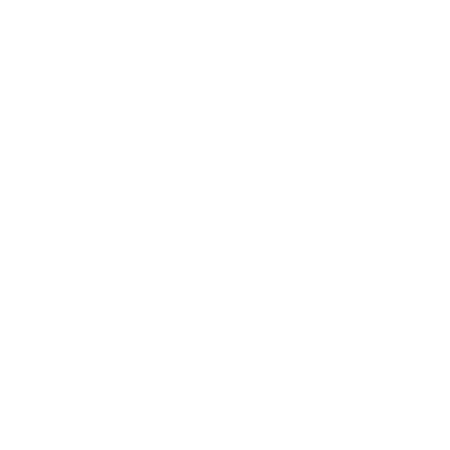


**Mục lục**

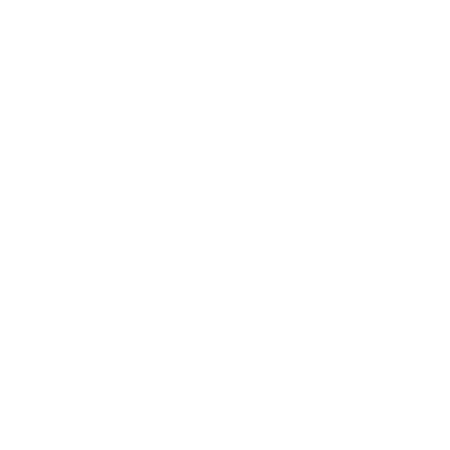
****

***Lời mở đầu ................................................................................................................ 1 Hướng dẫn sử dụng sách ........................................................................................ 2 Chương 1 - Nhập môn lập trình............................................................................... 5***

**1. Mục tiêu..........................................................................................................................5 2. Giới thiệu........................................................................................................................5 3. Máy tính hoạt động như thế nào?................................................................................5 4. Phần mềm được tạo ra như thế nào? .........................................................................9 5. Quy trình tạo ra một phần mềm .................................................................................10 6. Các vai trò trong lập trình...........................................................................................11 7. Ngôn ngữ lập trình ......................................................................................................13 8. Thuật toán ....................................................................................................................14 9. Mô tả thuật toán bằng mã giả.....................................................................................15 10. Mô tả thuật toán bằng lưu đồ...................................................................................16 11. Một số cấu trúc thường gặp trong thuật toán........................................................17 12. Ngôn ngữ lập trình JavaScript.................................................................................22 13. Chương trình đầu tiên ..............................................................................................22 14. Cài đặt công cụ lập trình...........................................................................................23 15. Bài thực hành ............................................................................................................24 16. Bài tập.........................................................................................................................25 17. Bài kiểm tra ................................................................................................................26 18. Tổng kết......................................................................................................................28** 

***Chương 2 - Biến, kiểu dữ liệu và toán tử ............................................................. 30* 1. Mục tiêu........................................................................................................................30 2. Giới thiệu......................................................................................................................30 3. Biến...............................................................................................................................30 4. Kiểu dữ liệu..................................................................................................................34 5. Phép toán toán học .....................................................................................................36 6. Phép toán logic............................................................................................................40 7. Phép toán so sánh.......................................................................................................43 8. Đọc dữ liệu từ bên ngoài............................................................................................44**

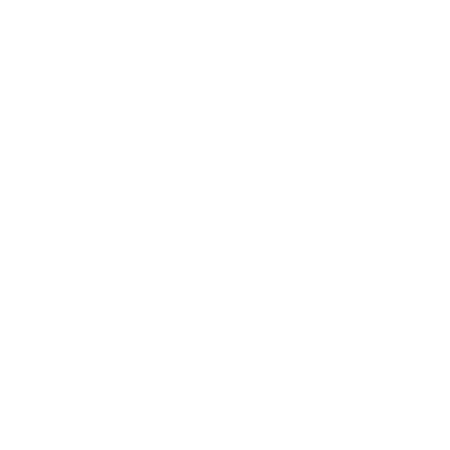
****

**9. Hiển thị dữ liệu ............................................................................................................46 10. Mã sạch ......................................................................................................................47 11. Các lỗi thường gặp ...................................................................................................48 13. Bài thực hành ............................................................................................................49 14. Bài tập.........................................................................................................................50 15. Bài kiểm tra ................................................................................................................50 16. Tổng kết......................................................................................................................52** 

***Chương 3 - Câu lệnh điều kiện.............................................................................. 53* 1. Mục tiêu........................................................................................................................53 2. Giới thiệu......................................................................................................................53 3. Cấu trúc điều kiện .......................................................................................................54 4. Cấu trúc điều kiện if-else............................................................................................54 5. Cấu trúc điều kiện if-else lồng nhau..........................................................................56 6. Cấu trúc điều kiện if-else bậc thang..........................................................................57 7. Cấu trúc điều kiện switch-case..................................................................................58 8. Mã sạch ........................................................................................................................61 9. Các lỗi thường gặp .....................................................................................................62 10. Bài thực hành ............................................................................................................66 11. Bài tập.........................................................................................................................77 12. Bài kiểm tra ................................................................................................................77 13. Tổng kết......................................................................................................................79**

***Chương 4 - Câu lệnh lặp ........................................................................................ 80* 1. Mục tiêu........................................................................................................................80 2. Giới thiệu......................................................................................................................80 3. Câu lệnh lặp .................................................................................................................80 4. Câu lệnh lặp while .......................................................................................................81 5. Câu lệnh lặp do-while..................................................................................................82 6. Câu lệnh lặp for ...........................................................................................................82 7. Câu lệnh lặp lồng nhau...............................................................................................84 8. Câu lệnh break.............................................................................................................85 9. Câu lệnh continue .......................................................................................................86 10. Mã sạch ......................................................................................................................86**

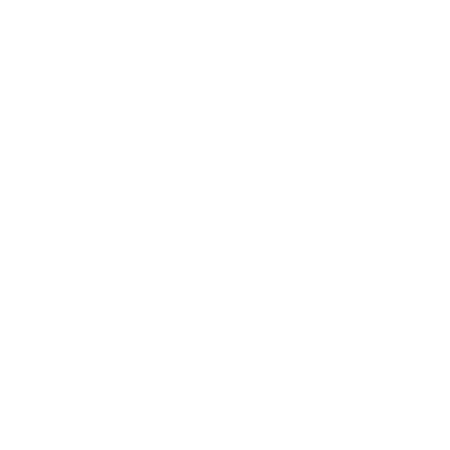
****

**11. Các lỗi thường gặp ...................................................................................................87 12. Bài thực hành ............................................................................................................88 13. Bài tập.........................................................................................................................93 14. Bài kiểm tra ................................................................................................................94 15. Tổng kết......................................................................................................................96** 

***Chương 5 - Mảng .................................................................................................... 97* 1. Mục tiêu........................................................................................................................98 2. Giới thiệu......................................................................................................................98 3. Mảng .............................................................................................................................98 4. Duyệt qua các phần tử của mảng............................................................................102 5. Mảng nhiều chiều ......................................................................................................103 6. Các thao tác thông dụng với mảng .........................................................................106 7. Mã sạch ......................................................................................................................108 8. Các lỗi thường gặp ...................................................................................................109 9. Bài thực hành ............................................................................................................110 10. Bài tập.......................................................................................................................115 11. Bài kiểm tra ..............................................................................................................116 12. Tổng kết....................................................................................................................118**

***Chương 6 - Hàm.................................................................................................... 119* 1. Mục tiêu......................................................................................................................119 2. Giới thiệu....................................................................................................................119 3. Hàm.............................................................................................................................120 4. Giá trị trả về của hàm................................................................................................124 5. Tham số của hàm ......................................................................................................125 6. Phạm vi của biến .......................................................................................................125 7. Hàm đệ quy ................................................................................................................126 8. Mã sạch ......................................................................................................................127 9. Các lỗi thường gặp ...................................................................................................129 10. Bài thực hành ..........................................................................................................130 11. Bài tập.......................................................................................................................133 12. Bài kiểm tra ..............................................................................................................134 13. Tổng kết....................................................................................................................136**

****

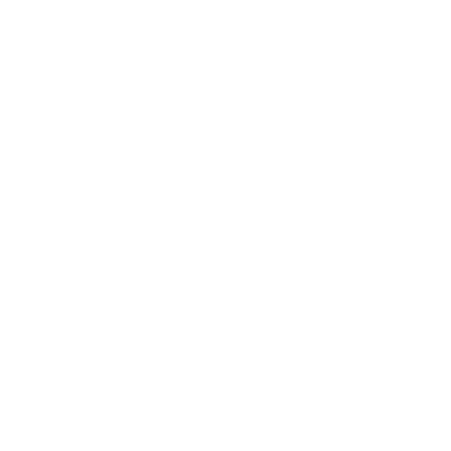
***Chương 7 - Thuật toán tìm kiếm.......................................................................... 137* 1. Mục tiêu......................................................................................................................137 2. Giới thiệu....................................................................................................................137 3. Tìm kiếm tuyến tính...................................................................................................137 4. Tìm kiếm nhị phân.....................................................................................................139 5. Độ phức tạp của thuật toán......................................................................................142 6. Các lỗi thường gặp ...................................................................................................144 7. Bài thực hành ............................................................................................................146 8. Bài tập.........................................................................................................................150 9. Bài kiểm tra ................................................................................................................150 10. Tổng kết....................................................................................................................151** 

***Chương 8 - Thuật toán sắp xếp........................................................................... 152* 1. Mục tiêu......................................................................................................................152 2. Giới thiệu....................................................................................................................152 3. Thuật toán sắp xếp nổi bọt.......................................................................................152 4. Thuật toán sắp xếp chèn ..........................................................................................155 5. Thuật toán sắp xếp chọn ..........................................................................................157 6. Các lỗi thường gặp ...................................................................................................159 8. Bài thực hành ............................................................................................................159 9. Bài tập.........................................................................................................................161 10. Bài kiểm tra ..............................................................................................................162 11. Tổng kết....................................................................................................................162**

***Tổng kết................................................................................................................. 164 Đội ngũ tác giả ...................................................................................................... 167***

******

Lời mở đầu

Với sự phát triển nhanh chóng của ngành Công nghệ Thông tin trong những năm gần đây, nhu cầu nhân sự trong ngành này tăng mạnh ở cả quy mô thế giới lẫn Việt Nam. Học lập trình không còn là một lựa chọn xa lạ với nhiều thanh niên Việt Nam nữa. Càng ngày càng có nhiều bạn trẻ muốn tìm đến với công việc lập trình, bắt đầu bằng việc tham gia vào các trường đào tạo chính quy, các trung tâm dạy lập trình, hoặc tự 

học.

Với hàng chục năm làm việc trong ngành Công nghệ Thông tin và trực tiếp tham gia vào công tác đào tạo lập trình viên, chúng tôi nhận ra rằng các bạn trẻ gặp rất nhiều khó khăn trong những thời điểm đầu tiên tiếp xúc với công việc này. Có rất nhiều rào cản khiến cho việc học lập trình trở nên khó khăn hơn, có thể kể đến như do chương trình đào tạo được thiết kế không tốt, giáo trình khó hiểu, chương trình không sát với thực tế, giáo trình không giúp người học xây dựng được kiến thức một cách bài bản và vững vàng. Bên cạnh đó, mặc dù có rất nhiều sách dạy lập trình bằng tiếng Anh rất tốt, nhưng nhiều người lại không sử dụng được tiếng Anh. Đó là những thiệt thòi và khó khăn mà những bạn trẻ đang muốn tìm đến ngành nghề lập trình đang gặp phải hiện nay.

Với mục tiêu giúp những người mới bắt đầu học lập trình được thuận lợi hơn, đội ngũ Giảng viên của Hệ thống Đào tạo Lập trình Hiện đại CodeGym đã biên soạn nên cuốn sách này và gửi đến các bạn. Cuốn sách này dành cho những người bắt đầu từ con số 0 và mong muốn phát triển năng lực lập trình trong thời gian nhanh nhất. Cuốn cẩm nang được thiết kế một cách khoa học để giúp người học tự xây dựng năng lực của mình thông qua từng hoạt động một. Không có quá nhiều thuật toán phức tạp, nhanh chóng làm ra được các sản phẩm để chứng minh sự tiến bộ của người học, sử dụng ngôn ngữ lập trình Javascript với độ tương tác cao là những đặc điểm nổi bật của cuốn cẩm nang này.

Mặc dù đội ngũ tác giả chúng tôi đã nỗ lực trong việc hoàn thiện cuốn sách này với tiêu chí dễ hiểu, khoa học và hiệu quả dành cho người mới bắt đầu, tuy nhiên khó để tránh khỏi các sai sót trong quá trình biên soạn. Vì vậy, chúng tôi rất mong nhận được

các ý kiến phản hồi và đóng góp của mọi người thông qua email info@codegym.vn. Cảm ơn và chúc các bạn gặt hái được nhiều giá trị từ cuốn sách này.

*Trân trọng!*

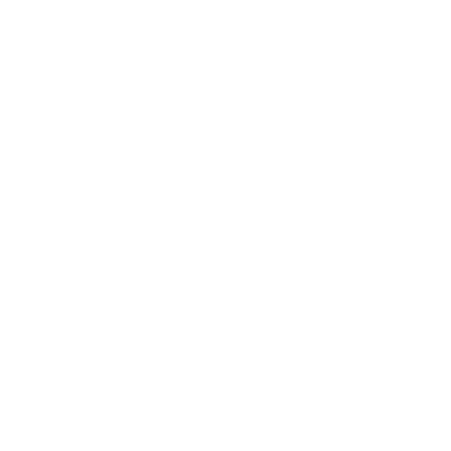
*Đội ngũ tác giả.*

/codegym.vn/ - 1



Hướng dẫn sử dụng sách

Cuốn sách này hướng đến mục tiêu giúp người học phát triển cả *kiến thức* lẫn *kỹ năng*. Cho nên, để khai thác cuốn sách một cách hiệu quả nhất, người học nên thực hiện các công việc sau đây:

● **Ghi chép**: Cần có một cuốn sổ để ghi chép, hoặc sử dụng các công cụ số như các trình soạn thảo hoặc Mind Map. Việc ghi chép sẽ giúp người học đảm bảo được mức độ hiểu của mình đối với các khái niệm quan trọng. Ghi chép cũng giúp cho người đọc ghi nhớ tốt hơn các khái niệm. 

● **Tìm kiếm thông tin ngoài cuốn sách**: Có những khái niệm mà người đọc không hiểu được, ngay cả khi đã đọc đi đọc lại nhiều lần. Người đọc nên tìm kiếm thông tin về các khái niệm đó ở những nguồn khác ngoài cuốn sách, để tìm cách hiểu chúng. Lí do là bởi vì có nhiều cách để trình bày một khái niệm nào đó, các tác giả mặc dù đã cố gắng để trình bày chúng một cách dễ hiểu và phù hợp với phần đông các độc giả, nhưng có thể ở một vài chỗ và đối với một vài người thì vẫn bị khó hiểu. Đừng đầu hàng, mà nên tìm cách để hiểu chúng ở phạm vi ngoài cuốn sách.

● **Luyện tập**: Thực hành là thao tác bắt buộc để có thể phát triển kỹ năng. Người học không thể có được kỹ năng tốt nếu chỉ đọc cuốn sách từ đầu tới cuối mà không thực hành trong thực tế. Việc thực hành nên tăng theo cấp độ từ dễ đến khó. Đối với các bài thực hành có hướng dẫn, người học có thể làm theo các hướng dẫn đó. Nhưng sau khi làm theo thì cần phải tiếp tục thực hành để mình có thể tự làm được mà không cần làm theo hướng dẫn.

● **Tổng kết sau mỗi chương**: Sau khi hoàn thành mỗi chương, người học cần xem lại phần ghi chép của mình, đối chiếu với các mục tiêu để xem thử mình đã phát triển được những năng lực gì. Nếu có phần nào mình chưa hiểu hoặc chưa làm được thì hãy cố gắng tìm thêm sự trợ giúp để có thể giải quyết đầy đủ.

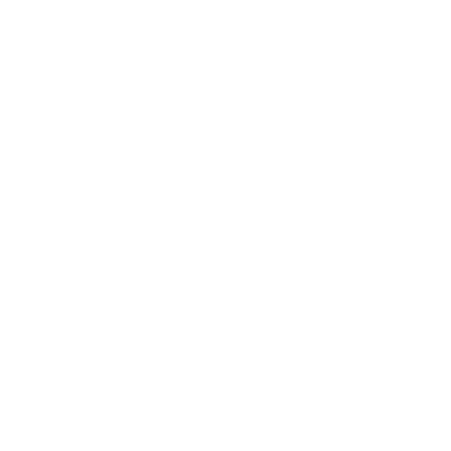
Cuốn sách này bao gồm 8 chương, lần lượt đề cập đến các khái niệm nền tảng trong lập trình. Tuy nhiên, nội dung của các chương không hoàn toàn được trình bày một cách tuyến tính, mà được tổ chức theo lối tích hợp. Nghĩa là một khái niệm có thể được trình bày trải đều qua nhiều chương khác nhau, giúp cho người học có cơ hội cọ xát nhiều lần, lặp đi lặp lại các thao tác thực hành, giúp xây dựng được một nền tảng hiểu biết và kỹ năng bền vững. Sau đây là cấu trúc của mỗi chương, người đọc nên hiểu cấu trúc này để có cách khai thác nội dung một cách hiệu quả nhất:

● **Mục tiêu**: Là nơi định nghĩa rõ ràng và ngắn gọn về những kiến thức và kỹ năng mà người đọc sẽ xây dựng được khi hoàn thành mỗi chương. Người học cần ghi chép lại mục tiêu của từng chương, để sau này tự đánh giá lại xem mình đã đạt được các mục tiêu đó hay chưa. Nếu không nắm được các mục

/codegym.vn/ - 2



tiêu thì người học sẽ rơi vào trạng thái hoang mang, bởi vì không biết mình đang hướng đến điều gì.

● **Giới thiệu**: Là nơi trình bày ngắn gọn về mục đích và nội dung của chương. Phần này giúp người học trả lời được câu hỏi WHAT (tôi đang học cái gì?) và WHY (thấy được sự liên quan giữa những khái niệm mình sẽ học với các ứng dụng thực tế). 

● **Các mục nội dung**: Phần chính của mỗi chương sẽ trình bày về các *khái niệm* và *ứng dụng* của chúng. Mỗi khái niệm sẽ được trình bày đầy đủ *định nghĩa*, *diễn giải*, *các tình huống ứng dụng, các vị dụ cụ thể* và những thông tin khác xung quanh khái niệm đó.

● **Mã sạch**: Mã sạch là một phần đặc biệt của cuốn sách để định hướng người đọc trở thành một lập trình viên tốt ngay từ đầu. Trong lập trình, có nhiều cách để giải quyết một vấn đề, nhưng chỉ có một số cách thực sự gọn gàng và hiệu quả. Mã sạch là cách để chúng ta tổ chức mã nguồn thực sự khoa học và hướng đến sự hiệu quả nói chung.

● **Các lỗi thường gặp**: Phần này liệt kê các lỗi mà một lập trình viên non kinh nghiệm thường mất nhiều thời gian để nhận ra và khắc phục. Chúng ta không thể liệt kê hết tất cả các lỗi có thể xảy ra khi làm việc với một cấu trúc lập trình nào đó, tuy nhiên việc nhận biết được các lỗi thường gặp sẽ giúp cho những người mới học lập trình tiết kiệm được khá nhiều thời gian.

● **Bài thực hành**: Đây là nội dung được thiết kế theo hướng “*cầm tay chỉ việc*”, có tác dụng giúp người đọc bắt đầu rèn luyện kỹ năng. Các bài thực hành được thiết kế theo hướng tăng dần về độ khó và quy mô, tương ứng với sự kỳ vọng tăng trưởng về kỹ năng của người đọc. Kết thúc phần *Bài thực hành*, người đọc có thể bắt tay vào tự thực hiện các bài tập để nâng cao hơn nữa kỹ năng của mình.

● **Bài tập**: Là cách thức để người học tự rèn luyện kỹ năng. Nếu nhìn vào lộ trình tổng quan, chúng ta sẽ đi từ việc học lí thuyết (thông qua phần *Mục tiêu, Giới thiệu, Các khái niệm, Các ví dụ*,…) rồi sau đó đến rèn luyện kỹ năng (thông qua phần *Thực hành* và *phần Bài tập*). Khi hoàn thành phần *Bài tập* thì kỹ năng của người đọc sẽ thành thạo hơn so với khi chỉ mới hoàn thành phần *Bài thực hành*.

● **Bài kiểm tra**: Đây là mục giúp người đọc tự đánh giá lại được kiến thức và kỹ năng của mình sau khi hoàn thành được một chương. Đáp án của các câu hỏi được đặt ở phần cuối của bài kiểm tra, giúp người học tự chấm điểm được. Nếu có những khái niệm nào mà mình chưa trả lời được thì người học nên quay trở lại phần trước đó để đọc kỹ hơn.

/codegym.vn/ - 3



● **Tổng kết**: Là nơi giúp người học rà soát lại nhanh những kiến thức và kỹ năng mình đã học được. Phần tổng kết này được viết sẵn với nội dung khá ngắn gọn, nhưng chúng tôi vẫn khuyến khích người học nên tự viết lại phần tổng kết của riêng mình với mức độ chi tiết hơn theo cách hiểu của mình. Người học cũng tự đối chiếu lại với các mục tiêu được liệt kê ở đầu chương, để xem mình đã đạt được các mục tiêu đó hay chưa.

/codegym.vn/ - 4



Chương 1 - Nhập môn lập trình

Các khái niệm căn bản về máy tính và lập trình

1. Mục tiêu 

● Giải thích được tổng quan cách hoạt động của máy tính

● Phân biệt được phần cứng và phần mềm

● Liệt kê được một số phần cứng thông dụng

● Liệt kê được một số phần mềm thông dụng

● Biết được vai trò của phần cứng, phần mềm và người dùng

● Trình bày được tổng quan quá trình tạo ra một phần mềm

● Giải thích được vai trò và ý nghĩa của ngôn ngữ lập trình

● Biết được tổng quan quy trình sản xuất một phần mềm

● Phân biệt được các vai trò thông dụng trong một quy trình sản xuất phần mềm ● Giải thích được ý nghĩa của thuật toán

● Mô tả được thuật toán bằng mã giả

● Mô tả được thuật toán bằng lưu đồ

● Cài đặt được phần mềm để bắt đầu viết mã

● Tạo được ứng dụng phần mềm đầu tiên

2. Giới thiệu

Trong chương đầu tiên của cuốn sách, chúng ta sẽ tìm hiểu về những khái niệm căn bản nhất về thế giới máy tính và lập trình. Đây là những khái niệm cơ bản nhất, cần thiết nhất trước khi bắt đầu viết những dòng mã đầu tiên. Chúng ta sẽ tìm hiểu về những khái niệm liên quan đến máy tính, phần mềm, người dùng, công việc sản xuất phần mềm, các bước để làm ra phần mềm. Cuối cùng, chúng ta sẽ cài đặt môi trường và chuẩn bị cho việc tạo ra một phần mềm đầu tiên.

Kết thúc chương này, chúng ta có thể bắt tay vào viết được những dòng mã đầu tiên để tạo ra các phần mềm.

3. Máy tính hoạt động như thế nào?

3.1. Định nghĩa máy tính

Thuật ngữ máy tính thường được sử dụng để chỉ đến các thiết bị điện tử có khả năng xử lý dữ liệu và chuyển đổi dữ liệu thành những thông tin hữu ích. Máy tính có được các khả năng này là nhờ sự kết hợp giữa các vi mạch điện tử và các tập lệnh do các lập trình viên tạo nên.

/codegym.vn/ - 5



Có thể phân loại máy tính theo nhiều tiêu chí khác nhau, chẳng hạn như về mục đích sử dụng, kích thước, cấu tạo hoặc công nghệ. Một số loại máy tính thường được nhắc đến là: Máy tính tính lớn (mainframe), máy tính nhỏ (minicomputer), siêu máy tính (supercomputer), máy vi tính (microcomputer) hoặc máy tính cá nhân (personal computer), máy tính xách tay (laptop). Các thiết bị như điện thoại thông minh (smart

phone) hoặc đồng hồ thông minh (smart watch) cũng là một dạng máy tính. Thông thường, khi người ta nói đến một hệ thống máy tính thì bao gồm 3 thành phần là: Phần cứng, Phần mềm và Người dùng. 

3.2. Phần cứng

Các thành phần vật lý kiến tạo lên máy tính được gọi là *phần cứng*. Phần cứng là bất cứ bộ phận nào của máy tính mà bạn có thể chạm tay vào được. Phần cứng máy tính bao gồm các *thiết bị điện tử tích hợp với nhau* được sử dụng để điều khiển các hoạt động nhập xuất và xử lí của máy tính. Một số phần cứng cơ bản bao gồm bộ nguồn điện, CPU (Central Processing Unit - Bộ xử lý trung tâm), RAM (Random Access Memory – Bộ nhớ Truy xuất Ngẫu nhiên), Motherboard (Bảng mạch chính, còn gọi là Mainboard), một số card mở rộng, thiết bị ngoại vi và các thành phần khác.

*Hình 1.1: Các thành phần cấu tạo của máy tính*

● Bảng mạch chính (Motherboard): Là bảng mạch quan trọng nhất, có nhiệm vụ kết nối các thành phần khác lại với nhau để hoạt động trong một thể thống nhất ● CPU – Bộ xử lý trung tâm: Có thể coi như bộ não của máy tính; thiết bị này tổ chức và thực hiện các chỉ thị của người dùng hoặc phần mềm

/codegym.vn/ - 6



● Bộ nhớ: Là một bảng mạch điện tử nhỏ bên trong máy tính. Chương trình chạy trên máy tính sẽ được nạp vào bộ nhớ và chạy từ đó. Bộ nhớ được phân thành hai loại là bộ nhớ sơ cấp và bộ nhớ thứ cấp. Bộ nhớ sơ cấp còn được gọi là bộ nhớ chính. Chúng bao gồm RAM (Random Access Memory – Bộ nhớ Truy xuất Ngẫu nhiên) hoặc ROM (Read-Only Memory – Bộ nhớ Chỉ đọc). Bộ nhớ thứ cấp đề cập tới các bộ lưu trữ trong hoặc ngoài được sử dụng cho các dữ liệu bền vững như đĩa mềm, ổ băng từ, đĩa quang (CD) hoặc ổ USB, v.v. 

● Thiết bị đầu vào: Cho phép nhận dữ liệu và các chỉ thị từ người dùng hoặc từ hệ thống máy tính khác. Chẳng hạn như: bàn phím, chuột, đầu đọc đĩa CD, camera, màn hình cảm ứng, v.v.

● Thiết bị đầu ra: Cho phép hiển thị kết quả thực thi các mệnh lệnh. Chẳng hạn như: màn hình, máy in, loa, máy chiếu, v.v.

3.3. Phần mềm

Phần mềm máy tính là tập hợp các chương trình máy tính và các dữ liệu có liên quan để cung cấp cho máy tính các chỉ dẫn cần thiết về những gì mà nó phải thực hiện. Trái ngược hẳn với phần cứng, phần mềm là một thứ vô hình, chúng ta không thể chạm được vào nó. Bạn có thể liên tưởng rằng máy tính như một thực thể sống với *phần xác* (đó là phần cứng) và *phần hồn* (đó là phần mềm). Với phần cứng, bạn có thể lắp ráp thành một máy tính. Tuy nhiên, máy tính cần phải có phần mềm để thực hiện các nhiệm vụ của mình.

Phần mềm máy tính có thể phân ra thành một số loại chính như phần mềm hệ thống, phần mềm ứng dụng và phần mềm lập trình.

● Phần mềm hệ thống: là những phần mềm máy tính được thiết kế để vận hành các thiết bị phần cứng và cung cấp, duy trì một nền tảng để chạy các phần mềm ứng dụng. Chẳng hạn như: Hệ điều hành, phần mềm diệt vi-rút, phần mềm quản lý mạng cho máy tính, v.v.

● Phần mềm ứng dụng (hoặc còn gọi tắt là ứng dụng): được thiết kế để giúp người dùng thực hiện một hay nhiều các công việc cụ thể nào đó. Ví dụ như các phần mềm doanh nghiệp, phần mềm kế toán, bộ phần mềm văn phòng, phần mềm xử lý ảnh và các phần mềm nghe nhạc, xem video, v.v.

● Phần mềm lập trình, những phần mềm này giúp các lập trình viên máy tính tạo ra các phần mềm khác. Ví dụ như phần mềm để viết mã nguồn, phần mềm để biên dịch, phần mềm để cài đặt, v.v.

3.4. Hệ điều hành

Hệ điều hành là một dạng phần mềm đặc biệt, trực tiếp được cài đặt lên các phần cứng để điều khiển chúng, cung cấp môi trường để các phần mềm khác có thể hoạt động, và đồng thời cũng cung cấp môi trường để người dùng tương tác với máy tính.

/codegym.vn/ - 7





Hình 1.2: Mối liên quan giữa Người dùng - Ứng dụng – Hệ điều hành – Phần cứng 

Có rất nhiều các hệ điều hành khác nhau được sử dụng cho các hệ thống máy tính, có thể kể đến như: Microsoft Windows, MacOS, các dòng Linux khác nhau (như Ubuntu, Debian, CentOS, Fedora…). Ngoài ra, còn có các hệ điều hành dành riêng cho các thiết bị di động, chẳng hạn như: Android, iOS, KaiOS,… Thậm chí, ngày nay, một số đơn vị đã nghĩ đến việc hợp nhất các hệ điều hành cho máy tính cá nhân với các hệ điều hành dành cho các thiết bị di động để gia tăng tính tương thích.

3.5. Người dùng

Những người sử dụng máy tính để làm các công việc cụ thể nào đó được gọi là Người dùng (User). Do máy tính chỉ là các công cụ, cho nên chúng cần phải có người điều khiển thì mới đem lại các lợi ích cụ thể. Người dùng có thể được phân loại là power user (Người dùng quyền lực), đó là những người dùng hiểu biết về hệ thống máy tính, hoặc end user (Người dùng cuối), đó là những người dùng không cần có quá nhiều hiểu biết về máy tính, mà chỉ cần học cách sử dụng các phần mềm đã được tạo sẵn để xử lý các nghiệp vụ hằng ngày.

3.6. Dữ liệu

Dữ liệu bao gồm các sự việc độc lập hoặc các mẩu thông tin, bản thân chúng thường không mang lại ý nghĩa cho con người. Máy tính đọc và lưu trữ dữ liệu ở dạng như văn bản, số liệu, hình ảnh hoặc âm thanh dưới cùng một dạng đó là các con số. Do đó, dữ liệu máy tính là dữ liệu số, nghĩa là chúng được tối giản xuống thành số. Nhìn chung, nghiệp vụ tính toán quan trọng nhất của máy tính là tập hợp dữ liệu (được gọi là đầu vào - input), xử lý chúng thành các dữ liệu đầu ra (output) hữu ích cho con người.

Hầu hết các máy tính ngày nay đều sử dụng hệ thống số nhị phân, bao gồm 2 giá trị là 0 và 1. Ngoại trừ một số máy tính rất đặc biệt khác, có thể có các dạng dữ liệu khác,

/codegym.vn/ - 8



chẳng hạn như máy tính lượng tử. Ở trong máy tính, có thể sử dụng một số cơ chế khác nhau để biểu diễn giá trị 0 và 1, chẳng hạn cơ chế quang học (phản xạ thì là 1, không phản xạ là 0), cơ chế từ trường (nam thì 0, bắc thì 1), cơ chế hiệu điện thế (không có điện thế là 0, có điện thế là 1),…

Mỗi một giá trị 0 hoặc 1 khi lưu trữ trong bộ nhớ thì được gọi là 1 *bit*. Một nhóm 8 *bit* thì được gọi là 1 *byte*. Byte là đơn vị lưu trữ nhỏ nhất mà máy tính sử dụng. Sau byte còn có các đơn vị khác lớn hơn để biểu diễn độ lớn của dữ liệu, chẳng hạn như: kilobyte (**kb**), megabyte (**MB)**, gigabyte (**GB**), terabyte (**TB**)... Bảng chuyển đổi giữa các đơn vị đo lường bộ nhớ được mô tả dưới đây: 

| **Đơn vị** | **Ký hiệu** | **Giá trị xấp xỉ (byte)** | **Giá trị chính xác (byte)** |
| --- | --- | --- | --- |
| Kilobyte | KB | 1 000 | 1024 |
| Megabyte | MB | 1 000 000 | 1 048 576 |
| Gigabyte | GB | 1 000 000 000 | 1073 741 824 |
| Terabyte | TB | 1 000 000 000 000 | 1099 511 627 776 |

*Bảng 1.1: Giá trị của các đơn vị đo độ lớn của dữ liệu*

4. Phần mềm được tạo ra như thế nào?

Phát triển phần mềm là một công việc bao gồm rất nhiều hoạt động khác nhau, đòi hỏi sự cộng tác, hợp tác giữa nhiều người với nhiều kỹ năng khác nhau. Có thể hình dung một cách đơn giản thì công việc này bao gồm các hoạt động như Định nghĩa yêu cầu, Thiết kế phần mềm, Lập trình, Sử dụng.

*Hình 1.3: Các công đoạn chính của việc phát triển phần mềm*

/codegym.vn/ - 9



● Định nghĩa yêu cầu: Tìm hiểu, phân tích để nắm rõ nhu cầu của người dùng về phần mềm mà mình sắp phát triển.

● Thiết kế phần mềm: Dựa trên các thông tin đã thu thập được, chúng ta phân tích và đưa ra các cách để xử lý vấn đề của người dùng thông qua các tính năng của phần mềm.

● Lập trình: Dựa trên bản thiết kế về các tính năng đã có, người lập trình viên sẽ viết ra các dòng lệnh để biến bản thiết kế đó trở thành một phần mềm có thể chạy thực sự. 

● Sử dụng: Sau khi phần mềm đã hoàn tất thì nó được bàn giao cho người dùng để xử lý các tác vụ mà trước đó đã định nghĩa.

Trên đây, chúng ta chỉ mới liệt kê các công đoạn chính để làm ra một phần mềm. Trong thực tế, có thể có thêm rất nhiều các công đoạn khác, hoặc là cách sắp xếp các công việc khác, tuỳ thuộc vào quy trình phát triển phần mềm của nhóm hoặc tổ chức.

5. Quy trình tạo ra một phần mềm

Để tạo ra một phần mềm, cần có sự cộng tác giữa rất nhiều người, sử dụng và chia sẻ các tài nguyên trong một khoảng thời gian, do đó việc đưa ra các quy trình làm việc là cần thiết.

Thời kỳ đầu, khi mới xuất hiện máy tính và phần mềm, các phần mềm thường nhỏ và đơn giản. Nhưng ngày nay, các hệ thống phần mềm thường rất lớn, đòi hỏi rất nhiều công sức để phát triển. Do đó, rất khó để một lập trình viên có thể hoàn thành được hết các công việc trong một khoảng thời gian cần thiết. Các phần mềm thường được sản xuất bởi các nhóm, hoặc nhiều nhóm cộng tác với nhau. Để cho việc cộng tác giữa các cá nhân và các nhóm được diễn ra thuận lợi thì chúng ta thiết lập các quy trình.

Quy trình là gì? Quy trình được hiểu đơn giản là các quy định về trình tự các bước để làm việc. Ai làm việc gì? Vào lúc nào? Sử dụng công cụ gì? Tiêu chuẩn gì?... Có nhiều dạng quy trình khác nhau được sử dụng trong các nhóm phần mềm, có thể liệt kê như: Thác nước (Waterfall), Xoắn ốc (Spiral), Scrum… Trong những năm gần đây, triết lý Agile và các phương pháp của nó đã được truyền bá và trở nên rất thông dụng.

/codegym.vn/ - 10





*Hình 1.4: Minh họa về quy trình thác nước (waterfall)*

6. Các vai trò trong lập trình

Có nhiều người tham gia vào trong quá trình làm ra các phần mềm, họ sẽ đóng các vai trò khác nhau và thực hiện các công việc khác nhau. Sau đây là danh sách một số các vai trò thường thấy:

**Kỹ sư phần mềm (Software Engineer)**

Kỹ sư phần mềm là người thiết kế và lập trình phần mềm ở mức hệ thống. Kỹ sư phần mềm là người hiểu các chức năng của hệ thống, trao đổi với khách hàng để xác định các chức năng của hệ thống đang xây dựng. Kỹ sư phần mềm là người giao tiếp nhiều và đồng thời cũng có nền tảng kỹ thuật và các kỹ năng lập trình tốt. Kỹ sư phần mềm còn được biết đến với các tên gọi tương tự như: Kiến trúc sư phần mềm, Kỹ sư hệ thống.

**Chuyên gia phân tích hệ thống (Systems Analyst)**

Chuyên gia Phân tích Hệ thống là người nghiên cứu và phân tích các vấn đề nghiệp vụ để sau đó đưa ra các thiết kế hệ thống thông tin nhằm cung cấp giải pháp, việc này thường xuất phát từ yêu cầu từ các bộ phận kinh doanh hoặc từ khách hàng. Chuyên gia Phân tích Hệ thống thu thập các yêu cầu và xác định chi phí cũng như thời gian cần thiết để triển khai dự án. Công việc này đòi hỏi việc kết hợp giữa các kỹ năng nghiệp vụ và kiến thức kỹ thuật, đồng thời phải giao tiếp tốt với các bên. Chuyên

/codegym.vn/ - 11



gia phân tích hệ thống còn được biết đến với các tên gọi như: Chuyên gia sản phẩm, Kỹ sư hệ thống, Chuyên gia giải pháp, Nhà thiết kế kỹ thuật.

**Chuyên gia phân tích nghiệp vụ (Business Analyst)**

Chuyên gia Phân tích Nghiệp vụ là người đóng vai trò trung gian quan trọng, làm việc với cả đội ngũ kỹ thuật, các cấp quản lý và với người dùng cuối. Chuyên gia Phân tích Nghiệp vụ là người đưa ra các cải tiến về quy trình và hoạt động nghiệp vụ thông qua việc sử dụng các công nghệ kỹ thuật. Vai trò này được xác định theo từng dự án, bắt đầu bằng việc phân tích các nhu cầu của khách hàng, thu thập và tài liệu hoá các yêu cầu, lập kế hoạch để xây dựng thiết kế cho giải pháp công nghệ. Chuyên gia Phân tích Nghiệp vụ cần phải có hiểu biết về công nghệ, tuy nhiên không nhất thiết phải là một chuyên gia công nghệ. Chuyên gia phân tích nghiệp vụ còn được biết đến với các tên gọi: Kiến trúc sư nghiệp vụ, Chuyên gia thông tin. 

**Chuyên viên Hỗ trợ Kỹ thuật (Technical Support)**

Chuyên viên Hỗ trợ Kỹ thuật là người giải quyết các vấn đề trong quá trình hoạt động của các hệ thống. Nhiều chuyên gia hỗ trợ kỹ thuật làm việc trong các công ty sản xuất và cung cấp phần cứng, và cũng có nhiều chuyên gia hỗ trợ kỹ thuật ở các doanh nghiệp nhằm hỗ trợ, theo dõi và bảo trì các hệ thống được sử dụng hằng ngày. Nhiều công việc đòi hỏi các chuyên gia với nền tảng và kinh nghiệm kỹ thuật tốt.

**Kỹ sư mạng (Network Engineer)**

Kỹ sư mạng là một trong các công việc rất cần thiết trong ngành IT, là người thực hiện các thao tác cài đặt, quản trị, duy trì và nâng cấp các hệ thống giao tiếp, xử lý các vấn đề liên quan đến mạng lưới trong các công ty. Kỹ sư mạng cũng là người chịu trách nhiệm về bảo mật, lưu trữ dữ liệu và các chiến lược khôi phục nếu có sự cố xảy ra. Kỹ sư mạng còn biết đến với tên gọi: Kỹ sư phần cứng, Chuyên gia mạng.

**Quản lý Dự án (Project Manager)**

Quản lý Dự án là người tổ chức các nhóm phát triển, phân bố thời gian và tài nguyên để đảm bảo các dự án đạt được các yêu cầu về chức năng, đúng thời gian và nằm trong ngân sách cho phép. Quản lý Dự án điều phối tất cả các hoạt động từ khi mới bắt đầu dự án cho đến khi kết thúc. Vai trò này đòi hỏi kinh nghiệm và nền tảng vững chắc về kỹ thuật cũng như các kỹ năng mềm để làm việc tốt với các nhóm phát triển và các nhà quản lý cấp cao. Quản lý Dự án còn biết đến với tên gọi: Trưởng dự án.

**Nhà phát triển (Developer)**

Nhà phát triển là người trực tiếp tạo ra phần mềm thông qua việc sử dụng các ngôn ngữ lập trình và các công cụ hỗ trợ. Nhà phát triển thường làm việc cộng tác trong các nhóm để đảm bảo các tính năng của sản phẩm được xây dựng và đáp ứng được các yêu cầu như thiết kế ban đầu. Vai trò này đòi hỏi kiến thức và kỹ năng tốt về công nghệ, công cụ và các ngôn ngữ lập trình. Nhà phát triển còn được biết đến với các

tên gọi: Lập trình viên (programmer), Nhân viên viết mã (coder).

**Kiểm thử viên Phần mềm (Software Tester)**

/codegym.vn/ - 12



Kiểm thử viên phần mềm là người tham gia vào công tác đảm bảo chất lượng phần mềm thông qua việc phát hiện các lỗi tiềm tàng và hỗ trợ nhóm phát triển trong việc xử lý các lỗi. Kiểm thử viên phần mềm thực hiện các thao tác phân tích nghiệp vụ, lập kế hoạch kiểm thử, viết các kịch bản kiểm thử, thực thi các ca kiểm thử và viết các báo cáo kiểm thử. Kiểm thử viên Phần mềm còn được biết đến với tên gọi: Kiểm thử viên, Nhân viên kiểm thử. 

**Chủ Sản phẩm (Product Owner)**

Chủ sản phẩm là người chịu trách nhiệm xác định và đảm bảo các chức năng của hệ thống. Chủ sản phẩm tìm hiểu và nghiên cứu các yêu cầu của người dùng cuối, đưa ra các giải pháp để đáp ứng được các yêu cầu, quản lý tiến độ và chất lượng của các chức năng trong suốt quá trình phát triển. Chủ sản phẩm là người có hiểu biết về thị trường, về các hệ thống phần mềm và sử dụng các công cụ quản lý sản phẩm và quản lý dự án. Chủ Sản phẩm còn được biết đến với các tên gọi: Quản lý sản phẩm.

**ScrumMaster (ScrumMaster)**

ScrumMaster là người chịu trách nhiệm trong việc đảm bảo các nhóm Scrum hoạt động tốt và chuyển giao được sản phẩm chất lượng cao. ScrumMaster nắm rõ khung làm việc Scrum, các kỹ thuật phát triển và là người làm việc thường xuyên với nhóm phát triển. ScrumMaster có thể là người am hiểu về công nghệ hoặc không.

7. Ngôn ngữ lập trình

Ngôn ngữ lập trình là công cụ để lập trình viên viết ra các chỉ dẫn cho máy tính thực thi. Có thể hiểu ngôn ngữ lập trình như là một cách để giao tiếp giữa lập trình viên và máy tính. Ngôn ngữ lập trình được sử dụng để tạo ra các phần mềm. Các ngôn ngữ lập trình quy định các chỉ dẫn, dựa vào đó lập trình viên sẽ sắp xếp chúng để tạo ra các chỉ dẫn có ý nghĩa. Một ví dụ để minh hoạ cho ý nghĩa của ngôn ngữ lập trình: Nếu chúng ta có các chỉ dẫn **ĐI THẲNG**, **RẼ TRÁI**, **RẼ PHẢI** thì chúng ta sẽ sắp xếp các chỉ dẫn đó để đưa được chú ong ở bản đồ sau đi đến được vị trí của bông hoa.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

/codegym.vn/ - 13



*Hình 1.5: Chỉ dẫn để chú ong đi đến bông hoa*

Chẳng hạn, các chỉ dẫn có thể là:

● RẼ PHẢI – ĐI THẲNG – ĐI THẲNG

● ĐI THẲNG – ĐI THẲNG – RẼ PHẢI

● ĐI THẲNG – RẼ PHẢI – ĐI THẲNG 

Ngày nay, có hàng trăm, thậm chí hàng nghìn ngôn ngữ lập trình khác nhau. Có những ngôn ngữ lập trình thông dụng như: Java, JavaScript, PHP, C#, Python, Ruby,… và cũng có nhiều ngôn ngữ lập trình rất chuyên biệt, chỉ sử dụng để xử lý những tình huống đặc thù nào đó. Mỗi ngôn ngữ lập trình đều có những đặc điểm riêng, phù hợp với những tình huống khác nhau, do đó việc lựa chọn ngôn ngữ lập trình phù hợp với một tình huống cụ thể cũng là một nhiệm vụ của lập trình viên.

Tuy có nhiều ngôn ngữ lập trình với các đặc trưng riêng, nhưng có một số chỉ dẫn cơ bản luôn hiện diện trong bất kỳ ngôn ngữ lập trình nào:

● Nhập dữ liệu: những chỉ dẫn giúp chương trình thu thập được dữ liệu từ bàn phím, thiết bị ngoại vi, tập tin, mạng, v.v.

● Xuất dữ liệu: những chỉ dẫn giúp chương trình có thể hiển thị dữ liệu lên màn hình, lưu dữ liệu vào tập tin, thay đổi trạng thái của thiết bị ngoại vi, gửi đi tín hiệu mạng, v.v.

● Tính toán: thực hiện những phép toán cơ bản như cộng hay nhân ● Thực thi có điều kiện: kiểm tra một số điều kiện và theo đó lựa chọn khối chỉ dẫn phù hợp để thực thi.

● Lặp lại: thực hiện một số chỉ dẫn một cách lặp đi lặp lại

Trong cuốn sách này, chúng ta sẽ sử dụng ngôn ngữ lập trình JavaScript để viết các phần mềm. Nên nhớ rằng, khi học lập trình, ngôn ngữ lập trình chỉ là một công cụ chứ không phải là toàn bộ những gì chúng ta cần học. Điều quan trọng nhất là chúng ta cần học tư duy lập trình, tư duy giải quyết vấn đề bằng các chỉ dẫn. Khi nắm được tư duy lập trình rồi thì chúng ta hoàn toàn có thể sử dụng một số ngôn ngữ khác nhau để tạo ra phần mềm. Điều này có nghĩa là, sau một thời gian lập trình thì việc học thêm các ngôn ngữ khác là một việc khá dễ dàng, hầu hết các lập trình viên ngày nay đều thuần thục một vài ngôn ngữ lập trình chứ không chỉ sử dụng một ngôn ngữ duy nhất.

8. Thuật toán

Khi chúng ta học lập trình thì không chỉ cần học một ngôn ngữ lập trình nào đó mà cần phải học tư duy giải quyết vấn đề. Khi đã có tư duy giải quyết vấn đề thì chúng ta có thể sử dụng các ngôn ngữ lập trình khác nhau để xây dựng các ứng dụng. Hay nói

/codegym.vn/ - 14



cách khác, ngôn ngữ lập trình chính là công cụ để hiện thực hoá tư duy giải quyết vấn đề cho một bài toán cụ thể.

Thuật toán, còn gọi là giải thuật, là một tập hợp hữu hạn các chỉ thị hay cách thức được định nghĩa rõ ràng cho việc hoàn tất một số sự việc từ một trạng thái ban đầu cho trước. Chúng ta có thể sử dụng các cách khác nhau để mô tả thuật toán, chẳng hạn như bằng lời nói, bằng các hình vẽ hoặc bằng các ký hiệu khác. 

**Ví dụ:**

Có hai bình A và B đựng hai loại chất lỏng khác nhau, chẳng hạn bình A đựng cồn, bình B đựng dấm. Yêu cầu tráo đổi (swap) chất lỏng đựng trong hai bình. **Thuật toán:**

● Yêu cầu phải có thêm một bình thứ ba gọi là bình C

● Bước 1: Đổ cồn từ bình A vào bình C

● Bước 2: Đổ dấm từ bình B vào bình A

● Bước 3: Đổ cồn từ bình C vào bình B

Trong lập trình, có 2 cách phổ biến để mô tả các thuật toán đó là Mã giả và Lưu đồ.

9. Mô tả thuật toán bằng mã giả

Mã giả (pseudo-code) là cách mô tả thuật toán bằng ngôn ngữ tự nhiên. Thông thường chúng ta sử dụng các từ tiếng Anh để mô tả thuật toán. Mã giả là mã không thực thi được và thông thường cũng không có các quy định chặt chẽ về cú pháp của mã giả. Chúng ta có thể tuỳ biến cách sử dụng từ, miễn sao đạt được mục đích là các bên liên quan có thể hiểu được thuật toán của chúng ta.

**Ví dụ**: Sử dụng mã giả để mô tả thuật toán giải phương trình bậc nhất với các cơ số a và b:

| 1. BEGIN  2. INPUT a, b  3. IF a = 0 THEN  4. IF b = 0 THEN  5. PRINT "Phương trình vô số nghiệm"  6. ELSE  7. PRINT "Phương trình vô nghiệm"  8. END IF  9. ELSE  10. PRINT "Phương trình có nghiệm x = -b / a"  11. END IF  12. END |
| --- |

**Giải thích:**

● *Dòng 1: BEGIN đánh dấu nơi bắt đầu thuật toán*

● *Dòng 2: INPUT dấu việc nhập dữ liệu cần thiết cho thuật toán*

● *Dòng 3: IF a = 0 đánh dấu việc thực thi theo điều kiện về tham số a = 0.*

● *Dòng 4: IF b = 0 đánh dấu việc thực thi theo điều kiện về tham số b = 0.*

● *Dòng 5: PRINT là chỉ thị để in ra nghiệm tương ứng với điều kiện a = 0 và b = 0.*

/codegym.vn/ - 15



● *Dòng 6: ELSE đánh dấu việc thực thi phương án thay thế khi điều kiện về tham số b = 0 không thỏa mãn.*

● *Dòng 7: PRINT là chỉ thị để in ra nghiệm tương ứng với điều kiện a = 0 và b ≠ 0.* ● *Dòng 8: END IF đánh dấu kết thúc quá trình thực thi theo điều kiện về tham sốba = 0.* ● *Dòng 9: ELSE chỉ thịthực thi phương án thay thế khi điều kiện về tham số a = 0 không thỏa mãn.*

● *Dòng 10: PRINT là chỉ thị để in ra nghiệm tương ứng với điều kiện a ≠ 0 và b ≠ 0 .* ● *Dòng 11: END IF đánh dấu kết thúc quá trình thực thi theo điều kiện về tham số a = 0.* ● *Dòng 12: END đánh dấu nơi kết thúc thuật toán.* 

Ưu điểm của việc sử dụng mã giả đó là gần với tự nhiên, ai cũng có thể sử dụng được. Nhược điểm của mã giả đó là nó không có các quy định chặt chẽ nên có thể dẫn đến tình huống là các bên không hiểu được nhau.

10. Mô tả thuật toán bằng lưu đồ

Lưu đồ là cách sử dụng các ký hiệu được quy định trước để mô tả thuật toán. Ưu điểm của lưu đồ là có các quy định chặt chẽ về từng ký hiệu, việc này giúp cho các bên thống nhất về cách sử dụng và dễ hiểu nhau hơn.

| **Biểu tượng** | **Mô tả** |
| --- | --- |
|  | Thể hiện trình tự thực hiện các thao tác |
|  | Điểm bắt đầu hoặc kết thúc của một tiến trình |
|  | Thực hiện các phép tính toán |
|  | Xét điều kiện và lựa chọn hướng đi tiếp dựa vào điều kiện đó |
|  | Nhập dữ liệu vào hoặc xuất thông tin ra |

*Bảng 1.2: Các biểu tượng thông dụng trong lưu đồ*

**Ví dụ**: Lưu đồ sau đây mô tả thuật toán để tìm nghiệm của phương trình bậc 2:

/codegym.vn/ - 16



**Giải thích:** 

● *Nút tròn rỗng mô tả nơi bắt đầu thuật toán*

● *Hình bình hành mô tả nơi nhập vào giá trị đầu vào*

● *Hình chữ nhật mô tả nơi thực hiện phép tính*

● *Hình bình hành mô tả nơi hiển thị kết quả*

● *Nút tròn với dấu X mô tả nơi kết thúc thuật toán.*

11. Một số cấu trúc thường gặp trong thuật toán

Thông thường, trình tự các bước thực hiện của thuật toán là tuyến tính từ trên xuống dưới. Nhưng trong nhiều tình huống, chúng ta cần thay đổi luồng thực thi đó thay đổi, chẳng hạn như ra các quyết định dựa trên một điều kiện nào đó, hoặc lặp đi lặp lại các hành động giống nhau. Trong những tình huống như vậy, chúng ta sẽ sử dụng các cấu trúc đặc trưng như cấu trúc điều kiện hoặc cấu trúc lặp.

11.1. Cấu trúc điều kiện

Cấu trúc điều kiện, còn được biết đến với tên gọi *cấu trúc lựa chọn*, là dạng cấu trúc được sử dụng trong các tình huống chúng ta cần ra các quyết định dựa trên một điều kiện cho trước.

Có một số dạng cấu trúc điều kiện cơ bản như sau:

▪ Cấu trúc 1: **Nếu** *< điều kiện>* (đúng) **thì** thực hiện *<công việc>* ▪ Cấu trúc 2: **Nếu** *< điều kiện>* (đúng) **thì** thực hiện *<công việc 1>*, **ngược lại** (điều kiện sai) **thì** thực hiện *<công việc 2>*

▪ Cấu trúc 3: Trường *hợp < i>* thì thực hiện *<công việc i>*

**Ví dụ 1:**

Bài toán kiểm tra xem một số có phải là số chẵn hay không, nếu là số chẵn thì hiển thị thông báo. Để xác định việc này, chúng ta căn cứ vào một đặc điểm là nếu một số là chẵn thì nó sẽ chia hết cho 2 (tức là dư 0).

Trong bài toán này chúng ta sử dụng dạng cấu trúc điều kiện **Nếu** *< điều kiện>* (đúng) **thì** thực hiện *<công việc>*. Trong đó, các bước thực hiện là:

/codegym.vn/ - 17



1. Nhập vào một số *num*

2. Tính *r* là phần dư của phép chia *num* cho 2

3. Kiểm tra xem *r* có bằng 0 hay không

4. Nếu *r* bằng 0 thì hiển thị thông báo “Number is even”

*Mã giả*:

| 1. BEGIN  2. INPUT num  3. r = num MOD 2  4. IF r=0  5. DISPLAY "Đây là số chẵn"  6. END IF  7. END |
| --- |

**Giải thích:** 

● *Dòng 1: BEGIN đánh dấu bắt đầu chương trình*

● *Dòng 2: Nhập giá trị num*

● *Dòng 3: Tính giá trị r theo công thức r bằng số dư của num chia cho 2* ● *Dòng 4: Kiểm tra xem thử r có bằng 0 hay không. Nếu r bằng 0 thì thực hiện dòng 5. Nếu r khác 0 thì bỏ qua dòng 5 và thực hiện dòng 6.*

● *Dòng 5: Hiển thị dòng chữ “Đây là số chẵn”*

● *Dòng 6: END IF đánh dấu kết thúc khối lệnh IF*

● *Dòng 7: END đánh dấu kết thúc chương trình*

*Lưu đồ:*

**

**Giải thích:**

Ở trong lưu đồ trên, hình quả trám được sử dụng để kiểm tra trường hợp r bằng 0, có hai trường hợp xảy ra được mô tả bằng hai hướng Yes và No. **Ví dụ 2:**

Bài toán kiểm tra xem một số là số chẵn hay là số lẻ, hiển thị thông báo tương ứng cho cả hai trường hợp.

Trong bài toán này chúng ta sử dụng dạng cấu trúc điều kiện **Nếu** *< điều kiện>* (đúng) **thì** thực hiện *<công việc 1>*, **ngược lại** (điều kiện sai) **thì** thực hiện *<công việc 2>*. Trong đó, các bước thực hiện là:

● Nhập vào một số *num*

/codegym.vn/ - 18



● Tính *r* là phần dư của phép chia *num* cho 2

● Kiểm tra xem *r* có bằng 0 hay không

● Nếu *r* bằng 0 thì hiển thị thông báo “Number is Even”

● Nếu ngược lại (tức là r khác 0) thì hiển thị thông báo “Number is Odd”. *Mã giả:*

| 1. BEGIN  2. INPUT num  3. r = num MOD 2  4. IF r = 0  5. DISPLAY "Đây là số chẵn"  6. ELSE  7. DISPLAY "Đây là số lẻ"  8. END IF  9. END |
| --- |

**Giải thích:** 

● *Dòng 1: BEGIN đánh dấu bắt đầu chương trình*

● *Dòng 2: Nhập giá trị num*

● *Dòng 3: Tính giá trị r theo công thức r bằng số dư của num chia cho 2* ● *Dòng 4: Kiểm tra xem thử r có bằng 0 hay không. Nếu r bằng 0 thì thực hiện dòng 5. Nếu r khác 0 thì thực hiện dòng 7.*

● *Dòng 5: Hiển thị dòng chữ “Đây là số chẵn”*

● *Dòng 6: ELSE đánh dấu đoạn mã sẽ thực thi trong trường hợp mà r khác 0* ● *Dòng 7: Hiển thị dòng chữ “Đây là số lẻ”*

● *Dòng 8: END IF đánh dấu kết thúc khối lệnh IF*

● *Dòng 9: END đánh dấu kết thúc chương trình*

*Lưu đồ:*

**

/codegym.vn/ - 19



11.2. Cấu trúc lặp

Cấu trúc lặp cho phép thực hiện lặp đi lặp lại các công việc nào đó dựa vào một điều kiện cho trước. Chúng ta thường sử dụng cấu trúc lặp để tự động hoá những công việc có tính chất giống nhau, giúp cho mã nguồn trở nên ngắn gọn hơn. Có hai dạng cấu trúc lặp cơ bản như sau:

● Lặp xác định trước số lần lặp: Là dạng lặp mà khi viết chương trình, người lập trình đã xác định được công việc sẽ lặp bao nhiêu lần. Chẳng hạn: hiển thị danh sách 100 khách hàng, tính tổng giá tiền của 10 sản phẩm, in bảng cửu chương (bảng tính nhân từ 1 đến 9) v.v. 

● Lặp không xác định số lần lặp: là loại lặp mà khi viết chương trình người lập trình chưa xác định được công việc sẽ lặp bao nhiêu lần. Số lần lặp sẽ được xác định tuỳ thuộc vào một số yếu tố cụ thể khi chương trình thực thi. Chẳng hạn: sao chép một file từ nơi này sang nơi khác (chúng ta không biết trước dung lượng của file), cho một nhân vật trong trò chơi chuyển động (chúng ta không biết trước khi nào thì nhân vật dừng lại), kim đồng hồ chuyển động v.v.

**Ví dụ 1:**

Hiển thị 1000 lần dòng chữ “Scooby”.

Trong bài toán này, chúng ta đã biết trước số lần lặp là 1000, do đó chúng ta sử dụng dạng lặp thứ nhất. Các bước thực hiện là:

1. Khai báo biến *counter* với giá trị ban đầu là 0

2. Kiểm tra điều kiện xem liệu *counter* có nhỏ hơn 1000 hay không 3. Nếu *counter* nhỏ hơn 1000 thì:

4. Hiển thị dòng chữ “Scooby”

5. Tăng giá trị của biến *counter* thêm 1 giá trị

6. Quay lại Bước 2

7. Nếu *counter* không nhỏ hơn 1000 (tức là bằng hoặc lớn hơn) thì kết thúc chương trình

*Mã giả:*

| 1. BEGIN  2. counter = 0  3. WHILE (counter < 1000)  4. DO  5. DISPLAY "Scooby"  6. counter = counter + 1  7. END DO  8. END |
| --- |

*Lưu đồ:*

/codegym.vn/ - 20



**Ví dụ 2:**

****

Cho phép người dùng lần lượt nhập vào các số tự nhiên, tính tổng tất cả các số mà người dùng đã nhập. Không hạn chế số lượng lần nhập. Khi người dùng nhập vào số 0 thì hiển thị kết quả và kết thúc chương trình. 

Trong bài toán này, chúng ta không biết trước số lần lặp. Các bước thực hiện là: 1. Khai báo một biến *sum* với giá trị ban đầu là 0

2. Cho phép người dùng nhập vào một giá trị cho biến *addition*

3. Cộng dồn giá trị của biến *addition* vào biến *sum*

4. Kiểm tra xem biến *addition* có bằng 0 hay không

5. Nếu biến *addition* khác 0 thì lặp lại Bước 2

6. Nếu biến *addition* bằng 0 thì hiển thị giá trị của biến *sum* và kết thúc *Mã giả:*

| 1. BEGIN  2. sum = 0  3. DO  4. INPUT addition  5. sum = sum + addition  6. WHILE (addition != 0)  7. DISPLAY sum  8. END |
| --- |

*Lưu đồ:*

/codegym.vn/ - 21





12. Ngôn ngữ lập trình JavaScript

JavaScript có khởi nguồn là một ngôn ngữ lập trình để chạy trên các trình duyệt, chủ yếu là để gia tăng tính tương tác và trải nghiệm đối với người dùng web. Về sau, JavaScript và hệ sinh thái của nó đã được cải tiến và phát triển để có thể chạy được trên các môi trường khác, chúng ta có thể sử dụng JavaScript để phát triển các ứng dụng phía back-end, mobile hay thậm chí là các ứng dụng desktop. 

Trong phạm vi của cuốn sách này, chúng ta sẽ sử dụng JavaScript trên môi trường của trình duyệt. Để thực thi mã JavaScript, chúng ta cần tạo ra một trang web và nhúng mã JavaScript vào trang web đó.

***Lưu ý:*** *Nếu trình duyệt bị vô hiệu hoá tính năng thực thi JavaScript thì chương trình JavaScript trên trang web sẽ không được thực thi.*

13. Chương trình đầu tiên

Ứng dụng JavaScript đầu tiên của chúng ta sẽ có một chức năng là hiển thị một thông báo nhỏ lên trên trang web với nội dung *“Hello. JavaScript is running.”*. Đoạn mã của file HTML:

| 1. <html>  2. <head>  3. <title>My First JavaScript Application</title>  4. </head>  5. <body>  6. <script>  7. alert("Hello. JavaScript is running.");  8. </script>  9. </body>  10. </html |
| --- |

/codegym.vn/ - 22



Bạn có thể sao chép đoạn mã trên vào một text editor bất kỳ (chẳng hạn notepad) và lưu lại thành một file có phần mở rộng là *.html*, chẳng hạn *hello.html*. 

Mở file này bằng một trình duyệt web sẽ cho thấy một dialog được hiển thị như dưới: 

Điều gì đã diễn ra?

● Trong file HTML, chúng ta sử dụng thẻ <script> để nhúng các đoạn mã JavaScript vào, từ dòng 6 đến dòng 8

● Khi mở trang trang web bằng trình duyệt, trình duyệt sẽ phiên dịch đoạn mã JavaScript ở giữa dòng 6 và dòng 8 và thực thi

● Hàm *alert()* ở dòng 7 là một hàm được sử dụng rất phổ biến trong JavaScript để hiển thị các thông báo.

***Lưu ý:*** *Cấu trúc của file HTML luôn luôn bao gồm các thẻ <html>, <head>, <title>, <body> như ví dụ trên. Chúng ta sẽ chưa đi tìm hiểu sâu về HTML ở giai đoạn này.*

14. Cài đặt công cụ lập trình

Để bắt đầu tạo ra các phần mềm, chúng ta cần cài đặt các công cụ cần thiết. Trong đó, một trình soạn thảo mã nguồn (code editor) là điều cần thiết.

Có rất nhiều các trình soạn thảo mã nguồn trên thị trường hiện nay, hầu hết chúng đều hỗ trợ các tính năng cần thiết để giúp cho việc viết mã trở nên dễ dàng hơn. Các tính năng chính của chúng thường bao gồm:

● Hiển thị mã nguồn ở các màu sắc khác nhau, giúp dễ phân biệt ● Gợi ý khi viết, giúp viết mã nhanh hơn

● Hỗ trợ biên dịch hoặc phiên dịch mã nguồn

● Hỗ trợ thực thi mã nguồn để quan sát kết quả

/codegym.vn/ - 23



● Hỗ trợ debug (dò lỗi) để quan sát tiến trình thực thi của mã

● Và còn nhiều tính năng khác: Tích hợp trình quản lý phiên bản (SVN, Git...), tự động định dạng mã nguồn, tái cấu trúc mã nguồn…

**IDE – Integrated Development Environment**

IDE là thuật ngữ để nói đến các trình soạn thảo mã nguồn trong đó có tích hợp nhiều chức năng và môi trường để hỗ trợ cho việc lập trình. IDE là viết tắt của Integrated Development Environment (nghĩa là Môi trường Phát triển Tích hợp). Một IDE có thể chỉ hỗ trợ một công nghệ hoặc hỗ trợ nhiều công nghệ khác nhau. Một số IDE nổi tiếng như: WebStorm, PhpStorm, IntelliJ, Netbeans, Eclipse, Visual Studio, Visual Studio Code, Atom, Sublime Text, v.v. 

Đối với các tính năng cơ bản thì hầu hết các IDE đều đáp ứng được, chúng chỉ khác nhau ở một số tiện ích riêng. Trong khuôn khổ của cuốn sách này, chúng ta có thể dùng WebStorm hoặc Visual Studio Code để viết mã.

15. Bài thực hành

Các bài thực hành sau đây sẽ hướng dẫn cách giải các bài toán đơn giản và ghi giải pháp ra dưới dạng mã giả hoặc lưu đồ. Chúng ta hãy lấy giấy và bút để thực hành các bài này.

Bài 1: Thuật toán chuyển đổi nhiệt độ

Mô tả thuật toán nhập một giá trị là độ 0C (Celsius) và chuyển nó sang độ 0F (Fahrenheit). Công thức chuyển đổi là:

� = 9 � �

5 + 32

**Hướng dẫn:**

Trong bài toán này, chúng ta sẽ lần lượt thực hiện các bước sau: ● Nhập vào giá trị độ C

● Tính giá trị độ F dựa theo công thức đã cho

● Hiển thị giá trị độ F

**Mã giả:**

| 1. BEGIN  2. INPUT c  3. f = (9 \* c) / 5 + 32  4. DISPLAY f  5. END |
| --- |

**Lưu đồ:**

/codegym.vn/ - 24





Bài 2: Thuật toán tính điểm trung bình

Mô tả thuật toán tính điểm trung bình của học sinh. Nhập vào điểm số 3 môn Toán, Lý và Hoá, sau đó hiển thị điểm số trung bình. 

**Hướng dẫn:**

Trong bài này, chúng ta sẽ thực hiện các bước sau:

● Nhập vào điểm số của 3 môn Toán, Lí và Hoá

● Tính điểm trung bình

● Hiển thị điểm trung bình

**Mã giả:**

| 1. BEGIN  2. INPUT math, physics, chemical  3. medium = (math + physics + chemical) / 3  4. DISPLAY medium  5. END |
| --- |

Bạn hãy thực hành vẽ lưu đồ cho thuật toán này.

16. Bài tập

Bài 1: Thuật toán chuyển đổi tiền tệ

Hãy viết mã giả và vẽ lưu đồ để mô tả thuật toán chuyển đổi từ Đô la Mỹ sang Việt Nam Đồng. Nhập vào giá trị Đô la Mỹ, sau đó hiển thị giá trị Việt Nam Đồng tương ứng. Tỉ giá chuyển đổi là 1/23000.

Công thức chuyển đổi:

1 Đô �� �ỹ = 23000 ��ệ� ��� Đồ��

Bài 2: Thuật toán tìm giá trị lớn nhất trong 3 số

Hãy viết mã giả và vẽ lưu đồ để mô tả thuật toán tìm giá trị lớn nhất trong 3 số. Nhập vào 3 số, sau đó thực hiện các phép so sánh lần lượt từng cặp số để tìm ra giá trị lớn nhất trong 3 số đó.

*Gợi ý: Chúng ta sẽ cần sử dụng cấu trúc điều kiện.*

/codegym.vn/ - 25



Bài 3: Thuật toán tìm giá trị lớn nhất trong dãy số

Hãy viết mã giả và vẽ lưu đồ để mô tả thuật toán tìm giá trị lớn nhất trong một loạt các số được nhập vào. Nhập vào số n là số lượng các số, sau đó nhập lần lượt n số và tìm ra giá trị lớn nhất trong đó.

*Gợi ý: Chúng ta sẽ cần sử dụng kết hợp cấu trúc lặp và cấu trúc điều kiện.* Bài 4: Thuật toán xếp hạng sinh viên 

Hãy viết mã giả và vẽ lưu đồ để mô tả thuật toán xếp hạng sinh viên. Nhập vào điểm thi của sinh viên, hiển thị phân loại sinh viên theo các khoảng:

| **Điểm thi** | **Xếp hạng** |
| --- | --- |
| Điểm >= 75 | Loại A |
| 60 <= Điểm < 75 | Loại B |
| 45 <= Điểm < 60 | Loại C |
| 35 <= Điểm < 45 | Loại D |
| Điểm < 35 | Loại E |

*Gợi ý: Chúng ta sẽ cần sử dụng nhiều cấu trúc điều kiện nối tiếp nhau.*

17. Bài kiểm tra

**Câu 1:** Điền từ còn thiếu vào định nghĩa sau:

Lập trình là quá trình tạo ra tập các \_\_\_ để \_\_\_ cho máy tính hoàn thành một \_\_\_ nào đó.

a) chỉ dẫn (instruction), ra lệnh, công việc (task)

b) hướng dẫn, yêu cầu, chương trình

c) chỉ dẫn (instruction), hướng dẫn, công việc (task)

d) chỉ dẫn (instruction), ra lệnh, sự kiện

**Câu 2:** Đâu là các hoạt động trong lập trình?

a) Viết code

b) Phân tích

c) Tìm hiểu yêu cầu

d) Thiết kế

**Câu 3:** Phát biểu nào sau đây là SAI về ưu nhược điểm của việc sử dụng mã giả hoặc lưu đồ để mô tả thuật toán?

a) Sử dụng lưu đồ giúp cho các bên thống nhất về cách sử dụng các ký hiệu và dễ hiểu nhau hơn

/codegym.vn/ - 26



b) Nhược điểm của lưu đồ là có các quy định quá chặt chẽ về từng ký hiệu không phải ai cũng nhớ để sử dụng được

c) Nhược điểm của mã giả là không có các quy định chặt chẽ dẫn đến tình huống các bên không hiểu được nhau

d) Mã giả gần với tự nhiên, ai cũng có thể sử dụng được.

**Câu 4:** Kết luận nào sau đây ĐÚNG khi nói về việc học lập trình 

a) Ngôn ngữ lập trình chính là công cụ để hiện thực hoá tư duy giải quyết vấn đề cho một bài toán cụ thể

b) Học lập trình thì không chỉ cần học một ngôn ngữ lập trình nào đó mà cần phải học tư duy giải quyết vấn đề

c) Học lập trình là học một loại ngôn ngữ lập trình để ra lệnh cho máy tính thực hiện 1 yêu cầu nào đó

d) Học một ngôn ngữ lập trình chúng ta chỉ cần quan tâm đến cú pháp của ngôn ngữ đó

**Câu 5:** Điền các từ còn thiếu vào phát biểu sau:

Mã giả là cách để mô tả thuật toán bằng \_\_\_ Thông thường chúng ta sử dụng \_\_\_ để mô tả thuật toán. Mã giả là mã \_\_\_ và thông thường cũng không có các quy định chặt chẽ về cú pháp của mã giả.

a) ngôn ngữ tự nhiên, các từ tiếng Anh, không thực thi được

b) ngôn ngữ tự nhiên, code, không thực thi được

c) ngôn ngữ tự nhiên, code, có thể thực thi được

d) ngôn ngữ máy, các từ của nước bản địa, không thể thực thi được **Câu 6:** Lưu đồ giúp chúng ta xem xét lại và gỡ rối chương trình một cách dễ dàng? a) Đúng

b) Sai

**Câu 7:** Phát biểu nào sau đây đúng về khái niệm "Thuật toán"?

a) Thuật toán là tập hợp các bước để đưa ra đáp án cho một bài toán b) Thuật toán bao gồm các chỉ thị để giải quyết một vấn đề

c) Thuật toán bao gồm một số bước để giải quyết một vấn đề

d) Thuật toán là cách giải một bài toán

**Câu 8:** Phát biểu nào sai về việc sử dụng các ký hiệu trong flow chart? a) Ký hiệu hình chữ nhật đánh dấu các thao tác nhập xuất dữ liệu b) Ký hiệu hình eclipse đánh dấu điểm bắt đầu hoặc kết thúc của thuật toán c) Ký hiệu hình thoi đánh dấu các bước cần rẽ nhánh

d) Ký hiệu hình bình hành đánh dấu các bước tính toán

**Câu 9:** Điền các từ còn thiếu vào phát biểu sau:

Một \_\_\_ cơ bản là việc thực thi tuần tự những câu lệnh đến khi một điều kiện cụ thể nào đó là đúng hay sai

a) Rẽ nhánh

/codegym.vn/ - 27



b) Khai báo biến

c) Vòng lặp

d) Thực thi hàm

**Câu 10:** Điền các từ còn thiếu vào phát biểu sau:

\_\_\_lựa chọn một công việc để thực hiện căn cứ vào một điều kiện nào đó. a) Biến 

b) Vòng lặp

c) Hàm

*d)* Cấu trúc điều kiện

***Đáp án:*** *Câu 1: a; Câu 2: a,b,c,d; Câu 3: c; Câu 4: b; Câu 5: a; Câu 6: a; Câu 7: b; Câu 8: a, d; Câu 9: c; Câu 10: d*

18. Tổng kết

● Một hệ thống máy tính bao gồm phần cứng, phần mềm và người dùng ● Phần cứng là những thành phần có thể sờ nắm được tạo nên máy tính ● Phần mềm được ví như linh hồn của máy tính, không thể sờ nắm được

● Các công việc cơ bản để tạo ra một phần mềm bao gồm: Thu thập yêu cầu, phân tích, thiết kế, viết mã, kiểm thử, phát hành, bảo trì

● Ngôn ngữ lập trình là công cụ để giao tiếp với máy tính bằng cách viết ra các chỉ dẫn

● Có nhiều vai trò khác nhau tham gia vào trong quá trình sản xuất phần mềm, chẳng hạn như: lập trình viên, kiểm thử viên, quản lý dự án, chuyên gia phân tích nghiệp vụ, ScrumMaster, Product Owner…

● Giải thuật (hay còn gọi là thuật toán) là các bước để xử lý một vấn đề ● Có nhiều cách để mô tả thuật toán, 2 cách thông dụng đó là dùng mã giả và lưu đồ

● Mã giả là cách sử dụng các từ ngữ tự nhiên quen thuộc với con người ● Máy tính không hiểu mã giả, hay nói cách khác, mã giả không thể thực thi được ● Lưu đồ là cách sử dụng các ký hiệu được quy định trước để mô tả thuật toán ● Luồng thực thi của mã nguồn là tuyến tính từ trên xuống dưới ● Có thể thay đổi luồng thực thi của chương trình thông qua các cấu trúc như cấu trúc điều kiện và vòng lặp

● Cấu trúc điều kiện cho phép lựa chọn thực thi một đoạn mã dựa vào một điều kiện

● Cấu trúc lặp cho phép thực hiện nhiều lần một đoạn mã nào đó ● JavaScript là một ngôn ngữ được sử dụng phổ biến trên các giao diện web ● Ngày nay JavaScript còn có thể sử dụng để tạo các ứng dụng back-end, mobile…

/codegym.vn/ - 28



● IDE (Integrated Development Environment) là thuật ngữ được sử dụng để chỉ đến các công cụ soạn thảo mã nguồn được tích hợp thêm các môi trường và tính năng khác để hỗ trợ cho việc lập trình

/codegym.vn/ - 29



Chương 2 - Biến, kiểu dữ liệu và toán tử Thực hiện các phép tính toán

1. Mục tiêu 

● Mô tả được khái niệm biến

● Khai báo và sử dụng được biến

● Mô tả được các kiểu dữ liệu thông dụng

● Sử dụng được đúng kiểu dữ liệu phù hợp

● Sử dụng được các toán tử số học

● Sử dụng được các toán tử logic

● Sử dụng được các toán tử so sánh

2. Giới thiệu

Ta đã biết từ chương trước, mọi chương trình máy tính đều thực hiện 4 công việc: nhận dữ liệu, lưu trữ, xử lý, xuất thông tin. Chẳng hạn, trong một ứng dụng bán hàng trực tuyến, người dùng sẽ nhập vào số lượng sản phẩm muốn mua, hệ thống sẽ dựa vào đó để tính được tổng số tiền mà người mua phải trả và hiển thị thông tin đó cho người dùng.

Chương này sẽ giúp chúng ta lựa chọn đúng kiểu dữ liệu phù hợp với từng tình huống. Chẳng hạn, để lưu trữ tên của người dùng thì chọn kiểu dữ liệu gì? Để lưu tuổi của người dùng thì chọn kiểu dữ liệu gì? Để lưu giới tính của người dùng thì chọn kiểu dữ liệu gì?...

Đối với các giá trị của từng kiểu dữ liệu, chúng ta có thể thực hiện được các phép toán khác nhau. Chẳng hạn, chúng ta có thể thực hiện các phép tính Cộng, Trừ, Nhân, Chia giữa các giá trị số. Chúng ta có thể thực hiện phép so sánh giữa các giá trị có kiểu ký tự,v.v.

Hoàn thành chương này, chúng ta có thể tạo được các ứng dụng trong đó thực hiện được các phép tính toán căn cứ vào dữ liệu mà mình nhập vào.

3. Biến

Khái niệm biến

Trong địa lý, nếu chúng ta được yêu cầu chỉ vị trí của vùng nằm ở tọa độ 16oB và 112oĐ ở trên quả địa cầu, phần lớn chúng ta sẽ mất khá nhiều thời gian để suy nghĩ, tìm kiếm và sau đó mới xác định được đúng tọa độ của vùng đó.

*Hình 2.1*

/codegym.vn/ - 30



Tuy nhiên, nếu chúng ta được yêu cầu chỉ vị trí của Quần 

đảo Hoàng Sa, phần lớn chúng ta đều dễ dàng xác định

được dễ dàng và nhanh chóng.

Tương tự như vậy, thật khó để có thể nhớ được toạ độ

của các thành phố thủ đô của các nước, nhưng chúng ta

lại khá dễ dàng để xác định được vị trí của thành phố 

thông qua tên của chúng, chẳng hạn như Paris, Berlin,

Bangkok, v.v.

Điều gì làm nên sự khác biệt trong việc sử dụng các toạ

độ và sử dụng các tên gọi? Rõ ràng, ai cũng biết là việc

sử dụng các tên gọi sẽ giúp dễ nhớ hơn, còn toạ độ

thường được dùng trong các tính toán khoa học. Việc sử

dụng tên gọi đại diện cho một vùng nào đó trên bản đồ sẽ giúp cho chúng ta dễ dàng hơn khi làm việc với nó.

Trong thế giới máy tính cũng vậy. Dữ liệu được lưu trong bộ nhớ. Bộ nhớ bao gồm nhiều ô nhớ. Mỗi ô nhớ có một địa chỉ riêng của nó để xác định ô nhớ đó. Cũng giống như toạ độ để xác định một vùng đất vậy. Tệ hơn, địa chỉ của các ô nhớ còn nhiều, dài dòng và khó nhớ hơn gấp nhiều lần so với kinh độ và vĩ độ trên trái đất. Có hàng triệu, thậm chí là hàng tỷ ô nhớ trong mỗi máy tính.

Giả sử, chúng ta cần thực hiện phép trừ 3 số. Như vậy chúng ta bắt đầu nhập vào số thứ nhất, lưu nó vào trong một ô nhớ ngẫu nhiên trong bộ nhớ, chúng ta phải ghi nhớ địa chỉ của ô này. Sau đó, chúng ta nhập số thứ 2 và lại lưu vào trong bộ nhớ, lại phải ghi nhớ địa chỉ ô nhớ này. Khi thực hiện phép cộng, chúng ta phải sử dụng các địa chỉ ô nhớ này để lấy các giá trị ra, rồi lại phải ghi kết quả của phép tính vào một ô nhớ khác, lại phải nhớ địa chỉ của ô nhớ mới.

*Hình 2.2: Các ô nhớ*

Ở hình trên, các giá trị 5, 4 và 9 đều được lưu trữ ở trong bộ nhớ máy tính ở các vùng khác nhau được gọi là các ô nhớ. Mỗi ô nhớ đều được xác định bởi một địa chỉ khá dài và khó nhớ.

/codegym.vn/ - 31



Cứ như vậy, điều gì sẽ xảy ra nếu chúng ta thực hiện một phép tính với sự tham gia của hàng chục giá trị khác nhau? Khả năng của lập trình viên khó có thể nhớ được địa chỉ của từng ấy ô nhớ. Đây là lúc chúng ta cần đặt tên cho các ô nhớ. *Hình 2.3: Sử dụng tên để đại diện cho các ô nhớ* 

Bây giờ, thay vì phải ghi nhớ địa chỉ của từng ô nhớ, chúng ta sẽ sử dụng các tên để đại diện cho ô nhớ mà mình mong muốn.

Ví dụ, khi nhập vào số thứ nhất, chúng ta đặt tên cho ô nhớ đó là *firstNumber*. Khi nhập vào số thứ hai, chúng ta đặt tên cho ô nhớ đó là *secondNumber*. Khi lưu trữ giá trị của phép tính, chúng ta đặt tên cho ô nhớ mới là *result*.

Chúng ta dễ dàng nhận thấy, cách làm này tốt hơn rất nhiều so với trước đây. Với cách làm này, chúng ta vừa tìm hiểu một khái niệm rất quan trọng trong lập trình, được gọi là *biến*.

**Định nghĩa biến:** *Biến là một tên gọi đại diện cho một vùng nhớ để lưu trữ dữ liệu trong máy tính.*

Khai báo biến

Khai báo biến là thao tác đặt tên cho một ô nhớ trong bộ nhớ. Sau khi khai báo biến thì chúng ta có thể sử dụng biến đó để thao tác với ô nhớ mà nó đại diện. Trong JavaScript, chúng ta khai báo một biến bằng cách sử dụng từ khoá *let*. Ví dụ sau đây khai báo lần lượt 3 biến là *radius*, *area* và *diameter*:

| 1. let radius;  2. let area;  3. let diameter; |
| --- |

***Lưu ý:*** *Tại các phiên bản JavaScript cũ hơn so với phiên bản phổ biến hiện nay, chúng ta khai báo biến bằng từ khoá var.*

Gán giá trị cho biến

Để gán giá trị cho một biến, chúng ta sử dụng toán tử gán (=).

**Ví dụ:**

/codegym.vn/ - 32



| 1. radius = 2.5;  2. area = 19.6;  3. diameter = 15.7; |
| --- |

Nói rằng “=” là một toán tử là bởi vì ngoài việc làm cho biến thay đổi giá trị, bản thân phép gán cũng trả về giá trị được gán giống như là kết quả của một phép tính. Vậy nên mặc dù câu lệnh sau trông rối và ít được dùng trong thực tế thì vẫn là câu lệnh đúng: 

| 1. radius1 = radius2 = 2.5; |
| --- |

Phép gán *radius2* làm cho *radius2* có giá trị là 2.5, đồng thời “phép tính” này cho kết quả là 2.5, kết quả này được gán vào cho biến *radius1*. Như vậy, kết quả là cả biến *radius1* và biến *radius2* đều có giá trị là 2.5.

Khai báo và khởi tạo giá trị cho biến

Hành động tạo giá trị cho biến lần đầu tiên (nhớ rằng sau đó giá trị của biến có thể được thay đổi bằng cách thực hiện lại phép gán) được gọi là khởi tạo giá trị cho biến. Chúng ta có thể khai báo một biến đồng thời với việc khởi tạo:

| 1. let radius = 2.5;  2. let area = 19.6;  3. let diameter = 15.7; |
| --- |

**Lưu ý**: một biến có thể được cấp lại giá trị, không giới hạn bao nhiêu lần. Trái lại, tại cùng một phạm vi mã (scope), bạn không thể khai báo một biến nhiều lần, chúng ta sẽ học về phạm vi mã sau, nhưng tại thời điểm này bạn có thể hiểu rằng chỉ dẫn sau sẽ gây lỗi bởi vì biến radius được khai báo hai lần:

| 1. let radius = 2.5;  2. let radius = 3;  // Uncaught SyntaxError: Identifier 'radius' has already been declared |
| --- |

Quy tắc đặt tên cho biến

Vì một số lý do, trong hầu hết các ngôn ngữ lập trình, chúng ta chỉ có thể sử dụng các chữ cái tiếng Anh và ký tự số để đặt tên cho biến. Tên biến sẽ không có các khoảng trắng (bao gồm cả dấu tab lẫn dấu xuống dòng), ký tự đặc biệt, hay chữ cái có dấu. Chữ cái đầu tiên của biến không được phép là ký tự số.

Riêng với ngôn ngữ JavaScript, ngoài hai quy tắc trên, chúng ta được phép sử dụng hai ký tự là *$* và *\_* (dấu gạch ngang dưới) như là chữ cái.

Chúng ta có thể kể ra một số tên biến hợp lệ và không hợp lệ như sau:

| 1. let money; // Hợp lệ  2. let 1000dolar; // Không hợp lệ  3. let my money; // Không hợp lệ  4. let \_radius; // Hợp lệ  5. let $username; // Hợp lệ |
| --- |

/codegym.vn/ - 33



***Lưu ý****: Các ràng buộc được nhắc tới ở đây là Quy tắc (rules), có nghĩa là nếu bạn không tuân thủ, trình phiên dịch và thực thi JavaScript sẽ báo lỗi. Bạn phân biệt Quy tắc với các Quy ước (convention) - là những ràng buộc mà nếu tuân thủ thì sẽ rất tốt chứ không phải là bắt buộc. Chúng ta có thể vi phạm các Quy ước mà không ảnh hưởng trực tiếp tới kết quả thực thi của chương trình.*

4. Kiểu dữ liệu 

Trong máy tính, một giá trị sẽ được thể hiện bằng *dữ liệu* đi kèm với *kiểu giữ liệu*. Cùng một vùng nhớ nhưng nếu được nhìn nhận dưới kiểu dữ liệu khác nhau thì sẽ thể hiện cho những giá trị khác nhau, chẳng hạn, trong một số trường hợp, ký tự 'A' nếu được nhìn nhận dưới kiểu dữ liệu số thì sẽ thể hiện cho số 65. Kiểu dữ liệu sẽ ảnh hưởng tới các toán tử có thể sử dụng lên dữ liệu (chẳng hạn, kiểu dữ liệu ký tự thì không thể áp dụng phép chia được) cũng như cách hành xử của mỗi toán tử. Chẳng hạn hai giá trị số là 1 và 2 khi được áp dụng phép tính cộng sẽ cho kết quả là 3, nhưng hai ký tự *“1”* và *“2”* cộng lại sẽ cho kết quả *“12”*.

Định kiểu động

JavaScript là một ngôn ngữ *định kiểu động*. Chúng ta không cần phải khai báo kiểu của các biến, thay vào đó, trình thực thi sẽ suy đoán kiểu của biến thông qua giá trị của nó. Nhờ đó mà đoạn mã sau cho chúng ta kết quả là *3* chứ không phải là *“12”:*

| 1. let foo = 1;  2. let bar = 2;  3. console.log(foo + bar); // 3 |
| --- |

Hai nhóm Kiểu dữ liệu

Trình thực thi JavaScript nhìn dữ liệu thành hai dạng: dữ liệu dạng nguyên thủy và dữ liệu dạng đối tượng. Trong khuôn khổ cuốn sách này chúng ta sẽ chỉ tập trung thao tác với kiểu dữ liệu nguyên thủy.

Kiểu dữ liệu nguyên thủy là kiểu dữ liệu được xây dựng sẵn trong ngôn ngữ JavaScript. Theo đó, hầu hết các toán tử được xây dựng để hỗ trợ xoay quanh các kiểu dữ liệu này.

Các kiểu dữ liệu nguyên thuỷ

Có các loại dữ liệu nguyên thủy khác nhau, trong đó kiểu Symbol mới được bổ sung gần đây. Kiểu Symbol có mục đích sử dụng khá đặc thù và chúng ta sẽ không bàn tới nó trong cuốn sách này. Các kiểu dữ liệu nguyên thủy còn lại là:

● boolean

● undefined

● number

● string

/codegym.vn/ - 34



Kiểu number

Kiểu *number* thể hiện các giá trị số, bao gồm cả số nguyên và số thực. JavaScript chỉ có một kiểu số duy nhất, có giá trị từ -(253 -1) đến 253 -1. Ngoài việc có thể chứa giá trị dấu phẩy động, kiểu số có thêm ba giá trị đại diện: *Infinity* (dương vô cùng), *-Infinity* (âm vô cùng), và *NaN* (Not-a-Number – không phải là một số). Kiểu dữ liệu số được dùng trong các trường hợp như để lưu tuổi của học sinh, giá của sản phẩm, nhiệt độ trong ngày, v.v. 

**Ví dụ:**

| 1. let age = 15;  2. let price = 20.05; |
| --- |

Trong ví dụ này, biến *age* và biến *price* có kiểu dữ liệu là số được dùng lần lượt để lưu tuổi của một người và giá của một sản phẩm.

Có một số nguyên duy nhất có hai đại diện: *0* được đại diện bởi *-0* và *+0*. (0 là một cách viết ngắn gọn của *+0*). Mặc dù điều này hầu như không có tác động nào tới chương trình (biểu thức (+0 === -0) cho chúng ta kết quả là *True*), tuy nhiên chúng ta có thể nhận thấy ảnh hưởng này khi chia một số cho *0*:

| 1. let x = 42 / 0; // Infinity;  2. let y = 42 / -0; // -Infinity; |
| --- |

Kiểu string

Kiểu string được dùng để biểu diễn dữ liệu dạng chuỗi ký tự. Mỗi ký tự có một *chỉ mục* (số thứ tự) của mình ở trong chuỗi. Trong ngôn ngữ JavaScript, các chỉ mục luôn được đánh số từ 0. Ký tự đầu tiên trong chuỗi sẽ chỉ số là *0*, tiếp theo là *1*, *2*, *3*...*Độ dài* của chuỗi là số ký tự của chuỗi đó. Kiểu dữ liệu string được sử dụng trong những

trường hợp như tên của học sinh, địa chỉ nhà ở, tiêu đề của một bài viết, v.v. **Ví dụ**: Chuỗi “HELLO” có độ dài là 5, bởi vì nó có chứa 5 ký tự. Chỉ mục của chữ H là 0, chỉ mục của chữ 0 là 4.

Để khai báo chuỗi thì chúng ta có thể sử dụng dấu nháy đơn hoặc dấu nháy kép. Chẳng hạn:

| 1. let brand = "CodeGym Việt Nam"; //Sử dung dấu nháy kép  2. let address = 'Hà Nội'; //Sử dụng dấu nháy đơn |
| --- |

***Lưu ý****: Trong một chương trình, nên thống nhất việc sử dụng dấu nháy đơn hoặc nháy kép. Tránh trường hợp sử dụng lộn xộn cả hai cách khiến cho mã nguồn trở nên xấu và khó đọc.*

Chuỗi trong JavaScript là bất biến (immutable). Chúng ta không thể thay đổi chuỗi sau khi đã khởi tạo giá trị. Tuy nhiên chúng ta có thể tính toán một chuỗi mới dựa trên giá trị cũ và gán ngược lại vào biến chuỗi.

/codegym.vn/ - 35



Kiểu boolean

Kiểu boolean đại diện có hai giá trị logic là *true* và *false*. Kiểu dữ liệu boolean được sử dụng trong các trường hợp cần phân biệt 2 trạng thái là *true* hoặc *false*. Chẳng hạn như là trạng thái bật và tắt của bóng đèn, kết quả của một biểu thức điều kiện trong đó có sử dụng các phép so sánh, v.v.

Trong JavaScript, tất cả các giá trị sau đều được coi là *false*: 

● Giá trị *false* của kiểu *boolean*

● Giá trị *+0* và *-0* của kiểu *number*

● Chuỗi rỗng (giá trị *""*)

● Giá trị *undefined* của kiểu *undefined*

● Gía trị *null* của kiểu *object*

Tất cả các giá trị khác sẽ được coi là *true* khi thực hiện phép chuyển đổi kiểu.

| 1. let f1 = Boolean(0) // false  2. let f2 = Boolean("") // false  3. let f3 = Boolean(undefined) // false  4. let f4 = Boolean(-1) // true  5. let f5 = Boolean('hello') // true |
| --- |

Kiểu undefined

Undefined mang ý nghĩa là *"chưa xác định".* Một biến chưa được gán giá trị thì có giá trị là *undefined*, đồng thời kiểu dữ liệu của nó cũng là *undefined*.

| 1. let x  2. console.log(x)  3. console.log(typeof x) // undefined  4. x = 3  5. console.log(x) // 3  6. console.log(typeof x) // number |
| --- |

5. Phép toán toán học

Toán tử (Operator)

Trong toán học, chúng ta đã quen thuộc với các toán tử (hay còn gọi là ký tự phép toán) như +, −, ×, : ... Chúng được sử dụng trong các biểu thức toán học, thực hiện các tính toán dựa trên giá trị của các toán hạng và trả về một kết quả cuối cùng. Trong các ngôn ngữ lập trình cũng vậy, có nhiều toán tử khác nhau được sử dụng cho nhiều mục đích khác nhau. Có những loại phổ biến như:

● Toán tử toán học được sử dụng trong các phép tính toán học

● Toán tử gán được sử dụng để gán giá trị cho các biến

● Toán tử cộng chuỗi được sử dụng để nối hai chuỗi

● Toán tử so sánh được sử dụng để so sánh các giá trị với nhau ● Toán tử logic được sử dụng để thay đổi giá trị thuộc kiểu dữ liệu boolean

/codegym.vn/ - 36



● Toán tử typeof được sử dụng để xác định kiểu dữ liệu của một giá trị Toán tử toán học (arithmetic)

Toán tử toán học được sử dụng trong các biểu thức toán học. Toán tử toán học làm việc với các giá trị số. Thông thường, trong một biểu thức thì toán tử toán học có 2 toán hạng, nhưng cũng có các trường hợp là một toán hạng. Chẳng hạn, trong biểu thức dưới đây thì toán tử là +, hai toán hạng là *100* và *50*: 



Danh sách các toán tử toán học được tóm tắt trong bảng sau:

| **Toán tử Mô tả** | |
| --- | --- |
| + | Cộng |
| - | Trừ |
| \* | Nhân |
| / | Chia |
| % | Chia lấy phần dư (modulus) |
| ++ | Tăng 1 giá trị |
| -- | Giảm 1 giá trị |

**Ví dụ 1:** Sử dụng với các giá trị số

| 1. let x = 100 + 50; |
| --- |

**Ví dụ 2:** Sử dụng với các biến

| 1. let a = 5;  2. let b = 10;  3. let c = a + b; |
| --- |

**Ví dụ 3:** Sử dụng với các biểu thức

| 1. let a = 2;  2. let x = (100 + 50) \* a; |
| --- |

Toán tử Cộng

Toán tử cộng *(+)* được sử dụng để tính tổng của hai số:

| 1. let x = 5;  2. let y = 2;  3. let z = x + y; //z = 7 |
| --- |

/codegym.vn/ - 37



Toán tử Trừ

Toán tử trừ *(-)* được sử dụng để tính hiệu của hai số:

| 1. let x = 5;  2. let y = 2;  3. let z = x - y; //z = 3 |
| --- |

Toán tử Nhân 

Toán tử nhân *(\*)* được sử dụng để tính tích của hai số:

| 1. let x = 5;  2. let y = 2;  3. let z = x \* y; // z = 10 |
| --- |

Toán tử Chia

Toán tử chia *(/)* được sử dụng để tính hiệu của hai số:

| 1. let x = 5;  2. let y = 2;  3. let z = x / y; // z = 2.5 |
| --- |

Toán tử Chia lấy số dư

Toán tử chia lấy số dư *(%)* được sử dụng để tính số dư của một phép chia:

| 1. let x = 5;  2. let y = 2;  3. let z = x % y; // z = 1 |
| --- |

Toán tử Tăng giá trị

Phép tăng giá trị là một phép toán cần duy nhất một toán hạng. Tác dụng của nó là tăng giá trị của một biến *number* được dùng làm toán hạng tăng lên một đơn vị.

| 1. let x = 5;  2. x++; // x = 6 |
| --- |

Phép tăng giá trị cũng tạo ra kết quả của bản thân phép tính. Giá trị đó phụ thuộc vào vị trí đặt toán tử. Nếu toán tử được đặt trước thì kết quả của phép toán sẽ là giá trị của toán hạng sau khi tăng, còn nếu toán tử đặt sau thì kết quả sẽ là giá trị của toán hạng trước khi tăng. Ví dụ:

| 1. let x = 5;  2. let y = x++; // x = 6, y = 5  3. x = 5  4. let z = ++x // x = 6, y = 6 |
| --- |

Trong cả hai trường hợp trên thì giá trị của biến x đều được tăng lên là 6. Nhưng kết quả của bản thân phép toán thì khác nhau cho nên giá trị của biến y là khác nhau.

/codegym.vn/ - 38



Toán tử giảm giá trị:

Toán tử giảm giá trị *(--)* được sử dụng để giảm giá trị của một biến *number* được dùng làm toán hạng giảm đi một đơn vị.

| 1. let x = 5;  2. x--; // x = 4 |
| --- |

***Lưu ý:*** *Tương tự như toán tử Tăng giá trị, toán tử Giảm giá trị cũng có kết quả trả về và kết quả này cũng bị ảnh hưởng bởi vị trí đặt toán tử.* 

Toán tử gán (assignment)

Toán tử gán được sử dụng để gán giá trị cho một biến, toán tử gán có thể sử dụng với tất cả các kiểu dữ liệu khác nhau.

| 1. let x = 10; |
| --- |

Trong đoạn mã trên, chúng ta gán giá trị *10* cho biến *x*. Sau khi biểu thức này được thực thi thì giá trị của biến *x* sẽ là *10*.

Toán tử gán cơ bản nhất là toán tử bằng, trong đó giá trị phía bên phải dấu bằng được gán cho biến ở phía bên trái của nó.

Ngoài ra, còn có thêm các toán tử gán phái sinh, kết hợp toán tử bằng với các toán tử khác. Chẳng hạn, toán tử cộng gộp là kết hợp giữa toán tử cộng và toán tử bằng. Biểu thức x += y tương đương với biểu thức x = x + y.

Bảng sau đây liệt kê các toán tử phái sinh từ toán tử gán.

| **Toán tử Ví dụ Tương đương với** | | |
| --- | --- | --- |
| = | x = y | x = y |
| += | x += y | x = x + y |
| -= | x -= y | x = x - y |
| \*= | x \*= y | x = x \* y |
| /= | x /= y | x = x / y |
| %= | x %= y | x = x % y |

Ví dụ 1:

| 1. let x = 5; |
| --- |

Ví dụ 2:

| 1. let x = 10;  2. x += 5; // x = 15 (Tương đương với thực hiện gán x = x + 5) |
| --- |

Ví dụ 3:

1. let x = 10;

/codegym.vn/ - 39



2. x -= 5; // x = 5 (Tương đương với thực hiện gán x = x - 5)

Ví dụ 4:

| 1. let x = 10;  2. x \*= 5; // x = 50 (Tương đương với thực hiện gán x = x \* 5) |
| --- |

Ví dụ 5: 

| 1. let x = 10;  2. x /= 5; // x = 2 (Tương đương với thực hiện gán x = x / 5) |
| --- |

Ví dụ 6:

| 1. let x = 11;  2. x %= 5; // x = 1 (Tương đương với thực hiện gán x = x % 5) |
| --- |

Toán tử cộng chuỗi (string concatenate)

Toán tử cộng chuỗi được sử dụng để nối các chuỗi. Cũng có thể sử dụng để nối chuỗi và các số.

Ví dụ 1:

| 1. let txt1 = "John";  2. let txt2 = "Doe";  3. let txt3 = txt1 + " " + txt2; // txt3 = "John Doe" |
| --- |

Trong ví dụ trên, giá trị của biến txt3 là một chuỗi *“John Doe”*.

Ví dụ 2:

| 1. let x = 5 + 5;// x = 10  2. let y = "5" + 5;// y = "55"  3. let z = "Hello" + 5;// z = "Hello5" |
| --- |

Trong ví dụ trên, giá trị của biến *x* là số *10* (cộng hai số). Giá trị của biến y là một chuỗi *“55”* (cộng một chuỗi với một số). Giá trị của biến *z* là một chuỗi *“Hello5”* (cộng một chuỗi với một số).

6. Phép toán logic

Toán tử logic được dùng trong các biểu thức logic (*true/false*). Toán tử “VÀ” được ký hiệu bởi hai dấu và *(&&)*. Toán tử “HOẶC” được ký hiệu bởi hai dấu gạch đứng *(||)*. Toán tử “PHỦ ĐỊNH” được ký hiệu bởi một dấu chấm than *(!)*.

| **Toán tử Mô tả** | |
| --- | --- |
| && | Và (AND) |
| || | Hoặc (OR) |
| ! | Phủ định (NOT) |

/codegym.vn/ - 40



Toán tử &&

Toán tử && trả về giá trị *true* nếu **cả hai** toán hạng đều có giá trị *true*. Chỉ cần một toán hạng có giá trị *false* thì kết quả sẽ là *false*.

Chúng ta có thể tổng hợp các trường hợp giá trị của *a*, *b* và kết quả như trong bảng.

| **Giá trị biến a Giá trị biến b Kết quả (a && b)** | | |
| --- | --- | --- |
| true | true | true |
| true | false | false |
| false | false | false |
| false | true | false |

Toán tử || 

Toán tử || trả về giá trị *true* nếu **một trong hai** toán hạng có giá trị *true*. Chỉ duy nhất trường hợp nếu cả hai toán hạng có giá trị *false* thì kết quả sẽ là *false*. Chúng ta có thể tổng hợp các trường hợp giá trị của a, b và kết quả như trong bảng.

| **Giá trị biến a Giá trị biến b Kết quả (a || b)** | | |
| --- | --- | --- |
| true | true | true |
| true | false | true |
| false | false | false |
| false | true | true |

Toán tử !

Toán tử phủ định đảo ngược giá trị boolean, chẳng hạn giá trị *true* thì chuyển thành *false* và giá trị *false* thì chuyển thành *true*.

| **Giá trị biến a Kết quả !a** | |
| --- | --- |
| true | false |
| false | true |

Độ ưu tiên của các toán tử

Trong toán học các phép tính trong một biểu thức có độ ưu tiên nhất định để xác định phép tính nào được tính toán trước. Trong lập trình cũng vậy, nếu trong một biểu thức có sự tham gia của nhiều toán tử thì thứ tự thực hiện được dựa trên độ ưu tiên được quy định sẵn.

Bảng bên dưới liệt kê các thứ tự ưu tiên ứng với từng toán tử:

/codegym.vn/ - 41



| **Toán tử Độ ưu tiên** | |
| --- | --- |
| Hậu tố (postfix) | expr++ expr-- |
| Một ngôi | ++expr --expr +expr -expr ~ ! |
| Nhân | \* / % |
| Cộng | + - |
| Dịch | << >> >>> |
| Quan hệ | < > <= >= instanceof |
| Bằng | == != |
| Bitwise AND | & |
| Bitwise exclusive OR | ^ |
| Bitwise inclusive OR | | |
| Và | && |
| Hoặc | || |
| Ba ngôi | ? : |
| Gán | = += -= \*= /= %= &= ^= |= <<= >>= >>>= |

Nếu muốn thay đổi độ ưu tiên của các toán tử, chúng ta có thể sử dụng phép đóng mở ngoặc tương tự như trong toán học. Đối với các phép toán có cùng độ ưu tiên, chẳng hạn như phép cộng và phép trừ thì thứ tự thực hiện là từ trái sang phải. **Ví dụ:** 

Khởi tạo hai biến x và y có giá trị lần lượt là 5 và 6:

| 1. let x = 5;  2. let y = 10; |
| --- |

Khởi tạo biến z có giá trị là kết quả của một biểu thức, các bước tính toán của biểu thức được thực hiện minh hoạ ở các dòng phía dưới.

| let z = (++x \* y) < 5 \* 10 && 6 > 3;  (6 \* y) < 5 \* 10 && 6 > 3;  60 < 50 && 6 > 3;  false && true;  false |
| --- |

Trong biểu thức trên, trước tiên toán tử *++* được ưu tiên thực thi *(++x = 6)*. Tiếp theo các toán tử nhân được thực thi (*6 \* y = 60* và *5 \* 10 = 50*). Tiếp theo sẽ thực thi các toán tử so sánh ( *60 < 50 = false* và *6 > 3 = true*). Cuối cùng, toán tử logic *&&* được thực thi. Kết quả biến *z* sẽ có giá trị là false.

/codegym.vn/ - 42



7. Phép toán so sánh

Các toán tử so sánh được dùng để đánh giá mức độ tương quan giữa các giá trị. Kết quả của phép toán so sánh sẽ là *true* hoặc *false*. Chẳng hạn, chúng ta có các toán tử rất quen thuộc là toán tử *Bằng*, toán tử *Lớn hơn*, toán tử *Nhỏ hơn*, toán tử *Lớn hơn hoặc Bằng*, toán tử *Nhỏ hơn hoặc Bằng*. 

***Lưu ý:*** *Có hai loại toán tử so sánh Bằng, một loại chỉ có 2 dấu bằng (==) dùng để so sánh giá trị mà không phân biệt kiểu dữ liệu của chúng, một loại có 3 dấu bằng (===), được sử dụng để so sánh trong trường hợp các giá trị có cùng kiểu. Trong ngôn ngữ Javascript, chúng ta thường sử dụng toán tử có 3 dấu bằng hơn.*

*Cũng có hai toán tử so sánh Khác, một loại bao gồm một dấu chấm than và một dấu bằng (!=), dùng để so sánh khác nhau giữa hai giá trị mà không phân biệt kiểu dữ liệu. Một loại bao gồm một dấu chấm than và hai dấu bằng (!==), được sử dụng để so sánh khác nhau giữa hai giá trị cùng kiểu dữ liệu.*

Danh sách các toán tử so sánh được liệt kê trong bảng sau:

| **Toán tử Mô tả** | |
| --- | --- |
| == | Bằng về giá trị (không phân biệt kiểu dữ liệu) |
| === | Bằng về giá trị, đồng thời cùng kiểu dữ liệu |
| != | Khác về giá trị (không phân biệt kiểu dữ liệu) |
| !== | Không bằng về giá trị, hoặc không cùng kiểu dữ liệu |
| > | Lớn hơn |
| < | Nhỏ hơn |
| >= | Lớn hơn hoặc bằng |
| <= | Nhỏ hơn hoặc bằng |

**Ví dụ 1:**

| 1. let x = 5;  2. let y = x == 5; // y = true  3. let z = x == "5"; // z = true |
| --- |

Trong ví dụ trên, chúng ta sử dụng toán tử so sánh Bằng với 2 dấu bằng, biến y có giá trị là true. Biến z cũng có giá trị là true, bởi vì mặc dù có 2 kiểu dữ liệu khác nhau nhưng giá trị của chúng đều là 5.

**Ví dụ 2:**

| 1. let x = 5;  2. let y = x === 5; //y = true  3. let z = x === "5"; //z = false  4. |
| --- |

/codegym.vn/ - 43



Trong ví dụ trên, chúng ta sử dụng toán tử so sánh *===*, giá trị của biến *y* là *true*. Giá trị của biến *z* là *false*, bởi vì các giá trị bằng nhau nhưng lại có kiểu dữ liệu khác nhau (*x* có kiểu dữ liệu là *number* còn *“5”* có kiểu dữ liệu là *string*).

**Ví dụ 3:**

| 1. let x = 5;  2. let y = x != 8; // y = true  3. let z = x != "5"; // z = false |
| --- |

Trong ví dụ trên, chúng ta sử dụng toán tử so sánh *Khác* với một dấu chấm than và một dấu bằng , giá trị của biến *y* là *true* và giá trị của biến *z* là *false*. **Ví dụ 4:** 

| 1. let x = 5;  2. let y = x !== 5; // y = false  3. let z = x !== "5"; // z = true |
| --- |

Trong ví dụ trên, chúng ta sử dụng toán tử so sánh *Khác* bao gồm một dấu chấm than và hai dấu bằng, giá trị của biến *y* là *false*. Giá trị của biến *z* là *true*, bởi vì mặc dù hai giá trị giống nhau nhưng lại khác kiểu dữ liệu.

8. Đọc dữ liệu từ bên ngoài

Trong phạm vi cuốn sách này, chúng ta thực thi chương trình thông qua trình phiên dịch của trình duyệt. Chương trình của chúng ta, theo đó, sẽ thu thập dữ liệu từ môi trường bên ngoài chúng thông qua trình duyệt. Có hai phương án mà chúng ta có thể sử dụng ngay được như sau:

● Sử dụng câu hàm *prompt()* để tạo dialog cho người dùng nhập liệu. ● Đọc giá trị của một phần tử *input* trong tài liệu HTML.

Nhận dữ liệu thông qua hộp thoại prompt.

Với phương án này, chúng ta sử dụng lời gọi hàm *prompt()* có sẵn trong môi trường thực thi của trình duyệt. Bạn sẽ nghiên cứu kỹ hơn về hàm nói chung ở các chương sau. Tại thời điểm này, hãy chỉ tập trung học cách sử dụng hàm prompt:

| 1. let radius = prompt("Vui long nhap ban kinh hinh tron:"); |
| --- |

Hàm *prompt()* ở trên sẽ yêu cầu trình duyệt mở một hộp thoại, với lời dẫn được mô tả trong tham số của nó, cùng với một ô nhập liệu và nút chấp nhận. Sau khi người dùng nhấn nút chấp nhận, giá trị họ nhập vào ô nhập liệu sẽ được gán cho biến *radius*:

| 1. console.log(radius); // 2.5 |
| --- |

/codegym.vn/ - 44



***Lưu ý:*** *Giá trị nhận được từ prompt luôn có kiểu dữ liệu là string, cho dù người dùng có nhập số hợp lệ đi chăng nữa. Trong trường hợp đó, để chuyển đổi giá trị này thành giá trị số, chúng ta có thể sử dụng hàm Number() như dưới đây:*

| 1. typeof radius; // "string"  2. radius = Number(radius);  3. console.log(radius); // 2.5  4. typeof radius; // "number" |
| --- |

Như vậy, chúng ta có thể viết ngắn gọn quá trình nhập một số vào như sau: 

| 1. let radius = Number(prompt("Vui long nhap ban kinh hinh tron:")); |
| --- |

Đọc giá trị của ô input từ trong tài liệu HTML

Trong tài liệu HTML có thể có những thẻ *input*. Các thẻ này được trình duyệt hiển thị dưới dạng những ô nhập liệu mà người dùng có thể nhập giá trị vào. Trình duyệt cũng cung cấp sẵn một số hàm JavaScript để lập trình viên có thể đọc được giá trị của các ô input đó.

Chẳng hạn, chúng ta có một tài liệu HTML với thẻ input như sau:

| 1. <h3>Circles Calculator</h3>  2. <label>  3. Ban kinh: <input type="text" id ="radius">  4. <button onclick="showArea()">Area</button>  5. </label>  6. <script type="text/JavaScript">  7. function showArea() { }  8. </script> |
| --- |

Chúng ta có thể sử dụng hàm JavaScript như sau để đọc giá trị nhập vào của phần tử input có id là radius ở trên:

| 1. let radius = document.getElementById("radius").value; |
| --- |

**Giải thích**

Ở dòng lệnh trên, chúng ta đã sử dụng hàm *getElementById()* có sẵn của đối tượng *document* để lấy về giá trị của thẻ input có tên là *radius*.

Hãy thử hoàn thành hàm *showArea()* với câu lệnh trên để giúp tính ra cho người dùng bán kính của hình tròn. Sau đó thực thi chương trình để xem kết quả.

| 1. function showArea() {  2. let radius = document.getElementById("radius").value;  3. console.log("Doc duoc gia tri ban kinh la " + radius);  4. let area = radius \* radius \* 3.14;  5. console.log("Tinh duoc gia tri dien tich la " + area + " voi PI = " + 3.14);  6. alert("Dien tich hinh tron co ban kinh " + radius + " la " + area);  7. console.log("Nguoi dung da nhan duoc ket qua!");  8. } |
| --- |

***Lưu ý:*** *cũng như prompt(), kết quả nhận được từ thuộc tính value ở đây cũng luôn có kiểu dữ liệu string.*

/codegym.vn/ - 45



9. Hiển thị dữ liệu

JavaScript có thể "hiển thị" dữ liệu theo nhiều cách khác nhau:

● Viết thành một phần tử HTML, sử dụng thuộc tính *innerHTML*. ● Viết vào đầu ra HTML sử dụng hàm *document.write()*.

● Viết vào một hộp cảnh báo sử dụng hàm *alert()*.

● Viết vào bảng điều khiển trình duyệt sử dụng hàm *console.log()*. Sử dụng innerHTML 

Để truy cập một phần tử HTML, JavaScript có thể sử dụng phương thức *document.getElementById(id)*. Trong đó, *id* là thuộc tính giúp xác định thành phần HTML mà chúng ta muốn truy cập. Thuộc tính *innerHTML* chính là nội dung bên trong của thành phần HTML đó:

**Ví dụ:**

| 1. <p id="demo"></p>  2. <script>  3. document.getElementById("demo").innerHTML = 7;  4. </script> |
| --- |

**Kết quả:** Nội dung của thẻ *<p>* với id là *demo* đã được thay đổi thành *7*. Sử dụng document.write()

Hàm *document.write()* thường được dùng để hiển thị dữ liệu bằng cách viết trực tiếp ra tài liệu HTML.

**Ví dụ:**

| 1. <h2>My First Web Page</h2>  2. <p>My first paragraph.</p>  3. <button type="button" onclick="document.write(5 + 6)">Try it</button> |
| --- |

Sử dụng hàm alert()

Hàm *alert()* có nhiệm vụ in một thông báo popup, nó có một tham số truyền vào là nội dung ta muốn thông báo với người dùng.

**Ví dụ:**

| 1. <h2>My First Web Page</h2>  2. <p>My first paragraph.</p>  3. <button type="button" onclick="alert(5 + 6)">Try it</button> |
| --- |

Sử dụng console.log()

Hàm *console.log()* được sử dụng thường xuyên trong việc tìm lỗi (debug). Hàm này có nhiệm vụ hiển thị ra giá trị của tất cả các loại dữ liệu như *number*, *integer*, *array*, *object...* ra cửa sổ console của trình duyệt.

Tùy thuộc vào các trình duyệt mà cách thức mở panel *console* của trình duyệt sẽ khác nhau, nhưng thường thì chúng ta sử dụng phím F12.

**Ví dụ:**

/codegym.vn/ - 46



| 1. <script>  2. console.log("xin chao")  3. </script> |
| --- |

Kết quả hiển thị sẽ nằm ở panel console như dưới đây.



10. Mã sạch

Quy tắc #1: Quy ước

Cú pháp, với mỗi cộng đồng hay mỗi nhóm phát triển sẽ có các quy định cách đặt tên biến khác nhau, nhưng đều sẽ tuân thủ theo những cách đặt bên dưới đây: ● Cú pháp lạc đà (camelCase): Ký tự đầu tiên của từ đầu tiên viết thường, những ký tự đầu tiên của những từ tiếp theo được viết hoa.

● Cú pháp con rắn (snake\_case): Với cú pháp con rắn, tất cả các chữ cái đều viết thường, và các từ cách nhau bởi dấu gạch dưới.

**Ví dụ:**

Các tên biến theo kiểu *camelCase*:

| 1. let userName;  2. let thisIsTheNameThatFollowTheCamelCase; |
| --- |

Các tên biến theo kiểu *snake\_*case:

| 1. let user\_name;  2. let this\_is\_the\_name\_that\_follow\_the\_snake\_case; |
| --- |

Quy tắc #2: Sử dụng đúng loại từ

Tên biến nên là danh từ, cụm danh từ, hoặc tính từ.

**Ví dụ:**

| 1. let name; // Nên  2. let color; // Nên  3. let visible; //Nên  4. let productPrice; // Nên  5. let move; // Không nên  6. let show; // Không nên |
| --- |

Quy tắc #3: Tên có ý nghĩa

Tên biến cần có ý nghĩa. Khi nhìn vào tên biến nó sẽ trả lời cho câu hỏi vì sao nó tồn tại, nó sẽ làm gì.

/codegym.vn/ - 47



| 1. let abc; // Không nên  2. let uName; // Không nên |
| --- |

Quy tắc #4: Mỗi biến khai báo trên một dòng

Khi khai báo biên nên tách mỗi dòng khai báo một biến.

Ví dụ: 

| 1. let i = 3, j = 5, k = 10; // Không nên  2. let i = 3; // Nên  3. let j = 5; // Nên  4. let k = 10; // Nên |
| --- |

11. Các lỗi thường gặp

Các lỗi thường gặp #1: Toán tử “===”

Toán tử “===” so sánh đồng thời giá trị và kiểu dữ liệu. Ví dụ:

| 1. let x = 5;  2. let y = x === 5;// y = true  3. let z = x === "5";// z = false |
| --- |

Trong ví dụ trên, biến *y* có giá trị là *true* vì biến *x* và *5* có cùng giá trị và cùng kiểu dữ liệu. Biến *z* có giá trị là *false* vì biến *x* và *“5”* có cùng giá trị nhưng khác nhau về kiểu dữ liệu.

Các lỗi thường gặp #2: So sánh số thực

Trong lập trình, không phải bao giờ các phép tính toán trên số thực cũng cho kết quả chính xác tuyệt đối. Việc so sánh bằng (==) giữa hai biểu thức số thực là không đáng tin cậy.

Ví dụ, đoạn mã sau sẽ hiển thị kết quả là *false* thay vì *true* như chúng ta thường nghĩ

| 1. let x = 1.0 - 0.1 - 0.1 - 0.1 - 0.1 - 0.1;  2. console.log(x == 0.5); // false  3. console.log(x); // 0.5000000000000001 |
| --- |

Trong ví dụ trên, giá trị của biến *x* không phải là *0.5* mà là *0.5000000000000001* **Giải pháp cho tình huống so sánh số thực:**

Để xử lý tình huống so sách giữa các số thực, chúng ta sử dụng cách so sánh gần đúng, dựa trên nguyên tắc: hiệu số của hai giá trị là một số rất nhỏ (epsilon), chẳng hạn như 10-7 hoặc 10-14.

**Ví dụ:**

| 1. const EPSILON = 1E-14;  2. let x = 1.0 - 0.1 - 0.1 - 0.1 - 0.1 - 0.1;  3. console.log(Math.abs(x - 0.5) < EPSILON ); // true |
| --- |

Hàm *Math.abs()* trên đây cho chúng ta giá trị tuyệt đối của biểu thức *x - 0.5*. Kết quả nhỏ hơn sai số cho phép và kết quả so sánh là *true*.

/codegym.vn/ - 48



13. Bài thực hành

**Bài 1:** Khai báo các biến thuộc các kiểu khác nhau, gán giá trị cho chúng và in ra tài liệu HTML.

● Biến *i* kiểu *number*, có giá trị là *10*

● Biến *f* kiểu *number*, có giá trị là *20.5*

● Biến *b* kiểu *boolean*, có giá trị là *true* 

● Biến *s* kiểu *string*, có giá trị là "Hà Nội".

Sử dụng hàm *document.write()* để viết kết quả ra màn hình.

Mã tham khảo:

| 1. let i = 10;  2. let f = 20.5;  3. let b = true;  4. let s = 'Hà Nội';  5.  6. document.write('i = ' + i);  7. document.write('<br/>');  8. document.write('f = ' + f);  9. document.write('<br/>');  10. document.write('b = ' + b);  11. document.write('<br/>');  12. document.write('s = ' + s); |
| --- |

**Bài 2:** Viết một đoạn mã JavaScript, khai báo biến *width* chứa giá trị độ rộng của hình chữ nhật, biến *height* chứa giá trị chiều cao của hình chữ nhật. In ra màn hình diện tích của hình chữ nhật đó.

Mã tham khảo:

| 1. let width = 20;  2. let height = 10;  3. let area = width \* height;  4. document.write('Area = ' + area); |
| --- |

**Bài 3:** Dùng hàm *prompt()* để nhập 2 số từ bàn phím, lưu vào *2* biến là *a* và b*.* Kiểm tra xem *a* có phải là bội số của *b* hay không. Hiển thị kết quả ra màn hình bằng cách sử dụng hàm *alert()*.

Mã tham khảo:

| 1. let a = Number(prompt("Enter a:"));  2. let b = Number(prompt("Enter b:"));  3. let result = a % b;  4. if(result === 0){  5. alert("a is the multiple of b");  6. } else {  7. alert("a is not a multiple of b");  8. } |
| --- |

/codegym.vn/ - 49



14. Bài tập

Bài 1: Ứng dụng chuyển đổi tiền tệ

Hãy viết một ứng dụng cho phép chuyển đổi giữa VNĐ và các loại tiền tệ khác. Giao diện của ứng dụng tương tự như sau, chương trình sẽ hiển thị hộp thoại để thông báo kết quả. 



Bài 2: Ứng dụng máy tính đơn giản

Hãy viết một ứng dụng máy tính đơn giản cho phép thực hiện các phép tính Cộng, Trừ, Nhân, Chia, Bình phương, Lập phương…

Giao diện của ứng dụng có thể như sau:



● Chương trình có ba ô nhập liệu và các nút chức năng. Hai ô nhập đầu tiên để người dùng nhập các toán hạng, ô thứ 3 để chương trình hiển thị kết quả tính. ● Các nút chức năng chỉ cần dùng đến một giá trị (ví dụ tính bình phương, lập phương, phân số) thì chỉ dùng đến giá trị của ô nhập liệu thứ nhất. ● Nhấn vào các nút chức năng, chương trình sẽ hiển thị kết quả phép tính tương ứng vào ô nhập liệu thứ 3.

● Khi nhấn vào nút chức năng “Use now”, giá trị vừa tính ra sẽ được đưa vào ô nhập liệu đầu tiên, giá trị ở ô nhập liệu thứ 2 và thứ 3 sẽ được xóa trắng.

15. Bài kiểm tra

**Câu 1:** Trong JavaScript, nếu x = 83 và y = 9 thì sau khi thực hiện x %= y, x sẽ có giá trị là bao nhiêu?

a) 0

b) 1

c) 2

d) 9

**Câu 2:** JavaScript có phân biệt hoa\thường hay không?

/codegym.vn/ - 50



a) Có

b) Không

**Câu 3:** Các định danh nào được đặt đúng?

a) 1\_Name

b) \_Name

c) name\_1 

**Câu 4.** Thẻ HTML nào cho phép đưa các mã JavaScript vào trang web? a) <jsscript>

b) <script>

c) <JavaScript>

d) <scripting>

**Câu 5:** Trong JavaScript, đâu là mã để viết chuỗi "Xin chào" vào trang web? a) document.write("Xin chào");

b) response.write("Xin chào");

c) windows.write("Xin chào");

d) writeln("Xin chào");

**Câu 6:** Làm thế nào để viết thông điệp "Xin chào" lên hộp thoại của trình duyệt? a) msgBox("Xin chào")

b) alert("Xin chào")

c) alertBox="Xin chào"

d) alertBox("Xin chào")

**Câu 7:** Đâu là các cú pháp đúng để viết chú thích trong JavaScript? a) Đây là chú thích

b) <!-- Đây là chú thích -->

c) /\* Đây là chú thích \*/

d) // Đây là chú thích

**Câu 8:** Đâu là dòng lệnh đúng dùng để khai báo một biến trong JavaScript? a) char[] name;

b) let name;

c) String name;

d) let $name;

**Câu 9:** Biểu thức *"1"+2+4* cho kết quả là gì?

a) "16"

b) Undefined

c) "124"

d) 7

**Câu 10:** Biểu thức *2+5+"8"* cho kết quả là gì?

a) "258"

b) 15

/codegym.vn/ - 51



c) "78"

d) undefined

***Đáp án:*** *Câu 1: c; Câu 2: b; Câu 3: b, c; Câu 4: b; Câu 5: a; Câu 6: b; Câu 7: c, d; Câu 8: b, d; Câu 9: c; Câu 10: c.*

16. Tổng kết 

● Biến là một khái niệm được dùng để đại diện cho một giá trị được lưu trữ trong bộ nhớ của máy tính

● Khi khai báo biến thì cần xác định tên của biến

● Tên của biến cần phải tuân thủ các quy định của từng ngôn ngữ lập trình ● Sau khi khai báo biến thì có thể gán giá trị cho biến

● Dữ liệu trong máy tính được chia thành nhiều kiểu khác nhau ● Một số kiểu dữ liệu thông dụng bao gồm: number, string, boolean, object… ● Có thể thực hiện các phép toán khác nhau trên các giá trị hoặc các biến ● Có các loại toán tử thông dụng như: Toán học, Logic, So sánh ● Các toán tử có độ ưu tiên nhất định, có thể thay đổi độ ưu tiên bằng cách sử dụng dấu ngoặc ()

● Có một vài cách khác nhau để đọc dữ liệu từ bên ngoài, chẳng hạn sử dụng hàm *prompt()* hoặc đọc dữ liệu từ thẻ *input*

● Có một vài cách khác nhau để hiển thị dữ liệu ra bên ngoài, chẳng hạn như *alert()*, *document.write()* hoặc *console.log()*

/codegym.vn/ - 52



Chương 3 - Câu lệnh điều kiện

Điều khiển luồng thực thi của chương trình dựa vào một điều kiện nhất định

1. Mục tiêu 

● Mô tả được các tình huống cần sử dụng câu lệnh điều kiện

● Trình bày được cú pháp của câu lệnh if-else

● Sử dụng được câu lệnh if để lựa chọn thực thi một khối lệnh

● Sử dụng được câu lệnh if-else để lựa chọn thực thi một trong hai khối lệnh ● Sử dụng được câu lệnh if lồng nhau để lựa chọn dựa trên nhiều điều kiện bao trùm lên nhau

● Sử dụng được câu lệnh if bậc thang để lựa chọn dựa trên nhiều điều kiện liên tiếp nhau

● Trình bày được cú pháp của câu lệnh switch-case

● Sử dụng được câu lệnh switch-case để lựa chọn dựa trên so sánh Bằng trong trường hợp có nhiều nhánh

● Lựa chọn được cấu trúc điều kiện phù hợp để sử dụng trong từng tình huống

2. Giới thiệu

Trong cuộc sống, chúng ta luôn phải đưa ra các quyết định khác nhau dựa trên việc đánh giá các tình huống hiện tại. Chẳng hạn, nếu đi đường gặp đèn đỏ thì chúng ta phải dừng chờ, gặp đèn xanh thì chúng ta đi tiếp. Nếu trời nắng thì chúng ta sẽ tìm các món ăn mát và bù nước, nếu trời lạnh thì chúng ta tìm các món ăn nóng ấm và giữ nhiệt, v.v.

Trong lập trình cũng vậy, chúng ta thường phải thay đổi luồng thực thi của một chương trình dựa trên các điều kiện hiện tại, chẳng hạn khi chúng ta đăng nhập vào một trang web, nếu mật khẩu nhập vào là đúng thì chúng ta sẽ được chuyển đến trang quản trị, còn nếu sai thì chúng ta sẽ phải đăng nhập lại.

Để làm được điều này, chúng ta sử dụng các cấu trúc trong lập trình được gọi là cấu trúc điều kiện, hay còn có một tên gọi khác là cấu trúc lựa chọn.

Hoàn thành chương này, chúng ta sẽ có thể viết được các ứng dụng phần mềm trong đó sẽ thực hiện một số hành động khác nhau dựa trên việc đánh giá các điều kiện.

/codegym.vn/ - 53



3. Cấu trúc điều kiện

Các câu lệnh điều khiển

Một chương trình phần mềm thực thi các câu lệnh theo trật tự từ trên xuống dưới. Khi đó chúng ta có thể thay đổi luồng thực thi của một chương trình bằng cách sử dụng các câu lệnh điều khiển luồng (control flow statement). 

Các câu lệnh điều khiển của JavaScript:

● Câu lệnh điều kiện (conditional statement)

● Câu lệnh lặp (Loop statement)

● Câu lệnh nhảy (jump statement)

Câu lệnh điều kiện

Trong khi viết mã lệnh, chúng ta muốn thực hiện các hành động khác nhau đối với các quyết định (điều kiện) khác nhau. Khi đó chúng ta có thể sử dụng các lệnh điều kiện trong mã của mình để làm điều này.

Câu lệnh điều kiện còn được gọi là câu lệnh ra quyết định (decision making). Câu lệnh điều kiện cho phép thay đổi luồng thực thi của chương trình. Việc lựa chọn thực thi một khối lệnh dựa trên việc đánh giá một điều kiện cho trước.

JavaScript hỗ trợ các câu lệnh điều kiện:

● Câu lệnh *if-else*: Được sử dụng trong trường hợp muốn đánh giá một biểu thức và rẽ sang 1 hoặc 2 nhánh khác nhau tuỳ thuộc vào giá trị của biểu thức đó. ● Câu lệnh *switch-case*: Được sử dụng trong trường hợp muốn đánh giá một

biểu thức và rẽ sang 1 hoặc nhiều nhánh khác nhau tuỳ thuộc vào giá trị của biểu thức đó.

4. Cấu trúc điều kiện if-else

Câu lệnh if

Câu lệnh if được sử dụng để xem xét việc thực thi một thao tác nào đó dựa trên điều kiện hiện tại.

**Cú pháp:**

| 1. if (biểu\_thức\_điều\_kiện) {  2. khối mã được thực thi nếu điều kiện là đúng  3. } |
| --- |

Câu lệnh *if* bao gồm một biểu thức điều kiện và một khối lệnh ở trong phần thân của nó. Tại thời điểm thực thi, nếu biểu thức điều kiện trả về kết quả *true* thì khối lệnh trong phần thân sẽ được thực thi, còn nếu biểu thức điều kiện trả về *false* thì khối lệnh trong phần thân sẽ được bỏ qua.

**Ví dụ:**

1. if (score > 9) {

2. classification = "Xuất sắc";

/codegym.vn/ - 54



3. }

Trong đoạn mã này biểu thức điều kiện là *score > 9*, còn phần thân là *classification = “Xuất sắc”*. Khi thực thi chương trình nếu biến *score* có giá trị lớn hơn 9 thì biến *classification* sẽ được gán giá trị là “Xuất sắc”, còn nếu biến *score* có giá trị bằng hoặc nhỏ hơn 9 thì biến *classification* sẽ không được gán giá trị mới.

***Lưu ý:*** *Nếu phần thân chỉ có một câu lệnh duy nhất thì chúng ta có thể không cần viết dấu mở và đóng ngoặc, còn nếu trong trường hợp phần thân có nhiều hơn 1 câu lệnh thì chúng ta bắt buộc phải mở và đóng ngoặc.* 

Câu lệnh if-else

Câu lệnh *if-else* là dạng đầy đủ của câu lệnh *if*.

Cú pháp:

| 1. if (biểu\_thức\_điều\_kiện) {  2. khối mã được thực thi nếu điều kiện là đúng  3. } else {  4. khối mã được thực thi nếu điều kiện trên là sai  5. } |
| --- |

Câu lệnh *if-else* cũng bao gồm một biểu thức điều kiện, nhưng lại có hai phần thân khác nhau. Một phần thân gắn với phần *if* và một phần thân gắn với phần *else*. Nếu biểu thức điều kiện trả về *true* thì phần thân gắn với *if* được thực thi, còn nếu biểu thức điều kiện trả về *false* thì phần thân gắn với phần *else* được thực thi. Luồng thực thi của câu lệnh if-else có thể được mô tả như lưu đồ dưới đây: 

*Hình 3.1: Luồng thực thi của câu lệnh if-else*

**Ví dụ 1:**

| 1. if (hour < 18){  2. greeting = "Good day";  3. } else {  4. greeting = "Good evening";  5. } |
| --- |

Trong đoạn mã trên, biểu thức điều kiện là *hour < 18*, còn phần thân của *if* là *greeting = “Good day”* và phần thân của *else* là *greeting = “Good evening”*.

/codegym.vn/ - 55



Khi thực thi, nếu biến *hour* có giá trị nhỏ hơn 18 thì biến *greeting* sẽ được gán giá trị là *“Good day”*, còn nếu biến *hour* có giá trị lớn hơn hoặc bằng 18 thì biến *greeting* sẽ được gán giá trị là *“Good evening”*.

**Ví dụ 2:**

| 1. if (number % 2 == 0){  2. alert(number + " is even.");  3. } else {  4. alert(number + " is odd.")  5. } |
| --- |

Trong ví dụ trên, chúng ta đánh giá một số nguyên và phân loại số đó là số chẵn hay là số lẻ. 

Trong trường hợp này, biểu thức điều kiện là để đánh giá xem số hiện tại có chia hết cho 2 hay không, còn hai phần thân là hiển thị thông báo tương ứng. Việc đánh giá chia hết cho 2 hay không được thực hiện thông qua phép chia lấy phần dư và sau đó so sánh với giá trị 0. Nếu phần dư bằng không thì chứng tỏ đây là số chẵn, còn nếu phần dư khác 0 thì số đó là số lẻ.

Chẳng hạn, nếu *number* có giá trị là 4 thì kết quả phép chia cho 2 lấy phần dư là 0, do vậy biểu thức điều kiện sẽ trả về đúng, và phần thân của *if* được thực thi. Nếu *number* có giá trị là 5 thì kết quả phép chia cho 2 lấy phần dư là 1, do vậy biểu thức điều kiện trả về sai, phần thân của *else* được thực thi.

if-else với một dòng lệnh bên trong

Trong trường hợp chỉ có một dòng lệnh bên trong *if* hoặc *else* thì có thể bỏ dấu ngoặc. Ví dụ:

| 1. if(number % 2 == 0)  2. alert(number + " is even.");  3. else  4. alert(number + " is odd.") |
| --- |

5. Cấu trúc điều kiện if-else lồng nhau

Ngoài việc sử dụng câu *lệnh if-else* riêng lẻ, chúng ta cũng có thể đặt câu lệnh *if-else* này trong câu lệnh *if-else* khác. Điều này xảy ra khi chúng ta muốn đánh giá nhiều biểu thức điều kiện trước khi thực thi một thao tác nào đó.

**Cú pháp:**

| 1. if (điều kiện 1) {  2. //khối lệnh thực thi nếu điều kiện 1 đúng  3. if (điều kiện 2) {  4. //khối lệnh thực thi nếu điều kiện 2 đúng  5. } else {  6. //khối lệnh thực thi nếu điều kiện 2 sai  7. }  8. } else {  9. //khối lệnh thực thi nếu điều kiện 1 sai  10. } |
| --- |

/codegym.vn/ - 56



**Ví dụ:**

| 1. if (a > b) {  2. if (a > c) {  3. console.log("Số lớn nhất là a = " + a);  4. } else {  5. console.log("Số lớn nhất là c = " + c);  6. }  7. } else {  8. if (b > c) {  9. console.log("Số lớn nhất là b = " + b);  10. } else {  11. console.log("Số lớn nhất là c = " + c);  12. }  13. } |
| --- |

Trong ví dụ này, chúng ta so sánh các giá trị của 3 biến là a, b và c để tìm ra giá trị lớn nhất trong 3 số đó. Chúng ta cần đặt các câu lệnh điều kiện bên trong các câu lệnh điều kiện khác. 

Chẳng hạn, với trường hợp *a = 5, b = 4, c = 3* thì biểu thức *a > b* sẽ trả về kết quả *true*, do đó phần thân của lệnh *if* này được thực thi, phần thân của lệnh *else* tương ứng sẽ được bỏ qua. Sau đó, biểu thức *a > c* cũng trả về kết quả *true*, do đó phần thân của câu lệnh *if* này cũng sẽ được thực thi, và phần thân của câu lệnh *else* tương ứng sẽ được bỏ qua. Như vậy, chuỗi in ra sẽ là *“Số lớn nhất là a = 5”*.

***Hỏi nhanh:*** *Hãy thử suy luận để xem luồng thực thi của đoạn mã trên sẽ là như thế nào trong trường hợp a = 3, b = 4, c = 5.*

6. Cấu trúc điều kiện if-else bậc thang

Khi chúng ta kết hợp nhiều câu lệnh *if-else* liên tiếp nhau theo *dạng if-else-if* thì chúng ta được một dạng mới thường được gọi là câu lệnh điều kiện *if-else bậc thang*. Gọi là bậc thang bởi vì khi viết chúng được trình bày dạng như các bậc thang hướng từ trên xuống dưới.

Các khối lệnh điều kiện được đặt trong các câu lệnh if nối tiếp nhau, nếu một điều kiện nào đó thoả mãn thì khối lệnh trong thân câu lệnh if sẽ được thực thi, còn nếu điều kiện sai thì sẽ chuyển sang đánh giá điều kiện if tiếp theo cho đến khi kết thúc. Luồng thực thi của khối lệnh *if-else* bậc thang có thể mô tả như lưu đồ sau:

/codegym.vn/ - 57



*Hình 3.2: Luồng thực thi của câu lệnh if-else bậc thang* 

**Cú pháp:**

| 1. if (điều kiện 1) {  2. //khối mã được thực thi nếu điều kiện 1 là đúng  3. } else if (điều kiện 2) {  4. //khối mã được thực thi nếu điều kiện 1 là sai và điều kiện 2 là đúng  5. } else {  6. //khối mã được thực thi nếu điều kiện 1 và điều kiện 2 đều sai  7. } |
| --- |

**Ví dụ:**

| 1. if (time < 10){  2. greeting = "Good morning";  3. } else if (time < 20){  4. greeting = "Good day";  5. } else {  6. greeting = "Good evening";  7. } |
| --- |

Trong ví dụ này, nếu giá trị của *time* là 9 thì điều kiện đầu tiên đúng, do đó biến *greeting* sẽ có giá trị là *“Good morning”*.

Nếu giá trị của *time* là 11 thì điều kiện đầu tiên không đúng, do đó chuyển sang đánh giá điều kiện của câu lệnh *if* tiếp theo, ở đây điều kiện *time < 20* là đúng, do đó biến *greeting* sẽ được gán giá trị là *“Good day”*.

Nếu giá trị của biến *time* là 21 thì điều kiện đầu tiên không đúng, chuyển sang đánh giá điều kiện thứ hai cũng không đúng, do đó chuyển đến thực thi câu lệnh *else* cuối cùng, và kết quả biến greeting sẽ được gán giá trị là *“Good evening”*.

7. Cấu trúc điều kiện switch-case

Câu lệnh *switch-case* là một bộ lựa chọn đa hướng, nó so sánh giá trị của một biểu thức với một danh sách các hằng số nguyên hoặc hằng ký tự. Chúng ta sử dụng câu lệnh *switch-case* trong những trường hợp muốn phân loại luồng thực thi của chương trình dựa vào các điều kiện so sánh bằng. Khi biểu thức so sánh bằng trả về đúng thì khối lệnh tương ứng được thực thi.

/codegym.vn/ - 58



**Cú pháp:**

| 1. switch (param) {  2. case value1:  3. //khối lệnh 1  4. case value2:  5. //khối lệnh 2  6. case value3:  7. //khối lệnh 3  8. default:  9. //Khối lệnh mặc định sẽ được thực thi nếu không chọn được khối lệnh nào khác  10. } |
| --- |

Trong đó *param* là một biểu thức. Trình thực thi sẽ so sánh giá trị của *param* lần lượt với giá giá trị đi theo sau các *case*. Nếu gặp một *case* mang theo giá trị bằng với giá trị của *param* thì các khối lệnh đi sau đó sẽ được thực thi. 

Trong trường hợp không tìm thấy case nào mang giá trị khớp với *param* thì khối lệnh đi theo sau *default* sẽ được thực thi. *default* là một khối lệnh không bắt buộc. ***Lưu ý:*** *Cho dù là thực thi mã đi theo case hay default, lệnh switch sẽ còn tiếp tục thực thi mã của tất cả các case (hay cả mã của default) đứng sau đó, mà không cần xem xét bất cứ điều kiện gì. Quá trình đó có thể được biểu diễn bằng lưu đồ như sau: *

*Hình 3.3: Luồng thực thi của câu lệnh switch-case*

Chúng ta có thể thấy luồng thực thi nối thẳng từ khối lệnh này tới khối lệnh khác mà không thông qua bất kỳ khối hình thoi (khối xét điều kiện) nào. Vì hình ảnh này mà cách thực thi này được gọi là *fall-though* (rơi xuyên qua) trong tiếng Anh. **Ví dụ:**

/codegym.vn/ - 59



| 1. let switcher = 1;  2. switch (switcher) {  3. case 0:  4. alert("Đèn sáng");  5. case 1:  6. alert("Đèn tắt");  7. default:  8. alert("Không tìm thấy công tắc");  9. } |
| --- |

Mặc dù chỉ có một *case* mang giá trị khớp với *switcher*, trình duyệt sẽ thực thi mã của cả *case 1* lẫn *default*, và kết quả là trình duyệt sẽ hiển thị 2 thông báo alert thay vì chỉ 1. 

Từ khóa break

Chúng ta có thể ứng dụng sự đặc biệt trong cách thực thi của lệnh *switch* để dùng chung mã giữa các *case*, nhờ đó tạo ra được những lệnh *switch* ngắn gọn. Tuy nhiên để khắc phục hiện tượng luồng thực thi rơi xuyên qua tất cả các case phía sau, thay vì chỉ thực thi những mã cần thiết, chúng ta cần vận dụng từ khóa *break.*

Từ khóa *break* sẽ làm luồng thực thi ngay lập tức thoát ra khỏi câu lệnh *switch* hiện tại.



*Hình 3.3: Luồng thực thi của câu lệnh switch-case khi có break sau mỗi case*

Từ khóa *break* thường được đặt làm câu lệnh cuối cùng của mỗi *case*, nhờ thế, hiện tượng fall-through được khắc phục.

**Ví dụ:**

/codegym.vn/ - 60



| 1. let switcher = 0;  2. switch (switcher) {  3. case 0:  4. alert("Đèn sáng");  5. break;  6. case null:  7. case undefined:  8. alert("Không tìm thấy công tắc");  9. case 1:  10. default:  11. alert("Đèn tắt");  12. } |
| --- |

Nhờ có *break*, giờ đây nếu switcher có giá trị là 0 thì chỉ có alert “Đèn sáng” hiện lên. Nếu *switcher* mang giá trị là *1* thì alert “Đèn tắt” vẫn hiện lên. Nếu chúng ta thực thi đoạn mã trên với giá trị *switcher* là *null* hay *undefined*, sẽ có 2 alert hiện lên. **Lưu ý:** 

Lệnh *switch* khớp giá trị của *param* với các *case* bằng toán tử === chứ không phải ==.

**Ví dụ:**

| 1. let switcher = "0";  2. switch (switcher) {  3. case 0:  4. alert("Đèn sáng");  5. break;  6. case null:  7. case undefined:  8. alert("Không tìm thấy công tắc");  9. case 1:  10. default:  11. alert("Đèn tắt");  12. } |
| --- |

Đoạn mã trên sẽ khớp giá trị của *switcher* với case *default* thay vì case *0* như chúng ta thường nghĩ.

8. Mã sạch

Quy tắc #1: Quy ước

● Trong các câu lệnh điều kiện, phía sau các từ khóa *if*, *else*, *switch* sẽ là khoảng trắng và tiếp theo là biểu thức điều kiện, kết thúc dòng đầu tiên sẽ là ký tự { để đánh dấu bắt đầu khối mã lệnh nằm bên trong câu lệnh điều kiện đó, trước dấu { sẽ là một khoảng trắng.

● Khối mã lệnh thực thi bên trong câu lệnh điều kiện phải lùi đầu dòng vào bốn khoảng trắng.

● Ký tự } đánh dấu kết thúc khối mã lệnh nằm trong câu điều kiện phải được viết riêng biệt tại dòng cuối cùng

/codegym.vn/ - 61



**Ví dụ:**

Không nên:

| 1. if(number > 0)  2. { number += 1;  3. alert('The number is' + number);} |
| --- |

Nên: 

| 1. if (number > 0) {  2. number += 1;  3. alert('The number is' + number);  4. } |
| --- |

Quy tắc #2: Viết có ý nghĩa

Viết mã nguồn cũng như viết một bài văn, chúng ta cần viết thế nào để cho người đọc mã nguồn dễ dàng hiểu được ý nghĩa của từng dòng mã. Một cách đơn giản để đạt được tính dễ hiểu đối với các biểu thức như trên đó là ”tách biến”: Đưa biểu thức của chúng ta ra bên ngoài khối lệnh *if* và gán giá trị của nó cho một biến (với tên gọi có ý nghĩa).

**Ví dụ:**

Trước khi tách biến:

| 1. if (number % 2 == 0){  2. alert(number + " is even.");  3. } else {  4. alert(number + " is odd.");  5. }  6. |
| --- |

Sau khi tách biến:

| 1. let isEven = number % 2 == 0;  2. if (isEven) {  3. alert(number + " is even.");  4. } else {  5. alert(number + " is odd.");  6. } |
| --- |

9. Các lỗi thường gặp

Lỗi thường gặp #1: Quên dấu ngoặc

Ví dụ:

| 1. if (radius >= 0)  2. area = radius \* radius \* PI;  3. alert("The area is " + area); // Luôn được thực thi, không phụ thuộc vào điều kiện |
| --- |

Trong đoạn mã bên trên, chúng ta muốn thực thi 2 câu lệnh ở trong điều kiện if, tuy nhiên lại quên mất việc đặt chúng vào trong một khối đóng mở ngoặc. Như vậy, khi

/codegym.vn/ - 62



điều kiện đúng thì chỉ có một câu lệnh ở trên được thực thi, dòng lệnh *alert(“The area is” + alert)* sẽ luôn được thực thi mà không bị phụ thuộc vào điều kiện *(radius >= 0).* Trong trường hợp này chúng ta cần phải sửa lại, bổ sung cặp dấu { } vào vị trí phù hợp.

Sau khi bổ sung cặp dấu ngoặc {}:

| 1. if (radius >= 0) {  2. area = radius \* radius \* PI;  3. alert("The area is " + area);  4. } |
| --- |

Lỗi thường gặp #2: Đặt dấu ; ở cuối câu lệnh điều kiện 

Lỗi thứ hai đó là đặt thừa dấu chấm phẩy sau điều kiện của if. Trong trường hợp này câu lệnh if không có tác dụng bởi vì nó kết thúc ngay sau khi đánh giá điều kiện. Có nghĩa là khối lệnh đằng sau nó sẽ luôn được thực thi cho dù điều kiện ở trước có đúng hay không.

Ví dụ:

| 1. if (radius >= 0); {  2. area = radius \* radius \* PI;  3. alert("The area is " + area);  4. } |
| --- |

Để khắc phục lỗi này, chúng ta chỉ cần xoá dấu chấm phẩy ở cuối câu lệnh if đi. Xóa dấu *(;)* ở cuối câu lệnh if:

| 1. if (radius >= 0) {  2. area = radius \* radius \* PI;  3. alert("The area is " + area);  4. } |
| --- |

Lỗi thường gặp #3: So sánh giá trị boolean

Lỗi thứ ba xảy ra khi chúng ta so sánh các giá trị thuộc kiểu dữ liệu boolean. Bản chất thì đây không phải là một lỗi, bởi vì các dòng lệnh vẫn thực thi đúng như mong đợi. Nhưng việc so sánh giá trị kiểu boolean như trong trường hợp này là không cần thiết. **Không nên:**

| 1. if (even == true) {  2. alert("It is even.");  3. } |
| --- |

Chúng ta sẽ dễ dàng thấy rằng kết quả của biểu thức so sánh *even = true* là hoàn toàn giống với giá trị của biến *even*. Nếu *even* có giá trị *true* thì biểu thức sẽ có giá trị *true*. Nếu *even* có giá trị là *false* thì biểu thức sẽ có giá trị là *false*. Như vậy thì việc sử dụng biểu thức ở đây là không cần thiết.

/codegym.vn/ - 63



| **Giá trị của biến**  **Kết quả của biểu thức (even == true)**  **even** | |
| --- | --- |
| true | true |
| false | false |

**Nên**: 

| 1. if (even) {  2. alert("It is even.");  3. } |
| --- |

Lỗi thứ ba xảy ra khi chúng ta so sánh các giá trị thuộc kiểu dữ liệu boolean. Bản chất thì đây không phải là một lỗi, bởi vì các dòng lệnh vẫn thực thi đúng như mong đợi. Nhưng việc so sánh giá trị kiểu boolean như trong trường hợp này là không cần thiết. Chẳng hạn, chúng ta sẽ dễ dàng thấy rằng kết quả của biểu thức so sánh even = true là hoàn toàn giống với giá trị của biến even. Nếu even có giá trị true thì biểu thức sẽ có giá trị true. Nếu even có giá trị là false thì biểu thức sẽ có giá trị là false. Như vậy thì việc sử dụng biểu thức ở đây là không cần thiết.

| **Giá trị của biến**  **Kết quả của biểu thức (even == true)**  **even** | |
| --- | --- |
| true | true |
| false | false |

Lỗi thường gặp #4: Nhầm lẫn khối if-else

Lỗi thứ 4 đó là nhầm lẫn khi kết hợp các khối lệnh if-else, lỗi này thường xảy ra khi chúng ta quên các dấu đóng mở ngoặc hoặc nhầm lẫn do khoảng cách đầu dòng của các câu lệnh không được căn chỉnh cẩn thận.

**Không nên:**

| 1. if (first\_number > second\_number)  2. if (first\_number > third\_number)  3. alert("A");  4. else  5. alert("B");  6. |
| --- |

Ở ví dụ trên, khi nhìn vào đoạn mã thứ nhất, chúng ta thường nghĩ rằng khối lệnh else được áp dụng cho câu lệnh điều kiện đầu tiên. Có nghĩa là nếu i nhỏ hơn hoặc bằng j thì sẽ hiển thị thông báo “B”.

Tuy nhiên thực tế thì không phải như thế. Câu lệnh else được áp dụng cho điều kiện if thứ hai ở bên trong câu lệnh if đầu tiên. Do đó, điều kiện else này chỉ được thực thi nếu điều kiện đầu tiên đúng và điều kiện thứ hai sai, tức là i lớn hơn j và i nhỏ hơn k. **Nên:**

/codegym.vn/ - 64



| 1. if (first\_number > second\_number) {  2. if (first\_number > third\_number) {  3. alert("A");  4. } else {  5. alert("B");  6. }  7. } |
| --- |

Lỗi thường gặp #5: Gán giá trị boolean 

Lỗi thường gặp thứ 5 là thực hiện việc so sánh để gán các giá trị tương ứng cho một biến. Đây thực chất cũng không phải là một lỗi, bởi vì chương trình vẫn thực thi đúng như mong đợi, nhưng việc sử dụng câu lệnh điều kiện trong trường hợp này là không cần thiết.

**Không nên:**

| 1. if (number % 2 === 0) {  2. even = true;  3. } else {  4. even = false;  5. }  6. |
| --- |

Lỗi thường gặp thứ 5 là thực hiện việc so sánh để gán các giá trị tương ứng cho một biến. Đây thực chất cũng không phải là một lỗi, bởi vì chương trình vẫn thực thi đúng như mong đợi, nhưng việc sử dụng câu lệnh điều kiện trong trường hợp này là không cần thiết.

Chẳng hạn, trong đoạn mã này, giá trị của biến even hoàn toàn tương ứng với giá trị của biểu thức so sánh kết quả phép chia lấy số dư và giá trị 0. Do đó, thay vì sử dụng if-else để gán giá trị thì chúng ta hoàn toàn có thể gán trực tiếp kết quả của biểu thức cho biến even.

| **Giá trị của biến**  **Kết quả của biểu thức (number % 2 ===**  **even**  **0)** | |
| --- | --- |
| true | true |
| false | talse |

Trong đoạn mã trên, giá trị của biến *even* hoàn toàn tương ứng với giá trị của biểu thức so sánh kết quả phép chia lấy số dư và giá trị 0. Do đó, thay vì sử dụng if-else để gán giá trị thì chúng ta hoàn toàn có thể gán trực tiếp kết quả của biểu thức cho biến even.

/codegym.vn/ - 65



| **Giá trị của biến**  **Kết quả của biểu thức (number % 2 ===**  **even**  **0)** | |
| --- | --- |
| true | true |
| false | talse |

**Nên:** 

| 1. let even = number % 2 === 0; |
| --- |

Lỗi thường gặp #6: Lặp mã

Lỗi thường gặp cuối cùng đó là việc lặp mã. Đây thực ra cũng không phải là một lỗi, bởi vì chương trình vẫn thực thi đúng như mong đợi nhưng việc mã bị lặp lại là một điều tối kỵ trong lập trình.

Ví dụ:

| 1. //Không nên  2. if (inState) {  3. tuition = 5000;  4. alert("The tuition is " + tuition);  5. } else {  6. tuition = 15000;  7. alert("The tuition is " + tuition);  8. } |
| --- |

Trong trường hợp này, hai câu lệnh *alert()* y hệt nhau cùng xuất hiện trong một đoạn mã ngắn. Chúng ta có thể dễ dàng xử lý bằng cách đơn giản là đưa câu lệnh đó ra cuối khối if-else. Như vậy thì câu lệnh *alert()* này được thực thi cả trong hai trường hợp là biến *inState* có giá trị là *true* hoặc *false*. Nghĩa là hành vi của chương trình không bị thay đổi.

Nên:

| 1. if (inState) {  2. tuition = 5000;  3. } else {  4. tuition = 15000;  5. }  6. alert("The tuition is " + tuition); |
| --- |

10. Bài thực hành

Bài 1: Kiểm tra năm nhuận

**Mục tiêu:**

Luyện tập sử dụng cấu trúc điều kiện if.

**Mô tả:**

/codegym.vn/ - 66



Trong phần này, chúng ta sẽ phát triển một ứng dụng nhằm kiểm tra xem một năm có phải là năm nhuận hay không. Ứng dụng cho phép người dùng nhập vào một năm, sau đó sẽ đưa ra thông báo là năm đó là năm nhuận hay không phải là năm nhuận. Năm nhuận là một năm đặc biệt, được cộng thêm một ngày để giữ cho lịch được đồng bộ với lịch thiên văn.

Cách xác định năm nhuận: Những năm chia hết cho 4 là năm nhuận, ngoại trừ những năm chia hết cho 100 mà không chia hết cho 400. Từ đó, có thể rút gọn thành các quy tắc xác định năm nhuận: “Những năm chia hết cho 4 mà không chia hết cho 100 là năm nhuận. Những năm chia hết cho 100 mà không chia hết cho 400 thì KHÔNG PHẢI là năm nhuận.Những năm chia hết đồng thời cho 100 và 400 là năm nhuận”. 

**Hướng dẫn:**

Bước 1: Tạo dự án mới với file index.html và khối lệnh JavaScript Bước 2: Nhập dữ liệu từ bàn phím

| 1. let year = parseInt(prompt("Enter a year")); |
| --- |

Bước 3: Phân loại và hiển thị kết quả

| 1. if (year % 4 == 0) {  2. if (year % 100 == 0) {  3. if (year % 400 == 0) {  4. alert(year + " is a leap year");  5. } else {  6. alert(year + " is NOT a leap year");  7. }  8. } else {  9. alert(year + " is a leap year");  10. }  11. } else {  12. alert(year + " is NOT a leap year");  13. } |
| --- |

Bước 4: Chạy và quan sát kết quả.

Sử dụng các năm sau để kiểm tra tính đúng đắn của chương trình:

| **Năm** | **Kết quả** |
| --- | --- |
| 12 | Năm nhuận |
| 13 | Năm không nhuận |
| 1000 | Năm không nhuận |
| 2000 | Năm nhuận |

**Lưu ý:**

Với các điều kiện như trên, chúng ta có thể chỉnh sửa để mã nguồn tốt hơn như sau:

/codegym.vn/ - 67



| 1. let year = parseInt(prompt("Enter a year"));  2. let isLeapYear = false;  3. if (year % 4 == 0) {  4. if (year % 100 == 0) {  5. if (year % 400 == 0) {  6. isLeapYear = true;  7. }  8. } else {  9. isLeapYear = true;  10. }  11. }  12. if (isLeapYear) {  13. alert(year + " is a leap year");  14. } else {  15. alert(year + " is NOT a leap year");  16. } |
| --- |

**Đảm bảo clean code:** 

Trong đoạn mã trên, các điều kiện bên trong các câu lệnh if sẽ là khó hiểu, bởi vì bản thân các biểu thức không trực tiếp nói rõ ý nghĩa của nó. Chúng ta có thể sử dụng kỹ thuật tách biến để các câu lệnh này dễ hiểu hơn. Ví dụ:

| 1. let isLeapYear = false;  2.  3. let isDivisibleBy4 = year % 4 == 0;  4. if (isDivisibleBy4) {  5. let isDivisibleBy100 = year % 100 == 0;  6. if (isDivisibleBy100) {  7. let isDivisibleBy400 = year % 400 == 0;  8. if (isDivisibleBy400) {  9. isLeapYear = true;  10. }  11. } else {  12. isLeapYear = true;  13. }  14. }  15.  16. if (isLeapYear) {  17. alert(year + " is a leap year");  18. } else {  19. alert(year + " is NOT a leap year");  20. } |
| --- |

*Mã nguồn tham khảo: https://github.com/codegym-vn/introduction-to-programming with-javascript/tree/dev/chapter-03/01-kiem-tra-nam-nhuan*

Bài 2: Luyện tập với cấu trúc if...else

**Mục tiêu:**

Luyện tập sử dụng cấu trúc if/else.

**Mô tả:**

Sử dụng cấu trúc if…else, viết chương trình nhập vào một chuỗi câu trả lời cho câu hỏi “What is the “official” name of JavaScript?”. Nếu câu trả lời nhập vào là

/codegym.vn/ - 68



“ECMAScript”, thì hiển thị thông báo: “Right!”, ngược lại hiển thị thông báo: “Didn’t know? ECMAScript!”

**Hướng dẫn:**

Bước 1: Tạo file index.html

Bước 2: Thêm thẻ *<script>* thực hiện viết các mã JavaScript

Bước 3: Khai báo biến value nhận giá trị được nhập vào từ hộp thoại 

| 1. let value = prompt('What is the "official" name of JavaScript?', ''); |
| --- |

Bước 4: Sử dụng cấu trúc if…else kiểm tra giá trị của value

| 1. if (value == 'ECMAScript') {  2. alert('Right!');  3. } else {  4. alert("You don't know? ECMAScript!");  5. } |
| --- |

Bước 5: Thực thi chương trình. Quan sát kết quả.

*Mã nguồn tham khảo: https://github.com/codegym-vn/introduction-to-programming with-javascript/tree/dev/chapter-03/02-luyen-tap-voi-cau-truc-if-else* Bài 3: Luyện tập với cấu trúc if...else...if

**Mục tiêu**

Luyện tập sử dụng cấu trúc if...else...if

**Mô tả**

Viết chương trình kiểm tra đăng nhập hệ thống của người dùng theo yêu cầu sau: ● Yêu cầu người dùng nhập tên từ bàn phím.

● Nếu tên nhập vào là “Admin” thì yêu cầu nhập mật khẩu.

● Nếu mật khẩu là “TheMaster” thì in ra chuỗi “Welcome”.

● Nếu mật khẩu nhập vào là null in ra chuỗi “Canceled”.

● Còn lại in ra chuỗi “Wrong password”.

● Nếu tên nhập vào là null in ra chuỗi “Canceled”.

● Còn lại in ra chuỗi “I don’t know you”.

**Hướng dẫn**

Bước 1: Tạo file login.html

Bước 2: Thêm thẻ <script> thực hiện viết các mã JavaScript

Bước 3: Tạo biến userName nhận giá trị nhập vào từ hộp thoại

| 1. let userName = prompt("Who's there?", ''); |
| --- |

Bước 4: Kiểm tra giá trị của userName

| 1. if (userName == 'Admin') {  2. //code vào đây  3. } else if (userName == null) {  4. alert('Canceled');  5. } else {  6. alert("I don't know you");  7. } |
| --- |

/codegym.vn/ - 69



Bước 5: Nếu userName là “Admin”. Khai báo biến pass nhận giá trị nhập vào từ hộp thoại.

| 1. let pass = prompt('Password?', ''); |
| --- |

Bước 6: Kiểm thử

| 1. if (pass == 'TheMaster') {  2. alert('Welcome!');  3. } else if (pass == null) {  4. alert('Canceled.');  5. } else {  6. alert('Wrong password');  7. } |
| --- |

Bước 7: Thực thi chương trình. Quan sát kết quả. 

*Mã nguồn tham khảo: https://github.com/codegym-vn/introduction-to-programming with-javascript/tree/dev/chapter-03/03-luyen-tap-voi-cau-truc-if-else-if* Bài 4: Luyện tập với cấu trúc switch...case

**Mục tiêu:**

Luyện tập với cấu trúc switch...case

**Mô tả:**

Viết lại cấu trúc if sau thành cấu trúc switch...case

| 1. if (browser == 'Edge') {  2. alert("You've got the Edge!");  3. } else if (browser == 'Chrome' || browser == 'Firefox' || browser == 'Safari' || browser == 'Opera') { 4. alert('Okay we support these browsers too');  5. } else {  6. alert('We hope that this page looks ok!');  7. } |
| --- |

**Hướng dẫn**

Cấu trúc if được viết lại với cấu trúc switch...case như sau:

| 1. switch (browser) {  2. case 'Edge':  3. alert("You've got the Edge!");  4. break;  5. case 'Chrome':  6. case 'Firefox':  7. case 'Safari':  8. case 'Opera':  9. alert('Okay we support these browsers too');  10. break;  11. default:  12. alert('We hope that this page looks ok!');  13. } |
| --- |

/codegym.vn/ - 70



*Mã nguồn tham khảo: https://github.com/codegym-vn/introduction-to-programming with-javascript/tree/dev/chapter-03/04-luyen-tap-voi-cau-truc-switch-case* Bài 5: Luyện tập với cấu trúc switch...case

**Mục tiêu:**

Luyện tập với cấu trúc switch...case.

**Mô tả:** 

Viết lại cấu trúc if sau thành cấu trúc switch...case.

| 1. let a = parseInt(prompt('a?', ''));  2. if (a == 0) {  3. alert(0);  4. }  5. if (a == 1) {  6. alert(1);  7. }  8. if (a == 2 || a == 3) {  9. alert('2,3');  10. } |
| --- |

**Hướng dẫn:**

Cấu trúc if được viết lại với cấu trúc switch...case như sau:

| 1. let a = parseInt(prompt('a?', ''));  2. switch (a) {  3. case 0:  4. alert(0);  5. break;  6. case 1:  7. alert(1);  8. break;  9. case 2:  10. case 3:  11. alert('2,3');  12. break;  13. } |
| --- |

*Mã nguồn tham khảo:*

*https://github.com/codegym-vn/introduction-to-programming-with javascript/tree/dev/chapter-03/05-luyen-tap-voi-cau-truc-switch-case* Bài 6: Luyện tập với cấu trúc if...else

**Mục tiêu:**

Luyện tập với cấu trúc if...else.

**Mô tả:**

Sử dụng cấu trúc if...else, nhập vào một số, kiểm tra giá trị của số nhập vào. ● Nếu số nhập vào lớn hơn 0 hiển thị 1

● Nếu số nhập vào nhỏ hơn 0 thì hiển thị -1

● Còn lại hiển thị 0

**Hướng dẫn:**

/codegym.vn/ - 71



Bước 1: Tạo file sign.html

Bước 2: Khai báo biến value nhận giá trị đầu vào từ hộp thoại, mặc định không nhập gì value = 0

| 1. let value = prompt('Type a number', 0); |
| --- |

Bước 3: Viết khối if…else..if

| 1. if (value > 0) {  2. alert(1);  3. } else if (value < 0) {  4. alert(-1);  5. } else {  6. alert(0);  7. } |
| --- |

Bước 4: Thực thi chương trình. Nhập giá trị 0, -5, 5. Quan sát kết quả nhận được Bước 5: Thay thế cấu trúc if bằng cấu trúc switch…case. Làm lại các bước như trên. *Mã nguồn tham khảo: https://github.com/codegym-vn/introduction-to-programming with-javascript/tree/dev/chapter-03/06-luyen-tap-voi-cau-truc-if-else* Bài 7: Tính tổng 2 số 

**Mục tiêu:**

Luyện tập sử dụng cấu trúc điều kiện với toán tử "?", " :"

**Mô tả:**

Hãy viết chương trình nhập vào giá trị cho a và b. Tính tổng a và b, nếu tổng nhỏ hơn 4, hiển thị chuỗi Below, ngược lại hiển thị Over. Lưu ý sử dụng toán tử ? : **Hướng dẫn:**

**Bước 1:** Tạo file add.html

**Bước 2:** Thêm thẻ <script> thực hiện viết các mã JavaScript

| 1. <!DOCTYPE html>  2. <html>  3. <body>  4. <script>  5. //code vào đây  6. </script>  7. </body>  8. </html> |
| --- |

**Bước 3:** Khai báo biến a, b nhập vào giá trị cho a, b từ hộp thoại

| 1. let a = parseInt(prompt("a: "));  2. let b = parseInt(prompt("b: ")); |
| --- |

**Bước 4:** Khai báo biến result lưu kết quả

| 1. let result = (a + b < 4) ? 'Below' : 'Over';  2. alert(result); |
| --- |

**Bước 5:** Thực thi chương trình. Quan sát kết quả.

*Mã nguồn tham khảo: https://github.com/codegym-vn/introduction-to-programming with-javascript/tree/dev/chapter-03/07-tinh-tong-2-so*

/codegym.vn/ - 72



Bài 8: Luyện tập với chuỗi

**Mục tiêu:**

Luyện tập với chuỗi.

**Mô tả:**

Viết chương trình nhập vào giá trị cho chuỗi message.

● Nếu giá trị nhập vào là Employee thì hiển thị chuỗi Hello. 

● Còn lại nếu giá trị nhập vào là Director thì hiển thị chuỗi Greetings ● Còn lại nếu giá trị nhập vào chuỗi rỗng thì hiển thị No login

● Còn lại hiển thị chuỗi rỗng

Sử dụng toán tử ? : để viết chương trình với cấu trúc điều kiện như sau:

| 1. let message;  2. if (login === 'Employee') {  3. message = 'Hello';  4. } else if (login === 'Director') {  5. message = 'Greetings';  6. } else if (login === '') {  7. message = 'No login';  8. } else {  9. message = '';  10. } |
| --- |

**Hướng dẫn:**

**Bước 1:** Tạo file *employee.html*

**Bước 2:** Thêm thẻ <script> thực hiện viết các mã JavaScript

| 1. <!DOCTYPE html>  2. <html>  3. <body>  4. <script>  5. //code vào đây  6. </script>  7. </body>  8. </html>  9. |
| --- |

**Bước 3:** Khai báo biến message

| 1. let login = prompt('Login?', '');  2. let message = (login === "Employee") ?  3. "Hello" :  4. (login == "Director") ?  5. "Greetings" :  6. (login == "") ?  7. "No login" :  8. ""; |
| --- |

**Bước 4:** Hiển thị thông báo

| 1. alert(message); |
| --- |

**Bước 5:** Thực thi chương trình. Quan sát kết quả.

/codegym.vn/ - 73



*Mã nguồn tham khảo: https://github.com/codegym-vn/introduction-to-programming with-javascript/tree/dev/chapter-03/08-luyen-tap-voi-chuoi*

Bài 9: Sự kiện bàn phím

**Mục tiêu:**

Luyện tập xử lý sự kiện bàn phím.

**Mô tả:** 

Di chuyển hình ảnh nhân vật Nobita lên, xuống, sang trái, sang phải sử dụng các phím tương ứng. Việc này được thực hiện bằng cách thay đổi thuộc tính tọa độ của thẻ khi xử lý event.



*Hình 3.5: Minh hoạ sự kiện bàn phím*

**Hướng dẫn:**

**Bước 1:** Chèn ảnh nobita.png vào tài liệu html

| 1. <img id="nobita" src="images/nobita.png" style="position:absolute;left:0; top:0;" height="100" width="80"/> |
| --- |

**Bước 2:** Xây dựng các hàm xử lý khi nhấn phím lên

| 1. function upArrowPressed() {  2. let element = document.getElementById("nobita");  3. element.style.top = parseInt(element.style.top) - 5 + 'px';  4. } |
| --- |

**Bước 3:** Xây dựng các hàm xử lý khi nhấn phím xuống

| 1. function downArrowPressed() {  2. let element = document.getElementById("nobita");  3. element.style.top = parseInt(element.style.top) + 5 + 'px';  4. } |
| --- |

**Bước 4:** Xây dựng các hàm xử lý khi nhấn phím sang trái

| 1. function leftArrowPressed() {  2. let element = document.getElementById("nobita");  3. element.style.left = parseInt(element.style.left) - 5 + 'px';  4. } |
| --- |

**Bước 5:** Xây dựng các hàm xử lý khi nhấn phím sang phải

/codegym.vn/ - 74



| 1. function rightArrowPressed() {  2. let element = document.getElementById("nobita");  3. element.style.left = parseInt(element.style.left) + 5 + 'px';  4. } |
| --- |

**Bước 6:** Xây dựng các hàm nhận các phím

| 1. function moveSelection(evt) {  2. switch (evt.keyCode) {  3. case 37:  4. leftArrowPressed();  5. break;  6. case 39:  7. rightArrowPressed();  8. break;  9. case 38:  10. upArrowPressed();  11. break;  12. case 40:  13. downArrowPressed();  14. break;  15. }  16. }  17. |
| --- |

**Bước 7:** Xây dựng hàm doReady() 

| 1. function docReady() {  2. window.addEventListener('keydown', moveSelection);  3. } |
| --- |

**Bước 8:** Gọi hàm doReady()

| 1. <body onload="docReady()"> |
| --- |

**Bước 9:** Chạy chương trình. Sử dụng các phím lên-xuống-sang trái-sang phải để di chuyển ảnh và quan sát kết quả.

*Mã nguồn tham khảo: https://github.com/codegym-vn/introduction-to-programming with-javascript/tree/dev/chapter-03/09-su-kien-ban-phim*

Bài 10: Sự kiện chuột

**Mục tiêu:**

Luyện tập thao tác sự kiện chuột.

**Mô tả:**

Làm sử dụng sự kiện click chuột để di chuyển hình ảnh một quả bóng. Mỗi lần click quả bóng sẽ được di chuyển sang trái/phải một khoảng cách.

**Hướng dẫn:**

**Bước 1:** Tạo giao diện như hình

/codegym.vn/ - 75