Lecture 8

C++ Features II

Namespace and STL

Prof. Hyeong-Seok Ko
Seoul National University
Graphics & Media Lab

Contents

- Namespace (3.1, 17.2.1)
- Standard Template Library (STL) (3.1-3.4, 9.2)

Namespace

- Every name used in the code must be unique within the current scope.
- **Namespace** provides a controllable mechanism for preventing name collisions.

Box_3D.h

```
class Box {
public:
   void print() {
      std::cout << "Box : ";
      std::cout << height << " " << width << " " << length << std::endl;
   }
   double height, width, length;
};
```

Box_2D.h

```
class Box {
public:
   void print() {
      std::cout << "### Box ###" << std::endl;
      std::cout << height << "\n" << width << "\n“ << std::endl;
   }
   double height, width;
};
```

main.cpp

```
#include "Box_3D.h"
#include "Box_2D.h"

void main() {
   Box box;
   box.height = 3; box.width = 5; box.length = 7;
   box.print();
}
```

Compilation Error !
Box redefinition

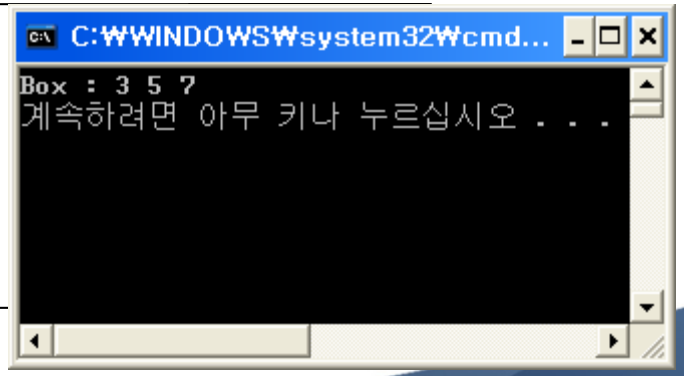
Box_3D.h

```
namespace three_dim {

class Box {
public:
   void print() {
      std::cout << "Box : ";
      std::cout << height << " " << width << " " << length << std::endl;
   }
   double height, width, length;
};
}
```

Box_2D.h

```
namespace two_dim {

class Box {
public:
   void print() {
      std::cout << "### Box ###" << std::endl;
      std::cout << height << "\n" << width << "\n“ << std::endl;
   }
   double height, width;
};
}
```

```
#include "Box_3D.h"
#include "Box_2D.h"

void main() {
   three_dim::Box box;
   box.height = 3; box.width = 5; box.length = 7;
   box.print();
}
```



Box_3D.h

```
namespace three_dim {

class Box {
public:
   void print() {
      std::cout << "Box : ";
      std::cout << height << " " << width << " " << length << std::endl;
   }
   double height, width, length;
};
}
```

Box_2D.h

```
namespace two_dim {

class Box {
public:
   void print() {
      std::cout << "### Box ###" << std::endl;
      std::cout << height << "\n" << width << "\n“ << std::endl;
   }
   double height, width;
};
}
```

main.cpp

```
#include "Box_3D.h"
#include "Box_2D.h"
using namespace three_dim;
void main() {
   Box box;
   box.height = 3; box.width = 5; box.length = 7;
   box.print();
}
```

Standard Template Library (STL)

- It brings huge innovation in C++ programming.

- It is **generic library** (i.e., based on **template**) which provides
  - Container (ex. vector, string)
  - Iterators
  - Algorithms
  - Functors

- It is very **general, efficient, and easy to use**.

Standard Template Library (STL)

- `string` Library
  - supports variable-length character strings
  - takes care of managing the memory associated with storing the characters
  - provides various useful operations
  - efficient

Standard Template Library (STL)

C style

```
#include <stdio.h>
#include <string.h>

void main() {
   const char* a = "Programming Methodology";
   const char* b = "is easy";

   char str[256] = "";
   strcat(str, a); strcat(str, " "); strcat(str, b);

   printf("%s\n",str);
}
```

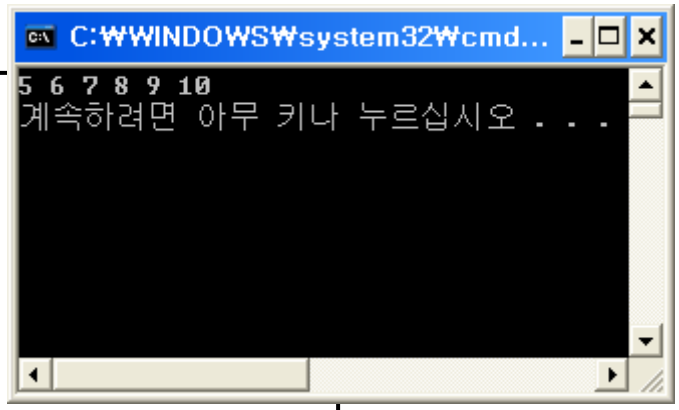
C++ style

```
#include <iostream>
#include <string>

using namespace std;

void main() {
   string a = "Programming Methodology", b = "is easy";
   cout << (a + " " + b) << endl;
}
```

Standard Template Library (STL)

- `vector<T>` Library
  - dynamic array
    - similar to array, but size is flexible
    - http://en.wikipedia.org/wiki/Dynamic_array
  - takes care of managing the memory associated with storing the elements
  - implemented as a class template

```
#include <iostream>
#include <vector>

void main() {
   std::vector<int> ivec;
   for(int i=5;i<=10;++i)
      ivec.push_back(i);

   for(std::vector<int>::size_type i=0;i!=ivec.size();++i)
      std::cout << ivec[i] << " ";
   std::cout << std::endl;
}
```



Standard Template Library (STL)

- Iterator
  - STL defines an **iterator** type for each of the standard containers.
    - such as `vector, list, deque`
  - **Iterators** are more general than subscripts.
  - similar to array pointer.

vector

```
#include <iostream>
#include <vector>

void main() {
   std::vector<int> ivec;
   for(int i=5;i<=10;++i)
      ivec.push_back(i);

   for(std::vector<int>::iterator it=ivec.begin();it!=ivec.end();++it)
      std::cout << *it << " ";
   std::cout << std::endl;
}
```

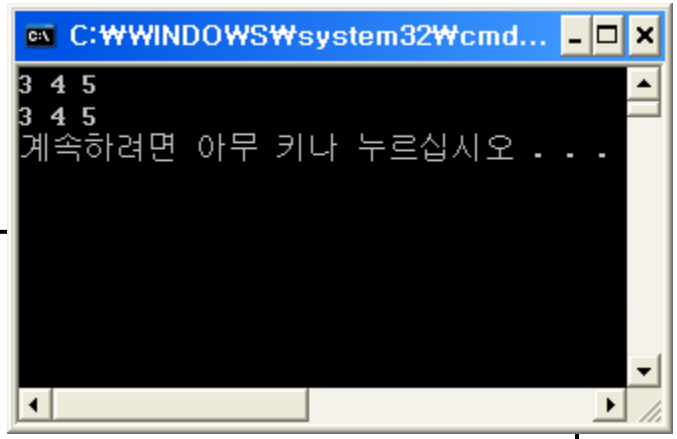
list

```
#include <iostream>
#include <list>

void main() {
   std::list<int> ilist;
   for(int i=5;i<=10;++i)
      ilist.push_back(i);

   for(std::list<int>::iterator it=ilist.begin();it!=ilist.end();++it)
      std::cout << *it << " ";
   std::cout << std::endl;
}
```

An iterator is a pointer.

Standard Template Library (STL)

- Algorithms
  - STL provides several generic algorithms.
    - such as `find`,`replace`,`sort`,...




```
#include <iostream>
#include <vector>
#include <algorithm>

void main() {
   int iarray[3] = {5,3,4};
   std::vector<int> ivec(3); ivec[0] = 5; ivec[1] = 3; ivec[2] = 4;

   std::sort(iarray, iarray + 3);
   std::sort(ivec.begin(), ivec.end());

   for(int i=0;i<3;++i)
      std::cout << iarray[i] << " ";
   std::cout << std::endl;

   for(std::vector<int>::iterator it=ivec.begin();it!=ivec.end();++it)
      std::cout << *it << " ";
   std::cout << std::endl;
}
```