A **pointer** is a variable that stores the **memory address** of another variable.

### Why use pointers?

* Dynamic memory management
* Passing large structures/arrays to functions efficiently
* Arrays and strings handling
* Accessing hardware-level data (e.g., embedded systems)
* For callbacks (pointers to functions)

## Pointer Declaration and De-referencing

Pointer declaration syntax:

int \*p; // declaration

p = &a; // storing address of variable a

De-referencing:

int a = 10;

Int \*ptr = &a;

Printf(“%d”, \*ptr); //out 10

**Problem Statement:Write a program to input a number and print its value and memory address using pointers.**

#include <stdio.h>

int main() {

int num;

int \*ptr;

printf("Enter a number: ");

scanf("%d", &num);

ptr = &num;

printf("Value: %d\n", \*ptr);

printf("Address: %p\n", ptr);

return 0;

}

## Pointer Arithmetic

### You can:

* Increment/decrement pointers
* Add/subtract integers

int arr[] = {10, 20, 30};

int \*p = arr;

printf("%d\n", \*p); // 10

printf("%d\n", \*(p+1)); // 20

### **Problem Statement: W**rite a program to traverse an array using pointers.

#### Sample Code:

#include<stdio.h>

int main(){

int arr[5] = {10, 20, 30, 50, 60};

int \*p = arr;

For (i=0; i<5; i++ ){

printf("%d\n", \*(p+i));}

return 0;

}

## Pointers to Arrays and Strings

### Array Pointer:

int arr[] = {10, 20, 30};

int \*p = arr; //pointer to arr[0] prints - 10

### String Pointer:

Chr \*str= “Hello”;

Printf(“%c”, \*(str)); //String to str[0] prints - H

### Problem Statement:Print each character of a string using a pointer.

#include <stdio.h>

int main() {

char \*str = "Pointer";

while(\*str != '\0') {

printf("%c ", \*str);

str++;

}

return 0;

}

## Double Pointers

A **double pointer** stores the address of another pointer.

### Use:

* Modify pointer values in functions
* Multilevel data structures
* Dynamic memory allocation for 2D arrays
* Managing arrays of strings or array of pointers
* Useful in function callbacks where the function modifies the pointer itself

#include <stdio.h>

void modify(int \*\*p){

\*\*p = 20;

}

int main(){

int a = 10;

int \*ptr = &a;

int \*\*p = &p;

printf(“%d”, a); // print value 10

modify(p);

printf(“%d”, a); // print value 20

return 0;

}

## **✅ Why/When Use Double Pointers?**

### 1. ****To Modify a Pointer Inside a Function****

C passes arguments **by value**, so to change a pointer's target inside a function, use a double pointer.

void allocateMemory(int \*\*ptr) {

\*ptr = (int \*)malloc(sizeof(int));

\*\*ptr = 42;

}

### **2. **2D Dynamic Arrays****

When you don’t know the size at compile time, double pointers help build a 2D matrix.

int \*\*matrix = (int \*\*)malloc(rows \* sizeof(int \*));

for (int i = 0; i < rows; i++) {

matrix[i] = (int \*)malloc(cols \* sizeof(int));

}

### **3. **Array of Strings****

Each string is a char\*, so char\*\* handles multiple strings.

char \*names[] = {"Alice", "Bob", "Charlie"};

char \*\*ptr = names;