Beginner's Guide to Competitive Programming With C++

Mohammad Osoolian

February 2023

Contents

1 Why C++? 3

2 C++ for competitive programming 3
- 2.1 Basic syntax 4
- 2.2 Important Data types 4
- 2.3 Important functions 5

3 Competitive programming conventions 6
- 3.1 Problem Conventions 6
- 3.2 Code Conventions 7
  - 3.2.1 Headers 9
  - 3.2.2 Constraints 11
  - 3.2.3 Global variables 12
  - 3.2.4 Fast I/O 12
  - 3.2.5 Typedefs 12
  - 3.2.6 Macros 13
  - 3.2.7 Templates 13

4 Practice makes perfect 14

1 Why C++?

There are many different options in programming languages to start competitive programming (CP). However, most of CP competitors choose C++ because of its many benefits. If you still aren't sure if using C++ is the right choice for you, here are some reasons why it might be the best option.

Faster: The most important reason is that C++ is much faster than many other languages. In real contest, there is no difference between time limit of various languages. Therefor, those submits with faster languages have more chance of being finished in the specified time limit.

More clear: C++ is a simple language and it's much closer to machine language. Therefor, in writing C++ code, every thing is just clear for you and no extra operatoin is done behind your code. This is an important adventage that prevents you from making mistakes in calculating time and memory complexity.

Various algorithm tools: C++ has wide different tools for algorithms that you probably need to use in CP problems. It also has different kinds of data structures that can be very helpful for you.

Widely used: C++ is considered to be the best choice for CP problems by 75% of the programmers across the world. This brings a huge community and great support for you. For example you can use snippets, templates, some data structure implementations and many algorithms implemented in C++.

2 C++ for competitive programming

Learning a language for CP problems is actually much easier than learning that language for developing applications. Furthermore, even the rules and conventions of competitive programming differs from programming as a developer. C++ is a powerful language with a lot of tools and libraries and learning all of them might be confusing for you. However, you don't have to learn everything about C++. The only things you need is basic syntax and some useful datatypes and functions and some special conventions for CP that we will talk about later.

2.1 Basic syntax

If you're already familiar with any programming language, it will probably be easy for you to learn C++ syntax as well. You only need to learn simple topics, including input, output, variables, conditions, loops and functions. Additionaly, C++ contains a variable type called "pointer" that you may have not heard of before. It's a good idea to search about it and get familiar with it. Here are some useful links:

- https://www.w3schools.com/cpp/cpp_syntax.asp
- https://www.tutorialspoint.com/cplusplus/cpp_basic_syntax.htm
- https://www.codecademy.com/learn/c-plus-plus-for-programmers/modules/basic-syntax-in-cpp/cheatsheet

Don't try to get too deep into it! You'll learn as much as you need as you go.

2.2 Important Data types

vector: Just like arrays, vectors use contiguous storage locations for their elements, which means that their elements can also be accessed using indexes, and just as efficiently as in arrays. But unlike arrays, their size can change dynamically, with their storage being handled automatically by the container.

https://www.mygreatlearning.com/blog/vectors-in-c/

pair: Pair is used to combine together two values that may be of different data types. Pair provides a way to store two heterogeneous objects as a single unit. It is basically used if we want to store tuples. Pair in C++ behaves similarly to tuple in python. It consists of two elements first, second. The first element is referred to as first while the second element is referred to as second. This order must be fixed (first, second).

https://www.scaler.com/topics/cpp/pair-in-cpp/

queue: Queues are a type of container adaptors that operate in a first in first out (FIFO) type of arrangement. Elements are inserted at the back (end) and are deleted from the front.

https://www.programiz.com/cpp-programming/queue

stack: Stacks are a type of container adaptors with LIFO(Last In First Out) type of working, where a new element is added at one end (top) and an element is removed from that end only

https://www.programiz.com/cpp-programming/stack

priority_queue: A C++ priority queue designed such that the first element of the queue is either the greatest or the smallest of all elements in the queue. We can also change it to the smallest element at the top

https://www.mygreatlearning.com/blog/priority-queue-in-cpp/

2.3 Important functions

min, max, minmax: These functions are used to get the minimum, maximum, or both between two or more objects. Additionally, a comparer function can be passed to these functions.

https://www.educative.io/courses/cpp-standard-library-including-cpp-14-and-
7nYWrLv9jkQ

max_element, min_element: max_element and min_element return an iterator pointing to the element with the largest or smallest value in the range [first, last). The comparisons can be performed using a predefined function.

https://www.geeksforgeeks.org/max_element-in-cpp/

swap: This function swaps the values of two variables of any data type.

https://www.geeksforgeeks.org/swap-in-cpp/

gcd: C++ builtin function gcd or __gcd in c++14 and before, calculates the gcd of two numbers.

https://www.geeksforgeeks.org/stdgcd-c-inbuilt-function-finding-gcd/

sort: This is a very useful functoin in C++ algorithm header that sorts an array or a vector of elements in O(nlogn). It is possible to pass a comparer function to sort elements in a specific order.

https://www.educative.io/answers/what-is-the-stdsort-function-in-cpp
You can also use a lambda function instead of defining a comparer function. Check this link:

https://stackoverflow.com/questions/5122804/how-to-sort-with-a-lambda

any_of, all_of, none_of: These functinos loop over a vector or an array and check if any, all or none of elements have a specific condition. You can pass a boolian function as the condition or write a lambda function in parameters.

https://www.geeksforgeeks.org/any_of-function-in-cpp-stl/

https://www.geeksforgeeks.org/stdnone_of-in-c/

https://www.geeksforgeeks.org/stdall_of-in-cpp/

unique: Removes all but the first element from every consecutive group of equivalent elements in the range [first,last)

https://www.geeksforgeeks.org/stdunique-in-cpp/

lower_bound, upper_bound: These functions are used to find bounds of an elements in vectors and arrays and sets. They work in $O(logN)$ for arrays and vectors and can be used instead of writing binary search functions.

https://jimmy-shen.medium.com/lower-bound-and-upper-bound-of-c-2f635619fa97

__builtin_popcount: Counts the number of bits set to 1 of an integer a. This function is specialy used in working with bitmasks.

https://www.geeksforgeeks.org/cpp-__builtin_popcount-function/

3 Competitive programming conventions

Competitive programming has its own rules and conventions. By being aware of these conventions, you can avoid unnecessary details and focus on more important aspects.

3.1 Problem Conventions

CP problems have a special form most of the time. They usually have **Title**, **Description**, **Input**, **Output**, **Example** and **Notes**.[1] Here is a sample problem on Codeforces(CF) for you to make the explanations below clearer.

Title: This part contains the name of the problem, the time limit, and the memory limit. These limitations are really important because it's impossible to answer the question without considering them. However, they usually don't change much. Time limits are usually around 1 or 2 seconds and memory limits are usually around 256 megabytes

[1]The conventions written in this guide is based on codeforces.com.

Description: The main explanation of problem is in the Description. You should read this section carefully to understand what exactly the problem is.

Input: In this section you will find out how inputs of problem are passed to you and how you should read them. There is another important piece of information here which is **input boundaries**. Input boundaries show you how much the inputs can get bigger. For example it shows you how long the length of input array could be, or how many queries exist in one test case at most. Input boundaries are really important because they can help you determine the order of time complexity of your answer.

Normal computers can do $3 \cdot 10^8$ *operations in one second.* Therefore, assuming that you have one array as input and boundary of length of array(N) is $1 < N < 10^5$, then your answer has to be at most in $O(NlogN)$. Sometimes you can discover what techniques and data structures to use if you know the suitable time complexity of the answer.

Output: the Output section tells you how you should print your answers. Sometimes they should be printed in oneline, sometimes in multiple lines, sometimes with spaces between, sometimes with out it, sometimes in lower case, and sometimes in upper case. Make sure you print exactly the same output as the problem wants. Answers are corrected by machine. Therefore, even a little differnece can make your answer incorrect.

Example: After explaining the problem and the form of input and output, an example is given to you in this part to ensure that you understand the concept of the problem. *Always try your code with the examples before submitting.*

Notes: Finally, in the last part, there are some explanations for the examples above. If you cannot understand the concept of the problem, read this section carefully and try to figure out the relationship between the input and output in the examples.

3.2 Code Conventions

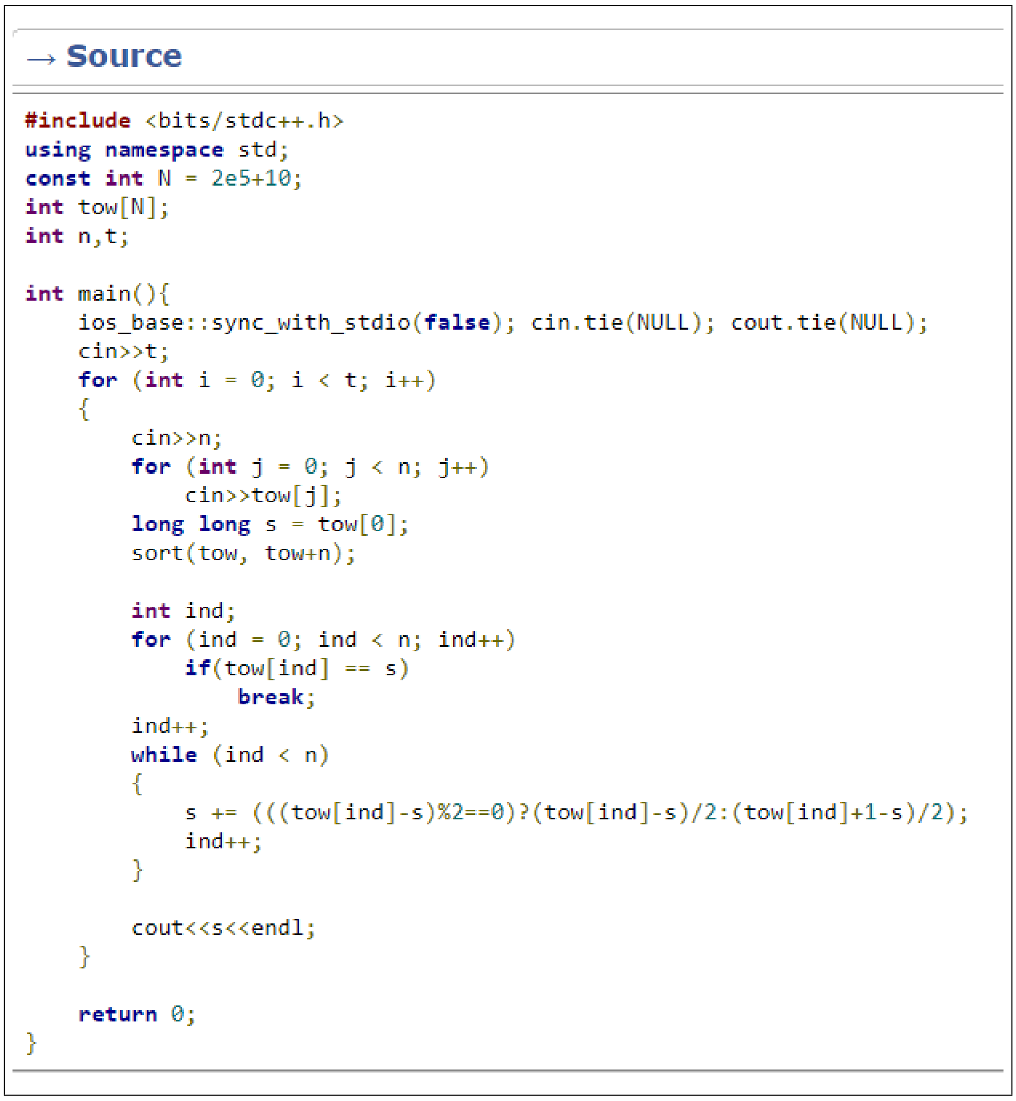
CP problems coding has a different style from other fields of coding. For example, when coding for a web project, the goal is to keep the project up and secure; it needs to be maintained, developed, and get new features. So your code must follow clean code rules to be easy to read and modifiable for other developers.

However, in CP, the only goal of coding is to pass the tests and get the score of problem in the shortest time possible. But that doesn't mean that you shouldn't follow clean code rules. Actually, to achieve this goal and get the highest score, your code must be clean, but in a different way.

That is because some of the clean code rules of big projects take so much time from you, causing you to lose points for latency. To solve this problem, some rules have been changed in a way that takes less time from you while still keeping your code clean and easy to read at the same time.

In conclusion, in CP, coding has different purposes and values compared to other contexts of coding, so don't expect the rules to be exactly the same as the coding rules you have heard before.

In this section, you will learn some conventions in CP coding, with help of this code submitted for this question in CF.



```
#include <bits/stdc++.h>
using namespace std;
const int N = 2e5+10;
int tow[N];
int n,t;

int main(){
    ios_base::sync_with_stdio(false); cin.tie(NULL); cout.tie(NULL);
    cin>>t;
    for (int i = 0; i < t; i++)
    {
        cin>>n;
        for (int j = 0; j < n; j++)
            cin>>tow[j];
        long long s = tow[0];
        sort(tow, tow+n);

        int ind;
        for (ind = 0; ind < n; ind++)
            if(tow[ind] == s)
                break;
        ind++;
        while (ind < n)
        {
            s += (((tow[ind]-s)%2==0)?(tow[ind]-s)/2:(tow[ind]+1-s)/2);
            ind++;
        }

        cout<<s<<endl;
    }

    return 0;
}
```

Figure 1: sample submitted code

3.2.1 Headers

There are some frequently used standard libraries in CP problems that may not be needed in every problem you solve. However, competitors usually include all of them regardless of whether they are necessary to use or not. The reason for that is because it is just easier and takes less time comparing to include every library just when you need it in the code. There could be also a `using namespace std;` after including libraries for convinience and to reduce code.

here are the list of frequent used standard libraries[2]:

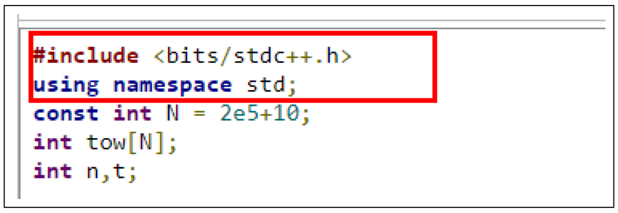
```
#include <iostream>
#include <string>
#include <vector>
#include <algorithm>
#include <sstream>
#include <queue>
#include <deque>
#include <bitset>
#include <iterator>
#include <list>
#include <stack>
#include <map>
#include <set>
#include <functional>
#include <numeric>
#include <utility>
#include <limits>
#include <time.h>
#include <math.h>
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <assert.h>

using namespace std;
```

However, rather than including all these libraries, you can only include `<bits/stdc++>` header. It's not a standard library but contains all necessary headers for CP.

```
#include <bits/stdc++>
using namespace std;
```

[2] source



```
#include <bits/stdc++.h>
using namespace std;
const int N = 2e5+10;
int tow[N];
int n,t;
```

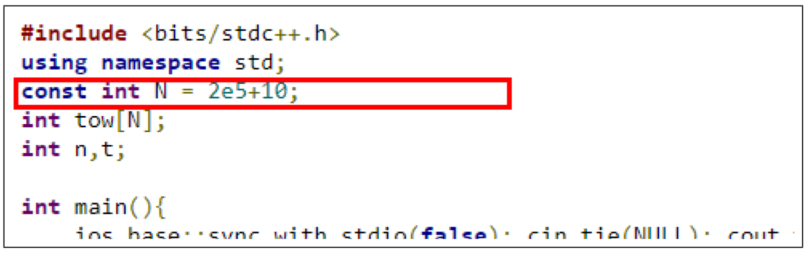
Figure 2: included headers

3.2.2 Constraints

Constraints come from the input boundaries of the problems. It is a convention among CP competitors to globaly declare `const int` variables for baoudaries. Thats because these boundaries are so important in creating arrays and memory needed variables to hold input values. In CP, memory efficiency is not a priority; therefore, the maximum possible number of inputs is considered for creating arrays, and this is where constraints are used. Look at the example problem and submitted code given at the beginning of this section.

It is specified in the problem that the number of towers is $2 \cdot 10^5$ at most. Here $2 \cdot 10^5$ is a constraint. Therefore, we can write `const int N = 2e5;` and use N to initialize arrays we need, or in the better choice we can write `const int N = 2e5 + 10`. This addition dosn't change the memory complexity but it could help in preveting from bugs that appear in corner cases.

- $n \cdot em$ stands for $n \cdot 10^m$.
- This syntax actually returns a `float` number, not an `int`. For example, in C++, `2e3` equals to `2000.0` not `2000`. Therefore, it is incorrect to write `int nums[2e5+10]`. But when you assign this float value to an integer variable like N, C++ casts it to an integer value and after that, you can use the variable to initialize arrays, as in the given example.



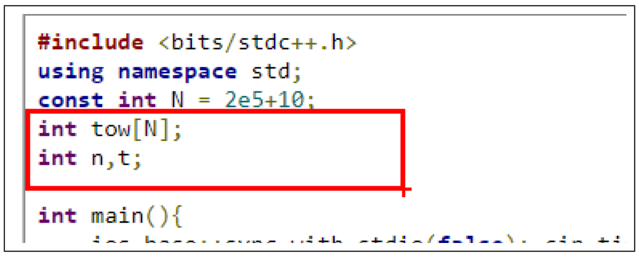
```
#include <bits/stdc++.h>
using namespace std;
const int N = 2e5+10;
int tow[N];
int n,t;

int main(){
    ios_base::sync_with_stdio(false); cin.tie(NULL); cout
```

Figure 3: constraints of code

3.2.3 Global variables

Basically, global variables hold everything you read from input, the result of preprocessing on input values, and anything you need in the main function. Number of test cases, length of arrays, the arrays that keep input data, DP arrays, etc, are some examples of frequently used global variables.It is not a good idea to use global variables in large projects; however, in CP contests, the main goal is to write the answer as quickly and easily as possible, and making use of global variables can help a lot. Declaring global variables makes it much easier for you to define functions because you don't have to pass them through function parameters. Instead, you can access them directly.



```
#include <bits/stdc++.h>
using namespace std;
const int N = 2e5+10;
int tow[N];
int n,t;

int main(){
```

Figure 4: global variables

3.2.4 Fast I/O

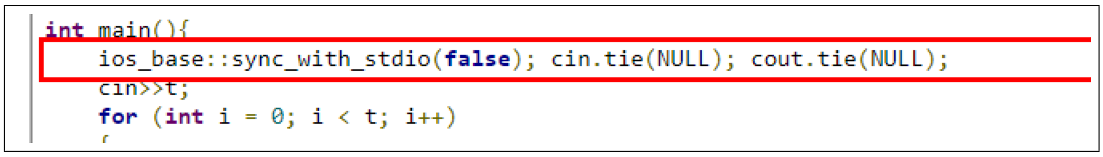
In some questions, the input size is too large and you lose a lot of time just reading the input, resulting in a time limit error In these cases, you can use `scanf` and `printf` as a faster method instead of `cin` and `cout`, or you can add these lines at the beginning of `main` function to make `cin` and `cout` faster.

```
ios_base::sync_with_stdio(false);
cin.tie(NULL);
cout.tie(NULL);
```

Both ways can solve the problem but the second one is probably easier for you.

3.2.5 Typedefs

`typedef` is a syntax in C++ that allows you to define abbreviations for variable types. Using this syntax, you can code faster and make it shorter and easier to read. Here are some famous typedefs used by many competitors:



```
int main(){
    ios_base::sync_with_stdio(false); cin.tie(NULL); cout.tie(NULL);
    cin>>t;
    for (int i = 0; i < t; i++)
```

Figure 5: fast I/O

```
typedef long long LL;
typedef pair<int, int> pii;
typedef pair<LL, LL> pll;
typedef pair<string, string> pss;
typedef vector<int> vi;
typedef vector<vi> vvi;
typedef vector<pii> vii;
typedef vector<LL> vl;
typedef vector<vl> vvl;
```

3.2.6 Macros

Macros are the `#define` statements in C++ and they are another technique of shortening code in CP. Macros are useful in many cased but it's not a good idea to use them too much. In the past, `#define` statements were used instead of `typedef`, but most of the macros have faster and safer alternatives now. Here is some useful macros:

```
#define PI 3.1415926535897932384626
#define mp make_pair
#define fi first
#define se second
#define DBG(vari) cerr<<#vari<<" = "<<(vari)<<endl;
#define pb push_back
```

3.2.7 Templates

After becoming familiar with all these conventions, you may like some of them and even have your own macros. Now, instead of typing macros every time, you can create a collection of them and make a snippet in your editor that copies and pastes this collection. This collection is called a **template** in CP. This is my template:

```
#include <bits/stdc++.h>
using namespace std;

#define mp make_pair
#define fi first
#define se second
#define DBG(vari) cerr<<#vari<<" = "<<(vari)<<endl;
#define pb push_back
typedef long long LL;
typedef pair<int, int> pii;
typedef pair<LL, LL> pll;


int main(){
    ios_base::sync_with_stdio(false); cin.tie(NULL); cout.tie(NULL);

    return 0;
}
```

You can also find more templates on the internet or make your own template and customize it as you like.

4 Practice makes perfect

Now that you are familiar with the rules of Competitive Programming, the only thing left to do is practice. Keep trying and work hard, and you will be able to solve more complex problems in less time. Don't give up easily on difficult problems and try not to look at the answers. Instead, take a break, come back later with fresh ideas.These websites are great for practicing. There are thousands of problems in various levels and topics. There are also exciting contests that you can attend and challenge your self.

- Codeforces
- TopCoder
- CodeChef
- HackerRank