14-06-2024

POINTER TO AN ARRAY :-

(*p) [5] → POINTER TO AN ARRAY ONLY POINTER
↓
COLUMNS

```
→ main ( )
{
  int a [5][5], r, c, i, j, (*p)[5];
  p = a;
  printf ("enter r & c");
  scanf ("%d %d", &r, &c);
  printf ("enter matrix");
  for (i=0; i<r; i++)
   for (j=0; j<c; j++)
     scanf ("%d", *(a+i)+j);
   for (i=0; i<10; i++, printf ("\n"))
    for (j=0; j<c; j++)
       printf ("%d", *(*(p+i)+j));
}
```

Array of Pointers

n POINTERS

ARRAY OF POINTERS :-

```
-> main()
{
    int a=5, b=6, c=7, *p[3];
    p[0] = &a;
    p[1] = &b;
    p[2] = &c;
        printf("%d %d %d", *p[0], *p[1], *p[2]);
}
```

POINTER TO AN ARRAY

~~* p[3]~~

-> (*p)[n]

one pointer

ARRAY OF POINTERS

-> ~~(*p)[n]~~

-> *p[n]

n - Pointers

```
-> main()
{
    char a[20] = "abc";
    char *p = "def";
    printf("%s %s", a, p);
}
```

```
-> main ( )
{
  char *p[5] = {" apple ", "bat ", "cat ", "dog ", "egg "};
  int i;
  for(i=0; i<5; i++)
  printf (" %s \n", p[i]);
}
```

```
-> void bubble_sort (int * , int );
 main ( )
 {
  int a[20], i, n;
  printf ("enter n ");
  scanf (" %d ", & n);
  printf ("enter elements ");
  for (i=0; i< n; i++)
    scanf (" %d ", & a[ i]);
  bubble_sort (a, n);
   for (i= 0; i< n; i++)
    printf (" %d ", a[i]);
 }
 void bubble_sort (int a[ ], int n)
 {
  int i, j, t ;
  for ( i =0; i <n-1; i++)
  {
```

```
for( j=0; j<n-i-1; j++)
{
   if (a[j] > a[j+1])
   {
      t = a[j];
      a[j] = a[j+1];
      a[j+1] = t;
   }
}
}
}
```

1. Searching
2. Adding of 2 Matrices
using POINTERS
[2 Pages back]

1. To search an element in an array using Pointers

```
-> ~~int~~ main ()
{
   int a[100], size, i, find, found = 0, *p;
   printf (" Enter size of an array : ");
   scanf ("%d", & size);
   p = a;
   printf ("enter elements : ");
   for( i=0; i< size; i++)
   {
      scanf ("%d", p+i);
   }
printf ("Enter the element to be searched ");
scanf (" %d ", & find );

for (i = 0; i< size; i++)
{
   if (*(p+i)= = find )
   {
      found = 1;
      break;
   }
}
if ( found= =1)
{
   printf ("%d is found at position %d\n", find, i+1);
   printf ("%d is found at index position %d\n",
                  find, i);
}
else
   printf ("%d is not present ", find );
}
```

O/P:-
```
Enter size of an array : 5
Enter elements :
   40
   50
   30
   10
   20
Enter the element to be searched: 30
30 is found at position 3
30 is found at index position 2
```

2. Addition of 2 matrices using Pointers :-

```
→ int main()
  {
    int a[100][100], b[100][100], c[100][100], *p, *q, *r;
    int i, j, r1, c1;
    printf ("Enter no of rows (between 1 and 100): ");
    scanf ("%d", &r1);
    printf ("Enter no of columns (between 1 and 100): ");
    scanf ("%d", &c1);

    printf ("\n Enter matrix A: \n");
    for (i=0; i<r1; i++)
       for (j=0; j<c1; j++)
          scanf ("%d", (a[i]+j));

    printf ("\n Enter matrix B: \n");
    for (i=0; i<r1; i++)
       for (j=0; j<c1; j++)
          scanf ("%d", (b[i]+j));

    p = &a[0][0];
    q = &b[0][0];
    r = &c[0][0];

    for (i=0; i<r1; i++)
       for (j=0; j<c1; j++)
          *(r+i*c1+j) = *(p+i*c1+j) +
                        *(q+i*c1+j);
```

```
printf ("\n Sum of 2 matrices: \n");
for (i = 0 ; i < r1; i++)
   for (j = 0; j < c1; j++)
   {
      ~~printf ("%d~~
      printf ("%d ", *(r + i * c1 + j));
      if (j == c1 - 1)
         printf ("\n");
   }
}
```

O/P :-

```
Enter no of rows (between 1 and 100): 2
Enter no of columns (between 1 and 100): 2
Enter Matrix A:
 1   2
 3   4
Enter Matrix B:
 5   6
 7   8
Sum of ~~two~~ 2 matrices:
 6   8
 10  12
```