Programming Seminar++: the Standard Template Library

Jun'ichi Ozaki

June 28, 2017

std::vector<*valuetype*>

std::vector is a variable-length array of *valuetype* (=int, double, std::complex, ...).
You don't have to worry about the array length at compile time.

1. Routines:

- void resize(size_t n) : Resizes the vector so that it contains n elements. (size_t = unsigned long)
- size_t size() : Returns the number of elements in the vector.
- *valuetype*& operator[] (size_t n) : Returns a reference to the element at position n.
- void push_back(*valuetype* x) : Add element at the end.
- ... and many other routines, please google them :)

2. Sample codes:

Note: compile using g++, not gcc.

```
#include<stdio.h>
#include<vector>
int main()
{
  std::vector<double> x;
  x.resize(4);
  x[0] = 10;
  x[1] = 20;
  x[2] = 30;
  x[3] = 40;
  for(int i = 0;i < x.size();i++)printf("%d,%lf\n",i,x[i]);
  return 0;
}
```

This is the same as the following:

```
#include<stdio.h>
#include<vector>
int main()
{
  std::vector<double> x;
  x.push_back(10);
  x.push_back(20);
  x.push_back(30);
  x.push_back(40);
  for(int i = 0;i < x.size();i++)printf("%d,%lf\n",i,x[i]);
  return 0;
}
```

std::map<*keytype, valuetype*>

std::map is an associative container that stores elements formed by a combination of a key value and a mapped value.
Briefly speaking, a generalization of std::vector to non-integer index.

1. Routines:

- size_t size() : Returns the number of elements in the container.
- *valuetype*& operator[] (*keytype* k) : If k matches the key of an element, the operator returns a reference to its mapped value. If k don't match any key in the container, the function creates a new element with that key and returns a reference to its element.
- size_t erase (const *keytype*& k) : Removes a element from the map container.
- ... and many other routines, please google them :)

2. Sample codes:

```
#include<stdio.h>
#include<map>
int main()
{
  std::map<double,int> x;
  x[0.0] = 1;
  x[0.1] = 2;
  x[0.2] = 3;
  x[0.3] = 4;
  printf("%d\n",x[0.0]);
  printf("%d\n",x[0.1]);
  printf("%d\n",x[0.2]);
  printf("%d\n",x[0.3]);
  return 0;
}
```

Advanced:

```
#include<stdio.h>
#include<string>
#include<map>
int main()
{
  std::map<std::string,std::string> firstnameof;
  firstnameof["Takayasu"]="Misako";
  firstnameof["Kanazawa"]="Kiyoshi";
  firstnameof["Tamura"]="Koutarou";
  firstnameof["Ozaki"]="Jun'ichi";
  printf("%s\n",firstnameof["Takayasu"].c_str());
  printf("%s\n",firstnameof["Kanazawa"].c_str());
  printf("%s\n",firstnameof["Tamura"].c_str());
  printf("%s\n",firstnameof["Ozaki"].c_str());
  return 0;
}
```

Iterator

Iterator is an object that enables a programmer to traverse a container. It can be used as a pointer.
For example, to output all the data in std::vector<double> x,

```
for(auto p = x.begin();p != x.end();p++)printf("%lf\n",*p);
```

in which *auto* is the variable type automatically deduced from the right hand side. That is the equivalent code to:

```
for(int i = 0;i < x.size();i++)printf("%lf\n",x[i]);
```

Iterator is used also for other contaners such as std::map, in which the integer index has no sense.
To output all the data (the pairs of key and value) in std::map<double,int> x;

```
for(auto p = x.begin();p != x.end();p++)printf("%lf,%d\n",p->first,p->second);
```

1. Routines:

- <*std::container*>.begin() : Returns an iterator referring to the first element in the container.
- <*std::container*>.end() : Returns an iterator referring to the past-the-end element.
- operator*, operator->, and operator++ : Enable a programmer to use an iterator as a pointer.
- ... and many other routines.

2. Sample codes:

Note: compile option -std=c++11 is necessary.

```
g++ -std=c++11 <filename>
```

```
#include<stdio.h>
#include<map>
int main()
{
  std::map<double,int> x;
  x[0.0] = 1;
  x[0.1] = 2;
  x[0.2] = 3;
  x[0.3] = 4;
  for(auto p = x.begin();p != x.end();p++)printf("%lf,%d\n",p->first,p->second);
  return 0;
}
```

Advanced:

```
#include<stdio.h>
#include<string>
#include<map>
int main()
{
  std::map<std::string,std::string> firstnameof;
  firstnameof["Takayasu"]="Misako";
  firstnameof["Kanazawa"]="Kiyoshi";
  firstnameof["Tamura"]="Koutarou";
  firstnameof["Ozaki"]="Jun'ichi";
  for(auto p = firstnameof.begin();p != firstnameof.end();p++)
    printf("Firstname of %s is %s\n",p->first.c_str(),p->second.c_str());
  return 0;
}
```