UNIT-2(ARRAY)

An array is defined as the collection of similar type of data items stored at contiguous memory locations.

Properties of Array

The array contains the following properties.

- Each element of an array is of same data type and carries the same size, i.e., int = 4 bytes.
- Elements of the array are stored at contiguous memory locations where the first element is stored at the smallest memory location.
- The array is the simplest data structure where each data element can be randomly accessed by using its index number.we can calculate the address of each element of the array with the given base address and the size of the data element.
- Arrays are the derived data type in C programming language which can store the primitive type of data such as int, char, double, float, etc.
- It also has the capability to store the collection of derived data types, such as pointers, structure, etc.

Advantage of C Array

1) Code Optimization: Less code to the access the data.

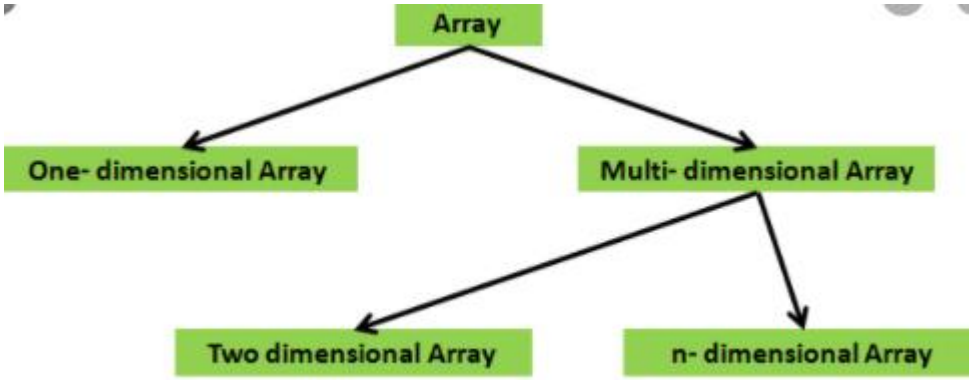
2) Ease of traversing: By using the for loop, we can retrieve the elements of an array easily.

3) Ease of sorting: To sort the elements of the array, we need a few lines of code only.

4) Random Access: We can access any element randomly using the array.

Disadvantage of C Array

1) **Fixed Size**: Whatever size, we define at the time of declaration of the array, we can't exceed the limit.



One Dimensional array:→ **single dimensional arrays are used to store list of values of same datatype. In other words, single dimensional arrays are used to store a row/column of values. In single dimensional array, data is stored in linear form. Single dimensional arrays are also called as** one-dimensional arrays**,** Linear Arrays **or simply** 1-D Arrays**.**

Multi Dimensional Array:→An array of arrays is called as multi dimensional array. In simple words, an array created with more than one dimension (size) is called as multi dimensional array. Multi dimensional array can be of **two dimensional array** or **three dimensional array** or **four dimensional array** or more.

- **The simplest form of multidimensional array is the two-dimensional array.**. The 2-D arrays are used to store data in the form of table. We also use 2-D arrays to create mathematical **matrices**.
- In C, when an array is initialized with size, then it assigns defaults values to its elements in following order.

Data Type	Default Value
bool	false
char	0
int	0
float	0.0
double	0.0f

void	
wchar_t	0

Declaration of 1D Array

We can declare an array in the c language in the following way.

1. data_type array_name[array_size];

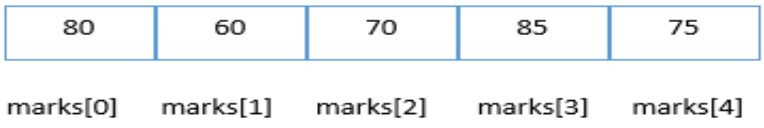
   Example:→ int marks[5];

   Here, int is the *data_type*, marks are the *array_name*, and 5 is the *array_size*.

Initialization of C Array

The simplest way to initialize an array is by using the index of each element. We can initialize each element of the array by using the index. Consider the following example.

1. marks[0]=80;//initialization of array
2. marks[1]=60;
3. marks[2]=70;
4. marks[3]=85;
5. marks[4]=75;



Initialization of Array

C array example

1. #include<stdio.h>
2. **int** main(){
3. **int** i=0;
4. **int** marks[5];//declaration of array
5. marks[0]=80;//initialization of array
6. marks[1]=60;

```
7.  marks[2]=70;
8.  marks[3]=85;
9.  marks[4]=75;
10. //traversal of array
11. for(i=0;i<5;i++){
12. printf("%d \n",marks[i]);
13. }//end of for loop
14. return 0;
15. }
```

C Array: Declaration with Initialization

We can initialize the c array at the time of declaration. Let's see the code.

```
1.  int marks[5]={20,30,40,50,60};
```

In such case, there is **no requirement to define the size**. So it may also be written as the following code.

```
1.  int marks[]={20,30,40,50,60};
```

```
1.  #include<stdio.h>
2.  int main(){
3.  int i=0;
4.  int marks[5]={20,30,40,50,60};//declaration and initialization of array
5.   //traversal of array
6.  for(i=0;i<5;i++){
7.  printf("%d \n",marks[i]);
8.  }
9.  return 0;
10. }
```

Two Dimensional Array in C

The two-dimensional array can be defined as an array of arrays. The 2D array is organized as matrices which can be represented as the collection of rows and columns. However, 2D arrays are created to implement a relational database lookalike data structure.

Declaration of two dimensional Array in C

The syntax to declare the 2D array is given below.

```
1. data_type array_name[rows][columns];
        int twodimen[4][3];
```

Initialization of 2D Array in C

```
1. int arr[4][3]={{1,2,3},{2,3,4},{3,4,5},{4,5,6}};
```

Two-dimensional array example in C

```
1. #include<stdio.h>
2. int main(){
3. int i=0,j=0;
4. int arr[4][3]={{1,2,3},{2,3,4},{3,4,5},{4,5,6}};
5. //traversing 2D array
6. for(i=0;i<4;i++){
7.  for(j=0;j<3;j++){
8.    printf("arr[%d] [%d] = %d \n",i,j,arr[i][j]);
9.  }//end of j
10.}//end of i
11.return 0;
12.}
```

Operation on Arrays in C:➔ There are a number of operations that can be performed on an array which are:

1. Traversal
2. Copying
3. Reversing
4. Sorting
5. Insertion
6. Deletion
7. Searching
8. Merging
9. **Traversal:**

1**)Traversal means.** Traversing means to access all the elements of the array starting from the first element to the last element one by one.

Aaccessing each array element for a specific purpose, either to perform an operation on them , counting the total number of elements or else using those values to calculate some other result.

LB: Lower Bound

UP: Upper Bound

Let A[] be the array.

- **Algorithm:** Procedure Traverse(A,LB,UB)

```
Step 1: Initialise counter i = lower_bound_index
Step 2: Repeat steps 3 to 4 while i < upper_bound
Step 3: Apply the specified operation on A[i]
Step 4: Increment counter : i = i + 1
[Loop Ends]
Step 5: Exit
```

```c
#include<stdio.h>
int main(){
int i=0;
int marks[5]={20,30,40,50,60};//declaration and initialization of array
 //traversal of array
for(i=0;i<5;i++){
printf("%d \n",marks[i]);
}
return 0;
}
```

Insertion Operation

Insert operation is to insert one or more data elements into an array. Based on the requirement,

- a new element can be added at the beginning, end, or
- any given index of array.
- Case1:(If n elements are present in an array & array size is more than n, then new element can added at the next position.)

Algorithm:

1. Declare the array as A[num]
2. Read N // N: number of elements already present.
3. Repeat for i=0 to(i<n)
   read a[i]
   i++
4. Read ELEMENT //the element have to add
5. A[N]=ELEMENT
6. N++
7. Display the element after insertion.
8. EXIT

Program:

```
#include <stdio.h>
#include <conio.h>
void main()
{ int a[10];
  int i,n,value;
  clrscr();
  printf("Enter the number of elements:\n");
  scanf("%d",&n);
  printf("Enter the elements:\n");
  for(i=0;i<n;i++)
  {scanf("%d",&a[i]);
  }
  printf("Enter the value of the new element:\n");
  scanf("%d",&value);
  a[n]=value;
  printf("The new array with the newly added elements is:\n");
  for(i=0;i<=n;i++)
  {printf("%d\n",a[i]);
  }
  getch();
}
```

Case2: Insert at a particular position

Algorithm:Procedure INSERT(A, pos, len, num)

1. Repeat for i=(len-1) down to (pos-1)
   . A[i+1]=A[i]
   end repeat
2. A[pos-1]=num
3. Len=len+1 //length of array resetted
4. For i=0 to i<len
   Print A[i]
   Display the new array after insertion
5. EXIT

Program:

```
#include <stdio.h>
#include <conio.h>
void main()
{ int a[100];
  int i,n,position,value;
  clrscr();
  printf("Enter the number of elements:\n");
  scanf("%d",&n);
  printf("Enter the elements:\n");
  for(i=0;i<n;i++)
  {scanf("%d",&a[i]);
  }
  printf("Enter the position where you want to enter the value:\n");
  scanf("%d",&position);
  printf("Enter the value you want to enter:\n");
  scanf("%d",&value);
  for(i=n-1;i>=position-1;i--)
  {a[i+1]=a[i];
  }
  a[position-1]=value;
  printf("The new array after insertion is:\n");
  for(i=0;i<=n;i++)
  {printf("%d\n",a[i]);
  }
  getch();
}
```

Deletion:

Deletion of element from an array can be done in 3 ways.

1. Either at the end

2. From a required position

Case1: Deletion of last element.

Procedure DELETE(A, len)

1. Item ← A[(len-1)]
2. len= len-1//length resetted
3. Display all the elements left after deletion
4. END

Program:

```
#include <stdio.h>
#include <conio.h>
void main()
{ int a[100];
  int i,n;
  clrscr();
  printf("Enter the number of elements:\n");
  scanf("%d",&n);
  printf("Enter the elements:\n");
  for(i=0;i<n;i++)
  { scanf("%d",&a[i]);
  }
  printf("The value removed from the array:%d\n",a[n-1]);
n--;
  printf("The remaining elements are:\n");
  for(i=0;i<n;i++)
  {printf("%d\n",a[i]);
  }
  getch();
}
```

Case 2: Deletion of element from a specific position.

Algorithm:➔Procedure Delete(A, pos, len)

1. Item ←a[pos]..element taken out
2. Repeat for i=(pos-1) to (len-1)
   A[i] = A[i+1]
   end repeat
3. len ← len-1 //length resetted
4. Display elements of array

5. END

Program:

```
#include <stdio.h>
#include <conio.h>
void main()
{ int a[100];
  int i,n,position;
  clrscr();
  printf("Enter the number of elements:\n");
  scanf("%d",&n);
  printf("Enter the elements:\n");
  for(i=0;i<n;i++)
  { scanf("%d",&a[i]);
  }
  printf("Enter the position where you to remove:\n");
  scanf("%d",&position);
  for(i=position-1;i<n;i++)
  { a[i]=a[i+1];
  }n--;
  printf("The new array after deletion of element:\n");
  for(i=0;i<n;i++)
  { printf("%d\n",a[i]);
  }
  getch();
}
```

3)Procedure to delete a specific element from array:➔

Procedure Delete(A,len,num)

1. **Read NUM**
2. **Repeat For i🡐0 to <len**
   If A(i)==num then
   Item🡐A(i)
   Pos🡐i
   exit loop
   end if

end repeat

3. **if pos<len then**
   Repeat For i←(pos) to (len-1)
   A[i]←A[i+1]
   end repeat
   Len←(Len-1)
   end if
4. **Display new list of elements**
   Repeat for i←0 to i<len
   printA[i]
   end repeat
5. **END**

Program:→

```
#include <stdio.h>
#include <conio.h>
void main()
{ int a[100];
  int i,n,pos,element,flag=0;;
  clrscr();
  printf("Enter the number of elements:\n");
  scanf("%d",&n);
  printf("Enter the elements:\n");
  for(i=0;i<n;i++)
  { scanf("%d",&a[i]);
  }
  printf("Enter the element you to remove:\n");
  scanf("%d",&element);
for(i=0;i<n;i++)
        {       if(a[i]==element)
                  {  pos=i;
                     flag=1;
                     break;
                  }
        }
If(flag==1)
{
        for(i=pos;i<n;i++)
```

```
        { a[i]=a[i+1];
        }
      n--;//resetted the lenght
      printf("The new array after deletion of element:\n");
       for(i=0;i<n;i++)
      { printf("%d\n",a[i]);
       }
}
else {printf("the element you want to delete  is not valid");
      }
 getch();
}
```

Search Operation➔ You can perform a search for an array element based on its value or its index.

```
Algorithm➔Search(A,n,ITEM)
   1. Start
   2. Repeat for i=0 to i < n
         IF A[i]== ITEM THEN GOTO STEP 3

   3. PRINT i, ITEM
   4. Stop
```

```
5. #include <stdio.h>
6.
7. void main() {
8.    int A[] = {1,3,5,7,8};
9.    int item = 5, n = 5;
10.    int i = 0, i = 0;
11.
12.    printf("The original array elements are :\n");
13.
14.    for(i = 0; i<n; i++) {
15.       printf("A[%d] = %d \n", i, A[i]);
16.    }
17.
18.    For(i=0;i<n;i++){
19.       if( A[i] == item ) {
20.          break;
21.       }
22.
23.         }
24.
25.    printf("Found element %d at position %d\n", item, i+1);
26. }
```

Program for addition of two matrix:→

Algorithm:→

1. Input matrix 1 and matrix 2.
2. If the number of rows and number of columns of matrix 1 and matrix 2 is equal,
3. for i=1 to rows[matrix 1]
4. for j=1 to columns [matrix 1]
5. Input matrix 1 [i,j]
6. Input matrix 2 [i,j]
7. matrix 3 [i,j]= matrix 1 [i,j]+ matrix 2 [i,j];
8. Display matrix 3 [i,j];

Program:→

```
//Program to Add Two Matrices In C.

#include <stdio.h>
int main()
{
int mat1[2][2], mat2[2][2], sum[2][2], i, j;
for(i=0;i<2;i++)
{
for(j=0;j<2;j++)
{
scanf("%d",&mat1[i][j]);
scanf("%d",&mat2[i][j]);
sum[i][j]=mat1[i][j]+mat2[i][j];
}
}
for(i=0;i<2;i++)
{
for(j=0;j<2;j++)
{printf("%d\t",sum[i][j]);
}
printf("\n");
}
return 0;
}
```

Matrix Multiplication:→

1. Start
2. Declare variables and initialize necessary variables
3. Enter the element of matrices by row wise using loops
4. Check the number of rows and column of first and second matrices

5. If number of rows of first matrix is equal to the number of columns of second matrix, go to step 6. Otherwise, print matrix multiplication is not possible and go to step 3.
6. Multiply the matrices using nested loops.
7. Print the product in matrix form as console output.
8. Stop

Program:➔

```
#include<stdio.h>
#include<stdlib.h>
int main(){
int a[10][10],b[10][10],mul[10][10],r,c,i,j,k;
system("cls");
printf("enter the number of row=");
scanf("%d",&r);
printf("enter the number of column=");
scanf("%d",&c);
printf("enter the  matrix element=\n");
for(i=0;i<r;i++)
{
for(j=0;j<c;j++)    {
scanf("%d",&a[i][j]);
scanf("%d",&b[i][j]);    }    }
printf("multiply of the matrix=\n");
for(i=0;i<r;i++)
{
for(j=0;j<c;j++)
{
mul[i][j]=0;
for(k=0;k<c;k++)
{
mul[i][j]+=a[i][k]*b[k][j];
}    } }
//for printing result
for(i=0;i<r;i++)
{    for(j=0;j<c;j++)
```

```
29.{    printf("%d\t",mul[i][j]);    }
30.printf("\n");
31.}    return 0;  }
```