STL[0]: Concepts, Pair and Vector

Dai@NeverLand / dailongao / EZ_dla

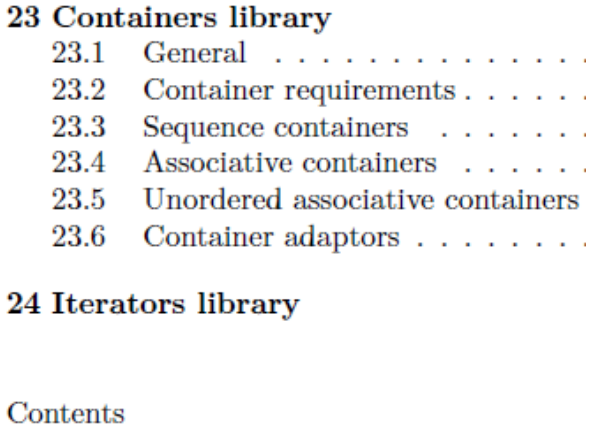
2013.7.17

Plan

- 7.17 – Concepts, Pair and Vector
- 7.18 – Map and Set
- 7.19 – Algorithms
- 7.20 – Queue and Priority Queue

What is STL?

- STL == Standard Template Library
- Part of the ISO Standard C++ Library
- Data Structure and algorithms for C++



23 Containers library

23.1 General . . . . . .
23.2 Container requirements . . . . . .
23.3 Sequence containers . . . . . .
23.4 Associative containers . . . . . .
23.5 Unordered associative containers
23.6 Container adaptors . . . . . .

24 Iterators library

Contents

What is STL?

Why should we use STL?

- Reduce development time.

uld we .?

- Reduc ient time.

↑ STL

← NO STL

Why should we use STL?

- Reduce development time.
- Code readability.
- Robustness.

Pair

- The simplest STL class
- Pairs of anything (int/int, int/char, etc…)

- Create it:
- **#include <utility>**
- **using namespace std;**
- pair<int, int> variable; // a variable of (int, int)

- What’s the meaning of <int, int>???

Template

- Suppose you need a list of X and a list of Y
- One is int, the other is double
- Similar code is hard to maintain!

- How can we reuse our code?
- Use template.
- More details in OOP

Template

- Only thing we should know is how to use it!
- It’s easy:
- **pair<type, type> x;**
- First is the name of the class
- And the template declaration (different class has different formula)

Pair

- Suppose we have a pair<int, int> x.
- How to use it?

Pair

- **Element access:**
- x.first -> access first element in pair
- x.second -> access second element
- Wait! It seems like a … struct?
- Right. You can use point(.) to use its functions and elements.

Pair

- How to set the value to a pair?
- x.first = a, x.second = b
- pair<int, int> x(a, b)
- x = make_pair(a, b)

Vector

- Top 3 data structure
- A simple and powerful structure
- A different way to write the code
- Forget linked list and malloc!

Vector

- How to create a vector?
- vector<int> a;
- Much easier than pair….?

Vector

- How vector looks like?
- Like c array…
- But auto-extending!
- We needn’t to set the size of vector; To instead, we use a different (but similar) way to insert the value.

Vector

- a is a empty vector.
- a.push_back(5);
- a.push_back(3);
- Now in a:
- a[0] = 5, a[1] = 3.
- Try printf(“%d\n”, a[0]);
- Vector supports random access.
- Push_back: O(1)

Vector

- What's more…
- a.size() – return how many elements in the vector
- a.empty() – return is the vector empty
- a.pop_back() – pop the last element in vector
- a.clear() – clear the vector
- a.erase(sth) – erase a element
- NOTICE: erase is O(N)!

Vector

- How can we visit the element?
- A simple way:
- for(i = 0; i < a.size(); ++i)
  - Gao(a[i]);
- But... What will happen if vector doesn't support random access one day?

Iterator

- STL provides a object for us to visit almost all containers in STL called **iterator**.
- You needn't know how it work, just use it!
- vector<int>::iterator it;
- for(it = **a.begin()**; it != **a.end()**; **++it**)
  - printf("%d\n", *it);
- Wait, is iterator a pointer? **NO**.

Iterator

- How to use iterator?
- http://www.cplusplus.com/reference/iterator/

Example

- The secret of Family CHEN…
- There are N (<= 1000000) members in Family CHEN.
- How to store the relationships between the members?

Homework

- HW5-1
  - Basic Vector Operations

Reference

- http://www.cplusplus.com