Subsections

Arrays and Strings

In principle arrays in C are similar to those found in other languages. As we shall shortly see arrays are defined slightly differently and there are many subtle differences due the close link between array and pointers. We will look more closely at the link between pointer and arrays later in Chapter 9.

Single and Multi-dimensional Arrays

Let us first look at how we define arrays in C:

```
  int listofnumbers[50];
```

BEWARE: In C Array subscripts start at **0** and end one less than the array size. For example, in the above case valid subscripts range from 0 to 49. This is a **BIG** difference between C and other languages and does require a bit of practice to get in ***the right frame of mind***.

Elements can be accessed in the following ways:-

```
  thirdnumber=listofnumbers[2];
                listofnumbers[5]=100;

Multi-dimensional arrays can be defined as follows:


  int tableofnumbers[50][50];

for two dimensions.

For further dimensions simply add more [ ]:


  int bigD[50][50][40][30].....[50];

Elements can be accessed in the following ways:


  anumber=tableofnumbers[2][3];
                tableofnumbers[25][16]=100;
```

Strings

In C Strings are defined as arrays of characters. For example, the following defines a string of 50 characters:

```
  char name[50];
```

C has no string handling facilities built in and so the following are all illegal:

```
char firstname[50],lastname[50],fullname[100];

              firstname= "Arnold"; /* Illegal */
              lastname= "Schwarznegger"; /* Illegal */
              fullname= "Mr"+firstname
                             +lastname; /* Illegal */
```

However, there is a special library of string handling routines which we will come across later.

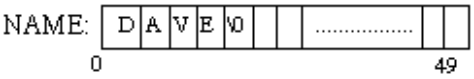
To print a string we use printf with a special **%s** control character:

```
printf(``%s'',name);
```

NOTE: We just need to give the name of the string.

In order to allow variable length strings the \0 character is used to indicate the end of a string.

So we if we have a string, char NAME[50]; and we store the ``DAVE'' in it its contents will look like:



Exercises

Exercise 12335

Write a C program to read through an array of any type. Write a C program to scan through this array to find a particular value.

Exercise 12336

Read ordinary text a character at a time from the program's standard input, and print it with each line reversed from left to right. Read until you encounter end-of-data (see below).

You may wish to test the program by typing

```
prog5rev | prog5rev
```

to see if an exact copy of the original input is recreated.

To read characters to end of data, use a loop such as either

```
char ch;
while( ch = getchar(), ch >= 0 ) /* ch < 0 indicates end-of-data */
```

or

```
char ch;
while( scanf( "%c", &ch ) == 1 ) /* one character read */
```

Exercise 12337

Write a program to read English text to end-of-data (type control-D to indicate end of data at a terminal, see below for detecting it), and print a count of word lengths, i.e. the total number of words of length 1 which occurred, the number of length 2, and so on.

Define a word to be a sequence of alphabetic characters. You should allow for word lengths up to 25 letters.

Typical output should be like this:

```
        length 1 : 10 occurrences
                length 2 : 19 occurrences
length 3 : 127 occurrences
        length 4 : 0 occurrences
        length 5 : 18 occurrences
                ....
```

To read characters to end of data see above question.

Dave Marshall
1/5/1999