# POINTERS

## Introduction:

- ✓ A pointer is a derived data type.
- ✓ This is the one which stores the address of data in memory, we will be in position to access the data directly and do calculations over it.
- ✓ The standard concept is, we access the data from memory using variable name it gets the data and operations are done over them. But the pointer is different that the accessing is done by address the data is stored so that it will be advantage of decreasing the instructions and overheads of standard usage.

## Definition:

A pointer is a variable which contains the address of another variable.

**Advantages of pointer**

– Enables us to access a variable that is defined outside the function.

– Can be used to pass information back and forth between a function and its reference point.

– More efficient in handling data tables.

– Reduces the length and complexity of a program.

– Sometimes also increases the execution speed.

## Declaring of a pointer variable

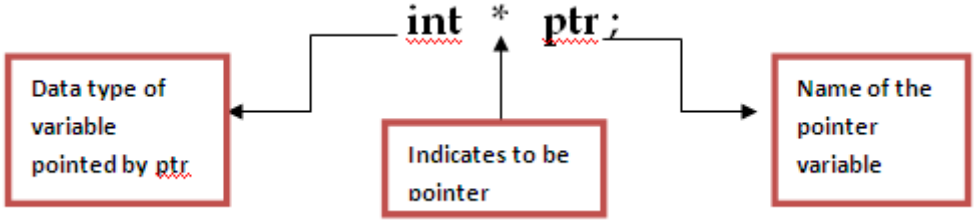
*General form:*

**data_type *pointer_name;**

Three things are specified in the above declaration:

- The asterisk (*) tells that the variable pointer_name is a pointer variable.
- pointer_name needs a memory location.
- pointer_name points to a variable of type data_type which may be in, float, double etc..

**Example:**



- ✓ Where, ptr is not an integer variable but ptr can hold the address of the integer variable i.e. it is a '**pointer to an integer variable'**, but the declaration of pointer variable does not make them point to any location.
- ✓ we can also declare the pointer variables of any other data types .

**For example:**

```
double  * dptr;
char * ch;
float *  fptr;
```

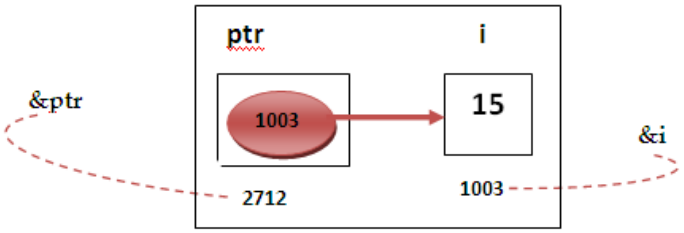
## Dereference operator (*) and Address operator (&) – Pointer operators

- ✓ The unary operator (*) is the dereferencing pointer or indirection pointer, when applied to the pointer can access the value the pointer points to.
- ✓ The address operator (&) is the referencing pointer or direction pointer, when applied to the pointer can access the address of the pointer holds.

To understand the concept more clearly,

let' s consider the program segment given below :

```
int  i= 15;
int * ptr;
ptr=&i;
```




```
printf("Value of  i  is :%d",i);
printf("Address of  i  is :%d",&i);
printf("Value of  i  is :%d",*ptr);
printf("address of i  is :%d",ptr);
printf("address of pointer is :%x",&ptr);
```

**Output:**

```
Value of  i  is : 15
Address of i is : 1003
Value of i:  15
Address  of  i  is  : 1003
Address  of  ptr  is : 2712
```

**Note:**

- *A variable is declared by giving it a type and a name (e.g.* ***int k;****)*
- *A pointer variable is declared by giving it a type and a name (e.g.* ***int *ptr****) where the asterisk tells the compiler that the variable named* ***ptr*** *is a pointer variable and the type tells the compiler what type the pointer is to point to (integer in this case).*
- *Once a variable is declared, we can get its address by preceding its name with the unary* ***&*** *operator, as in* ***&k****.*
- *The size of a pointer is dependent upon the architecture of the computer.*
- *The size of the pointer is always the same irrespective of the data type it is pointing to.*
- *We can "dereference" a pointer, i.e. refer to the value of that which it points to, by using the unary '*' operator as in* ****ptr****.*
- *An "lvalue" of a variable is the value of its address, i.e. where it is stored in memory. The "rvalue" of a variable is the value stored in that variable (at that address).*

## Initialization of pointers

Initializing a pointer variable is a important thing, it is done as follows

```
int x;
int *p = &x;
```

Here the two steps are done at once the two steps are

```
int *p;
*p=&x;
```

The combination of these two steps gives us the initialization of pointers.