Examples of Every Kind of Pointer

& → Address of operator

* → value of operator

---

1. int

```c
#include <stdio.h>

int main() {
    int x = 7;
    int *ptr = &x;
    printf("Value of x: %d\n", *ptr);
    printf("Address of x: %p\n", ptr);
    return 0;
}
```

2. char

```c
#include <stdio.h>

int main() {
    char c = 'A';
    char *ptr = &c;

    printf("Character: %c\n", *ptr);
    return 0;
}
```

3.float

```c
#include <stdio.h>

int main() {
    float f = 3.14;
    float *ptr = &f;

    printf("Float: %f\n", *ptr);
    return 0;
}
```

4.long

```c
#include <stdio.h>

int main() {
    long n = 123456789;
    long *ptr = &n;

    printf("Long value: %ld\n", *ptr);
    return 0;
}
```

Pass by Value vs Pass by Reference

Int fun (int a , int b);

a & b are **formal parameters** they are received by the function.

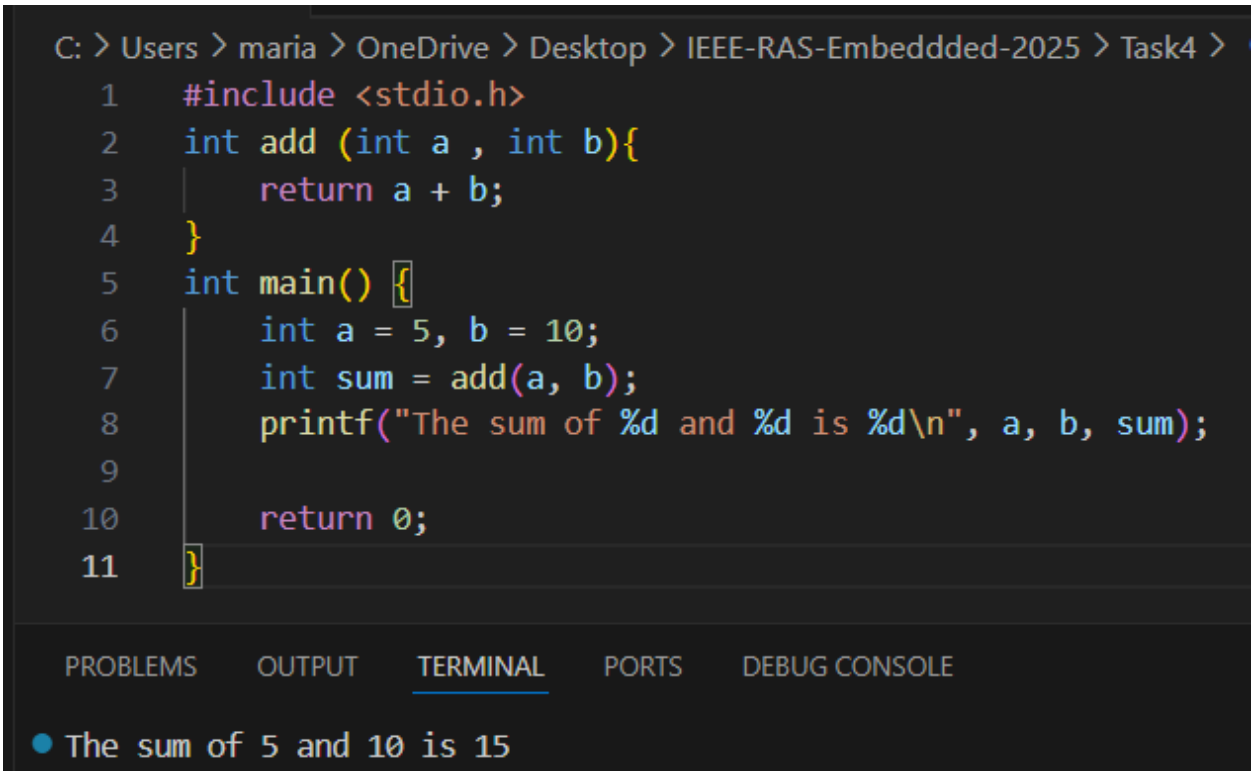
Fun (m , n);

M & n are **actual parameters** they are passed to the function

---

Passing By Value → The actual Parameters are copied to the formal parameters and these two different parameters are stored at different addresses.

(We are passing values to the function)



Passing by reference → The actual parameters and the formals parameters refer to the same memory locations therefore if the formal parameters changed the actual parameters will also change.

(We are passing addresses to the function)

C:\Users\maria\OneDrive\Desktop\IEEE-RAS-Embeddded-2025\Task4\bonus.c Task4 >

```c
#include <stdio.h>
void fun (int *aptr , int *bptr){
    *aptr = 20;
    *bptr = 10;
}
int main() {
    int a = 5, b = 10;
    printf("a: %d\nb: %d \n", a, b);
    fun(&a, &b);
    printf("a: %d\nb: %d \n", a, b);
    return 0;
}
```

PROBLEMS OUTPUT TERMINAL PORTS DEBUG CONSOLE

```
a: 5
b: 10
a: 20
b: 10
```