Желательно следовать рекомендациям, приведённым в статье https://habr.com/ru/post/172091/

1. Длина одной строки не должна превышать 100 символов.
2. Строки не должны заканчиваться пробелами и/или символами табуляции.
3. Каждая строка программы должна содержать не более одного выражения.
4. Имена переменных, функций, классов и т.д. должны быть осмысленными, должны быть английскими словами (не транслитом), должны записываться в стиле camelCase.
5. Имена переменных, функций, свойств и методов начинаются с маленькой буквы, а классов, структур и др. пользовательских типов – с большой.

```
double result = 0.0;
int count = 0;
class Person …
```

6. Имена констант записываются в верхнем регистре, слова разделяются нижними подчёркиваниями.

```
const int SECONDS_IN_HOUR = 60 * 60;
```

7. Не допускается применение «магических значений», вместо этого должны использоваться именованные константы. Исключения: 0, 1 и однократно встречающиеся строковые литералы.
8. Имена функций и методов желательно делать глаголами.

```
int findMax(const int *array, size_t size);
```

9. Переменные должны объявляться в как можно меньшей области видимости и как можно ближе к месту их использования.

10. Переменные желательно объявлять сразу с инициализацией.
11. Аргументы функции, переданные по ссылке или по указателю и не изменяющиеся внутри функции, должны иметь квалификатор `const`.
12. Не следует опускать пространство имён `std::`, это помогает читающему понять, что используемая сущность относится к стандартной библиотеке.
13. Тело ветвления, цикла или функции всегда располагается на новой строке и смещается на один отступ вправо (величина отступа – 2 или 4 пробела).
14. Любой блок кода, заключённый в фигурные скобки (тело цикла, функции, класса и т.п.), должен быть смещён вправо.
15. Разрыв строки должен быть отмечен дополнительным отступом.
16. Открывающая фигурная скобка должна располагаться либо на той же строке, что и оператор, к которому она относится (ветвление, цикл, функция), либо на новой. Два способа в одной программе не смешивать!

```
while (i > 0) {      // 1 способ
    …
}

while (i > 0)        // 2 способ
{
    …
}
```

17. Желательно всегда брать тело ветвления или цикла в фигурные скобки.

```
if (a % 2 == 0)
{
    a /= 2;
}
```

18. Бинарные операторы (присваивания, сравнения, арифметические, логические) должны отделяться пробелами с двух сторон.

```
int a = 2 * (b + c);
```

```
std::cout << a << '\n';
```

19. Ключевые слова `if`, `for`, `while`, `switch`, `catch` всегда отделяются пробелом от последующей открывающей скобки. В то же время список параметров при объявлении, определении и вызове функций никогда не отделяется пробелами.
20. Псевдонимы типов и шаблонов должны быть определены с использованием ключевого слова `using`, а не `typedef`.
21. Секции внутри класса должны быть упорядочены в порядке убывания интереса к ним со стороны читающего код: первой должна идти секция `public`, так как это интерфейс класса для клиентов; затем секция `protected`, поскольку она представляет собой интерфейс для наследников; завершает класс секция `private`, так как эта информация интересна только разработчику класса.
22. Имена полей класса следует заканчивать символом нижнего подчёркивания.
23. Все методы, не меняющие объект логически, должны быть отмечены квалификатором `const`.
24. Список инициализации в конструкторе должен инициализировать не более одной переменной или базового класса на строке. Двоеточие перед списком инициализации должно оставаться строке со списком параметров.

```
Complex::Complex(double re, double im) :
    re_(re),
    im_(im)
{}
```

25. Имена параметров шаблонов должны начинаться с заглавной буквы.
26. Не допускается использование конструкций `using namespace` на верхнем уровне в заголовочных файлах.
27. Заголовочные файлы должны содержать header guards.

```
#ifndef PERSON_H
#define PERSON_H
…
#endif
```

28. Заголовочные файлы не должны содержать определений функций, не отмеченных, как `inline`, допускаются только объявления. То же самое относится к методам, описанным вне тела класса.
29. В программе не должно быть закомментированного кода. Весь неиспользуемый код должен быть удален.