COMPETITIVE PROGRAMMING → CHEAT SHEET IN C++

How to scan two integers indefinitely?

```
while(scanf("%d %d\n",&m,&n)==2)
{
	cout << m << n << endl;
}
```

How to scan a line indefinitely?

```
string input_line;
while(getline(cin,input_line))
{
	cout << input_line << endl;
}
```

How to scan a string and unknown number of integers on a single line indefinitely?

```
#include <sstream>
string input_line;
string first_string;
vector<int> vec;
int i;
while(getline(cin,input_line))
{
	stringstream ss(input_line);
	ss >> first_string;

	while(ss >> i){
		vec.push_back(i);
	}
	vec.clear();
}
```

Sample Input :

Utsav 1 2 3
Chokshi 1 7 8

How to perform string formatting for I/O?

```
#include <iomanip>
//Setting alignment and word length
```

```
cout << setw(20) << left << "HEADER" << endl;

//Filling empty space with some character
cout << setfill('-') << setw(20) << left << "HEADER" << endl;
```

How to perform float/double formatting for I/O?

```
#include <iomanip>
float pi = 3.1456;
//Setting precision upto 2 decimal points
cout << fixed;
cout << setprecision(2) << pi << endl;
```

Defining multi-dimension array (statically and dynamically) and initializing it

```
// statically
bool mat[9][9][9];
memset(mat,false,sizeof(mat));// dynamically in C style
int n = 9;
bool *mat[n];
for(int i=0; i<n; i++){
      mat[i] = (int*)malloc(n*sizeof(int));
}
memset(mat,false,sizeof(mat));// dynamically using "new" keyword
int n = 9;
int** mat[n];
for(int i=0; i<n; i++){
  //empty brackets make it allocate memory and initialize with
zero
  mat[i] = new int[n]();
}
```

Important String functions and Techniques

Iterating over characters of string?

```
string sample_string = "utsav";
int length = sample_string.size();
for(int index=0; index < length; index++){
      cout << sample_string[index] << endl;
}
```

Sorting a string and Reveresal of a string

```
string sample_string = "utsav";
```

```
//It is similar to vector : inplace sorting
sort(string.begin(),string.end());
sort(string.begin(),string.begin()+2);

//Inplace reverse
reverse(sample_string.begin(),sample_string.end())
```

Checking a type of character

```
string str = "utsav";

isdigit(str[0])
isalpha(str[0])
isalnum(str[0])
ispunct(str[0])
isspace(str[0])
islower(str[0])
isupper(str[0])// converting character to upper or lower case
toupper(str[0])
tolower(str[0])
```

Converting string to integer and vice-versa

```
#include <cstdlib> //for atoi
#include <sstream> //for stringstream

//converting string to integer
string str = "124";
int i = atoi(str.c_str());

//converting integer to string
int i = 42;
stringstream ss;
ss << i;
string s = ss.str();//alternative method for converting integer
to string
string numStr = to_string(i);
```

Tokenizing and iterating over string with char delimiter

```
#include <string>
#include <sstream> //for stringstreamstring source =
"/users/home/docs";
string tmp;
char delim = '/';
stringstream ss(source);while(getline(ss,tmp,delim)) {
     cout << tmp << endl;
}
```

Finding next/previous permutation of string

```
#include <string>

string str= "abcd";

bool flag = next_permutation(str.begin(),str.end());

//True : next lexicographically larger permutation exists and
replaces str with it
//False : current string is last permutationif(flag){
	cout << "Next Perm :" << str << endl;
}
else{
	cout << "Last permutation of string reached." << endl;
}

//Previous lexicograhically smaller perm
flag = prev_permutation(str.begin(),str.end());
```

String comparison

```
#include <string>

string s1 = "abc";
string s2 = "abcd";
string s3 = "abd";
string s4 = "abc";
string s5 = "aba";

int i = s1.compare(s2); // s1 < s2 -> returns some val < 0 (-1)
i = s1.compare(s3); // s1 < s3 -> returns some val < 0 (-1)
i = s1.compare(s4); // s1 = s4 -> returns 0
i = s1.compare(s5); // s1 > s5 -> return some val > 0 (2)
```

SubString and Find-Replace

```
#include <string>

string fullname = "a12utsav chokshi"
string name = fullname.substr(2,5);
//substr(start_index,length)
string surname = fullname.substr(9);   //returns "chokshi"size_t
found = fullname.find("a"); // substring to be found can be
string or character
size_t found = fullname.find('a');

if(found != nopos){
	cout << "Pos :" << found << endl;
}
```

```
found = fullname.find("a",found+3,5);  //similar to finding a in
utsav

fullname.replace(found,1,"hello");  //a12utsav chokshi ->
a12utshellov chokshi
```

Important Vector functions and Techniques

Commonly used vector functions

```
#include <vector>vector<int> vec;

//getting length of vector
int length = vec.size();

//clearing content of vector
vec.clear();

//defining intial size of vector
vec.reserve(100);

//getting capacity of vector
vec.capacity();

//inserting elemets to vector
vec.push_back(5);

//accessing elements of vector
int temp = vec[0];

//intializing vector with some value
vector<float> vec2(10,0.0);

//removing nth element from vector
int n = 3;
vec.erase(vec.begin()+n);//removing(erasing) last element from
vector
vec.pop_back();//swaping two elements
swap(vec[0],vec[1]);//accessing last element
vec.back();
vec[vec.size()-1];//reversing vector
reverse(vec.begin(), vec.end());//finding element with max or
min value
vector<int> vec3({10,30,20});int mx =
*max_element(vec3.begin(),vec3.end());
int mn = *min_element(vec3.begin(),vec3.end());//finding sum of
all values
int baseVal = 100; //for finding sum keep baseVal zero
```

```
int sum = accumulate(vec3.begin(),vec3.end(),baseVal); //returns
160
```

Sorting vector of struct instances

```
#include <vector>struct Point{
	float x1;
	float y1;
}
bool pointComp(const Point& lhs, const point& rhs){
	// for sorting in ascending order
	return lhs.x1 < rhs.x1;
}

vector pointVec;

sort(pointVec.begin(),pointVec.end(),pointComp);
sort(pointVec.begin(),pointVec.begin()+5; pointComp);
```

If comparison function needs to be added under a class, put `static` keyword after `bool` !

Defining array of vectors

```
#include <vector>vector<int> arrayOfVectors[n];// need not
initialize each index arrayOfVectors, access directly
arrayOfVectors[0].push_back(1);
```

How to return an empty vector in one line ?

```
#include <vector>return vector<int>();
```

Initializing vector using set

```
#include <vector>
#include <set>set<int> s;
s.insert(1);
s.insert(2);vector<int> vec(s.begin(), s.end());
```

Important search functions and their behavior

- binary_search, upper_bound, lower_bound works only on sorted vector

- binary_search returns ‘true/false’.
- To find a position of the first occurrence of a search_element, use a lower_bound function
- To find a position of the last occurrence of a search_element, use a upper_bound function and reduce iterator by 1
- lower_bound returns: *the first occurrence of search_element if it is present otherwise next greater element of search_element*
- upper_bound returns: *the next greater element of search_element*
- If no element in the vector compares greater than search_element, the functions(lower_bound, upper_bound) returns vec.end()

```
#include <vector>vector<int> vec {10,20,20,30,30,40};//vector
needs to be sorted for binary_search, lower_bound
//and upper_bound to work correctly// binary_search
bool found = binary_search(vec.begin(),vec.end(),30);//
lower_bound
vector<int>::iterator it =
lower_bound(vec.begin(),vec.end(),30);
cout << "pos :" << it - vec.begin() << endl; // pos : 3//
lower_bound
vector<int>::iterator it =
lower_bound(vec.begin(),vec.end(),40);
cout << "pos :" << it - vec.begin() << endl; // pos : 5//
upper_bound
vector<int>::iterator it =
upper_bound(vec.begin(),vec.end(),30);
cout << "pos :" << it - vec.begin() << endl; // pos : 5//
upper_bound
vector<int>::iterator it =
upper_bound(vec.begin(),vec.end(),40);
cout << "pos :" << it - vec.begin() << endl; // pos : 6
```

Copying vector

```
#include <vector>vector<int> vec1(10,0);// Copying while
creating
vector<int> vec2(vec1);// Copying after creation
vector<int> vec3;
vec3 = vec1;// Copying specific section after creation
vector<int> vec4;
vec4.assign(vec1.begin()+1, vec1.end());
```

How to define two dimensional vector and initialize it ?

```
#include <vector>int numRows = 10; //Can be dynamically assigned
int numCols = 10; //Can be dynamically assigned// defining and
initiliazing
vector<vector<bool>> vec1(numRows, vector<bool>(numCols,
false));// only defining with size
vector<vector<bool>> vec1(numRows, vector<bool>(numCols));//
only defining
vector<vector<bool>> vec1;
```

Important Map functions and Techniques

- Maps in C++ are implemented using Red-Black Tree.
- So it takes O(logn) time for insertion/deletion/search.
- So it is better than array of pairs [O(n)] but worse than hash tables [O(1)].
- To use unordered_map : just replace map with unordered_map.
- unordered_map is feature of C++11 and it is implemented using hashing. Hence provides O(1) complexity for access.

Defining and accessing map

```
#include <map>

map<string,int> m1;
m1['utsav'] = 1;
```

Checking for key existence and iterating over keys in map

```
#include <map>map<string,int> m1;
m1["utsav"] = 1;
//Iterating over all keys
map<string,int>::iterator it;
vector<string> vec;for(it=m1.begin(); it!=m1.end(); it++){
	vec.push_back(it->first);
}//Checking for key existence
string key = "utsav";
if(m1.find("utsav") == m1.end()){
	cout << "Key not found." << endl;
}
```

Erasing keys

```
#include <map>map<char,int> m1;
m1['a'] = 1;
m1['b'] = 2;
m1['c'] = 3;
m1['d'] = 4;
m1['e'] = 5;// erases key 'a'
m1.erase('a');map<char,int>::iterator it1 = m1.find('c');//
erases keys : 'c','d','e'
m1.erase(it1,m1.end());
```

Finding lower and upper bound on keys

```
#include <map>map<char,int> m1;
m1['a'] = 1;
m1['b'] = 2;
m1['c'] = 3;
m1['d'] = 4;
m1['e'] = 5;// points to b
map<string,int>::iterator it1 = m1.lower_bound('b');// points to
e
map<string,int>::iterator it2 = m1.upper_bound('d');
```

Note difference between how you use lower_bound with vector and with map !

Obviously , lower_bound and upper_bound does not work with unordered_map :D

Important Stack functions and Techniques

Defining and accessing stack

```
#include <stack>

stack<int> s;

s.push(3);
int temp = s.top();
s.pop();        //returns nothing

bool flag = s.empty();
int  length = s.size();
```

There is no such thing as iterator for stack. So don't try ! :)

Important Queue functions and Techniques

Defining and accessing queue

```
#include <queue>

queue<int> q;

q.push(3);                //Inserts element at the end.
q.pop();                  //Returns nothing and deletes element
from the front.

int f = q.front();  //First element in queue
int b = q.back();   //Last element in queue

int length = q.size();
bool flag = q.empty();
```

There is no such thing as iterator for queue. So don't try ! :)

Important Set functions and Techniques

- Sets in C++ are containers that stores only unique elements.

- To use unordered_set : just replace set with unordered_set.
- unordered_set is feature of C++11

Defining and accessing set

```
#include <set>

set<int> s;

s.insert(3);        //Inserts element into the set.
s.erase(3);         //Deletes element from set.

int length = s.size();
bool flag = s.empty();

//clearing content
s.clear();
```

Initializing set with vector

```
#include <set>
#include <vector>vector<int> vec;set<int>
s2(vec.begin(),vec.end());
```

Finding element and iterating over all elements in set

```
#include <set>

set<int> s;
s.insert(2);

if(s.find(2) != s.end()){
	cout << "Found" << endl;
}
else{
	cout << "Not Found" << endl;
}

set<int>::iterator it;
for(it=s.begin(); it!=s.end(); it++){
	cout << *it << endl;
}
```

Important Priority Queue / Heap functions and techniques

- Building heap (i.e. initializing pq with vector) takes O(N)
- Insertion/Deletion (i.e. push/pop) of single element takes O(logN)
- Hence, If you build heap by inserting element one by one then it takes O(NlogN)
- Getting top element (i.e. finding largest/smallest) takes O(1)

Defining and accessing priority queue

```
#include <queue>priority_queue<int> maxHeap;
priority_queue<int, vector<int>, greater<int>>
minHeap;maxHeap.push(3);   //Inserts element into pq at
appropriate place
maxHeap.pop();     //Removes top element//Retrieves
greatest(maxHeap)/smallest(minHeap) element
int top = maxHeap.top()int length = maxHeap.size();
bool flag = maxHeap.empty();//clearing content
maxHeap.clear();
```

Initializing priority queue with vector

```
#include <queue>
#include <vector>vector<int> vec({1,5,10,2});//initializing
maxHeap
priority_queue<int> maxHeap(less<int>(), vec); //initializing
minHeap
priority_queue<int, vector<int>, greater<int>>
minHeap(greater<int>(), vec);
```

How to use Pair?

Defining and accessing pair structure

```
#include <utility>
```

```
pair<int,int> p1;
p1 = make_pair(10,20);

cout << p1.first << " " << p1.second << endl;
```

How to define Trie in C++ ?

```
const int ALPHABET_SIZE = 26;struct TrieNode{

    bool isWord;
    TrieNode* children[ALPHABET_SIZE];

    TrieNode(){
        isWord = false;
        for(int i=0; i<ALPHABET_SIZE; i++){
            children[i] = NULL;
        }
    }
};class Trie {
    private:
        TrieNode* root;
    public:
       Trie() {
            root = new TrieNode();
       }
}
```