

Competitive Programming first lesson



הסביר קוד פשוט

- כר נראה קוד ב++:

```
1 #include <bits/stdc++.h>
2 #define int long long
3 using namespace std;
4
5 ▶ signed main() {
6     cin.tie( tiestr: 0 )->sync_with_stdio( sync: false ), cout.tie( tiestr: 0 );
7     cout << "Hello, World!" << endl;
8     int x; cin >> x;
9     cout << 2*x << endl;
10    return 0;
11 }
```

הסבר שורה - שורה

- כר נראה קוד ב++:

```
1 #include <bits/stdc++.h>
```

הסבר שורה - שורה

זהו include כמעט כמעט כל מה שתצרכו לתוכנות תחרותי (מומלץ מאוד!) (לא תמיד עובד בVS).

```
1 #include <bits/stdc++.h>
```

הסבר שורה - שורה

זה עוזר לנו להפטר בנוחות מהבעיות של overflow (במחר שנראה בהמשך).

```
1 #include <bits/stdc++.h>
2 #define int long long
```

הסבר שורה - שורה

יאיר לא יאהב לראות את זה, אבל למי אכפת – חוסך הרבה זמן וקוד.

```
1 #include <bits/stdc++.h>
2 #define int long long
3 using namespace std;
```

הסבר שורה - שורה

מרוח נחמד

```
1 #include <bits/stdc++.h>
2 #define int long long
3 using namespace std;
4
```

הסבר שורה - שורה

זהו המחיר של ה`#define` - משנהים את הтипוא של `main` ל`signed` (מחיר פועל).

```
1 #include <bits/stdc++.h>
2 #define int long long
3 using namespace std;
4
5 ▶ signed main() {
```

הסבר שורה - שורה

השורה זו מאייצה את הקוד ע"י ביטול printf, scanf וע"י המתנה לפני קליטה ולפני פליטה (לא לעשות בשאלות interactive).

```
1 #include <bits/stdc++.h>
2 #define int long long
3 using namespace std;
4
5 ▶ signed main() {
6     cin.tie( tiestr: 0 )->sync_with_stdio( sync: false ), cout.tie( tiestr: 0 );
```

הסבר שורה - שורה

השורה זו מדפסה "שלום, עולם!" בשפה האנגלית (נדרשת רמת מתקדמים ב' להבין).

```
1 #include <bits/stdc++.h>
2 #define int long long
3 using namespace std;
4
5 ▶ signed main() {
6     cin.tie( tiestr: 0 )->sync_with_stdio( sync: false ), cout.tie( tiestr: 0 );
7     cout << "Hello, World!" << endl;
```

הסביר שורה -

פה אנחנו מגדירים משתנה וקולטים אותו.

```
1 #include <bits/stdc++.h>
2 #define int long long
3 using namespace std;
4
5 ▶ signed main() {
6     cin.tie( tiestr: 0 )->sync_with_stdio( sync: false ), cout.tie( tiestr: 0 );
7     cout << "Hello, World!" << endl;
8     int x; cin >> x;
```

הסבר שורה - שורה

פה אנחנו מדפיסים את הערך הנקלט כפול 2 ולאחר מכן שוטפים את הערך buffer (שימוש לב שפה לא רק יורד שורה אלא גם שוטף את הבאר, ולכן בלולאות ארוכות מומלץ '\n').

```
1 #include <bits/stdc++.h>
2 #define int long long
3 using namespace std;
4
5 ▶ signed main() {
6     cin.tie( tiestr: 0 )->sync_with_stdio( sync: false ), cout.tie( tiestr: 0 );
7     cout << "Hello, World!" << endl;
8     int x; cin >> x;
9     cout << 2*x << endl;
```

הסביר שורה - שורה

מחזירים 0, וסוגרים את הפונקציה.

```
1 #include <bits/stdc++.h>
2 #define int long long
3 using namespace std;
4
5 ▶ signed main() {
6     cin.tie( tiestr: 0 )->sync_with_stdio( sync: false ), cout.tie( tiestr: 0 );
7     cout << "Hello, World!" << endl;
8     int x; cin >> x;
9     cout << 2*x << endl;
10    return 0;
11 }
```

מבנה נתונים בספריית STL

- pair, tuple
 - string (you know from OOP)
 - vector
 - set, multiset
 - map, unordered_map
 - priority_queue
 - queue
 - stack
 - deque
- שאר מבני הנתונים הם גrouים או ניתנים להחלפה ע"י אלה.

pair, tuple

```
pair<int, int> pair1 = { &x, &y};  
int f = pair1.first;  
cout << f << '\n';
```

- Very useful!.

```
tuple<int, int, int> tuple1 = {x, y, 2*x};  
f = get<0>(&tuple1);  
cout << f << '\n';
```

- Useful, but you should just use pair instead.

string

```
string s = "abcab";
if(s[0] == 'a') cout << "Yay" << '\n';
cout << s.substr( pos: 2);
```

- Access: like arrays.
- Substr: this function returns a suffix from pos to end. output: 'cab'.

vector

```
vector<int> vec = {1, 2, 3, 3, 3};  
vec.push_back(4);  
for(auto a : long long : vec) cout << a << '\n';  
vector<pair<int, int>> v = {{x: 1, y: 2}, {x: 3, y: 4}};  
for(auto [a : long long , b : long long ] : v) cout << a << b << '\n';  
auto it : ptrdiff_t = lower_bound( first: vec.begin(), last: vec.end(), val: 3) - vec.begin();  
cout << it << '\n';  
sort( first: vec.begin(), last: vec.end());
```

- Vector is like an array but better. You can add new values.
- Line 3 (on the left): iterating by value.
- Line 5: iterating over all pairs in the vector v
- Line 6: returns the index to the first occurrence of val = 3 in a sorted array (binary search).
- Sort – sorts.

sorting:

- You can easily sort a vector in the most efficient way (optimized $n\log n$) using the `sort()` function. If you want custom sorting (for example if you sort pairs and want them to be sorted by the second and then the first), you can make a sort function and put it in the `sort`.

```
int n; cin >> n;
vi arr(n);
for(int i = 0; i < n; i++) cin >> arr[i];
sort(first: arr.begin(), last: arr.end());
//special sort method - sort(arr.begin(), arr.end(), sorter);
```

```
bool sorter(pair<int,int> a, pair<int,int> b){//special sorter
    if(a.second < b.second) return true;
    if(a.first < b.first) return true;
    return false;
}
```

```
bool sorter(int a, int b){//this is the default sorter
    if(a < b) return true;
    return false;
}
```

Set, multiset

- The idea is an avl tree (with values being the key too). Set cannot include a single element multiple times, multiset can.
- If you want set with multiple instances of the same element you can do `set<pair<element,int>>`, and give each element a different number, now they don't count as the same.

```
set<int> set1;
set1.insert( x: 10);
set1.insert( x: 5);
cout << *set1.begin() << '\n'; // prints minimum
auto it2 : iterator<long long> = set1.lower_bound( x: 5);
if(it2 != set1.end()) cout << *it2 << '\n'; // prints the value that is the greatest while being <= x
set1.erase( x: 5); // removes if contained
```

map

- This is an AVL tree that has keys that may differ from their corresponding value.

```
map<char, int> mp;
mp['1'] = 1;
mp['2'] = 3;
for(auto [key : const char , value : long long ] : mp)
    cout << key << " : " << value <<" ";
cout << '\n';
```

- This data structure is useful where $n \leq 2 \cdot 10^5$, but $x \leq 10^9$.
- Also useful for mapping from string to int or similar.

unordered_map

- This is NOT an AVL tree, but a hash map that has keys that may differ from their corresponding value. Usually use map instead.

```
// unordered_map  
  
unordered_map<char, int> ump;  
  
ump['1'] = 1;  
ump['2'] = 3;  
  
for(auto [key : const char , value : long long ] : ump)  
    cout << key << " : " << value <<", "  
cout << '\n';
```

- This data structure is useful where $n \leq 2 \cdot 10^5$, but $x \leq 10^9$, and you rather have some constant factor instead of logarithmic, OR that you cannot hash the key.

priority_queue

- Priority queue is a maximum heap, or if you want you can change it to be a maximum heap.
- Acts as a sorted queue.

```
priority_queue<int> q; //maximum heap
q.push( x: 1);
int a = q.top();
q.pop();
priority_queue<int, vector<int>, greater<int>> q2; //minimum heap
```

Queue

- Standard queue.

```
queue<int> q;  
q.push( x: 1);  
cout << q.front();  
q.pop();
```

stack

- Standard stack

```
stack<int> s;
s.push( x: 1 );
cout << s.top();
s.pop();
```

deque

- You pronounce it “dek”. Its like a queue and stack together (can choose if to push to front or back), but most of the time you just use queue or stack.

```
deque<int> d;
d.push_back( x: 1 );
d.push_front( x: 1 );
cout << d.front();
cout << d.back();
d.pop_back();
d.pop_front();
```

rep

- Competitive programming requires you to use `for()` a lot of times, but there is a way to make it look better and shorter! If you use this define, then you can write `rep(i, 0, n)` instead of

`for(int i = 0, i < n; i++)`. We highly recommend using it.

```
#define rep(a,b,c) for(int a=b; a<c; a++)
```

```
int n; cin >> n;
rep(i, 0, n){
    cout << i;
}
```

סיבוכיות זמן

- בתכנות תחרותי סיבוכיות הזמן לא ניתנת כ-(n)O לדוגמה, כי אין המחשב יודע שהקוד שלכם הוא (n)O?
- במקומות זאת סיבוכיות הזמן תיספר בשניות, נגיד לשאלת הזאת יש זמן של שתי שניות.
- כל שנייה היא בערך 8×10^8 פעולות (5 כפול 10 בחזקת 8).
- אתם תקבלו חסמים על המשתנים שלכם.
- דוגמאות (עבור זמן של שנייה אחת):
 1. $n \log(n)^2$ $n <= 10^5$ $n > 10^5$
 2. n^2 $n <= 10^4$ $n > 10^4$
 3. $n \log n$ $n <= 10^6$ $n > 10^6$

כל הדברים האלה הם בערך כי אם בכל צעד בלולאה תעשו 20 פעולות, זה כבר יעלה כמו $\log n$ פשוט תבינו כמה זמן בערך יש לכם, ותעשו פתרון שנכנו בו. ברוב השאלות $n <= 10^5$ והפתרון לינארי או $\log n$.

תוצאות תחרות

- בתוצאות התוצאות מחושבות קודם כל לפי כמה שאלות כל קבוצה פתרה, ולאחר מכן כמה penalty יש לכל קבוצה.
- penalty של קבוצה היא כמה זמן שנקח לה להכנס כל שאלה, ועל כל שאלה שהגשה לא נכונה לשאלה שלא נכונה יש נוספת penalty של בערך 20 (תלויה בתוצאות), אבל אם הגשת הגשה לא נכונה לשאלה שלא הכנסת בסוף אז אין penalty.

TEAM	SCORE	A	B	C	D	E	F	G	H	I	J	K	L	M
♥  dETHroners ETH Zürich	11 1052	98 1 try	77 1 try	200 2 tries	160 1 try	48 1 try	34 1 try	148 2 tries	6 tries	26 1 try	51 1 try	83 2 tries	67 1 try	
♥  Heroes of the C Universidade do Porto	11 1372	53 1 try	67 2 tries	253 3 tries	282 1 try	17 1 try	32 1 try	246 1 try	1 try	23 1 try	80 1 try	101 3 tries	118 1 try	
♥  cnXtv Ecole Polytechnique	11 1593	125 2 tries	92 1 try	202 5 tries	294 5 tries	55 3 tries	64 1 try	266 2 tries		44 2 tries	69 1 try	100 1 try	22 1 try	
♥  eXotic Ecole Polytechnique	10 1136	57 1 try	35 1 try	278 1 try	142 1 try	90 1 try	31 2 tries		10 tries	47 1 try	75 1 try	53 1 try	228 5 tries	
♥  UPC-1 Universitat Politècnica de Catalunya	10 1453	114 2 tries	104 1 try	283 3 tries	316 1 try	150 2 tries	25 2 tries	7 tries		26 2 tries	62 1 try	36 1 try	217 1 try	
♥  UNIBOis University of Bologna	10 1469	135 3 tries	233 1 try	67 1 try	316 2 tries	63 2 tries	80 1 try			52 1 try	131 1 try	100 1 try	192 2 tries	
♥  Volterra gng Università di Pisa	9 866	89 1 try	60 2 tries	182 3 tries		20 1 try	44 2 tries	5 tries		31 1 try	91 2 tries	26 1 try	223 1 try	
♥  EPFL Polympiads 1 Ecole Polytechnique Fédérale de Lausanne	9 1103	130 1 try	84 1 try		3 tries	71 1 try	12 1 try	291 1 try		17 2 tries	106 1 try	35 2 tries	277 3 tries	
♥  UPC-2 Universitat Politècnica de Catalunya	9 1146	41 1 try	102 1 try			186 1 try	44 3 tries	4 tries	268 1 try	47 1 try	60 1 try	79 4 tries	199 2 tries	
♥  ENS Ulm 1 École Normale Supérieure de Paris	9 1212	137 1 try	157 1 try	257 3 tries		53 3 tries	20 2 tries	3 tries		27 2 tries	68 1 try	63 2 tries	250 3 tries	

```
#include <bits/stdc++.h>
using namespace std;
#define x first
#define y second
#define int long long
#define rep(a,b,c) for(int a=(b); (a)<(c); a++)
#define per(a,b,c) for(int a=b-1; a>=c; a--)
typedef pair<int, int> pii;
typedef vector<int> vi;
void solve(){
    int n;
}
signed main() {
    int t = 1;
    cin >> t;
    while (t--) solve();
    return 0;
}
```

הצעה לשכותבים קוד

- כדאי להתחיל לכתוב קוד כשהקובץ בו כתובים נראה כך, כי נוח לעبور ככה בשאלות בהן יש `t` (בשאלות של codeforces). זהה רק הצעה ואין שום חובה כmoben.
- `t` הוא כמות `test cases`, לדוגמה יכולים להגיד לכם שצריך לבצע את הקוד `t` פעמים. אם לא אומרים לכם אז תהפכו את `cin >> t` ל-`comment`.

שאלות תרגול בסיסית ל-STL

- קודם תעשו את **codeforces** – **Watermelon** •
Cses – **Towers** •
Cses - **Playlist** •
Cses - **Distinct Values Subarrays** •
Cses - **Distinct Values Subarrays II** •
Cses - **Movie Festival** •
Cses - **Traffic Lights** •
Cses - **Movie Festival II** •