**Trường Đại học Cần Thơ**

**Trường Công Nghệ Thông Tin - Truyền Thông**



**NIÊN LUẬN CƠ SỞ NGÀNH KTPM**

**WEBSITE MINH HỌA THUẬT TOÁN BUBBLE SORT**

**Sinh viên thực hiện:**

Họ và Tên: Nguyễn Tiến Lĩnh

MSSV: B2012108

Lớp: Kỹ thuật phần mềm 03

Khóa: 46

**Cán bộ hướng dẫn:**

PGS.TS Huỳnh Xuân Hiệp

**HỌC KỲ 2, 2022-2023**

**ĐÁNH GIÁ CỦA GIÁO VIÊN HƯỚNG DẪN**

**MỤC LỤC**

**[CHƯƠNG 1: TỔNG QUAN 3](#_Toc3817)**

[I. MÔ TẢ BÀI TOÁN 3](#_Toc19944)

[II. MỤC TIÊU CẦN ĐẠT ĐƯỢC 3](#_Toc4358)

[III. HƯỚNG GIẢI QUYẾT 3](#_Toc779)

[IV. KẾ HOẠCH THỰC HIỆN 4](#_Toc2001)

**[CHƯƠNG 2: LÝ THUYẾT 5](#_Toc560)**

[I. CÁC KHÁI NIỆM SỬ DỤNG TRONG ĐỀ TÀI 5](#_Toc28012)

[2.1.1. Sắp xếp 5](#_Toc32606)

[2.1.2. Thuật toán sắp xếp 5](#_Toc26249)

[2.1.3. Sắp xếp nổi bọt 5](#_Toc21047)

[II. HƯỚNG DẪN LÝ THUYẾT 7](#_Toc783)

[III. KẾT QUẢ VẬN DỤNG LÝ THUYẾT VÀO ĐỀ TÀI 9](#_Toc20112)

**[CHƯƠNG 3: KẾT QUẢ ỨNG DỤNG 10](#_Toc25828)**

[I. PHÂN TÍCH YÊU CẦU BÀI TOÁN VÀ CẤU TRÚC DỮ LIỆU 10](#_Toc16016)

[II. THIẾT KẾ GIẢI THUẬT 11](#_Toc1540)

[3.2.1. Các biến dùng chung 11](#_Toc29422)

[3.2.2. Hàm createBubble 12](#_Toc24652)

[3.2.3. Hàm init 12](#_Toc16910)

[3.2.4. Hàm addNumber 13](#_Toc3235)

[3.2.5. Hàm check 13](#_Toc21767)

[3.2.6. Hàm checkFormat 14](#_Toc23653)

[3.2.7. Hàm addString 14](#_Toc23002)

[3.2.8. Hàm bubbleSort 15](#_Toc18242)

[3.2.9. Hàm MiniBubble 16](#_Toc11361)

[3.2.10. Hàm moveUp 16](#_Toc6381)

[3.2.11. Hàm swap 17](#_Toc21442)

[3.2.12. Hàm sorting 17](#_Toc25924)

[3.2.13. Hàm percent 18](#_Toc24682)

[3.2.14. Hàm setDescription 18](#_Toc16103)

[3.2.15. Hàm setColorCode 18](#_Toc22228)

[3.2.16. Hàm setColorBubble 18](#_Toc19283)

[3.2.17. Hàm executeAStep 19](#_Toc30129)

[3.2.18. Hàm createRowTable 20](#_Toc26872)

[3.2.19. Hàm executeAllStep 20](#_Toc23084)

[3.2.20. Hàm createHeadLineTable 22](#_Toc32489)

[3.2.21. Xử lý sự kiện cho nút “Sort” 22](#_Toc23382)

[3.2.22. Xử lý sự kiện cho nút “Pause” hoặc “Continue” 23](#_Toc2833)

[3.2.23. Xử lý sự kiện cho nút “Clear” 23](#_Toc32266)

[3.2.24. Xử lý sự kiện cho nút “Next Step” 24](#_Toc27413)

[3.2.25. Xử lý sự kiện cho nút “Random” 24](#_Toc11456)

[3.2.26. Xử lý sự kiện cho các nút tốc độ 25](#_Toc32078)

[3.2.27. Xử lý sự kiện cho nút “Import” 26](#_Toc31528)

[3.2.28. Xử lý sự kiện cho nút “Previous Step” 26](#_Toc25212)

[III. GIỚI THIỆU CHƯƠNG TRÌNH 27](#_Toc15245)

**[CHƯƠNG 4: KẾT LUẬN ĐÁNH GIÁ 28](#_Toc11806)**

[I. KẾT QUẢ ĐẠT ĐƯỢC 28](#_Toc7242)

[4.1.1. Về kiến thức 28](#_Toc670)

[4.1.2. Về kỹ năng 28](#_Toc30159)

[4.1.3. Về thái độ 28](#_Toc16293)

[II. HẠN CHẾ 28](#_Toc30230)

[III. NGUYÊN NHÂN 29](#_Toc23642)

[IV. HƯỚNG PHÁT TRIỂN 29](#_Toc31861)

**[PHỤ LỤC 30](#_Toc5214)**

**[TÀI LIỆU THAM KHẢO 31](#_Toc20774)**

# CHƯƠNG 1: TỔNG QUAN

## MÔ TẢ BÀI TOÁN

Giải thuật sắp xếp là một trong các nhóm giải thuật quan trọng của ngành lập trình, khoa học máy tính và cả toán học vì mang lại nhiều ứng dụng thực tiễn. Thuật toán sắp xếp bong bóng (Bubble sort) là một trong số các thuật toán sắp xếp phố biển do tính dễ tiếp thu, dễ cài đặt vì vậy mà được đưa vào giảng dạy trong nhiều chương trình đào tạo đại học về lập trình.

Tuy nhiên, việc tìm hiểu giải thuật sắp xếp nổi bọt nếu chỉ dừng lại ở mức độ lý thuyết vẫn chưa thể hiện hết bản chất của giải thuật, sự hình dung của người học với vấn đề còn chưa sinh động. Để giải quyết vấn đề nêu trên, Website minh họa giải thuật sắp xếp nổi bọt được xây dựng nhằm truyền tải đến người dùng đúng với tinh thần của giải thuật.

## MỤC TIÊU CẦN ĐẠT ĐƯỢC

* Hiểu rõ và nắm vững các kiến thức về cấu trúc dữ liệu và giải thuật sắp xếp có liên quan đến đề tài.
* Biết cách thiết kế giải thuật và cài đặt giải thuật vào chương trình.
* Sản phẩm có giao diện dễ nắm bắt đối với mọi đối tượng người dùng.
* Thực hiện được các chuyển động cần có để minh họa giải thuật.
* Hiển thị thông báo về các sự kiện đang xảy ra với người dùng.
* Thể hiện được các thao tác cơ bản với dữ liệu.
* Giải thuật hoạt động tốt với dữ liệu nhận vào từ người dùng.
* Có thể thực hiện được từng bước của giải thuật.
* Có bảng dữ liệu ghi lại quá trình xử lý giải thuật.

## HƯỚNG GIẢI QUYẾT

Trước hết cần tìm hiểu và ôn tập kiến thức về cấu trúc dữ liệu và giải thuật có liên quan đến sắp xếp nổi bọt và các kiến thức về lập trình web với HTML, CSS và JS. Ngoài ra, cần tìm hiểu về các thư viện hỗ trợ cho quá trình xây dựng website như Bootstrap hay các giao diện bản mẫu có sẳn (template) để giúp rút ngắn quá trình thiết kế, cần lưu ý lựa chọn các giao diện có bố cục gọn gàn, tươi sáng, dễ quan sát và nắm bắt các chức năng có trên giao diện.

Trong giai đoạn tiếp theo, sử dụng CSS và ngôn ngữ lập trình Javascript để xây dựng các hàm thực hiện các chuyển động cần thiết như chuyển động nổi bọt, chuyển động đi lên của bong bóng,... Đồng thời đặt sự kiện cho các nút, các hoạt động submit có thể có trên website, xây dựng các ô nhập liệu để người dùng đưa input vào hệ thống xử lí. Về giải thuật, để có thể minh họa được giải thuật cần kết hợp việc xử lí giải thuật bằng vòng lặp để giải quyết vấn đề và dùng mảng để lưu trữ các trạng thái ở từng cặp số được so sánh rồi dùng đệ quy để tái hiện lại quá trình sắp xếp một cách có thứ tự về mặt thời gian cùng với các hiệu ứng đi kèm.

## KẾ HOẠCH THỰC HIỆN

|  |  |
| --- | --- |
| **Thời gian** | **Công việc** |
| Tuần 01 | Nhận được hướng dẫn về đề tài niên luận |
| Tuần 02 | Ôn tập kiến thức về giải thuật và lập trình với Bubble sort |
| Tuần 03 | Tìm kiếm giao diện website phù hợp và điều chỉnh bổ sung chức năng cần thiết |
| Tuần 04 | Lập trình và kiểm thử các chức năng thành phần:   * Nhập một số. * Nhập một chuỗi số. * Nút “Sort” (sắp xếp). * Nút “Random” (sinh dữ liệu ngẫu nhiên). * Nút “Clear” (làm rỗng khu vực sắp xếp). * Nút “Pause” (dừng giải thuật). * Nút “Continue” (tiếp tục giải thuật). * Nút “Import” (đọc dữ liệu từ tập tin). |
| Tuần 05 - Tuần 07 | Lập trình và kiểm thử giải thuật sắp xếp nổi bọt:   * Sắp xếp bằng giải thuật. * Lưu trữ kết quả ở từng bước sắp xếp. * Dùng hàm đệ qui gọi lại từng bước sau một khoảng thời gian nhất định để tái hiện quá trình sắp xếp từng bước. * Hộp thoại hướng dẫn sử dụng. |
| Tuần 08 - Tuần 10 | Kết hợp giải thuật với giao diện website |
| Tuần 11 - Tuần 12 | * Kiểm thử phần mềm. * Viết tài liệu hướng dẫn sử dụng. |
| Tuần 13 | Đóng gói phần mềm và viết báo cáo |

# CHƯƠNG 2: LÝ THUYẾT

## CÁC KHÁI NIỆM SỬ DỤNG TRONG ĐỀ TÀI

### Sắp xếp

Sắp xếp là quá trình tái bố trí các phần tử trong một tập hợp cho trước thành một tập hợp mới theo một trình tự nào đó nhằm mục đích giúp quản lý và tìm kiếm các phần tử dễ dàng và hiệu quả hơn.

### Thuật toán sắp xếp

Trong khoa học máy tính và trong toán học, thuật toán sắp xếp là một thuật toán mà nó sắp xếp các phần tử của một danh sách (hoặc một mảng) theo thứ tự (tăng hoặc giảm). Người ta thường xét trường hợp các phần tử cần sắp xếp là các số [4]. Các thuật toán sắp xếp được chia thành 2 loại:

* Sắp xếp đơn giản: sắp xếp chọn (Selection sort), sắp xếp xen (Insertion sort), sắp xếp nổi bọt (Bubble sort), ...
* Sắp xếp phức tạp: sắp xếp nhanh (Quick sort), sắp xếp vun đống (Heap sort), sắp xếp nhị phân (Bin sort), ...

### Sắp xếp nổi bọt

#### Đặt vấn đề

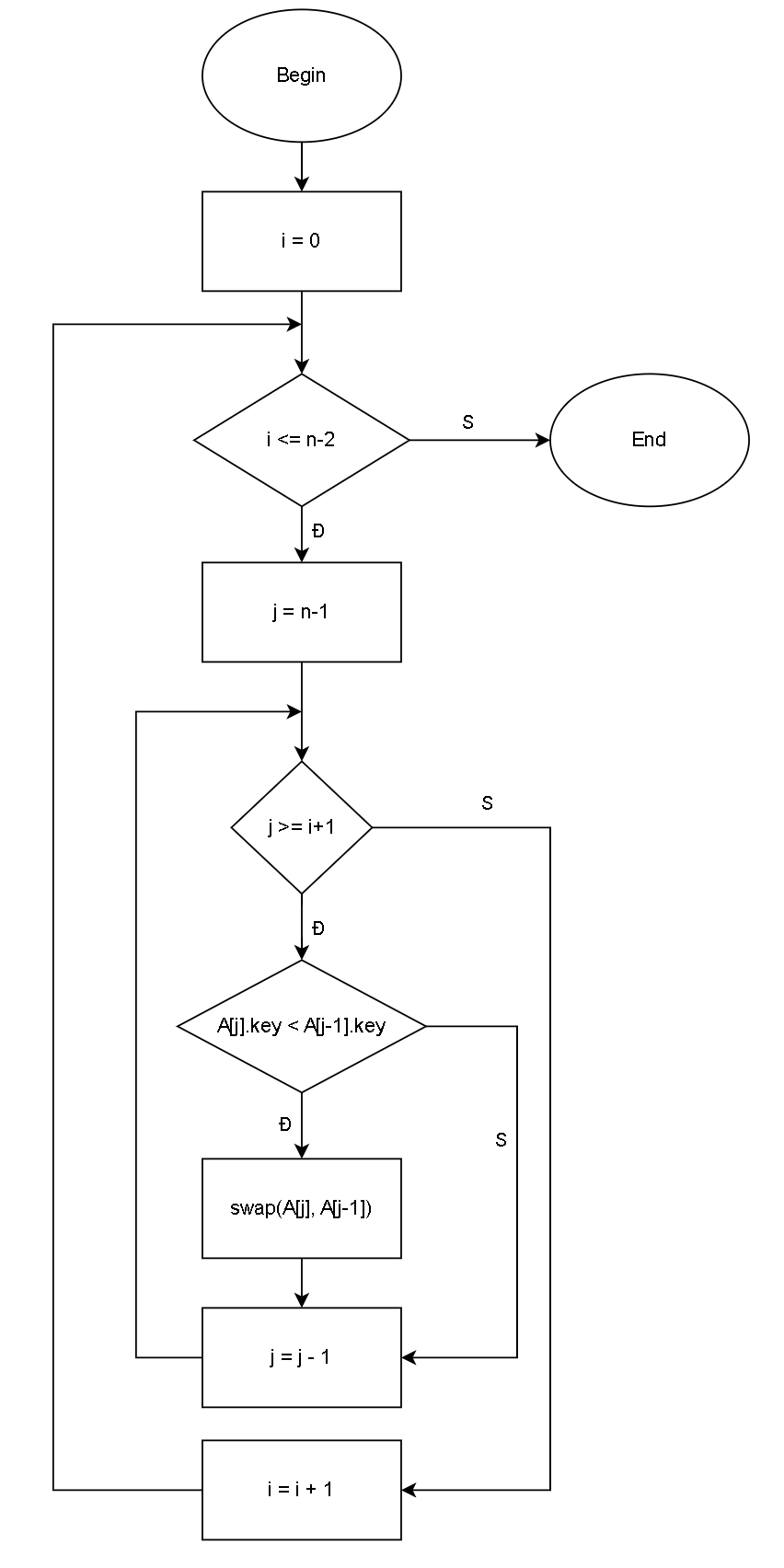
* Giả thuyết, cho mảng A có n phần tử số nguyên không có thứ tự với mỗi phần tử được biểu diễn dưới dạng A[i] với 0 ≤ i ≤ n-1. Ta có mảng A được biểu diễn như sau: A = {A[0], A[1], ..., A[n-1]}
* Vấn đề đặt ra: sắp xếp mảng A thành một dãy tăng dần\giảm dần bằng thuật toán sắp xếp nổi bọt.

#### Ý tưởng

Xuất phát từ cuối (hoặc đầu) mảng, đổi chổ các cặp phần tử kế cận để đưa phần tử nhỏ (lớn) hơn trong cặp phần tử đó về vị trí đứng đầu (cuối) mảng hiện hành. Sau đó sẽ không xét đến nó ở lần tiếp theo, do vậy ở lần xử lý thứ i sẽ có vị trí đầu mảng là i. Lặp lại xử lý trên cho đến khi không còn cặp phần tử nào để xét.

#### Giải thuật và lưu đồ

* Bước 1: Xét các a[j] (j giảm từ n-1 đến 1), so sánh khoá của a[j] với khoá của a[j-1]. Nếu a[j] < a[j-1] thì đổi a[j] và a[j-1]. Sau bước này thì a[0] có khoá nhỏ nhất.
* Bước 2: Xét các a[j] (j giảm từ n-1 đến 2), so sánh khoá của a[j] với khoá của a[j-1]. Nếu a[j] < a[j-1] thì đổi a[j] và a[j-1]. Sau bước này thì a[1] có khoá nhỏ thứ 2.
* Lặp lại tương tự bước 1 và bước 2.
* Bước n-1: mảng đã được sắp thứ tự

***Hình 1******.*** Lưu đồ giải thuật sắp xếp nổi bọt

#### Đánh giá giải thuật

* Độ phức tạp giải thuật là O(n2) [2](Trang 37) có dạng đa thức thuộc nhóm có thể chấp nhận được[3].
* Số lượng phép so sánh ở mỗi bước xác định theo số lần lặp trong vòng for mà không phụ thuộc vào tình trạng của mảng ban đầu.
* Các phần tử nhỏ được đưa về vị trí đúng rất chậm trong khi các phần tử lớn lại được đưa về vị trí đúng rất nhanh.

## HƯỚNG DẪN LÝ THUYẾT

* **Vấn đề:**

Cho mảng A={1, 6, 4, 10, 8, 2, 7} gồm 7 số nguyên không theo thứ tự. Hãy sắp xếp mảng trên theo thứ tự tăng dần từ trái sang phải.

* **Lí thuyết áp dung:**

Xuất phát từ bên trái mảng, ta lần lượt xét các phần tử A[j] với 0 ≤ j < n-1, tại A[j] ta xét phần tử A[j+1] kế cận nếu thỏa điều kiện A[j] > A[j+1] thì ta đổi chổ chúng và tiếp tục xét phần tử tiếp theo. Lặp lại giải thuật n-1 lần ta được mảng A có thứ tự tăng dần từ trái sang phải.

* **Giải quyết vấn đề:**

Xuất phát từ bên trái của mảng A. Với j=0, ta xét phần tử A[j]=A[0]=1 và phần tử kế cận của nó là A[j+1]=A[1]=6, xét thấy A[j]<A[j+1] không thỏa điều kiện hoán đổi. Tiếp tục với j=1, ta xét phần tử A[j]=6 và phần tử kế cận của nó A[2]=4, hoán đổi A[1] và A[2] và A[1]>A[2]. Mảng A từ:

A={1, **6**, **4**, 10, 8, 2, 7}

Trở thành:

A={1, **4**, **6**, 10, 8, 2, 7}

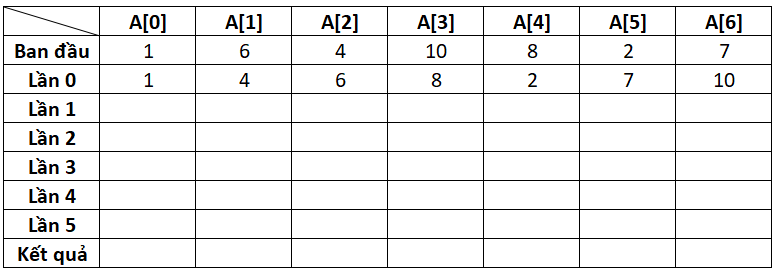
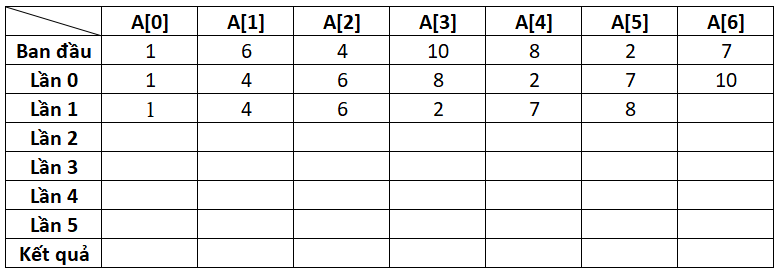
Tiếp tục với j=2, ta có A[2]=6 và A[3]=10. Vậy không đổi chổ A[2] và A[3].

Với j=3, ta có A[3] = 10 và A[4] = 8. Đổi chổ A[3] và A[4].

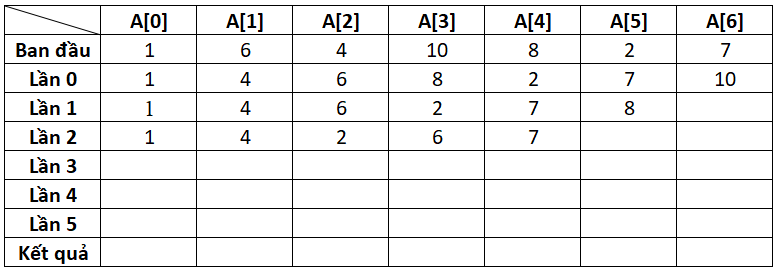
Với j=4, ta có A[4] = 10 và A[5] = 2. Đổi chổ A[4] và A[5].

Với j=5, ta có A[5] = 10 và A[6] = 7. Đổi chổ A[5] và A[6].

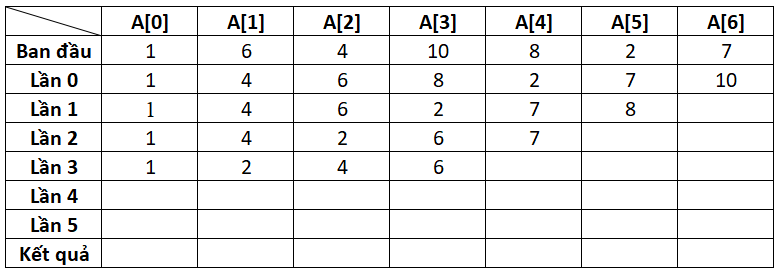
Tiếp tục lặp lại quá trình trên ta nhận được bảng A có thứ tự tăng dần.

***Hình 2******:*** Mảng A sau lần lặp đầu tiên

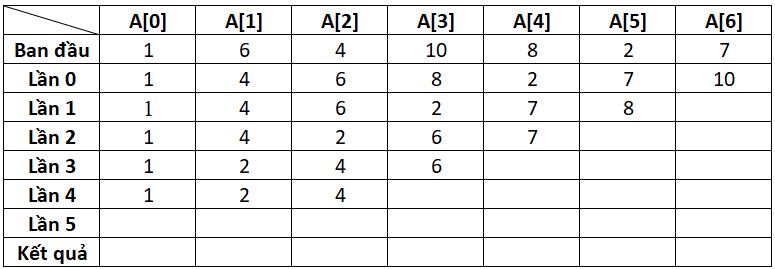
***Hình 3******:*** Mảng A sau lần lặp thứ 1



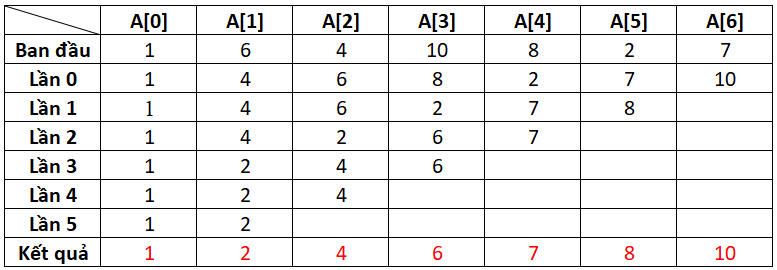
***Hình 4******:*** Mảng A sau lần lặp thứ 2



***Hình 5******:*** Mảng A sau lần lặp thứ 3



***Hình 6******:*** Mảng A sau lần lặp thứ 4

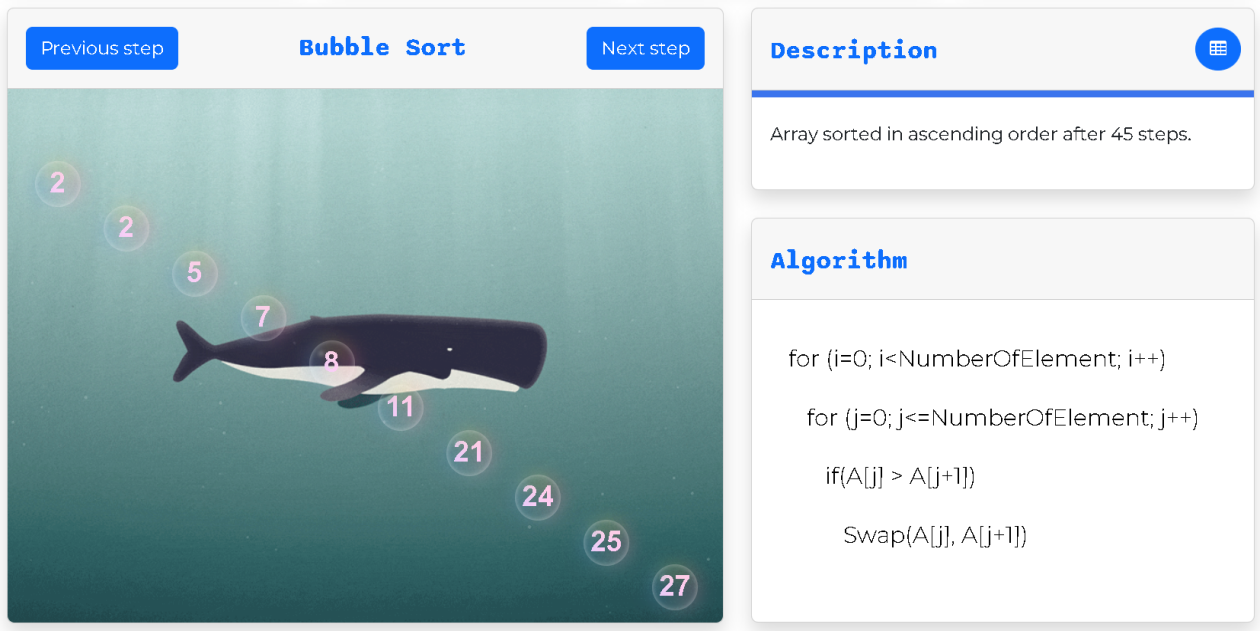
***Hình 7******:*** Mảng A sau lần lặp thứ 5 đã có thứ tự tăng dần

## KẾT QUẢ VẬN DỤNG LÝ THUYẾT VÀO ĐỀ TÀI

Sau khi vận dụng lý thuyết vào đề tài, nhận thấy:

- Giải thuật chạy đúng trên nhiều mẫu dữ liệu đầu vào khác nhau (bao gồm được tạo sẳn ngẫu nhiên hoặc nhập từ người dùng).

- Giải thuật cho ra kết quả có thứ tự sắp xếp theo chiều tăng dần từ trái sang.

***Hình 8******:*** Kết quả vận dụng giải thuật vào đề tài

# CHƯƠNG 3: KẾT QUẢ ỨNG DỤNG

## PHÂN TÍCH YÊU CẦU BÀI TOÁN VÀ CẤU TRÚC DỮ LIỆU

Trang web minh họa giải thuật sắp xếp nổi bọt có yêu cầu trọng tâm là việc thể hiện được ý nghĩa, cách thức hoạt động và kết quả giải thuật một cách sinh động trên nền tảng giao diện đồ họa.

Về dữ liệu đầu vào, với tham số đầu vào từ người dùng có thể là một con số (trong trường hợp nhập từng phần tử) hoặc một chuỗi số (trong trường hợp nhập nhiều phần tử cùng lúc) với số lượng phần tử qui ước là 10 đối với kích thước khung hình máy tính và 6 đối với kích thước thiết bị di động. Hệ thống cần xây dựng một hàm tiếp nhận tham số đầu vào và chuyển đổi chúng thành các đối tượng bong bóng trên giao diện đồ họa. Việc nhập liệu từ người dùng có thể xảy ra sơ xót, sai định dạng vì vậy các ô nhập liệu cần được thiết đặt các định dạng cho tham số đầu vào như giá trị nhận vào phải là số nguyên dương, chuỗi số nhận vào phải có định dạng hợp lệ để giúp quá trình xử lí dữ liệu trong nội hàm của hệ thống dễ dàng hơn.

Về thuật toán xử lí, sau khi người dùng bấm nút “Sort” (sắp xếp) trên giao diện, một hàm *bubbleSort* với tham số đầu vào *arrayNum* là mảng lưu trữ các số vừa nhập để thực hiện quá trình sắp xếp nổi bọt. Đồng thời trong quá trình sắp xếp, dùng 3 mảng *num1*, *num2* và *swapped* lần lượt lưu các giá trị số thứ nhất trong phép số sánh đổi chổ, số thứ hai trong phép so sánh đổi chổ và giá trị luận lý true nếu hai số đó đổi chổ với nhau, false nếu hai số đó không đổi chổ với nhau. Việc sắp xếp này giúp “lưu vết” giải thuật nhằm phục vụ cho quá trình tái thực thi từng bước của giải thuật trên giao diện đồ họa với hàm *executeAStep* với tham số đầu vào là k thể hiện bước k đang được thực hiện. Vì các vòng lặp for, while hay do...while không thể tiếp nhận các lệnh có độ trễ về mặt thời gian như setTimeout hay setInterval nên ta cần sử dụng hàm đệ qui gọi lại các giá trị có liên quan tại một bước thứ k được lưu trong 3 mảng nêu trên là num1[k], num2[k] và swapped[k] làm tham số cho mỗi lần đề qui. Công việc của hàm *executeAStep* là đổi chổ hai số num1[k] và num2[k] nếu swapped[k] là true và không đổi chổ nếu là false. Sau đó thiết kế hàm đệ qui *executeAllStep* để thực hiện tuần tự tất cả các bước của giải thuật với phần đệ qui gọi đến hàm *executeAStep* và phần dừng sẽ thông báo mảng đã được sắp xếp theo thứ tự. Tiếp tục tăng k và gọi đệ qui hàm *executeAllStep*.

Về minh họa chuyển động, trước hết cần thiết kế phương thức *createBubble* khởi tạo một bong bóng có chứa giá trị số là giá trị đầu vào mà người dùng đã nhập. Bên cạnh đó cần thiết kế các phương thức: *MiniBubble* tạo ra các bong bóng con nổi lên theo bong bóng chính mỗi khi chúng động, *MoveUp* tịnh tuyến bong bóng lên trên 30px, *setColorCode* tô màu dòng code đang được thực thi trong giải thuật minh họa hiện trên màn hình, *percent* tính toán phần trăm hoàn thành của giải thuật (số bước đã thực hiện/tổng số bước) và hiển thị lên thanh tiến trình.

Về hiển thị dữ liệu, xây dựng một phần tử bảng HTML để lưu trữ lại hiện trạng của mảng sau mỗi lần lặp của giải thuật và hiển thị khi người dùng ấn vào biểu tượng bảng trên giao diện đồ họa. Người dùng có thể ẩn bảng bằng cách nhấn vào biểu tượng đóng trên giao diện.

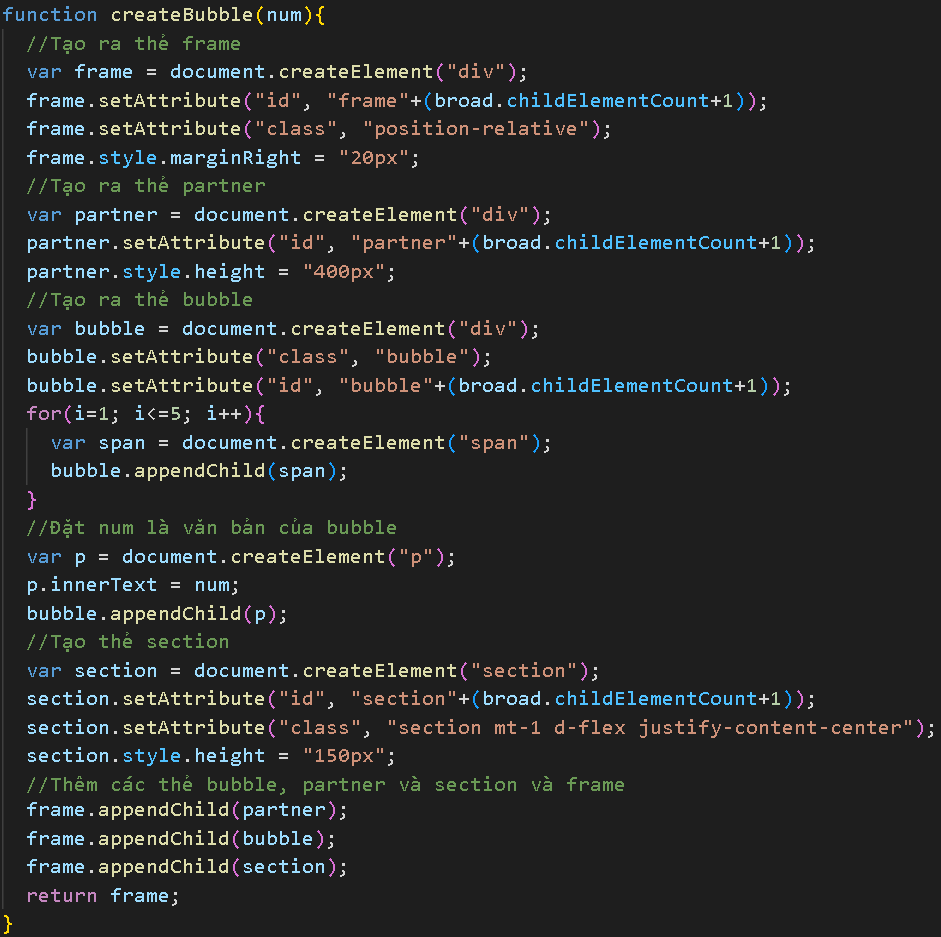
## THIẾT KẾ GIẢI THUẬT

### Các biến dùng chung

|  |  |  |  |
| --- | --- | --- | --- |
| **STT** | **Tên biến** | **Mô tả** | **Giá trị mặc định** |
| 1 | formNum | Tham chiếu đến form nhập một số |  |
| 2 | number | Giá trị người dùng nhập vào |  |
| 3 | formString | Tham chiếu đến form nhập một chuỗi |  |
| 4 | input | Giá trị chuỗi truyền vào |  |
| 5 | board | Tham chiếu đến phần tử HTML chứa các bong bóng của giải thuật |  |
| 6 | maxchild | Số phần tử tối đa của mảng được sắp xếp | 10 |
| 7 | arrayNum | Mảng được sắp xếp |  |
| 8 | speed | Trọng số dùng để nhân với các thông số cơ bản thành tốc độ xử lí của giải thuật | 2 |
| 9 | partnerHeight | Chiều cao của “partner” - thẻ div anh em với thẻ div chứa bong bóng trong giải thuật sắp xếp. Thay đổi theo từng loại thiết bị | 400 |
| 10 | pause | Biến trạng thái thể hiện trạng thái đang tạm dừng của giải thuật | true |

### Hàm createBubble

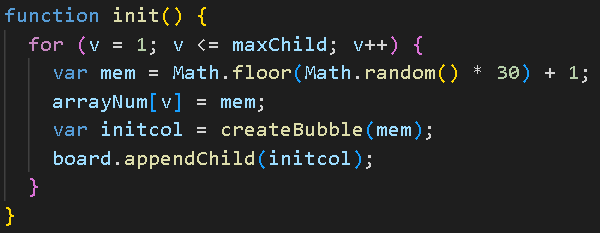
* Tham số: num (số nguyên)
* Mô tả: tạo ra một thẻ div với id là “frame” và có ba thẻ con lần lượt là div có id là “partner” dùng để kéo bong bóng chính nổi lên, thẻ div có class là “bubble” dùng để chứa bong bóng chính và thẻ section dùng để chứa các bong bóng con nổi theo bóng bóng chính. Tạo ra lần lượt 5 thẻ span và thêm chúng vào thẻ “bubble”, đặt num là nội dung của thẻ “bubble” và trả về thẻ “frame”.



***Hình 9******:*** Hàm createBubble

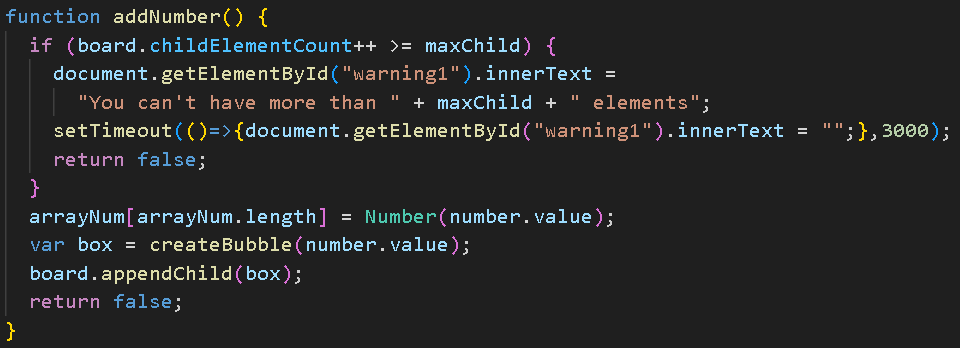
### Hàm init

* Mô tả: khởi tạo ngẫu nhiên 10 phần tử của mảng arrayNum bằng hàm random trong thư viện Math và gọi đến hàm createBubble để ra các bong bóng chứa giá trị ngẫu nhiên vừa có được.

***Hình 10******:*** Hàm init

### Hàm addNumber

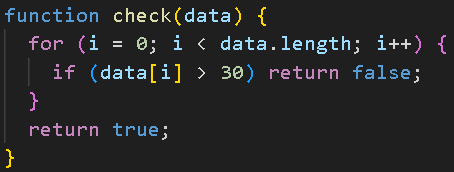
* Mô tả: được gọi mỗi khi người dùng ấn vào button “Add” tạo biểu mẫu “Add a number” trên giao diện. Hàm kiểm tra số phần tử hiện tại của board có nhỏ hơn số phần tử tối đa hay không. Nếu vẫn còn nhỏ hơn sẽ thêm giá trị đầu vào vào mảng arrayNum và truyền giá trị đó vào createBubble. Thêm thẻ được trả về từ hàm createBubble vào board.



***Hình 11******:*** Hàm addNumber

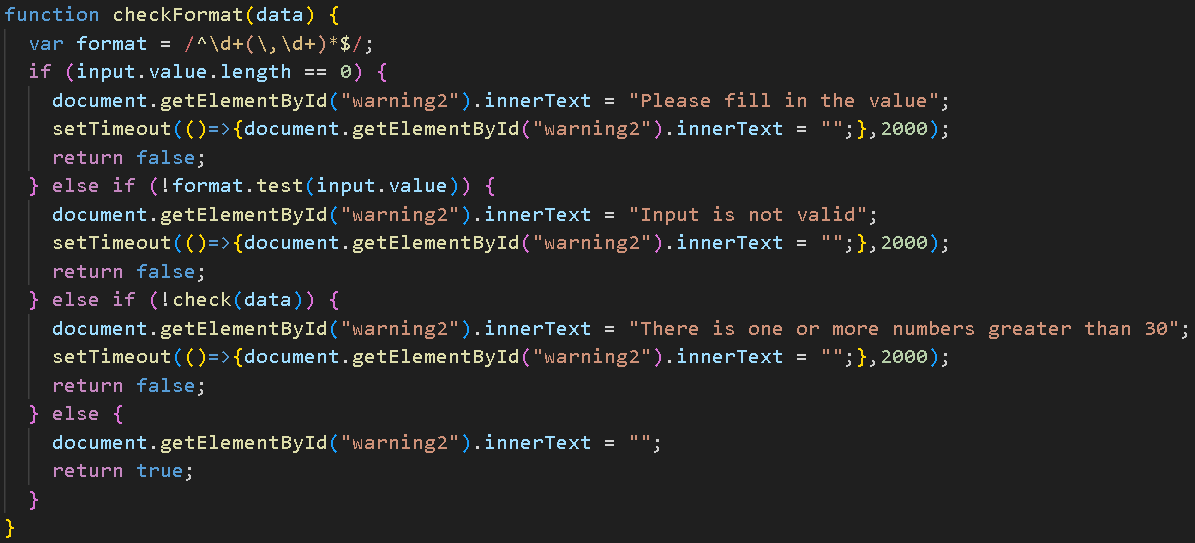
### Hàm check

* Tham số: data (mảng số nguyên).
* Mô tả: hàm duyệt qua mảng data. Tại mỗi phần tử, kiểm tra giá trị của chúng có lớn hơn 30, nếu có trả về false. Sau khi duyệt qua nếu không có giá trị nào lớn hơn 30 thì trả về true.

***Hình 12******:*** Hàm check

### Hàm checkFormat

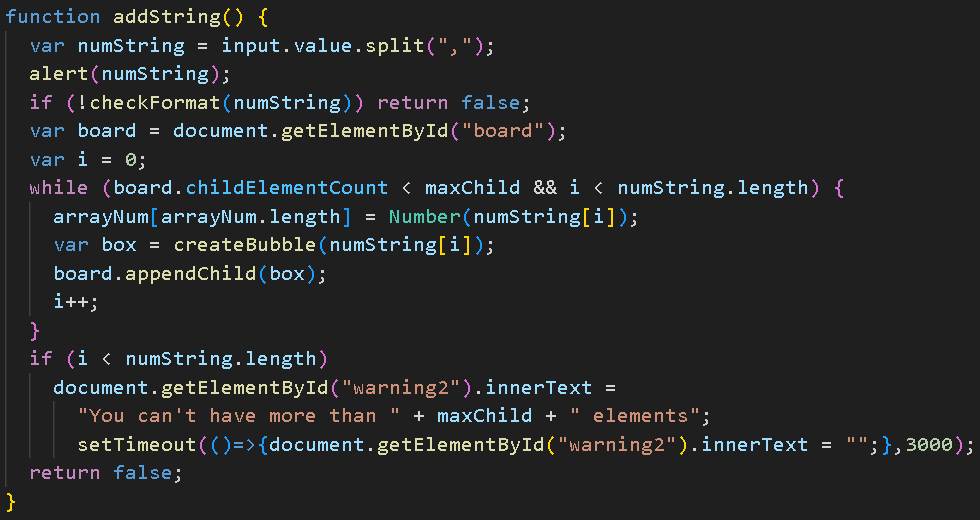
* Tham số: data (mảng số nguyên).
* Mô tả: lần lượt kiểm tra các định dạng của chuỗi dữ liệu nhập vào từ người dùng như kiểm tra nội dung rỗng, kiểm tra định dạng chuỗi hợp lệ, kiểm tra giá trị có lớn hơn 30 (gọi đến hàm check). Nếu tồn tại một vi phạm hàm hiển thị thông báo lỗi tương ứng và trả về giá trị false. Ngược lại, nếu không có vi phạm nào dữ liệu được xem là hợp lệ và hàm trả về giá trị true.



***Hình 13******:*** Hàm checkFormat

### Hàm addString

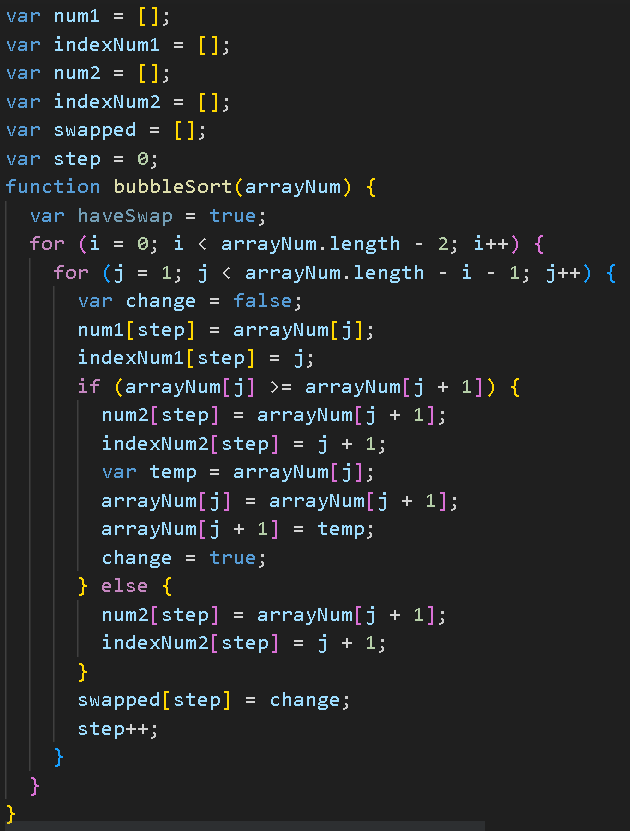
* Mô tả: hàm kiểm tra tính hợp lệ của dữ liệu nhận vào (gọi đến hàm *checkFormat*). Nếu hợp lệ, lần lượt thêm các giá trị đầu vào vào mảng *arrayNum* rồi gọi đến hàm *createBubble* để khởi tạo một bong bóng tương ứng. Thêm bong bóng vừa tạo vào khung chứa (*board*). Trong quá trình xử lý nếu mảng *arrayNum* đầy thì hàm sẽ hiển thị thông báo lên giao diện.



***Hình 14******:*** Hàm addString

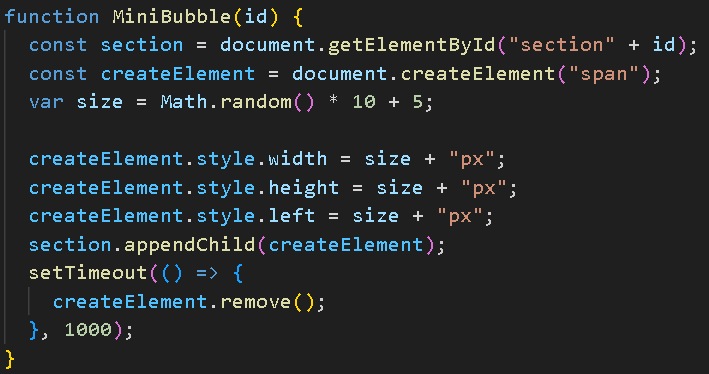
### Hàm bubbleSort

* Tham số: arrayNum (mảng số nguyên).
* Mô tả: sử dụng hai vòng lặp lồng nhau. Vòng lặp thứ nhất lần lượt duyệt qua n-1 phần tử của mảng arrayNum. Tại mỗi phần tử thứ i vòng lặp thứ hai sẽ tuần tự duyệt qua n-i phần tử của mảng arrayNum và đem phần tử A[j] so sánh với phần tử A[j+1]. Nếu A[j] >= A[j+1] thì hoán đổi giá trị của chúng và lưu lại giá trị hiện tại vào các mảng num1, num2, indexNum1, indexNum2 và swapped.

***Hình 15******:*** Hàm bubbleSort

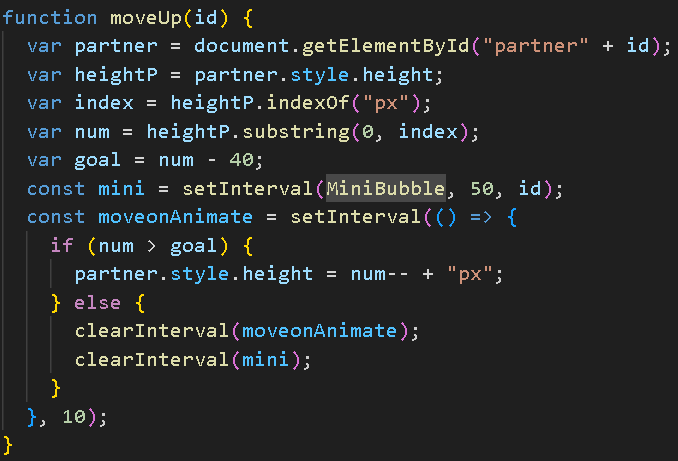
### Hàm MiniBubble

* Tham số: id (số nguyên).
* Mô tả: hàm tạo ra một bong bóng nhỏ nổi lên ngay sau bong bóng chính bằng cách tham chiếu đến thẻ section có id tương ứng với tham số đầu vào. Sau đó khởi tạo một thẻ span với 3 thuộc tính width, height và left ngẫu nhiên. Sau cùng chèn thẻ span vào thẻ section và cài đặt thời gian xóa thẻ span bằng hàm setTimeout (xóa sau 1 giây)[1].

***Hình 16******:*** Hàm MiniBubble

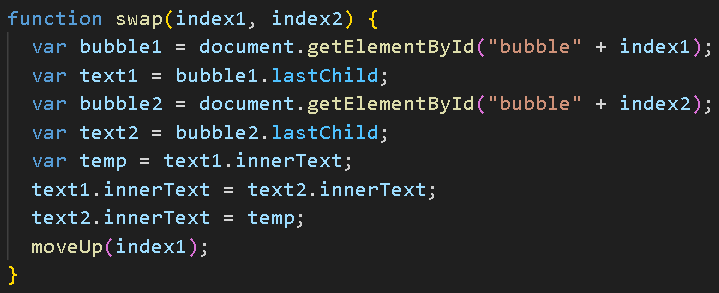
### Hàm moveUp

* Tham số: id (số nguyên).
* Mô tả: hàm thể hiện chuyển động đi lên của bong bóng chính bằng cách tham chiếu đến thẻ div “partner” có id tương ứng với tham số đầu vào. Sau đó sử dụng lệnh setInterval với thời gian lặp lại là 10 mi li giây để giảm dần chiều cao của thẻ “partner” tương ứng với việc kéo thẻ “bubble” phía dưới đi lên đồng thời cũng tạo ra các bong bóng nhỏ cùng nổi theo bằng hàm MiniBubble.

***Hình 17******:*** Hàm moveUp

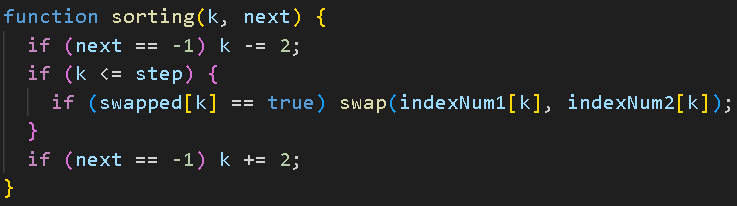
### Hàm swap

* Tham số: index1 (số nguyên), index2 (số nguyên).
* Mô tả: tham chiếu đến hai bong bóng có id tương ứng với giá trị index1 và index2 (cặp số đang được hoán đổi). Sau đó hoán đổi giá trị bên trong chúng và gọi đến hàm moveUp với tham số đầu vào là index1.

***Hình 18******:*** Hàm swap

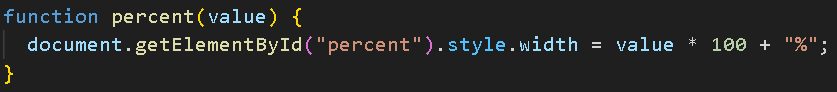
### Hàm sorting

* Tham số: k (số nguyên), next (số nguyên).
* Mô tả: hàm xử lí việc truy xuất hai phần tử được so sánh tại bước thứ k trong giải thuật. Kiểm tra giá trị next (thể hiện đối tượng gọi đến hàm này) nếu next=-1 thì đối tượng gọi đến là nút “Previous Step” trên giao diện. Tiến hành giảm giá trị k (bước sẽ được thực hiện) về bước trước đó đồng thời kiểm tra k<=số bước tối đa của giải thuật. Nếu không thì gọi đến hàm swap để hoán đổi giá trị của hai phần tử trong mảng arrayNum có chỉ số là indexNum1[k] và indexNum2[k].

***Hình 19******:*** Hàm sorting

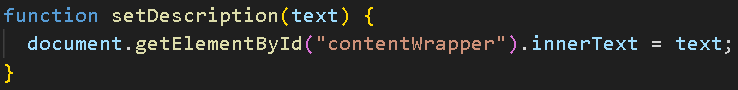
### Hàm percent

* Tham số: value (số nguyên).
* Mô tả: hàm tính toán phần trăm hoàn thành của giải thuật và đặt giá trị đó cho thanh tiến trình hiển thị lên giao diện.

***Hình 20******:*** Hàm percent

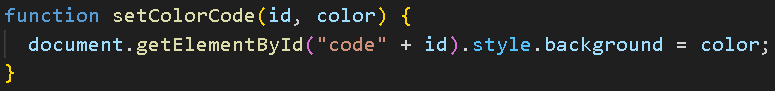
### Hàm setDescription

* Tham số: text (chuỗi)
* Mô tả: đặt nội dung cho mục lời mô tả trên giao diện là chuỗi text.

***Hình 21******:*** Hàm setDescription

### Hàm setColorCode

* Tham số: id (số nguyên), color (chuỗi).
* Mô tả: hàm thay đổi màu cho dòng mã có id tương ứng với tham số truyền vào thành màu color.

***Hình 22******:*** Hàm setColorCode

### Hàm setColorBubble

* Tham số: id (số nguyên), color (chuỗi).
* Mô tả: hàm thay đổi màu cho bong bóng có id tương ứng với tham số truyền vào thành màu color.

***Hình 23******:*** Hàm setColorBubble

### Hàm executeAStep

* Tham số: k (số nguyên), next (số nguyên).
* Mô tả: Trước hết gọi đến hàm percent với tham số truyền vào là (k/step). Tiếp đến, hàm so sánh giá trị của hai số trong phép so sánh tại bước k và đặt lời mô tả phù hợp với phép so sánh.

***Hình 24******:*** Hàm executeAStep

### Hàm createRowTable

* Tham số: tempArray (mảng số nguyên).
* Mô tả: hàm tạo ra và chèn một dòng mới vào trong bảng dữ liệu trên giao diện đồ họa. Bằng cách tham chiếu đến bảng dữ liệu có id là “data-table” và tạo ra các phần tử HTML như thẻ td và tr. Sử dụng vòng lặp duyệt qua mảng tempArray (mảng lưu trữ thứ tự hiện tại của các phần tử sau mỗi lần giải thuật duyệt qua toàn bộ mảng *arrayNum*), mỗi phần tử trong mảng tempArray ứng với một thẻ td. Với mỗi lần lặp, chèn thẻ td được tạo ra vào thẻ tr. Sau cùng chèn thẻ tr vào bảng dữ liệu “data-table”.

***Hình 25******:*** Hàm createRowTable

### Hàm executeAllStep

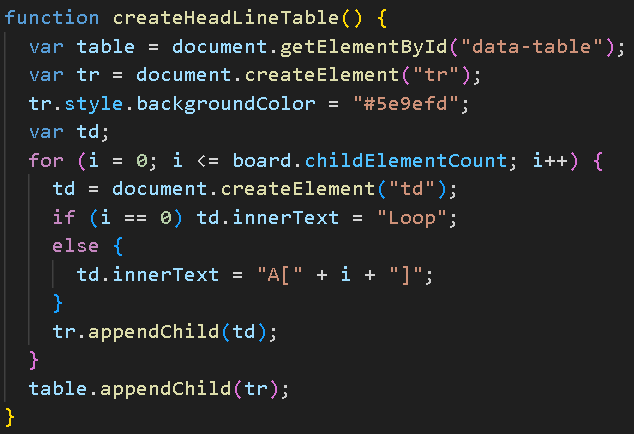
* Tham số: executeFirstStep (số nguyên).
* Mô tả: hàm thực hiện tuần tự tất cả các bước của giải thuật bằng cách kiểm tra tham số executeFirstStep. Nếu executeFirstStep=1 thì thực hiện bước đầu tiên của giải thuật sau đó kiểm tra biến pause xem người dùng có đang dừng giải thuật. Nếu pause==false tức giải thuật vẫn tiếp tục thì kiểm tra biến hiện tại k có lớn hơn tổng số bước của giải thuật (tức kiểm tra giải thuật đã hoàn thành hay chưa):
* Nếu k <= tổng số bước của giải thuật, thì thực hiện gọi “đệ qui” hàm executeAStep với tham số truyền vào là tham số k và tham số next=0 sau đó tăng k. Tiếp đến hàm kiểm tra giải thuật đã duyệt qua hết số phần tử của mảng *arrayNum* hay chưa, nếu đã duyệt qua hết thì nạp mảng hiện tại (theo đúng thứ tự các bong bóng trên giao diện) vào một mảng tạm và gọi hàm createRowTable với tham số là mảng đó để tạo ra một dòng dữ liệu của bảng dữ liệu trên giao diện.
* Nếu k > step, hiển thị lời mô tả mảng đã được sắp xếp hoàn tất và dừng gọi đệ qui hàm executeAStep.



***Hình 26******:*** Hàm executeAllStep

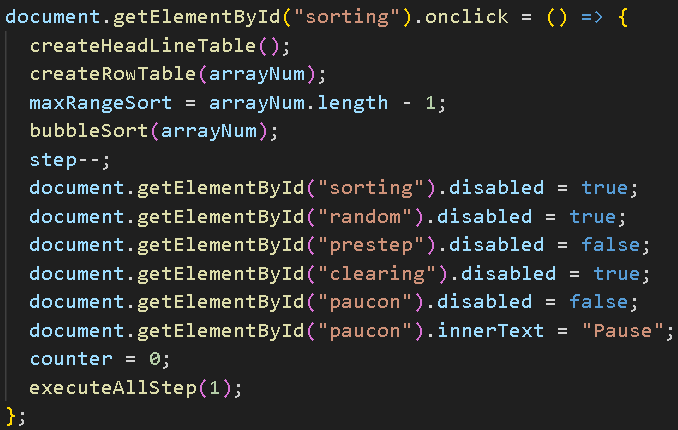
### Hàm createHeadLineTable

* Mô tả: hàm tạo ra dòng đầu tiên trong bảng dữ liệu chứa các trường dữ liệu gồm số lần lặp của giải thuật và các phần tử của mảng.

***Hình 27******:*** Hàm createHeadLineTable

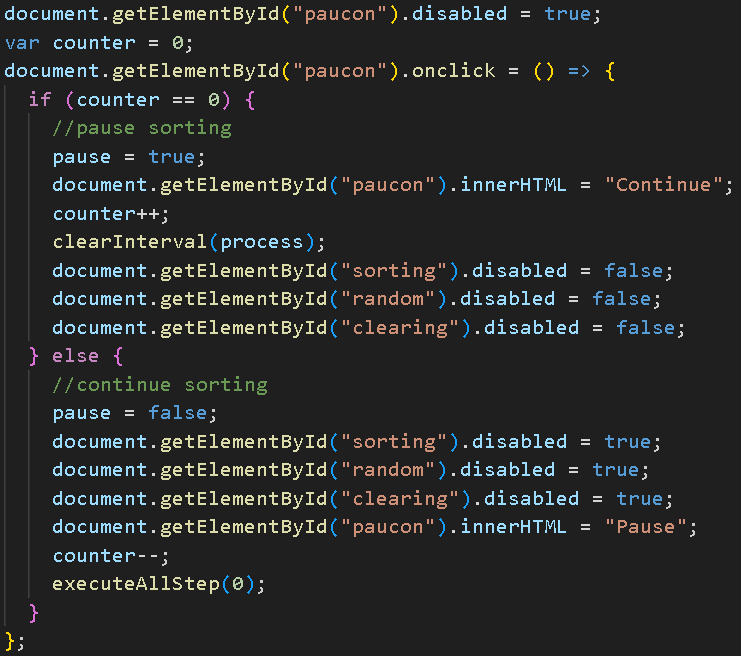
### Xử lý sự kiện cho nút “Sort”

* Mô tả: khi người dùng ấn vào nút “Sort” trên giao diện, website gọi đến hàm createHeadLineTable để tạo ra số cột tương ứng với số phần tử được sắp xếp sau đó gọi đến hàm createRowTable với tham số là mảng arrayNum để chèn mảng hiện tại vào bảng dữ liệu và gọi đến hàm bubbleSort để sắp xếp mảng arrayNum. Sau cùng điều chỉnh các

***Hình 28******:*** Xử lý sự kiện cho nút “Sort”

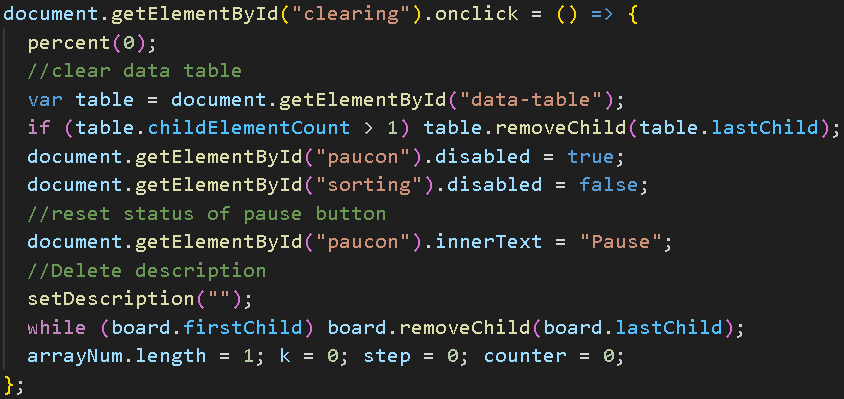
### Xử lý sự kiện cho nút “Pause” hoặc “Continue”

* Mô tả: khi người dùng nhấn vào nút “Pause” hoặc nút “Continue” trên giao diện website kiểm tra biến trạng thái counter. Nếu counter=0 tức giải thuật đang chạy, ta cập nhật giá trị cho biến pause=true để ngưng việc gọi đệ qui ở hàm executeAllStep đang chạy và đặt lại nội dung cho nút “Pause” thành “Continue” đồng thời tăng giá trị counter. Sau đó điều chỉnh trạng thái bật tắt của các nút điều khiển trên giao diện. Ngược lại khi counter=1, cập nhật giá trị cho pause=false, thay đổi nội dung nút “Continue” thành “Pause” và điều chỉnh trạng thái các nút trên giao diện. Sau cùng gọi hàm executeAllStep để tiếp tục thực thi giải thuật.

***Hình 29******:*** Xử lý sự kiện cho nút “Pause” hoặc “Continue”

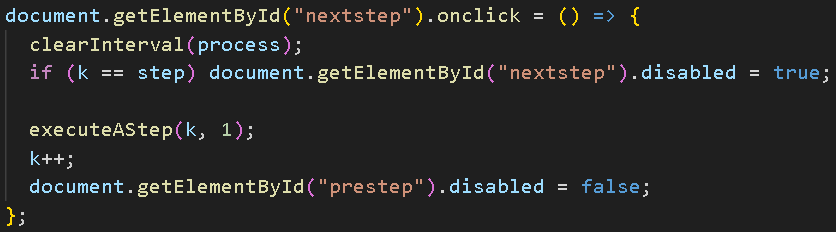
### Xử lý sự kiện cho nút “Clear”

* Mô tả: gọi hàm percent với tham số bằng 0 để làm mới thanh tiến trình trên giao diện sau đó gọi đến bảng dữ liệu và xóa toàn bộ con của nó rồi cập nhật lại trạng thái bật tắt của các nút điều khiển. Cuối cùng, xóa toàn bộ các bong bóng trong khung chứa (board).

***Hình 30******:*** Xử lý sự kiện cho nút “Clear”

### Xử lý sự kiện cho nút “Next Step”

* - Mô tả: khi người dùng ấn vào nút “Next step” trên giao diện thì website sẽ hủy việc gọi đệ qui hàm executeAStep thông qua biến process. Sau đó kiểm tra xem bước hiện tại (k) có bằng tổng số bước hay chưa, nếu bằng tức giải thuật đã đến bước cuối cùng sau khi nhấn nút “Next step” ta cần khóa nút này lại về không còn bước tiếp theo nào. Tiếp đến gọi hàm executeAStep thực hiện bước thứ k.



***Hình 31******:*** Xử lý sự kiện cho nút “Next step”

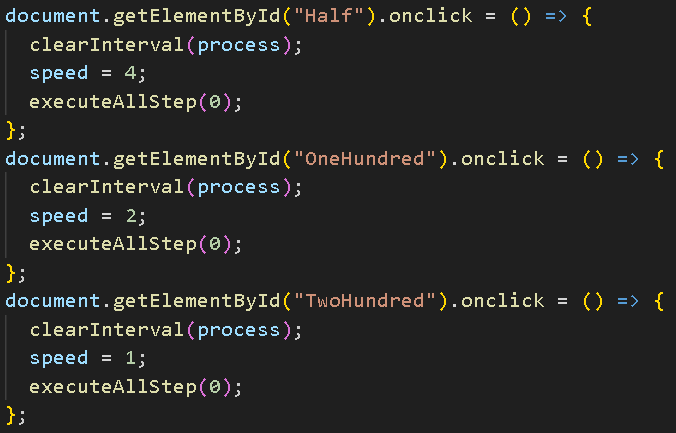
### Xử lý sự kiện cho nút “Random”

* Mô tả: gọi đến hàm percent với tham số bằng 0 để làm mới thanh tiến trình sau đó đặt lại trạng thái bật tắt của các nút cho phù hợp cũng như xóa nội dung trong phần mô tả. Tiếp theo, xóa tất cả các bong bóng trong khung chứa và gọi đến hàm init để khởi tạo lại ngẫu nhiên các bong bóng mới. Sau cùng, xóa các dòng trong bảng dữ liệu và gọi đến hàm bubbleSort.

***Hình 32******:*** Xử lý sự kiện cho nút “Random”

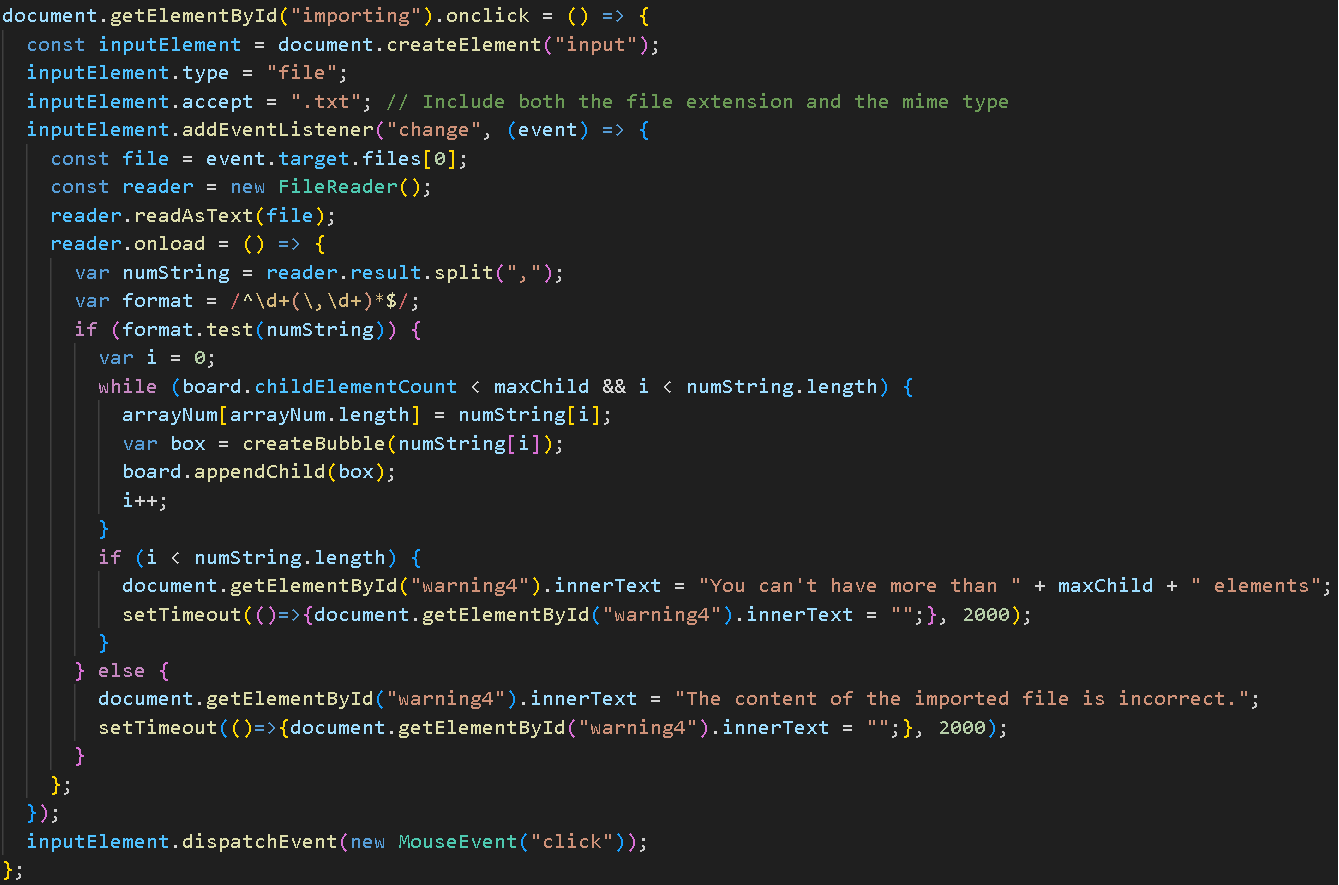
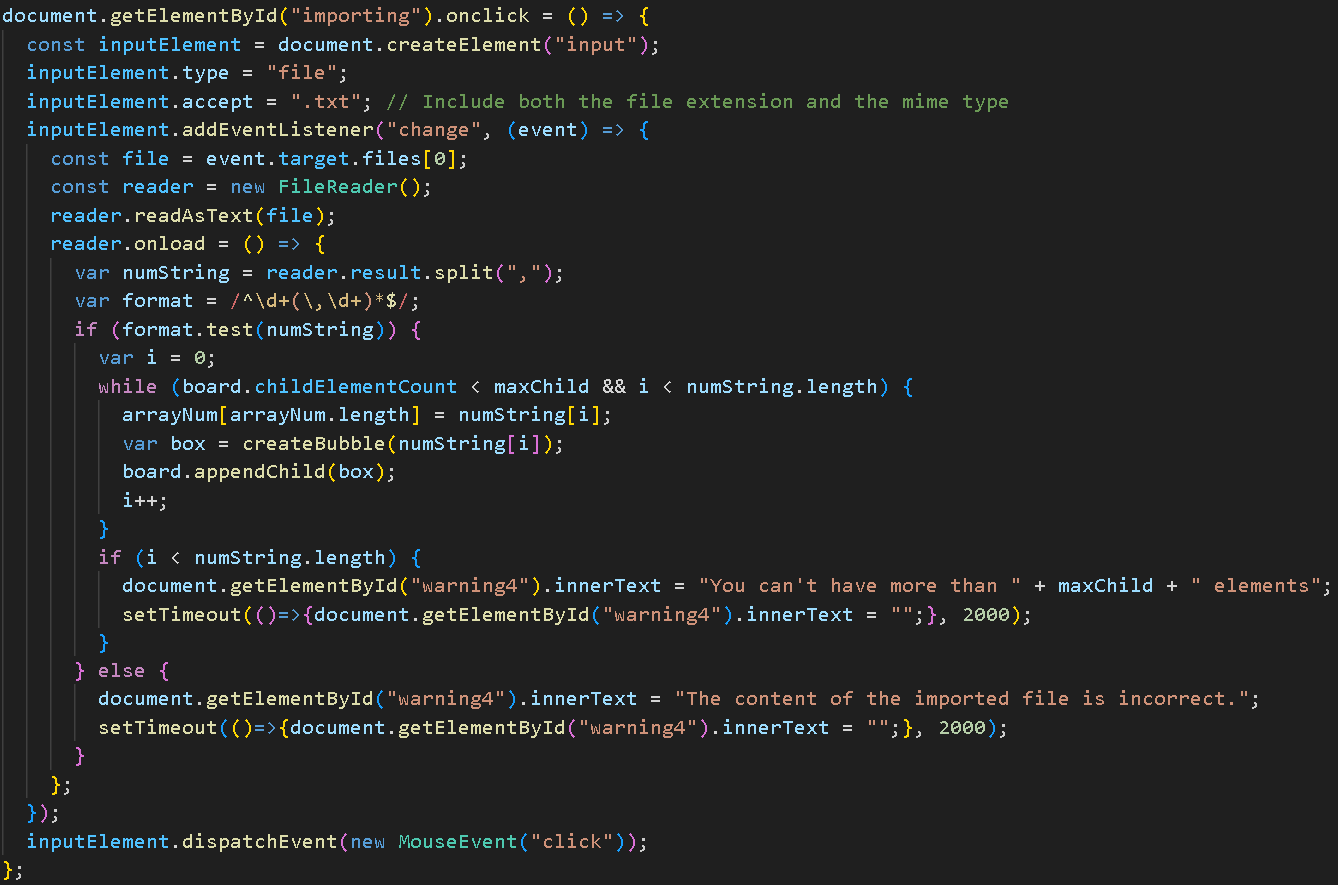
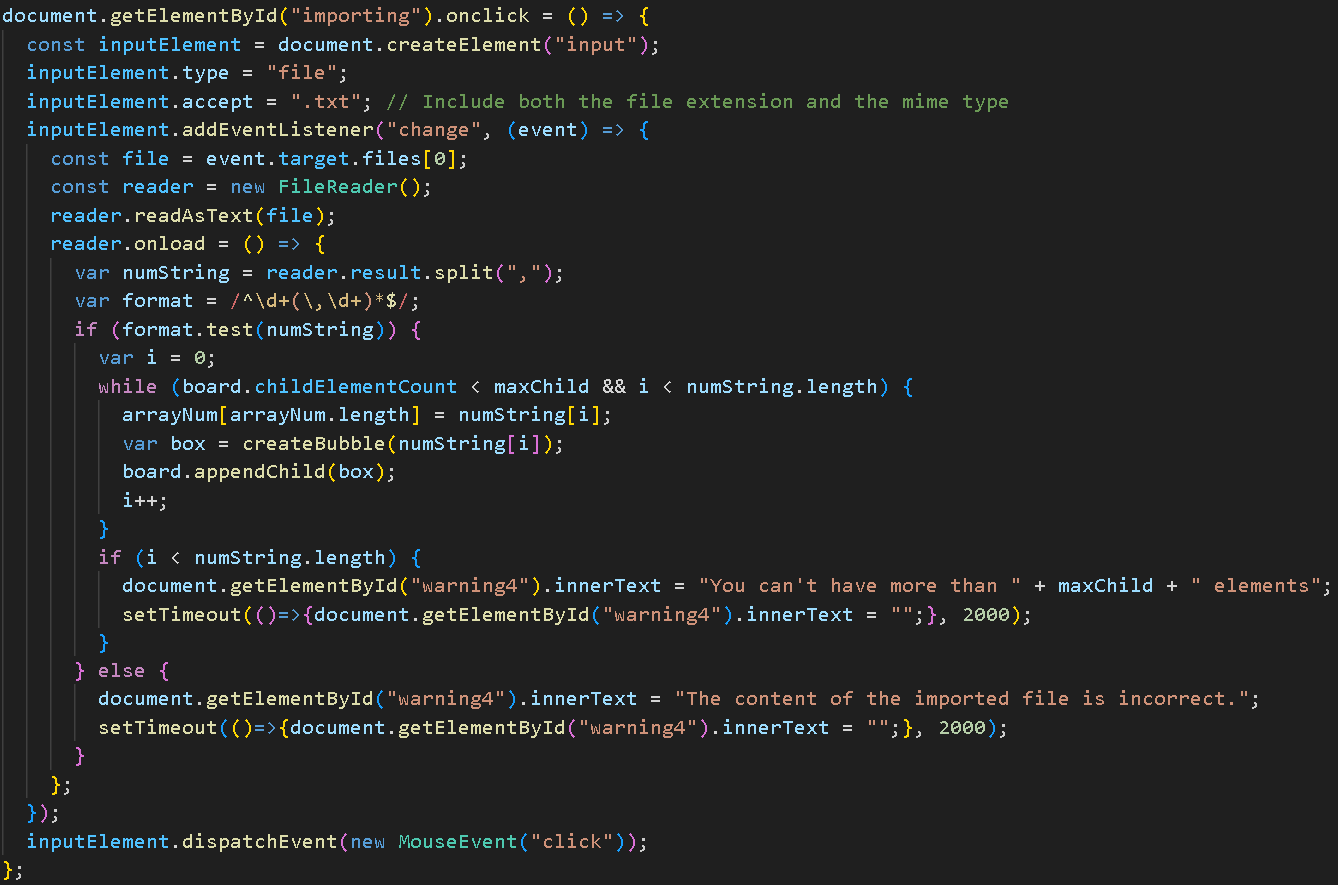
### Xử lý sự kiện cho các nút tốc độ

* Mô tả: điều chỉnh giá trị của biến speed và gọi hàm executeAllStep với tham số bằng 0 để thực hiện giải thuật tiếp tục.

***Hình 34******:*** Xử lý sự kiện cho các nút tốc độ

### Xử lý sự kiện cho nút “Import”

* Mô tả: tạo ra thẻ input với type=”file”, định dạng file nhận vào là “txt” và đặt sự kiện nhất nút cho thẻ input này. Khi ấn vào mở một trình duyệt chọn file trên thiết bị. Sau khi kiểm tra nội dung file nhận vào hợp lệ website sẽ thêm các tham số nhận vào thành các bong bóng và chèn vào khung chứa. Nếu định dạng không hợp lệ sẽ thông báo đến người dùng.



***Hình 33******:*** Xử lý chức năng cho nút “Import”

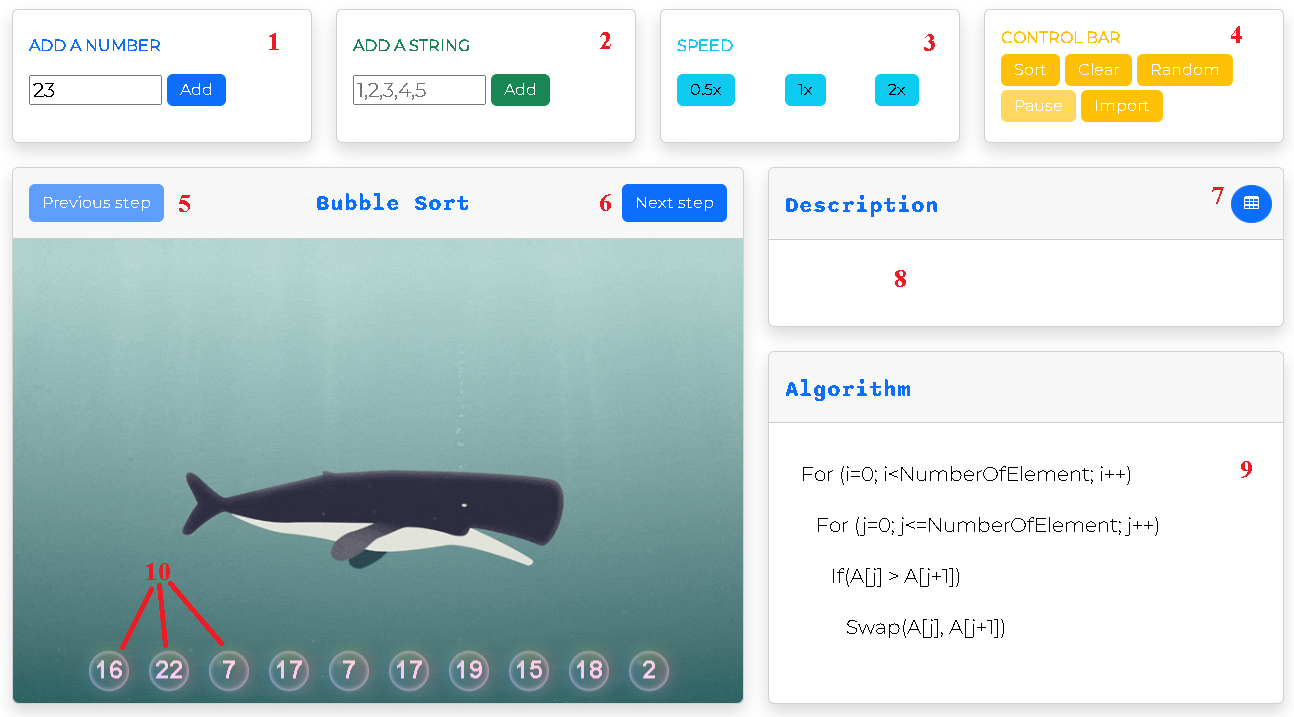
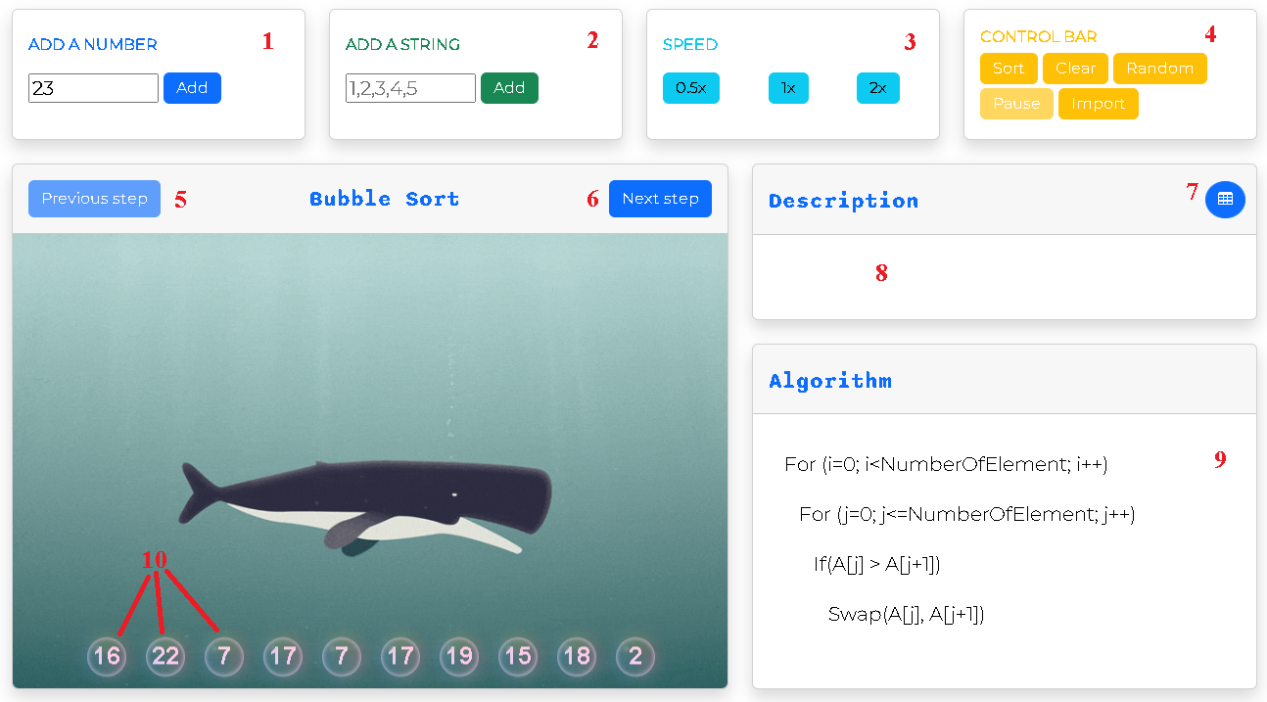
### Xử lý sự kiện cho nút “Previous Step”

* Mô tả: khi vừa truy cập vào trang nút “Previous step” sẽ được mặc định khóa lại vì giải thuật chưa bắt đầu. Khi người dùng nhấn vào, cập nhật lại giá trị cho biến pause=true đồng thay đổi màu sắc của 2 bong bóng đang được so sánh và gọi đến hàm.



***Hình 35******:*** Xử lý sự kiện cho nút “Previous step”

## GIỚI THIỆU CHƯƠNG TRÌNH

***Hình 36******:*** Giao diện chính của website minh họa Bubble Sort

|  |  |  |
| --- | --- | --- |
| **STT** | **Thẻ HTML sử dụng** | **Diển giải** |
| 1 | form, input, button | Form nhập giá trị số nguyên |
| 2 | form, input, button | Form nhập giá trị chuỗi số nguyên |
| 3 | form, button | Form điều chỉnh tốc độ thực hiện giải thuật |
| 4 | form, input, button | Form điều chỉnh các thao tác với giải thuật |
| 5 | button | Nút lùi giải thuật về một bước trước đó |
| 6 | button | Nút tăng giải thuật lên bước kế tiếp |
| 7 | button | Nút mở bảng dữ liệu |
| 8 | div | Khung chứa lời mô tả giải thuật |
| 9 | div | Khung chứa mã lệnh giải thuật |
| 10 | div, span | Các bong bóng chứa các phần tử được sắp xếp |

# CHƯƠNG 4: KẾT LUẬN ĐÁNH GIÁ

## KẾT QUẢ ĐẠT ĐƯỢC

### Về kiến thức

* Ôn lại và nắm vững kiến thức về cấu trúc dữ liệu, giải thuật sắp xếp nổi bọt.
* Ôn lại và nắm vững các kiến thức về lập trình web như HTML, CSS và JS.
* Thiết kế giao diện đồ họa trong sáng, dễ hiểu.
* Kết hợp thuật toán chạy đồng thời với các hoạt ảnh minh họa tạo ra từ Javascript.

### Về kỹ năng

* Có khả năng mô hình hóa các yêu cầu từ một bài toán thực tế được đặt ra.
* Sử dụng, kết hợp các kiến thức đã được giảng dạy để giải quyết vấn đề thực tế.
* Sử dụng tốt các công cụ, thư viện hỗ trợ lập trình như Visual Studio Code, Bootstrap.

### Về thái độ

* Nghiên cứu và thực hiện niên luận cơ sở theo đúng tiến độ đề ra của giáo viên hướng dẫn.
* Tham khảo ý kiến, trao đổi tích cực với giáo viên hướng dẫn đề hoàn thành tốt sản phẩm đầu ra.

## HẠN CHẾ

* Chưa thực hiện được giao diện website 3D.
* Chưa thể kết xuất được kết quả thực thi giải thuật ra các định dạng như pdf, png, jpg,..
* Chưa thể phát triển thêm các giao diện phụ khác như làm bài tập, giới thiệu hệ thống, bài giảng điện tử,...

## NGUYÊN NHÂN

* Việc xây dựng website 3 chiều cần thời gian lâu dài để chuẩn bị các công đoạn như vẽ mô hình, lập trình chuyển động cho mô hình cũng như kết hợp các mô hình lại với nhau tạo nên bối cảnh cho website. Vì nguyên nhân thời gian nên module này đã được chuyển sang giai đoạn phát triển trong tương lai.
* Ở giai đoạn hiện tại website chủ yếu tập trung vào việc mô tả hình ảnh, chuyển động và tính đúng đắn của giải thuật.

## HƯỚNG PHÁT TRIỂN

* Phát triển thêm các giao diện hỗ trợ người dùng ôn tập kiến thức sau khi đã trải nghiệm với giải thuật.
* Xây dựng giao diện 3 chiều cho website để tăng tính sống động, thực tế cho giải thuật.
* Cung cấp các bài giảng điện tử có liên quan đến giải thuật.

# PHỤ LỤC

*[Hình 1 . Lưu đồ giải thuật sắp xếp nổi bọt 6](#_Toc7173)*

*[Hình 2 : Mảng A sau lần lặp đầu tiên 8](#_Toc9762)*

*[Hình 3 : Mảng A sau lần lặp thứ 1 8](#_Toc17908)*

*[Hình 4 : Mảng A sau lần lặp thứ 2 8](#_Toc17942)*

*[Hình 5 : Mảng A sau lần lặp thứ 3 8](#_Toc24958)*

*[Hình 6 : Mảng A sau lần lặp thứ 4 9](#_Toc17008)*

*[Hình 7 : Mảng A sau lần lặp thứ 5 đã có thứ tự tăng dần 9](#_Toc23065)*

*[Hình 8 : Kết quả vận dụng giải thuật vào đề tài 9](#_Toc24055)*

*[Hình 9 : Hàm createBubble 12](#_Toc24946)*

*[Hình 10 : Hàm init 13](#_Toc30036)*

*[Hình 11 : Hàm addNumber 13](#_Toc17420)*

*[Hình 12 : Hàm check 13](#_Toc6353)*

*[Hình 13 : Hàm checkFormat 14](#_Toc28969)*

*[Hình 14 : Hàm addString 14](#_Toc29536)*

*[Hình 15 : Hàm bubbleSort 15](#_Toc14829)*

*[Hình 16 : Hàm MiniBubble 16](#_Toc8927)*

*[Hình 17 : Hàm moveUp 17](#_Toc3009)*

*[Hình 18 : Hàm swap 17](#_Toc26010)*

*[Hình 19 : Hàm sorting 18](#_Toc10742)*

*[Hình 20 : Hàm percent 18](#_Toc16658)*

*[Hình 21 : Hàm setDescription 18](#_Toc10922)*

*[Hình 22 : Hàm setColorCode 18](#_Toc1533)*

*[Hình 23 : Hàm setColorBubble 19](#_Toc15088)*

*[Hình 24 : Hàm executeAStep 19](#_Toc21901)*

*[Hình 25 : Hàm createRowTable 20](#_Toc28203)*

*[Hình 26 : Hàm executeAllStep 21](#_Toc3061)*

*[Hình 27 : Hàm createHeadLineTable 22](#_Toc7355)*

*[Hình 28 : Xử lý sự kiện cho nút “Sort” 22](#_Toc13354)*

*[Hình 29 : Xử lý sự kiện cho nút “Pause” hoặc “Continue” 23](#_Toc5596)*

*[Hình 30 : Xử lý sự kiện cho nút “Clear” 24](#_Toc26854)*

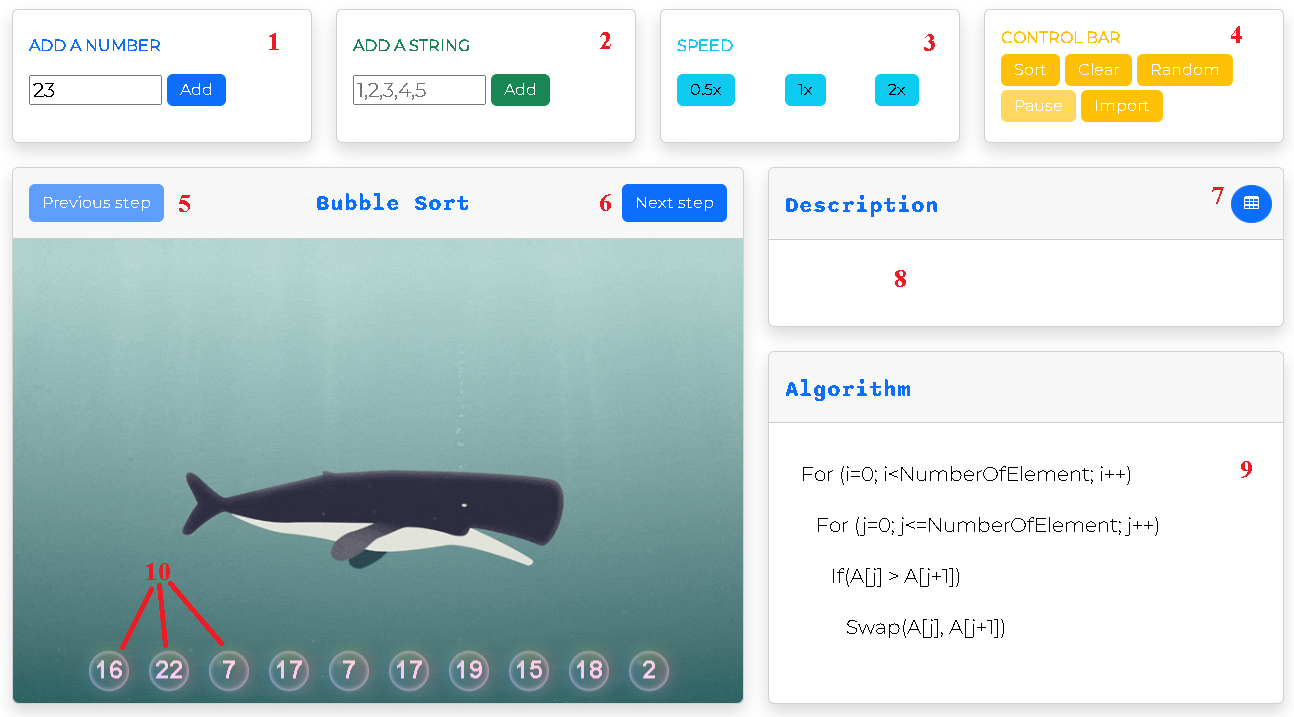
*[Hình 31 : Xử lý sự kiện cho nút “Next step” 24](#_Toc10442)*

*[Hình 32 : Xử lý sự kiện cho nút “Random” 25](#_Toc18163)*

*[Hình 34 : Xử lý sự kiện cho các nút tốc độ 25](#_Toc18730)*

*[Hình 33 : Xử lý chức năng cho nút “Import” 26](#_Toc21670)*

*[Hình 35 : Xử lý sự kiện cho nút “Previous step” 27](#_Toc21157)*

*[Hình 36 : Giao diện chính của website minh họa Bubble Sort 27](#_Toc682)*

# TÀI LIỆU THAM KHẢO

*Kênh Youtube “Online Tutorial”, Animated Realistic Bubbles using HTML, CSS & Vanilla Javascript. <https://www.youtube.com/watch?v=noC22oMVb44>*

*Nguyễn Văn Linh, Phân tích và Thiết kế thuật toán, Chương 2, trang 37, Nhà xuất bản Đại học Cần Thơ.*

*Võ Huỳnh Trâm, silde bài giảng Phân tích và Thiết kế thuật toán, Chương 1, Slide 26.*

*Wikipedia, Thuật toán sắp xếp. <https://vi.wikipedia.org/wiki/Thu%E1%BA%ADt_to%C3%A1n_s%E1%BA%AFp_x%E1%BA%BFp>*

**-- Hết --**