MỤC LỤC

[I. Scratch 3](#_Toc152087842)

[II. C++ 3](#_Toc152087843)

[a. Khái niệm: 3](#_Toc152087844)

[b. Cấu trúc cơ bản: 3](#_Toc152087845)

[d. Các thuật toán: 9](#_Toc152087846)

[e. Các câu lệnh vặt hàm int main() 13](#_Toc152087847)

[f. Lưu ý: 13](#_Toc152087848)

[III. HTML, CSS 13](#_Toc152087849)

[a. Khái niệm: 13](#_Toc152087850)

[b. Cấu trúc cơ bản: 13](#_Toc152087851)

[c. Các thẻ: 13](#_Toc152087852)

[d. Các thuộc tính 16](#_Toc152087853)

[e. Các kĩ năng và lưu ý: 22](#_Toc152087854)

[f. Các kĩ năng thiết kế 23](#_Toc152087855)

[g. Các nguồn tài nguyên: 25](#_Toc152087856)

[h. Lệnh viết tắt: 26](#_Toc152087857)

[i. Các đơn vị 26](#_Toc152087858)

[IV. Javascript 27](#_Toc152087859)

[a. Khái niệm 27](#_Toc152087860)

[b. Các loại biến 27](#_Toc152087861)

[c. Toán tử 27](#_Toc152087862)

[d. Câu lệnh 29](#_Toc152087863)

[e. Các sự kiện 32](#_Toc152087864)

[f. Các khái niệm: 32](#_Toc152087865)

[g. Các kĩ năng, lưu ý: 35](#_Toc152087866)

[h. DOM 36](#_Toc152087867)

[i. Github 37](#_Toc152087868)

[j. Local storage(Bộ nhớ cục bộ)(LS) 38](#_Toc152087869)

1. Scratch
2. C++
   1. Khái niệm:

* Là một ngôn ngữ lập trình kiểu tĩnh, dữ liệu trừu tượng, hỗ trợ lập trình hướng đối tượng và thủ tục.
  1. Cấu trúc cơ bản:

#include <bits/stdc++.h>

using namespace std;

int main()

{

}

* 1. **Các câu lệnh:**
* Khai báo biến:

+) Thứ nhất: Kiểu dữ liệu

* Int: Số nguyên, giới hạn 109.
* Long long: Số nguyên, giới hạn 1018.
* Long double: Số thực, giới hạn 1015.
* Bool: Kiểu logic(đúng hay sai)
* Char: Kiểu kí tự(ko chứa các kí tự rỗng như dấu cách)
* String: Kiểu xâu kí tự(có thể chứa các kí tự rỗng)
* Const: Kiểu chỉ đọc(không thể gán hay thay đổi giá trị của biến)

...

+) Thứ hai: Tên biến (không dấu, không có cách, không trùng với các tên hàm hay các lệnh khác)

VD: int n; long double tong;  
 long long a; bool check;

* Cin: Dùng để xác định đầu vào của bài toán, cụ thể là biến

+) Cấu trúc: cin>>(<tên biến>);

VD: cin>>a;

cin>>n;

+) Các trường hợp đặc biệt

* Đối với xâu kí tự: khi dùng câu lệnh thông thường thì khi nhập đến một kí tự rỗng chúng sẽ tự cắt đi phần sau kí tự rỗng đó. Lúc này để nhập được kí tự rỗng vào trong xâu, bắt buộc phải dùng getline(cin,(<tên biến>));

VD: getline(cin,s);

* Cout: Dùng để xuất ra đầu ra của bài toán, cụ thể là biến

+) Cấu trúc: cout<<(<tên biến>);

VD: cout<<a; cout<<n;

+) Các trường hợp đặc biệt:

* Đối với các số thập phân: Nếu như đầu ra được xác định là một số thập phân hay số thực, chúng sẽ mặc định chỉ in ra 1 chữ số đằng sau dấu phẩy, để thay đổi, sử dụng câu lệnh

cout<<fixed<<setprecision((<số nguyên>));

VD: cout<<fixed<<setprecision(3);

* If và else: Là câu lệnh thuộc cấu trúc điều khiển, giống như nếu như này thì làm như này, nếu thế kia thì làm như kia.

+) Cấu trúc:

if((<điều kiện>))

{

}

else

{

}

+) Các kiểu điều kiện: Chủ yếu dùng để so sánh giữa hai số liệu

* Lớn hơn: sử dụng dấu “>”.
* Bé hơn: sử dụng dấu “<”.
* Bằng: sử dụng cặp dấu “==”.
* Lớn hơn hoặc bằng: sử dụng cặp dấu “>=”.
* Bé hơn hoặc bằng: sử dụng cặp dấu “<=”.
* Khác: sử dụng cặp dấu “!=”.

+) **Lưu ý:**

* Viết lệnh nếu chồng: Trong vế else chúng ta có thể viết thêm vế if trong vế else;

VD:

if()

{

}

else if()

{

}

* For: Là câu lệnh thuộc cấu trúc lặp, có 3 giá trị tương tự như 1 hàm, các giá trị được quy ước theo tuần tự: biến, điều kiện tiếp tục, câu lệnh thực hiện mỗi khi hoàn thành vòng lặp.

+) Cấu trúc:

for(long long i=1;i<=n;i++)

{

}

* While: Là câu lệnh thuộc cấu trúc lặp giống for, tuy nhiên chỉ có 1 giá trị duy nhất là điều kiện tiếp tục, vì vậy nên while sẽ linh động hơn so với for.

+) Cấu trúc:

while(i<=n)

{

i++;

}

* Câu lệnh về mảng một chiều: Là câu lệnh dùng để khai báo mảng, sử dụng mảng, có giới hạn số phần tử là 107
* Khai báo mảng: giống như khai báo 1 biến, chỉ khác là có thêm số phần tử giới hạn có thể chứa

VD: long long a[1000];

* Sử dụng mảng: Để sử dụng mảng, chúng ta lấy phần tử của chúng bằng cách nhập lệnh: (<tên mảng>)[(<vị trí trong mảng>)].

VD: a[5];

* + **Lưu ý**: phần tử trong mảng được coi như là 1 biến.
  + Cấu trúc nhập và xuất mảng:

for(long long i=1;i<=n;i++)

{

cin>>a[i];

}

* + Sắp xếp mảng: dùng hàm sort để sắp xếp.
* Tăng dần: sort(<tên mảng + vị trí đầu>, <tên mảng + vị trí cuối>, hàm so sánh)
* VD:
  + sort(arr, arr+n): Sắp xếp mảng tên ‘arr’, theo thứ tự tăng dần, bắt đầu từ phần tử thứ 0 đến phần tử thứ n-1
  + sort(a+1,a+1+n): Sắp xếp mảng tên ‘a’, theo thứ tự tăng dần, bắt đầu từ phần tử thứ 1 đến phần tử thứ n.
* Giảm dần: sort(<tên mảng + vị trí đầu>, <tên mảng + vị trí cuối>, greater <(kiểu dữ liệu)>());
* VD: sort(a+1,a+1+n, greater <long long>()): Sắp xếp mảng mảng tên ‘a’ theo thứ tự giảm dần, theo kiểu dữ liệu long long
* So sánh tự tạo: sort(<tên mảng + vị trí đầu>, <tên mảng + vị trí cuối>, <tên\_hàm\_so\_sánh>);
  + Cách tạo hàm so sánh:
    - Khai báo hai tham số sử dụng để so sánh.
    - Sử dụng lệnh return cùng với toán tử so sánh để sắp xếp.
    - Dấu <: sắp xếp tăng dần(tử nhỏ - lớn)
    - Dấu >: sắp xếp giảm dần(từ lớn – nhỏ)
* **Lưu ý:** Đối với dữ liệu giảm dần, dữ liệu chứa của mảng phải trùng khớp với kiểu dữ liệu được khai báo trong hàm sort
* Câu lệnh về mảng hai chiều: Cũng là mảng nhưng được trình bày như một ma trận bao gồm cột và hàng, tích số cột và hàng có giới hạn là 107
  + Khai báo mảng: Giống mảng 1 chiều, chỉ khác là chúng có hai chỉ số về kích thước mảng khác nhau, bao gồm
* Giới hạn đầu tiên: số hàng
* Giới hạn thứ hai: số cột
* VD: int arr[105] [105];
* Sử dụng mảng: tương tự với mảng một chiều, ta sử dụng cấu trúc: (<tên mảng>)[(<vị trí hàng trong mảng>)][<vị trí cột trong mảng>].
* Cấu trúc nhập và xuất mảng:

for(long long i=1;i<=n;i++)

{

for(long long j=1; j<=m; j++)

{

cin>>a[i];

}

}

* Câu lệnh về xâu kí tự:

+) Số lượng phần tử xâu: để xác định kích cỡ của xâu

VD: s.length() hoặc s.size()

+) Lệnh tìm xâu con trong xâu cha: Dùng để xác định vị trí của xâu con trong xâu cha.

VD: s.find(tin) dùng để tìm theo thứ tự trái qua phải

s.rfind(tin) dùng để tìm theo thứ tự từ phải qua trái

+) Phép gán: Khác với mảng, ta có thể gán trực tiếp hai xâu với nhau

<xâu 1>=<xâu 2>

+) Phép ghép xâu: Sử dụng dấu “+” giống như phép tính cộng

+) Thứ tự lớn bé: theo thứ tự của từ điển, có thể áp dụng các phép toán giống như các số.

VD: “a” < “ab” < “b” < “bcd”

+) Lệnh xác định xâu rỗng: trả về kết quả là true nếu xâu rỗng và ngược lại

VD: s.empty().

+) Lệnh đổi chỗ: dùng để đổi chỗ hai xâu

VD: s.swap(t): đổi chỗ xâu s và t.

+) Lệnh xác định phần tử đầu và cuối: dùng để xác định phần tử đầu cũng như phần tử cuối của xâu.

VD: s.front(): in ra phần tử đầu tiên

s.back(): in ra phần tử cuối cùng

+) Lệnh lấy xâu con: dùng để lấy xâu con từ xâu cha.

VD: s.substr(5,3) trong đó 5 là vị trí, 3 là số kí tự

+) Lệnh chèn xâu: dùng để chèn xâu 2 vào xâu 1

VD: s.insert(8,st) trong đó 8 là vị trí, st là xâu 2.

+) Xoá xâu: dùng để xoá 1 đoạn xâu

VD: s.erase(0,4) trong đó cả hai số này là vị trí đầu và cuối của xâu cần xoá

+) Lệnh thay thế xâu vào xâu: dùng để thay 1 đoạn xâu này bằng 1 đoạn xâu khác

VD: s.replace(9,5,str) trong đó số 9 là vị trí đầu tiên của phần tử để thay, 5 là số phần tử cần được loại bỏ kể từ vị trí đầu tiên, str là xâu thay thế.

+) Lệnh auto x:n : tự động định nghĩa kiểu dữ liệu x theo kiểu dữ liệu của n, tuỳ thuộc vào các câu lệnh khác nữa

VD: for(auto x:n)

* Nếu n là xâu kí tự thì x là kí tự.
* Nếu n là số nguyên thì x là số nguyên.
* Khai báo và sử dụng hàm:
* Hàm: Là một thuật toán đã được định nghĩa sẵn, được sử dụng cho các mục đích cụ thể như: xác định giá trị của một số, dựng mảng,....
* Khai báo hàm:

<kiểu dữ liệu của hàm> <tên hàm> (các tham số đầu vào)

{

//các câu lệnh xử lí của hàm tại đây

}

VD:

long long tongchuso(long long a)

{

long long tong=0;

while(a>0)

{

tong+=a%10;

a/=10;

}

return tong;

}

* Return: Là loại lệnh trả giá trị mà hàm xử lí được về chương trình chính. Loại dữ liệu mà lệnh return trả phải tương đương với loại dữ liệu của hàm được khai báo.

VD:

* return a : trả giá trị là biến a.
* return true : trả giá trị là true, tương đương số 1.
* return false : trả giá trị là false, tương đương với số 0
* Các kiểu dữ liệu của hàm: Có nhiều kiểu dữ liệu khác nhau, chủ yếu tương tự với biến, cũng có các loại khác như
* Void: không trả về bất cứ giá trị gì, không thể dùng lệnh return trong loại hàm này. Loại này chủ yếu để tạo dựng nên những thứ cần thiết cho chương trình chính.
* Struct: sử dụng để tạo kiểu dữ liệu, trong đó nó có thể lưu nhiều biến trong cùng 1 phần tử. Hàm này không cần lệnh return.
  + Cú pháp sử dụng biến hàm: <tên phần tử>.<tên biến cần truy cập>.

VD: arr[i].name;

* + **Lưu ý:** Phải ghi dấu chấm phẩy sau dấu ngoặc đóng của hàm.
* Câu lệnh về mảng động(vector):
  + Khai báo vector: vector <kiểu dữ liệu> (tên mảng);
  + Các lệnh về mảng:
    - <tên mảng>.push\_back(): sử dụng để đẩy phần tử vào sau cùng của mảng.
    - <tên mảng>.size(): sử dụng để cho biết kích cỡ của mảng.
* Thuật toán mapping:
  + Khai báo một map: map<<kiểu dữ liệu thứ nhất>, <kiểu dữ liệu thứ 2>>;
  + Chúng được sử dụng tương tự như quy hoạch động, nhưng ngắn gọn hơn
  + Những câu lệnh về map:
    - <tên\_map>.first : thành phần thứ nhất của map, có kiểu dữ liệu thứ nhất được khai báo.

<tên\_map>.second: thành phần thứ hai của map, có kiểu dữ liệu thứ hai được khai báo.

* 1. Các thuật toán:
* Thuật toán tìm kiếm nhị phân: là một thuật toán sử dụng kĩ thuật chặt nhị phân để tìm kiếm, có tốc độ cực kì nhanh.
  + Giải thích vì sao nên dùng thuật toán tìm kiếm nhị phân:
    - Giả sử ta có một dãy số đã được sắp xếp, nếu như ta cần tìm kiếm một số nào đó theo thứ tự từ số đầu tiên đến số cuối cùng, mà số đó nằm ở cuối dãy, tức độ phức tạp thuật toán là O(n), với n là giới hạn dãy. Nếu cần kiểm tra nhiều lần cùng lúc mà số cần tìm kiếm vẫn nằm ở vị trí cuối dãy thì sẽ khiến máy chạy lâu, thậm chí có thể treo trong một thời gian dài.
    - Với thuật toán tìm kiếm nhị phân
* Sàng nguyên tố (Erathosthenes): Tạo ra một mảng đánh dấu tính nguyên tố của 1 đến 107.

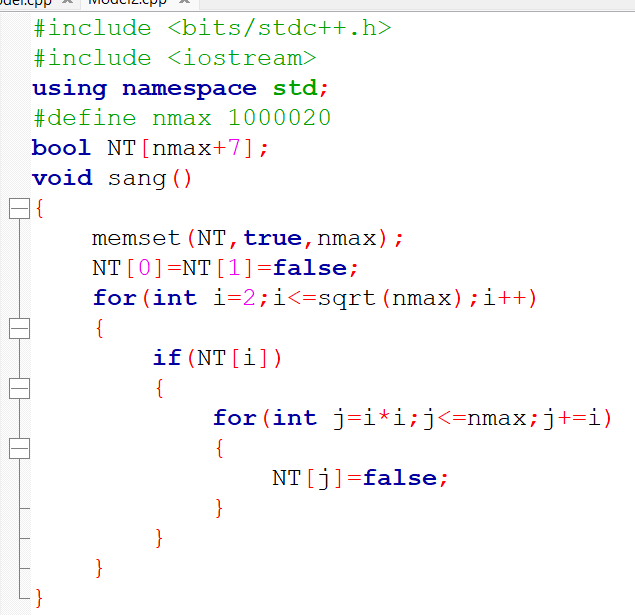
+) Cấu trúc:

B1: Định nghĩa nmax bằng 10000000 bằng lệnh #define nmax 10000000.

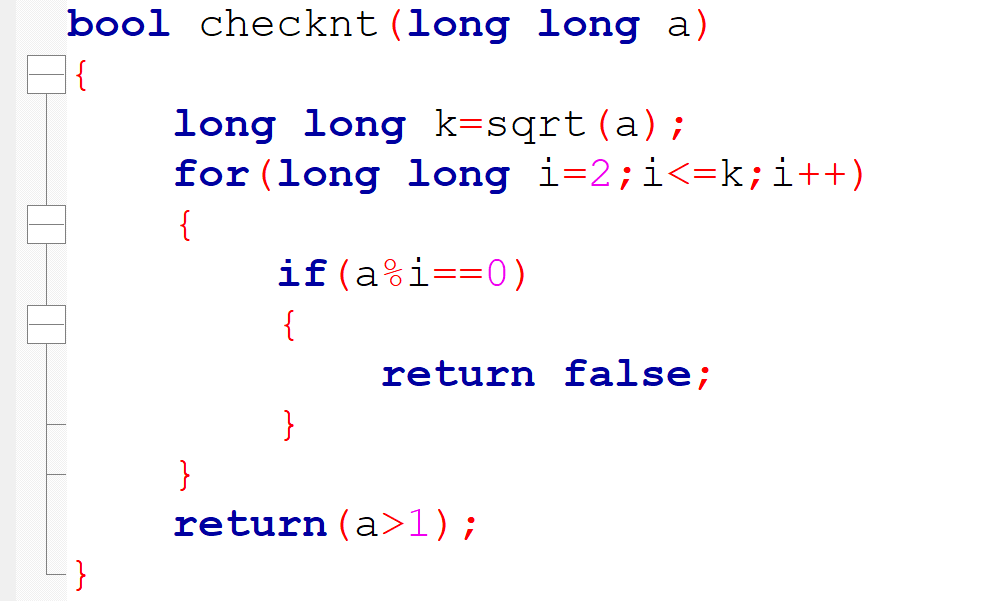
B2: Dựng mảng bool NT[nmax+7]

B3: Xây dựng hàm.

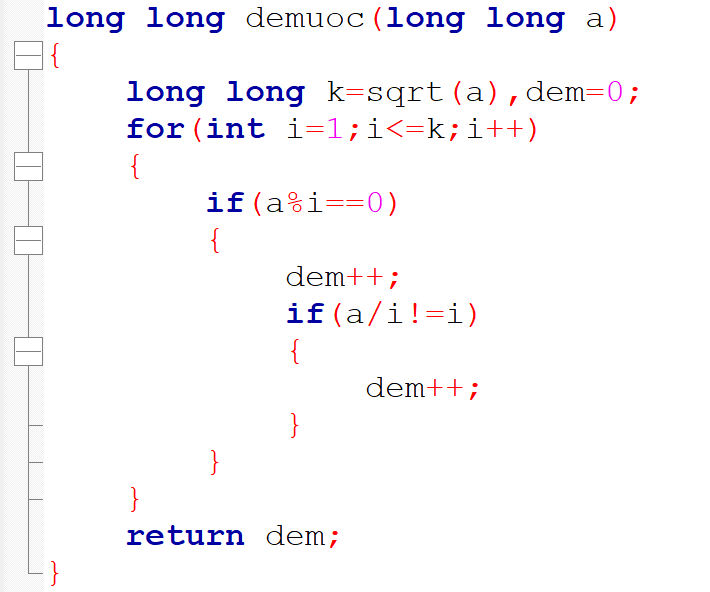
VD:



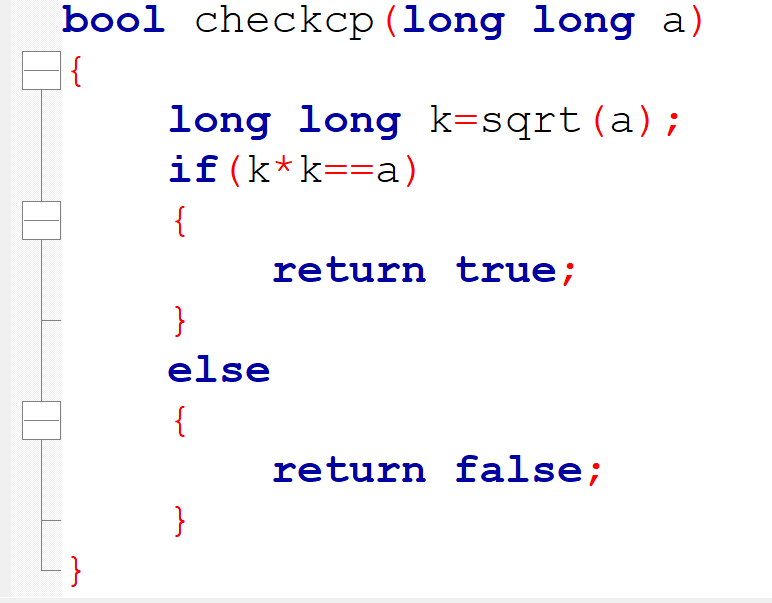
* Hàm kiểm tra nguyên tố(giới hạn 1012):



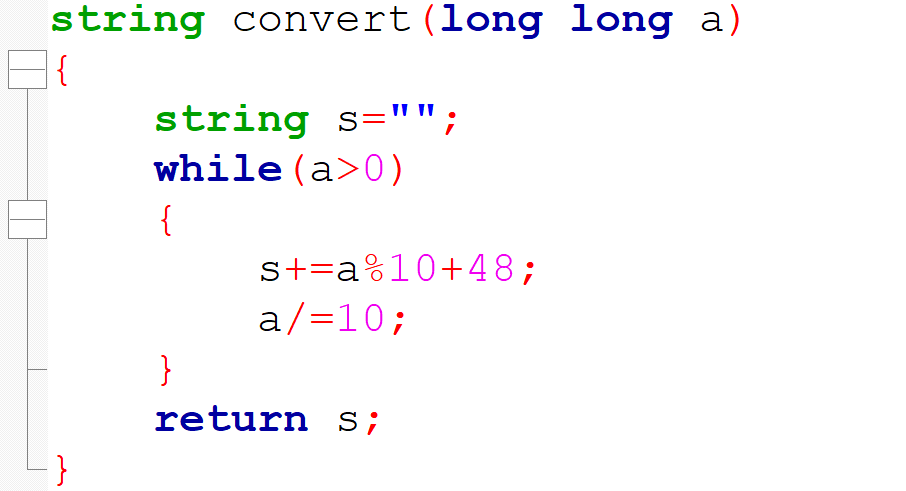
* Hàm đếm ước(giới hạn 1012):



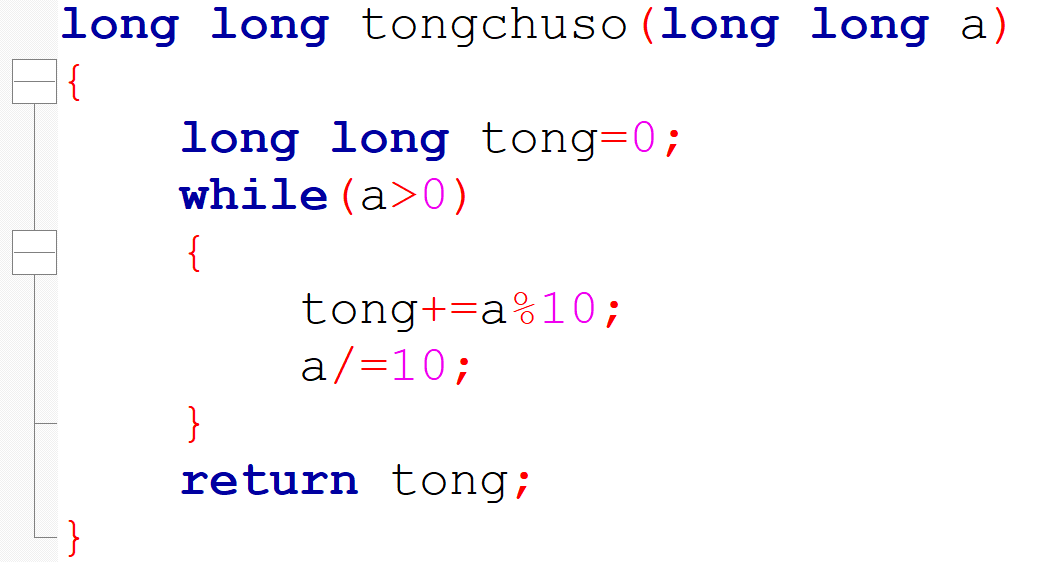
* Hàm kiểm tra số nguyên tố:



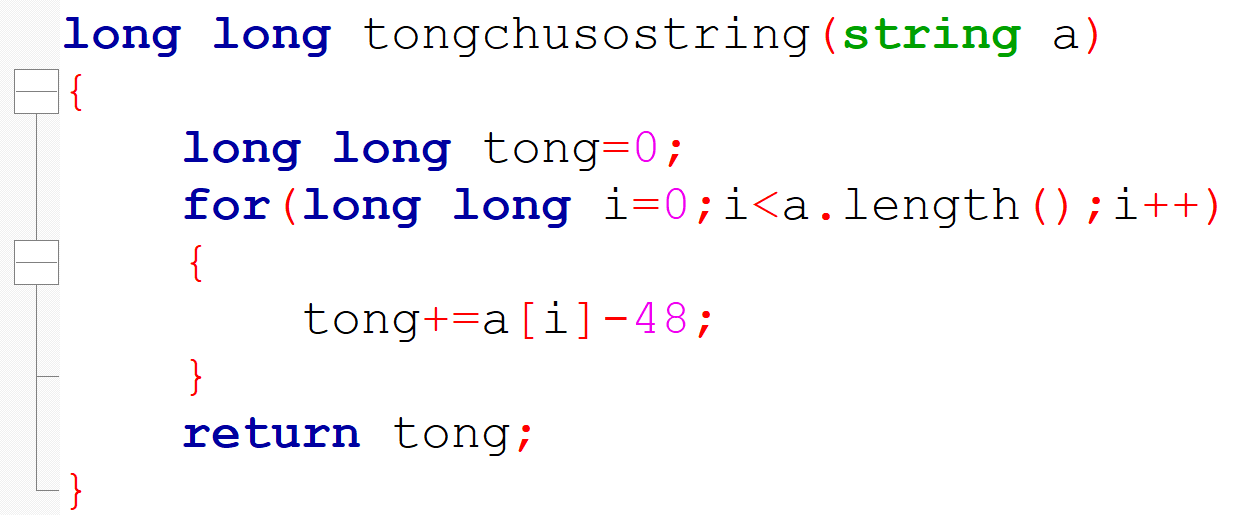
* Hàm chuyển đổi từ xâu sang số nguyên.



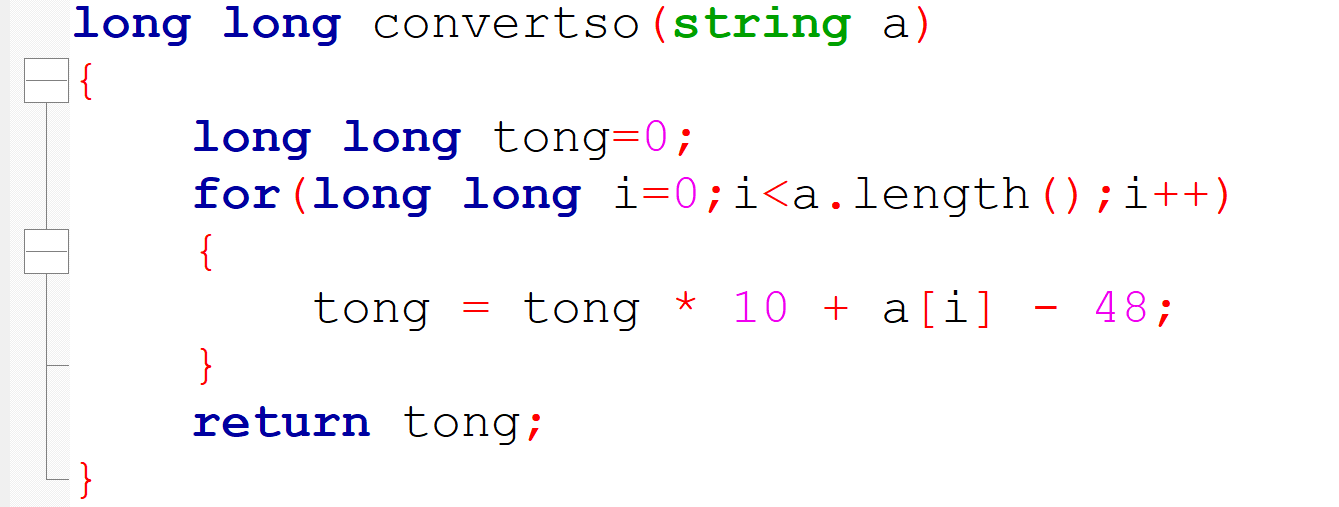
* Hàm tính tổng chữ số với đầu vào là số nguyên



* Hàm tính tổng chữ số với đầu vào là xâu.



* Hàm chuyển đổi từ xâu sang số nguyên.



* Công thức tính số chia hết cho x trong đoạn a,b

((b-a+1)/x)/2+1

* Công thức tính sô chia cho a dư k từ 1 ==> n

((n-k)/a) +1

* Công thức tính tổng từ 1 -> n: (n+1)\*n/2
* Công thức tính tổng từ a -> b thuộc cấp số cộng n: ((a - b)/n+1)\*(a+b)/2
  1. Các câu lệnh vặt hàm int main()
* Return 0: Khác với lệnh return của hàm, return 0 dùng để kết thúc chương trình chính.
* Break: Sử dụng để tạm dừng vòng lặp đang thực thi.
  1. Lưu ý:
* Cách chia lấy thập phân: cho số bị chia nhân với 1.0

1. HTML, CSS
   1. Khái niệm:

* **HTML** được sử dụng để tạo và cấu trúc các thành phần trong trang web hoặc ứng dụng, phân chia các đoạn văn, heading, titles, blockquotes.
* **CSS được sử dụng để định dạng, trang trí cho trang web, là ngôn ngữ hỗ trợ HTML.**
* **Javascript được sử dụng cho nhiều mục đích khác nhau như tạo web động, hỗ trợ cho HTML.**
  1. Cấu trúc cơ bản:
* Cấu trúc cơ bản của HTML bao gồm:

<!DOCTYPE html>  
<html>  
<head>  
  
</head>  
<body>  
  
</body>  
</html>

* 1. Các thẻ:
* Cấu trúc:

<tagname> Content goes here... </tagname>

* Thẻ kép: Là các loại thẻ bao gồm thẻ đóng và thẻ mở.
  + - Thẻ html: thẻ bao gồm toàn bộ nội dung trong trang web, cấu trúc:
    - Thẻ head: Thẻ chứa các nội dung chính của trang web, cấu trúc:
    - Thẻ title: Thẻ chứa tên của trang web, thường đứng phía trong thẻ head, cấu trúc:
    - Thẻ body: Thẻ chứa các nội dung trang web có thể hiển thị như tiêu đề, đoạn văn, hình ảnh, liên kết, bảng hay danh sách,...
    - Thẻ h: Là thẻ chứa tiêu đề của một đoạn văn.  
       + Thẻ h thông thường đi theo một con số đứng sau nó, vd như <h1>, dùng để chỉ về mức độ quan trọng của tiêu đề đó. Số càng cao tiêu đề đó càng nhỏ
    - Thẻ p: Thẻ chứa nội dung của một đoạn văn
    - Thẻ a: Thẻ chứa văn bản dẫn đến một liên kết, liên kết chứa trong một thuộc tính tên href
    - Thẻ div: Thẻ chứa các phần tử con khác, dùng để dễ dàng trong việc thêm thuộc tính bằng css
    - Thẻ ul,ol,li: Dùng để viết nên một danh sách
* Thẻ ul: Danh sách không sắp xếp
* Thẻ ol: Danh sách đã sắp xếp
* Thẻ li: Những phần tử con của các thẻ danh sách trên
* Thẻ form: Là thẻ để tạo nên 1 form, chủ yếu để thu thập dữ liệu của người truy cập và sử dụng trang web, thẻ này không trực tiếp xuất hiện trên trang web.
* Thuộc tính action: Là thuộc tính mà khi form được hoàn thiện sẽ được xử lí tại địa điểm nào đó.
* Thẻ input: Là một thành phần của thẻ form, là nơi mà người dùng nhập dữ liệu
* Có các thuộc tính như
  + Type: Là thuộc tính xác định loại input
    - Text: Trường nhập văn bản
    - Radio: Lựa chọn, chỉ có một lựa chọn trong tất cả các lựa chọn.
    - Checkbox: Lựa chọn, có thể có nhiều lựa chọn hoặc chẳng chọn cái nào.
    - Button: Tạo thành một cái nút có thể nhấp được
    - Submit: Là nút gửi theo nghĩa Tiếng Việt, là nút kết thúc form và gửi cho sv.
    - Number: Giống với thuộc tính text, chỉ khác là chỉ có thể nhập số, và có thể sử dụng nút tăng, giảm để tăng hoặc giảm giá trị trong input
    - Email: Tương tự với text, tuy nhiên chỉ dùng để nhập email
    - Password: Tương tự với text, tuy nhiên tất cả kí tự đều được ẩn đi và được tích hợp thêm 1 số tính năng khác của trường nhập mật khẩu
    - Reset: Tạo thành một chiếc nút có khả năng reset toàn bộ dữ liệu của form khi nhấn vào.
    - Color: Được sử dụng để nhập vào 1 màu sắc
    - Date: Được sử dụng để nhập ngày tháng năm.
  + Required: Bắt buộc người dùng phải điền vào trường này, không có giá trị
  + Autofocus: Tự động hướng con trỏ chuột vào đối tượng sở hữu lệnh này, không có giá trị
  + Disabled: Vô hiệu hóa trường sở hữu thuộc tính này, không có giá trị
* Value: Là thuộc tính xác định giá trị của chúng, được đặt bởi người thiết kế, có thể được hiện trên màn hình hoặc không tùy thuộc vào loại input
* Name: Là tên của input, sử dụng để xử lí dữ liệu được chính xác, do người thiết kế tự đặt. Những input có tên thì mới có thể xử lí.
* Placeholder: Là thuộc tính được sử dụng cho input thuộc loại text, được sử dụng để tạo một dòng chữ ‘giả’ bên trong trường nhập, biến mất khi người dùng nhập giá trị.
* Min: được sử dụng cho input loại date, là giới hạn tối thiểu của ngày tháng năm có thể chọn được, giá trị được trình bày theo yyyy-mm-dd
* Max: được sử dụng cho input loại date, là giới hạn tối đa của ngày tháng năm có thể chọn được, giá trị tương tự với min.
* Thẻ label: Là một loại thẻ văn bản, rất quan trọng trong form, được sử dụng kết hợp với input để giúp người dùng dễ dàng truy cập vào input.
* Thẻ fieldset: là thẻ bao trùm 1 vùng của thẻ form, có chức năng tương tự với thẻ div, chỉ khác nó có thể xuất hiện trực tiếp trên màn hình
* Thẻ legend: là thẻ dùng để gán tiêu đề cho thẻ fieldset, thường được gán trên đường biên của thẻ fieldset.
* Thẻ đơn(hay thẻ trống): Là thẻ chỉ có thẻ mở không có thẻ đóng.
* Thẻ br: Thẻ dùng để xuống dòng cho nội dung trong trang web, thường đứng trong hoặc sau một thẻ
  + - Thẻ img: thẻ dùng để chèn ảnh
      * Các thuộc tính trong thẻ:
        + Src: Nguồn ảnh
        + Alt: tiêu đề ảnh
        + Width: chỉnh về chiều ngang ảnh
        + Height: chỉnh về chiều cao ảnh
    - +) Thẻ link: có rất nhiều loại thẻ link, chúng đều nằm ở thẻ head, chủ yếu để liên kết các file với nhau  
       +) link:css: liên kết file html với file css

+) link:favicon: đặt icon cho tab của web

+) Có thuộc tính như

* Download: sử dụng để khi click vào thì
  + - +) Hr: Dùng để kẻ ngang phần tử
    - +) Thẻ nav: Dùng để chèn navigation
    - +) Thẻ script: dùng để đưa những thứ vốn không có trong máy tính vào web.
      * VD: file Javascript
    - Thẻ button: tạo một chiếc nút.
  1. Các thuộc tính
* Id: Thẻ định danh, giúp dễ dàng trang trí bằng css, sử dụng dấu # để edit.
* Class: Tương tự như id, sử dụng dấu . để edit.
* Width: Chỉnh chiều ngang của phần tử (đơn vị: px,%)

+) Max-content: chọn phần tử rộng nhất của thẻ làm chiều rộng chuẩn.

+) Min-content: chọn phần tử hẹp nhất của thẻ làm chiều rộng chuẩn.

* Height: Chỉnh chiều cao phần tử (đơn vị: px,%)

+) Đơn vị vh: Là đơn vị căn chiều cao theo tỉ lệ chiều cao của trang web(1vh = 1/100; 100vh=100/100)

* Color: Chỉnh màu chữ (đơn vị: mã màu hoặc màu có sẵn)
* Background-color: Chỉnh màu nền của đối tượng (đơn vị: mã màu hoặc màu có sẵn)
* Margin: Chỉnh về khoảng cách giữa các phần tử khác(đơn vị: px)

+) Margin-left: Phía trái

+) Margin-right: Phía phải

+) Margin-top: Phía trên

+) Margin-bottom: Phía dưới

+) **Lưu ý**: Nếu chỉ có 1 chỉ số thì nó sẽ hiểu là áp dụng cho 4 chiều, nếu có 2 chỉ số thì nó sẽ hiểu chỉ số đầu tiên là trên dưới, chỉ số thứ hai là trái phải, nếu có 4 chỉ số thì chỉ số đầu tiên bắt đầu từ phía trên, xoay theo chiều kim đồng hồ với thứ tự của chỉ số

* Padding: Chỉnh về khoảng cách giữa phần tử với viền của trang web(đơn vị: pixel)

+) Padding-left: Phía trái

+) Padding-right: Phía phải

+) Padding-top: Phía trên

+) Padding-bottom: Phía dưới

+) **Lưu ý**: Nếu chỉ có 1 chỉ số thì nó sẽ hiểu là áp dụng cho 4 chiều, nếu có 2 chỉ số thì nó sẽ hiểu chỉ số đầu tiên là trên dưới, chỉ số thứ hai là trái phải, nếu có 4 chỉ số thì chỉ số đầu tiên bắt đầu từ phía trên, xoay theo chiều kim đồng hồ với thứ tự của chỉ số

* Border: Thêm viền cho phần tử(đơn vị: pixel, mã màu hoặc màu có sẵn)
* Display: Định hình chiều ngang của phần tử chứa văn bản(đơn vị: tuỳ chỉnh).

+) Block: chiếm hết chiều ngang của thẻ mẹ, dành cho div, p....

+) Inline: chỉ chiếm một phần vừa đủ, không set được một số thuộc tính như width, height, text-align , dành cho a, span, u, i, b,...

+) Inline-block: dung hợp của block và inline, có thể set được thẻ width và height.

+) Flex: Dàn hàng các phần tử con của thẻ đó, tất cả các thẻ con đều phải dàn thành hàng, mặc cho các thuộc tính khác.

* Box-shadow: tạo nên bóng của phần tử(đơn vị: px, mã màu hoặc màu có sẵn)

+) Có tối đa 4 chỉ số của thuộc tính này: thứ tự của chỉ số ứng với thứ tự của các cạnh theo chiều kim đồng hồ, bắt đầu từ cạnh bên phải

* Border-radius: bo tròn cạnh của phần tử(đơn vị: px,%)

+) Có tối đa 4 chỉ số: thứ tự chỉ số ứng với thứ tự các góc theo chiều kim đồng hồ, bắt đầu từ góc trên trái.  
+) Nếu chỉ số là 50% thì chúng sẽ tạo ra hình tròn

* Transform: Thêm hiệu ứng cho đối tượng

+) rotate(): đơn vị: deg(độ)

* Text-transform: Chỉnh sửa đối tượng văn bản

+) uppercase: Viết hoa tất cả

+) lowercase: Viết thường tất cả

+) capitalize: Viết hoa đầu từ

* Flex-wrap: Dùng để quyết định có xuống dòng cho lệnh display:flex hay không.

+) Giá trị mặc định: nowrap. Giá trị khác: wrap  
+) Có tiếp nhận: width, height

+) Không tiếp nhận: display:block, display:inline,...

* Justify-content: Dùng để căn các phần từ của display:flex theo chiều ngang  
  +) Giá trị mặc định: flex-start (phía trái)  
  +) Giá trị khác: flex-end (phía phải), center (giữa), space-around(chèn hết diện tích), space-between(làm sao cho có khoảng cách ở giữa), space-evently(khoảng cách đều nhau)
* Align-items: Dùng để căn các phần tử của display:flex theo chiều dọc, có các giá trị như với justify-content.  
  +) Giá trị mặc định: flex-start (phía trên)

+) Giá trị khác: flex-end (phía dưới), center (giữa), ...

* Align-content: Giống với align-items, chỉ khác thay vì thuộc tính cho phần tử, chúng căn cho từng “ngăn” của display:flex

+) Giá trị: tương tự như justify-content

* Position: Dùng để xác định vị trí của thẻ.

+) Có các giá trị như:

* Static: Mặc định
* Relative: nằm ở vị trí tuyệt đối theo trục toạ độ giả tưởng
* Absolute: nằm ở vị trí tương đối, phụ thuộc vào thẻ có khai báo Position: relative; hoặc thẻ body nếu không có.
* Fixed: cố định tại điểm nào đó

+) Có 4 thuộc tính đi kèm theo: top, right, bottom, left.

* Hiệu ứng hover(:hover): dùng để tạo hiệu ứng khi có chuột chạm vào, áp dụng cho link.

+) Cách dùng: sau khi khai báo đối tượng trong css, gõ “:hover”

* List-style: dùng để định dạng marker của danh sách

+) Có các giá trị như:

* Square: Hình vuông
* Circle: Hình tròn
* Inside: gần hơn với phần tử
* Outside: xa hơn với phần tử
* Url: nguồn của icon.
* Text-decoration: dùng để trang trí cho thẻ.

+) Có các giá trị như

* None: không có.
* Underline: gạch chân
* Overline: gạch đầu
* Line-throught: gạch ngang.
* Font-weight: Dùng để tăng kích cỡ nét chữ cho văn bản

+) Giá trị: số, mặc định: 500.

* Background-image: Dùng để chèn hình ảnh vào nền

+) Có các giá trị như

* url(): Dùng để chèn nguồn ảnh.
* Background-repeat: thuộc tính đi kèm với background-image, xác định xem có lặp ảnh hay không.

+) Có các giá trị:

* + Repeat: Có, giá trị mặc định
  + No-repeat: Không.
  + Repeat-x: Lặp lại chỉ chiều ngang
  + Repeat-y: Lặp lại chỉ chiều dọc
* Background-origin: chọn điểm bắt đầu của background-image.

+) Có các giá trị như:

* Padding-box: Giá trị mặc định, bắt đầu từ góc trên trái phần padding.
* Border-box: Bắt đầu từ góc trên trái phần border.
* Content-box: Bắt đầu từ góc trên trái phần content.
* Background-size: chọn kích cỡ của ảnh nền, có các giá trị như:
  + Contain: bao toàn bộ vừa khít nội dung của thẻ đó
  + Cover: che lấp, theo kích cỡ của thẻ đó
* Outline: Đặt đường viền cho một khu vực nào đó, tương tự border, nhưng nằm ngoài border
  + Có các giá trị như:
    - None: Không đặt
* Border-image: Sử dụng ảnh để làm viền cho một phần tử, tương tự như border
  + Có các giá trị như:
    - Url(): dùng để chèn đường dẫn của ảnh
* Animation – name: là thuộc tính sử dụng để áp dụng một hiệu ứng được định nghĩa cụ thể trong file CSS hoặc thẻ style của HTML.
  + Giá trị:
    - Tên của hiệu ứng cần áp dụng.
    - None: không áp dụng
* Animation – duration: là thuộc tính sử dụng để tạo khoảng thời gian diễn ra hiệu ứng
  + Giá trị:
    - Tính theo đơn vị giờ phút giây (h, m, s)
    - Có thể có giá trị 0, tương đương với việc không thực thi hiệu ứng
* Animation – delay: là thuộc tính sử dụng để tạo khoảng thời gian trễ dành cho hiệu ứng
  + Giá trị
    - Tính theo đơn vị giờ phút giây
    - Có thể có giá trị 0, tức thực thi hiệu ứng ngay lập tức.
* Animation – iteration – count: là thuộc tính sử dụng để tạo vòng lặp cho hiệu ứng
  + Giá trị:
    - Số: dùng để chỉ nó sẽ lặp bao nhiêu lần.
    - Infinity: lặp lại liên tục cho đến khi hiệu ứng không thể thực thi được nữa.
* Z – index: là câu lệnh sử dụng để sắp xếp lại đối tượng theo lớp, tương tự như powerpoint
  + Có các giá trị như:
    - Số:
      * Đối với số dương, nó tiến lên số lớp đã khai báo.
      * Đối với số âm, nó lùi lại số lớp đã khai báo.
* List – style – image: câu lệnh dùng để lấy ảnh làm bullet cho danh sách
  + Có các giá trị như:
    - Url(): đường dẫn tới ảnh đó.
* Cursor: câu lệnh dùng để đặt hình dạng con trỏ chuột khi chạm vào
  + Có các giá trị như:
    - Pointer: hóa thành hình bàn tay có ngón trỏ chỉ vào.
* Transition: là hiệu ứng khi có một sự kiện gì đó xảy ra, tương tự như animation.
  + Có các giá trị như:
    - Thuộc tính được áp dụng(width, height, padding, margin,...)
    - Thời lượng(đơn vị s,m,h)
    - Có thể nhập nhiều lần giá trị như trên
  + Cách setup hiệu ứng transition:
    - B1: Ghi thuộc tính transition vào đối tượng cần ghi như trên
    - B2: Khi đối tượng xảy ra một sự kiện nào đó, VD như chạm vào con trỏ chuột(:hover) thì ghi thuộc tính của thẻ đó sau khi xảy ra sự kiện
* Transition – timing – function: sử dụng để chỉ định cách mà một hiệu ứng transition được thực thi
  + Có các giá trị như:
    - Ease: Đây là giá trị mặc định, hiệu ứng sẽ bắt đầu chậm, rồi nhanh dần, rồi lại kết thúc chậm dần.
    - Linear: Tốc độ từ khi bắt đầu đến kết thúc là như nhau.
    - Ease – in: Chậm khi bắt đầu và nhanh dần
    - Ease – out: Chậm dần khi kết thúc
    - Ease – in – out: kết hợp của ease – in và ease – out.
* Transiton – delay: độ trễ khi thực thi hiệu ứng
  + Có các giá trị như:
    - Thời lượng( đơn vị là s,m,h )
* Visibility: cho phép xem một đối tượng có hiển thị trên màn hình không, cho dù giá trị là như nào, chúng vẫn **ẢNH HƯỞNG**đến bố cục của trang web
  + Có các giá trị như:
    - Hidden: ẩn đi
    - Visible: hiện lên(mặc định)
  1. Các kĩ năng và lưu ý:
* Vô hiệu hoá dòng lệnh: Bôi đen ==> Ctrl + ?.
* Tạo ra cấu trúc html: gõ ‘html:5’ ==> nhấn Enter.
* Viết cùng thuộc tính cho nhiều thẻ: gõ tên thẻ đó và đánh dấu ‘,’
* Viết thuộc tính cho 1 thẻ con trong một thẻ mẹ: viết tên thẻ mẹ rồi cách và viết tên thẻ con.
* Chọn nhiều dòng lệnh để thao tác cùng lúc: nhấn giữ phím alt và chọn những dòng cần thao tác.
* Tạo nhiều class: trong thuộc tính class của thẻ, gõ tên class đầu tiên rồi gõ phím cách và gõ tiếp.
* Thẻ display:flex và các thuộc tính đi kèm phải ghi vào thẻ cha chứ không phải là thẻ con.
* Chiều ngang trang web là 1535px, chiều dọc là 745.6px.
* Để dùng display:flex cho thanh nav, ta nên đặt thuộc tính cho thẻ tạo nav là
* Top: 0;
* Width: 100%;
* Có thể chèn hai ảnh nền khác nhau trong cùng một dòng background-image, cũng có thể edit chúng trên cùng một dòng, tuy nhiên phải tuân theo thứ tự từ trái qua phải.

VD:



* 1. Các kĩ năng thiết kế
* Thanh navigation:

+) Gồm 2 phần: Logo mà menu

+) Logo là hình ảnh đại diện cho trang web

+) Menu hay thanh điều hướng là nơi giúp cho người dùng có thể đi đến những nơi khác nhau của trang web. Ở đây nên sử dụng danh sách để làm chúng.

* Đặt flex dễ dàng:

B1: Vào file css, gõ:

.flex-container{

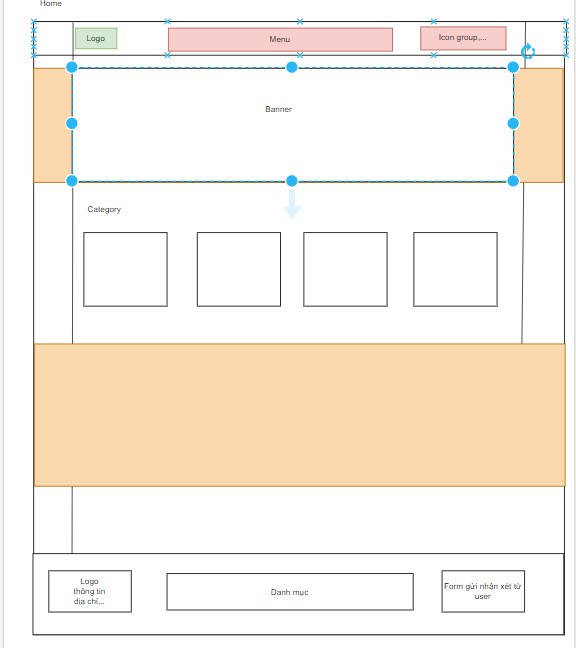
display: flex;

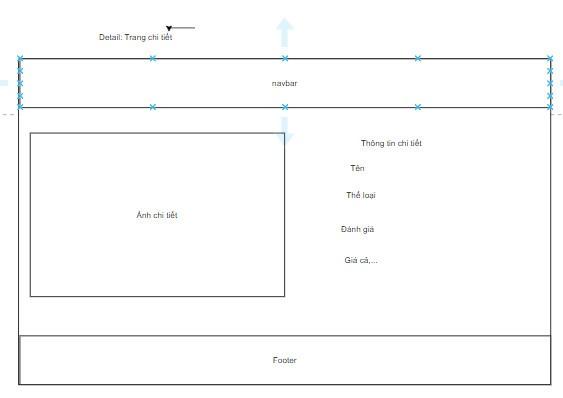
}

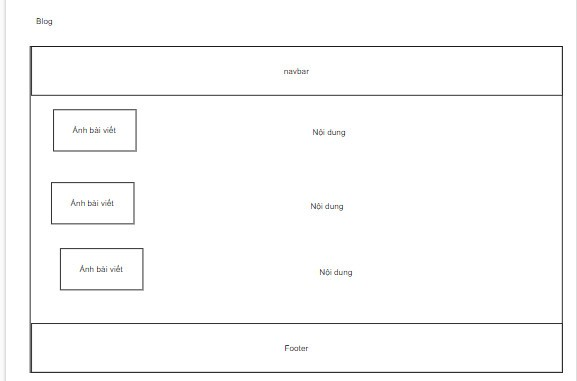
B2: Khi cần đến, vào file html, copy class này vào thẻ cần flex

**Lưu ý:** Có thể áp dụng cho bất kì thứ gì mà bạn muốn

* Bố cục chủ đạo của trang web
  + Nên sử dụng một container làm hộp chứa con cho mỗi phần lớn khác nhau của HTML như nav, .text-wrapper, footer, ....
  + Đối với trang home, nên cho một bức ảnh lồng chữ, được gọi là banner.
  + Tránh viết chữ lôi thôi, dài dòng, nên kết hợp với ảnh và các đối tượng khác
* Thanh Navigation thông thường gồm có các mục chính như: Trang chủ(Home), Giới thiệu(About), Liên lạc(Contact), ...
* Nền của trang web: nên phân bố sao cho độ tương phản đồng đều, giúp tránh bị đau mắt và tạo cảm giác hứng thú cho người đọc.
* Chân trang: Nên phân bố theo bố cục
  + Bên trái: Logo, Địa chỉ
  + Ở giữa: Các danh mục
  + Bên phải: Form nhỏ sử dụng để login
  + ....
* Các ảnh bố cục:







* 1. Các nguồn tài nguyên:
* Trang web chèn icon: <https://fontawesome.com>

+) Script: <script src="https://kit.fontawesome.com/b25957ae4f.js" crossorigin="anonymous"></script>

* Trang web tìm font: [Browse Fonts - Google Fonts](https://fonts.google.com/)
* Phông chữ đẹp:
* "Poppins", Arial, sans-serif.
* Open Sans, sans-serif
* Robotto, sans-serif
* Cookie, cursive
* Nguồn CSS: <https://getcssscan.com>

+) Box-shadow: <https://getcssscan.com/css-box-shadow-examples>

* Trang web thiết kế logo free: <https://www.freelogodesign.org/>
  1. Lệnh viết tắt:
* Margin: gõ “m” và giá trị liền sau
* Padding: Tương tự margin, thay vì gõ “m” thì gõ “p”
* Tạo một thẻ có class và id sẵn:

+) B1: Viết tên thẻ

+) B2: Viết # và id của nó(nếu có)

+) B3: Viết . và class của nó, có thể viết nhiều lần để tạo nhiều class khác nhau.

+) B4: Enter.

* Tạo nhiều thẻ một lúc

B1: Gõ tên thẻ

B2: Gõ số lượng thẻ: gõ “\*” rồi gõ số(1 thì ko cần gõ)

B3: Gõ dấu lớn(nếu có)

B4: Gõ phần tử con chứa trong phần tử đó bằng dấu “>”;

B5: Gõ phần tử đồng cấp(nếu có) bằng dấu “/”;

B6: Enter

* 1. Các đơn vị
* Pixel: đơn vị cơ bản nhất, đong theo kích cỡ của điểm ảnh tuỳ thuộc vào màn hình máy tính
* Em: đơn vị dựa trên kích cỡ mặc định của thẻ, 1em thì bằng với kích cỡ mặc định, 2em là gấp đôi
* Vw: đơn vị dựa trên kích cỡ chiều ngang của màn hình, 1vw = 1%, tức 10vw là 10%.
* Vh: đơn vị dựa trên kích cỡ chiều dọc của màn hình, tương tự với vw
* H, m, s: tương tự lần lượt là giờ, phút, giây

1. Javascript
   1. Khái niệm

* Javascript là một ngôn ngữ lập trình phổ biến, được sử dụng để tạo nên “bộ xử lí” cho 1 trang web.
  1. Các loại biến
* Gồm có các loại data chính:
* Undefined(không xác định): là kiểu dữ liệu không xác định, không được gán giá trị gì ngay từ lúc khai báo
* Null(không có hay rỗng): là kiểu dữ liệu object, nhưng giá trị lại là “null”
* Boolean(biến logic, chỉ có giá trị true hoặc false): là kiểu dữ liệu đúng sai, có thể nhận giá trị bằng cách gán trực tiếp hoặc thông qua các biểu thức
* String(xâu kí tự): là tập hợp các kí tự, có thể sử dụng nháy đơn, nháy kép, nháy huyền để gán giá trị
* Number(số): là tập hợp của toàn bộ các tập hợp số như số nguyên, số thực, số thập phân,...
* Integer: số nguyên
* Float: số thực
  1. Toán tử
* Toán tử là dấu hay kí tự đặc biệt, dùng để thực hiện các phép tính của biểu thức nào đó và cho ra kết quả cuối cùng
* Các loại toán tử cơ bản gồm
* Toán tử số học

|  |  |  |
| --- | --- | --- |
| **Toán tử** | **Mô tả** | **Ví dụ** |
| + | Cộng | 10 + 5 = 15 |
| - | Trừ | 10 – 5 = 5 |
| \* | Nhân | 10 \* 5 = 50 |
| \*\* | Luỹ thừa  (ES6) | 2 \*\* 4 =16  4 là số mũ |
| / | Phép chia | 10 / 5 = 2 |
| % | Phép chia lấy dư | 12 % 5 = 2  2: Là số dư của phép tính |
| ++ | Tăng 1 giá trị | 1++ = 1 + 1 = 2 |
| -- | Giảm 1 giá trị | 1-- = 1 – 1 = 0 |
| Math.sqrt(value)  (Là một hàm, nhưng được xem như là 1 toán tử) | Căn bậc 2 của một số | Math.sqrt(100) = 10 |

* Toán tử so sánh

|  |  |  |
| --- | --- | --- |
| **Toán tử** | **Mô tả** | **Ví dụ** |
| == | So sánh bằng | 5 == 5//true |
| === | So sánh giống hệt nhau(giá trị và kiểu dữ liệu) | 5 === “5”//false |
| != | Khác giá trị | 5 != 10 //true |
| !== | Khác giá trị và kiểu dữ liệu | 5 !== 10//false |
| > | Lớn hơn | 10>5//true |
| < | Nhỏ hơn | 10<5//false |
| >= | Lớn hơn hoặc bằng | 10>=10//true |
| <= | Nhỏ hơn hoặc bằng | 10<=10//true |

* Toán tử logic

|  |  |  |
| --- | --- | --- |
| **Toán tử** | **Mô tả** | **VD** |
| && | Phép toán and trả về true khi cả hai  phép a,b đều true | a&&b |
| || | Phép toán or trả về true khi một  trong các phép a,b có giá trị true | a||b |
| ! | Phép toán not trả về kết quả ngược  lại của biểu thức, có thể sử dụng để xác định biến có là undefined hay không | !a |

* Toán tử gán

|  |  |  |
| --- | --- | --- |
| **Toán tử** | **VD** | **Ý nghĩa** |
| = | x = y | Gán x vào y |
| += | x+=y | x = x + y |
| -= | x-=y | x = x - y |
| \*= | x \*= y | x = x \* y |
| /= | x/=y | x = x / y |
| %= | x%=y | x = x % y |

* Toán tử ba ngôi(Ternary): Là viết tắt của lệnh điều kiện if ... else

VD: 

Trong đó:

* tenBien: Tên biến
* dieuKien: biểu thức điều kiện
* giaTri1, giaTri2: nếu dieuKien là true, tenBien nhận giaTri1, nếu là false, nhận giaTri2
* Lưu ý:
  + Khi không ghi giaTri2, tức là nếu dieuKien là true thì nhận giaTri1, false thì cho qua.
  + Khi không ghi cả hai giaTri1, giaTri2, tức là tenBien giờ sẽ nhận giá trị True hoặc False trực tiếp từ dieuKien.
  1. Câu lệnh
* Xuất ra màn hình console: sử dụng console.log(<đầu ra>).
  + Nếu in bằng chuỗi kí tự: Phải đặt trong nháy kép hoặc đơn
  + Nếu in bằng biến: viết thẳng tên biến
  + VD:
    - console.log(x)
    - console.log(“Hello World!”)
* Khai báo biến: sử dụng lệnh var, let hoặc const sau đó kèm với tên biến.
* VD:
* let x;
* var y;
  + **Lưu ý:**  Các lệnh khai báo có tác dụng khác nhau:
    - Var: sau khi khởi tạo, có thể gán và khai báo lại lần nữa, thậm chí có thể dùng mà không cần khởi tạo(điều này rất dễ gây nhầm lẫn giữa biến này và biến khác).
    - Let: sau khi khởi tạo, có thể gán biến bằng giá trị mới khác với giá trị cũ, KHÔNG thể khai báo lại lần nữa.
    - Const: sau khi khởi tạo, KHÔNG thể gán biến bằng giá trị mới khác với giá trị cũ (sử dụng để lưu trữ những dữ liệu cố định, không bao giờ thay đổi).
* Gán giá trị của biến: sử dụng toán tử “=” để gán giá trị
* VD:
  + x = 10;
  + y = 5;
* Khởi tạo biến: Là hành động kết hợp của khai báo và gán giá trị
* VD: var x = 10;
* Cách để biết kiểu dữ liệu: sử dụng typeof để biết kiểu dữ liệu, có thể kết hợp với lệnh console.log() để in ra
  + VD: console.log(typeof x);
* Các câu lệnh về hàm:
* Khai báo hàm: sử dụng lệnh function
* Cấu trúc: function <tên hàm>(<các tham số đầu vào>)

{

<câu lệnh xử lí>

}

* Gọi hàm: <tên hàm>(<các tham số đầu vào>)

**Lưu ý:** Tất cả các tham số đầu vào đều phải điền đúng, đủ thì hàm mới hoạt động được. Hàm phải được gọi thì lệnh trong hàm mới được thực thi.

* Các lệnh liên quan đến hộp thoại thông báo:
  + Xuất ra dưới dạng một thông báo: alert(“tên thông báo”)
    - Sử dụng để tạo ra một thông báo.

VD: alert(“Hello World!”)

* + Nhập dữ liệu dới dạng một thông báo: prompt(“tên thông báo”)
    - Sử dụng để tạo ra một dữ liệu đầu vào dưới dạng một thông báo

VD: prompt(“Enter something?”)

* + - Có thể sử dụng lệnh này gán vào biến đầu vào để xử lí cho những câu lệnh sau.
* Lệnh ép kiểu: <kiểu dữ liệu>(value)
  + Sử dụng để chuyển đổi từ kiểu dữ liệu nào đó đến kiểu dữ liệu được khai báo trong hàm.

VD:

* + - String(input\_number): chuyển input\_number từ kiểu nào đó đến string
* Lệnh vòng lặp tìm kiếm - hàm filter: <tên\_mảng>.filter(<lệnh so sánh>)
  + Sử dụng để tạo vòng lặp có sẵn sử dụng thuật toán tìm kiếm tuần tự để tìm kiếm một đối tượng nào đó.
  + Lệnh so sánh chính là hàm được sử dụng làm lệnh tìm kiếm(tương tự như cách làm hàm so sánh dùng cho lệnh sort trong C++).
  + Toán tử được sử dụng để tìm kiếm thông thường là ‘==’.
  + Điểm đặc biệt: Khi ta sử dụng hàm làm lệnh tìm kiếm có khai báo một tham số, thì tham số đó sẽ được truyền trực tiếp phần tử của mảng mà vòng lặp filter này đang đến, thuận tiện hơn trong việc xây dựng hàm

VD:

function searchProduct(searchString){

    let founderProduct = productList.filter( function(product) {

        return product.name.toLowerCase().includes(searchString.toLowerCase());

    });

    renderProductList(founderProduct);

}

* Lệnh chuyển đổi toàn bộ kí tự về kí tự thường: <tên\_biến>.toLowerCase();
  + Được sử dụng cho xâu kí tự để chuyển tất cả các kí tự dạng chữ hoa về chữ thường.
  + Chúng hỗ trợ rất tốt cho lệnh hàm tìm kiếm, vì thông thường khi tìm kiếm, người ta chỉ gõ kí tự thường.
* Lệnh tìm xâu con: <tên\_xâu\_cha>.includes(tên\_xâu\_con)
  + Được sử dụng để tìm xem trong một xâu có xuất hiện một xâu con nào như đã khai báo hay không.
  + Hàm trả về giá trị là true khi tìm thấy, false khi ngược lại
  1. Các sự kiện
* Cách sử dụng sự kiện trong Javascript:
  + Document.<vị trí đối tượng trong HTML>.<tên sự kiện> = “<lệnh tại đây>”.
* Có các sự kiện chính
  + Onchange: Khi đối tượng thay đổi
  + Onclick(JS: ‘click’) Khi bạn nhấp 1 lần chuột vào đối tượng
  + Onmouseover: Khi bạn đưa trỏ chuột vào đối tượng
  + Onmouseout: Khi bạn đưa trỏ chuột ra khỏi đối tượng
  + Onkeydown: Khi bạn nhấn một phím gì đấy
  + Onload: Khi trang hoàn tất việc tải về
  + Input(chỉ dành cho JS): Khi giá trị của trường điền(input) thay đổi
  + Submit(chỉ dành cho JS): Khi nút submit của form được click
  1. Các khái niệm:
* Key and value
  + Mỗi một giá trị gì trong Javascript đều được lưu ở dạng key and value. Biến, mảng, kiểu đối tượng nhiều trường, ... đều được lưu ở dạng này.
  + Cấu trúc: <tên\_key> = <value>. Trong đó, key có thể hiểu là tên của đơn vị lưu dữ liệu, còn value là giá trị hay dữ liệu cần ghi của đơn vị lưu dữ liệu đó.
  + Có thể sử dụng toán tử hoặc “ || ” để tạo thành cấu trúc lựa chọn đầu vào. Khi vế đầu tiên trả về false, giá trị sẽ được đặt theo vế kế tiếp, nếu vế kế tiếp cũng false, đến vế tiếp theo... cho đến khi một vế là true hoặc giá trị khác thì biến được đặt theo đó.
* Kiểu đối tượng nhiều trường: là một kiểu dữ liệu cho phép người dùng tự tạo nhiều trường hay biến khác nhau trong một kiểu dữ liệu.
  + Cú pháp: var/let/const <tên\_biến> = {

var/let/const <tên\_biến\_con\_1>;

var/let/const <tên\_biến\_con\_2>;

var/let/const <tên\_biến\_con\_3>;

...

}

* For – in: một lệnh vòng for vắn tắt sử dụng để lấy chỉ số hoặc tên của mảng và object.
  + Đối với mảng: Dùng để lấy ra các chỉ số thứ tự của các phần tử có trong mảng.
  + Đối với đối tượng(object): Dùng để lấy ra tên ( KHÔNG phải là giá trị của thuộc tính ) các thuộc tính của object.
  + Cú pháp:

for (let/var <tên\_biến> in <tên\_mảng hoặc tên\_đối\_tượng>)

{

Khối lệnh;

}

VD:

for (let chi\_so in arr)

{

console.log(`Chỉ số ${chi\_so}`)

}

* For – of: là một lệnh vòng for sử dụng để lấy giá trị của phần tử hoặc thuộc tính của mảng và object.
  + Đối với mảng: Dùng để lấy ra các giá trị của các phần tử có trong mảng.
  + Đối với đối tượng(object): Dùng để lấy ra giá trị của các thuộc tính của object.
  + Cú pháp đối với mảng:

for (let/var <tên\_biến> of <tên\_mảng>)

{

Khối lệnh;

}

VD:

for (let chi\_so of arr)

{

console.log(`Chỉ số ${chi\_so}`);

}

* + Cú pháp đối với object:

for ( let gia\_tri of Object.values(<tên\_object>) )

{

Khối lệnh;

}

VD:

for (let gia\_tri of Object.values(<tên\_object>))

{

console.log(`gia\_tri ${gia\_tri}`);

}

* Mảng dữ liệu: là 1 loại biến có thể lưu trữ nhiều giá trị khác nhau.
  + Cú pháp khai báo một mảng: let/var/const <tên\_mảng> = [<GT1>, <GT2>, ...];

VD: let arr = [1, 2, 3];

* + Cú pháp lấy một giá trị trong mảng: <tên\_mảng>[<vị trí phần tử>];

**Lưu ý:** Vị trí đầu tiên của mảng được tính là số 0, vị trí cuối cùng là kích cỡ mảng trừ đi 1.

* + Lấy số phần tử của mảng: <tên\_mảng>.length;
  + Duyệt mảng: việc truy cập từng phần tử của mảng.
* Vòng lặp: sử dụng để thực hiện nhiều công việc mang tính chất tương tự nhau, có quy luật.
  + For: là một lệnh tạo vòng lặp cơ bản, tương tự như C++.
    - Cú pháp khai báo: for (<khai báo biến>, <điều kiên dừng vòng lặp>, <lệnh bước nhảy>)
    - Có ba trường hợp của vòng for:
      * Thứ nhất: khai báo đủ 3 mệnh đề trong cặp dấu ngoặc đơn của for.

VD:

for (let i = 0, len = cars.length, text = ""; i < len; i++) {  
  text += cars[i] + "<br>";  
}

* + - * Thứ hai: khai báo biến ngoài lệnh for

VD:

let i = 2;  
let len = cars.length;  
let text = "";  
for (; i < len; i++) {  
  text += cars[i] + "<br>";  
}

* + - * Thứ ba: sử dụng lệnh bước nhảy trong vòng for, chỉ để duy nhất điều kiện dừng vòng lặp.

VD:

let i = 0;  
let len = cars.length;  
let text = "";  
for (; i < len; ) {  
  text += cars[i] + "<br>";  
  i++;  
}

* + While: là vòng lặp chỉ cần có 1 mệnh đề: điều kiện dừng: vòng lặp sẽ dừng khi điều kiện bạn ghi vào trả về giá trị false.
    - Cú pháp: while (<điều\_kiện\_dừng>)

**Lưu ý:** Vòng lặp while rất nguy hiểm nếu như lệnh điều kiện dừng không bao giờ trả về giá trị false, nó có thể gây treo máy, treo trình duyệt trong một khoảng thời gian rất dài.

* 1. Các kĩ năng, lưu ý:
* Cách khai báo, khởi tạo biến, mảng, object:
  + Tên biến: sát với nội dung, mục đích sử dụng của biến. Tránh đặt tên quá trừu tượng.
    - Nếu là biến dùng để đánh dấu trong vòng lặp: index, chi\_so, ...
    - Nếu là biến dùng để ghi lại một giá trị gì đấy: tên có thể giúp mình hình dung được nội dung mà nó đang lưu
    - Nếu là mảng dùng để lưu dữ liệu dưới dạng danh sách: ...List, listOf...
  + Gán giá trị: gán giá trị sao cho có ích cho việc code về sau.
* Trật tự code: Code đẳng sau sẽ viết đè lên code trước (vì chúng được thực thi theo hướng tuần tự)
* Thả dữ liệu tự do vào string: sử dụng lệnh ${value}, để thả trực tiếp các biến có kiểu dữ liệu khác string vào chuỗi.
  + **Lưu ý:** Xâu string phải được khai báo bằng cặp dấu huyền ``
* Gõ comment cho Javascript: có 2 cách để gõ comment cho Javascript
  + Cách 1: Nhấn tổ hợp phím CTRL + ‘/’. Loại comment này sử dụng để làm chú thích ngắn, chỉ áp dụng được trên 1 dòng.
  + Cách 2: Gõ phím ‘/’ + ‘\*’ + ‘\*’. Khác với cách 1, chúng có thể comment một lúc nhiều dòng, dùng để vô hiệu hóa dòng lệnh thừa mà không cần xóa nó hoặc liệt kê một dãy chú thích dài.
  1. DOM
* Khái niệm
  + DOM (Document Object Model) là thứ giúp Javascript có thể truy cập và thay đổi các phần tử của HTML.
  + Khi tạo một trang web, DOM được tạo cùng với chúng. DOM phân bố theo sơ đồ hình cây như hình sau:



* Các câu lệnh về DOM:
* Document.getElementById(): Xác định phần tử chứa id trùng khớp với id được khai báo trong file HTML
* Document.getElementsByClass(): Xác định tất cả các phần tử chứa class trùng khớp với class được khai báo.
* Document.getElementsByTagName(): Xác định tất cả các phần tử chứa tên thẻ trùng khớp với tên được khai báo.
* Element.innerText = “<xâu\_kí\_tự>”: Thay đổi giá trị văn bản cho một phần tử chứa văn bản.
* Element.innerHTML = “<xâu\_kí\_tự>”: Thay đổi các PHẦN TỬ con của một phần tử. Viết giống như khi viết bên HTML
* Document.Element.style.<thuộc\_tính\_CSS> = “<giá\_trị\_thuộc\_tính>”: Thay đổi thuộc tính CSS của phần tử.
* Document.querySelector(): dùng để tìm các phần tử khớp với thứ được khai báo.
  + Đối với class: Document.querySelector(“<tên\_tag>. <tên\_class>”);
  + Đối với Id: Document.querySelector(“<tên\_tag># <tên\_class>”);
* Lệnh tạo sự kiện: là lệnh được thực thi khi một hay nhiều phần tử xảy ra một sự kiện gì đấy(xem danh sách sự kiện mục e)
  + Cú pháp: <đối\_tượng\_HTML>.addEventListener(‘<xâu\_kí\_tự\_của\_sự\_kiện>’, <lệnh xử lí khi sự kiện xảy ra>);
  + Điểm đặc biệt
    - Hàm này có biến đặc trưng là event, khi chúng ta khai báo lệnh xử lí khi sự kiện xảy ra có tham số tên event, chúng sẽ truyền trực tiếp sự kiện đó vào biến event, nhờ đó ta có thể làm nhiều thứ khác.
  + Thuộc tính:
    - Target: chính là đối tượng HTML mà ta đã khai báo ở trên. Cú pháp sử dụng: event.target.<thuộc tính khác của thẻ HTML được khai báo>
* Lệnh ngăn chặn sự kiện mặc định: event.preventDefault();
  + Được sử dụng để ngăn chặn sự kiện tiếp theo xảy ra theo mặc định sau khi một sự kiện nào đó xảy ra.
* Thuộc tính để lấy giá trị của một input: value
  + Được sử dụng để lấy giá trị của một input
  + Cú pháp: <DOM của input>.value (trả về dạng cơ bản là string)
  + Lưu ý: Đối với radio có đặc biệt hơn
  1. Github
* Nơi thao tác: Terminal( của Window hoặc Visual Studio Code đều được )
* Lệnh tạo thư mục git cho thư mục chứa project(nếu chưa có):
  + Cú pháp: git init
* Lệnh khởi tạo và cập nhật thư mục chứa dự án:
  + B1: Nhập lệnh: git pull (nếu muốn cập nhật lại các file từ đầu)
  + B2: Nhập lệnh: git add .
  + B3: Nhập lệnh: git commit -m “<chú\_thích>”
  + B4: Nhập lệnh: git push origin <tên\_nhánh> (nếu là lần đầu tiên đẩy lên github, sử dụng lệnh git push -u origin <tên\_nhánh>)
  + B5: Mở trình duyệt và kiểm tra
  1. Local storage(Bộ nhớ cục bộ)(LS)
* Khái niệm:
  + Local storage là một loại bô nhớ client (chỉ lưu và sử dụng được ở từng trình duyệt). Khi truy cập vào trang web, local storage tự động được tạo ra. Mỗi local storage có dung lượng tối đa là 5 MB.
  + Local storage được lưu ở dạng JSON.
* Các câu lệnh của local storage:
  + JSON.stringify(<tên key>): chuyển đổi dữ liệu được lưu bằng JS sang JSON, được thực hiện cho mảng hay kiểu object trước khi được lưu trữ trên LS.
  + JSON.parse(localStorage.getItem(“<tên\_key>”)): chuyển dữ liệu từ JSON trên LS sang JS.
  + localStorage.setItem(“<key\_value>” , “<giá\_trị\_hoặc\_biến>”): dùng để tạo một giá trị trong local storage, có kiểu key và value, lưu dưới dạng JSON
  + localStorage.getItem(“<tên\_key\_trên\_LS>”): dùng để lấy giá trị trong LS đã được lưu sẵn dưới dạng JSON về JS.