Arrays

An array is an identifier to store a set of data with common name. Note that a variable can store only a single data. Arrays may be one dimensional or multi dimensional.

<u>Defining an array one dimensional arrays</u>

Definition: Arrays are defined like the variables with an exception that each array name must be accompanied by the size (i.e. the max number of data it can store).For a one dimensional array the size is specified in a square bracket immediately after the name of the array.
The **syntax** is

data-type array name[size];

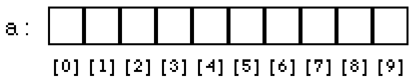
So far, we've been declaring simple variables: the declaration

```
int i;
```

declares a single variable, named `i`, of type `int`. It is also possible to declare an *array* of several elements. The declaration

```
int a[10];
```

declares an array, named `a`, consisting of ten elements, each of type `int`. Simply speaking, an array is a variable that can hold more than one value. You specify which of the several values you're referring to at any given time by using a numeric *subscript*. (Arrays in programming are similar to vectors or matrices in mathematics.) We can represent the array `a`

above with a picture like this: 

eg:

```
int x[100];
float mark[50];
char name[30];
```

<u>Note</u>: With the declaration int x[100],computer creates 100 memory cells with name x[0],x[1],x[2],………,x[99].Here the same identifier x is used but various data are distinguished by the subscripts inside the square bracket.

Array Initialization

Although it is not possible to assign to all elements of an array at once using an assignment expression, it is possible to initialize some or all elements of an array when the array is defined. The syntax looks like this:

```
int a[10] = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9};
```

The list of values, enclosed in braces `{}`, separated by commas, provides the initial values for successive elements of the array.

If there are fewer initializers than elements in the array, the remaining elements are automatically initialized to 0. For example,

```
        int a[10] = {0, 1, 2, 3, 4, 5, 6};
```

would initialize `a[7]`, `a[8]`, and `a[9]` to 0. When an array definition includes an initializer, the array dimension may be omitted, and the compiler will infer the dimension from the number of initialisers. For example,

```
        int b[] = {10, 11, 12, 13, 14};
```

Example :

```
int x[ ] ={0,1,2,3,4,5}; or
int x[6]={0,1,2,3,4,5};
Even if the size is not mentioned (former case) the values
0,1,2,3,4 are stored in x[0],x[1],x[2],x[3],x[4],x[5].If the
statement is like
                   int x[3]={0,1,2,3,4,5};
then x[0],x[1],x[2] are assigned the values 0,1,2.
```

Note: If the statement is like

int x[6]={0,1,2};

then the values are stored like x[0]=0, x[1]=1, x[2]=2, x[3]=0, x[4]=0 and x[5]=0.

Processing one dimensional array

1) Reading arrays: For this normally we use for- loop.
If we want to read n values to an array name called ‘mark’ , the statements look like

```
        int mark[200],i,n;
        for(i=1;i<=n;++i)
          scanf("%d",&x[i]);
```

Note: Here the size of array declared should be more than the number of values that are intended to store.

2) Storing array in another:
To store an array to another array. Suppose a and b are two arrays and we want to store that values of array a to array b. The statements look like

```
float a[100],b[100];
 int I;
 for(i=1;i<=100;++i)
 b[i]=a[i];
```

Problem: To find the average of a set of values.

```
#include<stdio.h>
main( )
{
 int x,i;
 float x[100],avg=0;
 printf("\n the no: of values ");
 scanf("%d",&n);
 printf("\n Input the numbers");
 for(i=1;i<=n;++i)
    {
      scanf("%f",&x[i]);
      avg=avg+x[i];
     }
 avg=avg/n;
 printf("\n Average=%f",avg);
 }
```

PASSING ARRAYS TO FUNCTION

Remember to pass a value to a function we include the name of the variable as an argument of the function.Similarly an array can be passed to a function by including arrayname (without brackets) and size of the array as arguments.In the function defined the arrayname together with empty square brackets is an argument.

Ex:

(calling function)-avg=average(n,x); where n is the size of the data stored in the array x[].

(function defined)- float average(int n,float x[]);

Now let us see to use a function program to calculate the average of a set of values.

```
#include<stdio.h>
 float average(int n,float y[]);
main()
{
  int n;
  float x[100],avg;
  printf("\n Input the no: of values");
  scanf("%d",&n);
  printf("\n Input the values");
  for(i=1;i<=n;++i)
     scanf("%f",&x[i]);
     avg=average(n,x);
      printf("\n The average is %f",avg);
}
float average(int n, float y[]);
{
  float sum=0;
  int i;
   for(i=1;i<=n;++i)
      sum=sum+y[i];
      sum=sum/n;
      return(sum);
}
```

Note:

1) In the function definition the array name together with square brackets is the argument. Similarly in the prototype declaration of this function too, the array name with square brackets is the argument

2) We know that changes happened in the variables and arrays that are in function will not be reflected in the main (calling) program even if the same names are usual. If we wish otherwise the arrays and variables should be declared globally. This is done by declaring them before the main program.

Ex:
```
   #include<stdio.h>
     void arrange(int n,float x[]);
     main();
   {
      ……….
      arrange(n,x);
      …………..
    }
      arrange(int n,float x[]);
      {
        ……….
        return;
       }
```
Problem : Write a program to arrange a set of numbers in ascending order by using a function program with global declaration.

MULTI-DIMENSIONAL ARRAYS

Multi-dimensional arrays are defined in the same manner as one dimensional arrays except that a separate pair of square brackets is required to each subscript.

```
Example:  float matrix[20][20]    (two dimensional)
                 Int x[10][10][5]  (3-dimensional)
```

Initiating a two dimensional array we do as int x[3][4]={1,2,3,4,5,6,7,8,9,10,11,12}

Or
```
int x[3][4]={
        {1,2,3,4};
        {5,6,7,8};
        {89,10,11,12};
     }
```
NOTE: The size of the subscripts is not essential for initialization. For reading a two dimensional array we use two for-loop.

Example:

```
for(i=1;i<=2;++i)
   for(j=1;j<=3;++j)
         scanf(“%f”,&A[i][j]);
```

NOTE: If `x[2][3]` is a two dimensional array, the memory cells are identified with name `x[0][0],x[0][1],x[0][2],x[1][0],x[1][1] and x[1][2].`

ARRAYS AND STRINGS.

A string is represented as a one dimensional array of character type.

Example : char name[20];

Here name is an array that can store a string of size 20.
If we want to store many strings(like many names or places) two dimensional array is used. Suppose we want to store names of 25 persons, then declare name as char name[25][ ]. Note that the second square bracket is kept empty if the length of string is not specified.

If the declaration is char name[25][30], 25 names of maximum size 30 can be stored. The various names are identified by name[0], name[1], name[2],…….., name[24]. These names are read by the command

```
For( i=0; i<25,++i)
   Scanf( “%[^\n]”, name(i));
```

PROBLEM: Write a program to store the names and places of students in your class.