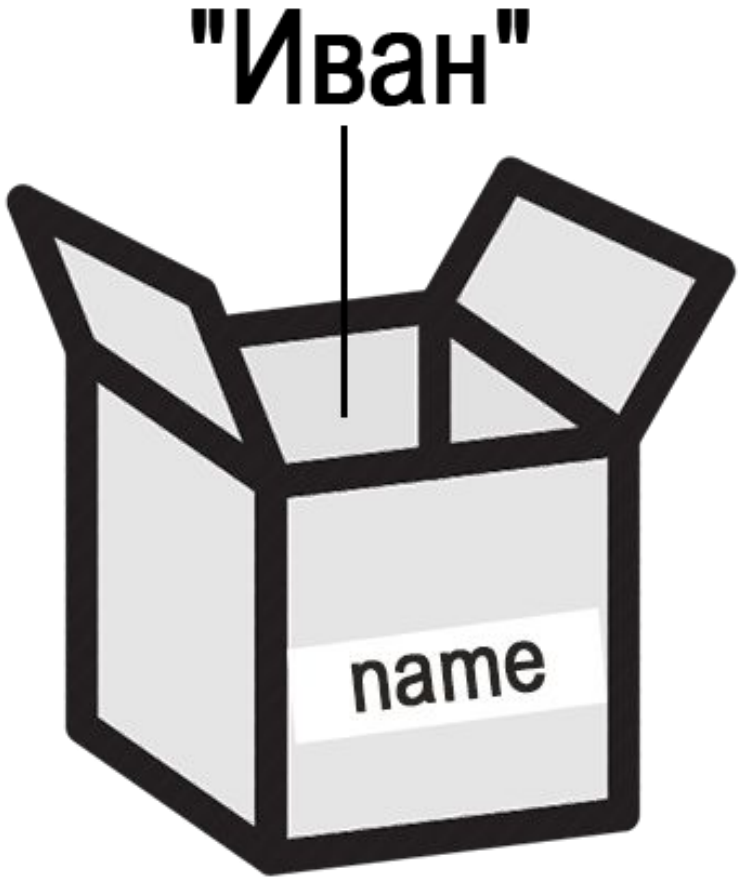
Переменные и типы данных в C++

Как и во многих языках программирования, в C++ для хранения данных используются **переменные**. Переменная имеет тип, имя и значение. Тип определяет, какую информацию может хранить переменная.



- **Переменная** — поименованная, либо адресуемая иным способом область памяти, адрес которой можно использовать для осуществления доступа к данным и изменять значение в ходе выполнения программы.

Перед использованием любую переменную надо определить. Синтаксис определения переменной выглядит следующим образом:

<тип переменой> <один или несколько идентификаторов переменных через запятую>;

Пример объявления переменных

- **int** a; *// объявление переменной a целого типа.*
- **float** b; *// объявление переменной b типа данных с плавающей запятой.*
- **double** c = 14.2; *// инициализация переменной типа double.*
- **char** d = 's'; *// инициализация переменной типа char.*
- **bool** k = true; *// инициализация логической переменной k.*

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

- int a; *// это объявление переменной*
- a = 8; *// а это присваивание переменной а значения 8*

- int a = 8; *// инициализируем переменную a значением 8*

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

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

Важные моменты:

- Переменная должна бывать объявлена **до** обращения к себе, буквально находиться **выше по коду**. Иначе вы получите ошибку “переменная не объявлена” – *Not declared in this scope*
- Глобальные переменные при объявлении имеют значение 0 по умолчанию (если не инициализировать)
- Локальные переменные (создаваемые внутри функций в процессе работы программы) при объявлении могут иметь случайное значение, т.к. выделяются из динамической памяти. Крайне рекомендуется их инициализировать, если в дальнейшем коде они используются как нулевое значение (пример ниже)

Типы данных

Название	Альт. название	Вес	Диапазон	Особенность
boolean	bool	1 байт	0 или 1, **true** или **false**	Логическая переменная. `bool` на Arduino тоже занимает 1 байт, а не бит!
char	–	1 байт	-128… 127	Хранит номер символа из таблицы символов ASCII
–	int8_t	1 байт	-128… 127	Целочисленный тип
byte	uint8_t	1 байт	0… 255	Целочисленный тип
int	int16_t, short	2 байта	-32 768… 32 767	Целочисленный тип
unsigned int	uint16_t, word	2 байта	0… 65 535	Целочисленный тип
long	int32_t	4 байта	-2 147 483 648… 2 147 483 647	Целочисленный тип
unsigned long	uint32_t	4 байта	0… 4 294 967 295	Целочисленный тип
float	–	4 байта	-3.4028235E+38… 3.4028235E+38	Хранит числа с плавающей точкой (десятичные дроби). Точность: 6-7 знаков
double	–	4 байта		Для AVR то же самое, что `float` . А так он 8 байт на более взрослом железе
–	int64_t	8 байт	-(2^64)/2… (2^64)/2-1	*Очень большие числа. Стандартный Serial не умеет такие выводить
–	uint64_t	8 байт	2^64-1	*Очень большие числа. Стандартный Serial не умеет такие выводить

Константы

Если значение переменной не будет изменяться в процессе выполнения программы – рекомендуется объявить её как константу, это позволит компилятору лучше оптимизировать код и в большинстве случаев он будет чуточку легче и

Задать константы можно:

- Точно так же как переменную, указав перед типом данных слово const.

```
const byte myConst = 10; // объявить константу типа byte
```

Задать константы можно:

- При помощи директивы препроцессору #define, которая делает следующее: на этапе компиляции кода препроцессор заменяет указанные все последовательности символов **в текущем документе** (напомню, что вкладки Arduino IDE являются одним документом) на соответствующие им значения. Константа, определённая при помощи #define **не занимает места в оперативной памяти**, а хранится как код программы во Flash памяти, это самый большой плюс данного способа.
  Синтаксис: #define имя значение

Точка запятой не ставится. Таким способом обычно указывают пины подключения, настройки, различные величины и всё такое.
Пример:

```
#define BTN_PIN 10
#define DEFAULT_VALUE 3423
```

Область видимости

Переменные, константы и другие типы данных (структуры и перечисления) имеют такое важное понятие, как область видимости. Она бывает

- Глобальной
- Локальной
- Формальной (параметр)

Глобальная переменная объявляется **вне функций** и доступна для чтения и записи в любом месте программы, в любой её функции.

```
byte var;
void setup() {
// спокойно меняем глобальную переменную
var = 50;
}
void loop() {
// спокойно меняем глобальную переменную
var = 70;
}
```

Локальная переменная живёт **внутри функции** или внутри любого блока кода, заключённого в **{** фигурные скобки **}**, доступна для чтения и записи только внутри него. При попытке обратиться к локальной переменной из другой функции (за пределами её **{** блока **}**) вы получите ошибку, потому что локальная переменная создаётся заново при выполнении содержащего её блока кода (или функции) и удаляется из памяти при завершении выполнения этого блока (или функции).

Формальная переменная, она же параметр, передаваемый в функцию, ведёт себя как обыкновенная локальная переменная, но появляется при немного других условиях: при вызове функции. Эту переменную можно читать и менять внутри её функции.