competition

2017-10-13

How to start

```
#include <bits/stdc++.h>
#define _ ios_base::sync_with_stdio(0);cin.tie(0);
/* sync_with_stdio will disable printf scanf */
#define endl '\n'

using namespace std;
int main() { _
        return 0;
}
```

pair

- Header `<utility>`
- Non-member funcs

```
            /* make a pair without assign types */
pair<int , string> pr = make_pair(1 , "str");
            /* assignment opertor overloading */

/* comparaision between pairs */
 cout << (make_pair(1 , 2) < make_pair(1 , 3)) << endl; /* 1 */

/* C++ 11 */
swap(pr0 , pr1);
```

- Member vars

```
cout << pr.first << endl; /* 1 */
cout << pr.second << endl; /* 2 */
```

tuple (*C++11*)

- Header `<tuple>`
- Non-member funcs

```
/* make a tuple */
tuple<int , char , string> t(10 , 'c' , "string");
/* make tuple quickly */
auto tp = make_tuple(1 , 1);
/* get element */
cout << get<2>(t) << endl;
/* string */
```

vector

- Header `<vector>`
- How to construct

```
const int size = 5;
const int init_val = 1;
vector<int> vi(size , init_val);
/* 1 1 1 1 1 */

vector<vector<int> > vvi(size , vector<int>(size , init_val));
// 5*5 's 1

// or you can alloc by

vector<vector<int> > vi2d;
for(int i = 0 ; i < size ; i++)
   vi2d.push_back(vector(size , init_val));
```

- How to iterate

```
/* iterator 為 .begin() .end() .rbegin() 及 .rend() */
for(vector<vector<int> >::iterator vit = vvi.begin() ;
      vit != vvi.end() ; vit++ , cout << endl)
  for(vector<int>::iterator it = vit->begin() ; it != vit->end() ; it++)
      cout << *it << ' ';

for(int i = 0 ; i < vvi.size() ; i++ , cout << endl)
   for(int j = 0 ; j < vvi[i].size() ; j++)
      cout << vvi[i][j] << ' ' << endl;
```

- Member funcs
    - Capacity and Accessor
        - `.size` , `.empty` , `.front` , `.back`
        - `.resize`

```
vi.resize(3)
/* reduce to 3 elms */
vi.resize(5 , 4)
/* expand to 5 elms and stuff new elms to 4 */
vi.resize(10)
/* expand to 10 elms and stuff new with default val (0) */
```

    - Modifiers
        - `.push_back` , `.pop_back` , `.clear`
        - `.insert`

          `.insert(iter , val)`

          `.insert(iter , size_t , val)`

          `.insert(iter , iter_beg , iter_end)`
        - `.erase`

          `.earse(iter)`

          `.earse(iter_beg , iter_end)`
        - `.swap - va.swap(vb)`
- function overlads
    - relational , `swap`

ref: ***std vector C++ — deep or shallow copy***

stack

- Header `<stack>`
- Member functions
    - `.empty` , `.size` , `.top` , `.push` , `.pop`
- Non-member func
    - relational operator

queue

- Header `<queue>`
- Member functions
    - `.empty` , `.size` , `.front` , `.back` , `.push` , `.pop`
- Non-member func
    - relational operator

priority_queue

- Header `<queue>`
- Member functions
    - `.empty` , `.size` , `.front` , `.back` , `.push` , `.pop`
- Non-member func
    - relational operator

set

- Header `<set>`
- How to construct

```
int data[] = {1 , 2 , 3 , 4};
set<int> iset(data , data + 4);
/* iter.begin() and iter.end() */

set<int> yset(iset);

set<float , bool(*fp)(float)) fset;
```

- How to iterate

```
/* iterator 為 .begin() .end() .rbegin() 及 .rend() */




for(set<int>::iterator it
   = iset.begin() ; it != iset.end() ; it++)
   cout << *it << ' ';

cout << endl;
```

- Member functions
  - Capacity
    - `.empty , .size`
    - `.max_size`

      check if the set has enough size to store elements
  - Modifiers
    - `.insert`

      insert element
    - `.erase`

      `.erase(iter)`

      `.erase(val)`

      `.erase(iter_beg , iter_end)`
    - `.swap`

      swap two set

      `sa.swap(sb)`
    - `.clear`
  - Observers

    read it yourself
  - Operations
    - `.find`
    - `.count`
    - `.lower_bound`

      Return iterator to lower bound

```
{1 , 2 , 3}.lower_bound(1)
 ^
{1 , 3 , 5 , 7}.lower_bound(2)
     ^
```

    - `.upper_bound`

      Return iterator to upper bound

```
{1 , 2 , 3}.upper_bound(2)
          ^
{1 , 3 , 5 , 7}.upper_bound(2)
     ^
```

- `.equal_range`

  Get range of equal elements.

  Return pair of iters

```
{1 , 2 , 3}.equal_range(2)
          ^ second iter
      ^ first iter
{1 , 3 , 5 , 7}.equal_range(2)
      ^ first & second iter
```

map

- Header `<map>`
- How to construct

```
map<string , int> dict;
dict["hello"] = 0;
dict["world"] = 1;
map<string , int> ydict(dict.begin() , dict.end());
map<string , int> zdict(dict); /* copy constructor */
```

- How to iterate

```
/* iterator is a pointer to pair */



for(map<string , int>::iterator
   it = dict.begin() ; it != dict.end() ; it++)
   cout << it->first << ' ' << it->second << endl;
```

- Member functions
  - Capacity & Element access
    - `.empty` , `.size`
    - `.max_size`

      check if the map has enough size to store kpr
    - `[]`

- Modifiers
  - `.insert`

    let me explain it
  - `.erase`

    `.erase(iter)`

    `.erase(key)`

    `.erase(iter_beg , iter_end)`
  - `.swap`
  - `.clear`
- Observers

  read it yourself
- Operations
  - `.find`

    get the iterator
  - `.count`

    check the kpr exist
  - `.lower_bound`

    Return iterator to lower bound
  - `.upper_bound`

    Return iterator to upper bound
  - `.equal_range`

    Get range of equal elements.

    Return pair of iters

algorithm

- Header `<algorithm>`
- Non-modifying
  - `iter find(iter_beg , iter_end , val)`
  - `iter find_if(iter_beg , iter_end , pred_fp)`
  - `iter search(iter_beg , iter_end , seq_beg , seq_end)`

    Search range for subsequence
  - `iter find_first_of(iter_beg , iter_end , range_beg , range_end)`

    Find element from set in range
  - `iter find_end(iter_beg , iter_end , sub_beg , sub_end)`

    Find last subsequence in range
  - `int count(iter_beg , iter_end , val)`

    (int -> std::ptrdiff_t)

        - `int count_if(iter_beg , iter_end , pred_fp)`
        - `bool equal`

          `equal(iter0_beg , iter0_end , iter1_beg)`

          `equal(iter0_beg , iter0_end , iter1_beg , pred_fp)`
    - Modifying
        - `copy(first_iter , last_iter , result_iter)`
        - `swap`
        - `reverse`
    - Sorting
        - `sort`

          `sort(first , last)`

          `sort(first , last , comp_fp)`
    - Heap
        - `push_heap` , `pop_heap` , `make_heap` , `sort_heap`

          `is_heap` (***c++11***)
    - Min/Max
        - `min`
        - `max`
        - `min_element`
        - `max_element`
    - Binery_search
        - do it yourself maybe
    - Other
        - next_permutation(first , end);
        - prev_permutation(first , end);

overload operator

```
1   inline bool operator==(const X& lhs, const X& rhs){ /* DIY */ }
2
3   inline bool operator!=(const X& lhs, const X& rhs){ return !(lhs == rhs); }
4
5
6   inline bool operator< (const X& lhs, const X& rhs){ /* DIY */ }
7
8   inline bool operator> (const X& lhs, const X& rhs){ return rhs < lhs; }
9   inline bool operator<=(const X& lhs, const X& rhs){ return !(lhs > rhs); }
10  inline bool operator>=(const X& lhs, const X& rhs){ return !(lhs < rhs); }
```

vim script

```
source /etc/vimrc
set nu "num line
set tabstop=4
set shiftwidth=4
set cindent
set smarttab
set expandtab "set tab to space
set autoindent
syntax on
"hi comment ctermfg=cyan
hi comment ctermfg=blue
"prefer ^
"let comment color be blue. instead of dark blue
"super TAB
function InsertTabWrapper()
    let col = col('.') - 1
    if !col || getline('.')[col - 1] !~ '\k'
        return "\<tab>"
    else
        return "\<c-p>"
    endif
endfunction
inoremap <TAB> <C-R>=InsertTabWrapper()<CR>
"my rc
imap jk <ESC>
let mapleader = "\<Space>"
nmap <Leader>x :x<CR>
nmap <Leader>w :w<CR>
nmap <Leader>q :q<CR>
nmap <Leader>n :n<CR>
nmap <Leader>N :N<CR>
nmap <Leader>jq :q!<CR>
nmap <Leader>/ :noh<CR>
nmap <Leader>; :
nmap <Leader>= mcHmhLmlgg=G`h`l`c
noremap H ^
noremap L $
set clipboard=unnamed
```


‹ Cpp STL Hack