POINTERS

Like any variable **Pointer** is a variable like reference that hold a memory to another variable, arrays.

And many tasks are easily done by pointers. Like,

- Always works on the original variable.
- We can easily create any data structures.
- Searching and sorting huge data very easily.
- Returning more than one values in the functions.

Now let's see the basic and simple form of pointers. Suppose, there is an integer variable age and we want to print the memory location of the variable and also the age.

```c
#include <stdio.h>

int main(){


    int age = 21;

    printf("I am %d years old.\n", age);
    printf("Memory address of the variable age is %p", &age); /* here '&' is the address
                                                                 allocation variable*/

}
```

%p is use to display the memory address. The output will be,

```
I am 21 years old.
Memory address of the variable age is 0061FF1C
```

the memory location will give the hexadecimal.

We can also store this '0061FF1C' address in a separate variable like reference. Which is the pointer. Now to make a pointer we have to make sure the pointer variable is same as the variable we are pointing to. If the variable is an integer, then the pointer type will be an integer; if the variable is a character, then the pointer type will be a character.

```c
#include <stdio.h>

int main(){


    int age = 21;
    int *pAge = &age; // here '*' is the indirection operator. or we can say it allows to store
                      // the value of address


    printf("The value of pAge is %p\n ", pAge );
    printf("The memory location of age is %p", &age);

}
```

Here, pAge has its own address but the value store within it is another address and we can store it using the **‘*’ or the indirection operator**.

```
The value of pAge is 0061FF18
 The memory location of age is 0061FF18
```

Here, we print the address stored in pAge using %p.

This is accessing the variable using the pointer variable only.

```c
int *pAge = &age;
```

THIS PART IS THE *POINTER DECLARATION AND POINTER INITIALIZATION.*

Now, suppose we want to access the value at the stored address, we can simple do it using the **‘*’ or the indirection operator**. Simply we want to find the value at the address which can also be called **dereferencing.**

```c
#include <stdio.h>

int main(){


    int age = 21;
    int *pAge = &age; // here '*' is the indirection operator. or we can say it allows to store
                      // the value of address


    printf("My age is %d year.\n", age);
    printf("The value stored at pAge is %d.\n", *pAge); //dereferencing.

}
```

```
My age is 21 year.
The value stored at pAge is 21.
```

There is another way by which we can find the value stored in a memory address without assigning a pointer using '%d' or '%c' or '%f' base on the type of variable. For example.

```c
#include <stdio.h>

int main(){


    int age = 21;
    char cht = 'r';
    float fx = 600.999645;

    printf("%d\n", *(&age));
    printf("%c\n", *(&cht));
    printf("%f\n", *(&fx));

    return 0;
}
```

OUTPUT:

```
21
r
600.999634
```

Since the pointer in C stores the memory address so their size it independent. It depends on the system architecture.

Pointers are used in many other cases. For example.

- to pass the values by reference
- To do system-level programming where memory addresses are useful.
- To access the values of an array easily.
- To return Multiple values from an function.

And many more other uses which we will talk about later.