【C++】Day82

Class	C++
Date	@April 23, 2022
Material	
Series Number	
Summary	

【Ch16】Templates and Generic Programming

16.3 Overloading and Templates

Function templates can be overloaded by other templates or by ordinary, nontemplate functions.

Functions with the same name must differ either as to the number or the types of their parameters.

Note: When two templates are viable, the compiler picks the one that matches better for the call.

The following two function templates are used to generate a string of bugs:

```
template <typename T> std::string bug_rep(const T& t) {
  std::ostringstream ret;
  ret << t; // Use T's output operator to print a representation of t
  return ret.str();
}
```

```
template <typename T> std::string bug_rep(T *p) {
  std::ostringstream ret;
  ret << "pointer: " << p;
  if(p)
    ret << bug_rep(*p);
```

```
  else
    ret << "Null pointer";
  return ret.str();
}
```

If we call the function as following:

```
std::string str("hi");
std::cout << bug_rep(&str) << std::endl;
```

The second version of `bug_rep` would be called as it is a more-matched version.

However, if we call the function as follows:

```
const std::string *strp = &str;
std::cout << bug_rep(strp) << std::endl;
```

The second version would be called. Both templates would be instantiated to `bug_rep(const std::string *)`. However, because the first one is more general while the second one is more specfic, the second version is instantiated.

Note: When there are seveal overloaded templates that provide an equally good match for a call, the most specialized version is preferred.

Nontemplate and Template Overloads

If we define a specified `bug_rep` for `string`:

```
std::string bug_rep(const string &s) {
  return '"' + s + '"';
}
```

In this case, both call of `bug_rep(const string &)` and `bug_rep<string>(const string &)` are valid. However, the string-specified version would be selected as it is more specific.