

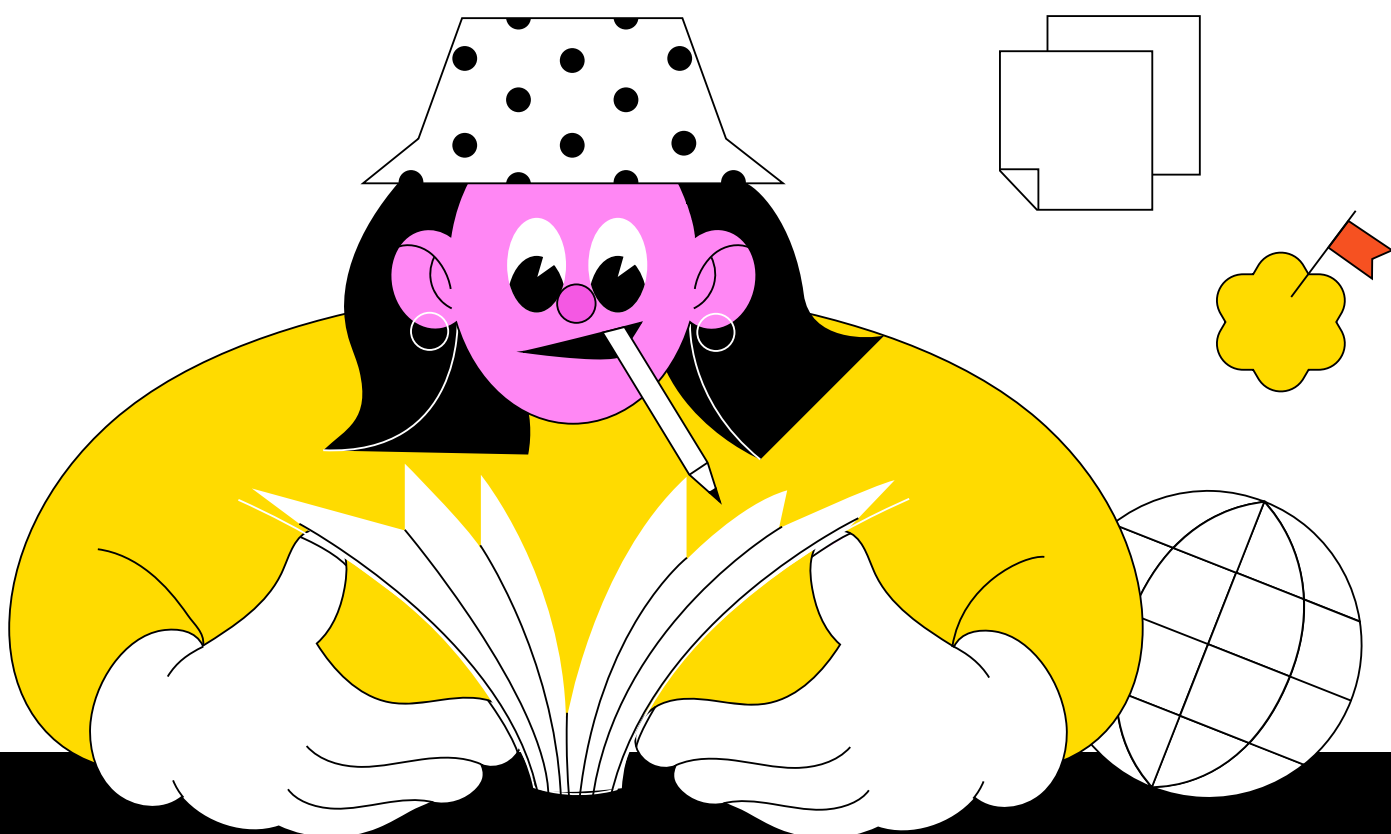
## Как английские слова в коде делают нашу жизнь проще

Что, если знание английского помогает дебажить быстрее?

Полезно разбираться в английском, если ты собираешься работать в ИТ. Как минимум потому, что большинство языков программирования написаны с использованием слов и конструкций английского языка, а большая часть библиотек и фреймворков разрабатывается на английском.

Да и практически вся документация пишется на английском языке.

Но как знание английского поможет разобраться в коде? И насколько глубоко удастся понять суть программы? Давайте разбираться.



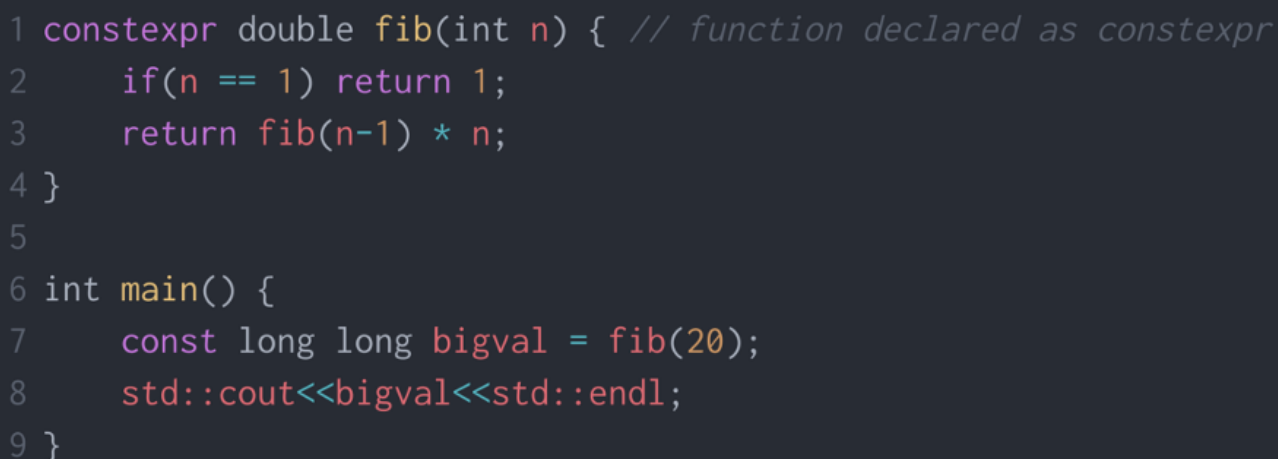
## Немного истории

В первых компьютерах языки высокого уровня, смысл которых может быть частично или полностью понят человеком на визуальном уровне, не использовались вовсе. Программисты вводили заранее закодированные строки — конкретные инструкции, понятные только машине. При этом стороннему зрителю прочитать и понять, что делает этот код, было невозможно.

Одним из первых высокоуровневых языков стали Shortcode и так называемый Автокод, написанные более «человеческим» языком. Но программирование на них всё равно требовало глубокой инженерной подготовки и времени на отладку. Только в 1957 году появился FORTRAN, программы на котором выглядели доступными для понимания человеком. FORTRAN имел синтаксис, который позволял разобраться в том, какие структуры данных использует программа и какие операции над ними может совершать. Теперь читать код стало гораздо проще.

## Как разобраться в несложных конструкциях популярных языков на примере C++

Начнём с типов переменных, которые можно использовать для работы.



```
1 constexpr double fib(int n) { // function declared as constexpr
2     if(n == 1) return 1;
3     return fib(n-1) * n;
4 }
5
6 int main() {
7     const long long bigval = fib(20);
8     std::cout<<bigval<<std::endl;
9 }
```



## Основные типы переменных в C++

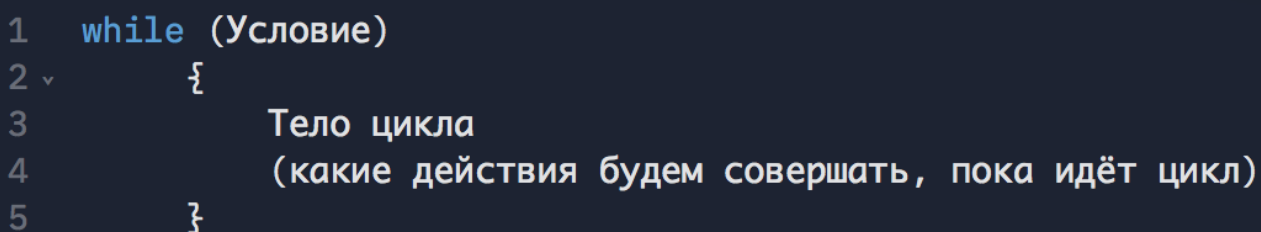
- `int` — целочисленный тип данных (от англ. `integer` — целое число)
- `float` — тип данных с плавающей запятой = дробное число (от англ. `float` — плавать)
- `double` — тип данных с плавающей запятой двойной точности (`double` — удвоение — повышает вдвое количество знаков после запятой).
- `char` — символьный тип данных (от англ. `char` — символ).
- `bool` — логический тип данных (от англ. `boolean` — логический, где подразумевается соответствие значения параметрам `TRUE` или `FALSE`).

Теперь несложно понять, что строка с операцией `{ float b = 7,9; }` указывает на создание дробного числа. С другой стороны, если бы `b` имела тип `int`, то значение бы автоматически округлилось.

## Простые структуры в C++

Логические структуры тоже написаны на относительно понятном языке в логичной, пусть и более схематичной форме. Рассмотрим классические конструкции с условиями и счётчиками. Условный оператор даёт программе выполнить определённое действие, только если выполняется указанное условие. Счётчик же предназначен для того, чтобы операция выполнялась несколько раз — в нём дополнительно указывается требуемое количество шагов.

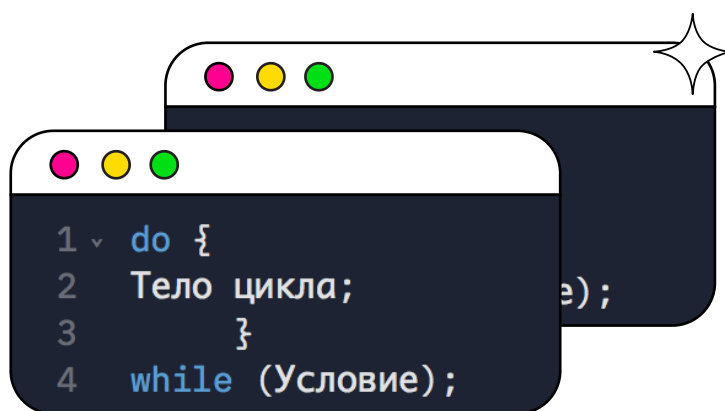
В виде кода обе конструкции выглядят следующим образом (на примере цикла `while`):



```
1  while (Условие)
2  {
3      Тело цикла
4      (какие действия будем совершать, пока идёт цикл)
5  }
```

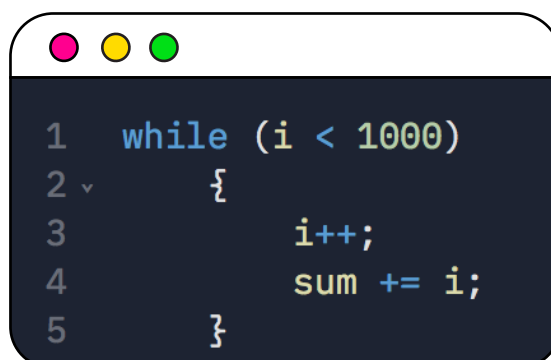
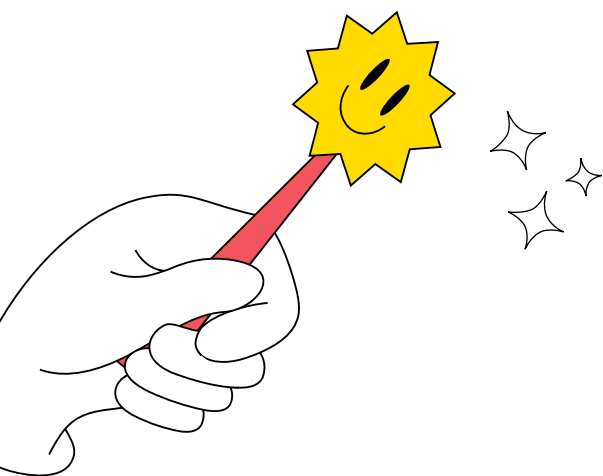


While в данном случае говорит программе: «пока условие X выполняется, делай следующие действия». У цикла есть и другой тип вида:



Смысл здесь изменится так же, как изменился бы в родном языке: «делай следующие действия, пока не наступит X». Теперь программа будет проверять условие на НЕвыполняемость, и остановится, как только оно будет достигнуто.

Приведём пример классического цикла While: код ниже будет суммировать все значения *i*, постепенно увеличивая *i* на единицу. Как только *i* станет равно 1000, выполнится условие отсечки и цикл завершится.



Если же нам известно точное количество итераций, стоит использовать цикл FOR (англ. – для).

```
1  for (действие до начала цикла;  
2      условие продолжения цикла;  
3      действия со счётчиком цикла на каждом шаге)  
4  {  
5      тело цикла;  
6  }
```

Интерпретация простая и тоже понятная путём прямого перевода:  
для каждого X из перечня чисел в скобках выполняй действие Y.

## Более сложные конструкции и методы

Тот же C++ имеет и ряд стандартных функций, названных так, чтобы можно было уже видя предложенный список доступных функций, выбрать подходящую. Поработаем с динамическими массивами. Ничего страшного, если этот термин вам пока не знаком. Название дано им неслучайно, а сами они во многом похожи на объекты из реального мира.

Динамический массив — набор данных, которые могут перемещаться внутри ячеек своей матрицы. Разновидностью динамического массива является очередь — процесс, когда некие данные занимают «места» в ячейках и по мере освобождения предыдущего места, переходят на него.

Для работы с такими массивами есть большой список функций. Облегчает задачу то, что зная английский, в них легко разобраться. Например, метод `clear()` — очищает очередь; `IsEmpty()` — проверяет, не является ли очередь пустой; `print()` — выводит очередь на экран. Покажем, как очередь передвигается на шаг вперёд.





```
1  for (int i = 0; i < count; i++) // проходим по всем номерам в очереди циклом FOR
2      p2[ i ] = p[ i + 1 ]; // копируем все элементы, чтобы не потерять их (кроме первого)
3      if (count > 0)
4          delete[ ] p; // освобождаем предыдущий номер
5      p = p2; // направляем на него следующий по списку
6      return item; // возвращаем указатель на нашу очередь
7  }
```

Надеемся, функции delete и return уже не вызвали у вас вопросов.

Кстати, в объектно-ориентированном программировании (ООП) есть концепции, которые позволяют описать с помощью английских терминов еще более сложные конструкции, например, паттерны.

Паттерны ООП (от английского слова «шаблон, образец») являются шаблонами для написания кода определённого узкого функционала. Многие из них были созданы исходя из частых запросов на одну и ту же логику в коде. Вот некоторые:

### REGISTRY (АНГЛ. – ЖУРНАЛ ЗАПИСЕЙ)

Паттерн предназначен для хранения записей и предоставления информации по ним в том случае, если к записи обратились по имени. Например, список номеров медицинских карт или телефонных номеров. Очевидно, что всегда проще использовать готовый шаблон, чем заново изобретать велосипед, точнее, список данных.

### FACTORY (АНГЛ. – ФАБРИКА)

Паттерн хранит информацию о том, как создаются определенные объекты и данные. Например, для создания новой анкеты обратной связи нам не нужно каждый раз писать процесс, описывающий множество необходимых полей. Имея паттерн Factory, мы можем обратиться к нему и сказать «создай новую анкету». В том случае, если анкета поменяла свою форму, изменения вносятся сразу на уровень Фабрики.



## DECORATOR (АНГЛ. – ДЕКОРАТОР, ОФОРМИТЕЛЬ)

Паттерн чаще всего используется для расширения исходного объекта до требуемого вида. Применяется для работы с интерфейсами и другими деталями, которые нужно «дотюнить».

По этой же логике появились Mediator (Посредник), Blackboard (Доска объявлений), Proxy (Заместитель) и так далее.

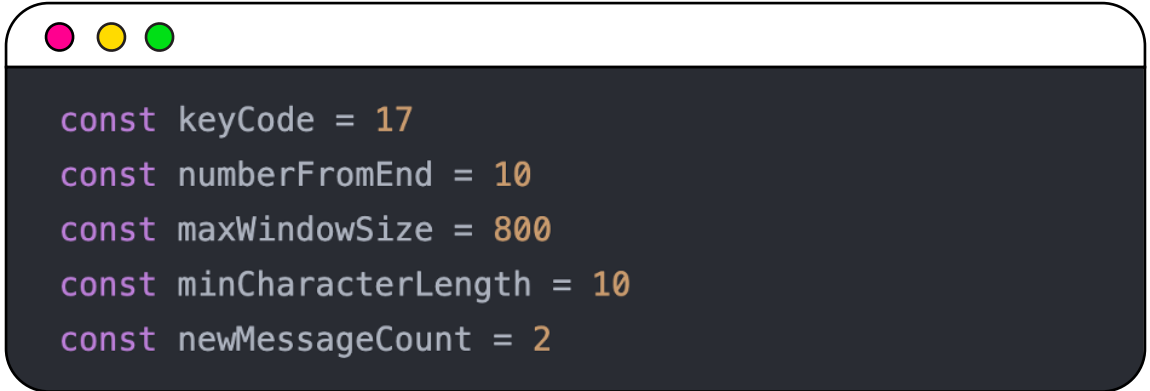
### Как назвать свои данные

Переменные и классы, которые вы задаёте сами, тоже стоит называть понятным языком, иначе есть риск уже через 1-2 недели забыть их смысл. Пара хороших примеров: `totalScore` — итоговый счёт в игре, `maxWidth` — максимальная ширина.

Если это уместно и соответствует сути, принято добавлять к названиям значения:

- количество (`count`), если идёт подсчёт,
- размер (`size`, `length`), если мы вычисляем длину и т.п.,
- номер (`number`), если возвращаем значение чего-то.

Оставим здесь хороший пример от [umatuhin.ru](https://umatuhin.ru):

A code snippet displayed in a window with a dark background and light-colored text. The window has a title bar with three colored buttons (red, yellow, green) on the left. The code consists of five lines, each starting with the keyword 'const' followed by a variable name and an equals sign and a value.

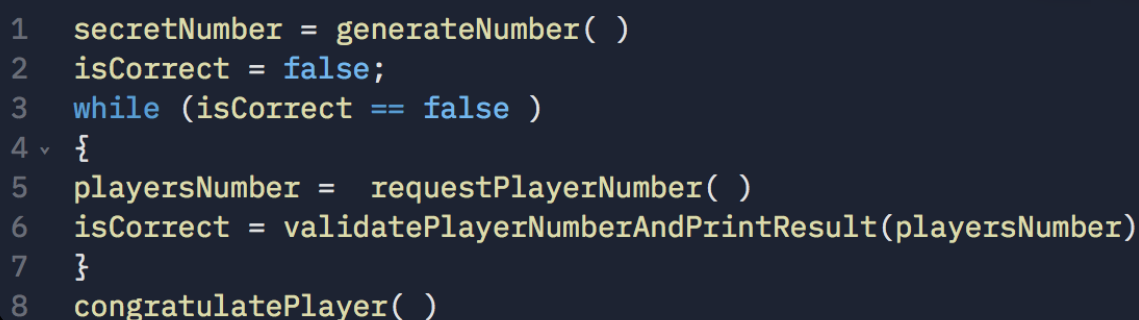
```
const keyCode = 17
const numberFromEnd = 10
const maxWindowSize = 800
const minCharacterLength = 10
const newMessageCount = 2
```

Кстати, массивы часто называют словами во множественном числе, которые имеют окончание на `s` и `es`. Таким образом становится понятно уже из названия, что внутри лежит целый набор данных. А в функциях на первое



место часто ставят глагол, кратко объясняющий, что она делает, например, find или create.

Зная эту деталь, уверены, вам будет несложно разобраться в задачке, которую мы подготовили для вас. Это простая игра, где компьютер загадывает число. Постарайтесь разобраться, какие правила у этой игры, и что вы можете делать, чтобы угадать число.



```
1 secretNumber = generateNumber( )
2 isCorrect = false;
3 while (isCorrect == false )
4 {
5   playersNumber = requestPlayerNumber( )
6   isCorrect = validatePlayerNumberAndPrintResult(playersNumber)
7 }
8 congratulatePlayer( )
```

ОТВЕТ: на самом деле у вас не так много возможностей угадать число, загаданное компьютером. Вы видите, что переменной “секретное число” было присвоено случайное значение. А дальше вы методом подбора пытаетесь его угадать. Как только введённое число совпадёт с загаданным, компьютер поздравит вас.

Надеемся, вам стало проще «читать» код визуально и не так страшно его изучать. Если что-то забудете, сориентируетесь по ходу дела, если у вас большой словарный запас :)

