Explicit Specification with Function Templates

Lee S. Koh

Initially (namely for pre-standard C++), there was no way to explicitly tell the compiler which types to use as template arguments for instantiation of a function template. In those days, a function template such as the following (part of ***E.g. 4*** of lecture note ***311Templating***)

```
template <class T>
T GetValue()
{
  T value;
  cout << "Enter value: ";
  cin >> value;
  return value;
}
```

was illegal. Every template parameter (like `T` in the example above) was required to appear at least once in the parameter list of a function template; otherwise the compiler was not able to deduce the template arguments during the instantiation of the function template (*i.e.*, resolve template arguments through *implicit deduction*).

During C++ standardization, however, *explicit specification* of template arguments for function templates (using a syntax similar to the instantiation of **`class`** templates) was added. With this feature (and provided the compiler we use supports it), we can explicitly tell the compiler which types it must use for instantiation of a function template. The current GCC C++ compiler on the CS Dept Linux servers does support the feature, so the following program (slightly modified version of ***E.g. 4*** of lecture note ***311Templating***) will successfully compile and run.

```
#include <iostream>
#include <cstdlib>
using namespace std;

template <class T>
T GetValue()
{
  T value;
  cout << "Enter value: ";
  cin >> value;
  return value;
}

int main()
{
  int intValue;
  double dblValue;
  char charValue;

  intValue = GetValue<int>();
  cout << "Value entered is "
       << intValue << endl;
  dblValue = GetValue<double>();
  cout << "Value entered is "
       << dblValue << endl;
  charValue = GetValue<char>();
  cout << "Value entered is "
       << charValue << endl;

  return(EXIT_SUCCESS);
}
```

In this light, we shall treat the "***Failed Unification Errors***" ***Pitfall*** appearing in our textbook (***Page 282*** for the 3rd edition, ***Page 294*** for the 4th edition) as no longer applicable.