bitset

bitset是一个类，定义在bitset头文件中。

- 可视为一个bool类型的数组，不过每个元素只占用1bit空间，bool里面则占用1byte。
- 模板类，template<size_t N> class bitset;
- 从0开始编号，一直到N-1。

定义和初始化bitset

bitset类具有固定大小，定义如下：

```
bitset<32> bitvec(1u);  // 32bit, 低位为1,其他位位0
```

其中大小（模板中的N）是一个常量表达式。0号位称为低位（low-order），N-1号位称为高位（high-order）。

用unsigned值初始化bitset

当我们用一个整型值初始化bitset时，该值会被转换位unsigned long long类型，并被当做位模式来处理。bitset中的二进制位数将是此模式的一个副本。如果bitset的大小大于一个unsigned long long的二进制数，则剩余的高位被置为0。如果小于，超出部分将会被丢弃。（这也是很intuitive的）。

```
// bitvec1比初始值小;初始值中的高位被丢弃
bitset<13> bitvec1(0xbeef);          // 二进制位序列位1 1110 1110 1111
// bitvec2比初始值大;高位被置0
bitset<20> bitvec2(0xbeef);          // 二进制位序列位0000 1011 1110 1110 1111
// 64-bit的机器中,long long 0ULL时64个0bit,以此~0ULL是64个1
bitset<128> bitvec3(~0ULL);          // 0~63位为1,64~127位为0
```

string初始化bitset

我们可以从一个string或一个字符数组指针来是初始化bitset。两种情况，都直接表示位模式。

```
bitset<32> bitvec4("1100"); // 2,3位为1,剩余为0
```

下标最小的对应高位，反之亦然。也就是说，❗ string的编号和bitset的刚好相反。

```
// 使用部分初始化
string str("111100001111");
bitset<32> bitvec5(str,5,4);     // 从str[5]开始的四个二进制位
                                 // 使用起来和string.substr(str,beg,cnt)类似。
bitset<32> bitvec6(str,str.size()-4);// 使用最后4个字符
                                 // 和string.substr(str,beg)类似
```

bitset的相关操作

bitset支持内置的位运算符。bitset的方法有如下：（置位和复位是对应的，简单理解位置位就是true或1，复位就是置为false或0）

- b.any()，b是否存在置位的二进制位
- b.all()，b所有位置是否都置位
- b.count()，b中置位的个数
- ...（用的时候再查即可）。

提取bitset的值

to_ulong和to_ullong返回一个值，保存了与bitset对象相同的位模式。只有当bitset的大小小于等于对应的大小(to_ulong位unsigned long,to_ullong为unsigned long long)时，我们才能使用这两个操作。

```
unsigned long ulong=bitvec3.to_ulong();
```

如果bitset的大小大于给定的类型，则这两个操作会抛出一个overflow_error异常。

bitset的IO运算符

输入运算符从一个输入流中读取字符，保存到一个临时的string对象中，直到读取的字符数达到相应bitset的大小时，或遇到不是1或0的字符时，或是遇到文件尾或输入错误，读取终止。随后，用临时的string对象来初始化bitset。

```
bitset<16> bits;
cin>>bits;
cout<<"bits:"<<bits<<endl;
```

使用bitset

略。