

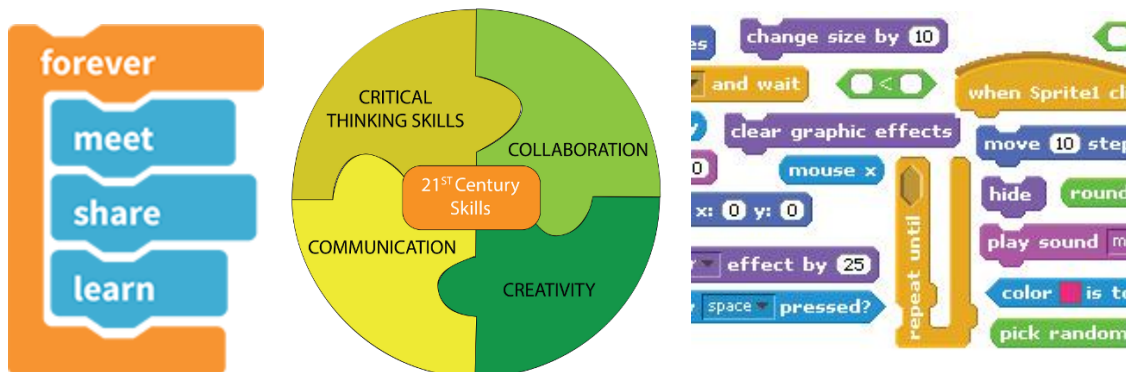
Bùi Việt Hà

Tự học lập trình



We support approaches to coding that engage young people in thinking creatively, reasoning systematically, and working collaboratively -- essential skills for everyone in today's society.

Chúng tôi hỗ trợ một công cụ lập trình mới giúp trẻ suy nghĩ hợp lý hơn, hệ thống hơn, sáng tạo hơn, làm việc nhóm và rèn luyện các kỹ năng cần thiết trong xã hội hôm nay.



Hà Nội 2016

Mục lục

Vì sao Scratch?	6
Mục đích.....	7
Bắt đầu.....	7
Nội dung bài học	7
Câu hỏi và bài tập.....	7
Mở rộng	7
CHƯƠNG 1: LÀM QUEN VỚI SCRATCH.....	8
Bài 1. Tư duy máy tính là gì.....	9
Mục đích.....	9
Bắt đầu.....	9
Nội dung bài học	10
Câu hỏi và bài tập.....	14
Mở rộng	15
Bài 2. Làm quen với Scratch	17
Mục đích.....	17
Bắt đầu.....	17
Nội dung bài học	18
Câu hỏi và bài tập.....	22
Mở rộng	23
CHƯƠNG 2: BẮT ĐẦU LẬP TRÌNH SCRATCH	24
Bài 3. Chuyển động 1	25
Mục đích.....	25
Bắt đầu.....	25
Nội dung bài học	25
Câu hỏi và bài tập.....	37
Mở rộng	38
Bài 4. Vẽ hình 1	39
Mục đích.....	39
Bắt đầu.....	39
Hoạt động bài học.....	39
Câu hỏi và bài tập.....	48
Mở rộng	49
Bài 5. Âm thanh 1.....	50

Mục đích.....	50
Bắt đầu.....	50
Nội dung bài học	50
Câu hỏi và bài tập.....	57
Mở rộng	58
Bài 6. Chuyển động 2.....	60
Mục đích.....	60
Bắt đầu.....	60
Nội dung bài học	60
Câu hỏi và bài tập.....	66
Mở rộng	67
Bài 7. Vẽ hình 2.....	70
Mục đích.....	70
Bắt đầu.....	70
Nội dung bài học	71
Câu hỏi và bài tập.....	76
Mở rộng	76
Bài 8. Âm thanh 2.....	77
Mục đích.....	77
Bắt đầu.....	77
Nội dung bài học	77
Câu hỏi và bài tập.....	87
Mở rộng	88
CHƯƠNG 3: TÌM HIỂU SÂU HƠN SCRATCH	89
Bài 9. Hội thoại	90
Mục đích.....	90
Bắt đầu.....	90
Nội dung bài học	91
Câu hỏi và bài tập.....	98
Mở rộng	98
Bài 10. Hội thoại và truyền thông	100
Mục đích.....	100
Bắt đầu.....	100
Nội dung bài học	100
Câu hỏi và bài tập.....	104
Mở rộng	105

Bài 11. Cảm biến	107
Mục đích.....	107
Bắt đầu.....	107
Nội dung bài học	108
Câu hỏi và bài tập.....	113
Mở rộng	113
CHƯƠNG 4: SCRATCH NÂNG CAO	115
Bài 12. Xử lý số 1.....	116
Mục đích.....	116
Bắt đầu.....	116
Nội dung bài học	116
Câu hỏi và bài tập.....	123
Mở rộng	123
Bài 13. Xử lý số 2.....	125
Mục đích.....	125
Bắt đầu.....	125
Nội dung bài học	125
Câu hỏi và bài tập.....	129
Mở rộng	129
Bài 14. Xử lý xâu ký tự 1	130
Mục đích.....	130
Bắt đầu.....	130
Nội dung bài học	130
Câu hỏi và bài tập.....	137
Mở rộng	137
Bài 15. Xử lý xâu ký tự 2	138
Mục đích.....	138
Bắt đầu.....	138
Nội dung bài học	138
Câu hỏi và bài tập.....	143
Mở rộng	144
Bài 16. Làm việc với List 1	145
Mục đích.....	145
Bắt đầu.....	145
Nội dung bài học	145
Câu hỏi và bài tập.....	155

Mở rộng	156
Bài 17. Làm việc với List 2	157
Mục đích.....	157
Bắt đầu.....	157
Nội dung bài học	157
Câu hỏi và bài tập	170
Mở rộng	170
Bài 18. Thủ tục 1	171
Mục đích.....	171
Bắt đầu.....	171
Nội dung bài học	171
Câu hỏi và bài tập	181
Mở rộng	182
Bài 19. Thủ tục 2	184
Mục đích.....	184
Bắt đầu.....	184
Nội dung bài học	185
Câu hỏi và bài tập	199
Mở rộng	199
Bài 20. Clone. Phân thân của nhân vật.....	202
Mục đích.....	202
Bắt đầu.....	202
Nội dung bài học	202
Câu hỏi và bài tập	208
Mở rộng	208

Vì sao Scratch?

Trên tay các bạn là cuốn sách **Tự học lập trình Scratch**, một môi trường, ngôn ngữ lập trình "kéo thả" rất mới đối với Việt Nam. Vì sao mọi người cần học môi trường lập trình này? Vì sao Scratch lại thích hợp cho lứa tuổi thiếu nhi, thiếu niên và phù hợp cho việc đưa các kiến thức lập trình cho các bậc học này?

Môi trường và ngôn ngữ lập trình Scratch do nhóm nghiên cứu Lifelong Kindergarten Group thuộc đại học MIT (Massachusetts Institute of Technology) thiết lập đầu năm 2008. Ý tưởng ban đầu của nhóm chỉ là thiết lập một ngôn ngữ lập trình mới, đơn giản, chỉ dùng kéo thả, dành cho trẻ con để thiết lập trò chơi, phim hoạt hình, ứng dụng đơn giản, kích thích sự sáng tạo trong môi trường làm việc nhóm của trẻ.

Tuy nhiên Scratch chỉ thực sự bùng nổ từ năm 2014 khi một số quốc gia như Anh, Mỹ đã đổi mới đột phá chương trình giảng dạy môn Tin học trong nhà trường, đưa nội dung kiến thức Khoa học máy tính vào nhà trường ngay từ cấp Tiểu học. Một trong những đề nghị quan trọng nhất của các chương trình này là cần đưa các ngôn ngữ lập trình đơn giản, dạng kéo thả như Scratch vào giảng dạy trong nhà trường ngay từ Tiểu học. Việc điều chỉnh chương trình môn Tin học này đã kéo theo sự gia tăng bùng nổ của Scratch trên phạm vi toàn thế giới. Số lượng học sinh đăng ký tham gia trang Scratch tăng đột biến cả về số lượng và chất lượng. Thực tế đã chứng minh tính hấp dẫn của các môi trường lập trình kéo thả như Scratch, sự đam mê lập trình của trẻ nhỏ. Scratch vô cùng thích hợp cho trẻ lứa tuổi từ 6 đến 14, tức là các cấp Tiểu học, THCS của Việt Nam. Chính vì vậy trong Chương trình đổi mới giáo dục của Việt Nam sau 2018, Bộ Giáo dục & Đào tạo cũng đã quyết định đưa nội dung kiến thức Khoa học máy tính trong môn Tin học vào ngay từ cấp Tiểu học, và những ngôn ngữ lập trình kéo thả như Scratch sẽ là một lựa chọn tốt cho các nhà trường và học sinh.

Scratch là gì?

Tóm tắt một vài ý để trả lời cho câu hỏi: **vậy Scratch là gì?**

- Scratch là 1 môi trường lập trình ứng dụng đặc biệt, trong đó việc “viết” lệnh sẽ được thực hiện bằng thao tác “kéo thả”.
- Đầu ra của Scratch hỗ trợ các công nghệ và ứng dụng mới nhất của CNTT-ICT, do vậy các ứng dụng của Scratch rất phong phú, hấp dẫn, nhất là trẻ nhỏ.
- Scratch có sự phát triển bùng nổ 2 năm trở lại đây. Đặc biệt là sau khi một số quốc gia có tiềm lực khoa học kỹ thuật mạnh trên thế giới đã quyết định đưa Scratch vào giảng dạy trong nhà trường cho học sinh từ cấp Tiểu học.
- Scratch hoàn toàn miễn phí và có thể chia sẻ rộng rãi trong cộng đồng.
- Scratch rất thích hợp để tạo ra các ứng dụng đồ họa, animation, bài học, bài giảng, mô phỏng kiến thức, trình diễn, sách điện tử, trò chơi, ... rất phù hợp với nhà trường, giáo viên, học sinh.
- Scratch là môi trường tốt nhất để dạy học sinh làm quen với tư duy máy tính, khoa học máy tính ngay từ lứa tuổi tiểu học.

Nội dung, đối tượng cuốn sách

Cuốn sách sẽ bao quát tất cả các chủ đề chính của môi trường lập trình Scratch bao gồm: chuyển động, đồ họa, âm thanh, hội thoại, cảm biến, biến nhớ, xử lý số - xử lý ký tự - mảng số, thủ tục và clone. Đối tượng của sách có thể là giáo viên tin học, giáo viên thường, học sinh tất cả các cấp từ Tiểu học, THCS, THPT.

Về định hướng nội dung của cuốn sách này sẽ là một trung dung giữa ứng dụng thuần túy thực tế và kiến thức hàn lâm của khoa học máy tính. Chúng tôi không đi quá sâu vào học thuật sẽ gây nhầm chán, khó hiểu với học sinh, nhưng cũng không sa đà quá nhiều vào các kỹ năng thiết kế trò chơi, phim hoạt hình, ...

Nội dung sách sẽ được chia thành nhiều bài học nhỏ. Mỗi bài học đều có chung 1 cấu trúc nhất định. Người học có thể tự học hoặc học, thực hành dưới sự hướng dẫn của giáo viên.

Cấu trúc 1 bài học sẽ thống nhất sẽ bao gồm các phần sau.

Mục đích

Giới thiệu ngắn mục đích, yêu cầu cần đạt về kiến thức và năng lực của người học.

Bắt đầu

Phần mở đầu của mỗi bài học sẽ thường bắt đầu bằng những câu hỏi, đặt vấn đề liên quan đến bài học để người đọc suy nghĩ và chuẩn bị, trước khi bước vào phần chính thức. Mô hình của phần khởi động này chính là khơi dậy nguồn cảm hứng của người học, giúp người học sẽ luôn chủ động trong quá trình học tập và luyện tập.

Nội dung bài học

Nội dung chính của mỗi bài học sẽ bao gồm một dãy các hoạt động hoặc trải nghiệm dành cho người học. Người học có thể tự học, đọc hoặc làm theo hướng dẫn của giáo viên để thực hiện các hoạt động này. Các hoạt động sẽ dần dần dẫn dắt người học khám phá và từng bước nắm bắt kiến thức, dựa trên đó sẽ hình thành năng lực theo yêu cầu của bài học.

Câu hỏi và bài tập

Các bài tập, bài luyện trắc nghiệm hoặc lập trình giúp người học củng cố kiến thức đã được học và rèn luyện kỹ năng lập trình, tư duy thuật toán và giải quyết vấn đề. Đa số các bài tập là đơn giản. Các bài tập khó sẽ có kèm thêm dấu *.

Mở rộng

Đây là phần thực sự có ý nghĩa vận dụng, mở rộng, tìm tòi trải nghiệm dành cho người học. Đa số các vấn đề, bài tập trong phần này là những bài khó, yêu cầu người học phải suy nghĩ, thử nghiệm, thực hành nhiều hơn. Nếu người học làm được tất cả các bài toán, vấn đề được đặt ra trong phần này thì chứng tỏ đã xuất sắc hoàn thành mục tiêu, mục đích của bài học.

CHƯƠNG 1: LÀM QUEN VỚI SCRATCH

Bài 1. Tư duy máy tính là gì

Mục đích

Học xong bài này, bạn có thể:

- Hiểu được: để giải quyết 1 vấn đề, máy tính sẽ "tư duy", "suy nghĩ" như thế nào.
- Biết được để giải quyết 1 bài toán trên máy tính cần được thực hiện như thế nào, thông qua các ví dụ minh họa cụ thể.

Bắt đầu

1. Hàng ngày chúng ta vẫn được nghe, xem, quan sát các chương trình máy tính được điều khiển bởi con người. Hãy kể ra 1 vài chương trình như vậy mà em biết?

Gợi ý:

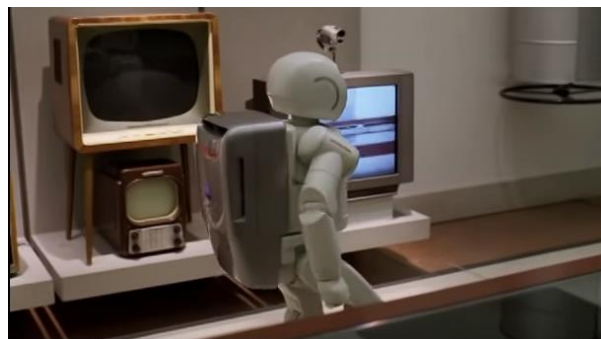
- Dây chuyền tự động lắp ráp máy tính.
- Máy tính bỏ túi tự động tính toán các phép tính.
- Các trò chơi trên máy tính.

2. Em đã nghe được cụm từ "tư duy máy tính" hay "máy tính nghĩ" ở đâu chưa? Hãy nêu ra 1 vài nguồn thông tin mà em biết. Tiếng Anh cụm từ này là "Computer thinking". Em hiểu thế nào về cụm từ này?

3. Quan sát Robot Asimo Nhật bản.



hoặc xem video:



<https://youtu.be/NZngYDDdfW4>

Em hãy trả lời các câu hỏi sau:

- Robot này có suy nghĩ giống người không?
- Robot thực chất hoạt động như 1 máy tính, vậy máy tính có suy nghĩ không? Máy tính có tư duy không?

4. Em hãy quan sát kết quả của 1 chương trình máy tính đơn giản bằng cách mở và chạy chương trình Start.Meo chạy.sb2.

- Mô tả chuyển động của con mèo. Mèo đã thực hiện các công việc gì, theo thứ tự các bước như thế nào?
- So sánh điều em đã mô tả với hình sau:



- Em có rút ra được kết luận gì không?

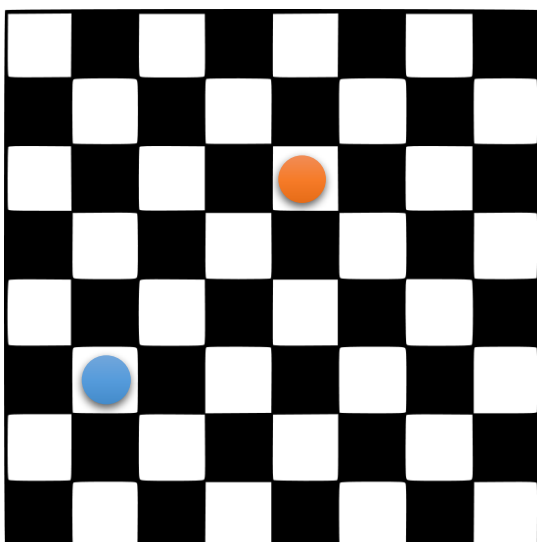
5. Câu hỏi cho tất cả:

- Máy tính sẽ tư duy như thế nào để thực hiện các chương trình, công việc được lên kế hoạch trước?
- Máy tính suy nghĩ và thực hiện công việc như thế nào?
- Vai trò của con người như thế nào đối với máy tính?

Nội dung bài học

1. Quân cờ trên bàn cờ vua

Em hãy nhìn lên bàn cờ ô vuông sau. các quân cờ trên bàn cờ này mỗi lần chỉ đi được sang ô bên cạnh theo hàng ngang hoặc hàng dọc. Câu hỏi: quân cờ màu xanh cần phải đi bao nhiêu bước và đi như thế nào để đến được vị trí quân cờ màu đỏ?



Từ vị trí quân cờ màu xanh đến vị trí màu đỏ có nhiều cách đi khác nhau, nhưng cần chỉ ra các cách đi với ít bước nhất.

Trả lời: Đi bằng 6 bước và có thể đi theo nhiều cách để đến đích.

Trên thực tế tất cả các bài toán cần giải, các vấn đề cần giải quyết cũng phải được thực hiện theo từng bước.

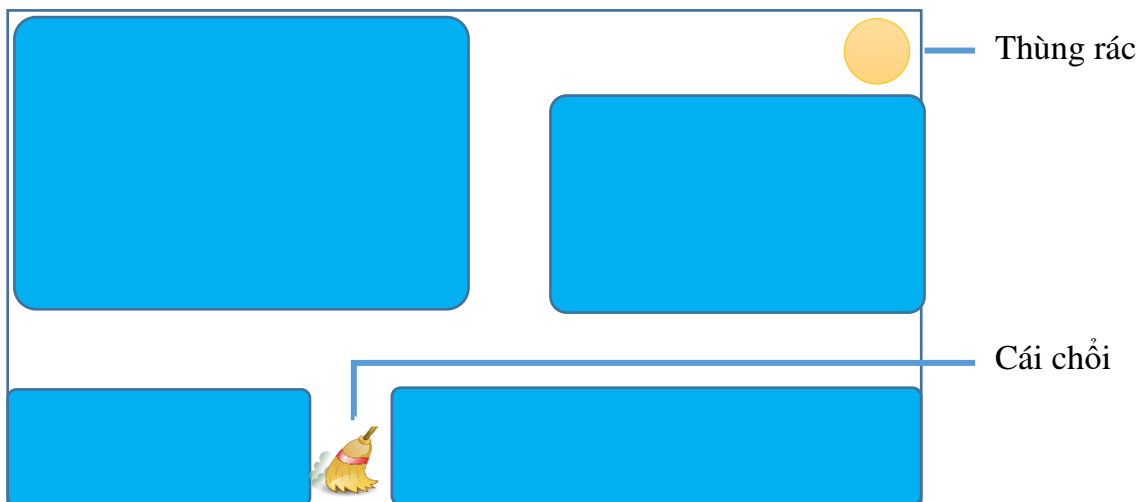
Ghi nhớ: máy tính thực hiện công việc theo từng bước.

Em có thể tìm được rất nhiều các ví dụ khác về việc giải quyết vấn đề theo các bước trên thực tế. Hãy cùng xem các ví dụ sau, em cần liệt kê các bước cụ thể để thực hiện các công việc đó.

- Hàng ngày đi từ nhà đến trường, em phải đi qua các đường, phố nào, hãy liệt kê lần lượt các đường, phố đó.
- Các bước để thực hiện việc nấu cơm.
- Giải 1 bài toán theo các bước, ví dụ bài toán khai triển 1 số thành tích các số nguyên tố, bài toán giải phương trình bậc nhất, bài toán chứng minh một đẳng thức, bất đẳng thức.

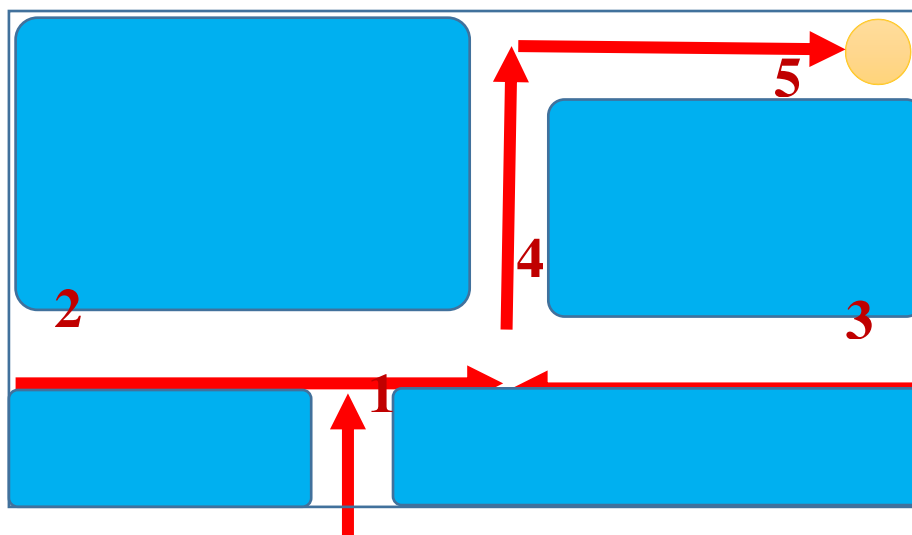
2. Quét nhà

Em quan sát căn phòng. Nhiệm vụ của em là cần quét rác và đưa rác đến góc nhà, vị trí có thùng rác. Em chỉ được dùng chổi quét theo các hướng ngang và dọc theo các lối đi trong phòng.



Hình ảnh dưới đây chỉ ra các công việc phải thực hiện đồng thời là thứ tự của chúng để hoàn thành công việc được giao trên.

Sơ đồ thực hiện là: 1 - 2 - 3 - 4 - 5.



Các em quan sát và trả lời các câu hỏi sau:

- Nếu thay đổi thứ tự các bước là: 3 - 1 - 2 - 4 - 5 thì công việc có hoàn thành không?
- Nếu chỉ thay đổi bước 1-2 thành: 2 - 1 - 3 - 4 - 5 thì kết quả công việc ra sao?
- Các bước 4, 5 có thể đổi chỗ cho nhau được không?

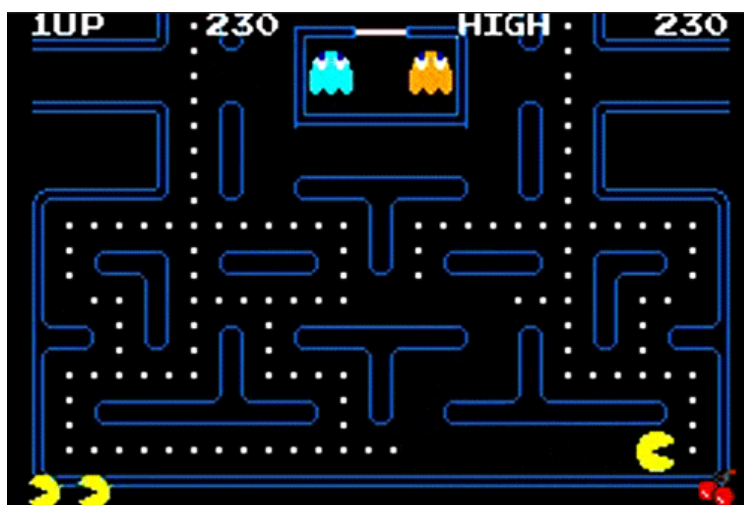
Kết luận

- Thông thường người ta thường giải quyết các bài toán thực tế thông qua các bước.
- Thứ tự các bước là quan trọng để hoàn thành nhiệm vụ.

3. Quan sát một chương trình máy tính mô phỏng giải quyết vấn đề

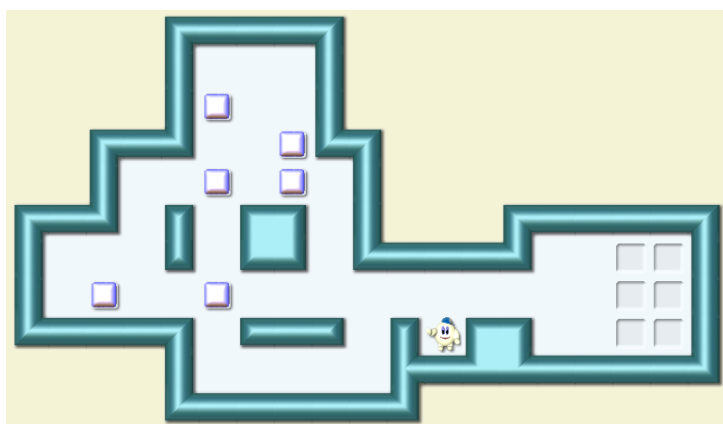
Có rất nhiều phần mềm hoặc trò chơi mô phỏng giải quyết vấn đề thông qua các bước.

- Trò chơi Pacman nổi tiếng một thời.



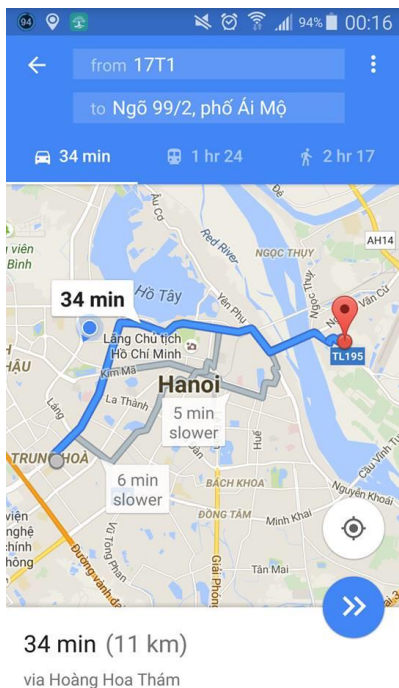
Trong trò chơi này, Pacman là người đào vàng sẽ có nhiệm vụ tìm vàng dọc theo các con đường trong 1 mê cung. Tuy nhiên trên đường đi thường xuất hiện các con quái sẽ tìm cách giết hại người đào vàng. Người đào vàng phải có nhiệm vụ tránh các con quái đồng thời đào được càng nhiều vàng càng tốt.

- Phần mềm trò chơi xếp đồ Socoban



Một bác công nhân có nhiệm vụ chuyển các thùng trong vườn đến vị trí đã được đặt trước. Mỗi lần bác công nhân chỉ có thể đẩy được 1 thùng đi 1 bước. Đường đi trong vườn rất khó vì có nhiều bức tường và nhiều thùng. Bác công nhân phải tìm ra được 1 cách dịch chuyển tối ưu nhất để hoàn thành nhiệm vụ.

- Tìm đường trên điện thoại



Bài toán tìm đường đi (tối ưu) giữa hai vị trí trong thành phố. Cho trước hai vị trí Đầu và Cuối, máy tính sẽ phải dựa trên bản đồ giao thông thành phố để tìm ra được 1 cách đi với khoảng cách hoặc thời gian tối ưu nhất.

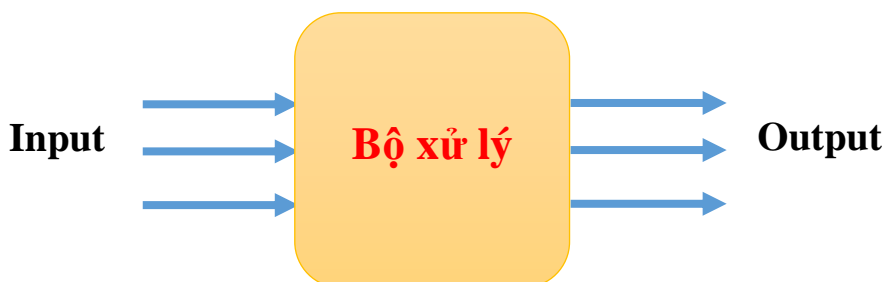
Trên màn hình máy tính sẽ hiện đường đi tối ưu đó với các thông tin có liên quan.

Các em có thể tìm hiểu và đưa ra nhiều ví dụ khác nữa.

Theo em có những điểm gì chung với các ví dụ trên?

4. Máy tính "nghĩ" như thế nào?

Em hãy quan sát sơ đồ sau hay được mô phỏng cho các công việc xử lý trên máy tính.



Input (đầu vào):

Đầu vào được hiểu là:

- Bài toán hay vấn đề cần giải quyết.
- Các thông số, dữ liệu ban đầu của bài toán.

Bộ xử lý:

Là chương trình máy tính, phần mềm hoặc thiết bị vi xử lý bất kỳ có chức năng tiếp nhận đầu vào, sau đó tiến hành giải quyết bài toán.

Output (đầu ra):

Kết quả của bài toán, vấn đề được thể hiện ở đầu ra.

Ví dụ bài toán tìm đường đi tối ưu.

Input (đầu vào)	Xử lý	Output (đầu ra)
<ul style="list-style-type: none"> - Vị trí 2 điểm Đầu và Cuối. - Bản đồ đường giao thông thành phố. - Các qui định khác về giao thông, các tuyến đường 1 chiều, 	<p>Thực hiện tìm đường đi tối ưu từ vị trí Đầu đến vị trí Cuối.</p> <p>Máy tính thực hiện bài toán theo 1 qui trình đã được con người xác định (lập trình) trước theo các bước chặt chẽ, tối ưu, bảo đảm tìm ra được kết quả.</p>	<p>Thể hiện trên màn hình đường đi tối ưu này.</p>

Như vậy "tư duy của máy tính" thực chất là việc con người cung cấp cho máy tính các dữ liệu đầu vào và cách xử lý vấn đề thông qua các chương trình, các bước, hay các lệnh một cách tuần tự, chặt chẽ, đảm bảo giải quyết được bài toán, vấn đề được đặt ra.

5. Con người sử dụng máy tính để giải quyết vấn đề gì và như thế nào?

Trên thực tế con người sử dụng máy tính để giải quyết các bài toán mà nếu không có máy tính sẽ rất khó giải quyết.

Qui trình con người sử dụng máy tính để giải quyết vấn đề thường theo các bước như sau:

- Xác định bài toán, vấn đề cần giải quyết (ví dụ: bài toán tìm đường đi trên bản đồ).
- Xác định các dữ liệu đầu vào ban đầu (vị trí 2 điểm Đầu và Cuối, bản đồ giao thông đường phố, ...).
- Xây dựng chương trình để đưa vào máy tính xử lý (lập chương trình bao gồm các lệnh, đưa vào máy tính để xử lý).
- Máy tính chạy theo chương trình đã nạp và đưa ra kết quả, thể hiện là đầu ra trên màn hình máy tính.

Câu hỏi và bài tập

1. Em hãy liệt kê các bước để thực hiện 1 số công việc hàng ngày ở nhà em (như thổi cơm, quét nhà, giặt quần áo,).
2. Em hãy chỉ ra thêm các trò chơi và phần mềm mô phỏng giải quyết vấn đề.
3. Cách giải quyết bài toán sau là đúng hay sai?

Bài toán: cho trước 3 số a , b , c , tính tổng $a^2 + b^2 + c^2$ nếu biết rằng mỗi bước chỉ được phép thực hiện 1 phép toán 2 số.

Input: a , b , c .

Xử lý theo các bước sau:

B1. Tính $a^2 + b^2$

B2. Tính c^2

B3. Lấy kết quả B1 cộng với B2.

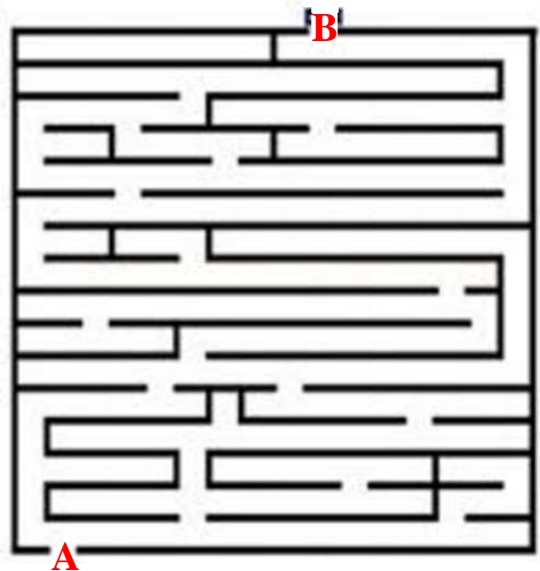
Output: in ra kết quả.

4. Cho trước số tự nhiên N ($N > 1$). Mỗi bước em chỉ có thể thực hiện 1 trong các điều sau:

- Giảm N đi 1.
- Chia đôi N nếu N là chẵn.

Giả sử cho $N = 100$. Hỏi em có thể thực hiện nhanh nhất là bao nhiêu bước để thu được số 1.

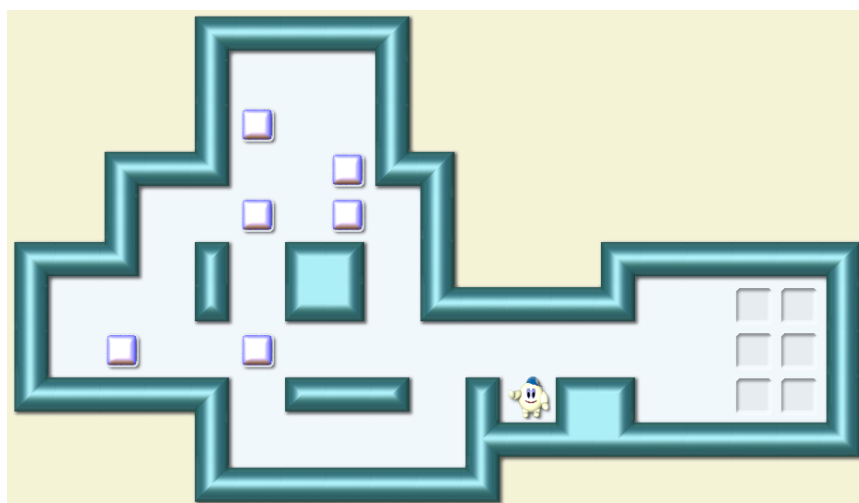
5. Em hãy tìm 1 đường đi từ vị trí A đến vị trí B.



Mở rộng

Sử dụng cách mô tả mô hình xử lý vấn đề ở trên, em hãy viết và trình bày sơ đồ xử lý với 1 số bài toán sau (viết và mô tả Input, Output và Bộ xử lý).

1. Đầu vào là 3 số bất kỳ a, b, c , đầu ra là các số này nhưng đã được sắp xếp theo thứ tự tăng dần. Ví dụ đầu vào là 5, -1, 3 thì đầu ra của bài toán phải là -1, 3, 5. Biết rằng mỗi bước chỉ cho phép thực hiện đổi chỗ 2 số cạnh nhau hoặc có thể so sánh 2 số bất kỳ để kiểm tra xem số nào lớn hơn, số nào nhỏ hơn.
2. Bài toán tính tổng của 3 số bất kỳ cho trước a, b, c nếu biết rằng mỗi bước chỉ cho phép thực hiện: lấy tổng 2 số dương, đổi dấu của 1 số (từ âm thành dương và ngược lại), kiểm tra 2 số xem số nào lớn hơn, hoặc lấy 1 hiệu của 1 số dương lớn hơn trừ đi 1 số dương nhỏ hơn.
3. Giải bài toán chuyển đồ Socoban sau.



Bài 2. Làm quen với Scratch

Mục đích

Học xong bài này, bạn sẽ biết:

- Làm quen và biết 1 môi trường ứng dụng mới: lập trình kéo thả.
- Lập trình tức là điều khiển máy tính hoạt động theo các bước, các lệnh tuần tự.
- Hiểu được qui trình hoạt động cơ bản của 1 chương trình trong môi trường Scratch.

Bắt đầu

Trong bài trước các em đã được làm quen với khái niệm "tư duy máy tính", tức là cách mà máy tính có thể "nghĩ" và "làm việc", "hành động" theo sự điều khiển của con người. Bây giờ chúng ta sẽ làm quen với một trong những hành động cụ thể đó của máy tính. Chúng ta sẽ được làm quen với một môi trường giao tiếp mới để con người có thể điều khiển máy tính có thể "suy nghĩ" và "làm việc", đó là môi trường Scratch.

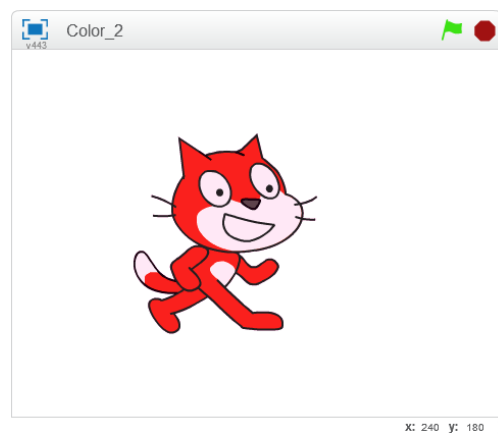
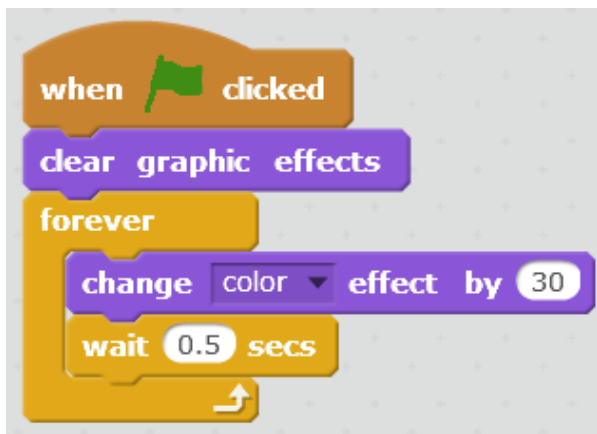
1. Em hãy quan sát 1 chương trình chạy trên môi trường Scratch và đưa ra các nhận xét của mình. Chương trình được ghi trong tệp Start.Dance.sb2.



Các gợi ý:

- Để chạy chương trình, em nhấp lên hình lá cờ màu xanh phía trên cửa sổ chính. Em có nhận xét gì về hoạt động của cậu bé trên màn hình?
- Để dừng chương trình em bấm nút tròn màu đỏ phía trên (cạnh hình lá cờ).
- Em hãy chú đến các hoạt động của cậu bé, âm thanh trống và quan sát nền sân khấu.

2. Em mở tệp Start.Color.sb2, chạy chương trình và viết lại nhận xét của em về hoạt động của nhân vật.



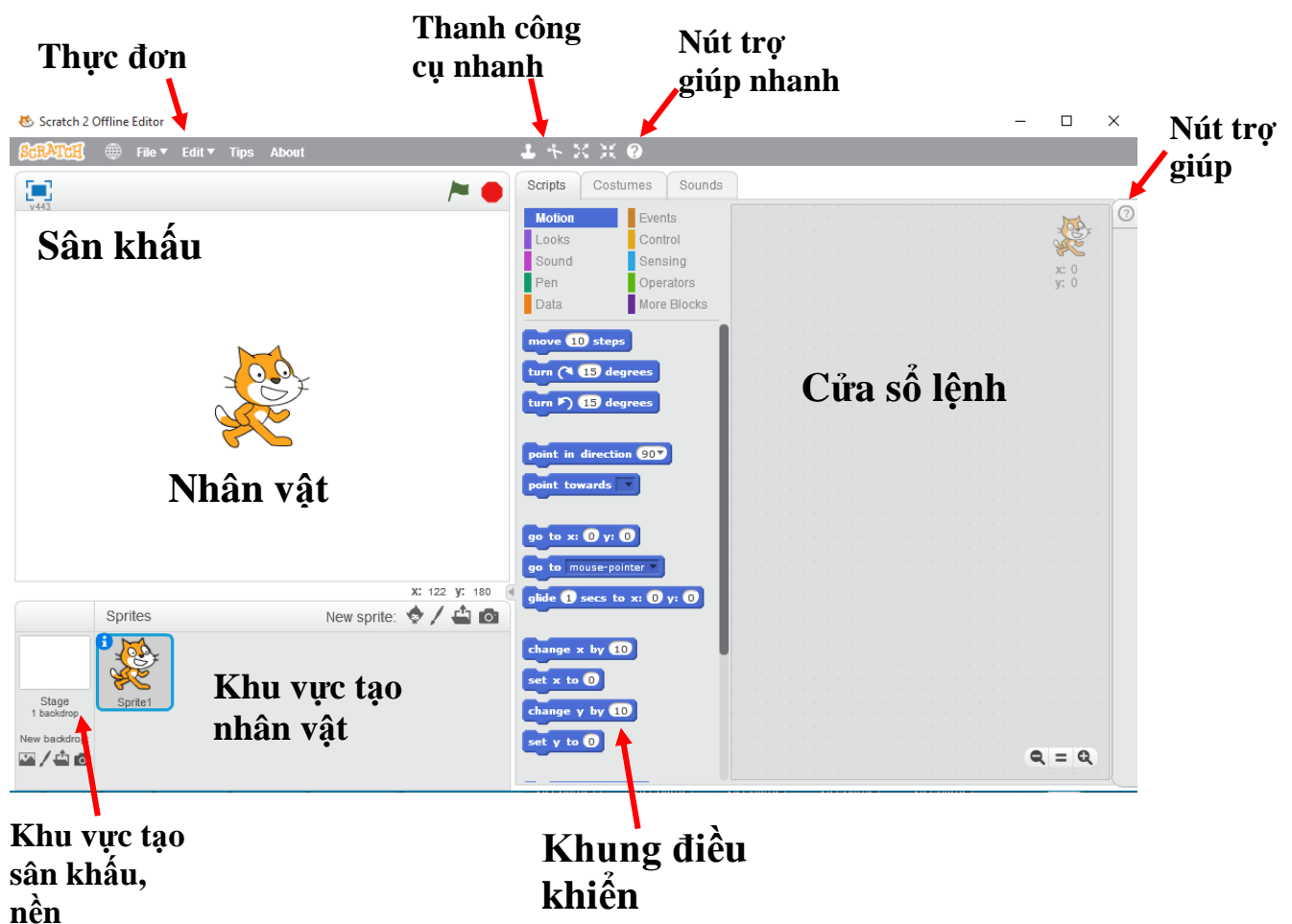
Gợi ý:

- Quan sát sự thay đổi màu sắc bên ngoài của con mèo.
- Em có thể giải thích vì sao mèo lại thay đổi màu của bộ lông của mình không?

Nội dung bài học

1. Làm quen với giao diện Scratch

Scratch là môi trường mà em sẽ được học trong môn học này. Các em hãy quan sát giao diện chính của Scratch:



Chúng ta cùng tìm hiểu nhanh các vị trí quan trọng trong giao diện Scratch.

Sân khấu

Sân khấu là cửa sổ thể hiện chính của phần mềm. Khi phần mềm chạy chúng ta quan sát phần mềm thông qua sân khấu, tương tự như khi chúng ta xem biểu diễn ca nhạc, xem phim, xem Tivi.

Nhân vật

Nhân vật xuất hiện trên sân khấu, là đối tượng chính của các hoạt động. Có thể có nhiều nhân vật, đa dạng về kích thước và chủng loại. Khi lần đầu tiên chạy Scratch, nhân vật chính là 1 chú mèo xinh xắn.

Cửa sổ lệnh

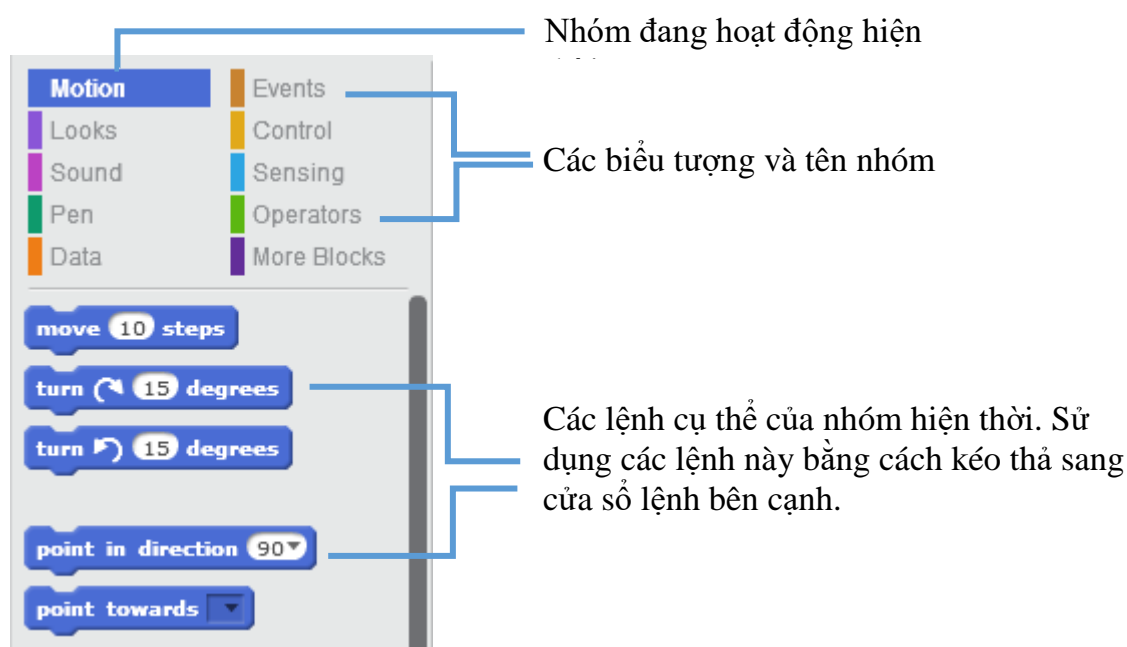
Cửa sổ lệnh chứa các "lệnh" để điều khiển hoạt động của nhân vật. Mỗi nhân vật có 1 cửa sổ lệnh riêng. Trong cửa sổ này không cần phải viết lệnh mà chỉ cần kéo thả các lệnh từ khung điều khiển bên cạnh sang. Vì vậy Scratch được gọi là môi trường lập trình kéo thả.

Khung điều khiển

Là nơi chứa các "công cụ" dùng để tạo ra chương trình. Tại khung này chứa các mẫu lệnh, người lập trình sử dụng các công cụ này để viết chương trình của mình.

Khung điều khiển này sẽ có 3 TAB thông tin: **Script** (Lệnh) , **Costume (Backdrop)** (Trang phục / Nền sân khấu) và **Sound** (Âm thanh).

Các lệnh trong Scratch được chia thành nhóm, mỗi nhóm bao gồm các lệnh có ý nghĩa gần tương tự nhau và thể hiện bằng 1 màu cụ thể. Có 10 nhóm các lệnh như vậy trong Scratch. Khi nhấp lên 1 nhóm, khung phía dưới sẽ xuất hiện các lệnh của nhóm này.



Thực đơn

Thực đơn chứa các lệnh chính của Scratch.

Thanh công cụ nhanh

Thanh công cụ nhanh chứa một số lệnh làm việc nhanh với nhân vật và các lệnh.

Nút trợ giúp nhanh

Nút này có ý nghĩa như như: nhấp chuột lên nút, sau đó nhấp chuột lên 1 lệnh bất kỳ sẽ hiển thị nội dung mô tả của lệnh này.

Nút trợ giúp

Nhấp nút này để hiển thị các trợ giúp nhanh của phần mềm.

Khu vực tạo nhân vật

Tại khu vực này em có thể thực hiện các thao tác như tạo thêm nhân vật, chỉnh sửa ngoại hình nhân vật (thay đổi trang phục), bổ sung âm thanh,

Khu vực tạo sân khấu, nền

Tại khu vực này em có thể thực hiện các thao tác với sân khấu như trang trí sân khấu, tạo thêm các cảnh sân khấu khác, tạo âm thanh nền cho sân khấu.

2. Điều khiển nhân vật bằng các lệnh

Các em hãy thực hiện các bước sau để hiểu cách điều khiển hoạt động của nhân vật con mèo.

(i) Nhấp nút **Scripts**, nhấp lên nhóm lệnh **Look** (màu tím)



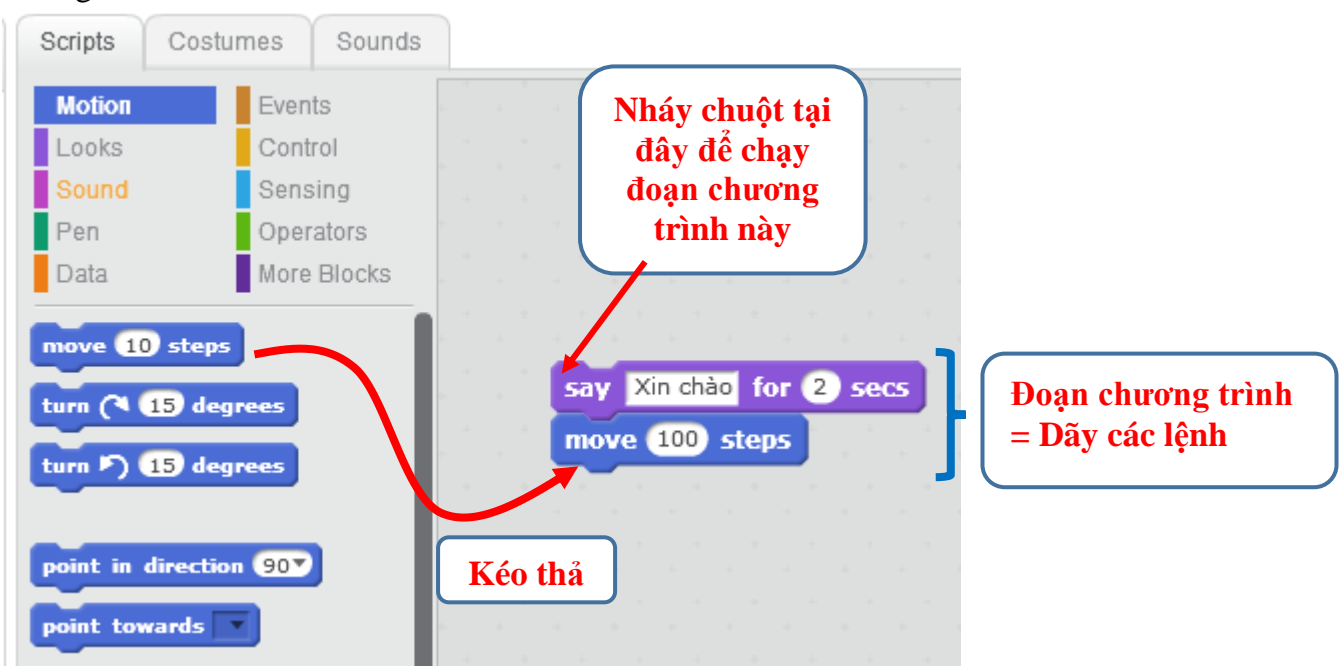
(ii) Kéo thả lệnh từ khung điều khiển phía dưới sang cửa sổ bên phải, sau đó sửa chữ "Hello!" thành "Xin chào".

(iii) Nhấp lên nút nhóm **Motion** (màu xanh thẫm)



(iv) Kéo thả lệnh từ khung điều khiển phía dưới sang cửa sổ bên phải, sau đó sửa số 10 thành 100.

(v) Dùng chuột kéo 2 lệnh trên sát vào nhau như hình dưới đây, ta thu được 1 đoạn chương trình gồm 2 lệnh.



Bây giờ để chạy đoạn chương trình này em hãy nhấp chuột lên vị trí bất kỳ của đoạn chương trình trên, và quan sát xem con mèo thay đổi như thế nào.

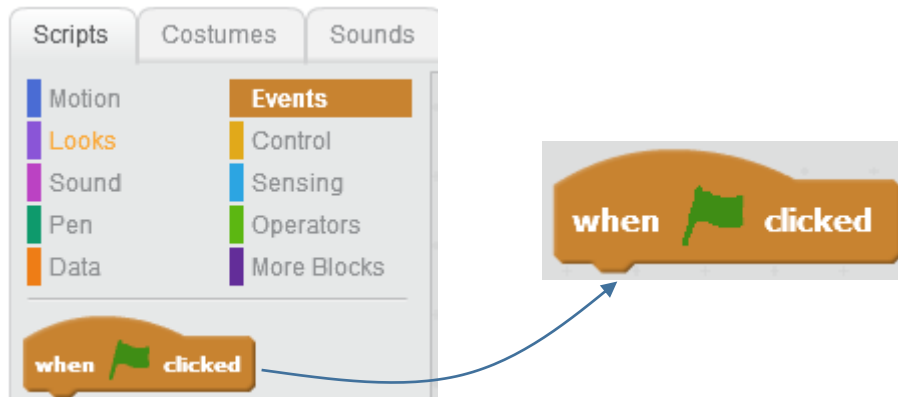
Các em sẽ thấy gì?

Nếu bây giờ em tách 2 lệnh trên rời ra và chạy thử em thấy có gì thay đổi?

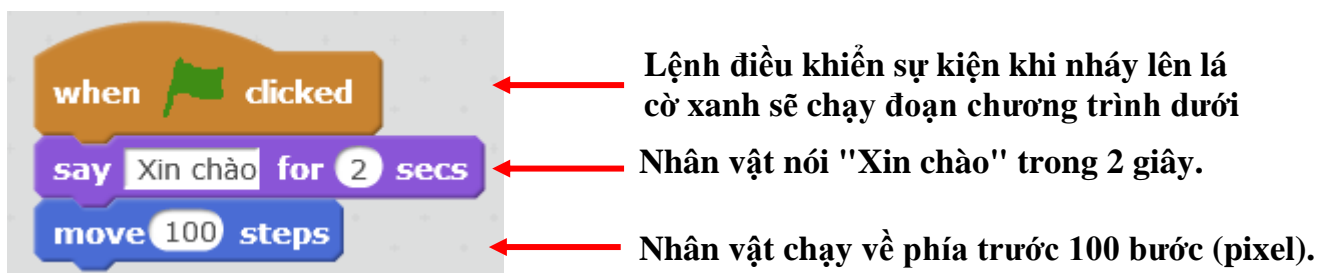
3. Bắt đầu 1 chương trình đơn giản bằng sự kiện "Bắt đầu chương trình"

Bây giờ chúng ta sẽ cùng nhau hoàn thiện đoạn chương trình trên thành một chương trình hoàn chỉnh.

Chọn nút nhóm lệnh có chữ Events (sự kiện), kéo thả lệnh sau sang cửa sổ lệnh chính.



Sau đó em điều chỉnh để có 1 đoạn chương trình gồm 3 lệnh sau.



Bây giờ chúng ta đã có 1 chương trình hoàn chỉnh trên Scratch. Em có thể phóng to cửa sổ chạy của chương trình và thực hiện lệnh chạy bằng cách nhấp biểu tượng lá cờ xanh. Khi chạy chương trình, nhân vật của chúng ta sẽ nói "Xin chào" trong 2 giây, sau đó sẽ chuyển động về phía trước 100 bước.

Chương trình trên có thể viết lại dưới dạng các dòng chữ tiếng Việt cho dễ hiểu như sau:

Nếu nhấp vào lá cờ xanh (nếu sự kiện này xảy ra)

Thì hiện dòng chữ "Xin chào" trong 2 giây.

Chạy về phía trước 100 bước.


2 lệnh này sẽ được thực hiện nếu sự kiện trên xảy

Cách viết như trên thường được dùng trong tin học, khi thiết lập các vấn đề, bài toán và viết các bước thực hiện để giải quyết trên máy tính.


Một vài kết luận ban đầu.

- Các nhân vật có thể điều khiển bằng các lệnh trong cửa sổ lệnh.
- Lệnh có thể được kéo thả từ khung các mẫu lệnh, không phải viết từng lệnh trên màn hình.

- Nhóm các lệnh gắn liền nhau tạo thành 1 đoạn chương trình. Khi chạy các lệnh sẽ lần lượt chạy từ trên xuống dưới.

- Để hoàn thiện 1 chương trình hoàn chỉnh, cần đưa lệnh sự kiện  lên đầu của đoạn chương trình.

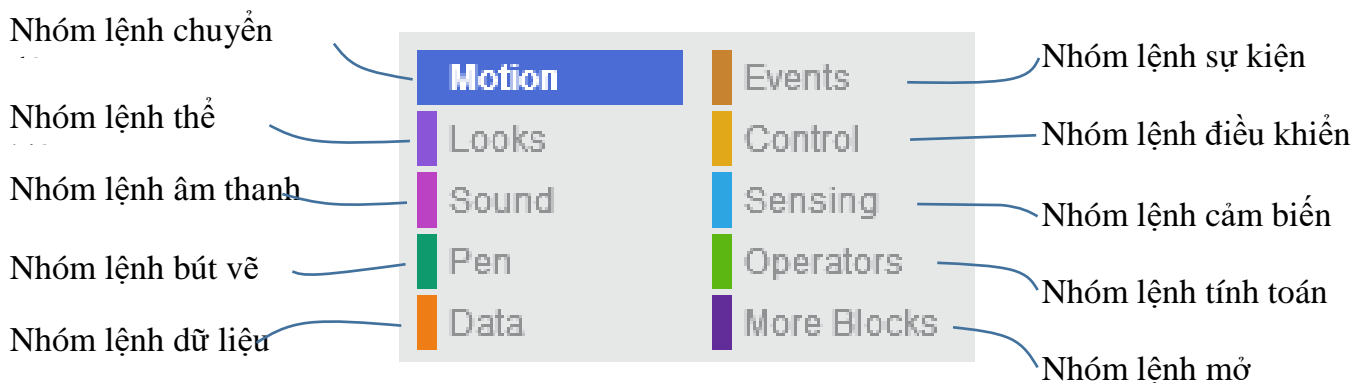
- Để chạy chương trình nháy lên nút có hình lá cờ xanh. Nháy nút tròn đỏ sẽ dừng chương trình đang chạy.

- Có thể phóng to cửa sổ chạy bằng cách nháy vào nút  góc trái trên màn hình.

6. Phân loại các nhóm lệnh điều khiển nhân vật

Em hãy quan sát vào khu vực khung điều khiển các lệnh của Scratch. Toàn bộ hệ thống lệnh được chia làm bao nhiêu nhóm? Mỗi nhóm có tên là gì và đặc trưng bởi màu sắc gì?

Em hãy nhìn vào sơ đồ sau để thấy toàn bộ các nhóm lệnh trong Scratch.



- Hãy liệt kê các nhóm lệnh của Scratch, màu sắc thể hiện của nhóm và nêu nhanh ý nghĩa các nhóm này.

- Với môi trường lập trình kéo thả Scratch chúng ta không cần học thuộc lòng và nhớ chi tiết từng lệnh. Chỉ cần hiểu ý nghĩa lệnh, cách sử dụng và kéo thả chúng sang cửa sổ lệnh là dùng được.


Câu hỏi và bài tập


1. Trong ví dụ trên, nếu em đặt 2 lệnh tách ra như hình sau:



thì em có thể chạy 2 lệnh trên như 1 chương trình không?

Kết luận: các lệnh rời rạc nhau không tạo thành 1 chương trình hoàn chỉnh. Các lệnh cần kết nối với nhau.

2. Nút  nằm ở góc phải dưới của cửa sổ lệnh có ý nghĩa gì? Hãy thao tác và trả lời câu hỏi trên.

3. Hai nút nhỏ  này nằm trên thanh công cụ nhanh phía trên màn hình có tác dụng gì? Để hiểu em hãy thực hiện thao tác sau:

- Nháy chuột lên 1 trong 2 nút trên.
- Sau đó em nháy chuột nhiều lần lên nhân vật.
- Em thấy điều gì xảy ra? Từ đó suy ra ý nghĩa của 2 nút trên.

4. Để tạo một chương trình mới, em thực hiện lệnh File / New.

Em hãy tạo 1 chương trình mới và tạo 1 đoạn chương trình đơn giản sau với nhân vật chính con mèo.



Chạy chương trình và quan sát xem con mèo thực hiện những hành động gì.

5. Để ghi lại chương trình đang làm việc, em thực hiện lệnh File / Save. Tệp chương trình Scratch cần có phần mở rộng là *.sb hoặc *.sb2

Mở rộng

Em hãy tìm hiểu các lệnh trong nhóm chuyển động (motion) và viết 1 chương trình hoàn chỉnh điều khiển nhân vật của em chuyển động trên màn hình.

CHƯƠNG 2: BẮT ĐẦU LẬP TRÌNH SCRATCH

Bài 3. Chuyển động 1

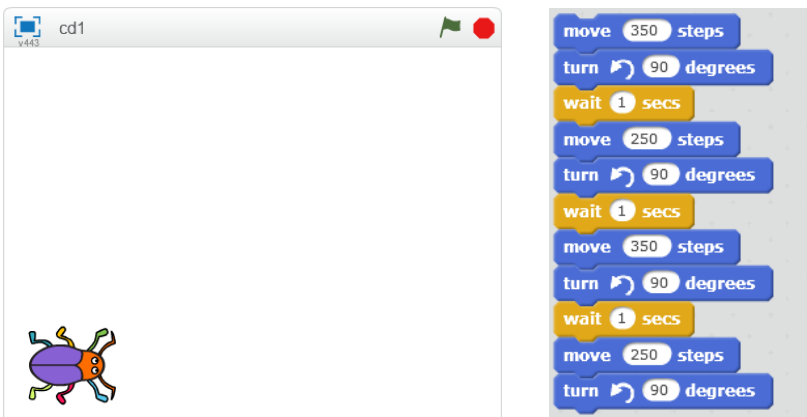
Mục đích

Học xong bài này, bạn có thể:

- Điều khiển nhân vật di chuyển trên màn hình (sân khấu) bằng dãy lệnh đơn giản.
- Thiết lập trang phục (costume, thể hiện bên ngoài, quần áo) cho nhân vật.
- Thay đổi nền sân khấu, màn hình.

Bắt đầu


Em hãy mở chương trình moving.1.khoidong.sb2, quan sát và chạy chương trình.




a) em hãy mô tả những gì xảy ra trên màn hình.

b) theo em các lệnh nào đã làm cho nhân vật (con cánh cam) chuyển động?

A. Lệnh  ?

B. Lệnh  ?

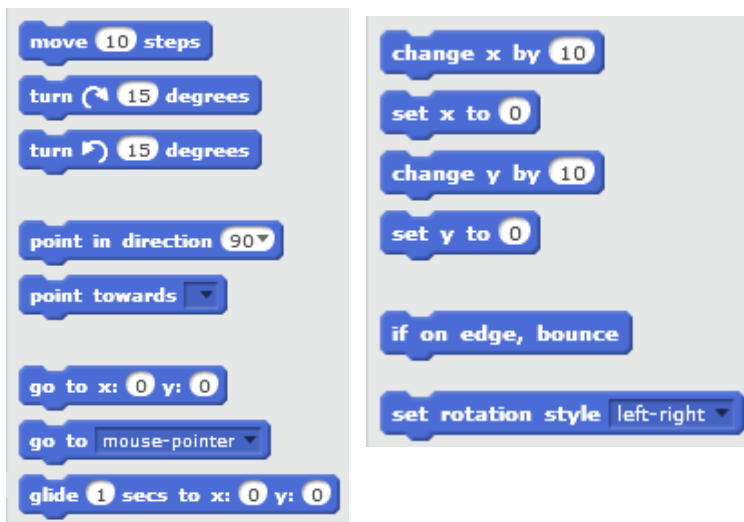
C. Lệnh  ?

Em có nhận xét gì về màu sắc của các lệnh làm nhân vật chuyển động?

Nội dung bài học

1. Cùng quan sát nhóm lệnh Motion

Em hãy quan sát nhóm các lệnh trong nhóm chuyển động (motion) của Scratch.



Em có nhận xét gì về các lệnh này?

- Tất cả các lệnh trong nhóm Chuyển động đều có cùng màu xanh.
- Các lệnh này có cùng chức năng điều khiển hoạt động của các nhân vật trên màn hình.

2. Tọa độ nhân vật và kích thước sân khấu

Để có thể điều khiển chuyển động nhân vật dễ dàng chúng ta cần tìm hiểu thêm thông tin về sân khấu mà trên đó các nhân vật đóng vai chính.

Kích thước sân khấu

Sân khấu luôn có kích thước hình chữ nhật 480 x 360. Chiều ngang = 480 (điểm / pixel), chiều cao = 360 (điểm / pixel).

Tâm của sân khấu chính là tâm của hệ trục tọa độ màn hình.

Mỗi vị trí trên sân khấu sẽ được đánh dấu bởi 2 số, 2 tọa độ (x, y). Ở đây:

x = khoảng cách từ vị trí này đến trục đứng (trục tung, trục Y); y = khoảng cách từ vị trí này đến trục ngang (trục hoành, trục X).

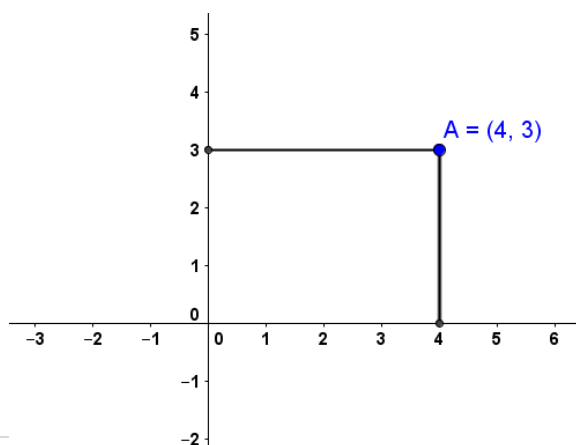
Cặp số (x, y) được gọi là tọa độ của vị trí, điểm đã cho.

Giá trị tọa độ x chạy từ -240 đến 240.

Giá trị tọa độ y chạy từ -180 đến 180.

Em hãy xác định trên sân khấu các vị trí có tọa độ sau:

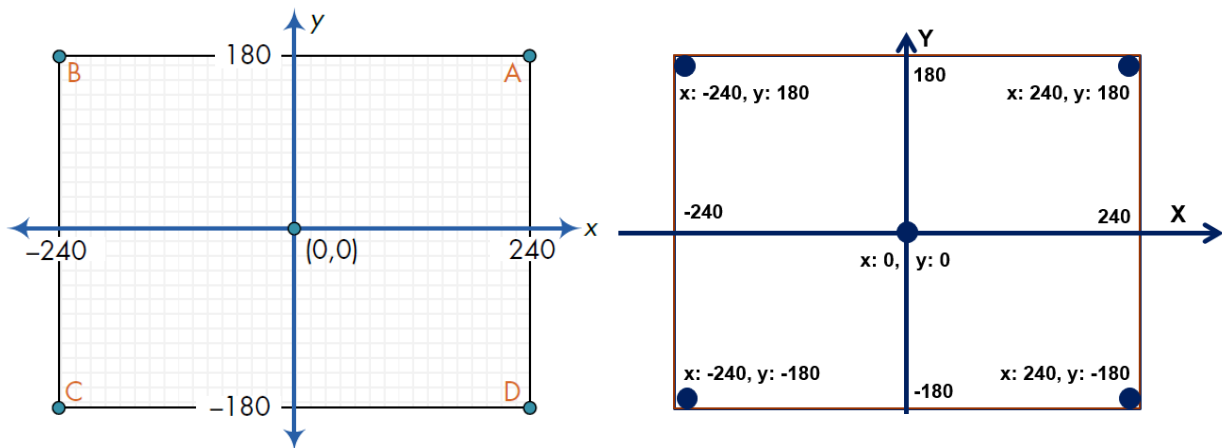
(0, 0); (10, -10); (100, 100).



Tọa độ của điểm trên mặt phẳng có trục tọa độ và cách tính tọa độ.

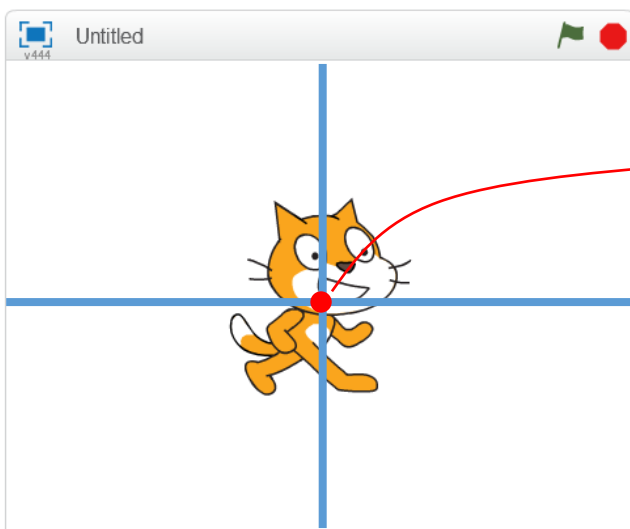
Điểm A trên hình có tọa độ x=4 theo trục hoành (ngang) và y=3 theo trục tung (ngang).

Mô hình tọa độ và kích thước sân khấu của Scratch.



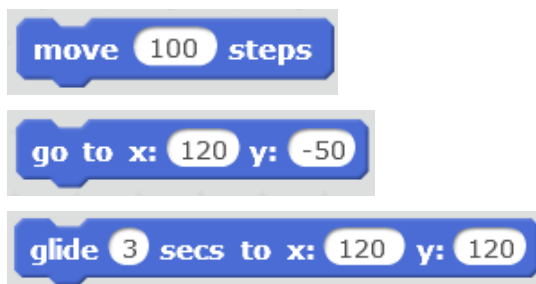
Tọa độ của nhân vật

Khi nhân vật chuyển động trên màn hình, chúng ta xác định nhân vật này bằng tọa độ của chúng. Tọa độ của nhân vật được tính là tọa độ của điểm chính tâm của nhân vật. Ví dụ điểm tính tọa độ của con mèo như hình dưới đây.



Vị trí này sẽ xác định tọa độ của nhân vật con mèo.

Em hãy kéo thả lần lượt các lệnh sau và kiểm tra xem nhân vật của chúng ta chuyển động như thế nào. Viết mô tả sang cột bên cạnh.

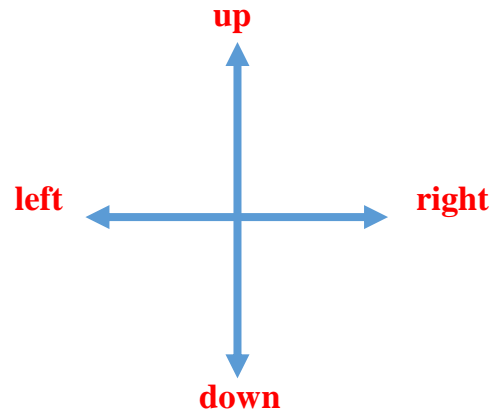
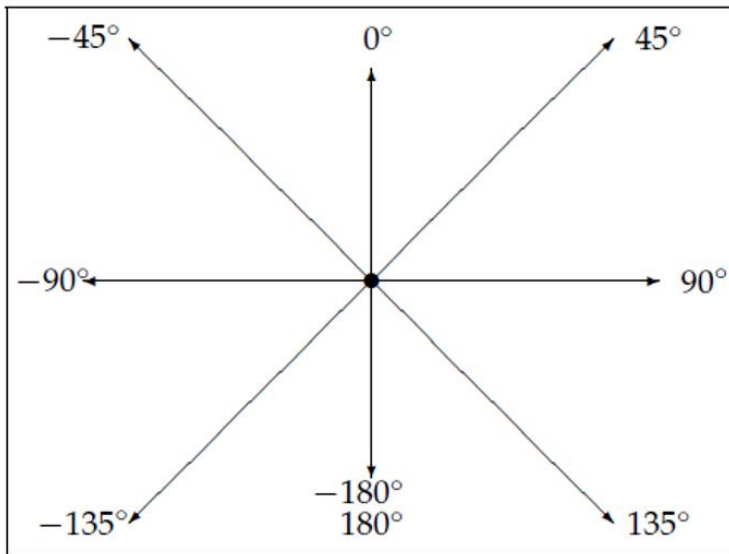


Biết được kích thước, tọa độ sân khấu chúng ta có thể điều khiển nhân vật chuyển động đến bất cứ vị trí nào chúng ta muốn.

2. Hướng chuyển động của nhân vật

move 100 steps

Khi chuyển động bởi lệnh , nhân vật bao giờ cũng di chuyển theo 1 hướng nhất định. Trong Scratch, các hướng được qui định bởi 1 số đo góc, như chỉ ra theo sơ đồ sau.



Giá trị góc của hướng tính từ -180° đến 180° .

turn 15 degrees

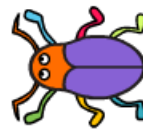
Lệnh cho phép nhân vật quay theo chiều kim đồng hồ 1 góc cho trước (ví dụ 15 độ theo hình ảnh).

turn 15 degrees

Lệnh cho phép nhân vật quay theo chiều ngược kim đồng hồ 1 góc cho trước (ví dụ 15 độ theo hình ảnh).

Khi 1 nhân vật mới được khởi tạo, nhân vật này luôn hướng theo hướng mặc định là hướng phải (right, 90°).

Nhân vật đang
xoay theo hướng
down (180° / -



Nhân vật đang
xoay theo hướng
left (-90°)

Mặc định nhân
vật có hướng
right (90°)



Nhân vật đang
xoay theo hướng
up (0°)

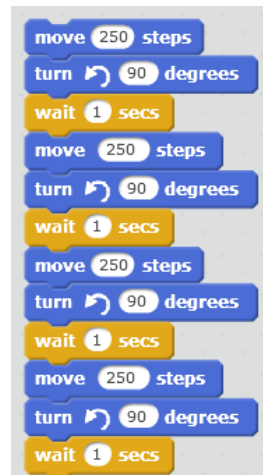
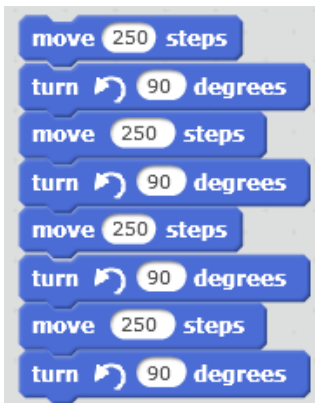
Hãy cho biết hướng các nhân vật này là hướng nào?



3. Hãy cho nhân vật chuyển động đơn giản trên sân khấu


Bây giờ các em hãy cùng thực hiện các chương trình đơn giản điều khiển các nhân vật chuyển động trên màn hình.


1. Thiết lập 2 chương trình sau và quan sát kết quả thực hiện.



Nhận xét:

- Kết quả thực hiện 2 chương trình này là như nhau (nhân vật chuyển động theo 1 hình vuông và quay về vị trí ban đầu), tuy nhiên với chương trình 1 rất khó quan sát vì tốc độ chuyển động nhanh quá, còn chương trình 2 thì dễ dàng quan sát sự chuyển động của nhân vật.

- Trong chương trình 2, lệnh  (ý nghĩa tạm dừng thực hiện chương trình trong 1 giây) có ý nghĩa giúp chúng ta quan sát được sự chuyển động rõ ràng hơn. Kinh nghiệm này hay được sử dụng.

- Lệnh  có màu vàng nhạt nằm trong nhóm lệnh Điều khiển (Control) mà các em sẽ được làm quen trong các bài học tiếp theo.

2. Bây giờ em sẽ cho nhân vật chính chuyển động theo chiều ngang trên sân khấu, ví dụ chạy từ trái sang phải và ngược lại.


Em hãy thiết lập chương trình có các lệnh sau:



Lệnh này có tác dụng cho nhân vật chính quay lại (xoay ngược kim đồng hồ)

Các em sẽ thấy gì?

Em sẽ thấy chú mèo khi quay ngược kim đồng hồ 180 độ sẽ bị lộn ngược!

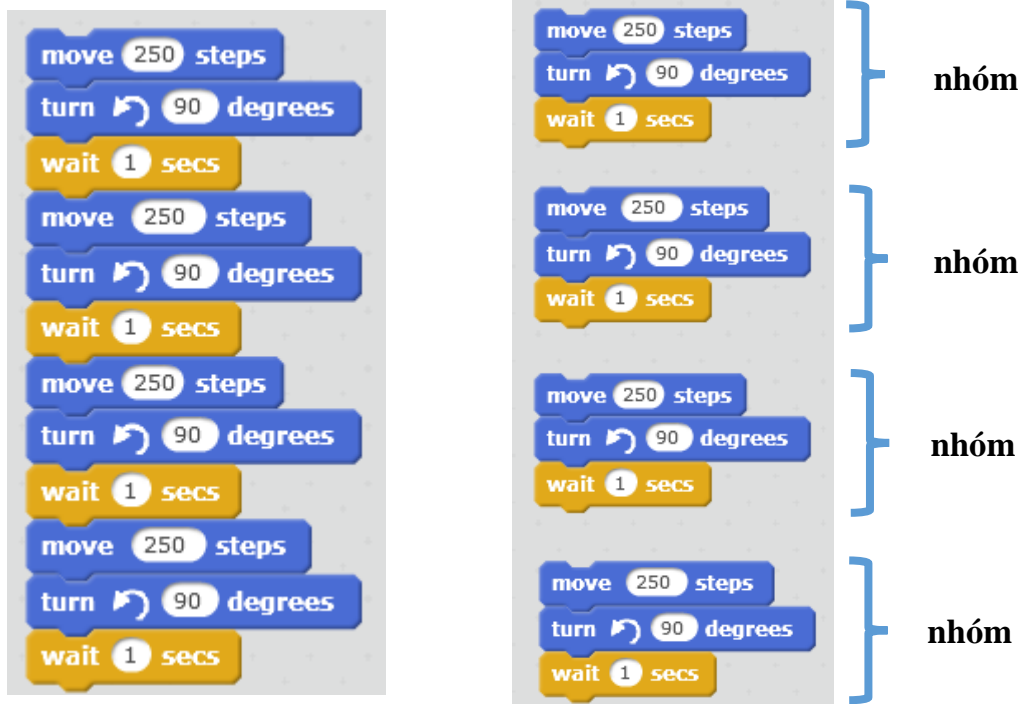
Để khắc phục lỗi này chúng ta sẽ dùng lệnh  để điều chỉnh cách xoay người của nhân vật. Lệnh này có 3 lựa chọn bằng cách dùng chuột nhấp chọn tại nút phía phải của lệnh. 3 lựa chọn quay là: chỉ **quay trái phải** (left-right); **không quay** (don't rotate); và **quay tròn** (all around). Chúng ta nhìn thấy chú mèo bị lộn ngược là do lựa chọn quay tròn.

Bây giờ hãy viết lại đoạn chương trình trên:



4. Thêm lệnh lặp

Chúng ta sẽ quan sát kỹ hơn chương trình đã làm trong phần đầu của mục trên và chú ý đến 4 nhóm lệnh lặp lại giống nhau.



Để khỏi phải viết lại các nhóm lệnh giống nhau nhiều lần, trong Scratch (và trong mọi môi trường lập trình khác), người ta sử dụng các lệnh để điều khiển các cấu trúc lặp này.

Trong Scratch lệnh điều khiển này có dạng sau:



Kéo thả các nhóm lệnh muốn lặp vào vị trí này. Có thể thay đổi giá trị số 10 bằng 1 số

Ý nghĩa của lệnh là cho phép thực hiện nhóm các lệnh nằm bên trong lệnh này có thể được thực hiện lặp lại 1 số lần. Số lần lặp có thể điều chỉnh trực tiếp trên lệnh.

Quay trở lại ví dụ trên, chúng ta có thể viết lại đoạn chương trình trên bằng 1 chương trình khác, ngắn gọn. Lệnh lặp được lấy từ nhóm lệnh Điều khiển.



1. Kéo thả nhóm lệnh này vào đây



2. Kết quả thu được, nhóm lệnh được thực hiện lặp 4

Kỹ thuật lặp chương trình được sử dụng rất nhiều trong thực tế, trong tất cả các môi trường lập trình máy tính.

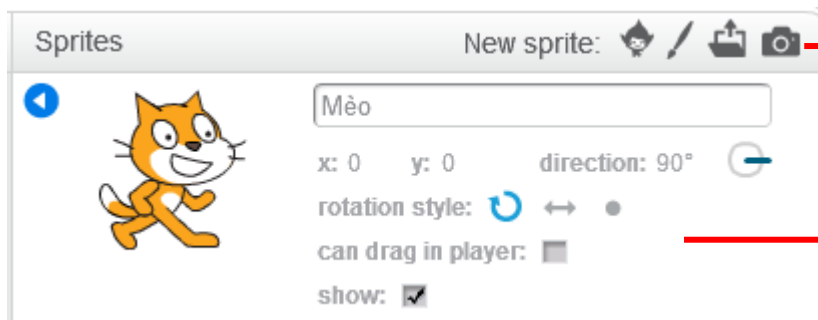
Em hãy viết lại đoạn chương trình lặp trên và chạy thử để kiểm tra kết quả.

5. Bổ sung nhân vật

Đối tượng điều khiển chính của 1 chương trình Scratch là nhân vật trên sân khấu. Chúng ta điều khiển nhân vật bằng chương trình được đưa ra (bằng cách kéo thả mẫu lệnh) trên Cửa sổ lệnh của nhân vật. Mỗi nhân vật có 1 cửa sổ lệnh (Script Window) của riêng mình. Điều đặc biệt thú vị là chúng ta có thể tạo ra nhiều nhân vật trên sân khấu.

1. Khu vực làm việc với nhân vật.

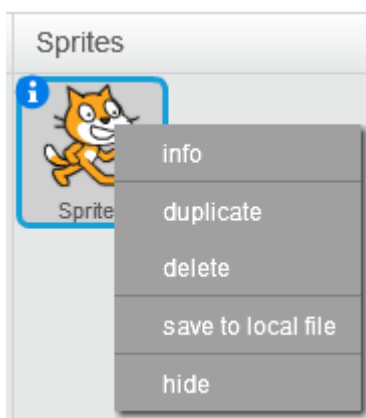
Khu vực phía dưới sân khấu chính là nơi tập trung thông tin của các nhân vật có trong chương trình. Nháy lên chữ i nhỏ sẽ xuất hiện các thông tin của nhân vật này. Khung thông tin nhân vật và các tính năng tương ứng như sau:



Các nút lệnh bổ sung thêm nhân vật.

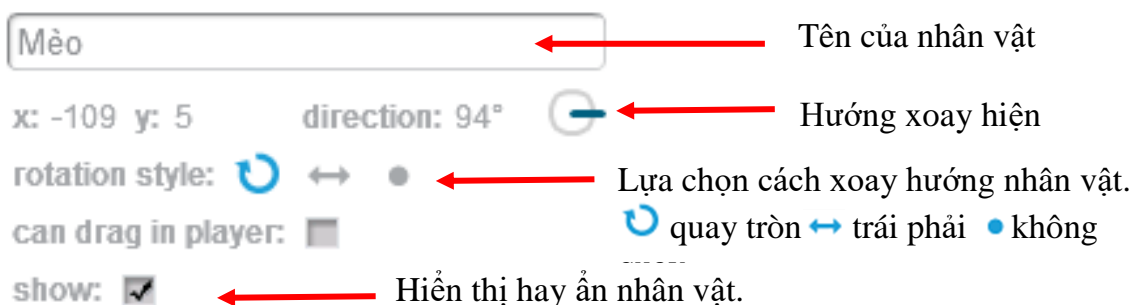
Cửa sổ thông tin của nhân vật. Tại đây có thể thay đổi, sửa trực tiếp thông tin của nhân vật.

Khi nhấp chuột phải lên biểu tượng nhân vật sẽ có xuất hiện thêm 1 bảng chọn các lệnh nữa đối với nhân vật này.




Bảng chọn các lệnh với nhân vật khi nhấp chuột phải lên nhân vật. Các chức năng từ trên xuống: **xem thông tin; tạo 1 bản sao; xóa; ghi hình ảnh ra file; ẩn không hiện trên màn hình.**

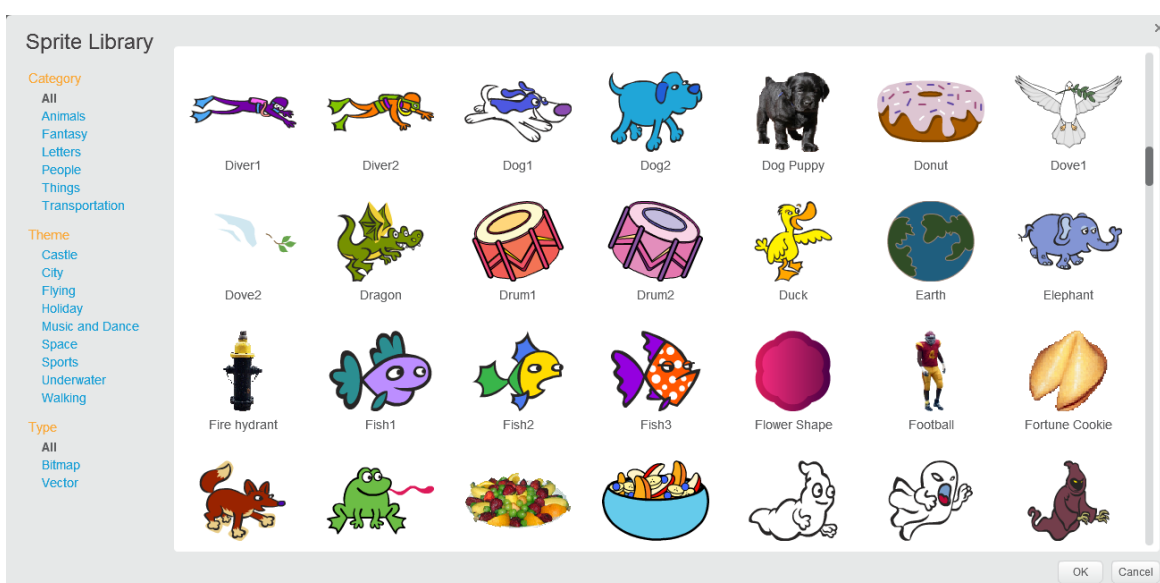
Tại cửa sổ thông tin của nhân vật chúng ta có thể thực hiện các thao tác sau.



2. Bổ sung thêm nhân vật từ thư viện dữ liệu

Để bổ sung thêm nhân vật từ 1 thư viện có sẵn, em hãy thực hiện các bước sau.

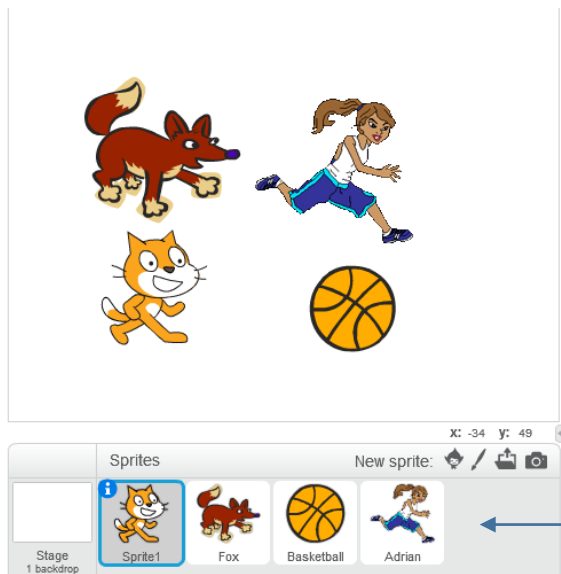
- Nhấp nút  tại khung nhân vật.
- Xuất hiện cửa sổ các nhân vật trong thư viện có sẵn.



Kho thư viện này có rất nhiều hình ảnh người, con vật, đồ dùng, ... có thể chọn làm nhân vật của chương trình.

- Nháy chuột chọn 1 hình và nháy nút OK để bổ sung hình này vào chương trình như 1 nhân vật mới.

Hình ảnh sau mô tả 1 chương trình có 4 nhân vật.



Danh sách các nhân vật hiện tại đây. Có thể bổ sung không hạn chế số lượng nhân vật cho mỗi chương trình Scratch.

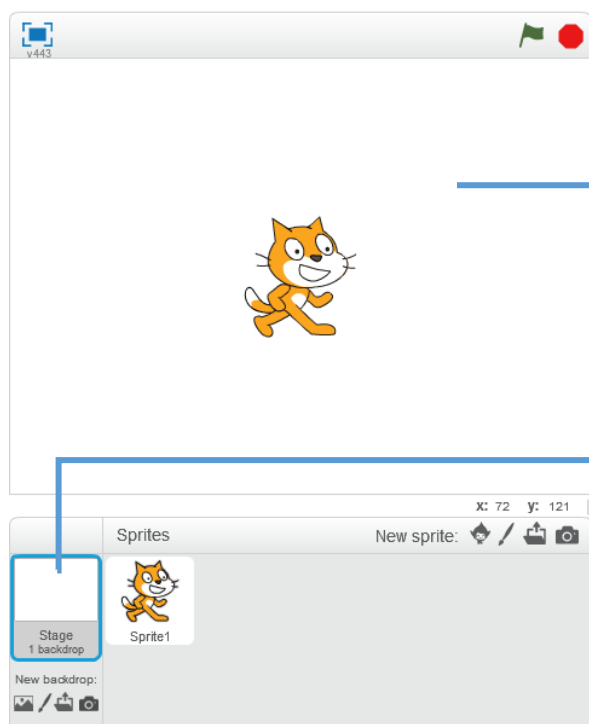
3. Hoạt động của em

Em hãy thiết lập 1 chương trình, bổ sung vào sân khấu từ 2 đến 5 nhân vật. Sau đó em viết các chương trình độc lập điều khiển các nhân vật này.

6. Thay đổi nền sân khấu

1. Nền sân khấu là gì?

Nền sân khấu là những gì thể hiện trên cửa sổ chính của chương trình. Các nhân vật đều hoạt động trên nền của sân khấu.

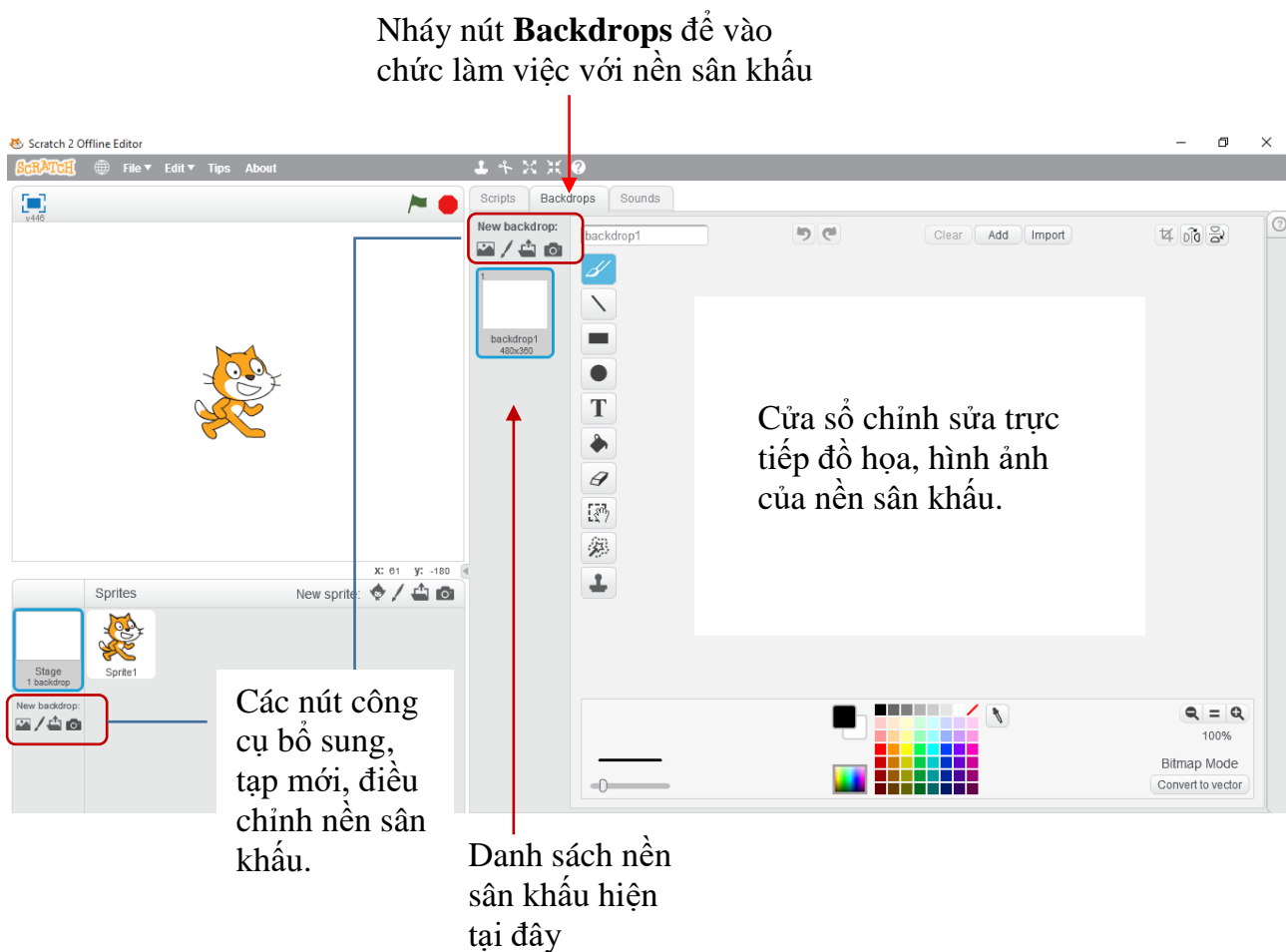


Hình nền sân khấu

Hình ảnh thu nhỏ của sân khấu và các chức năng với sân khấu ở đây


Trong hình trên, nền sân khấu là rỗng. Chúng ta có thể bổ sung thêm nhiều hình nền để đưa lên trang trí cho sân khấu.

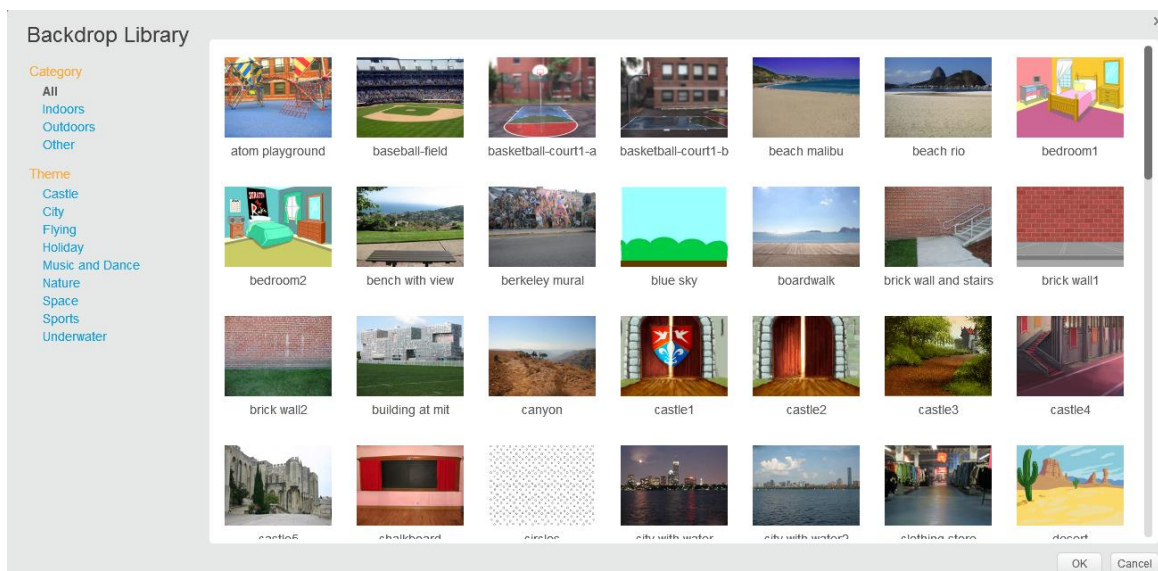
Cửa sổ điều chỉnh, bổ sung nền sân khấu có dạng như hình sau.



2. Bổ sung thêm nền sân khấu vào chương trình

Để bổ sung thêm hình ảnh sân khấu từ thư viện có sẵn, em hãy thực hiện các bước sau.

- Nháy nút  tại khung thông tin sân khấu.
- Xuất hiện cửa sổ các hình nền sân khấu trong thư viện có sẵn.

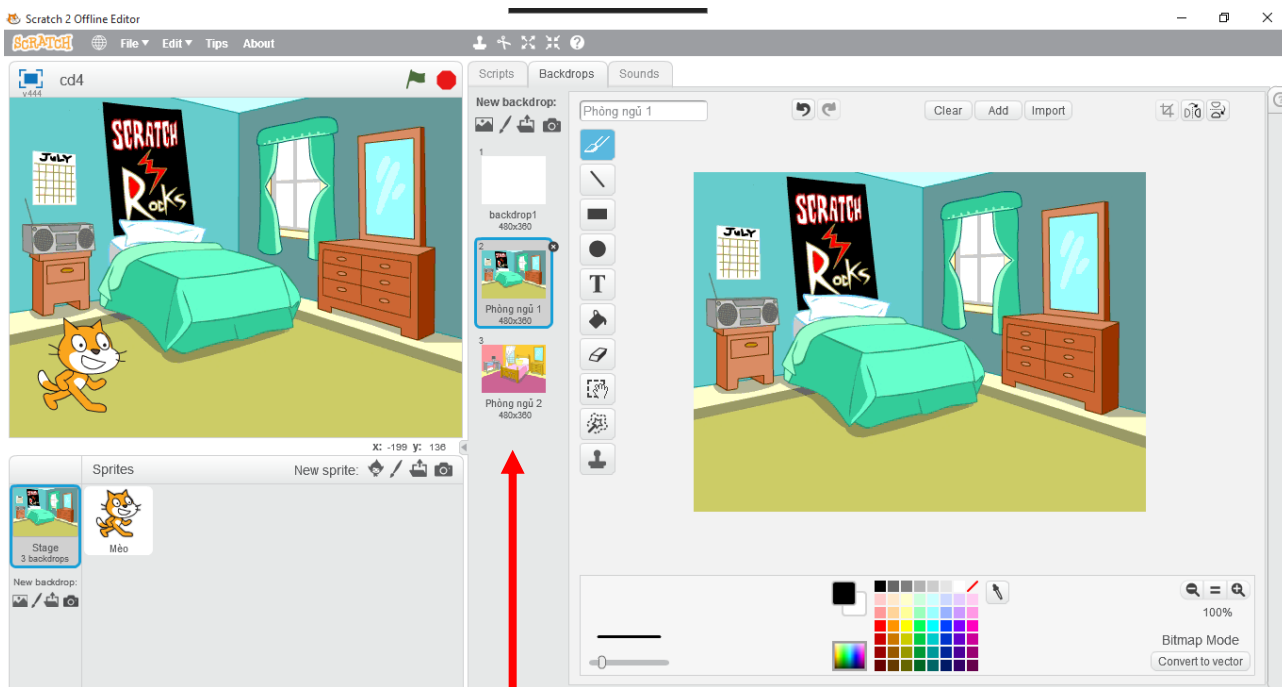


- Nháy chọn hình nền muốn đưa vào chương trình và nháy nút OK.
Hình nền sâu khấu sẽ đưa vào chương trình tương tự hình sau.



Khung điều khiển nhân vật và sân khấu sau khi đã bổ sung thêm nhiều nhân vật và nền sân khấu

Chú ý: khi nháy lên nút sân khấu phía dưới, cửa sổ **Backdrops** như hình sau xuất hiện cho phép chỉnh sửa trực tiếp hình ảnh đồ họa sân khấu. Có thể đặt tên các nền sân khấu trong cửa sổ này.



Dãy các hình nền sân khấu hiện ở

7. Cho nhân vật chào hỏi

Nhóm các lệnh Chuyển động chỉ điều khiển nhân vật di chuyển trên sân khấu, nhưng còn thể hiện bên ngoài của các nhân vật như ăn mặc, nói, hát thì phải dùng các lệnh của nhóm khác.

Chúng ta cùng làm quen với một vài lệnh đơn giản của nhóm lệnh Thể hiện (Look) của Scratch. Bắt đầu từ 1 chương trình như sau:

Chú mèo trước khi xuất phát sẽ chào khán giả: "Chào các bạn, tôi là Mèo con" trong 2 giây. Sau đó Mèo chạy vòng quanh sân khấu 1 vòng và quay trở lại vị trí cũ. Khi về đích, Mèo sẽ chào: "Chúc các bạn ngủ ngon".

Các lệnh được kéo thả vào chương trình như sau:



2 lệnh mới trong đoạn chương trình trên là:

say Chào các bạn, tôi là Mèo con **for 2 secs**

Lệnh này thể hiện nội dung câu nói trên màn hình và dừng toàn bộ chương trình trong 1 thời gian nhất định (ví dụ 2 giây trong ví dụ trên). Hình ảnh thể hiện có ý nghĩa nhân vật đang nói.

say Chúc các bạn ngủ ngon

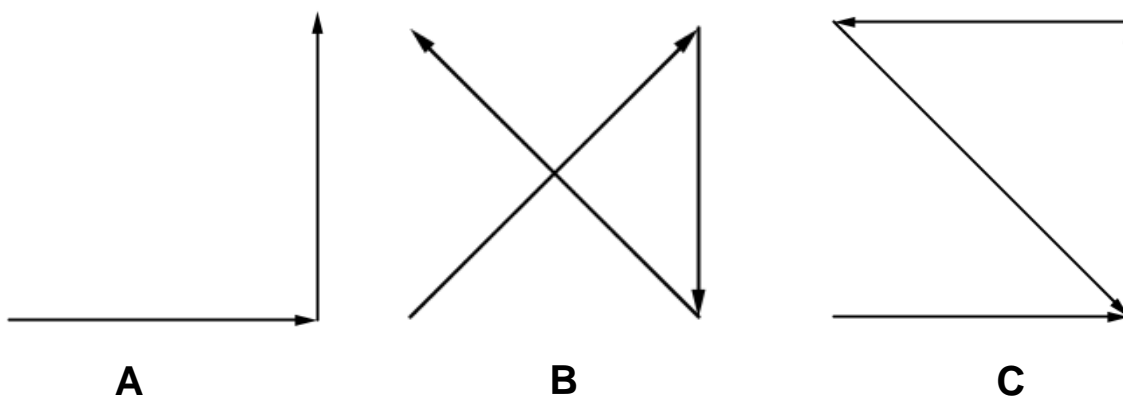
Lệnh này thể hiện nội dung câu nói trên màn hình (không dừng toàn bộ chương trình). Hình ảnh thể hiện có ý nghĩa nhân vật đang nói.

Câu hỏi và bài tập

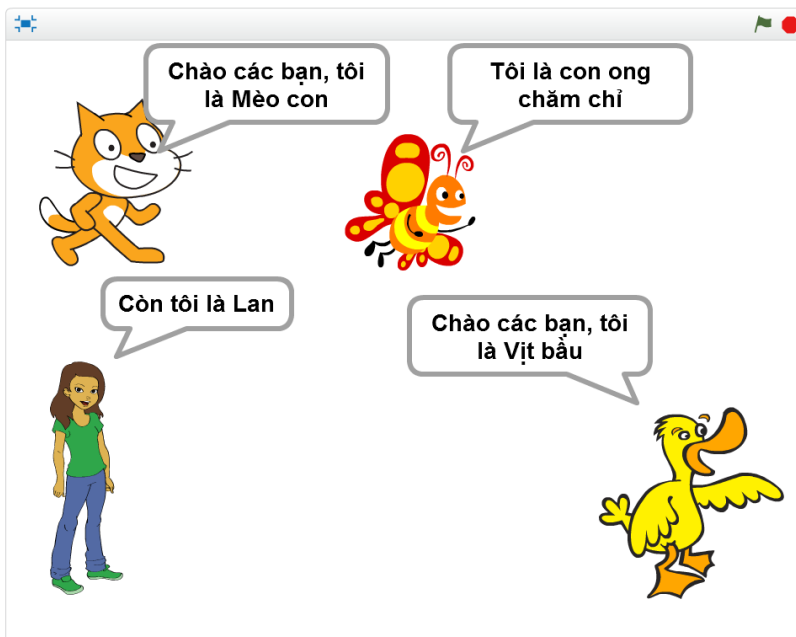
1. Viết chương trình điều khiển nhân vật thực hiện các công việc sau:

- Đi vòng quanh sân khấu 8 vòng theo chiều ngược kim đồng hồ.
- Đi vòng quanh sân khấu 5 vòng theo chiều kim đồng hồ.
- Trước khi đi nói "Chào các bạn, tôi chuẩn bị đi". Sau đó nhân vật sẽ đi vòng quanh sân khấu 4 lần, mỗi lần đi về sẽ dừng lại và nói câu: "Tôi vừa hoàn thành công việc" trong khoảng thời gian 2 giây trước khi đi tiếp.

2. Em hãy thiết lập các chương trình để điều khiển nhân vật chuyển động theo các sơ đồ sau, chú ý đến mỗi ngã rẽ nên dừng lại vài giây để nghỉ ngơi.



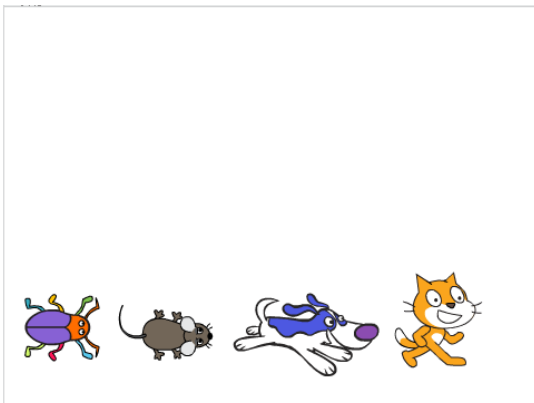
3. Em hãy viết 1 chương trình nhỏ tạo ra 4 nhân vật, khi chạy các nhân vật này chào hỏi như sau.



Mở rộng

Viết chương trình thực hiện các công việc sau:

- Thiết lập 4 nhân vật như hình dưới đây.



- Lăn lượt cho mỗi nhân vật chuyển động quanh sân khấu theo chiều ngược kim đồng hồ 4 lần. Con này chạy sau con kia.

Bài 4. Vẽ hình 1

Mục đích

Học xong bài này bạn sẽ biết:

- Sử dụng khái niệm bút vẽ để điều khiển nhân vật vẽ các đường, hình đơn giản trên màn hình.
- Thay đổi màu sắc, độ rộng, kiểu của nét vẽ.
- Làm quen với khái niệm cảm biến và các lệnh cảm biến đơn giản.
- Biết các câu lệnh điều khiển rẽ nhánh (selection).

Bắt đầu

1. Giả sử có 1 họa sĩ với 1 số bút vẽ, bảng màu và 1 tờ giấy trắng. Họa sĩ bắt đầu vẽ. Trong các từ sau, từ nào liên quan đến công việc vẽ của họa sĩ này:

- Trải giấy ra mặt bàn.
- Nhắc bút.
- Uống nước.
- Hạ bút.
- Hát.
- Nghỉ giải lao.
- Di chuyển bút trên giấy.
- Lia bút, rê bút.
- Chọn màu khác.
- Chọn bút khác.



2. Em đã từng làm quen với các phần mềm tập vẽ trên máy tính chưa? Hãy kể 1 vài chương trình như vậy.

3. Trong bài học này chúng ta sẽ được làm quen và học cách Scratch có thể vẽ được trên màn hình dưới sự điều khiển của chương trình. Chương trình sẽ điều khiển những công cụ sau đây:

- bút vẽ
- bảng màu
- nhắc bút, hạ bút

và nhiều tính năng nổi bật khác.

Hoạt động bài học






1. Chế độ vẽ theo chuyển động nhân vật

Trong Scratch, chế độ vẽ được thiết lập rất đơn giản theo nguyên tắc sau:

- Đặt chế độ hạ bút (pen down).
- Khi đó khi nhân vật chuyển động sẽ tạo ra vết trên đường đi của mình, đó chính là nét vẽ được tạo ra theo màu sắc, kiểu bút có sẵn của hệ thống.
- Trong quá trình vẽ có thể thay đổi màu vẽ, nét bút vẽ.
- Để kết thúc chế độ vẽ cần nhấc bút (pen up).

Các lệnh của nhóm vẽ (pen) đều có màu xanh lá cây.

Chúng ta chú ý các lệnh sau:

- Lệnh thiết lập chế độ vẽ: . Sau lệnh này mọi chuyển động của nhân vật sẽ đều để lại nét vẽ trên đường đi của mình.
- Lệnh hủy chế độ vẽ: . Chú ý sau lệnh này các hình vẽ không bị xóa trên màn hình.
- Lệnh thiết lập màu vẽ: . Cách xác định màu vẽ bởi lệnh này như sau: nháy chuột lên ô vuông của lệnh, sau đó nháy chuột lên 1 vị trí bất kỳ trên màn hình có màu sắc muốn xác định.
- Lệnh thiết lập độ rộng nét vẽ: . Giá trị
- Lệnh xóa tất cả các hình vẽ trên màn hình: .

Em hãy thiết lập 1 chương trình vẽ đơn giản như mô tả dưới đây và quan sát kết quả.

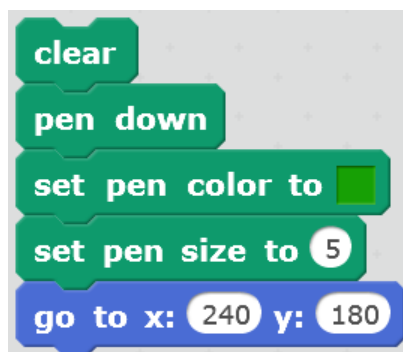
Xóa các hình được vẽ trước đó.

Thiết lập chế độ vẽ: hạ bút.

Đặt màu vẽ là màu: xanh.

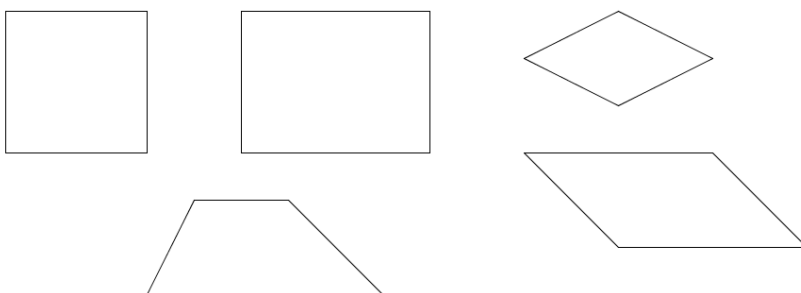
Thiết lập độ rộng nét bút = 5.

Cho nhân vật chuyển động lên vị trí góc phải trên của sân khấu.



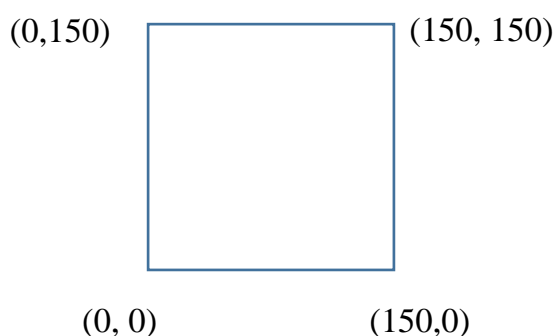
2. Vẽ 1 số hình hình học đơn giản

Các em sẽ cùng thực hành vẽ các hình hình học sau bằng Scratch: hình vuông, chữ nhật, hình thoi, hình bình hành, hình thang.



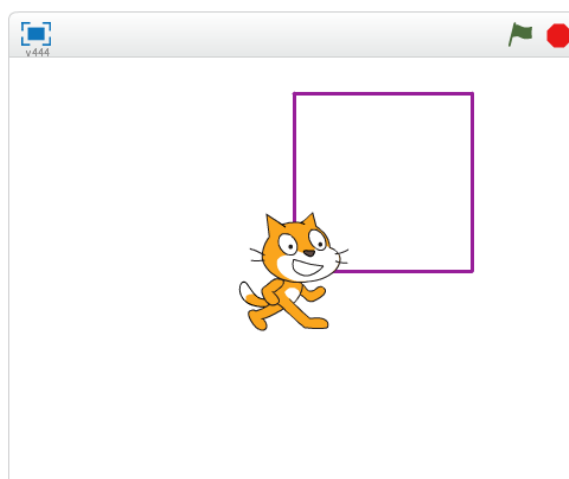
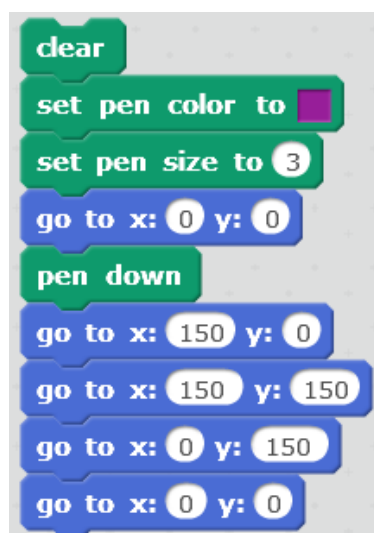
Để thực hiện các chương trình vẽ được các hình trên chúng ta cần xác định tọa độ các đỉnh của các hình này.

Ví dụ với hình vuông, chúng ta xác định tọa độ các đỉnh là:



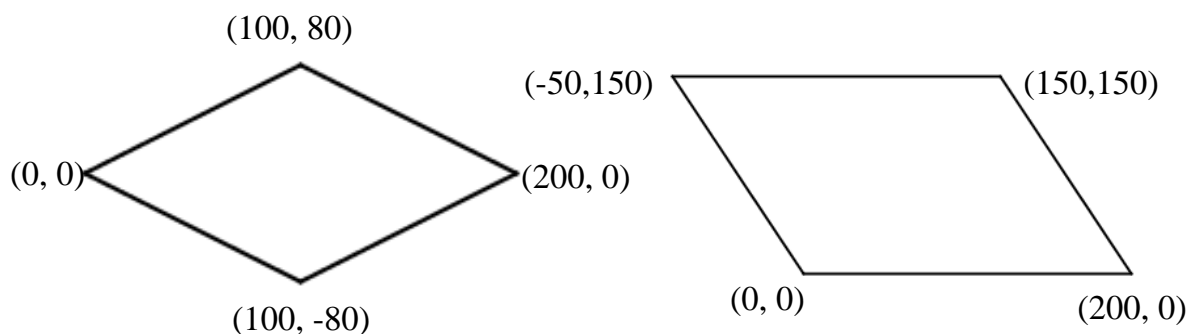
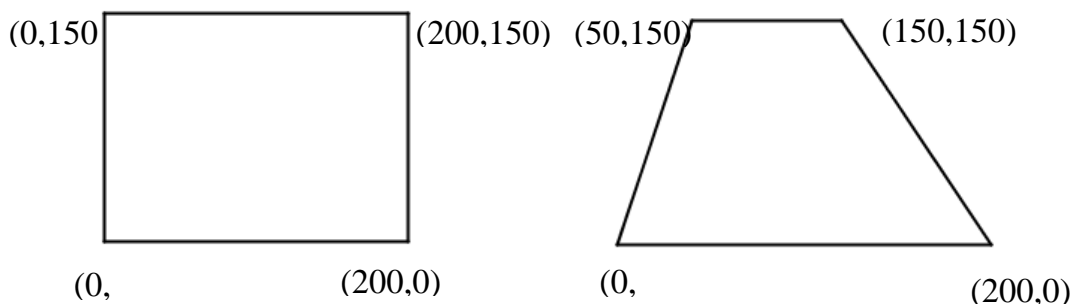
Em viết chương trình điều khiển nhân vật bắt đầu chuyển động từ vị trí (0,0), sau đó lần lượt di chuyển đến các vị trí (150,0), (150,150), (0,150), cuối cùng quay trở lại vị trí ban đầu (0,0).

Chương trình có thể được thiết lập như hình dưới đây. Chú ý rằng lệnh "go to x:0 y:0" được đặt trước lệnh "pen down" chỉ ra rằng chỉ sau khi nhân vật dịch chuyển đến vị trí (0,0) mới bắt đầu quá trình vẽ.





Kết quả của chương trình được thể hiện ở hình phải.

Tương tự chúng ta thiết lập tọa độ của các hình còn lại và tạo chương trình vẽ tương ứng.

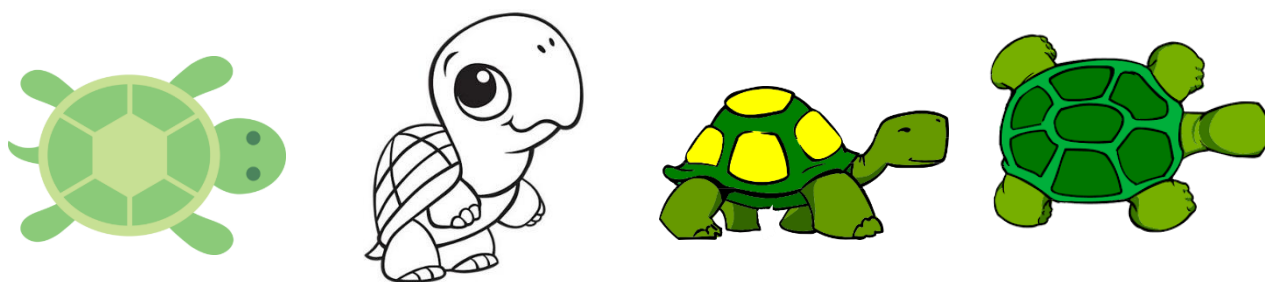


Ẩn nhân vật trong các bài toán vẽ hình

Trong 1 số bài toán vẽ hình nếu không có nhu cầu hiển thị nhân vật thì Scratch cho phép ẩn, không hiển thị (tạm thời) nhân vật bằng lệnh  từ nhóm Hiển thị (Look). Ngược lại lệnh  có tác dụng hiển thị trở lại hình ảnh nhân vật trên màn hình.

Sử dụng hình ảnh con rùa làm nhân vật của bài toán vẽ hình

Logo là phần mềm đầu tiên cho phép lập trình tạo bút vẽ trên màn hình. Trong phần mềm Logo sử dụng hình ảnh con rùa để vẽ. Chúng ta cũng có thể tạo ra các nhân vật hình ảnh Rùa trong các bài tập vẽ hình để tạo ra khung cảnh giống trong phần mềm Logo.



3. Thay đổi màu và nét bút

Quan sát các tính chất sau liên quan đến các tính chất của công việc vẽ.

Tính chất 1



Tính chất 2




Tính chất 3



Em hãy chỉ ra tên của các thuộc tính trong các cột của bảng trên.

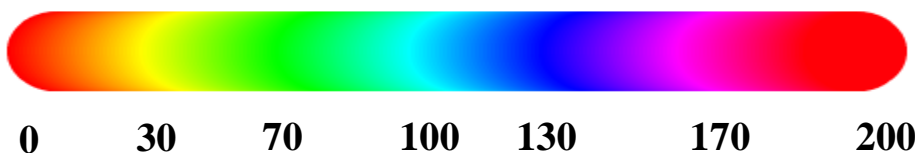
Có 3 thông số sau liên quan đến bút vẽ trong Scratch: màu vẽ (color), độ rộng nét bút (size) và độ mờ nét bút (shade).

Màu bút vẽ (pen color)

Màu sắc của bút vẽ được thiết lập bởi lệnh . Giá trị màu sắc của bút vẽ trong Scratch được đánh số từ 0 đến 200 theo dải màu cầu vồng.



Dải màu theo sắc cầu vồng được thiết lập trong Scratch, lấy giá trị số từ 0




Dải các giá trị màu của bút vẽ từ 0 đến 200.

Ta có bảng giá trị màu ngấn sau:

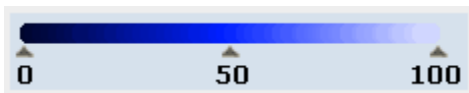
Giá trị	Màu	Giá trị	Màu	Giá trị	Màu
0	Đỏ	70	Xanh lá cây	130	Xanh thẫm
30	Vàng	100	Xa da trời	170	Hồng

Độ rộng nét bút (pen size)

Độ rộng của nét bút được tính bằng pixel (điểm). Lệnh  sẽ thiết lập kích thước của nét bút vẽ hiện thời.

Độ mờ nét bút (pen shade)

Độ mờ (hay độ đậm nhạt) của bút vẽ được đo bởi tham số shade có giá trị từ 0 đến 100. Giá trị 0 là đậm nhất (độ mờ = 0), giá trị = 100 là mờ hoàn toàn.



Lệnh  thiết lập độ mờ của nét bút.

Ngoài 3 lệnh thiết lập thông số chính thức cho màu bút, kích thước và độ mờ nét bút, trong Scratch còn có các lệnh làm thay đổi các giá trị này.



Thay đổi màu của bút theo giá trị được nhập trong lệnh.



Thay đổi kích thước nét bút theo giá trị được nhập trong lệnh.



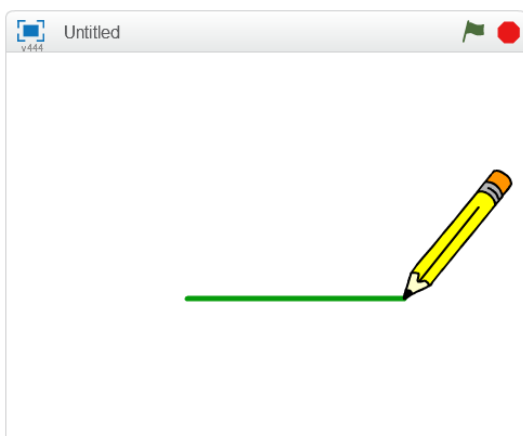
Thay đổi độ mờ bút theo giá trị được nhập trong lệnh.

Em hãy thiết lập các chương trình thực hiện kết quả thay đổi màu sắc, kích thước và độ mờ sử dụng các mẫu lệnh trên.

Em hãy viết lại các đoạn chương trình thể hiện trên màn hình các hình vẽ sau:



4. Thiết lập bút vẽ riêng

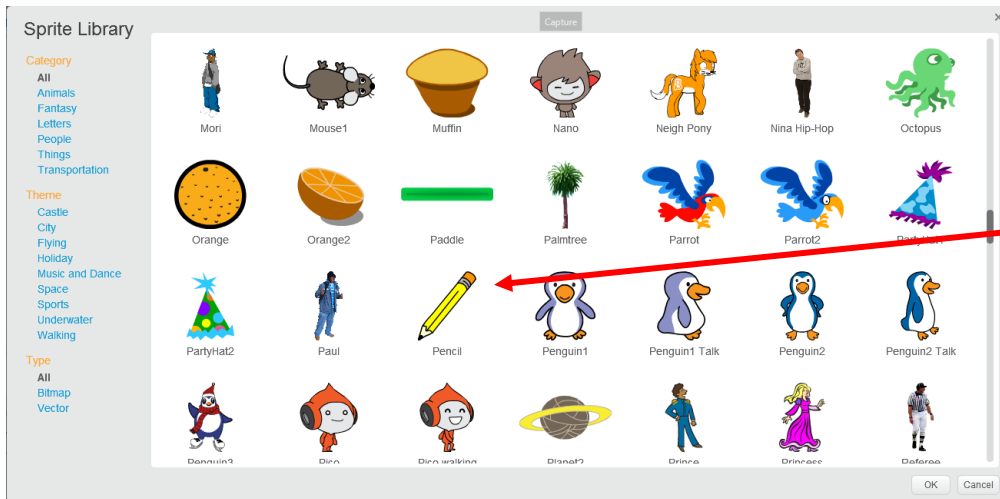


Mục đích của chúng ta trong bài luyện này là thiết lập một bút vẽ (như 1 nhân vật trên sân khấu) và có khả năng vẽ các nét bút đúng tại vị trí đầu bút như hình bên.

Chú ý nét vẽ của bút xuất phát từ chính đầu ngòi bút trong hình.

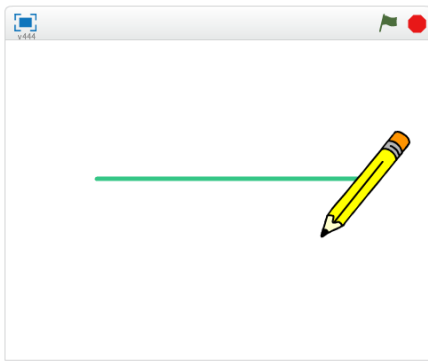
Em hãy thực hiện các bước sau đây:

- Bổ sung thêm 1 nhân vật vào chương trình. Lần này hãy chọn nhân vật bút chì (pencil) trong danh sách để đưa vào sân khấu.



Chọn bút vẽ theo hình ảnh này (pencil).

- Sau đó em có thể xóa nhân vật mèo bằng cách nhấp chuột phải lên mèo tại cửa sổ điều khiển nhân vật và chọn lệnh **delete**.



Bây giờ Bút chì đã hiện như nhân vật duy nhất trên màn hình. Tuy vậy nếu thực hiện lệnh vẽ chúng ta sẽ thấy nét vẽ xuất phát từ điểm giữa của bút chì (xem hình ảnh bên).

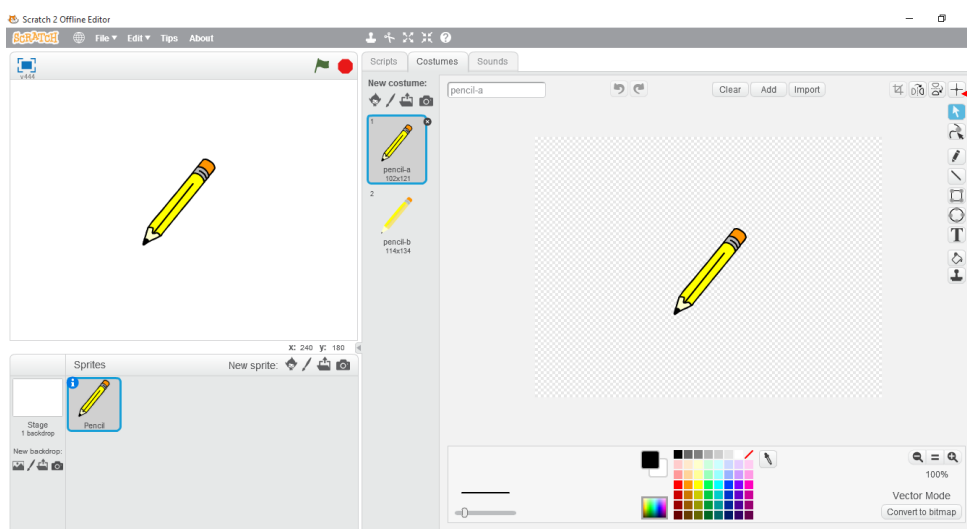
Trong Scratch, nét vẽ sẽ xuất phát từ **tâm** của nhân vật. Do vậy muốn nét vẽ đi qua đầu bút chì chúng ta cần chuyển tâm của hình về vị trí đầu bút chì.

Cách thực hiện như sau:

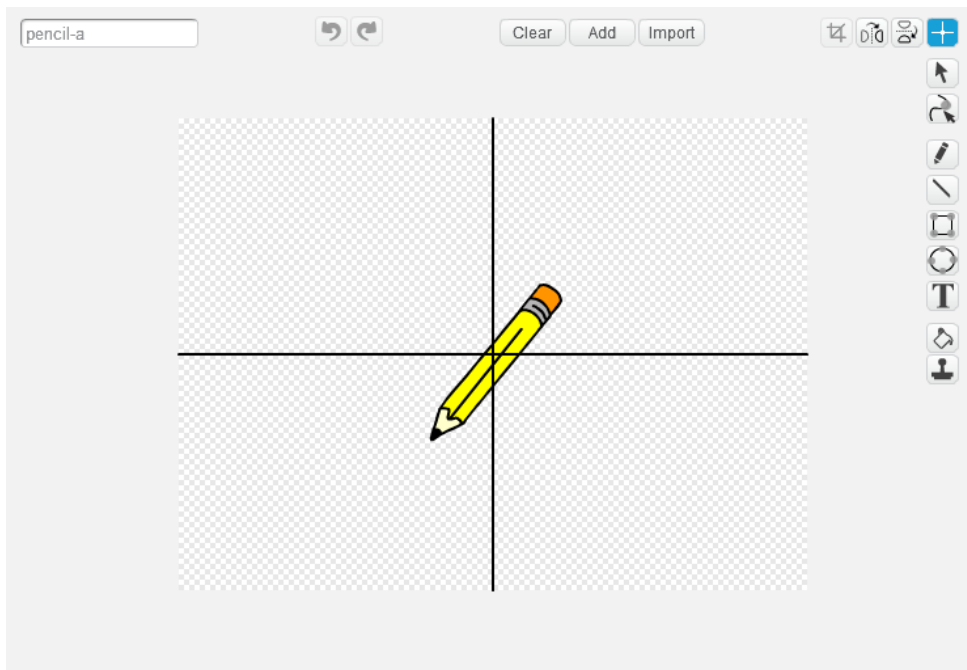
- Nhấp chọn bút chì trong khung điều khiển nhân vật.

- Nhấp lên TAB Costume. Sẽ xuất hiện màn hình chỉnh sửa hình ảnh trang phục của nhân vật hiện thời, cụ thể ở đây là bút chì.

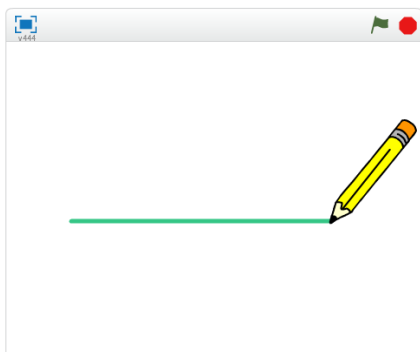
Bên phải là cửa sổ chỉnh sửa đồ họa của nhân vật đang chọn (ở đây là bút chì). Nhiệm vụ của chúng ta là chuyển tâm của nhân vật về đầu bút. Em thực hiện theo các bước sau.



1. Nhấp chuột lên nút có hình dấu cộng (+) này. Đây là chức năng chọn tâm của hình nhân vật.



2. Xuất hiện 2 đường vạch ngang, dọc như hình bên. Em hãy nhấn giữ và rê chuột đến điểm đầu bút và nhả chuột.



3. Công việc điều chỉnh vị trí tâm nhân vật bút chì đã hoàn thành. Em có thể kiểm tra để xem bút sẽ vẽ như thế nào.

Hãy sử dụng bút này để vẽ 1 hình vuông trên màn hình.

5. Vẽ theo điều kiện và sự kiện cảm biến

Những chương trình mà chúng ta đã biết đều là 1 dãy các lệnh bắt đầu bằng lệnh sự kiện



. Lệnh này điều khiển đoạn chương trình gắn phía dưới chạy khi nhấp chuột lên lá cờ.

Trong nhóm các lệnh sự kiện còn có 1 lệnh điều khiển theo sự kiện nhấp lên 1 phím bất kỳ.



Đó là lệnh . Lệnh này có ý nghĩa tương tự lệnh trên, chỉ khác là nhóm các lệnh gắn bên dưới sẽ thực hiện khi phím tương ứng được nhấp.

Chúng ta hãy cùng thiết lập chương trình sau:

Nếu có sự kiện nhấp lên lá cờ: khởi tạo chế độ vẽ đồ họa với các tham số ban đầu.

Nếu sự kiện một trong các phím ← → ↓ được bấm, nhân vật sẽ xoay các góc 90 độ trái, 90 độ phải và 180 độ.

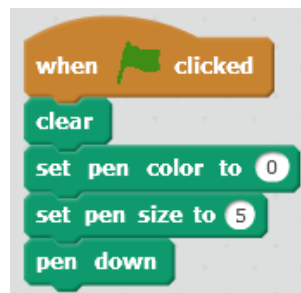
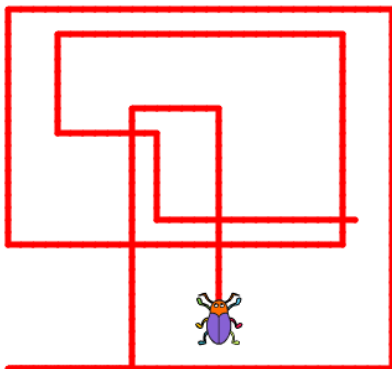
Nếu phím ↑ được bấm, nhân vật sẽ tiến lên phía trước 10 bước.

Chúng ta sử dụng lệnh **when space key pressed**, trong đó phím tương ứng được chọn bằng cách nhấp chuột lên nút hình tam giác và chọn tên phím tương ứng của sự kiện. Trên cửa sổ lệnh của nhân vật này sẽ đặt 5 chương trình như sau.

Nhấp chuột lên vị trí này sẽ hiện ra tất cả các phím trên bàn phím. Lựa chọn 1 để thiết lập lệnh cho sự kiện nhấp phím này.



Thiết lập chương trình như dưới đây và quan sát hoạt động của chương trình.



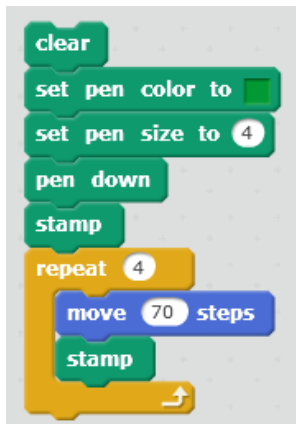
Các đoạn chương trình này được đặt trong cùng 1 cửa sổ lệnh của nhân vật, và được thực hiện đồng thời.



Chú ý: Trong các bài tiếp theo các em sẽ còn được học thêm cách điều khiển bàn phím bằng các công cụ cảm biến khác của Scratch.

6. Lệnh Stamp vẽ hình của nhân vật lên màn hình

Lệnh Stamp có chức năng in hình nhân vật lên màn hình tại vị trí và thời điểm thực hiện lệnh (với điều kiện đang ở chế độ hạ bút).



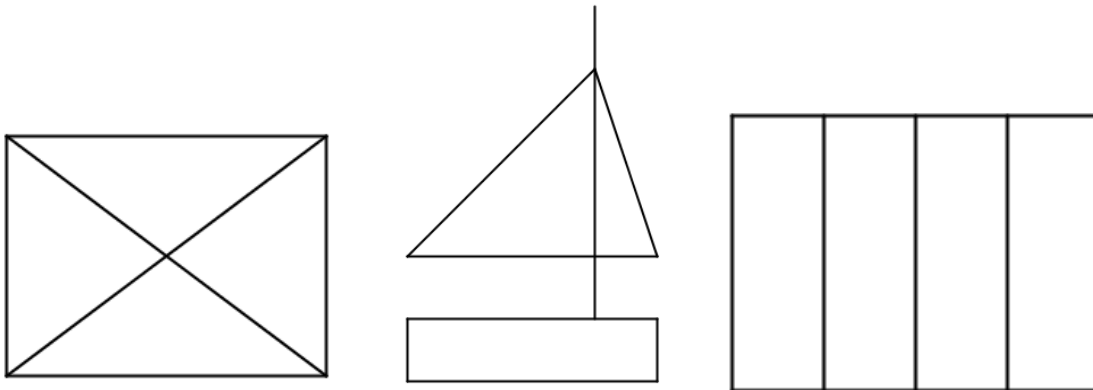
Em hãy thiết lập chương trình để có thể vẽ được hình dưới đây.



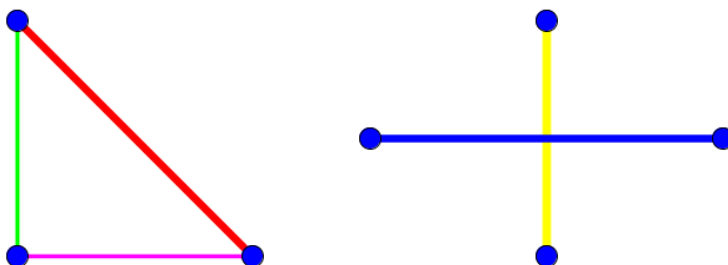
Em cần tạo một nhân vật mới có hình tròn màu xanh đặc, sau đó cần đặt tâm của nhân vật đúng vị trí tâm hình tròn.

Câu hỏi và bài tập

1. Hãy viết chương trình để thực hiện vẽ các hình sau.



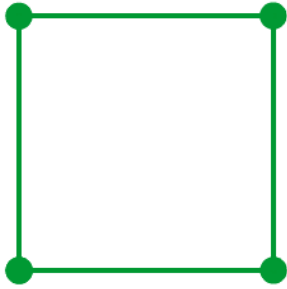
2. Hãy viết chương trình vẽ các hình sau, kết hợp các công cụ màu sắc và stamp.



3. Trong ví dụ của mục 5, em hãy thay đổi và bổ sung thêm yêu cầu sau:

- Khi nhân vật đi sang trái thì màu vẽ sẽ vàng, khi sang phải màu vẽ là xanh lá cây, đi lên với màu đỏ và đi xuống với màu xanh thẫm.

4. Sử dụng lệnh **Stamp** để viết chương trình vẽ hình sau:



Mở rộng

1. Xây dựng chương trình có các chức năng sau.

- Có 2 nhân vật là bút chì và quả bóng hình tròn.
- Khi chạy chương trình, cả 2 bút chì và quả bóng đều vẽ lên 1 hình vuông ở 2 bên màn hình.
- Bút chì kẻ hình vuông màu vàng, quả bóng tạo hình vuông màu xanh.

Bài 5. Âm thanh 1


Mục đích

Học xong bài này, bạn sẽ biết:

- Kết nối âm thanh với nhân vật. Nhân vật nói và thể hiện lời nói bằng chữ và âm thanh.
- Các lệnh tạo âm thanh, đánh trống và chơi nhạc trên nền sân khấu.
- Điều khiển nhân vật nhảy múa.

Bắt đầu

1. Chúng ta đã biết lệnh say có ý nghĩa như thế nào.

Lệnh  sẽ làm cho mèo con kêu meo meo như sau



Tuy vậy chúng ta chỉ nhìn thấy mèo kêu mà không nghe thấy giọng nói của mèo. Vậy có cách nào để nghe được giọng của mèo, hay của bất kỳ một nhân vật nào khác hay không?

2. Trong bài học này, em sẽ được làm quen và học được các lệnh liên quan đến âm thanh. Có rất nhiều điều thú vị đang ở phía trước.

- Âm thanh, giọng nói có thể phát ra từ nhân vật hay không? Nhân vật sẽ "nói" như thế nào?
- Nhạc nền của sân khấu có thể bật lên được hay không?
- Nhân vật của chúng ta có thể hát được không?
- Nhân vật có thể chơi các nhạc cụ, đánh trống, thổi kèn được hay không?

Nhưng câu hỏi như vậy em sẽ được lần lượt làm quen trong bài học này.

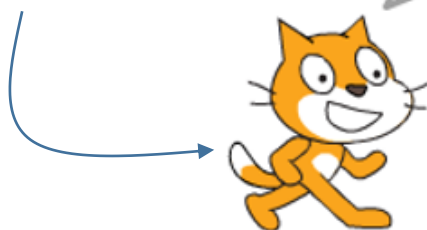
Nội dung bài học

1. Nhân vật có thể nói, hát

Chúng ta hãy bắt đầu từ nhóm lệnh Sound (âm thanh) và thực hiện lệnh trong nhóm lệnh này.



Chương trình ngắn sau đây sẽ cho chúng ta cảm giác nhân vật có thể nói, hội thoại và giao tiếp bằng âm thanh trong Scratch.



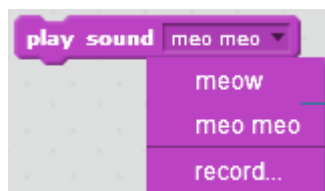
Em sẽ nghe và nhìn thấy gì?

Em sẽ nghe thấy tiếng kêu meo meo của Mèo và nhìn thấy dòng chữ Meo meo trên màn hình.

Như vậy các nhân vật trong Scratch không những có thể "nói" bằng chữ trên màn hình, mà còn có thể hát, nói bằng âm thanh thật của mình thông qua loa của máy tính.

Mỗi nhân vật trong Scratch đều có thể có một hay nhiều âm thanh, giọng nói đi kèm. Số lượng âm thanh của mỗi nhân vật là không hạn chế.

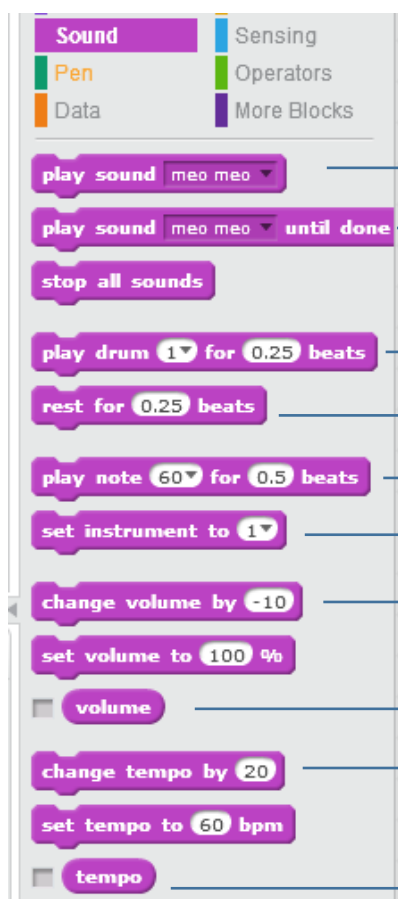
Lệnh **play sound** cho phép nhân vật thể hiện âm thanh của mình trực tiếp trên máy tính.



Trong lệnh play sound cho phép hiển thị và lựa chọn âm thanh tương ứng của nhân vật. Chọn âm thanh cần thể hiện cho nhân vật này.

2. Nhóm lệnh âm thanh

Nhóm lệnh âm thanh (sound) có màu tím sẫm. Các lệnh làm việc với âm thanh rất đa dạng, chúng ta sẽ bắt đầu bằng nhóm lệnh đơn giản đầu tiên: các lệnh bật, tắt âm thanh của nhân vật.



Các lệnh bật âm thanh (sound) của nhân vật (hoặc sân khấu)

Các lệnh đánh trống và liên quan đến nhịp trống

Các lệnh chơi nhạc và liên quan đến nhạc cụ

Các lệnh làm việc với cường độ âm thanh

Các lệnh làm việc với nhịp trống

Em hãy thử và phân biệt sự khác biệt khi thực hiện của các nhóm lệnh sau:



Lệnh đầu tiên phát âm thanh "meo meo", ngay sau đó thực hiện lệnh **say** mà không cần đợi âm thanh kết thúc.




Lệnh đầu tiên phát âm thanh "meo meo" và đợi cho đến khi âm thanh kết thúc mới thực hiện lệnh **say**.



Lệnh **say** được thực hiện trước, sau đó đến lệnh **play sound**.

Em hãy thực hiện cả 3 nhóm lệnh trên và trả lời các mệnh đề sau đúng hay sai.

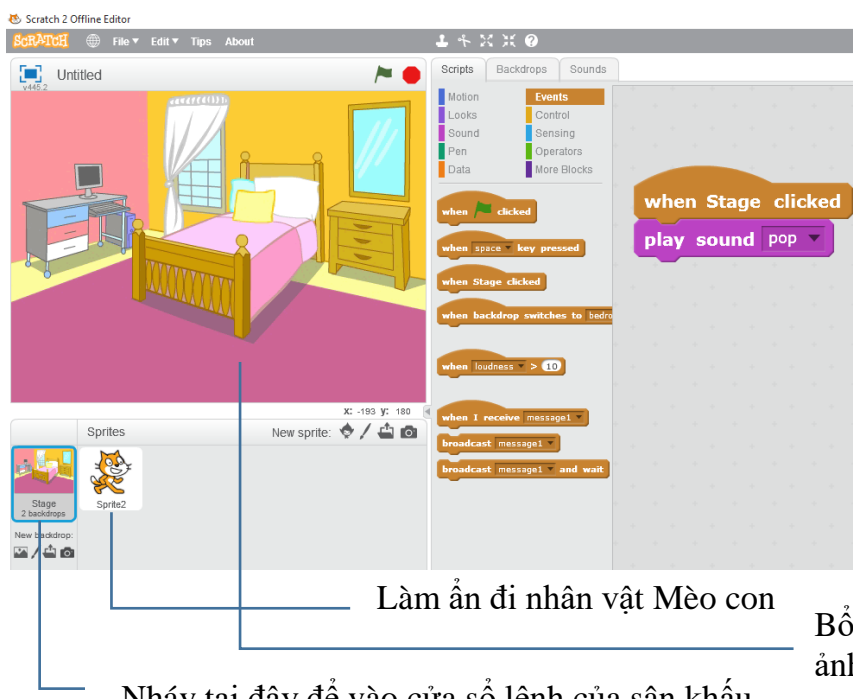
1. Cả 3 đoạn lệnh trên có tác dụng hoàn toàn giống nhau.
2. Các đoạn lệnh số 1 và 3 có tác dụng giống nhau.
3. Lệnh **play sound** sẽ thực hiện bật âm thanh có trong lệnh này đồng thời với các lệnh tiếp theo của lệnh này.
4. Lệnh **play sound** sẽ thực hiện bật âm thanh có trong lệnh, chờ nghe xong âm thanh sẽ thực hiện các lệnh tiếp theo.
5. Lệnh **play sound .. until done** sẽ thực hiện bật âm thanh có trong lệnh, chờ nghe xong âm thanh sẽ thực hiện các lệnh tiếp theo.

Chú ý: Lệnh  sẽ dừng tất cả các âm thanh đang bật trong chương trình.

3. Âm thanh, nhạc nền sân khấu

Âm thanh không chỉ có ở nhân vật, mà sân khấu cũng có thể có âm thanh, nhạc nền của riêng mình. Chúng ta hãy thực hiện một hoạt động đơn giản sau để hiểu ý nghĩa của âm thanh sân khấu.

- (a) Em hãy bổ sung nền sân khấu mới từ kho các hình ảnh đã có của phần mềm, ví dụ, lấy hình ảnh sau (bedroom).
- (b) Làm ẩn đi nhân vật chính là chú mèo.
- (c) Nháy chọn biểu tượng sân khấu trong khung điều khiển phía dưới và kéo thả đoạn chương trình như trong hình ảnh sau.



Kéo thả 2 lệnh này từ cửa sổ lệnh của sân khấu.

Làm ẩn đi nhân vật Mèo con

Bổ sung thêm hình ảnh sân khấu này.

Nháy tại đây để vào cửa sổ lệnh của sân khấu.

Chúng ta chú ý đến đoạn chương trình ngắn của sân khấu này.



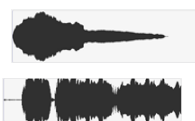
Lệnh này nằm trong nhóm sự kiện (event) của sân khấu. Các lệnh bên dưới sẽ thực hiện mỗi khi nháy chuột lên sân
Bật âm thanh của sân khấu được ghi trong lệnh

Khi chạy chương trình này, em sẽ thấy mỗi khi nháy chuột lên sân khấu sẽ nghe được 1 âm thanh nhỏ phát ra.

Trên thực tế mỗi nhân vật và sân khấu nền của 1 chương trình đều có thể gắn với một hay nhiều âm thanh khác nhau. Các âm thanh này rất đa dạng, có thể là lời nói, tiếng động, nhạc nền, bài hát. Các tệp âm thanh có thể dùng bao gồm các dạng wav, mp3,



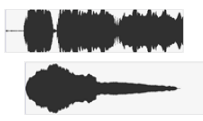
Sân khấu có các âm thanh, nhạc nền riêng của mình.



Nhân vật có âm thanh, nhạc, giọng nói riêng.



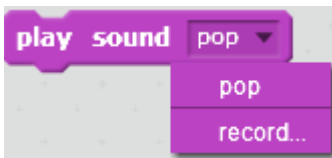
Số lượng âm thanh cho mỗi nhân vật là không hạn chế.




4. Thu âm trực tiếp âm thanh cho nhân vật

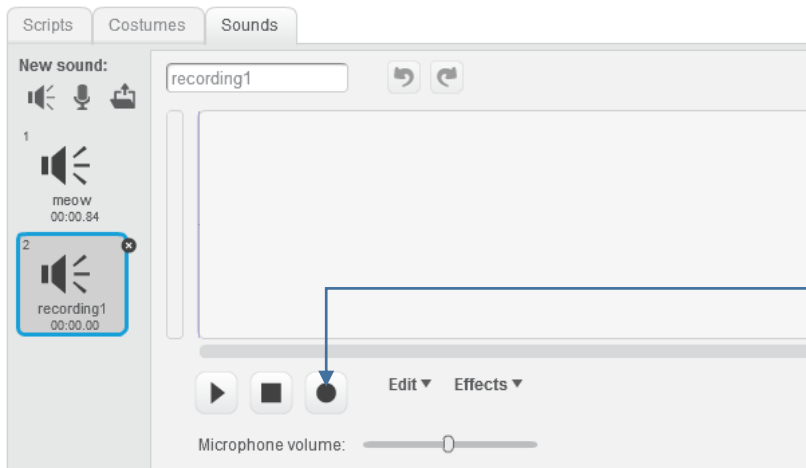
Trong hoạt động này, em sẽ thực hiện việc thu âm trực tiếp từ máy tính qua micro để tạo ra một âm thanh của nhân vật. Các bước thực hiện như sau.

(1) Lựa chọn **record** trong lệnh play sound của nhân vật.



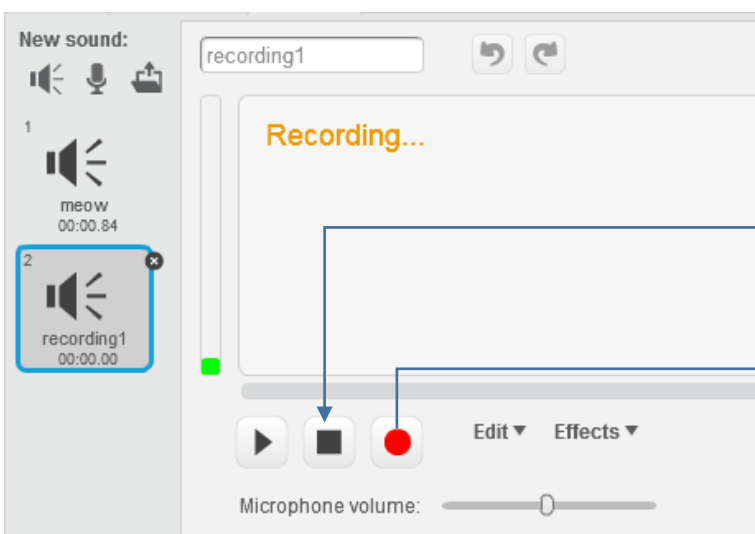
hoặc vào TAB **Sounds** của khung điều khiển và nháy lên biểu tượng micro .

Cửa sổ sau sẽ xuất hiện để chuẩn bị thu âm trực tiếp.



Nháy nút này để bắt đầu thu âm.

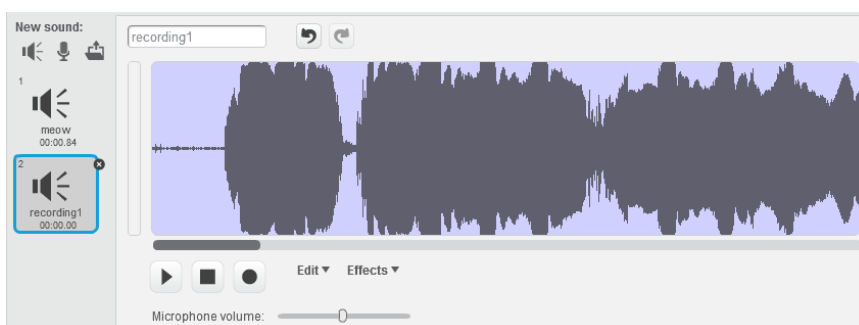
Trong quá trình đang thu âm, cửa sổ này có dạng như sau:



Nháy nút này để kết thúc thu âm

Trong thời gian thu âm, nút này chuyển màu đỏ.

Khi kết thúc thu âm, âm thanh vừa được thu sẽ hiển thị trên cửa sổ như sau:



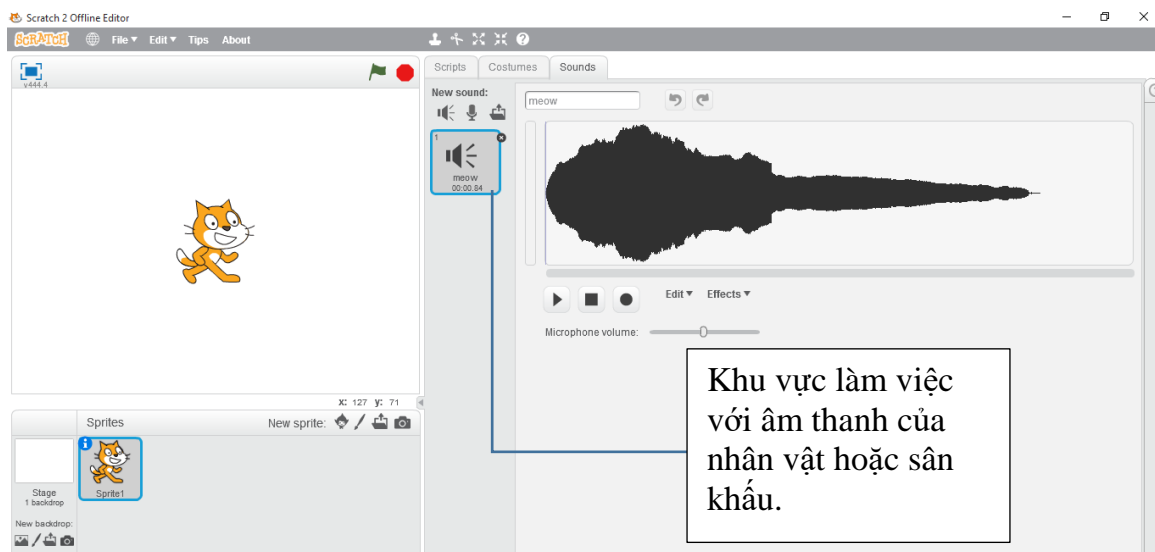
Bây giờ có thể thực hiện việc đặt tên cho âm thanh vừa thu âm.

5. Bổ sung âm thanh cho nhân vật và sân khấu

Trong hoạt động này, em sẽ thực hiện thao tác bổ sung âm thanh cho nhân vật (hoặc sân khấu) lấy từ 1 kho âm thanh có sẵn của phần mềm.

Cửa sổ làm việc với âm thanh của nhân vật hoặc sân khấu

Như chúng ta đã biết, trong Scratch, các nhân vật có thể nói, hát, phát ra tiếng nói của mình. Sân khấu cũng có thể có âm thanh nền. Trong màn hình Scratch nếu nhấn lên nút Sound chúng ta sẽ vào chức năng cho phép bổ sung thêm âm thanh cho nhân vật. Cũng trên màn hình này sẽ hiện toàn bộ danh sách các âm thanh hiện đang có của nhân vật. Với các âm thanh có sẵn chúng ta có thể thực hiện các thao tác sau:




- Chỉnh sửa âm thanh.
- Xóa âm thanh.
- Thay đổi tên âm thanh.

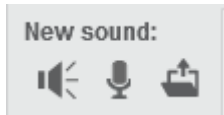


Mỗi nhân vật có thể đi kèm 1 hoặc nhiều âm thanh, em có thể bổ sung âm thanh từ các kho có sẵn, có thể thu âm trực tiếp hoặc lấy từ các tệp âm thanh trên máy tính.

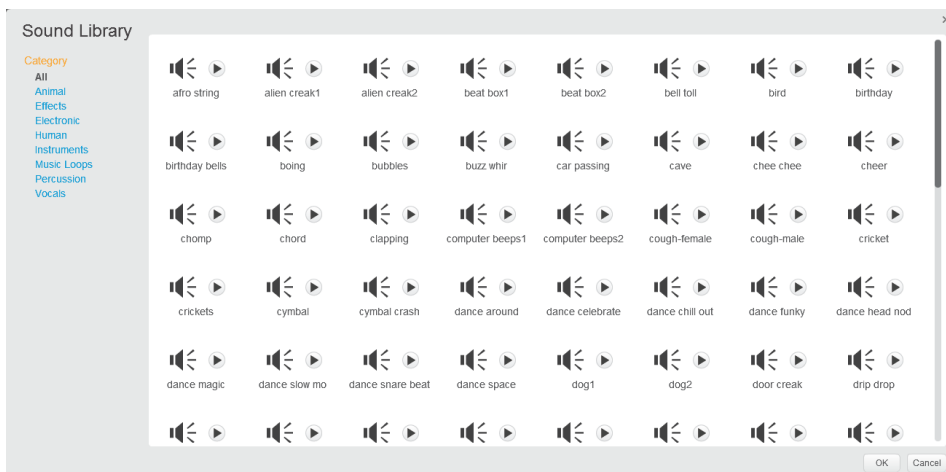
Em hãy thực hiện thao tác bổ sung âm thanh cho nhân vật bằng cách lấy từ 1 kho có sẵn.

Bổ sung âm thanh từ kho có sẵn

- Nháy vào nút hình loa  ở phía trên khung làm việc với âm thanh của nhân vật.

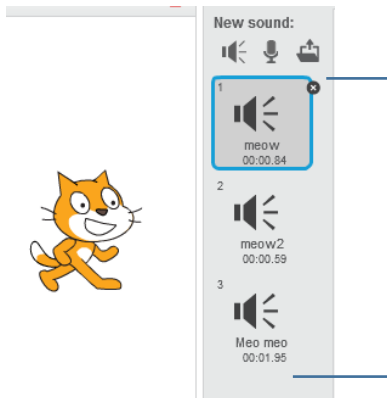


- Xuất hiện cửa sổ có danh sách các âm thanh.




- Em nháy chọn 1 âm thanh và nháy nút OK.

Em sẽ thấy âm thanh này đã được đưa vào khu vực âm thanh của nhân vật.

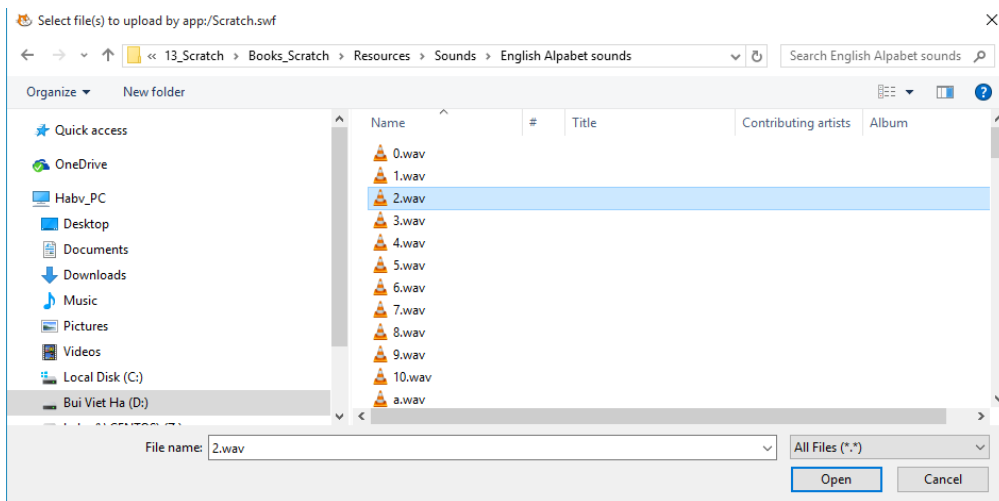


Danh sách các âm thanh của nhân vật hiển thị tại đây.

Bổ sung âm thanh từ tệp ngoài

- Nháy nút .

- Xuất hiện hộp hội thoại mở tệp có dạng tương tự hình sau. Tìm kiếm tệp âm thanh ngoài.



- Chọn tệp âm thanh muốn bổ sung và nhấn nút **Open**. Tệp âm thanh này sẽ được đưa vào danh sách âm thanh của nhân vật (hoặc sân khấu).

Câu hỏi và bài tập

1. Thiết lập nhân vật là thầy giáo, thu âm lời nói "Chào em" và viết đoạn chương trình thầy giáo nói và thể hiện dòng chữ "Chào em" trên màn hình.



2. Thiết lập nhân vật là học sinh, thu âm lời nói "Em chào thầy ạ" và viết đoạn chương trình em học sinh chào thầy giáo, nói và thể hiện dòng chữ "Em chào thầy" trên màn hình.



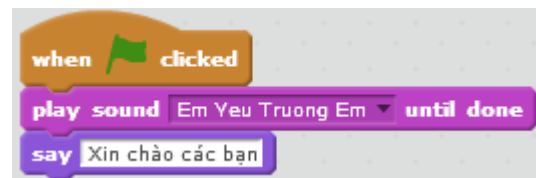
3. Thiết lập 2 nhân vật là thầy giáo và học sinh. Thu âm giọng nói của thầy "Chào em", giọng nói học sinh "Em chào thầy ạ" và thực hiện thiết kế chương trình như sau:

- Thầy nói "Chào em" và thể hiện lời chào trên màn hình.
- Sau 1 giây, học sinh sẽ chào lại "Em chào thầy ạ" và hiện dòng chữ trên màn hình.



4. Thiết lập chương trình như sau:

- Nền sân khấu là hình ảnh một ngôi trường.
- Nhân vật chính là hai bạn học sinh nhỏ đang đi học.
- Em bổ sung bài hát "Em yêu trường em" cho nhân vật hai bạn nhỏ này.
- Cho chương trình chạy thì hát bài hát trên. Khi hát xong thì hai bạn nhỏ sẽ chào mọi người.



Mở rộng

1. Em hãy viết 1 chương trình đơn giản có âm thanh với các yêu cầu sau.

- Nhân vật chính là 1 em bé.
- Em bé sẽ chạy vòng quanh sân khấu, mỗi bước chạy sẽ vọng lên tiếng thịch thịch của bàn chân.

2. Viết chương trình sau:

- Màn hình nền trắng, có 1 cái bút chì là nhân vật chính.
- Khi chạy chương trình, bút bắt đầu vẽ các hình đa giác đều trên màn hình, bắt đầu là tam giác, tứ giác và ngũ giác đều.
- Khi nháy chuột lên màn hình thì lời 1 bài hát quen thuộc vang lên. Em có thể tìm chọn một bài hát bất kỳ.

Bài 6. Chuyển động 2

Mục đích

Học xong bài này, bạn sẽ biết:

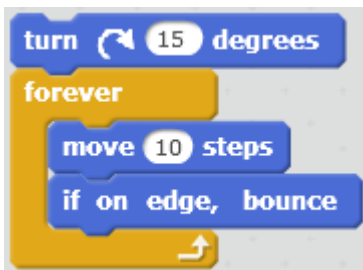
- Các lệnh chuyển động phức tạp hơn trên màn hình.
- Giao tiếp đơn giản giữa nhân vật và máy tính.
- Sử dụng lệnh lặp đơn giản và lặp vô hạn.
- Bước đầu làm quen với lệnh điều khiển if ... then ...
- Làm việc với 2 hoặc nhiều nhân vật. Chương trình song song.

Bắt đầu

1. Quan sát chương trình sau, bạn có nhận xét gì về chuyển động của nhân vật.




2. Quan sát tiếp chương trình sau.



- Bạn có nhận xét gì về chuyển động của nhân vật? Có điểm gì giống và khác nhau so với chương trình trên.

Nội dung bài học

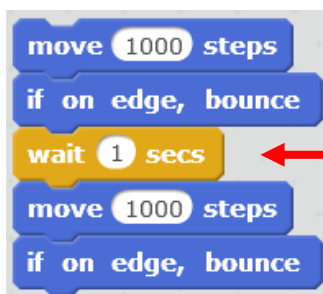
1. Điều khiển nhân vật khi chạm biên màn hình

Có bao giờ em nghĩ rằng nếu chúng ta cho các nhân vật chuyển động ra khỏi phạm vi màn hình thì sẽ như thế nào không? Hoạt động này sẽ giúp em điều khiển nhân vật tránh được hiện tượng trên. Trong Scratch có 1 lệnh riêng như vậy, đó là lệnh ,

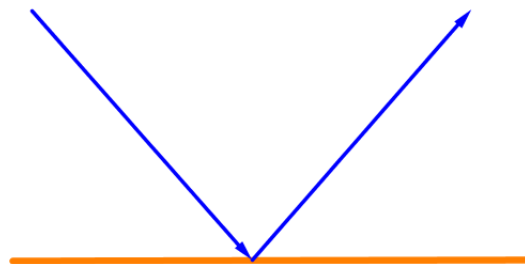
Lệnh này có tác dụng như sau: khi nhân vật chuyển động chạm cạnh sân khấu, nhân vật sẽ tự động dừng và quay lại theo hướng đối xứng gương với hướng ban đầu. Hướng ban đầu và hướng quay được chỉ ra trong hình bên.

Chú ý: lệnh này chỉ có trong Scratch và thường được đặt sau các chuyển động của nhân vật để điều khiển chuyển động này.

Em hãy thử đoạn chương trình sau để kiểm tra tác dụng của lệnh trên.



Lệnh **wait (1) secs** có tác dụng cho nhân vật dừng lại trong 1 thời gian để quan sát được dễ hơn.



Khi nhân vật chuyển động gặp cạnh sân khấu, nhân vật sẽ dừng lại và quay hướng theo góc đối xứng với hướng lúc đi đến.

2. Lệnh lặp vô hạn

Chúng ta đã được làm quen với lệnh trong nhóm điều khiển dùng để làm cho 1 đoạn chương trình thực hiện lặp lại 1 số lần (lệnh repeat). Nếu chúng ta muốn 1 đoạn chương trình được lặp lại vô hạn lần thì cần sử dụng lệnh dưới đây.

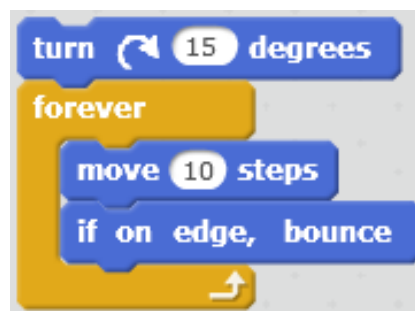


Nhóm các lệnh nằm trong khung này sẽ được lặp lại vô hạn lần cho đến khi người sử dụng nhấp nút đỏ trên màn hình hoặc có lệnh cho phép dừng vòng lặp hoặc chương trình.

Bây giờ thì em hãy thử lại và hiểu hai chương trình đã được nêu trong phần mở đầu, em có thể trả lời được câu hỏi đặt ra.



Nhân vật chuyển động liên tục, vô hạn dọc theo các cạnh của 1 hình vuông.



Nhân vật chuyển động liên tục, vô hạn, mỗi khi gặp cạnh sân khấu thì quay lại và đi tiếp.

3. Hội thoại giữa 2 nhân vật, chương trình song song

Trong hoạt động này chúng ta cùng thiết kế 1 chương trình đơn giản sau.

Nhân vật: Mèo và Cánh cụt.

Nội dung chương trình cần thực hiện.

Mèo (trông thấy Chim) chào "Xin chào bạn Cánh cụt".

Chờ trong 3 giây.

Cánh cụt (thấy Mèo chào) chào "Chào Mèo con".

Em hãy phân tích yêu cầu của đầu bài và viết ra các công việc cần thực hiện của Mèo và Cánh cụt.



Công việc của Mèo	Công việc của Cánh cụt

Em cần thực hiện các bước sau:

- Bổ sung thêm nhân vật chim cánh cụt như hình trên và đổi tên 2 nhân vật thành "Mèo" và "Cánh cụt".
- Trong cửa sổ lệnh của Mèo và Cánh cụt, em viết 2 chương trình riêng biệt cho Mèo và Cánh cụt như hình dưới đây. Chú ý: hai chương trình sẽ thực hiện đồng thời, song song, nên cần để câu chào của Mèo trong 5 giây, còn Cánh cụt thì cần chờ đợi 3 giây đúng theo yêu cầu, sau đó mới chào Mèo.



- Kết quả thực hiện chương trình.



Em hãy bổ sung thêm các lệnh để mở rộng yêu cầu của bài toán trên.

Mèo (trông thấy Chim) chào "Xin chào bạn Cánh cụt".

Đồng thời Mèo tiến 10 bước về phía Cánh cụt.

Chờ trong 3 giây.

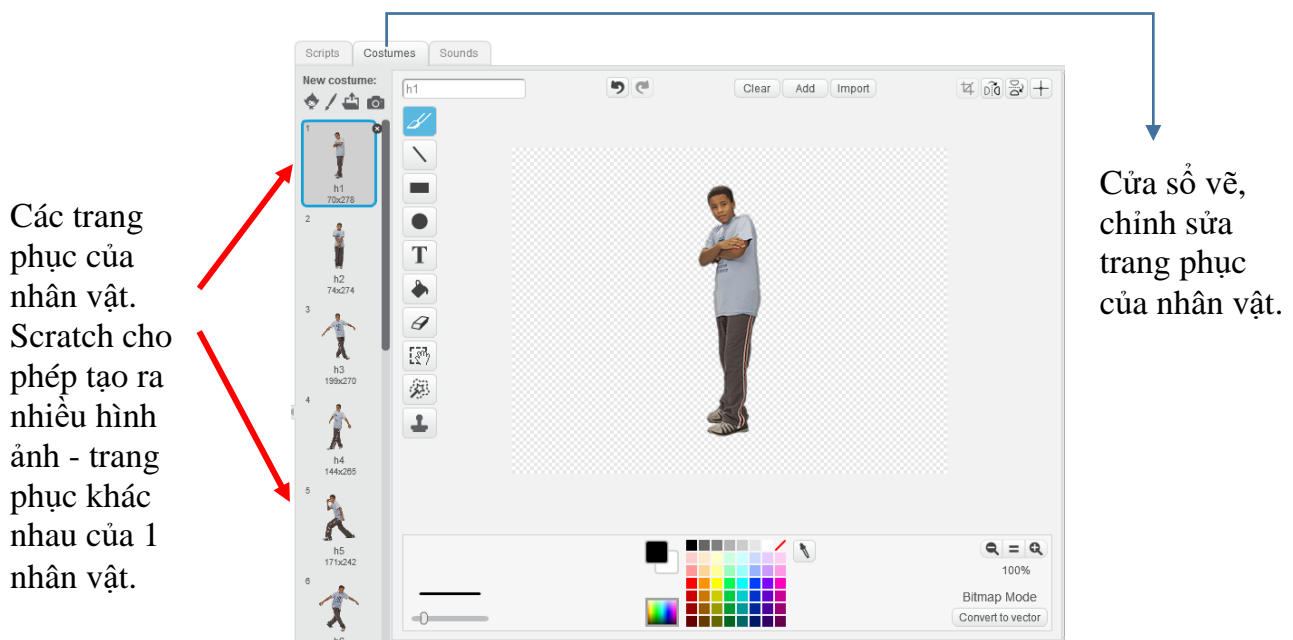
Cánh cụt quay lại phía Mèo.

Cánh cụt chào "Chào Mèo con".

4. Chuyển động bằng thay đổi trang phục

Để mô tả chuyển động của nhân vật, ngoài các lệnh cho phép nhân vật dịch chuyển vị trí trên màn hình, chúng ta còn có thể sử dụng hiệu ứng thay đổi hình ảnh nhân vật.

Mỗi nhân vật sẽ có thể có nhiều hình ảnh minh họa, được gọi là trang phục. Nút Costume cho phép em bổ sung và chỉnh sửa các hình ảnh trang phục này. Ví dụ nhân vật Hip-hop sau có đến 12 bộ trang phục. Như vậy kết hợp thay đổi trang phục và dịch chuyển trên màn hình sẽ tạo ra các hình ảnh chuyển động thực của nhân vật. Phim hoạt hình cũng được thiết kế dựa trên nguyên tắc này.



Trong nhóm lệnh **Thể hiện (Look)** có 2 lệnh điều khiển trang phục sau.

Lệnh **next costume** có tác dụng cho nhân vật chuyển sang trang phục tiếp theo trong danh sách.

Lệnh **switch costume to** có tác dụng cho nhân vật chuyển sang trang phục được chọn trong danh sách.

Bây giờ em hãy thiết kế đoạn chương trình cho nhân vật Hip-hop của chúng ta vừa dịch chuyển trên màn hình (dùng lệnh move), vừa thay đổi trang phục (dùng lệnh next costume).



Chương trình 1.

- Vòng lặp: 10 lần.
- Nhân vật thay đổi trang phục sau mỗi 10 bước.
- Sau khi thay trang phục thì dừng chương trình 1 giây.

Quan sát chương trình trên, em có nhận xét gì về chuyển động của chú bé Hip-hop?

Bây giờ chúng ta sẽ làm tốt lên chương trình trên bằng cách tăng số vòng lặp lên 100, nhân vật sẽ thay đổi trang phục chỉ sau 5 bước và thời gian chờ chỉ là 0.5 giây. Hiệu ứng chuyển động sẽ tốt lên.



Chương trình 2.

- Vòng lặp: 100 lần.
- Nhân vật thay đổi trang phục sau mỗi 5 bước.
- Sau khi thay trang phục thì nghỉ nhanh 0.5 giây.

5. Lệnh điều khiển có điều kiện

Trong hoạt động này em sẽ làm quen với 1 lệnh quan trọng của Scratch, lệnh điều khiển có điều kiện if ... then (nếu thì). Đây là 1 lệnh, 1 tư duy quan trọng của việc điều khiển máy tính làm việc.

Trên thực tế chúng ta thường rất hay gặp tình huống mà hành động được quyết định tùy theo các điều kiện cụ thể. Ví dụ:

- **Nếu** trời nắng **thì** tôi sẽ đi chơi.
- **Nếu** bạn gặp không làm được bài **thì** tôi sẽ giúp bạn.

Mệnh đề trên được thể hiện tổng quát thành câu lệnh sau:

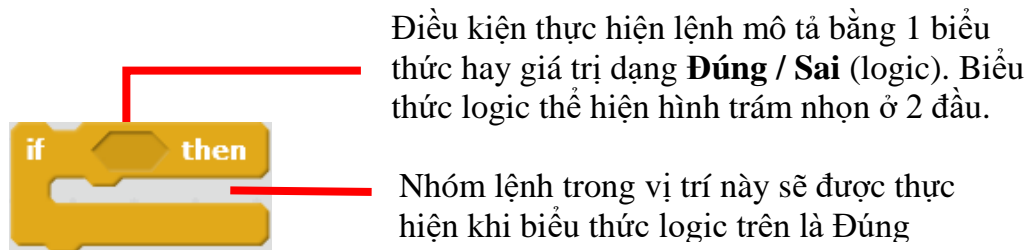
Nếu <điều kiện> **thì** <hành động>

Ở đây <điều kiện> là 1 biểu thức, hoặc mệnh đề luôn có giá trị Đúng hoặc Sai. Các mệnh đề, biểu thức chỉ mang ý nghĩa đúng, sai được gọi là biểu thức logic.

Ví dụ các biểu thức logic.

1 = 2 Nhận giá trị sai. 15 > 3.4 Nhận giá trị đúng.

Lệnh thực hiện lệnh theo điều kiện (logic) là lệnh If then, lệnh này nằm trong nhóm lệnh Điều khiển (Control) có màu nâu.



Trong Scratch có 1 số lệnh là các mệnh đề logic, chỉ có giá trị đúng và sai. Chúng ta cùng tìm hiểu 2 lệnh này trong nhóm lệnh Cảm biến (Sensing).

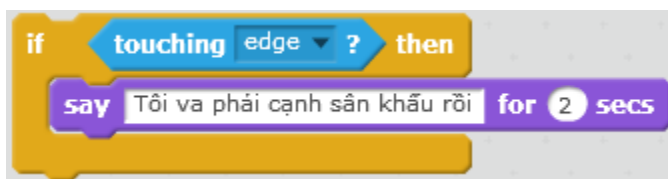


Lệnh này đóng vai trò 1 điều kiện logic. Lệnh chỉ trả lại giá trị Đúng nếu nhân vật hiện thời va chạm với 1 đối tượng khác chỉ ra trong lệnh (nhân vật khác hoặc cạnh sân khấu).

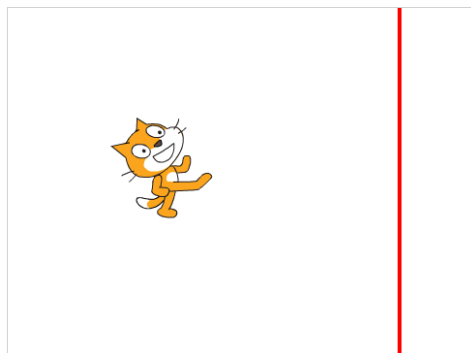


Lệnh này chỉ trả lại giá trị Đúng nếu nhân vật hiện thời tiếp xúc với màu sắc được chỉ ra ngay trong lệnh.

Ví dụ đoạn chương trình sau kiểm tra xem nhân vật có chạm vào cạnh sân khấu hay không, nếu đúng thì sẽ thông báo "Tôi va phải cạnh sân khấu rồi".



Bây giờ em hãy thực hiện chương trình với nhiệm vụ sau.



Vẫn là bài toán cho chú Mèo chuyển động tự do, vô hạn trên màn hình, nếu gặp cạnh sân khấu thì quay lại (xem ví dụ mục 1). Tuy nhiên bổ sung thêm yêu cầu sau:

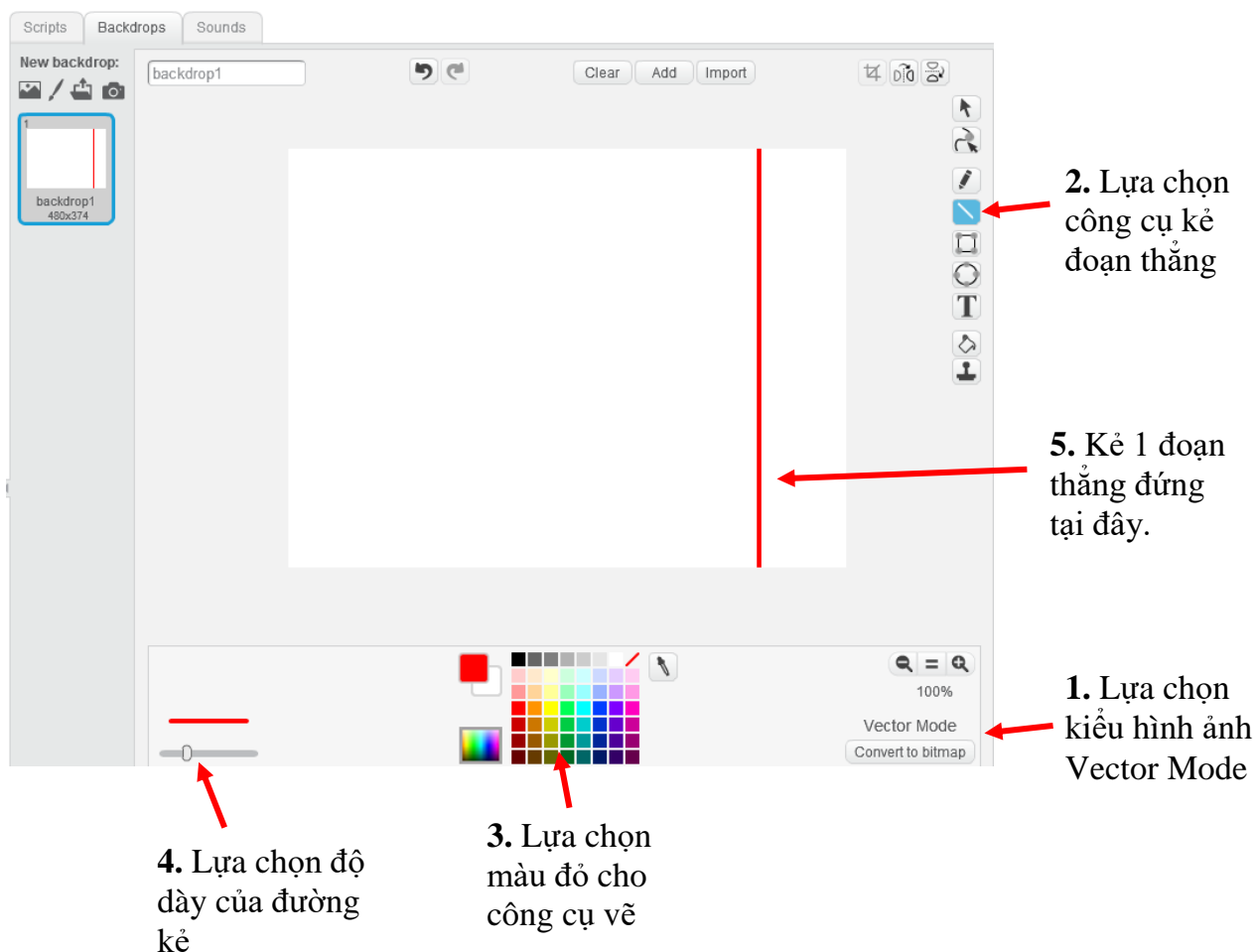
Trên sân khấu vẽ thêm 1 vạch đứng màu đỏ.

Mèo chỉ được phép chuyển động tự do ở vùng bên trái vạch đỏ. Nếu chạm vạch đỏ sẽ lập tức quay sáng trái với góc nghiêng 45 độ.

Em hãy cùng thực hiện chương trình này theo các bước sau.

1) Chỉnh sửa sân khấu, bổ sung thêm 1 đường thẳng đứng màu đỏ.

Em hãy nhấn chọn biểu tượng sân khấu, nhấn tiếp nút Backdrops, khi đó màn hình chỉnh sửa hình ảnh sân khấu như hình dưới đây. Em hãy thực hiện tiếp theo sơ đồ dưới đây.



2) Bây giờ em thiết kế lại chương trình đã có trong mục 1.

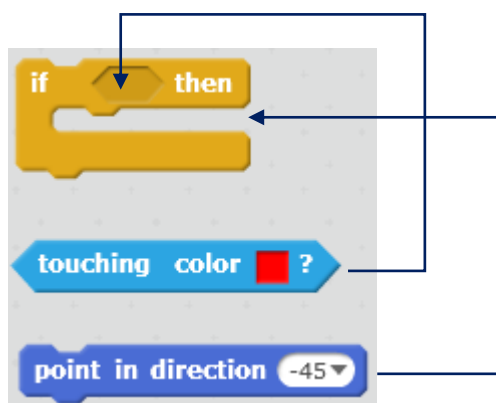


3) Trong vòng lặp này, em cần đưa vào thêm 1 số lệnh để điều khiển nhân vật nếu va chạm với đoạn thẳng đứng màu đỏ thì phải quay lại. Đoạn chương trình này có dạng sau:

Nếu <nhân vật tiếp xúc với màu đỏ> thì

Quay lại theo hướng 45 độ

Em kéo thả ra màn hình lệnh của nhân vật các lệnh sau, sau đó điền thông số và kéo thả chúng vào đúng vị trí.



Để thiết lập màu đúng của lệnh - biểu thức logic này em làm như sau:

- Nháy chuột lên vị trí ô vuông chọn màu của lệnh.
- Di chuyển chuột ra sân khấu và nháy lên đoạn thẳng đứng. Màu của đoạn thẳng này sẽ tự động điền vào biểu thức của lệnh này.

4) Kết quả em thu được đoạn chương trình sau:



5) Chạy chương trình và kiểm tra kết quả.

Câu hỏi và bài tập

1. Em hãy mô tả kết quả của đoạn chương trình sau:



2. Mở rộng đoạn chương trình trong mục 4, cho cậu bé Hip-hop chuyển động vô hạn từ trái qua phải, gặp cạnh sân khấu thì quay lại và cứ như vậy tiếp tục.

3. Các biểu thức logic sau là đúng hay sai?

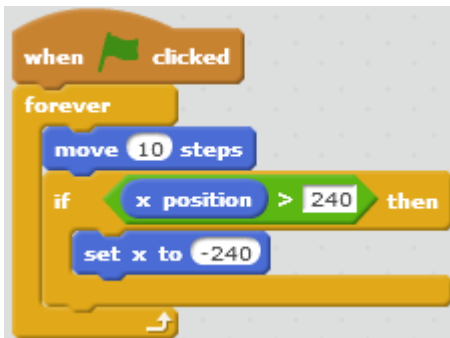
A. $4 < 5$.

B. $5 + 7 - 2 > 10$.

C. $2^3 = 3^2$

D. $\sqrt{2} > 1.4$

4. Em hãy chạy đoạn chương trình sau và giải thích kết quả chương trình.



Mở rộng

1. Thiết kế 1 trò chơi đơn giản

Em hãy thiết kế 1 trò chơi đơn giản "Thả bóng" theo các yêu cầu sau:

Nhân vật: quả bóng và thanh ngang.

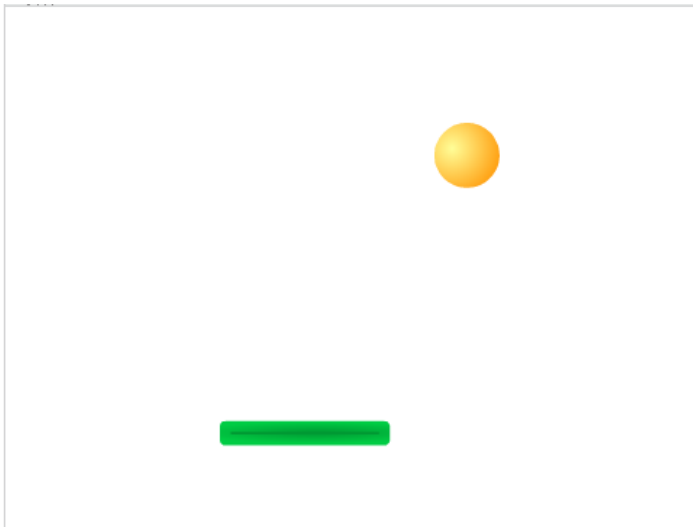
Quả bóng sẽ rơi tự do từ trên trần xuống. Nếu gặp cạnh sân khấu sẽ tự động bật lại và chuyển động liên tục.

Nếu quả bóng chạm thanh ngang, nó sẽ bật lên 1 góc 15 độ và chuyển động ngược lên trên.

Người dùng điều khiển dùng các phím phải, trái để điều khiển quả bóng sao cho bóng không bị rơi xuống phía dưới.

Gợi ý cách thực hiện.

Bước 1. Thiết lập 2 nhân vật **Bóng** và **Thanh ngang** trên sân khấu. Các nhân vật này có thể chọn từ kho các hình ảnh có sẵn của phần mềm.



Bước 2. Thiết lập chương trình điều khiển thanh ngang bằng các lệnh sự kiện khi bấm phím phải, trái.



Bước 3. Thiết lập chương trình cho Bóng. Có thể hình dung công việc của Bóng như sau:
Quay hướng xuống dưới.

Lặp vô hạn lần

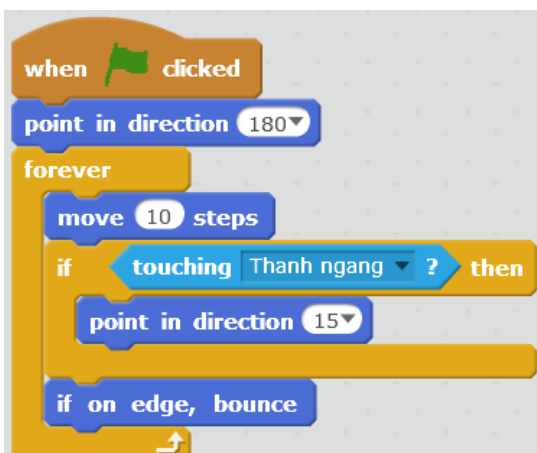
 dịch chuyển 10 bước (về phía trước)

 Nếu tiếp xúc với màu đỏ thì

 quay hướng lên trên 15 độ

 Nếu gặp cạnh sân khấu, quay lại.

Đoạn chương trình có dạng sau.



2. Làm tốt lên trò chơi này

Em hãy làm tốt trò chơi này bằng cách bổ sung yêu cầu sau:

Nếu để Bóng rơi xuống phía dưới thì dừng trò chơi.

Chú ý sử dụng các gợi ý sau:

- Có thể vẽ thêm 1 đường ngang phía dưới để kiểm soát khi Bóng rơi xuống.
- Sử dụng lệnh Stop All từ nhóm lệnh Điều khiển để dừng chương trình.

Bài 7. Vẽ hình 2

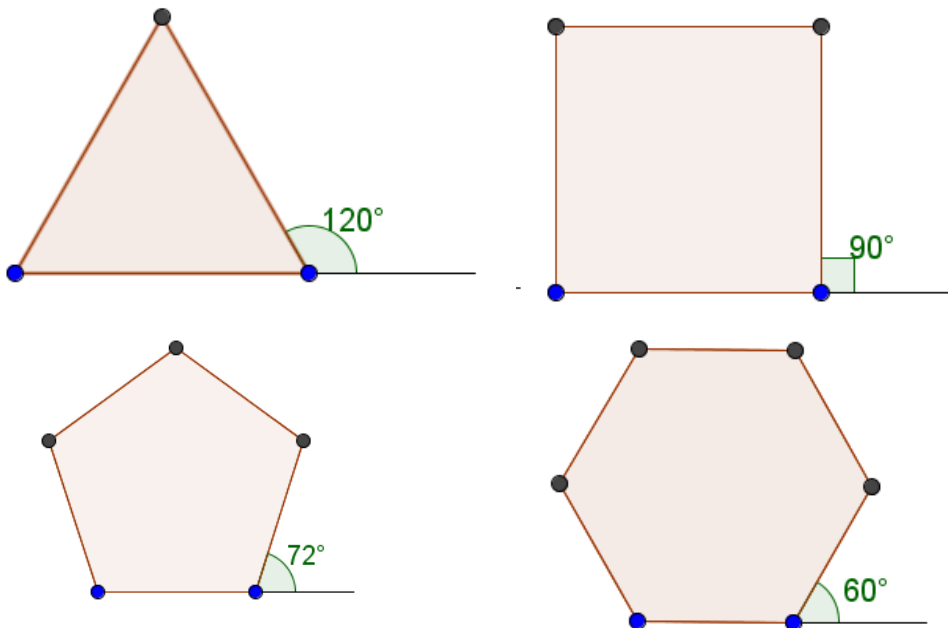
Mục đích

Học xong bài này, bạn có thể:

- Thực hiện vẽ hình phức tạp hơn: hình tròn, đa giác đều, xoắn ốc, hình nghệ thuật.
- Kết hợp màu sắc để vẽ hình nghệ thuật.
- Lệnh lặp có định, lặp có điều khiển.

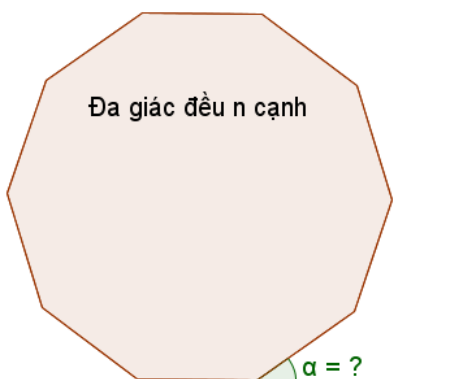
Bắt đầu

1. Quan sát các hình đa giác đều sau.



- Em có nhận xét gì về số đo các góc trong và ngoài của các đa giác này, nếu liên hệ với số cạnh của đa giác?

2. Em có thể tổng quát cách tính góc α với trường hợp đa giác đều n cạnh như hình dưới đây?

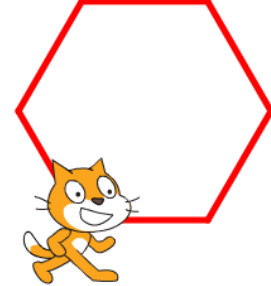
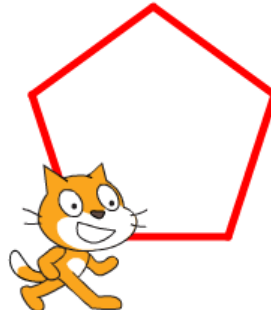
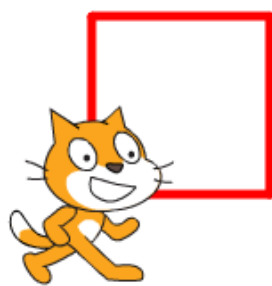
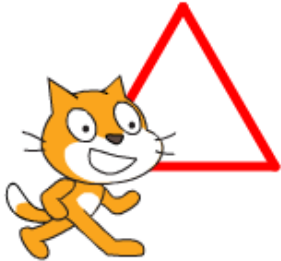


3. Em hãy trình bày theo cách riêng của em cách vẽ hình đa giác đều bằng Scratch.

Nội dung bài học

1. Thực hành vẽ 1 số hình đa giác đều

Dựa trên quan sát trong phần mở đầu, em hãy viết các chương trình đơn giản sau để vẽ các hình tam giác, tứ giác, ngũ giác và lục giác đều.



```
clear
pen down
set pen color to 0
set pen size to 5
repeat 3
  move 100 steps
  turn 120 degrees
```

```
clear
pen down
set pen color to 0
set pen size to 5
repeat 4
  move 100 steps
  turn 90 degrees
```

```
clear
pen down
set pen color to 0
set pen size to 5
repeat 5
  move 100 steps
  turn 72 degrees
```

```
clear
pen down
set pen color to 0
set pen size to 5
repeat 6
  move 100 steps
  turn 60 degrees
```

2. Tìm quy luật của cách vẽ đa giác đều

Nếu quan sát kỹ hơn các chương trình trên chúng ta sẽ tìm ra được quy luật (hay còn gọi là thuật toán) của các chương trình này.

```
repeat 5
  move 100 steps
  turn 72 degrees
```



Số cạnh đa giác

Chiều dài cạnh đa giác

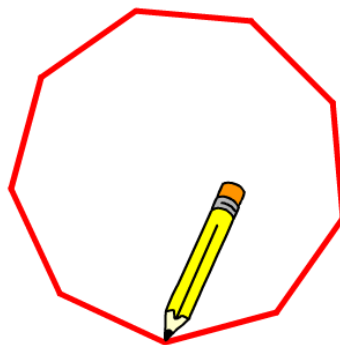
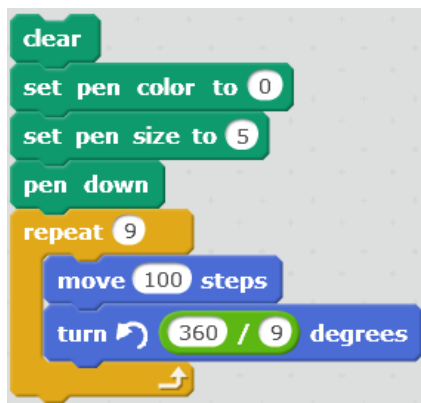
Giá trị này = $360 / \text{số cạnh đa giác}$

Chú ý: nếu số cạnh lớn thì có thể điều chỉnh độ dài cạnh để hình vẽ không vượt quá giới hạn sân khấu.

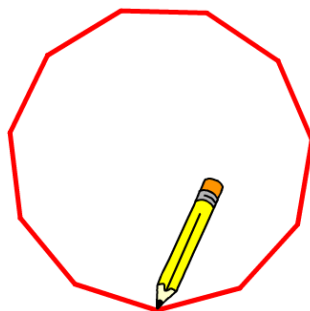
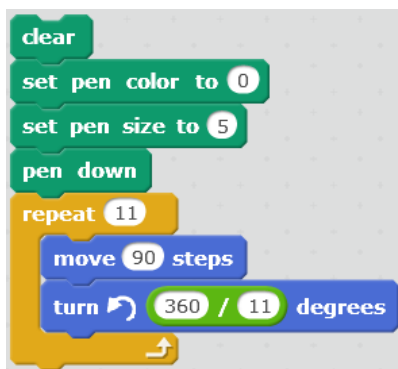
Em hãy áp dụng quy luật trên để vẽ các đa giác đều 9 và 11 cạnh, sử dụng nhân vật là bút chì.

Để thực hiện phép chia $360 / \text{<số cạnh>}$ em hãy dùng lện (toán tử) phép chia  từ nhóm lệnh **Tính toán (Operators)** màu xanh lá cây. Ví dụ nếu vẽ đa giác đều 9 cạnh chúng ta nhập các giá trị 360 và 9 vào toán tử này như sau: .

Với hình 9 cạnh, độ dài cạnh 100, em phải điều chỉnh vị trí ban đầu của bút chì sao cho hình nằm trong khung màn hình.



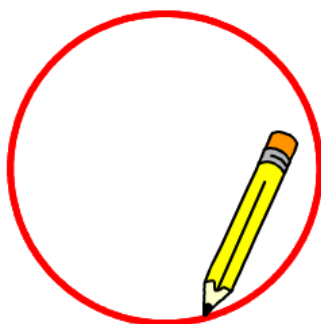
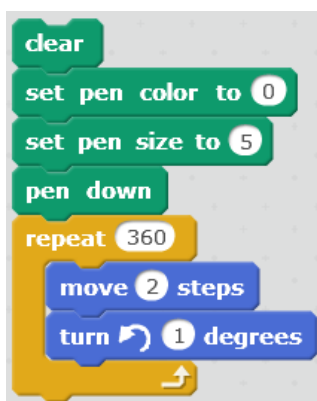
Với hình 11 cạnh, bắt buộc phải làm ngắn lại độ dài cạnh đa giác.



Dựa trên qui luật này, em hãy viết chương trình vẽ các hình đa giác đều có 7, 8, 10 cạnh.

3. Vẽ hình tròn

Để vẽ được hình tròn chúng ta hình dung như 1 đa giác đều n cạnh nhưng với n khá lớn. Em hãy thực hiện chương trình vẽ hình tròn theo cách vẽ 1 đa giác đều với số cạnh lớn.



Trong chương trình này, em dùng thuật toán đối với vẽ đa giác đều với số cạnh = 360, độ dài cạnh = 2.

Chú ý:

1. Em có thể không cần thiết lập số cạnh lớn như vậy. Có thể chỉ cần vẽ với số cạnh nhỏ hơn, ví dụ 180, hoặc thậm chí 90, vẫn thu được hình tròn.
2. Muốn vẽ vòng tròn nhỏ em có thể chọn độ dài cạnh là các số bé hơn 1, ví dụ 0.5.

Luyện tập 1: hãy vẽ hình sau



Gợi ý: Trước khi thực hiện đoạn chương trình vẽ vòng tròn, cần vẽ 1 đoạn thẳng từ vị trí hiện thời của nhân vật đến vị trí bắt đầu đường tròn. Thực hiện công việc này bằng lệnh

move 10 steps

thông thường. Ví dụ có thể sử dụng đoạn chương trình sau.



Lệnh này sẽ vẽ đoạn thẳng nối với đường tròn được vẽ trong lệnh lặp

Luyện tập 2: vẽ hình sau.



Gợi ý: tương tự trên, trước khi bắt đầu vẽ hình tròn, cho nhân vật quay 90 độ theo chiều kim đồng hồ. Ví dụ chương trình sau.



Lệnh này sẽ làm cho bút vẽ quay xuống dưới để vẽ được vòng tròn như trong hình vẽ yêu cầu.

Luyện tập 3: vẽ các hình sau.



Cách thực hiện tương tự trên.

Hình thứ nhất



Hình thứ 2



4. Vẽ hình phức tạp với vòng lặp lồng nhau

Chúng ta đã được làm quen và hiểu ý nghĩa của các lệnh lặp (lệnh **repeat**, **forever**). Các lệnh này cho phép thể hiện ngắn gọn, dễ hiểu hơn khi chương trình có các lệnh hoặc đoạn

lệnh lặp lại. Trong Scratch có 2 lệnh lặp: lặp với số lần cố định **repeat** và lặp vô hạn lần **forever**.

Trong chương trình các vòng lặp có thể được lồng vào nhau, tức là vòng lặp này nằm trong 1 vòng lặp khác. Có rất nhiều ví dụ thực tế mô tả các vòng lặp lồng nhau như vậy.

Ví dụ: khi xét các công việc hàng ngày em phải làm trong 1 tuần lễ (tứ thứ 2 đến thứ 7), có công việc đi học. Khi đó mô tả các công việc này trong 1 tuần, chúng ta có 1 vòng lặp đầu tiên:

```
Lặp lại 6 ngày trong tuần
    Sáng dậy sớm, ăn sáng
    Đi học
```

Bây giờ nếu xét trong cả 1 học kỳ bao gồm 17 tuần, mỗi tuần đều lặp lại lịch học trên, chúng ta sẽ có mô hình vòng lặp lồng nhau:

```
Lặp lại 17 tuần của học kỳ
    Lặp lại 6 ngày trong tuần
        Sáng dậy sớm, ăn sáng
        Đi học
```

Các vòng lặp có thể lồng nhau theo nhiều mức và kiểu khác nhau. Ví dụ trường hợp vòng lặp 1 mức ta có các trường hợp sau:



2 vòng lặp repeat lồng nhau

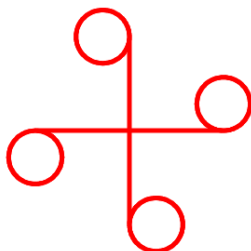


Vòng lặp repeat nằm trong vòng lặp forever



Vòng lặp vô hạn forever nằm trong vòng lặp repeat

Bây giờ em hãy thiết lập chương trình thể hiện hình vẽ dưới đây.

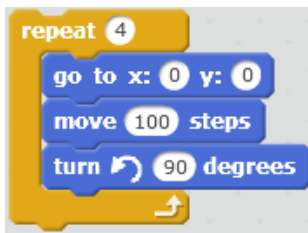


Phân tích bài toán này chúng ta thấy:

Vòng lặp ngoài cùng: thực hiện 4 lần việc vẽ 1 đoạn thẳng xuất phát từ tâm điểm của hình.

Vòng lặp bên trong: vẽ hình tròn.

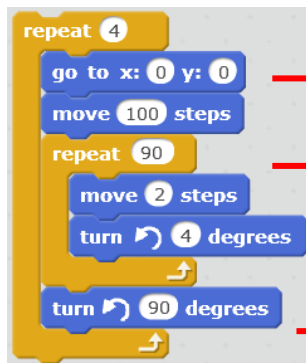
Ở vòng lặp bên ngoài chúng ta có đoạn lệnh:



Ở vòng lặp trong là đoạn lệnh vẽ đường tròn mà chúng ta đã biết:



Ghép 2 vòng lặp này lồng vào nhau chúng ta thu được.

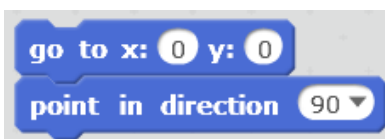


Vòng lặp ngoài, thực hiện 4 lần việc vẽ 1 đoạn thẳng từ vị trí tâm (0,0) độ dài 100.

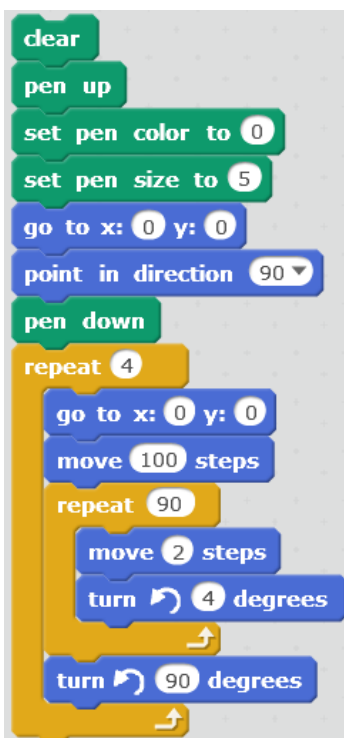
Vòng lặp trong vẽ đường tròn với 90 lần lặp.

Sau khi vẽ vòng tròn, quay hướng bút vẽ 90 độ sang trái để thực hiện vòng lặp ngoài tiếp theo.

Chú ý trước khi thực hiện vòng lặp đầu tiên em cần đưa bút vẽ về vị trí ban đầu là tâm của hình:

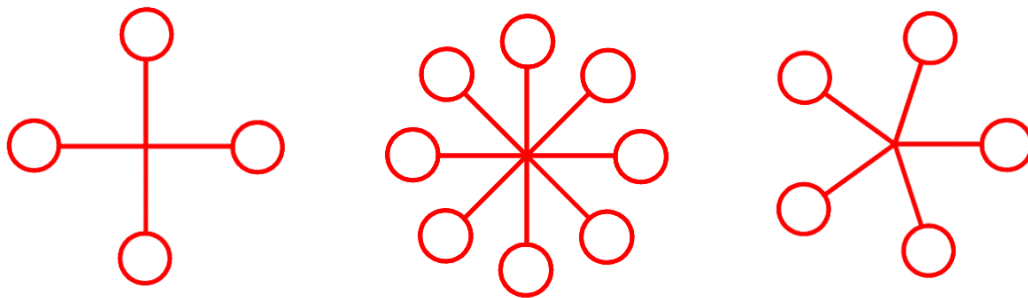


Kết quả chương trình như sau:

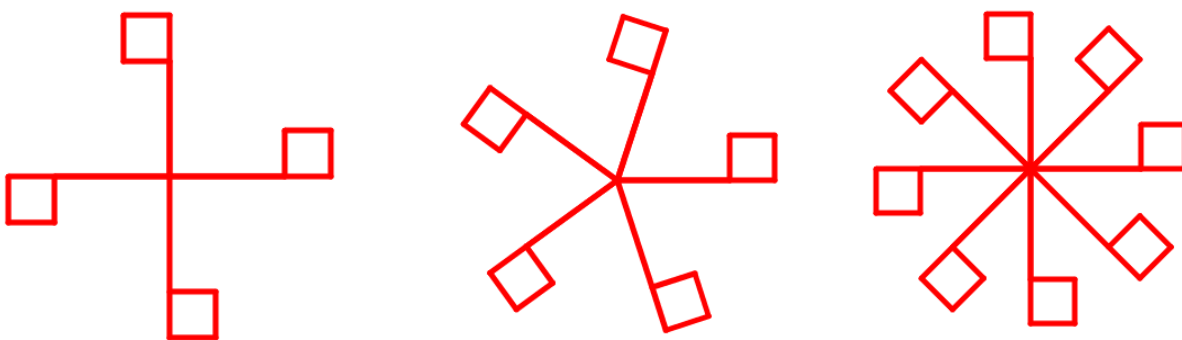


Câu hỏi và bài tập

1. Em hãy viết chương trình vẽ các hình sau:



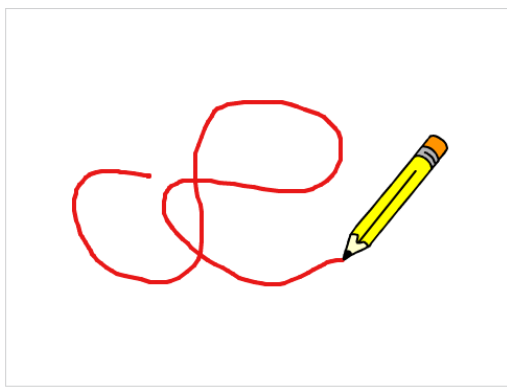
2. Viết chương trình vẽ các hình sau:



3. Viết lại các chương trình trên nhưng thể hiện mỗi nhánh 1 màu khác nhau.

Mở rộng

Thiết kế chương trình **Bút vẽ tự do** như sau:



Trên màn hình khi di chuyển chuột, bút vẽ luôn đi theo con trỏ chuột.

- Bấm phím Space: bắt đầu chế độ vẽ.
- Nháy chuột: kết thúc chế độ vẽ.
- Bấm phím x: chuyển màu vẽ là Xanh.
- Bấm phím d: chuyển màu vẽ là Đỏ.

Bài 8. Âm thanh 2

Mục đích

Học xong bài này, bạn sẽ biết:

- Kết nối âm thanh với nhân vật. Lập trình cho nhân vật nói và thể hiện lời nói bằng chữ và âm thanh.
- Điều khiển các lệnh tạo âm thanh, đánh trống và chơi nhạc trên nền sân khấu.
- Điều khiển nhân vật nhảy múa theo nhạc nền.

Bắt đầu

Em hãy quan sát bản nhạc sau. Em có hiểu cách ghi các nốt nhạc, cao độ, trường độ, nhịp, phách của bản nhạc này không?

Cả nhà thương nhau

Rhythm: Polka
Tone: Music Box

Nhạc và lời: Phan Văn Minh

♩ = 120

The musical score is written on three staves in 2/4 time. The melody is in F major. The lyrics are: Ba thương con vì con giống mẹ. Mẹ thương con vì con giống ba. Cả nhà ta cùng yêu thương nhau. Xa là nhớ gặp nhau là cười.

Chords indicated above the notes: F, Dm, Gm, C, F, Bb, F, C7, F.

Trong bài học này chúng ta sẽ cùng nhau tìm hiểu các lệnh để có thể đánh được đúng các nốt của bản nhạc này.

Nội dung bài học


1. Kết nối âm thanh với nhân vật, nền sân khấu. Ứng dụng kể chuyện



Như chúng ta đã biết, mỗi nhân vật trong Scratch có thể có nhiều trang phục và âm thanh đi kèm, số lượng các âm thanh hay trang phục này là không hạn chế.

Tương tự, sân khấu cũng có thể có nhiều hình ảnh nền sân khấu (backdrop) và âm thanh kèm theo với số lượng không hạn chế.

Các lệnh làm việc với trang phục, âm thanh, nền sân khấu trong Scratch bao gồm:

Ý nghĩa lệnh	Nhân vật	Sân khấu
Chuyển trang phục / nền sân khấu tương ứng.		

Ý nghĩa lệnh	Nhân vật	Sân khấu
Chuyển sang trang phục / nền sân khấu tiếp theo trong danh sách.		
Bật âm thanh tương ứng và thực hiện ngay các lệnh tiếp theo		
Bật âm thanh tương ứng và chờ xong mới thực hiện các lệnh tiếp theo		

Chú ý quan trọng: Dãy các trang phục, nền sân khấu hay âm thanh còn có thể xác định bằng số thứ tự hoặc tên của chúng. Ví dụ nếu chúng ta sử dụng các biến nhớ  để lưu số thứ tự hoặc  để lưu tên của các trang phục, nền sân khấu hay âm thanh này thì các lệnh trên có thể sử dụng các biến nhớ trên để khai thác.



Chuyển nền sân khấu sang nền có số thứ tự trong danh sách là **stt**.

Chuyển nền sân khấu sang nền có tên trong danh sách là **name**.

Chuyển trang phục của nhân vật sang trang phục có số thứ tự trong danh sách là **stt**.

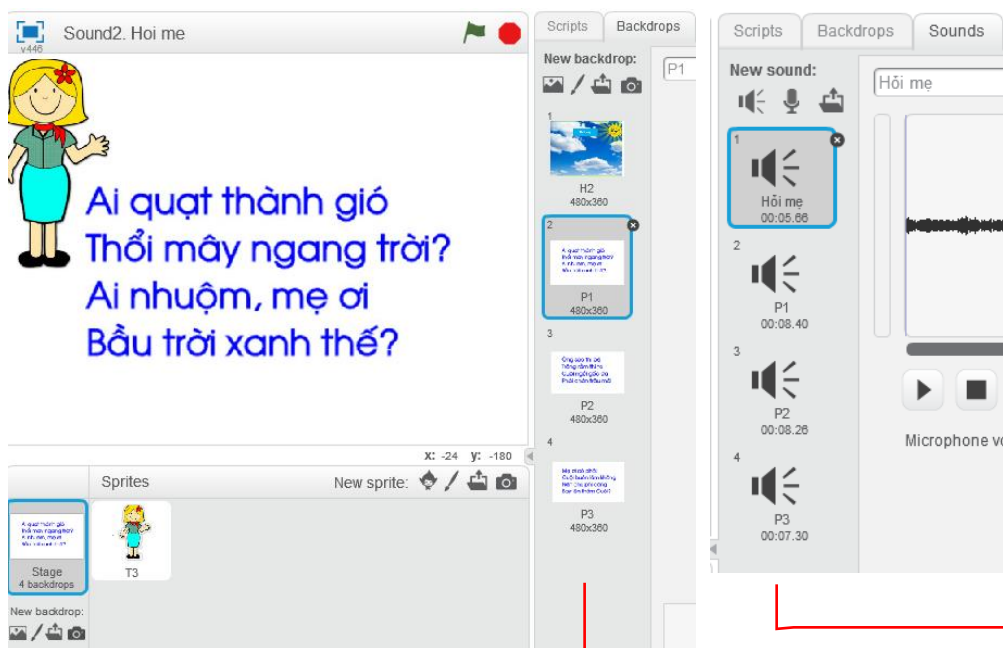
Chuyển trang phục của nhân vật sang nền có tên trong danh sách là **name**.

Bật âm thanh có số thứ tự trong danh sách là **stt**.

Bật âm thanh có tên trong danh sách là **name**.

Thiết lập ứng dụng kể chuyện

Chúng ta sẽ thiết lập một ứng dụng đơn giản có kết nối âm thanh với hình ảnh. Giả sử muốn thực hiện một bài kể chuyện bằng giọng nói theo tranh, ví dụ, bài thơ "hỏi mẹ". Em hãy thiết kế mô hình chương trình bao gồm 4 hình ảnh nền sân khấu và 4 tệp âm thanh kể chuyện tương ứng. Chú ý mỗi tệp âm thanh sẽ là giọng kể mô tả hình ảnh tương ứng.



Dãy âm thanh tương ứng với dãy hình ảnh nền sân khấu.

Dãy hình nền sân khấu chính của câu chuyện muốn đưa vào ứng dụng này.

Đoạn chương trình chính được xây dựng trên của số lệnh của sân khấu. Cần tạo thêm biến nhớ **stt** để lưu số thứ tự của các tệp âm thanh. Chương trình bao gồm 1 vòng lặp (số vòng lặp = số lượng hình ảnh nền). Tại mỗi bước, cần thực hiện lệnh dùng để bật nghe âm thanh cho đến khi xong thì tự động tăng **stt** lên 1 và chuyển sang nền sân khấu tiếp theo.



Thiết lập giá trị ban đầu cho **stt** = 1

Vòng lặp chính của chương trình. Số lần lặp = số lượng hình nền = số lượng âm thanh tương ứng.

2. Trống và nhịp trống

Chắc các em đã được làm quen với các bộ trống trong dàn nhạc và vai trò của trống giữ nhịp trong các biểu diễn âm nhạc. Trong hoạt động này chúng ta sẽ làm quen với các lệnh trống của Scratch.




Bộ các công cụ
trống hiện đại.

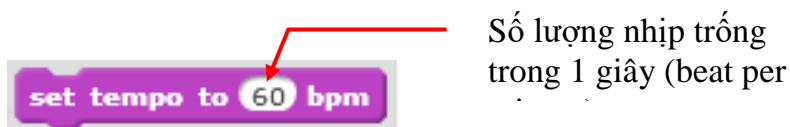
Lệnh gõ trống **play drum for beat**.

Lệnh này có tác dụng đánh 1 công cụ gõ (công cụ ghi trong tham số đầu tiên) và kéo dài trong khoảng thời gian tính theo nhịp (beat) trống, thời gian này ghi trong tham số thứ 2 của lệnh.



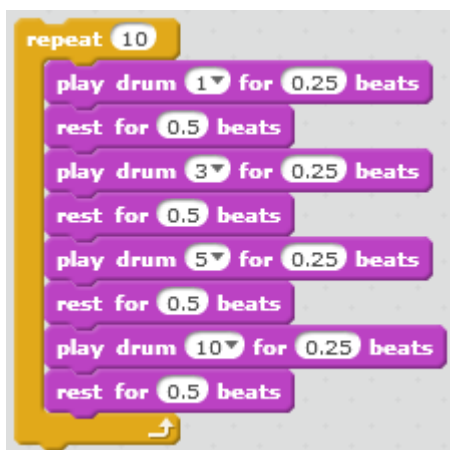
Ví dụ: lệnh `play drum 1 for 0.5 beats` sẽ đánh 1 lần trống nền (trống nền ) và đợi 1/2 nhịp trống tiếp theo. Ở đây nhịp trống được đo bởi giá trị **tempo** = số nhịp trống trong 1 phút. Ví dụ nếu tempo = 60 thì nhịp trống sẽ là theo từng giây, mỗi giây 1 lần đánh trống.

Lệnh thiết lập nhịp trống là **set tempo to**.



Chú ý: Nhịp trống (beat) còn có ý nghĩa trong việc thể hiện trường độ nốt nhạc sẽ được học trong phần sau.

Em hãy soạn và chạy thử đoạn chương trình trống sau đây.



Chương trình bao gồm 10 vòng lặp, mỗi vòng lặp sẽ bao gồm 2 nhịp trống.

Nhịp 1: sẽ lần lượt đánh trống chính và thanh ngang 1/4 nhịp, xen giữa nghỉ 1/4 nhịp.

Nhịp 2: sẽ lần lượt đánh mặt mũ trên và trống ổ 1/4 nhịp, xen giữa nghỉ

Bảng sau cho chúng ta nhận dạng thêm 1 số hình ảnh công cụ trong giàn trống.

Snare
Drum



Tambourine



Bongo



Bass Drum



Hand Clap



Conga



Side Stick



Claves



Cabasa



Crash
Cymbal



Wood Block



Guiro



Open Hi-
Hat



Cowbell



Vibraslap



Closed Hi-
Hat



Triangle

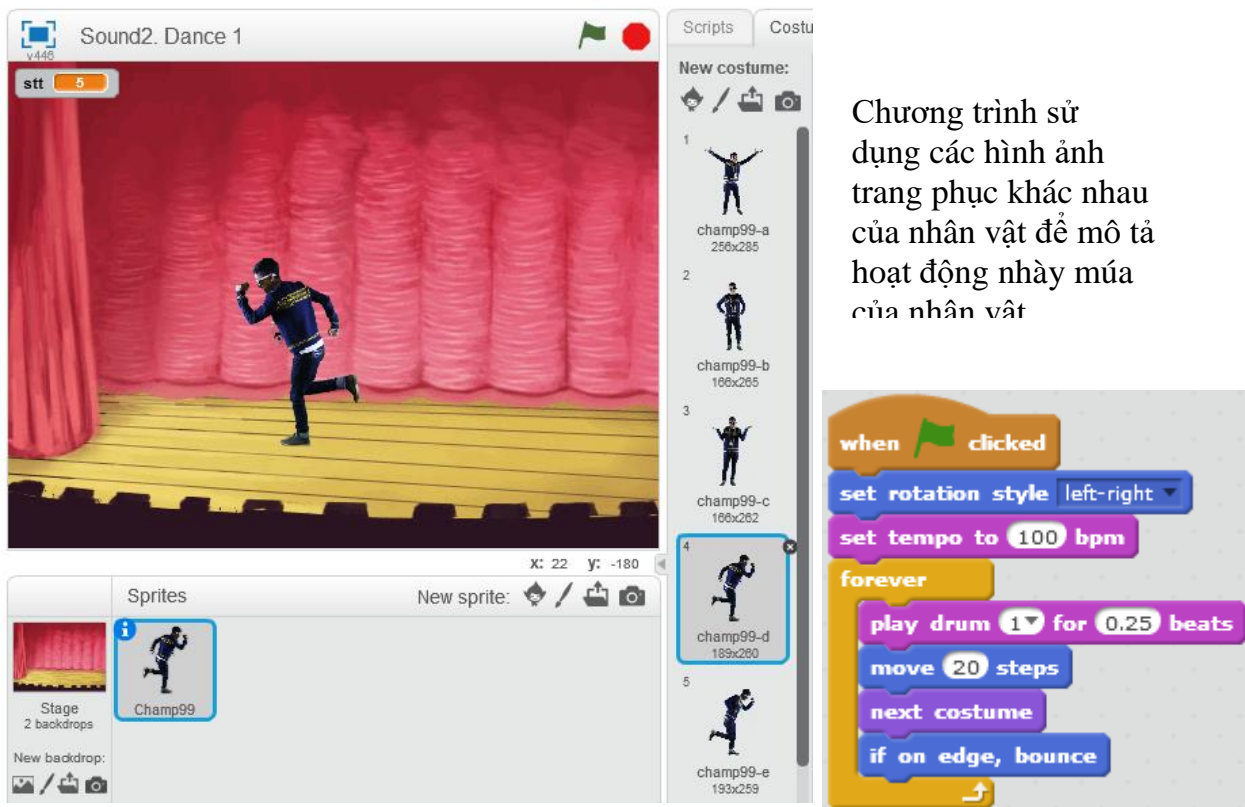


Open Cuica



3. Em bé nhảy và múa theo nhịp trống.

Em hãy thiết lập 1 chương trình đơn giản như thể hiện trong hình sau.



Chương trình sử dụng các hình ảnh trang phục khác nhau của nhân vật để mô tả hoạt động nhảy múa của nhân vật

4. Đánh nốt nhạc

Trong hoạt động này em sẽ được học cách Scratch điều khiển các công cụ chơi nhạc. Các em chú ý đến các yếu tố sau:

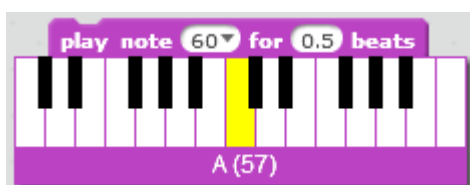
1. Công cụ chơi nhạc.
2. Độ cao nốt nhạc.
3. Trường độ nốt nhạc.

Độ cao của nốt nhạc

Lệnh phát nốt nhạc chính trong Scratch là **play note for beats**. Lệnh này có tác dụng phát 1 nốt nhạc đúng cao độ và trường độ được ghi ngay trong tham số của lệnh. Trường độ được tính theo nhịp trống (beat).



Khi lựa chọn cao độ nốt nhạc, em sẽ thấy hiện hình ảnh mô tả phím đàn Piano như hình dưới đây. Mỗi nốt nhạc có 1 ký tự chữ cái thể hiện và giá trị cao độ của nốt nhạc.



Khi nhấn chuột vào vị trí muốn nhập cao độ nốt nhạc sẽ xuất hiện hình ảnh bên thể hiện phím đàn cho người dùng để quan sát và nhập. Bên cạnh giá trị số còn có ký hiệu các nốt nhạc.

Bảng ký hiệu các nốt nhạc:

Đô	Rê	Mi	Pha	Son	La	Si	Đô
C	D	E	F	G	A	B	C

Ví dụ các giá trị nốt chính của bản nhạc được mô tả trong hình sau:



Note	48	50	52	53	55	57	59
-------------	-----------	-----------	-----------	-----------	-----------	-----------	-----------



Note	60	62	64	65	67	69	71
-------------	-----------	-----------	-----------	-----------	-----------	-----------	-----------

Trường độ của nốt nhạc

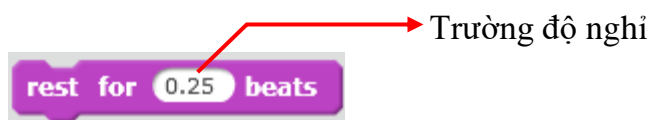
Trường độ chỉ ra độ dài theo thời gian của nốt nhạc. Trong Scratch, đơn vị đo trường độ là **beat** (nhịp trống) mà chúng ta đã biết. Giá trị trường độ được thể hiện trong bảng sau:

Ký hiệu nốt nhạc



Trường độ (beat)	4	2	1	0.5	0.25	0.125	0.0625
-------------------------	----------	----------	----------	------------	-------------	--------------	---------------

Với lệnh nghỉ (khoảng lặng của bản nhạc) chúng ta sử dụng lệnh sau:



Bảng các ký hiệu khoảng lặng trong khuôn nhạc như sau:

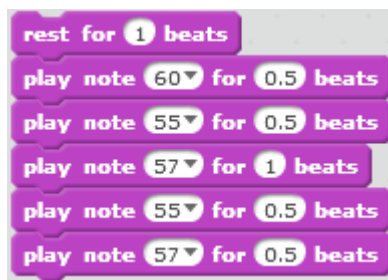
Ký hiệu nghỉ trên nốt nhạc



beat	4	2	1	0.5	0.25
-------------	----------	----------	----------	------------	-------------

Bây giờ nhìn vào 1 bản nhạc đơn giản, chúng ta có thể viết đoạn chương trình mô phỏng bản nhạc đó.

Em quan sát bản nhạc và thực hiện các lệnh Scratch tương ứng trong các ô trống của bảng sau:



Bây giờ chúng ta sẽ cùng thiết lập các đoạn nhạc dài hơn, dựa trên các bài hát quen thuộc mà em đã biết.

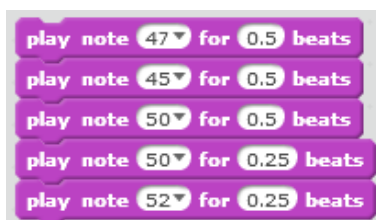
Ví dụ 1: một đoạn nhạc bài hát **Đi học** (nhạc Bùi Đình Thảo, thơ Minh Chính).



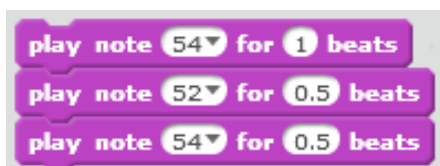
Đoạn nhạc này có 6 khuôn nhịp.

Em hãy điền tiếp các lệnh Scratch cho các khuôn nhịp 3, 4, 5.

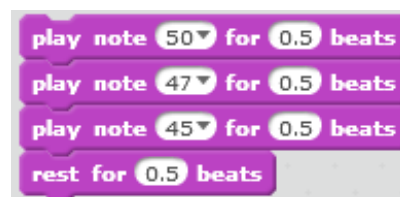
Nhịp 1



Nhịp 2



Nhịp 2



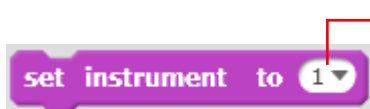
Ví dụ 2: một đoạn nhạc bài **Người Hà Nội** (nhạc: Nguyễn Đình Thi).



Đoạn đầu của bài hát được thiết lập như hình trên. Em hãy viết tiếp đoạn sau của bản nhạc này.

Lựa chọn công cụ chơi nhạc.

Lệnh thiết lập công cụ chơi nhạc trong Scratch là **set instrument to**.



Nhạc cụ lựa chọn tại đây.

Bảng sau mô tả bằng hình ảnh các công cụ chơi nhạc hỗ trợ trong Scratch.

Piano



Bass



Saxophone



Electric piano



Pizzicato



Flute



Organ



Cello



Wooden Flute



Guitar



Trombone



Bassoon



Electric guitar



Clarinet



Choir



5. Một số bài hát, bản nhạc hoàn chỉnh

Em hãy cùng thực hiện tạo chương trình chơi các bản nhạc hoàn chỉnh sau.

Bài 1. Chúc sinh nhật. Happy birthday.

Happy Birthday TRADITIONAL

MODERATO

The musical score for 'Happy Birthday' is shown in 3/4 time. The melody is written on a single staff. The lyrics are: HAP - PY BIRTH - DAY TO YOU. HAP - PY BIRTH - DAY TO YOU. HAP - PY BIRTH - DAY TO NA - ME HAP - PY BIRTH - DAY TO YOU. HAP - PY BIRTH - DAY TO YOU.

Below the score, there are two columns of Scratch code blocks, connected by a red arrow pointing from the first column to the second.

Column 1:

- play note 55▼ for 0.5 beats
- play note 55▼ for 0.5 beats
- play note 57▼ for 1 beats
- play note 55▼ for 1 beats
- play note 60▼ for 1 beats
- play note 59▼ for 2 beats
- play note 55▼ for 0.5 beats
- play note 55▼ for 0.5 beats
- play note 57▼ for 1 beats
- play note 55▼ for 1 beats
- play note 62▼ for 1 beats
- play note 60▼ for 2 beats
- play note 55▼ for 0.5 beats

Column 2:

- play note 55▼ for 0.5 beats
- play note 67▼ for 1 beats
- play note 64▼ for 1 beats
- play note 60▼ for 1 beats
- play note 59▼ for 1 beats
- play note 57▼ for 1 beats
- play note 65▼ for 0.5 beats
- play note 65▼ for 0.5 beats
- play note 64▼ for 1 beats
- play note 60▼ for 1 beats
- play note 62▼ for 1 beats
- play note 60▼ for 2 beats

Bài 2. Cả nhà thương nhau. Nhạc và lời: Phan Văn Minh.

Cả nhà thương nhau

Rhythm: Polka
Tone: Music Box
♩ = 120

Nhạc và lời: Phan Văn Minh

The musical score for 'Cả nhà thương nhau' is shown in 2/4 time. The melody is written on a single staff. The lyrics are: Ba thương con vì con giống mẹ. Mẹ thương con vì con giống ba. Cả nhà ta cùng yêu thương nhau. Xa là nhớ gặp nhau là cười.

Below the score, there are two columns of Scratch code blocks, connected by a red arrow pointing from the first column to the second.

Column 1:

- play note 53▼ for 1 beats
- play note 53▼ for 1 beats
- play note 53▼ for 1.5 beats
- play note 50▼ for 0.5 beats
- play note 53▼ for 1 beats
- play note 55▼ for 0.5 beats
- play note 53▼ for 0.5 beats
- play note 50▼ for 2 beats
- play note 50▼ for 1 beats
- play note 55▼ for 1 beats
- play note 55▼ for 1.5 beats
- play note 53▼ for 0.5 beats
- play note 55▼ for 1 beats
- play note 57▼ for 0.5 beats
- play note 60▼ for 0.5 beats
- play note 57▼ for 2 beats

Column 2:

- play note 53▼ for 1 beats
- play note 55▼ for 1 beats
- play note 58▼ for 1.5 beats
- play note 55▼ for 0.5 beats
- play note 58▼ for 1 beats
- play note 60▼ for 1 beats
- play note 60▼ for 2 beats
- play note 57▼ for 1 beats
- play note 55▼ for 0.5 beats
- play note 57▼ for 0.5 beats
- play note 60▼ for 1.5 beats
- play note 48▼ for 0.5 beats
- play note 55▼ for 1 beats
- play note 53▼ for 0.5 beats
- play note 55▼ for 0.5 beats
- play note 53▼ for 2 beats

Câu hỏi và bài tập

1. Em hãy tạo khuôn nhạc cho một bài hát hoàn chỉnh:

EM YÊU TRƯỜNG EM

Nhạc và lời: Hoàng Vân

Nhịp nhàng - Vui

Em yêu trường em với bao bạn thân và cô giáo
Em yêu trường em với bao bạn thân và cô giáo

hiên, như yêu quê hương cấp sách tới trường trong vườn yêu
hiên, như yêu quê hương cấp sách tới trường trong vườn yêu

thương. Nào bàn nào ghế, nào sách nào
thương. Mùa phượng phượng thắm, mùa các vàng

vỏ. Nào mực nào bút, nào phấn nào bảng. Cả tiếng chim
nở. Mùa huệ huệ trắng, đào thắm hồng đỏ. Trường chúng em

vui trên cành cây cao. Cả lá cờ sao trong ánh thu
đầy như vườn hoa tươi. Người tốt việc hay là cháu Bác

vàng, yêu sao yêu thể trường của chúng
Hò, yêu sao yêu thể trường của chúng

em. yêu sao yêu thể trường của chúng em.
em. yêu sao yêu thể trường của chúng em.

2. Em hãy tạo khuôn nhạc cho một bài hát hoàn chỉnh: Đếm ngón tay.

đếm ngón tay

Xoè bàn tay, đếm ngón tay. Một anh béo trông
Xoè bàn tay, đếm ngón tay. Nhìn anh giữa trông
Rồi bên anh, đứng thứ năm, người coi đáng trông

thật đến hay. Cả ngày vui, ai có việc,
thật đến cao. Hỏi tại sao cao thế nào,
thật đến xinh. Hỏi rằng ai em út nhà?

là anh giúp luôn không ngồi yên. Kề bên anh,
thì anh nói anh chăm thể thao. Cạnh bên anh
thì em hát ngay theo nhịp ca. Rằng là em

đứng thứ hai, một anh tính thật thà để thương, hỏi rằng
đứng thứ tư, hỏi anh đã biết đọc chữ chưa thì anh
bé rất ngoan từ nay khám tay sạch các anh, làm vệ

anh cao nhất nhà, thì anh lác luôn ngay cái đầu.
thưa, anh biết rồi, rồi anh đứng nghiêng dơ tay chào.
sinh, hay quét nhà, và múa hát cho vui ông bà.

Mở rộng

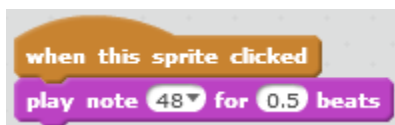
Thiết lập chương trình đơn giản sau.



Trên màn hình là 15 nút tròn dùng để mô tả các nốt nhạc. Trên mỗi nút có chữ cái ký hiệu tên nốt và giá trị tương ứng của lệnh thể hiện nốt nhạc này.

Nháy chuột lên các nút tròn này sẽ phát âm thanh chính là nốt nhạc tương ứng.

Gợi ý: thiết lập 15 nhân vật có hình các nút tròn trên. Với mỗi nhân vật - nút tròn sử dụng lệnh bắt sự kiện nháy chuột sau. Đoạn chương trình sau dành cho nút đỏ đầu tiên (C, 48).



CHƯƠNG 3: TÌM HIỂU SÂU HƠN SCRATCH

Bài 9. Hội thoại

Mục đích

- Thực hiện giao tiếp hội thoại người - máy đơn giản.
- Khái niệm biến nhớ, sử dụng biến nhớ trong cách giải quyết vấn đề.

Bắt đầu

1. Em hiểu thế nào là hội thoại người - máy, cho 1 vài ví dụ.
2. Em thực hiện đoạn chương trình trình diễn hội thoại đơn giản sau giữa 2 bạn Hoa và Lan.

Hoa: Chào Lan.

Lan: Chào bạn Hoa, lâu lắm mới gặp.

Hoa: Bạn bây giờ học trường nào?

Lan: Tôi học trường THCS Bế Văn Đàn. Bạn khỏe không?

Hoa: Cảm ơn bạn, tôi vẫn khỏe.

Ví dụ có thể viết chương trình sau mô tả đoạn hội thoại trên.

Hoa



Lan



Theo em, chương trình trên có phải là hội thoại người - máy hay không? Vì sao?

3. Đoạn văn sau mô tả 1 hội thoại người - máy tính.

Máy tính: Bạn hãy nhập 1 số tự nhiên.

Người: thực hiện động tác nhập 1 số từ bàn phím, ví dụ nhập số 12.

Máy tính: Bạn hãy nhập tiếp 1 số tự nhiên khác.

Người: thực hiện động tác nhập 1 số từ bàn phím, ví dụ nhập số 18.

Máy tính (sau 1 giây suy nghĩ): Bạn đã nhập 2 số tự nhiên 12, 18. Ước số chung lớn nhất 2 số này là 6.

Em có nhận xét gì về cuộc hội thoại trên.

Nội dung bài học

1. Bắt đầu 1 chương trình hội thoại đơn giản

Em hãy thiết lập chương trình để thực hiện hoạt động hội thoại sau.

Nhân vật chính là 1 thầy giáo, học sinh là người sử dụng máy tính.

Thầy giáo: Chào học sinh, em tên là gì?

Học sinh nhập từ bàn phím: Hòa Anh.

Thầy giáo: Chào Hòa Anh, rất vui được làm quen với em.
Sau 2 giây.

Thầy giáo: Bây giờ em hãy nhập từ bàn phím 1 số tự nhiên.

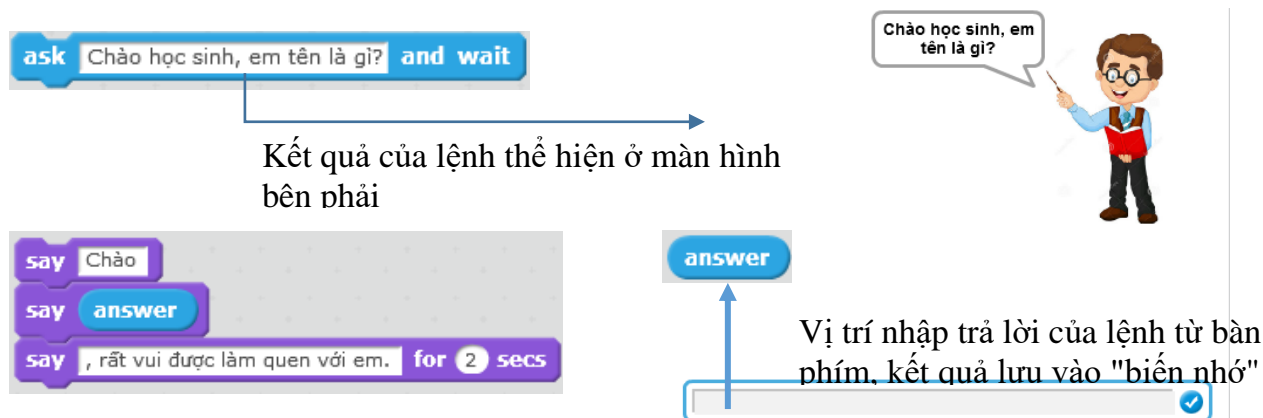
Học sinh nhập từ bàn phím: 15

Thầy giáo: Tốt lắm, em đã nhập số 15.

Trong ví dụ trên, chúng ta được làm quen với 1 dạng hội thoại thực sự giữa máy tính và người sử dụng. Máy tính đưa ra yêu cầu; học sinh trả lời thông qua bàn phím; máy tính nhận kết quả và đưa ra thông điệp tiếp theo tùy thuộc vào kết quả đã nhận được.

Để mô tả quá trình hội thoại trên chúng ta sử dụng lệnh sau từ nhóm lệnh Cảm biến (Sensing).

Lệnh **ask** **What's your name?** **and wait** (**ask and wait**) sẽ cho phép nhân vật hiện câu hỏi trên màn hình và chờ người dùng nhập câu trả lời từ bàn phím vào 1 dòng nhập liệu. Câu trả lời này sẽ được lưu trong "biểu thức" hay "lệnh" **answer**. Giá trị **answer** này sau đó có thể được sử dụng lại vào các công việc khác. Chúng ta cùng xét ví dụ cụ thể của lệnh này.

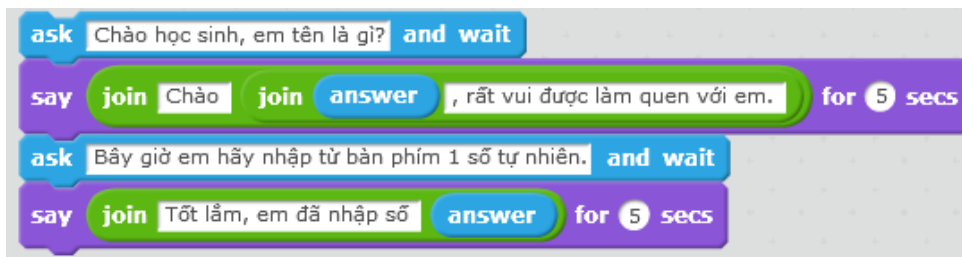


Giá trị **answer** được sử dụng để giáo viên tiếp tục hội thoại với học sinh dưới dạng đầy đủ như hình trên hoặc dạng rút gọn như hình



Trong lệnh trên chúng ta đã sử dụng lệnh **join** **hello** **world** trong nhóm lệnh **Tính toán** dùng để kết nối 2 giá trị hoặc 2 cụm từ.

Chương trình đầy đủ của hoạt động hội thoại này như sau:



2. Biến nhớ là gì?

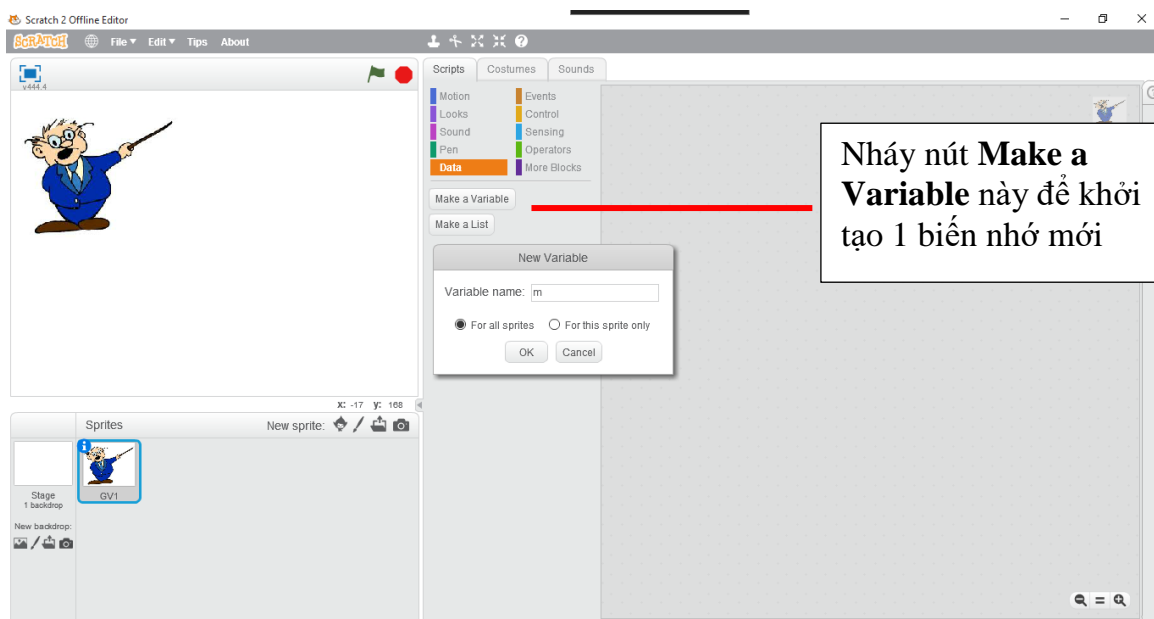
Trong hoạt động 1, chúng ta đã làm quen với lệnh hay "biến nhớ" **answer**, đây là 1 kỹ thuật được sử dụng thường xuyên trong Tin học, trong các môi trường lập trình. Cụm từ **answer** luôn được hiểu với ý nghĩa là 1 vùng trong bộ nhớ máy tính để lưu trữ giá trị mà người sử dụng nhập từ bàn phím thông qua lệnh **ask and wait**. Hiểu đơn giản biến nhớ được tạo ra để lưu trữ dữ liệu. Khi được tạo ra rồi thì chúng ta có thể thực hiện các phép tính với biến nhớ tương tự như với dữ liệu lưu trong biến nhớ này.

Em hãy thực hiện các chương trình đơn giản sau, qua đó hiểu được ý nghĩa của biến nhớ trong Scratch.

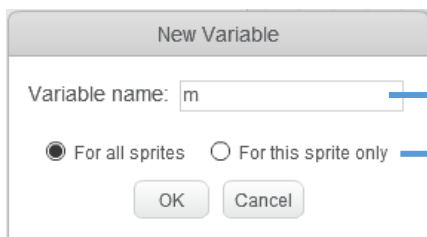
Chương trình 1.

Thầy giáo yêu cầu học sinh nhập từ bàn phím 2 số tự nhiên m, n, sau đó thông báo 2 số này và tổng của chúng.

Bước 1. Tạo 1 chương trình mới, bổ sung thêm nhân vật Thầy giáo, khởi tạo 2 biến nhớ mới đặt tên là m, n.

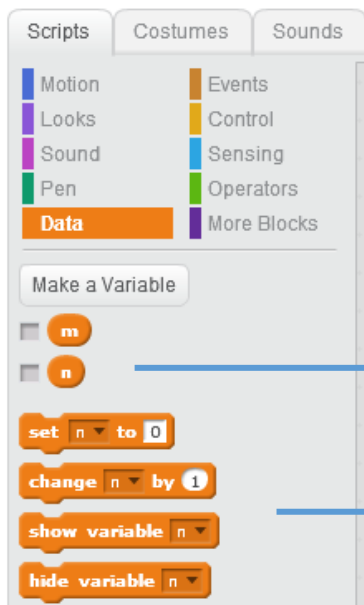


Để tạo 1 biến nhớ mới, em vào nhóm lệnh Dữ liệu (Data) và nháy nút Make a Variable. Cửa sổ tạo biến nhớ có dạng sau, em nhập m tại ô **Variable name**, chọn **For all sprites** và nháy **OK**.



Đặt tên biến nhớ ở đây
Chọn kiểu biến nhớ chung hay chỉ riêng cho nhân vật hiện thời.

Tương tự em tạo thêm biến nhớ có tên n. Hình ảnh sau trên khung mẫu lệnh của nhóm Dữ liệu (Data) sẽ xuất hiện các lệnh làm việc với biến nhớ.



Danh sách các biến nhớ đã khởi tạo hiện ở đây.

Các lệnh của nhóm lệnh làm việc với biến nhớ.

Có hai lệnh làm việc quan trọng với biến nhớ là:

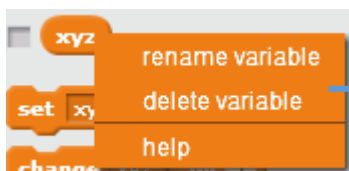


Lệnh thiết lập, gán 1 giá trị cụ thể cho biến nhớ.



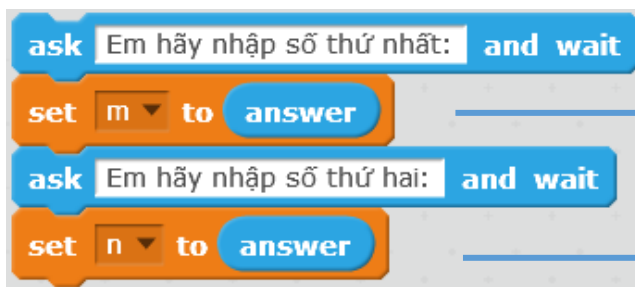
Lệnh thay đổi giá trị của biến nhớ (tăng lên hoặc giảm đi) theo 1 giá trị cụ thể.

Chú ý: Nháy chuột phải lên 1 biến nhớ sẽ xuất hiện 1 bảng chọn nhỏ để thực hiện thêm các lệnh bổ sung cho biến nhớ này.



Các lệnh bổ sung với biến nhớ hiện thời:
rename variable: đổi tên biến nhớ.
delete variable: xóa biến nhớ.

Bước 2. Thực hiện hội thoại giữa thầy giáo và học sinh (người dùng máy tính).



Gán giá trị của **answer** vào biến nhớ **m**.

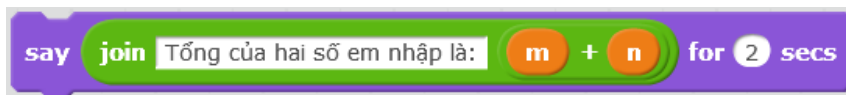
Gán giá trị của **answer** vào biến nhớ **n**.

Bước 3. Thông báo tổng 2 số m, n .

Để thể hiện thông báo này chúng ta cần hiển thị 1 thông tin chữ và giá trị tổng $m + n$. Dùng lệnh / biểu thức **join** **hello** **world** (từ nhóm lệnh Tính toán) để thực hiện yêu cầu này.

Lệnh hay toán tử **+** dùng để tính giá trị $m + n$.

Kết quả của thông báo này như sau:



Chương trình 2.

Thay đổi chương trình trên, yêu cầu học sinh nhập 1 số tự nhiên với yêu cầu số này phải > 100 . Giáo viên sẽ liên tục thông báo cho học sinh cần nhập đúng và yêu cầu nhập lại nếu học sinh nhập không đúng số theo yêu cầu.

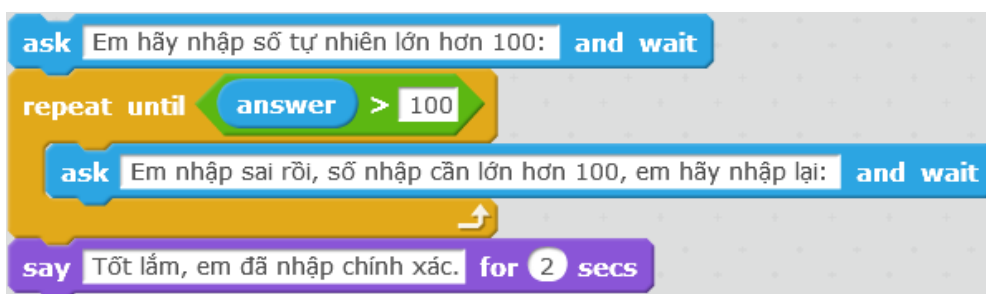
Để thực hiện chương trình này chúng ta sẽ sử dụng một lệnh lặp mới, lệnh lặp có điều kiện **repeat until** để kiểm tra học sinh có nhập đúng yêu cầu hay không. Lệnh lặp này không chỉ ra trước số lần lặp, lệnh sẽ thực hiện việc lặp cho đến khi 1 điều kiện logic được thực hiện.



Biểu thức logic là điều kiện của lệnh lặp: nhóm lệnh này sẽ được lặp lại liên tục cho đến khi biểu thức này có giá trị đúng.

Với yêu cầu cụ thể của chương trình 2, điều kiện cần kiểm tra là giá trị số mà học sinh nhập cần > 100 . Nếu điều kiện này đúng thì việc lặp sẽ dừng lại. Chúng ta sẽ sử dụng lệnh, biểu thức logic so sánh **>** từ nhóm lệnh tính toán, cụ thể biểu thức cần đưa vào vị trí kiểm tra của lệnh lặp là **answer > 100**.

Kết quả chương trình 2 được xây dựng như sau:



3. Lệnh, biểu thức, hàm có giá trị dùng như biến nhớ

Trong Scratch có rất nhiều lệnh có ý nghĩa của 1 biểu thức hàm số và được dùng tương tự như biến nhớ **answer** mà em đã biết.

Em cần chú ý đến hình dạng các lệnh của Scratch để phân biệt.



Lệnh có 2 đầu thẳng góc

Đây là các lệnh bình thường, không có giá trị trả lại như 1 biểu thức.



Lệnh có 2 đầu nhọn

Đây là các lệnh trả lại giá trị logic, tức là Đúng hoặc Sai (True or False).



Lệnh có 2 đầu tròn

Đây là các lệnh trả lại giá trị số hoặc chữ.

Trong các loại lệnh trên, 2 loại sau có thể sử dụng như 1 biến nhớ, có thể đưa vào trong các điều kiện, biểu thức tính toán của các lệnh khác.

Các biểu thức, lệnh hay biến nhớ có giá trị này có thể chèn vào các vị trí trống trong các lệnh khác nhau của Scratch. Hình sau mô tả qui định của các giá trị có thể chèn.



Tại các vị trí **tròn** chỉ cho phép nhập hoặc chèn giá trị số, nguyên hoặc thập phân.

Tại các vị trí **vuông** cho phép nhập hoặc chèn giá trị số, chữ hoặc logic.

Sau đây là 1 vài lệnh có ý nghĩa biểu thức và có thể sử dụng như các biến nhớ trong Scratch.



Nhóm **Chuyển động (Motion)**

Trả lại giá trị là tọa độ X của nhân vật tại thời điểm hiện thời.



Nhóm **Chuyển động (Motion)**

Trả lại giá trị là tọa độ Y của nhân vật tại thời điểm hiện thời.



Nhóm **Cảm biến (Sensing)**

Trả lại trạng thái logic Đúng / Sai, đúng nếu nhân vật va chạm với 1 nhân vật khác hoặc cạnh sân khấu.



Nhóm **Cảm biến (Sensing)**

Trả lại trạng thái logic Đúng / Sai, đúng nếu nhân vật tiếp xúc với màu sắc được chọn trong lệnh.



Nhóm **Cảm biến (Sensing)**

Trả lại trạng thái logic Đúng / Sai, đúng nếu người dùng nhấn phím tương ứng.



Nhóm **Tính toán (Operators)**

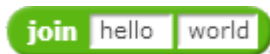
Trả lại giá trị số ngẫu nhiên nằm giữa 2 giá trị số được ghi trên thanh lệnh.



Nhóm **Cảm biến** (Sensing)

Trả lại giá trị thời gian hiện thời (giây, phút, giờ, năm, ...). Các thông số có thể lựa chọn.

- year: năm (2016)
- month: tháng (1→12)
- date: ngày của tháng (1→31)
- day of week: ngày của tuần (1→7)
- hour: giờ hiện thời (0→23)
- minute: phút hiện thời (0→59)
- second: giây hiện thời (0→59)

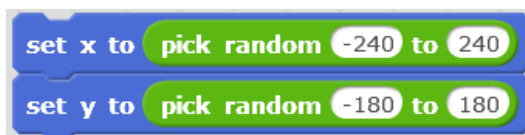


Nhóm **Tính toán** (Operators)

Trả lại giá trị dãy ký tự là ghép nối của 2 biểu thức trong ô vuông của lệnh.

4. Một số ví dụ sử dụng biến nhớ và hàm số

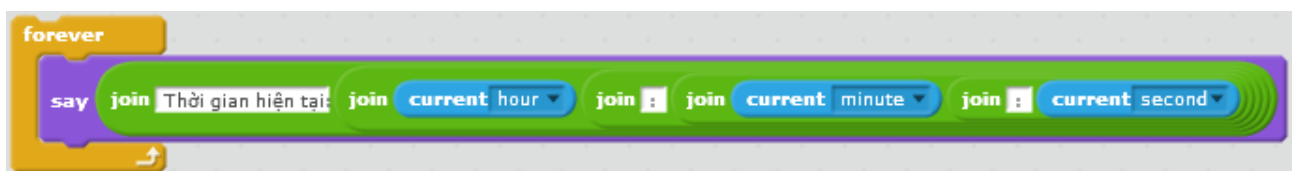
4.1. Để điều khiển nhân vật chính xuất hiện ngẫu nhiên trên màn hình, em có thể sử dụng đoạn lệnh sau:



4.2. Chúng ta muốn thể hiện thời gian chạy online trên màn hình, ví dụ trong hình sau.



thì cần thực hiện lệnh sau:



4.3. Điều kiện kiểm tra xem nhân vật đã chạm đáy màn hình chưa.



4.4. Điều kiện kiểm tra xem nhân vật đã chạm đỉnh trên màn hình hay chưa.





5. Tạo nút cho phép nhập điều chỉnh biến nhớ trên màn hình

Hoạt động này cho phép thiết lập giao diện cho phép người sử dụng nhập hoặc điều chỉnh dữ liệu ngay trên màn hình trong quá trình chạy chương trình mà không cần nhập bằng lệnh **ask and wait**.

Chúng ta cùng thực hiện bài toán sau.

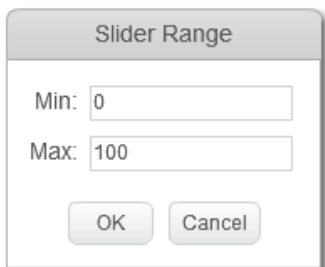
Chương trình cho phép người dùng nhập điều chỉnh 2 giá trị số m , n ngay trên màn hình và sau khi người dùng nhấp chuột lên nhân vật chính thì chương trình thông báo tổng của hai số này.

Để cho biến nhớ luôn hiển thị giá trị trên màn hình em cần nhấp chọn vào ô hiển thị bên cạnh biểu tượng biến nhớ  . Làm tương tự như vậy với biến nhớ n . Các biến nhớ hiện trên màn hình sẽ có dạng như sau:



Bây giờ chúng ta hãy thực hiện thao tác cho phép người dùng có thể thay đổi trực tiếp giá trị của các biến nhớ này trên màn hình. Với mỗi biến nhớ thực hiện 2 thao tác sau:

- Nhấp chuột phải lên vị trí biến nhớ trên sân khấu và nhấp chọn lệnh **slider**.
- Nhấp chuột phải lên vị trí biến nhớ 1 lần nữa và chọn lệnh **set slider min and max** để thiết lập vùng dữ liệu muốn điều chỉnh trực tiếp giá trị của biến nhớ. Xuất hiện hộp hội thoại nhỏ sau, cần nhập 2 giá trị **Min** và **Max**.



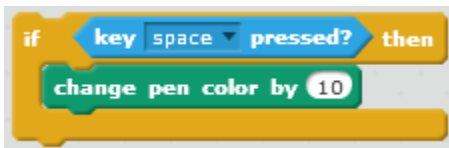
Bây giờ em có thể thiết lập các lệnh để thực hiện theo yêu cầu của chương trình.



Câu hỏi và bài tập

1. Các lệnh sau đây thực hiện những công việc gì?

A.



B.



C.



2. Em hãy viết các lệnh hoặc đoạn chương trình mô tả các yêu cầu sau.

A. Mô tả thời gian hiện thời bằng tiếng Việt, ví dụ:

Bây giờ là 15 giờ 20 phút 14 giây.

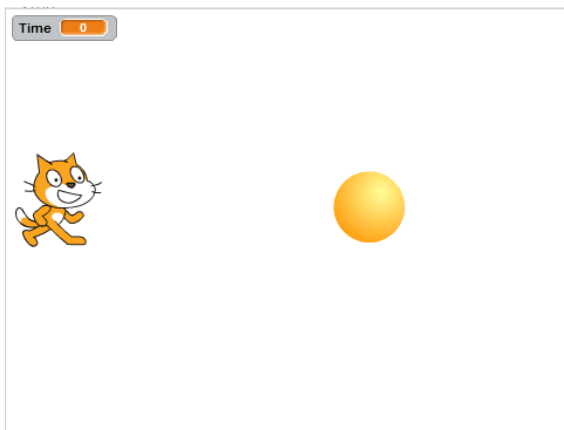
B. Cho nhân vật chuyển động ngẫu nhiên trên sân khấu, nếu gặp cạnh sân khấu thì quay lại.

C. Đếm ngược thời gian: biến nhớ Time được gán ban đầu giá trị 10 và sẽ hiển thị đếm ngược trên màn hình về 0.

Mở rộng

1. Hãy thiết kế 1 trò chơi đơn giản **Mèo và Bóng** sau:

Trên màn hình có con Mèo và quả Bóng.



Thời gian cho mỗi cuộc chơi là 15 giây. Trên màn hình biến nhớ Time sẽ đếm ngược từ 15 về 0.

Khi bắt đầu chương trình, quả bóng sẽ dịch chuyển nhanh trên sân khấu một cách ngẫu nhiên. Nhiệm vụ của người chơi là nhấp chuột nhanh lên quả bóng, càng nhiều lần càng tốt.

Khi thời gian lùi về 0, trò chơi dừng lại. Mèo con thông báo "Game Over" và sau đó thông báo số lần bạn nhấp chuột chính xác lên quả bóng.

Thiết lập 2 nhân vật Mèo và Bóng. Tạo 2 biến nhớ là **Time** - thời gian của mỗi lần chơi và **ClickCount** dùng để đếm số lần người chơi nhấp đúng lên quả bóng.

Chương trình cho Mèo.



Chương trình cho Bóng.



2. Bổ sung thêm âm thanh vào trò chơi trên cho chương trình thêm sinh động khi chơi.

Bài 10. Hội thoại và truyền thông

Mục đích

- Thực hiện bài toán hội thoại có điều khiển thông qua thông điệp.
- Giải quyết vấn đề thông qua thông điệp truyền thông giữa các nhân vật.

Bắt đầu

1. Em đã bao giờ tham dự 1 cuộc họp ở lớp mà các bạn đều đồng thanh nói, không ai chịu nghe ai cả chưa? Hãy kể 1 vài ví dụ thực tế như vậy.
2. Em hãy mô tả theo hiểu biết của em về hoạt động của mô hình thư điện tử (email) trên thực tế.
3. Trên lớp học, cô giáo ra 1 bài tập yêu cầu cả lớp làm. Sau 5 phút, cô nói: bạn nào xung phong lên bảng làm. Một số cánh tay giơ lên. Cô gọi 1 bạn lên bảng làm cho cả lớp xem. Em có nhận xét gì về mô hình hội thoại trong ví dụ trên.



Nội dung bài học

1. Bài toán hội thoại có điều khiển giữa 3 nhân vật

Chúng ta cùng nhau xây dựng chương trình với bài toán sau:

Có 3 bạn: Lan, Bình, Việt trên sân khấu,



Lan

Bình



Việt

Đầu tiên Lan chào: "Xin chào bạn Bình, bạn đi đâu đấy"?.

Bình nghe hiểu và trả lời: "Tôi đi chơi với mẹ".

Sau đó Lan nói: "Việt đây à, chào bạn".

Việt nghe thấy và trả lời: "Chào Lan, bạn đi đâu đấy?"

Quy trình thực hiện bài toán hội thoại - truyền thông này như sau.

- Lan thiết lập 2 thông điệp "Hỏi Bình" và "Chào Việt".
- Lan nói "Xin chào bạn Bình, bạn đi đâu đấy?" và gửi đi thông điệp "Hỏi Bình".
- Bình nhận được thông điệp "Hỏi Bình" thì thực hiện 3 việc sau:
 - + Thiết lập thông điệp "Bình xong rồi".
 - + Nói: "Tôi đi chơi với mẹ".
 - + Gửi đi thông điệp "Bình xong rồi".
- Lan nhận được thông điệp "Bình xong rồi" thì thực hiện tiếp 2 việc sau:
 - + Lan nói: "Việt đấy à, chào bạn".
 - + Gửi đi thông điệp "Chào Việt".
- Việt khi nhận được thông điệp "Chào Việt" thì nói: "Chào Lan, bạn đi đâu đấy".

Chương trình được xây dựng và thiết kế như sau.

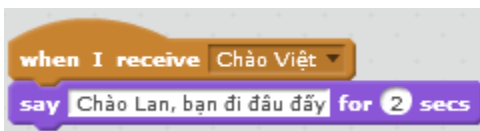
Chương trình của Lan.



Chương trình của Bình.



Chương trình của Việt.



2. Gửi và nhận thông điệp

Đọc và ghi nhớ các lệnh truyền thông trong Scratch.

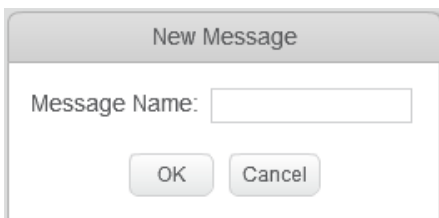
Để xử lý các bài toán hội thoại - truyền thông, trong Scratch có 2 lệnh quan trọng sau trong nhóm **Sự kiện**.



Lệnh phát / truyền thông điệp, đồng thời có thể tạo ra 1 thông điệp mới. Mỗi nhân vật có thể tạo ra và phát không hạn chế các thông điệp. Mỗi thông điệp có 1 tên riêng và tồn tại trong suốt thời gian thực hiện chương trình.



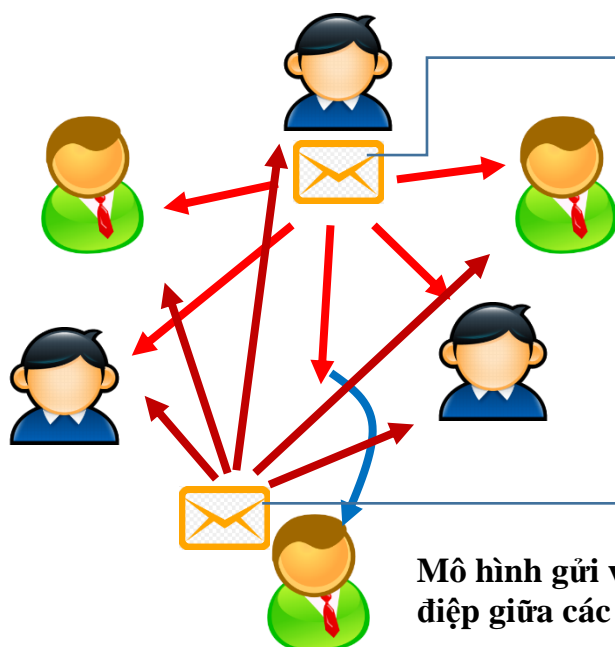
Sự kiện nhận thông điệp. Lệnh sự kiện này cho phép người nhận thông điệp xử lý chính xác



Cửa sổ tạo 1 thông điệp mới.

- Vào lệnh **broadcast**, chọn **New Message**.
- Nhập tên của thông điệp tại vị trí **Message Name**.
- Nháy **OK**.

Như vậy mô hình gửi và nhận thông điệp, xử lý khi nhận thông điệp là các công cụ chính xử lý bài toán truyền thông, quan hệ giữa các nhân vật trong mô hình Scratch.



Mỗi nhân vật được phép tạo và gửi thông điệp của mình ra cộng đồng cho tất cả.

Người nhận đúng thông điệp sẽ xử lý thông điệp này và tạo ra thông điệp trả lời, gửi lại thông điệp này cho cộng

Mô hình gửi và nhận thông điệp giữa các nhân vật.

Xét 1 vài ví dụ sau, em hãy viết suy nghĩ của mình cần đưa ra quan hệ, thông điệp gì giữa các nhân vật để giải quyết các bài toán sau.

Bài toán, vấn đề

Có 1 thầy giáo và nhiều học sinh trên sân khấu.
Tại 1 thời điểm thầy chỉ nói được với 1 học sinh.
Làm thế nào để học sinh nào biết được thầy đang nói và đang hỏi ai?

Quan hệ truyền thông giữa nhân vật

Trên sân khấu có 3 nhân vật: 1 bút chì, 2 nút có màu xanh và đỏ. Khi người dùng nháy lên nút đỏ thì bút chì vẽ màu đỏ, khi người dùng nháy

lên nút xanh, bút sẽ vẽ màu xanh. Mô tả mô hình truyền thông của sân khấu này.

Trên sân khấu có 2 vận động viên, 1 quả bóng và 1 trọng tài. Khi trọng tài thổi còi, 2 vận động viên bắt đầu đá quả bóng sang nhau. Hãy mô tả mô hình truyền thông của sân khấu này.

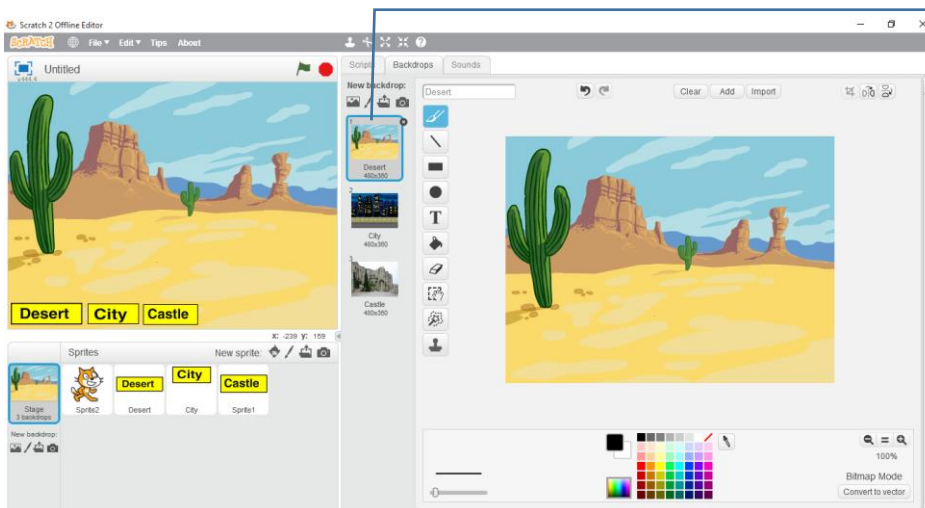
3. Sử dụng "nút lệnh" trong các bài toán giao tiếp

Chúng ta cùng nhau thực hiện một chương trình đơn giản có sử dụng các "nút lệnh" để hiểu thêm vai trò của thông điệp trong các bài toán có giao tiếp.

Thiết lập 1 chương trình bao gồm có 3 nút lệnh: Desert, City và Castle. Khi nháy lên các nút này thì nền sân khấu sẽ thay đổi hình ảnh tương ứng.

Để xây dựng chương trình này, em cần thực hiện các việc sau:

- Thiết lập 1 dự án Scratch mới, bổ sung 3 nền sân khấu và đặt tên Desert, City và Sastle.
- Tạo mới 3 nhân vật có hình nút lệnh có tên Desert, City và Sastle.

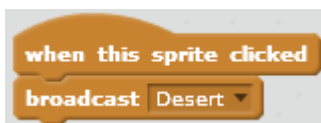


Thiết lập 3 hình nền sân khấu với các tên gọi desert (sa mạc), city (thành phố), castle

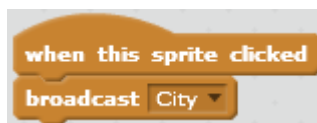
Thiết lập 3 nhân vật dạng nút lệnh với tên gọi desert (sa mạc), city (thành phố), castle

- Làm ẩn nhân vật chính (Mèo con) trên sân khấu.
- Với 3 nhân vật (nút lệnh) vừa khởi tạo, mỗi nhân vật thiết lập 1 thông điệp có tên trùng với tên của nhân vật.
- Xây dựng 3 đoạn chương trình tương ứng với 3 nút lệnh này như sau:

Desert



City



Castle



Bây giờ có thể thiết lập đoạn chương trình của nhân vật chính như sau:



4. Vai trò truyền thông của thông điệp

Trong hoạt động trên em đã thấy rõ vai trò của thông điệp quan trọng như thế nào trong việc điều khiển kết nối giữa các nhân vật trên sân khấu và giải quyết các bài toán nói chung trong môi trường Scratch.

Thông điệp có tính năng truyền thông tin tuyệt đối chính xác.

Thông điệp được truyền qua các Message có tên chính xác, khi nhân vật nhận thông điệp này sẽ xử lý thông qua lệnh **when I receive <message>**, do vậy sẽ tuyệt đối chính xác.

Truyền thông điệp rất nhanh chóng, gần như tức thời.

Thông điệp được gửi bằng lệnh **broadcast** gần như tức thời, không có hiệu ứng chậm chễ về thời gian.

Việc lập trình sử dụng thông điệp truyền thông rất sáng sủa, rõ ràng và đơn giản.

Em hãy thử kiểm tra bằng cách thực hiện chương trình trong mục 3 bằng 2 cách. Cách 1 sử dụng thông điệp, cách 2 sử dụng biến nhớ để điều khiển.

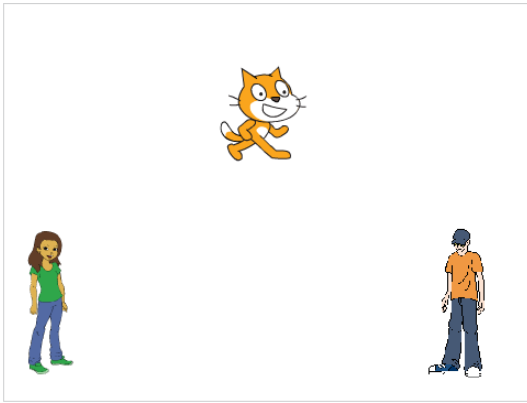
Câu hỏi và bài tập

1. Một nhân vật có thể tạo đồng thời nhiều nhiều thông điệp và gửi đồng thời tất cả các thông điệp đó được hay không?
2. Một nhân vật có thể nhận và xử lý đồng thời 2 thông điệp từ 2 địa chỉ khác nhau được hay không? Cho ví dụ.
3. Viết chương trình cho bài yêu cầu sau.

Trên màn hình có 3 nhân vật: Mèo con, Lan và Hùng đang đứng ở vị trí như trong hình. Cả Lan và Hùng đều muốn Mèo chạy về phía mình.

Khi nháy chuột lên Lan, Mèo sẽ quay về hướng Lan và chạy về phía Lan.

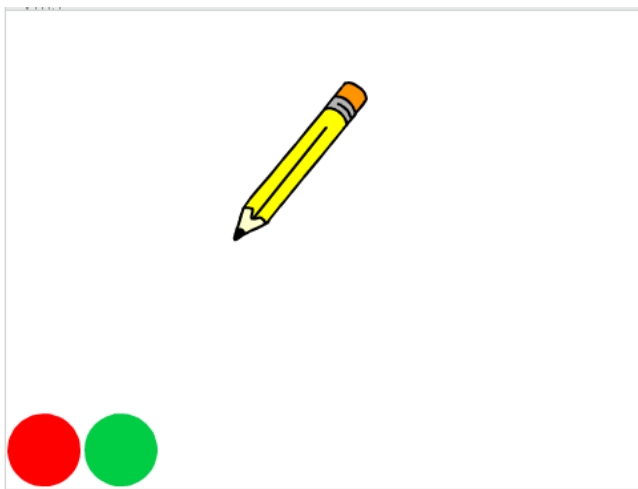
Khi nháy chuột lên Hùng, Mèo sẽ quay về hướng Hùng và chạy về phía Hùng.



Vị trí ban đầu của Mèo con, Lan và Hùng.

Mở rộng

1. Mở rộng bài tập lớn của bài Vẽ hình 2, em hãy thiết lập chương trình Vẽ tự do sau.



Yêu cầu:

- Giao diện bao gồm 1 cây bút chì và 2 nút Xanh, Đỏ góc dưới màn hình.
- Khi di chuyển chuột trên màn hình, bút vẽ luôn đi theo vị trí con trỏ.
- Nhấn phím Space sẽ bật / tắt chế độ vẽ.
- Nháy chuột lên nút màu xanh hoặc đỏ sẽ có tác dụng đổi màu vẽ thành xanh hoặc đỏ.

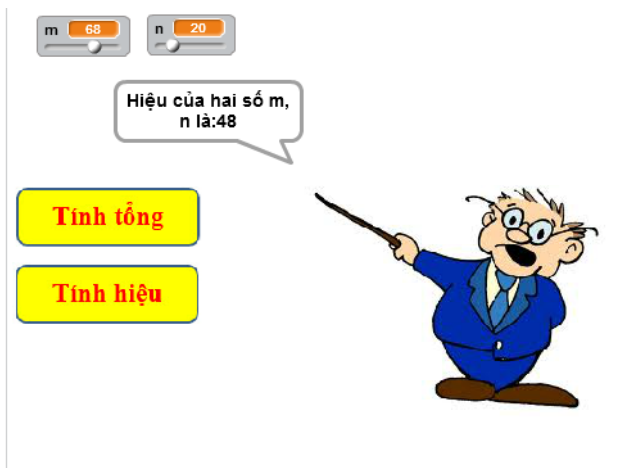
2. Thiết lập chương trình đơn giản sau.



Yêu cầu:

- Giao diện phần mềm bao gồm: 1 Thầy giáo, 2 nút lệnh "Tính tổng" và "Tính hiệu". 2 giá trị số m, n có thể nhập, điều chỉnh ngay trên màn hình.
- Khi người dùng nháy lên các nút lệnh "Tính tổng" và "Tính hiệu", thầy giáo sẽ thông báo kết quả của phép tính tương ứng ngay trên màn hình.

Ví dụ kết quả của phép **Tính hiệu** sẽ như sau:



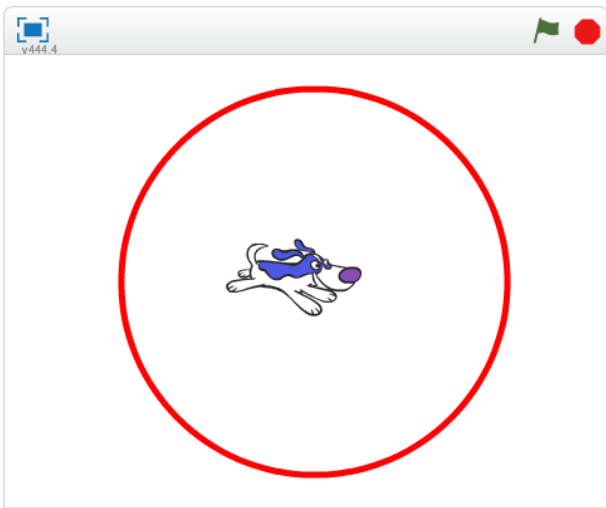
Bài 11. Cảm biến

Mục đích

- Khái niệm và mô hình cảm biến trong Scratch.
- Hiểu được các lệnh cảm biến đơn giản: tọa độ, khoảng cách, màu sắc.
- Cảm biến va chạm giữa các nhân vật.
- Mô hình biến nhớ chung và riêng.

Bắt đầu

1. Em có nghe, đọc sách báo nói về 6 giác quan, em hãy nhớ lại đó là các giác quan nào?
2. Giả sử em muốn viết 1 chương trình cho 1 con vật (nhân vật) của em chạy tự do trong 1 vòng tròn màu đỏ, không cho nhân vật chạy ra ngoài. Em sẽ phải làm gì?



3. Trong các câu sau, câu nào có liên quan đến sự kiện, câu nào liên quan đến cảm biến?

- Sáng chủ nhật em đến trường tập hát.
- Đi trên đường em bị lóa mắt vì mặt trời chiếu thẳng vào mắt.
- Bạn An rất thích màu đỏ.
- Sáng em thức giấc đúng 6h khi nghe chuông báo thức.
- Sáng nào bố mẹ em cũng dậy sớm vào lúc 6h để tập thể dục.
- Khi nhận được thư mẹ, bố em lập tức viết thư trả lời.
- Khi nhấn phím Enter, dữ liệu được nhập sẽ đưa lên mạng Internet.
- Em bé rất sợ tiếng ồn, mỗi khi nghe tiếng động lớn thì khóc.

Qua các ví dụ trên em hãy chỉ ra sự giống nhau và khác nhau giữa 2 khái niệm **Sự kiện** và **Cảm biến**. Hãy điền suy nghĩ của em vào bảng sau:

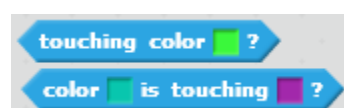
Vấn đề / câu hỏi	Sự kiện	Cảm biến
Hành động xuất phát từ bên ngoài hay từ bản thân nhân vật		
Sự việc có phụ thuộc vào bản thân nhân vật hay không?		
Sự việc có được lên kế hoạch trước hay không?		
Sự việc có được thực hiện theo thời gian định trước hay không?		
Sự việc có các tham số liên quan đến các giác quan con người hay không?		

Nội dung bài học

1. Các câu lệnh cảm biến trong Scratch.

Các lệnh cảm biến nằm trong nhóm lệnh Sensing (Cảm biến). Các lệnh này thường điều khiển, xử lý thông qua các thông số mang tính cảm nhận, định tính và không phụ thuộc vào các tác động bên ngoài (sự kiện). Chính vì vậy chúng ta gọi chúng là các lệnh cảm biến. Để so sánh cảm biến và sự kiện chúng ta thường nghĩ đến cảm nhận của các giác quan của con người và các sự kiện độc lập từ bên ngoài.

Trong Scratch, các lệnh cảm biến thường là các biểu thức, hàm số. Các hàm số này sẽ trả lại giá trị của các thông số mang tính "cảm biến", chỉ phụ thuộc vào nội tại nhân vật. Bảng sau cho ta một tổng quan về các hàm hay dùng nhất trong nhóm lệnh Cảm biến.



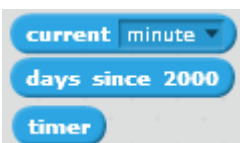
Cảm biến màu sắc.



Cảm biến khoảng cách, va chạm.



Cảm biến chuột và bàn phím.



Cảm biến thời gian.



Thông tin thuộc tính của nhân vật và sân khấu.



Lệnh hỗ trợ hội thoại nhập dữ liệu trực tiếp từ bàn phím.


2. Cảm biến màu sắc

Cảm biến màu sắc là những cảm nhận, nhận biết liên quan đến màu sắc. Em hãy cùng thực hiện các hoạt động sau để hiểu các lệnh cảm biến liên quan đến màu sắc trong Scratch.

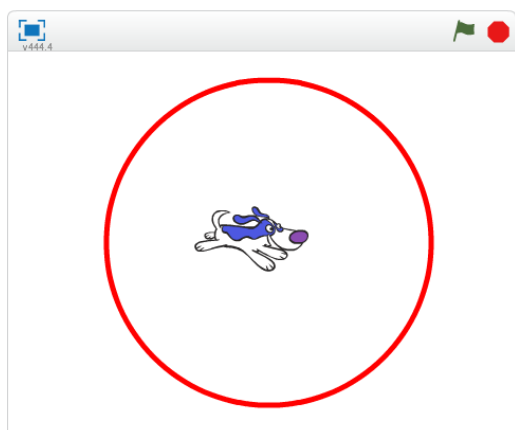
Nhân vật chính của chúng ta là 1 con chó nhỏ hiện đang nằm giữa sân khấu.

Yêu cầu thực hiện chương trình.

Cho con chó nhỏ chạy theo các hướng ngẫu nhiên, mỗi lần chạy 10 bước và nghỉ 0.3 giây. Nếu chó chạm vạch đỏ thì lập tức bị đặt vào vị trí tâm của vòng tròn, và chương trình lại tiếp tục.

Em hãy sử dụng lệnh cảm biến  để điều khiển nhận biết khi chó chạm vạch đỏ. Để chọn màu cảm biến chính xác em thực hiện: sau khi kéo thả lệnh trên ra cửa sổ lệnh, em nhấp chuột lên ô màu bên phải lệnh, sau đó nhấp chuột lên vạch đỏ trên sân khấu để chọn màu cảm biến.

Chương trình được mô tả trong hình dưới đây.



3. Cảm biến khoảng cách, va chạm

Trong môi trường Scratch có 2 lệnh liên quan đến cảm biến khoảng cách, va chạm.



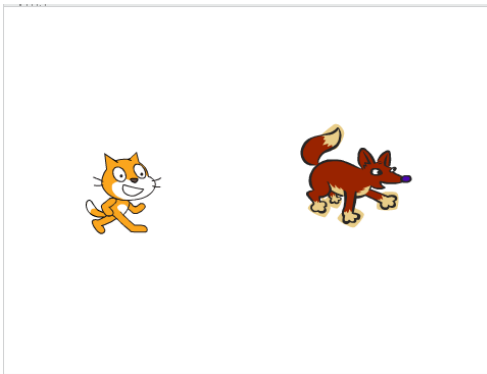
Lệnh này sẽ trả lại giá trị đúng nếu nhân vật hiện thời va chạm với 1 nhân vật khác / hoặc va chạm với con trỏ chuột / hoặc va chạm với cạnh sân khấu. Ngược lại nó trả về giá trị sai.



Lệnh này sẽ tính và trả về giá trị khoảng cách từ nhân vật hiện thời đến 1 nhân vật khác / hoặc đến vị trí con trỏ chuột.

Chúng ta cùng tìm hiểu cảm biến khoảng cách, va chạm thông qua ví dụ sau.

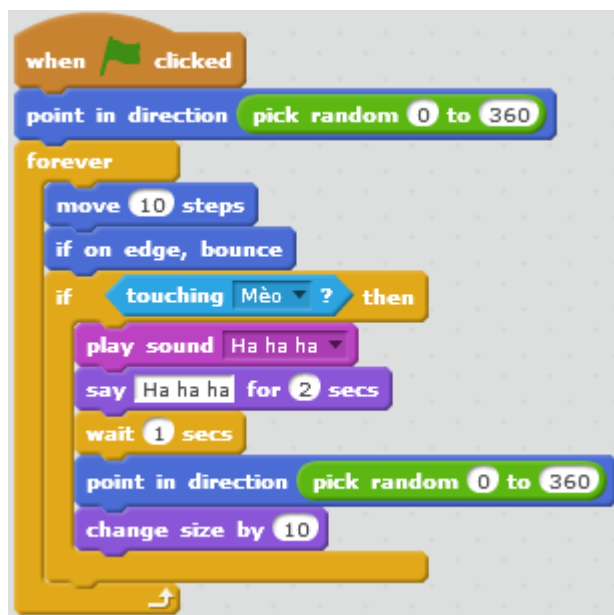
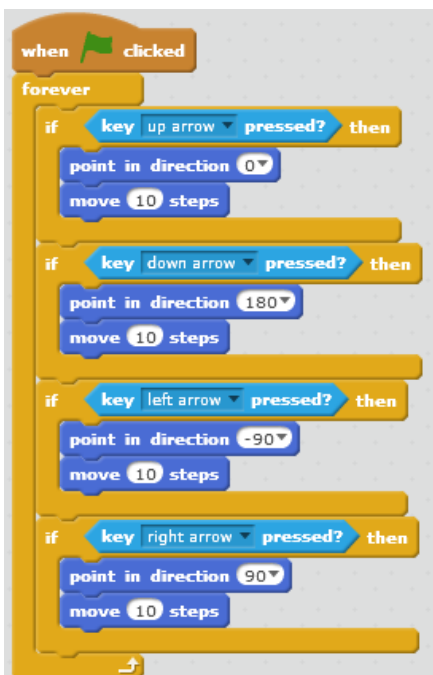
Trò chơi Chó đuổi Mèo.



Trên sân khấu có 2 nhân vật chính là Chó và Mèo. Mèo được điều khiển chuyển động bằng bàn phím. Người dùng sử dụng các phím up, down, left, right để dịch chuyển mèo theo các hướng lên, xuống, trái, phải tương ứng. Cần điều khiển để Mèo tránh gặp Chó.

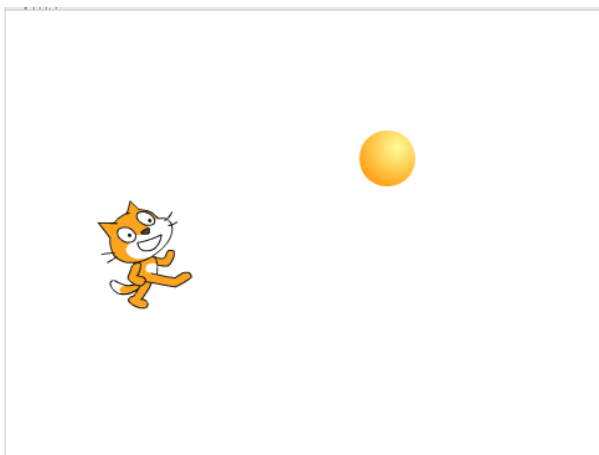
Chó chuyển động ngẫu nhiên trên màn hình. Nếu Chó va chạm với Mèo, Chó sẽ kêu "Ha ha ha" và tự to lên 10%.

Chương trình riêng cho Mèo và Chó được thể hiện trong hình sau. Có thể tạo thêm âm thanh tiếng kêu mừng rỡ của Chó khi bắt được Mèo.



4. Cảm biến chuột và bàn phím

Em hãy thiết kế trò chơi đơn giản Mèo đuổi Bóng.



Yêu cầu của trò chơi như sau:

Mèo sẽ luôn hướng đến Bóng và chạy về phía Bóng. Nếu bắt được bóng thì Mèo sẽ kêu lên tiếng "Bắt được bóng rồi" (có thể thêm âm thanh để trò chơi thêm hấp dẫn) và kết thúc.

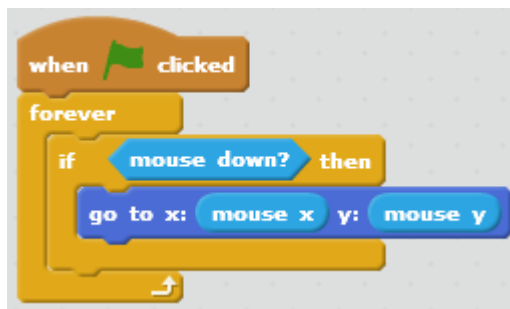
Người chơi sẽ dùng chuột để điều khiển quả bóng. Nháy chuột lên màn hình sẽ lập tức làm cho Bóng dịch chuyển nhanh đến vị trí chuột.

Người dùng cũng có thể click và rê chuột trên màn hình để kéo thả Bóng chạy trốn khỏi Mèo.

Em cần sử dụng lệnh cảm biến **touching** và lệnh chuyển động **point towards** để điều khiển Mèo luôn hướng về Bóng và chạy về phía Bóng.

Với Bóng, em cần sử dụng các lệnh cảm biến chuột là **mouse down?** để xác định thời điểm người dùng nhấn chuột, dùng các hàm **mouse x**, **mouse y** để tính tọa độ chính xác của chuột tại thời điểm nhấn chuột.

Chương trình dành cho Bóng (trái) và Mèo (phải) như hình dưới đây.



5. Cảm biến thời gian

Các hàm cảm biến thời gian trong Scratch bao gồm biến **timer** tự động đếm thời gian theo giây, và hàm **current minute** trả lại các giá trị của thời gian hệ thống. Lệnh **reset timer** có tác dụng đặt lại từ đầu cho biến đếm **timer**.

Áp dụng: thiết kế chương trình mô tả đồng hồ kim chạy trên màn hình.

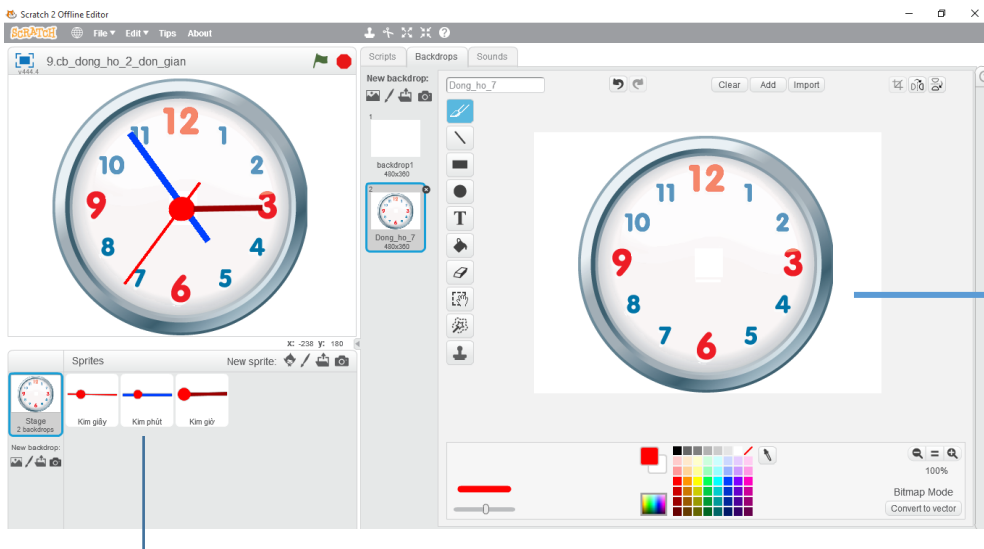


Thiết lập dự án chương trình như sau:

- Sân khấu nền là mặt đồng hồ kim (analog clock)
- Có 3 nhân vật là: kim giây, kim phút, kim giờ.

Chú ý thiết lập tâm của các kim này tại vị trí tâm hình tròn.

- Đưa tất cả các kim về vị trí xuất phát: tâm của các kim này trùng với tâm của sân khấu.

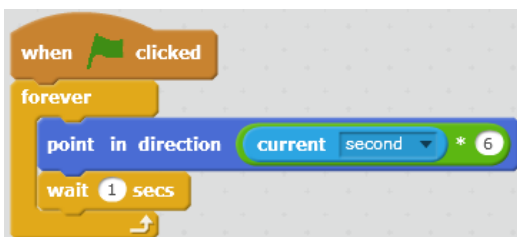


Vẽ hình nền sân khấu là mặt tròn của đồng hồ kim.

Thiết lập 3 nhân vật là kim giây, kim phút, kim giờ.

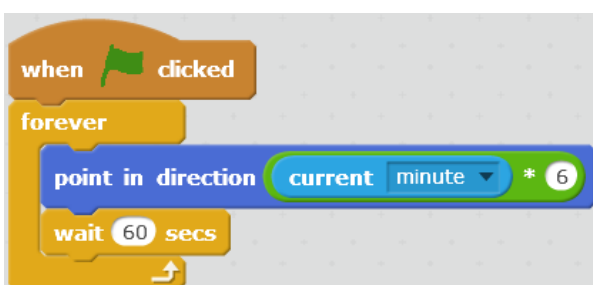
Để viết chương trình điều khiển, em sẽ viết cho từng kim đồng hồ. Vấn đề là tính góc quay của từng kim đồng hồ chính xác theo thời gian. Sử dụng hàm **current second** để tính giá trị của giây, phút, giờ theo thời gian hệ thống.

Chương trình cho kim giây.



Thời gian theo giây có giá trị từ 0 đến 59, tương ứng với giá trị góc quay là 0 đến 360. Do đó mỗi giây, góc cần quay kim giây là $\text{<giá trị giây>} * (360/60) =$

Chương trình cho kim phút.



Thời gian theo phút có giá trị từ 0 đến 59, tương ứng với giá trị góc quay là 0 đến 360. Do đó mỗi phút, góc cần quay kim phút là $\text{<giá trị phút>} * (360/60) =$

Chương trình cho kim giờ.

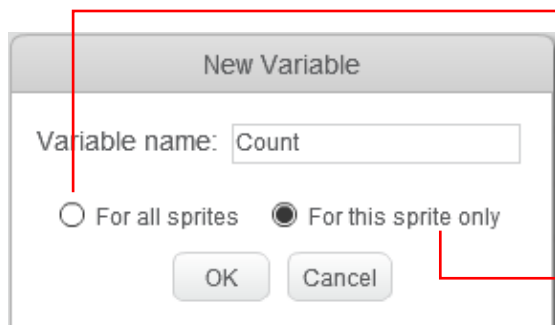


Thời gian theo giờ có giá trị từ 0 đến 23, tương ứng với giá trị góc quay là 0 đến $360 \times 2 = 720$. Do đó mỗi giờ, góc cần quay kim giờ là $\text{<giá trị giờ>} * (720/24) =$

Câu hỏi: em có thấy chương trình chạy chính xác không?

6. Kiểu biến nhớ: dùng chung và riêng

Khi khởi tạo 1 biến nhớ trong Scratch, chúng ta cần lựa chọn 1 trong 2 kiểu, biến nhớ dùng chung và biến nhớ dùng riêng.



Biến nhớ dùng chung: tất cả mọi nhân vật đều có quyền nhập, thay đổi giá trị của biến nhớ này. Biến nhớ chung còn có tên gọi **Global Variable**.

Biến nhớ dùng riêng: chỉ nhân vật là chủ của biến này có quyền nhập, thay đổi giá trị, các nhân vật khác chỉ có quyền xem, khai thác giá trị. Biến nhớ riêng còn có tên gọi **Local Variable**.

Chú ý: Nếu chúng ta khởi tạo 1 biến nhớ riêng cho 1 nhân vật thì biến nhớ này chỉ xuất hiện trong bảng lệnh của nhân vật này mà không hiện trong bảng lệnh của các nhân vật khác. Tuy nhiên biến nhớ này sẽ xuất hiện như 1 thuộc tính của nhân vật này. Các nhân vật khác muốn truy cập giá trị này cần thực hiện thông qua lệnh, hàm cảm biến truy xuất thuộc tính của nhân vật.

Trong ví dụ trên, biến Count là riêng được khởi tạo cho nhân vật Lan thì các nhân vật khác sẽ truy cập biến nhớ này thông qua lệnh .

Câu hỏi và bài tập

1. Mở rộng trò chơi Chó đuổi Mèo, bổ sung thêm biến đếm cho phép Chó va chạm vào Mèo 3 lần thì kết thúc chương trình.
2. Mở rộng trò chơi Mèo bắt Bóng, bổ sung thêm biến đếm thời gian từ khi bắt đầu cho đến khi Mèo bắt được bóng, thể hiện thời gian này trên màn hình.

Mở rộng

1. Trò chơi **Ăn táo**. Viết chương trình điều khiển con Cánh cam bằng bàn phím, chuyển động đến vị trí Quả táo ở góc phải dưới sân khấu, dọc đường đi không được va chạm với bất cứ vết màu nào trên màn hình. Nếu chạm vào các vết màu này thì bị thua.



2. Mở rộng chương trình thể hiện đồng hồ kim chạy chính xác trên màn hình.



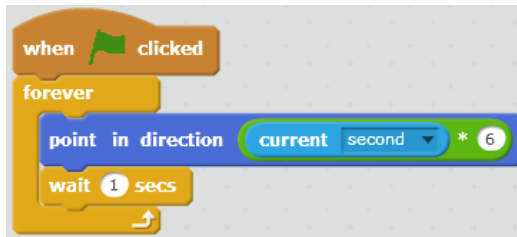
Gợi ý:

Cần điều khiển kim phút chạy theo từng giây và kim giờ theo từng phút.

Muốn vậy đối với kim phút và kim giờ cần thiết lập thêm biến nhớ (riêng) angle cho mỗi nhân vật này và tính toán chính xác góc cần quay theo thời gian.

Bảng sau mô tả chi tiết các điều khiển các kim giờ, phút, giây chính xác.

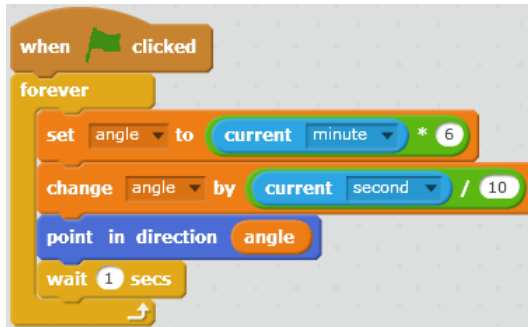
Điều khiển kim giây theo từng giây.



Kim giây không cần thay đổi điều khiển, cập nhật theo từng giây.

Thời gian theo giây có giá trị từ 0 đến 59, tương ứng với giá trị góc quay là 0 đến 360. Do đó mỗi giây, góc cần quay kim giây là $\langle \text{giá trị giây} \rangle * (360/60) = \langle \text{giây} \rangle * 6$.

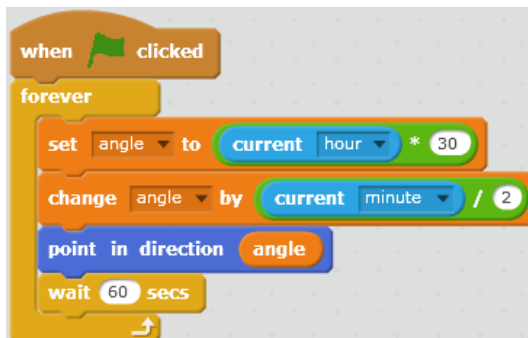
Điều khiển kim phút theo từng giây.



Nâng cấp điều khiển kim phút theo từng giây.

Thời gian theo phút có giá trị từ 0 đến 59, tương ứng với giá trị góc quay là 0 đến 360. Do đó mỗi phút, góc cần quay kim phút là $\langle \text{giá trị phút} \rangle * (360/60) = \langle \text{phút} \rangle * 6$.

Điều khiển kim giờ theo từng phút.



Nâng cấp điều khiển kim giờ theo từng phút.

Thời gian theo giờ có giá trị từ 0 đến 23, tương ứng với giá trị góc quay là 0 đến $360 \times 2 = 720$. Do đó mỗi giờ, góc cần quay kim giờ là $\langle \text{giá trị giờ} \rangle * (720/24) = \langle \text{giờ} \rangle * 30$.

CHƯƠNG 4: SCRATCH NÂNG CAO

Bài 12. Xử lý số 1

Mục đích

- Biến nhớ và vai trò của biến nhớ.
- Phân biệt biến nhớ và biểu thức.
- Các phép tính, tính toán đơn giản với giá trị số và logic.
- Thực hiện một vài bài toán xử lý số đơn giản.

Bắt đầu

1. Chúng ta nhớ lại 2 đoạn chương trình vẽ hình vuông và ngũ giác đều đã học trong bài Vẽ hình 2.

Vẽ hình vuông



Vẽ hình ngũ giác đều



Em có suy nghĩ gì về việc có thể tổng quát hóa đoạn chương trình này để vẽ 1 hình đa giác đều n cạnh bất kỳ?

2. Các biểu thức số học, ví dụ:

$$\frac{7}{8} + \frac{9}{10}$$

$$7 + \frac{5}{6} - \frac{23}{50}$$

$$\sqrt{2} + \frac{1}{2}45 - \cos 15$$

sẽ được tính toán như thế nào trong Scratch?

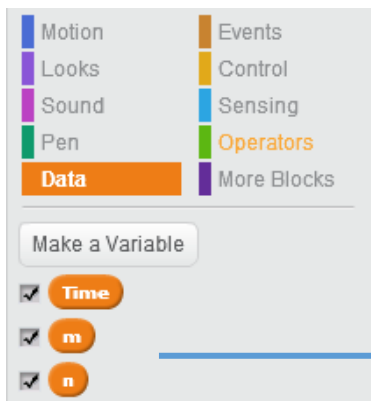
Nội dung bài học

1. Biến nhớ và hàm số

Chúng ta đã làm quen với biến nhớ và các lệnh trả lại giá trị (hay còn gọi là hàm số) trong Scratch. Hoạt động này là một củng cố kiến thức đã biết.

Định nghĩa biến nhớ:

Biến nhớ là 1 vùng trong bộ nhớ, được đặt tên bằng một dãy chữ và số, dùng để lưu trữ các giá trị (số, chữ hoặc logic). Biến nhớ có thể do hệ thống tạo sẵn hoặc do người dùng tự tạo ra để giúp giải quyết các bài toán lập trình.



Các biến nhớ do người lập trình tạo ra sẽ thể hiện trong khung lệnh **Dữ liệu (Data)**.

Qui định đặt tên biến nhớ: Scratch không đặt các điều kiện chặt cho cách đặt tên biến nhớ. Tên biến nhớ phân biệt chữ hoa, chữ thường và có thể chứa dấu cách. Tuy nhiên với các ngôn ngữ lập trình bậc cao thì tên biến nhớ thường được qui định khá chặt chẽ. Các em cần nhớ 1 số qui tắc chung này để áp dụng cho các ngôn ngữ lập trình bậc cao trong tương lai.

Tên biến nhớ phải là dãy chữ hoặc số, không có dấu cách, không nên chứa các ký tự đặc biệt như : ; , \ / + - [] { } (). Nên đặt tên biến nhớ là 1 cụm từ có ý nghĩa để dễ nhớ. Không nên đặt tên biến nhớ với quá nhiều chữ hoa

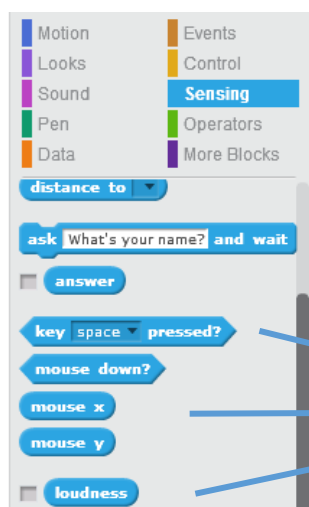
Ví dụ tên biến nhớ như sau là hợp lệ: x1, abc, solution, time_out, click_count, con_meo, chim_bay, thoi_gian,

Các tên biến nhớ như sau thì không nên và không hợp lệ: 12+23, sinx, ANHSang, C://thu_muc,

Chú ý: các biến nhớ dùng chung cần có tên khác nhau từng đôi một trong chương trình. Biến nhớ riêng nhân vật có thể đặt tên trùng nhau.

Định nghĩa hàm số:


Hàm số do người dùng hoặc hệ thống tạo ra, được đặt tên bằng một dãy chữ và số, dùng để tính toán một giá trị nào đó, được lưu trữ trong bộ nhớ và được sử dụng tương tự như các biến nhớ trong các câu lệnh.







Các hàm số được tạo sẵn trong hệ thống, mang ý nghĩa khác nhau và đều trả lại các giá trị có ý nghĩa trong quá trình giải quyết vấn đề.

Hàm số có thể có hoặc không có các tham số đầu vào. Ví dụ.

- Hàm **mouse x** trả lại tọa độ X của vị trí con chuột hiện thời. Hàm này không có tham số.

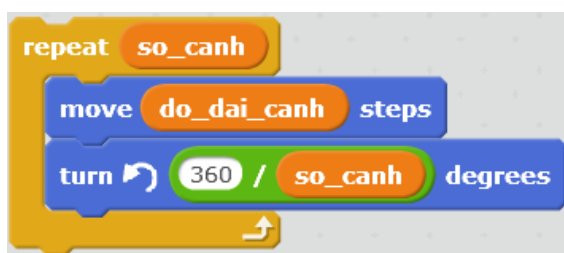
- Hàm  có 1 tham số là tên phím tương ứng trên bàn phím. Hàm sẽ trả lại giá trị Đúng nếu phím này được nhấn, và trả lại Sai nếu phím này không được nhấn.

- Hàm tính tổng  có 2 tham số đầu vào là 2 số. Hàm sẽ trả lại giá trị là tổng của hai số này. Tham số đầu vào có thể lấy giá trị cụ thể hoặc lấy từ các biến nhớ hoặc hàm số khác.

Ví dụ các cách truyền tham số cho hàm tính tổng này: , , .




Biến nhớ có rất nhiều ý nghĩa trên thực tế. Chúng ta hãy cùng xét 1 ví dụ sau, lấy cảm hứng từ ví dụ vẽ các hình tứ giác, ngũ giác đều trong phần mở đầu của bài học này.

Sử dụng các biến nhớ **so_canh** để lưu số cạnh của đa giác muốn vẽ, biến nhớ **do_dai_canh** để lưu độ dài cạnh đa giác, khi đó đoạn chương trình vẽ 1 đa giác đều tổng quát như sau:



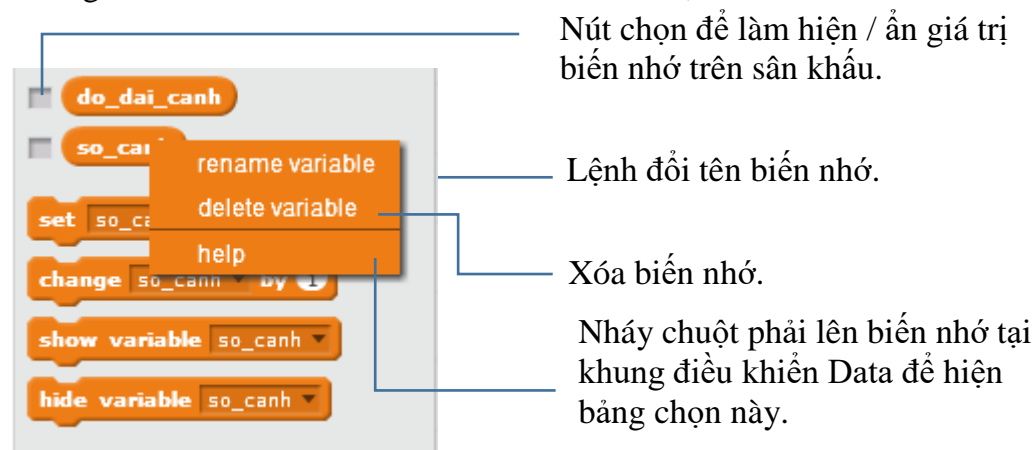
2. Các thao tác làm việc với biến nhớ

Đối với biến nhớ, 2 lệnh quan trọng nhất là lệnh gán giá trị và lệnh thay đổi giá trị của biến nhớ. Các lệnh này được thực hiện trong Scratch như sau, so sánh với cách viết trong 1 số ngôn ngữ lập trình bậc cao khác.

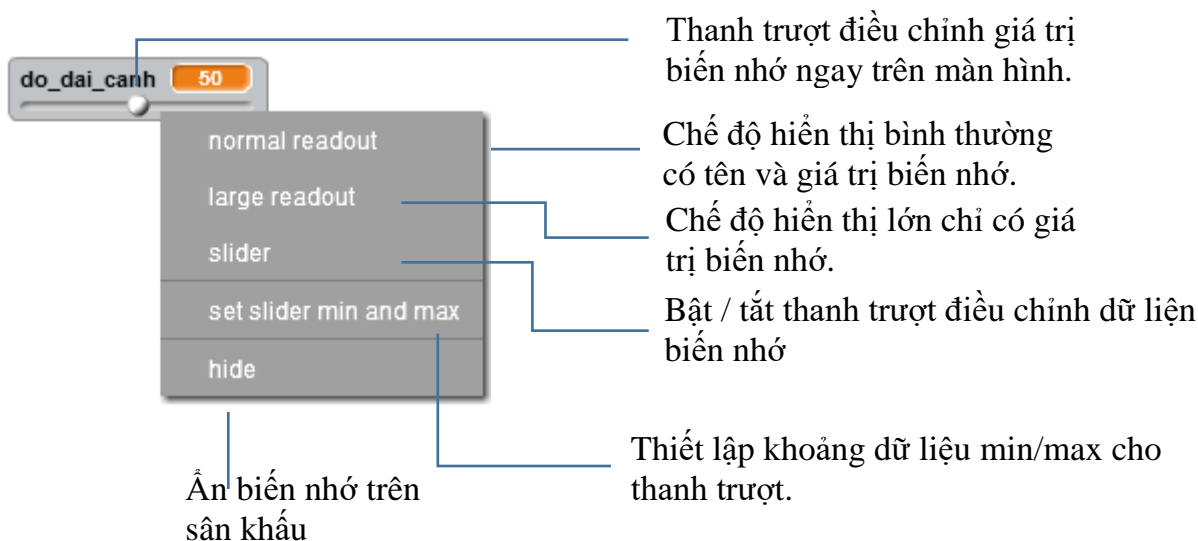
Lệnh	Scratch	Pascal	C, Java
Gán giá trị cho biến nhớ		<code>m:=10;</code>	<code>m = 10;</code>
Thay đổi giá trị của biến nhớ + 1		<code>m:=m+1;</code>	<code>m++;</code>
Thay đổi giá trị của biến nhớ - 1		<code>m:=m-1;</code>	<code>m--;</code>

Một số lệnh, thao tác khác với biến nhớ.

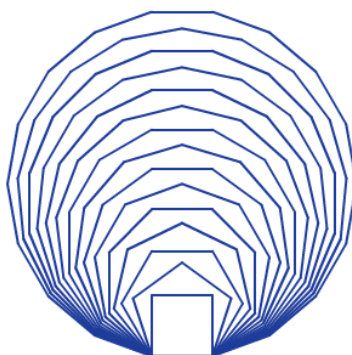
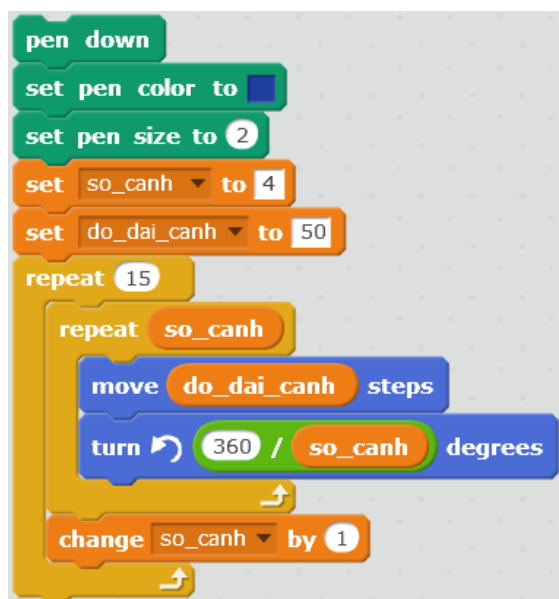
Hiện giá trị biến nhớ trên sân khấu, xóa biến nhớ, đổi tên biến nhớ.



Khi biến nhớ đã được hiển thị trên sân khấu thì có thể thực hiện các thao tác sau bằng cách nhấp chuột phải lên vị trí biến nhớ.



Đoạn chương trình sau mô tả bài toán vẽ liên tục các hình đa giác đều với số cạnh tăng dần từ 4 đến 18. Kết quả thể hiện trong hình phải.



3. Kiểu dữ liệu trong Scratch

Khi 1 biến nhớ được khởi tạo trong Scratch, chúng ta không cần khai báo biểu dữ liệu cho biến nhớ này. Trong quá trình làm việc, tùy thuộc vào dữ liệu nhập, biến nhớ tự động nhận biết 1 trong 3 kiểu dữ liệu sau:

Số (Number). Bao gồm số nguyên và thập phân.

Logic (Boolean). Kiểu dữ liệu chỉ có 2 giá trị Đúng (true) và Sai (false).

Xâu (String). Kiểu dữ liệu là dãy các chữ, ký tự.

Ví dụ nếu biến nhớ được khởi tạo thì có thể gán cho n các giá trị sau:



Gán giá trị số cho n



Gán chuỗi ký tự cho n

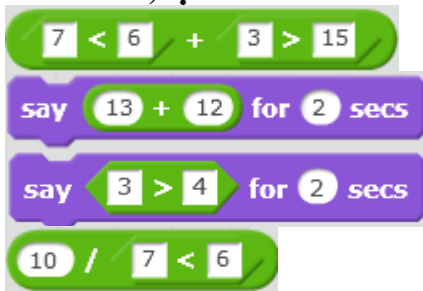


Gán giá trị logic cho n

Em hãy thực hiện các lệnh sau và điền kết quả vào cột phải trong bảng dưới đây, từ đó có nhận xét gì về các kiểu dữ liệu và cách thể hiện chúng trong Scratch.

Biểu thức, lệnh

Kết quả



4. Các phép tính đơn giản với số trong Scratch

Nhóm lệnh Operators (màu xanh lá cây) bao gồm các lệnh, phép tính làm việc với dữ liệu số, logic và chuỗi ký tự. Trong bài học này chúng ta làm quen với các phép tính đơn giản với số.

Phép tính cộng, trừ nhân, chia số thập phân



(hàm) phép cộng 2 số



(hàm) phép trừ 2 số



(hàm) phép nhân 2 số



(hàm) phép chia 2 số, kết quả thập phân.

Làm tròn số



(hàm) làm tròn số của 1 số, lấy số nguyên gần nhất số này.



(hàm) làm tròn (lên) của 1 số, lấy số nguyên gần nhất trên số này.



(hàm) làm tròn (dưới) của 1 số, lấy số nguyên gần nhất dưới số này.

Phép chia số nguyên



(hàm) lấy số dư của 2 số nguyên



(hàm) lấy thương nguyên của 2 số nguyên

Phép tính lũy thừa



(hàm) lấy lũy thừa của 2 số thập phân n^a



(hàm) lấy lũy thừa của 2 số nguyên n^a

Phép lấy căn thức, trị tuyệt đối



(hàm) lấy giá trị tuyệt đối của 1 số.



(hàm) lấy căn bậc 2 của 1 số dương a
($a^{1/2}$)



(hàm) lấy căn bậc n của 1 số a ($a^{1/n}$)

Ví dụ:

$$1/2 + 3/4$$



$$m^2 + n^2 - mn$$



$$|x| + |x + 2| - |x - 1|$$



$$x^2 - x + 1$$

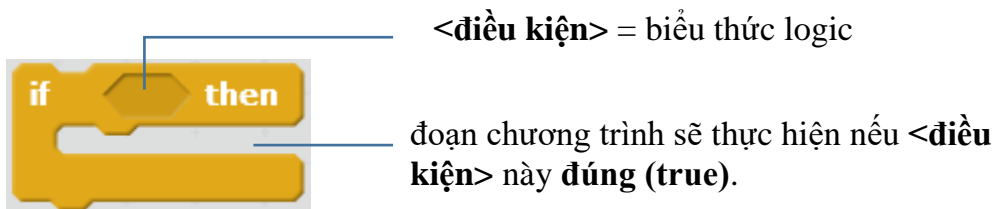


$$(a + b)^2$$

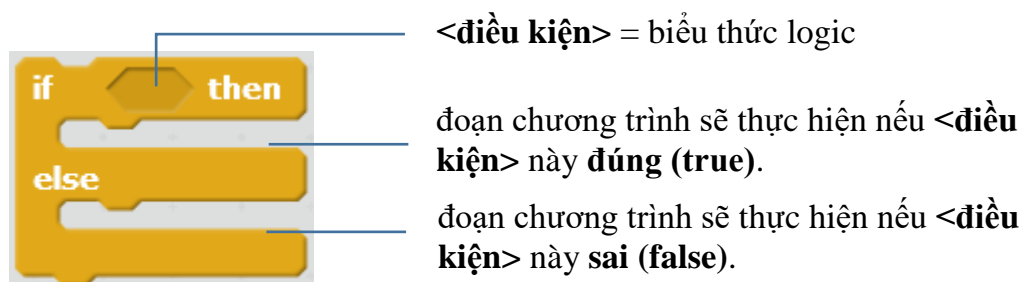


5. Kiểu dữ liệu logic

Em đã được làm quen với lệnh điều khiển có điều kiện **if .. then**. Biểu thức sau **if** là 1 điều kiện logic. Khi thực hiện, lệnh sẽ kiểm tra điều kiện này, nếu **đúng (true)** thì sẽ thực hiện đoạn lệnh sau từ khóa **then**, nếu **sai (false)** thì bỏ qua lệnh này.



Tương tự chúng ta có 1 lệnh tổng quát hơn: lệnh điều khiển đầy đủ có điều kiện **if .. then ... else**. Biểu thức sau **if** là 1 điều kiện logic. Khi thực hiện, lệnh sẽ kiểm tra điều kiện này, nếu **đúng (true)** sẽ thực hiện đoạn lệnh sau từ khóa **then**, nếu **sai (false)** sẽ thực hiện đoạn lệnh sau từ khóa **else**.



Các toán tử, phép tính với biểu thức và biến nhớ logic trong Scratch bao gồm.

Phép toán logic



Kết quả

(hàm) toán tử logic **and** (và).

Trả lại true (đúng) khi và chỉ khi cả 2 biểu thức thành phần đều true (đúng).

(hàm) toán tử logic **or** (hoặc).

Trả lại true (đúng) nếu 1 trong 2 biểu thức thành phần là true (đúng), các trường hợp khác sẽ trả lại false (sai).

(hàm) toán tử **not** (phủ định / không).

Trả lại true (đúng) nếu biểu thức là false (sai) và ngược lại, trả lại false nếu biểu thức là true (đúng).

Bảng cụ thể các phép tính logic **and**, **or**, **not** như sau.

Toán tử and (và)

A	B	A and B
true	true	true
true	false	false
false	true	false
false	false	false

Toán tử or (hoặc)

A	B	A or B
true	true	true
true	false	true
false	true	true
false	false	false

Toán tử not (không)

A	not A
true	false
false	true

6. Biểu diễn biểu thức logic

Em hãy biểu diễn các điều kiện logic sau trong môi trường Scratch.

$a > 10$ và $a < 100$



$m \geq 100$

$0 < m < 20$

ngày = 10 và tháng = 3 và năm = 2016

m là số lẻ và $0 < m < 100$

m là số chẵn và $m > 20$

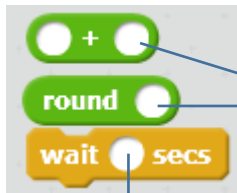
Chú ý đến các ô nhập trực tiếp dữ liệu trên các dòng lệnh. Có 2 loại ô chữ nhật và ô tròn.

Ô vuông, chữ nhật



Ô dữ liệu vuông, chữ nhật có thể nhập số, chữ hoặc biểu thức logic.

Ô tròn



Ô dữ liệu tròn chỉ thể nhập số hoặc biểu thức logic, không nhập được chữ.

Câu hỏi và bài tập

1. Viết hàm kiểm tra xem năm **year** có phải là năm nhuận không?

Tính chất toán học của năm nhuận: là năm chia hết cho 4 và không chia hết cho 100, hoặc chia hết cho 400.

2. Viết chương trình nhập hai giá trị month (tháng) và year (năm) từ bàn phím và kiểm tra xem giá trị nhập vào có hợp lệ hay không?

3. Biểu diễn các biểu thức sau dùng lệnh Scratch\

$$(7/8) + (9/10)$$

$$ax^2 + bx + c$$

$$m*n - 2(m+n)$$

$$|x| - |2x - 1|$$

$$10^2 + (3 + 7/8)$$

4. Biểu diễn các điều kiện logic sau bằng lệnh Scratch

$$(m > 100) \text{ và } (n < 100)$$

$$mn < 0$$

$$(m + n) > (m^2 + n^2)$$

$$(x \geq 2) \text{ hoặc } (y \leq 10)$$

m không chia hết cho 4

m không bằng 0

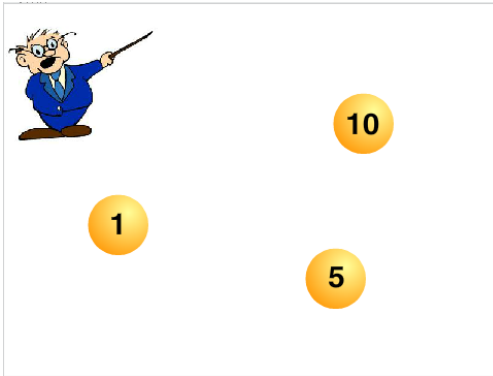
Mở rộng

1. Viết chương trình hiển thị trên màn hình thời gian đầy đủ hệ thống, ví dụ:

Ngày 16 tháng 5 năm 2016

12 giờ 45 phút 24 giây

2. Thiết kế trò chơi đơn giản sau:



Trên sân khấu có 3 quả bóng với các số hiệu 1, 5, 10 chuyển động ngẫu nhiên, lúc ẩn lúc hiện trên màn hình.

Nhiệm vụ của em là nháy chuột chính xác lên các quả bóng, em sẽ được tích lũy điểm theo số được ghi trên quả bóng.

Thời gian cho mỗi lần chơi là 10 giây. Bạn nào đạt được điểm cao thì càng giỏi. Khi kết thúc 1 lần chơi,

giáo viên sẽ thông báo điểm của em trên màn hình.

Bài 13. Xử lý số 2

Mục đích

- Một vài thuật toán đơn giản với số nguyên, phân số.
- Bài toán tìm ước số, tìm số nguyên tố, tìm ước số chung của hai số.
- Bài toán rút gọn phân số, tính ước số chung lớn nhất, bội số chung nhỏ nhất.
- Thiết lập 1 vài trò chơi số đơn giản.

Bắt đầu

1. Em đã bao giờ đặt bút tính $100!$ chưa? Thử lấy bút và tính xem có khó không?
 2. Em hãy nhớ lại một số khái niệm về số học để chuẩn bị cho bài học mới này.
- Thế nào là số nguyên tố, hợp số?
 - Khai triển 1 số tự nhiên thành tích các thừa số nguyên tố.
 - Khái niệm và cách tính ƯSCLN và BSCNN của 2 số tự nhiên.
 - Thế nào là 1 phân số tối giản?

Nội dung bài học

1. Một số thuật toán số đơn giản

Kiểm tra n có chia hết cho m hay không?

Sử dụng hàm lấy số dư phép chia (mod): . Nếu giá trị này = 0 thì n chia hết cho m , ngược lại n không chia hết cho m .



Kiểm tra a có phải là ước số thực sự của m hay không? (Ước thực sự là ước số khác 1 và chính số đó).

Hàm kiểm tra như sau:



Kiểm tra năm **year** có phải là năm nhuận hay không? (năm nhuận là năm chia hết cho 4 và không chia hết cho 100, hoặc chia hết cho 400).

Hàm kiểm tra năm nhuận như sau:



Em hãy viết tiếp các hàm kiểm tra sau:

1. Hàm kiểm tra hai phân số m/n và p/q có bằng nhau hay không?
2. Hàm kiểm tra số p có là bội số của cả hai số n, m hay không?

3. Hàm kiểm tra số p có là ước số đồng thời của cả hai số n, m hay không?

4. Hàm kiểm tra số p có phải là nghiệm của phương trình bậc nhất $px + q = 0$ hay không?

2. Bài toán tìm các ước số thực sự của 1 số tự nhiên cho trước

Bài toán: yêu cầu nhập 1 số tự nhiên n từ bàn phím, chương trình sẽ thông báo tất cả các ước số thực sự của n trên màn hình.

Ví dụ nhập số 100 thì chương trình thông báo các giá trị 1, 2, 5, 10, 20, 25, 50 trên màn hình.

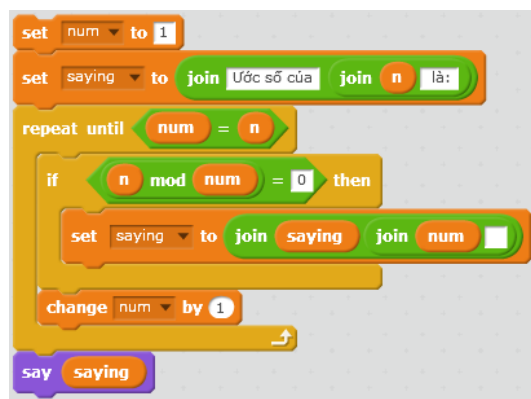
Ý tưởng cách giải (thuật toán) của bài toán này như sau:

Thiết lập 1 biến chạy num , cho num chạy từ 1 đến $n-1$ (chú ý không tính khi $num = n$), với mỗi bước lặp, kiểm tra xem num có phải là ước số của n hay không. Nếu đúng thì đưa num vào DS các số sẽ được in ra kết quả.

Mô tả thuật toán này một cách tường minh:

```
Gán num = 1
Lặp cho đến khi num = n
    Nếu n mod num = 0 thì
        thông báo num
    tăng n lên 1 đơn vị
```

Để viết chương trình trên Scratch, chúng ta thiết lập 2 biến nhớ **num** và **saying**. Biến num để chạy và kiểm tra ước số của n . Biến **saying** để lưu lại các ước cụ thể và hiển thị thông báo ra màn hình.



Câu hỏi: trong bài toán trên nếu thay đổi yêu cầu "ước số thực sự" bằng "ước số" thì chương trình sẽ phải thay đổi như thế nào?

3. Bài toán tìm ƯSCNN của hai số tự nhiên

Bài toán: yêu cầu nhập 2 số tự nhiên n, m từ bàn phím, chương trình sẽ thông báo kết quả là ước số chung lớn nhất của hai số n, m trên màn hình.

Chúng ta sử dụng biến nhớ **gcd** (greatest common divisor) để lưu trữ kết quả phép tính. Thuật toán đơn giản nhất là dùng 1 biến nhớ tạm, ví dụ **num**, cho **num** chạy từ 1 đến n và kiểm tra xem num có là ước số của đồng thời n, m hay không, nếu đúng thì gán giá trị này vào **gcd**.

Cách thứ 2 hay hơn xuất phát từ nhận xét: nếu $n=m$ thì rõ ràng $\text{gcd} = n$. Giả sử $n > m$, khi đó gcd , là ước của n , m nên cũng sẽ là ước của $n-m$, bằng cách thay thế n bằng $n-m$ chúng ta sẽ tìm tiếp với cặp $n-m, m$, cách này đi nhanh hơn đến kết quả cuối cùng. Thuật toán sau dựa trên ý tưởng trên và được coi là thuật toán chuẩn để tính ƯỚC LỚN NHẤT CỦA 2 SỐ TỰ NHIÊN n, m .

Lặp cho đến khi $m = n$

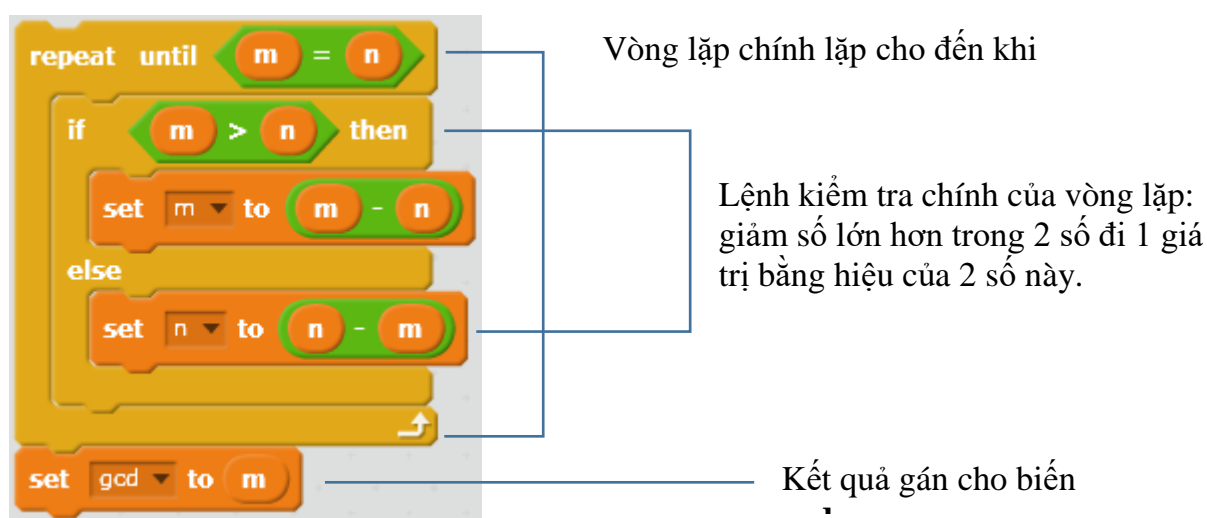
```

    Nếu  $m > n$  thì
         $m = m - n$ 
    còn không thì
         $n = n - m$ 

```

Gán $\text{gcd} = m$.

Đoạn chương trình thể hiện thuật toán trên như sau:



Em hãy hoàn thiện chương trình này trên Scratch.

4. Bài toán kiểm tra số nguyên tố

Bài toán: yêu cầu nhập 1 số tự nhiên n từ bàn phím. Chương trình sẽ thông báo số này là nguyên tố hay hợp số.

Ý tưởng của lời giải này là đếm số các ước số thực sự của n . Nếu số ước số thực sự < 2 thì n sẽ là nguyên tố.

Thuật toán sau đếm số các ước số thực sự của n . Sử dụng biến **count** để đếm số các ước số thực sự của n . Biến nhớ **num** được tạo ra dùng để chạy theo 1 vòng lặp từ 1 đến $n-1$ và kiểm tra xem **num** có phải là ước của n hay không. Nếu **num** là ước của n thì tăng **count** lên 1. Biến **count** được gán giá trị ban đầu = 0.

Gán $\text{num} = 1, \text{count} = 0$

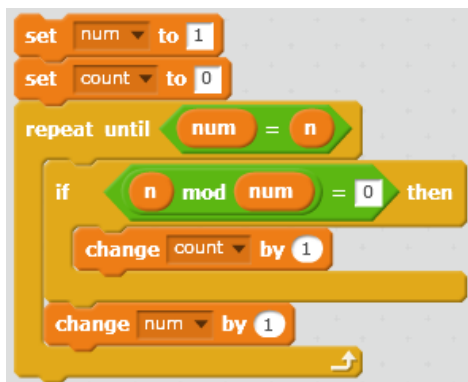
Lặp cho đến khi $\text{num} = n$

```

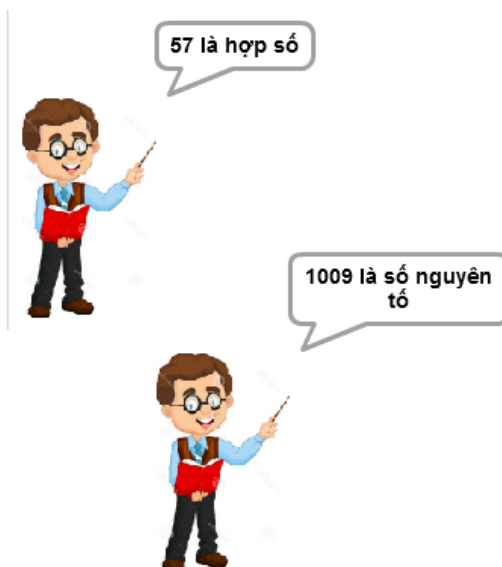
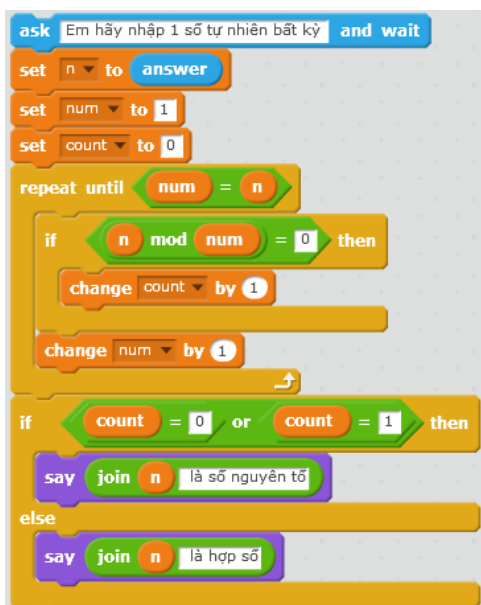
    Nếu  $\text{num}$  là ước của  $n$  thì
        Tăng  $\text{count}$  lên 1 đơn vị.
    Tăng  $\text{num}$  lên 1 đơn vị.

```

Đoạn chương trình trong Scratch mô tả thuật toán trên như sau:



Chương trình hoàn chỉnh cho phép nhập từ bàn phím 1 số và thông báo kết quả trên màn hình.

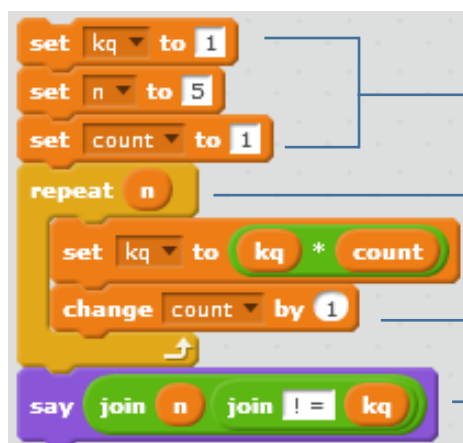


5. Bài toán tính nhanh 100!

Chúng ta đã biết cách tính **n giai thừa** $n! = 1.2.3....n$ (tích của **n** số tự nhiên đầu tiên, tính từ 1). Với **n** lớn thì việc tính **n!** rất khó. Máy tính sẽ giúp em tính rất nhanh bài toán này.

Đoạn chương trình sau mô tả việc tính **n!**

Biến nhớ **count** sẽ tăng dần từ 1 đến **n** để góp phần tính tăng cho kết quả **kq**. Vòng lặp ngoài có **n** vòng. Các biến **count** và **kq** được gán giá trị ban đầu = 1.



Các thiết lập ban đầu.

Vòng lặp chính (n vòng)

Nhóm lệnh chính trong vòng lặp:
tăng **kq** theo phép nhân với **count**
và tăng **count** lên 1.

Hiển thị kết quả

Viết lại chương trình tính $n!$ hoàn chỉnh, giá trị n được nhập từ bàn phím.

Câu hỏi và bài tập

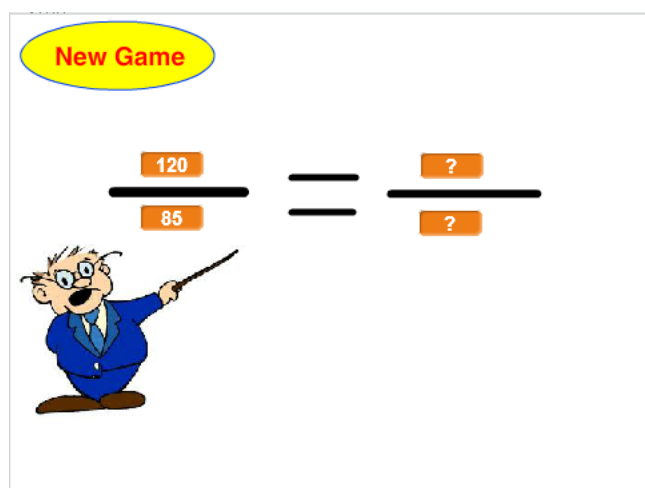
- Viết chương trình nhập số m từ bàn phím và đếm số các ước số của m , tính cả 1 và n . Ví dụ nếu $m = 10$ thì thông báo đáp số là 4.
- Viết đoạn chương trình với dữ liệu đầu vào là n, m và kiểm tra xem n, m có nguyên tố cùng nhau hay không? Hai số được gọi là nguyên tố cùng nhau nếu $U\text{SCLN}$ của chúng $= 1$.
- Viết đoạn chương trình nhập 3 số dương a, b, c và kiểm tra xem có thể vẽ được tam giác ABC với các cạnh có số đo a, b, c hay không.

Mở rộng

- Viết chương trình cho phép nhập số tự nhiên n từ bàn phím và in ra khai triển số n thành tích của các thừa số nguyên tố.

Ví dụ đầu vào là 12 thì thông báo ra là: $12 = 2.2.3$

- Thiết kế trò chơi **Rút gọn phân số** sau.



Mỗi lần nhấp lên nút New Game sẽ bắt đầu trò chơi.

Thầy giáo sẽ đưa ra ngẫu nhiên 1 phân số trên màn hình và yêu cầu người chơi rút gọn phân số này và nhập 2 số p/q là phân số tối giản của phân số trên màn hình.

Sau khi nhập thầy giáo sẽ thông báo ngay là đúng hay sai, nếu sai thì cần nhập lại, nếu đúng thì kết thúc.

Bài 14. Xử lý chuỗi ký tự 1

Mục đích

- Giá trị không là số hoặc logic.
- Các phép tính, tính toán đơn giản với giá trị là chữ hoặc văn bản.
- Thực hiện 1 vài bài toán đơn giản xử lý chữ, ký tự, văn bản.

Bắt đầu

1. Trong các biểu thức giá trị sau, hãy đánh dấu các giá trị là số hoặc logic:

- Heal The World
- $\text{Min} < \text{Max}$
- $321 + 231 + 123$
- Sông Mê Kông dài hơn sông Hồng
- Nếu hôm nay trời mưa em sẽ học ở nhà
- $(x < 10) \text{ and } (x > 1)$

2. Em hiểu thế nào là 1 chuỗi ký tự? Các nội dung sau có phải là 1 dãy ký tự không?

Hanoi1 + Hanoi2

123456789

Hà Nội là thủ đô của nước Việt Nam

$$x^2 + y^2 + z^2 = 192$$

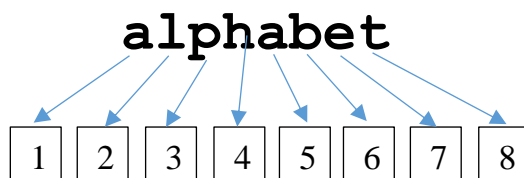
Giải phương trình $ax^2 + bx + c = 0$

Nội dung bài học

1. Cách chuỗi ký tự được lưu trữ trong máy tính

Chuỗi ký tự được hiểu là 1 dãy các ký tự, ví dụ "Hà Nội", "English", "letters", "hòa bình". Dãy các ký tự tạo nên 1 chuỗi sẽ được đánh số từ 1.

Ví dụ từ alphabet.

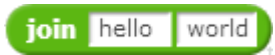




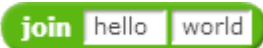

Chú ý:



- Dãy ký tự của chuỗi có thể chứa các ký tự đặc biệt như : ; / \ { } [] ()
 - Dấu cách (Space) cũng được coi là 1 ký tự, ta dùng ký hiệu \sqcup để chỉ dấu cách.
- Tổng số các ký tự của 1 chuỗi được gọi là **độ dài** của chuỗi ký tự này.



2. Các hàm xử lý chuỗi ký tự trong Scratch

Trong môi trường Scratch có 3 hàm xử lý chuỗi ký tự sau:

-  (hàm) toán tử nối 2 chuỗi ký tự.
-  (hàm) trả lại ký tự thứ <1> của chuỗi ký tự <world>
-  (hàm) trả lại độ dài của chuỗi ký tự <world>

Toán tử  có tác dụng nối 2 chuỗi ký tự và trả lại kết quả là chuỗi được kết nối. Ví dụ lệnh  sẽ trả lại chuỗi ký tự "Hòa bình" là nối của 2 từ "Hòa " và "bình". Lệnh **join** có thể lồng nhau nhiều mức.

Hàm  sẽ trả lại 1 ký tự cụ thể trong 1 chuỗi, từ. Ví dụ lệnh  sẽ trả lại giá trị là chữ "t".

Hàm  trả lại độ dài của 1 chuỗi, từ cho trước. Ví dụ  trả lại số 7.

Sau đây là 1 số thao tác, lệnh đơn giản khác liên quan đến chuỗi ký tự.

Gán chuỗi ký tự **Str** là rỗng



Gán giá trị của chuỗi **String1** cho chuỗi **Str**.



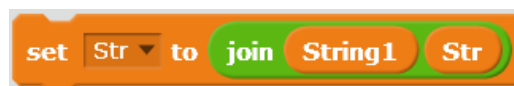
Lấy ra 1 ký tự ngẫu nhiên của chuỗi **Str**.



Bổ sung chuỗi **String1** vào cuối của chuỗi **Str**.



Bổ sung chuỗi **String1** vào đầu của chuỗi **Str**.



3. Spelling English Word. Bài toán học phát âm tiếng Anh

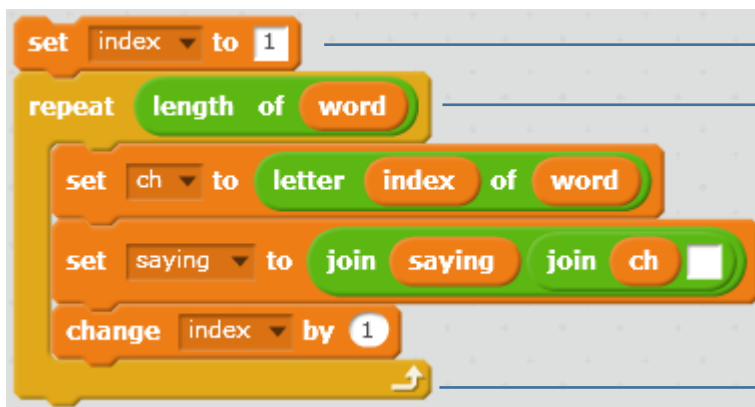
Em bắt đầu bài học bằng 1 ví dụ đơn giản sau: Bài học phát âm tiếng Anh (spelling word).

Thầy giáo yêu cầu học sinh nhập 1 từ tiếng Anh, sau đó thầy sẽ diễn giải cách phát âm từng chữ của từ tiếng Anh này.

Ví dụ: nếu học sinh nhập từ **peace** thì giáo viên sẽ đưa ra cách phát âm "**p e a c e**".

Em tạo biến nhớ **word** để lưu trữ từ do người dùng nhập, sau đó phần mềm sẽ tách từng chữ của từ này và đưa lên màn hình. Sử dụng biến chạy **index** để đưa từng chữ của từ này ra màn hình. Biến nhớ **saying** để lưu kết quả cần đưa ra màn hình.

Đoạn chương trình chính như sau.

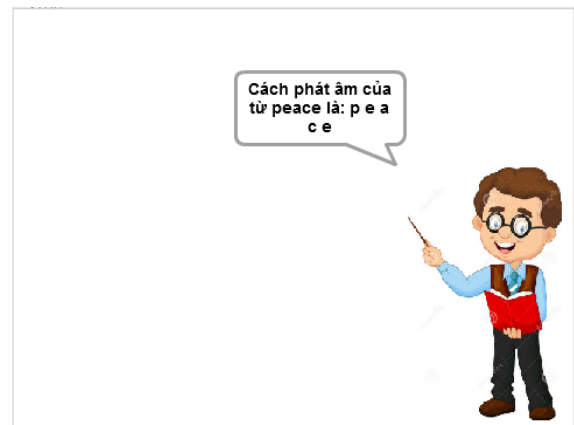
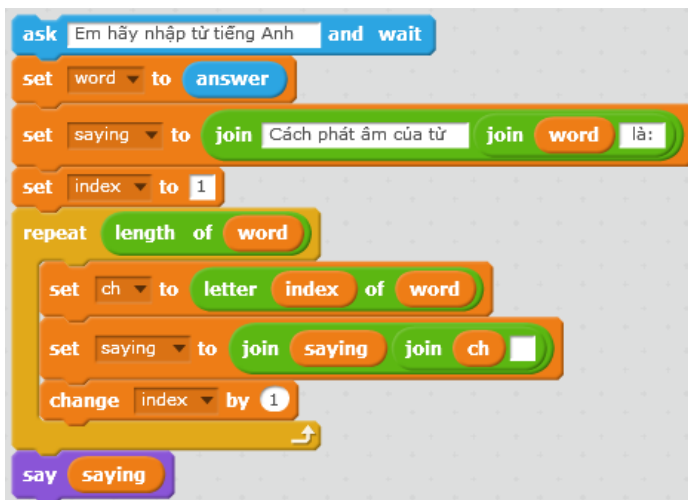


Gán giá trị ban đầu cho biến

Vòng lặp chính.

Với mỗi lần lặp, lấy ra **ch** = 1 ký tự tương ứng của xâu **word**, rồi đưa vào cuối của biến **saying**, và bổ xung thêm 1

Chương trình hoàn chỉnh của bài học này, kết quả hiện trong hình bên phải.

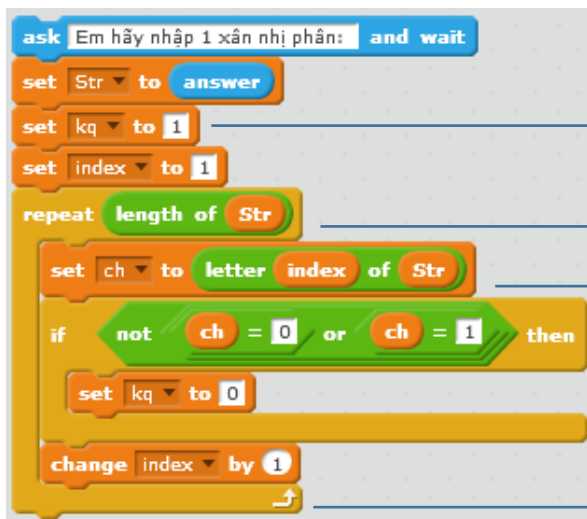


4. Kiểm tra tính chất của từ, xâu ký tự

4.1. Kiểm tra 1 xâu ký tự có phải là nhị phân hay không.

Xâu nhị phân là xâu chỉ bao gồm các ký tự 0 hoặc 1. Để kiểm tra 1 xâu ký tự **Str** có phải là nhị phân hay không, em hãy thực hiện theo cách sau. Biến nhớ **kq** dùng để lưu kết quả kiểm tra, xâu là nhị phân nếu $kq = 1$, ngược lại nếu $kq = 0$ thì xâu không là nhị phân. Để kiểm tra

ký tự **ch** là 0 hoặc 1 hay không chúng ta dùng biểu thức **ch = 0 or ch = 1**. Bắt đầu chương trình, em gán $kq = 0$, trong quá trình kiểm tra trong vòng lặp nếu gặp 1 ký tự không là 0 hoặc 1 thì gán ngay $kq = 1$.



Ban đầu gán $kq = 1$

Vòng lặp với độ dài của chuỗi **Str**

Gán **ch** cho 1 ký tự theo vòng

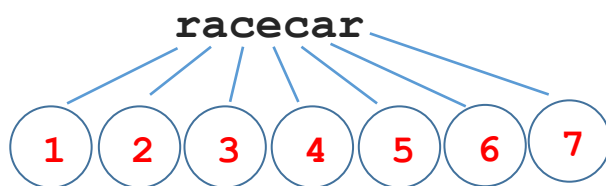
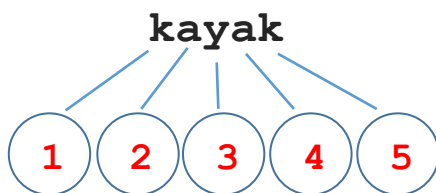
Kiểm tra nếu ch không là nhị phân thì gán $kq = 0$

Em hãy viết hoàn thiện chương trình này.

4.2. Kiểm tra 1 chuỗi ký tự có phải là đối xứng (palindrome) hay không.

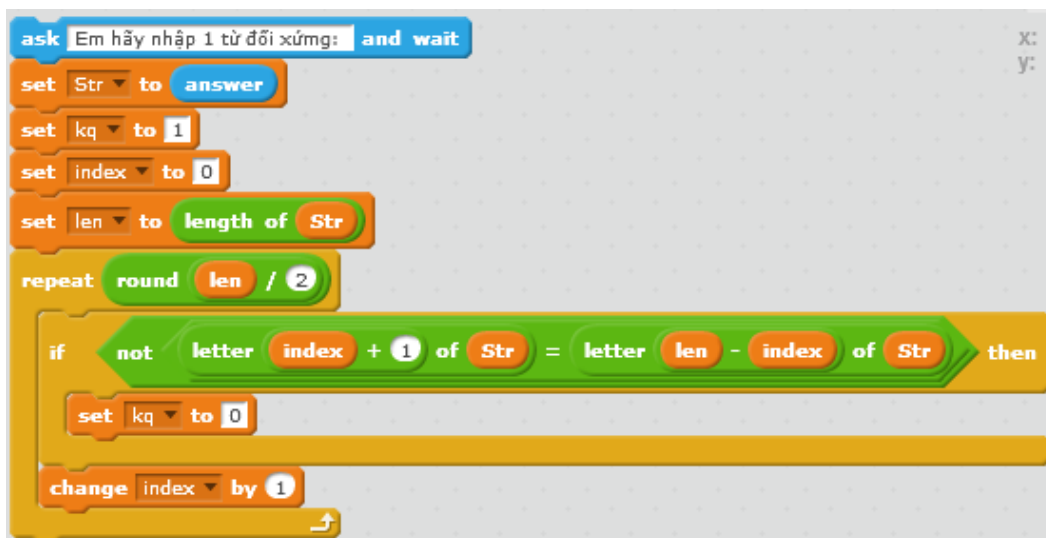
Một chuỗi được gọi là đối xứng (palindrome) nếu các ký tự, chữ tạo thành chuỗi này đối xứng qua trục thẳng đứng. Hay nói cách khác các từ nếu đọc xuôi hay ngược đều như nhau được gọi là palindrome. Ví dụ:

redivider, noon, civic, radar, level, rotor, kayak, reviver, racecar, madam, refer.



Phân tích: Nếu độ dài của chuỗi là **len** thì chuỗi đã cho sẽ là đối xứng nếu các cặp chữ với chỉ số $(1, len), (2, len-1), \dots$ phải có giá trị bằng nhau. Do vậy chỉ cần kiểm tra theo 1 vòng lặp chỉ số từ 1 đến $len/2$ là đủ. Chuỗi gốc ký hiệu là **Str**.

Chương trình



Em hãy viết hoàn thiện chương trình này.

5. Hàm lấy chuỗi con của 1 chuỗi hoặc từ

Bài toán: cho trước 1 chuỗi ký tự **Str** với độ dài **len**. Cần viết 1 chương trình để lấy ra 1 phần của chuỗi này (chuỗi con), tính từ vị trí **istart** đến vị trí **iend** của chuỗi này.

Ví dụ với chuỗi "happy new year", **istart** = 3, **iend** = 5, độ dài chuỗi con = **iend - istart + 1** = 3. Chuỗi con được trả lại được lưu trong biến **string** là "ppy"



Thuật toán của bài toán này khá đơn giản. Thiết lập 1 vòng lặp với số bước lặp **iend - istart + 1**, biến index bắt đầu từ vị trí **istart**, mỗi bước lấy 1 chữ từ chuỗi gốc và đưa vào cuối của chuỗi kết quả **string**.

Đoạn chương trình tính chuỗi con như sau.

Thiết lập chuỗi **string** = trống

Thiết lập giá trị ban đầu của biến

Vòng lặp chính số vòng = **iend - istart + 1**

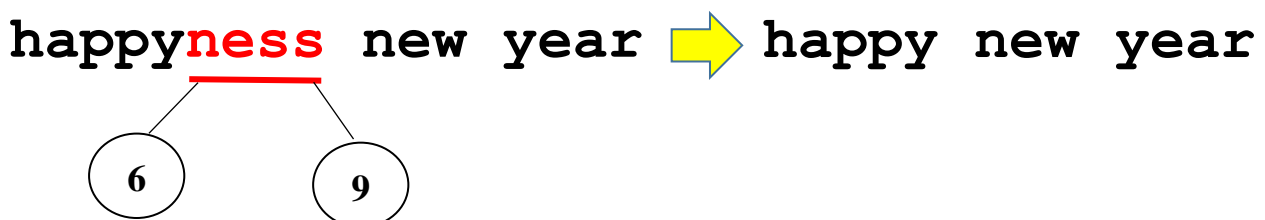
Bổ sung 1 ký tự vào cuối của

Em hãy hoàn thiện chương trình này bổ sung phần yêu cầu nhập chuỗi ký tự gốc **Str**, nhập 2 chỉ số bắt đầu và kết thúc chuỗi con **istart** và **iend**.

6. Hàm xóa 1 ký tự (1 phần) của chuỗi

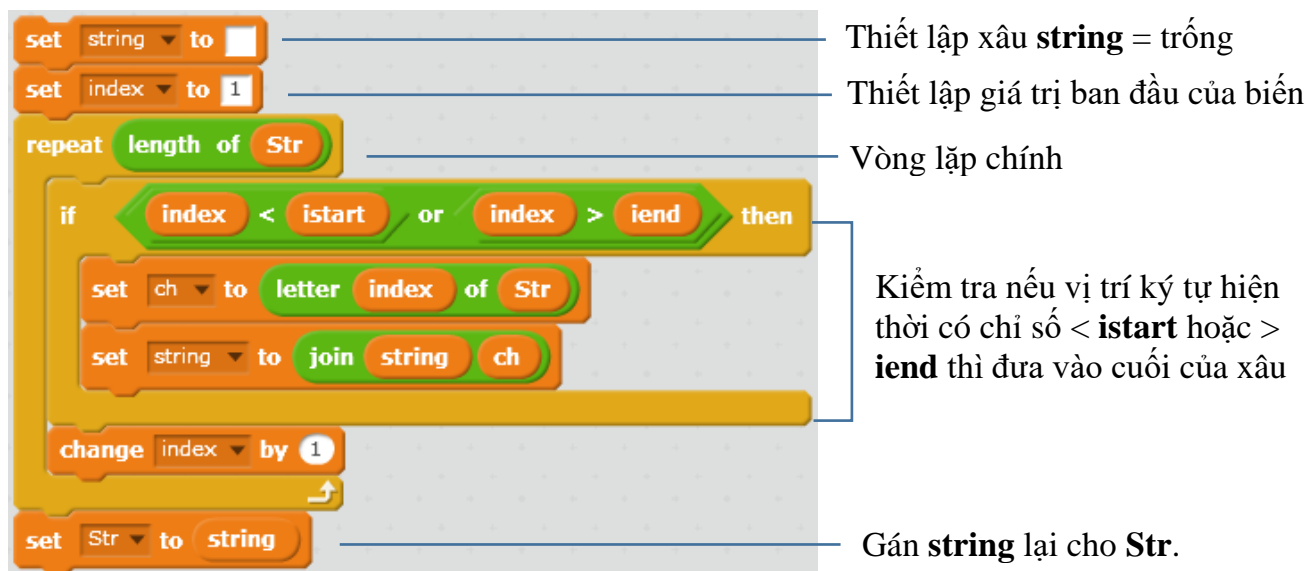
Bài toán: cho trước 1 chuỗi ký tự **Str** với độ dài **len**. Cần viết chương trình xóa đi chuỗi con của chuỗi này, tính từ vị trí **istart** đến vị trí **iend**.

Ví dụ với chuỗi "happyness new year", **istart** = 6, **iend** = 9, chuỗi mới sau khi đã xóa chuỗi con có giá trị là "happy new year"



Thuật toán của bài toán này như sau. Tạo biến nhớ phụ **string** dùng để lưu tạm kết quả. Vòng lặp chính chạy suốt chiều dài của chuỗi gốc **Str**. Với mỗi ký tự của **Str**, chương trình kiểm tra nếu vị trí này < **istart** hoặc > **iend** thì đưa ký tự nào vào cuối của chuỗi **string** (ngược lại không làm gì cả).

Đoạn chương trình xóa chuỗi con như sau.

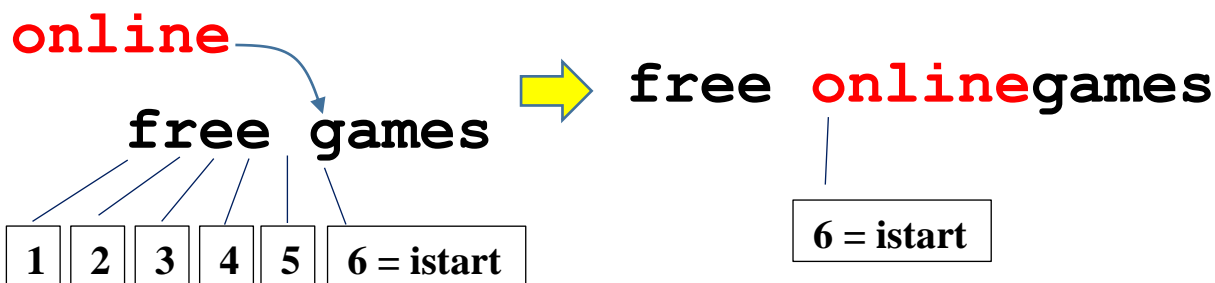


Em hãy hoàn thiện chương trình này bổ sung phần yêu cầu nhập chuỗi ký tự gốc **Str**, nhập 2 chỉ số bắt đầu và kết thúc chuỗi cần xóa **istart** và **iend**.

7. Hàm chèn chuỗi (ký tự) vào chuỗi khác tại vị trí cho trước

Bài toán: cho 2 chuỗi ký tự: chuỗi gốc **Str** và chuỗi thứ 2 **Str1**. Cần chèn chuỗi **Str1** vào chuỗi gốc **Str** tại vị trí **istart**, kết quả ghi lại trong chính chuỗi gốc **Str**.

Để chuẩn bị chương trình chúng ta cùng tìm hiểu qua 1 ví dụ cụ thể. Chuỗi gốc **Str** = "free games", chuỗi cần chèn **Str1** = "online", vị trí chèn **istart** = 6.



Đầu tiên chúng ta cho biến nhớ **index** chạy và gán từng chữ của chuỗi **Str** vào 1 biến trung gian **string**. Khi đến vị trí **istart**, quá trình này tạm dừng để chuyển sang việc bổ sung chuỗi **Str1** vào **string**. Công việc này được thực hiện bằng 1 biến nhớ khác là **index1**. Khi đã bổ sung xong **Str1** thì quá trình bổ sung **Str** lại tiếp tục bằng **index**. Chương trình này sẽ có 2 vòng lặp lồng nhau.

Thuật toán trên có thể viết lại như sau:

Gán các giá trị ban đầu $index = 1$, $index1 = 1$

Thực hiện lặp cho đến khi $index > \text{độ dài Str}$

Nếu $index = istart$ thì

Thực hiện vòng lặp có số bước bằng độ dài chuỗi **Str1**

Bổ sung **Str1** vào cuối **string**

Gán ký tự thứ $index$ của chuỗi **Str** vào **string**

Gán **Str** = **string**

Đoạn chương trình sau mô tả phần thuật toán chính của bài toán.

Vòng lặp ngoài, điều kiện dừng khi duyệt xong xâu Str

Kiểm tra điều kiện để thực hiện vòng lặp trong

Vòng lặp trong, bổ sung xâu Str1 vào cuối của string. Vòng lặp này chỉ thực hiện đúng 1 lần.

Các lệnh vòng lặp ngoài: bổ sung từng chữ của xâu Str vào cuối của string.

Gán trả lại string vào Str là xâu gốc ban đầu

Chương trình hoàn chỉnh sẽ yêu cầu học sinh nhập lần lượt các thông số:

- Xâu gốc Str
- Xâu cần chèn Str1
- Vị trí cần chèn

Chương trình thực hiện công việc chèn và thông báo kết quả ra màn hình.

Câu hỏi và bài tập

1. Các biểu thức nào dưới đây viết đúng?

$$(m^2 + n^2) \text{ và } (m^2 - n^2)$$

$$(a + b)^2 = a^2 + 2ab + b^2$$

$$abc < bce$$

2. Sau 2 lệnh sau thì biến nhớ Str sẽ lưu trữ chuỗi ký tự gì?



Mở rộng

Thiết kế 1 trò chơi **Ghép chữ tạo từ chính xác** sau.

Phần mềm sẽ đưa ra trên màn hình 2 từ, người chơi cần ghép 2 từ này lại với nhau để tạo thành 1 từ đúng.

Ví dụ nếu 2 từ cho ban đầu là **ped** và **wikiia** thì từ cần ghép đúng phải là **wikipedia**. Người chơi được yêu cầu nhập từ cần ghép từ bàn phím cho đến khi tạo được từ đúng thì dừng lại. Nếu sau 10 lần vẫn nhập sai thì thua và phần mềm sẽ đưa ra đáp án đúng.

Bài 15. Xử lý xâu ký tự 2

Mục đích

Học xong bài này bạn sẽ hiểu:

- Một số thuật toán đơn giản xử lý chữ và xâu ký tự.
- Thiết lập 1 vài trò chơi đơn giản với chữ và xâu ký tự.

Bắt đầu

Em hãy đọc để hiểu 1 số trò chơi liên quan đến chữ, xâu ký tự dưới đây. Em đã biết và đã từng chơi các trò chơi này chưa? Em hãy phát biểu suy nghĩ của mình xem có thể lập trình để mô phỏng chúng được hay không?

1. Trò chơi sắp xếp từ

Trên màn hình xuất hiện 1 dãy các từ. Nhiệm vụ của người chơi là cần sắp xếp lại các từ này theo 1 thứ tự nhất định có ý nghĩa nào đó, ví dụ, sắp xếp lại để trở thành 1 câu có nghĩa. Người chơi thao tác bằng cách kéo thả các chữ hoặc đánh số các từ trên màn hình.



2. Trò chơi đoán từ hangman

Trên màn hình hiện yêu cầu đoán 1 từ khi cho biết ý nghĩa của từ này. Người chơi đoán từ bằng cách nhập từng chữ cái cấu tạo nên từ này (bằng bàn phím hoặc chuột). Nếu nhập đúng thì chữ cái đó sẽ hiện trên màn hình tại đúng vị trí trong từ. Người chơi chỉ được phép nhập sai 1 số lần nhất định.



Nội dung bài học

1. Số nhị phân

Em có hiểu số nhị phân là gì không?

Trong các số sau, số nào là nhị phân?

345671 1011101 1001002 1010101

11011a1b 3220011 1111111 0000111

Những số mà chúng ta làm việc hàng ngày đều là các số thập phân, hay còn gọi là số được viết trong hệ thập phân. Các số thập phân sử dụng 10 chữ số 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 để thể hiện. Nhưng trong máy tính chỉ có thể hiểu 2 chữ số là 0 và 1. Do vậy các số lưu trữ trong máy tính chỉ bao gồm các chữ số 0, 1. Các số này được gọi là số **nhị phân**. Hệ đếm chỉ dùng chữ số 0 và 1 gọi là **hệ nhị phân**. Hệ thập phân tiếng Anh gọi là **decimal**, hệ nhị phân gọi là **binary**.

Bảng sau cho ta biết tương ứng giữa 1 số số nhị phân và thập phân.

decimal	binary	decimal	binary	decimal	binary	decimal	binary
1	1	5	101	9	1001	13	1101
2	10	6	110	10	1010	14	1110
3	11	7	111	11	1011	15	1111
4	100	8	1000	12	1100	16	10000

2. Chuyển số nhị phân sang thập phân

Bài toán: cho 1 số nhị phân $a_n a_{n-1} \dots a_1 a_0$ (binary), cần chuyển đổi số này sang hệ thập phân.

Chúng ta quan sát qui luật biểu diễn số dưới dạng nhị phân và tìm ra qui luật tổng quát cho việc chuyển đổi số viết theo hệ nhị phân sang hệ thập phân.

$$100 \rightarrow 100 = 1.2^2 + 0.2 + 0 = 4 \text{ (hệ thập phân)}$$

$$111 \rightarrow 111 = 1.2^2 + 1.2 + 1 = 7 \text{ (hệ thập phân)}$$

Tổng quát chúng ta thấy:

$$a_n a_{n-1} \dots a_1 a_0 \text{ (binary)} = a_n.2^{n-1} + a_{n-1}.2^{n-2} + \dots + a_1.2 + a_0 \text{ (decimal)}$$

Nhìn vào công thức trên chúng ta chưa hình dung được thuật toán cần thực hiện trên máy tính. Chúng ta sẽ viết lại quá trình tính toán theo từng bước nhỏ để tính được số thập phân.

a_n

$$2.a_n + a_{n-1}$$

$$2.(2.a_n + a_{n-1}) + a_{n-2}$$

$$2.(2.(2.a_n + a_{n-1}) + a_{n-2}) + a_{n-3}$$

.....

$$2.(a_n.2^{n-3} + a_{n-1}.2^{n-4} + \dots + a_0) + a_1$$

$$2.(a_n.2^{n-2} + a_{n-1}.2^{n-3} + \dots + a_1) + a_0.$$

Như vậy sau đúng $n+1$ bước thì quá trình tính trên kết thúc và kết quả thu được số thập phân cần tìm.

Số nhị phân ban đầu được đưa vào biến nhớ **Binary**. Số thập phân là kết quả được lưu trong biến nhớ **Decimal**. Các biến nhớ trung gian bao gồm **index**, **len** và **ch**.

Gán ban đầu: `Decimal = 0, index = 1`

`len = độ dài xâu Binary`

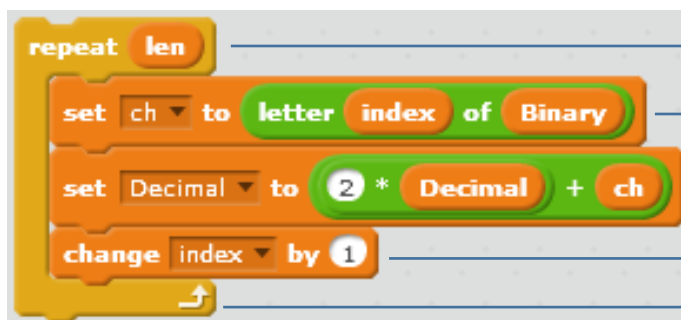
Thực hiện vòng lặp với độ dài len

`ch = chữ số tương ứng chỉ số index của Binary`

`Decimal = 2.Decimal + ch`

`Tăng index lên 1`

Đoạn chương trình mô tả thuật toán biến đổi hệ nhị phân sang thập phân `Binary → Decimal` như sau.



Bắt đầu vòng lặp độ dài

Lấy ra ký tự tương ứng của xâu

Thay đổi $Decimal = 2 * Decimal + ch$

Tăng index lên 1

Chương trình hoàn chỉnh của bài toán này như sau:



Số thập phân kết quả là: 255



3. Chuyển số thập phân sang nhị phân

Bài toán: cho trước số thập phân, hãy biểu diễn số này sang hệ nhị phân.

Bài toán này giải quyết vấn đề ngược lại so với bài toán trên.

Để tìm được cách làm bài này, chúng ta lại quan sát công thức:

$$a_n a_{n-1} \dots a_1 a_0 \text{ (binary)} = a_n \cdot 2^{n-1} + a_{n-1} \cdot 2^{n-2} + \dots + a_1 \cdot 2 + a_0 \text{ (decimal)}$$

Gọi số thập phân ban đầu là D, chúng ta sẽ phân tích quá trình tính toán để tính được lần lượt các số $a_0, a_1, \dots, a_{n-1}, a_n$ như sau.

Số thập phân ban đầu: D

$$a_0 = D \bmod 2 \text{ (tính số dư); } a_n \cdot 2^{n-2} + a_{n-1} \cdot 2^{n-3} + \dots + a_1 = D/2 \text{ (làm tròn số); (gán } D = D/2).$$

$$a_1 = D \bmod 2; D = D/2.$$

.....

$$a_{n-1} = D \bmod 2, D = D/2.$$

$$a_n = D \bmod 2, D/2 = 0, \text{ kết thúc.}$$

Như vậy qui trình trên sẽ dừng lại khi $D = 0$.

Số thập phân đầu vào lưu trong biến nhớ **Decimal**. Xâu nhị phân đầu ra lưu trong biến nhớ **Binary**. Biến **ch** dùng để tính các giá trị trung gian a_i . Thuật toán chuyển đổi ở trên được viết tường minh như sau:

Gán `Binary = rỗng`

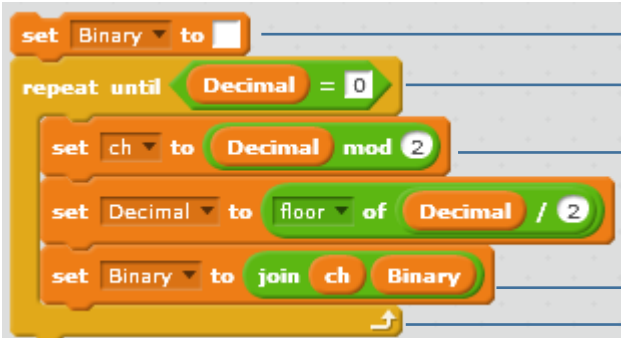
Thực hiện lặp cho đến khi `Decimal = 0`

 Đặt `ch = Decimal mod 2`;

`Decimal = Decimal / 2` (làm tròn xuống)

 Bổ sung `ch` vào bên trái của `Binary`

Đoạn chương trình mô tả thuật toán chuyển đổi số Nhị phân sang Thập phân trên được viết trong Scratch như sau:




The image shows a Scratch code block with the following steps:

- `set Binary to` (empty text field)
- `repeat until` loop with condition `Decimal = 0`
 - `set ch to` `Decimal mod 2`
 - `set Decimal to` `floor of` `Decimal / 2`
 - `set Binary to` `join` `ch` `Binary`

 Annotations on the right:

- Line 1: Gán `Binary = rỗng`.
- Line 2: Vòng lặp chính với điều kiện dừng `Decimal = 0`.
- Line 3: Tính `ch = số dư Decimal chia 2`.
- Line 4: Giảm `Decimal` khi chia cho 2.
- Line 5: Bổ sung `ch` vào **bên trái** của `Binary`.

Chương trình hoàn chỉnh của bài toán này như dưới đây.



The complete Scratch program includes:

- `ask` block: "Em hãy nhập 1 số thập phân bất kỳ: and wait"
- `set` block: `Decimal` to `answer`
- `set` block: `Saying` to `join` "Số thập phân" `join` `Decimal` `chuyển sang nhị phân sẽ là:`
- `set` block: `Binary` to (empty text field)
- `repeat until` loop (same as the snippet above)
- `say` block: `join` `Saying` `Binary`

Số thập phân 256
chuyển sang nhị
phân sẽ là:
100000000



4. Kiểm tra 1 ký tự / từ có nằm trong 1 từ khác hay không

Bài toán: Cho trước 2 xâu **Str1** và **Str**. Cần kiểm tra xem xâu **Str1** có nằm trong xâu **Str** không, nếu có thì cần chỉ ra vị trí ký tự đầu tiên của **Str1** trong **Str**. Nếu không nằm trong thì trả lại giá trị 0.

Ví dụ : 2 xâu là **bab** và **ababab**.

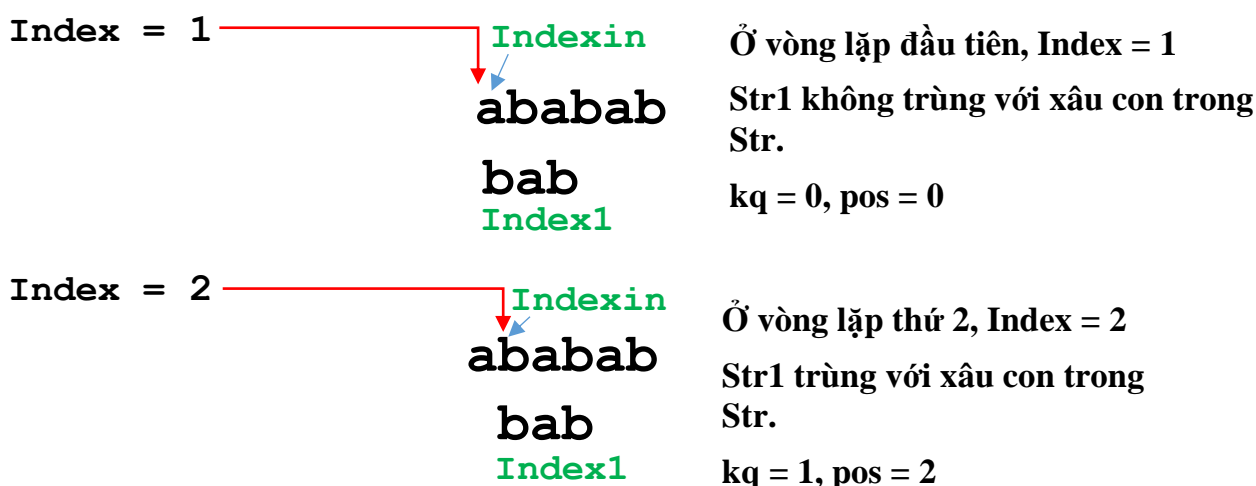
bab **ababab**

 2

Xâu **bab** mặc dù được tìm thấy trong xâu **ababab** 2 lần nhưng hàm sẽ trả lại vị trí đầu tiên, tức là giá trị 2.

Giả sử 2 xâu **Str1** và **Str** có độ dài là **len1** và **len**. Kết quả của thuật toán được trả lại trong biến **pos**. Nếu không tìm thấy (**Str1** không nằm trong **Str**) thì **pos = 0**, ngược lại **pos > 0** là vị trí đầu tiên **Str1** nằm trong **Str**.

Ý tưởng của thuật toán như sau: Bắt đầu từ vị trí đầu tiên của xâu **Str**, tiến hành kiểm tra lần lượt các vị trí bắt đầu từ 1 bằng biến **index**. Với mỗi vị trí như vậy cần tiến hành kiểm tra **len1** phần tử tiếp theo xem có trùng với **Str1** không. Nếu trùng thì lập tức dừng vòng lặp, gán giá trị **pos = index**. Nếu không trùng thì tiếp tục tăng **index** và kiểm tra tiếp. Trong vòng lặp kiểm tra bên trong sử dụng thêm các biến nhớ: **indexin**, **index1**, **ch1**, **kq**. Biến **kq** có ý nghĩa **kq = 1** nếu đã tìm thấy **Str1** trong **Str**. Trong vòng lặp trong, sử dụng biến nhớ **indexin** để chạy trên xâu **Str** đồng thời với biến **index1** chạy trên **Str1**.



Mô tả thuật toán trên bằng lời tường minh như sau:

Gán các giá trị ban đầu: `pos = 0, index = 1`

Thực hiện lặp cho đến khi `pos > 0` hoặc `index > len - len1 + 1`

Gán `index1 = 1, kq = 1, indexin = index`

Lặp cho đến khi `index1 > len1` hoặc `kq = 0`

Đặt `ch` = ký tự thứ `index` của `Str`

Đặt `ch1` = ký tự thứ `index1` của `Str1`

Nếu `ch` khác `ch1` thì gán `kq = 0`

Tăng `index1` lên 1

Tăng `indexin` lên 1

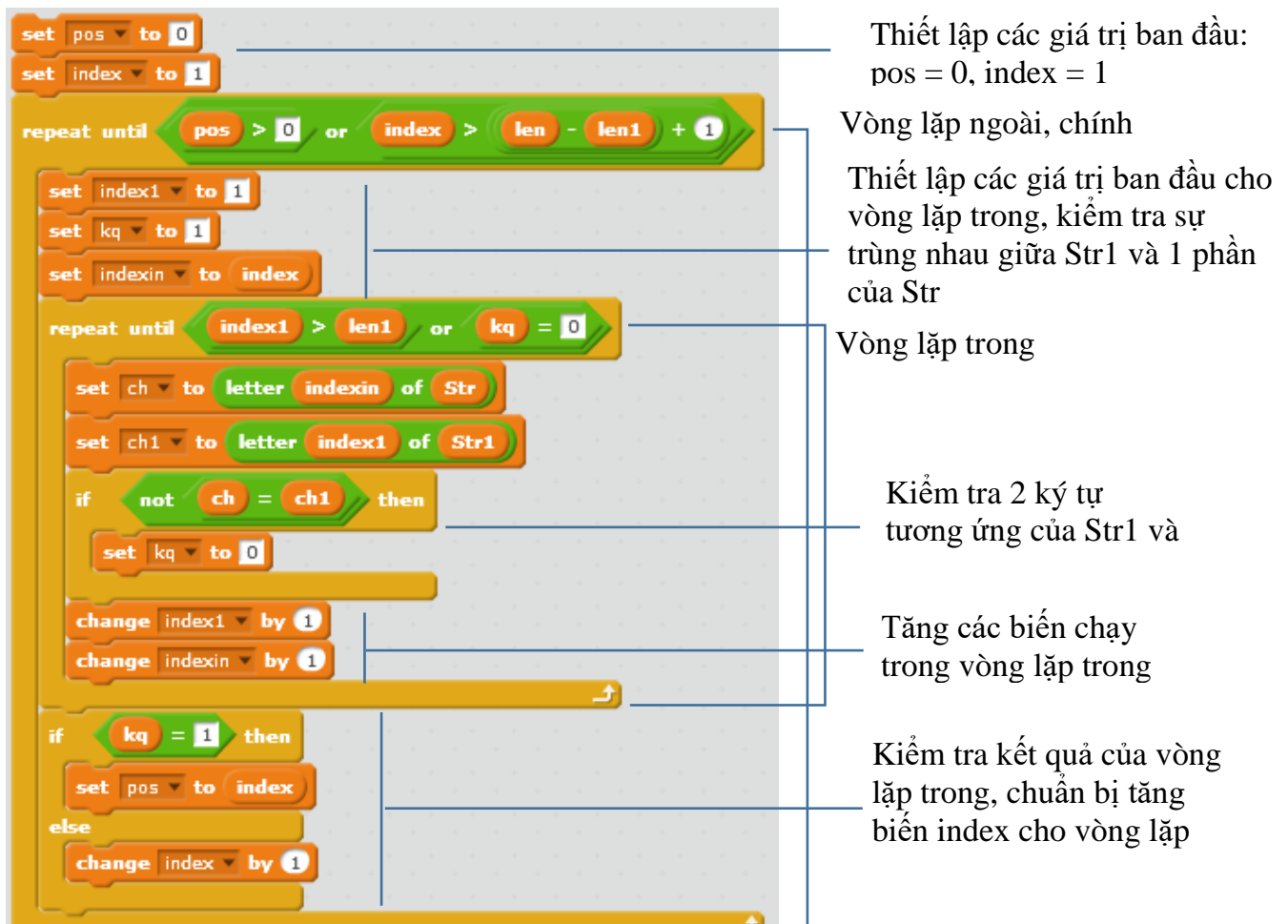
Nếu `kq = 1` thì

Gán `pos = index`

nếu không thì

Tăng `index` lên 1

Đoạn chương trình Scratch mô tả thuật toán trên như sau:



Em hãy hoàn thiện chương trình đầy đủ của bài toán này.

Chương trình sẽ yêu cầu học sinh nhập 2 dữ liệu:

- Xâu dữ liệu gốc Str.
- Xâu dữ liệu cần kiểm tra Str1

Chương trình sẽ thông báo kết quả có tìm thấy hay không. Nếu tìm thấy thì chỉ ra vị trí mà **Str1** nằm trong **Str**.

Câu hỏi và bài tập

1. Viết chương trình nhập từ bàn phím 1 xâu ký tự bất kỳ Str. Sau đó tiến hành đổi chỗ 2 phần tử bất kỳ trong xâu trên và hiển thị kết quả trên màn hình.

2. Viết chương trình thực hiện công việc sau:

Nhập từ bàn phím 1 xâu ký tự bất kỳ Str, sau đó tiến hành hoán vị ngẫu nhiên các ký tự của Str, kết quả đưa vào xâu Str1. Kết quả thể hiện ra màn hình ca 2 xâu Str và Str1.

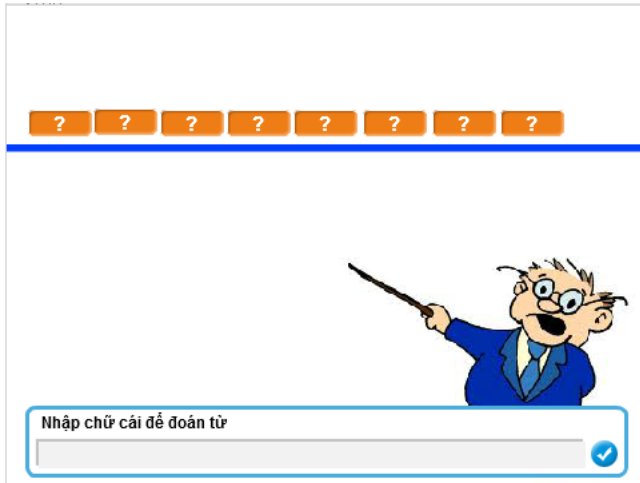
3. Viết chương trình thực hiện công việc sau:

- Nhập 1 xâu ký tự bất kỳ **Str** từ bàn phím.
- Thực hiện việc thiết lập xâu **Str1** bằng cách thay đổi ngược lại thứ tự các ký tự của xâu **Str**. Ví dụ nếu ban đầu **Str** = "abcdef" thì kết quả xâu **Str1** = "fedcba".

- Thể hiện kết quả xâu **Str1** trên màn hình.

Mở rộng

Thiết kế trò chơi tìm từ (Hangman) dạng đơn giản sau.



Từ cần tìm được lưu trong biến nhớ Word (được gán cứng trong chương trình). Giao diện chương trình là trò chơi dự đoán từ này. Từ Word có độ dài ≤ 8 .

Trên màn hình sẽ thể hiện các ô tương ứng với độ dài của từ cần tìm. Các chữ chưa được đoán sẽ hiện dấu ?, các chữ đã đoán rồi sẽ hiện chính xác trên màn hình.

Chương trình sẽ liên tục đưa ra câu hỏi: "Nhập chữ cái để đoán từ". Nhiệm vụ của người chơi là nhập 1 chữ cái. Nếu chữ cái này có trong từ thì nó sẽ hiện ra trên màn

hình (có thể 1 hoặc nhiều). Nếu không có thì chương trình thông báo "không có chữ cái này" và yêu cầu nhập tiếp tục, cho đến khi nào lật được hết các chữ cái của từ đã cho.

Bài 16. Làm việc với List 1

Mục đích

Học xong bài này bạn sẽ:

- Hiểu ý nghĩa của List (mảng, dãy giá trị), biết cách thiết lập dãy trong Scratch.
- Ứng dụng List giải 1 số bài tập đơn giản.
- Thuật toán sắp xếp, đổi chỗ, tìm min, max trong dãy.

Bắt đầu

Em đã từng xem, từng biết các thông tin được liệt kê như một danh sách, ví dụ như danh sách học sinh lớp học, danh sách các món ăn, danh sách các tỉnh, thành phố,

Số TT	Họ và tên thí sinh	
1	PHẠM VIỆT	HOÀNG
2	NGUYỄN ĐỖ	VĂN
3	NGUYỄN ĐỖ	VŨ
4	VŨ THỊ THANH	DUYÊN
5	VŨ ANH	KIỆT
6	NGUYỄN TRƯƠNG	TUẤN
7	NGUYỄN ĐÌNH	BẢO
8	LÊ NHẬT	LINH
9	LÊ VIỆT	QUANG
10	HUYỀN NGUYỄN ANH	THU
11	TRẦN ĐÌNH	HIỂN
12	PHẠM NHẬT	TUẤN
13	LÊ HOÀNG	VŨ
14	NGUYỄN THỊ THANH	HƯƠNG
15	VƯƠNG THỊ ANH	MINH

THỰC ĐƠN 1

Mục chiên
Tôm hấp
Gà rô ty
Cá thu sốt
Dạ dày bóp
Hoa lơ xào nạc
Rau luộc
Canh chua cá
Cơm + Tráng miệng

Các tỉnh			Thành phố
An Giang	Hà Nam	Quảng Nam	Cần Thơ
Bà Rịa - Vũng Tàu	Hà Tĩnh	Quảng Ngãi	Đà Nẵng
Bắc Giang	Hải Dương	Quảng Ninh	Hải Phòng
Bắc Kạn	Hậu Giang	Quảng Trị	Hà Nội
Bạc Liêu	Hòa Bình	Sóc Trăng	TP HCM
Bắc Ninh	Hưng Yên	Sơn La	
Bến Tre	Khánh Hòa	Tây Ninh	
Bình Định	Kiên Giang	Thái Bình	
Bình Dương	Kon Tum	Thái Nguyên	
Bình Phước	Lai Châu	Thanh Hóa	
Bình Thuận	Lâm Đồng	Thừa Thiên Huế	
Cà Mau	Lạng Sơn	Tiền Giang	
Cao Bằng	Lào Cai	Trà Vinh	
Đắk Lắk	Long An	Tuyên Quang	
Đắk Nông	Nam Định	Vĩnh Long	
Điện Biên	Nghệ An	Vĩnh Phúc	
Đồng Nai	Ninh Bình	Yên Bái	
Đồng Tháp	Ninh Thuận	Phú Yên	
Gia Lai	Phú Thọ		
Hà Giang	Quảng Bình		

Em hãy liệt kê thêm các danh sách thông tin mà em đã từng biết đến ở nhà, ở trường và ngoài xã hội.

Chúng ta đã được làm quen với khái niệm biến nhớ, là công cụ để lưu trữ các thông tin giúp người lập trình có thể tạo ra được các chương trình hiệu quả hơn. Tuy nhiên mỗi biến nhớ chỉ lưu trữ được 1 giá trị tại 1 thời điểm. Nếu như, ví dụ, chúng ta cần lưu trữ đồng thời tên của tất cả học sinh trong lớp học, thì phải làm như thế nào. Rõ ràng từng biến nhớ cụ thể không thể giải quyết được yêu cầu trên.

Bài học này sẽ giúp em hiểu được một công cụ mới để thực hiện yêu cầu trên đây.

Nội dung bài học

1. Biến nhớ kiểu danh sách (List)

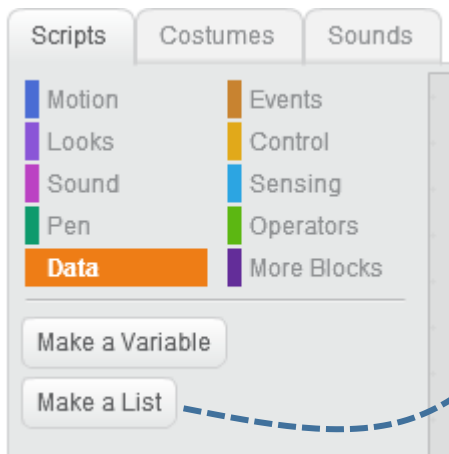
Biến nhớ kiểu danh sách dùng để lưu trữ một dãy giá trị, có thể là số hoặc chữ. Ví dụ dãy tên học sinh sau được đánh số từ 1 đến 6, có thể được lưu trong 1 biến nhớ kiểu danh sách (list).

Hà	Bình	Nguyệt	Vũ Hùng	Vân	Bình
1	2	3	4	5	6

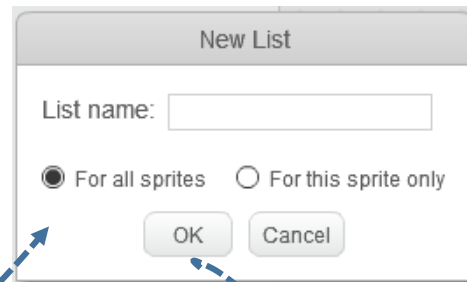
Để mô tả 1 dãy (số, chữ) tổng quát, người ta hay dùng ký hiệu: a_1, a_2, \dots, a_n . Số hạng tổng quát sẽ ký hiệu là a_i .

Trong Scratch, biến nhớ kiểu danh sách không cần khai báo trước kiểu dữ liệu và số lượng phần tử. Thao tác tạo ra các biến nhớ kiểu danh sách rất đơn giản, trong nhóm lệnh Dữ liệu (Data).

Việc khởi tạo biến nhớ danh sách rất đơn giản, bởi nút lệnh **Make a List** từ nhóm lệnh **Data**.



Nút **Make a List** dùng để khởi tạo 1 biến nhớ dạng





Giao diện tạo biến nhớ dạng List. Có 2 loại: biến nhớ chung và biến nhớ riêng cho mỗi nhân vật.




Xuất hiện các lệnh làm việc với biến nhớ List.

Cũng giống như biến nhớ bình thường, biến nhớ dạng danh sách có thể là biến nhớ chung hoặc riêng. Nếu là biến nhớ chung thì tất cả các nhân vật đều có thể sử dụng, cập nhật và thay đổi dữ liệu. Nếu là biến nhớ riêng thì chỉ có nhân vật là chủ mới có các quyền truy cập, sử dụng và thay đổi thông tin.

Việc nhập dữ liệu vào biến nhớ List có thể thực hiện bằng lệnh  hoặc có thể nhập trực tiếp trên sân khấu của Scratch.

Lệnh  có tính năng bổ sung thêm 1 thông tin vào cuối danh sách của biến nhớ dạng List.

Ví dụ lệnh  sẽ bổ sung thêm 1 tên học sinh là "Việt Anh" vào cuối của biến nhớ HS (biến nhớ dạng List).

Chú ý: khác với biến nhớ thông thường, biến dạng List nếu khai báo là dùng riêng cho nhân vật sẽ không hiện trong các thuộc tính của nhân vật, do đó các nhân vật khác sẽ không thể truy cập hay sử dụng các biến nhớ riêng này.

Em hãy giải thích ý nghĩa của các lệnh sau và ghi sang cột bên phải.

add answer to HS	
add name to HS	
add m to dayso1	

Chúng ta sẽ cùng nhau tìm hiểu sâu sắc hơn các ứng dụng của biến nhớ List trong các hoạt động tiếp theo.

2. Nhập danh sách học sinh lớp học

Em hãy thực hiện chương trình đơn giản sau.



Chương trình yêu cầu người dùng nhập họ tên học sinh trong lớp học vào một danh sách, trong thông báo ghi rõ là đang nhập học sinh thứ mấy. Để kết thúc chỉ cần nhập 1 dữ liệu rỗng, tức là bấm Enter ngay.

Biến nhớ danh sách lưu trữ tên học sinh sẽ được đặt tên là HS. Chúng ta sử dụng thêm 1 biến nhớ nữa **Stt** dùng để lưu số thứ tự của học sinh hiện thời đang nhập. Dùng ngay biến **Stt** để kiểm tra khi nào thì kết thúc quá trình nhập liệu từ bàn phím.

Chương trình hoàn chỉnh của bài toán này như sau:



Điều kiện kiểm tra vòng lặp là $Stt = 0$

Kiểm tra nếu dữ liệu nhập vào khác rỗng thì bổ sung vào danh sách HS, nếu người dùng không nhập, chỉ nhấn Enter thì đặt $Stt = 0$ để kết thúc quá

3. Các thao tác trực tiếp trên danh sách

Có nhiều cách để thao tác với dữ liệu của biến nhớ danh sách: thông qua các lệnh, nhập trực tiếp trên màn hình Scratch hoặc nhập thông qua tệp (Text File)..

Các thao tác trực tiếp với dữ liệu trên biến nhớ danh sách: bổ sung, chèn, xóa,



Bổ dung dữ liệu <thing> vào cuối của biến nhớ. Lệnh này sẽ làm cho dãy dữ liệu của biến nhớ tăng thêm 1 phần tử nằm cuối danh sách.

Lệnh này sẽ xóa, hủy khỏi danh sách 1 phần tử được xác định trước. Nếu chỉ số của phần tử ghi trên lệnh nằm ngoài phạm vi của dãy (ví dụ = 0 hoặc > độ dài dãy) thì lệnh không có tác dụng. Một lựa chọn khác của lệnh nếu thiết lập tham số **all** thì lệnh có dạng sau



sẽ xóa toàn bộ dữ liệu của biến nhớ này (hay đưa biến nhớ về trạng thái như lúc mới khởi tạo).

Lệnh này sẽ chèn thêm 1 phần tử có giá trị <thing> vào trước phần tử thứ <1> của dãy. Nếu thay thế chỉ số cụ thể bằng <random>



thì lệnh sẽ chèn vào 1 vị trí bất kỳ của dãy. Sau lệnh này dãy sẽ tăng lên 1 phần tử.

Lệnh này sẽ thay thế phần tử thứ <1> của dãy bằng giá trị <thing>. Nếu thay thế chỉ số cụ thể bằng <random>



thì lệnh sẽ thay thế vào vào 1 vị trí bất kỳ của dãy. Chú ý: lệnh này không làm tăng số phần tử của dãy.

Các lệnh sau sẽ có chức năng truy cập, khai thác dữ liệu danh sách:



(hàm) trả lại giá trị cụ thể của một phần tử của dãy. Nếu tham số thay bằng <random>



, lệnh sẽ trả lại 1 phần tử bất kỳ trong dãy.

(hàm) trả lại độ dài của dãy.



(hàm logic) kiểm tra xem dãy có chứa phần tử được ghi tại vị trí <thing> hay không.

Lệnh này có tính năng hiển thị tất cả các phần tử của dãy.

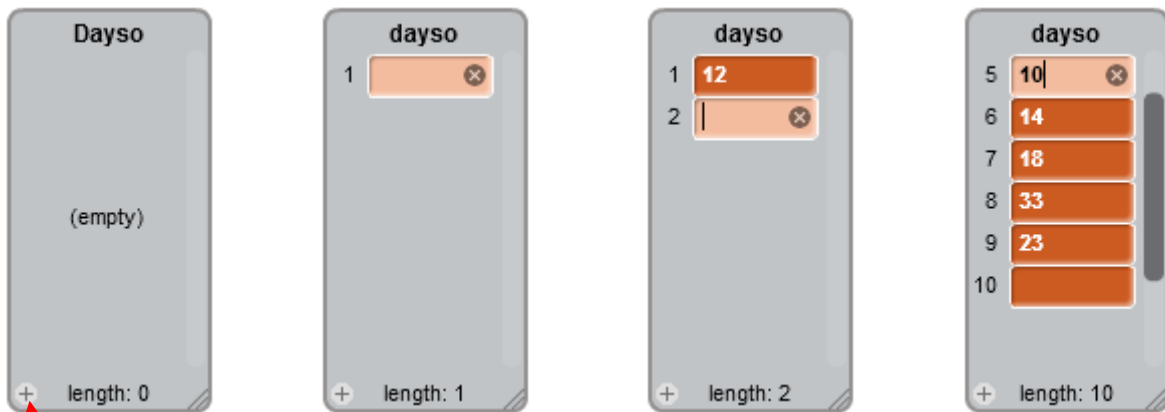


Nhập dữ liệu trực tiếp trên màn hình Scratch

Có thể nhập nhanh và trực tiếp dữ liệu kiểu danh sách trên màn hình của Scratch. Muốn vậy chúng ta đặt chế độ hiển thị biến nhớ này trên màn hình.



Hình ảnh của biến nhớ danh sách sẽ hiện ra như hình dưới đây. Bây giờ chúng ta có thể thao tác trực tiếp trên biến nhớ này để nhập dữ liệu.

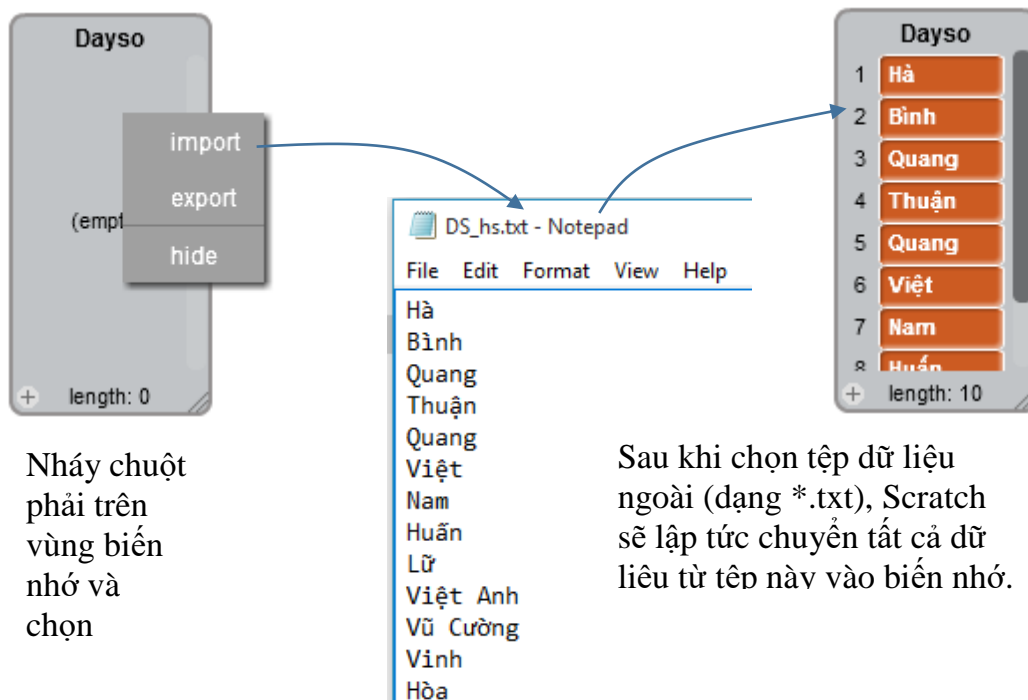


Nháy nút này để bắt đầu tiến hành nhập liệu.

Bước tiếp theo chúng ta nhập trực tiếp dữ liệu theo từng phần tử của danh sách. Có thể dùng các phím lên, xuống để chuyển lên, xuống trong danh sách. Tiến hành nhập cho đến khi xong. Muốn xóa 1 phần tử nhấn dấu x bên

Chuyển nhập dữ liệu từ Plain Text File

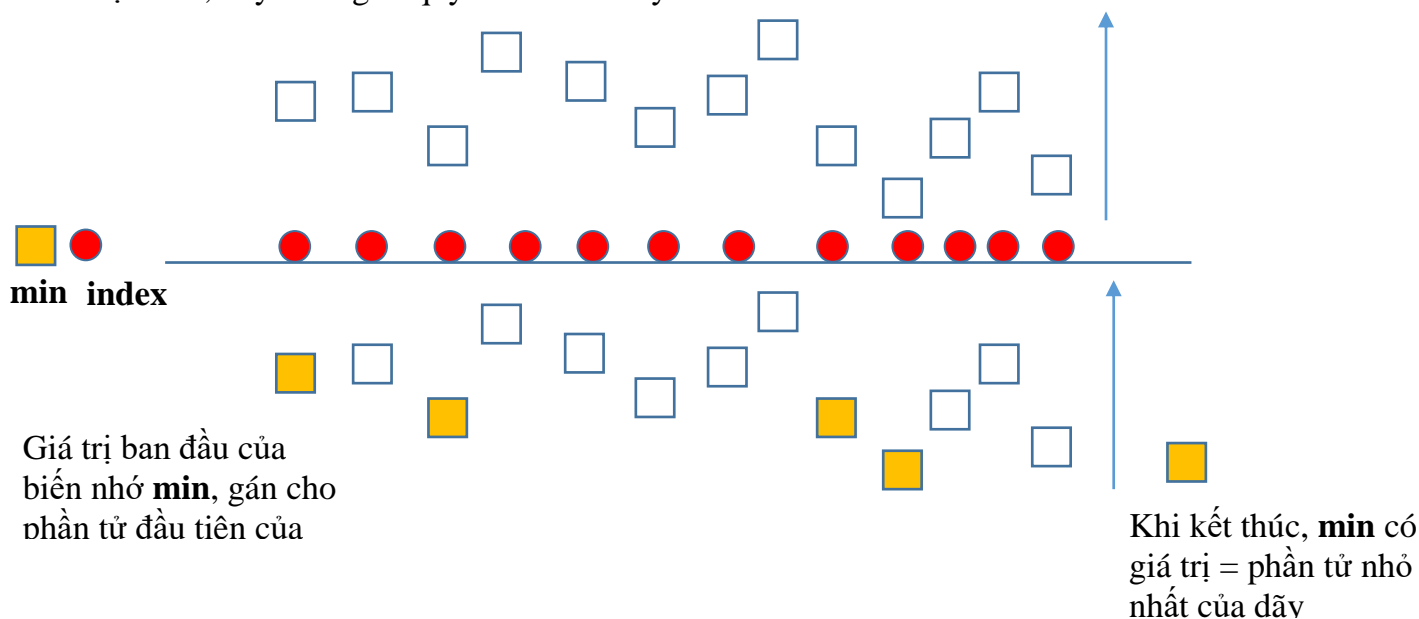
Một cách nhập dữ liệu nhanh và hiệu quả nữa là nhập trước dữ liệu vào 1 tệp văn bản dạng plane text (tệp *.txt), mỗi đơn vị dữ liệu ghi trên 1 dòng, sau đó chuyển nhập nhanh vào danh sách của Scratch. Quy trình nhập này được mô tả trong hình sau.



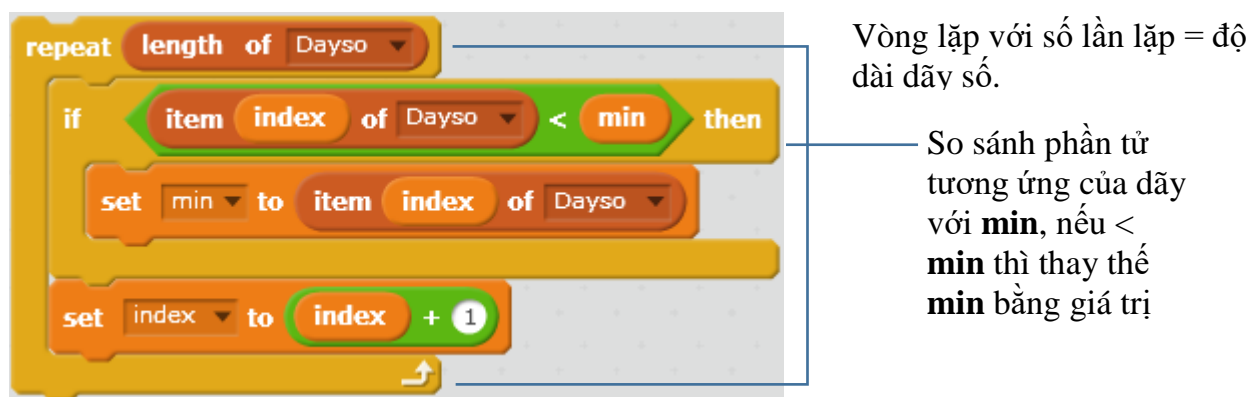
4. Bài toán tìm Min, Max

Bài toán: cho trước 1 dãy số chứa trong 1 biến nhớ danh sách. Yêu cầu cần tìm và hiển thị phần tử nhỏ nhất (Min) và lớn nhất (Max) của dãy này.

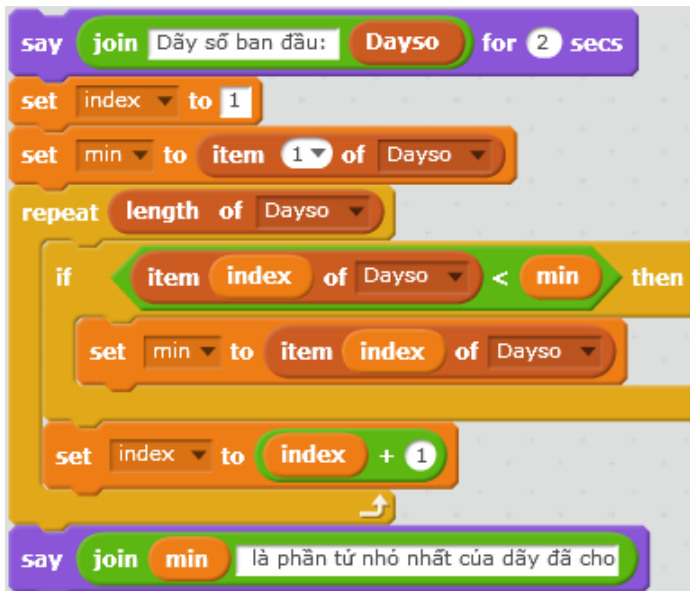
Thuật toán, hay cách giải quyết bài toán này như sau:



Tạo ra 2 biến nhớ: biến **index** để chạy dọc theo dãy số, biến **min** để lưu kết quả so sánh trung gian. khi di chuyển dọc theo dãy số, so sánh **min** và giá trị phần tử của dãy, nếu giá trị này < **min** thì lập tức thay thế **min** bằng giá trị này. Trong ví dụ mô phỏng trên, biến nhớ **min** được thay đổi 3 lần. Thuật toán mô tả bằng lệnh Scratch như sau:



Chương trình hoàn chỉnh như sau:



Dãy số ban đầu: 12
11 9 -8 -5 21 54 -15
89 45 32 32

-15 là phần tử nhỏ nhất của dãy đã cho



Em hãy viết chương trình tương tự tìm giá trị phần tử lớn nhất (Max) của 1 dãy số cho trước.

5. Bài toán tìm kiếm giá trị trong dãy

Tìm kiếm thông tin là 1 bài toán lớn có rất nhiều ứng dụng rộng rãi trên thực tế. Trong hoạt động này chúng ta cùng tìm hiểu một vài trường hợp đơn giản nhất của công việc này.

Bài toán tìm 1 phần tử

Cho trước 1 danh sách dữ liệu, cần tìm ra 1 phần tử thỏa mãn 1 điều kiện nào đó. Ví dụ thực tế của bài toán này rất nhiều. Ví dụ:

- Tìm trong lớp 1 bạn nam có chiều cao lớn hơn 1,7m.
- Tìm trong dãy 1 số là số nguyên tố.

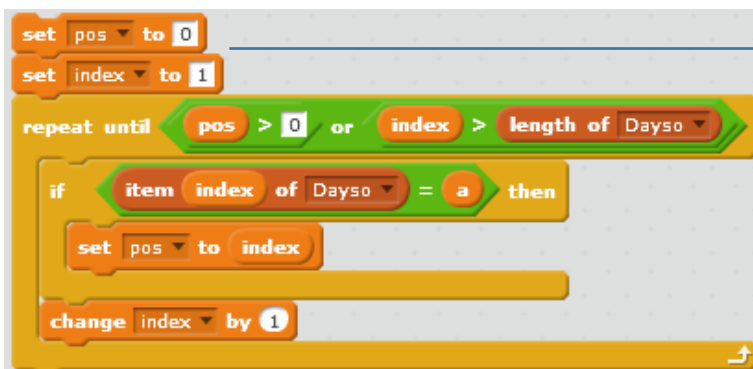
Chúng ta cùng tìm hiểu và giải quyết bài toán này.

Bài toán cụ thể có thể đơn giản hóa như sau: Giả sử dãy dữ liệu là dãy số, cần tìm 1 phần tử trong dãy trên có giá trị = **a**. Dãy số trên được lưu trong biến nhớ danh sách có tên **dayso**.

Đầu vào (Input): Danh sách **dayso**, giá trị **a**.

Đầu ra (Output): Thông báo "không tìm thấy" hoặc "có" và chỉ ra số thứ tự của phần tử được tìm thấy có giá trị **a**.

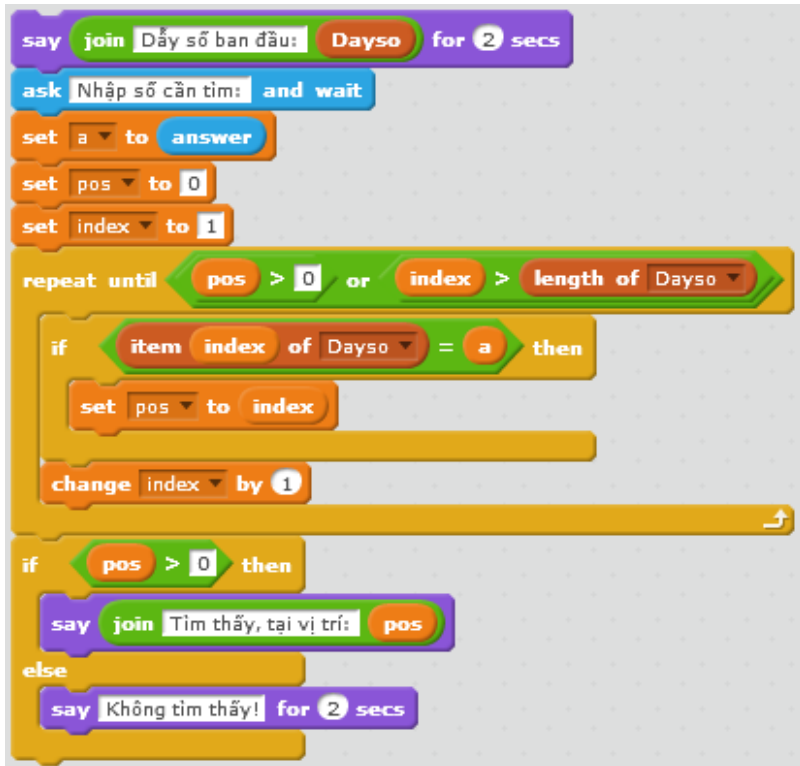
Chúng ta sử dụng biến nhớ **pos** để lưu lại chỉ số của phần tử tìm thấy trong dãy. Nếu **pos = 0** tức là không tìm thấy. Thuật toán đơn giản này mô tả như sau:



Gán các giá trị ban đầu: **pos = 0**, **index = 1**.

Vòng lặp chính có điều kiện dừng nếu **pos > 0**.

Chương trình hoàn chỉnh như sau:



The Scratch code block is as follows:

```
say join Dãy số ban đầu: Dayso for 2 secs
ask Nhập số cần tìm: and wait
set a to answer
set pos to 0
set index to 1
repeat until pos > 0 or index > length of Dayso
  if item index of Dayso = a then
    set pos to index
  change index by 1
if pos > 0 then
  say join Tìm thấy, tại vị trí: pos
else
  say Không tìm thấy! for 2 secs
```

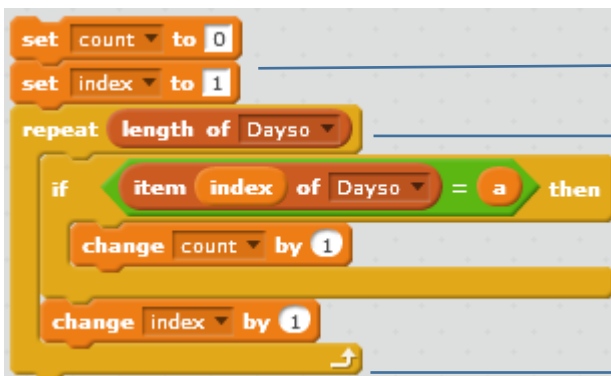
Two Scratch cat characters are shown with speech bubbles. The first cat says: "Dãy số ban đầu: 10 -2 23 45 67 101 -45 79 -1 -13 7 9 25 21". The second cat says: "Tìm thấy, tại vị trí: 13".

Bài toán tìm tất cả các phần tử

Cho trước 1 danh sách dữ liệu, cần tìm ra tất cả các phần tử của dãy thỏa mãn 1 điều kiện nào đó, đếm số lượng các phần tử đã tìm được. Ví dụ:

- Tìm trong lớp tất cả các bạn có tên "Hùng".
- Tìm tất cả các số chia hết cho 3 trong dãy số cho trước.

Cách làm bài này tương tự như bài toán tìm 1 phần tử, chỉ khác là chúng ta sẽ duyệt dãy từ đầu đến cuối và có thêm 1 biến nhớ nữa **count** dùng để đếm số lượng các phần tử tìm được. Nếu **count = 0** tức là không tìm thấy.



The Scratch code block is as follows:

```
set count to 0
set index to 1
repeat length of Dayso
  if item index of Dayso = a then
    change count by 1
  change index by 1
```

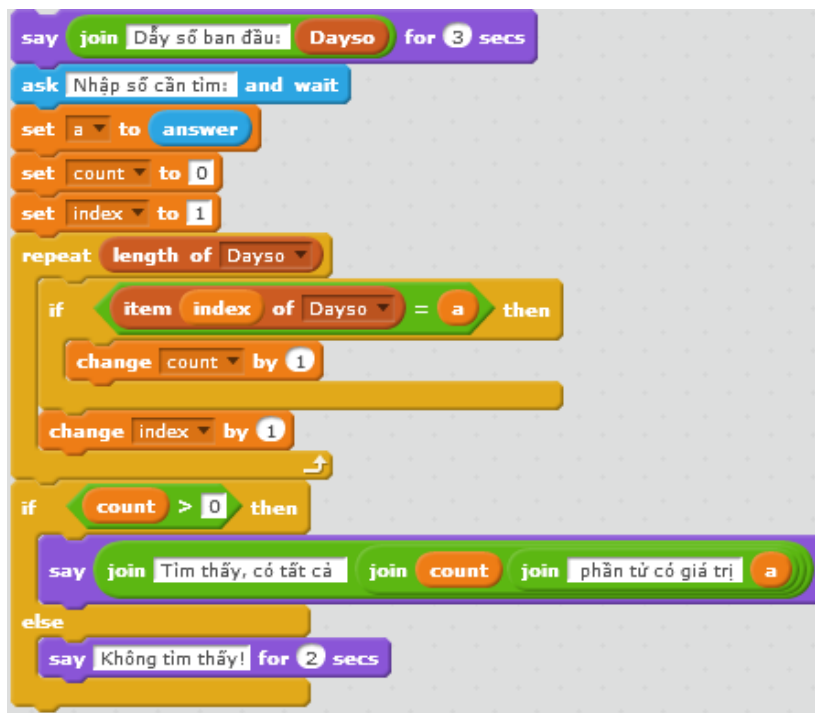
Annotations for the code:

- Thiết lập các giá trị ban đầu. (for the first two 'set' blocks)
- Vòng lặp với số bước bằng độ dài của (for the 'repeat' block)
- Kiểm tra: nếu giá trị phần tử của dãy = a thì tăng biến count lên 1 (for the 'if' block)

Chương trình chính yêu cầu như sau:

- Dãy số cho trước đã được nhập từ trước và
- Chương trình yêu cầu nhập giá trị số cần tìm, sau đó đưa ra kết quả: hoặc thông báo không tìm thấy, hoặc thông báo đã tìm thấy và số lượng phần tử tìm được.

Chương trình hoàn chỉnh như sau.



6. Bài toán sắp xếp dãy

Sắp xếp 1 dãy dữ liệu cho trước là 1 bài toán rất hay gặp trên thực tế. Ví dụ:

- Sắp xếp lớp đứng xếp hàng theo thứ tự từ thấp đến cao.
- Sắp xếp tên học sinh trong lớp theo thứ tự ABC (thứ tự từ điển).
- Sắp xếp các tỉnh thành phố theo diện tích, dân số.

Chúng ta sẽ cùng nhau tìm hiểu bài toán sắp xếp dãy số trong hoạt động này. Hãy bắt đầu từ 1 ví dụ đơn giản: sắp xếp dãy 4 số sau theo thứ tự tăng dần.

Dãy số gốc: **10 7 9 4**

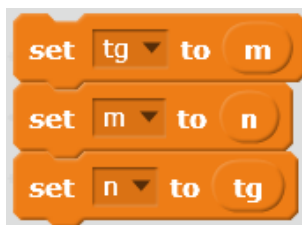
1. Tìm phần tử nhỏ nhất là 4, đổi chỗ với 7, thu được dãy: **4 10 9 7**
2. Tìm phần tử nhỏ nhất trong 3 số cuối, đó là 7, đổi chỗ cho 10, thu được: **4 7 9 10**

Thuật toán sử dụng trên được gọi là **Sắp xếp chọn**.

Đổi chỗ 2 phần tử của dãy

Trong cách làm trên chúng ta thấy thao tác lỗi là "đổi chỗ" hai phần tử trong dãy, hay tổng quát hơn là đổi chỗ 2 biến nhớ bất kỳ trong bộ nhớ máy tính.

Cho 2 biến nhớ **m**, **n** với các giá trị đã gán. Tìm cách đổi giá trị giữa 2 biến nhớ này. Cách thực hiện phổ biến nhất là sử dụng thêm 1 biến nhớ trung gian, ký hiệu là **tg**. Các bước như sau:



Gán m cho tg , $tg = m$

Gán n cho m , $m = n$

Gán tg cho n , $n = tg$

Đoạn chương trình đổi chỗ 2 phần tử thứ i, j của dãy số **dayso** như sau.



Bài toán sắp xếp dãy: Thuật toán chọn

Giả sử dãy cần sắp xếp theo thứ tự tăng dần là: a_1, a_2, \dots, a_n

Thuật toán chọn thực hiện như sau:

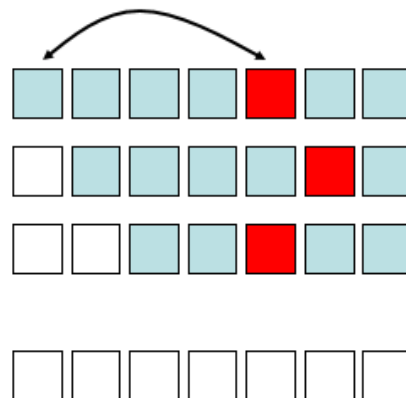
Bước 1. Tìm trong các số a_2, a_3, \dots, a_n phần tử nhỏ nhất, giả sử là a_j . Đổi chỗ a_1, a_j nếu $a_1 > a_j$

Bước 2. Tìm trong các số a_3, a_4, \dots, a_n phần tử nhỏ nhất, giả sử là a_j . Đổi chỗ a_2, a_j nếu $a_2 > a_j$

.....

Bước $n-2$. Tìm trong các số a_{n-1}, a_n phần tử nhỏ nhất, giả sử là a_j . Đổi chỗ a_{n-2}, a_j nếu $a_{n-2} > a_j$

Bước $n-1$. Đổi chỗ a_{n-1}, a_n nếu $a_{n-1} > a_n$.



Có thể mô tả thuật toán này theo cách viết sau:

Vòng lặp $n-1$ lần, cho i chạy từ 1 đến $n-1$

Chọn ra phần tử nhỏ nhất trong các số $a_{i+1}, a_{i+2}, \dots, a_n$

Giả sử chỉ số này là j

Nếu $a_i > a_j$ thì đổi chỗ a_i, a_j

Để thiết kế chương trình chúng ta tạo ra các biến nhớ sau:

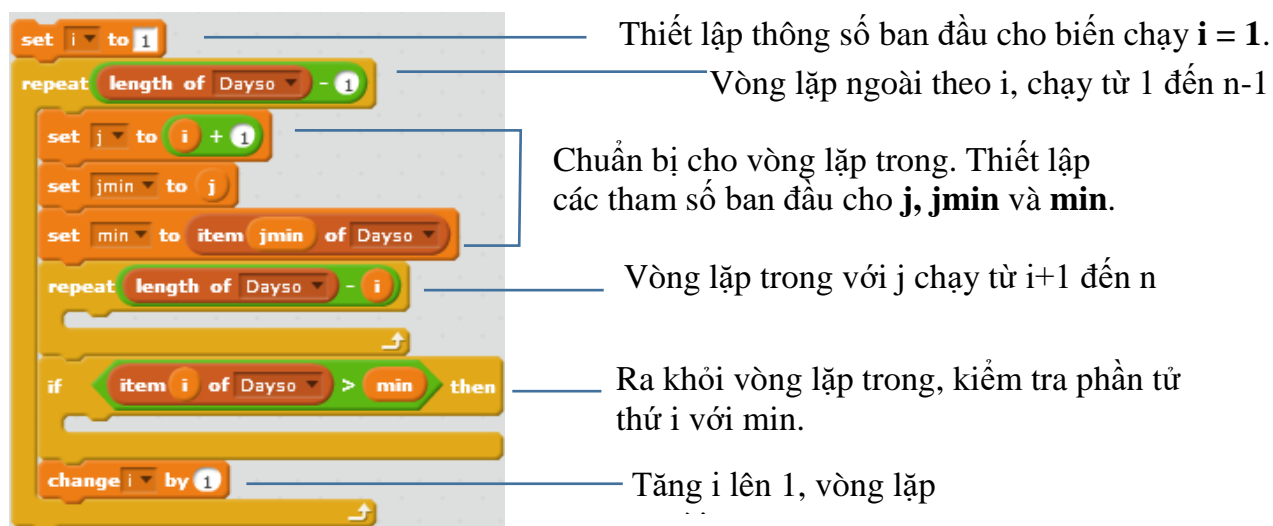
i, j - các biến nhớ chạy ở vòng ngoài và vòng lặp trong.

$jmin$ - chỉ số phần tử nhỏ nhất được tìm thấy ở vòng lặp bên trong.

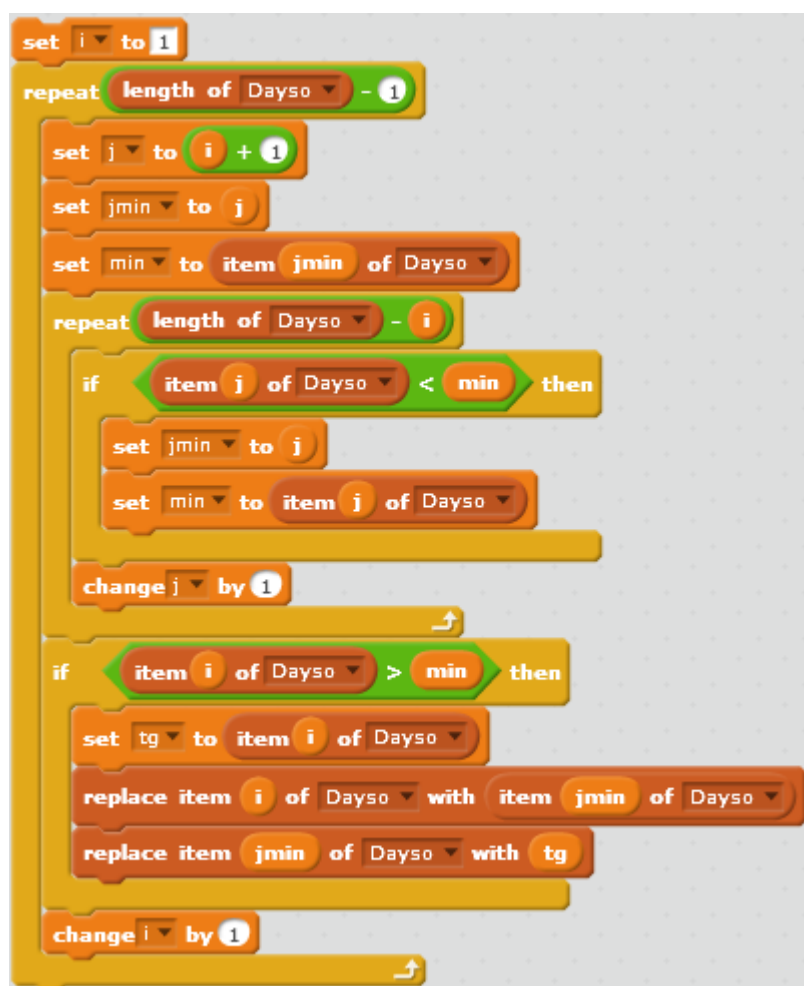
min - giá trị phần tử nhỏ nhất, dùng trong vòng lặp trong tìm min .

tg - biến nhớ trung gian dùng cho việc đổi chỗ 2 phần tử.

Sơ đồ thuật toán được mô tả như sau:



Toàn bộ đoạn chương trình chính như sau:



Em hãy hoàn thiện chương trình này.

Câu hỏi và bài tập

1. Tạo 1 biến danh sách và nhập trực tiếp thành 1 dãy số.

2. Tạo 1 biến danh sách và nhập trực tiếp 1 danh sách tên học sinh trong lớp.
3. Thực hiện bài toán chèn 1 phần tử vào dãy đã sắp xếp đúng: cho trước 1 dãy số đã được sắp xếp theo thứ tự tăng dần, cho trước 1 số P. Hãy viết đoạn chương trình chèn số P này vào dãy trên sao cho dãy vẫn được sắp xếp đúng.
4. Cho trước 1 dãy số, hãy viết chương trình để sắp xếp lại dãy số này theo các yêu cầu sau:
 - (a) Dãy giảm dần.
 - (b) Các số âm lên trước, các số dương ở phía sau.
 - (c) Các số chẵn lên trước, số lẻ ở phía sau.
5. Cho trước 1 danh sách tên học sinh trong lớp. Hãy viết chương trình để sắp xếp lại danh sách học sinh này theo thứ tự từ điển.
6. Cho trước 1 dãy số. Hãy viết chương trình sinh 1 dãy số là 1 hoán vị ngẫu nhiên của dãy số ban đầu.

Mở rộng

Thiết kế 1 chương trình ứng dụng thao tác với dữ liệu là danh sách học sinh như sau:



Input Data

Delete Data

View Data

Khi khởi động chương trình, Mèo sẽ thông báo tên các học sinh hiện có trong danh sách.

Trên màn hình có 3 nút lệnh.

Khi nháy lên nút **Input Data**, quá trình nhập trực tiếp dữ liệu từ bàn phím bắt đầu. Nhập liên tục cho đến khi nhấn Enter không nhập gì cả.

Khi nháy lên nút **Delete Data**, chương trình sẽ yêu cầu người dùng nhập tên học sinh muốn xóa, sau đó kiểm tra và xóa tên này

trong danh sách.

Khi nháy lên nút **View Data**, chương trình hiển thị toàn bộ danh sách học sinh hiện có.

Bài 17. Làm việc với List 2

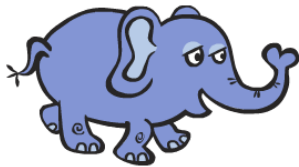
Mục đích

Trong bài này bạn sẽ học và biết:

- Một số ứng dụng đơn giản của dữ liệu List
- Một số thuật toán nâng cao đối với dữ liệu List
- Ứng dụng thực tế của danh sách.

Bắt đầu

Em hãy đọc yêu cầu của 1 trò chơi **Tìm hiểu động vật** sau đây và suy nghĩ xem có thể thiết kế hệ thống dữ liệu như thế nào để có 1 chương trình, bài học hay.



Em thích con vật gì?

Phần mềm sẽ ra liên tục các câu hỏi, em cần trả lời trực tiếp bằng cách gõ bàn phím. Nếu gõ đúng tên con vật thì hình ảnh của động vật này sẽ hiện trên màn hình. Nếu gõ không đúng, gõ sai thì thông báo "sai" hoặc hình ảnh sẽ không hiện.

Cứ như vậy cho đến khi em nhấn Enter ngay thì trò chơi kết thúc.

Các gợi ý bằng câu hỏi:

- Giả sử em có 10 hình ảnh con vật dùng làm dữ liệu cho trò chơi này, em sẽ thiết kế bộ dữ liệu như thế nào.
- Có 1 cách đơn giản nhất là tạo ra 10 biến nhớ và 10 nhân vật trên sân khấu dùng để lưu trữ tên của các con vật và thể hiện hình ảnh con vật.
- Nếu giả sử em muốn tăng số lượng con vật lên thì em sẽ phải làm gì? Có cách nào khi tăng con vật mà không phải tăng thêm biến nhớ hoặc nhân vật không?

Chúng ta hãy tìm hiểu bài toán này và nhiều bài toán tương tự khác trong bài học này.

Nội dung bài học

1. Tìm hiểu động vật

Trong hoạt động đầu tiên này, chúng ta sẽ quay trở lại bài toán, trò chơi Tìm hiểu động vật và cùng xem nếu sử dụng các biến nhớ danh sách thì chương trình sẽ trở nên rất mạch lạc, sáng sủa.



Chương trình sẽ liên tục đưa ra câu hỏi "Em thích con vật gì?". Người chơi trả lời bằng cách nhập tên con vật từ bàn phím.

Nếu nhập đúng hình ảnh con vật sẽ hiện trên màn hình và giáo sư sẽ thông báo thông tin kỹ hơn về động vật này.

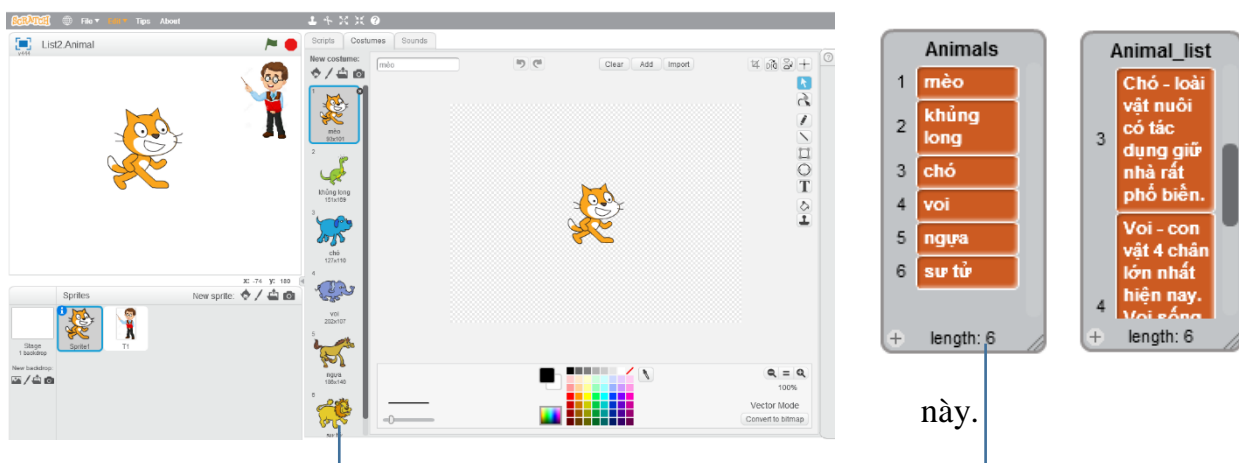
Trò chơi kết thúc khi người chơi

Chúng ta sẽ sử dụng 2 biến danh sách chính là **Animal** và **Animal_list**. Biến **Animal** là danh sách tên các con vật cần tìm hiểu. Biến **Animal_list** sẽ lưu thông tin tra cứu chi tiết tương ứng của các con vật đã có trong danh sách **Animal**. Ví dụ 1 bảng thông tin như sau. Chú ý bảng này có thể có số dòng lớn tùy ý.

Animal **Animal_list**

mèo	Mèo - loài vật nuôi trong nhà rất phổ biến trong mọi gia đình.
khủng long	Khủng long - loài vật khổng lồ nhất trên trái đất, đã tuyệt chủng từ cách đây hàng triệu năm
chó	Chó - loài vật nuôi có tác dụng giữ nhà rất phổ biến.
voi	Voi - con vật 4 chân lớn nhất hiện nay. Voi sống nhiều ở châu Phi và châu Á.
ngựa	Ngựa - loài vật thường dùng để kéo xe, chở đồ, rất phổ biến trên thế giới.
sư tử	Sư tử - là động vật ăn thịt lớn nhất trên cạn, được mệnh danh là chúa rừng xanh.

Tiếp theo chúng ta sẽ thiết kế cách lưu trữ hình ảnh các con vật. Chỉ cần tạo 1 nhân vật cho công việc này. Hình ảnh các con vật khác nhau sẽ tương ứng với trang phục của nhân vật



này.

Thiết lập hệ thống trang phục (costume) với tên trùng khớp với bảng **Animal**.

Bây giờ chúng ta sẽ thiết kế sơ đồ hoạt động của chương trình này. Sơ đồ này có thể tóm tắt trong hình sau, bắt đầu từ **sân khấu**.



Yêu cầu học sinh nhập tên 1 con vật từ bàn phím. Thông tin nhập sẽ được lưu trong biến nhớ **name**.

Nếu **name** khác rỗng → xử lý tiếp.

Nếu **name** = rỗng → dừng chương trình.

Chuyển điều khiển sang nhân vật **thầy giáo**.

Tìm kiếm thông tin trong bảng **Animal_list** về con vật ghi trong biến nhớ **name**. Nếu tìm thấy thì hiển thị thông tin con vật này trên



Chuyển điều khiển sang nhân vật **con mèo**.

Chuyển trang phục sang tên lưu trong biến nhớ **name** và thể hiện.



Gui thông điệp:

Gui thông điệp: **End**



Chuyển điều khiển sang nhân vật **con mèo**.

Àn, không thể hiện trên màn hình.

Chúng ta sẽ bắt đầu từ sân khấu. Sân khấu trong Scratch cũng có cửa sổ lệnh của riêng

mình. Trong bài toán này, sân khấu sẽ bắt đầu chương trình với sự kiện



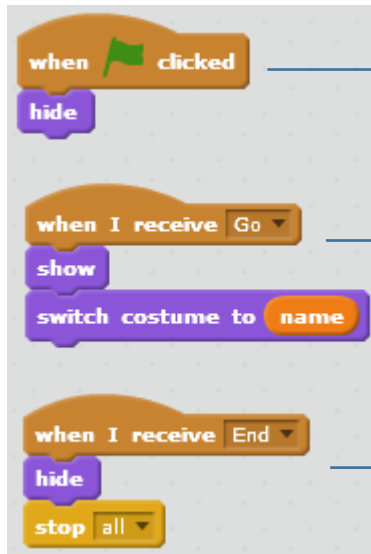
Thực hiện lệnh yêu cầu nhập tên con vật trong 1 vòng lặp vô hạn.

Kiểm tra xem dữ liệu nhập có rỗng hay không.

Nếu dữ liệu nhập không rỗng thì lưu kết quả nhập vào biến **name** và truyền thông điệp **Go** cho 2 nhân vật của chương trình.

Nếu dữ liệu nhập là rỗng thì truyền thông điệp **End**.

Đây là cửa sổ lệnh của nhân vật con mèo (con mèo có 5 trang phục khác).

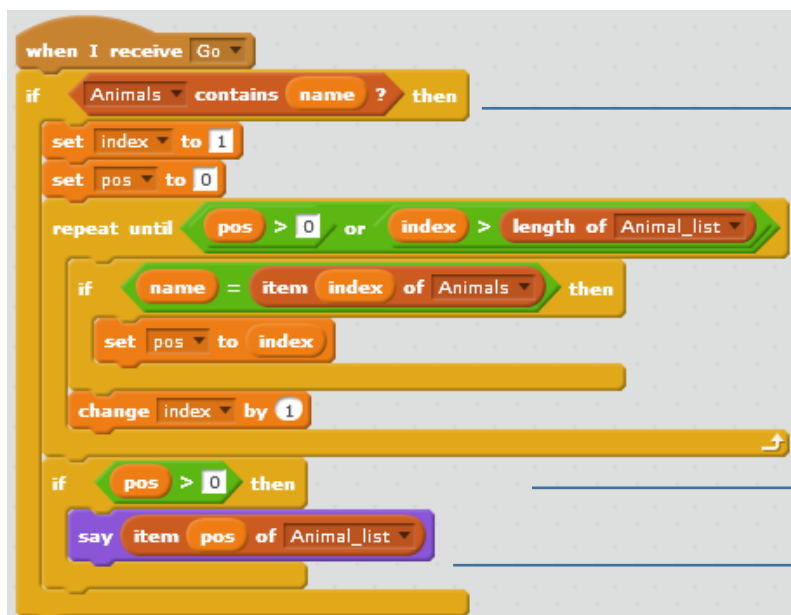


Khi bắt đầu chương trình, nhân vật không hiển thị trên màn hình.

Khi nhận thông điệp **Go**, chuyển trang phục sang giá trị ghi trong biến nhớ **name** và hiển thị trên

Khi nhận thông điệp **End**, ẩn không hiển thị và dừng toàn bộ chương trình.

Khi nhận thông điệp **Go**, nhân vật thầy giáo cần thực hiện xử lý biến nhớ **name**, kiểm tra xem **name** có tên trong bảng **Animal** không. Nếu có thể tìm ra chỉ số chính xác dòng chứa tên tương ứng với biến **name** này, sau đó sẽ thể hiện trên màn hình dữ liệu có trong bảng **Animal_list**. Chỉ số chính xác dòng chứa tên con vật tương ứng với **name** được lưu trong biến nhớ **pos**.



Kiểm tra điều kiện tên con vật trong biến **name** có nằm trong danh sách **Animal** hay không, nếu có mới xử lý

Vòng lặp này có tính năng tìm ra giá trị **pos** là vị trí của biến **name** trong danh sách

Thông báo thông tin chi tiết về con vật, lấy thông tin từ bảng **Animal_list**.

2. Bài toán sinh hoán vị, tập con ngẫu nhiên

Trên thực tế có rất nhiều bài toán đòi hỏi việc sinh ngẫu nhiên các tập hợp con, sinh ngẫu nhiên các hoán vị, ... Trong hoạt động này chúng ta sẽ cùng giải quyết một vài bài toán đó trong môi trường Scratch.

Bài toán 1. Cho trước 1 xâu ký tự **Str**, cần sinh ra 1 xâu **Strout** là hoán vị ngẫu nhiên của xâu **Str** ban đầu.

Bài toán 2. Cho trước dãy số (hoặc chữ) a_1, a_2, \dots, a_n . Cần sinh ra 1 hoán vị ngẫu nhiên của dãy này, đưa ra dãy sau: b_1, b_2, \dots, b_n là hoán vị của dãy ban đầu.

Bài toán 3. Cho trước dãy số (hoặc chữ) a_1, a_2, \dots, a_n . Cần sinh ra một dãy ngẫu nhiên m phần tử của dãy trên: b_1, b_2, \dots, b_m với m cho trước.

Bài toán 1. Cho trước 1 xâu ký tự **Str**, cần sinh ra 1 xâu **Strout** là hoán vị ngẫu nhiên của xâu **Str** ban đầu.

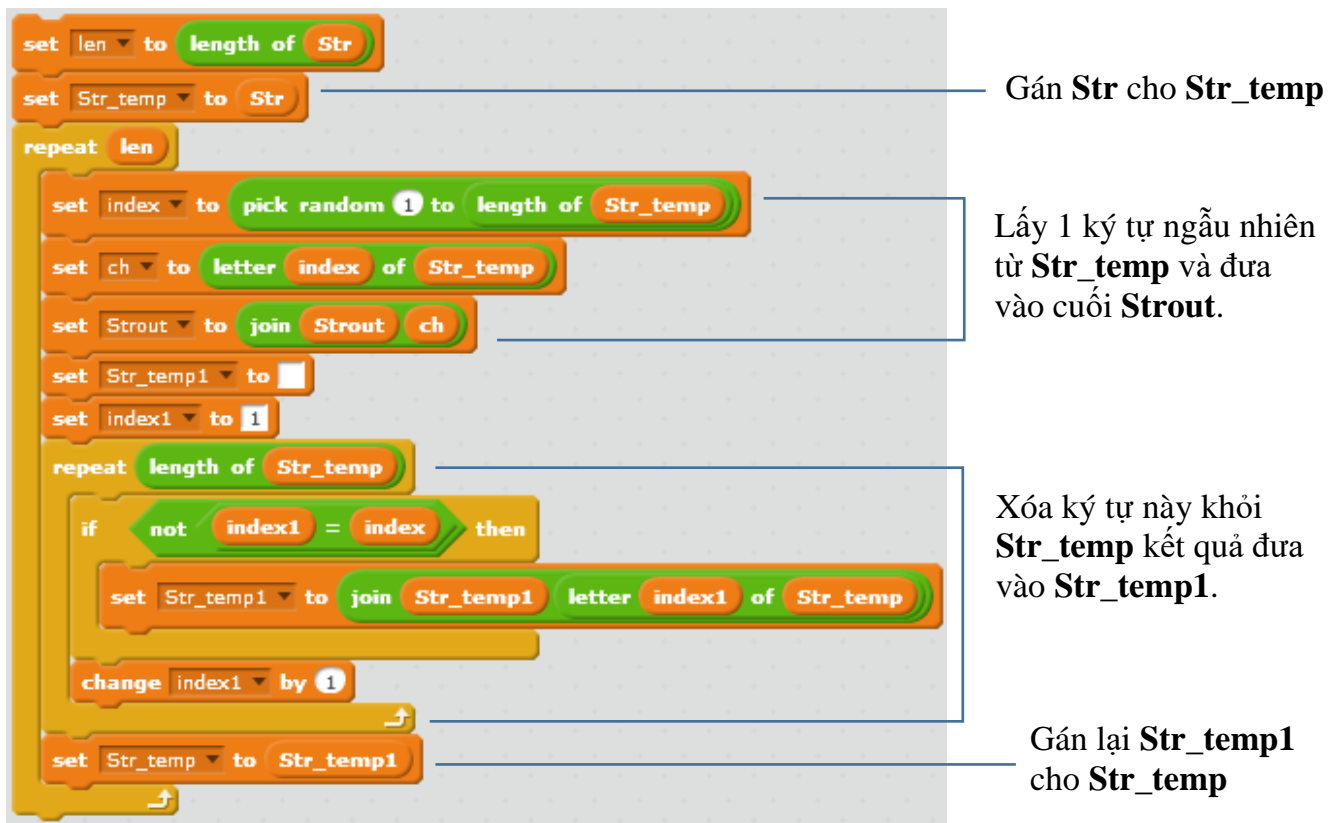
Ý tưởng ban đầu của thuật toán này khá đơn giản: lấy ngẫu nhiên các ký tự từ xâu **Str** và bổ sung vào xâu **Strout**. Yêu cầu của bài toán là sau khi thực hiện giá trị xâu **Str** cần giữ nguyên, không thay đổi. Sử dụng các lệnh cụ thể của Scratch có thể viết lại cách làm trên dưới dạng cây lệnh như sau:

```
Đặt Strout = rỗng, len = độ dài xâu Str
Đặt Str_temp = Str
Thiệp lặp lặp với số vòng lặp = len
    Lấy ra 1 ký tự bất kỳ của xâu Str_temp
    Đưa ký tự này vào cuối của Strout
    Xóa ký tự này từ Str_temp
```

Trong Scratch không có lệnh xóa 1 ký tự của xâu, do đó dòng cuối cùng của thuật toán trên cần thực hiện như 1 chương trình. Thuật toán xóa 1 ký tự có chỉ số **index** khỏi 1 xâu **Str_temp** như sau:

```
Đặt Str_temp = rỗng, index1 = 1, len = độ dài xâu Str
Thiệp lặp lặp với số vòng lặp = len
    Kiểm tra: nếu index1 <> index thì
        Lấy ra ký tự thứ index1 của Str
        Đưa ký tự này vào cuối của Str_temp
        Tăng index lên 1
Thiết lập Str = Str_temp
```

Kết hợp 2 thuật toán trên, chúng ta có chương trình trong Scratch.



Bài toán 2. Cho trước dãy số (hoặc chữ) a_1, a_2, \dots, a_n . Cần sinh ra 1 hoán vị ngẫu nhiên của dãy này, đưa ra dãy sau: b_1, b_2, \dots, b_n là hoán vị của dãy ban đầu.

Input: Dãy List, **Output:** Dãy List_out

Bài toán 2 cách thực hiện tương tự bài toán 1, chỉ có điểm khác là thực hiện trên biến danh sách, chứ không phải trên biến xâu ký tự.

Cách đơn giản: lấy từ phần tử ngẫu nhiên của List và đưa vào cuối của Listout, sau đó xóa phần tử này khỏi danh sách gốc List.



Cách trên sẽ xóa dữ liệu từ danh sách gốc.

Chương trình dưới đây được viết lại hoàn chỉnh, danh sách gốc List không bị xóa dữ liệu. Do vậy cần tạo thêm 1 biến danh sách trung gian List_temp.



Dãy phần tử gốc:
0123456789



Dãy hoán vị ngẫu
nhiên: 1934206857



Bài toán 3. Cho trước dãy số (hoặc chữ) a_1, a_2, \dots, a_n . Cần sinh ra một dãy ngẫu nhiên m phần tử của dãy trên: b_1, b_2, \dots, b_m với m cho trước.

Input: Dãy List, số m ; Output: Dãy List_out

Ý tưởng và cách thực hiện bài toán này tương tự bài toán 2 và khá đơn giản như sau, chú ý rằng dãy gốc List yêu cầu không bị xóa.

Thuật toán lấy ra ngẫu nhiên m phần tử từ 1 dãy gốc ban đầu **List** được viết như sau:

Đặt dãy List_temp và Listout = rỗng.

Gán từng phần tử của dãy List vào List_temp.

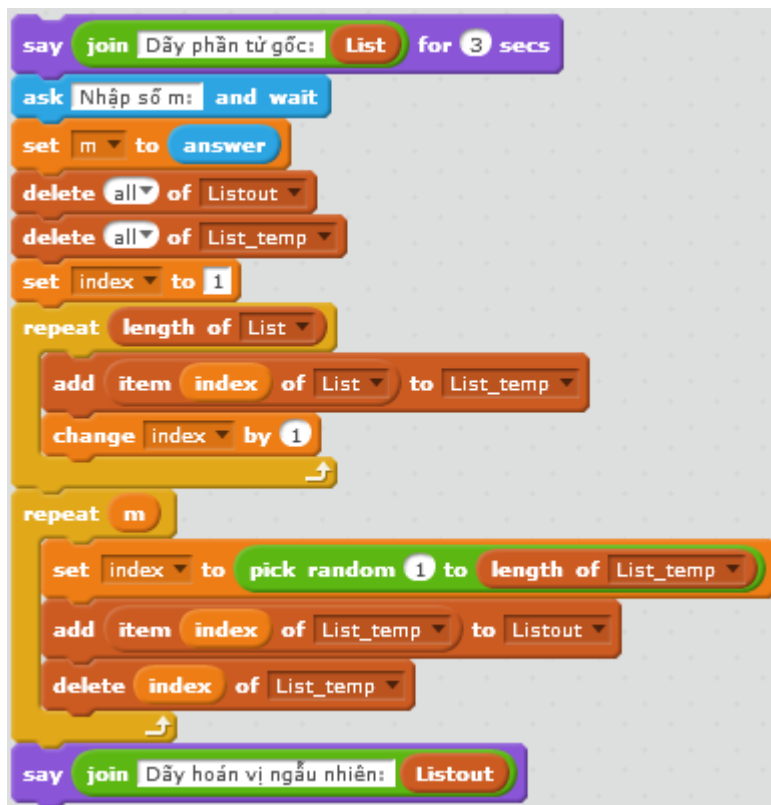
Thiết lập vòng lặp độ dài m

Lấy ra 1 phần tử ngẫu nhiên của List_temp.

Đưa phần tử này vào cuối của danh sách Listout.

Xóa phần tử này từ List_temp.

Chương trình hoàn chỉnh trên Scratch như sau.



3. Từ điển sinh bài kiểm tra trắc nghiệm

Chúng ta sẽ cùng tìm hiểu một ứng dụng nữa của dữ liệu danh sách, khi chúng được dùng như dữ liệu từ điển để sinh ngẫu nhiên các bài kiểm tra trắc nghiệm.

Mô hình bài toán cụ thể như sau.

Giả sử có 2 bảng dữ liệu, bao gồm 2 danh sách quốc gia và thủ đô tương ứng của quốc gia đó. Ví dụ 1 bảng như vậy.

Quốc gia	Thủ đô
Argentina	Buenos Aires
Armenia	Yerevan
Australia	Canberra
Austria	Viên
Azerbaijan	Baku
Croatia	Zagreb
Cuba	La Habana
Síp	Nicosia
Cộng hòa Czech	Praha
Venezuela	Caracas
Việt Nam	Hà Nội
Liên bang Nga	Moscow

Vương quốc Anh	Luân Đôn
Hoa kỳ	Washington
Uruguay	Montevideo
Uzbekistan	Tashkent

Chương trình cần đưa ra các loại câu hỏi trắc nghiệm sau, dữ liệu được lấy ngẫu nhiên từ các bảng dữ liệu trên.

(a) <Thủ đô> là thủ đô của nước nào?

(b) <Quốc gia> có thủ đô là ?



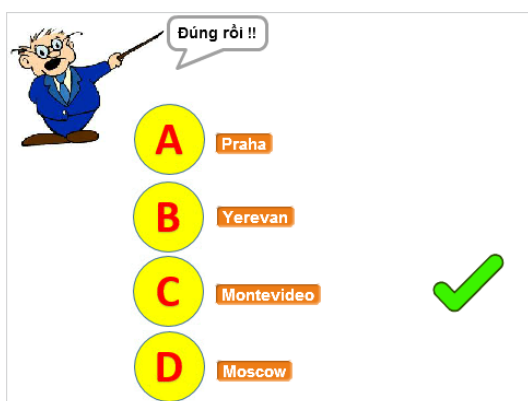
Phần mềm sẽ tự động sinh bài kiểm tra trắc nghiệm theo 1 trong 2 loại câu hỏi như trên.

Trên màn hình có 4 nút A, B, C, D. Bên cạnh là bộ 4 dữ liệu được lấy ra và hiển thị ngẫu nhiên trên màn hình.

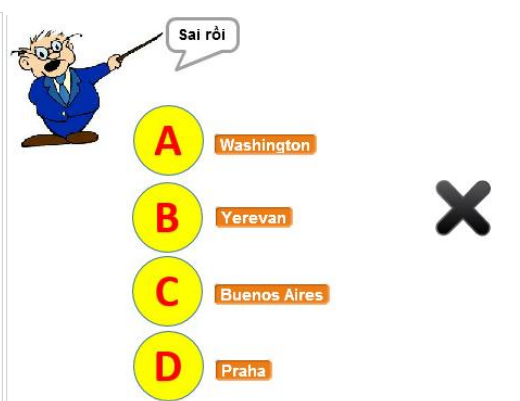
Người thực hiện bài kiểm tra bằng cách nháy chuột lên 1 trong các đáp án.

Chương trình sẽ thông báo ngay đúng hay sai, hiển thị dấu đúng, sai và chuyển sang câu hỏi tiếp theo.

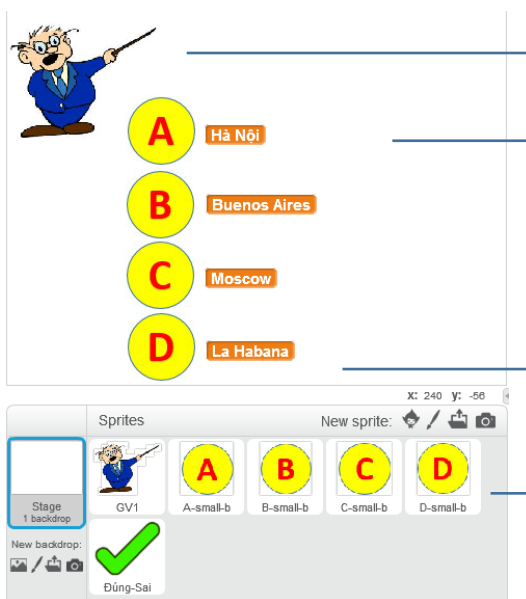
Giao diện khi làm bài như sau:



Giao diện khi nháy chọn phương án đúng, chú ý dấu tích bên cạnh phương án đã click.



Giao diện khi nháy chọn phương án sai, chú ý dấu tích chéo bên cạnh phương án đã click.



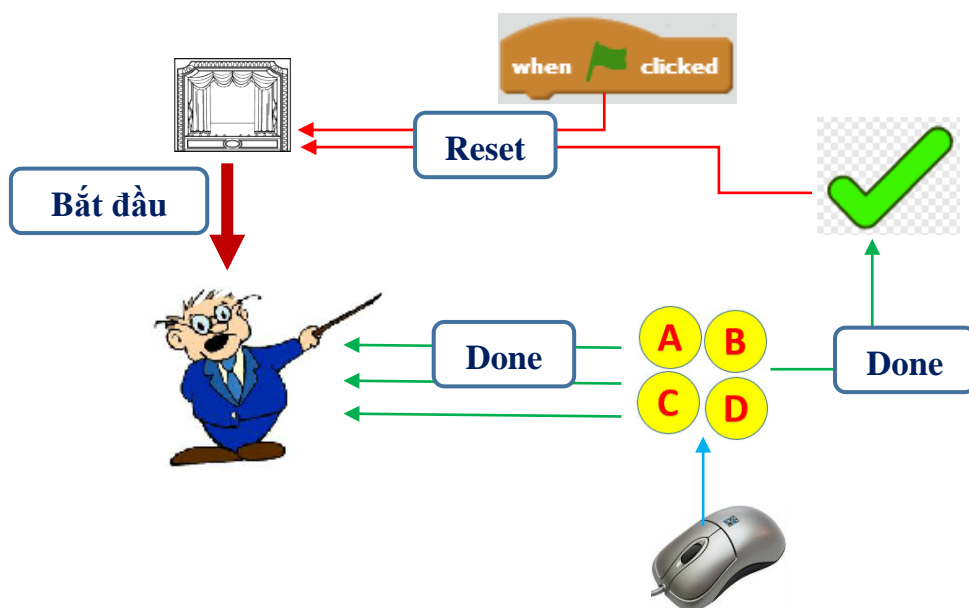
Nhân vật thầy giáo sẽ ra câu hỏi trắc nghiệm trên màn hình.

Vị trí các nút A, B, C, D và 4 biến nhớ dùng để thể hiện bộ dữ liệu được sinh ngẫu nhiên cho câu hỏi trắc nghiệm.

Nhân vật và sân khấu chính: thầy giáo, 4 nút A, B, C, D và nút dấu Đúng - Sai.

Sân khấu sẽ tham gia giải quyết bài toán

Sơ đồ hoạt động của chương trình như sau:



Sau đây là giải thích cho sơ đồ trên.

- Bắt đầu chương trình, thông điệp **Reset** (làm lại) được đưa ra và chuyển tới sân khấu nền để tính toán, xử lý các thông tin ban đầu.

- Nhận được thông điệp **Reset**, sân khấu sẽ thực hiện việc sinh ngẫu nhiên dạng câu hỏi trắc nghiệm, sinh ngẫu nhiên bộ 4 phương án, trong đó có 1 phương án đúng. Dữ liệu được lấy từ 2 bảng **dsQG** và **dsThudo**. Sau khi sinh xong bộ dữ liệu sẽ phát đi thông điệp **Bắt đầu** (**Start**). Chú ý là sau khi bộ dữ liệu được sinh, các thông tin này sẽ tự động hiển thị trên màn hình.

- Thầy giáo khi nhận được thông điệp **Bắt đầu** thì thực hiện công việc sau: thể hiện thông tin câu hỏi trên màn hình và chờ người chơi trả lời.

- Người sử dụng trả lời câu hỏi do thầy giáo đưa ra bằng cách nhấp chuột lên 1 trong các nút A, B, C, D.

- Nút ABCD (1 trong A, B, C, D) khi nhận được cảm biến chuột thì lập tức gán biến nhớ choice cho phương án mà người dùng trả lời, sau đó phát thông điệp **Done** (đã xong). Thông điệp này sẽ phát đồng thời cho thầy giáo và nút dấu Đúng-Sai.

- Thầy giáo khi nhận thông điệp **Done** sẽ lập tức kiểm tra xem đáp án người dùng đã chọn là đúng hay sai và thông báo đúng / sai ngay trên màn hình.

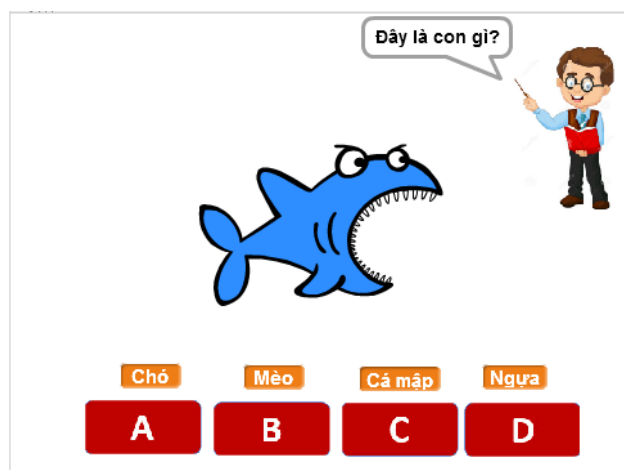
- Nút dấu Đúng - Sai khi nhận thông điệp **Done** sẽ lập tức kiểm tra xem đáp án người dùng đã chọn là đúng hay sai và tính toán, thay đổi trang phục và dùng lệnh **stamp** (sau khi đã **show**) để in dấu Check (đúng) hoặc Chéo (sai) tại vị trí tương ứng mà người dùng đã nhấp chuột.

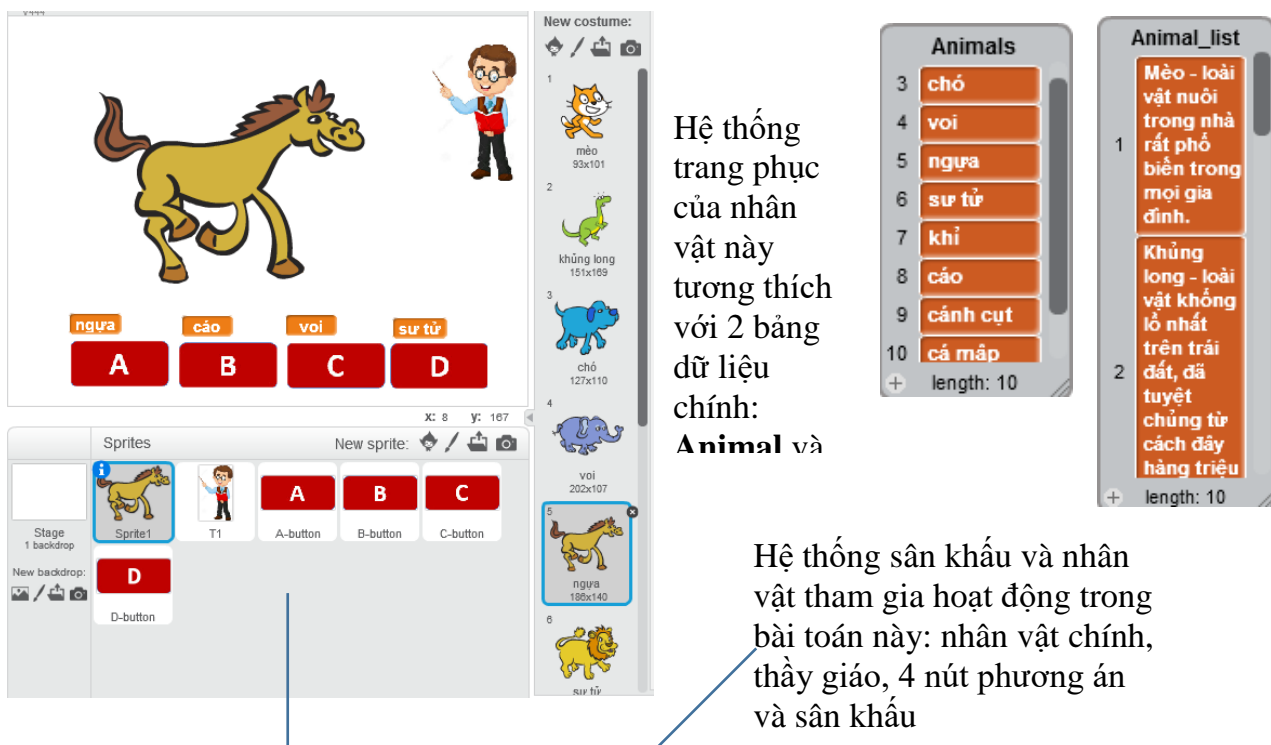
- Nút Đúng - Sai chờ 3 giây sau thì phát thông điệp **Reset**. Khi nhận thông điệp Reset thì nút dấu Đúng - Sai sẽ ẩn bằng cách thực hiện lệnh **hide**.

Bài tập: em hãy viết chương trình hoàn chỉnh cho bài toán trên.

4. Bài luyện trắc nghiệm có hình ảnh

Bài toán trong hoạt động này có ý nghĩa gần tương tự như trong hoạt động 3 về dữ liệu sinh bài kiểm tra trắc nghiệm. Dữ liệu và hình ảnh của hoạt động này lấy từ bài toán Tìm hiểu động vật của hoạt động 1.





Hệ thống sân khấu và nhân vật tham gia hoạt động trong bài toán này: nhân vật chính, thầy giáo, 4 nút phương án và sân khấu

Sơ đồ làm việc của chương trình thể hiện trong mô hình sau.

Bài tập: em hãy viết chương trình hoàn chỉnh cho bài toán trên.

Trong ví dụ trên chúng ta sử dụng 2 biến nhớ dạng danh sách ds_QG và ds_Thudo có liên quan đến nhau theo nghĩa 1-1, nghĩa là mỗi dòng của bảng này tương ứng với chính dòng đó

của bảng kia. Đây là cách tạo quan hệ đơn giản giữa hai bảng. Thực tế các quan hệ giữa các bảng có thể phức tạp hơn. Chúng ta hãy quan sát 1 ví dụ của quan hệ như vậy.

Giả sử chúng ta có 3 biến nhớ dạng danh sách như sau:

Tỉnh	Chỉ số	Vùng
Thái Nguyên	1	1. Trung du Bắc Bộ
Hà Nội	2	2. Đồng bằng Bắc Bộ
Thanh Hóa	3	3. Trung Bộ
Thái Bình	2	4. Tây Nguyên
Quảng Ngãi	3	5. Nam Bộ
Lâm Đồng	4	
Cần Thơ	5	
Kiên Giang	5	
Gia Lai	4	
Cao Bằng	1	

Bài toán 1. Viết chương trình nhập các bảng dữ liệu có liên kết trên.

Giả sử đã có danh sách các vùng miền, cần lập chương trình nhập danh sách các tỉnh, thành phố với chỉ số liên kết vùng miền chính xác.



Đầu tiên chương trình thông báo vị trí của nút lệnh nhập dữ liệu. Khi nháy lên nút Input Data, em bắt đầu nhập dữ liệu. Việc nhập dữ liệu được tiến hành theo 2 bước:

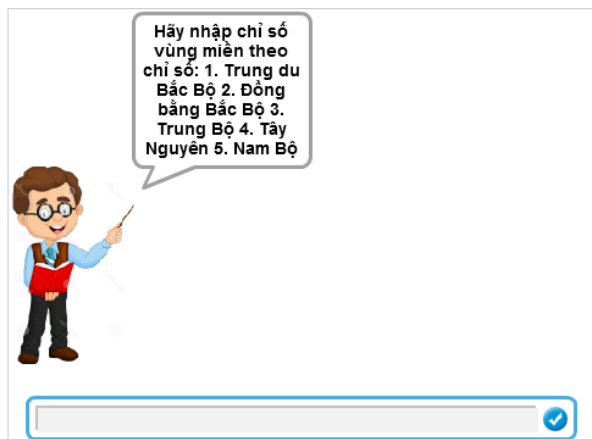
Bước 1: nhập tên tỉnh, thành phố (không được trùng với tên đã có).

Bước 2: nhập chỉ số vùng, miền.

Em hãy thiết kế và hoàn thiện chương trình này.

Bài toán 2. Bài toán tra cứu dữ liệu

Giả sử đã có danh sách các vùng miền, cần lập chương trình yêu cầu nhập chỉ số vùng miền, chương trình sau đó sẽ đưa ra danh sách tất cả các tỉnh thành phố thuộc vùng miền này.



Đầu tiên chương trình yêu cầu em nhập 1 số là chỉ số vùng, miền muốn tra cứu dữ liệu.

Sau đó chương trình sẽ tính toán và hiện kết quả tra cứu là số lượng và danh sách cụ thể các tỉnh, thành phố thuộc vùng, miền đã nhập.

Nếu nhập không đúng chỉ số vùng, miền,

Em hãy thiết kế và hoàn thiện chương trình này.

Câu hỏi và bài tập

1. Cho trước 1 dãy số tự nhiên bất kỳ, hãy viết chương trình thực hiện các công việc sau:

- Đếm và liệt kê các số chẵn của dãy trên.
- Đếm và liệt kê các số là nguyên tố trong dãy trên.

2. Giả sử chúng ta đã có 1 dãy số hoặc chữ: a_1, a_2, \dots, a_n . Viết chương trình nhập số tự nhiên m và sinh ra ngẫu nhiên các số b_1, b_2, \dots, b_m lấy theo thứ tự từ trái sang phải của dãy ban đầu.

3. Dãy số Fibonacci là dãy số 1, 1, 2, 3, 5, 8, 13, tức là dãy a_1, a_2, \dots, a_n thỏa mãn điều kiện:

$$a_1 = a_2 = 1$$

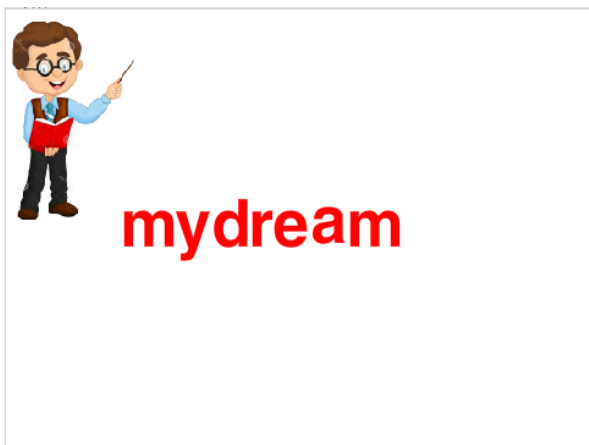
$$a_n = a_{n-1} + a_{n-2} \text{ với } n > 2$$

Hãy viết chương trình nhập số m từ bàn phím và thể hiện trên màn hình m số hạng đầu tiên của dãy Fibonacci này.

Mở rộng

Hãy thiết kế và viết chương trình mô phỏng bài học phát âm tiếng Anh có âm thanh sau.

Phần mềm yêu cầu nhập 1 từ tiếng Anh bất kỳ, sau đó thông báo và thể hiện cách phát âm của từ này với âm thanh thật.



Yêu cầu bổ sung của chương trình:

Khi thực hiện việc phát âm, chương trình sẽ đưa ra màn hình lần lượt các chữ cái của từ đã cho ngay trên màn hình, đưa chữ và âm thanh đồng thời.

Bài 18. Thủ tục 1

Mục đích

Học xong bài này em sẽ:

- Biết và hiểu được vai trò của thủ tục như 1 nhóm con chương trình nhằm giúp giải quyết công việc dễ hơn.

Bắt đầu

1. Tư duy chia 1 bài toán lớn thành các bài toán nhỏ hơn.

Trên thực tế khi gặp một công việc lớn, con người luôn tìm cách chia nhỏ công việc đó ra để làm từng bước.

- Khi được giao quét 1 ngôi nhà lớn, em sẽ chia thành những phòng nhỏ và quét dọn từng phòng một.

- Nếu phải bê 1 chồng sách lớn em sẽ khuân và bê chuyển từng cuốn sách hoặc từng vài ba cuốn sách cho nhẹ.

- Khi được giao nhiều bài tập về nhà, em thường chia ra mỗi ngày làm 1 ít bài tập để cuối tuần khỏi dồn lại thành nhiều.

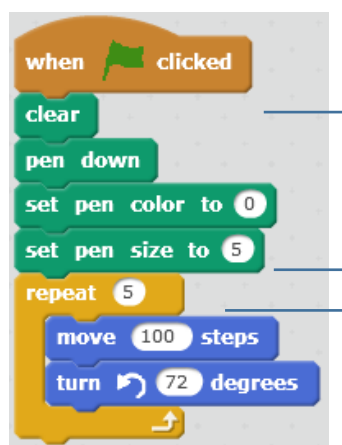
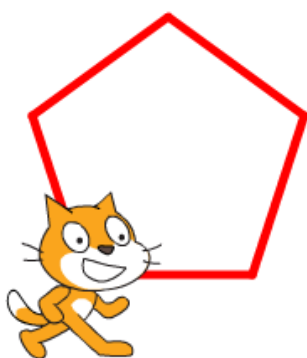
Em có thể chỉ ra rất nhiều ví dụ thực tế nữa.

Trong bài toán tin học, việc chia 1 công việc lớn thành nhiều việc nhỏ hơn để làm từng việc được gọi chung là "chia để trị". Đây là một kỹ thuật hay dùng nhất trong các bài toán tin học, nhất là lập trình máy tính.

Nội dung bài học

1. Có thể rút gọn chương trình bằng cách nào?

Chúng ta hãy bắt đầu hoạt động này bằng chương trình vẽ 1 đa giác đều đã biết.



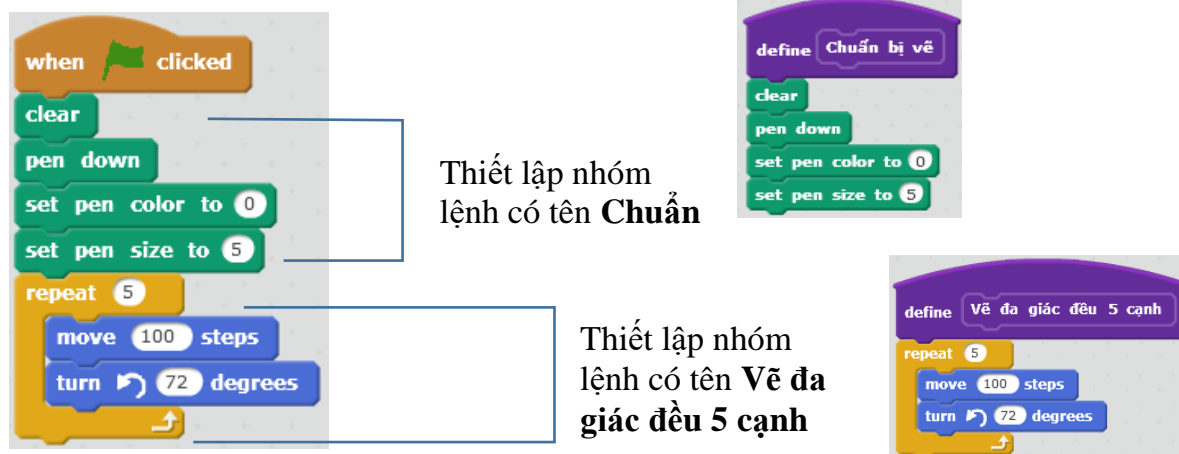
Đoạn chương trình chuẩn bị cho việc vẽ.

chương trình vẽ hình đa giác đều 5 cạnh.

Đoạn chương trình, như em thấy, có thể chia làm 2 phần với chức năng tách biệt rõ rệt.

- Phần 1 bao gồm các lệnh thiết lập ban đầu cho việc vẽ.
- Phần 2 là các lệnh vẽ hình đa giác đều.

Chương trình trên bây giờ có thể tách làm 2 phần, phần 1 có tên "**Chuẩn bị vẽ**", phần 2 có tên "**Vẽ đa giác đều 5 cạnh**".



Khi đó chương trình gốc của chúng ta bây giờ rút gọn lại chỉ còn 2 "lệnh".



Em có nhận xét gì về hoạt động trên?

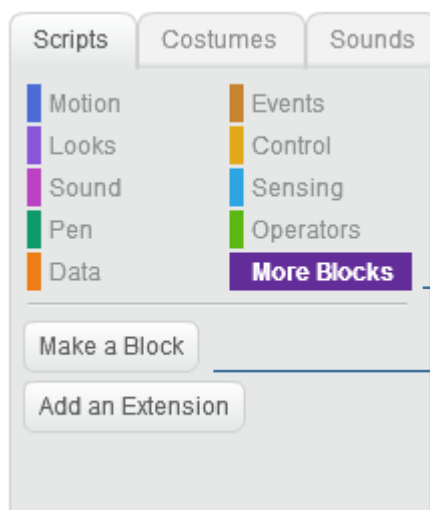
Các gợi ý:

- Nhóm các lệnh vào thành 1 lệnh mới?
- Việc phân nhóm thế này có lợi ích gì?

2. Thiết lập và sử dụng khái niệm thủ tục trong Scratch

Trong hoạt động này, em sẽ thực hiện thao tác thiết lập các thủ tục cụ thể cho một chương trình cụ thể. Chúng ta hãy bắt đầu từ chương trình vẽ hình đa giác đều 5 cạnh trên.

Lệnh thiết lập thủ tục có trong nhóm lệnh **More Blocks** như hình sau.



Nhóm lệnh **More Blocks** (thêm các lệnh khác), trong đó có lệnh làm việc với

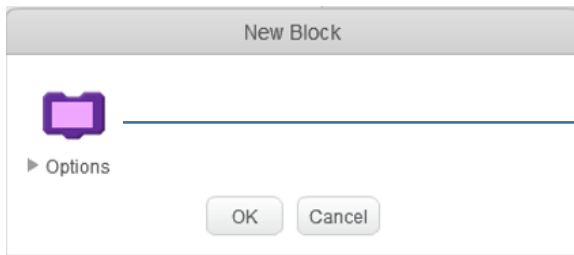
Lệnh khởi tạo thủ tục.

Để khởi tạo 1 thủ tục mới, nhấn nút **Make a Block**

Make a Block

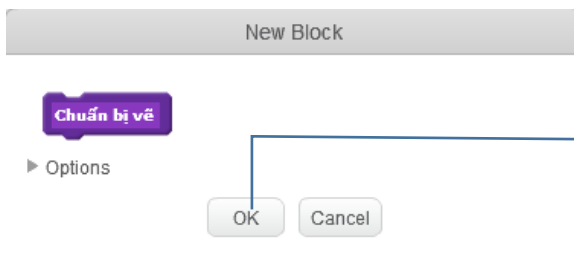
trong nhóm lệnh More Blocks.

Cửa sổ New Block xuất hiện.

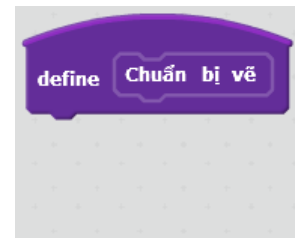


Nhập tên thủ tục tại vị trí này.
Nhập xong nhấn nút OK.

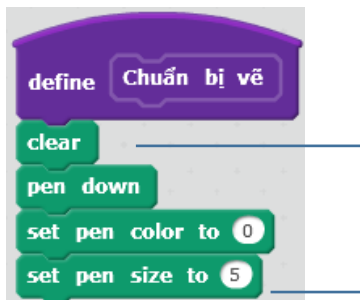
Ví dụ em tạo ra thủ tục với tên "Chuẩn bị vẽ" như hình sau.



Nhấn nút OK em sẽ
thấy xuất hiện lệnh
define trong cửa sổ



Bây giờ em hãy kéo thả nhóm lệnh có chức năng là các lệnh chuẩn bị cho việc vẽ vào khung của lệnh **define** là lệnh định nghĩa thủ tục như sau. Khung bên dưới của lệnh **define** dùng để định nghĩa nội dung của thủ tục.



Kéo thả nhóm các lệnh này vào dưới và gắn
với lệnh định nghĩa thủ tục (**define**).

Tương tự em hãy tạo thêm 1 thủ tục nữa có tên **Vẽ đa giác đều 5 cạnh** và định nghĩa thủ tục này như sau:



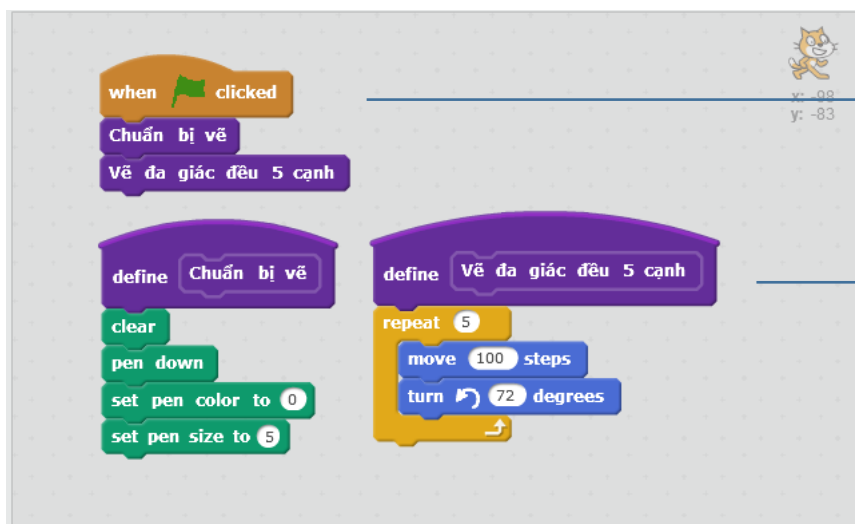
Bây giờ, khi nhìn vào khu vực mẫu lệnh của nhóm More Blocks chúng ta sẽ thấy xuất hiện 2 lệnh tương ứng với 2 thủ tục vừa tạo ra.



2 lệnh tương ứng với 2 thủ tục vừa khởi tạo xuất hiện trong khung lệnh.

Các lệnh mới này sẽ được sử dụng hoàn toàn giống như các lệnh khác mà em đã biết, và có thể sử dụng nhiều lần.

Bây giờ trong khung chương trình chính, em chỉ cần sử dụng 2 lệnh trên để tạo thành đoạn chương trình chính thức của bài toán. Cửa sổ lệnh của nhân vật có khuôn dạng như sau:



Đoạn chương trình chính chỉ còn 2

2 thủ tục được định nghĩa cùng trên 1 màn hình lệnh.

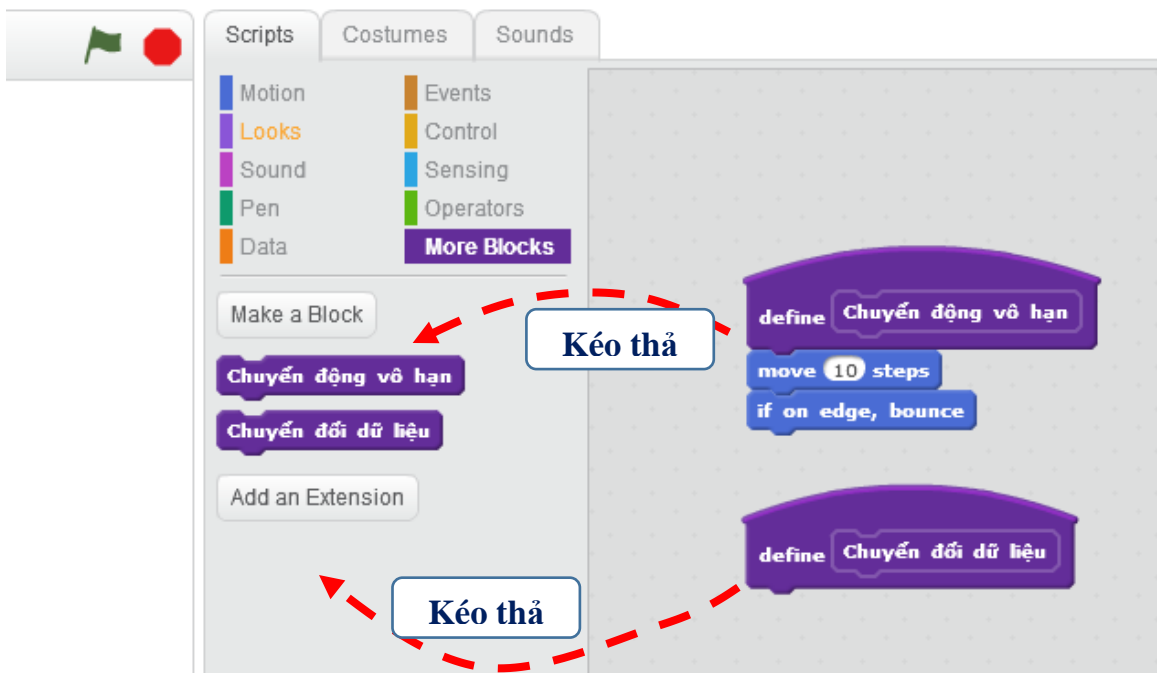
Theo em sử dụng thủ tục mang lại những lợi ích nào? Chọn các phương án em thấy đúng, giải thích vì sao.

- A. Nhìn chương trình thấy dễ hiểu hơn, đơn giản hơn.
- B. Chương trình chính được viết ngắn gọn hơn.
- C. Sử dụng thủ tục sẽ làm cho số lệnh được viết ra ít hơn, ngắn gọn hơn.
- D. Mỗi thủ tục có thể hiểu là 1 bài toán con.
- E. Làm cho chương trình trở nên dễ hiểu, có cấu trúc hơn.

3. Các thao tác khác với thủ tục

1. Xóa thủ tục

Muốn xóa 1 thủ tục chúng ta kéo thả dòng lệnh **define** thủ tục nào vào khung điều khiển lệnh ở giữa màn hình. Chú ý trước khi xóa thủ tục cần hủy hết các dòng lệnh của thủ tục này trong cửa sổ lệnh.

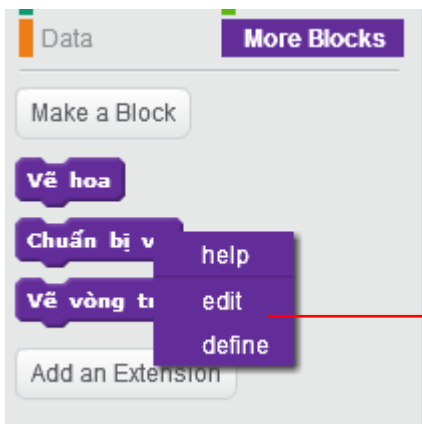


Chú ý:

- Nếu chỉ kéo thả 1 dòng lệnh define thủ tục thì sẽ xóa thủ tục này.
- Nếu kéo thả thủ tục và các lệnh gắn kèm theo thì sẽ xóa cả thủ tục và các lệnh này.

2. Đổi tên thủ tục

Muốn đổi tên 1 thủ tục cần thực hiện lệnh: nhấp chuột phải lên dòng chứa thủ tục đó tại khung điều khiển lệnh, sau đó chọn chức năng **edit**.



Chức năng đổi tên thủ tục đã xác định trước

4. Một số ví dụ đơn giản về thủ tục

Trong hoạt động này chúng ta sẽ cùng thực hiện các bài toán thiết kế các thủ tục cụ thể.

1. Thủ tục vẽ đa giác đều n cạnh với độ dài cạnh d , các số n, d cho trước

Thủ tục này đã được mô tả trong hoạt động trên. Chúng ta sử dụng lệnh **Make a Block** để tạo ra 1 thủ tục có tên **Vẽ đa giác đều**, sau đó kéo thả các lệnh tương ứng vào khung của thủ tục này như sau.



Nội dung của thủ tục **Vẽ đa giác đều**. Các biến nhớ **n** và **d** được khởi tạo từ trước.

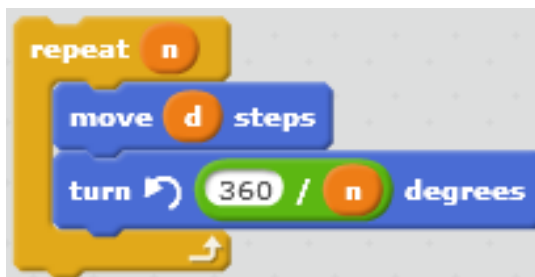


Sau đây là minh họa cho việc chạy thủ tục này. Trước khi chạy các biến nhớ **n** và **d** cần gán trước các giá trị.

2. Thủ tục vẽ vòng tròn biết tâm và bán kính

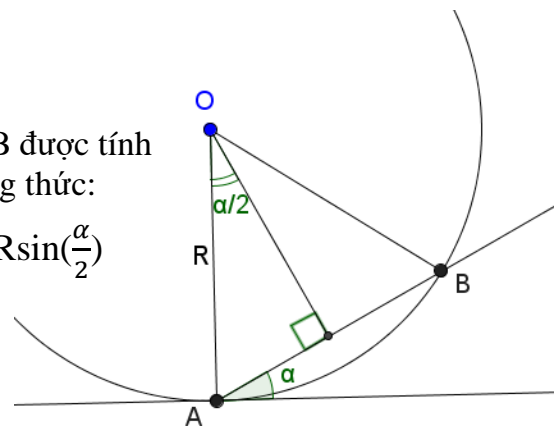
Thủ tục này yêu cầu có 3 tham số là (**T_x**, **T_y**) - tọa độ tâm của vòng tròn và **R** - bán kính vòng tròn cần vẽ.

Để phân tích thủ tục này chúng ta xuất phát từ thủ tục vẽ đa giác đều trong hoạt động trước

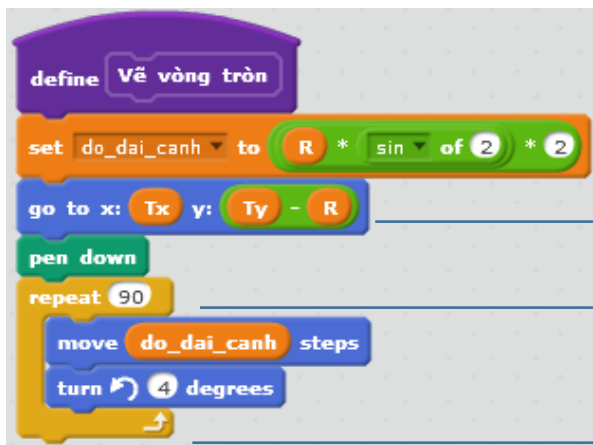


Cạnh AB được tính theo công thức:

$$AB = 2R\sin\left(\frac{\alpha}{2}\right)$$



Để tạo được hình tròn, chúng ta chỉ cần tăng số **n** trong thủ tục trên lên 1 số khá lớn, ví dụ 90, 180, 360. Tuy nhiên độ dài cạnh **d** cần tính lại chính xác. Cạnh **d** có thể tính qua bán kính **R** theo công thức đã ghi ở trên: $d = 2R\sin\left(\frac{\alpha}{2}\right)$. Ví dụ với $n = 90$, khi đó $\alpha = 360/90 = 4$. Do đó theo công thức trên ta sẽ có $d = 2R\sin(2)$.



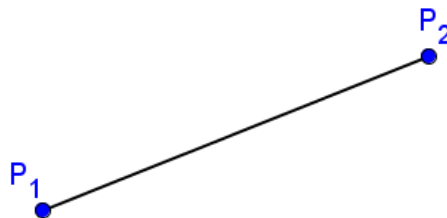
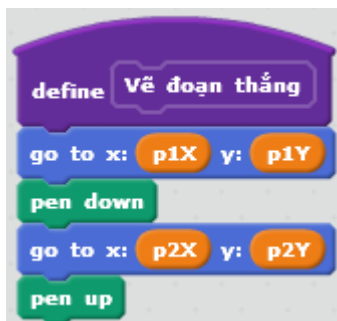
Tính độ dài cạnh $d = 2R\sin(\frac{\alpha}{2})$ với $\alpha = 4$

Tọa độ điểm bắt đầu vẽ = $(Tx, Ty - R)$

Đoạn chương trình vẽ đường tròn.

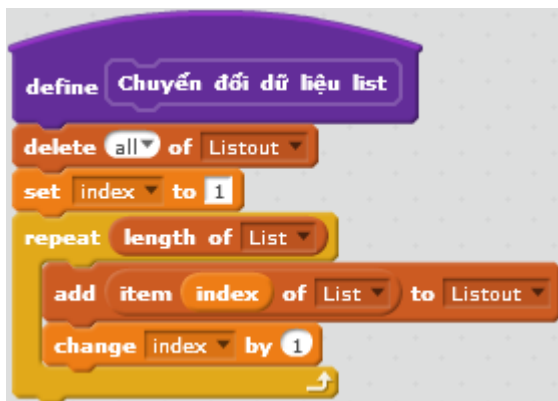
3. Thủ tục vẽ đường thẳng đi qua 2 điểm A và B cho trước

Giả sử điểm A có tọa độ (P_1X, P_1Y) , điểm B có tọa độ (P_2X, P_2Y) , khi đó thủ tục vẽ đoạn thẳng nối 2 điểm này như sau:



3. Thủ tục gán dữ liệu từ 1 biến nhớ danh sách này sang biến nhớ danh sách khác

Giả sử cho trước 1 biến nhớ kiểu danh sách **List**, cần xây dựng thủ tục chuyển dữ liệu từ **List** sang **Listout**.



Thuật toán của thủ tục này đơn giản như sau:

- Xóa nội dung của dãy Listout.
- Chuyển đổi lần lượt các phần tử của dãy **List** sang **Listout**.

4. Thủ tục sinh ngẫu nhiên 1 hoán vị các số 1, 2, 3, ..., n

Cho trước số tự nhiên **n**, thủ tục cần xây dựng sẽ sinh ngẫu nhiên 1 hoán vị của dãy số 1, 2, 3, ..., n và kết quả đưa vào biến nhớ **List**.

Để thực hiện thủ tục này, chúng ta cần tạo ra thêm 1 dãy phụ nữa, ví dụ List_temp. Đoạn chương trình của thủ tục như sau.



Dãy các số từ 1 đến n đã được đưa vào dãy `List_temp` từ trước đó.

Thuật toán của thủ tục: chọn ngẫu nhiên 1 số từ `List_temp`, đưa số này vào `List`, sau đó xóa số này từ `List_temp` và lặp lại quá trình trên cho đến khi dãy

Dưới đây là đoạn chương trình hoàn chỉnh để sử dụng thủ tục trên.



Các lệnh chuẩn bị ban đầu được đưa vào 1 thủ tục **Init**, khởi tạo 2 dãy **List**, **List_temp** và đưa dãy 1.. n

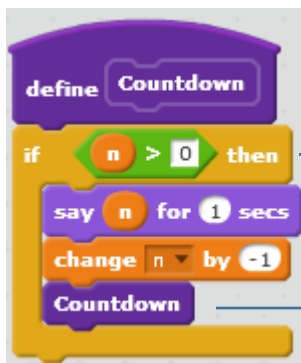
Đoạn chương trình

5. Thủ tục gọi thủ tục khác và gọi chính nó

Như chúng ta đã thấy, thủ tục sau khi được khởi tạo sẽ hiện ra trong khung các mẫu lệnh và được sử dụng như một lệnh bình thường. Do vậy hoàn toàn có thể xảy ra là trong nội dung của 1 thủ tục này có lời gọi đến thủ tục khác, hoặc một thủ tục sẽ có lời gọi đến chính nó. Trong hoạt động này chúng ta sẽ làm quen với các hiện tượng này.

a. Thủ tục có thể gọi chính nó

Khi trong thủ tục có lời gọi đến chính nó, điểm cần đặc biệt chú ý nhất là phải có điều kiện đảm bảo không để thủ tục chạy vô hạn. Trong ví dụ sau, thủ tục Countdown có nhiệm vụ hiển thị giá trị biến nhớ n và đếm ngược cho đến 0. Quá trình này cần phải kết thúc với điều kiện $n > 0$.

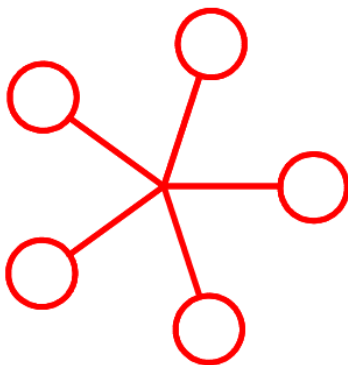


Điều kiện để dừng thủ

Lời gọi chính

b. Thủ tục gọi thủ tục

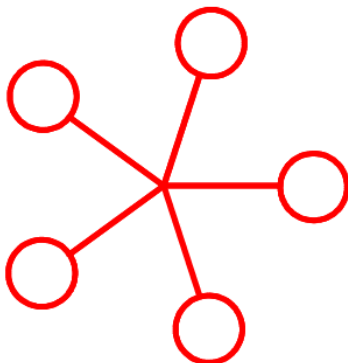
Chúng ta quay lại 1 ví dụ đã biết trong các bài trước, vẽ bông hoa 5 cánh như hình sau.



Chúng ta nhớ lại qui trình vẽ bông hoa này như sau:

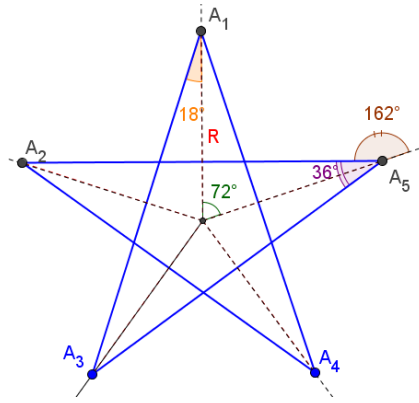
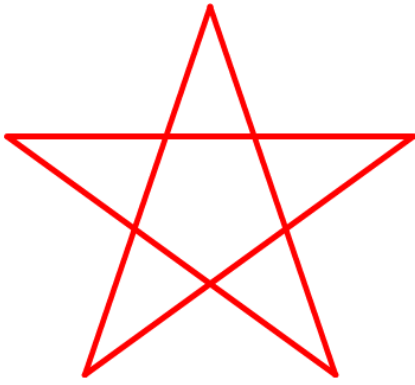
- Ở vòng lặp ngoài sẽ lặp 5 lần và vẽ 5 đoạn thẳng từ tâm đến vị trí tiếp cận vòng tròn.
- Ở vòng lặp trong là vẽ vòng tròn

Dựa trên chương trình gốc này, chúng ta sẽ tạo 2 thủ tục: **Vẽ hoa** ở vòng lặp ngoài và **Vẽ vòng tròn nhỏ** ở vòng lặp trong. Thủ tục **Vẽ hoa** sẽ gọi **Vẽ vòng tròn nhỏ** bên trong các lệnh của mình. Nội dung cụ thể 2 thủ tục này trong hình dưới đây.



5. Chương trình vẽ ngôi sao 5 cánh

Trong hoạt động này chúng ta sẽ thiết lập chương trình vẽ ngôi sao 5 cánh như hình dưới đây với bán kính **R** cho trước. Để thuận tiện cho việc phân tích bài toán, chúng ta cần vẽ thêm hình 1 ngôi sao 5 cánh ở bên với các tham số mô tả kỹ hơn cấu trúc của ngôi sao này.



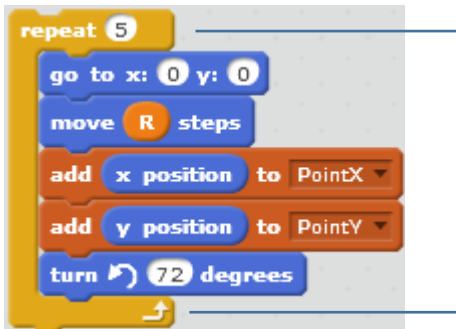
Phân tích bài toán.

Quan sát hình trên bên phải cho chúng ta gợi ý cách thực hiện yêu cầu bài toán. Thiết lập 2 bảng, biến nhớ dạng List để lưu trữ tọa độ các điểm A_1, A_2, A_3, A_4, A_5 . Các biến mảng này sẽ ký hiệu là **PointX** và **PointY**. Bảng **PointX** sẽ lưu trữ các tọa độ X, mảng **PointY** sẽ lưu trữ các tọa độ y.

Như vậy các bước để giải quyết bài toán này như sau:

1. Thiết lập tọa độ các điểm A_1, A_2, A_3, A_4, A_5 và đưa vào 2 mảng **PointX**, **PointY**.
2. Vẽ hình ngôi sao bằng cách nối vẽ các điểm $A_1-A_3, A_2-A_4, A_3-A_5, A_4-A_1, A_5-A_2$.

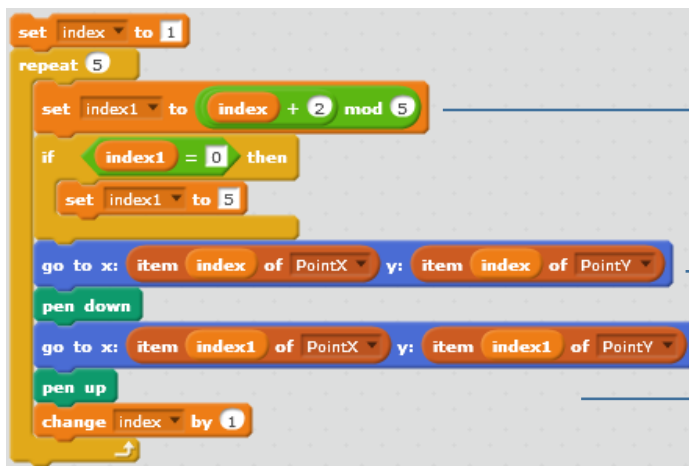
Đoạn chương trình sau mô tả cách thiết lập tọa độ dãy điểm A_1, A_2, A_3, A_4, A_5 . Các tọa độ này lần lượt được bổ sung vào 2 danh sách **PointX**, **PointY**.



Vòng lặp 5 lần, mỗi lần thực hiện:

- Bắt đầu từ vị trí (0, 0), dịch chuyển 1 đoạn bằng R, nạp thông tin tọa độ X, Y vào 2 bảng PointX, PointY.

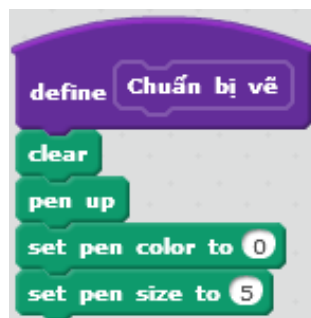
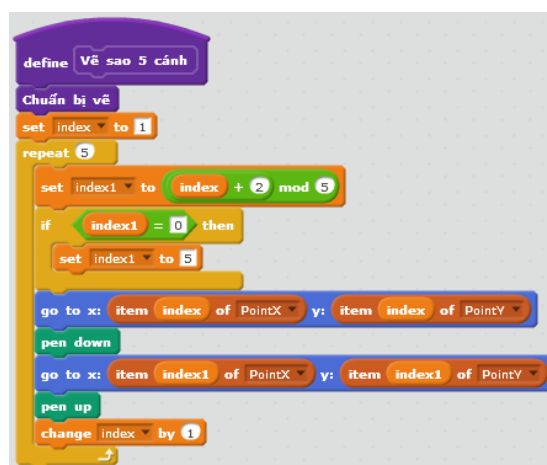
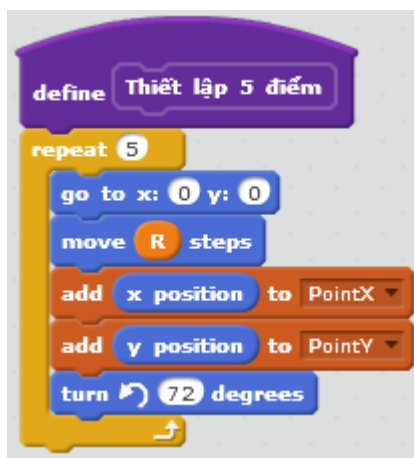
Đoạn chương trình sau thực hiện công việc vẽ các nét nối các cặp điểm $A_1-A_3, A_2-A_4, A_3-A_5, A_4-A_1, A_5-A_2$. Việc này cũng được thực hiện trong 1 vòng lặp 5 lần.



Xác định các chỉ số **index** và **index1**

Vẽ đoạn thẳng nối điểm từ điểm có chỉ số **index** đến điểm có chỉ số **index1**

Toàn bộ các thủ tục của chương trình như sau:



Câu hỏi và bài tập

1. Cho trước dãy số **List**, hãy viết 1 thủ tục sinh ngẫu nhiên 1 hoán vị của dãy này, kết quả sẽ đưa vào mảng **Listout**.
2. Cho trước 1 xâu ký tự **Str**. Viết 1 thủ tục sinh ngẫu nhiên 1 hoán vị của xâu này, kết quả đưa vào xâu **Strout**.
3. Hoàn thiện chương trình Vẽ hoa.

4. Hãy thực hiện bài toán vẽ hình ngôi sao theo cách khác, ví dụ: tính độ dài cạnh ngôi sao (khoảng cách $d = A_1A_3$, thuật toán vẽ chính sẽ như sau: xuất phát từ vị trí A_1 , thực hiện lặp 5 lần thao tác sau: xoay trái 36 độ, di chuyển theo độ dài d).

Mở rộng

Viết chương trình thực hiện bài luyện theo yêu cầu sau.

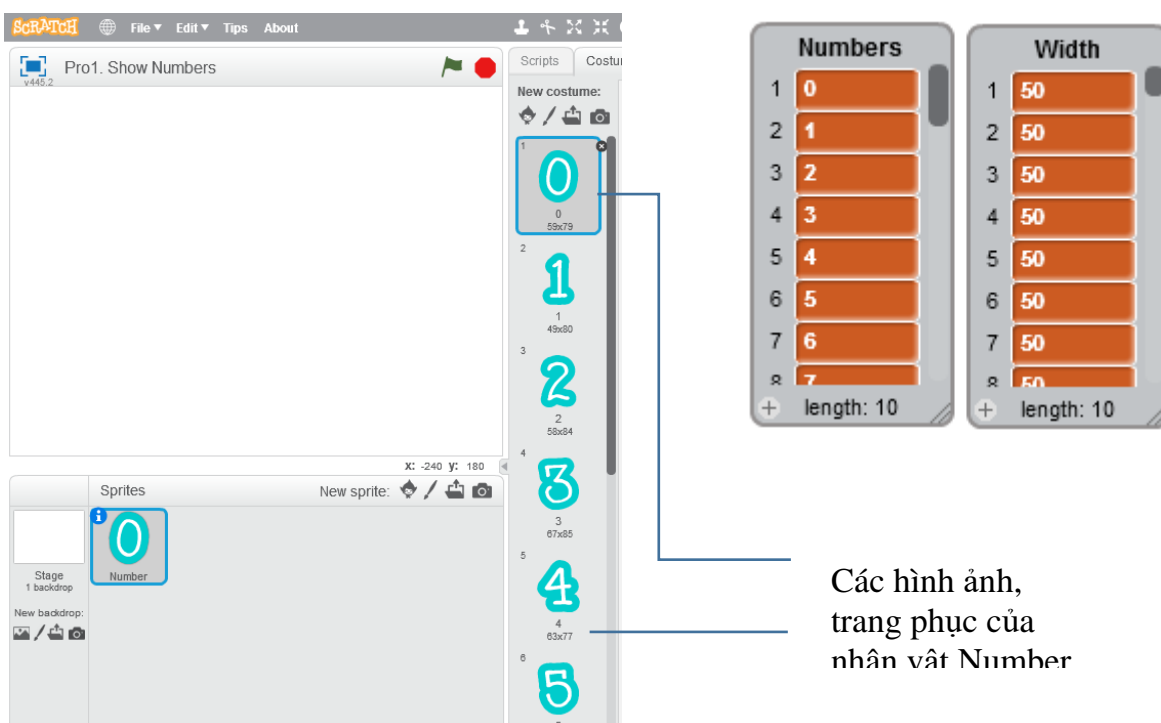
78890

Chương trình sẽ yêu cầu người dùng nhập 1 số tự nhiên từ bàn phím, sau đó thể hiện số này trên màn hình bằng các chữ số lớn, rõ

Em hãy nhập 1 số tự nhiên < 1000000

Ý nghĩa của chương trình này rất lớn và có nhiều ứng dụng trong các chương trình khác nhau, vì các giá trị số của biến nhớ được thể hiện trên màn hình rất khó, khó quan sát.

Để thực hiện bài toán này chúng ta tạo ra 1 nhân vật có tên Number có đầy đủ toàn bộ 10 trang phục chính là các chữ số 0, 1, 2, 3, 4, 5, 6, 7, 8, 9. Các chữ số này và độ rộng của chúng trên màn hình được lưu trong 2 dãy **Numbers** và **Width** như hình dưới đây.

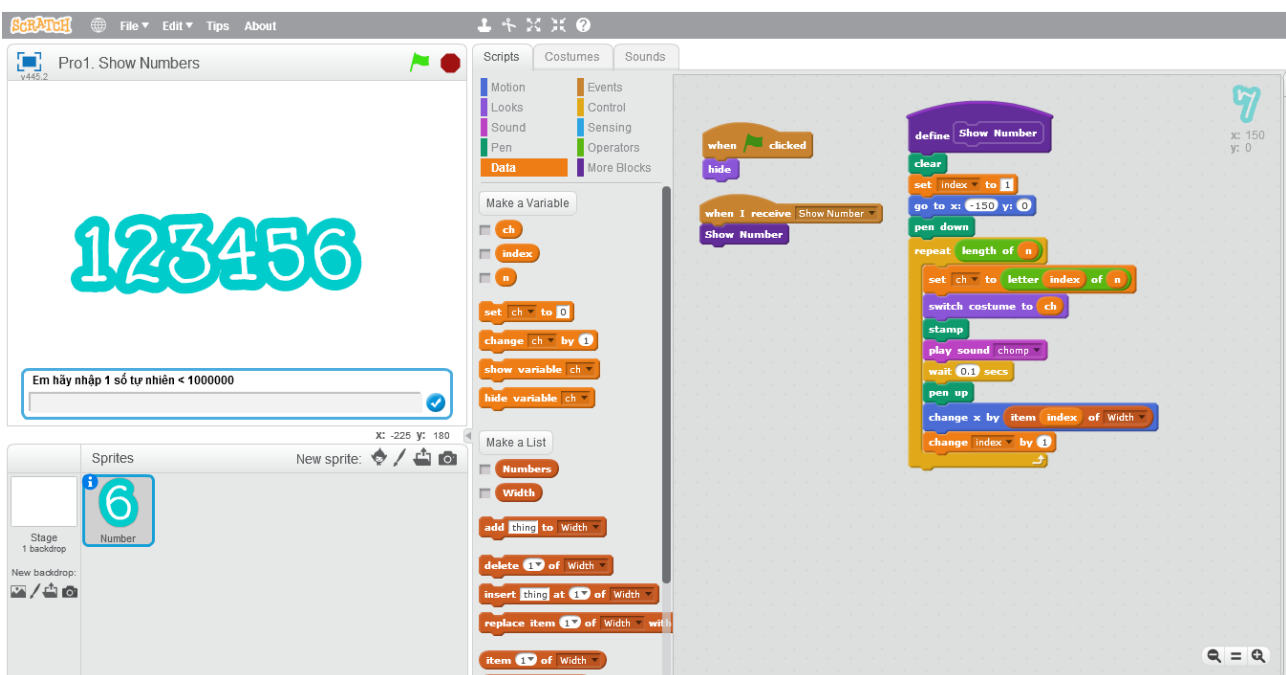


Các hình ảnh, trang phục của nhân vật Number

Biến nhớ **n** dùng để lưu số do người dùng nhập. Thủ tục sau có nhiệm vụ thể hiện số **n** trên màn hình bằng nhân vật **Number**.



Yêu cầu nhập số n được tiến hành bởi sân khấu. Sau khi nhập xong dữ liệu, thông điệp **Show Number** được gửi đi và khi nhận thông điệp này, thủ tục **Show Number** sẽ được kích hoạt để thể hiện số đã nhập trên màn hình.



Bài 19. Thủ tục 2

Mục đích

Học xong bài này, bạn sẽ biết:

- Thiết lập thủ tục có tham số.
- Phân biệt biến nhớ riêng của thủ tục và biến nhớ của nhân vật.
- Ứng dụng tiếp theo của thủ tục.

Bắt đầu

Trong bài học trước, em đã được làm quen với khái niệm thủ tục. Thủ tục sau khi được định nghĩa sẽ hiện ra trong khung điều khiển lệnh như 1 lệnh mẫu và em có thể sử dụng lệnh này như mọi lệnh khác. Ví dụ thủ tục **Vẽ đa giác đều** sau.



Chúng ta cùng nhau xem lại nội dung của thủ tục này, em có nhận xét gì khi quan sát nội dung này?

Em hãy quan sát các biến nhớ **n** và **d**.



Các biến nhớ **n** và **d** có màu cam sẫm là biến nhớ của nhân vật đang thực hiện thủ tục này.

Như vậy các biến nhớ **n**, **d** được tạo ra bởi nhân vật đang thực hiện thủ tục này và không phụ thuộc vào thủ tục. Do vậy khi gọi thủ tục này, các biến nhớ **n**, **d** cần được gán dữ liệu trước. Ví dụ:



Trong Scratch, em đã biết có rất nhiều lệnh có tham số, tức là chúng ta cho thể nhập trực tiếp dữ liệu trên dòng định nghĩa của lệnh này.

Ví dụ nếu như chúng ta có thủ tục vẽ 1 đa giác đều, trong đó tham số số cạnh và độ dài cạnh có thể nhập trực tiếp như lệnh sau thì việc áp dụng thủ tục này trên thực tế sẽ thuận tiện và đơn giản hơn rất nhiều.



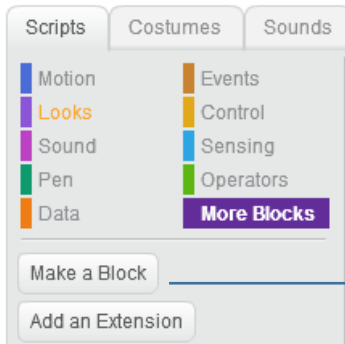
Trong bài học này chúng ta sẽ làm quen với loại thủ tục có tham số này.

Nội dung bài học

1. Thiết lập tham số cho thủ tục

Trong hoạt động này chúng ta sẽ cùng nhau thiết lập các thủ tục có tham số trong Scratch, bắt đầu từ bài toán vẽ đa giác đều trên.

(a) Để thiết lập thủ tục chúng ta chọn nhóm lệnh **More Blocks** và sau đó nhấp lên nút **Make a Block**.



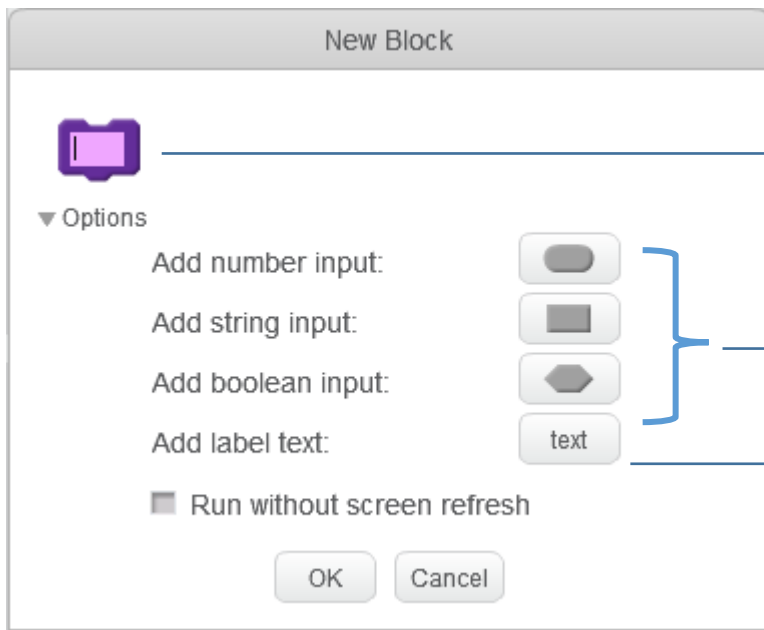
Nháy nút này để bắt đầu khởi tạo 1 thủ tục mới.

(b) Xuất hiện hộp hội thoại New Block như hình dưới đây. Nháy lên nút Options để mở rộng hộp hội thoại.



Nháy lên nút Options để mở rộng hộp hội thoại này đầy đủ sẵn sàng cho việc khởi tạo thủ tục có

(c) Hộp hội thoại tạo thủ tục mở rộng như sau.



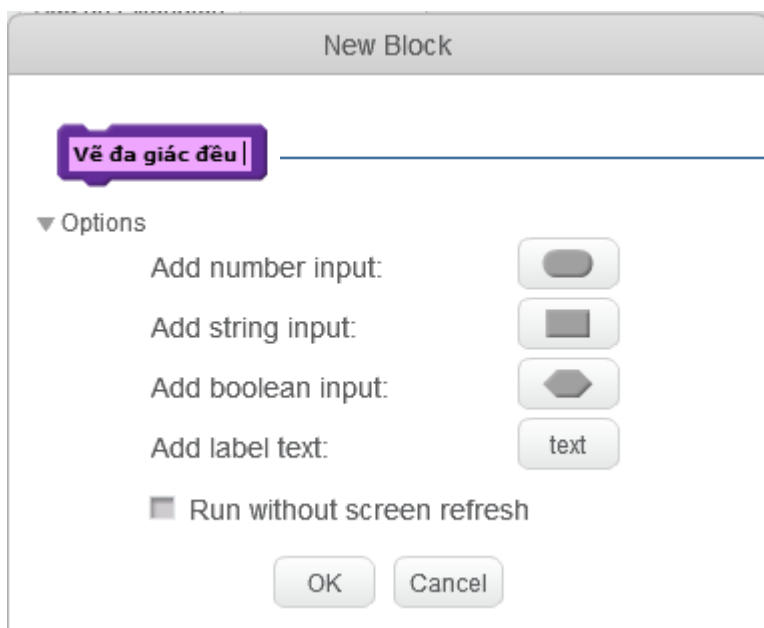
Nhập tên thủ tục và các tham số của thủ tục tại đây.

Các nút này dùng để đưa tham số vào thủ tục. Cho phép 3 loại tham số: số, xâu

Chèn giữa các tham biến là chữ được chèn bằng nút này.

Chú ý ý nghĩa các nút chèn tham số và chữ vào tên của thủ tục.

(d) Đầu tiên cần nhập tên của thủ tục trước. Ví dụ nhập tên **"Vẽ đa giác đều"**.



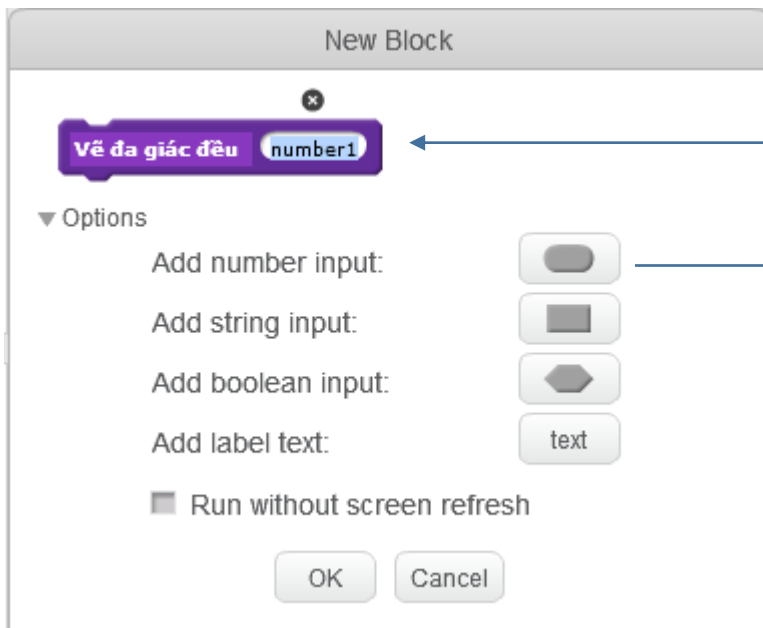
Nhập tên của thủ tục tại đây.

(e) Sau khi nhập tên chính của thủ tục thì bắt đầu có thể nhập tham số.

Trong ví dụ dưới đây, sau khi nhập tên thủ tục là Vẽ đa giác đều, em sẽ nhấn nút để tạo 1 tham số dạng số (number) đầu tiên.

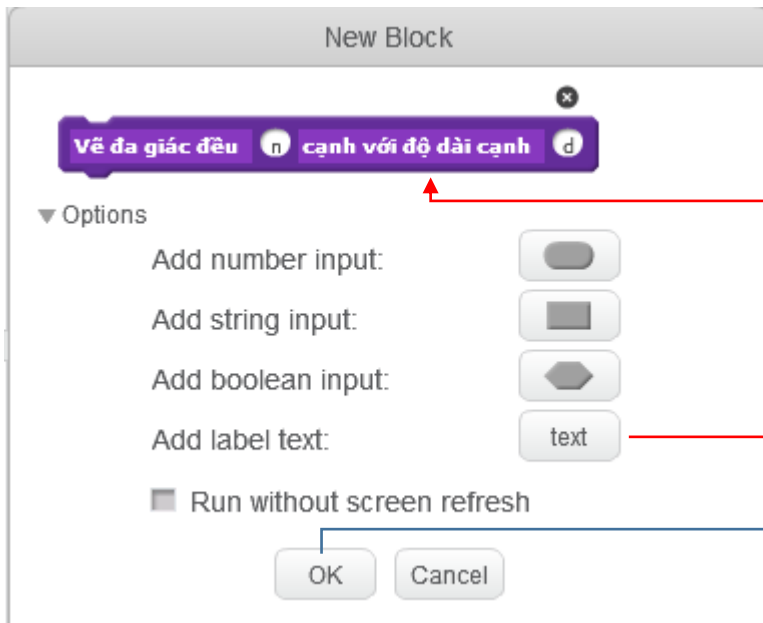


Xuất hiện khung tham số với tên **"number1"**. Em hãy thay đổi thành **"n"**.



Nháy nút này để tạo ra tham số đầu tiên dạng số của thủ tục. Tên của tham số sẽ mặc định là **"number1"**, em hãy đổi thành **"n"**.

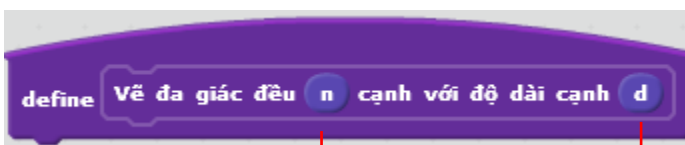
(f) Tiếp theo, em hãy chèn 1 đoạn chữ vào thủ tục ("**cạnh với độ dài cạnh**"), sau cùng bổ sung thêm 1 tham biến dạng số nữa, đặt tên là **"d"** như hình sau.



Nháy nút này để tạo ra thêm dòng chữ "**cạnh với độ dài cạnh**" trong tên

Nháy OK để kết thúc khởi tạo tên của thủ tục.

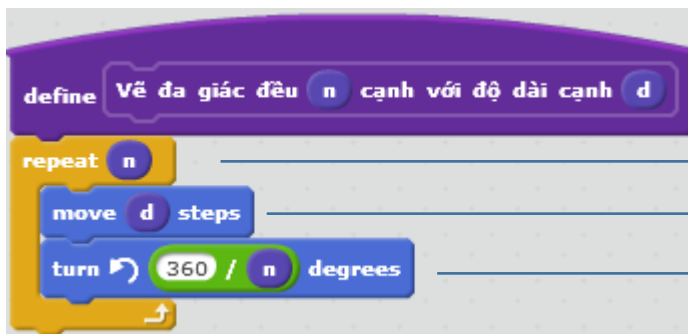
(g) Sau khi tạo xong tên của thủ tục, em sẽ thấy xuất hiện lệnh **define**, biểu tượng sau trên cửa sổ lệnh của nhân vật. Em hãy chú ý quan sát các tham biến đã được định nghĩa trong thủ tục.



Các tham biến trong dòng định nghĩa thủ tục có màu xanh.

Các tham biến được định nghĩa trong thủ tục sẽ có màu xanh. Trong quá trình viết các lệnh mô tả thủ tục chúng ta có thể sử dụng các tham biến này như các biến nhớ bình thường, ngoại trừ các lệnh thao tác trực tiếp thay đổi giá trị của các biến nhớ này.

(h) Bây giờ chúng ta điền nội dung của thủ tục bằng các lệnh đã biết. Chú ý sử dụng các tham biến của thủ tục như các biến nhớ thông thường.

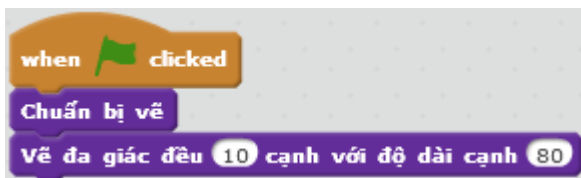


Trong nội dung của thủ tục cần sử dụng các tham biến như các biến nhớ bình

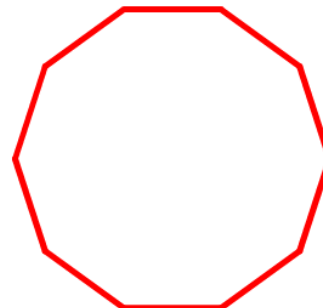
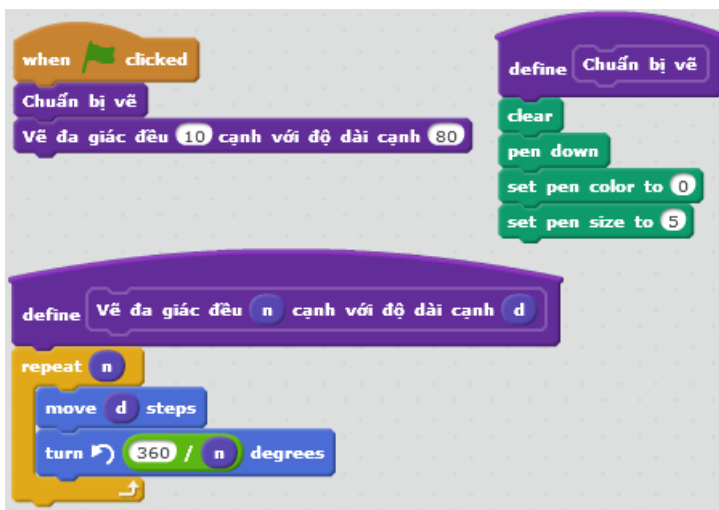
(i) Sau khi thủ tục được định nghĩa xong, tên của thủ tục sẽ xuất hiện tại khung mẫu lệnh như 1 lệnh có tham số bình thường của Scratch.



(j) Bây giờ chương trình chính của bài này sẽ được viết đơn giản bằng 2 lệnh như sau.



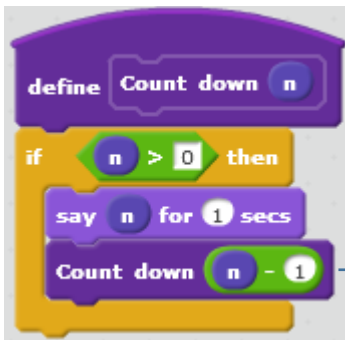
Toàn bộ cửa sổ lệnh và kết quả thực hiện chương trình như sau.



2. Thủ tục đếm ngược Count Down

Thủ tục đếm ngược Count Down từ bài trước bây giờ có thể viết dưới dạng có tham số như sau.

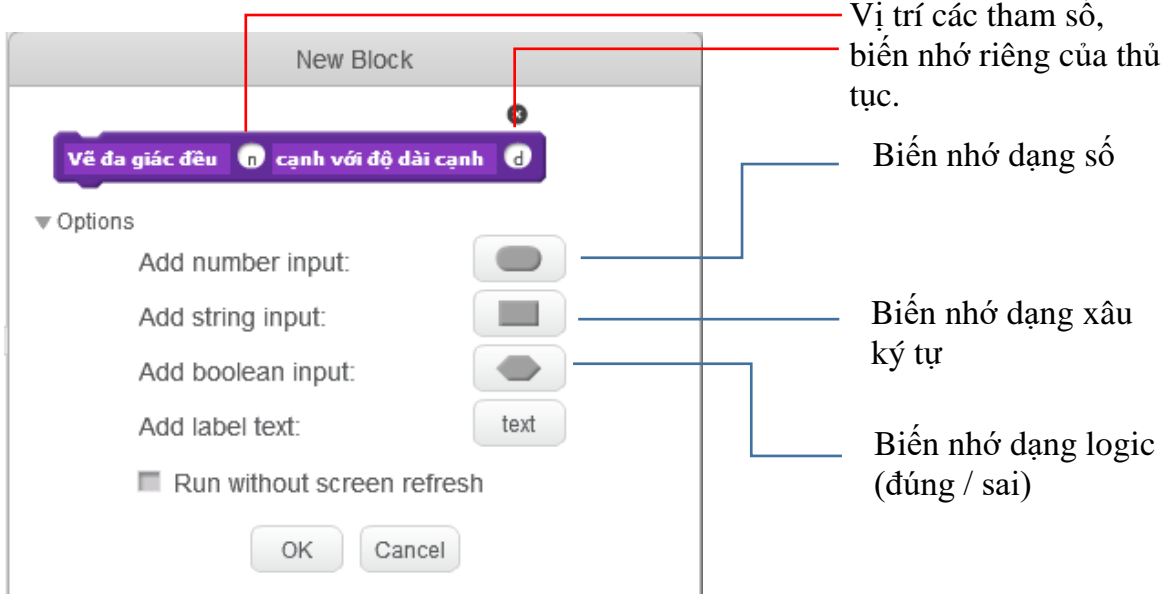
Chú ý lời gọi chính nó sẽ có tham số thay đổi.



Lời gọi chính thủ tục gốc
Count down với giá trị tham
số $n - 1$.

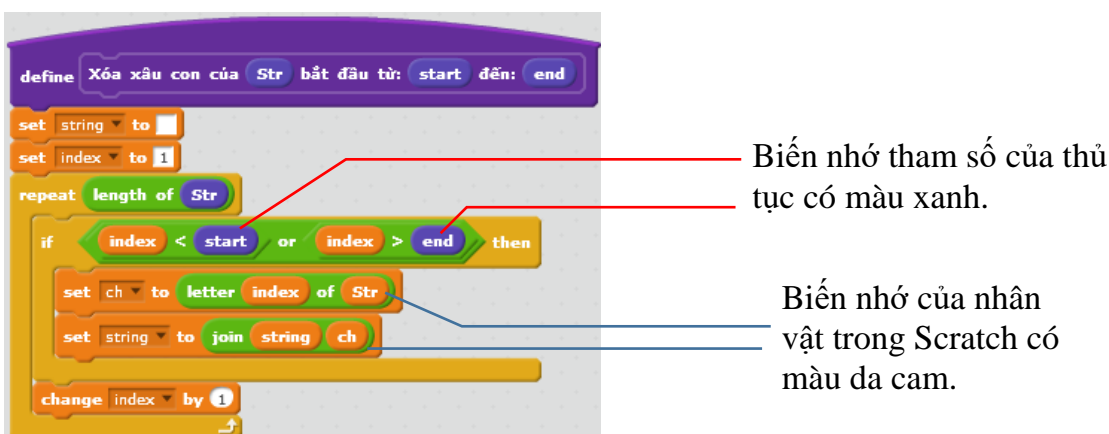
3. Biến nhớ của thủ tục

Các tham biến được định nghĩa trong thủ tục chính là các biến nhớ riêng của thủ tục này. Chúng ta có thể tạo ra nhiều biến nhớ như vậy trong quá trình định nghĩa thủ tục mới. Trong hoạt động này chúng ta cùng phân tích sâu hơn về các tham biến hay biến nhớ riêng này của thủ tục trong Scratch.



Các tham biến được định nghĩa trong thủ tục có thể hiểu như các biến nhớ riêng của thủ tục này. Vậy các biến nhớ này có đặc điểm gì khác biệt với khái niệm biến nhớ thông thường mà ta đã biết.

(1) Tất cả các tham biến, hay biến nhớ của thủ tục đều có màu xanh để phân biệt với biến nhớ của nhân vật trong Scratch có màu da cam. Do vậy 2 hệ thống biến nhớ này là độc lập hoàn toàn với nhau.



(2) Các giá trị được truyền vào tham số của thủ tục có thể là giá trị số cụ thể, có thể là biến nhớ.

Trong ví dụ sau cả 3 tham số của thủ tục đều được truyền giá trị thông qua các biến nhớ.



Trong ví dụ trên, giá trị của biến nhớ nhân vật **Str** được truyền vào thủ tục (tại thời điểm gọi thủ tục này) thông qua tham biến **Str**. Chú ý biến **Str** màu da cam khác biệt hoàn toàn với tham biến **Str** màu xanh.

Ví dụ: lệnh sau đây nếu được thực hiện trong thủ tục trên sẽ gán giá trị của tham biến **Str** của thủ tục cho biến nhớ **Str**.



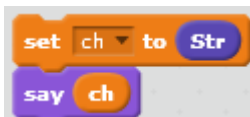
Còn trong ví dụ sau, sau khi gán trực tiếp biến nhớ Str thành "Hà Nội" thì giá trị tham biến Str của thủ tục vẫn không thay đổi.



(3) Các tham biến, biến nhớ riêng của thủ tục chỉ có ý nghĩa bên trong thủ tục này.

Các biến nhớ màu xanh chỉ có tác dụng nếu việc sử dụng chúng được đặt trong các lệnh bên trong thủ tục này.

Ví dụ: 2 lệnh sau nếu đặt ngay trong chương trình chính sẽ không có ý nghĩa vì biến nhớ màu xanh Str không có tác dụng.



(4) Không thể thay đổi giá trị của các tham biến hay biến nhớ của thủ tục.

Các tham biến hay biến nhớ bên trong thủ tục không thể thực hiện được các lệnh thay đổi giá trị như xóa, gán hoặc thay đổi thông tin. Khi thực hiện lời gọi thủ tục, các tham biến này được truyền 1 giá trị từ bên ngoài và giữ nguyên giá trị đó trong suốt thời gian thủ tục có hiệu lực.

4. Một số thủ tục với xâu ký tự

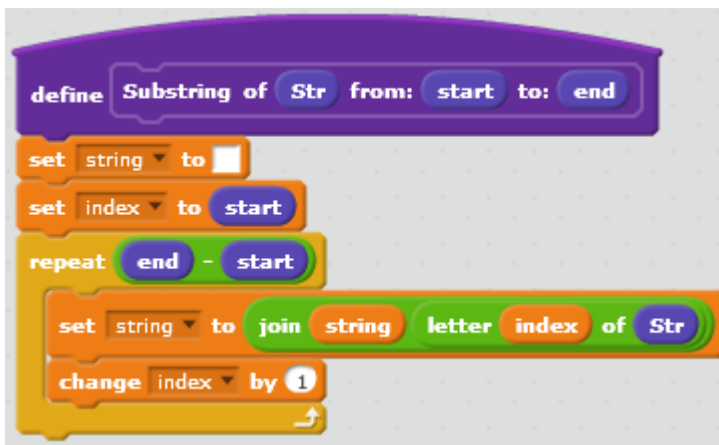
4.1. Thủ tục lấy ra 1 xâu con từ xâu gốc

Thủ tục này có các tham biến đầu vào:

Str: xâu gốc ban đầu.

start, end: vị trí các ký tự bắt đầu và kết thúc cần lấy ra xâu con từ xâu gốc **Str**.

Kết quả xâu con được gán cho biến nhớ **string**.



4.2. Thủ tục chèn 1 xâu vào một xâu khác

Các tham số của thủ tục:

Str1: xâu, nội dung cần chèn.

Str: xâu ký tự gốc ban đầu.

start: vị trí cần chèn trong xâu gốc **Str**.

Kết quả xâu gốc **Str** sau khi đã chèn **Str1** vào vị trí **start** sẽ được gán cho biến nhớ **string**.



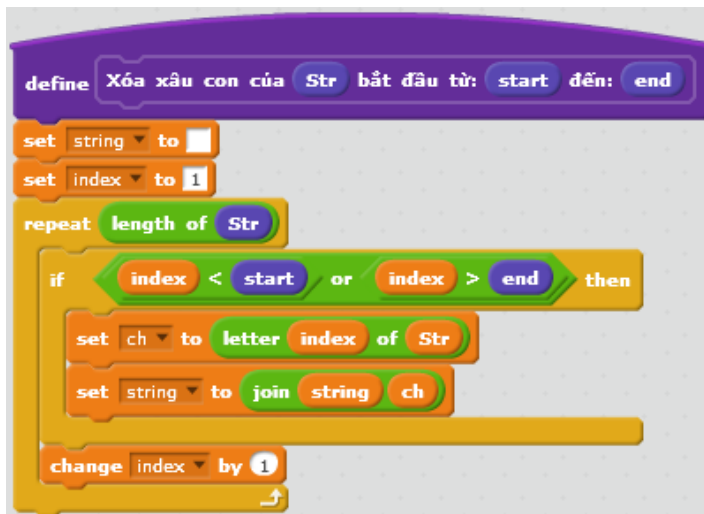
4.3. Thủ tục xóa 1 xâu con trong 1 xâu ký tự

Các tham biến của thủ tục này:

Str: chuỗi ký tự gốc.

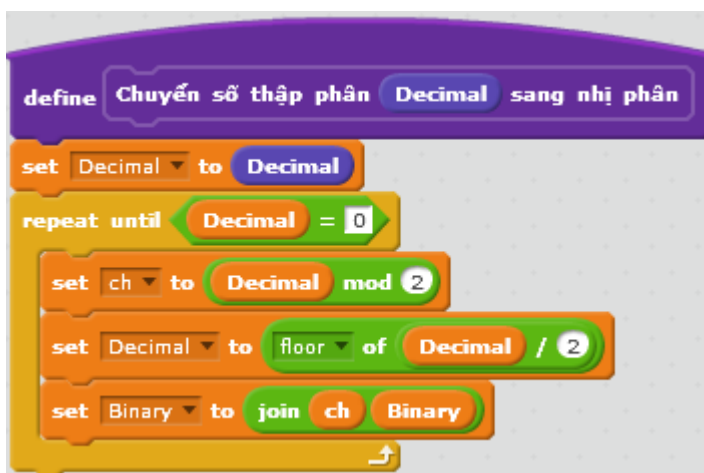
start, end: vị trí đầu và cuối trong chuỗi **Str** cần xóa.

Kết quả của việc xóa các ký tự trong chuỗi **Str** sẽ gán vào chuỗi **string**.



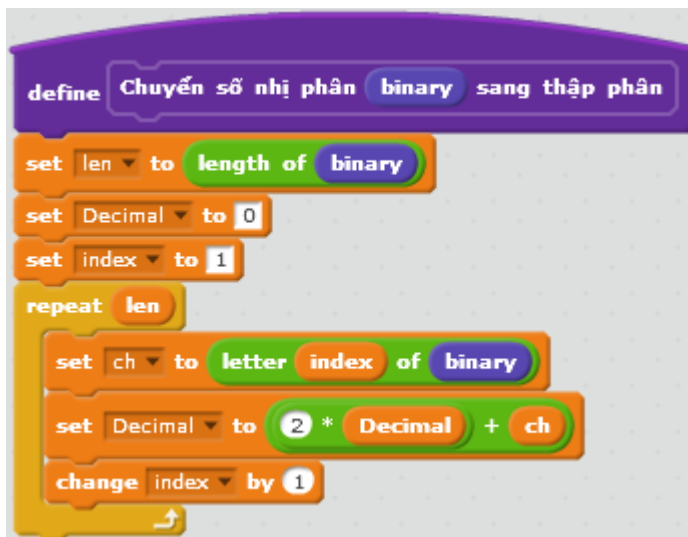
4.4. Thủ tục chuyển số thập phân sang nhị phân

Thủ tục này có 1 tham biến là **Decimal**. Kết quả việc chuyển đổi sang chuỗi nhị phân được lưu trong biến nhớ **Binary**.



4.5. Thủ tục chuyển số nhị phân sang thập phân

Thủ tục này có 1 tham biến là **Binary**. Kết quả chuyển đổi chuỗi nhị phân này sang số thập phân được lưu trong biến nhớ **Decimal**.



5. Một số thủ tục với số

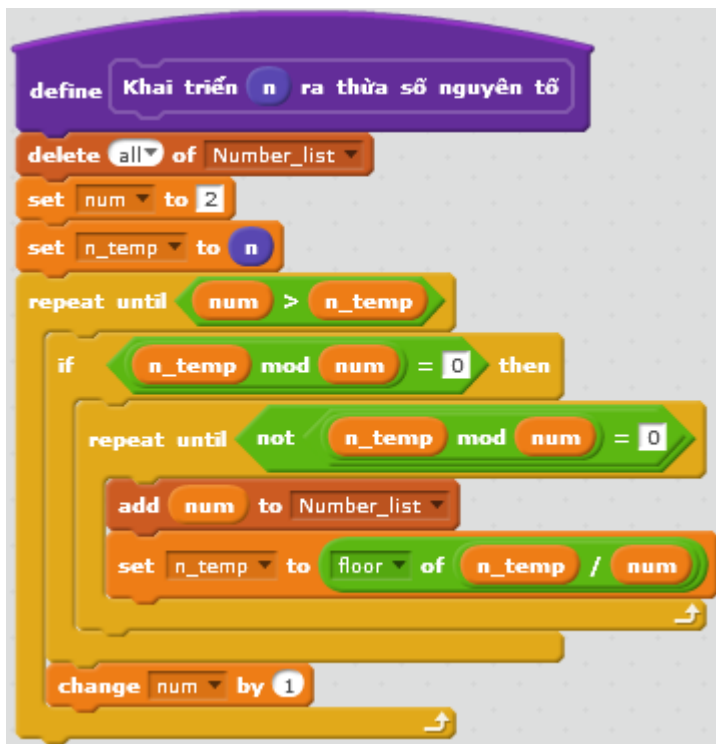
5.1. Thủ tục kiểm tra 1 số có phải là nguyên tố hay không.

Thủ tục này có 1 tham số duy nhất là số tự nhiên **n**. Kết quả kiểm tra ghi trong biến nhớ **nguyento**. Nếu **nguyento = 1** thì **n** là số nguyên tố, ngược lại nếu **nguyento = 0** thì **n** là hợp số.



5.2. Thủ tục khai triển 1 số tự nhiên thành tích các thừa số nguyên tố

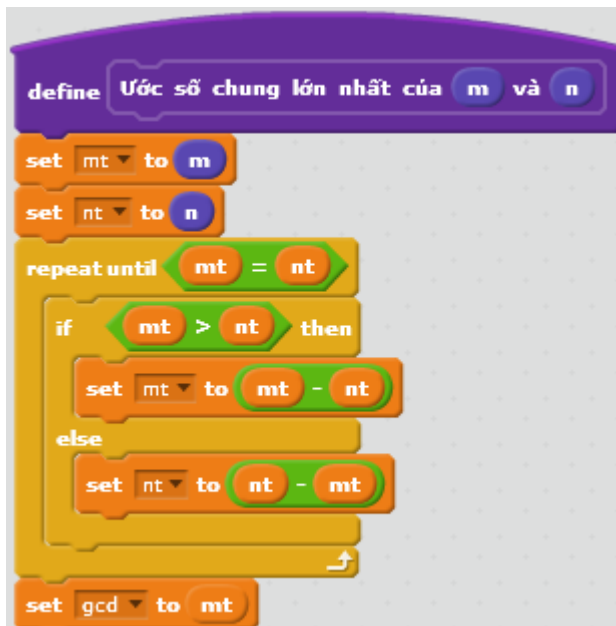
Thủ tục này chỉ có 1 tham số duy nhất là **n**. Kết quả khai triển **n** thành tích các thừa số nguyên tố được lưu trong dãy **Number_list**.



5.3. Thủ tục tính USCLN và BSCNN của 2 số tự nhiên

Cả 2 thủ tục này đều có các tham biến là các số **m** và **n**.

Kết quả ước số chung lớn nhất của 2 số trên sẽ được lưu trong biến nhớ **gcd**.



Kết quả của thủ tục tính bội số chung nhỏ nhất của 2 số **m** và **n** được lưu trong biến nhớ **lcm**.

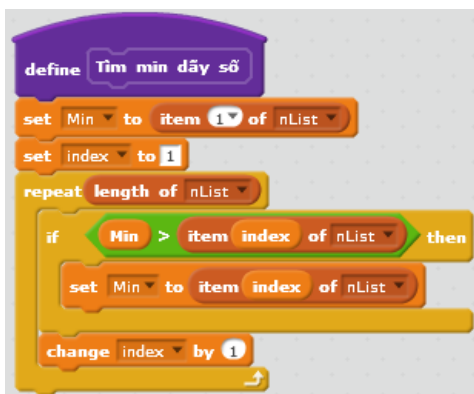


6. Một số bài toán xử lý liên quan đến dãy số

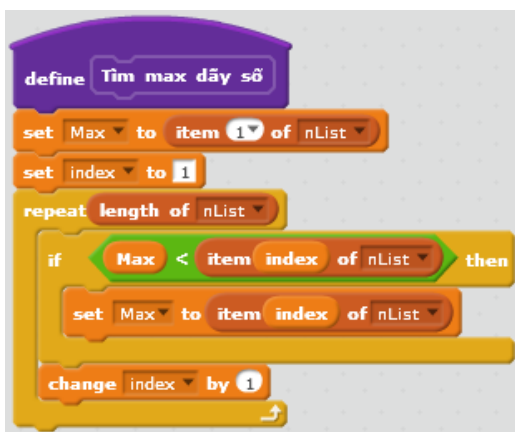
Trong hoạt động này chúng ta sẽ cùng nhau giải quyết một số bài toán cổ điển liên quan đến dãy số. Dãy số cho trước sẽ được ký hiệu là lưu trữ trong biến nhớ dạng danh sách **nList**.

6.1. Bài toán tìm phân tử nhỏ nhất và lớn nhất của 1 dãy số cho trước

Giá trị nhỏ nhất của dãy số **nList** được lưu trong biến nhớ **Min**.



Giá trị lớn nhất của dãy số **nList** được lưu trong biến nhớ **Max**.



6.2. Bài toán tính số các khoảng đơn điệu của 1 dãy số cho trước

Khoảng đơn điệu của 1 dãy số là 1 dãy con liên tục đơn điệu (tăng hoặc giảm thực sự). Để hiểu rõ hơn định nghĩa này, chúng ta quan sát 1 số ví dụ sau:

Dãy số gốc	Phân tích	Số các khoảng đơn điệu
1 1 1 1 1 1 1	dãy này chỉ bao gồm các số hạng bằng nhau nên không có bất kỳ khoảng đơn điệu nào.	0
1 2 3 3 3 -2 -3	khoảng đơn điệu tăng đầu tiên (1 2 3), tiếp theo là 1 khoảng đi ngang, sự đi ngang không được tính là khoảng đơn điệu. Do vậy dãy này chỉ có 2 khoảng đơn điệu.	2
1 1 2 3 4 4 3 2 1 1 1 2	khoảng đi ngang đầu tiên (1 1) không được tính là khoảng đơn điệu. Dãy này chỉ có 3 khoảng đơn điệu.	3

Chúng ta sử dụng các biến nhớ sau:

nList: dãy số đã cho ban đầu.

count: biến đếm các khoảng đơn điệu tăng thực sự của dãy số. Đây chính là đáp án cần tìm.

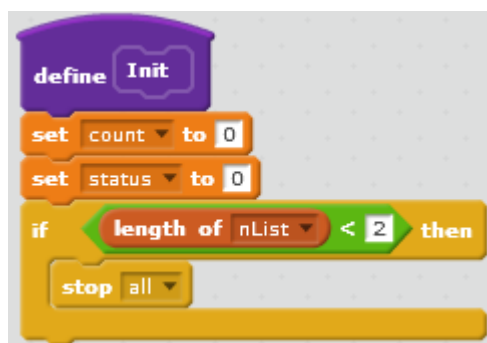
status: biến nhớ ghi lại trạng thái của phần tử hiện thời của dãy khi chúng ta đang phân tích. Ý nghĩa của biến này như sau:

status = 0, phần tử này chưa xét hoặc không cần xét.

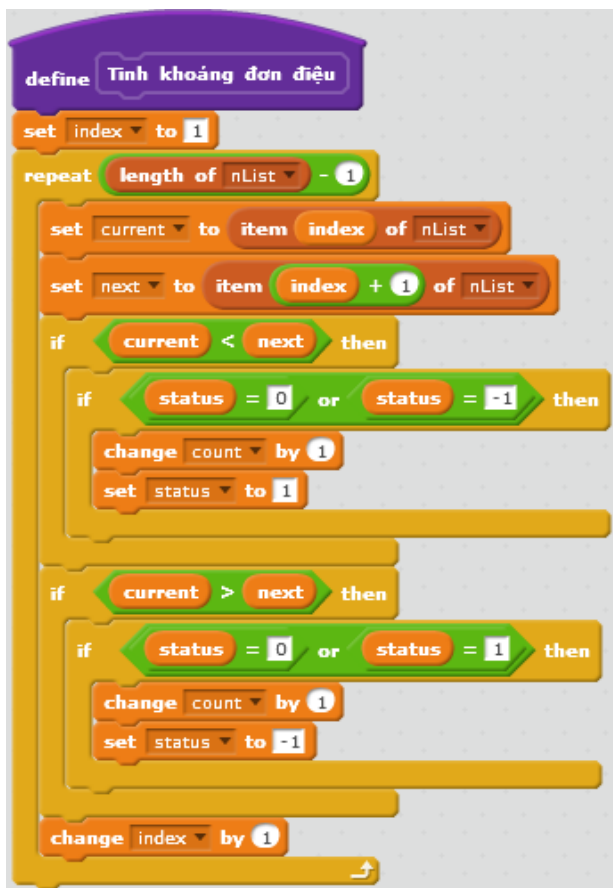
status = 1, phần tử hiện thời đang nằm trong 1 khoảng đơn điệu tăng thực sự.

status = -1, phần tử hiện thời không nằm trong 1 khoảng đơn điệu tăng thực sự.

Thủ tục **Init** sẽ đặt các giá trị ban đầu cho biến count và status. count = 0. status = 0. Kiểm tra dãy nList nếu có > 1 phần tử thì mới tiếp tục làm việc.



Thủ tục chính của bài toán.



6.3. Bài toán tìm 1 dãy con liên tục đơn điệu tăng có độ dài lớn nhất của một dãy số cho trước

Phân tích bài toán.

Gọi dãy đã cho là a_1, a_2, \dots, a_N . Vì dãy cần tìm là dãy liên tục nên chúng ta có thể đánh dấu và tìm được tất cả các dãy con liên tục đơn điệu tăng chỉ cần 1 lần duyệt từ đầu dãy. Nhận xét trên cho ta ý tưởng làm bài toán này chỉ cần 1 vòng duyệt tuyến tính từ 1 đến N .

Trong quá trình duyệt chúng ta sẽ sử dụng các biến nhớ sau:

Thông tin về dãy cần tìm được lưu trong các biến sau:

bmax – chỉ số phần tử đầu tiên của dãy con đơn điệu tăng cực đại.

lengthmax – độ dài của dãy con đơn điệu tăng cực đại.

Như vậy dãy này sẽ có các chỉ số là $bmax, bmax+1, \dots, bmax+lengthmax-1$

Chúng ta cần lưu thông tin của dãy con liên tục đơn điệu tăng hiện thời, tức là dãy mà chúng ta đang khảo sát:

bcurr – chỉ số phần tử đầu tiên của dãy

lengthcurr – độ dài hiện thời của dãy.

Phân khai báo ban đầu như sau:

```
bmax = 1; lengthmax = 1;
```

```
bcurr = 1; lengthcurr = 1;
```

Phần chương trình duyệt chính như sau:

cho k chạy từ 2 đến N

if $a_{k-1} < a_k$ then

lengthcurr:=lengthcurr + 1;

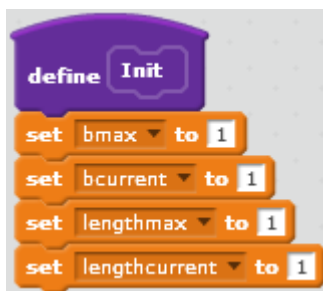
if lengthmax < lengthcurr then

bmax := bcurr;

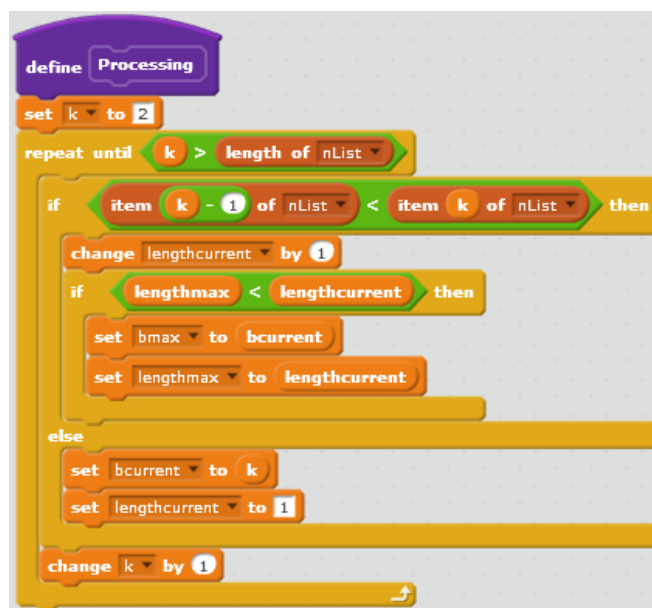
lengthmax := lengthcurr;

else

bcurr := k; lengthcurr := 1;

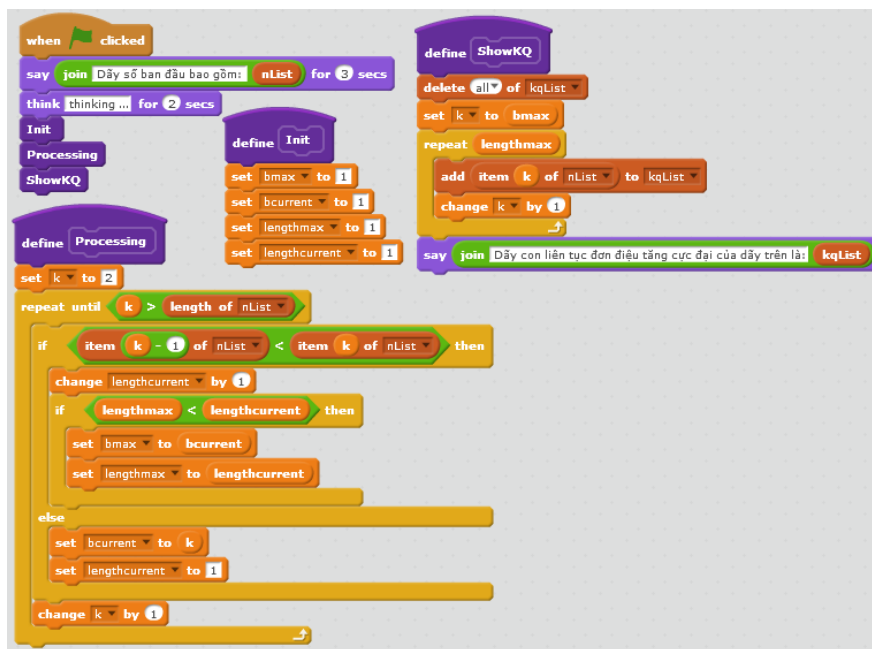


Thủ tục **Init** khai báo các giá trị ban đầu của các biến nhớ.



Thủ tục chính
Processing

Toàn bộ chương trình hoàn chỉnh như sau.



The Scratch code is as follows:

```

when clicked
  say join "Dãy số ban đầu bao gồm: " nList for 3 secs
  think thinking... for 2 secs
  Init
  Processing
  ShowKQ

define Init
  set bmax to 1
  set bcurrent to 1
  set lengthmax to 1
  set lengthcurrent to 1

define Processing
  set k to 2
  repeat until k > length of nList
    if item k - 1 of nList < item k of nList then
      change lengthcurrent by 1
      if lengthmax < lengthcurrent then
        set bmax to bcurrent
        set lengthmax to lengthcurrent
      else
        set bcurrent to k
        set lengthcurrent to 1
      change k by 1

define ShowKQ
  delete all of kqList
  set k to bmax
  repeat lengthmax
    add item k of nList to kqList
    change k by 1
  say join "Dãy con liên tục đơn điệu tăng cực đại của dãy trên là: " kqList
  
```

Two cartoon characters provide explanations:

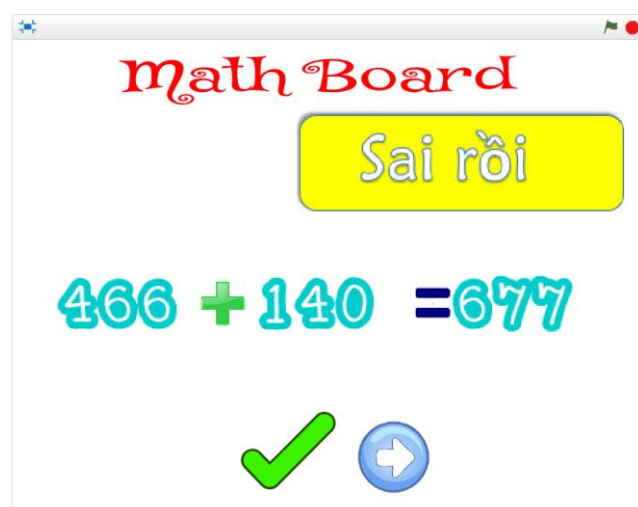
- Character 1: "Dãy số ban đầu bao gồm: 1 3 1 -1 1 2 3 2 1 3 5 6 7 7"
- Character 2: "Dãy con liên tục đơn điệu tăng cực đại của dãy trên là: 13567"

Câu hỏi và bài tập

1. Có người nói "tạo một thủ tục tức là tạo thêm một lệnh mới", đúng hay sai?
2. Viết thủ tục với tham số n tính số hạng dãy Fibonacci thứ n .
3. Cho trước 1 dãy số. Viết chương trình tìm ra 1 dãy con liên tục có các số bằng 0 có độ dài cực đại trong dãy trên.
4. Cho trước 2 dãy số **List1**, **List2**. Viết chương trình gộp 2 dãy này vào thành 1 dãy và sắp xếp theo thứ tự tăng dần. Kết quả đưa vào dãy **Listout**.

Mở rộng

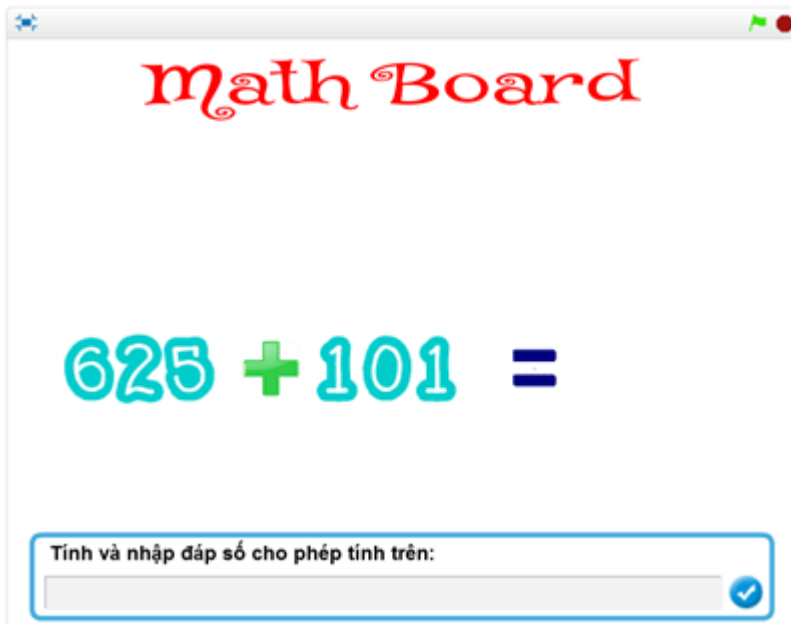
Thiết kế chương trình mô phỏng bài học tính cộng, trừ đơn giản sau.



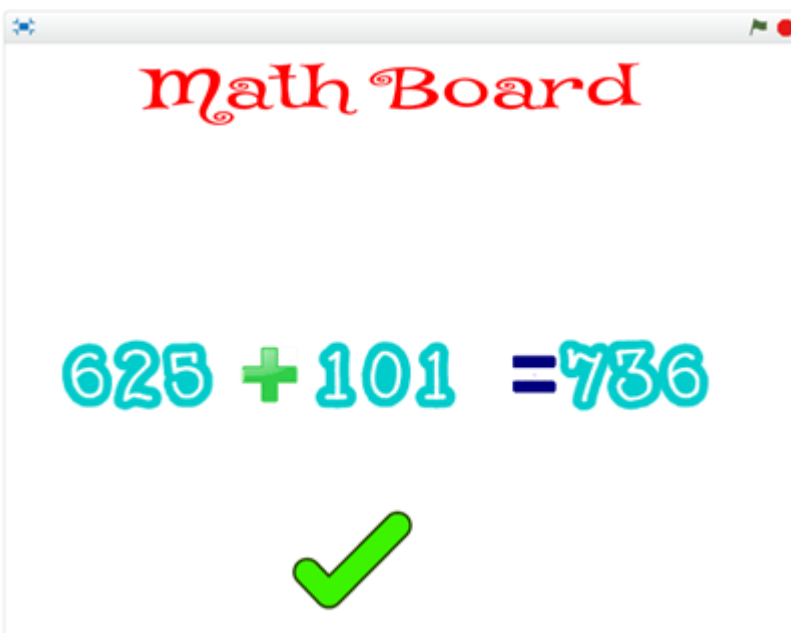
Chương trình sẽ tự động sinh các phép toán cộng, trừ số đơn giản. Người dùng nhập trực tiếp từ bàn phím đáp án. Nút Check sẽ kiểm tra đáp án đó là đúng hay sai và thông báo trên màn hình. Nếu làm sai, chương trình sẽ ghi đáp án đúng bên cạnh đáp án sai.

Sau khi làm xong, nhấn nút Next để tự động sinh và chuyển

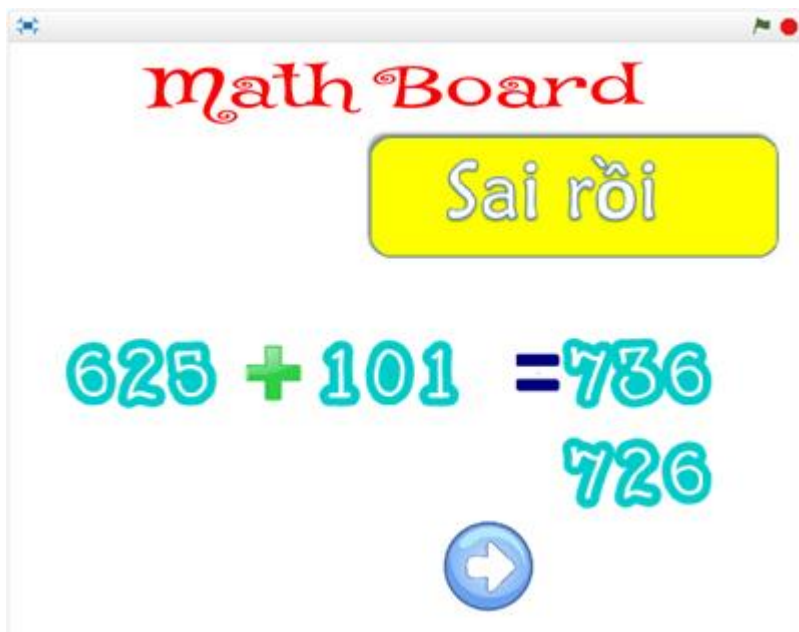
Cụ thể hơn, yêu cầu các bước thực hiện của chương trình như sau:



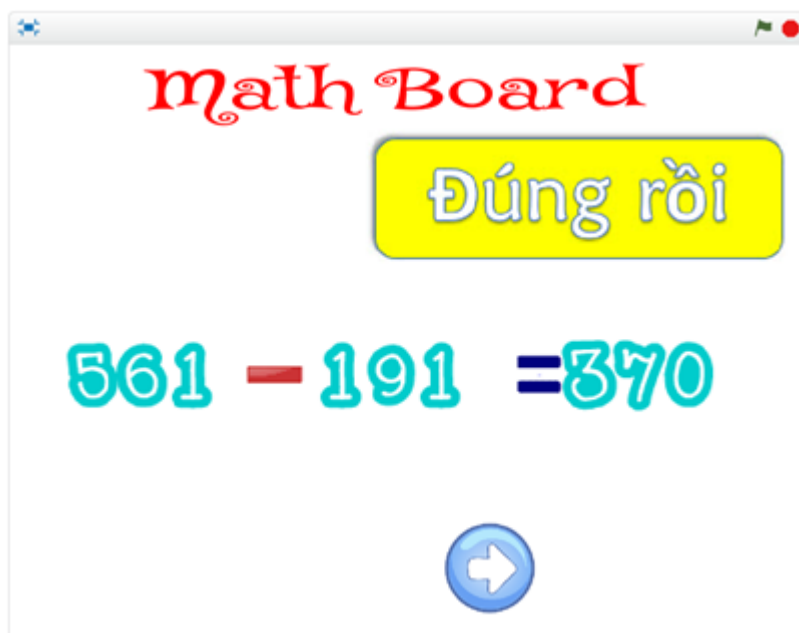
Phần mềm tự động sinh phép toán và dữ liệu bài toán, yêu cầu người chơi nhập đáp án từ bàn phím.



Khi người dùng nhập xong, đáp án do người dùng nhập sẽ hiện trên phép tính ở màn hình. Khi nó nút Check xuất hiện. Nháy nút này để kiểm tra đúng hay sai.



Nếu sai, chương trình sẽ hiện thông báo sai rồi và hiện đáp án đúng bên dưới. Đồng thời xuất hiện nút Next. Nháy lên nút này để bắt đầu 1 phép tính mới.



Nếu đúng, chương trình sẽ xuất hiện thông báo Đúng rồi. Đồng thời xuất hiện nút Next. Nháy lên nút này để bắt đầu 1 phép tính mới.

Em hãy thiết kế và hoàn thiện chương trình này.

Bài 20. Clone. Phân thân của nhân vật

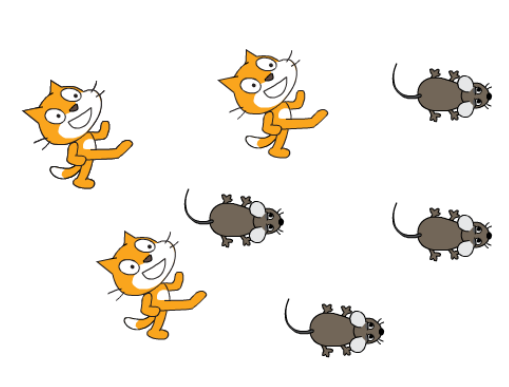
Mục đích

Học xong bài học này, bạn sẽ hiểu được:

- Khái niệm phân thân (clone) của nhân vật và ý nghĩa của Clone.
- Tính chất và thuộc tính của Clone.
- Một vài ứng dụng Clone trong các bài toán thực tế.

Bắt đầu

Em hãy nhìn vào hình ảnh 1 ứng dụng của Scratch sau và có nhận xét gì?



Trên màn hình em sẽ thấy nhiều con mèo và chuột cùng chuyển động. Rõ ràng đây không phải là hình ảnh của lệnh **stamp**, mà phải là các nhân vật.

Để thiết kế chương trình trên em phải làm gì?

- Sử dụng lệnh **stamp** liên tục?
- Tạo ra nhiều nhân vật có hình dạng giống nhau và cho chúng chuyển động?
- Em có cách nào khác hay không?

Trong bài học này, chúng ta sẽ làm quen với 1 khái niệm hoàn toàn mới trong Scratch có thể giải quyết dễ dàng bài toán trên.

Nội dung bài học

1. Khái niệm phân thân - clone trong Scratch

Tất cả các nhân vật trong Scratch đều có thể "phân thân", tức là tạo ra các nhân vật khác là song sinh với chính bản thân mình. Các nhân vật được phân thân đó gọi là Clone.

Chúng ta cùng xem 1 chương trình ngắn để bước đầu làm quen và phân biệt được 2 khái niệm: nhân vật chính (gốc) và phân thân (Clone) của nhân vật này.

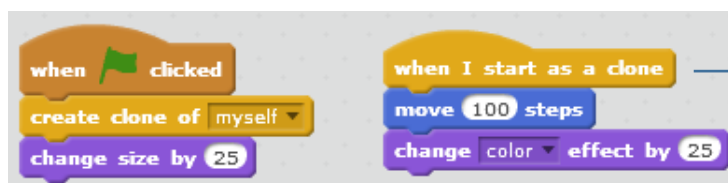


Nhân vật chính, gốc

Nhân vật được phân thân, là song sinh của nhân vật gốc.

Nhân vật này được gọi là **clone** của nhân vật chính.

Chúng ta cùng xem đoạn chương trình của nhân vật chính con mèo.



Đoạn chương trình điều khiển **clone**.

Đoạn chương trình khởi tạo **clone**.

Nhân vật gốc và Clone



Nhân vật gốc là khái niệm Nhân vật (Sprite) mà chúng ta vẫn biết từ trước đến nay trong môi trường Scratch.

Mỗi nhân vật khi được tạo ra sẽ có các tính chất, thuộc tính riêng như hình ảnh, tọa độ x, y, hướng quay, trang phục, âm thanh, kích thước, các biến nhớ riêng. Mỗi nhân vật có 1 cửa sổ lệnh riêng của mình.

Mỗi nhân vật có thể tạo ra các phân thân (clone) của riêng mình.




Clone được khởi tạo từ 1 nhân vật gốc. Clone là 1 phân thân của nhân vật gốc và có mọi thuộc tính của nhân vật gốc.

Khi bắt đầu khởi tạo, clone kế thừa toàn bộ tính chất, thuộc tính của nhân vật gốc. Tuy nhiên sau khi ra đời, có thể điều khiển clone bằng tất cả các lệnh của Scratch như 1 nhân vật bình thường. Điểm khác biệt chỉ ở chỗ:

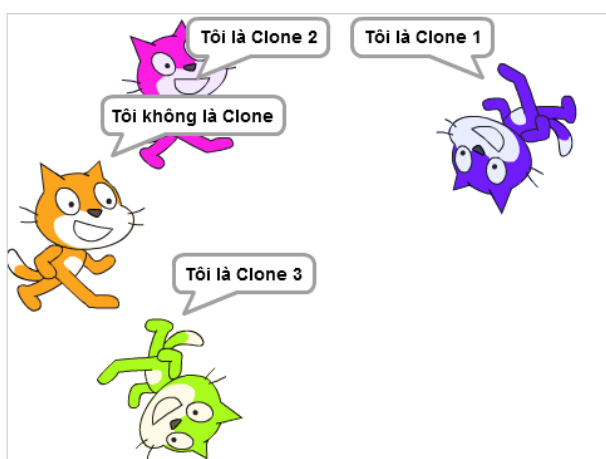
- Các chương trình điều khiển Clone bắt buộc nằm trong lệnh sự kiện **when I start as a clone**.
- Bản thân clone có thể xóa chính mình.

Bảng sau liệt kê các thông tin nhanh liên quan đến nhân vật gốc và clone.

	Nhân vật gốc	Clone
Mô tả nhanh	Là nhân vật hoạt động chính trên sân khấu cho người sử dụng khởi tạo, hoạt động vĩnh viễn.	Là nhân vật phân thân từ 1 nhân vật gốc, được khởi tạo bởi lệnh create clone of . Clone sau khi được tạo ra sẽ có đầy đủ tính chất như 1 nhân vật bình thường và kế thừa mọi

	Nhân vật gốc	Clone
		thuộc tính từ nhân vật gốc của mình. Clone không hoạt động vĩnh viễn.
Khởi tạo Clone	Nhân vật chính dùng lệnh  để tạo Clone. Có thể tạo Clone của mình hoặc của các nhân vật khác. Chú ý: Sân khấu cũng có quyền tạo Clone cho mọi nhân vật.	
Điều khiển Clone		Clone sau khi được tạo ra sẽ chịu sự điều khiển của câu lệnh sự kiện  . Cho phép nhiều chương trình cùng điều khiển 1 Clone.
Xóa Clone		Clone tự xóa bản thân mình bằng lệnh  .
Thời gian sống	Vĩnh viễn	Chỉ hoạt động trong thời gian chạy chương trình.

Chúng ta cùng xét 1 ví dụ sau. Ví dụ này minh họa cho quan hệ giữa nhân vật chính và các phân thân - clone của chính mình.



Tạo 1 biến nhớ riêng có tên CloneID của nhân vật chính. Chương trình sẽ lần lượt tạo ra 3 clone, và trước mỗi lần tạo sẽ gán giá trị CloneID lần lượt là 1, 2, 3. Các clone này sẽ kế thừa biến nhớ CloneID cho riêng mình với các giá trị lần lượt là 1, 2, 3. Khi tạo ra, các clone sẽ di chuyển ngẫu nhiên, tự do trên màn hình và luôn hiện giá trị CloneID của riêng mình để phân biệt với các clone khác.

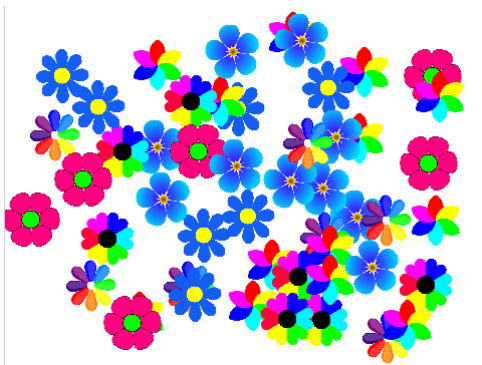
Chương trình cụ thể trên Scratch như sau:



Clone có ý nghĩa gì trong các ứng dụng thực tế của Scratch, chúng ta cùng tìm hiểu các hoạt động tiếp theo.

2. Rừng hoa

Chương trình đơn giản sau cho em hiểu thêm hoạt động của clone.



Chương trình đơn giản này chỉ có đúng 1 nhân vật là bông hoa, nhưng với nhiều hình ảnh đẹp mắt khác nhau.

Mục đích của chương trình là tạo ra 1 rừng hoa với nhiều màu sắc sặc sỡ.

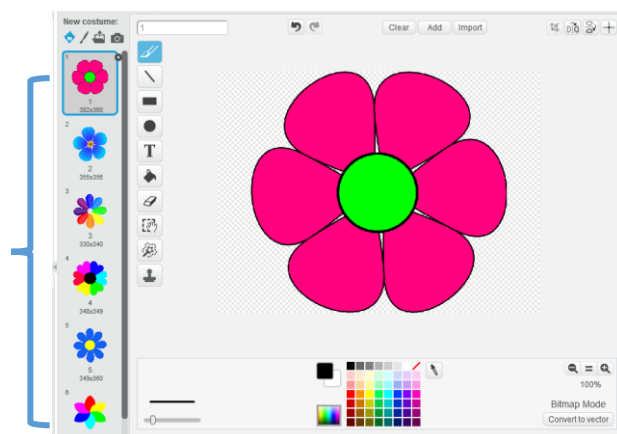
Để làm như vậy, em cần sưu tầm nhiều hình ảnh hoa với màu sắc đa dạng khác nhau và đưa vào thành các trang phục của nhân vật này.

Chương trình được xây dựng đơn giản như sau:

- Nhân vật chính bông hoa cần tạo ra nhiều hình ảnh khác nhau nhưng có kích thước gần giống nhau làm trang phục. Ví dụ:

Thiết lập các hình bông hoa có kích thước gần giống nhau và màu sắc đa dạng khác nhau.

Dãy các trang phục (costume) được đánh số từ 1



- Trong mạch chương trình chính, em cho bông hoa lần lượt tạo ra 50 clone của chính mình.



Lệnh này sẽ tạo ra 50 bản clone của nhân vật chính. Các phân thân clone này khi tạo ra có trang phục và khuôn dạng giống với nhân vật chính và ở tại đúng vị trí của nhân vật chính (nhưng nằm phía dưới).

- Mỗi phân thân, clone sẽ di chuyển nhanh tới 1 vị trí ngẫu nhiên trên màn hình với thể hiện trang phục ngẫu nhiên.



3. Trò chơi mèo đuổi chuột

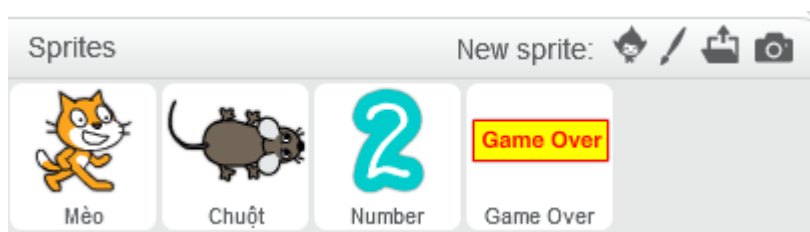
Trong hoạt động này, em hãy thiết kế trò chơi Mèo đuổi Chuột.

Nhân vật chính của chúng ta là Mèo và Chuột. Trò chơi được mô tả đơn giản như sau:

- Lũ chuột sẽ được phân thân và chạy lung tung (ngẫu nhiên) trên màn hình. Thấy mèo từ xa chuột đã quay đầu bỏ chạy.
- Mèo được điều khiển bởi con người (dùng các phím điều khiển). Nhiệm vụ của người chơi là điều khiển mèo đuổi và bắt chuột càng nhiều càng tốt.
- Thời gian mỗi lần chơi chỉ là 1 phút. Trên màn hình luôn hiển thị số lượng chuột đã bị bắt. Khi kết thúc trò chơi, bạn nào bắt được số chuột nhiều hơn sẽ chiến thắng. Khi kết thúc, thông báo Game Over sẽ hiện trên màn hình.



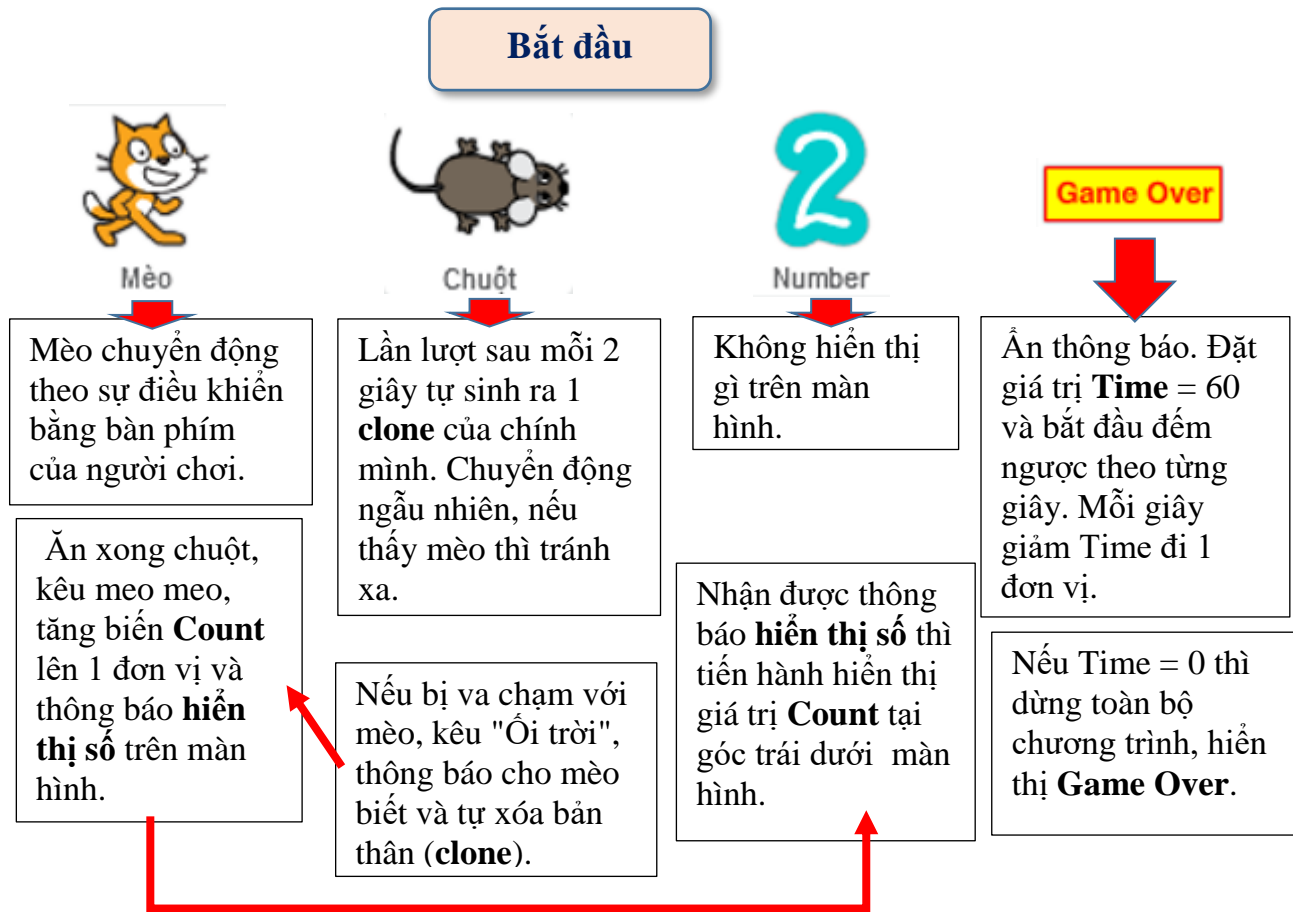
Để thiết kế chương trình chúng ta bắt đầu từ các nhân vật. Hệ thống nhân vật bao gồm: mèo, chuột, các chữ số (từ 0 đến 9) và nút thông báo Game Over.



Sử dụng 2 biến nhớ tổng thể:

Time - thời gian chơi và
Count - số lượng chuột bị

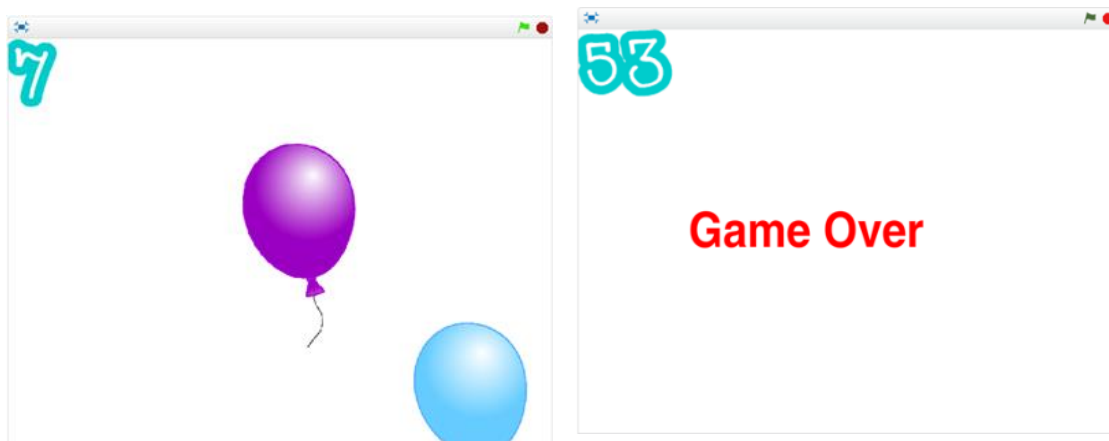
Sơ đồ hoạt động của trò chơi như sau:



4. Trò chơi bóng bay

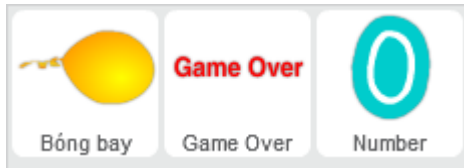
Trò chơi được mô tả như sau:

Từ phía dưới các quả bóng bay sẽ xuất hiện ngẫu nhiên và bay lên trên. Nhiệm vụ của người chơi là nháy chuột lên các quả bóng bay này. Mỗi khi nháy lên quả bóng bay sẽ nổ và biến mất. Thời gian chơi là 1 phút. Góc trái trên màn hình sẽ thể hiện số bóng bay đã được nháy trúng. Người thắng cuộc là người đã làm nổ nhiều số bóng bay nhất. Khi hết thời gian, màn hình xuất hiện dòng chữ và âm thanh "Game Over".



Gợi ý thiết kế chương trình.

Em hãy thiết lập 3 nhân vật sau: **Bóng bay**, **Chữ số** và biển thông báo **Game Over**. Ngoài ra cần có thêm 2 biến nhớ chung là **Time** - thời gian dùng để đếm ngược khi chơi và **Count** - biến nhớ ghi lại số lượng bóng bay đã được đánh trúng.



Nhiệm vụ mỗi nhân vật trên như sau:

1. Bóng bay

Bóng bay sẽ được tự động tạo các **clone** của mình và bay lên từ 1 vị trí ngẫu nhiên ở phía dưới. Bóng bay được thiết kế có 1 số hình ảnh với màu sắc khác nhau. Người chơi sẽ nhấp chuột lên các quả bóng này. Nhiệm vụ của người chơi là nhấp càng nhiều càng tốt lên các quả bóng bay. Bóng bay sẽ biến mất nếu vượt ra ngoài màn hình hoặc bị người chơi nhấp đúng.

2. Chữ số

Góc trái trên màn hình sẽ luôn thể hiện số quả bóng bay đã bấm đúng. Nhân vật này sử dụng biến nhớ **Count** để đếm số bóng bay bị nhấp chuột đúng và thể hiện con số này tại góc trái trên màn hình.

3. Biển thông báo Game Over

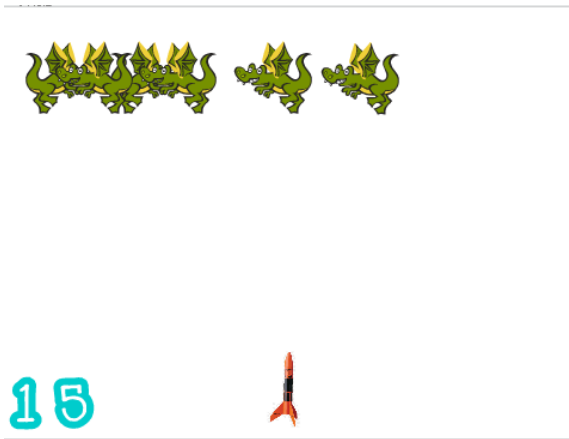
Trò chơi kết thúc sau 60 giây. Nhân vật Biển thông báo này có nhiệm vụ đọc thời gian và thông báo kết thúc trò chơi sau đúng 60 giây. Biến **Time** dùng làm nhiệm vụ đếm ngược thời gian.

Câu hỏi và bài tập

- Viết chương trình cho con Mèo gốc phân thân thành 3 clone, 1 con chạy sang trái, 1 con chạy sang phải và 1 con chạy lên trên.
- Em hãy làm cho trò chơi **Rừng hoa** đẹp lên như sau:
Sau khi xuất hiện các bông hoa sẽ tự động đổi màu và xoay tròn trên màn hình. Em sẽ thấy một bức tranh sắc màu sắc sỡ, lung linh đẹp mắt.
- Các phân thân của nhân vật gốc có thể đặt tên được hay không? Vì sao?
- Có thể lập trình điều khiển riêng cho từng Clone được hay không? Lấy ví dụ cụ thể.

Mở rộng

Thiết kế trò chơi vui sau: **Tên lửa bắn rồng**.



Phía trên các con rồng sẽ xuất hiện ngẫu nhiên và bay ngang qua màn hình theo chiều ngang. Phía dưới có hình 1 tên lửa. Nhiệm vụ của người chơi là điều khiển tên lửa này đến bắn rồng. Người chơi thắng cuộc nếu bắn được > 100 con rồng. Số con rồng bắn được luôn hiện tại góc trái dưới. Người chơi sẽ thua nếu số lượng rồng xuất hiện quá đông > 10. Người chơi dùng phím Space để bắn, dùng phím trái,

Các nhân vật chính của trò chơi này.



Sơ đồ hoạt động của chương trình như sau.



Sau mỗi giây lại sinh ra 1 clone, tăng biến **gragon-count** lên 1 và chuyển động ngang

Nếu gặp tên lửa thì tăng biến **count**, tự biến mất và giảm biến **dragon-count** đi 1 đơn vị. Thông báo **Show Number**.



Chịu sự điều khiển của người chơi, hướng theo cách người dùng phím điều khiển.

Người dùng bấm phím Space - sinh ra 1 clone và bắn tên lửa theo hướng lên trên.

Nếu bắn trúng rồng, tự biến mất. Nếu gặp cạnh trên màn hình cũng biến mất.



Nếu nhận thông điệp **Show Number** thì hiển thị số **count** trên màn hình.



Kiểm soát 2 giá trị: **count** và **dragon-**

Nếu **count** > 100, dừng chương trình và hiển thị **Victory**.

Nếu **dragon-count** > 10 thì dừng chương trình và hiển thị **Game Over**.