Templates and Generic Programming

OOP (通过多态) 和泛型 (Generic)都能处理在编写程序时不知道类型的情况。不同之处在于，

- 动态绑定可以处理类型在运行之前都未知的情况
- 泛型编程中，编译时可以确定类型

Templates are the foundation of generic programming in C++. A template is a blueprint or formula for creating classes or functions. When we use a generic type, such as `vector`, or a generic function, such as `find`, we supply the information needed to transform that blueprint into a specific class or funtion. That transformation happens during compilation.

Defining a Template

Avoid the redundant overloaded function:

```
int cmp(const string &a,const string &b)
int cmp(const int &a,const int &b)
int cmp(const double &a,const double &b)
```

> May be similar to `void *` in C to do the same works.
> `int cmp(const void *,const void *)`

Function Templates

A function template is a formula from which we can generate type-specific versions of that function. The template version of `cmp` looks like:

```
template <typename T>
int cmp(const T &a,const T &b){}
```