POINTER

- **Pointer**

A pointer is a variable that holds the address of a variable i.e.; a pointer contains memory address, not values of that variable.

The syntax of pointer declaration is

data_type *variable_name;

Where data_type is the type of pointer variable such as int, float, char etc. and variable name is a name of pointer variable.

Example **int *ptr;**

declares an integer pointer called ptr.The ***** operator is used to define that ptr is an integer type pointer variable.

//Basic program in pointer

```
#include<stdio.h>
#include<conio.h>
void main()
{
 int j=5;
 int ptr,**ptr2;
 ptr=&j;
ptr2=&ptr;
 printf("\nThe value of j=%d",j);
 printf("\nThe address of j=%u",&j);
 printf("\n\nthe address of j=%u",ptr);
 printf("\nthe value of j=%d",*ptr);
printf("\nthe value of j=%d",**ptr2);
 getch();
 }
```

Output:

```
 The value of j=5
 The address of j=6524
 the address of j=6524
 the value of j=5
the value of j=5
```

- **POINTER ARITHMETIC**

Various arithmetic operations are performed on pointers. The basic pointer arithmetic are Addition (+), Subtraction (-), Increment (++), Decrement (--).

```
int *ptr ;
int k=10;
ptr=&k;
```

Here ptr integer type pointer variable. ptr holds the address of the variable k. Suppose the value of ptr=6700, then value of ptr++ will be 6702.Similarly, if the value of ptr=6700, then value of ptr— will be 6698.Because of, 2 bytes memory spaces are allocated for integer data type. Similarly increment (++) and decrement (--) operators are acted as a same fashion according to its data type.

Suppose;

```
 char *ptr;
 char k='C';
  ptr=&k;
```

if value of ptr=7000, ptr++ will be 7001.if value of ptr=7000, ptr—will be 6999. Because of, 1 byte memory space is allocated for character data type. Same way we can use addition and subtraction operation on pointer arithmetic. If value of ptr=7000, ptr=ptr+10 will be 7010. If value of ptr=7000, ptr=ptr-10 will be 6990.

- **POINTERS AND ARRAYS**

In C, a strong relationship exits between pointers and arrays. Any operation on array can be done faster and efficiently using pointer.

Suppose, int a[5]={7,9,4,8,2};

You want to access i^{th} element of that array where i is the integer. It can be written as *(a+i) instead of a[i].In C programmingconcept a[i] and *(a+i) are equivalent.

9) i. One Dimension Array representation

```
#include<stdio.h>
#include<conio.h>
void main()
{
 int a[5]={7,9,4,8,2};
 int *ptr;
 int i=0;
 ptr=&a[0];
 clrscr();
 while(*ptr!='\0')
```

```
 {
 printf("\n\nthe address of a[%d]=%u",i,ptr);
 printf("\nthe value of a[%d]=%d",i,*ptr);
 ptr++;
 i++;
 }
 getch();
 }
```

Output:

```
the address of a[0]=65516
the value of a[0]=7

the address of a[1]=65518
the value of a[1]=9

the address of a[2]=65520
the value of a[2]=4

the address of a[3]=65522
the value of a[3]=8

the address of a[4]=65524
the value of a[4]=2
/* The elements of array are  stored in the memory as shown*/
                        a[0]      a[1]      a[2]      a[3]      a[4]
               value     7         9         4         8         2

        65516    65518    65520   65522    65524
```

- **ARRAY OF POINTERS:**

An array of pointer is similar to an array of any predefined data type. As a pointer variable always contains an address, an array of pointers is collection of address. These can be addresses of ordinary isolated variables or array elements.

//One Dimension Array representation by pointer

```
#include<stdio.h>
#include<conio.h>
void main()
{
```

```
int i,*a[10],n;
clrscr();
printf("Enter the size of array----");
scanf("%d",&n);
printf("\nEnter the array elements----");
for(i=0;i<n;i++)
scanf("%d",&a[i]);
printf("\n The displayed array\n\n");
for(i=0;i<n;i++)
printf("\t%d",*a[i]);
getch();
}
```

Output

```
Enter the size of array----5
Enter the array elements----
12
1
34
45
56
 The displayed array
      12      1       34      45      56
```

- **CONSTANT POINTER**

The general declaration of constant pointer is
data_type *const variable_name=K; where **K** is a constant parameter.

```
int *const ptr=20;
```

Here any pointer arithmetic operations are invalid.

- **POINTER TO POINTER**

C supports, a pointer pointing to another pointer that points target value.
The general format of declaring a pointer to pointer is

```
data_type ** variable_name;
```

Here ** refers to a pointer to a pointer to a target value.

Suppose,

```
int P=10;
```

```
int *ptr1;
int **ptr2;
ptr1=&P;
ptr2=&ptr1;
```

- **NULL pointer**

A NULL pointer does not point to any object or function. There is conceptual difference between uninitialized pointer and NULL pointer. An uninitialized pointer may point to anywhere but NULL pointer compares a pointer to any object or function.

A NULL pointer is initiated by

(i) #define NULL ((void*)0)

Or

(ii) #define NULL(type) (type*)0

Both assign the value 0 to the pointer.