

Ravesli [Ravesli](#)


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


## Урок №10. Переменные, Инициализация и Присваивание

 [Юрий](#) |

- [Уроки C++](#)

|

 Обновл. 29 Авг 2020 |

 95758

[1](#)  37

Программируя на языке C++, мы создаем, обрабатываем и уничтожаем объекты. **Объект** — это часть памяти, которая может хранить значение. В качестве аналогии мы можем использовать почтовый ящик, куда мы помещаем информацию и откуда её извлекаем. Все компьютеры имеют **оперативную память**, которую используют программы. При создании объекта, часть оперативной памяти выделяется для этого объекта. Большинство объектов, с которыми мы будем работать в языке C++, являются переменными.

Оглавление:

1. [Переменные](#)
2. [l-values и r-values](#)
3. [Инициализация vs. Присваивание](#)
4. [Неинициализированные переменные](#)
5. [Тест](#)
6. [Ответы](#)

## Переменные

**Стейтмент** `a = 8;` выглядит довольно простым: мы присваиваем значение 8 переменной `a`. Но что такое `a`? `a` — это переменная, объект с именем.

На этом уроке мы рассмотрим только целочисленные переменные. **Целое число** — это число, которое можно записать без дроби, например: -11, -2, 0, 5 или 34.

Для создания переменной используется стейтмент объявления (разницу между объявлением и определением переменной мы рассмотрим несколько позже). Вот пример объявления целочисленной переменной `a` (которая может содержать только целые числа):

```
1 | int a;
```

При выполнении этой инструкции центральным процессором часть оперативной памяти выделяется под этот объект. Например, предположим, что переменной `a` присваивается ячейка памяти под номером 150. Когда программа видит переменную `a` в выражении или в стейтменте, то она понимает, что для того, чтобы получить значение этой переменной, нужно заглянуть в ячейку памяти под номером 150.

Одной из наиболее распространенных операций с переменными является операция присваивания, например:

```
1 | a = 8;
```

Когда процессор выполняет эту инструкцию, он понимает её как «поместить значение 8 в ячейку памяти под номером 150».

Затем мы сможем вывести это значение на экран с помощью `std::cout`:

```
1 | std::cout << a; // выводим значение переменной a (ячейка памяти под номером 150) на экран
```

## l-values и r-values

В языке C++ все переменные являются l-values. **l-value** (в переводе «л-значение», произносится как «ел-ва́лю») — это значение, которое имеет свой собственный адрес в памяти. Поскольку все переменные имеют адреса, то они все являются l-values (например, переменные `a`, `b`, `c` — все они являются l-values). От слова «left», так как только значения l-values могут находиться в левой стороне в операциях присваивания (в противном случае, мы получим ошибку). Например, стейтмент `9 = 10`; вызовет ошибку компилятора, так как 9 не является l-value. Число 9 не имеет своего адреса в памяти и, таким образом, мы ничего не можем ему присвоить (`9 = 9` и ничего здесь не изменить).

Противоположностью l-value является r-value (в переводе «р-значение», произносится как «ер-ва́лю»). **r-value** — это значение, которое не имеет постоянного адреса в памяти. Примерами могут быть единичные числа (например, 7, которое имеет значение 7) или выражения (например, `3 + x`, которое имеет значение `x` плюс 3).

Вот несколько примеров операций присваивания с использованием r-values:

```
1 | int a;           // объявляем целочисленную переменную a
2 | a = 5;           // 5 имеет значение 5, которое затем присваивается переменной a
3 | a = 4 + 6;       // 4 + 6 имеет значение 10, которое затем присваивается переменной a
4 |
5 | int b;           // объявляем целочисленную переменную b
6 | b = a;           // a имеет значение 10 (исходя из предыдущих операций), которое затем присваивается переменной b
7 | b = b;           // b имеет значение 10, которое затем присваивается переменной b (ничего не меняется)
8 | b = b + 2;       // b + 2 имеет значение 12, которое затем присваивается переменной b
```

Давайте детально рассмотрим последнюю операцию присваивания:

```
1 | b = b + 2;
```

Здесь переменная `b` используется в двух различных контекстах. Слева `b` используется как l-value (переменная с адресом в памяти), а справа `b` используется как r-value и имеет отдельное значение (в данном случае, 12). При выполнении этого стейтмента, компилятор видит следующее:

```
1 | b = 10 + 2;
```

И здесь уже понятно, какое значение присваивается переменной `b`.

Сильно беспокоиться о l-values или r-values сейчас не нужно, так как мы еще вернемся к этой теме на следующих уроках. Всё, что вам нужно сейчас запомнить — это то, что в левой стороне операции присваивания всегда должно находиться l-value (которое имеет свой собственный адрес в памяти), а в правой стороне операции присваивания — r-value (которое имеет какое-то значение).

## Инициализация vs. Присваивание

В языке C++ есть две похожие концепции, которые новички часто путают: присваивание и инициализация.

После объявления переменной, ей можно **присвоить** значение с помощью оператора присваивания (знак равенства `=`):

```
1 | int a; // это объявление переменной
2 | a = 8; // а это присваивание переменной a значения 8
```

В языке C++ вы можете объявить переменную и присвоить ей значение одновременно. Это называется **инициализацией** (или «*определением*»).

```
1 | int a = 8; // инициализируем переменную a значением 8
```

Переменная может быть инициализирована только после операции объявления.

Хотя эти два понятия близки по своей сути и часто могут использоваться для достижения одних и тех же целей, все же в некоторых случаях следует использовать инициализацию, вместо присваивания, а в некоторых — присваивание вместо инициализации.

**Правило:** Если у вас изначально имеется значение для переменной, то используйте инициализацию, вместо присваивания.

## Неинициализированные переменные

В отличие от других языков программирования, языки Си и C++ не инициализируют переменные определенными значениями (например, нулем) по умолчанию. Поэтому, при создании переменной, ей присваивается ячейка в памяти, в которой уже может находиться какой-нибудь мусор! Переменная без значения (со стороны программиста или пользователя) называется **неинициализированной переменной**.

Использование неинициализированных переменных может привести к ошибкам, например:

```
1 | #include <iostream>
2 |
3 | int main()
```

```
4 {  
5     // Объявляем целочисленную переменную a  
6     int a;  
7  
8     // Выводим значение переменной a на экран (a - это неинициализированная переменная)  
9     std::cout << a;  
10  
11     return 0;  
12 }
```

В этом случае компилятор присваивает переменной *a* ячейку в памяти, которая в данный момент свободна (не используется). Затем значение переменной *a* отправляется на вывод. Но что мы увидим на экране? Ничего, так как компилятор это не пропустит — выведется ошибка, что переменная *a* является неинициализированной. В более старых версиях Visual Studio компилятор вообще мог бы вывести какое-то некорректное значение (например, 7177728, т.е. мусор), которое было бы содержимым той ячейки памяти, которую он присвоил нашей переменной.

Использование неинициализированных переменных является одной из самых распространенных ошибок начинающих программистов, но, к счастью, большинство современных компиляторов выдадут ошибку во время компиляции, если обнаружат неинициализированную переменную.

Хорошей практикой считается всегда инициализировать свои переменные. Это будет гарантией того, что ваша переменная всегда имеет определенное значение и вы не получите ошибку от компилятора.

**Правило: Убедитесь, что все ваши переменные в программе имеют значения (либо через инициализацию, либо через операцию присваивания).**

## Тест

Какой результат выполнения следующих стейтментов?

```
1 int a = 6;  
2 a = a - 3;  
3 std::cout << a << std::endl; // №1  
4  
5 int b = a;  
6 std::cout << b << std::endl; // №2  
7  
8 // В этом случае a + b является r-value  
9 std::cout << a + b << std::endl; // №3  
10  
11 std::cout << a << std::endl; // №4  
12  
13 int c;  
14 std::cout << c << std::endl; // №5
```

## Ответы

Чтобы просмотреть ответ, кликните на него мышкой.

**Ответ №1**

Программа выведет 3:  $a - 3 = 3$ , что и присваивается переменной  $a$ .

**Ответ №2**

Программа выведет 3: переменной  $b$  присваивается значение переменной  $a$  (3).

**Ответ №3**

Программа выведет 6:  $a + b = 6$ . Здесь не используется операция присваивания.

**Ответ №4**

Программа выведет 3: значением переменной  $a$  до сих пор является 3.

**Ответ №5**

Результатом будет ошибка, так как  $c$  — это неинициализированная переменная.

Оценить статью:



(906 оценок, среднее: 4,93 из 5)



[← Урок №9. Комментарии](#)

[Урок №11. cout, cin и endl](#)



## Комментариев: 37



1. *Миха:*

[26 сентября 2020 в 02:11](#)

"Объект — это часть памяти, которая может хранить значение. В качестве аналогии мы можем использовать почтовый ящик, куда мы помещаем информацию и откуда её извлекаем. Все компьютеры имеют оперативную память, которую используют программы. При создании объекта, часть оперативной памяти выделяется для этого объекта. Большинство объектов, с которыми мы будем работать в языке C++, являются переменными."

Хмм. насколько верно, что переменная — это объект? Это минимальная единица манипулирования объектом (переменная — ссылка на ячейку памяти.). Если рассматривать вашу аналогию, то объект-почтовый ящик, а переменная — письмо (Хотя и письмо — тоже объект. А вот буква в письме — переменная).

[Ответить](#)



2. *Сергей:*

[21 мая 2020 в 02:41](#)

Простите , может, за глупый вопрос ( только начал изучать) , а почему одной и той же переменной необходимо присваивать разные значения ?

```
1 | int a;      // объявляем целочисленную переменную a
2 | a = 5;      // 5 вычисляется в 5, которое затем присваивается переменной a
3 | a = 4 + 6;  // 4 + 6 вычисляется в 10, которое затем присваивается переменной a
```

#### [Ответить](#)



1. Нурис:

[26 мая 2020 в 22:38](#)

Пните когда ответят

#### [Ответить](#)



1. Сергей:

[11 июня 2020 в 20:00](#)

Пнул. Ответили. Не прошло и ....))))

#### [Ответить](#)



2. Адольф:

[9 июня 2020 в 12:21](#)

Это не необходимость, просто демонстрация некой возможности, переменным можно всегда присваивать новые значения, а можно и не присваивать. Так например, если для разных задач нужна 1 переменная, можно просто менять ее значение для каждой операции, а не создавать новую.

#### [Ответить](#)



3. Владислав:

[6 июля 2020 в 15:22](#)

Читай комментарий(5 вычисляется в 5, которое затем присваивается переменной a). Это для наглядности сделано, типа что бы ты понял как присвоить значение. Но на практике такое особо не делается без вывода. Если не понятно то что я сказал то напиши `cout <<a<<endl;` после(`a = 5;`) и напиши `cout <<a <<endl;` после (`a = 4 + 6;`)

#### [Ответить](#)



3. ХейЛонг:

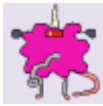
[22 марта 2020 в 23:37](#)

Левое значение, оперативка. Сразу видно чёткого программера, не заморачивающегося стилистикой речи. Но это я так, не по теме =) По крайней мере, с начала изучения твоего ресурса я не испытываю ощутимого дискомфорта в понимании материала. Надеюсь, эта тенденция сохранится впредь.

[Ответить](#)4. *Дима:*[17 июля 2019 в 14:52](#)

Я написал такой код но на первой скобке пишет "требуется объявление" и "отсутствует заголовок функции" как исправить

```
1  #include <iostream>
2
3  int main();
4
5  {
6
7      int a;
8      int b;
9      int c;
10     a = 1;
11     b = 2;
12     b + a = c;
13     std::cout << "c" ;
14     return 0;
15
16 }
```

[Ответить](#)1. *Андрей:*[9 августа 2019 в 14:53](#)

Для того, чтобы присвоить 'c' выражение 'a+b' нужно сперва указать переменную, в которую поместишь выражение: `c = a + b;`

А для того чтобы вывести переменную нужно: `std::cout<<"c= "<<c;`

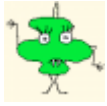
[Ответить](#)2. *Дороу:*[22 августа 2019 в 09:25](#)

После `main()` точка с запятой не нужна(

[Ответить](#)3. *Жека:*[27 декабря 2019 в 23:55](#)

ты мог бы просто убрать кавычки. И объявить переменную. Кавычки нужны для вывода текста на экран.

[Ответить](#)



5. *Алексей:*

[19 июня 2019 в 17:03](#)

Все отлично, было 13 лет назад баловался на С. Универ требовал и надо было.

Сейчас вспоминаю, изучаю.

Пишу эти простые проги на Ubuntu, неинициализированная переменная при выводе выдаст 0.

[Ответить](#)



1. *SagePtr:*

[8 октября 2019 в 08:44](#)

Далеко не всегда 0. Тут как повезёт. Локальные переменные не инициализируются точно. Если у вас всегда 0, возможно, проверяете на какой-нибудь простецкой программе, которую компилятор вполне мог заоптимизировать и поменять переменную на константу, видя, что с ней ничего кроме вывода на экран не происходит.

[Ответить](#)



6. *Dulat:*

[29 апреля 2019 в 09:15](#)

Храни тебя господь, Юрий!)

Благодарю вас урок очень понравился, так доступно объясняете.

[Ответить](#)



1. *Юрий:*

[29 апреля 2019 в 11:47](#)

Спасибо, что читаете 😊

[Ответить](#)



7. *Nub:*

[26 апреля 2019 в 03:01](#)

[quote]

```
1 | b = b + 2; // b + 2 вычисляется в 12, которое затем присваивается переменной b
```

При выполнении этого стейтмента, компилятор видит следующее:

```
1 | b = 10 + 2;
```

[/quote]

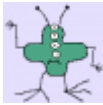
Откуда компилятор видит, что `b == 10`?

То, что `b` равно 10 будет известно только в рантайме, нет?

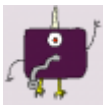


[Ответить](#)8. *Rizo:*[21 апреля 2019 в 20:53](#)

Скажите пожалуйста. Задумался об изучении программирования, учусь в техунивере по профессии программная-инженерия. В приоритете такие языки как C++ и JavaScript. скажите пожалуйста, что лучше изучить сначала. Если C++ то за какое время смогу освоить ее нормально. JS рассматриваю в качестве быстро устроиться на работу, т. к. стипендия маленькая. Нужно как ты жить? Спасибо!

[Ответить](#)1. *jaroslavez:*[25 апреля 2019 в 19:25](#)

Освой html+css+js и устройся в какую-нибудь конторку верстальщиком на небольшую зарплату (если Москва, то минимум 20 тысяч, на еду и одежду с рынка хватит, но если ты квартиру снимаешь, то надо исходить из стоимости аренды). Если ты беспокоишься о "порче" мозгов из-за изучения js, после которого будет, якобы, сложнее изучить C++ (лично меня этим пугали), то нет, ничего подобного.

[Ответить](#)2. *Nub:*[26 апреля 2019 в 03:25](#)

Чувак, если тебе нужно на что-то жить, то устройся лучше в МакДональдс. Я абсолютно серьезно.

Чтобы зарабатывать деньги программированием нужно быть профи, с опытом, для C++ с БОЛЬШИМ опытом. Опыта набираются на бесплатных работах: контрибути в опенсорс (это если согласятся еще принять) или придумывают себе свой проект.

Верстальщики (а они вообще нужны кому-нибудь в 2019?) не исключение, просто путь покороче. Но это путь в никуда.

Так что лучше подрабатывай в Маке, а по ночам пили свой проект. Советую начать с Python — он очень универсален.

Сейчас большой хайп вокруг machine learning, вот там реально найти работу юнцу, т.к. тема новая и седовласых просто нет. Про сайтики лучше забыть — эта поляна давно вытоптана

[Ответить](#)1. *никто:*[19 июля 2019 в 16:06](#)

груСНые реалии суровой жизни

[Ответить](#)

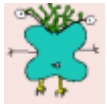
9. *mB13uk:*[30 марта 2019 в 09:19](#)

а есть ли разница в написании инициализации? ну может быть время выполнения или объем кода?

```
1 int a = 16;
```

или

```
1 int b{16};
```

[Ответить](#)10. *Rikko:*[29 марта 2019 в 16:40](#)

А можно написать вначале "using namespace std", чтобы при записи кода не писать "std"?

[Ответить](#)11. *Алексей:*[20 января 2019 в 23:52](#)

Программа выведет 3.  $a-3$  равно 3, что и присваивается переменной  $a$ . Нужно бы пересмотреть ответы.  $a = 6$ .  $6 - 3 = 3$ . Не подготовленный пользователь, не всегда может понять.

[Ответить](#)1. *Константин:*[24 февраля 2019 в 05:59](#)

...нет электричества — нет никаких битов-байтов-переменных-постоянных... Подали питание — появляются напряжения на кристаллах оперативной памяти, скажем, 1 мВ соответствуют так называемому двоичному "0", а 3 мВ — двоичной "1". Таким образом, каждый кристаллик может хранить потенциал либо 1 мВ, либо 3 мВ. (Если там появится какое-то другое напряжение — кристалл вышел из строя). Как я полагаю, минимальный кластер "склеен" из четырёх таких кристалликов. Для поиска его внутри микросхемы каждый кластер имеет свои координаты — например: ячейка 170. А высокоуровневый язык позволяет использовать её (менять потенциалы на кристаллах) как место постоянной дислокации некоего значения — парадоксально, но это и называется "переменной"; а для удобства обращения к ней, предоставлена возможность называть её неким вымышленным именем собственным. Фиксированное же значение не имеет такого постоянного адреса.

[Ответить](#)12. *Алексей:*[20 января 2019 в 23:31](#)

Не слишком много раз повторяется слово — "Переменная"? 😊 — "здесь мы присваиваем значение 7 переменной а. Но что такое «а»? а — это переменная. Переменная в C++ — это просто объект с именем."

[Ответить](#)



13. *Алексей:*

[20 января 2019 в 23:22](#)

В начале путаница возникает. "Объект — это часть памяти..." И потом — "При определении объекта часть этой памяти выделяется для него." Как бы получается если подставить выше написанное, то выйдет следующее — При определении части памяти, часть этой памяти выделяется для него. Как — то уточнить бы всё таки понятие объекта.

[Ответить](#)



14. *Мимо проходил:*

[7 января 2019 в 14:51](#)

Автор, не вводи людей в заблуждения, используя ради выпендрежа английские слова. В русском языке есть понятный перевод и свои устоявшиеся понятия по теме программирования.

"Стэйтмент" — по-русски говорят "выражение"

"л и р — вэлю" — левостороннее и правостороннее присваивание, выражение, значение или ещё что.

Зачем ты плодишь свои слова уродцы?) Ты до этого программирование хоть изучал?

Читай книги, это фундамент.

[Ответить](#)



1. *Юрий:*

[7 января 2019 в 22:31](#)

Хорошо, давай по порядку.

**Вопрос №1:** "Зачем я использую английские слова?"

*Ответ №1:* "Потому что никаких 100% устоявшихся переводов всех понятий/терминов/слов ни в каком учебнике или книге, которую ты, возможно, читал — нет. Есть много терминов, перевод которых идентичен в большинстве ресурсов, но есть термины, перевод которых отличается из одного ресурса в другой. ПОЭТОМУ, ЧТОБЫ НЕ ВВОДИТЬ В ЗАБЛУЖДЕНИЕ читателей, я даю англицизм (английский "statement" = русский "стейтмент"), вместо корявого и непонятного перевода (и никак не ради выпендрежа).

И когда человек ищет дополнительную информацию в Гугле, то лучше он наберёт англицизм и получит релевантный ответ со списком из 10-30 РЕЛЕВАНТНЫХ сайтов, нежели введёт "левостороннее присваивание" и получит НЕРЕЛЕВАНТНЫЙ ответ из всего 5 ссылок, где упоминается этот чудо-термин".

**Вопрос №2:** "Почему я не использую "выражение" в качестве перевода "statement"?"

*Ответ №2:* "Полностью развернутый ответ — это урок №8, который ты, видимо, не удосужился хотя бы просмотреть. В нём сообщается, что стейтмент — это «полное

предложение», которое сообщает компилятору выполнить определенное задание. Выражение — это математический объект, который производит результат-значение. Выражения, как правило, используются внутри стейтментов. Детальнее — см. урок №8".

**Вопрос №3:** "Почему у меня перевод l-value и r-value, а не "левостороннее и правостороннее присваивание, выражение, значение или ещё что"?"

*Ответ №3:* "См. ответ №1. Да можно было бы использовать левостороннее/правостороннее значение (но никак не "левостороннее/правостороннее присваивание"), но этот вариант, по моему мнению, хуже того, что есть (см. второй абзац Ответа №1)".

**Вопрос №4:** "Читал ли я книги по программированию и изучал ли программирование до этого?"

*Ответ №4:* "Не дочитал ни одну книгу по программированию. Да, изучал ещё в колледже и изучаю (если так можно выразиться вообще) в университете".

P.S.

Слова уродцы — это конечно что-то новенькое 😊

[Ответить](#)



2. Джо:

[5 февраля 2019 в 18:55](#)

> "л и р — вэлью" — левостороннее и правостороннее присваивание  
 > "вэлью"  
 > присваивание  
 Иди лучше дальше, не останавливайся

[Ответить](#)



15. максим:

[6 декабря 2018 в 10:50](#)

я правильно делаю? вроде работает

```

1  #include <iostream>
2
3  int getValueFromUser()
4  {
5      std::cout << "Enter an integer: ";
6      int x;
7      std::cin >> x;
8      return x;
9  }
10
11 int main()
12 {
13     int a = getValueFromUser();
14     int b = 2;

```

```

15 |     std::cout << a << " * " << b << " = " << a * b << std::endl;
16 |
17 |     return 0;
18 | }

```

[Ответить](#)

16. Павел:

[23 ноября 2018 в 06:18](#)

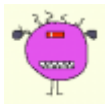
Можно вместо термина "стейтмент" использовать "оператор" или "инструкция"? Мне кажется, что будет понятнее.

[Ответить](#)

1. Юрий:

[23 ноября 2018 в 16:35](#)

Оператор — это оператор, стейтмент — это не один лишь оператор, это определенная совокупность кода. Термин "инструкция" может быть, но не всегда это корректно.

[Ответить](#)

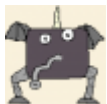
17. Валерий:

[10 октября 2018 в 16:52](#)

Какова область действия объявленных переменных?

Только в теле функции где они объявлены?

Как объявить переменную, что бы она действовала во всех используемых функциях? (Во всей программе...)

[Ответить](#)

1. NiK:

[25 октября 2018 в 12:35](#)

Для того чтобы переменная была видна в любом месте программы (если программа состоит из одного файла .cpp) необходимо сделать ее глобальной. Для этого нужно объявить требуемую переменную выше функции `main()`, например, так:

```

1 | // тут что-то есть...
2 |
3 | // Вот тут объявляем переменную, если хотим чтобы она была
4 | // глобальной
5 |
6 | int value;
7 |
8 | // Теперь переменная value будет видна в любой части программы.
9 |
10 | int main()
11 |

```

```

12 | {
13 |
14 | ...
15 |
16 | return 0;
   | }

```

Но старайтесь не злоупотреблять глобальными переменными. Используйте их только там, где они действительно нужны.

#### [Ответить](#)



1. *Романов Артём:*

[14 декабря 2019 в 12:33](#)

Чтобы глобальная переменная была видна за пределами функции, можно перед типом переменной написать `static` (для видимости во всём файле) или `extern` (для видимости во всех файлах программы):

```

1 | static int sVar //Эта переменная видна во всём файле
2 | extern int eVar //Эта переменная видна во всех файлах программы

```

#### [Ответить](#)



18. *Zufar:*

[5 августа 2017 в 14:20](#)

Грозит ли чем-нибудь мусор в неинициализированной переменной программе? Например если я сначала объявлю переменную и через строчек 40 кода, я ей присвою значение.

#### [Ответить](#)



1. *Юрий:*

[5 августа 2017 в 14:47](#)

Нет, всё будет нормально, если программа не будет обращаться к неинициализированной переменной до того момента, когда вы ей присвоите значение. Т.е. объявить переменную можно и инициализировать её позже также можно, даже компилировать программу можно — вы получите предупреждение, а не ошибку, что переменная неинициализированная. Но если вы будете использовать переменную в каком-то выражении до её инициализации (например, в  $z = x + y$ , где  $x$  — не инициализирован), то это уже чревато неожиданными результатами программы (вплоть до переполнения переменной или простого сбоя).

#### [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