

Ravesli [Ravesli](#)


- [Уроки по C++](#)
- [OpenGL](#)
- [SFML](#)
- [Qt5](#)
- [RegExр](#)
- [Ассемблер](#)
- [Купить .PDF](#)


Урок №9. Комментарии

 [Юрий](#) |

- [Уроки C++](#)

|

 Обновл. 19 Сен 2020 |

 86288

[1](#)  17

Комментарий — это строка (или несколько строк) текста, которая вставляется в исходный код для объяснения того, что делает код. В языке C++ есть 2 типа комментариев: однострочные и многострочные.

Оглавление:

1. [Однострочные комментарии](#)
2. [Многострочные комментарии](#)
3. [Как правильно писать комментарии?](#)
4. [Закомментировать код](#)

Однострочные комментарии

Однострочные комментарии — это комментарии, которые пишутся после символов `//`. Они пишутся в отдельных строках и всё, что находится после этих символов комментирования, — игнорируется компилятором, например:

```
1 | std::cout << «Hello, world!» << std::endl; // всё, что находится справа от двойного
```

Как правило, однострочные комментарии используются для объяснения одной строчки кода:

```
1 | std::cout << «Hello, world!» << std::endl; // cout и endl находятся в библиотеке ios
2 | std::cout << «It is so exciting!» << std::endl; // эти комментарии усложняют чтение
3 | std::cout << «Yeah!» << std::endl; // особенно, когда строки разной длины
```

Размещая комментарии справа от кода, мы затрудняем себе как чтение кода, так и чтение комментариев. Следовательно, однострочные комментарии лучше размещать над строками кода:

```
1 // cout и endl находятся в библиотеке iostream
2 std::cout << «Hello, world!» << std::endl;
3
4 // теперь уже легче читать
5 std::cout << «It is so exciting!» << std::endl;
6
7 // не так ли?
8 std::cout << «Yeah!» << std::endl;
```

Многострочные комментарии

Многострочные комментарии — это комментарии, которые пишутся между символами `/* */`. Всё, что находится между звёздочками, — игнорируется компилятором:

```
1 /* Это многострочный комментарий.
2 Эта строка игнорируется
3 и эта тоже. */
```

Так как всё, что находится между звёздочками, — игнорируется, то иногда вы можете наблюдать следующее:

```
1 /* Это многострочный комментарий.
2 * Звёздочки слева
3 * упрощают чтение текста
4 */
```

Многострочные комментарии не могут быть вложенными (т.е. одни комментарии внутри других):

```
1 /* Это многострочный /* комментарий */ а это уже не комментарий */
2 // Верхний комментарий заканчивается перед первым */, а не перед вторым */
```

Правило: Никогда не используйте вложенные комментарии.

Как правильно писать комментарии?

Во-первых, на уровне библиотек/программ/функций комментарии отвечают на вопрос «**ЧТО?**»: «Что делают эти библиотеки/программы/функции?». Например:

```
1 // Эта программа вычисляет оценку студента за семестр на основе его оценок за модули
2
3 // Эта функция использует метод Ньютона для вычисления корня функции
4
5 // Следующий код генерирует случайное число
```

Все эти комментарии позволяют понять, что делает программа, без необходимости смотреть на исходный код. Это особенно важно специалистам, работающим в команде, где не каждый специалист будет знаком со всем имеющимся кодом.

Во-вторых, внутри библиотек/программ/функций комментарии отвечают на вопрос «**КАК?**»: «Как код выполняет задание?». Например:

```
1 /* Для расчета итоговой оценки ученика, мы складываем все его оценки за уроки и дома  
2    а затем делим получившееся число на общее количество оценок.  
3    Таким образом, мы получаем средний балл ученика. */
```

Или:

```
1 // Чтобы получить рандомный (случайный) элемент, мы выполняем следующее:  
2  
3 // 1) Составляем список всех элементов.  
4 // 2) Вычисляем среднее значение для каждого элемента, исходя из его веса, цвета и ц  
5 // 3) Выбираем любое число.  
6 // 4) Определяем соответствие элемента случайно выбранному числу.  
7 // 5) Возвращаем случайный элемент.
```

Эти комментарии позволяют читателю понять, каким образом код выполняет поставленное ему задание.

В-третьих, на уровне стейтментов (однострочного кода) комментарии отвечают на вопрос «ПОЧЕМУ?»: «Почему код выполняет задание именно так, а не иначе?». Плохой комментарий на уровне стейтментов объясняет, что делает код. Если вы когда-нибудь писали код, который был настолько сложным, что нужен был комментарий, который бы объяснял, что он делает, то вам нужно было бы не писать комментарий, а переписывать этот код.

Примеры плохих и хороших однострочных комментариев:

👎 Плохой комментарий:

```
1 // Присваиваем переменной sight значение 0  
2 sight = 0;
```

(По коду это и так понятно)

👍 Хороший комментарий:

```
1 // Игрок выпил зелье слепоты и ничего не видит  
2 sight = 0;
```

(Теперь мы знаем, ПОЧЕМУ зрение у игрока равно нулю)

👎 Плохой комментарий:

```
1 // Рассчитываем стоимость элементов  
2 cost = items / 2 * storePrice;
```

(Да, мы видим, что здесь подсчет стоимости, но почему элементы делятся на 2?)

👍 Хороший комментарий:

```
1 // Нам нужно разделить все элементы на 2, потому что они куплены по парам  
2 cost = items / 2 * storePrice;
```

(Теперь понятно!)

Программистам часто приходится принимать трудные решения по поводу того, каким способом решить проблему. А комментарии и существуют для того, чтобы напомнить себе (или объяснить другим) причину, почему вы написали код именно так, а не иначе.

👍 Хорошие комментарии:

```
1 // Мы решили использовать список вместо массива,  
2 // потому что массивы осуществляют медленную вставку.
```

Или:

```
1 // Мы используем метод Ньютона для вычисления корня функции,  
2 // так как другого детерминистического способа решения этой задачи - нет.
```

И, наконец, комментарии нужно писать так, чтобы человек, который не имеет ни малейшего представления о том, что делает ваш код — смог в нем разобраться. Очень часто случаются ситуации, когда программист говорит: «Это же совершенно очевидно, что делает код! Я это точно не забуду!». Угадайте, что случится через несколько недель или даже дней? Это не совершенно очевидно, и вы удивитесь, как скоро вы забудете то, что делает ваш код. Вы (или кто-то другой) будете очень благодарны себе за то, что оставите комментарии, объясняя на человеческом языке что, как и почему делает ваш код. Читать отдельные строки кода — легко, понимать их логику и смысл — сложно.

Подытожим:

- ➔ На уровне библиотек/программ/функций оставляйте комментарии, отвечая на вопрос «*ЧТО?*».
- ➔ Внутри библиотек/программ/функций оставляйте комментарии, отвечая на вопрос «*КАК?*».
- ➔ На уровне стейтментов оставляйте комментарии, отвечая на вопрос «*ПОЧЕМУ?*».

Закомментировать код

Закомментировать код — это конвертировать одну или несколько строк кода в комментарии. Таким образом, вы можете (временно) исключить часть кода из компиляции.

Чтобы закомментировать одну строку кода, используйте однострочные символы комментирования `//`.

Не закомментировано:

```
1 std::cout << 1;
```

Закомментировано:

```
1 // std::cout << 1;
```

Чтобы закомментировать блок кода, используйте однострочные символы комментирования `//` на каждой строке или символы многострочного комментария `/* */`.

Не закомментировано:

```
1 std::cout << 1;  
2 std::cout << 2;
```

```
3 | std::cout << 3;
```

Закомментировано символами однострочного комментария:

```
1 | // std::cout << 1;
2 | // std::cout << 2;
3 | // std::cout << 3;
```

Закомментировано символами многострочного комментария:

```
1 | /*
2 |     std::cout << 1;
3 |     std::cout << 2;
4 |     std::cout << 3;
5 | */
```

Есть несколько причин, почему следует использовать «закомментирование»:

- ➔ **Причина №1:** Вы работаете над новой частью кода, которая пока что не рабочая, но вам нужно запустить программу. Компилятор не позволит выполнить программу, если в ней будут ошибки. Временное отделение нерабочего кода от рабочего комментированием позволит вам запустить программу. Когда код будет рабочий, то вы сможете его легко раскомментировать и продолжить работу.
- ➔ **Причина №2:** Вы написали код, который компилируется, но работает не так, как нужно и сейчас у вас нет времени с этим возиться. Закомментируйте код, а затем, когда будет время, исправьте ошибки.
- ➔ **Причина №3:** Поиск корня ошибки. Если вас не устраивают результаты работы программы (или вообще происходит сбой), полезно будет поочерёдно «отключать» части вашего кода, чтобы понять какие из них рабочие, а какие — создают проблемы. Если вы закомментируете одну или несколько строчек кода и программа начнет корректно работать (или пропадут сбои), шансы того, что последнее, что вы закомментировали, является ошибкой — очень велики. После этого вы сможете разобраться с тем, почему же этот код не работает так, как нужно.
- ➔ **Причина №4:** Тестирование нового кода. Вместо удаления старого кода, вы можете его закомментировать и оставить для справки, пока не будете уверены в том, что ваш новый код работает так, как нужно. Как только вы будете уверены в новом коде, то сможете без проблем удалить старые фрагменты кода. Если же новый код у вас будет работать не так, как нужно, то вы сможете его удалить и откатиться к старому коду.

Примечание: Во всех следующих уроках я буду использовать комментарии в иллюстративных целях. Внимательные читатели смогут заметить, что по вышеуказанным стандартам большинство из этих комментариев будут плохими. Но помните, что использовать я их буду в образовательных целях, а не для демонстрации хороших примеров.

Оценить статью:

★★★★★ (807 оценок, среднее: 4,95 из 5)



[← Урок №8. Структура программ](#)[Урок №10. Переменные, Инициализация и Присваивание](#)

Комментариев: 17



1. *Юрий:*

[7 июля 2020 в 17:32](#)

Спасибо Вам за этот прекрасный и доступный курс. Я сам преподаватель информатики и этот курс читаю с огромным удовольствием!!! Спасибо Вам за Ваш громадный и прекрасный курс!!!

[Ответить](#)



1. *Юрий:*

[7 июля 2020 в 18:09](#)

Пожалуйста 😊

[Ответить](#)

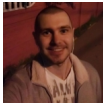


2. *Александр:*

[20 сентября 2019 в 22:01](#)

А если необходимо символы /* */ записать в строковую переменную, то как это сделать правильно, не сломав его?

[Ответить](#)



1. *Дмитрий Бушуев:*

[23 сентября 2019 в 11:36](#)

Используйте символ обратного слэша — \
Например:

```
1  std::string name("\\* *\\");
2  std::cout << name;
3
4  //Или можно сразу вывести на печать
5  std::cout << "\\* *\\";
```

[Ответить](#)



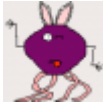
3. *АРТЕМ:*

[22 июня 2019 в 20:04](#)

Зачем заключать нерабочий код в комментарий?

Помойму легче скопировать его и куда нибудь сохранить.. Потом достать вставить и доделать

[Ответить](#)



1. *Сергей:*

[9 июля 2019 в 19:01](#)

1)чтоб был под носом (никуда его не переносить)

2)удобство — гораздо проще кликнуть пару слешей или обозначать область /* */ . Чем держать в дампе или переносить куда-либо.

[Ответить](#)



4. *Андрей:*

[16 мая 2019 в 22:17](#)

Мне 50, я одинок, имею кучу времени, некую давнюю мечту — что то сделать самому... — И вот: C++ мне в помощь!!

Очень благодарен Автору! Вы даёте импульс жить дальше!.

[Ответить](#)



1. *Юрий:*

[6 июня 2019 в 15:13](#)

Приятно, что оказался вам полезен 😊

[Ответить](#)



5. *Demien:*

[30 марта 2019 в 18:16](#)

Ты забыл о третьем виде комментариев — тройной слеш.

[Ответить](#)



6. *Александр Казаков:*

[30 марта 2019 в 17:10](#)

Новое понимание комментария. Спасибо! Это другой взгляд или взгляд с другой стороны.

[Ответить](#)



7. *Testik:*

[26 февраля 2019 в 16:22](#)

В третьей части урока мы создаём второй уровень для игры Кот в лабиринте и закрепляем команды Scratch для реализации движения спрайтов по координатам.

[Ответить](#)

8. Эзиз:

[19 февраля 2019 в 21:31](#)

Самая легкая тема в программировании))))

[Ответить](#)

1. Наталья:

[23 сентября 2020 в 12:54](#)

Написание хороших комментариев — одна из наиболее сложных задач в программировании, а отнюдь не самая лёгкая. Спасибо автору за грамотное изложение того, что именно надо писать в комментариях, и за наглядные примеры.

[Ответить](#)

9. Feliz:

[10 февраля 2019 в 19:06](#)

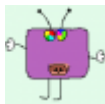
А почему "ravesli" ?

[Ответить](#)

1. Юрий:

[10 февраля 2019 в 22:56](#)

Просто так. Название ни к чему не привязано.

[Ответить](#)

1. Георгий:

[24 июня 2019 в 20:44](#)

Я то думаю что так сложно запомнить название , проще курс запомнить

[Ответить](#)

10. Денис:

[14 января 2019 в 18:04](#)

Спасибо, узнал, что можно "закомментировать"

[Ответить](#)

Добавить комментарий

Ваш E-mail не будет опубликован. Обязательные поля помечены *

Имя *

Email *

Комментарий

☐ Сохранить моё Имя и E-mail. Видеть комментарии, отправленные на модерацию






☐ Получать уведомления о новых комментариях по электронной почте. Вы можете [подписаться](#) без комментирования.

[TELEGRAM](#)  [КАНАЛ](#)



[ПАБЛИК](#) 

ТОП СТАТЬИ

-  [Словарь программиста. Сленг, который должен знать каждый кодер](#)
-  [Урок №1. Введение в программирование](#)
-  [70+ бесплатных ресурсов для изучения программирования](#)
-  [Урок №1: Введение в создание игры «Same Game»](#)
-  [Урок №4. Установка IDE \(Интегрированной Среды Разработки\)](#)

- [Ravesli](#)
- - [О проекте](#) -
- - [Пользовательское Соглашение](#) -
- - [Все статьи](#) -
- Copyright © 2015 - 2020