Pointer

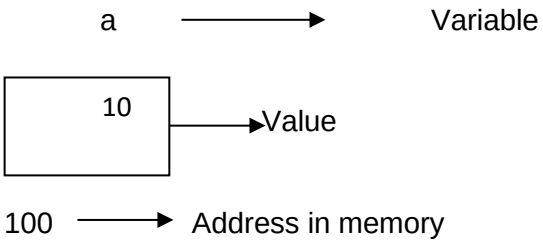
Definition:

Pointer is variable which can hold address of another variable. Pointer is a variable that contains the address which is the location of another variable in the memory.

Concept:

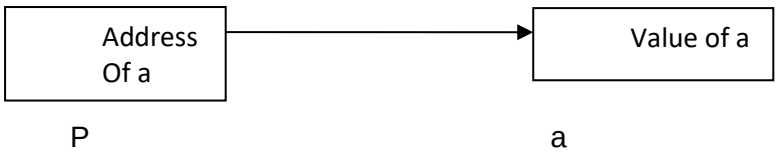
Whenever user declares any variable, memory is allocated for that variable depending on the data type. E.g. If we are declaring integer value , 2 bytes will be allocated at some memory location.

int a = 10;



The variable a is associated with the memory address 100. Since memory addresses is integer value , we can store it in another variable, Such variables that can contain the memory address are called Pointers.

Suppose , we assign the address of a to variable P , the P is called the pointer as it points to the variable a.



Now P is again a variable, so P will also get some memory location.

All Pointers gets 2 bytes, it is not depending on the data type. Pointer can only store the address of another variable and address is Integer value. So all the pointer gets 2 bytes in memory.

In above example:

Variable	value	address (memory location)
a	10	100
P	100(address of a)	1200

value of P = address of a

Syntax:

Data type * variable name;

e.g. To declare integer pointer i.e Pointer to integer value

int *p; //p is integer pointer (*)indicates you are declaring pointer variable

char *ch; //ch is character pointer

Address and Dereferencing (& and *) Operators:

The & operator is called 'address of operator' . It is unary operator.

e.g.

int x = 10;

The memory address is automatically assigned for this data item (variable) by compiler.

We can assign the address of x to another variable P as,

P = &x; // it assigns 100(address of variable x) to variable P.

Accessing the value variable through pointer (using * operator)

x = *p; // *p and x represents the same value.

Accessing to an object (variable) pointer is called Dereferencing and the asterisk (*) operator is called 'Dereferencing or indirection operator'.

Pointer Initialization:

- After declaration, we have to initialize the pointer variable. Use of the 'address of (&) 'operator as a prefix to the variable name assigns its address to the pointer.
- We can also assign a pointer value to another variable of the same data type.

Example:

int a=5, x,*p;

p= &a; //assigns the address of variable 'a' to pointer 'p'

x = *p; //assigns the value at address pointed by 'p' to variable 'a' i.e. x=5

*p = 0; // assigns value 0to 'a' since *p is same as 'a'.

Two Statements,

p = &a; and x=*p;

are equivalent to single statement, x = *(&a); or x=a;

That is & operator is the inverse of * operator.

Pointer Arithmetic:

Consider,

int i = 3;

int *ptr;

ptr=&i;

Addition of two pointers, multiplication of a number with pointer and division of a pointer with number is not possible and following operations on ptr are valid.

Operation	Expression	Meaning
Increment	ptr++	Increments ptr to point to next location of same type. If ptr is integer pointer and ptr = 100 then ptr++ gives 102. If ptr is character pointer then it gives 101.
Decrement	ptr--	Decrements ptr to point to previous location of same type. If ptr is integer pointer and ptr = 100 then ptr— gives 98. If ptr is character pointer then it gives 99.
Adding a number to pointer	ptr=ptr+8	ptr pointes to 9th integer location after current location. If ptr is 100 then ptr+8 will gives 116(8 integer requires 16 bytes)
Subtracting a number from a pointer	ptr=ptr-8	ptr pointes to 9th integer location before current location. If ptr is 100 then ptr-8 will gives 84(8 integer requires 16 bytes)
Subtracting one pointer from another	p= p1 – p2 (p,p1,p2 are pointers)	We can subtract one pointer from another if and only if both are pointing two one array. Here p,p1 and p2 are pointing to same array.