

Ravesli [Ravesli](#)

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





## Урок №28. Инициализация, присваивание и объявление переменных

 [Юрий](#) |

- [Уроки C++](#)

|

 Обновл. 2 Сен 2020 |

 51064

[16](#)

Этот урок является более детальным продолжением [урока №10](#).

Оглавление:

1. [Адреса и переменные](#)
2. [Фундаментальные типы данных в C++](#)
3. [Объявление переменных](#)
4. [Инициализация переменных](#)
5. [uniform-инициализация](#)
6. [Присваивание переменных](#)
7. [Объявление нескольких переменных](#)
8. [Где объявлять переменные?](#)

### Адреса и переменные

Как вы уже знаете, переменные — это имена кусочков памяти, которые могут хранить информацию. Помним, что компьютеры имеют оперативную память, которая доступна программам для использования. Когда мы определяем переменную, часть этой памяти отводится ей.

Наименьшая единица памяти — **бит** (англ. «*bit*» от «*binary digit*»), который может содержать либо значение 0, либо значение 1. Вы можете думать о бите, как о переключателе света — либо свет выключен (0), либо включен (1). Чего-то среднего между ними нет. Если просмотреть случайный кусочек памяти, то всё, что вы увидите, — будет ...011010100101010... или что-то в этом роде.

Память организована в последовательные части, каждая из которых имеет свой **адрес**. Подобно тому, как мы используем адреса в реальной жизни, чтобы найти определенный дом на улице, так и здесь: адреса позволяют найти и получить доступ к содержимому, которое находится в определенном месте памяти. Возможно, это удивит вас, но в современных компьютерах, у каждого бита по отдельности нет своего собственного адреса. Наименьшей единицей с адресом является **байт** (который состоит из 8 битов).

Поскольку все данные компьютера — это лишь последовательность битов, то мы используем **тип данных** (или просто «*тип*»), чтобы сообщить компилятору, как интерпретировать содержимое памяти. Вы уже видели пример типа данных: `int` (целочисленный тип данных). Когда мы объявляем целочисленную переменную, то мы сообщаем компилятору, что «кусочек памяти, который находится по *такому-то* адресу, следует интерпретировать как целое число».

Когда вы указываете тип данных для переменной, то компилятор и процессор заботятся о деталях преобразования вашего значения в соответствующую последовательность бит определенного типа данных. Когда вы просите ваше значение обратно, то оно «восстанавливается» из этой же последовательности бит.

Кроме `int`, есть много других типов данных в языке C++, большинство из которых мы детально рассмотрим на соответствующих уроках.

## Фундаментальные типы данных в C++

В языке C++ есть встроенная поддержка определенных типов данных. Их называют **основными типами данных** (или «*фундаментальные/базовые/встроенные типы данных*»).

Вот список основных типов данных в языке C++:

Категория	Тип	Значение	Пример
Логический тип данных	<code>bool</code>	true или false	true
Символьный тип данных	<code>char</code> , <code>wchar_t</code> , <code>char16_t</code> , <code>char32_t</code>	Один из ASCII-символов 'с'	
Тип данных с плавающей запятой	<code>float</code> , <code>double</code> , <code>long double</code>	Десятичная дробь	3.14159
Целочисленный тип данных	<code>short</code> , <code>int</code> , <code>long</code> , <code>long long</code>	Целое число	64
Пустота	<code>void</code>	Пустота	

## Объявление переменных

Вы уже знаете, как объявить целочисленную переменную:

```
1 int nVarName; // int - это тип, а nVarName - это имя переменной
```

Принцип объявления переменных других типов аналогичен:

```
1 type varName; // type - это тип (например, int), а varName - это имя переменной
```

Объявление пяти переменных разных типов:

```
1 bool bValue;  
2 char chValue;
```

```
3 | int nValue;  
4 | float fValue;  
5 | double dValue;
```

Обратите внимание, переменной типа `void` здесь нет (о типе `void` мы поговорим детально на следующем уроке).

```
1 | void vValue; // не будет работать, так как void не может использоваться в качестве типа
```

## Инициализация переменных

При объявлении переменной мы можем присвоить ей значение в этот же момент. Это называется **инициализацией переменной**.

Язык C++ поддерживает 2 основных способа инициализации переменных.

**Способ №1: Копирующая инициализация** (или «*инициализация копированием*») с помощью знака равенства `=`:

```
1 | int nValue = 5; // копирующая инициализация
```

**Способ №2: Прямая инициализация** с помощью круглых скобок `()`:

```
1 | int nValue(5); // прямая инициализация
```

Прямая инициализация лучше работает с одними типами данных, копирующая инициализация — с другими.

## uniform-инициализация

Прямая или копирующая инициализация работают не со всеми типами данных (например, вы не сможете использовать эти способы для инициализации списка значений).

В попытке обеспечить единый механизм инициализации, который будет работать со всеми типами данных, в C++11 добавили новый способ инициализации, который называется **uniform-инициализация**:

```
1 | int value{5};
```

Инициализация переменной с пустыми фигурными скобками указывает на инициализацию по умолчанию (переменной присваивается `0`):

```
1 | int value{}; // инициализация переменной по умолчанию значением 0 (ноль)
```

В `uniform-инициализации` есть еще одно дополнительное преимущество: вы не сможете присвоить переменной значение, которое не поддерживает её тип данных — компилятор выдаст предупреждение или сообщение об ошибке. Например:

```
1 | int value{4.5}; // ошибка: целочисленная переменная не может содержать нецелочисленное значение
```

**Правило: Используйте `uniform-инициализацию`.**

## Присваивание переменных

Когда переменной присваивается значение после её объявления (не в момент объявления), то это **копирующее присваивание** (или просто «*присваивание*»):

```
1 int nValue;  
2 nValue = 5; // копирующее присваивание
```

В языке C++ нет встроенной поддержки способов прямого/uniform-присваивания, есть только копирующее присваивание.

## Объявление нескольких переменных

В одном **стейтменте** можно объявить сразу несколько переменных одного и того же типа данных, разделяя их имена запятыми. Например, следующие 2 фрагмента кода выполняют одно и то же:

```
1 int a, b;
```

И:

```
1 int a;  
2 int b;
```

Кроме того, вы даже можете инициализировать несколько переменных в одной строке:

```
1 int a = 5, b = 6;  
2 int c(7), d(8);  
3 int e{9}, f{10};
```

**Есть 3 ошибки, которые совершают новички при объявлении нескольких переменных в одном стейтменте:**

**Ошибка №1:** Указание каждой переменной одного и того же типа данных при инициализации нескольких переменных в одном стейтменте. Это не критичная ошибка, так как компилятор легко её обнаружит и сообщит вам об этом:

```
1 int a, int b; // неправильно (ошибка компиляции)  
2  
3 int a, b; // правильно
```

**Ошибка №2:** Использование разных типов данных в одном стейтменте. Переменные разных типов должны быть объявлены в разных стейтментах. Эту ошибку компилятор также легко обнаружит:

```
1 int a, double b; // неправильно (ошибка компиляции)  
2  
3 int a; double b; // правильно (но не рекомендуется)  
4  
5 // Правильно и рекомендуется (+ читабельнее)  
6 int a;  
7 double b;
```

**Ошибка №3:** Инициализация двух переменных с помощью одной операции:

```
1 int a, b = 5; // неправильно (переменная a остаётся неинициализированной)
2
3 int a = 5, b = 5; // правильно
```

Хороший способ запомнить эту ошибку и не допускать в будущем — использовать прямую или `uniform`-инициализацию:

```
1 int a, b(5);
2 int c, d{5};
```

Этот вариант наглядно показывает, что значение 5 присваивается только переменным `b` и `d`.

Так как инициализация нескольких переменных в одной строке является отличным поводом для совершения ошибок, то я рекомендую определять несколько переменных в одной строке только в том случае, если вы будете каждую из них инициализировать.

**Правило:** Избегайте объявления нескольких переменных в одной строке, если не собираетесь инициализировать каждую из них.

## Где объявлять переменные?

Более старые версии компиляторов языка Си вынуждали пользователей объявлять все переменные в верхней части функции:

```
1 #include <iostream>
2
3 int main()
4 {
5     // Все переменные в самом верху функции
6     int x;
7     int y;
8
9     // А затем уже весь остальной код
10    std::cout << "Enter a number: ";
11    std::cin >> x;
12
13    std::cout << "Enter another number: ";
14    std::cin >> y;
15
16    std::cout << "The sum is: " << x + y << std::endl;
17    return 0;
18 }
```

Сейчас это уже неактуально. Современные компиляторы не требуют, чтобы все переменные обязательно были объявлены в самом верху функции. В языке C++ приоритетным является объявление (а также инициализация) переменных как можно ближе к их первому использованию:

```
1 #include <iostream>
2
```

```
3 int main()
4 {
5     std::cout << "Enter a number: ";
6     int x; // мы используем X в следующей строке, поэтому объявляем эту переменную
7     std::cin >> x; // первое использование переменной x
8
9     std::cout << "Enter another number: ";
10    int y; // переменная y понадобилась нам только здесь, поэтому здесь её и объявляем
11    std::cin >> y; // первое использование переменной y
12
13    std::cout << "The sum is: " << x + y << std::endl;
14    return 0;
15 }
```

Такой стиль написания кода имеет несколько преимуществ:

- ➔ Во-первых, возле переменных, которые объявлены как можно ближе к их первому использованию, находится другой код, который способствует лучшему представлению и пониманию происходящего. Если бы переменная `x` была объявлена в начале функции `main()`, то мы бы не имели ни малейшего представления, для чего она используется. Для понимания смысла переменной пришлось бы целиком просматривать всю функцию. Объявление переменной `x` возле операций ввода/вывода данных позволяет нам понять, что эта переменная используется для ввода/вывода данных.
- ➔ Во-вторых, объявление переменных только там, где они необходимы, сообщает нам о том, что эти переменные не влияют на код, расположенный выше, что делает программу проще для понимания.
- ➔ И, наконец, уменьшается вероятность случайного создания неинициализированных переменных.

В большинстве случаев, вы можете объявить переменную непосредственно в строке перед её первым использованием. Тем не менее, иногда, могут быть случаи, когда это будет не желательно (по соображениям производительности) или невозможно. Детально об этом мы поговорим на следующих уроках.

**Правило: Объявляйте переменные как можно ближе к их первому использованию.**

Оценить статью:

★★★★★ (510 оценок, среднее: 4,97 из 5)



← [Глава №1. Итоговый тест](#)

[Урок №29. Тип данных void](#) →



**Комментариев: 16**



1. *Владимир:*  
[15 ноября 2019 в 17:19](#)

В разделе "определение переменных" вроде примеры их объявления, а не определения ))

[Ответить](#)



1. *Юрий:*  
[16 ноября 2019 в 13:55](#)

Спасибо за замечание, всё исправил)

[Ответить](#)



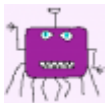
2. *Алексей:*  
[24 июня 2019 в 16:56](#)

Вопрос следующий:

ubuntu, vim, не работает uniform инициализация — типа `a{2}`.

Требует c++11. Поставил, требует так же.

[Ответить](#)



3. *Никита:*  
[17 июня 2019 в 19:21](#)

Сайт отличный не только содержанием, но и быстротой с красотой. Знаю я одного ютубера РНР-шера, который якобы профессионал. Ужасное оформление своего сайта у него. Вы бы видели. Глаза кровью обливались  
p.s.: тот, который "Си за 1 час" видеоролик имеет )

[Ответить](#)



1. *Юрий:*  
[18 июня 2019 в 13:23](#)

Ахахах, спасибо за столь лестный отзыв. Этот сайт тоже не идеален, но вот компромиссное решение между быстротой, красотой и содержанием, кажется, удалось достигнуть.

[Ответить](#)



4. *Andrey:*  
[30 июля 2018 в 09:34](#)

Где можно скачать весь пройденный курс? Иногда приходится находиться в таких местах, где нет интернета и нет возможности заниматься.

[Ответить](#)



1. Юрий:

[30 июля 2018 в 23:39](#)

Я думаю немного позже, я сделаю pdf-ник с половиной курса. Потом, когда все уроки доперевожу — сделаю pdf-ник целого курса.

[Ответить](#)



5. Torgu:

[30 июня 2018 в 19:23](#)

Здравствуйте, где можно скачать IDE с поддержкой C++11? Не хочется упускать возможности использования "Uniform Initialization". Хотя порограмма не старая — Visual Studio 2017

[Ответить](#)



1. Юрий:

[2 июля 2018 в 22:32](#)

Смотрите урок 4 — в нем детально расписывается как устанавливать Visual Studio и где скачать. Если вы скачиваете с оф. сайта и ничего лишнего в настройках не изменяете, то C++11 и uniform инициализация в Visual Studio 2017 используются по умолчанию.

[Ответить](#)



6. Виталий:

[10 апреля 2018 в 07:07](#)

```
"int value{5};"
```

а здесь точно нет ошибок?

по логике должно быть `"int nValue{5};"`

или я ошибаюсь?

[Ответить](#)



1. Юрий:

[10 апреля 2018 в 16:04](#)

Это названия переменных. Имена переменных могут быть разными. Здесь они разные — не одинаковые во всех примерах.

[Ответить](#)



7. Сергей:

[19 июля 2017 в 04:18](#)

Хороший урок про определение переменных, автору + 100 к карме, надеюсь сайт не умрёт и дальше будет развиваться и пополняться статьями, которые изложены в такой простой и доходчивой форме.



[Ответить](#)

1.  Юрий:  
[19 июля 2017 в 16:49](#)

Будем продолжать дальше 😊

[Ответить](#)

2.  Илья:  
[28 августа 2017 в 19:58](#)

Поддерживаю предыдущего комментатора — сайт отличный! Автору спасибо и успехов! Супер! Надеюсь на продолжение работы по переводу!

[Ответить](#)

1.  Юрий:  
[31 августа 2017 в 17:07](#)

И Вам спасибо 😊

[Ответить](#)

3.  илья:  
[17 июня 2018 в 13:34](#)

сайт отличный, не поспоришь!

[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.

[Отправить комментарий](#)[TELEGRAM](#)  [КАНАЛ](#)[Электронная почта](#)[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