

# Битовые операции. Битовые поля.

# Побитовые операции

Язык Си поддерживает все битовые операции:

|                 |                 |
|-----------------|-----------------|
| <b>&amp;</b>    | и               |
| <b> </b>        | или             |
| <b>^</b>        | исключающее или |
| <b>~</b>        | дополнение      |
| <b>&gt;&gt;</b> | сдвиг вправо    |
| <b>&lt;&lt;</b> | сдвиг влево     |

- Битовые операции применимы только к целочисленным переменным.
- Битовые операции обычно выполняют над беззнаковыми целыми, чтобы не было путаницы со знаком.

# Побитовые операции

| Операция          | Название                          | Нотация             | Класс      | Приоритет | Ассоциат.     |
|-------------------|-----------------------------------|---------------------|------------|-----------|---------------|
| <b>~ (унар. )</b> | Побитовое «НЕ»                    | <b>~X</b>           | Префиксные | 15        | Справа налево |
| <b>&lt;&lt;</b>   | Сдвиг влево                       | <b>X &lt;&lt; Y</b> | Инфиксные  | 11        | Слева направо |
| <b>&gt;&gt;</b>   | Сдвиг вправо                      | <b>X &gt;&gt; Y</b> |            |           |               |
| <b>&amp;</b>      | Побитовое «И»                     | <b>X &amp; Y</b>    | Инфиксные  | 8         | Слева направо |
| <b>^</b>          | Побитовое<br>исключающее<br>«ИЛИ» | <b>X ^ Y</b>        | Инфиксные  | 7         | Слева направо |
| <b> </b>          | Побитовое<br>«ИЛИ»                | <b>X   Y</b>        | Инфиксные  | 6         | Слева направо |

# Побитовые операции

| Операция         | Название  | Нотация              | Класс     | Приоритет | Ассоциат.     |
|------------------|---|----------------------|-----------|-----------|---------------|
| <b>&lt;&lt;=</b> | Присваивание<br>со сдвигом<br>влево                 | <b>X &lt;&lt;= Y</b> | Инфиксные | 2         | Справа налево |
| <b>&gt;&gt;=</b> | Присваивание<br>со сдвигом<br>вправо                | <b>X &gt;&gt;= Y</b> |           |           |               |
| <b>&amp;=</b>    | Присваивание<br>с побитовым<br>«И»                  | <b>X &amp;= Y</b>    |           |           |               |
| <b>^=</b>        | Присваивание<br>с побитовым<br>исключающим<br>«ИЛИ» | <b>X ^= Y</b>        |           |           |               |
| <b> =</b>        | Присваивание<br>с побитовым<br>«ИЛИ»                | <b>X  = Y</b>        |           |           |               |

# Проверка битов (&)

```
unsigned char a      = 0x46;  // 01000110b
unsigned char b      = 0x44;  // 01000100b
unsigned char mask = 0x06;  // 00000110b
```

```
printf("a & mask %x, res %d\n", a & mask, (a & mask) == mask);
printf("b & mask %x, res %d\n", b & mask, (b & mask) == mask);
```

# Обнуление битов (&)

```
unsigned char a          = 0x46;  // 01000110b
unsigned char mask_1 = 0xbf;  // 10111111b
unsigned char mask_2 = 0xf9;  // 11111001b
```

```
printf("a & mask_1 %x\n", a & mask_1);
printf("a & mask_2 %x\n", a & mask_2);
```

# Установка битов (|)

```
unsigned char a      = 0x40;  // 01000000b
unsigned char mask_1 = 0x06;  // 00000110b
unsigned char mask_2 = 0x44;  // 01000100b

printf("a | mask_1 %x\n", a | mask_1);
printf("a | mask_2 %x\n", a | mask_2);
```

# Смена значений битов (^)

```
unsigned char a      = 0x46;  // 01000110b
unsigned char mask_1 = 0x44;  // 01000100b
unsigned char mask_2 = 0xFF;  // 11111111b

printf("a ^ mask_1 %x\n", a ^ mask_1);
printf("a ^ mask_2 %x\n", a ^ mask_2);
```

# Сдвиг вправо (>>)

```
unsigned char a = 0xFF; // 11111111b
```

```
printf("a >> 1 = %2x\n", a >> 1);
```

```
printf("a >> 4 = %2x\n", a >> 4);
```

# Сдвиг влево (<<)

```
unsigned char a = 0x01; // 00000001b
```

```
printf("a << 1 = %2x\n", a << 1);
```

```
printf("a << 4 = %2x\n", a << 4);
```

# Отличие между & и &&, | и ||

```
#include <stdio.h>

int main(void)
{
    unsigned char a = 0x01;
    unsigned char b = 0x02;

    if (a && b)
        printf("a && b true\n");
    else
        printf("a && b false\n");

    if ((b && 1) == 1)
        printf("odd\n");
    else
        printf("even\n");

    a = 0;
    if (a && b / a)
        printf("true\n");
    else
        printf("false\n");

    return 0;
}
```

```
#include <stdio.h>

int main(void)
{
    unsigned char a = 0x01;
    unsigned char b = 0x02;

    if (a & b)
        printf("a & b true\n");
    else
        printf("a & b false\n");

    if ((b & 1) == 1)
        printf("odd\n");
    else
        printf("even\n");

    a = 0;
    if (a & b / a) // !!!
        printf("true\n");
    else
        printf("false\n");

    return 0;
}
```

8



```
#define HIDE      0
#define SHOW     1
#define BORDER   1
#define CAPTION  2
#define RED       1
#define GREEN    2
#define BLUE     4
```

```
struct window
{
    ...
    unsigned char flags;
};
```

Пусть в поле flags

бит 0 – видимо/скрыто (SHOW/HIDE)

биты 1 – 2 – стиль окна (BORDER, CAPTION)

биты 3 – 5 – цвет окна (RED, GREEN, BLUE)

```
struct window w;
unsigned char part;

w.flags = SHOW | (BORDER << 1) | ((RED | BLUE) << 3);

part = (w.flags >> 1) & 0x3;
if (part & BORDER)
    printf("BORDER is present\n");
else
    printf("BORDER is not present\n");

if ((w.flags >> 3) & GREEN)
    printf("GREEN is present\n");
else
    printf("GREEN is not present\n");

if ((w.flags >> 3) & BLUE)
    printf("BLUE is present\n");
else
    printf("BLUE is not present\n");
```

# Битовые поля

*Битовое поле* - особый тип структуры, определяющей, какую длину имеет каждый член в битах.

Стандартный вид объявления битовых полей следующий:

```
struct имя_структуры
{
    тип имя1: длина;
    тип имя2: длина;
    ...
    тип имяN: длина;
};
```

Битовые поля должны объявляться как целые, unsigned или signed.

```
struct wnd_flags
{
    unsigned char show    : 1;
    unsigned char style   : 2;
    unsigned char color   : 3;
};
```

```
struct window
{
    struct wnd_flags flags;
};
```

...

```
w.flags = SHOW | (BORDER << 1) | ((RED | BLUE) << 3);
```

```
w.flags.show    = SHOW;
w.flags.style    = BORDER;
w.flags.color    = RED | BLUE;
```

```
part = (w.flags >> 1) & 0x3;
if (part & BORDER)
    printf("BORDER is present\n");
else
    printf("BORDER is not present\n");
if (w.flags.style & BORDER)
    printf("BORDER is present\n");
else
    printf("BORDER is not present\n");

if ((w.flags >> 3) & GREEN)
    printf("GREEN is present\n");
else
    printf("GREEN is not present\n");
if (w.flags.color & GREEN)
    printf("GREEN is present\n");
else
    printf("GREEN is not present\n");
```

# Битовые поля

```
w.flags = 5; // ошибка компиляции
```

```
unsigned char f;
```

```
...
```

```
f = w.flags; // ошибка компиляции
```

```
struct window
```

```
{
```

```
    // другие обычные поля
```

```
    unsigned char show : 1;
```

```
    unsigned char style : 2;
```

```
    unsigned char color : 3;
```

```
};
```