Codebook

Jin, Lai, Lam from YZU

September 26, 2019

Contents

1 Environment

1.1 .vimrc

```
set number
set mouse=a
set shiftwidth=4
set tabstop=4
set autoindent
set cindent
filetype indent on
set cursorline
set t_Co=256
colorscheme slate
syntax on
```

1.2 compile

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

1.3 copy

```
#copy template file
#!/bin/bash
for name in {A..M};
do
  cp template.cpp $name.cpp
done
```

1.4 template

```
//template to code in C++
#include <bits/stdc++.h>
using namespace std;

int main(){

  return 0;
}
```

2 Container

2.1 vector

```
//template
template <class value_type>
//init
vector <value_type>
//iterator
iterator begin()
iterator end()
//capacity
size_type size()
void reserve(size_type)
bool empty()
//access
reference operator[](size_type)
reference at(size_type)
//modifiers
void push_back(value_type)
void pop_back()
iterator insert(const_interator, value_type)
iterator erase(const_interator)
```

2.2 stack

```
//template
template <class value_type>
//init
stack <value_type>
//capacity
size_type size()
bool empty()
//access
reference top()
//modifiers
void push(value_type)
void pop()
```

2.3 queue

```
//template
template <class value_type>
//init
queue <value_type>
//capacity
size_type size()
bool empty()
//access
reference front()
reference back()
//modifiers
```

```
void push(value_type)
void pop()
```

2.4 priority_queue

```
//template
template <class value_type>
//init
priority_queue <value_type> //priority
    larger
priority_queue <value_type, vector<
    value_type>, greater<value_type> > //
    priority smaller
//capacity
size_type size()
bool empty()
//access
reference top()
//modifiers
void push(value_type)
void pop()
```

2.5 set

```
//template
template <class value_type>
//init
set <value_type>
//iterator
iterator begin()
iterator end()
//capacity
size_type size()
bool empty()
//oprations
iterator find(value_type)
size_type count(value_type)
//modifiers
pair<iterator, bool> insert(value_type)
size_type erase(value_type)
```

2.6 map

```
//template
template <class key_type, class mapped_type>
typedef pair<key_type, mapped_type>
    value_type
//init
map <key_type, mapped_type>
//iterator
iterator begin()
iterator end()
//capacity
size_type size()
bool empty()
//access
mapped_type& operator[](key_type)
map<key_type, mapped_type>::iterator->first
    //key value
map<key_type, mapped_type>::iterator->second
    // mapped value
//oprations
iterator find(key_type)
size_type count(key_type)
//modifiers
pair<iterator, bool> insert(pair<key_type,
    mapped_type>(key_type, mapped_type))
size_type erase(key_type)
```

2.7 list

```
//template
template <class value_type>
//init
list <value_type>
//iterator
iterator begin()
iterator end()
//capacity
size_type size()
void reserve(size_type)
bool empty()
//access
reference front(size_type)
reference back(size_type)
//operations
void remove(value_type)
//modifiers
void push_front(value_type)
void pop_front()
void push_back(value_type)
void pop_back()
iterator insert(const_interator, value_type)
iterator erase(const_interator)
```

3 Method

3.1 algorithm

```
template <class InputIterator, class
    value_type>
InputIterator find(InputIterator first,
    InputIterator last, value_type val)

template <class RandomAccessIterator>
void sort(RandomAccessIterator first,
    RandomAccessIterator last)

template <class RandomAccessIterator, class
    Compare>
void sort(RandomAccessIterator first,
    RandomAccessIterator last, Compare comp)

template <class ForwardIterator, class
    value_type>
bool binary_search(ForwardIterator first,
    ForwardIterator last, value_type val)
```

3.2 bitset

```
//template
template <class size_t>
//init
bitset <size_t>(unsigned long long)
bitset <size_t>(string)
bitset <size_t>(char *)
//access
bool operator[](size_t) const
reference operator[](size_t)
size_t count() // return the number of 1
size_t size() // size()-count() = return the
     number of 0
bool any()
bool none()
//operations
```

```
reference set() //all
reference set(size_t, bool) //single
reference reset() //all
reference reset(size_t) //single
string to_string()
string to_ulong()
string to_ullong()
```

```
//for DOMjudge
CORRECT
COMPILER-ERROR
TIMELIMIT
RUN-ERROR
WRONG-ANSWER
```

3.3 cmath

```
double cos(double)
double acos(double) //PI = acos(0.0)*2.0
double exp(double) //exponential
double log(double)
double log10(double)
double log2(double)
double pow(double, double)
double sqrt(double)
double cbrt(double)
double ceil(double) //round up
double floor(double) //round down
double round(double) //round
double abs(double)
```

3.4 iomanip

```
//template to code in C++
#include <bits/stdc++.h>
using namespace std;

int main(){

  return 0;
}
```

4 Note

4.1 Preparing

```
check keyboard
check mouse
build environment(vim, g++, shell)
check judge system
check response message
```

4.2 Response Message