**Pointer:**

Pointer is variable which stores address of another variable of same datatype.

Pointer variable contains address of another variable of same datatype.

Pointers are frequently used in ‘C’ as they provide number of benefits to programmer, some of them are as follows:

1. Pointers are more efficient in handling array and database.
2. Pointers allows to alter function arguments.
3. By using pointer we can implement dynamic memory allocation.
4. Pointer increases the execution speed of program and reduce length of program.
5. Pointers help us to build complex datastructures like stack, queue, linked list, trees, etc.

**Declaration of pointer:**

Pointer can be declared in 2 ways as

**Syntax :**

**1.** datatype \*pointer\_name; eg. int \*p;

2. datatype\* pointer\_name; eg. int\* p;

This declaration tells that

1. \* (asterisk) tells that variable pointer\_name is pointer variable.
2. Pointer\_name points to variable of datatype.

**Initialization of pointer:**

Process of assigning address of variable is known as initialization.

The address-of operator (&) is used to assign address to pointer.

Eg. char i ;

char \*p;

p=&a;

**OR**

char i ;

char \*p=&i;

Here, we can access value of i or can assign value to i using value-at (\*) operator.

ie. \*p=’a’;

**Program :**

/\* Program for pointer introduction \*/

#include<stdio.h>

#include<conio.h>

void main()

{

float i, \*p;

p=&i;

printf(“Enter value of float”);

scanf(“%f”,p);

printf(“ Value %f is stored at %u”,\*p,p);

}

**Chain of pointer:**

It is possible to make pointer to pointer to point another pointer , thus creating chain of pointer.

Eg. int i=10;

int \*p;

int \*\*q;

int \*\*\*r;

p=&i;

q=&p;

r=&q;

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Name | i | P | q | r |
| Value | 10 | 2000 | 2002 | 2004 |
| Address | 2000 | 2002 | 2004 | 2006 |

Here, r points to q , q points p, p points to i and we can access value of i as

\*p=10; or

\*\*q =10; or

\*\*\*r=10;

**Pointer to array:**

Array is collection of elements of same datatype.

Eg. int a[10];

This will create array of integers of size 10 with name a ie. a[0] , a[1] , a[2]……. a[9].

Now , we can create pointer points to this array as

int \*p;

p=a;

Here , & is not used because array name itself is address of first element in array and it can be also initialized as p=&a[0]; or

p=&a[1]; or

p=&a[2]; and

so on….

When we increment p, it will point next element in array ie. to print all elements in array ‘a’, we can write as

for(i=0;i<10;i++){

printf(“%d”,\*p);

p++;

}

**Pointer to string:**

String is array of characters. Pointer to string is same as pointer to array except that we can access string only by (pointer\_name) or not by (\*pointer\_name).

Eg. char a[]=”C\_LANGUAGE”,\*p;

p=a;

printf(“%s=%s”,a,p);

**Pointer to structure:**

Structure is user defined datatype which can hold information of different datatype;

Eg. struct stud

{

char name[10];

int roll\_no;

float fees;

};

Struct stud s;

This is structure of student which hold information of name,roll\_no and fees of student. We can access information of structure by . (DOT) operator.

Eg. s.name

s.roll\_no

s.fees

Where , s is structure variable. We can create pointer points to this structure as

struct stud \*p;

p=&s;

We can access information as

p->roll\_no or (\*p).roll\_no

p->name or (\*p).name

p->fees or (\*p).fees

**Pointer to Function:**

Function is block of statements used to perform particular task.

Eg. int add(int a, int b)

{

int sum=0;

sum=a+b;

return sum;

}

This function returns addition of a and b. We can create pointer which points to this function as

**Declaration:**

Type (\*pointer\_name)(parameter\_list);

**Initialization:**

**(**pointer\_name)=(function\_name);

**Accessing:**

(\*pointer\_name)(parameters);

**Eg…**

int (\*p)(int,int); //Declaration

p=add; //Initialization

(\*p)(5,10); //Accessing