|  |
| --- |
| https://www.cnblogs.com/yyxt/p/5090942.html |

模板类的 成员模版函数不能为 虚函数：

编译器在编译一个类的时候，需要确定该类的虚函数表大小，一般来说，一个类由N个虚函数，其虚函数表大小时N，按字节时 4\*N

如果允许一个成员模版函数未虚函数的话，因为我们可以为该成员模版函数实例化出很多不同的版本，也就是可以实例化出很多不同版本的虚函数。此时编译器为确定类的虚函数表大小，就必须确认我们一共未该成员模版函数实例化了多少个不同版本的虚函数。即编译器需要查找所有代码文件，才能直到到底由几个虚函数。

成员模版函数不能有默认参数：

这里所说默认参数有两种：第一种是 函数的默认参数 第二种是 函数模版的默认模版参数

成员模版函数和普通函数一样，可以有函数的默认参数（第一种），但是不能有函数模版的默认模版参数（第二种）

|  |
| --- |
| //第一种的情况：  template <typename T>  T sum(T\* b,T\* e,T result = T()) //这里的result就是函数的默认参数，允许  {  while(b!=e)  result += \*b++; return result;  };  //第二种的情况：  template <typename T = char>//这里的char为默认模板参数，不允许  T f(T a, T b)  {  return a + b;  }; |

不允许有函数模版的默认模版参数，因为函数模版的模版参数在调用的时候编译器根据实参的类型来确定，对于上述第二种情况，此时template中的T缺省值char就无意义

|  |
| --- |
| int result = f(1,2) //int f(int a,int b)，确定T为int  double result = f(1.0,2.0) //double f(double a,duoble b))，确定T为double |

注意：上面所说通过函数模版的实参演绎模版参数类型情况，指定缺省的模版实参没有意义和必要，但是某些情况下，有一些模版参数（比如作为函数的返回值类型）无法通过函数的实参演绎来确定，此时指定缺省的模版实参就很有必要，比如

|  |
| --- |
| // 返回类型RT无法通过实参演绎获得，这个时候指定RT的缺省实参就是非常必要的  template <typename RT, typename T1, typename T2>  inline RT add(T1 a, T2 b)  {  return a + b;  } |

1、模板类支持默认模板参数

2、模板函数或者成员模板函数不支持默认模板参数