Содержание

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 → тыкнуть на плюсик, список должен стать из Debug и Release, через некоторое время они подтянутся в выпадайку 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 выпадайке)
- 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

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


using namespace std;


using ll = long long;


mt19937 mt(736);


void solve(istream &cin = std::cin, ostream &cout = std::cout)
{}


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

    cout << fixed;

#ifdef LOCAL
    auto st = clock();

    ifstream fin("../input.txt");

    solve(fin);

    cout << "clock: " << setprecision(4) << (clock() - st) / (
        ↪ double) CLOCKS_PER_SEC << endl;
#else
    solve();
#endif

    return 0;
}
```

3 Общее

Избегаем глобальных переменных, кроме констант, кооторые 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 убережет 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.