# 模版

## 函数模版

**使用**：以template关键字开始，后面跟一个模版参数列表，列表中是一个或多个模版参数，不能为空。 **例子**：

template <typename T>  
Int compare(const T &v1, const T &v2)  
{  
 return 0;  
}

**实例化**：编译器用函数实参v1和v2推断模版实参，并把它绑定到T上。实例化后函数模版就变成了一个普通的函数，这个函数被称为模版的实例。 **类型模版参数**：模版参数列表中的参数可以是类型参数，需加关键词typename，类型参数在函数体内用于变量声明和类型转换。 **非类型模版参数**：不加关键词typename，而是使用具体类型修饰的参数，使用时必须是一个常量表达式，而不是局部变量或动态内存，由用户提供或者编译器推断。 **内联函数模版**：inline 放在返回类型之前。 **constexpr函数模版**：同上。 **函数模版的特点**：尽量减少对实参类型的要求。 **模版编译**：当实例化时模版才会编译生成代码，而不是定义时声称代码。模版的声明和定义放在同一个文件中，but why？因为要保证模板实例化时，模板的定义，包括类模板的成员的定义也必须是可见的。大多数编译错误是在实例化期间发现的。

## 类模板

**使用**：类似函数模板。 **例子**：

template <typename T>  
class Blob {  
/\*...\*/  
}

**实例化**：不能为类模板推断模板参数类型，必须提供显示模版实参列表。 **类模板的函数成员**：定义在类模板之外的成员函数必须以关键字typename开始，后接类模板参数列表。 **例子**：

template <typename T>  
RetType ClassName<T>::method\_name(parameters);

**实例化**：对于实例化了的类模板，其成员只有在使用时才被实例化。

## 成员模板

## 显示实例化

## 可变参数模板