Bài 3: Bitmask

Phan Hoàng Trung

Khái niệm

Bitmask là một kỹ thuật sử dụng các bit để lưu trữ và thao tác với các cờ (flags) hoặc trạng thái. Có thể sử dụng bitmask để đặt, xóa và kiểm tra trạng thái của các bit cụ thể trong một từ (word).

Bitmask thường được sử dụng để tối ưu hóa bộ nhớ, thực hiện các phép toán logic trên một cụm bit, và quản lý các trạng thái, quyền truy cập, hoặc các thuộc tính khác của một đối tượng.

NOT bitwise

Dùng để thực hiện phép NOT bitwise trên từng bit của một số. Kết quả là bit đảo ngược của số đó.

```
int result = ~num ;
```

AND bitwise

Dùng để thực hiện phép AND bitwise giữa từng cặp bit của hai số. Kết quả là 1 nếu cả hai bit tương ứng đều là 1, ngược lại là 0.

```
int result = num1 & num2;
```

OR bitwise

Dùng để thực hiện phép OR bitwise giữa từng cặp bit của hai số. Kết quả là 1 nếu có hơn một bit tương ứng là 1.

```
int result = num1 | num2;
```

XOR bitwise

Dùng để thực hiện phép XOR bitwise giữa từng cặp bit của hai số. Kết quả là 1 nếu chỉ có một bit tương ứng là 1.

```
int result = num1 ^ num2;
```

Shift left và Shift right bitwise

Dùng để di chuyển bit sang trái hoặc sang phải.

Trong trường hợp <<, các bit ở bên phải sẽ được dịch sang trái, và các bit trái cùng sẽ được đặt giá trị 0.

Trong trường hợp >>, các bit ở bên trái sẽ được dịch sang phải, và các bit phải cùng sẽ được đặt giá trị 0 hoặc 1 tùy thuộc vào giá trị của bit cao nhất (bit dấu).

```
int resultLeftShift = num << shiftAmount;
int resultRightShift = num >> shiftAmount;
```

Ví dụ

uint8_t options =

Ví dụ

uint8_t options =

Ví dụ

uint8_t options =

Ví dụ

```c
#include <stdio.h>
#include <stdint.h>


#define GENDER        1 << 0  // Bit 0: Giới tính (0 = Nữ, 1 = Nam)
#define TSHIRT        1 << 1  // Bit 1: Áo thun (0 = Không, 1 = Có)
#define HAT           1 << 2  // Bit 2: Nón (0 = Không, 1 = Có)
#define SHOES         1 << 3  // Bit 3: Giày (0 = Không, 1 = Có)
// Tự thêm tính năng khác
#define FEATURE1      1 << 4  // Bit 4: Tính năng 1
#define FEATURE2      1 << 5  // Bit 5: Tính năng 2
#define FEATURE3      1 << 6  // Bit 6: Tính năng 3
#define FEATURE4      1 << 7  // Bit 7: Tính năng 4

void enableFeature(uint8_t *features, uint8_t feature) {
    *features |= feature;
}

void disableFeature(uint8_t *features, uint8_t feature) {
```

Ví dụ

```c
#include <stdio.h>

#define LED1 1 << 0 // 0001
#define LED2 1 << 1 // 0010
#define LED3 1 << 2 // 0100
#define LED4 1 << 3 // 1000


void enableLED(unsigned int *GPIO_PORT, unsigned int LED) {
    *GPIO_PORT |= LED;
}

void disableLED(unsigned int *GPIO_PORT, unsigned int LED) {
    *GPIO_PORT &= ~LED;
}


int main() {
    unsigned int GPIO_PORT = 0; // Giả sử là biến điều khiển cổng GPIO
```

Ví dụ

```c
#include <stdio.h>
#include <stdint.h>

#define ENABLE 1
#define DISABLE 0

typedef struct {
    uint8_t LED1 : 1;
    uint8_t LED2 : 1;
    uint8_t LED3 : 1;
    uint8_t LED4 : 1;
    uint8_t LED5 : 1;
    uint8_t LED6 : 1;
    uint8_t LED7 : 1;
    uint8_t LED8 : 1;
} LEDStatus;

void displayAllStatusLed(LEDStatus status) {
    uint8_t* statusPtr = (uint8_t*)&status;
```

Ví dụ

```
#include <stdio.h>
#include <stdint.h>
#define COLOR_RED 0
#define COLOR_BLUE 1
#define COLOR_BLACK 2
#define COLOR_WHITE 3
#define POWER_100HP 0
#define POWER_150HP 1
#define POWER_200HP 2
#define ENGINE_1_5L 0
#define ENGINE_2_0L 1

typedef uint8_t CarColor;
typedef uint8_t CarPower;
typedef uint8_t CarEngine;



#define SUNROOF_MASK 1 << 0     // 0001
```