Bài 6: Bitmask

# Khái niệm cơ bản

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.

# Các toán tử bitwise

## 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.

Ví dụ:

| **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.

Ví dụ:

| **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 một trong hai bit tương ứng là 1.

Ví dụ:

| **int** result = num1 ^ num2; |
| --- |

## NOT bitwise

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

Ví dụ:

| **int** result = ~num; |
| --- |

## Shift left và Shift right

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).

Ví dụ:

| **int** resultLeftShift = num << shiftAmount; **int** resultRightShift = num >> shiftAmount; |
| --- |

Một số ví dụ và ứng dụng:

| #**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 5 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) {  \*features &= ~feature; }   **int** **isFeatureEnabled**(**uint8\_t** features, **uint8\_t** feature) {  **return** (features & feature) != 0; }  **void** **listSelectedFeatures**(**uint8\_t** features) {  printf("Selected Features:\n");   **if** (features & GENDER) {  printf("- Gender\n");  }  **if** (features & TSHIRT) {  printf("- T-Shirt\n");  }  **if** (features & HAT) {  printf("- Hat\n");  }  **if** (features & SHOES) {  printf("- Shoes\n");  }  // Thêm các điều kiện kiểm tra cho các tính năng khác }  **void** **removeFeatures**(**uint8\_t** \*features, **uint8\_t** unwantedFeatures) {  \*features &= ~unwantedFeatures; }    **int** **main**() {  **uint8\_t** options = 0;   // Thêm tính năng   enableFeature(&options, GENDER | TSHIRT | HAT);   removeFeatures(&options, TSHIRT);   // Liệt kê các tính năng đã chọn  listSelectedFeatures(options);    **return** 0; } |
| --- |

Ví dụ 2:

| #**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   // Bật LED1 và LED3  enableLED(&GPIO\_PORT, LED1 | LED3);  **if** (GPIO\_PORT & LED1 )  {  printf("LED1 is on\n");  }   **if** (GPIO\_PORT & LED2)  {  printf("LED2 is on\n");  }   **if** (GPIO\_PORT & LED3)  {  printf("LED3 is on\n");  }    // Tắt LED1 và bật LED2  disableLED(&GPIO\_PORT, LED1);  enableLED(&GPIO\_PORT, LED2);   **if** (GPIO\_PORT & LED1 )  {  printf("LED1 is on\n");  }   **if** (GPIO\_PORT & LED2)  {  printf("LED2 is on\n");  }   **if** (GPIO\_PORT & LED3)  {  printf("LED3 is on\n");  }   // Cập nhật trạng thái của GPIO\_PORT tương ứng với hardware   **return** 0; } |
| --- |

Ví dụ 3:

| #**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;    **int** **main**() {  LEDStatus ledStatus = {.LED7 = ENABLE};   // Bật LED 1 và 3  ledStatus.LED1 = ENABLE;  ledStatus.LED3 = ENABLE;   // Kiểm tra trạng thái của LED 1  **if** (ledStatus.LED1) {  printf("LED1 is on\n");  }    **return** 0; } |
| --- |

Ví dụ 4:

| #**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 #**define** PREMIUM\_AUDIO\_MASK 1 << 1 // 0010 #**define** SPORTS\_PACKAGE\_MASK 1 << 2 // 0100 // Thêm các bit masks khác tùy thuộc vào tùy chọn   **typedef** **struct** {  **uint8\_t** additionalOptions : 3; // 3 bits cho các tùy chọn bổ sung  CarColor color : 2;  CarPower power : 2;  CarEngine engine : 1;   } CarOptions;    **void** **configureCar**(CarOptions \*car, CarColor color, CarPower power, CarEngine engine, **uint8\_t** options) {  car->color = color;  car->power = power;  car->engine = engine;  car->additionalOptions = options; }  **void** **setOption**(CarOptions \*car, **uint8\_t** optionMask) {  car->additionalOptions |= optionMask; }  **void** **unsetOption**(CarOptions \*car, **uint8\_t** optionMask) {  car->additionalOptions &= ~optionMask; }   **void** **displayCarOptions**(**const** CarOptions car) {  **const** **char** \*colors[] = {"Red", "Blue", "Black", "White"};  **const** **char** \*powers[] = {"100HP", "150HP", "200HP"};  **const** **char** \*engines[] = {"1.5L", "2.0L"};   printf("Car Configuration: \n");  printf("Color: %s\n", colors[car.color]);  printf("Power: %s\n", powers[car.power]);  printf("Engine: %s\n", engines[car.engine]);  printf("Sunroof: %s\n", (car.additionalOptions & SUNROOF\_MASK) ? "Yes" : "No");  printf("Premium Audio: %s\n", (car.additionalOptions & PREMIUM\_AUDIO\_MASK) ? "Yes" : "No");  printf("Sports Package: %s\n", (car.additionalOptions & SPORTS\_PACKAGE\_MASK) ? "Yes" : "No"); }  **int** **main**() {  CarOptions myCar = {COLOR\_BLACK, POWER\_150HP, ENGINE\_2\_0L};   setOption(&myCar, SUNROOF\_MASK);  setOption(&myCar, PREMIUM\_AUDIO\_MASK);    displayCarOptions(myCar);   unsetOption(&myCar, PREMIUM\_AUDIO\_MASK);   displayCarOptions(myCar);   printf("size of my car: %d\n", **sizeof**(CarOptions));   **return** 0; } |
| --- |