

Codebook

Jin, Lai, Lam from YZU

September 26, 2019

Contents

1 Environment

1.1	.vimrc	1
1.2	compile	1
1.3	copy	1
1.4	template	1

2 Container

2.1	vector	1
2.2	stack	1
2.3	queue	1
2.4	priority_queue	1
2.5	set	1
2.6	map	1
2.7	list	1

3 Method

3.1	algorithm	1
3.2	bitset	1
3.3	cmath	1
3.4	iomanip	1

4 Note

4.1	Preparing	1
4.2	Response Message	1

1 Environment

1.1 .vimrc

```
1 set number
2 set mouse=a
3 set shiftwidth=4
4 set tabstop=4
5 set autoindent
6 set cindent
7 filetype indent on
8 set cursorline
9 set t_Co=256
10 colorscheme slate
11 syntax on
```

1.2 compile

```
1 #shell script to compile program and execute
2 #!/bin/bash
3 g++ -Wall -O2 -std=c++14 -static -pipe -o $1
   $1.cpp && ./$1 < $1.in
```

1.3 copy

```
1 #copy template file
2 #!/bin/bash
3 for name in {A..M};
4 do
5     cp template.cpp $name.cpp
6 done
```

1.4 template

```
1 //template to code in C++
2 #include <bits/stdc++.h>
3 using namespace std;
4
5 int main(){
6
7     return 0;
8 }
```

2 Container

2.1 vector

```
1 //template
2 template <class value_type>
3 //init
```

```
4 vector <value_type>
5 //iterator
6 iterator begin()
7 iterator end()
8 //capacity
9 size_type size()
10 void reserve(size_type)
11 bool empty()
12 //access
13 reference operator[](size_type)
14 reference at(size_type)
15 //modifiers
16 void push_back(value_type)
17 void pop_back()
18 iterator insert(const_iterator, value_type)
19 iterator erase(const_iterator)
```

2.2 stack

```
1 //template
2 template <class value_type>
3 //init
4 stack <value_type>
5 //capacity
6 size_type size()
7 bool empty()
8 //access
9 reference top()
10 //modifiers
11 void push(value_type)
12 void pop()
```

2.3 queue

```
1 //template
2 template <class value_type>
3 //init
4 queue <value_type>
5 //capacity
6 size_type size()
7 bool empty()
8 //access
9 reference front()
10 reference back()
11 //modifiers
```

```

12 void push(value_type)
13 void pop()

```

2.4 priority_queue

```

1 //template
2 template <class value_type>
3 //init
4 priority_queue <value_type> //priority
   larger
5 priority_queue <value_type, vector<
   value_type>, greater<value_type> > //
   priority smaller
6 //capacity
7 size_type size()
8 bool empty()
9 //access
10 reference top()
11 //modifiers
12 void push(value_type)
13 void pop()

```

2.5 set

```

1 //template
2 template <class value_type>
3 //init
4 set <value_type>
5 //iterator
6 iterator begin()
7 iterator end()
8 //capacity
9 size_type size()
10 bool empty()
11 //operations
12 iterator find(value_type)
13 size_type count(value_type)
14 //modifiers
15 pair<iterator, bool> insert(value_type)
16 size_type erase(value_type)

```

2.6 map

```

1 //template
2 template <class key_type, class mapped_type>

```

```

3 typedef pair<key_type, mapped_type>
   value_type
4 //init
5 map <key_type, mapped_type>
6 //iterator
7 iterator begin()
8 iterator end()
9 //capacity
10 size_type size()
11 bool empty()
12 //access
13 mapped_type& operator[](key_type)
14 map<key_type, mapped_type>::iterator->first
   //key value
15 map<key_type, mapped_type>::iterator->second
   // mapped value
16 //operations
17 iterator find(key_type)
18 size_type count(key_type)
19 //modifiers
20 pair<iterator, bool> insert(pair<key_type,
   mapped_type>(key_type, mapped_type))
21 size_type erase(key_type)

```

2.7 list

```

1 //template
2 template <class value_type>
3 //init
4 list <value_type>
5 //iterator
6 iterator begin()
7 iterator end()
8 //capacity
9 size_type size()
10 void reserve(size_type)
11 bool empty()
12 //access
13 reference front(size_type)
14 reference back(size_type)
15 //operations
16 void remove(value_type)
17 //modifiers
18 void push_front(value_type)
19 void pop_front()

```

```

20 void push_back(value_type)
21 void pop_back()
22 iterator insert(const_iterator, value_type)
23 iterator erase(const_iterator)

```

3 Method

3.1 algorithm

```

1 template <class InputIterator, class
   value_type>
2 InputIterator find(InputIterator first,
   InputIterator last, value_type val)
3
4 template <class RandomAccessIterator>
5 void sort(RandomAccessIterator first,
   RandomAccessIterator last)
6
7 template <class RandomAccessIterator, class
   Compare>
8 void sort(RandomAccessIterator first,
   RandomAccessIterator last, Compare comp)
9
10 template <class ForwardIterator, class
   value_type>
11 bool binary_search(ForwardIterator first,
   ForwardIterator last, value_type val)

```

3.2 bitset

```

1 //template
2 template <class size_t>
3 //init
4 bitset <size_t>(unsigned long long)
5 bitset <size_t>(string)
6 bitset <size_t>(char *)
7 //access
8 bool operator[](size_t) const
9 reference operator[](size_t)
10 size_t count() // return the number of 1
11 size_t size() // size()-count() = return the
   number of 0
12 bool any()
13 bool none()
14 //operations

```

```
15 reference set() //all
16 reference set(size_t, bool) //single
17 reference reset() //all
18 reference reset(size_t) //single
19 string to_string()
20 string to_ulong()
21 string to_ullong()
```

```
1 //for DOMjudge
2 CORRECT
3 COMPILER-ERROR
4 TIMELIMIT
5 RUN-ERROR
6 WRONG-ANSWER
```

3.3 cmath

```
1 double cos(double)
2 double acos(double) //PI = acos(0.0)*2.0
3 double exp(double) //exponential
4 double log(double)
5 double log10(double)
6 double log2(double)
7 double pow(double, double)
8 double sqrt(double)
9 double cbrt(double)
10 double ceil(double) //round up
11 double floor(double) //round down
12 double round(double) //round
13 double abs(double)
```

3.4 iomanip

```
1 //template to code in C++
2 #include <bits/stdc++.h>
3 using namespace std;
4
5 int main(){
6
7     return 0;
8 }
```

4 Note

4.1 Preparing

```
1 check keyboard
2 check mouse
3 build environment(vim, g++, shell)
4 check judge system
5 check response message
```

4.2 Response Message