Lesson 7 Arrays and Strings

Arrays in C

An array is a collection of data that all have the same type. The data values can change but the data type will not. Since an array is thus a variable, it must be declared before use. The compiler needs to be told the size of the array and the type of elements that are to be stored in the array.

The following code defines an array to hold ten integers.

```
int hold[10];
```

Now that the array has been declared, we need to know how to access the elements of the array. In C, the first element of an array is indexed with the number 0. Thus the elements in our array have index from 0 through to 9 ie 10 elements. We can assign the array elements as follows

```
for(i=0;i<=9;i++) hold[i] = i;
```

Data can be entered into an array using scanf as follows

```
 for(i=0;i<=9;i++)
    {
     printf(" Enter interger number \n");
     scanf("%d",&hold[i]);
    }
```

To read data from an array, say for example to calculate the average of the ten numbers in the array we can use the following

```
total =0.;
for(i=0;i<=9;i++)
total += hold[i];
printf("mean = %f \n"total/10.);
```

If we need to change the size of an array throughout a program before compiling, its sometimes useful to use the **#define** statement. By using

```
#define SIZE 10

int hold[SIZE]
```

we can change the size of the array by just editing one line. This is common practice in C when using arrays. Thus changing the size of the array from 10 to 20 is achieved by

```
#define SIZE 20
```

A word of warning. C does not warn you when an array subscript exceeds the size of the array. This can be the cause of many bugs in a program using arrays. It might be useful to check that array subscripts are greater than or equal to 0 and less than SIZE.

Array Initialisation

We can initialise an array by any of the following statements.

```
int hold[4]  = {5,6,7,0};
int hold[]   = {5,6,7,8};
int hold[4]  = { 0 } ;
```

For characters arrays the following statements can be used

```
char s1[3]  = "Hi";
char s2[10] = "there,";
char s3[]   = "Everyone";
char name[4]   = {'T','o','m','\n'}
```

Strings in C

A string is a group of characters, usually letters of the alphabet. In order to format your printout in such a way that it looks nice, has meaningful names and titles, and is aesthetically pleasing to you and the people using the output of your program, you need the ability to output text data. In C, a string is not a formal data type but an array of type char. So the above statements can be used to store the string s1="Hi" .

Be mindful that it is actually a character array with the first element s[0]='H' and second element s[1]='i'.

String Constants

We have already seen the use of string constants. For example using the printf statement to print a persons name: `printf("%s","Patrica");`

Note that each character of the string occupies one byte in memory and the last character of the string is '\0' , the NULL character (character with a value of zero). The NULL character is used to signal the end of the string.

String Variables

The following code gives a typical example of how a string variable is declared and used

```
int main(void)
{
 char name[20];
 printf("Please enter your name: ");
 scanf("%s",name);
 printf("Greetings%s.",name);
 return(0);
}
```

Note that in the above program, although the variable name can hold twenty characters, we can only type in nineteen since one byte is needed for the NULL character.

Note also that when using the scanf() function the address operator(&) was not required. This is because the name of the array variable *name* is an address. This last point is very important, the address of standard variable types, for example int, char, float, double require the the & in front of the name of the variable. For arrays, just the name of the variable is required. This will be discussed in greater detail later.

Input and Output functions for Strings

The C programming language contains many useful functions for string manipulation. See string.h for more details. For now we will look at just two functions used in input and output. The gets() function gets a string from the keyboard. The function puts() outputs a string to the screen. The following program reads in and outputs a string.

```
#include<stdio.h>
int main(void)
     {
      char name[20];
      puts("INPUT YOUR NAME (19 characters)");
      gets(name);
      puts("Hello");
      puts(name);
      puts(strcat("Hello ",name));
      return(0);
      }
```

The above code may compile with a depreciation warning around the gets function. You can replace the line with the following to get rid of the warning.

```
fgets(name,sizeof(name),stdin);
```

The code below use fgets and other functions from string.h

```
#include<string.h>
#include<stdio.h>
int main(void)
    {
     char name[20];
     char mess[]= "Hi there, ";
      int len;
      puts("INPUT YOUR NAME (19 characters)");
      fgets(name,sizeof(name),stdin);
      puts("Hello");
      puts(name);
       strcat(mess,name);
       puts(mess);
       len = strlen(mess);
       printf("The length of the message is  %d\n",len);
       return(0);
     }
```

Initialising Strings

To initialise a string in C, there are two methods. The first is similar to the example above *char nam[]={`D`,`a`,`n`,`\0`};*. The second is *char nam[] = "Dan"*. The NULL character is inserted automatically when the second method is used. As an exercise to help you fully understand strings, you should write a program which will get you name from the keyboard and print it out one character at a time. There are functions in ctype.h which can check that each character is upper case, lower case, alphanumeric etc

You can also print out the address in memory of each character in the string.

ARRAYS OF MORE THAN ONE DIMENSION

It is possible in C to have an array of arrays. This is really just a two dimensional array. Consider for example plotting a graph of a function y = x*x, between 0 and 5. If we wish to store the data for plotting then we will need a two dimensional array. Six rows and two columns are needed. The example below shows how easy it is to set up such a structure.

```
int main(void)
   {
    int graph[6][2];
    int row,col;
    for(row=0;row<=5;row++)
     for(col=0;col<=1;col++)
        graph[row][col]=pow(row,2);
     for(row=0;row<=5;row++)
      for(col=0;col<=1;col++)
        printf("x[%d]=%d  \n",row,graph[row][col]);
      return(0);
    }
```

Initialising a two dimensional array

To initialise a two dimensional array we could use the following. This may not be the best solution for so large arrays – example to store 10 years of rainfall. (10 rows, 365 columns)

```
int mat[5][2]= {   {0,0},
                   {1,1},
                   {2,4},
                   {3,9},
                   {4,16} };
```

Arrays as Arguments to Functions

Since an array is a variable, it can be passed as an argument to a function. An array is passed to a function by its address. Since the name of the array is the address, we pass this down to the function. Suppose we have an array called list which holds 10 integers. The address of the first element of the array is given by *list* or *&list[0]*. Thus its enough to pass *list* down to the function requesting it. This is called passing by reference. The actually array is not passed down or copied to the function – but the address in memory of the beginning of the array.

The function can access the original array since it not has its address.

Suppose that we want to write a function to assign values to a one dimensional array, the following code describes how this is achieved

```
#include<stdio.h>
void fill_array(int[],int); /*function prototype*/

void main(void)
   {
    int j;
    int list[5];
    fill_array(list,5); /* function call */
    for(j=0;j<5;j++)
       printf("list[%d]=%d \n",j, list[j]);
    }
```

```
void fill_array(int hold[],int n) /*function defination*/
   {
   int i;
   for(i=0;i<n;i++)
    {
     printf(" Enter interger number \n");
     scanf("%d",&hold[i]);
    }
   }
```

In some some ways, passing a two dimensional array to a function is similar to passing an array of one dimension, there is however one major difference.

As before we pass the name of the array to the function ie the address in memory of the array. Now though instead of list[] been the argument to the function we write list[][cols] ie we don't need to tell the compiler how many rows there are. This is because all that is needed is the number of cols. For example, suppose that we want to access the variable at address list[3][1]. The function multiplies the the row index 3 by the number of of elements per row (ie no. of cols), then adds the col index 1. Thus the result is 3*col + 1.

How would you modify the above code to work with a 2x2 array of integers ?

Array of Strings

Suppose we want to store a list of four names. To do this we need an array of size four. Each entry in the array will be a string. Thus we are using a two dimensional array. The following program asks the user to type in there name. If there name is on the list there are allowed into the club.

```
#define num 4
#define size 20
#include<stdio.h>
#include<string.h>

int main(void)
    {
     int i,check=0;
     char name[size];
     static char list[num][size] = { "John",
                                     "Pat",
                                     "June",
                                     "Kate" };
     puts(" ENTER YOUR NAME");
     scanf("%s",&name);
     for(i=0;i<=3;i++)
        if(strcmp(&list[i][0],name)==0) check=1;

     if(check)
        printf("YOU MAY ENTER");
     else
        printf("SORRY ,NO ENTRY");
     return(0);
     }
```

When initialising an array of strings, we do not need to put the names in braces indicating that the string is an array, the compiler already knows this fact. However the list of names must be put in braces to indicate that we have an array of strings. The string function strcmp() is used to compare the names. The function has two arguments, string1 and string2. It returns a zero if both are the same and a non zero if they are different.

Programming Exercises 1

1. Write a program that uses an array and that reads in the exam marks of a class (10 students) and calculates the average class mark.
2. Write a program which reads 10 numbers into an array and then sorts the values from smallest to largest.
3. Repeat exercise 2 only this time pass the array to a function which will do the sorting.
4. Write a program, using an array of integers with 2 functions, that will return the largest and smallest value of the array.
5. Write a program which will fill an array with values for the function

```
        y = x^2 + 6x -2         for x = 1
to 10 with step size 1.
```

6. Repeat exercise 5 for a range of values of -1 to 1 with a step size of 0.1.
7. Write a program that reads in the years monthly temperatures and calculates the average temperature for the year. Calculate the standard deviation about the mean from the formula.
8. A vector is a mathematical object which is just an array of numbers eg v = (3,4,1) and u = (6,7,2) are examples of three dimension vectors. Vector addition and multiplication are defined as follows

```
          v + u = (3+6,4+7,1+2)   = (9,11,3)
v * u = 3*6 + 4*7 + 1*2 = ?
```

   Write two functions, one that adds two vectors and one that multiplies two vectors.

Programming Exercises 2

1. Write a program that reads in and prints out your name and address into an array using the scanf and printf functions.
2. Repeat 1 using gets and puts.
3. Write a short program that counts the number of letters in a persons name which they have type in at the keyboard.
4. Write a program which reads in a line of text from the keyboard and also a character from the keyboard, and then prints out the number of times that the character appeared in the line of text.
5. Write a function that will capitalise all the characters in a string.
6. A matrix can be defined as a 2-dim array. Conside the matrices below

$$\begin{pmatrix} 4 & 3 \\ 2 & 4 \end{pmatrix} \quad \begin{pmatrix} -1 & 0 \\ 2 & 3 \end{pmatrix},$$

   These are examples of *2x2* matrices. We can add/subtract and multiply matrices as defined earlier. Write a program which allows the program user to enter the elements of two matrices and also the operation *+,- or* * which they want to be performed on the matrices. Print the results out in matrix form.

7. Write a program which prints out a table of values to be plotted by a graphics package of the following function

$$y = -3x + 11 \text{ for } -10 < x < 10$$

x	y=-3x +11
-9	38
-8	35
.	.
.	.
.	.
.	.
.	.
.	.
9	-16

8. Write a program in which the user enters 5 cities. The program should store them first and then display them to the user.