

Đoạn mã trên sử dụng toán tử bitwise OR (`|`) và toán tử dịch trái (`<<`) để thao tác với giá trị kiểu `char` và in ra các ký tự theo cách đặc biệt. Hãy cùng phân tích chi tiết mã này.

```
c Sao chép mã  
  
#include<stdio.h>  
  
int main()  
{  
    char c = 48;  
    int i, mask = 01;  
    for(i = 1; i <= 5; i++)  
    {  
        printf("%c", c | mask);  
        mask = mask << 1;  
    }  
    return 0;  
}
```

Đoạn mã trên sử dụng toán tử bitwise OR (`|`) và toán tử dịch trái (`<<`) để thao tác với giá trị kiểu `char` và in ra các ký tự theo cách đặc biệt. Hãy cùng phân tích chi tiết mã này.

```
c  
Sao chép mã  
#include<stdio.h>  
  
int main()  
{  
    char c = 48;  
    int i, mask = 01;  
    for(i = 1; i <= 5; i++)  
    {  
        printf("%c", c | mask);  
        mask = mask << 1;  
    }  
    return 0;  
}
```

Phân tích chi tiết


1. Khai báo và khởi tạo giá trị:

- o `char c = 48;`
 - Biến `c` có kiểu `char` và được khởi tạo với giá trị 48. Trong bảng mã ASCII, giá trị 48 tương ứng với ký tự `'0'`. Vì vậy, `c` đại diện cho ký tự `'0'`.
- o `int mask = 01;`
 - `mask` là một biến kiểu `int` được khởi tạo với giá trị 01. Ở đây, giá trị 01 thực ra là số nhị phân 1 (vì tiền tố 0 chỉ ra rằng đây là giá trị hệ 8, nhưng giá trị thực tế là 1).

2. Vòng lặp `for` và cách hoạt động của mã:

- **Vòng lặp** này chạy từ $i = 1$ đến $i = 5$, tức là lặp 5 lần.
- Trong mỗi vòng lặp, thực hiện các thao tác sau:

c

 Sao chép mã

```
printf("%c", c | mask);
```

1.

- $c | \text{mask}$ là phép toán **bitwise OR** giữa giá trị của c và mask . Phép toán này sẽ thực hiện phép OR theo từng bit của c và mask . Do $c = 48$ (tương đương 00110000 trong hệ nhị phân), và mask thay đổi mỗi lần, ta sẽ xem xét sự thay đổi này ở từng vòng lặp:
 - Lần 1: $\text{mask} = 1$ (nhị phân: 00000001)
 - $c | \text{mask} = 48 | 1 = 49$ (tương đương ký tự '1')
 - In ra '1'
 - Lần 2: $\text{mask} = 2$ (nhị phân: 00000010)
 - $c | \text{mask} = 48 | 2 = 50$ (tương đương ký tự '2')
 - In ra '2'
 - Lần 3: $\text{mask} = 4$ (nhị phân: 00000100)
 - $c | \text{mask} = 48 | 4 = 52$ (tương đương ký tự '4')
 - In ra '4'
 - Lần 4: $\text{mask} = 8$ (nhị phân: 00001000)
 - $c | \text{mask} = 48 | 8 = 56$ (tương đương ký tự '8')
 - In ra '8'
 - Lần 5: $\text{mask} = 16$ (nhị phân: 00010000)
 - $c | \text{mask} = 48 | 16 = 64$ (tương đương ký tự '@')
 - In ra '@'
- Sau mỗi vòng lặp, giá trị mask sẽ được dịch sang trái một bit ($\text{mask} = \text{mask} << 1$), làm cho mask tăng gấp đôi (hoặc dịch tất cả các bit sang trái).

2. Tổng kết kết quả:


- Ở mỗi vòng lặp, ký tự được in ra sẽ là kết quả của phép toán $c | \text{mask}$. Như đã phân tích, kết quả của từng vòng lặp là:
 - Vòng 1: '1' (mã ASCII 49)
 - Vòng 2: '2' (mã ASCII 50)
 - Vòng 3: '4' (mã ASCII 52)
 - Vòng 4: '8' (mã ASCII 56)
 - Vòng 5: '@' (mã ASCII 64)

Kết quả là: **1248@**

Kết quả đầu ra

Khi chương trình chạy, kết quả in ra sẽ là:

```
yaml
```

 Sao chép mã

```
1248@
```

Tóm tắt

- Trong đoạn mã này, bạn sử dụng phép toán bitwise OR (`|`) để kết hợp các bit của `c` (ban đầu là `48`, ký tự `'0'`) với `mask` (bắt đầu là `1`, và sau đó dịch sang trái trong mỗi vòng lặp).
- Mỗi lần, kết quả của phép OR được in ra dưới dạng ký tự, và do giá trị của `mask` thay đổi theo mỗi vòng lặp, các ký tự được in ra là các ký tự `'1'`, `'2'`, `'4'`, `'8'`, và `'@'`.