CÁC CÂU LỆNH LẶP

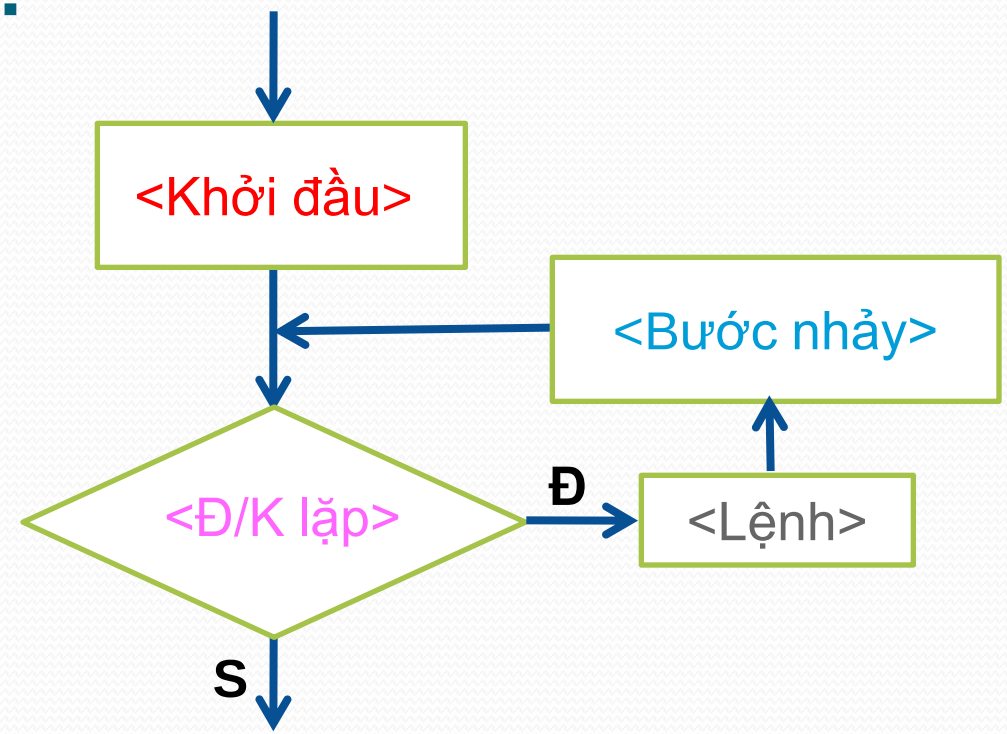
Nội dung

1. Câu lệnh for
2. Câu lệnh while
3. Câu lệnh do… while

Đặt vấn đề

- Ví dụ
  - Viết chương trình xuất các số từ 1 đến 10
  - => Dùng 10 câu lệnh printf
  - Viết chương trình xuất các số từ 1 đến 1000
  - => Dùng 1000 câu lệnh printf!!!
- Giải pháp
  - Sử dụng cấu trúc lặp lại một hành động trong khi còn thỏa một điều kiện nào đó.
  - 3 lệnh lặp: FOR, WHILE, DO… WHILE

Câu lệnh for



for (<Khởi đầu>; <Đ/K lặp>; <Bước nhảy>)

<Lệnh>;

<Khởi đầu>, <Đ/K lặp>, <Bước nhảy>: là biểu thức C bất kỳ có chức năng riêng

<Lệnh>: đơn hoặc khối lệnh.

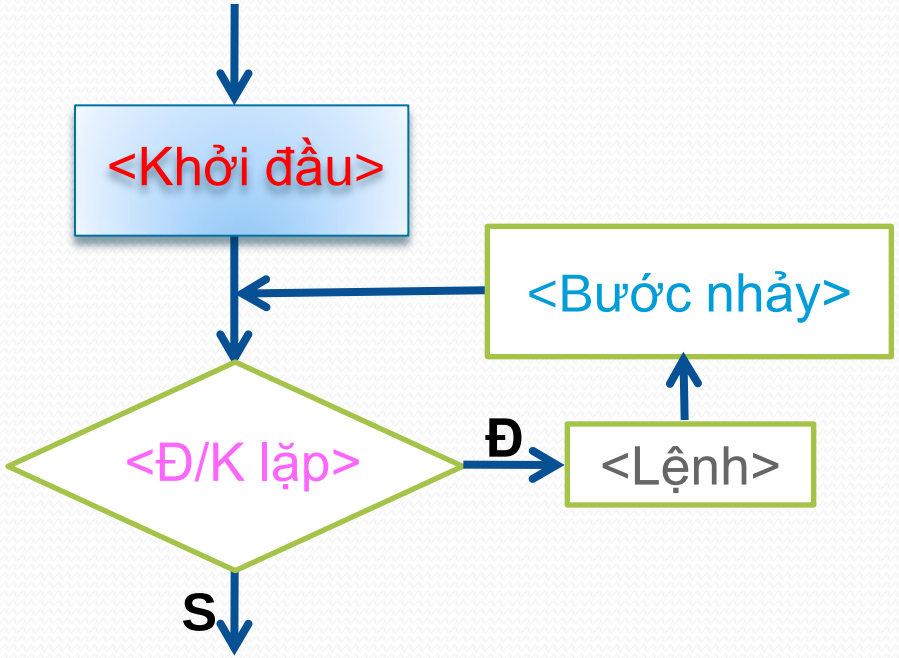
Câu lệnh for

```
main()
{
    int i;
    for (i = 0; i < 10; i++)
        printf("%d\t", i);

    for (int j = 0; j < 10; j = j + 1)
        printf("%d\n", j);

    for (int k = 0; k < 10; k += 2)
    {
        printf("%d", k);
        printf("\n");
    }
}
```

Câu lệnh for - Một số lưu ý

- Trong câu lệnh for, có thể sẽ không có phần <Khởi đầu>

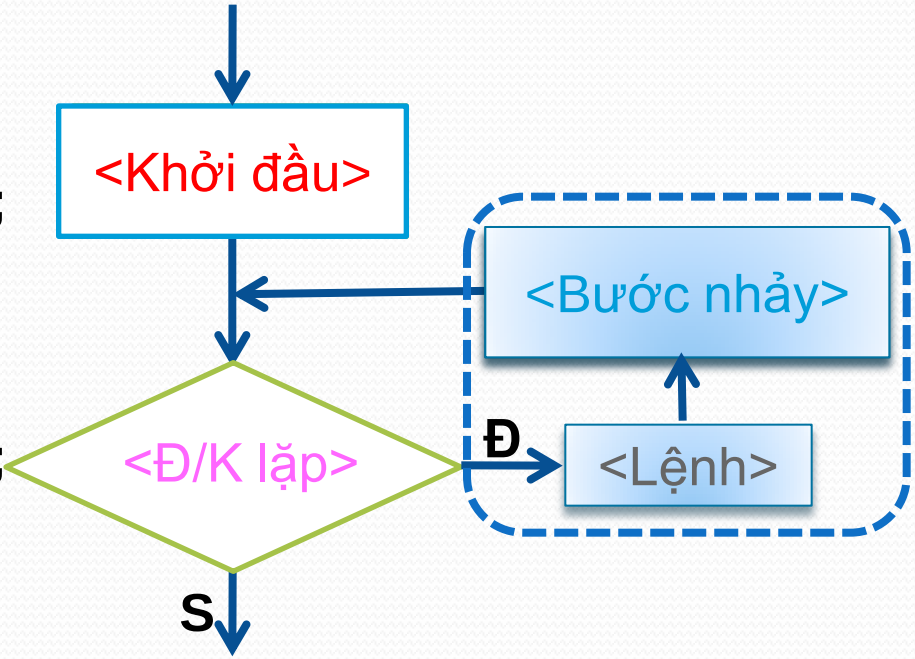
```
int i;
for (i = 0; i < 10; i++)
        printf("%d\n", i);

int i = 0;
for (; i < 10; i++)
        printf("%d\n", i);
```



Câu lệnh for - Một số lưu ý

- Trong câu lệnh for, có thể sẽ không có phần <Bước nhảy>

```
int i;
for (i = 0; i < 10; i++)
        printf("%d\n", i);

for (i = 0; i < 10; )
{
        printf("%d\n", i);
        i++;
}
```



Câu lệnh for - Một số lưu ý

- Trong câu lệnh for, có thể sẽ không có phần <Đ/K lặp>

```
int i;
for (i = 0; i < 10; i++)
        printf("%d\n", i);

for (i = 0; ; i++)
        printf("%d\n", i);

for (i = 0; ; i++)
{
        if (i >= 10)
                break;
        printf("%d\n", i);
}
```

Câu lệnh for - Một số lưu ý

- Lệnh break làm kết thúc câu lệnh.
- Lệnh continue bỏ qua lần lặp hiện tại.

```
for (i = 0; i < 10; i++)
{
        if (i % 2 == 0)
                break;
        printf("%d\n", i);
}

for (i = 0; i < 10; i++)
{
        if (i % 2 == 0)
                continue;
        printf("%d\n", i);
}
```

Câu lệnh for - Một số lưu ý

- Không được thêm ; ngay sau lệnh lệnh for.

  => Tương đương câu lệnh rỗng.

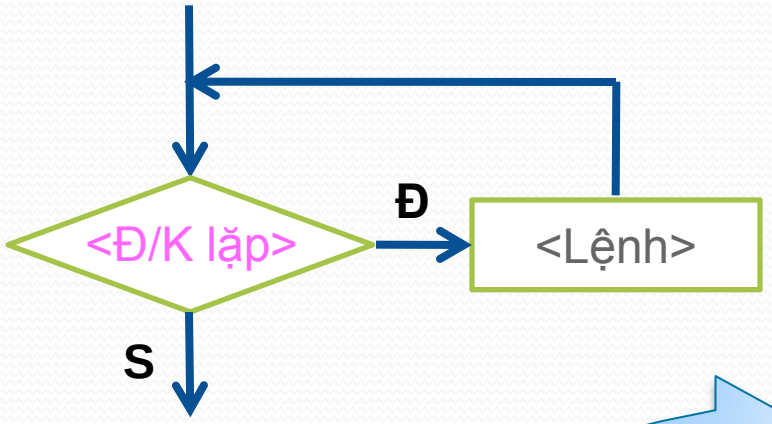
```
for (i = 0; i < 10; i++);
{
        printf("%d", i);
        printf("\n");
}

for (i = 0; i < 10; i++)
{
};
{
        printf("%d", i);
        printf("\n");
}
```

Câu lệnh for - Một số lưu ý

- Các thành phần <Khởi đầu>, <Đ/K lặp>, <Bước nhảy> cách nhau bằng dấu ;
- Nếu có nhiều thành phần trong mỗi phần thì được cách nhau bằng dấu ,

```
for (int i = 1, j = 2; i + j < 10; i++, j += 2)
      printf("%d\n", i + j);
```

Câu lệnh while



while (<Đ/K lặp>**)**

<Lệnh>;

Biểu thức C bất kỳ, thường là biểu thức quan hệ cho kết quả 0 (sai) và != 0 (đúng)

Câu lệnh đơn hoặc Câu lệnh phức (kẹp giữa { và })

Câu lệnh while

```
int i = 0;
while (i < 10)
{
        printf("%d\n", i);
        i++;
}

for (int i = 0; i < 10; i++)
        printf("%d\n", i);

int i = 0;
for (; i < 10; )
{
        printf("%d\n", i);
        i++;
}
```

Câu lệnh while - Một số lưu ý

- Câu lệnh while có thể không thực hiện lần nào do điều kiện lặp ngay từ lần đầu đã không thỏa.

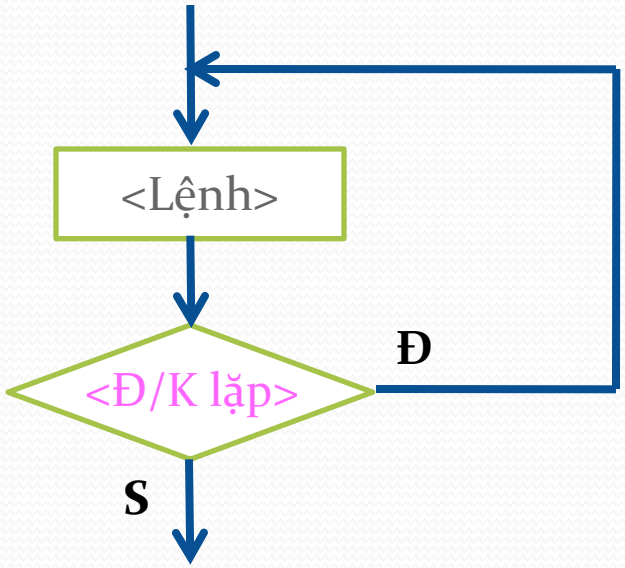
```
main()
{
        int n = 1;
        while (n > 10)
        {
                printf("%d\n", n);
                n--;
        }
        …
}
```

Câu lệnh while - Một số lưu ý

- Không được thêm ; ngay sau lệnh lệnh while.

```
int n = 0;
while (n < 10);
{
        printf("%d\n", n);
        n++;
}

while (n < 10)
{
};
{
        printf("%d\n", n);
        n++;
}
```

Câu lệnh do… while



do

<Lệnh>;

while (<Đ/K lặp>);

Câu lệnh đơn hoặc Câu lệnh phức (kẹp giữa { và })

Biểu thức C bất kỳ, thường là biểu thức quan hệ cho kết quả 0 (sai) và != 0 (đúng)

Câu lệnh do… while

```
int i = 0;
do
{
        printf("%d\n", i);
        i++;
}
while (i < 10);
```

```
int i = 0;
printf("%d\n", i);
i++;
for (; i < 10; )
{
        printf("%d\n", i);
        i++;
}
```

Câu lệnh do… while - Một số lưu ý

- Câu lệnh do… while là một câu lệnh đơn và có thể lồng nhau.

```
int a = 1, b;
do
{
        b = 1;
        do
        {
                printf("%d\n", a + b);
                b = b + 2;
        }
        while (b < 20);
        a++;
}
while (a < 20);
```

Câu lệnh do while- Một số lưu ý

- Câu lệnh do… while sẽ được thực hiện ít nhất 1 lần do điều kiện lặp được kiểm tra ở cuối.

```
main()
{
        int n;
        do
        {
                printf("Nhap n: ");
                scanf("%d", &n);
        }
        while (n < 1 || n > 100);
}
```

FOR, WHILE & DO… WHILE

- Đều có khả năng lặp lại nhiều hành động.

```
int n = 10;
for (int i = 1; i <= n; i++)
        printf("%d\n", i);

int i = 1;
while (i <= n)
{
        printf("%d\n", i); i++;
}

int i = 1;
do {
        printf("%d\n", i); i++;
} while (i > n);
```

FOR, WHILE & DO… WHILE

- Số lần lặp xác định ngay trong câu lệnh for

```
int n = 10;
for (int i = 1; i <= n; i++)
        …;
```

```
int i = 1;
while (i <= n)
{
        …;
}
```

```
int i = 1;
do {
        …;
} while (i > n);
```

WHILE & DO… WHILE

- while có thể không thực hiện lần nào.
- do… while sẽ được thực hiện ít nhất 1 lần.

```
int n = 100;
while (n < 10)
{
        …;
}
…
do
{
        printf("Nhap n: ");
        scanf("%d", &n);
}
while (n > 10);
```

CÁC CÂU LỆNH ĐẶC BIỆT

- Lệnh break làm kết thúc câu lệnh.
- Lệnh continue bỏ qua lần lặp hiện tại.

Ví dụ:

```
while (x != y)
        {     …
            if (x==a) continue;
            b+=6;
            …
            if (y==b) break;
            …
        }
```

Bài tập thực hành

1. Nhập một số nguyên dương n (n > 0).

   Hãy cho biết:

   a. Có phải là số đối xứng? Ví dụ: 121, 12321, …
   b. Có phải là số chính phương? Ví dụ: 4, 9, 16, …
   c. Có phải là số nguyên tố? Ví dụ: 2, 3, 5, 7, …
   d. Chữ số lớn nhất và nhỏ nhất?
   e. Các chữ số có tăng dần hay giảm dần không?

Bài tập thực hành

2. Nhập một số nguyên dương n. Tính:

a. $S = 1 + 2 + \ldots + n$

b. $S = 1^2 + 2^2 + \ldots + n^2$

c. $S = 1 + 1/2 + \ldots + 1/n$

d. $S = 1*2*\ldots*n = n!$

e. $S = 1! + 2! + \ldots + n!$

3. Nhập 3 số nguyên a, b và n với a, b < n. Tính tổng các số nguyên dương nhỏ hơn n chia hết cho a nhưng không chia hết cho b.

4. Tính tổng các số nguyên tố nhỏ hơn n (0 < n < 50)

Bài tập thực hành

5. Nhập một số nguyên dương n. Xuất ra số ngược lại. Ví dụ: Nhập 1706 ➔ Xuất 6071.

6. Tìm và in lên màn hình tất cả các số nguyên trong phạm vi từ 10 đến 99 sao cho tích của 2 chữ số bằng 2 lần tổng của 2 chữ số đó.

7. Tìm ước số chung lớn nhất của 2 số nguyên dương a và b nhập từ bàn phím.

8. Nhập n. In n số đầu tiên trong dãy Fibonacy.

   a. $a_0 = a_1 = 1$

   b. $a_n = a_{n-1} + a_{n-2}$

Bài tập 1a: kiểm tra đối xứng

```
        …//Nhap n
        sogoc = n; sodao = 0;
        while (sogoc > 0)
        {
                donvi = sogoc % 10;
                sodao = sodao*10 + donvi;
                sogoc = sogoc / 10;
        }
        if (sodao == n) printf("DX");
        else printf("Khong doi xung");
}
```

Bài tập 1b

```
//số chính phương
n_can_nguyen = int(sqrt(n));
if (n_can_nguyen*n_can_nguyen == n)
        printf("%d la so CP.", n);
else
        printf("%d khong la so CP.", n);
```

Bài tập 1c

```
//kiểm tra nguyên tố
souoc = 0;
for (i = 1; i <= n; i++)
        if (n % i == 0)
                souoc++;

if (souoc == 2)
        printf("%d la so nguyen to",n);
else
        printf("%d ko la so nguyen to", n);
```