

# Обзор языка программирования для Arduino

Официальной средой разработки является Arduino IDE, где программирование осуществляется на языке C++ – одном из самых популярных и мощных языков. Сами разработчики называют язык, на котором пишутся программы для ардуино Wiring, так как в стандартной библиотеке Arduino.h используются функции и инструменты из программной платформы Wiring. Но языком, именно языком, из которого берётся синтаксис, является C++.

Простейшая программа для Arduino, которая управляет миганием светодиода:

```
// ----- //

int led = 13; // Объявляем переменную led с номером 13
void setup()
{
  pinMode(led, OUTPUT); // Устанавливаем led в режим ВЫХОД
}

void loop()
{
  digitalWrite(led, HIGH); // Подаём на 13 вывод высокий сигнал (+5v)
  delay(1000);             // Ожидаем 1 сек.
  digitalWrite(led, LOW);  // Подаём на 13 вывод низкий сигнал (0v)
  delay(1000);             // Ожидаем 1 сек.
}

// -----//
```

В 1 секунде 1000 миллисекунд.

Программа выполняет бесконечный цикл мигания встроенным светодиодом Arduino.

То же самое, но немного по другому:

```
// -----//

int led = 13; // Объявляем переменную led с номером 13
void setup()
{
  pinMode(led, OUTPUT); // Устанавливаем led в режим ВЫХОД
}

void loop()
{
  digitalWrite(led, !digitalRead(led)); /* Считываем состояние 13 выхода и инвертируем его
                                         *значение */
  delay(1000);
}

// -----//
```

Примечание: инверсия, это отрицание — переворачивание смысла, замена «белого» «чёрным», 1 на 0.

Структура программы Ардуино достаточно проста и в минимальном варианте состоит из двух обязательных частей **setup()** и **loop()**.

```
void setup()
{
    // Тело функции или программы
    // код выполняется один раз при запуске программы
    // используется в основном для настроек
}
void loop()
{
    // Тело функции или программы
    // основной код, выполняемый циклично
}
```

Функция `setup()` выполняется один раз, при включении питания или перезагрузки контроллера. Обычно в ней происходят начальные установки переменных. Функции `setup()` и `loop()` должны присутствовать в программе для Arduino, даже если в них ничего нет!

После завершения `setup()` управление переходит к функции `loop()`. Она в бесконечном цикле выполняет команды, записанные в её теле (между фигурными скобками). Собственно эти команды и совершают все командные действия микроконтроллера.

## **Первоначальные правила синтаксиса языка C (Си по-русски)**

### **Понятийный аппарат**

Программа — последовательность инструкций для исполнения процессором.

Код — текст компьютерной программы на каком-либо языке программирования, который может быть прочтён человеком.

Переменная — место (ящик) для хранения значения, которое обязательно имеет своё название. Например: `x = 2`, где `x` - это имя переменной, а `2` - это значение переменной. Объявление переменной происходит в момент её создания.

Тип переменной - этот тип определяет, какие значения может иметь переменная, какие операции с ней можно производить и сколько байт в памяти компьютера она будет занимать.

Функция — фрагмент программного кода (подпрограмма), к которому можно обратиться из другого места программы.

Параметр функции — это переменная, которая используется в функции и значение которой предоставляет вызывающий объект. Например: `void printValue(int a)`

Аргумент — это значение, которое передаётся из вызывающего объекта в функцию и которое указывается в скобках при вызове функции. Например: `printValue(7);`

Метод — это функция или процедура, принадлежащая какому-то классу или объекту

Объект — область памяти, способная содержать данные и обладающая типом (похожи на переменные)

Вызов функции — применение функции в программе.

Массив — пронумерованная последовательность величин одинакового типа.

Компиляция - сборка программы, т.е. перевод текста с языка программирования C++ на язык компьютера, который понимает только единицы и нули.

## Оформление и форматирование

Есть такое понятие, как форматирование (оформление) кода, т.е. соблюдение сдвигов, пробелов и интервалов. Сдвиг операторов внутри фигурных скобок подчёркивает логическую структуру программы. Форматирование необходимо, т. к. это значительно облегчает чтение и понимание кода.

Сравните форматированный и неформатированный код

<pre>void setup() {   pinMode(10, OUTPUT); }  void loop() {   int knob = analogRead(0);   knob = map (knob, 0, 1023, 0, 255);   analogWrite(10, knob); }</pre>	<pre>void setup(){pinMode(10,OUTPUT);} void loop(){int knob=analogRead(0); knob=map (knob,0,1023,0,255); analogWrite(10,knob);}</pre>
------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	---------------------------------------------------------------------------------------------------------------------------------------------------

Структура форматирования:

- Между математическими действиями, знаками сравнения, присваивания и т.п. ставится пробел;
- Как и в обычном тексте, пробел ставится после и не ставится перед запятой, двоеточием, точкой с запятой;
- Отступ от левого края экрана – знак табуляции или 2-а пробела, код сдвигается вправо и на одном расстоянии формируются команды из одного блока кода. В Arduino IDE одна табуляция, равна двум пробелам;
- Каждое действие выполняется с новой строки;
- Фигурные скобки начала и окончания блока кода принято писать на отдельной строке. Также очень многие пишут открывающую скобку на строке с оператором, это экономит место.

## Имена переменных

Типы данных - определяют возможные значения и их смысл, операции, а также способы хранения значений типа.

Общепринятые соглашения:

- Имена переменных принято писать начиная с маленькой буквы, называть их так, чтобы было понятно в соответствии с их назначением. Да, английский неплохо бы подтянуть! Пример: `value`;
- Если название переменной хочется составить из двух и более слов, они разделяются верхним регистром первой буквы каждого нового слова. Пример: `myButtonState`;
- Имена типов данных и классов принято писать с большой буквы. Пример: `Signal`, `Servo`;
- Имена констант принято писать в верхнем регистре, разделение – нижнее подчеркивание. Например: `MOTOR_SPEED`;
- При написании библиотек и классов, имена внутренних переменных принято писать начиная со знака нижнего подчеркивания. Пример: `_position`;

Несколько общепринятых сокращений для названий переменных, вы часто будете встречать их в других программах:

- `button` – `btn`, кнопка
- `index` – `idx` – `i`, индекс
- `value` – `val`, значение
- `variable` – `var`, переменная

Имена глобальных переменных принято писать без сокращений, например `EARTH_RADIUS`.

Имена функций и методов принято начинать с глагола, кратко описывающего действие функции. Вот те из них, которые вы будете встречать постоянно:

- `get` – получить значение (`getValue`)
- `set` – установить значение (`setTime`)
- `print`, `show` – показать что-то
- `read` – прочитать
- `write` – записать
- `change` – изменить
- `clear` – очистить
- `begin`, `start` – начать
- `end`, `stop` – закончить, остановить
- Имена функций, в которых логические значения используются в качестве возвращаемых очень часто начинаются со слов `is` (например, `isEqual`) или `has` (например, `hasCommonDivisor`)

### Структура кода

- Переменная любого типа должна вызываться после своего объявления. Иначе будет ошибка.
- Объявление и использование классов или типов данных из библиотеки/файла должно быть после подключения библиотеки/файла.

### Синтаксис

(набор правил, описывающий комбинации символов алфавита)

- Тела функций и процедур заключаются в фигурные скобки `{ }`

- Каждая команда (инструкция) заканчивается точкой с запятой ;
- Метод применяется к объекту через точку. Пример: Serial.begin()
- Вызов функции, процедуры или метода всегда заканчивается скобками, даже если функция не принимает параметров или аргументов. Пример: loop()
- Разделитель десятичных дробей – точка. Пример: 0.25. У запятой другое применение!
- Запятыми перечисляются аргументы функций и методов, а также членов массива. Пример: digitalWrite(3, HIGH); массив – int myArray[4] = {3, 4, 5 ,6}; Также запятая является самостоятельным оператором
- Одиночный символ заключается в одиночные кавычки 'a'
- Строка заключается в двойные кавычки “строка”
- Имена переменных могут содержать латинские буквы в верхнем и нижнем регистре (большие и маленькие), цифры и нижнее подчёркивание. Пример: myVal\_35 .
- Имена переменных не могут начинаться с цифры. Только с буквы или нижнего подчёркивания
- Регистр имеет значение, т.е. большая буква отличается от маленькой. Пример: переменные val и Val – не одно и то же
- В угловые кавычки < > заключаются заголовочные файлы и стандартные библиотеки языка, например #include<Arduino.h>
- В двойные кавычки заключаются ваши собственные файлы, например #include "constants.h"
- В одинарные кавычки заключаются символы (например 'a'), а в двойные кавычки — строки - "cost"

К синтаксису также можно отнести комментарии, т.к. в разных языках они выделяются по-разному. Комментарий это обычный текст, который игнорируется на этапе компиляции. Комментарии нужны для пояснения кода, как себе самому, так и другим возможным его читателям. В C++ у нас два типа комментариев:

### Однострочный комментарий

```
// однострочный комментарий
// компилятор меня игнорирует =(
```

### Многострочный комментарий

```
/* Многострочный
комментарий */
```

## Переменные

Переменная (variable) - именованная область памяти, которой могут манипулировать программы.

У каждой переменной в C++ есть определённый тип. Тип определяет размер и расположение в памяти, диапазон возможных значений, набор применимых операций.

Объявление (declaration) — происходит в процессе её создания, что делает её тип и имя известными программе.

Переменную имеет смысл объявлять ближе к месту его первого использования.

В языке C++ существуют следующие типы данных:

Тип данных	Диапазон чисел и значений
bool	true, false
char	-128 ... 127
unsigned char	0 ... 255
short, int	-32768 ... 32767
unsigned int	0 ... 65535
long	-2147483648 ... 2147483647
unsigned long	0 ... 4294967295
float	-3.4028235+38 ... 3.4028235+38
double	-3.4028235+38 ... 3.4028235+38

Знаковый тип (signed) способен представлять отрицательные, а не только положительные числа.

Беззнаковый тип (unsigned) - только положительные числа и 0 (unsigned int может быть сокращён до unsigned).

### Несколько эмпирических правил, способных помочь при выборе используемого типа

1. Используйте беззнаковый тип, когда точно знаете, что значения не могут быть отрицательными;
2. Используйте тип int для целочисленной арифметики. Если ваши значения больше, чем минимально гарантируемый тип int, то используйте long long;
3. Не используйте базовые типы char и bool в арифметических выражениях. Используйте их только для хранения символов и логических значений;
4. Используйте тип double для вычислений с плавающей точкой. У типа float обычно не хватает точности. Точность, определяемая типом long double обычно чрезмерна и не нужна.

Ключевые слова C++, которые нельзя использовать в именах переменных, функций, объектов и классов:

alignas	enum	return
alignof	explicit	short
and	export	signed
and_eq	extern	sizeof
asm	false	static
auto(1)	float	static_assert
bitand	for	static_cast
bitor	friend	struct
bool	goto	switch
break	if	template
case	inline	this
catch	int	thread_local
char	long	throw
char16_t	mutable	true
char32_t	namespace	try
class	new	typedef

compl	noexcept(начиная с C++11)	typeid
const	not	typename
constexpr	not_eq	union
const_cast	nullptr(начиная с C++11)	unsigned
continue	operator	using(1)
decltype	or	virtual
default	or_eq	void
delete	private	volatile
do	protected	wchar_t
double	public	while
dynamic_cast	register	xor
else	reinterpret_cast	xor_eq