

pointer

The pointer in C language is a variable which stores the address of another variable. This variable can be of type int, char, array, function, or any other pointer.

Consider the following example to define a pointer which stores the address of an integer.

```
int n = 10;
```

```
int* p = &n; // Variable p of type pointer is pointing to the address of the variable n of type integer.
```

Declaring a pointer

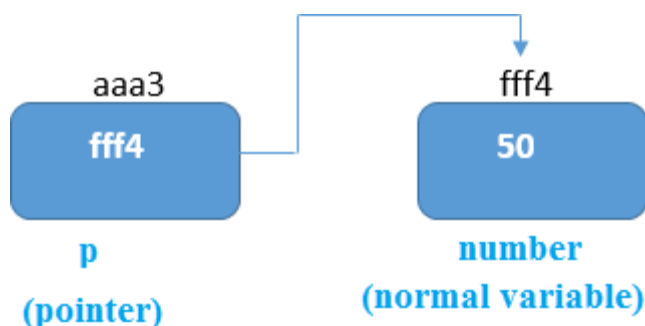
The pointer in c language can be declared using * (asterisk symbol). It is also known as indirection pointer used to dereference a pointer.

```
int *a; // pointer to int
```

```
char *c; // pointer to char
```

Pointer Example

An example of using pointers to print the address and value is given below.



javatpoint.com

pointer

As you can see in the above figure, pointer variable stores the address of number variable, i.e., fff4. The value of number variable is 50. But the address of pointer variable p is aaa3.

By the help of * (**indirection operator**), we can print the value of pointer variable p.

Let's see the pointer example as explained for the above figure.

```
#include<stdio.h>
#include<conio.h>
int main(){
    int number=50;
    int *p;
    p=&number;//stores the address of number variable
    printf("Address of p variable is %x \n",p); // p contains the address of the number therefore printing p gives the address of number.
    printf("Value of p variable is %d \n",*p); // As we know that * is used to dereference a pointer therefore if we print *p, we will get the value stored at the address contained by p.
    return 0;
}
```

Output:

Address of p variable is c1e5fb0c

Value of p variable is 50.

Advantage of pointer

- 1) Pointer **reduces the code** and **improves the performance**, it is used to retrieving strings, trees, etc. and used with arrays, structures, and functions.
- 2) We can **return multiple values from a function** using the pointer.

3) It makes you able to **access any memory location** in the computer's memory.

Usage of pointer

There are many applications of pointers in c language.

1) Dynamic memory allocation

In c language, we can dynamically allocate memory using malloc() and calloc() functions where the pointer is used.

2) Arrays, Functions, and Structures

Pointers in c language are widely used in arrays, functions, and structures. It reduces the code and improves the performance.

Address Of (&) Operator

The address of operator '&' returns the address of a variable.

```
#include<stdio.h>
```

```
#include<conio.h>
```

```
int main()
```

```
{
```

```
int number=50;
```

```
printf("value of number is %d, address of number is %x",number,&number);
```

```
return 0;
```

```
}
```

Output:

value of number is 50, address of number is effaea34

NULL Pointer

A pointer that is not assigned any value but NULL is known as the NULL pointer. If you don't have any address to be specified in the pointer at the time of declaration, you can assign NULL value. It will provide a better approach.

```
int *p=NULL;
```

Pointer Program to swap two numbers without using the 3rd variable.

```
#include<stdio.h>
#include<conio.h>
int main()
{
    int a=10,b=20,*p1=&a,*p2=&b;

    printf("Before swap: *p1=%d *p2=%d",*p1,*p2);
    *p1=*p1+*p2;
    *p2=*p1-*p2;
    *p1=*p1-*p2;
    printf("\nAfter swap: *p1=%d *p2=%d",*p1,*p2);

    return 0;
}
```

Output

```
Before swap: *p1=10 *p2=20
After swap:  *p1=20 *p2=10
```

Pointer to Array:

Pointers are variables which stores the address of another variable. When we allocate memory to a variable, pointer points to the address of the variable. Unary operator (*) is used to declare a variable and it returns the address of the allocated memory. Pointers to an array points the address of memory block of an array variable.

We can declare pointer to array like following :

```
int a[5];
int *p;
p=a; p=&a[0];
```

The following is the syntax of array pointers.

```
datatype *variable_name[size];
```

Here,

datatype – The datatype of variable like int, char, float etc.

variable_name – This is the name of variable given by user.

size – The size of array variable.

Example

```
#include <stdio.h>
#include<conio.h>
int main () {
    int *arr[3];
    int *a;
    a=&arr;
    printf( "Value of array pointer variable : %d\n", arr);
    printf( "Value of pointer variable : %d\n", &a);
    return 0;
```

```
}
```

Output:

Value of array pointer variable : 1481173888

Value of pointer variable : 1481173880

Array of Pointer:

```
int a[]={ 10,20,30,40};
```

```
int *p[4];/*p[0],*p[1],*p[2], *p[3]
```

Example:

```
#include<stdio.h>
```

```
#include<conio.h>
```

```
void main ( )
```

```
{
```

```
    int a[3] = { 10,20,30};
```

```
    int *p[3],i;
```

```
    for (i=0; i<3; i++)
```

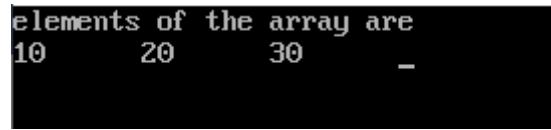
```
        p[i] = &a[i];
```

```
    printf ("elements of the array are \n");
```

```
    for (i=0; i<3; i++)
```

```
        printf ("%d \t", *p[i]);
```

```
}
```

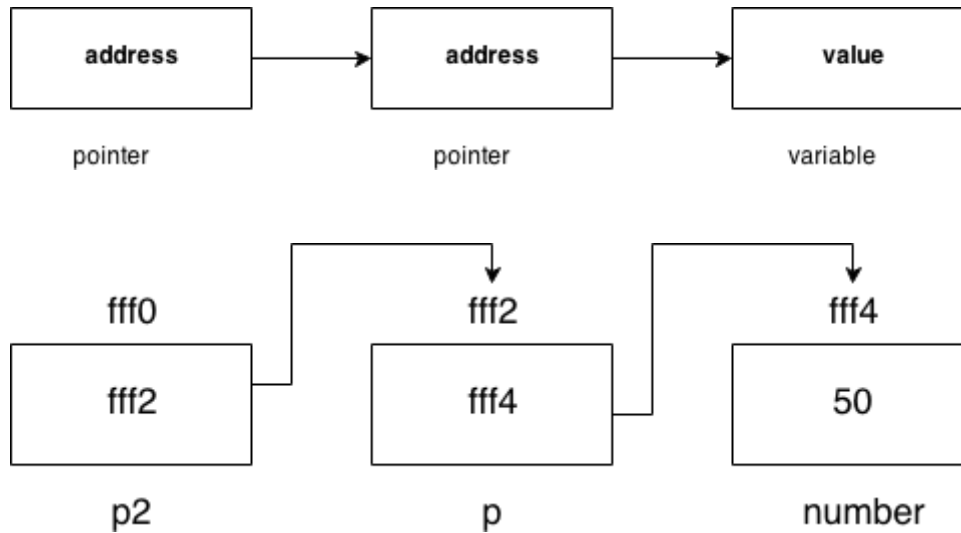


```
elements of the array are
10      20      30      -
```

Pointer to Pointer

As we know that, a pointer is used to store the address of a variable in C. Pointer reduces the access time of a variable. However, In C, we can also define a pointer to store the address of another pointer. Such pointer is known as a double pointer (pointer to pointer). The first pointer is used to store the address of a variable whereas the second pointer is used to store the address of the first pointer.

pointer



The syntax of declaring a double pointer is given below.

int **p; // pointer to a pointer which is pointing to an integer.

Example.

```
#include<stdio.h>
```

```
#include<conio.h>
```

```
void main ()
```

```
{
```

```
    int a = 10;
```

```
    int *p;
```

```
    int **pp;
```

```
    p = &a; // pointer p is pointing to the address of a
```

```
    pp = &p; // pointer pp is a double pointer pointing to the address of pointer p
```

```
    printf("address of a: %x\n",p); // Address of a will be printed
```

```
    printf("address of p: %x\n",pp); // Address of p will be printed
```

```
printf("value stored at p: %d\n",*p); // value stored at the address contained by  
p i.e. 10 will be printed
```

```
printf("value stored at pp: %d\n",**pp); // value stored at the address  
contained by the pointer stored at pp  
}
```

Output:

address of a: 9a695534

address of p: 9a695538

value stored at p: 10

value stored at pp: 10

Function Pointer

As we know that we can create a pointer of any data type such as int, char, float, we can also create a pointer pointing to a function. The code of a function always resides in memory, which means that the function has some address. We can get the address of memory by using the function pointer.

Declaration of a function pointer

Till now, we have seen that the functions have addresses, so we can create pointers that can contain these addresses, and hence can point them.

Syntax of function pointer

```
return type (*ptr_name)(type1, type2...);
```

For example:

```
int (*ip) (int);
```

pointer

In the above declaration, `*ip` is a pointer that points to a function which returns an `int` value and accepts an integer value as an argument.

float (*fp) (**float**);

In the above declaration, ***fp** is a pointer that points to a function that returns a float value and accepts a float value as an argument.

We can observe that the declaration of a function is similar to the declaration of a function pointer except that the pointer is preceded by a `*`. So, in the above declaration, `fp` is declared as a function rather than a pointer.

Example:

Write a program to sum of two numbers using pointer to function.

Program

```
1 #include<stdio.h>
2 int Sum(int,int);
3 int (*ptr)(int,int);
4 int main()
5 {
6     int a,b,rt;
7     printf("\nEnter 1st number : ");
8     scanf("%d",&a);
9     printf("\nEnter 2nd number : ");
10    scanf("%d",&b);
11    ptr = Sum;
12    rt = (*ptr)(a,b);
13    printf("\nThe sum is : %d",rt);
14    return 0;
15 }
16 int Sum(int x,int y)
17 {
18     return x + y;
19 }
```

Output

```
Enter 1st number : 5
Enter 2nd number : 10
The sum is : 15
```

