函数指针，指针函数，数组指针，指针数组 区分

函数指针：重点在指针，表示它是一个指针，它指向的是一个函数。eg: int (*fun)();

指针函数：重点在函数，表示它是一个函数，它的返回值是指针。 eg: int* fun();

数组指针：重点在指针，表示它是一个指针，它指向的是一个数组。int (*fun)[8];

指针数组：重点在数组，表示它是一个数组，它包含的元素是指针 itn* fun[8];

类模板（class template） — 模板类（template class） 区分：

类模板：重点在模板，表示它是一个模板，专门用于产生类的模子。

```
template <tempname T>
class Vector
{
.....................................
}
```

使用这个 Vector 模板就可以产生很多的 class（类），Vector <int> 、Vector <char> 、Vector < Vector <int> >。

模板类：重点在类，表示的是由一个模板生成而来的类。

例子：上面的 Vector <int> 、Vector <char> 、……全是模板类。

函数模板（function template）——模板函数（template function）

函数模板的重点是模板。表示的是一个模板，专门用来生产函数。

eg:

```
template <tempname T>
void fun(T a)
{
        …
}
```

在运用的时候，可以显式（explicitly）生产模板函数,fun(int) ,fun(double) , fun(shape*)。

也可以在使用的过程中由编译器进行模板参数推导，帮你隐式（implicitly）生成。

```
fun(1)            //隐式的生成 fun(int)
func(1.2)     //隐式的生成 fun(double)
func('a')     //隐式的生成 fun(char)
Shape*   ps = new   ch;
 fun(ps);     //隐式的生成 fun(Shape*)
```

模板函数:重点在函数，表示的是由一个模板生成而来的函数。

面显式（explicitly）或者隐式（implicitly）生成的 fun <int> 、fun <Shape*> ……都是模板函数。