BATU-EXAM

Made by batuexams.com

at MET Bhujbal Knowledege City

Computer Programing in C Department

UNIT - IV

ARRAYS in C

* Introduction :-

— As we know, in C language there are three types of data types:

1. Primary / Fundamental data types: int, float, char.
2. Derived data types: array, function, pointer
3. User defined data types: structure, union, enumeration.

— Primary data types are int, float and char. We already learned primary data types in detail.

— Arrays are called derived data types because it uses primary data types as its basic data types.

* What is Array?

— Let's consider a situation where a program needs number of similar types of data elements to be stored.

— A variable is used to store one data elements at a time. So to store number of data elements; number of variables are required. This solution has many problems:

→ As the list of variables increases the length of program also increases.

→ To manipulate those variables several assignment statements also needed.

→ Programmer needs to remember names of all the variables.

— An alternate solution to above situation is to store similar type of data elements is create an array.

Definition

- Array is collection of elements of similar data types referred by the same variable name. Contiguous memory block is allocated to all these array elements.
- Array elements are of same data type i.e. once array is declared as integer, then all values which are present in an array will be of type integer only.

✱ Need of Array :-

* First Scenario

To store weight of 5 children we need 5 variables as follows.

```
Void main()
 {
  int rajesh, suvarna, surya, pritesh, chitra;
  rajesh = 25;
  suvarna = 20;
  surya = 30;
  chitra = 22;
  pritesh = 21;
 printf("\n **** Weight of Children ****");
 printf("\n Suvarna = %d \n Chitra = %d. \n Rajesh = %d \n
        pritesh = % \n Surya = %d, suvarna, chitra, rajesh,
         pritesh, surya);
 }
```

- If count of variables is small then this idea will work. Similarly to store weight of 100, 1000, or more children programmer should declare that much

number of variables which is bad idea. Also it is difficult to retrieve particular child's weight as the programmer needs to remember all the variables.

Second Scenario

— Another idea is to use single variable to store these information as shown in following program.

```
Void main()
 {
   int weight;
   weight = 25;
   weight = 20;
   printf("\n weight of child = %d", weight);
 }
```

— No doubt, this program will print the most recently updated value of the variable i.e. 20. Because initially the variable weight is assigned to 25; and when 20 is assigned to weight. Previous value of weight i.e. 25 is lost new value i.e. 20 is assigned to weight.

— A situation in which more than one values needs to be stored at a time, this idea will not work.

— Array can handle lots of elements using single name. Instead of declaring lots of variables, used a subscripted variable called as array to store the weight of children. Using the subscript we can manage random elements of array.

Example

— consider the following group

weight = { 25, 20, 30, 21, 22 }

- It represents weight of five children. To refer the elements 20 of the group the notation weight 2 is used. Similarly, to refer the element 22 of the group notation weight 5 is used.
- But in case of array, the element 25 is referred as weight[0]; because the array

```
Weight[i]
   ↓      └→ subscript.
 array
 name
```

- Element's counting start with 0 instead of 1. Similarly, the element 22 referred as weight[4]. In general to refer any element of an array following notation should be used.
- Here weight is the subscripted variable (array), whereas i is subscript. Where is in the range 0-4 in our example.

Need of array -

- Let's consider the following scenario for proper understanding the need of arrays.
- Array variables are needed because of following requirements of user:
  - → Store multiple values in the variable with same name.
  - → Store multiple values (with the same datatypes) together i.e. store them one after another.
  - → For easy access or retrieval of stored data elements.

— Note that the elements should be of same data type means it can be group of integers (int s), group of real numbers (float s or double s), group of characters (char s) etc. Usually array of characters called as string.

✯ Characteristics of an Array :-

— Array can hold only similar type of data.
— Array is a subscripted variable which holds several elements at a time.
— Using subscript random elements can be referred.
— Contiguous memory is allocated for elements of array
— Always the array-name is followed by [ ] which tells the compiler that we are dealing with an array. Between [and] bracket int type of value is needed.
— An array is a collection of similar elements.
— The first elements in the array is numbered as 0, so the last element is 1 less that the size of the array.
— Before using an array its type and dimension must be declared.

✯ Array Declaration and Initialization

✯ Array Declaration –

— While using array in a program the first step is to declare an array.

- An array declaration is performed to tell the compiler what type of data the array will hold and how much data elements it can hold.

Syntax

Following syntax is used to declare an array.

```
data_type array_name[size];
```

Example

- To declare an array which holds age of five persons following statement is used.

```
int age[5];
```

- Here, int is the data type of the array age and the number 5 specifies the maximum number of elements which the age array can hold.

Arrangement of array elements in memory after declaration:-

- As soon as the control goes to the statement of declaring array age[], Immediately contiguous memory block of 10 bytes get reserved in memory for 5 integers.
- According to 16 bit compiler each integer occupies 2 bytes in memory; so 5*2 = 10 bytes should be reserved for array of 5 integers.

The memory structure would be as follows:

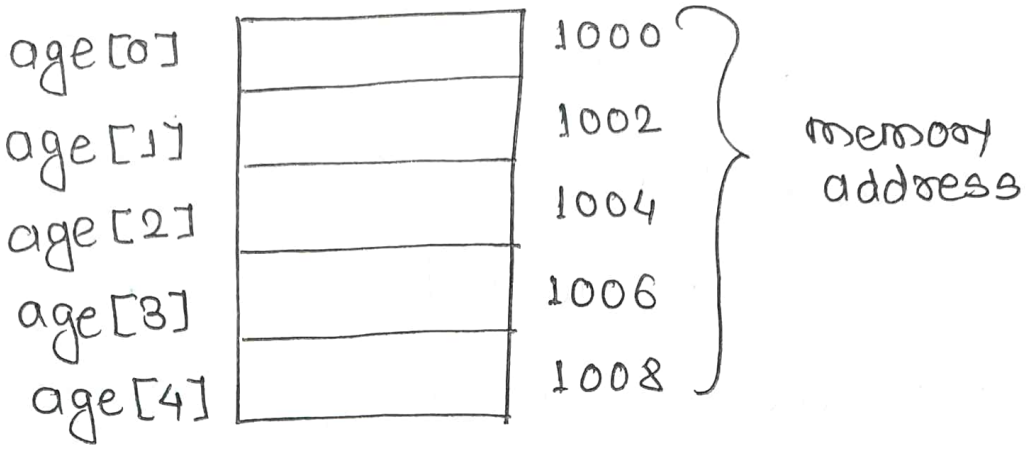


figure:- Memory structure after declaring array of 5 integers.

- Size of array can be calculated by using the sizeof() operator. Like:

```
sizeof(age);
```

it will give the memory space occupied by the array in bytes. Suppose age is an intger array and has the capacity of storing 5 numbers. Then the sizeof(age) will output 5*2 = 10.

- All the locations will contain garbage values until ~~the~~ we initialize the array.

✯ Array Initialization :-

- Array initialization is the process of assigning the value to array elements. After declaring the array it will contain garbage values until we initialize it. There are different ways to initialize an array.

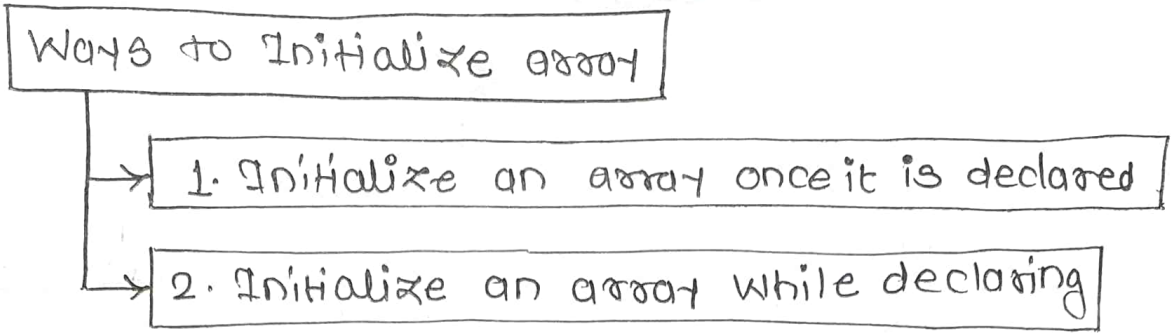


fig:- Ways to initialize array.

1. Initialize an array once it is declared :-

Example:-

```
int age[5];    // declaring array.
age[0] = 20;   // initialize 20 to first location.
age[1] = 25;   // initialize 25 to second location.
age[2] = 30;   // initialize 30 to third location.
age[3] = 40;   // initialize 40 to fourth location.
age[4] = 45;   //initialize 45 to fifth location.
```

2. Initialize an array while declaring :-

— We can combine the declaration and initialization of array like:

Example:-

```
int age[5] = {20, 25, 30, 40, 45};
int age[] = {20, 25, 30, 40, 45};
```

— If the array is initialized and declared at a time then the size of array is optional (as shown in example). The values going to be initialized are written between two curly braces (i.e. { and }) and separated by comma (,).

— Above two statements assigns the value 20 to first element i.e. age[0], 25 is assigned to second element i.e. age[1], 30 is assigned to third element i.e. age[2], 40 is assigned to fourth element i.e. age[3], and 45 is assigned to fifth element i.e. age[4].

— Arrangement of array elements in memory after initialization is shown below.

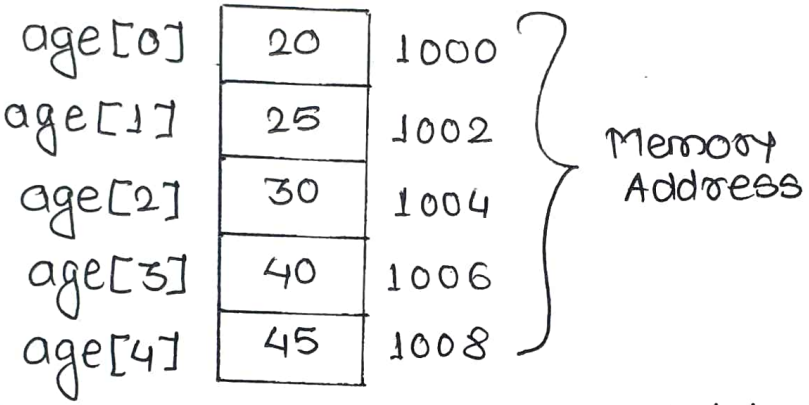


figure:- Memory structure after initializing the array of 5 integers.

* Types of Array :-

— Arrays are categorized according to the dimensions used to define it. Here dimensions indicates the number of rows and columns used to set size of array.

— Array has categorized into following types.

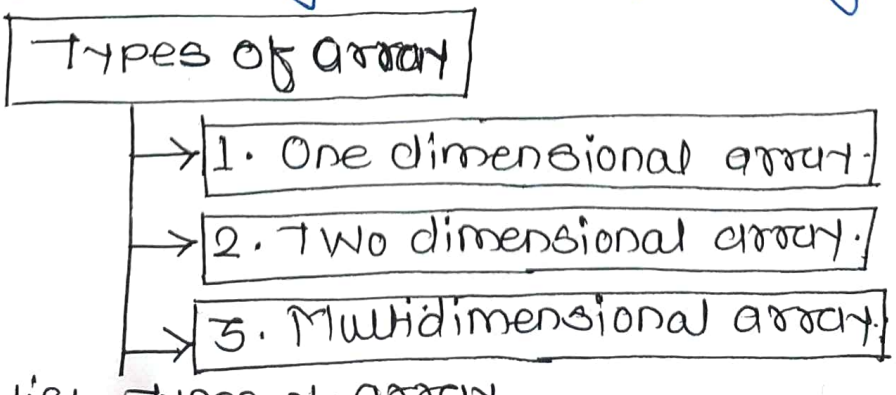


fig:- Types of array.

* One Dimensional Array :-

— An array with single subscript is called as one dimensional array.

Declaration of one dimensional array -

Syntax:-

```
data_type  array_name [size];
```

Example -

- Declare an array to store percentage of 10 students.

```
float percentage [10];
```

Initialization of one dimentional array -

Syntax:-

```
array_name [subscript] = value;
```

Example:-

- Initialize an array to store percentage of 10 students.

```
Percentage [0] = 99.07;
Percentage [1] = 54.04;
Percentage [2] = 60.60;
percentage [3] = 79.47;
Percentage [4] = 92.70;
Percentage [5] = 80.60;
percentage [6] = 85.27;
percentage [7] = 90.30;
percentage [8] = 99.70;
percentage [9] = 94.50;
```

- Using subscript we can assign value to specific array element. The order of assignment statement doesn't matter in above example. It will form following structure in memory.

- Memory arrangement after declaring one dimentional array.

Entering or Writting Data into One Dimentional Array-

- We can enter data into array by array initialization or by accepting array elements from user. We have seen how to initialize data, now we are going to study how ~~array~~ to accept array element from user & store it in array.

```
for (i=0 ; i<10; i++)
 {
   printf("\nEnter percentage of student %d :", i+1);
   scanf( "%f", &percentage[i]);
 }
```

Accessing or Reading Data from an one Dimentional Array -

- While accessing array element we can use loop just like used in writting data into array. Only difference is that reading data from array doesn't require the scanf() function.

```
for(i=0; i<10; i++)
 {
    Printf("\n percewage of student %d: %f ", i+1,
                                    percewage[i]);
 }
```

* Write a program to declare array to store percentage of 10 students. Accept percentage from user and point on the screen.

```
#include <stdio.h>
#include <conio.h>
int main()
 {
  float percewage[10];
  int i;
  for(i=0; i<10; i++)
   {
    Printf("\nEnter percewage of student %d: ", i+1);
    scanf("%f", &percewage[i]);
   }
  for(i=0; i<10; i++)
   {
    printf("\n percewage of student %d: %4.2f", i+1,
                                    percewage[i]);
   }
  return 1;
 }
```

* Write the output of the following program.

```
Void main()
 {
    int sub[10]; i;
    for(i=0 ; i<=8 ; i++)
     {
        sub[i] = i;
        Printf("\n%d", sub[i]);
     }
 }
```

Solution -

It will produce compilation error: "i was not declare". After declaring i following output will be displayed.

```
0
1
2
3
4
5
6
7
8
---------
```

* Write the output of following program.

```
#include<stdio.h>
 Void main()
 {
    int a[5] = {1, 2, 3, 4, 5};
    Printf("%d", a[4]);
 }
```

Solution –

It will produce following output.

5

---

* Find the error in following program and justify it:

```
#include<stdio.h> void main(){
 int i, a[5]={7,5,2,1,9,14};
 for(i=0; i<5; i++)
  Printf("%f", a[2]);
  getch();
}
```

Solution :–

It will produce following error:

– Too many initialization for 'int[5]'

– As the capacity of array is to store 5 elements, in code 6 elements are initialized.

– %f – Wrong modifier.

– getch() will show error as conio.h is not included.

* Find the error in following program & justify it:

```
#include<stdio.h>
 Void main()
  {
    int x;
    int xx[3] = {111, 222, 333, 444};
    for(i=0; j<3; i++)
      Printf("%f", a[i]);
  }
```

Solution:-

It will produce following errors

— Too many initializers for 'int [3]'

As the capacity of array is to store 3 elements, in code 4 elements are initialized.

'i' was not declared in this scope.

'j' was not declared in this scope.

'd' was not declared in this scope.

Every variable should be declared first before it is used.

%f is used for float and not for this.

expected '}' at end of input

Every { should have }.

* Find error in following program and justify it.

```
#include<stdio.h>
void main()
  {
     int I, a[5]={7,5,2,1,9,14};
     Printf("%f", a[2]);
     getch();
  }
```

Solution:-

— Too many initializers for 'int [5]'

— %f is used for floats and not for ints.

— getch() is not declared in the scope.

getch() will show error at conio.h is not included.

* Two Dimensional Array :-

- An array with two subscripts is called as two dimentional array. 2D arrays are mostly used to perform matrix operations.

Declaration of two dimentional array -

Syntax :-

```
data_type array_name [row size] [column size];
```

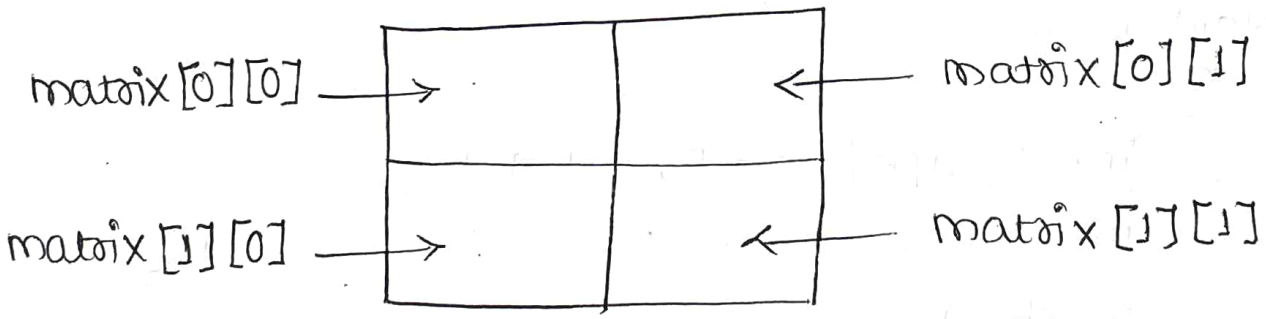
Example :-

- Declare an array to store a 2x2 matrix

```
int matrix [2] [2];
```

The 2x2 matrix store 2*2=4 values. In our example data type of array is integer means the four values are of int type.

- In user's view the matrix looks like;



- But in memory it will form different structure.
- Memory ~~management~~ arrangement after declaring two dimentional array shown in figure on next page.

matrix[0][0]		3000	memory Address
matrix[0][1]		3002	
matrix[1][0]		3004	
matrix[1][1]		3006	

Initialization of two dimentional array

Syntax:-

```
array_name[row-number][column-number]=value;
```

Example:-

- Initialize an array to store values for 2x2 matrix

```
matrix[1][1] = 9;
matrix[0][0] = 7;
matrix[0][1] = 6;
matrix[1][0] = 5;
```

- Using subscript we can assign value to specific array element. The order of assignment statement doesn't matter in above example. It will assign the values given as following fig.

matrix[0][0]	7	6	matrix[0][1]
matrix[1][0]	5	9	matrix[1][1]

- In memory initialized values are stored in sequential row as shown in figure on next page.

matrix[0][0]		3000
matrix[0][1]		3002
matrix[1][0]		3004
matrix[1][1]		3006

memory address

Entering or writting Data into Two Dimentional Array:-

- We can enter data into array by array initialization or by accepting array elements from user.

- We have seen how to initialize data. Now we are going to study how to accept array element from user and store into 2D array.

- The following code is used to accept values for 2x2 matrix from user & store in into 2D Array.

```
printf("\n Enter data for 2D matrix\n");
for (i=0; i<2; i++)
{
    for (j=0; j<2; j++)
    {
        printf ("\n matrix [%d] [%d]:\t", i, j);
        scanf ("%d", &matrix[i][j]);
    }
}
```

- For 2D array, two for loops are used. One for loop (i.e. outer for loop) for row and another for loop (i.e. inner for loop) is for column.

- Printf() function is used to request user to enter the data.

- scanf() function is used to take input from user. &matrix[i][j] tells the compiler to store the entered value as ith row and jth column in matrix. Here value of i and j varies from 0 to 1.

Accessing or Reading Data from an Two Dimensional Array -

- As we use two for loops for writting the data into 2D array, likewise we can use two for loops to read the data from 2D array.

- Only difference is that in reading data from array doesn't require the scanf() function. The following code is used to access elements of two dimentional array:

```
printf("\n The matrix is: \n");
 for (i=0; i<2; i++)
 {
    for (j=0; j<2; j++)
    {
       printf ("%d", matrix[i][j]);
    }
   printf("\n");
 }
```

* Write a program to accept values for 2x2 matrix & print them.

```
#include <stdio.h>
#include <conio.h>
 int main()
 {
   int matrix[2][2], i, j;
   printf("\n Enter data for 2D matrix\n");
```

```
for(i=0; i<2; i++)
  {
    for(j=0; j<2; j++)
      {
        printf("\n matrix[%d][%d]: \t", i,j);
        scanf("%d", &matrix[i][j]);
      }
  }
printf("\n The matrix is: \n");
for(i=0; i<2; i++)
  {
    for(j=0; j<2; j++)
      {
        printf("%d", matrix[i][j]);
      }
    printf("\n");
  }
return 1;
}
```

Output:-

```
Enter data for 2D matrix

matrix [0][0] :   12
matrix [0][1] :   1
matrix [1][0] :   32
matrix [1][1] :   10

The matrix is:
 12   1
 32   10
```

* Multidimensional Array :-

- An array with more than two subscripts is called as multidimensional array.

Declaration of multidimensional array

- For simplicity we will study 3D array which has 3 subscripts.

Syntax -

Syntax for declaring multidimensional array is given below:

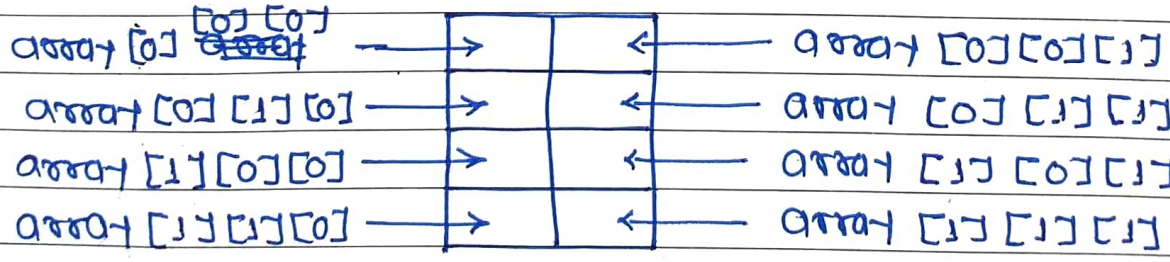
```
data_type array_name [size1] [size2] [size3];
```

Example -

Declare an multidimensional array

```
int array [2] [2] [2]
```

- This array store 2*2*2 = 8 values. In this example data type of array is integer means the value should be of int type.
- In user's view the multidimensional array looks like:



- But in memory it will from different structure.
- Memory arrangement after declaring multidimensional array shown in figure on next page.

array [0] [0] [0]		1000
array [0] [0] [1]		1002
array [0] [1] [0]		1004
array [0] [1] [1]		1006
array [1] [0] [0]		1008
array [1] [0] [1]		1010
array [1] [1] [0]		1012
array [1] [1] [1]		1014

memory Address

Initialization of multidimentional array

Syntax -

```
array-name [row-number] [row_number] [column number]
```

Example -

Initialize the multidimentional array

```
array [0] [1] [1] = 9;
      [0] [0] [0] = 7;
      [0] [0] [1] = 6;
      [0] [1] [0] = 5;
      [1] [0] [0] = 2;
      [1] [0] [1] = 3;
      [1] [1] [0] = 4;
      [1] [1] [1] = 8;
```

array [0] [0] [0]	7	6	array [0] [0] [1]
array [0] [1] [0]	5	9	array [0] [1] [1]
array [1] [0] [0]	2	3	array [1] [0] [1]
array [1] [1] [0]	4	8	array [1] [1] [1]

- In memory initialized values are stored in sequential row by row manner as shown on next page.

array [0] [0] [0]

7	1000
6	1002
5	1004
9	1006
2	1008
4	1010
8	1012
3	1014

memory Address

Entering or writing data into multidimentional array

- The following code is used to accept 10 elements from user & store it into 3D array:

```
printf ("\nEnter data for 3D array \n");
  for(i=0; i<2; i++)
  {
   for (j=0; j<2; j++)
    {
     for (k=0; k<2; k++)
      {
       printf("\narray [%d] [%d] [%d]: \t", i, j, k);
       scanf("%d", &array [i] [j] [k]);
      }
    }
  }
```

* Array of Characters:

- Up till now we have studied array of integer, or float types, now let's study array of characters/ character array.
- An array with character (char) data type is called as character array.
- character arrch forms a string. String is collection of characters, numbers and special symbols.
- Numbers or special symbols can be stored in character array but not recommended.
- At the end of character ~~key~~ array or string, '\0' (NULL value) will be stored.

Declaration of array of Characters

Syntax:-

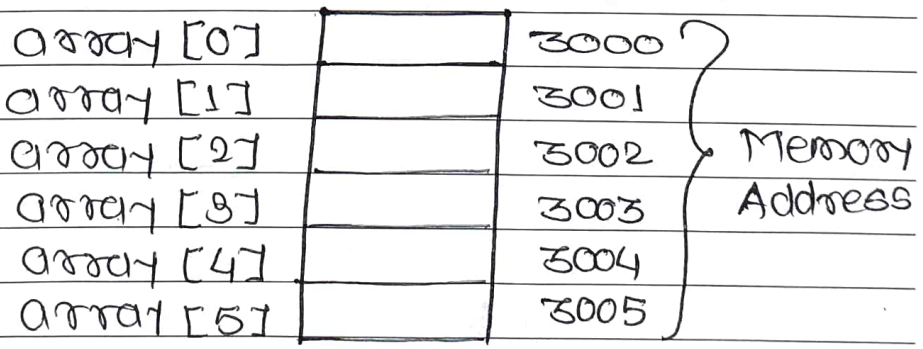
```
char array-name [size];
```

Example -

Declare an array of character to store 6 characters.

```
Char array [6];
```

- Memory arrangement after declaring character array shown in following figure.



Initialization of array of character

- Syntax for initializing character array is given below:

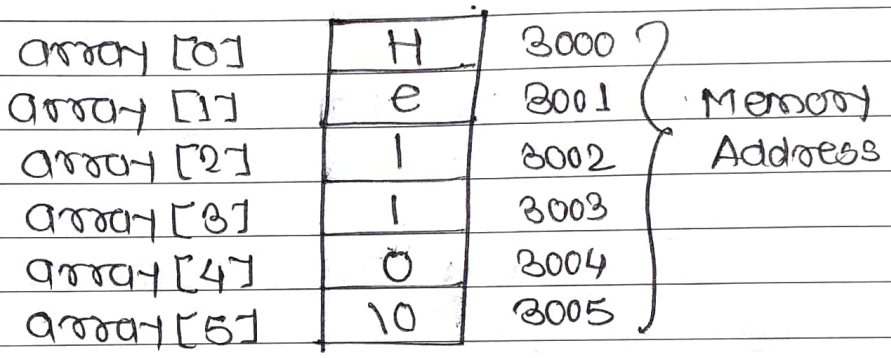
```
array-name [subscript] = value;
```

- For example: Initialize an array to store "Hello" word.

array [0] = 'H'
array [1] = 'e'
array [2] = 'l'
array [3] = 'l'
array [4] = 'o'
array [5] = '\0'

- It will form following structure in memory.

- Memory arrangement after declaring character array shown in figure.



* Character and String Input/Output

- For taking a character array as input %c access-specifier is used with scanf() function.
- The following code is used to accept a character from user and store it in array.

```
for (i=0; i<10; i++)
{
    printf ("\n Enter a character : \t");
    scanf ("%c", &array[i]);
}
```

- And for displaying entered character array as output following code will be used:

```
printf ("\n Entered array is :");
for (i=0; i<10; i++)
```

```
}
  printf ("%c", array[i]);
}
```

* Write a program to declare a character array to accept and store 10 characters and print it.

Soln →

```
#include <stdio.h>
#include <conio.h>
int main()
{
   char array[10];
   int i;
   printf ("\n Enter 10 characters: \t");
   for(i=0; i<10; i++)
   {
      scanf("%c", &array[i]);
   }
   Printf ("\n Entered array is: ");
   for(i=0; i<10; i++)
   {
      printf("%c", array[i]);
   }
  return 1;
}
```

Enter 10 characters: good day..
Entered array is: good day..

* Difference between Character Array and Integer Array

Parameters	Character Array	Integer Array
Data type	Data type of a character array is char.	Data type of a integer array is int.
Access specifier	%c is used to access elements of characters array.	%d is used to access elements of integer array.
Includes	The character array can contain numbers, symbols as well as alphabets. But all values will be treated as characters.	The integer array can contain only integer values.
Declaration	char array [5];	int age [5];
Initialization	char array [5] = {'a', 'b', 'c', '1', '2'};	int age [5] = {12, 22, 23, 42, 25};
Size	Size of an array depends on its data type. A character uses 1 byte to store so the size of a character array is equal to the number of characters it stores. for example: Char array [5]; Need 5 bytes to reserve.	An integer uses 2 byte to store so the size of an integer array is double of the number of integers it stores. for example: int array [5]; Need 5 * 2 = 10 bytes to reserve.

* Write a program to accept a string from user and print it.

```
#include <stdio.h>
#include <conio.h>
int main()
 {
    char str[10];
    printf("\n Enter a string: \t");
    scanf("%s", str);
    printf("\nEntered string is: %s", str);

    return 1;
 }
```

output

```
Enter a string: programming
Entered string is: programming
```

* Differences bet^n Array & String

Parameters	Array	String
Data Type	Array stores elements of same data type.	String is collection of numbers, letters & characters; but every single unit of the string treated as character only.
Space	The space character is not allocated in array. Elements of array should be countinious.	String contains space as a character.
Contain	Array can have multiple types like integer array, character array.	string is made up of characters.

* Operations on Array :-

— Apart from inputing and outputting elements into and ~~array~~ from array, many operations can be performed on array.

Operations on Array

1. Deleting elements in an array
2. Calculating sum and average of array elements
3. Counting positive and negative numbers in the array.
4. Searching an element in an array.
5. Finding the smallest or largest element from an array.
6. Sorting the array elements in ascending or descending order.
7. Addition, substraction, multiplication and division of two ~~numbers~~ arrays.
8. Finding Transpose of matrix.
9. Addition, substraction, multiplication and division of two matrices.
10. Converting the case of characters.
11. Calculate the length of string.
12. Check Weather string is palindrome or not.
13. Compare two string.
14. Copy string into another string.

Deleting elements from array:-

- The deletion of elements of array doesn't affect array size.
- Following program shows the example of deleting the element at specified location in array. And also check whether the given position is less than array size or not.

* Write the program to accept the position from user and delete the element at that position from array.

```
#include <stdio.h>
#include <conio.h>
int main()
{
  int array[20], pos, i, n;
  printf("\n Enter the elements you wants to insert
          into array\n");
  scanf("%d", &n);
  printf("Enter the elements \n", n);
  for(i=0; i<n; i++)
    {
      scanf("%d", &array[i]);
    }
```

```
printf("Enter the position of elements to be
        deleted\n");
scanf("%d", &pos);

if(pos >= n+1)
{
    printf("Deletion not possible.\n");
}
else
{
    for(i=pos-1; i<n-1; i++)
    {
        array[i] = array[i+1];
    }
    printf("After deleting elements at %d
        location the array is\n", pos);
    for(i=0; i<n-1; i++)
    {
        printf("%d\n", array[i]);
    }
}
return 0;
}
```

Output:-

```
Enter number of elements want to insert in
the array
4
Enter 4 elements
1
2
3
4
Enter the position of element to be deleted
3
After deleting element at 3 location the array is
1
2
4
```

2. Calculating sum and average of array elements:-

* Write a program to find sum and average marks obtained by a class of 10 students in a test.

```
#include<stdio.h>
#include<conio.h>
int main()
{
   int avg, sum = 0;
   int i;
   int marks[10];
   for(i=0; i<10; i++)
   {
      printf("\n Enter mark of student %d:", i);
      scanf("%d", &marks[i]);
   }
   for(i=0; i<10; i++)
   {
      sum = sum + mark[i];
   }
   printf("\n sum of marks = %d", sum);

   avg = sum / 10;
   printf("\n Average marks = %d", avg);
}
```

Output:-

```
Enter Mark of student 1: 70
Enter Mark of student 2: 89
Enter Mark of student 3: 82
Enter Mark of student 4: 97
Enter Mark of student 5: 76
Enter Mark of student 6: 66
Enter Mark of student 7: 64
Enter Mark of student 8: 72
Enter Mark of student 9: 88
Enter Mark of student 10: 60
  Sum of Marks    = 764
Average of Marks = 76
```

3. Counting positive and negative numbers in the array

* Write a program to find out number of positive, negative and zero elements from an array.

```
#include <stdio.h>
#include <conio.h>
void main()
{
   int numbers[5];
   int i=0, cnt_pos=0, cnt_neg=0, cnt_zero=0;

for(i=0; i<5; i++)
 {
   printf("\n Enter the Number:");
   scanf("%d", &numbers[i]);
   if (numbers[i]<0)
     {
        cnt_neg++;
     }
   if(numbers[i]==0)
     {
       cnt_zero++;
     }
  if(numbers[i]>0)
     {
        cnt_pos++;
     }

getche();
}

printf("\n Number of +ve numbers=%d", cnt_pos);
printf("\n Number of -ve numbers=%d", cnt_neg);
printf("\n Number of zeros=%d", cnt_zero);
getch();
}
```

4. Write a program to search a particular number in an array. If the number is present print "Number found" else print "Number not found".

Soln ⇨

```
#include<stdio.h>
#include<conio.h>
int main()
{
  int array[6] = {20,80,60,40,10,45};
  int i, num, flag=1;
  printf("array elements are:");
  for (i=0; i<6; i++)
  {
     printf("\t%d", array[i]);
  }
  printf("\n Enter an element to search:\t");
  scanf("%d", &num);

  for (i=0; i<6; i++)
  {
     if(array[i]==num)
     {
        flag=0;
     }
  }

  if(flag==0)
  {
    printf("\n Number Found");
  }
  else {
       printf("\n Number not found");
  }
}
```

Output :-

array elements are: 20 80 60 40 10 45
Enter an element to search: 40
Number Found.

5. Write a program to find smallest Number in 5-element integer array.

Soln ⇒

```
#include <stdio.h>
#include <conio.h>
int main()
{
  int array[5] = {33, 30, 34, 31, 32}, j;
  printf("\n Array elements are:");

  for(j=0; j<5; j++)
    {
      printf("%d\t", array[j]);
    }

  int min = array[0];

  for(j=1; j<5; j++)
    {
      if(array[j] < min)
        {
          min = array[j];
        }
    }

  Printf("\n smallest number in 5-element
          integer array is: %d", min);

  return 1;
}
```

Output:-

Array elements are: 33 30 34 31, 32

smallest number in 5-element integer array is: 30.

6. Write a program to find largest number in 5-elements integer array.

Solution ⇒

```
#include<stdio.h>
#include<conio.h>
int main()
{
   int array[5] = { 38, 30, 34, 31, 32 }, j;
   printf("\n Array elements are: \t");

   for(j=0; j<5; j++) {
      printf("%d\t", array[j]);
   }
  int max = array[0];

  for(j=1; j<5; j++)
   {
      if(array[j] > max)
        {
          max = array[j];
        }
   }
printf("\n largest number in 5-element integer
        array is: \t %d", max);

 return 1;
 }
```

Output:-

Array elements are: 38 30 34 31 32
largest number in 5-element integer array is: 34

☆ Sorting the array elements in ascending order

7. Write a program to sort elements of an array in ascending order. Read elements of array from user [using scanf function].

Soln →

```
#include <stdio.h>
#include <conio.h>
int main()
{
  int array[10], i, j, temp;
  printf("Enter 10 elements for array: ");
  for(i=0; i<10; i++)
  {
    scanf("%d", &array[i]);
  }
 for(i=0; i<9; i++)
  {
    for(j=i+1; j<10; j++)
    {
      if(array[j] < array[i])
      {
        temp= array[i];
        array[i]= array[j];
        array[j]= temp;
      }
    }
  }
printf("\n Ascending ordered array: \n");
  for(i=0; i<10; i++)
  {
    printf(" %d", array[i]);
  }
return 1;
}
```

Output:-

```
Enter 10 elements for array :40
 46
 24
 73
 80
 10
 79
 44
  21
  23
```

Ascending ordered array:

10 21 23 24 40 44 46 73 79 80

8. Write a program to sort elements of an array in descending order.

Soln ⇒

```
#include <stdio.h>
#include <conio.h>
int main ()
 {
   int array [6], i, j, temp;
   printf ("Enter 6 elements for array: ");
   for (i=0; i<6; i++)
     {
       scanf ("%d", &array [i]);
     }
```

```
for( i=0; i<5; i++)
{
  for(j=i+1; j<6; j++)
  {
    if(array[j] > array[i])
    {
      temp= array[i];
      array[i]= array[j];
      array[j]= temp;
    }
  }
}
printf("\n decending ordered array: \n");
  for(i=0; i<6; i++)
  {
    printf("%d \t", array[i]);
  }
  return 1;
}
```

Output:-

Enter 6 elements for array:3
7
4
0
6
8

descending ordered array:
8 7 6 4 3 0

7. Addition, Subtraction, Multiplication, and Division of two arrays

9. Write a program to accept values for 2 integer arrays and add and multiply them and print the result of addition and multiplication.

```
#include <stdio.h>
#include <conio.h>
void main ()
{
   int array1[5], array2[5], array3[5];
   int i=0;
   printf("\n Enter elements of first array: ");
   for (i=0; i<5; i++)
   {
      printf("Enter number for array1[%d]:", i);
      scanf("%d", &array1[i]);
   }
   Printf("\n Enter elements for array second: ");
   for(i=0; i<5; i++)
   {
      printf("Enter number for array2[%d]:", i);
      scanf("%d", &array2[i]);
   }

   Printf("\n Addition of array1 and array2");
   for(i=0; i<5; i++)
   {
      array3[i] = array1[i] + array2[i];
      printf("\n array1[%d] + array2[%d] = array3[%d]
                  =%d", i,i,i, array3[i]);
   }
   printf("\n multiplication of array1 and array2");
   for (i=0; i<5; i++)
   {
      array3[i] = array1[i] * array2[i];
      printf("\n array1[%d] * array2[%d] = array3
                  [%d]=%d", i,i,i, array3[i]);
   }
}getch();
```

Output:-

addition of array1 and array2
array1[0] + array2[0] = array3[0] = 6
Enter elements for first array
Enter element for array1[0] : 1
Enter element for array1[1] : 2
Enter element for array1[2] : 3
Enter element for array1[3] : 4
Enter element for array1[4] : 5
Enter elements for array second
Enter element for array2[0] : 5
Enter element for array2[1] : 4
Enter element for array2[2] : 3
Enter element for array2[3] : 2
Enter element for array2[4] : 1
Enter element for array3[5] :

addition of array1 and array2
array1[0] + array2[0] = array3[0] = 6
array1[1] + array2[1] = array3[1] = 6
array1[2] + array2[2] = array3[2] = 6
array1[3] + array2[3] = array3[3] = 6
array1[4] + array2[4] = array3[4] = 6

Multiplication of array1 and array2
array1[0] * array2[0] = array3[0] = 5
array1[1] * array2[1] = array3[1] = 8
array1[2] * array2[2] = array3[2] = 9
array1[3] * array2[3] = array3[3] = 8
array1[4] * array2[4] = array3[4] = 5

10. Write a program to accept values for 2 integer arrays and substract and divide them and print the result of substraction and division.

```
#include <stdio.h>
#include <conio.h>
```

```
void main()
{
 int array1[5], array2[5], array3[5];
 int i=0;
 printf("\n Enter the elements for first array");
 for (i=0; i<5; i++)
  {
   printf("\n Elements of array");
   for(i=0; i<5; i++)
   {
    printf("\n Enter number of array1[%d]:", i);
    scanf("%d", &array1[i]);
   }
 printf("\n Enter the elements for second array");
 for (i=0; i<5; i++)
  {
    printf("\n Enter number of array2[%d]:", i);
    scanf("%d", &array2[i]);
  }

 printf("\n Substraction of array1 and array2");
  for (i=0; i<5; i++)
   {
     array3[i] = array1[i] - array2[i];
     printf("\n Array1[%d] - array2[%d] =
              array3[%d] = %d", i, i, i, array3[i]);
   }

 printf("\n division of array1 and array2");
  for (i=0; i<5; i++)
   {
     array3[i] = array1[i] / array2[i];
     printf("\n array1[%d] / array2[%d] = array3[%d]
          = %d", i, i, i, array3[i]);
   }

  getch();
 }
```

8. Finding Transpose of matrix

11. Write a program to find transpose of any (3 x 3) matrix.

```
#include<stdio.h>
#include<conio.h>
 void main()
 {
 int origin_matrix[3][3], trans_matrix[3][3], i=0,j=0;

  for(i=0; i<3; i++)
   {
    for(j=0; j<3; j++)
     {
       printf("\n Enter matrix [%d][%d] element", i,j);
       scanf("%d", &origin_matrix[i][j]);

      trans_matrix[j][i] = origin_matrix[i][j];
     }
   }
printf("\n Original Matrix: \n\t");
 for(i=0; i<3; i++)
   {
    for(j=0; j<3; j++)
     {
       printf("%d\t", origin_matrix[i][j]);
     }
    printf("\n\t");
   }
printf("\n Transposed matrix: \n\t");
  for(i=0; i<3; i++)
   {
    for(j=0; j<3; j++)
     {
       printf("%d\t", trans_matrix[i][j]);
     }

    printf("\n\t");
  }
 getch();
}
```

9. Addition, Subetraction, Multiplication and Division of two matrices.

12. Write a program to read two matrices and print their addition.

⇨

```
#include<stdio.h>
#include<conio.h>
int main()
{
  int matrix1[2][2], matrix2[2][2], matrix3[2][2];
  int i=0, j=0;
  printf("\n Enter array element of first matrix:");
  for(i=0; i<2; i++)
  {
    for(j=0; j<2; j++)
    {
      printf("\n Enter matrix[%d][%d] element\t", i, j);
      scanf("%d", &matrix[i][j]);
    }
  }
  printf("\nEnter array elements of second matrix:");
  for(i=0; i<2; i++)
  {
    for(j=0; j<2; j++)
    {
      printf("\nEnter matrix2[%d][%d] element\t", i, j);
      scanf("%d", &matrix2[i][j]);
    }
  }
  printf("\n matrix1 : \n\t\t");
  for(i=0; i<2; i++)
  {
    for(j=0; j<2; j++)
    {
      printf("%d\t", matrix1[i][j]);
    }
    print("\n\t\t");
  }
```

```
printf("\n matrix2: \n\t\t");
for(i=0; i<2; i++)
{
  for(j=0; j<2; j++)
  {
    printf("%d\t", matrix2[i][j]);
  }
  printf("\n\t\t");
}
printf("\n Addition of two 2X2 matrix is: \n\t\t");
for(i=0; i<2; i++)
{
  for(j=0; j<2; j++)
  {
    matrix3[i][j] = matrix1[i][j] + matrix2[i][j];
    printf("%d\t", matrix3[i][j]);
  }
  printf("\n\t\t");
}
}
```

Output:-

```
Enter array elements of first matrix:
Enter matrix 1 [0][0] element 1
Enter matrix 1 [0][1] element 2
Enter matrix 1 [1][0] element 3
Enter matrix1 [1][1] element 4
```

```
Enter array elements of second matrix:
Enter matrix2[0][0] element 4
Enter matrix2[0][1] element 3
Enter matrix2[1][0] element 2
Enter matrix2[1][1] element 1

matrix1:
        1    2
        3    4
matrix2:
        4    3
        2    1
Addition of two 2X2 matrix is:
        5    5
        5    5
```

13. Write a program to read two matrices and print their substraction, division, and multiplication.

Sol. ⇨

```
#include <stdio.h>
#include <conio.h>
int main()
{
int matrix1[2][2], matrix2[2][2], matrix3[2][2];
int i=0,j=0;
printf("\n Enter array elements of first matrix:");
for(i=0;i<2;i++)
{
 for(j=0;j<2;j++)
 {
   scanf("%d",&matrix1[i][j]);
 }
}
printf("\n Enter array elements of second matrix:");
 for(i=0;i<2;i++)
 {
```

```
for(j=0;j<2;j++)
{
 scanf("%d",&matrix2[i][j]);
}
}
printf("\n matrix1: \n\t\t");
for(i=0;i<2;i++)
{
 for(j=0;j<2;j++)
 {
  printf("%d\t", matrix1[i][j]);
 }
 printf("\n\t\t");
}
printf("\n matrix2: \n\t\t");
for(i=0;i<2;i++)
{
 for(j=0;j<2;j++)
 {
  printf("%d\t", matrix2[i][j]);
 }
 printf("\n\t\t");
}

printf("\n substraction of two 2x2 matrix is: \n\t\t");
for(i=0;i<2;i++)
{
 for(j=0;j<2;j++)
 {
  matrix3[i][j]=matrix1[i][j]-matrix2[i][j];
  printf("%d\t", matrix3[i][j]);
 }
 printf("\n\t\t");
}
printf("\n division of two 2x2 matrix is: \n\t\t");
for(i=0;i<2;i++){
 for(j=0;j<2;j++){
  matrix3[i][j]= matrix1[i][j]/matrix2[i][j];
  printf("%d\t", matrix3[i][j]);
 }
```

```
    printf("\n\t\t");
}

printf("\n multiplication of two 2X2 matrix is: \n\t\t");
 for(i=0;i<2;i++)
  {
   for(j=0;j<2;j++)
    {
    matrix3[i][j]= matrix1[i][j] * matrix[i][j];
    printf("%d\t", matrix3[i][j]);
   }
   printf("\n\t\t");
  }
}
```

Converting the case of characters

15. Write a program to accept lower case characters and convert them to upper case characters.

801 ⇨

```
#include<stdio.h>
#include<conio.h>
void main()
{
  char array[5];
  int i=0;
  for(i=0; i<5; i++)
  {
    printf("\nEnter character in lowercase:");
    array[i] = getche();

    array[i] = array[i]-32;

  }
for(i=0; i<5; i++)
  {
    printf("\nThe converted character is %c",
                          array[i]);
  }
  getch();
}
```

```
Enter character in lowercase : h
Enter character in lowercase : e
Enter character in lowercase : l
Enter character in lowercase : l
Enter character in lowercase : o

The converted character is H
The converted character is E
The converted character is L
The converted character is L
The converted character is O
```

11. Calculate length of string

16. Write a program to accept a string and display the length of it without using standard library function.

```
#include <stdio.h>
#include <conio.h>
Void main()
{
  int i=0;
  char str[10];
  printf("\n Enter String:");
  scanf("%s", str);

  While (str[i] != '\0')
  {
     i++;
  }
printf("\n Length of string %s = %d", str, i);
getch();
}
```

```
Enter String : programming
Length of string programming = 11
```

BATU-EXAM

Made by batuexams.com

at MET Bhujbal Knowledege City