Bài 8
Mảng

Module: BOOTCAMP PREPARATION

Kiểm tra bài trước

Hỏi và trao đổi về các khó khăn gặp phải trong bài “Cáu trúc lặp”

Tóm tắt lại các phần đã học từ bài “Cấu trúc lặp”

Mục tiêu

- Trình bày được khái niệm mảng
- Mô tả được cú pháp khai báo và sử dụng mảng
- Mô tả được cách sử dụng vòng lặp for để duyệt mảng
- Mô tả được cách sử dụng vòng lặp for/in để duyệt mảng
- Khai báo và sử dụng được mảng một chiều
- Trình bày được khái niệm mảng đa chiều
- Mô tả được cú pháp khai báo và sử dụng mảng hai chiều
- Sử dụng for để làm việc với mảng
- Sử dụng for-in để làm việc với mảng

Thảo luận

Khai báo mảng

Khởi tạo mảng

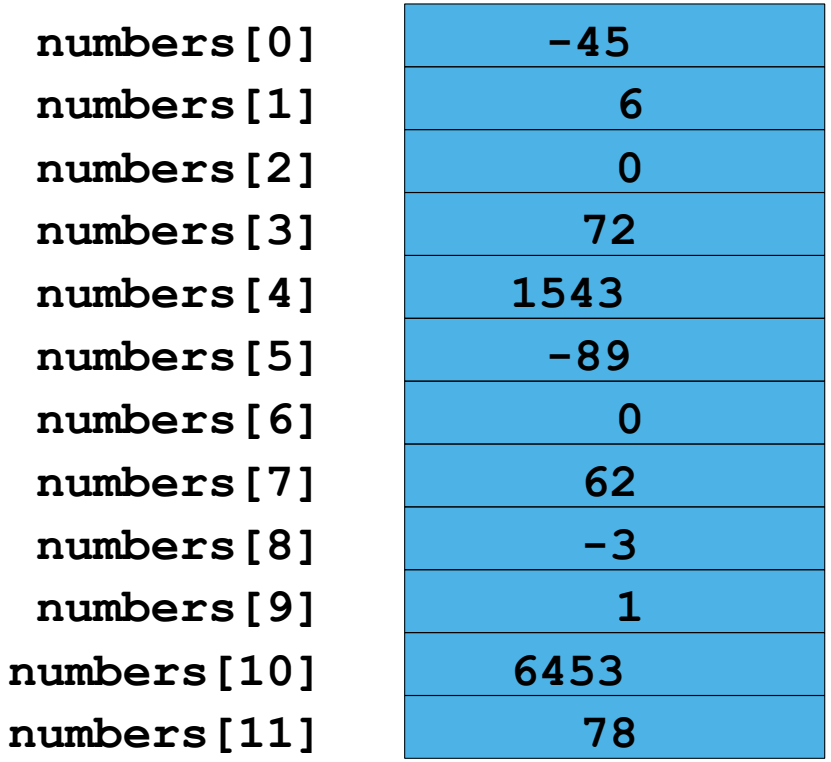
Gán giá trị cho mảng

Lưu trữ nhiều dữ liệu

- Trong các chương trình máy tính, có những trường hợp chúng ta phải lưu trữ rất nhiều dữ liệu
- Chẳng hạn, cần lưu trữ danh sách tên của hằng trăm sinh viên, cùng điểm thi của từng người
- Việc khai báo hằng trăm biến để lưu trữ các dữ liệu kiểu này rất mất thời gian và không thực tế
- JavaScript và phần lớn các ngôn ngữ khác hỗ trợ một cấu trúc dữ liệu là mảng để giải quyết các tình huống này

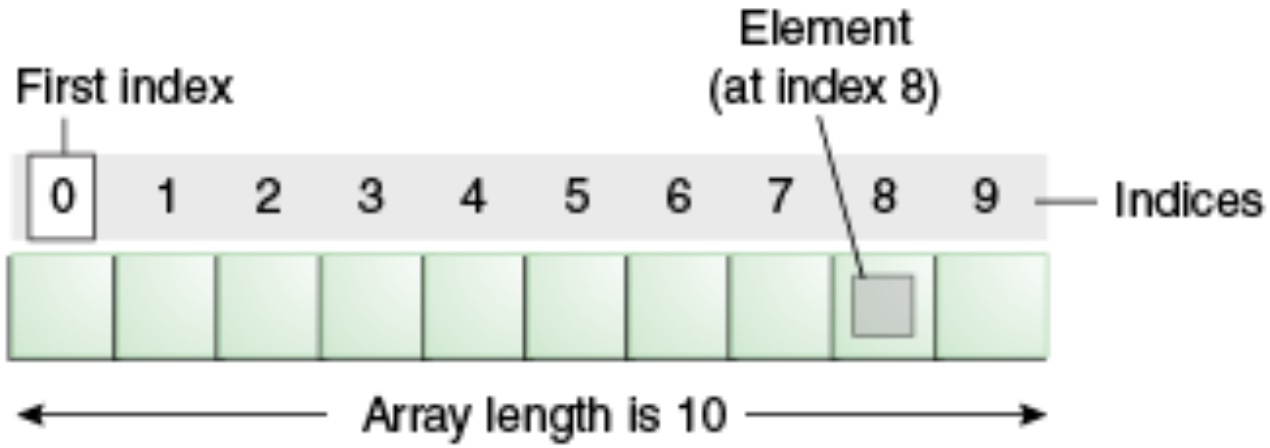
Mảng là gì

- Mảng là một loại biến đặc biệt, có thể lưu được nhiều giá trị thay vì chỉ một giá trị như các biến thông thường
- Mỗi giá trị trong mảng được gọi là một phần tử
- Các phần tử được lưu trữ ở các vị trí kế tiếp nhau trong bộ nhớ
- Chẳng hạn, mảng numbers chứa 12 phần tử là các số nguyên



Các khái niệm của mảng

- Tên mảng: Tuân thủ theo quy tắc đặt tên của biến
- Phần tử: Các giá trị được lưu trữ trong mảng
- Chỉ số: Vị trí của các phần tử trong mảng. Chỉ số bắt đầu từ 0.
- Độ dài: Số lượng các phần tử của mảng



Khai báo mảng: Cách 1

- Cú pháp:

```
var array_name = [item1, item2, ...];
```

- Ví dụ:

```
var cars = ["Saab", "Volvo", "BMW"];
```

Khai báo mảng: Cách 2

- Sử dụng từ khoá new
- Cú pháp:

```
var array_name = new Array(item1, item2, ...) ;
```

- Ví dụ:

```
var cars = new Array("Saab", "Volvo", "BMW");
```

Demo

Khai báo mảng sử dụng 2 cách

Thảo luận

Khai báo mảng

Khởi tạo mảng

Gán giá trị cho mảng

Phần tử và chỉ số của mảng

- Mỗi phần tử (item) được xác định bằng một số thứ tự còn gọi là chỉ số (index) duy nhất trong mảng
- Chỉ số là một số nguyên dương
- Chỉ số của phần tử đầu tiên là 0
- Chỉ số của phần tử cuối cùng là n – 1, trong đó n là độ dài của mảng
- Có thể truy xuất đến phần tử của mảng thông qua chỉ số

Truy xuất phần tử trong mảng

- Mỗi phần tử trong mảng được thao tác giống như một biến
- Truy xuất các phần tử thông qua chỉ số đặt trong dấu []
- Ví dụ, gán giá trị cho một phần tử đầu tiên của mảng

```
cars[0] = "Opel";
```

- Ví dụ, lấy giá trị của một phần tử đầu tiên của mảng

```
var name = cars[0];
```

Vòng lặp for-in

- Vòng lặp for-in (còn gọi là *enhanced for*) được sử dụng để duyệt qua các phần tử của một collection, chẳng hạn như mảng, danh sách…
- Cú pháp:

```
for (var in collection) {

}
```

Trong đó:

- var: Biến đại diện lần lượt cho từng phần tử của collection trong mỗi lần lặp
- collection: đối tượng cần lặp

for-in: Ví dụ

- Duyệt qua các phần tử của một mảng:

```
var array = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10];

for (var element in array) {
    console.log("element = " + array[element]);
}
```

- Tương đương với câu lệnh for:

```
for(var i = 0; i < numbers.length; i++){
    console.log("element = " + numbers[i]);
}
```

Duyệt các phần tử của mảng với for

- Sử dụng vòng lặp for để duyệt qua tất cả các phần tử của mảng

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

BMW
Volvo
Saab
Ford
Fiat
Audi

Khác biệt khi sử dụng for và for-in

- Có thể sử dụng index khi duyệt bằng for, không có index nếu sử dụng for-in
- Có thể sử dụng for để duyệt theo các chiều khác nhau (từ đầu đến cuối, từ cuối đến đầu…)
- For in chỉ duyệt từ đầu đến cuối mảng

Các thuộc tính và phương thức của mảng

Độ dài của mảng

- Thuộc tính *length* là độ dài của mảng
- Ví dụ:

```
var fruits = ["Banana", "Orange", "Apple", "Mango"];
alert(fruits.length);
```

Có giá trị là 4

Thêm phần tử vào mảng: push()

- Phương thức push() cho phép thêm phần tử vào cuối mảng
- Ví dụ:

```
var fruits = ["Banana", "Orange", "Apple", "Mango"];
fruits.push("Lemon");
```

Phần tử mới được thêm vào tại vị trí 4

Thêm phần tử vào mảng: length

- Có thể thêm phần tử vào mảng bằng cách gán giá trị cho phần tử ở vị trí length
- Ví dụ:

```
var fruits = ["Banana", "Orange", "Apple", "Mango"];
fruits[fruits.length] = "Lemon";
```

Độ dài 4

Phần tử mới được thêm vào tại vị trí 4

join()

- Phương thức join() chuyển tất cả các phần tử trong mảng thành chuỗi và nối chúng lại với nhau
- Ví dụ:

```
var a = [1, 2, 3];     // Create a new array with these three elements
a.join();              // => "1,2,3"
a.join(" ");           // => "1 2 3"
a.join("");            // => "123"
var b = new Array(10); // An array of length 10 with no elements
b.join('-')            // => '---------': a string of 9 hyphens
```

reverse()

- Phương thức reverse() đảo ngược các phần tử trong mảng, trả về một mảng các phần tử bị đảo ngược
- Ví dụ:

```
var a = [1,2,3];
a.reverse().join()  // => "3,2,1" and a is now [3,2,1]
```

sort()

- Sắp xếp các phần tử của mảng sử dụng phương thức sort()
- Ví dụ - 1:

```
var a = new Array("banana", "cherry", "apple");
a.sort();
var s = a.join(", ");  // s == "apple, banana, cherry"
```

sort()

- Ví dụ - 2:

```
var a = [33, 4, 1111, 222];
a.sort();                       // Alphabetical order: 1111, 222, 33, 4
a.sort(function(a,b) {          // Numerical order: 4, 33, 222, 1111
   return a-b;                  // Returns &lt; 0, 0, or &gt; 0, depending on order
});
a.sort(function(a,b) {          // Reverse numerical order
   return b-a;
});
```

sort()

- Ví dụ - 3:

```
a = ['ant', 'Bug', 'cat', 'Dog']
a.sort();                    // case-sensitive sort: ['Bug','Dog','ant',cat']
a.sort(function(s,t) {   // Case-insensitive sort
            var a = s.toLowerCase();
            var b = t.toLowerCase();
            if (a < b) return -1;
            if (a > b) return 1;
            return 0;
        });                  // => ['ant','Bug','cat','Dog']
```

concat()

- Nối các phần tử của hai mảng với nhau. Trả về mảng mới chứa phần tử của mảng gốc và mảng được nối.
- Ví dụ:

```
var a = [1,2,3];
a.concat(4, 5)          // Returns [1,2,3,4,5]
a.concat([4,5]);        // Returns [1,2,3,4,5]
a.concat([4,5],[6,7])   // Returns [1,2,3,4,5,6,7]
a.concat(4, [5,[6,7]])  // Returns [1,2,3,4,5,[6,7]]
```

push() và pop()

- Hai phương thức push() và pop() hoạt động giống như cấu trúc ngăn sếp (stack – first in, last – out):
  - Phương thức push() để nối một hoặc nhiều phần tử vào cuối mảng
  - Phương thức pop() xoá đi phần tử cuối cùng của mảng, làm giảm kích thước của mảng, trả về gía trị của phần tử bị xoá.
- Ví dụ:

```
var stack =  [];          // stack: []
stack.push(1,2);          // stack: [1,2]     Returns 2
stack.pop();              //stack: [1]        Returns 2

stack.push(3);            // stack: [1,3]     Returns 2
stack.pop();              // stack: [1]       Returns 3
stack.push([4,5]);        // stack: [1,[4,5]] Returns 2
stack.pop()  ;            // stack: [1]       Returns [4,5]
stack.pop();              // stack: []        Returns 1
```

shift() và unshift()

- Phương thức shift() để nối phần tử vào đầu mảng
- Phương thức unshift() đế xoá phần tử đầu tiên của mảng
- Ví dụ:

```
var a = [];              // a:[]
a.unshift(1);            // a:[1]          Returns: 1
a.unshift(22);           // a:[22,1]       Returns: 2
a.shift();               // a:[1]          Returns: 22
a.unshift(3,[4,5]);      // a:[3,[4,5],1]  Returns: 3
a.shift();               // a:[[4,5],1]    Returns: 3
a.shift();               // a:[1]          Returns: [4,5]
a.shift();               // a:[]           Returns: 1
```

toString()

- Phương thức chuyển toàn bộ các phần tử trong mảng sang dạng chuỗi.
- Ví dụ:

```
[1,2,3].toString()          // Yields '1,2,3'
["a", "b", "c"].toString()  // Yields 'a,b,c'
[1, [2,'c']].toString()     // Yields '1,2,c'
```

Demo

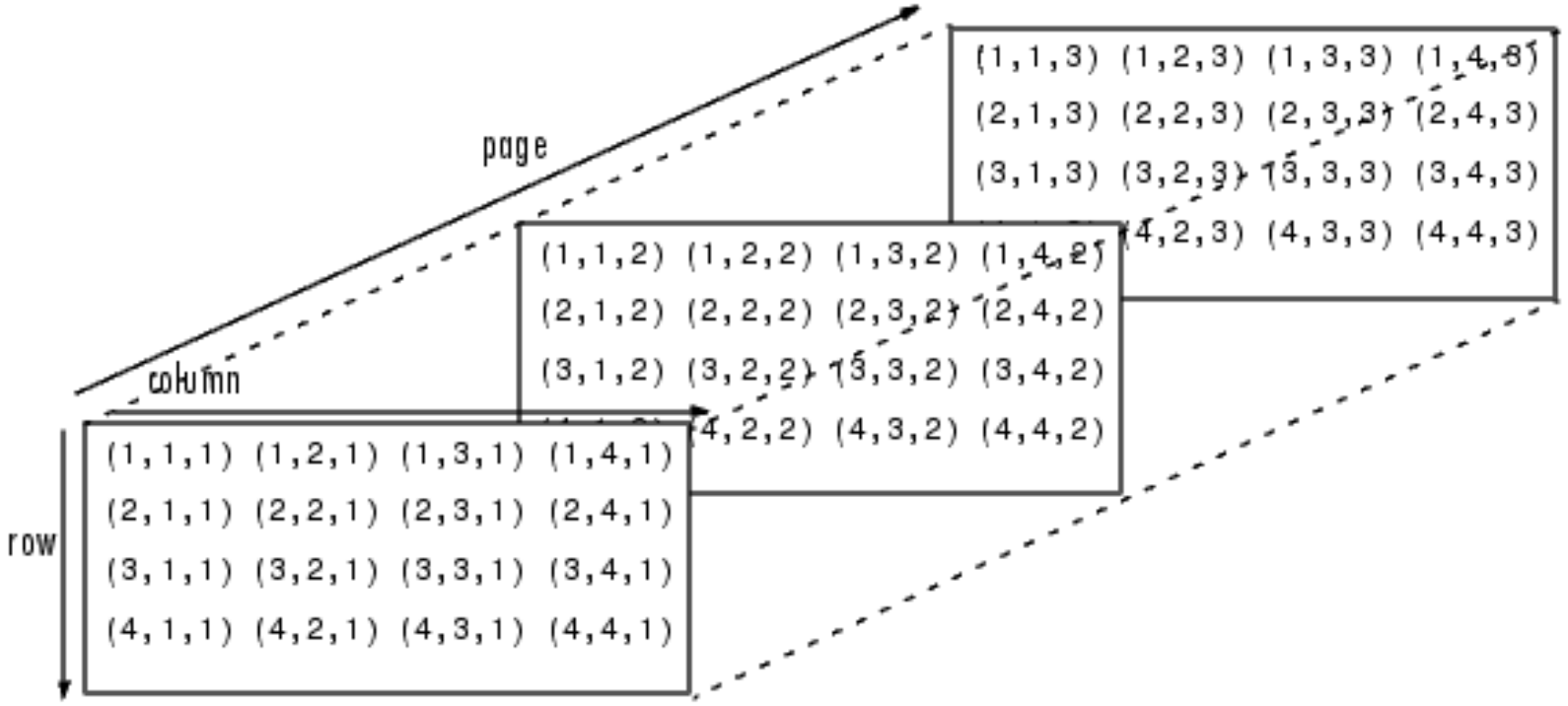
Một số phương thức thao tác với mảng

Thảo luận

Mảng hai chiều

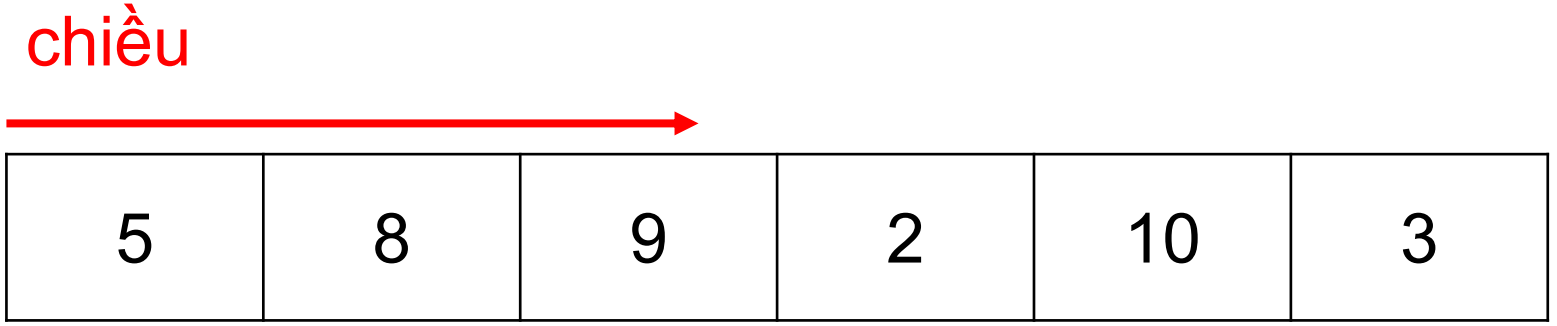
Mảng đa chiều

- Mảng đa chiều là mảng có các phần tử là các mảng khác
- Có thể có mảng 2 chiều, 3 chiều… hoặc nhiều hơn
- Mảng càng nhiều chiều thì độ phức tạp khi xử lý càng cao



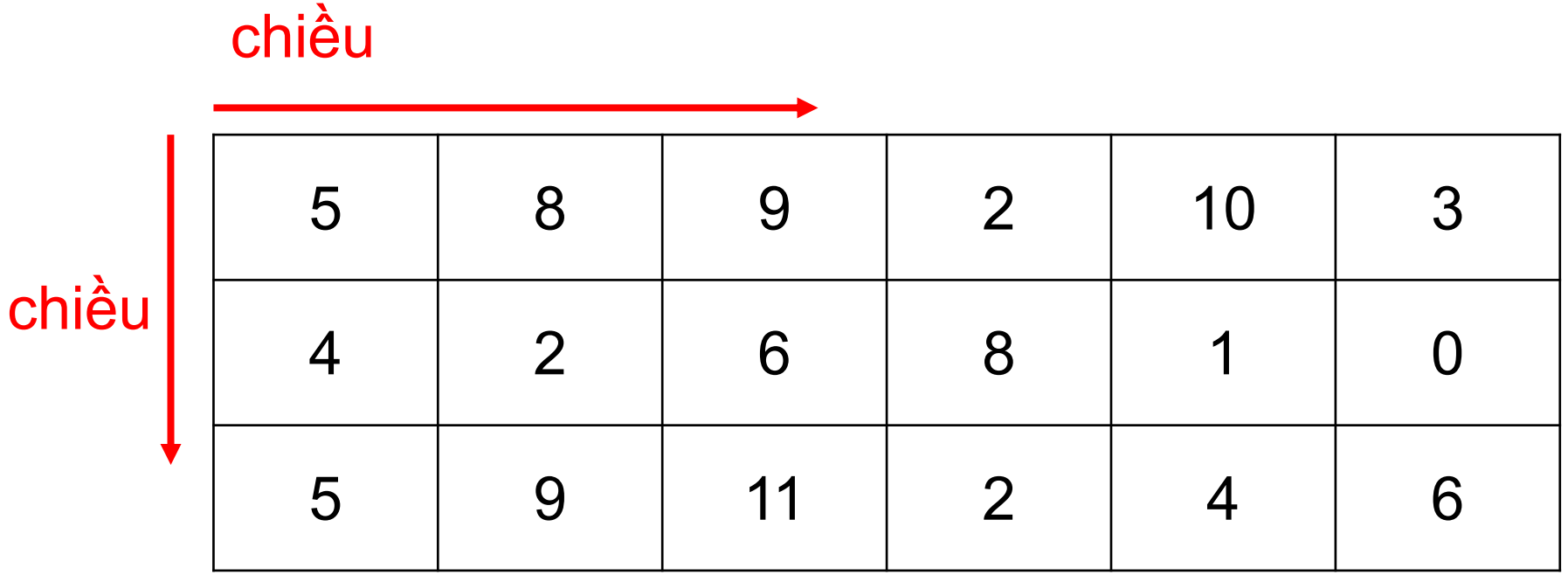
Số chiều của mảng

- Mảng một chiều cần 1 chỉ số để xác định vị trí của phần tử mảng



Số chiều của mảng

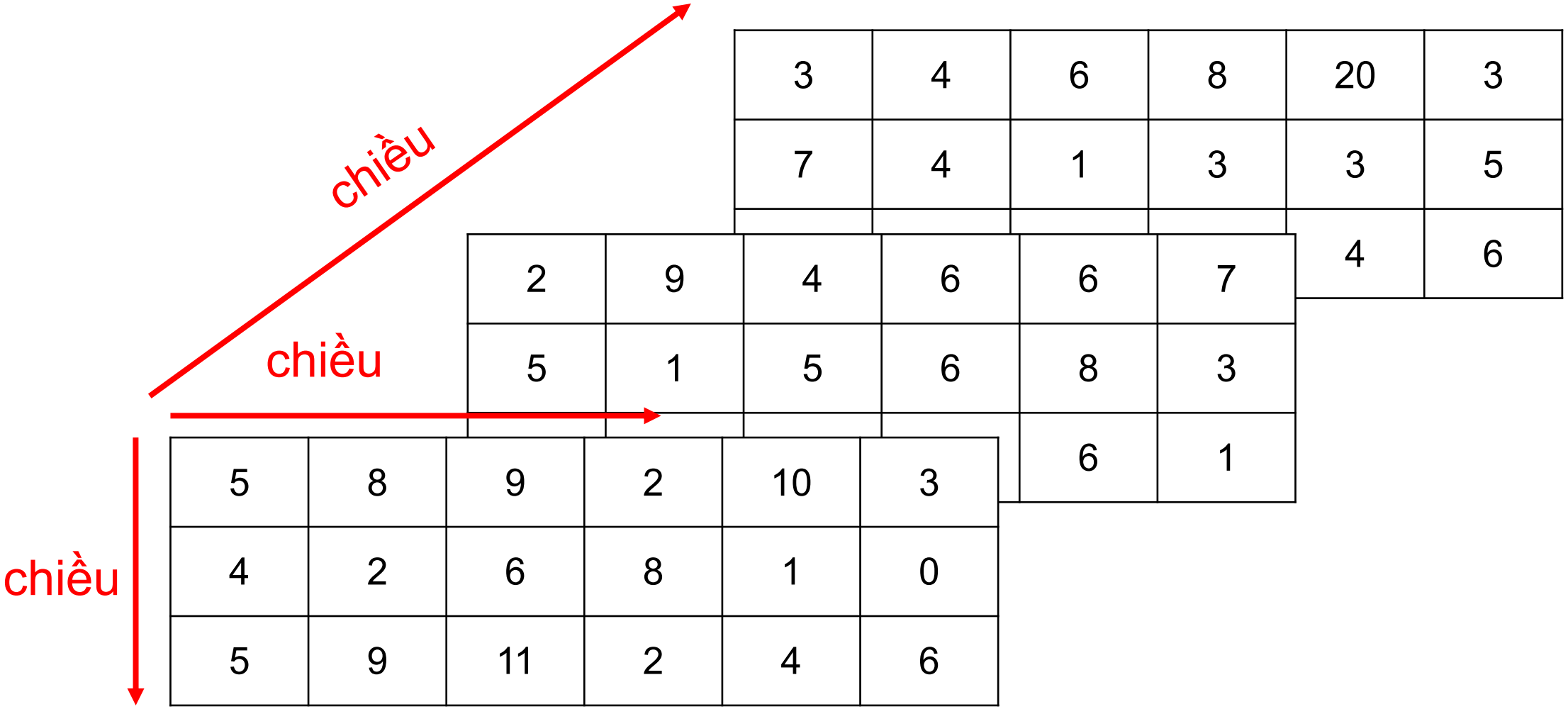
- Mảng hai chiều cần 2 chỉ số để xác định vị trí của phần tử mảng



5	8	9	2	10	3
4	2	6	8	1	0
5	9	11	2	4	6

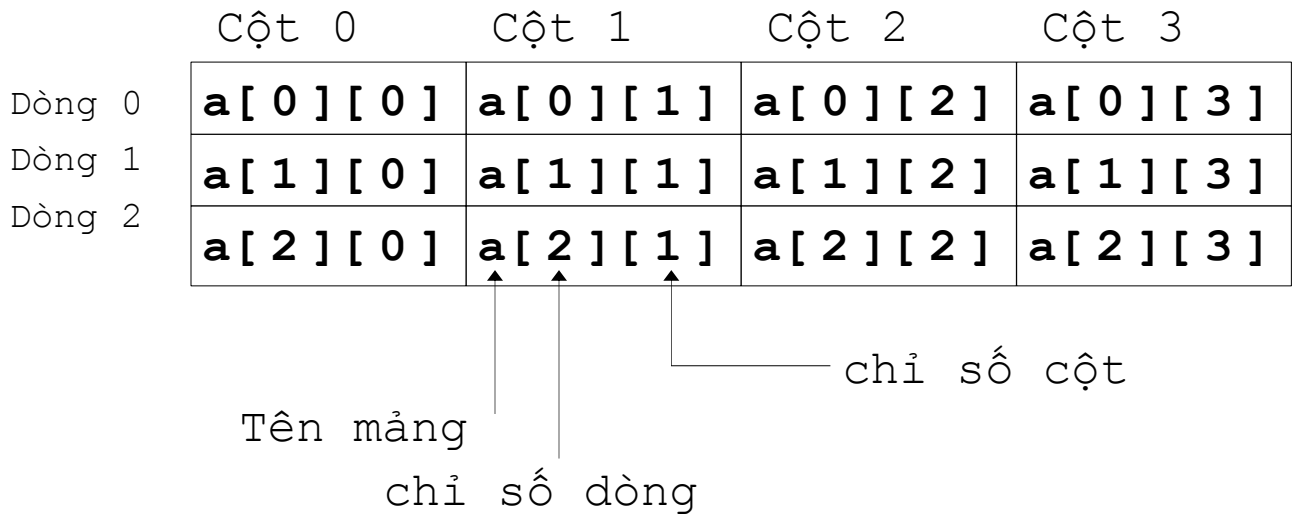
Số chiều của mảng

- Mảng 3 chiều cần 3 chỉ số để xác định vị trí của phần tử mảng



Mảng hai chiều

- Mảng hai chiều là mảng nhiều chiều được sử dụng phổ biến
- Mảng hai chiều bao gồm các phần tử, mỗi phần tử là một mảng một chiều.
- Mảng hai chiều là một bảng gồm n dòng và m cột:



- Số phần tử trong mảng là n * m

Mảng hai chiều: Ví dụ

- Ví dụ: mảng 2 chiều gồm 3 dòng, mỗi dòng có 5

index

	0	1	2	3	4
0	c[0][0] = 11	c[0][1] = 2	c[0][2] = 7	c[0][3] = 9	c[0][4] = 5
1	c[1][0] = 23	c[1][1] = 34	c[1][2] = 65	c[1][3] = 67	c[1][4] = 2
2	c[2][0] = 38	c[2][1] = 28	c[2][2] = 43	c[2][3] = 72	c[2][4] = 65

- Tổng số phần tử trong mảng 3 * 5 = 15
- **Index:** các chỉ số dòng và cột

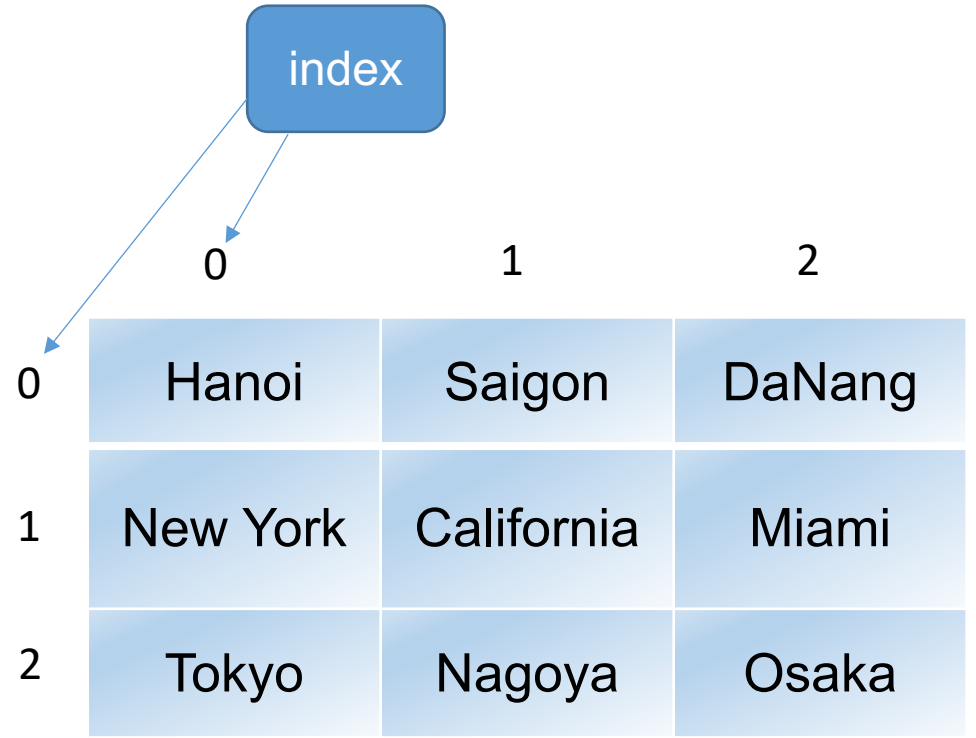
Phần tử và chỉ số mảng 2 chiều

- Một phần tử được xác định bằng chỉ số dòng và chỉ số cột trong mảng
- Ví dụ mảng *numbers* với 3 cột x 5 dòng, có tổng cộng15 phần tử lần lượt là:
  - *numbers[0][0]*
  - *numbers[1][0]*
  - *...*
  - *numbers[2][4]*

Khởi tạo mảng hai chiều: Cách 1

- Sử dụng dấu ngoặc vuông để khởi tạo mảng với các phần tử
- Ví dụ, khởi tạo mảng có 3 dòng và 3 cột:

```
var cities = [
    ['Hanoi', 'Saigon', 'DaNang'],
    ['New York', 'California', 'Miami'],
    ['Tokyo', 'Nagoya', 'Osaka']
];
```



Khởi tạo mảng hai chiều: Cách 2

- Sử dụng từ khoá new để khởi tạo mảng
- Ví dụ, khởi tạo mảng với 3 dòng và 3 cột

```
var cities = new Array(3);
for(var i = 0; i < 3; i++){
    cities[i] = new Array(3);
}
cities[0][0] = 'Hanoi'; cities[0][1] = 'Saigon'; cities[0][2] = 'DaNang';
cities[1][0] = 'New York'; cities[1][1] = 'California'; cities[1][2] = 'Miami';
cities[2][0] = 'Tokyo'; cities[2][1] = 'Nagoya'; cities[2][2] = 'Osaka';
```

Duyệt mảng đa chiều

- Sử dụng vòng lặp lồng nhau để duyệt mảng đa chiều
- Sử dụng 2 vòng lặp lồng nhau để duyệt mảng 2 chiều
- Ví dụ:

```
for(var i = 0; i < cities.length; i++){
    for(var j = 0; j < cities[i].length; j++){
        document.write(cities[i][j] + '<br/>');
    }
}
```

Demo

Cách sử dụng mảng hai chiều

Thảo luận

Các thao tác với mảng hai chiều

Khởi tạo giá trị ngẫu nhiên

- Ví dụ:

```
var matrix = new Array(10).fill(new Array(10));

for (var row = 0; row < matrix.length; row++) {
    for (var column = 0; column < matrix[row].length; column++) {
        matrix[row][column] = Math.floor((Math.random() * 100) + 1);
    }
}
```

Hiển thị các phần tử của mảng

- Ví dụ:

```
for (var row = 0; row < matrix.length; row++) {
    for (var column = 0; column < matrix[row].length; column++) {
        console.log(matrix[row][column] + " ");
    }
}
```

Tính tổng các phần tử số

- Ví dụ:

```
var total = 0;
for (var row = 0; row < matrix.length; row++) {
    for (var column = 0; column < matrix[row].length; column++) {
        total += matrix[row][column];
    }
}
```

Tính tổng các phần tử số theo từng cột

- Ví dụ:

```
for (var column = 0; column < matrix[0].length; column++) {
   var total = 0;
   for (var row = 0; row < matrix.length; row++)
   total += matrix[row][column];
   console.log("Sum for column " + column + " is " + total);
}
```

Tìm hàng có tổng lớn nhất

```
var maxRow = 0;
var indexOfMaxRow = 0;

for (var column = 0; column < matrix[0].length; column++) {
   maxRow += matrix[0][column];
}
for (var row = 1; row < matrix.length; row++) {
   var totalOfThisRow = 0;
   for (var column = 0; column < matrix[row].length; column++)
   totalOfThisRow += matrix[row][column];
   if (totalOfThisRow > maxRow) {
      maxRow = totalOfThisRow;
      indexOfMaxRow = row;
   }
}
console.log("Row " + indexOfMaxRow + " has the maximum sum of " + maxRow);
```

Trộn ngẫu nhiên các phần tử

- Ví dụ:

```
for (var i = 0; i < matrix.length; i++) {
    for (var j = 0; j < matrix[i].length; j++) {
        var i1 = (int)(Math.random() * matrix.length);
        var j1 = (int)(Math.random() * matrix[i].length);
        var temp = matrix[i][j];
        matrix[i][j] = matrix[i1][j1];
        matrix[i1][j1] = temp;
    }
}
```

Tóm tắt bài học

- Mảng cho phép lưu trữ nhiều giá trị cùng kiểu
- Các khái niệm của mảng: Tên mảng, kiểu dữ liệu, kích thước, phần tử, chỉ số
- Tên của mảng tuân theo quy tắc của tên biến
- Chỉ số của phần tử đầu tiên là 0
- Chỉ số của phần tử cuối cùng là length – 1
- Có thể sử dụng vòng lặp for và for-each để duyệt mảng
- Để sao chép mảng thì cần sao chép lần lượt từng phần tử của mảng

Tóm tắt bài học

- Mảng đa chiều được sử dụng phổ biến là 2 chiều
- Số lượng "chiều" của mảng bằng với số lượng chỉ số để truy xuất đến một phần tử của mảng
- Ragged array là mảng hai chiều trong đó kích thước của các dòng là không bằng nhau
- Có thể sử dụng 2 vòng lặp lồng nhau để duyệt qua các phần tử của mảng hai chiều

Hướng dẫn

Hướng dẫn làm bài thực hành và bài tập

Chuẩn bị bài tiếp theo: *Hàm*