**Strings and Arrays of Characters**

Declaring a string

In C, a string is an **array** of characters.

An array is a list of things which are all of the same **type** (integers, characters, doubles etc).

The things in the array are called **elements**.

What makes an array useful, is that each element can be accessed by its **index** – its place in the array.

To declare a string, we do this-

char smallString[5]; // a string that can hold 5 characters

This reserves a piece of memory for the string, big enough to hold 5 characters. It also associates the memory area with the name ‘smallString’.

To declare and initialise a string, we do this -

char myString[] = “Hello World!”;

This statement declares an array of characters, of unknown size – hence [].

It also initialises the string with the text (“Hello World!”);

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| H | e | l | l | O |  | W | o | r | l | d | ! | \0 |

How strings a stored in memory

Strings are stored as a series of characters in memory locations, each one following directly after the other. They are **null-terminated** – they have a null character (zero) at the end, to show where the end is. The \0 is a way of writing null, just as \n and \t are newline and tab in printf()

When we declare a string like :-

char myString[] = “Hello World!”;

The compiler puts the null at the end for us. We don’t need to put it in, in this case.

Accessing characters in a string

The string is treated as an array of characters. We use the array operator – square brackets to access the elements.

So

mystring[0] is ‘H’

mystring[1] is ‘e’

mystring[2] is ‘l’

Printing strings

We can print the whole string in one go:

printf(“my string is %s\n”, myString);

We just have to give the name of the string – myString.

We could print it one character at a time …. if we knew how long it was.

Some String Functions

The declarations live in the header file string.h

strlen – length of a string

strlen gives us the length of a string, without the final null.

char myString[] = “Hello World!”;

int length = strlen(myString);

printf(“myString is %d characters long\n”, length);

Now that we know the length, we can do things with the individual characters.

What kind of loop do we use when we know the number of times to loop?

char myString[] = “Hello World!”;

int length = strlen(myString);

printf(“myString is %d characters long\n”, length);

for (int i = 0; i < length; i++) {

printf(“character %n is %c\n”, i, myString[i]);

}