|  |
| --- |
| МИНОБРНАУКИ РОССИИ |
| Федеральное государственное бюджетное образовательное учреждение  высшего образования  **«МИРЭА – Российский технологический университет»**  **РТУ МИРЭА** |
| Институт искусственного интеллекта |
| Кафедра программного обеспечения систем радиоэлектронной аппаратуры |

РЕФЕРАТ

по «Введению в профессиональную деятельность»

на тему: «Стандарты оформления исходного кода на С++»

|  |  |  |  |
| --- | --- | --- | --- |
| Обучающийся | \_\_\_\_\_\_\_\_\_\_\_\_\_ | Кочнов Игорь Олегович \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | |
|  | *Подпись* | *Фамилия Имя Отчество* | |
| Шифр | 22К0009 |  |  |
| Группа | КМБО-02-22 |  |  |
|  |  |  |  |
| Руководитель  работы | \_\_\_\_\_\_\_\_\_\_\_\_\_ | *\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_* | |
|  | *Подпись* | *Фамилия Имя Отчество* | |

Москва 2022

**Оглавление**

[Введение 3](#_Toc121064722)

[Немного о языке 4](#_Toc121064723)

[Наименования 4](#_Toc121064724)

[Файлы исходных кодов 5](#_Toc121064725)

[Выражения 6](#_Toc121064726)

[Разное 9](#_Toc121064727)

[Оформление кода 9](#_Toc121064728)

[Заключение 11](#_Toc121064729)

[Используемая литература 12](#_Toc121064730)

# Введение

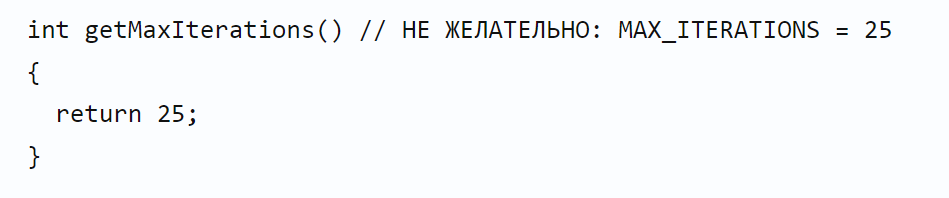
Начинающему разработчику может показаться, что соблюдение стандартов написания кода на любом языке программирования не важно, ведь “Я и так понимаю что у меня написано, зачем мне все это делать, это же долго”. Но тут же они сталкиваются с рядом проблем. Например, захотят они посмотреть свой же код через неделю, и ничего в нем не поймут, так как всё в кучу, отступов нет, непонятно, где начало одной функции, где конец другой. Приходится долго сидеть и разбираться. Бывают ситуации хуже, когда уже другому разработчику приходится лезть в ваш код. Если опытный разработчик не увидит соблюдение стандартов, то он просто не будет смотреть ваш код. Итак, несоблюдение стандартов сильно отразится на пригодноспособности вашего кода, а следовательно, и вас как разработчика. Именно поэтому очень важно разобраться, как делать надо, как не надо, как делать нужно, а как ни в коем случае делать нельзя. Об этом и будет моя работа. Но для начала стоит отметить, что все рекомендации достаточно условные, так как нельзя дать один алгоритм действий на все случаи жизни. Все рекомендации созданы чтобы улучшить читаемость кода. Поэтому программист может пренебрегать некоторыми из них в угоду улучшения внешнего вида кода. Грубо говоря, не стоит опираться только на правила, нужно мыслить гибко.

# Немного о языке

Язык программирования С++ представляет высокоуровневый компилируемый язык программирования общего назначения со статической типизацией, который подходит для создания самых различных приложений. На сегодняшний день С++ является одним из самых популярных и распространенных языков. Почему это так? В первую очередь, он быстрый. Программы компилируются и выполняются быстрее, чем у большинства других языков программирования. Также он универсален и большинтство языков, такие как Java, C#, Java Script и другие построены именно на C++. Поэтому практически каждый программист когда-нибудь работал с ним.

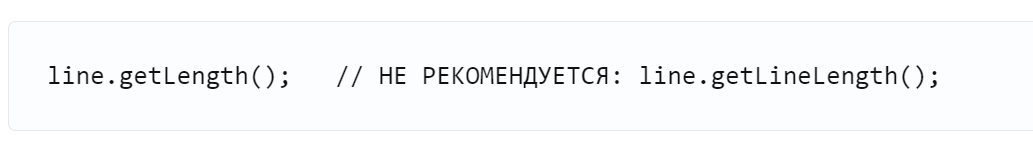
# Наименования

1. Все переменные должны иметь осмысленные названия, например CountOfElements – правильно, а n – неправильно. Это одна из самых важных рекомендаций, она обязательна и не придерживаться ее нельзя, так как это может нести за собой ужасные последствия в виде абсолютного непонимания разработчика, что происходит и как;
2. Все имена должны быть написаны по английски, например nameSurname – правильно, а imyaFamilya – нет, так как программирование – это интернациональная профессия;
3. Имена переменных типов данных должны быть написаны в смешанном регистре, начиная с нижнего, например countOfElements – правильно, а countofelements – нет. Как видно, во втором случае очень тяжело разобрать что это за переменная. В крайнем случае можно сделать так: count\_of\_elements, однако первый вариант приветствуется больше;
4. Константные переменные должны быть записаны в верхнем регистре с нижними подчеркиваниями, например MAX\_ITERATIONS. Однако, использование констант в виде переменных должно быть сведено к минимуму (см. Рис.1);



*Рис. 1*

1. Названия функций должны быть глаголами. Например, getName – правильно, name – неправильно;
2. Аббревиатуры и сокращения в именах должны записываться в нижнем регистре. openDvdPlayer – правильно, openDVDPlayer – неправильно. Во втором варианте Player не выделяется так как нужно, поэтому используют первый;
3. Настраиваемым переменным следует давать то же имя, что и у их типа. Например, void setTopic(Topic\* topic) – правильно, а void setTopic(Topic\* value) – нет, так как второе считается сложнее из-за разнообразия переменных;
4. Переменные, используемые в программе много где, следует называть длинными, понятными именами, а переменные, которые используются в нескольких строчках кода – короткими. Например, переменную, используемую в одном цикле, нужно называть i,j,k,l; не iterations и т.п;
5. Следует избегать указания названий объектов в именах методов. (см. Рис.2);

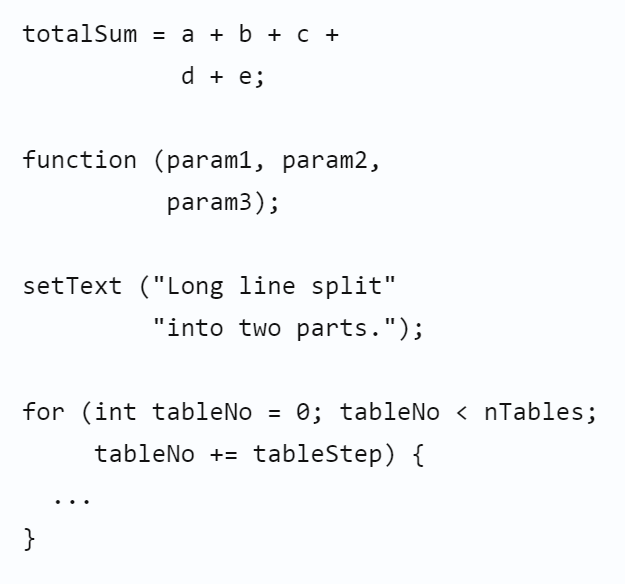


*Рис. 2*

1. При прямом доступе к атрибуту, лучше использовать get/set. Например, array.getElement. Это общая практика разработчиков, ее лучше придерживаться;
2. Слово compute может быть использовано в методах, вычисляющих что-либо. Например, array.computeAverage. Дайте читающему сразу понять, что это времязатратная операция;
3. Слово find может быть использовано в методах, осуществляющих какой-либо поиск**.** Например, array.findMinElement. Дайте читающему понять, что это малозатратная операция.

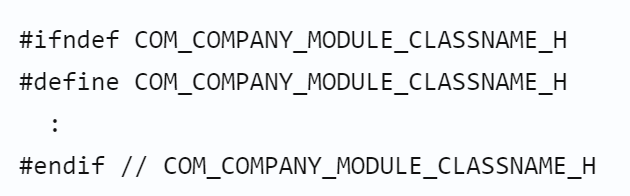
# Файлы исходных кодов

1. Заголовочным файлам следует давать расширение .h , файлам исходных кодов следует давать расширение .c++ ;
2. Все определения должны находиться в файлах исходного кода. Так разработчик, если ему нужно будет найти какое-то определение, будет искать его там где ему и следует находиться;
3. Содержимое файлов не должно превышать 80 колонок. Это стандарт многих редакторов, и так как файлы передаются от одного человека к другому, следует грамотно разбивать строки;
4. Незавершенность разбитых строк должна быть понятна и очевидна. На рисунке 3 приведен ряд примеров. Например, разделять строку после запятой, оператора. Следует выравнивать разбивку по месту, где начались выражения на предыдущей строке;



*Рис.3*

1. Должна быть защита от вложенного включения (см. Рис.4);



*Рис. 4*

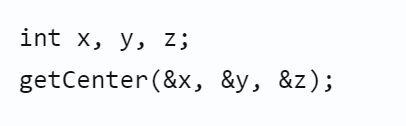
# Выражения

1. Локальные типы, используемые в одном файле, должны быть объявлены только в нём;
2. Приведение типов должно быть явным (см. Рис.5) ;



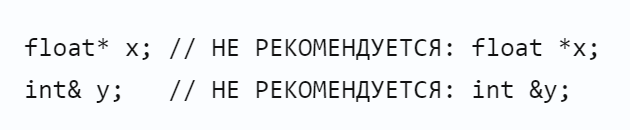
*Рис.5*

1. Следует инициализировать переменные в местах их объявления. Это позволяет использовать их в любой момент и не задумываться о проблемах. Иногда это сделать не получается, например как на рисунке 6. Тогда лучше оставить переменные неинициализированными;



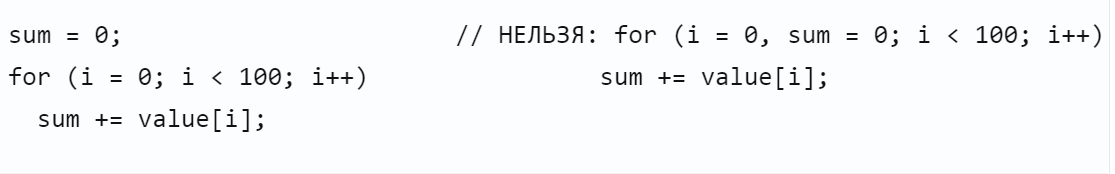
*Рис.6*

1. Переменные всегда должны иметь однозначный смысл, разночтения недопустимы;
2. Следует избегать использования глобальных переменных. Почти всегда можно обойтись без них;
3. Символ указателя стоит ставить сразу после типа, а не перед переменной. Пример на рисунке 7;



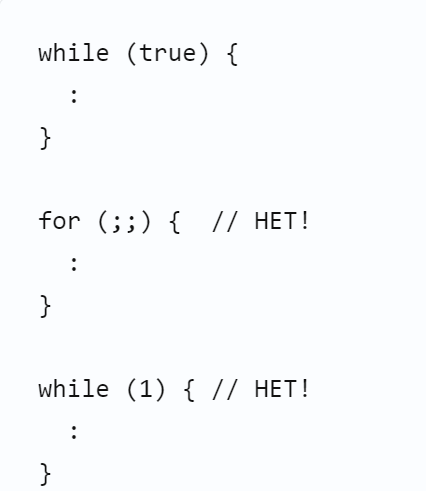
*Рис.7*

1. Не стоит включать в цикл выражения, не относящиеся к управляению цикла(см. Рис. 8);



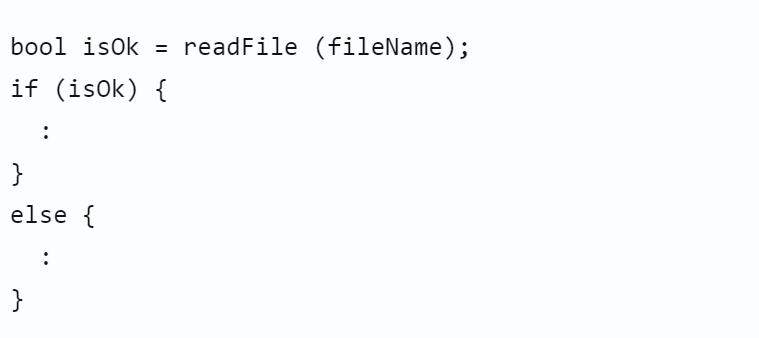
*Рис.8*

1. Следует избегать циклов do – while. Они ухудшают читаемость;
2. Лучше не использовать break и continue, если это не улучшает читаемость;
3. Бесконечный цикл делать так, как на рисунке 9. Остальные формы следует избегать;



*Рис.9*

11)Ожидаемую часть следует располагать в части if, исключение — в части else (см. Рис. 10). Это позволяет убедиться, что исключения не вносят неясности в нормальный ход выполнения. Важно для читаемости и производительности.



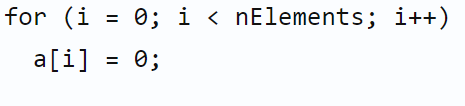
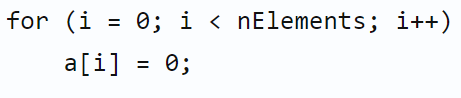
*Рис.10*

# Разное

1. Следует избегать использования любых чисел кроме 0 и 1. Если не понятно, откуда это число, следует делать именованные константы;
2. При работе с числами с плавающей точкой, следует писать их через точку и указывать как минимум один знак после запятой. Так становится очевидна разница между такими числами и целыми, ведь это совершенно разные модели.

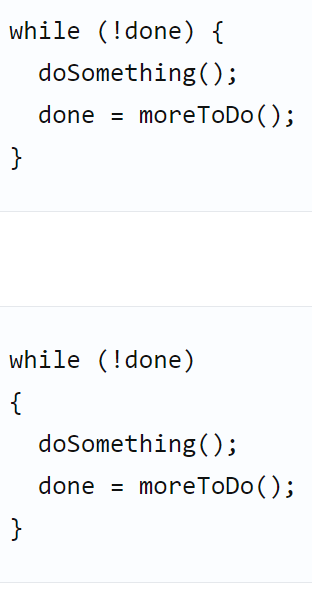
# Оформление кода

1. Следует делать отступ в 2 или 4 пробела. Меньше – ухудшает читаемость, больше – увеличивается вероятность того, что придется разбивать строки в глубоко вложенном коде. Пример на рисунке 11;

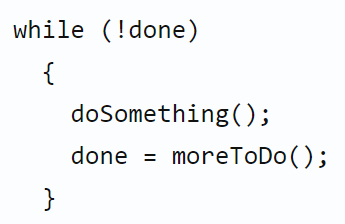
 

*Рис. 11*

1. Блоки кода следует оформлять как на рисунке 12, но ни в коем случае не как на рисунке 13, там используются лишние пробелы, что может вызвать проблемы;

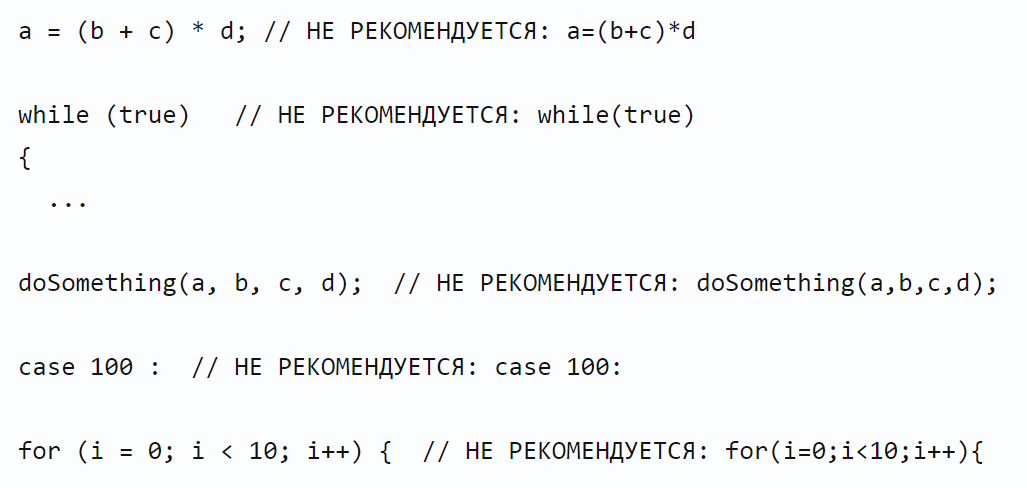


*Рис.12*



*Рис.13*

1. Пробелы. Операторы следует отбивать пробелами. После зарезервированных ключевых слов языка C++ следует ставить пробел. После запятых следует ставить пробелы. Двоеточия следует отбивать пробелами. После точек с запятой в цикле *for* следует ставить пробелы. Пример показан на рисунке 14.



*Рис.14*

# Заключение

Итак, правил по написанию кода достаточно много, есть общие, которые есть почти в каждом проекте, а есть и такие, которые возможно никогда не придется использовать разработчику. В своём реферате я сосредоточился на именно на первых, чтобы дать общее понимание стандартов. Как видно, несоблюдение стандартов может привести к недопониманию, а его не должно быть в работе, которая опирается на взаимодействие многих людей.

# 

# Используемая литература

1. Рейзлин В.И. Р35 Язык С++ и программирование на нём: учебное пособие / В.И. Рейзлин ; Томский политехнический университет. – 3-е изд., перераб. – Томск : Изд-во Томского политехнического университета, 2021. – 208 с
2. <https://habr.com/ru/post/172091/>
3. <https://metanit.com/cpp/tutorial/1.1.php>
4. <https://ru.wikipedia.org/wiki/C%2B%2B>
5. <https://vc.ru/hr/50161-pochemu-c-krut-aktualen-i-bessmerten>