evandrix's blog

Topcoder STL article summary

By **evandrix**, 3 years ago,

Vector: []++

Declaration

- empty vector: `vector<int> v;`
- array of *10* `vector<int>` s: `vector<int> v[10];`
- vector with *10* *int* elements: `vector<int> v(10);`

Initialisation

```
vector<int> v1;
// …
vector<int> v2 = v1;   // make a copy?
vector<int> v3(v1);   // identical to v2's init
// …
vector<int> v4(1000);   // specific size: 1000 0's
// …
vector<int> v5(20, "Unknown");   // initial value
// …
vector<int> v6(v1.begin(), v1.end()); // [begin,end)
int data[] = { 1,2,3,…,8,9 };
vector<int> v7(data, data+(sizeof(data)/sizeof(data[0]))); //
data+length=.end()
vector<int> v8(v1.begin(), v1.begin()+(v1.size()/2)); // 1st half of v1
// 1st half of v1, ordered back-to-front
vector<int> v9(v1.rbegin()+(v.size()/2), v.rend());
```

size()

- unsigned: macro `#define sz(C) return (int) C.size()`
- use `empty()` instead of `size() == 0` , because of runtime complexity

push_back(elem)

resize(new_size)

- smaller: delete elements
- larger: pad with zeros (obj: NULL)
- `resize(), push_back()` : elements are added AFTER newly allocated size, not INTO it, eg.

```
vector<int> v(20);
…
v.resize(25);
…
for (int i=0; i<5; i++)
{
   v.push_back(…); // writes to v[25..30], not v[20..25]!
}
```

clear()

→ **Pay attention**

Before contest
Codeforces Round #315 (Div. 1)
43:05:59

Before contest
Codeforces Round #315 (Div. 2)
43:05:59

Like 98 people like this. Be the first of your friends.

→ **A7maSa3d**

Contribution: **0**

- Settings
- Blog
- Teams
- Submissions
- Talks
- Contests

A7maSa3d

→ **Top rated**

#	User	Rating
1	tourist	3503
2	Petr	3103
3	vepifanov	2963
4	qwerty787788	2947
5	rng_58	2941
6	TooSimple	2903
7	niyaznigmatul	2853
7	WJMZBMR	2853
9	scott_wu	2824
10	yeputons	2783

Countries | Cities | Organizations View all →

→ **Top contributors**

#	User	Contrib.
1	Endagorion	157
1	PrinceOfPersia	157
3	Petr	156
3	Egor	156
5	Swistakk	147
6	I_love_Hoang_Yen	144
7	I_love_Tanya_Romanova	142
8	Rubanenko	140
8	dreamoon	140
10	Enchom	139

View all →

→ **Find user**

- makes vector contain 0 elements, not zero out elements

Multidimensional arrays: `vector< vector<> >`

```
int N,M;
// matrix of size N*M filled with -1's
vector< vector<int> > Matrix(N, vector<int>(M, -1));
```

insert(idx)

- add data to somewhere other than the end (=push_back())

```
// insert value 42 after first element
// other elements 2nd-last are shifted downward
v.insert(1, 42);

// interval form
// shift elements 2nd-last by appropriate offset
// copy contents of v2 into v
v.insert(1, all(v2));
```

erase(idx)

- single element: `erase(iterator);`
- interval form: `erase(begin iterator, end iterator);`

function(vector)

```
void some_function(vector<int> v) { // NO: makes a copy
   // …
}

void some_function(const vector<int>& v) { // YES: unmodifiable ref
   // …
}
// … or …
void modify_vector(vector<int>& v) {
   v[0]++;
}
```

reverse()

```
int data[10] = { 1,3,5,7,9,11,13,15,17,19 };
reverse(data+2, data+6);
// range {5,7,9,11} -> {11,9,7,5}
```

find()

- search for appropriate elements in an interval

1. element found: pointer to instance of first occurrence
   `int index = find(v.begin(), v.end(), 49) - v.begin(); < v.size()`
2. otherwise: end of interval `find(v.begin(), v.end(), 49) == v.end();`

min/max_element()

- returns iterator to respective element
  `int index = min/max_element(v.begin(), v.end()) - v.begin();`
  `int value = *min/max_element(v.begin(), v.end());`

`#define all(c) c.begin(),c.end()`

sort()

- ascending: `sort(v.begin(), v.end());`
- descending: `sort(v.rbegin(), v.rend());`

Pair

```
pair<string, pair<int, int> > P;
```

Handle:

Find

→ Recent actions

Claveo → Distance matrix — tree problem

agaurav77 → Counting Problem (DP)

Tribbiani → Team training for ICPC

natsukagami → Inverse sum of coprime pairs

ashish1610 → Procon 2015 Editorials

ironman7453 → Maximum Longest Increasing Subsequence

Zlobober → VK Cup 2015: first day

JUST_Yousef → Graph problem, TLE and RTE

love_you_ma → Reducing Run Time

P1kachu → Editorial Codeforces Round #Pi

ashish1610 → Invitation to Procon 2015

PrinceOfPersia → The 1st Hunger Games

P1kachu → Codeforces Round #Pi (Div. 2)

SinchiCodo → Java Vs C++ !!

indy256 → Dynamic Programming Optimizations

vsanjay_nitdgp → get the idea

dalex → Разбор задач Codeforces Round #301 (Div. 2)

roger-that → Easy Problem

Noufaln26 → Minimum number of operations

qmaruf → Why output is different in my PC and in online judge?

star569 → IOI website down?

Noufaln26 → I got output in IDE but not in codevita server.why?????????

BWC → Invitation to BWC Round #1

Alioth → Is this a bug?

reality → Sum of distances

Detailed →

```
string s = P.first;
int x = P.second.first;
int y = P.second.second;
```

- advantage: builtin comparison operators, ie. lexicographical
- `pair[]` / `vector<pair<> >` sorted by STL internally

```
// sort array of integer points so they form a polygon
vector< pair<double, pair<int,int> > points; // { polar angle, (x,y) }
```

- also used in associative containers, eg. `map`

Iterator: generalised container data access

performs following operations only

- take value (unary *)
- comparison (</!=)
- increment/decrement (++/--)
- *add immediate, ie.* `it+=20` *: shift 20 elements forward*
- *get distance b/w iterators, ie.* `int n=it2-it1;`

types: *normal* vs. *random* access

- normal can be compared with ==/!=, allows ++/--
- but not subtracted, nor added: cannot implement in O(1) for all container types

`<algorithm>` : `.begin()` & `.end()` point to *first invalid* object, NOT last element

- `c.begin() == c.end() iff c.empty()`
- `c.end() - c.begin() = c.size()`

```
void reverse_array(int *A, int N) {
   int first = 0, last = N-1;
   while (first < last)
   {
      swap(A[first], A[last]); // cannot index all container types in
O(1), eg. doubly linked list
      first++;
      last--;
   }
}

void reverse_array(int *A, int N) { // iterator-style
   int *first = A, *last = N-1;
   while (first < last)
   {
      swap(*first, *last);
      first++;
      last--;
   }
}

// does not use '<'
template<typename T> void reverse_array(T *first, T *last) {
   if (first != last)
   {
      while (true)
      {
         swap(*first, *last);
         first++;
         if (first == last)
         {
            break;
         }
         last--;
         if (first == last)
         {
            break;
         }
      }
   }
}

/** identical to std::reverse(T begin, T end) in <algorithm> */
// .end() = *last + 1;
// STL-compliant
template<typename T> void reverse_array(T *begin, T *end) {
   if (begin != end)
   {
      end--;
      if (begin != end)
      {
         while (true)
         {
            swap(*begin, *end);
            begin++;
            if (begin == end)
            {
               break;
            }
            end--;
            if (begin == end)
            {
               break;
            }
         }
      }
   }
}
```

- types: ::iterator, const_iterator, reverse_iterator, const_reverse_iterator

```
// '!=' instead of '<'
for(vector<int>::iterator it=v.begin(); it!=v.end(); it++)
{
   *it++;
}

#define tr(container, it) \
   for(typeof(container.begin()) it=container.begin();
it!=container.end(); it++)
```

String: vs. `vector<char>`

- *string manipulation functions & memory management policy*

substring

- `s.substr(start_index[, end_index]);`, eg. `(0,s.length()-1)`, `(1)`
- `s.length()-1` on empty string `.empty()` : `s.length()` is unsigned; `unsigned(0)-1` ?

- split string function?

Set

- init: `int data[5]={5,1,4,2,3}; set<int> S(data, data+5);`
- no duplicates
- order-independent
- check membership
- comparable elements
- add,remove,check in O(log N); count in O(1)
- **insert(elem)**
- **erase(elem)** interval form:

`s.erase(s.find(10), s.find(100)); // erase [10,100)`
- **size()**
- iterator traversal
- `<algorithm>` find() in O(N); `<set>` find() in O(log N)

```
#define present(container, element) (container.find(element) !=
container.end())
#define cpresent(container, element) (find(all(container),element) !=
container.end())
```

- remove duplicates from vector
- `sort(all(v)); v.resize(unique(all(v)) – v.begin());`
- v is now de-duplicated and sorted ascending

Map

```
map<string, int> M;
M["A"] = 1;
M.find("A") != M.end();
M.erase("A");
```

- traversing iterator = pair<key, value>, ie. `it->second;`
- operator[] vs. map::find()
- find() preserves map contents
- [] creates non-existent elements
- **use find() in loops**

Set & Map are stored as R-B trees

++/-- *defined on Set & Map*

`<algorithm>`

- `min(a,b)`
- `max(a,b)`
- `swap(a,b)`
- `sort(begin,end)`
- `find(begin,end,element)` : set/map have their own defined
- `count(begin,end,element)` : set/map have their own defined
- `prev/next_permutation(begin,end)` : return false if exhausted; CONTAINER MUST BE SORTED FIRST!

`<sstream>`

```
const string& s;
istringstream is(s);
int tmp;
is >> tmp;

int tmp;
ostringstream os;
os << tmp;
string s = os.str();
```

Macros

```
typedef vector<int> vi;
typedef vector<vi> vii;
typedef pair<int, int> ii;
#define sz(a) int((a).size())
#define pb push_back
#define all(c) (c).begin(),(c).end()
#define tr(c,l) for (typeof((c).begin()) i=(c).begin(); i!=(c).end();
i++)
#define present(c,x) ((c).find(x) != (c).end())
#define cpresent(c,x) (find(all(c),x) != (c).end())
```

topcoder, **algorithm**, **summary**