**TRƯỜNG ĐẠI HỌC FPT**

****

**ASSIGNMENT 2**

**DATA STRUCTURE AND ALGORITHM**

*Người thực hiện*: **Vũ Thiên Ân – SE160296**

Lớp **: SE1616**

Khoá  **: 16**

**THÀNH PHỐ HỒ CHÍ MINH, NĂM 2021**

PHẦN ĐÁNH GIÁ CỦA GIẢNG VIÊN

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Tp. Hồ Chí Minh, ngày tháng năm

(kí và ghi họ tên)

**MỤC LỤC**

***Trang***

**1.** **Giới thiệu bài toán 3**

**2.** **Giải quyết bài toán**

**2.1 Mô tả cấu trúc dữ liệu 3**

**2.2 Sơ đồ giải thuật 10**

**2.3 Hiện thực 27**

**2.4 Kết quả và thảo luận 49**

**3. Kết luận 58**

**Tự đánh giá 59**

1. **Giới thiệu bài toán**

BÀI TOÁN CON KHỈ VÀ NẢI CHUỐI

Trong một căn phòng có một con khỉ, một cái ghế và một cây gậy. Các vật có vị trí ban đầu bất kỳ.

Một nải chuối được treo trên trần phòng có vị trí ban đầu bất kì.

Con khỉ có thể thực hiện những hành động sau:

* Di chuyển trên sàn nhà
* Di chuyển cái ghế (kéo và đẩy)
* Trèo lên 1 vật
* Cầm 1 vật
* Khều nải chuối

Tìm hành động tốt nhất để con khỉ lấy được nải chuối

1. **Giải quyết bài toán**
   1. **Mô tả cấu trúc dữ liệu**
      1. **Điều kiện kết thúc bài toán**

Con khỉ chỉ có thể lấy được nải chuối khi các điều kiện sau đều thỏa:

* Vị trí của ghế ở dưới nải chuối
* Khỉ đứng trên ghế
* Khỉ có gậy
* Khỉ khều nải chuối

Tuy nhiên, ta cũng có thể nói: khỉ lấy được chuối nếu sau các hành động của khỉ xảy ra trạng thái (S) mà các điều kiện sau đều thỏa:

* Khỉ có gậy.
* Vị trí của ghế ở dưới nải chuối.
* Khỉ đứng kế bên ghế hoặc đứng ở trên ghế.

Ở đây, “khỉ đứng kế bên ghế” có thay thế hai điều kiện “khỉ đứng trên ghế” và “khỉ khều nải chuối”. Nếu ghế đã đúng vị trí, khỉ đã có gậy và khỉ đứng kế bên ghế, thì việc khỉ sau đó leo lên đứng trên ghế và khều chuối chỉ mang tính chất thủ tục để hoàn tất bài toán. Phân tích lời giải ở phần sau sẽ dùng trạng thái (S) này làm trạng thái phải đạt được để hoàn tất bài toán.

* + 1. **Kí hiệu và số trường hợp ban đan đầu**

Chuỗi hành động tốt nhất của khỉ được quyết định bởi ví trí ban đầu của 4 vật: khỉ, chuối, ghế và gậy. Bài toán sẽ được phân tích trong không gian 2 chiều, mỗi vật đều có một tọa độ theo chiều ngang.

Gọi:

Khỉ là M,

Chuối là B,

Ghế là C,

Gậy là S.

Khỉ và các vật có thể được đại diện bằng những khối hộp. Sau đây là hình minh họa căn phòng và các vật:

S

B

C

M

Thứ tự các vật, từ trái qua phải, có thể là [M- B - C - S], [B - C - M - S] , v.v.  
Có tổng cộng 4! = 24 cách sắp xếp vị trí ban đầu của 4 vật này. Ở đây, ta chưa bàn đến trường hợp các vật có tọa độ ban đầu xung đột với nhau, tức là các hình hộp ở hình minh họa trên chiếm cùng một khoản không gian, hoặc có tọa độ một cạnh bên giống nhau (2 vật “tiếp xúc” nhau). Các trường hợp này có thể được xem như: khỉ có gậy ngay từ ban đầu, ghế ở đúng vị trí ngay từ ban đầu v.v. Các trường hợp này sẽ được phân tích trong phần **2.2.5**.

* + 1. **Tính đối xứng và số trường hợp thật sự cần xét**

Xét 2 trường hợp [M - B - C - S] và [S - C - B - M]. Cách sắp xếp các vật trong 2 trường hợp này đối xứng qua một trục ở giữa, được mình họa ở dưới. Nếu diễn tả di chuyển của khỉ theo vị trí của vật mà khỉ muốn đi đến, thay vì đề cập đến việc di chuyển trái hay phải, thì chuỗi hành động để giải 2 trường hợp này như nhau.

M

B

C

S

M

B

C

S

M đi đến C **→** M leo qua bên kia của C

M

B

C

S

M

B

C

S

M đi đến S, lấy được S **→** M quay lại C

ĐẨY

ĐẨY

M  
Có S

B

C

M  
Có S

B

C

M đẩy C đến dưới B

M đi đến C **→** M leo qua bên kia của C

**→** M đi đến S, lấy được S **→** M quay lại C

**→** Mđẩy C đến dưới B

Trong 24 cách sắp xếp các vật, 12 cách sắp xếp là “đối xứng” với 12 trường hợp khác. Nhờ có tính đối xứng này và cách diễn tả hành động nêu trên, ta chỉ cần phân tích một nửa số trường hợp, cụ thể là 12 trường hợp. Việc khỉ di chuyển về hướng trái hoặc phải sẽ được hiện thực hóa trong code và ta không cần phải phân tích toàn bộ 24 trường hợp.

* + 1. **Minh họa 12 trường hợp và chuỗi hành động tốt nhất**

Sau đây là minh họa 12 trường hợp được chọn và chuỗi hành động tốt nhất của mỗi trường hợp. Chuỗi hành động kết thúc khi đạt được trạng thái (S) nêu ở phần **2.1.1**.

**Trường hợp 1 đến 6**

M nằm ở ngoài cùng bên trái của danh sách. Thứ tự các vật có dạng [M – x – x – x], với x có thể là C, B hoặc S . Các trường hợp đối xứng của 6 trường hợp này có M là vị trí ngoài cùng bên phải, thứ tự các vật có dạng [x – x – x – M].

M

B

C

S

M

B

C

S

M

B

C

S

M đi đến C **→** M leo qua bên kia của C

**→** Mkéo theo C đến khi C dưới B

**→** M di chuyển tiếp đến S, lấy được S

**→** M quay lại C

M đi đến C **→** M leo qua bên kia của C

**→** Mkéo theo C đến khi C dưới B

**→** MLấy được S trên đường

M đi đến C **→** M leo qua bên kia của C

**→** M đi đến S lấy được S **→** M quay lại C

**→** Mđẩy C đến dưới B

11111

21111

31111

6

5

41111

M

B

C

S

M

B

C

S

M

B

C

S

M đi đến C **→** M Lấy được S trên đường

**→** Mđẩy C đến dưới B

M đi đến C **→** M Lấy được S trên đường

**→** Mkéo C đến dưới B

M đi đến C **→** M Lấy được S trên đường

**→** Mkéo C đến dưới B

**Trường hợp 7 – 12**

M là vị trí thứ 2 từ trái sang, tức là [x – M – x – x]. Các trường hợp đối xứng với 6 trường hợp có M là vị trí thứ 3 từ trái sang, [x – x – M– x].

M

B

C

S

M

B

C

S

M

B

C

S

M

B

C

S

M đi đến C

**→** Mkéo C đến dưới B

**→** M Lấy được S trên đường

M đi đến C.

**→** Mkéo C đến dưới B

**→** M di chuyển tiếp đến S, lấy được S

**→** M quay lại C

M đi đến C **→** M Lấy được S trên đường

**→** Mkéo C đến dưới B

M đi đến C **→** M leo qua bên kia của C

**→** M đi đến S lấy được S **→** M quay lại C

**→** Mđẩy C đến dưới B

10

9

81111

7

111111

12

M

B

C

S

M

B

C

S

M đi đến S lấy được S

**→** M đi đến C

**→** Mkéo C đến dưới B

M đi đến S lấy được S

**→** M đi đến C

**→** Mđẩy C đến dưới B

**Phân tích trường hợp 11 và 12**

Hành động tốt nhất cho trường hợp 11 và 12 là “đi đến gậy và lấy gậy”, thay vì “đi đến ghế” như các trường hợp khác. Trường hợp 11 và 12 có chung một cách sắp xếp các vật:

*“Chuối (B) cùng phía ghế (C) so với khỉ (M) VÀ {B, C} khác phía gậy (S) so với M”*

Trong hai trường hợp này, nếu khỉ chọn đi đến ghế đầu tiên, di chuyển ghế, sau đó quay lại lấy gậy, thì khỉ sẽ phải lặp lại một quãng đường không cần thiết. Ở dưới là minh họa so sánh hai lựa chọn nàycho trường hợp 11. Trường hợp 12 tương tự.

111111

111111

Đi đến S lấy S

C

B

M

S

C

B

M

S

Phần trùng không cần thiết

Kéo C đến B

Đi đến C

Quay lại C

Đi đến S

Kéo C đến B

Đi đến C

* 1. **Sơ đồ giải thuật**
     1. **Quyết định hành động đầu tiên**

Trong toàn bộ các trường hợp, hành động đầu tiên của khỉ có thể là:

Đi đến ghế, (M đi đến C) - trường hợp 1 – 10

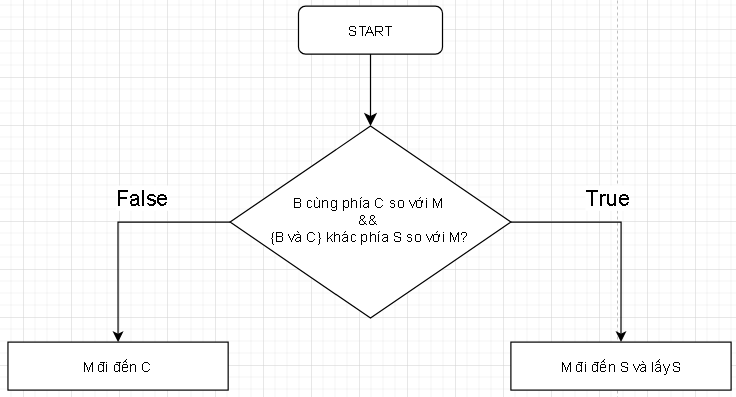
Hoặc

Đi đến gậy, (M đến S lấy được S) - trường hợp 11 và 12

Điều kiện cho hành động này là:

*“Chuối (B) cùng phía ghế (C) so với khỉ (M) VÀ {B, C} khác phía gậy (S) so với M”*

Đến đây, ta có thể vẽ flowchart để quyết định hành động đầu tiên như sau:



Điều kiện 1

Case 1 - 10

Case 11, 12

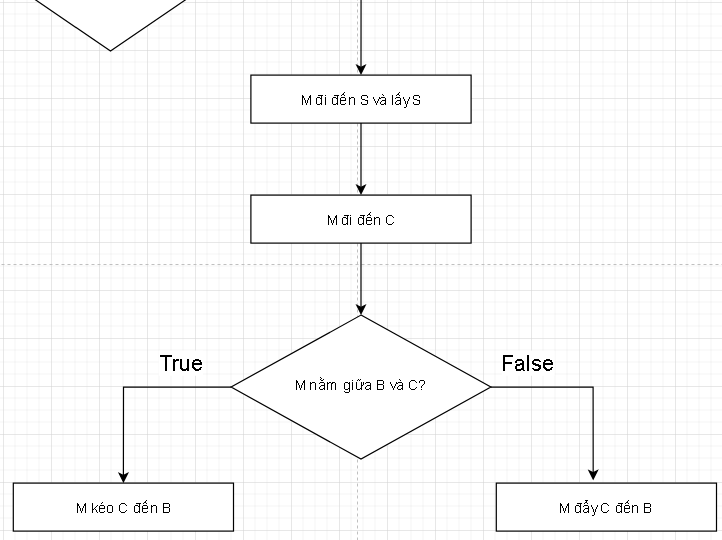
(2)

(1)

Ta gọi điều kiện duy nhất hiện tại là điều kiện 1.

* + 1. **Lời giải cho trường hợp 11 và 12**

Tiếp tục xét 2 trường hợp 11 và 12, sau khi lấy được gậy, khỉ sẽ đi đến ghế. Sau đó khỉ **đẩy** ghế đến chuối, nếu như lúc đó ghế nằm giữa khỉ và chuối, hoặc khỉ **kéo** ghế đến chuối, nếu như lúc đó khỉ nằm giữa chuối và ghế. Ta vẽ tiếp nhánh (2) của flowchart:



(2)

Điều kiện 2

Case 11

Case 12

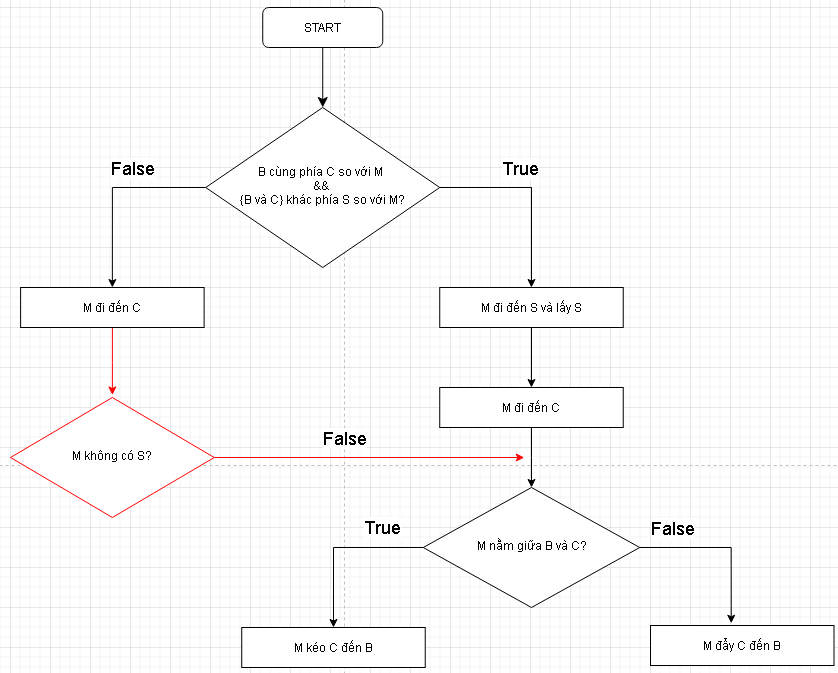
Ta sẽ gọi điều kiện kiểm tra *“M nằm giữa B và C”* là điều kiện 2.

Khi đến bước cuối cùng của flowchart trên, ta đạt được trạng thái (S) và kết thúc bài toán. Vậy ta đã có lời giải cho trường hợp 11 và 12.

* + 1. **Lời giải cho trường hợp 4, 5, 6 và 9**

Trong các trường hợp 4, 5, 6 và 9, trên đường đi đến ghế, khỉ “vô tình” lấy được gậy. Khi đến được ghế, vì đã có gậy rồi, việc làm tiếp theo của khỉ chỉ là di chuyển ghế đến dưới chuối. Điều này tương tự trường hợp 11 và 12 sau khi khỉ đã lấy được gậy.

Như vậy, trong flowchart, khi khỉ đến ghế, ta kiểm tra xem khỉ đã có gậy hay chưa, nếu có thì chuỗi hành động tiếp theo tương tự như trường hợp 11 và 12.Ta có thể thêm vào flowchart phần màu đỏ như sau:



Điều kiện 2

Điều kiện 1

(2)

(1)

Case 4, 5, 6, 9

* + 1. **Lời giải cho trường hợp 1, 2, 3, 7, 8 và 10**

Với các trường hợp 1, 2, 3, 7, 8 và 19, khi khỉ đến được ghế, có thể chia làm 2 tình huống sau:

Trong 4 trường hợp 1, 2, 3 và 10, khỉ chưa có gậy và phải leo bên kia của ghế bởi vì:

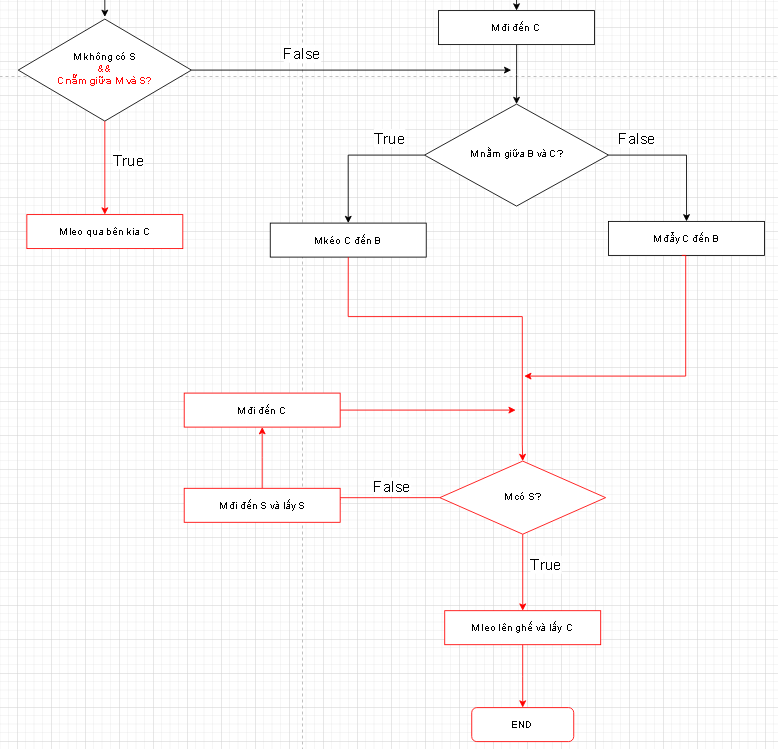
“*Ghế nằm giữa khỉ và gậy.”* (trường hợp 1, 2, 3 và 10)

Ngược lại, trong trường hợp 7 và 8, mặc dù khi đến ghế khỉ chưa có gậy, khỉ không cần phải leo qua bên kia của ghế bởi vì:

“*Khỉ nằm giữa ghế và gậy.”* (trường hợp 7 và 8)

Với trường hợp 7, khỉ sẽ “vô tình” lấy được gậy trên đường kéo ghế đến dưới chuối.Với trường hợp 8, hành động tốt nhất sau khi đến được ghế vẫn là kéo theo ghế đến dưới chuối. Khỉ sau đó bỏ lại ghế, đi lấy gậy rồi quay lại ghế.

B

Tổng kết lại các suy luận do trên, ta có thể thêm vào flowchart (phần màu đỏ) như sau: 

B

Case 4, 5, 6, 7, 9

Case 8

Case 1, 2, 3, 10

Case 4, 5, 6, 7, 8, 9

Điều kiện 2

(2)

(1)

Điều thứ nhất được thêm vào ở nhánh (1) là điều kiện *“C nằm giữa M và S?”* được kết hợp với *“M không có S?”*. Nhờ có điều kiện kép này mà ta tách được 2 nhóm trường hợp 4, 5, 6, 7, 8, 9 và nhóm trường hợp 1, 2, 3, 10.

Ngoài ra, ta đã thêm vào flowchart một điều kiện *“M có S?”* sau việc khỉ di chuyển ghế đến dưới chuối. Điều kiện này đảm bảo chuỗi hành động của trường hợp số 8 sẽ kết thúc ở trạng thái (S), tức là thỏa *“khỉ có gậy”*.

Cuối cùng ta thêm vào thủ tục khỉ leo lên ghế, khều chuối và kết thúc bài toán.

Ta đã có lời giải cho trường trường hợp 4, 5, 6, 7, 8, 9. Tuy nhiên, ta vẫn phải hoàn tất lời giải cho nhóm trường hợp 1, 2, 3 và 10.

**Tiếp tục theo dõi hành động của khỉ trong trường hợp 1 và 2** sau khi khỉ đã leo qua bên kia ghế.

11111

C

B

M

S

81111

C

B

M

S

71111

21111

M

B

C

S

M

B

C

S

Sau khi khỉ leo qua bên kia ghế, trường hợp 1 trở thành gần giống như trường hợp 8 trong khi trường hợp 2 trở thành gần giống như trường hợp 7. Khỉ đều ở giữa ghế và gậy. Điều khác biệt duy nhất là trong trường hợp 1 và 2, sau khi nhảy qua ghế khỉ đã đứng sát ghế rồi, do đó có thể ngay lập tức kéo ghế về phía chuối.

**Tiếp tục xét các trường hợp 3 và 10**

101111

31111

M

B

C

S

M

B

C

S

1231111

M

B

C

S

M

B

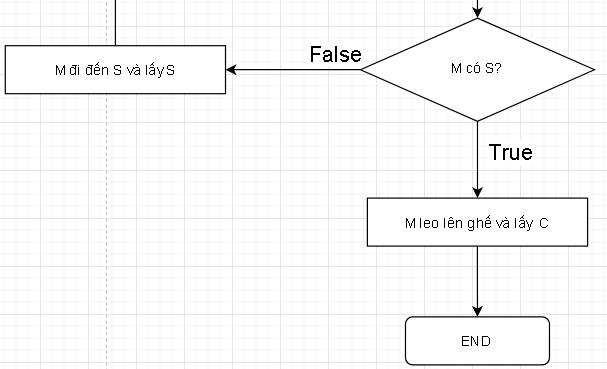
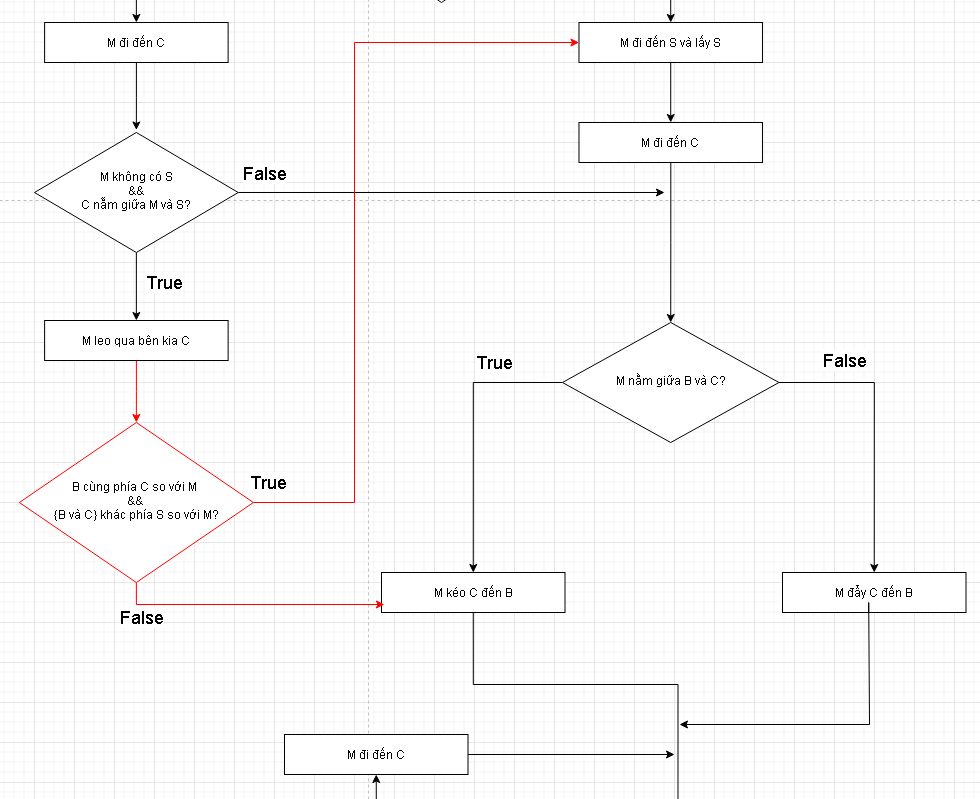
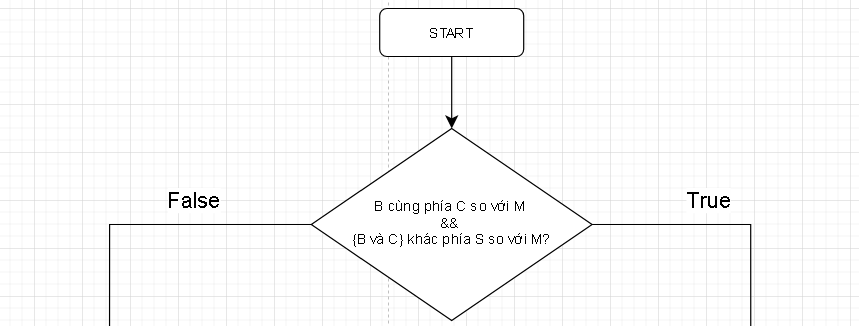
C

S

Sau khỉ khỉ leo qua bên kia ghế, trường hợp 3 và 10 trở trường hợp đối xứng với trường hợp 12. Điều này nghĩa là chuỗi hành động sau đó sẽ tương tự như trường hợp 12. Ta sẽ cần thêm vào flowchart một điều kiện giống như điều kiện để quyết định hành động đầu tiên đã bàn ở phần 2.2.1, tức là kiểm tra lại điều kiện 1, đó là:

*“Chuối (B) cùng phía ghế (C) so với khỉ (M) VÀ {B, C} khác phía gậy (S) so với M.”*

Điều kiện này đúng trong trường hợp 3 và 10 và không đúng trong trường hợp 1 và 2. Flowchart sẽ được chỉnh thành như sau:

Case 1, 2

Điều kiện 1

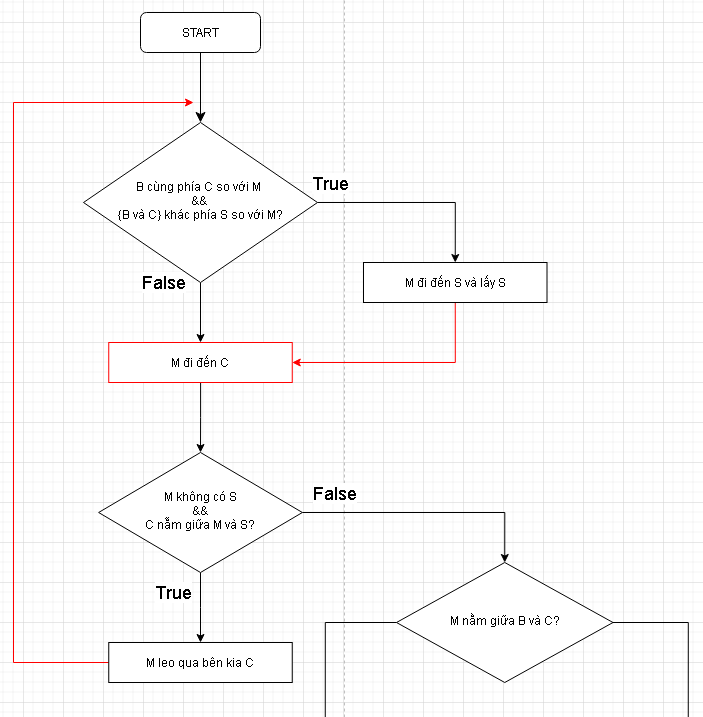
Case 3, 10

Điều kiện 2

Điều kiện 1

B

Ta có thể làm gọn flowchart bằng cách kết hợp yếu tố giống nhau, cụ thể là điều kiện 1 xuất hiện 2 lần và hành động “M đi đến C” cũng xuất hiện 2 lần.



Điều kiện 1

Điều kiện 2

Đối với trường hợp 1 và 2, sau khi leo qua bên kia của ghế thì hành động *“M đi đến C”* là không cần thiết vì khỉ đã đứng kế bên ghế rồi. Do đó yếu tố *“M đi đến C”* để trong flowchart có vẻ là dư thừa cho hai trường hợp này. Tuy nhiên, ta có thể hiểu hành động *“M đi đến C”* là *“khỉ đi đến ghế, nếu như đã đứng kế bên ghế rồi thì khỏi cần đi”*. Trong code hiện thực hóa hành động này, ta sẽ kiểm tra khỉ đã đứng kế bên ghế hay chưa trước khi đi, nếu đã đứng kế bên ghế rồi thì sẽ không thực hiện hành động.

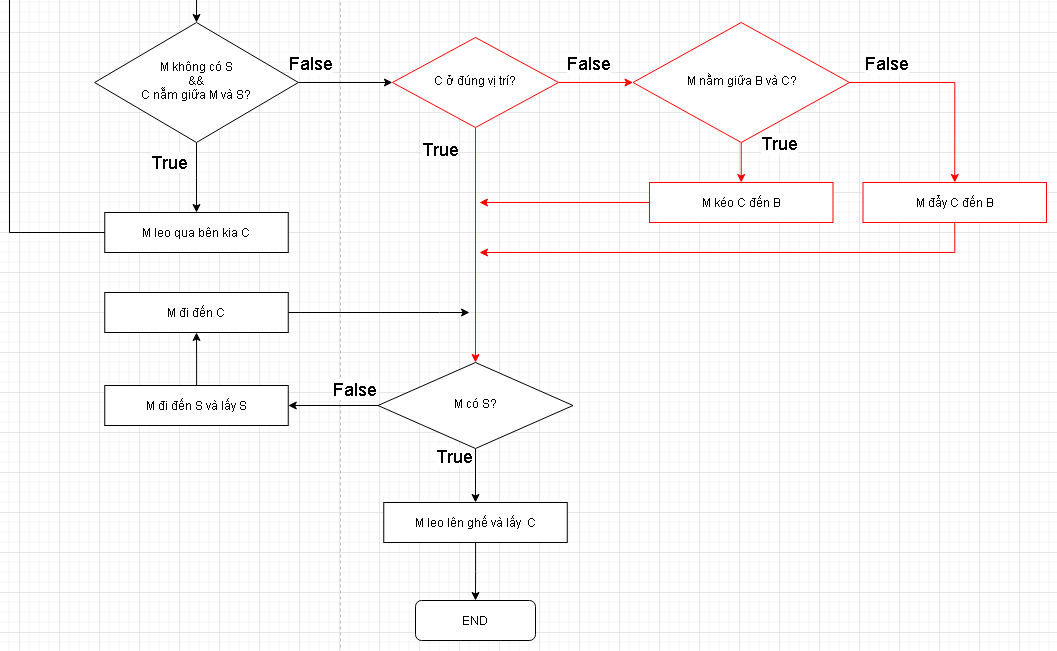
* + 1. **Các trường hợp có các vật tọa độ xung đột với nhau**

Flowchart hiện tại danh cho các trường hợp mà các vật không có vị trí (tọa độ) ban đầu xung đột nhau, tức là không kể đến các trường hợp như khỉ có gậy ngay từ đầu hoặc ghế ở đúng vị trí ngay từ đầu, v.v.Ta cần xét các trường hợp này và thay đổi flowchart nếu cần thiết để tăng tính khách quan.

* + - 1. **Trường hợp ghế và chuối cùng tọa độ ban đầu**

**Xét trường hợp ghế (C) và chuối (B) có tọa độ giống nhau hoặc “tiếp xúc nhau”.** “Tiếp xúc” nghĩa là các hình hộp đại diện 2 vật này có tọa độ một cạnh bên giống nhau. Điều này nghĩa là ghế có vị trí như mong muốn (dưới chuối) ngay từ ban đầu và khỉ không cần kéo hay đẩy ghế chuyển ghế. Ta cần xem lại các điều kiện trong flowchart có liên quan đến cả hai tọa độ B và tọa độ C.

* + Điều kiện 1: nếu như B và C có cùng tọa độ thì hiển nhiên B cùng phía C so với M. Do đó điều kiện 1 không cần được thay đổi.
  + Điều kiện 2: nếu C đã có vị trí đúng rồi thì việc di chuyển C là không cần thiết, và không cần xét tiếp điều kiện 2. Ta sẽ kiểm tra điều kiện *“C ở đúng vị trí?”* ngay trước điều kiện 2. Ta gọi đây là điều kiện 3 Việc này giống như đặt câu hỏi: *“Ghế đã ở đúng chỗ chưa? Nếu rồi thì khỏi cần di chuyển.”*

Flowchart sẽ được thay đổi thành: 

Điều kiện 2

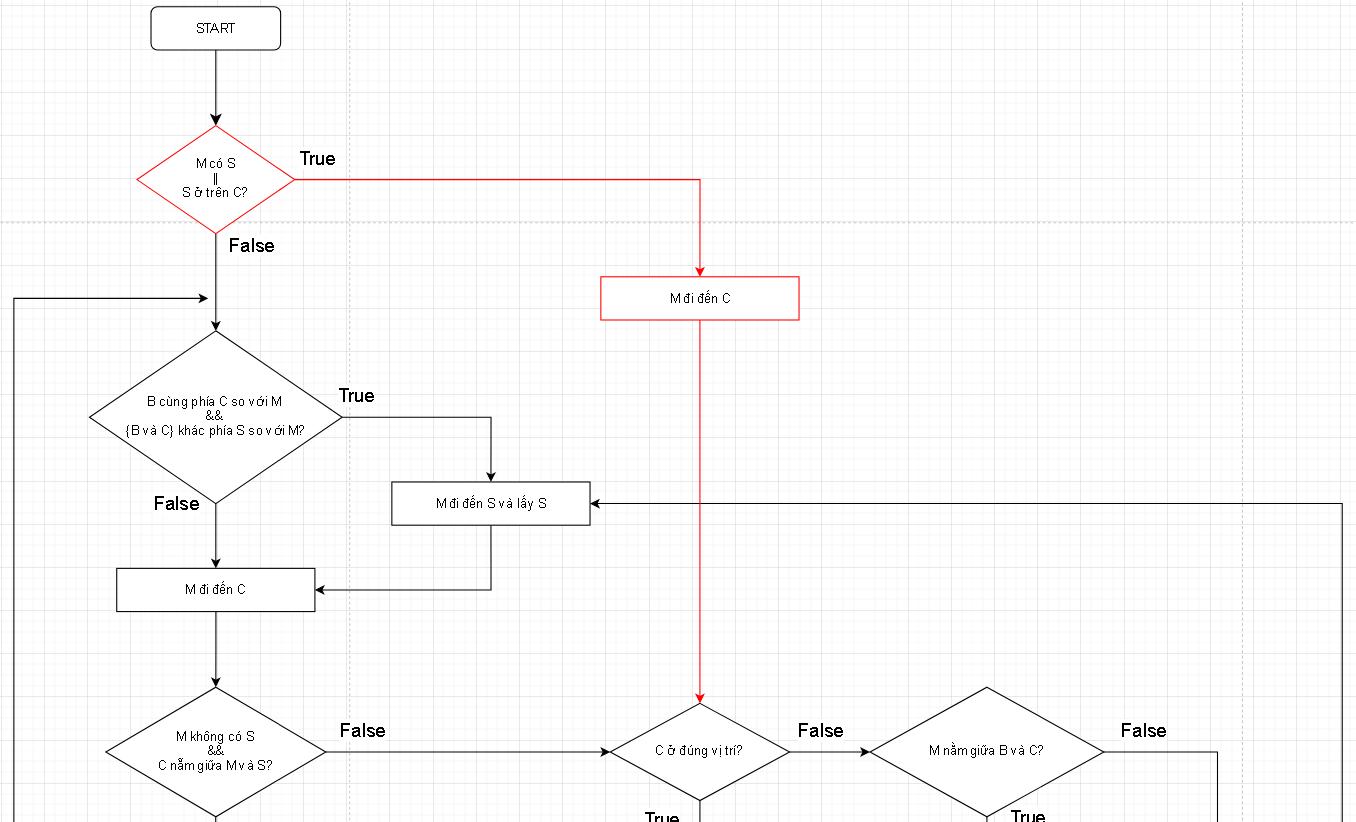
Điều kiện 3

B

* + - 1. **Trường hợp khỉ và gậy cùng tọa độ ban đầu và trường hợp ghế và gậy cùng tọa độ ban đầu**

**Xét trường hợp khỉ (M) và gậy (S) có tọa độ giống nhau hoặc “tiếp xúc nhau”.** “Tiếp xúc” nghĩa là các hình hộp đại diện 2 vật này có tọa độ một cạnh bên giống nhau. Ta xem điều này như việc khỉ có gậy ngay từ đầu, khỉ sẽ không cần phải lo việc đi lấy gậy và có thể đi thẳng ngay đến ghế. Khỉ tiếp theo có thể xét điều kiện 3 để di chuyển ghế nếu cần thiết hoặc ngược lại leo lên ghế, khều chuối và kết thúc bài toán ngay lập tức. Flowchart ta đang có dành cho các trường hợp khỉ chưa có sẵn gậy, và bắt đầu với điều kiện 1. Ta sẽ thêm vào điều kiện *“M có S?”* ngay trước điều kiện 1. Nếu đúng thì khỉ sẽ đi ngay đến ghế, nếu sai thì ta mới xét tiếp điều kiện 1.

**Xét trường hợp ghế (C) và gậy (S) có cùng vị trí**, tức là gậy ở trên ghế ngay từ đầu. Ta không cần xét trường hợp “tiếp xúc” của 2 vật này. Trong trường hợp này, hành động tốt nhất đầu tiên của khỉ là đi đến ghế. Khi đến ghế, khỉ sẽ có được gậy và chuỗi hành động sau đó tương tự như trường hợp khỉ có gậy ngay từ đầu đã bàn ở trên. Ta gộp 2 điều kiện: *“M có S” hoặc “S ở trên C”* thành điều kiện 4 và đặt trước điều kiện 1. Vòng lặp quay lại trước điều kiện 1 sẽ được chỉnh lại thành quay lại trước điều kiện mới này, để vẫn đảm bảo kiểm tra được điều kiện *“M có S”*, rồi sau đó mới xét điều kiện 1. Flowchart sẽ được thay đổi thành:



Điều kiện 4

Điều kiện 2

Điều kiện 3

Điều kiện 1

* + - 1. **Các trường hợp trùng vị trí khác**

**Xét các trường hợp ban đầu chuối (B) và gậy (S) có cùng vị trí**, tức là gậy ở dưới chuối. Điều này đồng nghĩa việc B cùng phía S so với M, chuỗi hành động của khỉ sẽ theo nhánh False của điều kiện 1. Flowchart hiện có thật ra đã bao gồm tình huống này và không cần điều chỉnh.

**Xét các trường hợp ban đầu khỉ (M) và chuối (B) có cùng vị trí**, tức là khỉ ở ngay dưới chuối. Sẽ có hai nhóm trường hợp chính:

**Nếu gậy cùng phía ghế so với khỉ** thì chuỗi hành động của khỉ sẽ theo nhánh False của điều kiện 1. Dưới đây là minh họa của 2 tình huống như vậy. 2 trường hợp “đối xứng” của 2 trường hợp này cũng có chuỗi hành động theo nhánh False của điều kiện 1. Flowchart hiện có đã bao gồm lời giải cho các trường hợp này.

C

B

M

S

C

B

M

S

**Nếu gậy và ghế khác phía so với khỉ**, hành động đầu tiên tốt nhất có thể là đi đến ghế hoặc cũng có thể là đi đến gậy. Quãng đường phải đi của 2 chuỗi hành động được minh họa ở dưới là như nhau.

Lựa chọn 1: M đi đến S lấy được S

**→** M đi đến C **→** Mkéo C đến dưới B

Lựa chọn 2: M đi đến C **→** Mkéo C đến dưới B

**→** M di chuyển tiếp đến S, lấy được S

M

B

C

S

Vì khỉ có thể chọn 1 trong hai chuỗi hành động, ta có thể chọn giúp khỉ nhánh True của điều kiện 1, tức là đi lấy gậy trước, và vẫn có kết quả đúng. Để bao gồm tình huống này, ta đổi điều kiện 1 thành như sau:

*(B cùng vị trí với M || B khác phía C so với M) && S khác phía C so với M*

**Xét trường hợp khỉ (M) và ghế (C) có vị trí ban đầu như nhau, tức là khỉ ở trên ghế**. Trường hợp này khá đặc biệt vì ta cần thêm một nhánh riêng vào flowchart ngay trước điều kiện 4. Ta kiểm tra điều kiện này ngay khỉ bắt đầu và gọi nó là điều kiện 5.

Tùy vào khỉ đã có gậy hay chưa, hành động đầu tiên của khỉ sẽ khác nhau. Nếu khỉ chưa có gậy, khỉ sẽ bước xuống ghế để lấy gậy. Hướng xuống ghế tốt nhất là về hướng của gậy. Cách sắp xếp của các vật lúc đó sẽ tương tự như một trong các tình huống 7, 8, 12 và các tình huống “đối xứng” của chúng. Chuỗi hành động tiếp theo của khỉ sẽ đi theo nhánh flowchart bắt đầu với điều kiện 4.

Nếu khỉ ban đầu ở trên ghế và cũng đã có gậy, ta xét tiếp điều kiện ghế có ở đúng vị trí. Nếu ghế chưa đúng vị trí, khỉ phải bước xuống ghế để di chuyển ghế. Vì không còn phải lo đi lấy gậy, khỉ bước xuống hướng nào của ghế cũng được, vì sau đó việc kéo hay đẩy ghế đến dưới chuối đều có cùng độ dài quãng đường. Ta chọn giúp khỉ lựa chọn bước xuống ghế về hướng của chuối. Chuỗi hành động tiếp theo của khỉ gần giống như nhánh True của điều kiện 4, vì khỉ đã có gậy rồi. Điều khác biệt là khỉ không cần “đi đến C” vì đã đứng sát ở bên ghế rồi.

Cuối cùng, nếu khỉ ban đầu ở trên ghế, đã có gậy, và ghế đã đúng vị trí, thì khỉ chỉ cần lấy chuối và bài toán kết thúc.

* + - 1. Phiên bản cuối cùng của flowchart

Sau đây là **phiên bản cuối cùng** của flowchart, phần mới được bổ sung được bàn ở phần 2.2.5.3 có màu đỏ:

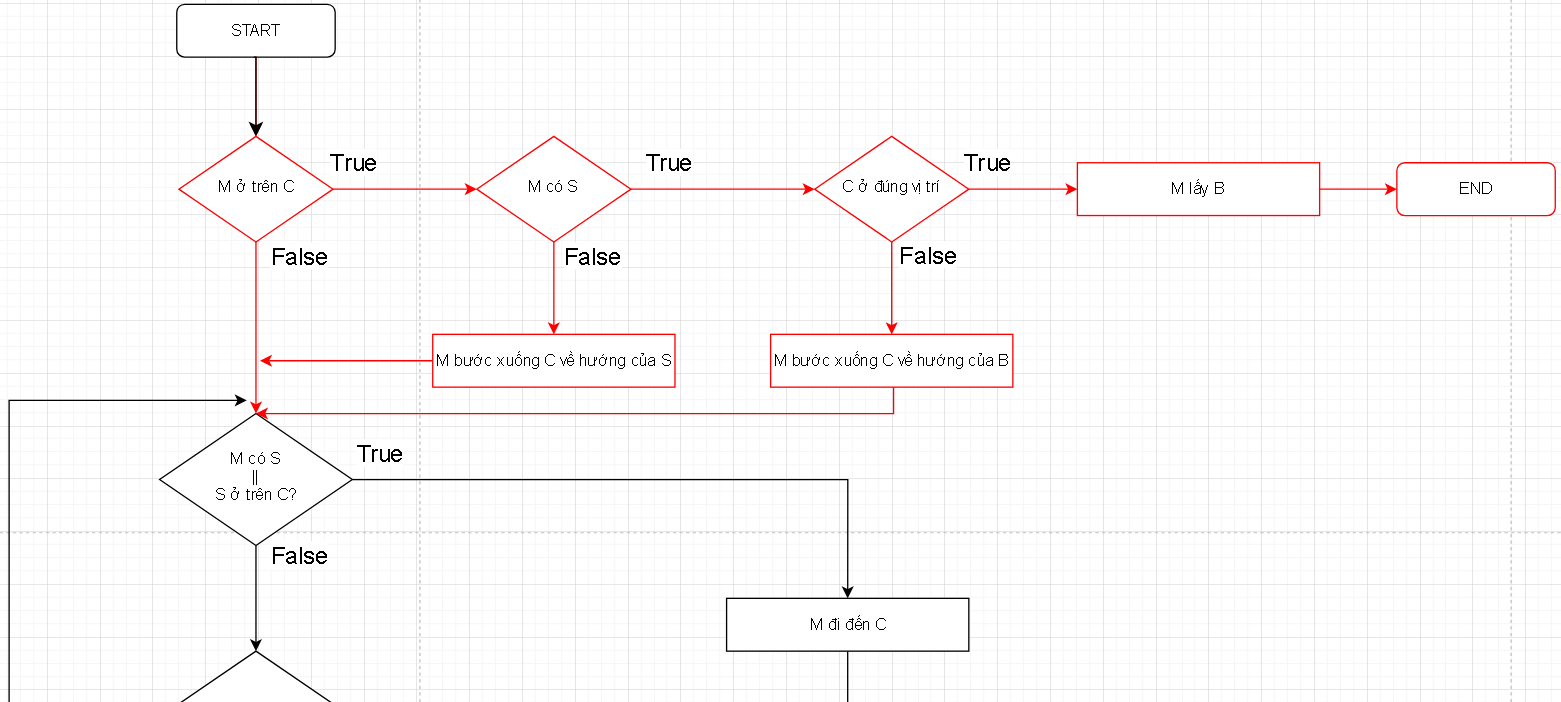
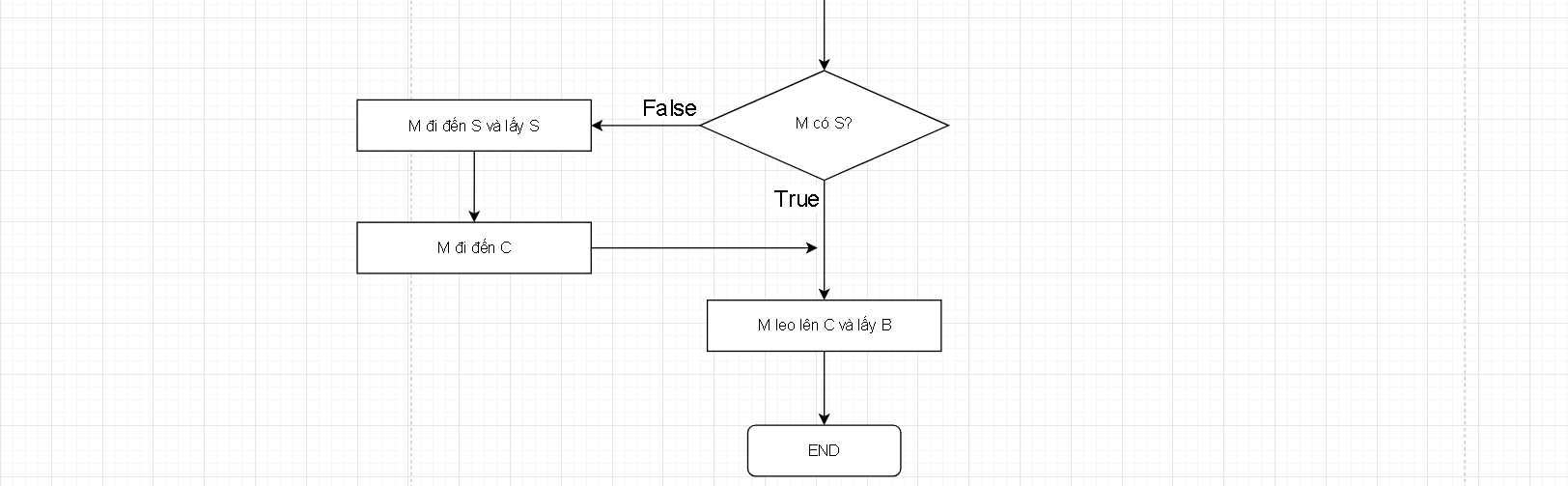
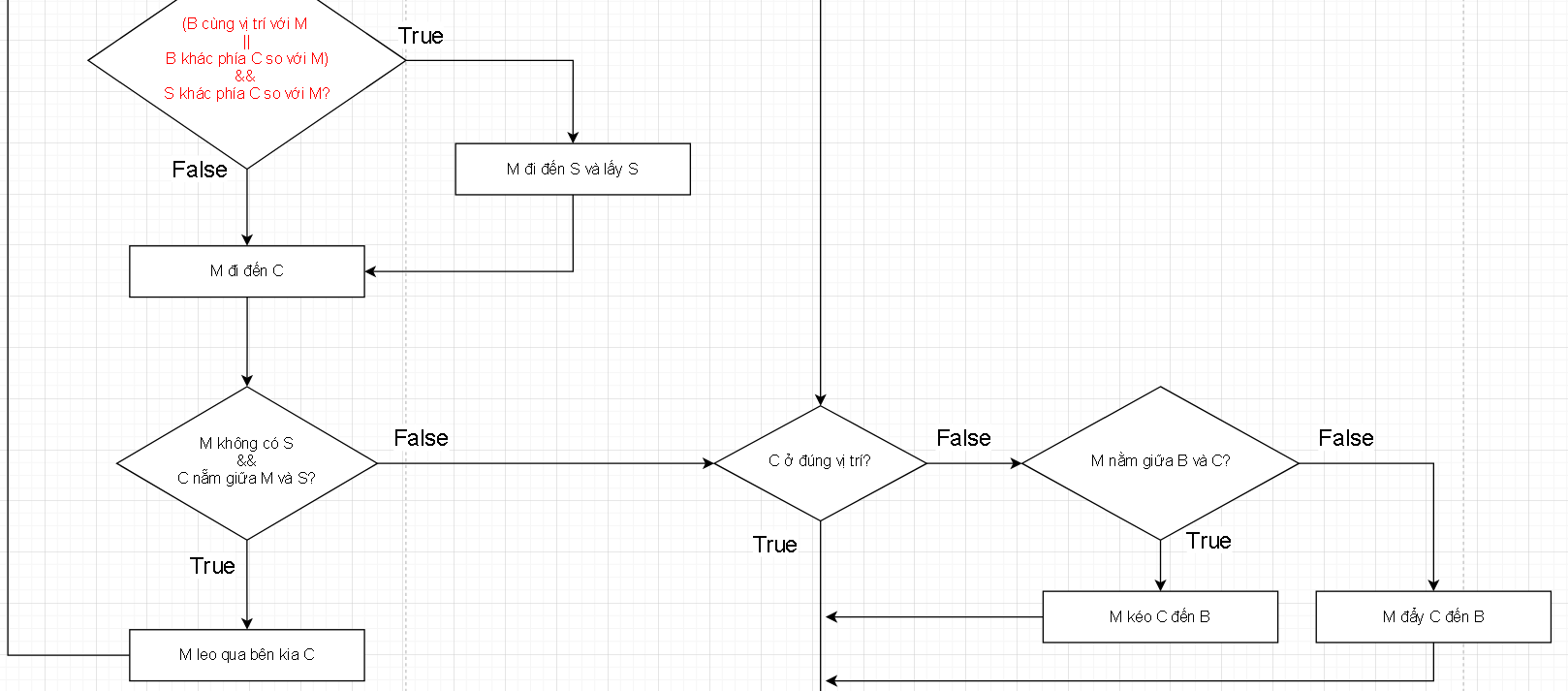
Điều kiện 5

Điều kiện 2

Điều kiện 3

Điều kiện 4

Điều kiện 1



* 1. **Hiện thực bài toán**
     1. **Mô phỏng các vật:**

Mỗi vật có thể được mô phỏng bằng một khối hộp trong một căn phòng 2 chiều. Căn phòng được đại diện bằng một khoảng tọa độ chiều ngang (x) tùy chọn.

S

B

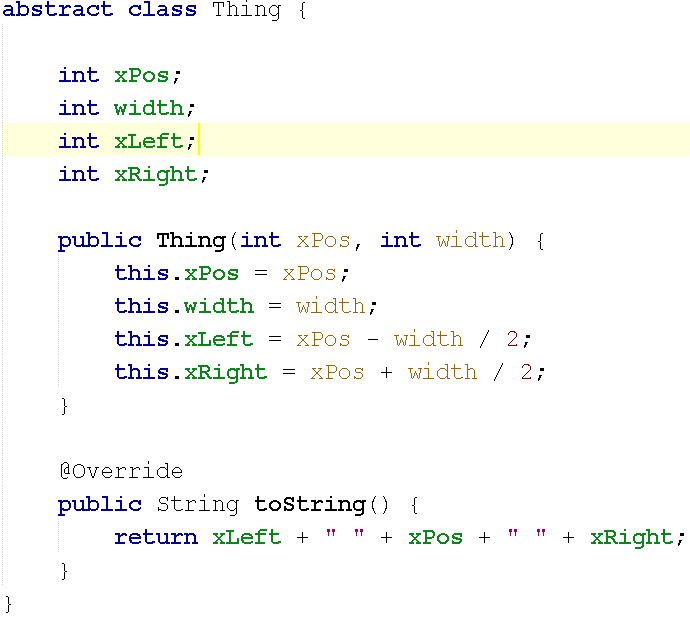
C

M

Mỗi vật đều có 1 tọa độ ở tâm là xPos, 1 tọa độ xLeft, đại diện cho góc dưới bên trái, và 1 tọa độ xRight, đại diện cho góc dưới bên phải. Nếu một vật có xLeft == xRight của vật khác hoặc xRight == xLeft của vật khác, 2 vật này được quy định là “tiếp xúc nhau”. Điều kiện này rất quan trọng, vì giúp ta kiểm tra các sự kiện:

1. Khỉ đã đi đến ghế hay chưa
2. Khỉ đã đi đến gậy và lấy được gậy hay chưa. Nếu gậy và khỉ tiếp xúc nhau, khỉ được xem là đã lấy được gậy
3. Ghế đã được di chuyển đến dưới chuối hay chưa. Ta quy ước điều kiện ghế được ở đúng vị trí khi ghế có xRight == xLeft của chuối hoặc có xLeft == xRight của chuối. Dĩ nhiên, nếu ghế và chuối chiếm cùng một khoảng tọa độ ngang, ghế nằm ở đúng vị trí.

Mỗi vật còn có thêm một tính chất width, là chiều ngang của mỗi vật. xLeft và xRight là từ xPos đi về trái và phải một nửa chiều ngang của vật. Không mất tính tổng quát, ta có thể để cho chiều ngang của chuối (B), ghế (C) và (M) bằng nhau. Chiều ngang của gậy (S) sẽ nhỏ hơn, đơn giản vì tính thẩm mỹ. Ta tạo một abstract class gọi là Thing, và class đại diện cho các vật sẽ kế thừa class này. Class Thing bao gồm một class toString() để in ra tọa độ của các vật.



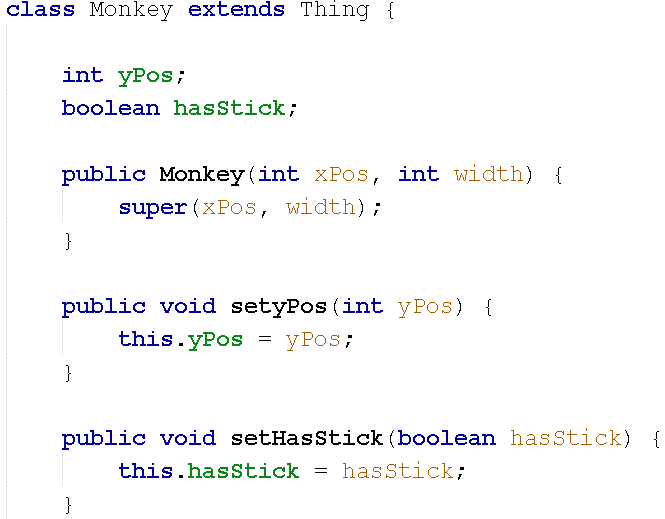
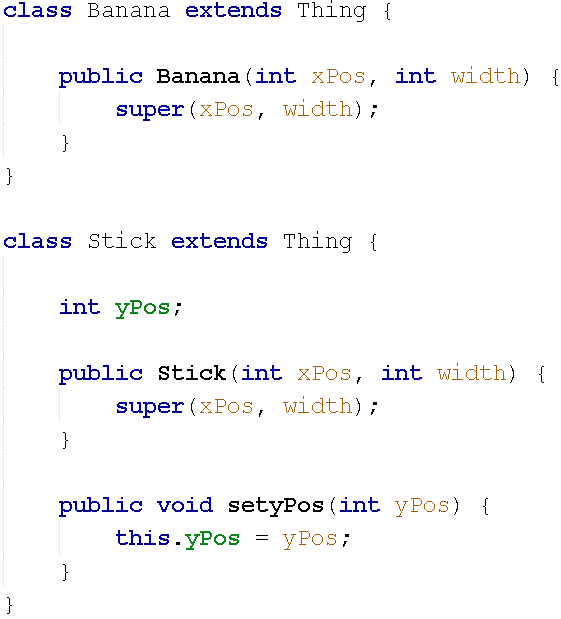
**Các điểm dị biệt của các class**

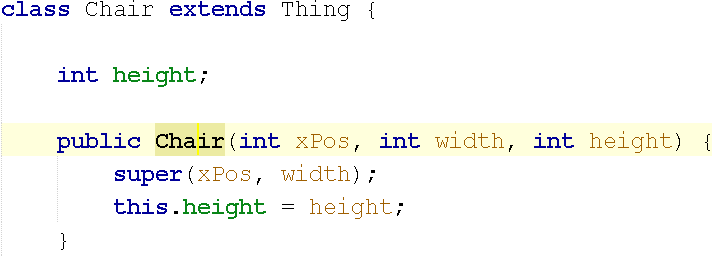
Ghế có thêm đặc tính height, đại diện cho chiều cao của ghế.

Khỉ và gậy có thêm tọa độ yPos, để theo dõi xem có đang ở trên ghế hay không. Giá trị ban đầu của tọa độ này là 0. Nếu trong quá trình code chạy 2 vật này ở trên ghế, ta cần một method setter để thay đổi yPos thành chiều cao của ghế.

Khỉ có thêm một tính chất boolean hasStick để kiểm tra khỉ có gậy hay không. Ta cũng cần một method setter để thay đổi biến này trong quá trình code chạy.

Đến đây, code hiện thực các vật thể như sau:



* + 1. **Mô phỏng các hành động trong class**

Ghế và khỉ có thể di chuyển, ta cần thêm vào method move(int step) để mô phỏng hành động này. Method này nhận một biến int step, là số đơn vị tọa độ thay đổi bởi một bước di chuyển cho trước. Giá trị của step tốt nhất là 1 đơn vị để tránh việc một vật đi lố không gian cho phép.

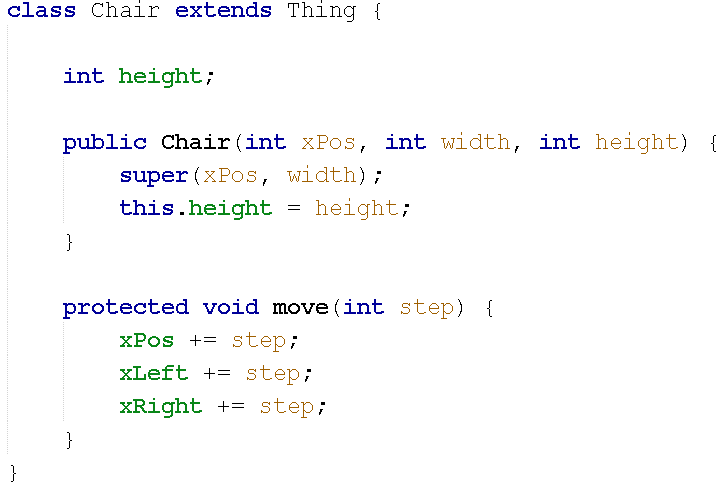
Khỉ có thêm hành động “*Leo lên ghế”* được mô phỏng bởi method climbOnChair(Chair c). Cách hoạt động của method này:

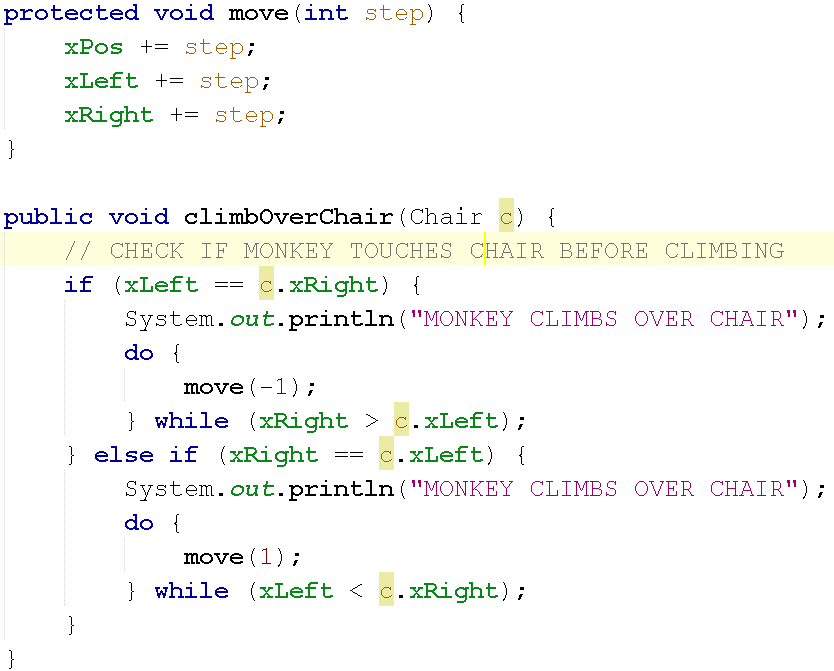
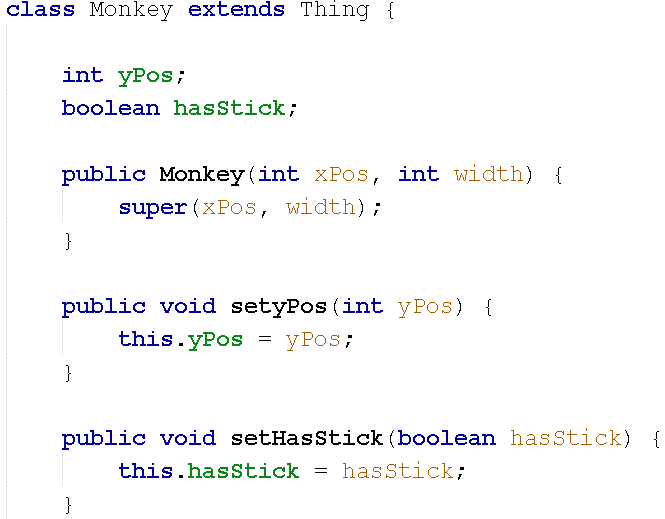
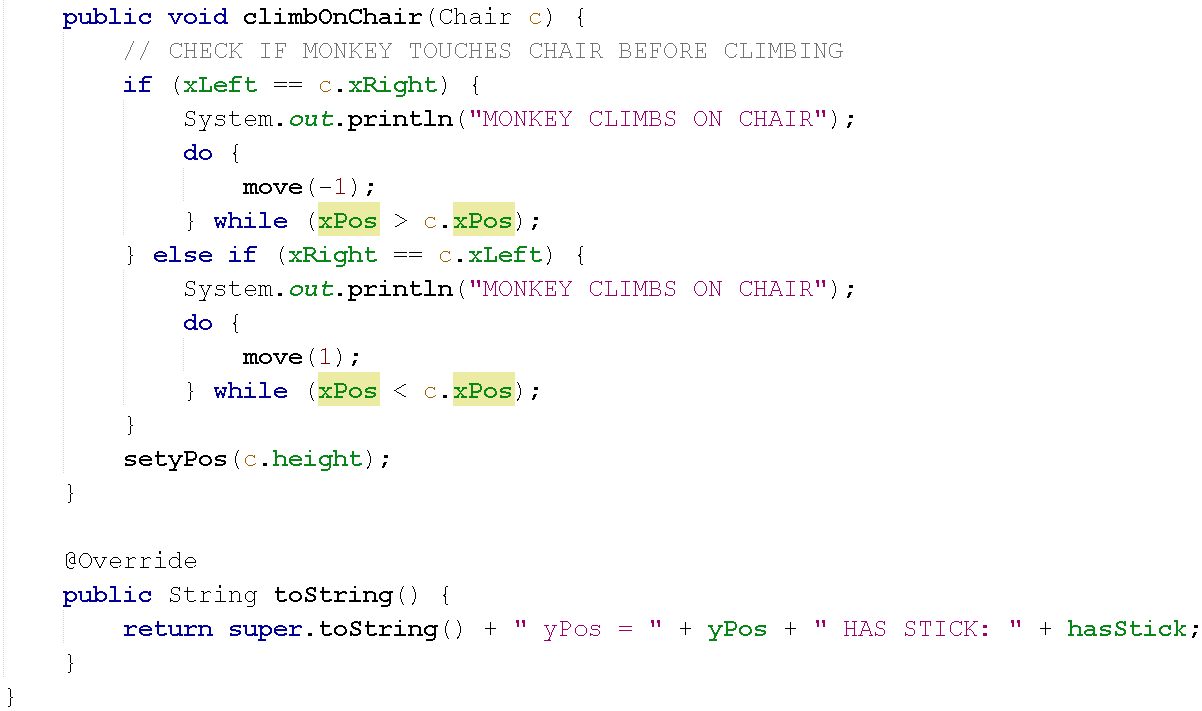
1. Kiểm tra khỉ đã tiếp xúc ghế chưa.
2. Nếu có, khỉ sẽ di chuyển về trái (step = -1) nếu ghế ở bên trái khỉ, hoặc khỉ sẽ di chuyển về bên phải (step = 1) nếu ghế ở bên phải khỉ.
3. Khỉ sẽ dừng di chuyển nếu có xPos == xPos của ghế.
4. Thay đổi tọa độ y của khỉ thành độ cao của ghế.

Ngoài ra, khỉ có hành động “*Leo qua bên kia ghế”*, được mô phỏng bởi method climbOverChair(Chair c). Method này hoạt động như bước 1 và 2 của method mô phỏng “*Leo lên ghế”*, nhưng khác điều kiện kết thúc và không cần thay đổi tọa độ y của khỉ:

1. Kiểm tra khỉ đã tiếp xúc ghế chưa.
2. Nếu có, khỉ sẽ di chuyển về trái (step = -1) nếu ghế ở bên trái khỉ, hoặc khỉ sẽ di chuyển về bên phải (step = 1) nếu ghế ở bên phải khỉ.
3. Khỉ sẽ dừng khi có xLeft == xRight của ghế hoặc xRight == xLeft của ghế.

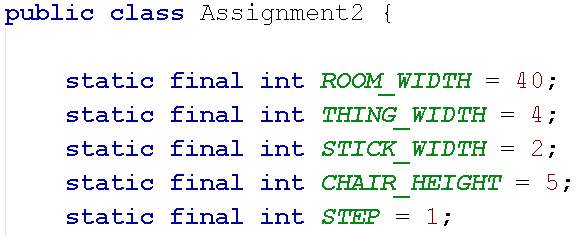
Sau đây là code hoàn tất mô phỏng ghế và khỉ, có thêm các hành động:



* + 1. **Khởi tạo chương trình**

Đầu tiên, chọn các hằng số: chiều ngang căn phòng, chiều ngang của class Thing, chiều ngang của gậy, chiều cao của ghế và độ dài bước đi của vật.

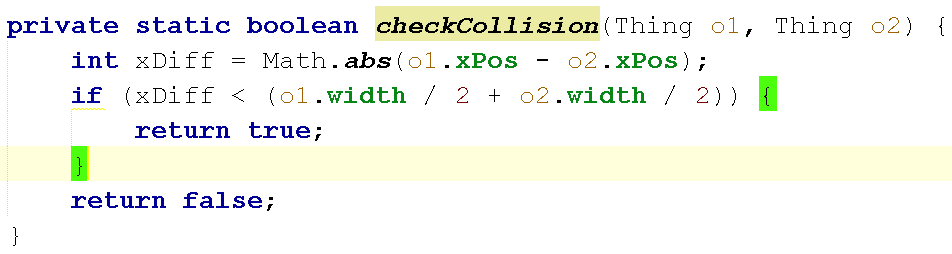


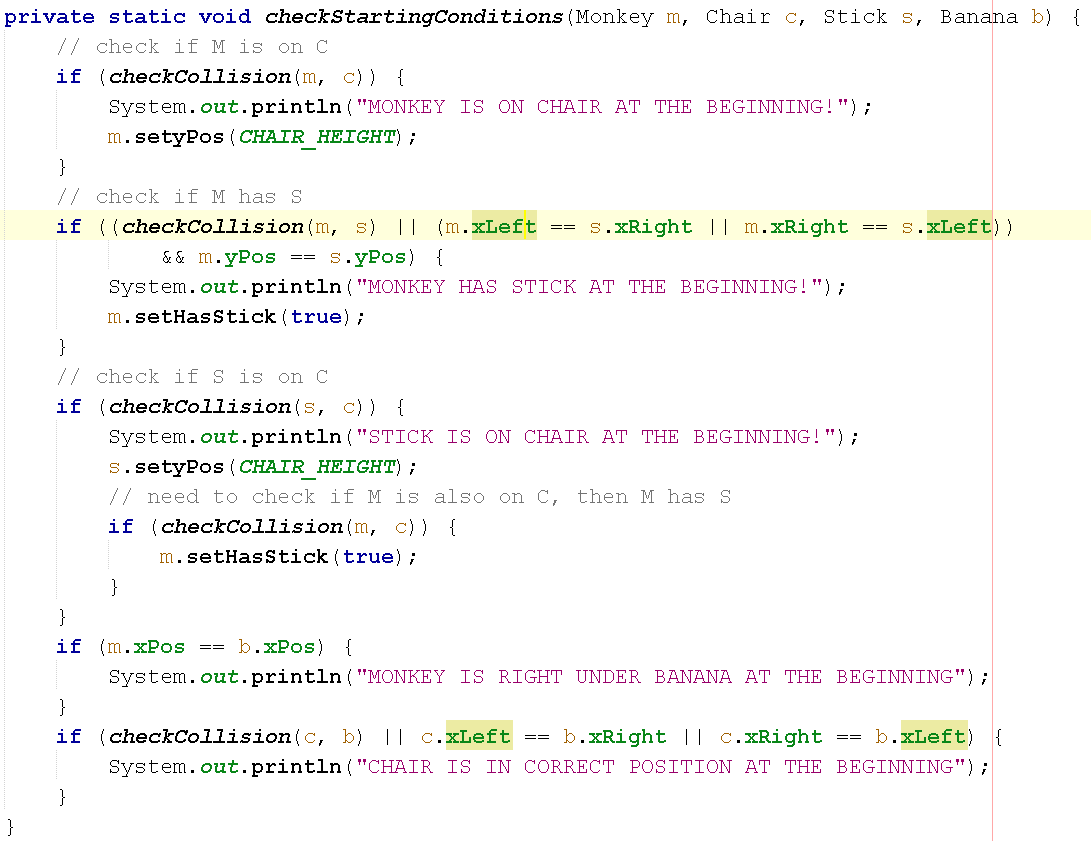
Tọa độ của các vật được khởi tọa ngẫu nhiên với Math.random. Để tránh một vật có chiều ngang THING\_WIDTH bị lố ra bên ngoài không gian cho phép, ta giới hạn miền giá trị ngẫu nhiên thành:

[THING\_WIDTH / 2, ROOM\_WIDTH – THING\_WIDTH / 2]

Sau khi tạo ngẫu nhiên tọa độ các vật, ta cần xét các trường hợp đặc biệt, ví dụ như “khỉ có sẵn gậy ngay từ đầu”, “ghế ở đúng vị trí ngay từ đầu” v.v. Ta viết một method checkStartingConditions() kiểm tra và hiện thực các điều kiện này. Method này dùng một method hỗ trợ checkCollision(Thing o1, Thing o2) để kiểm tra các vật có trùng tọa độ sau khi khởi tạo. Cách hoạt động của method checkCollision(Thing o1, Thing o2):

1. Tìm giá trị xDiff = giá trị tuyệt đối của o1.xPos – o2.xPos
2. Nếu xDiff nhỏ hơn tổng của một nửa chiều ngang o1 và một nửa chiều ngang o2, 2 vật có trùng tọa độ với nhau



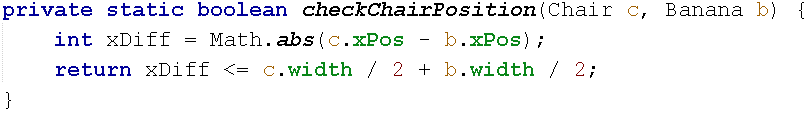


Trong method main, ta gọi method checkStartingConditions(), in ra tọa độ các vật để kiểm tra và thông báo đã khởi tạo xong.Sau đây là code hiện thực hóa việc khởi tạo:



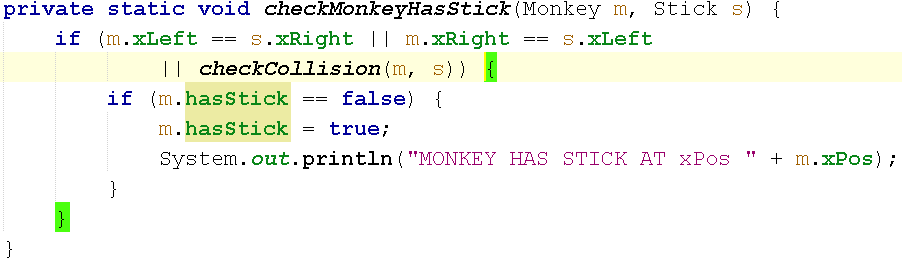
* + 1. **Các method hỗ trợ**
       1. **Method kiểm tra “*Ghế ở đúng vị trí?”(Điều kiện 3)***

Method này hoạt động tương tự như method checkCollision() nhưng bao gồm thêm điều kiện “xdiff == tổng nửa chiều ngang hai vật*”.*



* + - 1. **Method kiểm tra “*Khỉ có gậy?*”**

Method này dùng method checkCollision(Monkey m, Stick s) == true hoặc kiểm tra khỉ và gậy có tiếp nhau hay không. Nếu true mà khỉ chưa có gậy thì đổi tính chất của khỉ thành có gậy.

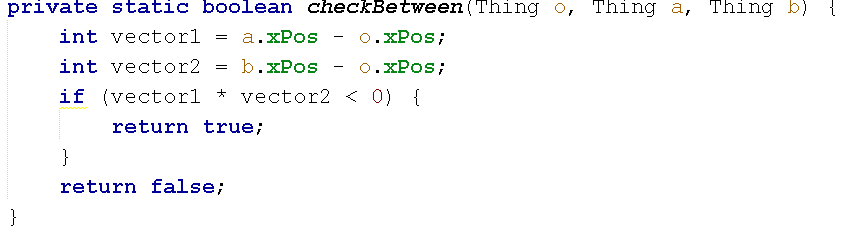


* + - 1. **Method kiểm tra 1 vật ở giữa 2 vật khác**

Method này khá quan trọng vì nó giúp ta kiểm tra điều kiện 1 và vài điều kiện khác trong flowchart. Kiểm tra *“B khác phía C so với M”* tương tự như kiểm tra “*M KHÔNG nằm giữa B và C”.*

Method này kiểm tra một object Thing o có nằm giữa 2 object Thing a và Thing b hay không. Cách hoạt động:

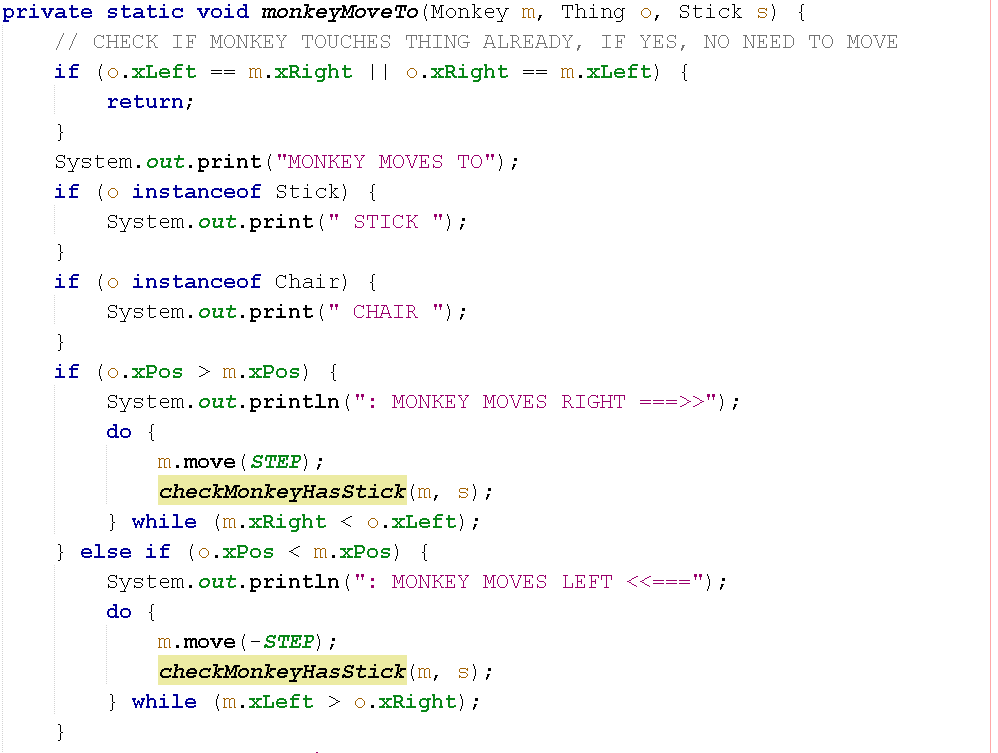
1. Tìm vector o → a, và vector o → b, dùng tọa độ ở tâm vật xPos
2. Nếu tích 2 vector trên là số âm, tức là o nằm giữa a và b

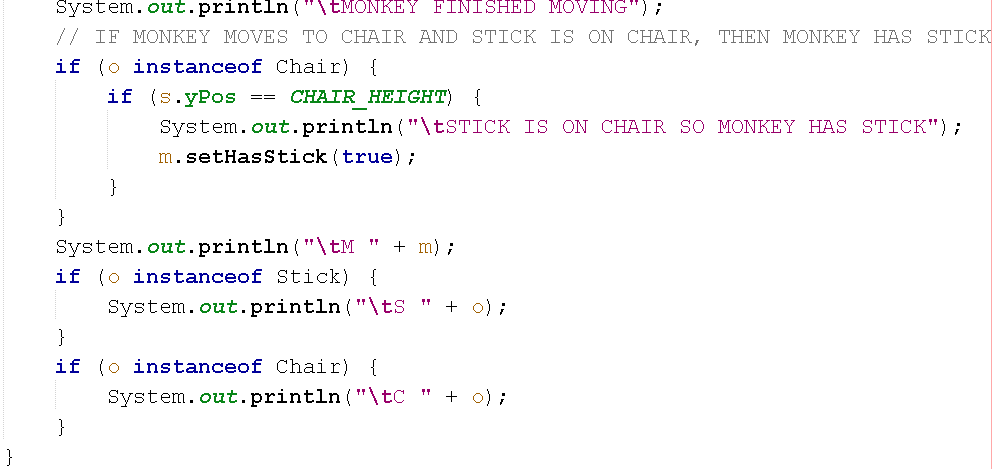


* + 1. **Mô phỏng các sự kiện**
       1. **Mô phỏng sự kiện *“Khỉ đi đến ghế”* và *“Khỉ đi đến gậy”***

Method monkeyMoveTo(Monkey m, Thing o, Stick s) mô phỏng sự kiện khỉ đi đến một object Thing, có thể là gậy hoặc ghế. Cách hoạt động của method này:

1. Kiểm tra khỉ có tiếp xúc vật hay chưa, nếu tiếp xúc thì thoát và không cần thực hiện việc di chuyển.
2. Nếu tọa độ xPos của khỉ nhỏ hơn tọa đọ xPos của vật, khỉ di chuyển về bên trái. Ngược lại, khỉ di chuyển về bên phải.
3. Kiểm tra sau một bước đi khỉ có “vô tình” nhặt được gậy hay không. Tức là gọi method checkMonkeyHasStick() ở phần 2.3.4.2.
4. Khỉ sẽ dừng di chuyển khi tiếp xúc với vật.
5. Trường hợp khỉ đi đến ghế, và gậy ở trên ghế, thay đổi tính chất của khỉ thành có gậy.





* + - 1. **Mô phỏng sự kiện *“Khỉ kéo ghế đến chuối”* và *“Khỉ đẩy ghế đến chuối”***

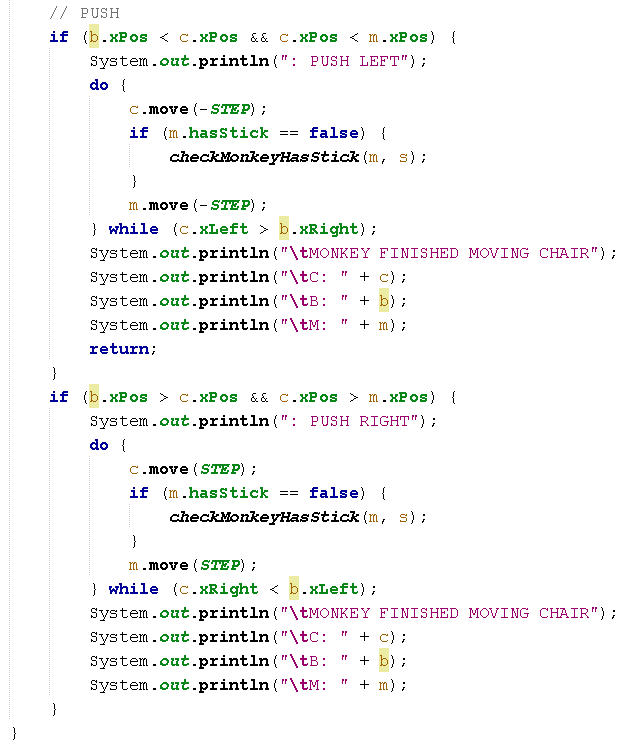
2 sự kiện khỉ kéo hoặc đẩy ghế đến dưới chuối được mô phỏng bởi method monkeyMoveChair(). Cách hoạt động của method:

1. Kiểm tra khỉ có tiếp xúc với ghế trước khi bắt đầu, nếu chưa, cho khỉ đi đến ghế.
2. Nếu khỉ ở giữa ghế và chuối
   1. Nếu chuối ở bên phải khỉ, khỉ và ghế cùng di chuyển về bên phải, thông báo “khỉ kéo ghế về phải”
   2. Nếu chuối ở bên trái khỉ, khỉ và ghế cùng di chuyển về bên

trái, thông báo “khỉ kéo ghế về trái” Nếu ghế ở giữa khỉ và chuối

1. Nếu ghế ở giữa khỉ và chuối
   1. Nếu chuối ở bên phải ghế, khỉ và ghế cùng di chuyển về bên phải, thông báo “khỉ đẩy ghế về phải”
   2. Nếu chuối ỏ bên trái ghế, khỉ và ghế cùng di chuyển về bên phải, thông báo “khỉ đẩy ghế về trái”
2. Sau mỗi bước di chuyển, kiểm tra khỉ có “vô tình” nhặt được gậy hay không, tức là gọi method checkMonkeyHasStick() phần 2.3.4.2.

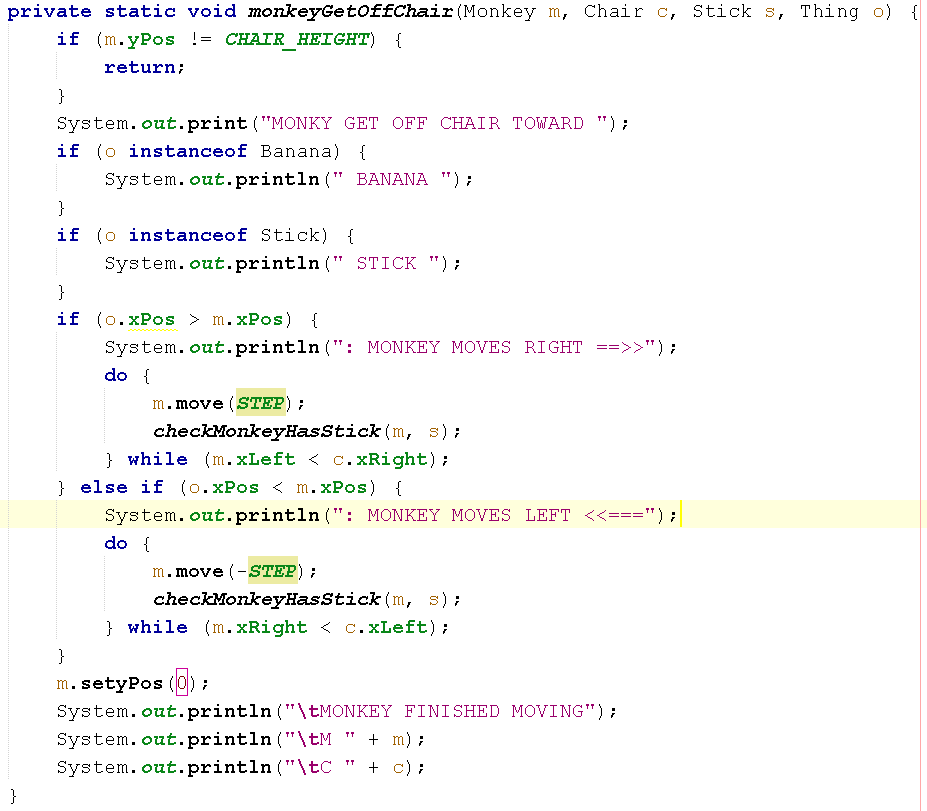




* + - 1. **Mô phỏng sự kiện*“Khỉ bước xuống ghế về hướng chuối hoặc gậy”***

Method monkeyGetOffChair() mô phỏng sự kiện khỉ bước xuống ghế về hướng một object Thing o, có thể là gậy hoặc chuối. Cách hoạt động của method:

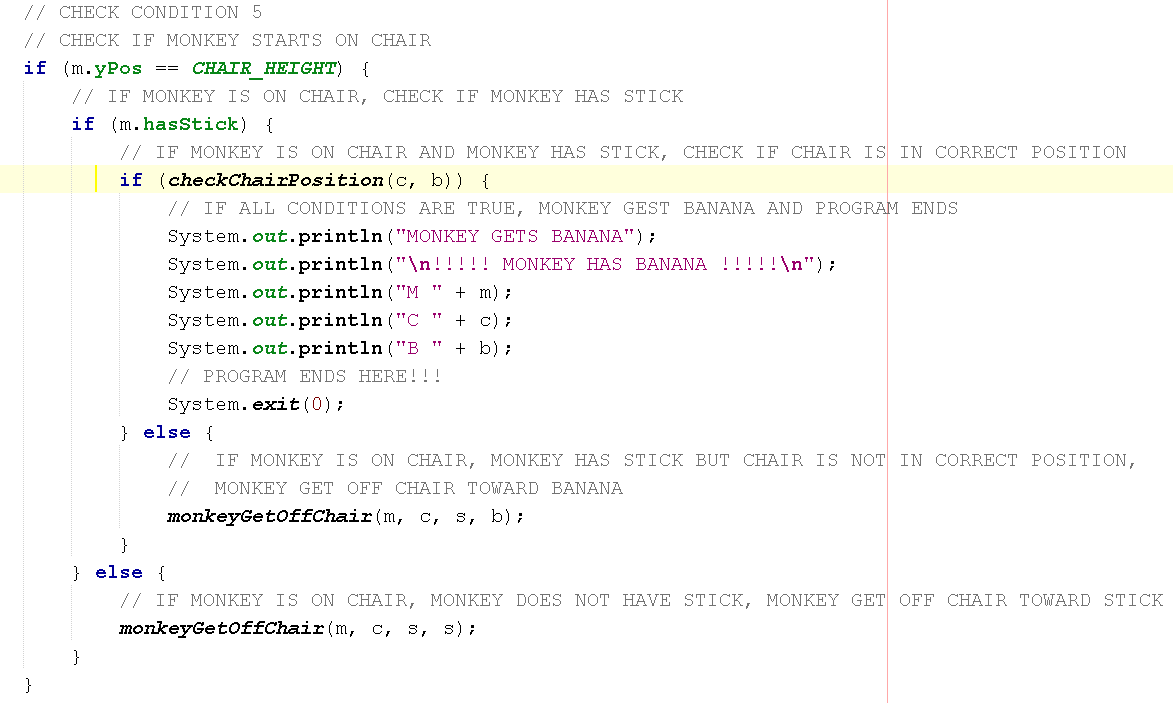
1. Kiểm tra khỉ có ở trên ghế hay không, nếu không thì không tiếp tục.
2. Nếu vật ở bên phải khỉ, khỉ di chuyển về phía phải
3. Nếu vật ở bên trái khỉ, khỉ di chuyển về phía trái
4. Kiểm tra việc khỉ vô tình nhặt được gậy sau mỗi bước đi, tức là gọi method checkMonkeyHasStick() phần 2.3.4.2
5. Khỉ sẽ dừng di chuyển khi có xLeft = xRight của ghế hoặc có xRight == xLeft của ghế.
6. Đổi yPos của khi thành 0.



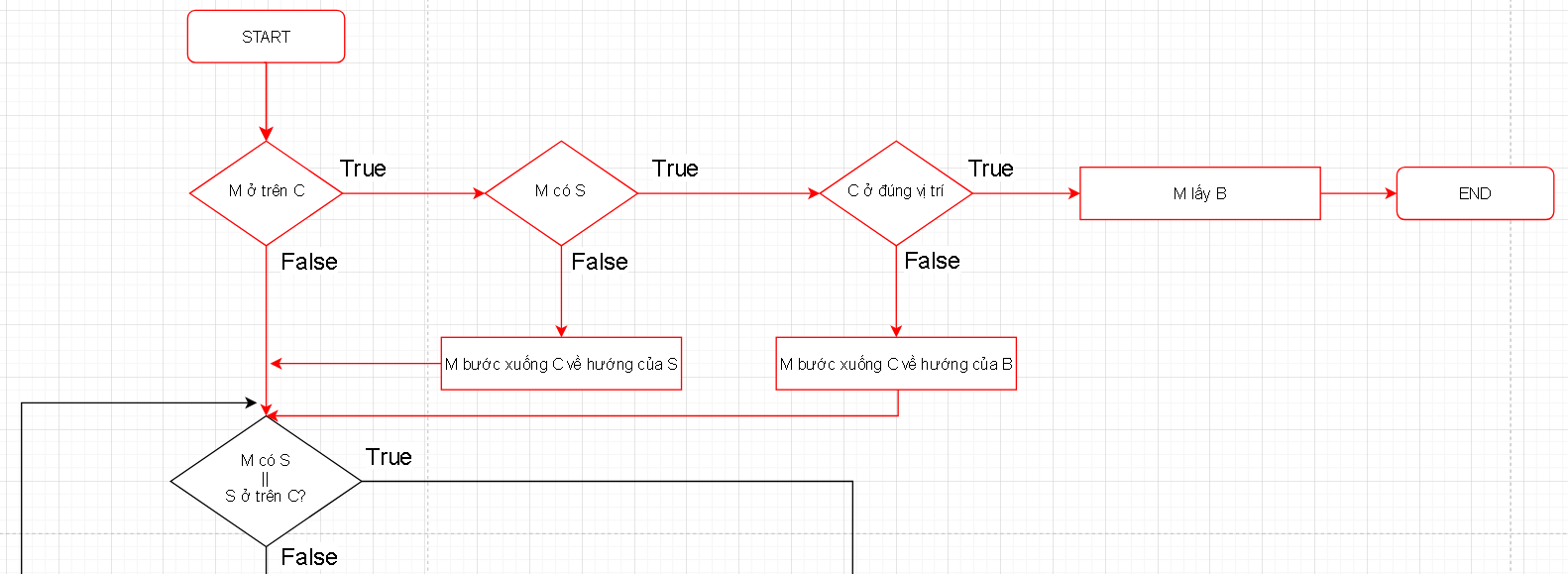
* + 1. **Hiện thực lời giải bài toán**

**Mỗi hình chụp code sẽ hiện thực / liên quan đến phần màu đỏ của hình chụp flowchart đi kèm**

Sau khi khởi tạo, ta kiểm tra điều kiện 5 và hiện thực nhánh True của điều kiện này



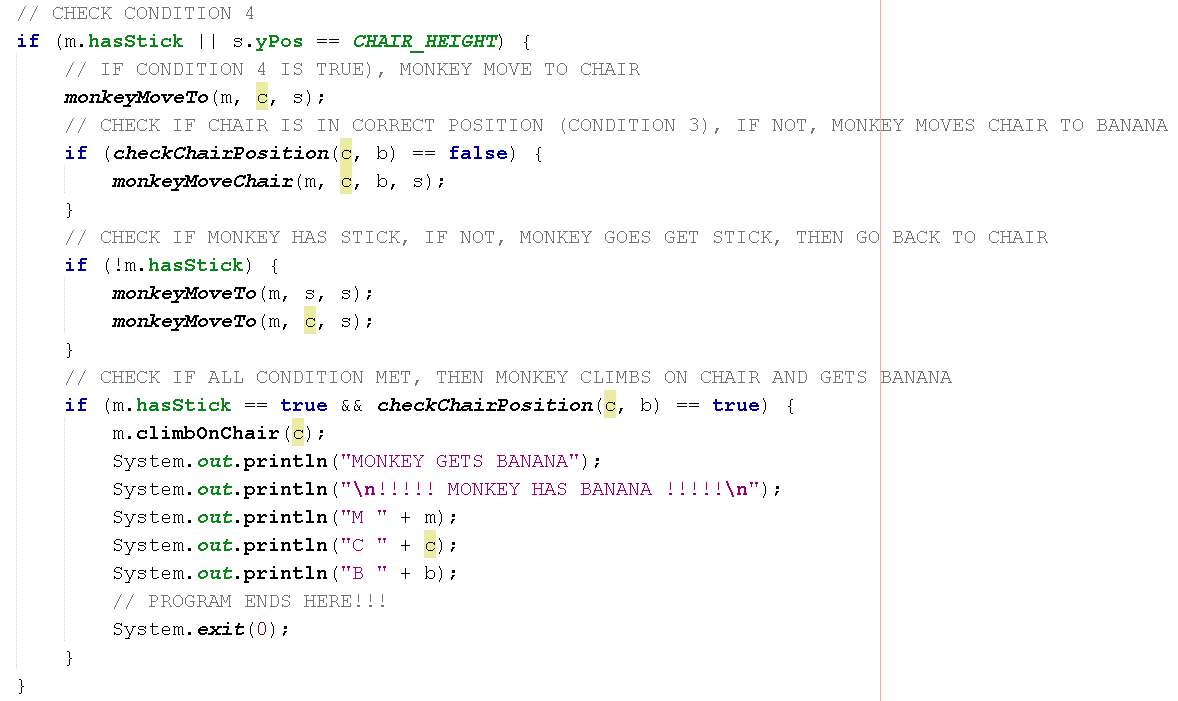
Phần flowchart tương ứng.



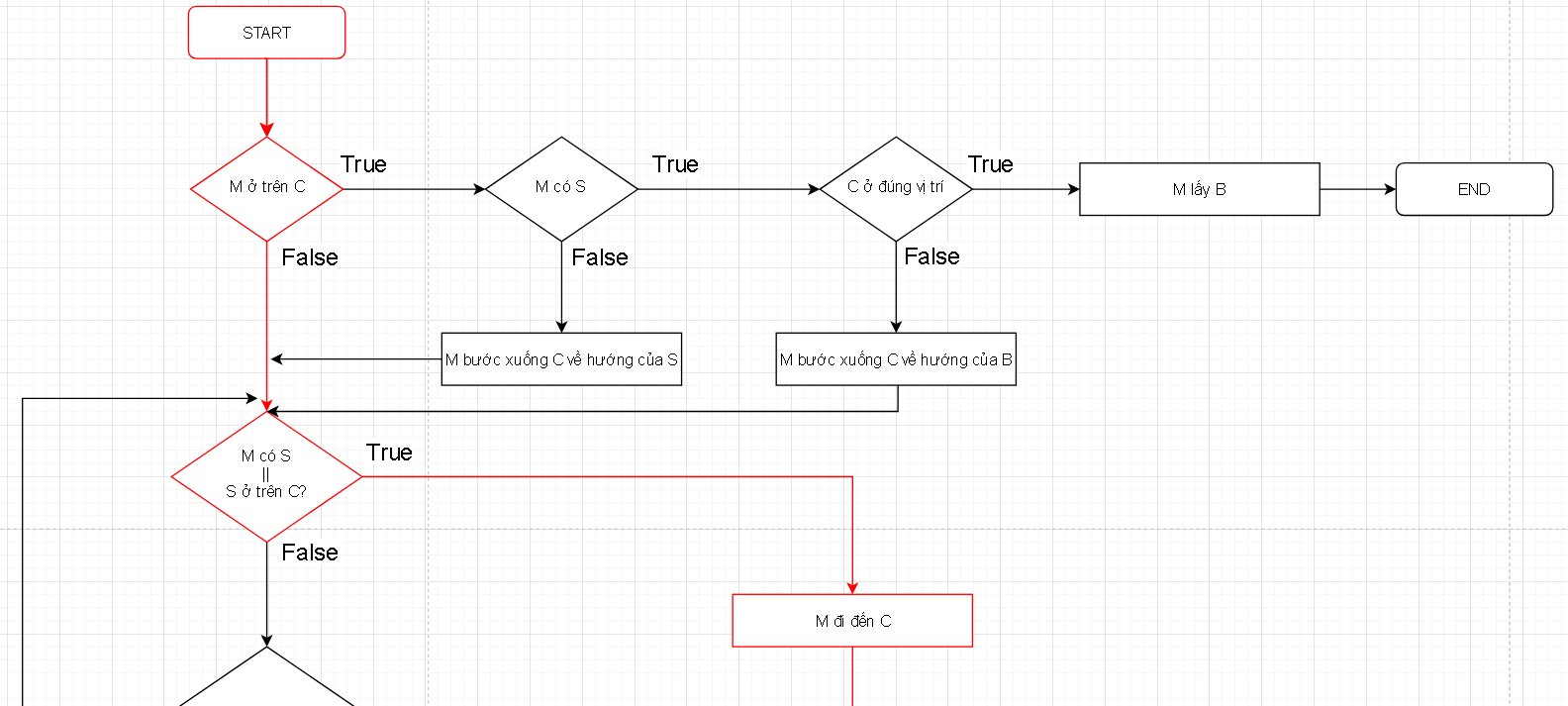
Điều kiện 4

Điều kiện 5

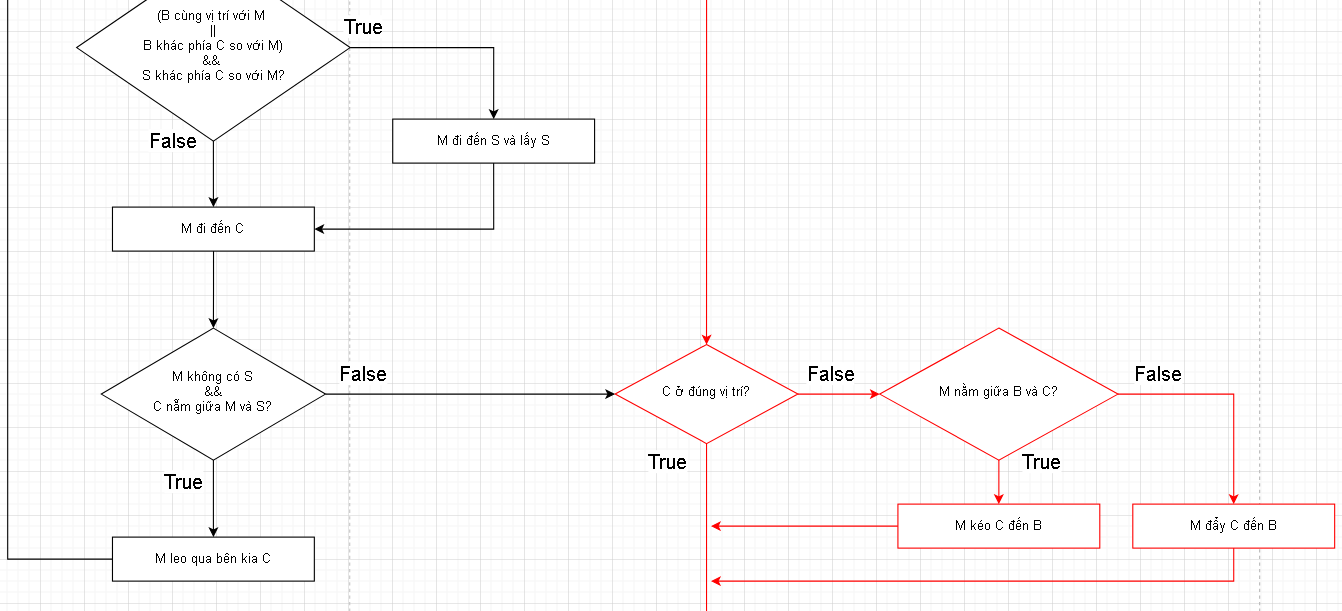
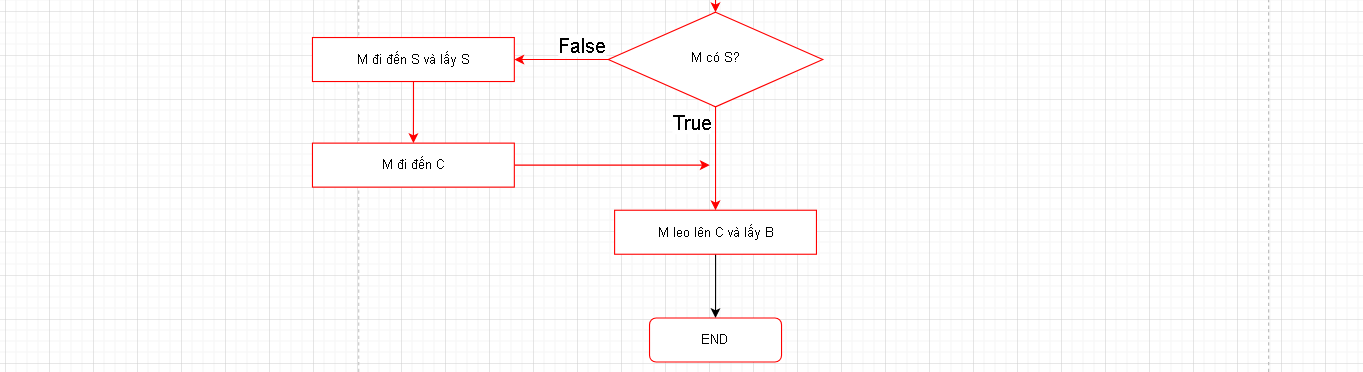
Sau đó, ta kiểm tra điều kiện 4 và hiện thực nhánh True của điều kiện này



Phần flowchart tương ứng:



Điều kiện 5



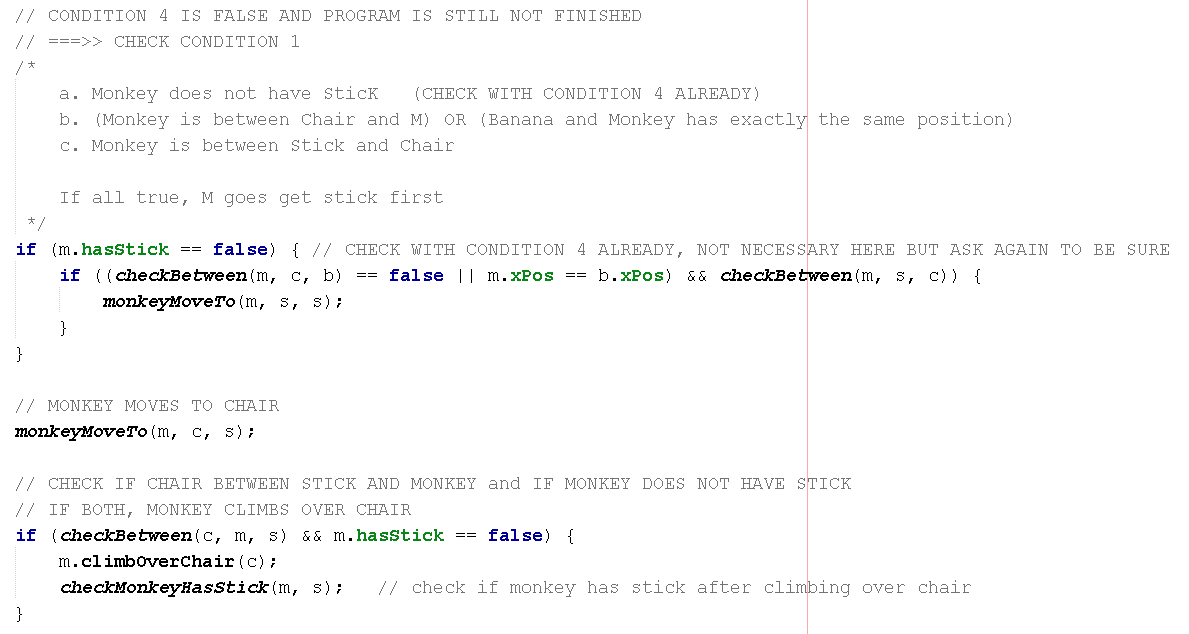
Điều kiện 1

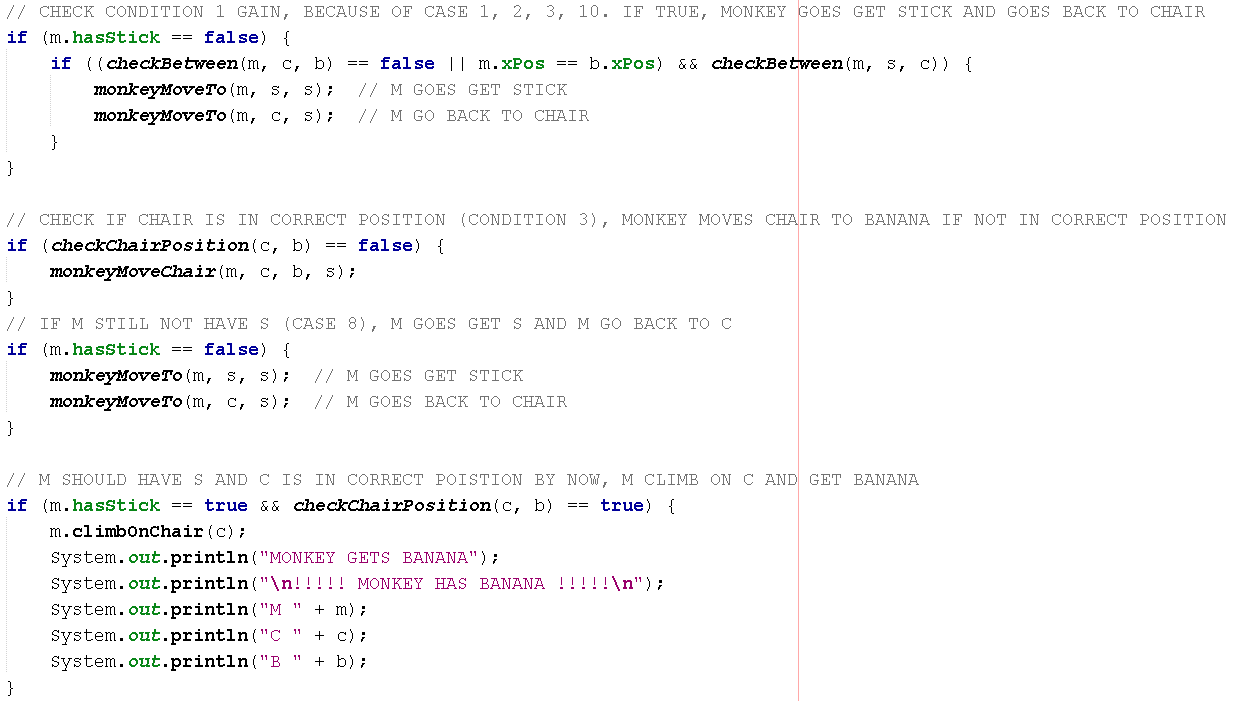
Điều kiện 2

Điều kiện 3

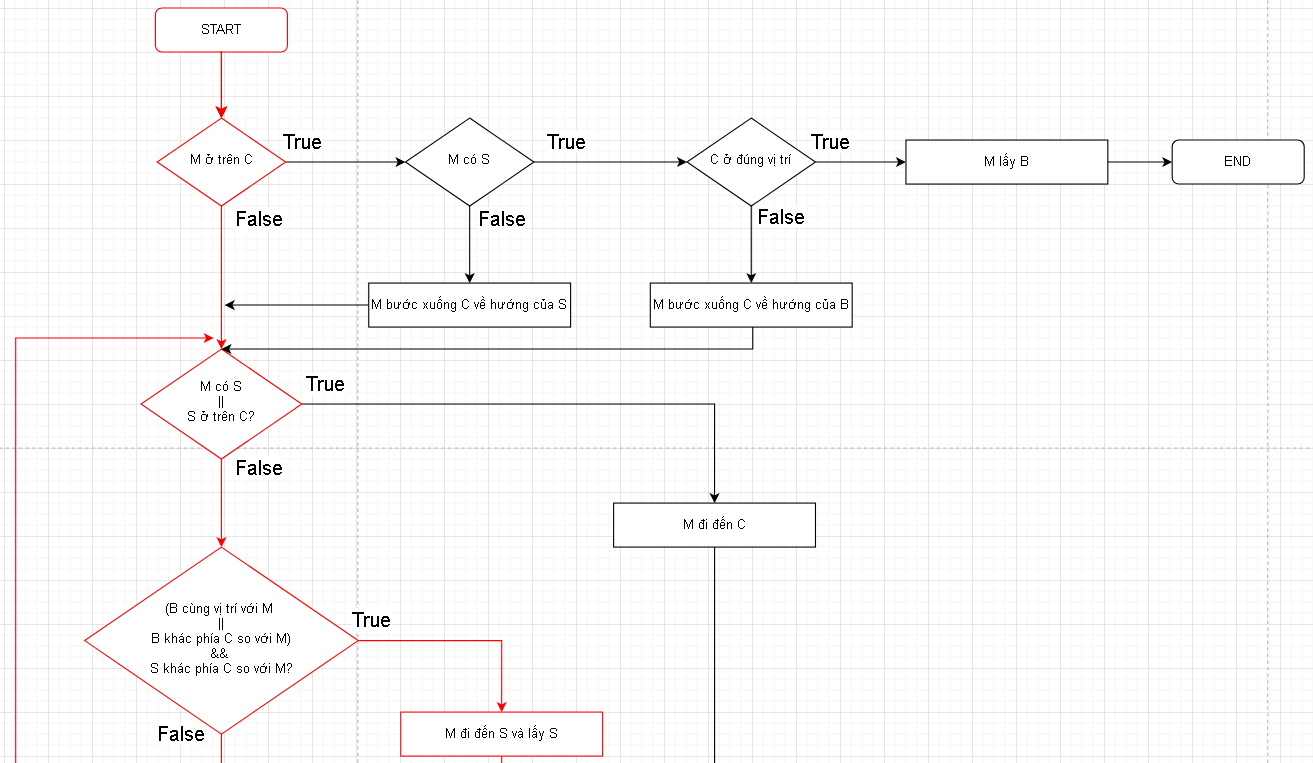
Điều kiện 4

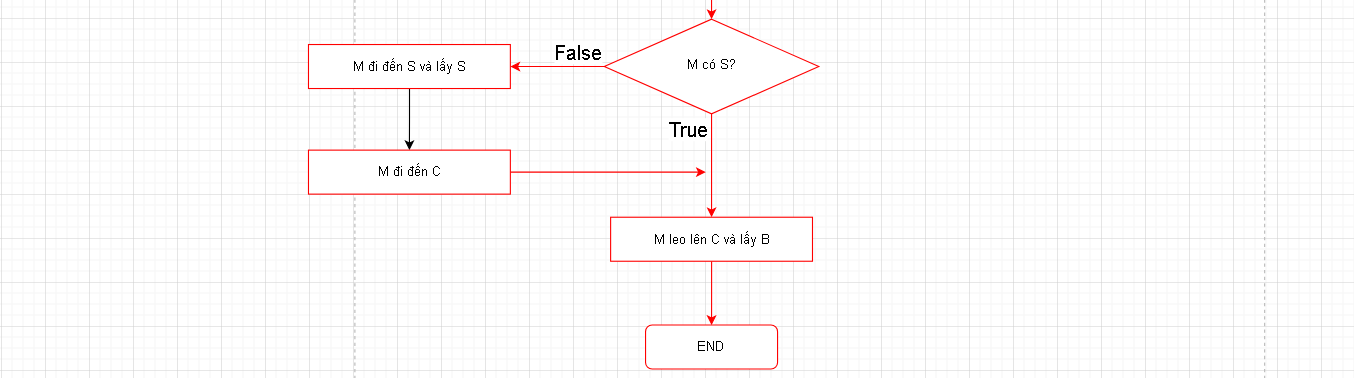
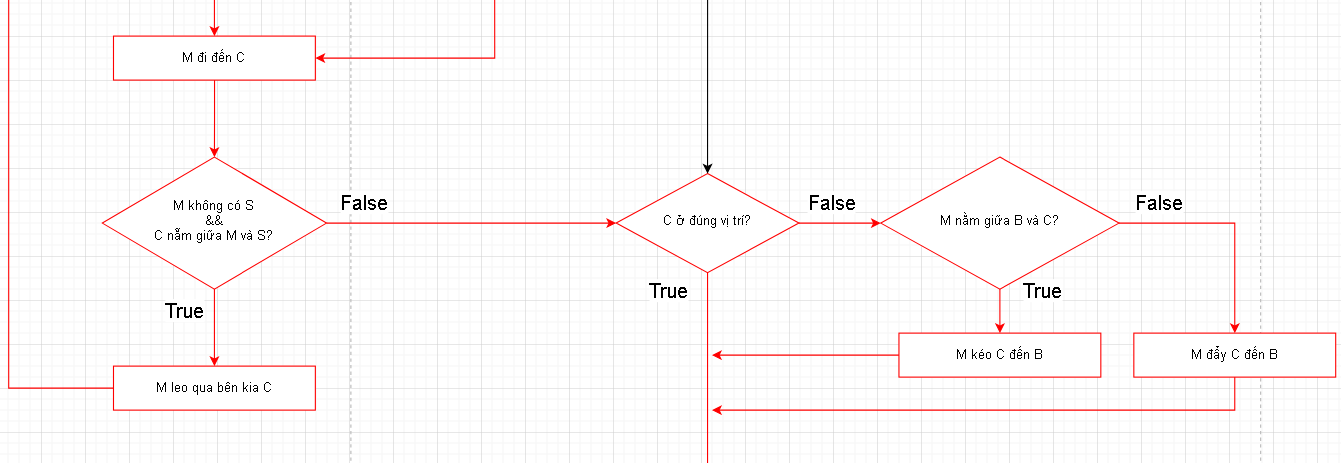
Cuối cùng, ta hiện thực phần flowchart bắt đầu với điều kiện 1.





Phần flowchart tương ứng và cũng là phần cuối cùng cần hiện thực:





Điều kiện 5

Điều kiện 4

Điều kiện 1

Điều kiện 2

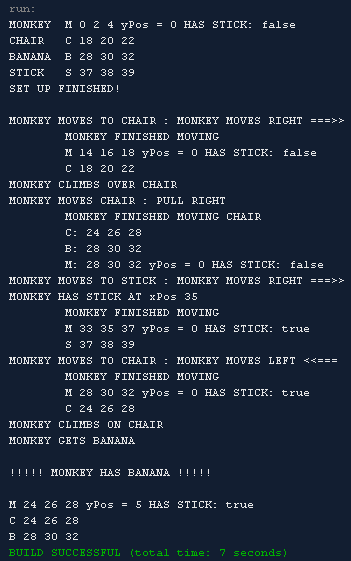
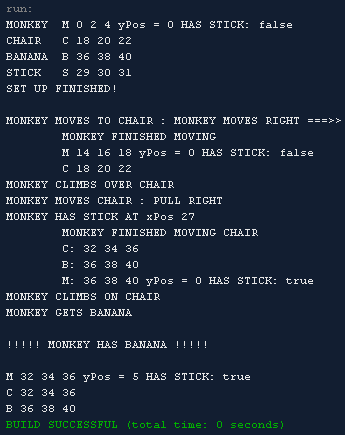
Điều kiện 3

* 1. **Kết quả và thảo luận**
     1. **Kết quả**

Sau đây là kết quả của toàn bộ 12 trường hợp được phân tích ở các phần trước. Các vật sẽ được liệ kê theo thứ tự từ trái qua phải. Tọa độ của mỗi vật sẽ được liệt kê cùng với mỗi vật. Trong code hiện thực, tọa độ được in ra màn hình lần lượt là xLeft – xPos – xRight. Gậy có chiều ngang 2 đơn vị, các vật khác có chiều ngang 4 đơn vị.

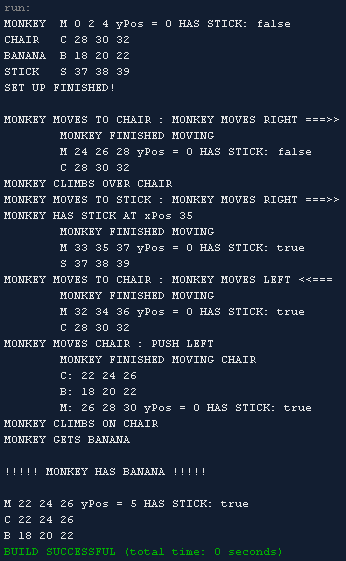
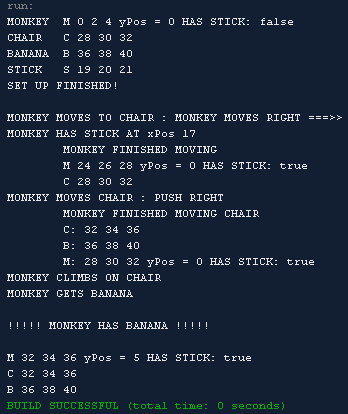
**Trường hợp 1** **Trường hợp 2**

(M, 2) – (C, 20) – (B, 30) – (S, 38) (M, 2) – (C, 20) – (S, 30) – (B, 38)



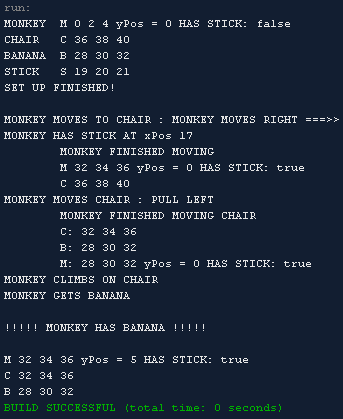
**Trường hợp 3 Trường hợp 4**

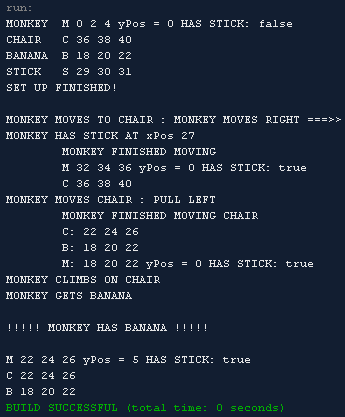
(M, 2) – (B, 20) – (C, 30) – (S, 38) (M, 2) – (S, 20) – (C, 30) – (B, 38)



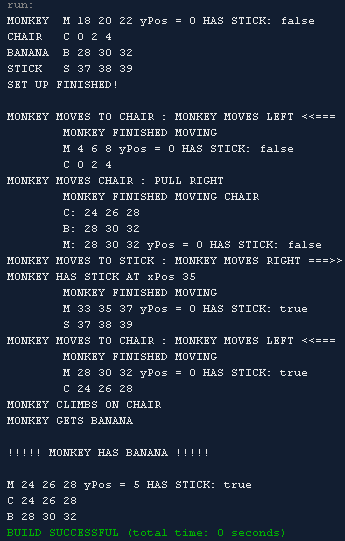
1. **Kết luận**

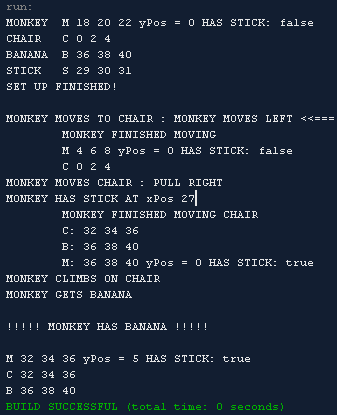
**Trường hợp 5 Trường hợp 6**

(M, 2) – (S, 20) – (B, 30) – (C, 38) (M, 2) – (B, 20) – (S, 30) – (C, 38)

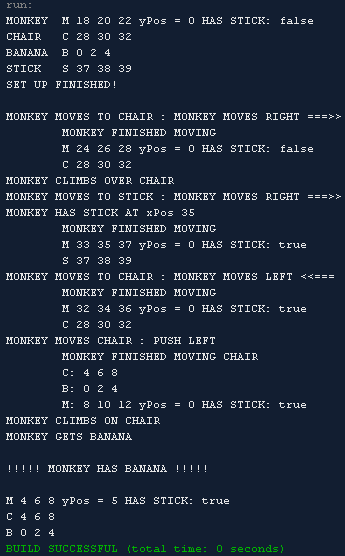
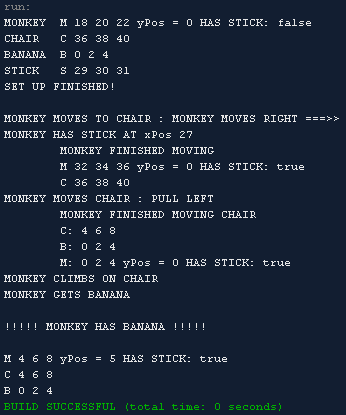


**Trường hợp 7**  **Trường hợp 8**

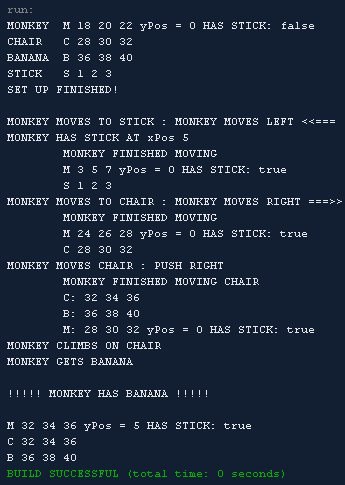
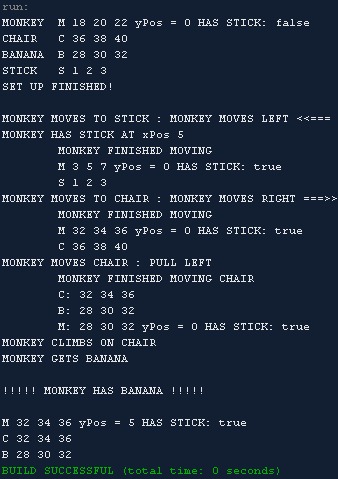
(C, 2) – (M, 20) – (S, 30) – (B, 38) (C, 2) – (M, 20) – (B, 30) – (S, 38)



**Trường hợp 9**  **Trường hợp 10**

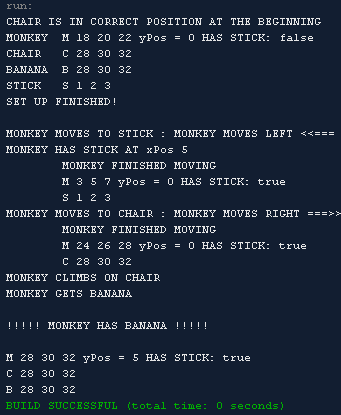
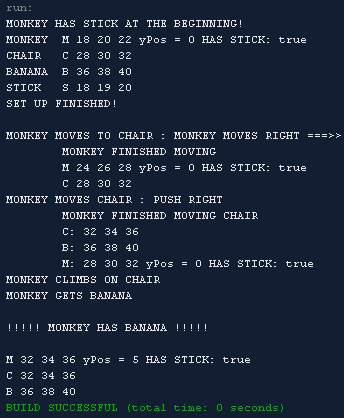
(B, 2) – (M, 20) – (S, 30) – (C, 38) (B, 2) – (M, 20) – (C, 30) – (S, 38)

**Trường hợp 11 Trường hợp 12**

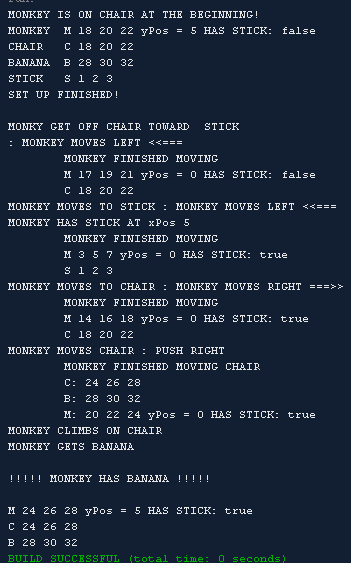
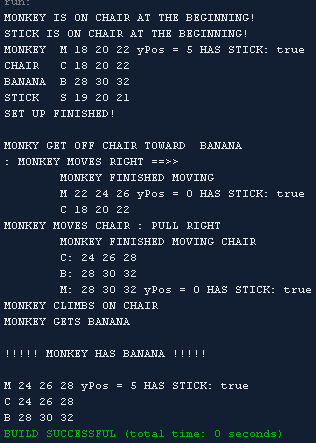
(S, 2) – (M, 20) – (B, 30) – (C, 38) (S, 2) – (M, 20) – (C, 30) – (B, 38)

**Trường hợp khỉ có gậy sẵn Trường hợp ghế ở đúng vị trí sẵn**

(S, 19) – (M, 20) – (B, 30) – (C, 38) (S, 2) – (M, 20) – (B, 30) – (C, 30)

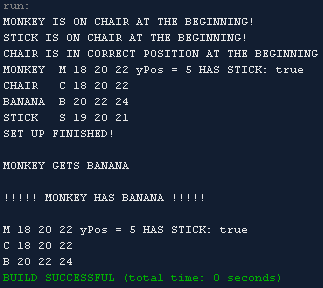


**Khỉ đứng sẵn trên ghế, chưa có gậy Khỉ đứng sẵn trên ghế, có gậy**

(S, 2) – (M, 20) – (C, 20) – (B, 30) (S, 20) – (M, 20) – (C, 20) – (B, 30)

**Trường hợp khỉ đứng sẵn trên ghế, có gậy, ghế ở đúng vị trí**

(S, 20) – (M, 20) – (C, 20) – (B, 22)



Kết quả của code hiện thực đều đúng với chuỗi hành động tốt nhất đã phân tích.

* + 1. **Thảo luận**

Đối với bài toán hiện có, có tổng cộng 4 vật: khỉ, chuối, ghế và gậy. Nếu chỉ xét không gian 2 chiều, có tổng cộng 24 cách sắp xếp vị trí các vật, khi không vật nào cùng tọa độ. Nhờ “tính đối xứng”, ta có thể giảm số trường hợp thành 12. Chuỗi hành động tốt nhất cho mỗi trường hợp đều được liệt kê ở phần 2.1.2. Flowchart được trình bày sau đó là tổng hợp lại các chuỗi hành động tốt nhất cho 12 trường hợp này, và code hiện thực dựa trên flowchart. Chuỗi hành động cho các trường hợp các vật có tọa độ xung đột nhau cũng được liệt kê toàn bộ và thêm vào flowchart.

Việc ta có được flowchart và code hiện thực đều dựa trên việc ta biết trước toàn bộ các chuỗi hành động tốt nhất. Nếu bài toán gồm nhiều vật hơn, tổng số trường hợp là một con số lớn hơn thì việc biết trước toàn bộ các chuỗi hành động tốt nhất có thể không khả thi. Ngoài ra, nếu xét không gian 3 chiều, ta có thể không dùng được “tính đối xứng” để giảm số trường hợp. Các điều kiện liên quan đến vị trí tương đối của vật như *“khỉ nằm giữa ghế và vậy”*, v.v có thể phức tạp hơn hoặc không thể dùng được.

Tóm lại, cách tiếp cận hiện có là đủ để giải bài toán với các điều kiện đã nêu, nhưng cần chỉnh sửa để có thể giải quyết một bài toán khái quát hơn.

1. **Kết luận**

Những điều em đã học được hoặc trải nghiệm được sau khi hoàn tất assignment 2:

1. Cách mô tả chi tiết và có hệ thống quá trình thực hiện đề án liên quan đến kỹ thuật phần mềm và trí tuệ nhân tạo, thông qua việc viết báo cáo.
2. Cách thiết kế flowchart cho một phần mềm hoặc bài toán.
3. Cách mô phỏng các vật thể và hành động trong một chương trình hoặc game.
4. Tính chất iterative trong việc tạo ra một chương trình phần mềm.

Ngoài ra, em có thêm các kinh nghiệm sau:

1. Việc tìm ra giải thuật đúng từ ban đầu quan trọng hơn việc hiện thực bằng code. Nếu đã có code hiện thực nhưng giải thuật sai thì công sức là vô ích và việc chỉnh sửa rất tốn công sức và thời gian.
2. Nên có những deadline rõ ràng cho các quy trình trong một đề án.
3. Cố gắng hết sức nhưng không cầu tòa quá mức.

Sau khi hoàn tất assignment 2, em có thêm hứng thú với ngành trí tuệ nhân tạo và lập trình game.

**TỰ ĐÁNH GIÁ – ĐỀ TÀI NHÓM**

**SE1616 – Vũ Thiên Ân – SE160296**

|  |  |  |
| --- | --- | --- |
| **Nội dung** | **Điểm** | **Ghi chú** |
| Giới thiệu bài toán (0.25 đ) | 0.25 |  |
| Mô tả cấu trúc dữ liệu (1.25 đ) | 1.25 |  |
| Sơ đồ giải thuật (1.5 đ) | 1.5 |  |
| Hiện thực (5 đ) | 4.5 |  |
| Kết quả và thảo luận (0.5 đ) | 0.5 |  |
| Điểm nhóm (0.75 đ) | 0.75 | Toàn bộ đều do em làm |
| Các điều rút ra cho bản thân (0.25 đ) | 0.25 |  |
| Báo cáo đúng theo mẫu (0.5 đ) | 0.5 |  |
| Tổng điểm | 9.5 |  |