**Code Cùng Casio**

*Contact: minhmok095@gmail.com*

**I. Giới thiệu**

Trong tài liệu này mình sẽ sử dụng Casio fx580 VN X, và bạn nên có kiến thức về lập trình trước khi tiếp tục.

Thực tế, bên Trung Quốc đã tìm ra cách để thao túng máy để có thể gõ được mã máy trong các dòng fx991 và fx580 từ hồi trước 2018 rồi.

Dưới đây mình sẽ sử dụng hoàn toàn những phép tính được cho chứ không dùng trick này, cách mình viết dưới đây **hoàn toàn không hữu ích bằng**, nhưng mà dễ hiểu hơn. Nếu bạn muốn up level hơn thì có một số nguồn sau:

* http://casiocalc.wikidot.com/
* https://community.casiocalc.org/

**II. Các khái niệm cơ bản**

**II.1. Khởi chạy**

Ta có thể coi từng câu lệnh là các phép toán, và kể từ đời Casio fx500 trở đi thì ta đã có thể tính được nhiều phép tính sử dụng dấu “:” như sau:

Bấm “CALC”, bạn sẽ thấy kết quả là 2 của phép tính , bấm “=”, kết quả sẽ là 5 của phép tính, bấm “=” tiếp, kết quả sẽ là của phép tính , bấm “=” tiếp, ta sẽ về lại phép tính 1+1 ban đầu.

Cơ bản là ta coi các phép toán là các câu lệnh, mỗi câu lệnh ngăn cách nhau bằng dấu “:”, khi chạy hết thì nó sẽ về lại ban đầu. Lý do tại sao phải bấm “CALC” ở đầu thì mình sẽ giải thích ở phần tiếp theo.

**II.2. Biến**

Trong Casio ta có 10 biến A, B, C, D, E, F, x, y, z, M, với giá trị ban đầu bằng 0. Lý do phải bấm “CALC” ở đầu là do nó sẽ giúp ta đặt giá trị các biến trước khi khởi chạy.

Ví dụ, ta có một “chương trình” như sau:

Khi bấm “CALC”, màn hình sẽ hiện lên yêu cầu ta nhập giá trị của A, sau đó là của B. Mình sẽ đặt A là 1, B là 6. Khi chạy, màn hình sẽ đưa ra kết quả 2, sau đó là 9.

Ta cũng có thể gán giá trị cho biến trong 1 câu lệnh như sau:

Ví dụ, giả sử mình có chương trình như sau:

Bấm “CALC”, đặt A bằng 1, thì khi ta chạy đến câu lệnh thứ 2 ta sẽ thấy kết quả là 2, vì biến A đã được sửa thành A+1 (mà A=1 nên 1+1=2).

Nhưng chỉ có vậy thì không có tác dụng gì lắm, ta sẽ sử dụng biến trong phần tiếp theo về lặp.

**Giới hạn lưu trữ của Casio**

Tuy ta có thể thực hiện được các thuật toán, ta cần hiểu được về giới hạn lưu trữ của Casio. Các số trong Casio sẽ chỉ chuẩn xác cho đến 9999999999, nếu lớn hơn thì sẽ được đưa về dạng xấp xỉ và không còn chính xác nữa. Nên 1 biến sẽ lưu được xấp xỉ 4.15 bytes, mà Casio có 10 biến nên tổng cộng ta lưu được 41.5 bytes.

**II.3. Lặp**

Một mảnh ghép quan trọng trong lập trình chính là lặp. Trong Casio ta không trực tiếp có cách để lặp các câu lệnh của chúng ta, nhưng ta nhận ra rằng khi chạy hết các câu lệnh, máy sẽ nhảy về phép tính/câu lệnh đầu tiên, và các giá trị của biến nếu bị thay đổi thì vẫn được giữ nguyên. Ta sẽ sử dụng đặc điểm này như một vòng lặp.

**Giải fibonacci**

Để làm một ví dụ thì mình sẽ ứng dụng lặp để tìm số Fibonacci. Nếu bạn chưa biết thì dãy Fibonacci là dãy mà các số trong đó là tổng của 2 số trước đó, và 2 số đầu tiên là 0, 1.

Ví dụ: 0, 1, 1, 2, 3, 5, 8, 13,…

Để giải bài này, mình có thuật toán như sau, đặt A=0, B=1, C=0, coi C là số Fibonacci ở vị trí thứ n, A và B là 2 số liên tiếp trước nó, mình sẽ lặp liên tục, mỗi lần lặp mình sẽ cho C = A + B, rồi thay A = B, B = C.

Và trong Casio thì ta sẽ có chương trình sau:

Bấm “CALC”, đặt A=0, B=1, C=0, bấm “=” liên tục, mỗi lần ta về lại phép tính đầu ta sẽ thấy C được thay đổi thành số Fibonacci tiếp theo trong dãy.

**Vấn đề về tự động hóa**

Nhưng trong thực tế, nếu phải bấm “=” liên tục để có thể lặp được nhiều lần thì vô cùng mệt mỏi, nên khi code trong Casio ta phải cố đưa các vấn đề về lại một phép toán đơn giản mà không cần lặp, hoặc sử dụng các vòng lặp khác sau đây.

**Lặp tổng và tích**

Ta có thể lặp để tính tổng và tích như sau:

Tổng:

Tích:

Nếu đã học qua thì ta biết kí hiệu đầu là sigma và nó sẽ lặp từ start đến end (cho x = start, cộng 1 vào x đến khi bằng end thì dừng), và sẽ cộng phần bên trong liên tục vào 1 tổng. Kí hiệu ở dưới (pi notation) cũng như vậy nhưng thay vì tổng thì nó là tích.

Hạn chế của 2 cách lặp trên là ta không thể thay đổi được biến trong vòng lặp, và kết quả trả về phải là tổng hoặc tích. Nhưng chúng có thể được sử dụng cho một số mục đích khá thú vị, điển hình là 1 ví dụ ở trong phần sau về cấu trúc rẽ nhánh, điều kiện.

**II.4. Làm tròn số bằng hàm Int**

Trong Casio có một hàm rất mạnh mà mình sẽ sử dụng trong hầu hết các vấn đề cơ bản nhất, đó là hàm Int. Hàm Int có tác dụng làm tròn xuống các số thập phân, ví dụ thì sẽ trả về 3, nhưng dựa vào đây mà ta có thể mở rộng nó hơn.

**Làm tròn bình thường**

Nếu hàng thập phân >= 0.5 thì làm tròn lên, còn không thì làm tròn xuống. Ta có thể làm vậy như sau:

Do 1 số lớn hơn hoặc bằng 0.5 cộng 0.5 sẽ ra 1 số lớn hơn hoặc bằng 1, làm tròn xuống sẽ là 1, còn nếu không thì vẫn bé hơn 1 và làm tròn xuống là 0.

**Làm tròn lên**

Để làm tròn lên thì ta có thể làm như sau:

Ý tưởng của ta sẽ là, nếu 1 số là 0 thì 0 + 0.999…99 làm tròn xuống vẫn là 0, còn nếu số từ 0.000…01 trở lên thì cộng với 0.999…99 sẽ ra 1 số lớn hơn hoặc bằng 1, làm tròn xuống là 1. Vì đã nêu trên là Casio chỉ chính xác với 10 chữ số, nên ta lấy là đủ ổn để ra 0.999…99 mà ta nêu trên.

**II.5. Cấu trúc rẽ nhánh, điều kiện**

Trong Casio, không có lệnh if nào cả, nên ta sẽ sử dụng toán để giải quyết vấn đề này. Thứ ta cần đạt được là nếu một điều kiện thỏa mãn thì trả về 1 giá trị này, còn không thì trả về 1 giá trị khác. Thực tế có rất nhiều cách, nhưng sau đây mình sẽ liệt kê các cách mình hay dùng nhất.

**Giá trị tuyệt đối**

Giá trị tuyệt đối có thể được coi là một cấu trúc rẽ nhánh, nếu n dương thì trả về n, âm thì trả về -n.

Ví dụ, ta có thể áp dụng giá trị tuyệt đối trong một số vấn đề, chẳng hạn như tìm số lớn hơn và bé hơn. Ta được biết rằng tổng của tổng và hiệu 2 số chia 2 ra số lớn hơn, hiệu của tổng và hiệu chia 2 ra số bé hơn. Nhưng làm sao tính được hiệu mà không biết số nào lớn hơn số nào bé hơn? Chúng ta sẽ sử dụng giá trị tuyệt đối, và từ đó ta có:

Số lớn hơn:

Số bé hơn:

**Nhân với 0 (và 1)**

Cách trung lập nhất để tạo ra một cấu trúc rẽ nhánh là coi một điều kiện là một biểu thức, lấy nó nhân với một biểu thức khác. Nếu biểu thức bằng 0 thì kết quả sẽ chỉ ra 0, còn không thì nó sẽ trả về tích của 2 biểu thức.

Nhưng thường thì như vậy vẫn chưa đủ, để có thể tính toán được với biểu thức còn lại thì kết quả của biểu thức điều kiện nên chỉ được đưa về 0 và 1, do nhân với 1 thì biểu thức còn lại giữ nguyên chứ không bị phá. Nên dưới đây mình có làm một số điều kiện cơ bản mà giá trị trả về là 0 và 1.

Phần này hơi khó hiểu nên giờ không cần quan tâm lắm, vì trong phần ví dụ ở cuối mục này mình sẽ sử dụng cách này một cách dễ hiểu.

**Hàm tanh**

Trong Casio đã có sẵn hàm tanh. Hàm tanh(x) sẽ trả về 1 số trong khoảng 0 đến 1 nếu x >= 0, trả về 1 số trong khoảng -1 đến 0 nếu x <= 0, tỉ lệ thuận với x, và trả về 0 nếu x=0. Sử dụng tính chất này, ta có thể kiểm tra xem 1 số là âm hay dương, nhưng cũng chính vì thế nó còn có thể được sử dụng để tạo phép so sánh.

**So sánh hơn/kém**

Thứ mà ta muốn đạt được là 1 phương trình gồm A và B, nếu A >= B thì phương trình trả về 1, nếu A < B thì chương trình trả về 0.

Một số lớn hơn hoặc bằng số khác là khi hiệu số đó và số kia lớn hơn hoặc bằng 0, bé hơn khi khi hiệu âm. Do đó bước đầu ta có:

Phép tính trên sẽ trả về 1 số từ 0 đến -1 nếu A <= B, 0 đến 1 nếu A >= B. Nhưng ta muốn kết quả trả về là 0 hoặc 1, nên ta sẽ cộng 1 vào phép tính trên rồi chia 2, do đó bây giờ ta có:

Giờ phép tính của ta sẽ trả về kết quả trong khoảng 0 và 1, và khi A – B = 0 thì phép tính ra 0.5, giờ ta chỉ cần làm tròn để đưa kết quả về chỉ 0 hoặc 1:

Rút gọn:

Phép tính trên giờ sẽ trả về 1 nếu A >= B, 0 nếu A < B, đúng thứ mà ta cần.

**So sánh bằng**

Thứ ta muốn đạt được trong trường hợp này là trả về 1 nếu A = B, 0 nếu A khác B. Ta nhận thấy tanh(0) sẽ trả về 0, tanh bất kì số nào khác trả về 1 số khác 0 trong khoảng từ -1 đến 1, mà A-B sẽ bằng 0 nếu A = B, nên ta có ý tưởng như sau: Làm tròn xuống của 1-0 luôn bằng 1, làm tròn xuống của 1-n với n là số bất kì lớn hơn 0 bé hơn hoặc bằng 1 sẽ luôn bằng 0. Do đó ta có công thức sau:

**Một số mẹo nhỏ**

Ta có thể kiểm tra xem nhiều cặp AB, CD, EF xem tất cả có bằng nhau không một cách ngắn gọn bằng cách:

Ta có thể làm điều tương tự với phép so sánh hơn:

Vì các công thức ở trên trả về 1 và 0, ta có thể lấy điều kiện ngược lại của chúng bằng cách lấy 1 trừ đi nó. Ví dụ ta có công thức sau để trả về 1 nếu A khác B và 0 nếu A bằng B:

**Áp dụng**

Sau đây ta sẽ sử dụng những gì ta học được từ trước để giải quyết một số vấn đề cần đến điều kiện nhé.

**Kiểm tra số nguyên tố**

Để giải được bài này mình sẽ sử dụng lặp tính tích, nhân với 0 và 1, và so sánh bằng với tanh.

Với n là số cần kiểm tra, ta lặp từ 2 đến căn n, nếu n chia hết cho bất kì số nào thì n không phải là số nguyên tố.

Một số chia hết cho số kia chỉ khi thương là một số nguyên, nên để kiểm tra chia hết thì ta chỉ cần so xem làm tròn của thương có bằng thương hay không là được, mà mình đã có cách so sánh bằng ở trên, nên để kiểm tra chia hết thì ta có code sau:

Phép tính sẽ trả về 0 nếu A chia hết cho x, 1 nếu A không chia hết cho x.

Nếu giờ ta dùng vòng lặp tính tích, thì chỉ cần 1 thừa số bằng 0, tức chỉ cần chia hết cho 1 số trong vòng lặp thì số mà chúng ta kiểm tra không phải là số nguyên tố, do đó cả tích sẽ bằng 0. Ngược lại, nếu tất cả các số đều không thỏa mãn, thì tích các số sẽ là 1 vì mọi thừa số là 1 do không chia hết cho số nào cả.

Dựa vào ý tưởng trên, ta có chương trình sau:

Nó sẽ trả về 0 nếu A là hợp số, trả về 1 nếu A là số nguyên tố.

**II.6. Cắt và chỉnh sửa số**

Giả sử số là 123456, muốn cắt được số từ chữ số thứ 3 đến chữ số thứ 5 (tức 345).

Bước 1, ta cần cắt bỏ “12” để còn lại 3456. Ta nhận thấy 123456 mod 10000 bằng 3456, nhưng Casio không có cách để lấy phần dư (dấu ÷R không cho ta một kết quả có thể gán được vào biến nên không dùng được), nên ta có thể sử dụng công thức sau:

Bước 2, ta cần cắt bỏ “6” để còn lại 345. Ta nhận thấy 3456 chia 10 bằng 345.6, lấy phần nguyên còn 345, bây giờ phép tính của ta thành:

Tổng quát hóa, coi số lớn là A, B là độ dài số biết trước, ta lấy 1 số con với C là điểm bắt đầu, D là điểm kết thúc bằng công thức: