额外加分题-第 4 题

1850059 计 1 班 杨志远

代码：

```
#include <iostream>
using namespace std;
int main()
{
    int a = 1, b = 32;
    cout << (a << b) << endl;
    cout << (1 << 32) << endl;
    return 0;
}
```

第二句 cout 语句：warning C4293: “<<”: Shift 计数为负或过大，其行为未定义

原因分析：

第一句 cout 语句中，在变量的左移运算中，当<<右边的变量大于变量类型的字节数时，必然会发生溢出，这时会先将右边的变量进行一次取余运算（此处相当于 b%32），也就避免了溢出的问题，不会报告 warning。1<<(32%32) -> 1<<0 -> 1，与输出结果相符

第二句 cout 语句中，常量的左移运算完全遵照定义，$(0000\ 0001)_{16}$ 经过左移 32 位后后，变为了$(1\ 0000\ 0000)_{16}$。由于 int 只有 32 位，所以截断越界的“1”，同时报告 warning，最后的结果就是 0，与输出结果相符。

进一步研究：

```
    int a = 1, b = 32;
    cout << (a << b) << endl;
    cout << (a << 31) << endl;
    cout << (a << 32) << endl;              //warning
    cout << (a << 63) << endl;              //warning
    cout << (a << 64) << endl;              //warning
    cout << (a << 70) << endl << endl;  //warning
    cout << (1 << (b - 6)) << endl;
    cout << (1 << b) << endl;
    cout << (1 << (b + 6)) << endl;
    cout << (1 << 31) << endl;
    cout << (1 << 32) << endl;              //warning
    cout << (1 << 63) << endl;              //warning
    cout << (1 << 64) << endl;              //warning
    cout << (1 << 70) << endl << endl;  //warning

    long long c = 1, d = 32, e = 64;
    cout << (c << d) << endl;
    cout << (c << 31ll) << endl;
```

```
    cout << (c << 32ll) << endl;
    cout << (c << 63ll) << endl;
    cout << (c << 64ll) << endl;                //warning
    cout << (c << 70ll) << endl << endl;        //warning
    cout << (1ll << d) << endl;
    cout << (1ll << (e - 1)) << endl;
    cout << (1ll << e) << endl;
    cout << (1ll << (e + 6)) << endl;
    cout << (1ll << 31ll) << endl;
    cout << (1ll << 32ll) << endl;
    cout << (1ll << 63ll) << endl;
    cout << (1ll << 64ll) << endl;              //warning
    cout << (1ll << 70ll) << endl << endl;      //warning
```

运行的结果为：

```
1
-2147483648
1
-2147483648
1
64

67108864
1
64
-2147483648
0
0
0
0

4294967296
2147483648
4294967296
-9223372036854775808
0
0

4294967296
-9223372036854775808
0
0
2147483648
4294967296
```

-9223372036854775808
0
0

经过观察可以发现，int 类型的输出的结果与预期一致，证明结论正确。而根据 long long 类型的输出结果，可以发现，long long 类型在左移运算时，<<右边的数字不会进行取余运算。

总的来说：

1、在同一整数类型下，<<运算左边的数字无论是常量还是变量都不会影响结果，不同的整数类型决定了最后的结果是否会溢出。

2、<<运算右边的数字如果是过大（超过整数类型的字节数）的常量则**会**报告 warning，如果是变量则无论是否过大都**不会**报告 warning。

3、<<运算右边的数字如果是 int 类型则会进行取余运算（%32）得出某个数字（例如 1<<3202 的结果是 4），猜测可能是为了提高运算速度，如果是 long long 类型则会直接使运算结果为 0。