html>.

[51] Tang Peiyi. 2010, Multi-core parallel programming in Go (PDF). Proc. First International Conference on Advanced Computing and Communications, truy cập ngày 18 tháng 01 năm 2021, <https://ualr.edu/pxtang/papers/acc10.pdf>.

[52] Đỗ Hoàng. 2019, "Tổng quan về Golang?Golang là gì? Tại sao nên dùng Golang?" blog, ngày 8 tháng 11, *Nordic Coder*, truy cập ngày 18 tháng 01 năm 2021, <https://nordiccoder.com/blog/golang-la-gi/>.

[53] Cohen Stanley. 1971, "The Delphi-speakeasy system. I. Overall description". Computer Physics Communications. 2: 1-10.

[54] Nygaard Kristen. 1978, "The Development of the Simula Languages" (PDF), The Development of the SIMULA, truy cập ngày 18 tháng 01 năm 2021, <https://hannemyr.com/cache/knojd_acm78.pdf>.

[55] Fruderica. 2020, *History of C*, truy cập ngày 18 tháng 01 năm 2021, <https://en.cppreference.com/w/c/language/history>.

[56] Kay Alan. 2017, *The Early History of Smalltalk*, Apple Computer, Gagne.homedns.org, truy cập ngày 18 tháng 01 năm 2021, <http://gagne.homedns.org/~tgagne/contrib/EarlyHistoryST.html>.

[57] Bratko Ivan. 2012, Prolog programming for artificial intelligence (4th ed.). Harlow, England ; New York: Addison Wesley.

[58] Robin Milner. 1978, A theory of type polymorphism in programming. Journal of Computer and System Sciences, 17(3):348-375.

[59] Hoàng Nam. 2014, *C sharp là gì? Những điều cần biết về c sharp*, Long Vân System Solution, truy cập ngày 18 tháng 01 năm 2021, <https://longvan.net/c-sharp-la-gi.html>.