POINTERS:

A pointer is a variable that points to a memory location in which data is stored. We can access and change the contents of this memory location via the pointer.

Pointer declaration

A pointer is a variable that contains the memory location of another variable. The syntax is as shown below. You start by specifying the type of data stored in the location identified by the pointer. The asterisk tells the compiler that you are creating a pointer variable. Finally you give the name of the variable.

type * variable name

Example:

```
int *ptr;
float *string;
```

Address operator

ptr=#

This places the address where num is stores into the variable ptr. If num is stored in memory 21260 address then the variable ptr has the value 21260.

/* A program to illustrate pointer declaration*/

```
main()
{
int *ptr;
int sum;
sum=45;
ptr=∑
printf (“\n Sum is %d\n”, sum);
printf (“\n The sum pointer is %d”, ptr);
}
```

The pointer contains the address 21260 the value 45

Pointer expressions & pointer arithmetic

Like other variables pointer variables can be used in expressions. For example if p1 and p2 are properly declared and initialized pointers, then the following statements are valid.

```
y=*p1*p2;
sum=sum+*p1;
z= 5* - *p2/p1;
*p2= *p2 + 10;
```

C allows us to add integers to or subtract integers from pointers as well as to subtract one pointer from the other. We can also **use short hand operators** with the pointers **p1+=; sum+=*p2;** etc.

We can also **compare pointers by using relational operators** the expressions such as **p1 >p2,** p1==p2 and p1!=p2 are allowed.

/*Program to illustrate the pointer expression and pointer arithmetic*/

```
#include< stdio.h >
main()
{
int p1,p2;
int a,b,x,y,z;
a=30;b=6;
p1=&a;
p2=&b;
x=*p1+ *p2 –6;
y=6*- *p1/ *p2 +30;
printf(“\nAddress of a +%u”,p1);
printf(“\nAddress of b %u”,p2);
printf(“\na=%d, b=%d”,a,b);
printf(“\nx=%d,y=%d”,x,y);
p1=p1 + 70;
p2= p2;
printf(“\na=%d, b=%d”,a,b);
}
```

Pointer to arrays

An array is actually very much like pointer. We can declare the arrays first element as a[0] or as int *a

- a[0] is an address
- *a is also an address

The difference is pointer is a variable and can appear on the left of the assignment operator. The array name is constant and cannot appear as the left side of assignment operator.

/* A program to display the contents of array using pointer*/

```
void main()
{
int a[100];
int i,j,n;
printf(“\nEnter the elements of the array\n”);
scanf(“%d”,&n);
printf(“Enter the array elements”);
for(I=0;I< n;I++)
{
```

```
scanf("%d",&a[I]);
printf("Array element are");
}
for(ptr=a, ptr< (a+n); ptr++)
printf("Value of a[%d]=%d stored at address %u",j+=,*ptr,ptr);
}
```

Strings are characters arrays and here last element is 0 arrays and pointers to char arrays can be used to perform a number of string functions.

Pointers and structures

We know the name of an array stands for the address. Suppose item is an array variable of struct type. Consider the following declaration:

```
struct products
{
char name[30];
int manufac;
float net;
item[2],*ptr;
}
```

This statement declares item as array of two elements, each type struct products and ptr as a pointer data objects of type struct products, the

assignment ptr=item;

would assign the address of zeroth element to product[0]. Its members can be accessed by using the following notation.

```
ptr- >name;
ptr- >manufac;
ptr- >net;
```

The symbol - > is called arrow pointer and is made up of minus sign and greater than sign.

When the pointer is incremented by one it is made to print to next record ie item[1]. The following statement will print the values of members of all the elements of the product array.

for(ptr=item; ptr< item+2;ptr++)
printf("%s%d%f\n",ptr- >name,ptr- >manufac,ptr- >net);

We could also use the notation (*ptr).number to access the member number. The parenthesis around ptr are necessary because the member operator. Has a higher precedence than the operator *.