Pointers

A pointer is a variable that represents the location or address of a variable or array element.

Uses

1. They are used to pass information back and forth between a function and calling point.
2. They provide a way to return multiple date items.
3. They provide alternate way to access individual array elements.

When we declare a variable say x, the computer reserves a memory cell with name x. the data stored in the variable is got through the name x. another way to access data is through the address or location of the variable. This address of x is determined by the expression &x, where & is a unary operator (called address operator). Assign this expression &x to another variable px(i.e. px=&x).this new variable px is called a pointer to x (since it points to the location of x. the data stored in x is accessed by the expression *px where * is a unary operator called the indirection operator.
Ex: if x=3 and px=&x then *px=3

Declaration and Initialisation

A pointer variable is to be declared initially. It is done by the syntax.
Data type *pointer variable
Int *p declares the variable p as pointer variable pointing to a an integer type data. It is made point to a variable q by p= &q. In p the address of the variable q is stored. The value stored in q is got by *p.
If y is a pointer variable to x which is of type int, we declare y as int *y ;

Ex:
```
float a;
float *b;
b=&a;
```

Note : here in 'b' address of 'a' is stored and in '*b' the value of a is stored.

Passing pointers to a function

Pointers are also passed to function like variables and arrays are done. Pointers are normally used for passing arguments by reference (unlike in the case if variable and arrays, they are passed by values). When data are passed by values the alteration made to the data item with in the function are not carried over to the calling point; however when data are passed by reference the case is otherwise. Here the address of data item is passed and hence whatever changes occur to this with in the function, it will be retained through out the execution of the program. So generally pointers are used in the place of global variables.

Ex:):

```
#include<stdio.h>
void f(int*px, int *py
main()
{
int x = 1;
int y=2;
f(&X,&y);
printf("\n %d%d", x,y);
}
Void f(int *px, int *py);
*px=*px+1;
*py=*py+2;
return;
}
```

Note:

1. here the values of x and y are increased by 1 and 2 respectively.
2. arithmetic operations *, +, -, / etc can be applied to operator variable also.

Pointer and one dimensional arrays

An array name is really a pointer to the first element in the array i.e. if x is a one dimensional array, the name x is &x[0] and &x[i] are x + i for i= 1,2,……. So to read array of numbers we can also use the following statements

```
int x[100],n;
for (i=1 ; i<=n;  ++i)
scanf (‘%d”, x + i )
```

(in the place of scanf (“%d”, &x[i]))

Note : the values stored in the array are got by * (x + i) in the place x[i].

Dynamic memory allocation

Usually when we use an array in c program, its dimension should be more than enough or may not be sufficient. To avoid this drawback we allocate the proper (sufficient) dimensions of an array during the run time of the program with the help of the library functions called memory management functions like ‘malloc’, ‘calloc’, ‘realloc’ etc. The process of allocating memory at run time is known as dynamic memory allocation.

Ex;

to assign sufficient memory for x we use the following statement

x= (int *) malloc (n* sizeof (int)) , for in the place of initial declaration int x[n]

Similarly in the place of float y [100] we use y = (float *) malloc (m* sizeof (float));

Example to read n numbers and find their sun

```
Main()
{
      int *x, n, i, sum=0;
      Printf("\n Enter number of numbers");
      Scanf("%d", &n);
      x=(int *)malloc(n * sizeof(int));
      for(i=1;i<=n,++i)
      {
            scanf("%d", x+i):
            sum +=  *(x+i);
      }
      Printf("\nThe sum is %d ", sum);
}
```

Passing function to other function

A pointer to a function can be passed to another pointer as an assignment. Here it allows one function to be transferred as if the function were a variable.

Structures and Unions

We know an array is used to store a collection of data of the same type. But if we want to deal with a collection of data of various type such as integer, string, float etc we use structures in C language. It is a method of packing data of different types. It is a convenient tool for handling logically related data items of bio-data people comprising of name, place, date etc. , salary details of staff comprising of name, pay da, hra etc.

Defining a structure.

In general it is defined with the syntax name **struct** as follows

```
Struct  structure_name
{
        Data type variable1;
        Data type  variable2;
```