- Example

Let's have a look at an example of dependent names in this lesson.

WE'LL COVER THE FOLLOWING

Example: Template Lookup

```
// templateLookup.cpp

#include <iostream>

void g(double) { std::cout << "g(double)\n"; }

template<class T>
struct S {
    void f() const {
        g(1);              // non-dependent
    }
};

void g(int) { std::cout << "g(int)\n"; }

int main(){
    g(1);                // calls g(int)

    S<int> s;
    s.f();               // calls g(double)
}
```

Explanation

If we access the defined functions `g` with `double` or `int` type object, they work fine. We have created the `struct` object `S` of `int` type in line 19. When we try to access the `g` function then it follows the same order and calls, the `g` with a `double` type parameter is defined first. The call to `s()` on line 17 calls with a `double` type parameter is defined first. The call to `g()` on line 17 calls

the `g(int)` version and the call to `g()` through the call to `f()` on line 20 calls `g(double)`.

In this chapter, we have learned about the details of templates. In the next chapter, we'll familiarize ourselves with the techniques used in templates. Let's start with Automatic Return type in the next lesson.