C++

Generic programming: introduction

Generic programming

- Generic programming deals with finding abstract representation of data structures and efficient algorithms
- A simplified view for generic programming is that of “programming with parameterized types”

Generic programming in C++

- C++ supports generic programming with the mechanism of **template**
- There are two kinds of template: **function template** and **class template**
- Template programming is a form of polymorphism called **parametric polymorphism**

Example

- Max() function with code reuse by copying:

```
int max (int a, int b)
{
 return a > b ? a : b;
}

string max (string a, string b)
{
 return a > b ? a : b;
}
```

Note the usage of overloaded operator “>“.
It is defined in the header <string>:

```
bool operator> (const string& c1,
                const string& c2);
```

Example rewritten using template

```
template <typename T>
T max (const T& a, const T& b)
{
  return a > b ? a : b;
}

int main(){
 int a=1, b=2;
 cout << max(a, b) << endl;
 string c="max", d="what";
 cout << max(c, d) << endl;
}
```

- Max() now can be used with any type as long as they are comparable by ">"

Standard Template Library

- The Standard Template Library (STL) is a foundation library providing basic types (list, vector, set, …) and basic algorithms (find, sort)
- It is part of the C++ standard library
- The programming paradigm underlying STL is called generic programming
- In the following slides and lessons we will study generic programming and learn how to use the STL
- Especially we will look at the following topics:
  - Templates
  - Containers
  - Iterators
  - Function objects

Standard Template Library

- The official documentation for the STL is available online at: http://www.sgi.com/tech/stl/
- Another online documentation that also include information on the STL is at (a Japanese translation of this site exists): http://www.cppreference.com/wiki/start