CÁC THAO TÁC XỬ LÝ TRÊN BIT

I. Các toán tử thao tác bit

Trong ngôn ngữ máy tính, các phép toán thao tác bit thực hiện tính toán (theo từng bit) trên một hoặc hai chuỗi bit, hoặc trên các số nhị phân.

Với nhiều loại máy tính, việc thực thi các phép toán thao tác bit thường nhanh hơn so với khi thực thi các phép toán cộng, trừ, nhân, hoặc chia.

1. Toán tử and

- Ký hiệu trong C ++: '&'
- Ý nghĩa: đây là phép nhân logic trên các bit, kết quả nhân logic tương ứng trên từng cặp bit của 2 dãy bit như bảng sau:

a	b	a & b
0	0	0
0	1	0
1	0	0
1	1	1

- Quy tắc nhớ: kết quả tích logic 2 bit bằng 1 khi cả hai bit bằng 1. Các trường hợp khác tích bằng 0.

2. Toán tử or

- Ký hiệu trong C ++: '|'
- Ý nghĩa: đây là phép cộng logic trên các bit, kết quả cộng logic tương ứng trên từng cặp bit của 2 dãy bit như bảng sau:

a	b	a | b
0	0	0
0	1	1
1	0	1
1	1	1

- Quy tắc nhớ: kết quả cộng logic 2 bit bằng 0 khi cả hai bit bằng 0. Các trường hợp khác kết quả cộng bằng 1.

3. Toán tử not

- Ký hiệu trong C ++ : '~'
- Ý nghĩa: đây là phép toán đảo bit. Đảo mọi giá trị bit 0 thành bit 1

a	b	~a	~b
0	0	1	1
0	1	1	0
1	0	0	1
1	1	0	0

4. Toán tử xor

- Ký hiệu trong C ++ : '^'
- Ý nghĩa: đây là phép tổng loại trừ logic trên các bit, kết quả thực hiện tương ứng trên từng cặp bit của 2 dãy bit như bảng sau:

a	b	a ^ b
0	0	0
0	1	1
1	0	1
1	1	0

- Quy tắc nhớ: kết quả tổng loại trừ logic 2 bit bằng 0 khi cả hai bit bằng 0 hoặc bằng 1. Các trường hợp khác kết quả bằng 1.

5. Toán tử dịch trái

- Ký hiệu trong C ++ : '<<'
- Ý nghĩa: x << i : dịch x sang trái i bit
- VD:
  - x có dãy bit như sau: 01010101.
  - Thực hiện dịch x sang trái 3 bit: x << 3
  - Kết quả dãy bit nhận được: 10101**000**

6. Toán tử dịch phải

- Ký hiệu trong C ++ : '>>'
- Ý nghĩa: x >> i : dịch x sang phải i bit
- VD:
  - x có dãy bit như sau: 01010101.
  - Thực hiện dịch x sang phải 3 bit: x >> 3
  - Kết quả dãy bit nhận được: **000**01010

7. Ví dụ

```
void main()
{
	int a = 5; // 0000 0000 0000 0101
	int b = 6; // 0000 0000 0000 0110
	int z1, z2, z3, z4, z5, z6;
	z1 = a & b; // 0000 0000 0000 0100
	z2 = a | b; // 0000 0000 0000 0111
	z3 = a ^ b; // 0000 0000 0000 0011
	z4 = ~a; // 1111 1111 1111 1010
	z5 = a >> 2;// 0000 0000 0000 0001
	z6 = a << 2;// 0000 0000 0001 0100
}
```

II. Các kiểu dữ liệu trong C++

1byte tương ứng 8 bit

Loại dữ liệu	Tên kiểu	Số ô nhớ	Miền giá trị
Ký tự	char	1 byte	$-2^7 \dots 2^7-1$
	unsigned char	1 byte	$0 \dots 2^8-1$
Số nguyên	short	2 byte	$-2^{15} \dots 2^{15}-1$
	unsigned short	2 byte	$0 \dots 2^{16}-1$
	int	4 byte	$-2^{31} \dots 2^{31}-1$
	unsigned int	4 byte	$0 \dots 2^{32}-1$
	long	4 byte	$-2^{31} \dots 2^{31}-1$
Số thực	float	4 byte	$\pm 10^{-37} \dots \pm 10^{+38}$
	double	8 byte	$\pm 10^{-307} \dots \pm 10^{+308}$

III. Ví dụ ứng dụng:

1. Ví dụ 1: viết chương trình tìm dãy bit nhị phân của số nguyên 4 byte có dấu

```
#include<stdio.h>
#include<conio.h>

// Hàm lấy 1 bit của x tại vị trí i tính (từ phải sang)
bool GetBit(int x, int i)
{
	return (x>>i)  & 1;
}

// Tìm dãy bit của x và gán  vào mảng bit kết quả a
void TimDayBit(int x, bool a[32])
{
	int k=0;
	for(int i = 31; i>=0;  i--)
	{
		bool bit = GetBit(x,i);
		a[k++] = bit;
	}
}

// Hàm xuất dãy bit
void XuatDayBit(bool a[32])
{
	for(int i = 0; i < 32; i ++)
		printf("%d",a[i]);
}
void main()
{
	// Nhập số nguyên cần tính dãy bit
	int x; //số nguyên có dấu
	printf("Nhap so nguyen: ");
	scanf("%d",&x);

	// Tìm dãy bit của x
	bool a[32];
	TimDayBit(x,a);

	// Xuất dãy bit tìm  được
	printf("Day nhi phan la: ");
	XuatDayBit(a);

	getch();
}
```

2. **Ví dụ 2:** viết chương trình tìm dãy bit nhị phân của số chấm động chính xác đơn. (float)

❖ **Hướng dẫn:** dựa vào bản chất lưu trữ nhị phân, có thể dễ dàng xuất dạng biểu diễn nhị phân của số chấm động hay chuyển dãy bit biểu diễn thành các giá trị lưu trữ tương ứng dựa vào các thao tác luận lý AND, OR, SHIFT LEFT, SHIFT RIGHT,…

❖ **Chú ý:**

- Không thể thực hiện các thao tác luận lý **trực tiếp trên số thực**

```
float x = -5.25;
x >> 1; // báo lỗi
```

➔ Cần lấy con trỏ tới vùng nhớ chứa số thực:

```
float x = -5.25;
// Thực hiện ép kiểu như sau:
long *p = (long *)&x;
// Và thực hiện các thao tác luận lý:
*p >> 1;
```

- Thực hiện tương tự như trên số nguyên để tìm dãy bit.

References:

1. http://vi.wikipedia.org/wiki/Ph%C3%A9p_to%C3%A1n_thao_t%C3%A1c_bit
2. http://www.cs.umd.edu/class/sum2003/cmsc311/Notes/BitOp/bitshift.html

---

còn tiếp…