COMPUTER PROGRAMMING - I

Need of an Array

- main( )
- { int x ;
- x = 5 ;
- x = 10 ;
- printf ("\nx = %d", x) ;
- }
-
- Value of x is overwritten

Need of an Array

- Suppose we wish to arrange the percentage marks obtained by 100 students in ascending order.
- (a) Construct 100 variables to store percentage marks obtained by 100 different students, i.e. each variable containing one student's marks.
- (b) Construct one variable (called array or subscripted variable) capable of storing or holding all the hundred values.

Array

- An array is a collection of similar elements/data types.
- These similar elements could be all **int**s, or all **float**s, or all **char**s, etc.
- Usually, the array of characters is called a 'string', whereas an array of **int**s or **float**s is called simply an array.
- Any given array must be of the same type. i.e. we cannot have an array of 10 numbers, of which 5 are **int**s and 5 are **float**s.

Array

- The elements of the array are stored in consecutive memory locations and are referenced by an index or subscript.
- Arrays are declared using the following syntax :
- type name[size]
- For example: int marks[5]

marks[0]	10
marks[1]	12
marks[2]	13
marks[3]	14
marks[4]	15

Array

- Important points about array
- 1. The starting index i.e index of the first element of an array is always zero
- 2. The index of last element is n-1 where n is the size of the array
- 3. An array has static memory allocation i.e memory size once allocated for an array cannot be changed.

Array Initialization

- int n[5] = { 1, 2, 3, 4, 5 };
- If not enough initializers, rightmost elements become 0

- int n[5] = { 0 }
- All elements 0

- If size omitted, initializers determine it
- int n[ ] = { 1, 2, 3, 4, 5 };
- 5 initializers, therefore 5 element array

Array

- WAP to accept n integers from user into an array and display them one on each line

```
int main()
{
   int n, i, a[100];
   printf("Enter number of elements:");
   scanf("%d",&n);
   for(i = 0;i<n; i++)
   {
      printf("Enter a value");
      scanf("%d",&a[i]);
   }
```

Array

- printf("The numbers entered are \n");
-     for(i=0;i<n;i++)
-    {
-                     printf("%d\n",a[i]);
-    }
- }

Array

- WAP to accept marks and display average
- int main( )
- {
-   int avg, sum = 0 ;
- int i ; int marks[5] ; /* array declaration */
- for (i = 0 ; i <5 ; i++)
-    {
-       printf ("\nEnter marks ") ;
-       scanf ("%d", &marks[i]) ; /* store data in array */
-    }

Array

```
/* read data from an array*/
for ( i = 0 ; i <5 ; i++ )
{
    sum = sum + marks[i] ;
}
avg = sum / 5 ;
printf ( "\nAverage marks = %d", avg ) ;
}
```

```
Enter marks 20

Enter marks 30

Enter marks 40

Enter marks 50

Enter marks 60

Average marks = 54
```

Array

- //WAP to accept n integers from user into an array and display the count of even and odd numbers of these

```
int main()
{
    int n, i,a[100], even =0;
    printf("Enter number of elements:");
    scanf("%d",&n);

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

```

Array

```
     for(i=0;i<n;i++)
    {

             if(a[i]%2 == 0)
                even++;
    }
    printf("The count of even numbers is %d and count of odd numbers is %d",even,(n-even));
}
```

Array

- // WAP to evaluate the value of the following series and display the result

```
int main()
{
    int n, i,a[100],sum1 =0,sum2 =0,sum =0;
    printf("Enter number of elements:");
    scanf("%d",&n);
    for(i=0;i<n;i++)
    {
        printf("Enter a value");
        scanf("%d",&a[i]);
    }
```

```
     for(i=0;i<n;i++)
    {
           sum1 = sum1 + a[i] * a[i];
           sum2 = sum2 + a[i];

    }
    sum = sum1-sum2*sum2;
    printf("The value of the series is %d",sum);

}
```

Array

```
Enter number of elements:5
Enter a value1
Enter a value2
Enter a value3
Enter a value4
Enter a value5
The value of the series is -170
```

Array

- Find max element in the array

```
#include<stdio.h>
#define MAX 5
int main()
{
    int arr[MAX],n,i,large;
    printf("Enter number of elements:");
    scanf("%d",&n);
    for(i=0;i<n;i++)
    {
        printf("Enter a value");
        scanf("%d",&arr[i]);
    }
```

Array

```
large = arr[0];
    for(i=0;i<n;i++)
    {
        if(large<arr[i])
            large = arr[i];
    }
    printf("Largest element is %d \n",large);
}
```

WAP to convert number from decimal to binary

- Decimal to binary

```
int main()
{
     int a[20];
     int dec,i=0;
     int j;
     printf("Enter the decimal number to find its binary number\n");
     scanf("%d",&dec);
     while(dec>0)
     {
         a[i]=dec%2;
         i++;
         dec=dec/2;
     }

printf("Binary number of %d is = ",dec);

    for(j=i-1;j>=0;j--)
    {
        printf("%d",a[j]);
    }

}
```

Insertion

- Inserting an element in the array means adding a new data element in an already existing array.
- If the element has to be inserted at the end of existing array, then we just have to add 1 to upperBound and assign value.
- If the memory space allocated to the array is not available then we will not be able to add another element to it.

Insertion

- Algorithm to insert an element in the array
- INSERT(A,N, pos, val)

```
Step 1: [Initialization] Set i = N -1
Step 2: Repeat steps 3 and 4 while i > = pos
Step 3:                 Set A[ i + 1] = A[ i ]
Step 4:                 Set i = i – 1
              [End of loop]
Step 5 : Set A[pos] = val
Step 6: Set  N = N+1
Step 7: Exit
```

Insertion

- // Write a program to insert a number in an array that is already sorted in ascending order

```
#include<stdio.h>
void main( )
{
    int a[20],n,item,i;
    printf("Enter the size of the array");
    scanf("%d",&n);
   printf("Enter elements of the array in the sorted order");

    for(i=0; i<n; i++)
     {
          scanf("%d", &a[i]);
     }
     printf("\nEnter ITEM to be inserted : ");
     scanf("%d", &item);
```

Insertion

```
    i = n-1;
    while(item<a[i] && i>=0)
    {
        a[i+1] = a[i];
        i--;
    }
    a[i+1] = item;
    n++;
    printf("\n\nAfter insertion array is :\n");
    for(i=0; i<n; i++)
    {
            printf("\n%d", a[i]);
    }

}
```

Deletion

- Deleting an element from an array means removing a data element from an already existing array.
- If the element has to be deleted from the end of the existing array, then we just have to subtract 1 from the upperBound.

Deletion

- Algorithm to delete an element from the array
- DELETE (A, N, pos)

```
Step 1: [Initialization] Set i = pos
Step 2: Repeat steps 3 and 4
            while i <= N – 1
Step 3:     Set A[ i ] = A[ i + 1]
Step 4:     Set i = i+1
        [End of loop]
Step 5 : Set N = N -1
Step 6: Exit
```

Deletion

- //Write a program to delete an element at desired position from an array

```
#include<stdio.h>
int main(){
  int a[50],i,pos,size;
  printf("\nEnter size of the array: ");
  scanf("%d",&size);

  printf("\nEnter %d elements in to the array: ",size);
  for(i=0;i<size;i++)
          scanf("%d",&a[i]);
  printf("\nEnter position where to delete: ");
  scanf("%d",&pos);
```

Deletion

- for(i = pos; i<n; i++)
  - a[i] = a[i+1];
- size--;
- for(i=0;i<size;i++)
- printf(" %d",a[i]);

- return 0;
- }

Two dimensional array

- The two-dimensional array is also called a matrix.
- Here is a sample program that stores roll number and marks obtained by a student side by side in a matrix.
- There are two parts to the program—in the first part through a **for** loop we read in the values of roll no. and marks, whereas, in second part through another **for** loop we print out these values.

```
int main()
{
    int stud[4][2] ;
    int i, j ;
for ( i = 0 ; i <= 3 ; i++ )
    {
        printf ( "\n Enter roll no. and marks" ) ;
        scanf ( "%d %d", &stud[i][0], &stud[i][1] ) ;
     }
for ( i = 0 ; i <= 3 ; i++ )
    printf ( "\n%d %d", stud[i][0], stud[i][1] ) ;
}
```

2D array Initialization

```
int stud[4][2] = {
              { 1234, 56 },
             { 1212, 33 },
             { 1434, 80 },
            { 1312, 78 } } ;
  OR
int stud[4][2] = { 1234, 56, 1212, 33, 1434, 80, 1312, 78 } ;
```

2D array Initialization

- It is important to remember that while initializing a 2-D array it is necessary to mention the second (column) dimension, whereas the first dimension (row) is optional.
- Thus the declarations,
- int arr[2][3] = { 12, 34, 23, 45, 56, 45 } ;
- int arr[ ][3] = { 12, 34, 23, 45, 56, 45 } ;

2D array Initialization

- int arr[2][ ] = { 12, 34, 23, 45, 56, 45 } ; int arr[ ][ ] = { 12, 34, 23, 45, 56, 45 } ;
- would never work.

Addition of two matrices

Addition of two matrices:
Rule: Addition of two matrices is only possible if both matrices are of same size.

Suppose two matrices A and B is of same size m X n
Sum of two matrices is defined as

$(A + B)_{ij} = A_{ij} + B_{ij}$

Where $1 \leq i \leq m$ and $1 \leq j \leq n$

Addition of two matrices

$$A = \begin{pmatrix} 5 & 10 & 20 \\ 8 & 6 & 5 \end{pmatrix} \qquad B = \begin{pmatrix} 3 & 8 & 5 \\ 2 & 9 & 3 \end{pmatrix}$$

Addition of two matrixes:

$$A + B = \begin{pmatrix} 5+3 & 10+8 & 20+5 \\ 8+2 & 6+9 & 5+3 \end{pmatrix} = \begin{pmatrix} 8 & 18 & 25 \\ 10 & 15 & 8 \end{pmatrix}$$

Addition of two matrices

- int main() {
-    int i, j, mat1[10][10], mat2[10][10], mat3[10][10];
-    int row1, col1, row2, col2;

-    printf("\nEnter the number of Rows of Mat1 : ");
-    scanf("%d", &row1);
-    printf("\nEnter the number of Cols of Mat1 : ");
-    scanf("%d", &col1);

-    printf("\nEnter the number of Rows of Mat2 : ");
-    scanf("%d", &row2);
-    printf("\nEnter the number of Columns of Mat2 : ");
-    scanf("%d", &col2);

Addition of two matrices

- // Before accepting the Elements Check if no of
-   // rows and columns of both matrices is equal
-   if (row1 != row2 || col1 != col2) {
-      printf("\nOrder of two matrices is not same ");
-      exit(0);
-   }

Addition of two matrices

```
//Accept the Elements in Matrix 1
   for (i = 0; i < row1; i++)
{
      for (j = 0; j < col1; j++)
 {
         printf("Enter the Element a[%d][%d] : ", i, j);
         scanf("%d", &mat1[i][j]);
      }
   }
```

Addition of two matrices

```
 //Accept the Elements in Matrix 2
   for (i = 0; i < row2; i++)
{
      for (j = 0; j < col2; j++)
 {
         printf("Enter the Element b[%d][%d] : ", i, j);
         scanf("%d", &mat2[i][j]);
      }
}
```

Addition of two matrices

```
//Addition of two matrices
   for (i = 0; i < row1; i++)
{
      for (j = 0; j < col1; j++)
{
         mat3[i][j] = mat1[i][j] + mat2[i][j];
      }
}
```

Addition of two matrices

```
 //Print out the Resultant Matrix
   printf("\nThe Addition of two Matrices is : \n");
   for (i = 0; i < row1; i++)
{
      for (j = 0; j < col1; j++)
{
         printf("%d\t", mat3[i][j]);
      }
      printf("\n");
   }

   return (0);
}
```

Transpose of matrix

```
int main() {
   int i, j, mat[10][10], transmat[10][10];
   int row, col;

   printf("\nEnter the number of Rows of Matrix : ");
   scanf("%d", &row);
   printf("\nEnter the number of Cols of Matrix : ");
   scanf("%d", &col);
```

Transpose of matrix

```
 //Accept the Elements in Matrix
   for (i = 0; i < row; i++) {
      for (j = 0; j < col; j++) {
         printf("Enter the Element a[%d][%d] : ", i, j);
         scanf("%d", &mat[i][j]);
      }
   }
```

Transpose of matrix

```
//Transpose of  matrix
   for (i = 0; i < row; i++)
{
      for (j = 0; j < col; j++)
{
         transmat[j][i] = mat[i][j];
      }
}
```

Transpose of matrix

```
//Print out the Resultant Matrix
   printf("\nThe Transpose of  Matrix is : \n");
   for (i = 0; i < col; i++) {
      for (j = 0; j < row; j++) {
         printf("%d\t", transmat[i][j]);
      }
      printf("\n");
   }

   return (0);
}
```

Matrix multiplication

Multiplication of two matrixes:
Rule: Multiplication of two matrixes is only possible if first matrix has size m X **n** and other matrix has size **n** x r.
Where m, n and r are any positive integer.

Multiplication of two matrixes is defined as

$$[AB]_{i,j} = \sum_{s=1}^{n} A_{i,s} B_{s,j}$$

Matrix multiplication

Multiplication of two matrixes is defined as

$$[AB]_{i,j} = \sum_{s=1}^{n} A_{i,s} B_{s,j}$$

Where $1 \leq i \leq m$ and $1 \leq j \leq n$

Matrix multiplication

Suppose two matrixes A and B of size of 2 x 2 and 2 x 3 respectively:

$$A = \begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix} \quad B = \begin{pmatrix} 5 & 6 & 7 \\ 8 & 9 & 10 \end{pmatrix}$$

Multiplication of two matrixes:

$$A * B = \begin{pmatrix} 1*5 + 2*8 & 1*6 + 2*9 & 1*7 + 2*10 \\ 3*5 + 4*8 & 3*6 + 4*9 & 3*7 + 4*10 \end{pmatrix}$$

$$A * B = \begin{pmatrix} 21 & 24 & 27 \\ 47 & 54 & 61 \end{pmatrix}$$

Matrix multiplication

- int main(){
- int a[5][5],b[5][5],c[5][5],i,j,k,sum=0,m,n,o,p;
- printf("\nEnter the row and column of first matrix");
- scanf("%d %d",&m,&n);
- printf("\nEnter the row and column of second matrix");
- scanf("%d %d",&o,&p);
- if(n!=o){
- printf("Matrix mutiplication is not possible");
- printf("\nColumn of first matrix must be same as row of second matrix");
- }

Matrix multiplication

```
else{
      printf("\nEnter the First matrix->");
      for(i=0;i<m;i++)
      for(j=0;j<n;j++)
           scanf("%d",&a[i][j]);
      printf("\nEnter the Second matrix->");
      for(i=0;i<o;i++)
      for(j=0;j<p;j++)
           scanf("%d",&b[i][j]);

```

Matrix multiplication

```
printf("\nThe First matrix is\n");
      for(i=0;i<m;i++){
      printf("\n");
      for(j=0;j<n;j++){
           printf("%d\t",a[i][j]);
      }
      }
      printf("\nThe Second matrix is\n");
      for(i=0;i<o;i++){
      printf("\n");
      for(j=0;j<p;j++){
           printf("%d\t",b[i][j]);
      }
      }
```

Matrix multiplication

```
for(i=0;i<m;i++)
      for(j=0;j<p;j++)
           c[i][j]=0;
      for(i=0;i<m;i++){ //row of first matrix
      for(j=0;j<p;j++){  //column of second matrix
           sum=0;
           for(k=0;k<n;k++)
               sum=sum+a[i][k]*b[k][j];
           c[i][j]=sum;
      }
      }
  }
```

Matrix multiplication

```
 printf("\nThe multiplication of two matrix is\n");
  for(i=0;i<m;i++){
      printf("\n");
      for(j=0;j<p;j++){
           printf("%d\t",c[i][j]);
      }
  }
  return 0;
}
```

C Program to Copy all elements of an array into Another array

```
#include<stdio.h>

int main() {
   int arr1[30], arr2[30], i, num;

   printf("\nEnter no of elements :");
   scanf("%d", &num);

   //Accepting values into Array
   printf("\nEnter the values :");
   for (i = 0; i < num; i++) {
      scanf("%d", &arr1[i]);
   }

   /* Copying data from array 'a' to array 'b */
   for (i = 0; i < num; i++) {
      arr2[i] = arr1[i];
   }

   //Printing of all elements of array
   printf("The copied array is :");
   for (i = 0; i < num; i++)
      printf("\narr2[%d] = %d", i, arr2[i]);

   return (0);
}
```

C Program to Search an element in Array

```
#include<stdio.h>

int main() {
   int a[30], ele, num, i;

   printf("\nEnter no of elements :");
   scanf("%d", &num);

   printf("\nEnter the values :");
   for (i = 0; i < num; i++) {
      scanf("%d", &a[i]);
   }

   //Read the element to be searched
   printf("\nEnter the elements to be searched :");
   scanf("%d", &ele);

   //Search starts from the zeroth location
   i = 0;
   while (i < num && ele != a[i]) {
      i++;
   }

   //If i < num then Match found
   if (i < num) {
      printf("Number found at the location = %d", i + 1);
   } else {
      printf("Number not found");
   }

   return (0);
}
```

C program to merge two arrays

- #include<stdio.h>
-
- int main() {
- int arr1[30], arr2[30], res[60];
- int i, j, k, n1, n2;
-
- printf("\nEnter no of elements in 1st array :");
- scanf("%d", &n1);
- for (i = 0; i < n1; i++) {
- scanf("%d", &arr1[i]);
- }
-
- printf("\nEnter no of elements in 2nd array :");
- scanf("%d", &n2);
- for (i = 0; i < n2; i++) {
- scanf("%d", &arr2[i]);
- }
-
- i = 0;
- j = 0;
- k = 0;
-

C program to merge two arrays

- k = 0;
-
- // Merging starts
- while (i < n1 && j < n2) {
- if (arr1[i] <= arr2[j]) {
- res[k] = arr1[i];
- i++;
- k++;
- } else {
- res[k] = arr2[j];
- k++;
- j++;
- }
- }

C program to merge two arrays

- j++;
- }
-
- //Displaying elements of array 'res'
- printf("\nMerged array is :");
- for (i = 0; i < n1 + n2; i++)
- printf("%d ", res[i]);
-
- return (0);
-
- }

Sort an array

- Bubble sort is a very simple method that sorts the array elements by repeatedly moving the largest element to the highest index position of the array (in case of arranging elements in ascending order).
- In bubble sorting, consecutive adjacent pairs of elements in the array are compared with each other.
- If the element at lower index is greater than the element at the higher index, the two elements are interchanged so that the smaller element is placed before the bigger one.
- This process is continued till the list of unsorted elements exhaust.

Bubble Sort

```
              30  52  29  87  63  27  18  54
Pass1:        30 29  52   63  27  18  54  87
Pass2:        29  30  52  27  18   54  63 87
Pass3:        29  30  27  18  52   54  63 87
Pass4:        29  27  18  30  52   54  63 87
Pass5:        27  18  29  30  52   54  63 87
Pass6:        18  27  29  30  52   54  63 87
Pass7:        18  27  29  30  52   54  63 87
```

Bubble Sort

```
SORT(A, N)

Step 1: Repeat steps 2 For I = 0 to N-1
Step 2:  Repeat For J = 0 to N – I -1
Step 3:     If A[J] > A[J + 1], then
         SWAP A[J] and A[J+1]
         [End of Inner Loop]
    [End of Outer Loop]
Step 4: EXIT
```