各个STL容器模板都接受一个可选的模板参数，该参数指定使用那个分配器对象来管理内存。

例如，vector模板的开头：

template <class T,class Allocator = allocator<T>>

       class vector{…}

如果省略该模板参数的值，则容器模板将默认使用allocator<T>类。这个类使用new和delete。

可对矢量执行的操作：

size()--返回容器中的元素数目；

swap()--交换两个容器的内容；

begin()--返回一个指向容器中第一个元素的迭代器；

end()--返回一个表示超过容器为的迭代器；

### template<class T> class Eigen::aligned\_allocator< T >

STL compatible allocator to use with with 16 byte aligned types.

Example:

\* // Matrix4f requires 16 bytes alignment:

\* std::map< int, Matrix4f, std::less<int>,

\* aligned\_allocator<std::pair<const int, Matrix4f> > > my\_map\_mat4;

\* // Vector3f does not require 16 bytes alignment, no need to use Eigen's allocator:

\* std::map< int, Vector3f > my\_map\_vec3;

new创建类对象需要指针接收，一处初始化，多处使用

* new创建类对象使用完需delete销毁
* new创建对象直接使用堆空间，而局部不用new定义类对象则使用栈空间
* new对象指针用途广泛，比如作为函数返回值、函数参数等
* 频繁调用场合并不适合new，就像new申请和释放内存一样

使用普通方式创建的类对象，在创建之初就已经分配了内存空间。而类指针，如果未经过对象初始化，则不需要delete释放

C++11随机数

C++11 introduces several pseudo-random number generators designed to replace the good-oldrand from the C standard library. I’ll show basic usage examples of std::mt19937, which provides a random number generation based on [Mersenne Twister](https://en.wikipedia.org/wiki/Mersenne_twister) algorithm. Using the Mersenne Twister implementation that comes with C++1 has advantage over rand(), among them:

1. mt19937 has much longer period than that of rand, e.g. it will take its random sequence much longer to repeat itself.
2. It much better statistical behavior.
3. Several different random number generator engines can be initiated simultaneously with different seed, compared with the single “global” seed srand() provides.

The downside is that mt19937 is a bit less straight-forward to use. However, I hope this post will help with this point :-).  
  
We start with a basic example:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13 | #include <iostream>  #include <random>    using namespace std;    int main()  {  mt19937 mt\_rand(time(0));    cout << mt\_rand() << endl;    return 0;  } |

Line 7 creates a new std::mt19937 object and seeds it with time(0). This is just like callingsrand(time(0)). Subsequent calls to the newly created object, mt\_rand will return a random 32-bit unsigned int.

If you want integers in a specific range, instead all kinds of arithmetics, C++11 provides convinient wrappers:

|  |
| --- |
| mt19937::result\_type seed = time(0);  auto dice\_rand = std::bind(std::uniform\_int\_distribution<int>(1,6),  mt19937(seed)); |
| mt19937::result\_type seed = time(0);  auto real\_rand = std::bind(std::uniform\_real\_distribution<double>(0,1),  mt19937(seed)); |

In the first example, each call to dice\_rand() will return an integer between 1 and 6 (inclusive). In the second example each call to real\_rand() will return a double in the range [0,1) (including 0, excluding 1). Note that usage of std::bind requires #include <functional>.

Finally if you want to go C++11 all the way, you can also replace time(0) with a proper call forstd::chrono when seeding the random number generator:

|  |
| --- |
| auto seed = chrono::high\_resolution\_clock::now().time\_since\_epoch().count();  mt19937 mt\_rand(seed); |

The above code requires adding #include <chrono> to the list of includes.

如果你只是想简单的不用科学计算法来表示一个数，那么在输出的时候用  
  
cout<<std::fixed<<yournumber;

## 将double型数字保留几位小数(这个方法是抄的啦)

**[cpp]** [view plain](http://blog.csdn.net/kid_u_forfun/article/details/8628732) [copy](http://blog.csdn.net/kid_u_forfun/article/details/8628732)

[print?](http://blog.csdn.net/kid_u_forfun/article/details/8628732)

1. <span style="font-size:14px;">Double d = 10000.0 / 333;
2. String s = **new** DecimalFormat("#.###").format(d);</span>

关于 C++ 输出时的小数点后的位数是很基础的东西，应该掌握。

double a = 3.141592, b = 2213242.329843;  
cout.precision(6);  
cout << a << endl << b;

输出的是：

3.14159  
221324

上面输出的是“6个有效数字”，而想要输出 3.141592 和 2213242.329843 的数字，不管小数点前面有几位，那就要：

double a = 3.141592, b = 2213242.329843;  
cout.setf(ios::showpoint); //设置为始终输出小数点后的数字，就是说 a = 3，它也输出 3.00000 这样  
cout.precision(6);  
cout.setf(ios::fixed); //设置为小数位始终有 6 位，没有这个的话就会像上面那个代码那样固定的不是小数点后面的数字了。  
cout << a << endl << b;

输出就是：

3.141592  
2213242.329843

比如pi=3.14159265333,保留三位小数：

**[cpp]** [view plain](http://blog.csdn.net/kid_u_forfun/article/details/8628732) [copy](http://blog.csdn.net/kid_u_forfun/article/details/8628732)

[print?](http://blog.csdn.net/kid_u_forfun/article/details/8628732)

1. cout<<fixed<<setprecision(3);
2. cout<<pi<<endl;

备注：必须加头文件iomanip，忘记这个，你就等着吧。