bitset

> 注意：
> 左边是高位b[9]， 右边的才是低位b[0]。
> 0001000010 ，注意 高 -----> 低

bitset容器是一个bit位元素的序列容器，每个元素只占一个bit位，取值为0或1，有节省内存空间，下面是bitset的存储示意图，它的16个元素只使用了两个字符的空间。【就是一个字符占一个char的内存】

7	6	5	4	3	2	1	0
0	1	0	0	0	1	1	1
						1	0
15	14	13	12	11	10	9	8

1- 方法总纲

- b.any() b中是否存在置为1的二进制位?
- b.none() b中不存在置为1的二进制位吗?
- b.count() b中置为1的二进制位的个数
- b.size() b中二进制位的个数
- b[pos] 访问b中在pos处的二进制位
- b.test(pos) b中在pos处的二进制位是否为1?
- b.reset() 把b中所有二进制位置为0
- b.reset(pos) 把b中在pos处的二进制位置为0
- b.flip() 把b中所有二进制位函数取反
- b.flip(pos) 把b中在pos处的二进制位取反
- b.to_ulong() 用b中同样的二进制位返回一个unsigned long值 。还有最大的long long unsigned int
- b.set() 把b中所有二进制位都置为1
- b.set(pos) 把b中在pos处的二进制位置为1

2- 设置元素值

- 采用下标法
- 采用set()方法，一次性将元素设置为1
- 采用set(pos)方法，将pos位设置为1
- 采用reset(pos)方法，将某pos位设置为0

```
#include<iostream>
#include<bitset>        /// header bitset<>
```

```cpp
using namespace std;

int main()
{
    bitset<10> b;       /// [0,9]
    ///1- index set
    b[1] = 1;
    b[6] = 1;
    b[9] = 1;

    for(int i = b.size()-1; i >= 0; i --)
        cout<<b[i] << " ";       /// 1 0 0 1 0 0 0 0 1 0
    cout<<endl;

    /// 2- all is zero
    b.reset();
    for(int i = b.size()-1; i >= 0; i --)
        cout<<b[i] << " ";       /// 0 0 0 0 0 0 0 0 0 0
    cout<<endl;

    ///3- 指定位置设1
    b.set(1, 1);
    b.set(6, 1);
    b.set(9, 1);

    for(int i = b.size()-1; i >= 0; i --)
        cout<<b[i] << " ";       ///同上,  1 0 0 1 0 0 0 0 1 0
    cout<<endl;

    ///4- 所有位设置为1
    b.set();
    for(int i = b.size()-1; i >= 0; i --)
        cout<<b[i] << " ";      /// 1 1 1 1 1 1 1 1 1 1
    cout<<endl;

    cout << b.to_ulong() << endl;    /// 10 个1, 是1023
    cout <<sizeof(long long unsigned int ) << endl; // 8 字节, 64bit
    return 0;
}
```

3- 输出元素

有两种方法。把它当成容器就好。

- 下标法
- 直接向输出流输出全部元素

```cpp
#include<iostream>
#include<bitset>
using namespace std;

int main()
{
    bitset<10> b;
    b.set(1, 1);
    b.set(6, 1);

    for(int i = b.size()-1; i >= 0; i --)
        cout<<b[i];     /// 0001000010 , 注意 高 -----> 低
    cout<<endl;

    cout<<b<<endl;      /// 0001000010
    return 0;
}
```