

C++ notes for CoP

Daniel M. German

January 11, 2017

Contents

1	General	2
1.1	Use typedefs	2
2	Vectors	2
2.1	Multi-dimensional	2
3	Collections	2
3.1	Reverse a collection	2
3.2	Back inserter	2
3.3	map (as in functional map)	2
3.4	for_each	2
4	Priority queue	2
4.1	Descending	2
4.2	ascending	3
5	queue/deque	3
5.1	operations	4
6	stack	4
7	Maps	4
7.1	order/unordered	4
7.2	constructors	4
7.3	Add elements	4
7.4	traversal	4
7.5	does the map have it? so we can access it	5
7.6	find: with an iterator to it	5
8	Input	5
8.1	Using scanf	5
8.2	split string by delimiter	5
9	Union find	6

1 General

1.1 Use typedefs

```
std::unordered_map<std::string, std::string> stringmap;
```

2 Vectors

2.1 Multi-dimensional

And allocate their size automatically

```
vector<vector<int> > A(dimension1, vector<int>(dimension2));  
//or  
vector<vector<int> > A(dimension1, vector<int>(dimension2), initValue);
```

3 Collections

3.1 Reverse a collection

```
std::reverse(v.begin(), v.end());
```

3.2 Back inserter

```
std::back_inserter_iterator< std::vector<int> > back_it (v2);
```

3.3 map (as in functional map)

Using a backinserter, probably the best way to do it

if it does not create the vector, then why bother? Perhaps just use for_each instead

```
std::back_inserter_iterator< std::vector<int> > back_it (v2);  
std::transform(v.begin(), v.end(), back_it, [](int x)->int {  
    return -x;  
});
```

3.4 for_each

```
std::for_each(v.begin(), v.end(), [](int x) {  
    std::cout << x ;  
});
```

4 Priority queue

4.1 Descending

```
#include <iostream>  
#include <queue>  
#include <vector>  
#include <algorithm>
```

```

int main()
{
    std::priority_queue<int, std::vector<int>, std::greater<int>> q;
    for(int n : {1, 8, 5, 6, 3, 4, 0, 9, 7, 2})
        q.push(n);
    for(int i = 0; i < 10; i++) {
        std::cout << q.top() << std::endl;
        q.pop();
    }
    return 0;
}

```

4.2 ascending

```

#include <iostream>
#include <queue>
#include <vector>
#include <algorithm>

int main()
{
    std::priority_queue<int, std::vector<int>, std::greater<int>> q;
    for(int n : {1, 8, 5, 6, 3, 4, 0, 9, 7, 2})
        q.push(n);
    for(int i = 0; i < 10; i++) {
        std::cout << q.top() << std::endl;
        q.pop();
    }
    return 0;
}

```

5 queue/deque

```

#include <iostream>
#include <deque>

int main()
{
    // Create a deque containing integers
    std::deque<int> d = {7, 5, 16, 8};

    // Add an integer to the beginning and end of the deque
    d.push_front(13);
    d.push_back(25);

    // Iterate and print values of deque
    for(int n : d) {
        std::cout << n << '\n';
    }
}

```

5.1 operations

push_back	
push_front	
pop_back	
pop_front	
top	
pop	
front	inspect front
back	inspect back

6 stack

remember, pop pops, but top inspects

```
std::stack<int>    s;

s.push( 2 );
s.push( 6 );
s.push( 51 );

s.pop();
s.top();
```

7 Maps

7.1 order/unordered

```
std::map<char,int> mymap;
std::unordered_map<char,int> mymap;
```

7.2 constructors

Add "pairs" in the constructor"

```
stringmap second ( {{"apple","red"},"lemon","yellow"}} );
```

7.3 Add elements

```
second["apple"] = "red";
```

7.4 traversal

each element is a pair: with first and second

```
for (auto& x: sixth)
    std::cout << " " << x.first << ":" << x.second;
```

7.5 does the map have it? so we can access it

```
if (mymap.count(x)>0)
    std::cout << "mymap has " << x << std::endl;
else
    std::cout << "mymap has no " << x << std::endl;
```

7.6 find: with an iterator to it

- gets specific element
- just use count instead, unless you want to erase it

```
std::map<char,int> mymap;
std::map<char,int>::iterator it;

it = mymap.find('b');
if (it != mymap.end())
    mymap.erase (it);
```

8 Input

8.1 Using scanf

```
int j = scanf("%d %d\n", &n,&q);
assert(j == 3)
int j = scanf("%d %d\n", &n,&q);
assert(j == 2);
to strings by a delimiter

#+BEGIN_SRC C++
string st;
vector<string> tokens;
while (getline(std::cin, st, delim)) {
    tokens.push_back(item);
}
```

p** input into an integer

std::stoi	int
std::stol	long
std::stoll	long long

```
string st;
vector<int> tokens;
while (getline(std::cin, st, delim)) {
    tokens.push_back(std::stoi(st));
}
```

8.2 split string by delimiter

```
#+END_SRC
```

9 Union find

```
#include <vector>
#include <assert.h>

std::vector<int> id {};
std::vector<int> rank {};

void init_union_find(int n)
{
    id.resize(n);
    rank.resize(n, 0);
    for(int i=0; i<n; i++) {
        id.at(i) = i;
    }
}

int findSet(int i)
{
    if (id.at(i) == i)
        return i;
    else {
        id.at(i) = findSet(id.at(i));
        return id.at(i);
    }
}

bool isSameSet(int p, int q)
{
    return (findSet(p) == findSet(q));
}

void unionSet(int i, int j)
{
    if (!isSameSet(i, j)) {
        int x = findSet(i);
        int y = findSet(j);
        if (rank.at(x) > rank.at(y))
            id.at(y) = x;
        else {
            id.at(x) = y;
            if (rank.at(x) == rank.at(y))
                rank.at(y)++;
        }
    }
}
```