Biostatistics 615 - Statistical Computing

Lecture 7 STL and Boost library

Jian Kang

Oct 6, 2015

STL containers

What are STL containers?

- Data structure for convenient storage and access of multiple elements
- Behaviors are robust for both call-by-value and call-by-reference.
- http://www.cplusplus.com/reference/stl/ serves a great reference to look up.

Three popular STL containers

- `std::vector` - Array. $O(1)$ insert, $O(n)$ search.
- `std::set` - Container of unique elements. $O(\log n)$ insert/search.
- `std:map` - Container for key-value pairs. $O(\log n)$ insert/search.

Using std::vector : Initialization

```
int a1[4];                  // OK: initialize an array of size 4
std::vector<int> v1(4);     // OK: make a vector of size 4
```

the value of n MUST be assigned in defining an array

```
int* a3 = new int[n];       // OK: Allocate with new[] operator
delete [] a3;               // OK: Must be deleted after using
std::vector<int> v2(n);     // OK: For vector, n can be determined in run time
```

the value of n is not set in vector, the default is 0

```
v2.resize(2*n);             // OK: And you can resize a vector, but not an array

int a4[4] = {2,0,1,2};      // OK : Multiple element initialization is simple
std::vector<int> v3(4) = {2,0,1,2}; // ERROR : Not allowed for vector
std::vector<int> v4(4);     // OK : Each element has to be assigned separately
v4[0] = 2; v4[1] = 0; v4[2] = 1; v4[3] = 2; // access elements using []
```

uses array as pointer , from a4 to a4+4

```
std::vector<int> v5(a4, a4+4); // OK : Initialization using an array
int a6[4] = {-1,-1,-1,-1}; // OK : Need to write redundantly
std::vector<int> v6(4,-1); // OK : Allocate size 4 vector with value -1
```

Using std::vector : Insert, search and remove

Inserting elements

```
int a6[4];      // in an array, the size needs to be specified a priori
for(int i=0; i < 4; ++i) // and assign each value
   a6[i] = (i*i);
std::vector<int> v6; // with vector, initially define an empty vector
for(int i=0; i < 4; ++i)
   v6.push_back(i*i); // using push_back() function, size dynamically changes
```

Search for values

```
for(int i=0; i < 4; ++i)
  if ( a6[i] == 4 ) std::cout << "Found 4" << std::endl;
for(unsigned int i=0; i < v6.size(); ++i) // v6.size() is unsigned
  if ( v6[i] == 4 ) std::cout << "Found 4" << std::endl;
std::vector<int>::iterator it;  // use iterator
for(it = v6.begin(); it != v6.end(); ++it)
  if ( *it == 4 ) std::cout << "Found 4" << std::endl;
v6.clear();          // remove all contents
```

the last element of pointer V6

Using std::set for repetitive and fast search

lookup.cpp

```
#include <iostream>
#include <fstream> //Input/output stream class to operate on files.
#include <set>
#include <string>
int main(int argc, char** argv) {
  //Input stream class to operate on files.
  std::ifstream ifs(argv[1], std::ifstream::in );
  std::set<std::string> words;
  std::string word;
  while( ifs >> word ) words.insert(word);  // load file to set
  std::cout << "Type any word to lookup: ";
  while( std::cin >> word ) {
    if ( words.find(word) != words.end() )
      std::cout << "Found " << word << std::endl;
    else
      std::cout << "Could not find " << word << std::endl;
    std::cout << std::endl << "Type any word to lookup: ";
  }
  return 0;
}
```

Running lookup.cpp

```
jiankang@luigi:~$ ~jiankang/Public/bin/lookup ~jiankang/Public/data/words
Successfully loaded input file /afs/umich.edu/user/j/i/jiankang/Public/data/words
Type any word to lookup: Hello
Could not find Hello

Type any word to lookup: hello
Found hello

Type any word to lookup: world
Found world

Type any word to lookup: biostat615
Could not find biostat615

Type any word to lookup: (Ctrl + D to stop)
```

Using std::map as a dictionary

countSubstr.cpp

```
#include <iostream>
#include <fstream>
#include <map>
#include <vector>
#include <string>

using namespace std; // to avoid typing std:: repetitively

int main(int argc, char** argv) {
  if ( argc != 3 ) {
    cerr << "Usage: " << argv[0] << " [input_file.txt] [length]" << endl;
    return -1;
  }

  ifstream ifs(argv[1], ifstream::in );
  int length = atoi(argv[2]);
  map<string,int> mCnt;                  // (substr)->(count) map
  map<string, vector<string> > mWord; // (substr)->(list to all words) map
  string word;                           // variable to store a word
  string ss;                             // variable to store a substring
```

Using std::map as a dictionary

countSubstr.cpp (cont'd)

```
  while( ifs >> word ) {
    ss = word.substr(0,length);    // make substring
    ++mCnt[ss];                    // update count map (use map like array)
    mWord[ss].push_back(word);     // update word list map
  }
  cout << "Successfully loaded input file " << argv[1] << endl;
  cout << endl << "Type a substring of length " << length << ": ";
  while( cin >> ss ) {
    int cnt = mCnt[ss];
    cout << "There are " << cnt << " words starting with " << ss << endl;
    if ( cnt > 0 ) {                        // print each word in the list
      vector<string>& words = mWord[ss]; // reference type to avoid copy
      for(int i=0; i < cnt; ++i)
         cout << words[i] << endl;
    }
    cout << endl << "Type a substring of length " << length << ": ";
  }
  return 0;
}
```

mCnt contains the num of chars of the string ? ? ?

Running countSubstr.cpp

```
jiankang@luigi:~$ ~jiankang/Public/bin/countSubstr ~jiankang/Public/data/words 4
Successfully loaded input file /afs/umich.edu/user/j/i/jiankang/Public/data/words

Type a substring of length 4: bios
There are 4 words starting with bios
bioscience
biosphere
biostatistic
biosynthesize
```

Input/output handling

iomanip for output formatting

```
#include <iomanip>
// write floating point values in fixed point notation
std::cout << std::setiosflags(std::ios::fixed); // see also std::ios::scientific
std::cout << std::setprecision(4); // print up to 4 significant digits
std::cout << 3.14159;               // 3.142 will be printed
// managing alignment between output
std::cout << std::setw(10);        // set the minimum width of output to 10
std::cout << std::setiosflags(std::ios::right); // right align the output
std::cout << 100;                  // 100 printed after 7 blanks
```

ifstream for reading files

```
#include <fstream>
std::ifstream ifs("myfile.txt");
std::string s;
while( ifs >> s ) std::cout << "Read " << s << std::endl;
```

More STL examples

std::sort for sorting an array

```
#include <algorithm>
// ...
int myints[] = {32,71,12,45,26,80,53,33};
vector<int> myvector (myints, myints+8);
std::sort(myvector.begin(), myvector.begin()+4); // 12 32 45 71 26 80 53 33
std::sort(myvector.begin(), myvector.end());      // 12 26 32 33 45 53 71 80
```

std::next_permutation for enumerating permutation

```
#include <iostream>
#include <algorithm>
int main(int argc, char** argv) {
  int myints[] = {1,2,3};
  do {
    std::cout << myints[0] << " " << myints[1] << " " << myints[2] << std::endl;
  } while ( std::next_permutation (myints,myints+3) );
  return 0;
}
```

Using boost C++ libraries

Boost C++ library

- An extensive set of libraries for C++
- Supports many additional classes and functions beyond STL
- Useful for increasing productivity

Using boost C++ libraries

Boost C++ library

- An extensive set of libraries for C++
- Supports many additional classes and functions beyond STL
- Useful for increasing productivity

Examples of useful libraries

- Math/Statistical Distributions
- Tokenizer
- Random Numbers
- Graph Algorithms
- Regular expressions

Test Program for Boost Library : boostTest.cpp

```cpp
#include <iostream>
#include <boost/tokenizer.hpp>
#include <string>

using namespace std;
using namespace boost;

int main(int argc, char** argv) {
  // default delimiters are spaces and punctuations
  string s1 = "Hello, boost library";
  tokenizer<> tok1(s1);  // tokenize string delimited by whitespace
  for(tokenizer<>::iterator i=tok1.begin(); i != tok1.end() ; ++i) {
    cout << *i << endl;  // print each element
  }
  return 0;
}
```

标点

Using boost libraries

At scs.itd.umich.edu

```
$ g++ -o boostTest -I ~jiankang/Public/include/ boostTest.cpp
$ ./boostTest
Hello
boost
library
```

Adding the inclusion path will allow to include boost headers

Installing boost library in your system

- from *http://www.boost.org/*
- `tar xzvf boost_1_59_0.tar.gz`
- `mkdir ~/include` (under your home directory)
- `cp -R boost_1_59_0/boost ~/include/boost`
- `g++ -I ~/include -o boostTest boostTest.cpp` or modify inclusion path in your development environment

boost example 1 : Chi-squared test

```
#include <iostream>
#include <boost/math/distributions/chi_squared.hpp>
using namespace std;
using namespace boost::math;
int main(int argc, char** argv) {
  if ( argc != 5 ) {
    cerr << "Usage: chisqTest [a] [b] [c] [d]" << endl;
    return -1;
  }
  int a = atoi(argv[1]);  // read 2x2 table from command line arguments
  int b = atoi(argv[2]);
  int c = atoi(argv[3]);
  int d = atoi(argv[4]);
  // calculate chi-squared statistic and p-value
  double chisq = (double)(a*d-b*c)*(a*d-b*c)*(a+b+c+d)/(a+b)/(c+d)/(a+c)/(b+d);
  chi_squared chisqDist(1);
  cout << "Chi-square statistic = " << chisq << endl;
  cout << "p-value = " << cdf(complement(chisqDist, chisq)) << endl;
  return 0;
}
```

cdf(comlement(chisqDist, chisq)) Pr(x>=chisq) x~chisq(1)

cdf(chisqDist, chisq) Pr(x< chisq) x~chisq(1)

Running examples of chisqTest

```
user@host~:/$ ./chisqTest 2 7 8 2
Chi-square test statistic = 6.34272
p-value = 0.0117864

user@host~:/$ ./chisqTest 20 70 80 20
Chi-square test statistic = 63.4272
p-value = 1.66408e-15

user@host~:/$ ./chisqTest 200 700 800 200
Chi-square test statistic = 634.272
p-value = 5.88561e-140

user@host~:/$ ./chisqTest 2000 7000 8000 2000
Chi-square statistic = 6342.72
p-value = 0
```

boost Example 2 : Tokenizer

```
#include <iostream>
#include <boost/tokenizer.hpp>
#include <string>
using namespace std;
using namespace boost;
int main(int argc, char** argv) {
  // default delimiters are spaces and punctuations
  string s1 = "Hello, boost library";
  tokenizer<> tok1(s1);
  for(tokenizer<>::iterator i=tok1.begin(); i != tok1.end() ; ++i) {
    cout << *i << endl;
  }

  // advanced use : you can parse csv-like format
  string s2 = "Field 1,\"putting quotes around fields, allows commas\",Field 3";
  tokenizer<escaped_list_separator<char> > tok2(s2);
  for(tokenizer<escaped_list_separator<char> >::iterator i=tok2.begin();
       i != tok2.end(); ++i) {
    cout << *i << endl;
  }
  return 0;
}
```

only treat comma as seperator

only treat comma as seperator, ignore white space

A running example of `tokenizerTest`

```
user@host~:/$ ./tokenizerTest
Hello
boost
library
Field 1
putting quotes around fields, allows commas
Field 3
```

boost Example : Reading matrix from file

```
#ifndef __MATRIX_615_H
#define __MATRIX_615_H
#include <vector>
#include <iostream>
#include <fstream>
#include <cstdlib>
#include <boost/tokenizer.hpp>
#include <boost/lexical_cast.hpp>
template <class T>//for generic programming, independent of any particular type.
class Matrix615 {          通用的
public:
  std::vector< std::vector<T> > data;

  Matrix615(int nrow, int ncol, T val = 0) {
    data.resize(nrow); // make n rows
    for(int i=0; i < nrow; ++i)
      data[i].resize(ncol,val); // make n cols with default value val
  }
  int rowNums() { return (int)data.size(); }
  int colNums() { return ( data.size() == 0 ) ? 0 : (int)data[0].size(); }
  void readFromFile(const char* fileName);
};
```

if data.size != 0

Matrix615.h

```
void Matrix615::readFromFile(const char* fileName) {
  // open input file
  std::ifstream ifs(fileName);
  if ( ! ifs.is_open() ) {
    std::cerr << "Cannot open file " << fileName << std::endl;
    abort();
  }

  // set up the tokenizer
  std::string line;
  boost::char_separator<char> sep(" \t");
  // typedef is used to replace long type to a short alias
  typedef boost::tokenizer< boost::char_separator<char> > wsTokenizer;

  // clear the data first
  data.clear();
  int nr = 0, nc = 0;
```

```
  // read from file to fill the contents
  while( std::getline(ifs, line) ) {
    if ( line[0] == '#' ) continue; // skip meta-lines starting with #
    wsTokenizer t(line,sep);
    data.resize(nr+1);
    for(wsTokenizer::iterator i=t.begin(); i != t.end(); ++i) {
      data[nr].push_back(boost::lexical_cast<T>(i->c_str()));
      if ( nr == 0 ) ++nc;  // count # of columns at the first row
    }                                i=lexical_cast<int>(p); // 将字符串转化为整数
    if ( nc != (int)data[nr].size() ) {
      std::cerr << "The input file is not rectangle at line " << nr << std::endl;
      abort();
    }
    ++nr;
  }
}
```

Read data into a vector

```
template <class T>
void readFromFile(std::vector<T>& v, const char* fileName) {
  // open input file
  std::ifstream ifs(fileName);
  if ( ! ifs.is_open() ) {
    std::cerr << "Cannot open file " << fileName << std::endl;
    abort();
  }

  v.clear();
  std::string tok;
  while( ifs >> tok ) {
    v.push_back(boost::lexical_cast<T>(tok));
  }
}
```