例：程序C++11\_test14

// 可变参数类模板

template<typename T, typename... Args>

class Test

{

};

template<typename T>

*ostream*& Print(*ostream*& os, const T& t)

{

return os << t;

}

// 可变参数函数模板

template<typename T, typename... Args>

*ostream*& Print(*ostream*& os, const T& t, const Args&... rest)

{

os << t << ", ";

// 递归调用

return Print(os, rest...);

}

// Args代表模板参数

// rest代表通过Args的类型定义的函数形参

template<typename T, typename... Args>

void Foo(const T& t, const Args&... rest)

{

*cout* << "Args count " << sizeof...(Args) << *endl*;

*cout* << "rest count " << sizeof...(rest) << *endl*;

}

int main(int argc, char\* argv[])

{

Test<int, *string*> t1;

Test<int, int, float> t2;

Print<int, *string*>(*cout*, 1, "ok");

*cout* << *endl*;

Print<int, *string*>(*cout*, 1, "ok", "abc");

*cout* << *endl*;

Print<int, *string*, int>(*cout*, 1, "ok", 4);

*cout* << *endl*;

Print<int, *string*, int>(*cout*, 1, "ok", 4, 5);

*cout* << *endl*;

// 在实际使用中，可变模板参数包与函数参数包是对应的

// 但不一定要模板参数包的所有模板参数都罗列出来，

// 编译器会根据函数参数包进行自动推导模板参数，

// 但指明的模板参数必须有相应的函数参数来对应，

// 并且顺序必须一致

Foo<int, *string*, int, float>(1, "ab", 4, 6.5, "ab", 5);

return 0;

}

输出为：

1, ok

1, ok, abc

1, ok, 4

1, ok, 4, 5

Args count 5

rest count 5