**Pointers in C Programming Language**

Before diving into Pointers in “C”, let understand this

In a computer, **memory is uniquely identified by an address**, much like a PO Box in a post office or an apartment mailbox. Just as each PO Box has a unique number that helps you locate and retrieve your mail, each memory location in a computer has a **unique address** that helps the processor locate and access stored data.

For instance, when we write int dayOfTheWeek = 1; in a program:

1. The computer **allocates a specific memory space** for the variable dayOfTheWeek.
2. It stores the value 1 in that allocated memory space.
3. This memory space is identified by its **unique address**, just like a PO Box number.

In essence, **a memory space is allocated for each variable**, which holds the value of that variable, and this memory space can be accessed using its unique address—just like retrieving mail from a specific PO Box.

In computer the memory address is denoted in hexadecimal format, like 0x88ffed45ab33

Let’s look into this C program.

*A computer code with text

Description automatically generated*

The result is:

A black background with white text

Description automatically generated

Let’s get the pictorial representation:

A diagram of a memory

Description automatically generated

**Let’s look into Pointers:**

In C, a pointer is a **variable that stores the memory address of another variable**. Let try to understand this through the below example.

Int \*p = NULL; // This is how we declare a pointer of type int. If we are not initializing any value, the best practice is to assign NULL. It indicates that the pointer does not point to any valid memory address.

As p can store/hold address, we can assign p as

p = &dayOfTheWeek; // This mean address of the variable “dayOfTheWeek” is now stored in p.

Let’s get the pictorial representation

A diagram of a memory

Description automatically generated with medium confidence

Let’s look into the below program:

A screenshot of a computer program

Description automatically generated

The result is:

A computer screen with white text

Description automatically generated

Hope you understood the basic of Pointers in “C”

Let's look into another program to understand pointers better.

#include <stdio.h>

int main()

{

int dayOfTheWeek = 1;

int\* p = NULL;

\*p = 5; **// this line you give you an error, why?**

int num = 6;

\*p = num; **// this line you give you an error, why?**

p = &num;

\*p = dayOfTheWeek;

printf("%d", \*p);

\*p = 7;

printf("%d", \*p); **// what it will print ?**

printf("%d", dayOfTheWeek); **// what it will print ?**

printf("%d", \*num); **// what it will print ?**

printf("%d", num); **// what it will print ?**

int\* pNum = 4; **// this line you give you an error, why?**

int\* pInt; **// declaration, not initialized to anything not even NULL, what does it mean?**

\*pInt = 10; **// this line you give you an error, why?**

return 0;

}

Let’s look into another program

#include <stdio.h>

int main()

{

int num = 10;

printf("1 %d\n", num);

int\* pNum = num; //is it allowed?

printf("2 %p\n", pNum);

pNum = 10; //is it allowed?

printf("3 %p\n", pNum);

pNum = &num; //is it allowed?

printf("4 %p\n", pNum);

pNum = num; //is it allowed?

printf("5 %p\n", pNum);

\*pNum = num; //is this allowed?

printf("6 %d\n", \*pNum);

\*pNum = &num; //is it allowed?

printf("7 %d\n", d);

}

In int datatype we can do mathematical operations +, -, /, x.

Can we do the same in int pointers (int\*)?

**Let’s explore character pointers by building on the concepts we learned about integer pointers.**

**Declaration of a Character Pointer:**

A character pointer is declared as: char\* cp;

If we are not assigning any value to it during declaration, it is a good practice to initialize it to NULL to avoid undefined behavior. This would look like: char\* cp = NULL;

**Assigning a Value to a Character Pointer:**

If we have a value to assign, the assignment should be as follows:

char c = 'A';

char\* cp = &c; // '&c' is used because pointers store/hold the memory address of a variable.

**Incorrect Assignment:**

If you assign a value directly without referencing a variable, as shown below:

char c = 'A';

char\* cp = c;

This assignment will compile, but it results in unexpected behavior. Here, the character 'A' is implicitly converted to its ASCII value (65), which is treated as a memory address. Since 65 is not a valid memory address, this leads to undefined behavior, potentially causing your program to crash or produce unpredictable results.