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

```
// -=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-//

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);
}
// -=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-//
```

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. Функция setup() должна присутствовать в программе для Arduino, даже если в ней 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 коммандные действия микроконтроллера.

Первоначальные правила синтаксиса языка С (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 (ящичек) для хранения значения. Например: x = 2, где x - это переменная, а 2 - это значение. Объявление переменной происходит в момент её создания.

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.

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

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

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, код сдвигается вправо и на одном расстоянии формируются команды из одного блока кода. В Arduino IDE одна табуляци, равна двум пробелам;
- Каждое действие выполняется с новой строки (автоформатирование обычно это не исправляет);
- 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 понятнои 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 подчёркивание. Например: MOTOR_SPEED
- При написании библиотек и классов, имена внутренних переменных принято писать начиная со знака нижнего подчёркивания. 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 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

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 заключаются в фигурные скобки { }
- Каждая команда заканчивается точкой с запятой ;
- Метод применяется к объекту через точку. Пример: Serial.begin()
- Вызов функции 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[] = {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, например #include<Arduino.h>

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 определён 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. Используете 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 имён переменных:

alignas alignof and and_eq asm auto(1) bitand bitor bool break case catch char char16_t char32_t class compl const constexpr const_cast continue decltype default delete do double dynamic_cast else	enum explicit export extern false float for friend goto if inline int long mutable namespace new noexcept(начиная с C++11) not not_eq nullptr(начиная с C++11) operator or or_eq private protected public register reinterpret_cast	return short signed sizeof static static_assert static_cast struct switch template this thread_local throw true try typedef typeid typename union unsigned using(1) virtual void volatile wchar_t while xor xor_eq