Buổi 8
Vòng lặp

I. Khái niệm vòng lặp

Trong lập trình, khi giải quyết 1 bài toán, đôi lúc có những thao tác, hành động lặp đi lặp lại nhiều lần.

Ví dụ:

- Đếm số lượng ước số của 100.
- Xuất ra màn hình 10 số nguyên tố đầu tiên kể từ số 30.

Khi gặp những thao tác lặp đi lặp lại, chúng ta sử dụng vòng lặp (*loop*).

II. Vòng lặp for

Vòng lặp `for` cho phép chương trình thực hiện lặp đi lặp lại một công việc với số lần lặp xác định. Để làm được việc này, cần 1 biến để đếm lần chạy hiện tại là lần thứ mấy.

Cú pháp:

```
for (<khởi tạo biến đếm>; <điều kiện>; <tăng/giảm biến đếm>) {
    // Các câu lệnh cần thực hiện
}
```

Trong đó:

- `<khởi tạo biến đếm>` là biểu thức dùng để tạo và gán giá trị ban đầu cho biến đếm (thường là biến `i`). Sau khi chạy câu lệnh này, vòng lặp sẽ kiểm tra `<điều kiện>`.
- `<điều kiện>` nếu đúng thì các câu lệnh trong vòng lặp sẽ được chạy, sau đó sẽ thực hiện `<tăng/giảm biến đếm>`. Ngược lại, vòng lặp sẽ kết thúc.
- `<tăng/giảm biến đếm>` là biểu thức để tăng hoặc giảm biến đếm. Sau khi thực hiện xong câu lệnh này, vòng lặp sẽ tiếp tục kiểm tra `<điều kiện>`.
- Cả 3 biểu thức đều không bắt buộc, nhưng nếu không có thì vẫn phải giữ dấu `;`.

Ví dụ: Xuất ra màn hình 10 câu "Xin chao":

```
for (i = 1; i <= 10; i++) {
    console.log("Xin chào");
}
```

III. Vòng lặp while

Vòng lặp `while` cho phép chương trình thực hiện lặp đi lặp lại một công việc miễn là một điều kiện nào đó còn đúng.

Cú pháp:

```
while (<điều kiệu>) {
    // Các câu lệnh cần thực hiện nếu điều kiện đúng
}
```

Trong đó: `<điều kiện>` nếu đúng thì các câu lệnh trong vòng lặp sẽ được chạy. Ngược lại, vòng lặp sẽ kết thúc.

Ví dụ: Xuất ra màn hình các số nguyên từ 1 đến 10:

```
var i = 1;
while (i <= 10) {
```

```
        console.log(i);
        i++;
    }
```

IV. Vòng lặp do-while

Vòng lặp `do-while` cũng cho phép chương trình thực hiện lặp đi lặp lại một công việc miễn là một điều kiện nào đó còn đúng, tương tự như vòng lặp `while`. Tuy nhiên, vòng lặp `do-while` sẽ thực hiện công việc trước khi kiểm tra điều kiện.

Cú pháp:

```
do {
    // Các câu lệnh cần thực hiện
} while (<điều kiệu>);
```

Trong đó: Vòng lặp sẽ thực hiện các câu lệnh trước, sau đó kiểm tra `<điều kiện>`. Nếu `<điều kiện>` đúng thì các câu lệnh trong vòng lặp sẽ tiếp tục được chạy. Ngược lại, vòng lặp sẽ kết thúc.

Ví dụ: Xuất ra màn hình các số nguyên từ 1 đến 10:

```
var i = 1;
do {
    console.log(i);
    i++;
} while (i <= 10);
```

V. Điều khiển vòng lặp

1. Câu lệnh break

Đôi khi, việc viết biểu thức điều kiện để dừng vòng lặp là rất khó, hoặc biểu thức điều kiện có nhiều trường hợp khác nhau, hoặc chỉ đơn giản là muốn dừng vòng lặp trước thời hạn. Khi đó chúng ta có thể dùng câu lệnh `break;` để thoát khỏi vòng lặp.

Ví dụ: Tìm số k lớn nhất sao cho tổng các số từ 1 đến k không vượt quá n cho trước:

```
var k = 1, sum = 0;
while (true) {
    sum += k;
    if (sum > n) {
        break;
    }
    else {
        k++;
    }
}
alert("Số k cần tìm là " + (k - 1));
```

2. Câu lệnh continue

Khi vòng lặp đang chạy, câu lệnh `continue;` có tác dụng bỏ qua lần chạy hiện tại và ngay lập tức chuyển sang lần chạy tiếp theo.

Ví dụ: Xuất ra màn hình các số nguyên dương từ 1 đến 10, ngoại trừ số 7:

```
for (i = 1; i <= 10; i++) {
    if (i == 7) {
        continue;
    }
    else {
        console.log(i);
```

```
        }
    }
```

VI. Vòng lặp vô tận

Khi sử dụng vòng lặp, cần chú ý viết biểu thức điều kiện cho đúng, đảm bảo rằng vòng lặp có thể dừng được. Một vòng lặp không thể dừng được (vì biểu thức điều kiện luôn đúng) gọi là vòng lặp vô tận (*infinite loop*).

Ví dụ:

```
for (i = 10; i >= 0; i++) {
    console.log(i);
}
while (i <= 10) {
    cout << i << " ";
}
```

VII. Bài tập

Thực hiện các bài tập sau, mỗi bài tập nằm trong 1 trang web riêng.

Vòng lặp for

Yêu cầu chung: Nhập số nguyên dương n.

1. Xuất ra màn hình các số nguyên dương chẵn từ n đến 0.
2. Đếm số lượng số lẻ trong khoảng từ 0 đến n.
3. Tính tổng và tích các số từ 1 đến n.
4. Đếm số lượng ước số của n, và xuất danh sách ước số đó ra màn hình
5. Cho biết n có phải là số nguyên tố hay không.

Vòng lặp while và do-while

6. Nhập 2 số nguyên dương a và b. Xuất ra màn hình bội chung nhỏ nhất của a và b.
7. Nhập 2 số nguyên dương a và b. Xuất ra màn hình ước chung lớn nhất của a và b.

Bài tập chung

Yêu cầu chung: Nhập số nguyên dương n.

8. Tính:

$$S_1 = 1^2 + 2^2 + 3^2 + \ldots + n^2$$

$$S_2 = 1 + \frac{1}{2} + \frac{1}{3} + \frac{1}{4} + \ldots + \frac{1}{n}$$

$$S_3 = \frac{1}{2} + \frac{2}{3} + \frac{3}{4} + \ldots + \frac{n}{n+1}$$

$$S_4 = 1 + (1 + 2) + (1 + 2 + 3) + \ldots + (1 + 2 + 3 + \ldots + n)$$

9. Tính tổng các chữ số của n.
10. Xuất ra màn hình số m là số đảo ngược của n, và cho biết n có phải là số đối xứng hay không.
    Ví dụ: Nhập n = 74 thì xuất ra m = 47 và do đó n không phải là số đối xứng.
11. Xuất ra màn hình n số nguyên tố đầu tiên.
12. Cho biết n có phải số hoàn thiện[1] hay không.
13. Tìm các số nguyên dương có 3 chữ số sao cho tổng bình phương của 3 chữ số đó bằng 25.

[1] Số hoàn thiện là số bằng tổng các ước số của nó (ngoại trừ chính nó). Ví dụ, 6 là số hoàn thiện vì 6 = 1 + 2 + 3.