Anjuman — I — Islam's

M.H SABOO SIDDIK COLLEGE OF ENGINEERING

8, Saboo Siddik Polytechnic Rd., Byculla, Mumbai — 400 008.

Department of Basic Science and Humanity

Module-05: Cheat Sheet – Pointer

Subject: C Programming **Sem: I**

1. Introduction to Pointers

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

Example

```
int x = 10;
int *p = &x;   // p stores the address of x
```

2. Definition of Pointer

A pointer is defined using the * operator:

data_type *pointer_name;

Example

```
int *p;     // pointer to int
float *q;   // pointer to float
char *ch;   // pointer to char
```

3. Uses of Pointers

- ✓ Dynamic memory allocation
- ✓ Passing large data to functions efficiently
- ✓ Implementing data structures (linked lists, trees)
- ✓ Returning multiple values from functions
- ✓ Array and string manipulation
- ✓ Hardware-level programming

4. Address Operator &

& gives the address of a variable.

Example

```
int x = 5;
printf("%p", &x);   // prints address of x
```

5. Pointer Variables

A pointer variable stores a memory address.

Example

```
int x = 20;
int *p = &x;
printf("%p", p);   // address stored in p
```

6. Dereferencing Pointer (*)

Used to access the value stored at the pointer's address.

Example

```
int x = 30;
int *p = &x;

printf("%d", *p);  // prints 30 (value at p)
```

7. Void Pointer

A pointer that can hold the address of any data type.
But it cannot be dereferenced without casting.
Example

```
void *ptr;
int x = 10;
ptr = &x;

printf("%d", *(int*)ptr);  // cast to int* before dereference
```

8. Pointer Arithmetic

Valid operations on pointers:

✓ p++ (next memory block)
✓ p--
✓ p + n
✓ p - n

Pointer increment depends on data type size.
Example

```
int arr[3] = {1, 2, 3};
int *p = arr;   // points to arr[0]

p++;          // moves to next integer (4 bytes ahead)
printf("%d", *p);  // prints 2
```

9. Pointers to Pointers (**)

A pointer storing the address of another pointer.
Example

```
int x = 50;
int *p = &x;     // pointer to x
int **pp = &p;   // pointer to pointer

printf("%d", **pp);  // prints 50
```

10. Pointers and Arrays

Array name acts like a pointer to the first element.
Example

```
int arr[3] = {10, 20, 30};
int *p = arr;

printf("%d", *p);     // 10
printf("%d", *(p+1)); // 20
```

📌 Quick Summary Table

Concept	Meaning	Example
Pointer	Stores address	int *p
&	Address operator	p = &x
*	Dereference	*p = value
Void Pointer	Generic pointer	void *ptr
Pointer Arithmetic	Move pointer	p++
Pointer to Pointer	Pointer storing pointer address	int **pp
Array + Pointer	Array name = 1st element address	p = arr

Program to add two numbers using pointers

```
**
 * C program to add two number using pointers
 */

#include <stdio.h>

int main()
{
   int num1, num2, sum;
   int *ptr1, *ptr2;

   ptr1 = &num1; // ptr1 stores the address of num1
   ptr2 = &num2; // ptr2 stores the address of num2

   printf("Enter any two numbers: ");
   scanf("%d%d", ptr1, ptr2);

   sum = *ptr1 + *ptr2;

   printf("Sum = %d", sum);

   return 0;
}
```

Program to swap two numbers using call by reference

```
/**
 * C program to swap two number using call by reference
 */

#include <stdio.h>

/* Swap function declaration */
void swap(int * num1, int * num2);

int main()
{
   int num1, num2;

   /* Input numbers */
   printf("Enter two numbers: ");
   scanf("%d%d", &num1, &num2);

   /* Print original values of num1 and num2 */
   printf("Before swapping in main n");
   printf("Value of num1 = %d \n", num1);
   printf("Value of num2 = %d \n\n", num2);

   /* Pass the addresses of num1 and num2 */
   swap(&num1, &num2);

   /* Print the swapped values of num1 and num2 */
```

```
    printf("After swapping in main n");
    printf("Value of num1 = %d \n", num1);
    printf("Value of num2 = %d \n\n", num2);

    return 0;
}


/**
 * Function to swap two numbers
 */
void swap(int * num1, int * num2)
{
    int temp;

    // Copy the value of num1 to some temp variable
    temp = *num1;

    // Copy the value of num2 to num1
    *num1= *num2;

    // Copy the value of num1 stored in temp to num2
    *num2= temp;

    printf("After swapping in swap function n");
    printf("Value of num1 = %d \n", *num1);
    printf("Value of num2 = %d \n\n", *num2);
}
```

```
Enter two numbers: 10 20
Before swapping in main
Value of num1 = 10
Value of num2 = 20

After swapping in swap function
Value of num1 = 20
Value of num2 = 10

After swapping in main
Value of num1 = 20
Value of num2 = 10
```