eecrazy的专栏

个人资料

令羽

访问： 273859次
积分： 3640
等级： BLOG 5
排名： 第8162名

原创： 116篇 转载： 10篇
译文： 1篇 评论： 34条

bitset 用法整理

标签： string vector constructor ubuntu os linux

2012-08-30 16:27 11628人阅读 评论(

分类：

STL（4） ACM（49）

在项目中需要使用到10进制48位的数字按二进制由高到低解释，然后按每一位是0还是1来判断报警或错误状态。所以，在**Linux**中的C++下需要用到二进制转换以及按位解析。收集到了一些资料，自己保存一下啊。

如下：

bitset 用法整理

构造函数

bitset<n> b;

b有n位，每位都为0.参数n可以为一个表达式.

如bitset<5> b0;则"b0"为"00000";

bitset<n> b(unsigned long u);

b有n位,并用u赋值;如果u超过n位,则顶端被截除

如:bitset<5>b0(5);则"b0"为"00101";

bitset<n> b(string s);

b是string对象s中含有的位串的副本

string bitval ("10011");

bitset<5> b0 (bitval4);

则"b0"为"10011";

bitset<n> b(s, pos);

b是s中从位置pos开始位的副本,前面的多余位自动填充0;

string bitval ("01011010");

bitset<10> b0 (bitval5, 3);

则"b0" 为 "0000011010";

bitset<n> b(s, pos, num);

b是s中从位置pos开始的num个位的副本,如果num<n,则前面的空位自动填充0;

string bitval ("11110011011");

bitset<6> b0 (bitval5, 3, 6);

则"b0" 为 "100110";

os << b
把b中的位集输出到os流
os >>b
输入到b中,如"cin>>b",如果输入的不是0或1的字符,只取该字符前面的二进制位.

bool any( )
是否存在置为1的二进制位？和none()相反

bool none( )
是否不存在置为1的二进制位,即全部为0？和any()相反.

size_t count( )
二进制位为1的个数.

size_t size( )
二进制位的个数

flip()
把所有二进制位逐位取反

flip(size_t pos)
把在pos处的二进制位取反

bool operator[](size_type _Pos)
获取在pos处的二进制位

set()
把所有二进制位都置为1

set(pos)
把在pos处的二进制位置为1

reset()
把所有二进制位都置为0

reset(pos)
把在pos处的二进制位置为0

test(size_t pos)
在pos处的二进制位是否为1？

unsigned long to_ulong( )
用同样的二进制位返回一个unsigned long值

string to_string ()
返回对应的字符串.

详细请翻阅msdn.

另转：http://blog.csdn.net/auto_ptr/archive/2010/10/22/5958924.aspx

bitset的使用示例

std::bitset是STL的一个模板类，它的参数是整形的数值，使用位的方式和数组区别不大，相当于只能存一个位的数组

看一个例子

view plaincopy to clipboardprint?

```
bitset<20> b1(5);
  cout<<"the set bits in bitset<5> b1(5) is:"
    << b1 <<endl;
bitset<20> b1(5);
  cout<<"the set bits in bitset<5> b1(5) is:"
  << b1 <<endl;
```

结果是 the set bits in bitset<5> b1(5) is:00000000000000000101

它是以整数5传递进去，而以二进制数打印出来。

bitset还可以用作字符串转为整型

view plaincopy to clipboardprint?

```
string bitval2;
  cin>>bitval2;
//  int length = strlen("11101101100");
// bitset<11>b2(bitval2);
   bitset<11> b2(bitval2);
  cout<<b2<<"is  "<<b2.to_ulong()<<endl;
string bitval2;
  cin>>bitval2;
//  int length = strlen("11101101100");
// bitset<11>b2(bitval2);
   bitset<11> b2(bitval2);
  cout<<b2<<"is  "<<b2.to_ulong()<<endl;
```

以及整形转为字符串

view plaincopy to clipboardprint?

```
int interge1;
cin>>interge1;

cout<<"**********整数转为字符串**********"<<endl;
bitset<11>b3(interge1);
cout<<b3.to_ulong()<<"is  "<<b3.to_string()<<endl;
  int interge1;
  cin>>interge1;

  cout<<"**********整数转为字符串**********"<<endl;
  bitset<11>b3(interge1);
  cout<<b3.to_ulong()<<"is  "<<b3.to_string()<<endl;
```

在网上看到还有一篇关于bitset写的不错的文章，不知到作者是谁，粘贴自用之：

bitset如何初始化、如何转化为double类型的小数、如何进行交叉（可以尝试用string作为中间量，因为bitset可以

初始化的，但是这样的构造和传递会消耗很多的时间——我讨厌这种不必要的消耗！）

假如说我希望计算的精度足够高，将bitset取为64位，那么什么类型的数才能输出？如果不需要输出，那么在取精度

候，如何将一个64位的bitset转化为double类型的小数？（可能需要自己编程实现了）

如何将一个double类型的数字转化为bitset，也就是二进制编码，方便我们做交叉、变异。

(说得简单点，以上两个就是解码和编码的问题)——文字很乱，整理一下！

如何实现两个bitset的合并？小数部分、整数部分，如果能够合并，那写程序又会方便多了！比如：两个32位的bits

成一个64位的bitset！（是不是又要利用string进行转换呢？如何转换？）

代码说明：将bitset的某一位置为1

```
bitset<32> bits;
for（int  i =0;i<5;i++)
bits. set (i); //i为需要被置为1的位数
cout<<bits<<endl;
```

bitset的函数用法

注意事项

你看得出来下面的代码为什么输出7和9吗？

```
#include<iostream>
#include<bitset>
using  namespace  std;
void  main()
{
   bitset<4> bit(1111);
   cout<<bit.to_ulong()<<endl;
   bitset<4> ait(1001);
   cout<<ait.to_ulong()<<endl;
}
```

原因很简单：bitset调用的构造函数，1111为十进制，换成二进制为 0x10001010111，最后4位为0111，输出就是

你想规定bitset里面的每一位，那么最好用string类型：bitset<4> bits("1111"); 这样输出就是15了。

字符串合并以及输出的问题，要搞定，还真麻烦.....为了偷懒，在多个类型之间转来转去的.....不过写起来真的很简单哈！有现成的方法就用呗！不管效率了.....

```
#include <iostream>
#include <string>
#include <iostream>
#include "afxwin.h"
using  namespace  std;
int  main(){
   CString s1 =  "abcd" ;
   CString s2 =  "xyzw" ;
   CString s3 = s1+s2;
   cout<<(LPCTSTR)s1<<endl<<(LPCTSTR)s3<<endl;
    string  s4 = (LPCTSTR)s3;
   cout<<s4<<endl;
    return  0;
}
```

下面是2个bitset合并的代码例子

```
#include <bitset>
#include <iostream>
#include <string>
#include <iostream>
#include "afxwin.h"
using  namespace  std;
int  main(){
   bitset<4> bits1( "1111" );
   bitset<4> bits2( "0000" );
    int  i = bits1.size()+bits2.size();
//  bitset<i> bits3;  //不能使用动态参数作为模板参数，能不能想办法解决？
```

```
    bitset<128> bits3;
    int  j=0;
    for  (j=0;j<bits1.size();j++)
    {
        if  (bits1[j]==1)
          bits3. set (j);
    }
    for  (j=bits1.size();j<bits1.size()+bits2.size();j++)
    {
        if  (bits2[j-bits1.size()]==1)
       {
          bits3. set (j);
       }
    }
    cout<<bits3<<endl<<bits3.to_ulong()<<endl;
    return  0;
}
```

bitset能够达到的最大长度

```
#include <bitset>
#include <vector>
#include <iostream>
#include <string>
#include <iostream>
#include "afxwin.h"
using   namespace  std;
int  main(){
    bitset<1000000> bits;// 一百万差不多到顶了，如果再加一个0，到达一千万，就会崩溃。为什么？
    cout<<bits[0];
    return  0;
}
```

想使用动态的bitset吗？

dynamic_bitset可以满足我的需求！这实在太棒了！boost万岁！ps：不知道会造成多大的效率影响？和固定长度的
较起来，虽然固定一点、浪费一点空间，但是如果更快的话，也是值得了。另外：dynamic_bitset不能在 vc6下通过
译.....

bit_vector

这个“位向量组”在SGI STL中实现，VC6中没有。从名字和功能介绍上就可以看出来：这是一个可以像操作vector一
便的容器，可以push_back每一位。效率有待实验，我是在一本书上偶然看到这个库的。

然而，令我失望的是：在ubuntu和VC6下，都没有bit_vector，必须安装SGI 版本的stl才行呢。

结论：对于这方面，看样子还是凑合着用吧！（实现简单的bitset，空间方面嘛，稍微浪费一点也就是了)。

对于上例中提及的7,9的问题，有人提问如下：

编译结果为7 和 9， 请高手解释 不太理解

```
#include<iostream>
#include<bitset>
using namespace std;
void main(){
bitset<4> bit(1111);
cout<<bit.to_ulong()<<endl;
bitset<4> ait(1001);
cout<<ait.to_ulong()<<endl;
}
```

问题补充：

那对于以下的代码呢 编译结果为9 如何解释呢 #include<iostream>#include<bitset>using namespace std;voi

main(){ bitset<5> a(00111); cout<<a.to_ulong();}

回答如下：

MSDN上如是说：

bitset(unsigned long val);

The constructor sets only those bits at position j for which val & 1 << j is nonzero.

理解上面的这段话后， 用计算器计算得到：

D(1111) = B(10001010111)

D(1001) = B(1111101001)

D为十进制,B为二进制 而你定义的bitset<4>，只有四位，那么取1111的二进制的最后四位即0111,得十进制7，同理
得到1001，即十进制9

ps:对于你上面补充的，我想说如果是bitset<5> (111),那么这个111是十进制，但是当你变成bitset<5>(00111) 或
bitset<5>(0111),那么这个111是八进制，也就是十进制的73,即1001001，取后面五位即9 再说得细一点，bitset<N
中的N指明内存中的位数，二进制。

M指明数的大小，至于这个数的进制，你可以查一下资料，C语言中在一个整数前面加0表示八进制，在一个整数前
表十六进制。

参考资料：MSDN

我的代码：

[html]

```
#include <iostream>
#include <bitset>
using namespace std;
//bitset的所有操作：
//any();none();test();bit[];set();reset();to_string();to_ulong();
//count();flip();分别用十进制，八进制，十六进制，字符串赋值。
int main()
{
    bitset<32> bitvec(8);//0~31
    bool flag = bitvec.any();//判断是否存在某位或者多位为1,有则返回true
    bool flag1 = bitvec.none();//判断是否所有的位都是0，是则返回true
    bool flag2 = bitvec.test(3);//测试第4位是否为1，是则返回true<<
    cout<<"bitvec的值为："<<bitvec<<endl;
    cout<<"第4位为："<<bitvec[3]<<endl;//输出第4位的值
    bitvec.reset(3);//将第4位设置为0，或者bitvec[3] = 0
    cout<<"第4位为："<<bitvec[3]<<endl;//输出第4位的值
    bitvec.reset();//将所有位设置为0
    cout<<"bitvec的值为："<<bitvec.to_string()<<endl;
    bitvec.set();//将所有位设置为1
    cout<<"bitvec的值为："<<bitvec.to_string()<<endl;
    cout<<"bitvec的值为："<<bitvec<<endl;
    cout<<"bitvec的值为："<<bitvec.to_ulong()<<endl;
    cout<<"bitvec中1的个数为："<<bitvec.count()<<endl;
    bitvec = 8;
    cout<<"bitvec的值为："<<bitvec.to_string()<<endl;
    bitvec.flip();//将所有的位翻转
    cout<<"bitvec的值为："<<bitvec.to_string()<<endl;
    bitvec.flip(0);//翻转第一位
    cout<<"bitvec的值为："<<bitvec.to_string()<<endl;
    bitvec = 0xffff;//设置低16位为1
    cout<<"bitvec的值为："<<bitvec.to_string()<<endl;
    bitvec = 012;//用八进制值012设置bitvec
    cout<<"bitvec的值为："<<bitvec.to_string()<<endl;
    string bit = "1011";
    bitset<32> bitvec1(bit);//用字符串对象初始化bitset<32>对象
    cout<<"bitvec1的值为："<<bitvec1.to_string()<<endl;
    string bit1 = "11111101011000011010101";
    bitset<32> bitvec2(bit1,6);//用从第6位开始到字符串结束这一部分初始化bitvec2
    cout<<"bitvec2的值为："<<bitvec2.to_string()<<endl;
    bitset<32> bitvec3(bit1,6,4);//用从第6位开始，长度为4这一部分初始化bitvec3;
    cout<<"bitvec3的值为："<<bitvec3.to_string()<<endl;
}
```

顶 0 踩 0

上一篇 hdu4389-X mod f(x)-多校9-1010题解
下一篇 学习Linux的优势

相关文章推荐

- bitset 用法整理
- bitset用法整理
- C++标准库：bitset 用法整理
- C++标准库：bitset 用法整理
- C++ STL bitset 用法整理
- bitset 用法整理
- bitset 用法整理
- C++标准库：bitset 用法整理

参考知识库

.NET知识库
3990 关注 | 839 收录

Linux知识库
12240 关注 | 3980 收录

C语言知识库
9176 关注 | 3472 收录

软件测试知识库
4819 关注 | 318 收录

猜你在找

Linux环境C语言编程基础
《C语言/C++学习指南》Linux开发篇
Linux C高级编程进阶之指针与数组解析
Linux下的C语言程序设计
零基础学会在Linux上编译调试C++项目
4.5.数组&字符串&结构体&共用体&枚举-C语言高级专题第5
基于Ubuntu Core系统的DragonBoard 410c开发案例解析
VC++游戏开发基础系列从入门到精通
VC++Windows多线程实战图片编辑器
I2C总线和触摸屏驱动移植实战-linux驱动开发第9部分

查看评论

暂无评论

您还没有登录,请[登录]或[注册]

* 以上用户言论只代表其个人观点，不代表CSDN网站的观点或立场

核心技术类目

全部主题 Hadoop AWS 移动游戏 Java Android iOS Swift 智能硬件 Docker OpenStack
VPN Spark ERP IE10 Eclipse CRM JavaScript 数据库 Ubuntu NFC WAP jQuery
BI HTML5 Spring Apache .NET API HTML SDK IIS Fedora XML LBS Unity
Splashtop UML components Windows Mobile Rails QEMU KDE Cassandra CloudStack

FTC coremail OPhone CouchBase 云计算 iOS6 Rackspace Web App SpringSide
Maemo Compuware 大数据 aptech Perl Tornado Ruby Hibernate ThinkPHP HBase
Pure Solr Angular Cloud Foundry Redis Scala Django Bootstrap