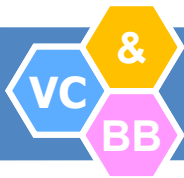


NHẬP MÔN LẬP TRÌNH

CÂU LỆNH LẶP





Nội dung

1

Câu lệnh for

2

Câu lệnh while

3

Câu lệnh do... while

4

Một số kinh nghiệm lập trình

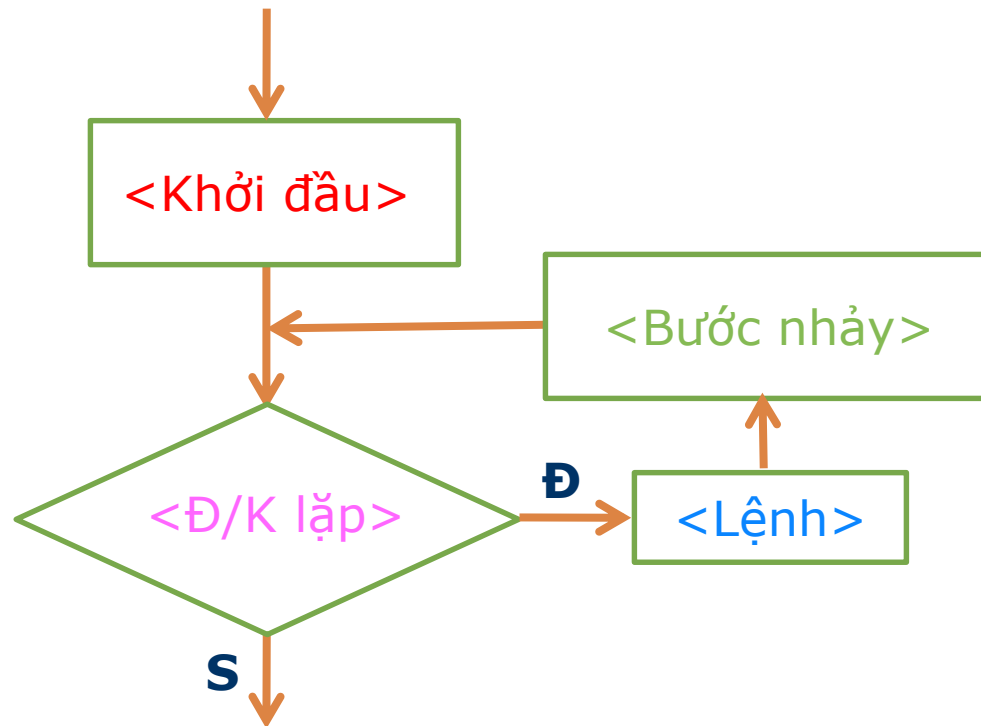
❖ Ví dụ

- Viết chương trình xuất các số từ 1 đến 10
=> Sử dụng 10 câu lệnh printf
- Viết chương trình xuất các số từ 1 đến 1000
=> Sử dụng 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

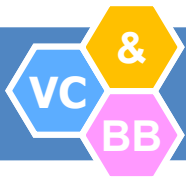
Câu lệnh for



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

<Lệnh>;

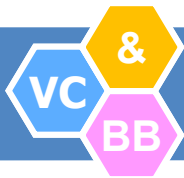
<Khởi đầu>, <Đ/K lặp>, <Bước nhảy> :
là biểu thức C bất kỳ có chức năng riêng
<Lệnh> : đơn hoặc khối lệnh.



Câu lệnh for

```
void main()  
{
```

```
}
```



Câu lệnh for - Một số lưu ý

❖ Câu lệnh **FOR** là một câu lệnh đơn và có thể lồng nhau.

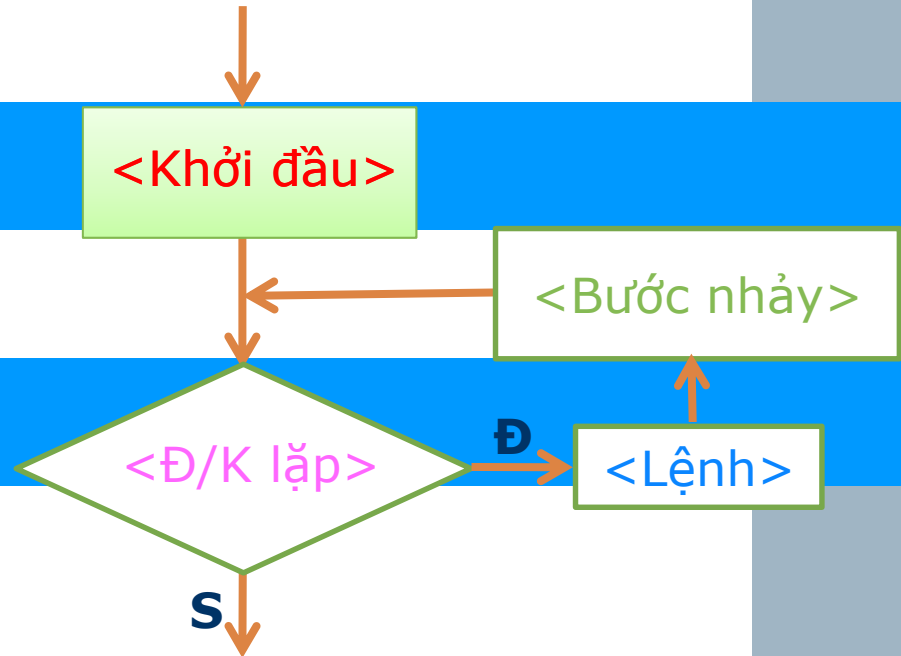
```
if (n < 10 && m < 20)
```

Câu lệnh for - Một số lưu ý

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

```
int i;
```

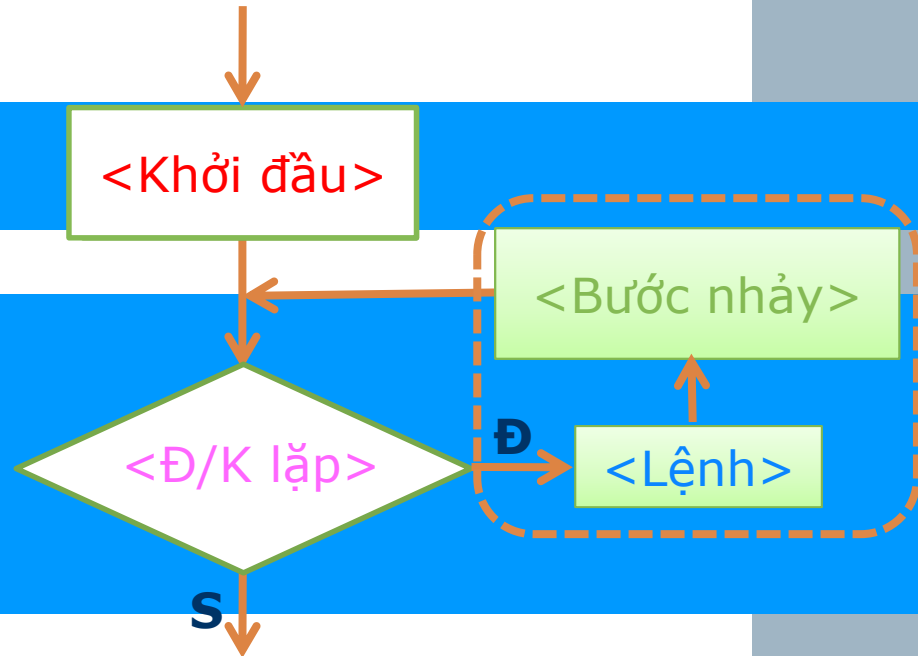
```
int ;
```

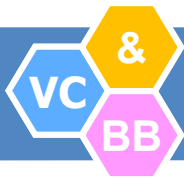


Câu lệnh for - Một số lưu ý

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

```
int i;
```

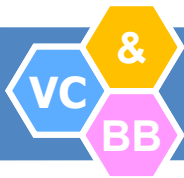




Câu lệnh for - Một số lưu ý

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

```
int i;
```



Câu lệnh for - Một số lưu ý

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

```
{
```

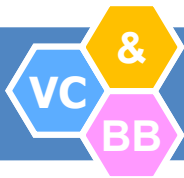
```
    printf("%d\n", i);
```

```
}
```

```
{
```

```
    printf("%d\n", i);
```

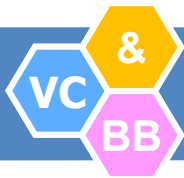
```
}
```



Câu lệnh for - Một số lưu ý

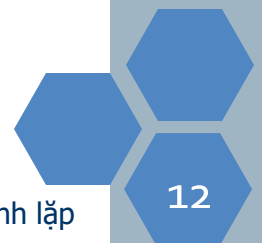
- ❖ Không được thêm **;** ngay sau lệnh lệnh for.
=> Tương đương câu lệnh rỗng.

```
{  
    printf("%d", i);  
    printf("\n");  
}
```

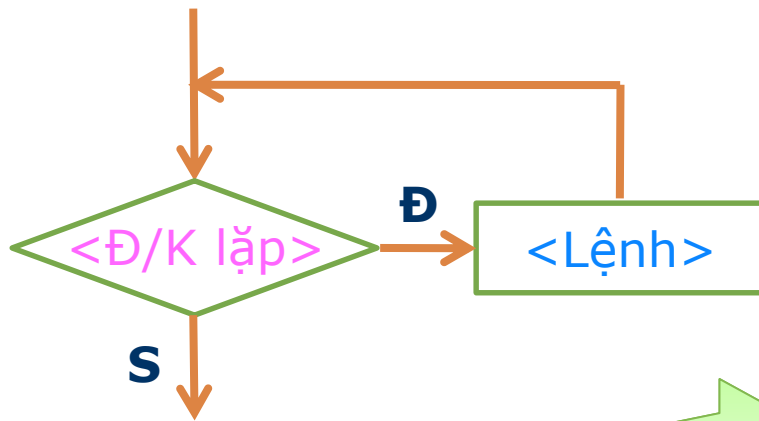


Câu lệnh for - Một số lưu ý

- ❖ 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 ,



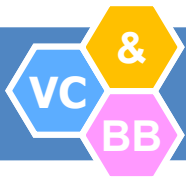
Câu lệnh while



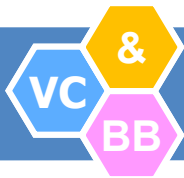
while (<Đ/K lặp>)
 <Lệnh>;

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 })



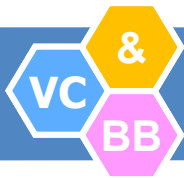
Câu lệnh while



Câu lệnh while - Một số lưu ý

❖ Câu lệnh **while** là một **câu lệnh đơn** và **có thể lồng nhau**.

```
if (n < 10 && m < 20)
```

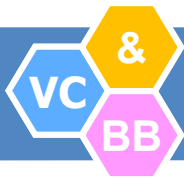


Câu lệnh while - Một số lưu ý

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

```
void main()  
{  
  
    {  
        printf("%d\n", n);  
        n--;  
    }  
  
}
```

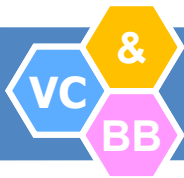




Câu lệnh for - Một số lưu ý

❖ Không được thêm **;** ngay sau lệnh `while`.

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

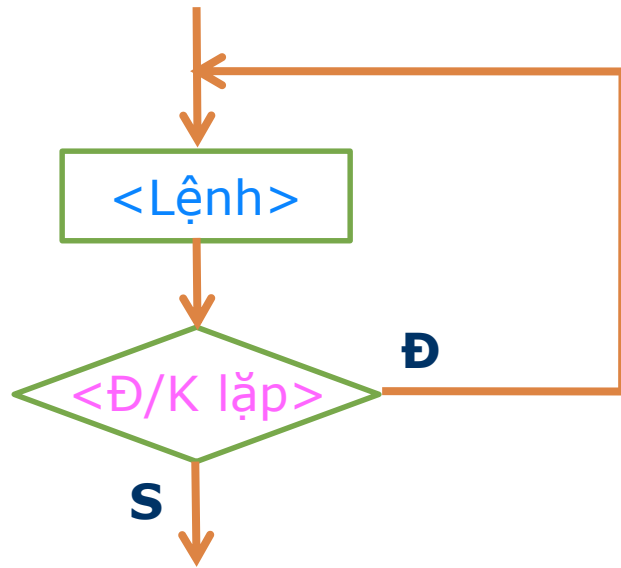


Câu lệnh while - Một số lưu ý

❖ Câu lệnh **while** có thể bị lặp vô tận (**loop**)

```
void main()  
{  
  
    {  
        printf("%d", n);  
    }  
  
}
```

Câu lệnh do... while



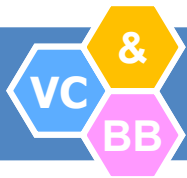
do

<Lệnh>;

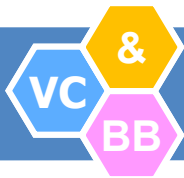
while (<Đ/K lặp>);

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



Câu lệnh do... while - Một số lưu ý

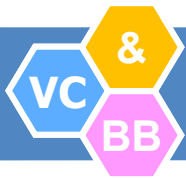
❖ Câu lệnh **do... while** là một câu lệnh đơn và có thể lồng nhau.

```
int a = 1, b;
```



```
void main()
{
    int n;
```

NMLT - Câu lệnh lặp



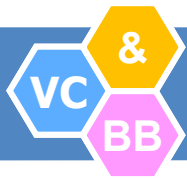
Câu lệnh do... while - Một số lưu ý

❖ Câu lệnh **do... while** có thể bị lặp vô tận (**loop**)

...

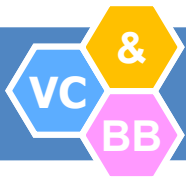
...

...



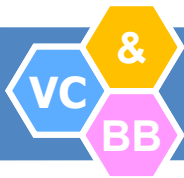
for, while, do... while

❖ Điều có khả năng lặp lại nhiều hành động.



for, while, do... while

❖ Số lần lặp xác định ngay trong câu lệnh **for**



while & do... while

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






```
{  
    ...;  
}
```

```
do  
{
```

```
}
```

3. 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 kh



4. Nhập một số nguyên dương n . Tính:


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

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

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

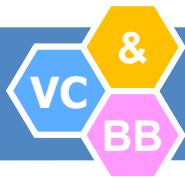
 d. $S = 1 * 2 * \dots * n = n!$

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

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

 6. Tính tổng các số nguyên tố nhỏ hơn n ($0 < r_1 < r_2 < \dots < r_k < n$)

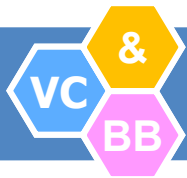




Bài tập thực hành

7. Nhập một số nguyên dương n . Xuất ra số ngược lại. Ví dụ: Nhập 1706 \rightarrow Xuất 6071.
8. 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.
9. Tìm ước số chung lớn nhất của 2 số nguyên dương a và b nhập từ bàn phím.
10. Nhập n . In n số đầu tiên trong dãy Fibonacci.
- a. $a_0 = a_1 = 1$
 - b. $a_n = a_{n-1} + a_{n-2}$

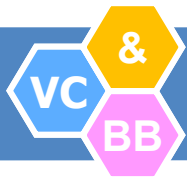




Bài tập 3a

```
void main()  
{
```

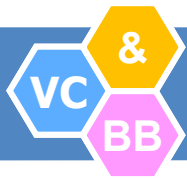
```
}
```



Bài tập 3b

```
void main()  
{
```

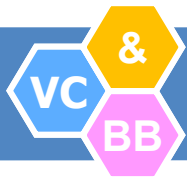
```
}
```



Bài tập 3c

```
void main()  
{
```

```
}
```

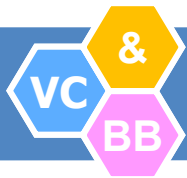



Bài tập 3d

```
void main()  
{
```

```
...
```

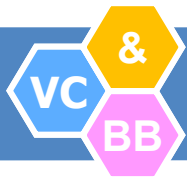
```
}
```



Bài tập 3e

```
void main()  
{
```

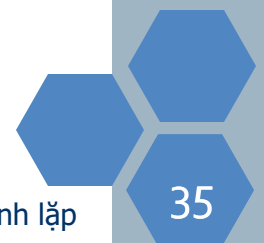
```
}
```

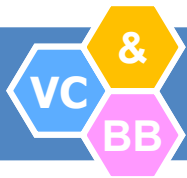


Bài tập 4a

```
void main()  
{
```

```
}
```

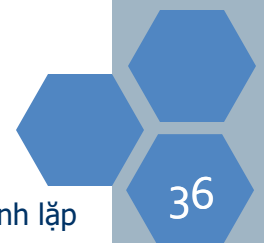


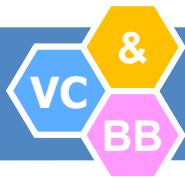


Bài tập 4b

```
void main()  
{
```

```
}
```

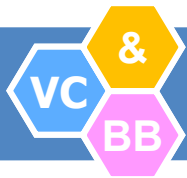




Bài tập 4c

```
void main()  
{
```

```
}
```

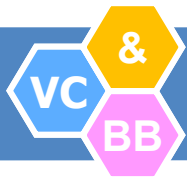


Bài tập 4d

```
void main()  
{
```

```
}
```





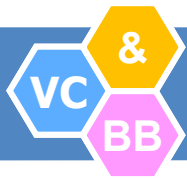
Bài tập 4e

```
void main()  
{
```

```
{
```

```
}
```

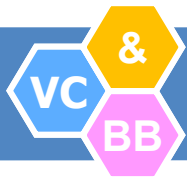
```
}
```



Bài tập 5

```
void main()  
{
```

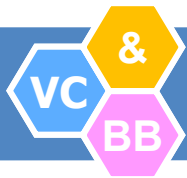
```
}
```

Bài tập 6

```
void main()  
{
```

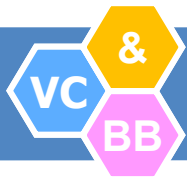
```
}
```



Bài tập 7

```
void main()  
{
```

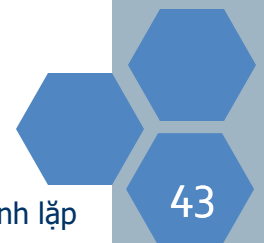
```
}
```



Bài tập 8

```
void main()  
{
```

```
}
```



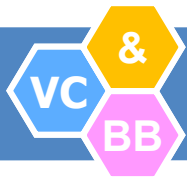
❖ Ví dụ: $a = 12, b = 8$

❖ Cách 1:

- Cho 1 biến i chạy từ 8 trở về 1, nếu cả a và b đều chia hết cho i thì dừng và i chính là uscln.
- 8, 7, 6, 5, 4 \Rightarrow USCLN của 12 và 8 là 4.

❖ Cách 2:

- USCLN của a & b (a khác b), ký hiệu (a, b) là:
 - $(a - b, b)$ nếu $a > b$
 - $(a, b - a)$ nếu $b > a$
- $(12, 8) = (4, 8) = (4, 4) = 4$

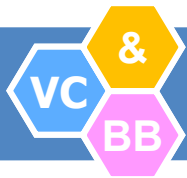


Bài tập 9

```
void main()
```

```
{
```

```
}
```



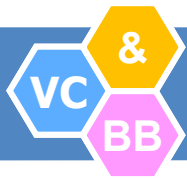
Bài tập 9

```
void main()  
{
```

```
}
```

Bài tập 10

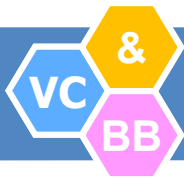
- ❖ Dãy Fibonacci: $a_0 a_1 a_2 \dots a_{n-2} a_{n-1} a_n$
 - Với $a_0 = a_1 = 1$, $a_n = a_{n-1} + a_{n-2}$
- ❖ Ví dụ: 1 1 2 3 5 8 13 21 ...
- ❖ Xuất n phần tử đầu tiên của dãy Fibonacci
 - $n = 1 \Rightarrow 1$, $n = 2 \Rightarrow 1\ 1$
 - $n > 2$
 - Lưu lại 2 phần tử trước nó là a và b
 - Mỗi lần tính xong cập nhật lại a và b.
- ❖ Nên thêm 2 phần tử ảo đầu tiên là a_{-2} , a_{-1}
 - 1 0 1 1 2 3 5 8 13 21 ...



Bài tập 10

```
void main()  
{
```

```
}
```

Bài tập

- ❖ $S = 1/2 + 1/4 + \dots + 1/2n$
- ❖ $S = 1 + 1/3 + 1/5 + \dots + 1/(2n+1)$
- ❖ $S = 1/(1x^2) + 1/(2x^3) + \dots + 1/(nx^{n+1})$
- ❖ $S = 1/2 + 2/3 + \dots + n/(n+1)$
- ❖ $S = 1 + 1/(1 + 2) + \dots + 1/(1 + 2 + \dots + n)$
- ❖ Liệt kê tất cả ước số của số nguyên dương n
- ❖ Tính tổng các ước số của số nguyên dương n
- ❖ Đếm số lượng ước số của số nguyên dương n
- ❖ Tính tổng các ước số chẵn của số nguyên dương n

