

# Содержание

<b>1</b>	<b>CLion</b>	<b>1</b>
1.1	Settings . . . . .	1
1.2	Shortcuts . . . . .	1
<b>2</b>	<b>Шаблон</b>	<b>2</b>
<b>3</b>	<b>Общее</b>	<b>3</b>

## 1 CLion

### 1.1 Settings

→ Editor

→ Code Style → C/C++

→ Wrapping and Braces

→ Braces placement → все next line

→ 'if()' statement → 'else' on new line

→ Tabs and Indents → Use tab character → Smart tabs

→ Font → Font: Fira Code → Enable font ligatures (делает красивый шрифт)

→ Build, Execution,...→ CMake → тыкнута на плюсики, список должен стать из Debug и Release, через некоторое время они подтянутся в выпадашку при компиляции

### 1.2 Shortcuts

- Ctrl+Alt+L — форматирование, можно тыкать постоянно
- Shift+F10 — запуск
- Shift+F9 — запуск с дебаггером (надо не забыть сменить на дебаг в выпадашке)
- Ctrl+F9 — компиляция
- Ctrl+F8 — breakpoint
- Ctrl+F7/F8/F9 при дебаге step into/step over/resume
- Ctrl+F2 останавливает программу
- Shift+F6 переименовывает переменную и все ее вхождения
- Ctrl+B переходит к объявлению функции/переменной/класса под курсором или переходит в файл
- Alt+вертикальные стрелочки ходит по функциям
- Alt+горизонтальные стрелочки ходит по файлам
- Alt+цифра открывает/закрывает соответствующую менюшку
- Если дважды нажать на Ctrl, зажать и потыкать на стрелочки вверх/вниз, это сделает мультикурсор, а Escape его потом обратно уберет

## 2 Шаблон

```
1 #include <bits/stdc++.h>
2
3 using namespace std;
4
5 using ll = long long;
6 using ull = unsigned long long;
7
8 mt19937 mt(736);
9
10
11 void solve(istream &cin = std::cin, ostream &cout = std::cout)
12 {}
13
14
15 int main()
16 {
17     ios_base::sync_with_stdio(false);
18     cin.tie(nullptr);
19
20     cout << fixed;
21
22 #ifdef LOCAL
23     auto st = clock();
24
25     ifstream fin("../input.txt");
26
27     do
28     {
29         solve(fin);
30
31         cout << "≡" << endl;
32
33         string str;
34         while (getline(fin, str) && str ≠ string(max(1, (int)str.
           ↪ size()), '='));
35     } while (fin);
36
37     cout << setprecision(6) << "clock: " << double(clock() - st) /
           ↪ CLOCKS_PER_SEC << endl;
38 #else
39     solve();
40 #endif
41
42     return 0;
43 }
```

### 3 Общее

Избегаем глобальных переменных, кроме констант, которые обязательно должны быть `const` (исключение — `mt`). Если есть какая-то глобальная динамика или еще что-нибудь такое, заводим для ее внутренностей класс или хотя бы функцию со `static` переменными.

DRY: Все константы в коде, которые используются несколько раз, должны быть обернуты в константы, код, который используется несколько раз, должен быть обернут в функцию или хотя бы в лямбду

KISS: Избегаем неиспользуемого и длинного кода. Если написали какую-то ерунду и поняли, что ее можно было написать вдвое короче и проще, лучше переписать, потому что это все равно потом дебажить.

Базу индукции в рекурсии лучше разбирать в начале функции, так эту проверку придется писать только один раз и будет легче проверять.

Переменные лучше называть так, чтобы об их назначении можно было догадаться только по их объявлению, писать длинные названия переменных в коде поможет автодополнение (`Ctrl+пробел`).

Если есть, например, таблица размера  $n \times m$  и она хранится в каком-то двумерном векторе, то для получения ее размеров лучше использовать метод `size` этого самого вектора, а не константы  $n$  и  $m$ . Аналогично, если какой-то фор должен идти до конца массива, лучше условие написать до конца массива, а не до  $n$ . Эту уберезет от проблем, когда выясняется, что жизнь, например, будет лучше, если в лабиринт добавить границу, или в граф — пару фиктивных вершин. В идеале, константы типа размера входа должны использоваться только при считывании.