DATA TYPES

Data types in C refers to an extensive system used for declaring variables or functions of different types. The type of a variable determines how much space it occupies in the storage and how the bit pattern stored is interpreted.

Void

The void keyword is used only in function declarations.It indicates that the function is expected to return no information.

Example:

```
1  void loop() {
2      //rest of code
3  }
4
```

Boolean

A boolean holds one of two values, true of false. Each boolean variable occupies one byte of memory.

Example:

```
1  /* Decleration of variable with type boolean
2   * and initialise it with false
3  */
4  boolean val = false;
5
6  /* Decleration of variable with type boolean
7   * and initialise it with true
8  */
9  boolean state = true;
10
```

Char

A data type that takes up one byte of memory that stores a character value. Character literals are written in single quotes like this: 'A' and for multiple characters, strings use double quotes: “ABC”.
Characters are stored as numbers. Specific encoding in ASCII chart.

Example:

```
/* Decleration of variable with type char
 * and initialise it with character a
*/
char chr_a = 'a';
/* Decleration of variable with type char
 * and initialise it with character 97
*/
char chr_c = 97; // 97 == a

```

Unsigned char

Unsigned char is an unsigned data type that occupies one byte of memory. The unsigned char data type encodes numbers from 0 to 255.

Example:

```
/* Decleration of variable with type unsigned char
 * and initialise it with character y
*/
unsigned char chr_y = 121;

```

Byte

A byte stores an 8-bit unsigned number, from 0 to 255.

Example:

```
/* Decleration of variable with type byte
 * and initialise it with 25
*/
byte m = 25;


```

Int

Integers are the primary data-type for number storage. int stores a 16-bit (2-byte) value. This yields a range of -32,768 to 32,767 (minimum value of -2^15 and a maximum value of (2^15) - 1).

Example:

```
/* Decleration of variable with type int
 * and initialise it with 32
*/
int counter = 32;

```

Unsigned int

Unsigned ints (unsigned integers) are the same as int in the way that they store a 2 byte value. Instead of storing negative numbers, however, they only store positive values, yielding a useful range of 0 to 65,535 (2^16) - 1). The Due stores a 4 byte (32-bit) value, ranging from 0 to 4,294,967,295 (2^32 - 1).

Example:

```
/* Decleration of variable with type unsigned int
 * and initialise it with 60
*/
unsigned int counter = 60;


```

Word

On the Uno and other ATMEGA based boards, a word stores a 16-bit unsigned number. On the Due and Zero, it stores a 32-bit unsigned number.

Example:

```
/* Decleration of variable with type word
 * and initialise it with 1000
*/
word w = 1000;

```

Long

Long variables are extended size variables for number storage, and store 32 bits (4 bytes), from 2,147,483,648 to 2,147,483,647.

Example:

```
/* Decleration of variable with type long
 * and initialise it with 1024789
*/
long lg = 1024789;

```

Unsigned long

Unsigned long variables are extended size variables for number storage and store 32 bits (4 bytes). Unlike standard longs, unsigned longs will not store negative numbers, making their range from 0 to 4,294,967,295 (2^32 - 1).

Example:

```
/* Decleration of variable with type unsigned long
 * and initialise it with 101776
*/
unsigned long lg = 101776;

```

Short

A short is a 16-bit data-type. On all Arduinos (ATMega and ARM based), a short stores a 16-bit (2-byte) value. This yields a range of -32,768 to 32,767 (minimum value of -2^15 and a maximum value of (2^15) - 1).

Example:

```
/* Decleration of variable with type short
 * and initialise it with 19
*/
short sh = 19;

```

Float

Data type for floating-point number is a number that has a decimal point. Floating-point numbers are often used to approximate the analog and continuous values because they have greater resolution than integers.
Floating-point numbers can be as large as 3.4028235E+38 and as low as 3.4028235E+38. They are stored as 32 bits (4 bytes) of information.

Example:

```
/* Decleration of variable with type float
 * and initialise it with 1.362
*/
float fl = 1.362;

```

Double

On the Uno and other ATMEGA based boards, Double precision floating-point number occupies four bytes. That is, the double implementation is exactly the same as the float, with no gain in precision. On the Arduino Due, doubles have 8-byte (64 bit) precision.

Example:

```
/* Decleration of variable with type double
 * and initialise it with 19.362
*/
double dl = 19.362;

```