Chapter 2 : Indexed variables

A series of variables ordered by index and treated as a single block is called indexed variables. These variables can be of the same type to take the form of an **array**. An array, that contains character data-type variables, is known as a **string**. Indexed variables with different data type are called a **structure**. This chapter has extensively covered two types of indexed variables array and string. The structure it will be discussed in the next chapter.

Part I : One dimensional Array

Definition

- An array is a method to treat an ensemble of multiple variables with the same data-type as a one class.
- Each variable in the array represents an **element** in this array.
- Each variable occupy a storage space in memory, and the storage space of all of the array elements is called **contiguous memory** allocation.
- Elements in array are ordered by **index** in which the 0th index refers to the first element in the array.

Array declaration

Declaring an array is done in the following way

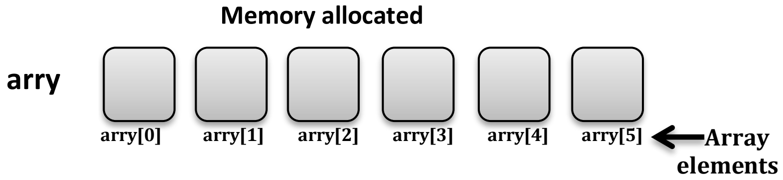
```
data_type array_name[array_size];
```

- **Data-type** : is defined according to the data-type of the array elements.
- **Array-name**: is chosen by the programmer and it is important and used to refer to the array element.
- **Array size**: is the number of elements in the declared array.

Example of array declaration:

```
int arry[6];
```

int is the data type of the six elements in this array, and this statement can be pictured as follows



It is very important to note the following:

- The array element is referred to by its name followed by an index placed between square brackets.
- The index of first element of the array is 0, and therefore always there is a difference of one between the index of the last element and the size of the array.

Array initialization

The method of assigning an initial value to a variable is known as initialization. There are many ways to initialize an array.

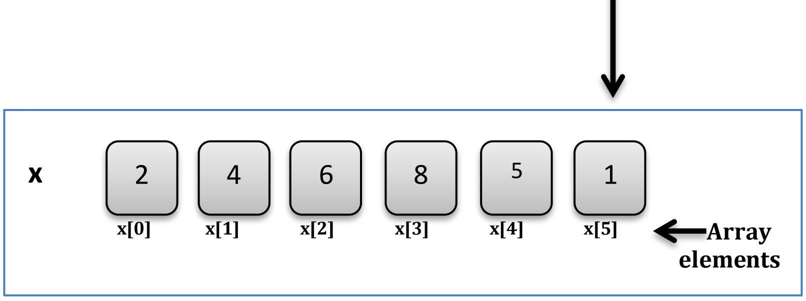
1. **Initialization during declaration step**

Syntax:

```
data_type array_name[array_size]={element 1, element 1, element 1,...};
```

Example:

`int x[6]={2, 4, 6, 8, 5, 1};`



N.B. In this case, it is not necessary to specify the size of the array and the initialization remains correct in this way

`int x[ ]={2, 4, 6, 8, 5, 1};`

2. Initialization using loops

Example:

Program	*Output*

#include <stdio.h>
int main()
{
int i,x[6];
for(i=0;i<6;i++)
{
printf("Enter the element x[%d]:\n", i);
scanf("%d", &x[i]);
}
return 0;
}
	Enter the element x[0]:
	2
	Enter the element x[1]:
	4
	Enter the element x[2]:
	6
	Enter the element x[3]:
	8
	Enter the element x[4]:
	5
	Enter the element x[5]:
	1

Access array elements

Array elements are manipulated as variables, and index is the only difference between elements and variables. The index requires an arrangement in memory to distinguish these elements from each other. Therefore, the element can only be accessed through this index.

Access, here, means two things:

i) Using the value of the element
ii) Dealing with the memory address of the element.

array_name[0] accesses the first element in the array, while array_name[1] accesses the second element, etc.

Example :

Program	*Output*

#include <stdio.h>
int main()
{
  int i,a,b=2, y[5]={5,b,1,4,6};
  a=y[0];
  printf("a=%d \n %d",a, y[1]);
  return 0;
}
	a=5
	2

An opportunity presents itself here to demonstrate a straightforward method for assigning values to elements of an array.

```
int arr[3] ;
arr[0]=5 ;
arr[1]=12 ;
arr[2]=16 ;
```

Input/output an element of array

- To input the first and second elements of the array float arr[n], we just use scanf() function as follows

  scanf("%f%f", &arr[0], &arr[1]) ;

- To output the first and the second elements of the last array we just use printf() function in this way

  printf("%f%f", arr[0], arr[1]) ;

The length of array

The meaning of length is identical to that of size in most cases but the length in some how have a relation with number of bytes. Anyway, determining the number of bytes in the array can be achieved by simply using this relation.

Total number of bytes=number of bytes in one element × the number of elements.

Example :

int Arr[]={5, 4, 2, 51, 25, 45} ;

The size of this array is **6** and the size of a one integer element is **4** bytes, and this gives

The total number of bytes in Arr[]=6×4=24 bytes

In C, the number of bytes in a variable is given by using **sizeof()** operator, and the following program explain how we can do this

Program	*Output*

#include <stdio.h>
 	
int main() {
   int b, Arr[]={5,4,2,51,25,45};
   b=sizeof(Arr);
 	
   printf("Number of bytes in one element=%d bytes\n", sizeof(Arr[0]));
 	
   printf("Total number of bytes in array=%d bytes",b);
 	
   return 0;
}
	Number of bytes in one element=4 bytes
	Total number of bytes in array=24 bytes

The length (size) of an array is given by this relation

$$length = \frac{sizeof(array)}{sizeof(array[0])}$$

Example :

Program	*Output*

#include <stdio.h>
 	
int main() {
   int length, A[]={5,10,15,20,25,30,35,40};
   length=sizeof(A)/sizeof(A[0]);
 	
   printf("length=%d", length);
 	
   return 0;
}
	
	length=8

Operations on Arrays

1- Program to find sum of two arrays elements

Program

```
#include <stdio.h>

int main()
{

int  k, ar1[5]={10, 12, 5, 1, 7};
int ar2[5]={2, 1, 4, 8, 6}, ar3[5];


     for(k=0;k<5;k++)
     {
        ar3[k]=ar1[k]+ar2[k];
        printf("ar3[%d]=%d\n",k, ar3[k]);
     }
   return 0;
}
```

Output

```
ar3[0]=12
ar3[1]=13
ar3[2]=9
ar3[3]=9
ar3[4]=13
```

2- Program to swap the values of the elements of two arrays

Program

```
#include <stdio.h>

   int main() {
      int i,temp, a1[3]={1,2,3}, a2[3]={4,5,6};
      for(i=0; i<3;i++)
      {
         temp=a1[i];
         a1[i]=a2[i];
         a2[i]=temp;
         printf("a1[%d]=%d  \t  a2[%d]=%d\n",i,
a1[i], i, a2[i]);
      }


   return 0;
}
```

Output

```
a1[0]=4            a2[0]=1
a1[1]=5            a2[1]=2
a1[2]=6            a2[2]=3
```

3- Program to find sum and average of elements in array

Program	*Output*

#include <stdio.h>
int main()
{
int b, Ar[]={5,2,6,4,7,2};
b=sizeof(Ar)/sizeof(Ar[1]);
float average;
int a,i, sum=0;
for(i=0;i<b; i++)
{
sum=sum+Ar[i];
}
 	
average=sum/b;
printf("sum=%d \naverage=%f",sum, average);
 	
return 0;
}
	sum=26
	average=4.000000