Module 4 :-

Arrays (part 1)

* Introduction
* Declaratn & initializatn of 1-D array
* Reading & printing of 1-D array
* Bubble sort
* Linear & Binary search
* Reading & printing of 2-D array
* programs on matrix operations: Addition, subtraction, multiplication, transpose

Introduction :-

```
int marks = 20;
```

In the above declaration, variables marks stores only one student marks.

Suppose if we want to store marks of 100 students, then we cant declare 100 variables. Instead of that we can use <u>array.</u>

we use arrays to store set of data items of same type.

```
int marks[90];
```

In this, 90 fields will be allocated to store marks.

* Array :- Array is an <u>ordered set</u> of similar data items.

eg:- int marks[5];

Here, 5 fields will be allocated.

0	1	2	3	4
50	70	80	100	90
marks[0]	marks[1]	marks[2]	marks[3]	marks[4]

marks[0] = 50 → marks of 1st student
marks[4] = 90 → marks of 5th student

i.e., value can be accessed by using array index.

array index starts with 0 (zero).

In this example marks of 5 students are accessed by specifying marks[0] till marks[n-1] = marks[4].

Types of arrays

1) single dimensional array (1-D array)
2) Multi dimensional array (2-D, 3-D)

1) Single-dimensional array (1-D array)

It is a linear list consists of similar data items.

In memory data items are stored contigues one after the other.

Array size depends on the data type we store.

```
int a[5];
```

In this, array is 10 bytes.

i.e., Array size = n * sizeof(datatype)
= 5 * sizeof(int);
= 5 * 2
= 10 bytes

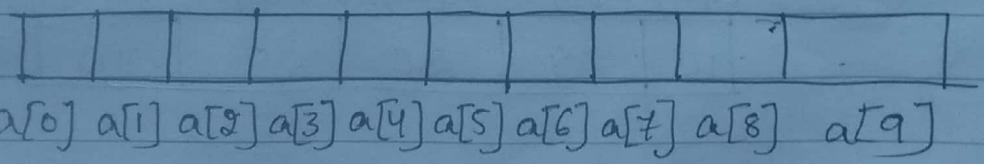
** Declaration of 1-D array

Syntax :- data_type array_name[size];

Eg:- int a[10]; (size: positive integer)

10 spaces will be reserved for a locations.

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



** Initialization of 1-D array

* Assigning values to an array.

Syntax :- data_type array_name[size] = {v1, ---, vn};

(v1, ---, vn: variable values.)

* data types can be int, float, char etc.,
* array_name is name of array
* size or expression should be always gives positive integer.

Different ways of initializing arrays

* * Compile time initialization :-

1) Initializing all specified memory location

Arrays can be initialized at the time of declaration when their initial values are known in advance.

eg:- int a[5] = { 10, 15, 20, 25, 30 };

During compilation, 5 contiguos memory locations are reserved by the compiler for variable a & all these locations are initialized as shown below :-

a[0]	a[1]	a[2]	a[3]	a[4]
10	15	20	25	30

For characters,

int b[5] = { 'H', 'E', 'L', 'L', 'O' };

b[0]	b[1]	b[2]	b[3]	b[4]
H	E	L	L	O

2) partial array initialization

Here, number of values initialized is less than the size of array.

Elements are initialized in the order from 0th location. The remaining locations will be initialized to zero automatically.

Eg:- int a[5] = {10, 20};

5 memory locations will be allocated & compiler initializes first 2 location with 10 & 20. The remaining memory locations are automatically initialized to 0's.

a[0]	a[1]	a[2]	a[3]	a[4]
10	20	0	0	0

3) Initialization without size

Consider,

char b[] = {'H', 'E', 'L', 'L', 'O'};

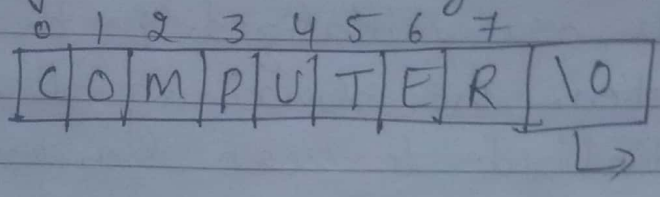
In this example, the array size is not specified. But array size will be set to total number of initial values specified. i.e., array size is automatically set to 5.

b[0]	b[1]	b[2]	b[3]	b[4]
H	E	L	L	O

4) Array initialization with a string

consider, char b[] = "COMPUTER";

Array b is initialized as below,



String "COMPUTER" contains 8 characters, but string always ends with null characters. So array size is 9 bytes (string length + 1).

** Run time initialization (Reading array)

* Reading & printing one-D array

consider, int a[5];

* 5 memory locations are allocated.
* Each elements can be accessed by specifying index.
* using a[0] through a[4], we can access 5 data items.
* The array can be read by using scanf() as follows:

```
scanf("%d", &a[0]);
:
:
scanf("%d", &a[n-1]);
```

In general, scanf("%d", &a[i]) where i = 0, 1, ---, n-1.
We can read n data items from keyboard as follows.

```
for(i=0; i<n; i++)
{
    scanf("%d", &a[i]);
}
```

Similarly, the n data items can be displayed using printf() as follows:

```
for(i=0; i<n; i++)
{
    printf("%d", a[i]);
}
```

WAP to read n elements & display the same using array.

```
void main()
{
    int n, a[10], i;
    printf("Enter no. of elements \n");
    scanf("%d", &n);
    printf("Enter n elements \n");
    for(i=0; i<n; i++)
    scanf("%d", &a[i]);
```

```
printf("The N elements are \n");
for(i=0; i<n; i++)
printf("%d\t", a[i]);
}
```

WAP to read n elements, find sum & display result using array.

```
void main()
{
  int n, i, a[10], sum=0;
  printf("Enter number of elements \n");
  scanf("%d", &n);
  printf("Enter N elements \n");
  for(i=0; i<n; i++)
  scanf("%d", &a[i]);
  for(i=0; i<n; i++)
  sum=sum+a[i];
  printf("sum=%d", sum);
  getch();
}
```

WAP to find sum of positive numbers, negative number using array.

```
void main()
{
  int n, i, a[10], psum=0, nsum=0;
  clrscr();
```

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

for(i=0; i<n; i++)
{
  if(a[i]>=0)
    psum = psum+a[i];
  else
    nsum = nsum+a[i];
}
printf("Sum of +ve nos = %d\n", psum);
printf("Sum of -ve nos = %d\n", nsum);
getch();
}
```

WAP to input n integer nos & conduct linear search for a given array elements.

```
void main()
{
  int n, i, a[100], key, found=0;
  clrscr();
  printf("Enter number of elements\n");
  scanf("%d", &n);
```

```
printf("Enter elements\n");
for(i=0;i<n;i++)
scanf("%d",&a[i]);
printf("Enter key element to search\n");
scanf("%d",&key);
for(i=0;i<n;i++)
{
  if(a[i]==key)
    {
      found=1;
      break;
    }
}
  if(found==1)
  printf("key found at position=%d",i+1);
  else
  printf("not found");
  getch();
}
```

* Lab program 4(Bubble sort)
* Lab program 5(Binary search)

WAP to find largest of N elements using array

```
void main()
{
  int a[10], n, i, big;
  printf("Enter n\n");
  scanf("%d", &n);
  printf("Enter elements\n");
  for(i=0; i<n; i++)
  scanf("%d", &a[i]); → big = a[0];
  for(i=1; i<n; i++)
  {
      if (a[i] > big)
      {
        big = a[i];
      }
  }
  printf("Largest = %d\n", big);
}
```

Two-dimensional array (2D arrays)

It is an array where data items will be stored in the form of rows & columns in a table fashion.

*Declaration :-

syntax :- data_type array_name [exp1][exp2];

where data_type can be int, float, char etc,
* array_name is the name of array.
* exp1 & exp2 are constant expression.

* exp1 specifies rows to be accessed.
* exp2 specifies columns to be accessed.

Eg:- int a[3][4];

In this, array has 3 rows & 4 columns. (3*4=12) memory locations are allocated.

Pictorial representation of array

	col-0	col-1	col-2	col-3
Row 0	a[0][0]	a[0][1]	a[0][2]	a[0][3]
Row 1	a[1][0]	a[1][1]	a[1][2]	a[1][3]
Row 2	a[2][0]	a[2][1]	a[2][2]	a[2][3]

Initializatn of 2-D array

Syntax:-

```
data-type array-name[exp1][exp2] = {
                                    {a1, ---, an},
                                    ..........
                                    ..........
                                    {z1, ---, zn}
                                  };
```

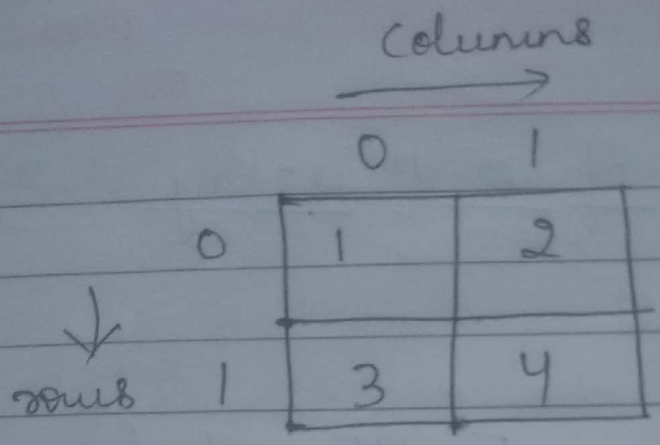
Here, a1 to ann are the values assigned to 1st row & so on.

1) Initializing all specified memory locations

consider, int a[2][2] = {
                          {1, 2},
                          {3, 4}
                        };

Array a has 2 rows & 2 columns.

pictorial representation of this 2-D array is shown below:-

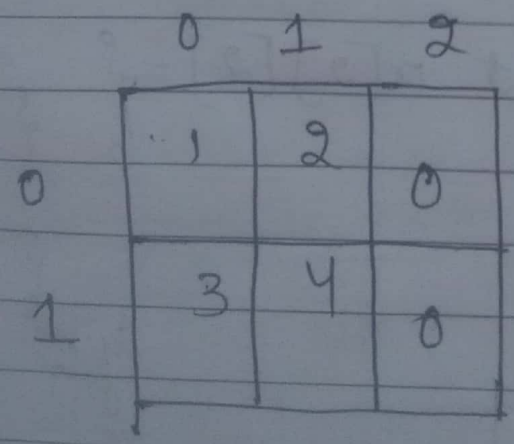


2) partial array initialization

Here, number of values initialized are less than the size of array.

The remaining locations are initialized to zero automatically.

```
int a[2][3] = {
                {1, 2},
                {3, 4}
              };
```



Reading & printing of 2-D array

To access each item row wise in 2D-array, the row index i should be in outer loop & column index j should be in inner loop.

To read 2-D array of size m×n :-

```
for(i=0; i<m; i++)  /* m - rows */
{
    for (j=0; j<n; j++)  /* n - columns */
    {
        scanf ("%d", &a[i][j]);
    }                    ↳ /* read m×n matrix */
}
```

To print 2-D array of size m×n :-

```
for (i=0; i<m; i++)
{   for( j=0; j<n; j++)
    {  printf("%d", a[i][j]);
    }  printf("\n");
}
```

* WAP to read mxn matrix & display the same.

```
void main()
{
    int m, n, a[10][10], i, j;
    printf("Enter size of matrix\n");
    scanf("%d%d", &m, &n);
    printf("Enter elements of matrix A\n");
    for(i=0; i<m; i++)
    {
        for(j=0; j<n; j++)
        {
            scanf("%d", &a[i][j]);
        }
    }
    printf("Elements of matrix A\n");
    for(i=0; i<m; i++)
    {
        for(j=0; j<n; j++)
        {
            printf("%d\t", a[i][j]);
        }
        printf("\n");
    }
}
```

* WAP to add 2 matrices & store result in 3rd matrix.

```
void main()
{
    int m,n,i,j,a[10][10],b[10][10],
    c[10][10];
    printf("Enter size of matrix\n");
    scanf("%d%d",&m,&n);
    printf("Enter elements of matrix A\n");
    for(i=0;i<m;i++)
    {
        for(j=0;j<n;j++)
        {
            scanf("%d",&a[i][j]);
        }
    }
    for(i=0;i<m;i++)
    {
        for(j=0;j<n;j++)
        {
            scanf("%d",&b[i][j]);
        }
    }
    for(i=0;i<m;i++)
    {
        for(j=0;j<n;j++)
        {
            c[i][j]=a[i][j]+b[i][j];
        }
    }
```

```
printf("Resultant matrix\n");
for(i=0;i<m;i++)
{
for(j=0;j<n;j++)
  {
    printf("%3d", c[i][j]);
  }
  printf("\n");
}
getch();
}
```

*** WAP to find transpose of matrix.

```
void main()
{
int i, j, m, n, a[10][10], t[10][10];
printf("Enter size of matrix\n");
scanf("%d%d", &m, &n);
printf("Enter elements\n");
for(i=0; i<m; i++)
{
  for(j=0; j<n; j++)
  {
    scanf("%d", &a[i][j]);
  }
}
```

```
for(i=0; i<m; i++)
{
 for(j=0; j<n; j++)
 {
    t[i][j] = a[i][j];
 }
}

printf("Transpose matrix\n");
for(i=0; i<n; i++)
{
 for(j=0; j<m; j++)
 {
    printf("%d\t", t[j][i]);
 }
 printf("\n");
}
getch();
}
```

Note:- You can find transpose of matrix without using t (temporary matrix). P.T.O

* Sum of elements of matrix pgm
* Lab program 6 (matrix multiplication)
* Trace of matrix.

WAP to find sum of elements of matrix.

```
#include<stdio.h>
#include<conio.h>
void main()
{
    int a[10][10], i, j, m, n, sum=0;
    clrscr();
    printf("Enter row & column size\n");
    scanf("%d%d", &m, &n);
    printf("Enter matrix elements\n");
    for(i=0; i<m; i++)
    {
      for(j=0; j<n; j++)
      {
        scanf("%d", &a[i][j]);
      }
    }
    for(i=0; i<m; i++)
    {
      for(j=0; j<n; j++)
```

```
    {
        sum = sum + a[i][j];
    }
}
printf("sum = %d", sum);
getch();
}
```