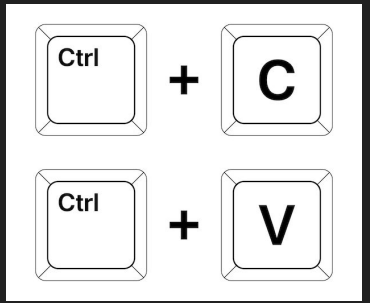
Plantilla de código para programación competitiva

Mamá: Tenemos un entorno de desarrollo en casa.
El entorno de desarrollo en casa:



¿Qué ponemos en una plantilla?

En una plantilla ponemos cosas que son útiles en muchos problemas para no escribir tanto y minimizar la posibilidad de error. ¿Qué cosas?

- Constantes
- librerias
- Aliases de tipo
- Macros

En OIA no podemos llevar código preparado más allá del provincial, así que uno tiene que memorizar su plantilla. Por eso es recomendable mantener la plantilla ligera y sin cosas innecesarias.

Constantes

Hay algunos números que son comunes en problemas de programación competitiva, y es cómodo tenerlos a mano. Por ejemplo, pi, o algunos números primos muy conocidos.

```
const double PI = acos(-1);      // 3.1415926…
const int primo1 = 1e9+7;        // 1000000007
const int primo2 = (119<<23)+1; // 998244353
```

Estos no surgen mucho en OIA (y cuando sí surgen, el enunciado los menciona), así que no recomiendo ponerlos en la plantilla.

Librerias

Para no andar escribiendo los “include” en todos los archivos, podemos ponerlos en la plantilla. Por ejemplo, podríamos tener en nuestra plantilla:

```
#include <vector>
#include <string>
#include <iostream>

// etc...
```

Una forma alternativa es usar “el bits”, una cabecera que trae todas las bibliotecas de C++ y C. (ojo que tarda más en compilar y no anda en todos lados)

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

Aliases de tipo

En C++ (al igual que en C) podemos ponerle nombres alternativos a los tipos de datos. Esto sirve cuando el nombre real es difícil de tipear, o es muy largo. También está bueno para colapsar nombres de varias palabras en una sola.

Un ejemplo claro es el tipo `long long` (para enteros grandes, hasta 10^{18}). Es largo y son dos palabras, así que usamos un alias para acortarlo.

```
using ll = long long;
```

Otro uso común es para pares o vectores, que tienen < y >.

```
using pii = pair<int, int>;    o    using vi = vector<int>;
```

Macros

Una macro nos permite escribir código que se transforma en otro código.

Las macros más útiles, por lejos, son forr y forn, unas macros que nos ahorran escribir el for a mano.

```
#define forr(i,a,b) for(int i = (a); i < int(b); ++i)
#define forn(i,n)   forr(i,0,n)
```

Por ejemplo, si escribimos forr(pepe, asd+1, v.size()), el compilador interpreta

```
for (int pepe = (asd+1); pepe < int(v.size()); ++pepe)
```

Plantilla recomendada para pruebas oficiales

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

#define forr(i,a,b) for(int i = (a); i < int(b); ++i)
#define forn(i,n) forr(i,0,n)
#define sz(v) int(v.size())

using namespace std;
using ll = long long;
using pii = pair<int,int>;

int main() {
}
```

Plantilla recomendada para práctica online

Esta es la plantilla que uso para ICPC, donde se puede llevar material impreso.

Es más larga porque no hace falta memorizarla, pero no tan larga porque hay que transcribirla a mano.

Hay gente que tiene plantillas aún más largas para usar en competencias por internet (buscar en Codeforces).

https://gist.github.com/SebastianMestre/59e74def2e1a7f431735ea91834bc095

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

#define forr(i,a,b) for(int i=int(a);i<int(b);++i)
#define forn(i,n) forr(i,0,n)
#define dforr(i,a,b) for(int i=int(b)-1;i>=int(a);--i)
#define dforn(i,n) dforr(i,0,n)
#define db(v) cerr<<#v" = "<<(v)<<'\n'
#define vecp(v) cerr<<#v<<" = "; for(auto ee:v)cerr<<ee<<' '; cerr<<'\n'
#define nn cout<<'\n'
#define sz(v) (int(v.size()))
#define all(v) v.begin(), v.end()
#define pb push_back
#define pp pop_back
#define fst first
#define snd second

typedef long long ll; typedef unsigned long long ull;
typedef long double ld; typedef pair<int,int> pii;
typedef pair<ll,ll> pll; typedef vector<ll> vll;

const ll MAXN = 2e5+100;
const ll INF = 1e18+100; const ll MOD = 1e9+7;
const ld EPS = 1e-9; const ld PI = acosl(-1);

int main(){
    //freopen("input.txt", "r", stdin);
    //freopen("output.txt", "w", stdout);
    ios::sync_with_stdio(false); cin.tie(nullptr);
    return 0;
}
```

Yapa, mi plantilla de cuando hacía OIA

```
#include <iostream>

using namespace std;

#define forn(i,n) for(int i = 0; i < int(n); ++i)

using ll = long long;

int main() {
}
```