C Pointers

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. The size of the pointer depends on the architecture. However, in 32-bit architecture the size of a pointer is 2 byte.

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.
```

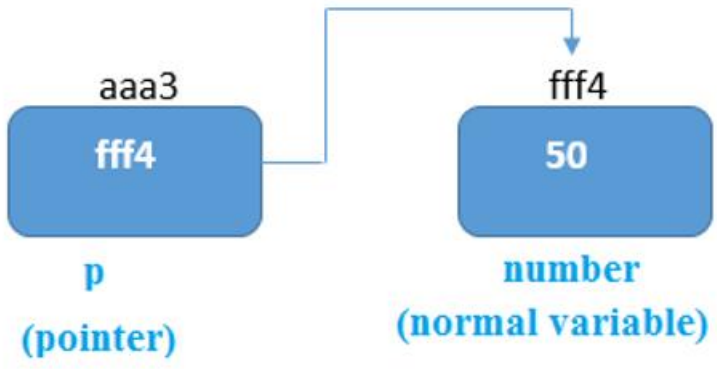
Declaring a pointer

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

```
1. int *a;//pointer to int
2. char *c;//pointer to char
```

Pointer Example

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



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>
int main(){
int number=50;
int *p;
p=&number;//stores the address of number variable
printf("Address of number 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 po
inter therefore if we print *p, we will get the value stored at the address contained by p.
return 0;
}
```

Output

```
Address of number variable is fff4
Value of *p variable is 50
```

Pointer to array

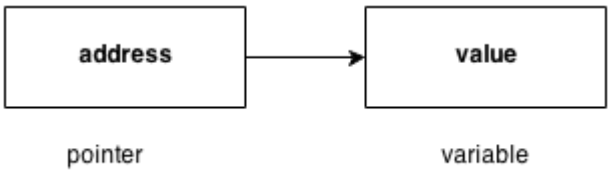
```
int arr[10];
int *p[10]=&arr; // Variable p of type pointer is pointing to the address of an integer array
 arr.
```

Pointer to a function

```
void show (int);
void(*p)(int) = &display; // Pointer p is pointing to the address of a function
```

Pointer to structure

```
struct st {
   int i;
   float f;
}ref;
struct st *p = &ref;
```



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. But, we need to use %u to display the address of a variable.

```
#include<stdio.h>
int main(){
int number=50;
printf("value of number is %d, address of number is %u",number,&number);
return 0;
}
```

Output

```
value of number is 50, address of number is fff4
```

NULL Pointer

A Null Pointer is a pointer that does not point to any address value. 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;
```

In the most libraries, the value of the pointer is 0 (zero).

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

```
#include<stdio.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
```

Reading complex pointers

There are several things which must be taken into the consideration while reading the complex pointers in C. Lets see the precedence and associativity of the operators which are used regarding pointers.

Operator	Precedence	Associativity

(), []	1	Left to right
*, identifier	2	Right to left
Data type	3	-

Here,we must notice that,

- (): This operator is a bracket operator used to declare and define the function.
- []: This operator is an array subscript operator
- * : This operator is a pointer operator.
- Identifier: It is the name of the pointer. The priority will always be assigned to this.
- Data type: Data type is the type of the variable to which the pointer is intended to point. It also includes the modifier like signed int, long, etc).

How to read the pointer: int (*p)[10].

To read the pointer, we must see that () and [] have the equal precedence. Therefore, their associativity must be considered here. The associativity is left to right, so the priority goes to ().

Inside the bracket (), pointer operator * and pointer name (identifier) p have the same precedence. Therefore, their associativity must be considered here which is right to left, so the priority goes to p, and the second priority goes to *.

Assign the 3rd priority to [] since the data type has the last precedence. Therefore the pointer will look like following.

- char -> 4
- * -> 2
- p -> 1
- [10] -> 3

The pointer will be read as p is a pointer to an array of integers of size 10.

Example

How to read the following pointer?

```
int (*p)(int (*)[2], int (*)void))
```

Explanation

This pointer will be read as p is a pointer to such function which accepts the first parameter as the pointer to a one-dimensional array of integers of size two and the second parameter as the pointer to a function which parameter is void and return type is the integer.