ARRAYS

- An array is a consecutive group of memory locations that are of the same type.
- To refer a particular location or element in the array, we specify the name of the array and the position number of the particular element.

Declaring Arrays

Arrays occupy space in memory. To specify the type of the elements required by an array, use a declaration of the form:

type arrayName [arraySize] ;

- The compiler reserves the appropriate amount of memory.
- The arraySize must be an integer constant greater than zero.

  ex: int C[12]; // C is an array of 12 integers

Arrays can be declared to contain values of any non-reference data type. For example, an array of type string can be used to store character strings.

Examples Using Arrays

EXAMPLE 1: Declaring an Array & using Loop to Initialize the Array's Elements

- The program declares a 10-element integer array **n**.
- Lines a-b use **For** statement to initialiaze the array elements to zeros.
- Like other automatic variables, automatic arrays are not implicit initialised to zero.
- The first output statement (line c) displays the column headings for the columns printed in the subsequent for statement (lines d-e), which prints the array in tabular format.

```
int n[ 10 ] ; // n is an array of 10 integers

void setup () {

}

void loop () {
   for ( int i = 0; i < 10; ++i ) // initialize elements of array n to 0 {
      n[ i ] = 0; // set element at location i to 0
      Serial.print (i) ;
      Serial.print ('\r') ;
   }
   for ( int j = 0; j < 10; ++j ) // output each array element's value {
      Serial.print (n[j]) ;
      Serial.print ('\r') ;
   }
}
```

RESULT

ELEMENT	VALUE
0	0
1	0
2	0
3	0
4	0
5	0
6	0
7	0
8	0
9	0

EXAMPLE 2: Initializing an Array in a Declatration with an Initializer List

- The elements of an array can be initialised in the array declaration by following the array name with an equal-to sign and a brace-delimited comma-separated list of initialisers.
- The program uses an initialiser list to initialise an integer array with 10 values (line a) and points the array in tabular format (lines b-c).

```
// n is an array of 10 integers
int n[ 10 ] = { 32, 27, 64, 18, 95, 14, 90, 70, 60, 37 } ;

void setup () {

}

void loop () {
   for ( int i = 0; i < 10; ++i ) {
      Serial.print (i) ;
      Serial.print ('\r') ;
   }
   for ( int j = 0; j < 10; ++j ) // output each array element's value {
      Serial.print (n[j]) ;
      Serial.print ('\r') ;
   }
}
```

RESULT

ELEMENT	VALUE
0	32
1	27
2	64
3	18
4	95
5	14
6	90
7	70
8	60
9	37

EXAMPLE 3: Summing the Elements of an Array

Often, the elements of an array represent a series of values to be used in a calculation. ex: if the elements of an array represent exam grades, a professor may wish to total the elements of the array and use the sum to calculate the class average for the exam.
The program sums the values contained in the 10-element integer array **a**.

```
const int arraySize = 10; // constant variable indicating size of array
int a[ arraySize ] = { 87, 68, 94, 100, 83, 78, 85, 91, 76, 87 };
int total = 0;

void setup () {

}
void loop () {
   // sum contents of array a
   for ( int i = 0; i < arraySize; ++i )
      total += a[ i ];
   Serial.print ("Total of array elements : ") ;
   Serial.print(total) ;
}
```

RESULT

Total of array elements: 849