WRITE A POST

Satwik Kansal FOLLOW
Python | Data Science | Decentralized Applications

STL Cheatsheet for Competitive Programming

Published Aug 11, 2017

I've prepared a C++ cheatsheet that covers most of the data structures and STL functionalities you'll ever use for programming competitions or interview tests for tech companies. Think about the rest of this post as a quick refresher to the most commonly used parts of STL. Without further ado, here it is!

```
/*
This a header file that includes every standard library.
You can use it to save time.

NOTE: This header file may only be recognized by the GCC compiler.

*/
#include <bits/stdc++.h>

/*
//Use this if the above header file doesn't work.

#include <iostream>

#include <algorithm>
```

7 2

```
#include <vector>
*/

using namespace std;


// These will be used later
struct Person {
  string name;
  int age;
} p1, p2, p3;

struct is_older {
  bool operator()(struct Person p1, struct Person p2) {
        return p1.age > p2.age;
    }
};

bool compare_names(struct Person p1, struct Person p2) {
        return p1.name < p2.name;
}

bool way_to_sort(int i, int j) { return i > j; }


int main() {

  /*
  ========
   STACK
  ========
  */

  stack <string> distros; //Create a stack of strings.

  distros.push("Ubuntu");  //Pushes elements into the stack.
  distros.push("Mint");

  cout << "Number of distros in the stack are " << distros.size() << endl
  cout << "Distro on the top is " << distros.top() << endl;
```

```
========
 VECTOR
========
*/

vector <int> numbers;

if (numbers.empty()){ //check if the vector is empty?
  cout << "The vector is empty :(" << endl;
}

for(int i=0; i<100; i+=10){ //Add some values to the vector
  numbers.push_back(i);
}

cout << "Size of the vector is " << numbers.size() << endl;

// iterating over the vector, declaring the iterator
vector <int>::iterator it;

cout << "The vector contains: ";
for (it=numbers.begin(); it!=numbers.end(); it++) {
  cout << "  " << *it;
}

// getting value at a particular position
int position = 5;
cout<<"\nVector at position "<<position<<" contains "<<numbers.at(posi

// deleting an element at a position
numbers.erase(numbers.begin() + position);
cout<<"Vector at position "<<position<<" contains "<<numbers.at(positi

// deleting a range of elements, first two elements
// NOTE: You may expect elements at 0, 1, and 2 to be deleted
// but index 2 is not inclusive.
numbers.erase(numbers.begin(), numbers.begin()+2);
cout << "The vector contains: ";
for (it=numbers.begin(); it!=numbers.end(); it++) {
```

```
if (numbers.empty()){
  cout << "\nThe vector is now empty again :(";
}


/*
=========
 HASHMAP
=========
*/

// Declaration <key type, value type>
map <string, string> companies;

companies["Google"] = "Larry Page";
companies["Facebook"] = "Mark Zuckerberg";

// insertion can also be done as
companies.insert(pair<string, string> ("Xarvis Tech", "xarvis"));
// or
companies.insert(map<string,string>::value_type("Quora", "Adam D'Angel
// or even
companies.insert(make_pair(string("Uber"), string("Travis Kalanick")))

// Iterating the map
map<string, string>::iterator itz;
cout << "\n\nCompanies and founders" << endl;
for (itz=companies.begin(); itz!=companies.end(); itz++){
  cout << "Company: " << (*itz).first << "\t Founder: " << itz->second
}

itz = companies.find("Google");
cout << itz->second;

/*
==============
 LINKED LISTS
==============
*/
list<int> mylist;
```

```
  // 10 20 30 40 50 60 70 80 90
  it1 = it2 = mylist.begin(); // ^^
  advance (it2,6);            // ^                 ^
  ++it1;                      //    ^              ^

  it1 = mylist.erase (it1);   // 10 30 40 50 60 70 80 90
  //    ^           ^

  it2 = mylist.erase (it2);   // 10 30 40 50 60 80 90
  //    ^           ^

  ++it1;                      //       ^        ^
  --it2;                      //       ^     ^

  mylist.erase (it1,it2);     // 10 30 60 80 90

  cout << "\nmylist contains:";
  for (itx=mylist.begin(); itx!=mylist.end(); ++itx)
      cout << ' ' << *itx;
  cout << '\n';

  // NOTE: it1 still points to 40, and 60 is not deleted
  cout << endl << *it1 << "\t" << *it2 <<endl;

  // This will print an unexpected value
  it1++;
  cout << *it1;

  cout << "\nmylist now contains:";
  for (it1=mylist.begin(); it1!=mylist.end(); ++it1)
      cout << ' ' << *it1;
  cout << '\n';

  /*
=======
 HEAPS
=======
*/
```

```
pq.push(5);
pq.push(1);
pq.push(10);
pq.push(30);
pq.push(20);

// Extracting items from the heap
while (!pq.empty())
{
    cout << pq.top() << " ";
    pq.pop();
}


// creating heap from user defined objects
// Let's initialize the properties of `Person` object first
p1.name = "Linus Torvalds";
p1.age = 47;

p2.name = "Elon Musk";
p2.age = 46;

p3.name = "Me!";
p3.age = 19;

// Initialize a min heap

// Note: We defined a comparator is_older in the beginning to
// compare the ages of two people.

priority_queue <struct Person, vector<struct Person>, is_older> mh;
mh.push(p1);
mh.push(p2);
mh.push(p3);

// Extracting items from the heap
while (!mh.empty())
{
    struct Person p = mh.top();
```

```
=========
 SORTING
=========
*/

  // The following list type initialization is only supported in versi
  //vector<int> int_vec = {56, 32, -43, 23, 12, 93, 132, -154};

  // If the above style of initialization doesn't work, use the follow
  static int arr[] = {56, 32, -43, 23, 12, 93, 132, -154};
  int arr_len = sizeof(arr) / sizeof(arr[0]);
  vector <int> int_vec(arr, arr + arr_len);

  cout << endl;
  // Default: sort ascending
  // sort(int_vec.begin(), int_vec.end());
  // To sort in descending order:
  // Do not include the () when you call wayToSort
  // It must be passed as a function pointer or function object
sort(int_vec.begin(), int_vec.end(), way_to_sort);
  for (vector <int>::iterator i = int_vec.begin(); i!=int_vec.end(); i
      cout << *i << " ";
  cout << endl;

  // sorting the array
  sort(arr, arr + arr_len);
  for (int i=0; i < arr_len; i++) {
      cout << arr[i] << " ";
  }

  // Sorting user-defined objects
  static struct Person persons[] = {p1, p2, p3};
  sort(persons, persons+3, compare_names);

  // This will print out the names in alphabetical order
  for (int i=0; i < 3; i++) {
      cout << persons[i].name << " ";
  }
```

Hope you found this helpful 😃

I've also created a gist for the above cheatsheet. Feel free to fork, suggest changes, or point out bugs!

Stl Cheatsheet C++ C++ with stl Competitive programming

2 Replies

But, interviewers usually ask two kinds of questions. 1.) How do you think a <ds_name> might have been implemented internally? 2.) How would you implement your own <ds_name> and ensure time complexity guarantees that STL provides?

Show more

Reply

Satwik Kansal 2 years ago

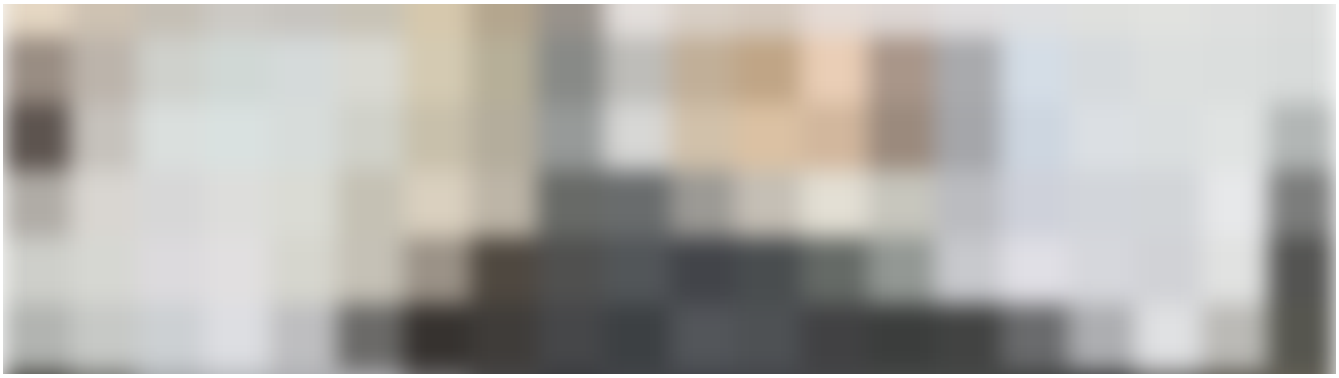
Agreed, this isn't actually intended to prepare you for interviewer questions. For those, only sky is the limit! This is intended to be used as a refresher before writing code. As far as I know, most companies do test your programming skills (either directly on the whiteboard or via any collaborative editor like Google Docs) and the knowledge of STL at that

Show more

Reply

Aditi Tipnis

JavaScript: How to make API calls for each value in an array and get an array of results.



7 2

So here's how I did it. (Skip to the end for full code)

Promise

```
new Promise(executor)
```

> *The Promise object represents the eventual completion (or failure) of an asynchronous operation, and its resulting value.*

executor