Extended CPP Notes

Nathan Warner

Northern Illinois
University

Computer Science
Northern Illinois University
April 17, 2024
United States

Contents

Templates

1.1 Class vs typename keyword

The choice between using **class** and **typename** in template declarations in C++ is largely a matter of style and historical context, as both keywords serve the same purpose

1.2 Handle friend functions

1.2.1 Friendship to a Non-Template Function

This is straightforward. You directly declare a non-template function as a friend inside your template class. This grants that specific function access to all instances of the template class, regardless of the type parameter.

```
template <typename T>
class MyClass {
    friend void someFunction(MyClass<T>&);
};
```

1.2.2 Friendship to a Template Function

More commonly, you want a template function to be a friend to a template class. This allows each instantiation of the function template to access the corresponding instantiation of the class template.To achieve this, you need to forward declare the function template and then declare it as a friend inside your class template. The tricky part is that the syntax for declaring a template function as a friend inside a template class can vary based on what you're trying to achieve:

How to forward declare:

```
    template<typename T>
    class myclass;

    template<typename T>
    void foo(myclass<T>&);


    template<typename T>
    class myclass {

    public:
        friend void foo <T>(const myclass<T>& obj);
    };

template <typename T>
void foo(myclass<T>& obj) {
    // Define
}
```

Different types:

- More general form used when you want the friendship to apply to all instantiations of the function template

```
template <typename T>
class MyClass {
    friend void someFunction<>(MyClass<T>&); // Specific instantiation
};
```

- All instantiations of the function template are friends:

```
template <typename T>
class MyClass {
    template <typename U>
    friend void someFunction(MyClass<U>&); // All instantiations
};
```

- This form ties the friendship to the specific template instantiation of both MyClass and someFunction using the same template argument T. The function template that takes the same template parameters:

```
template <typename T>
class MyClass {
    friend void someFunction<T>(MyClass<T>&); // Matched instantiation
};
```