

# Task 1G: Function Pointer Usage in C

## Objective:

To understand and demonstrate the use of function pointers in C. This task defines multiple arithmetic functions and passes them as arguments to another function using a function pointer to ensure the correct function is invoked at runtime.

## 1. Program Code

```
#include <stdio.h>

int add(int a, int b) {
    return a + b;
}

int subtract(int a, int b) {
    return a - b;
}

int multiply(int a, int b) {
    return a * b;
}

int divide(int a, int b) {
    return a / b;
}

void operation(int (*func)(int, int), int x, int y) {
    int result = func(x, y);
    printf("Result = %d\n", result);
}

int main() {
    operation(add, 10, 5);
    operation(subtract, 10, 5);
    operation(multiply, 10, 5);
    operation(divide, 10, 5);
    return 0;
}
```

## 2. Compilation Instructions

Compile the program using GCC:

```
gcc -g function_pointer.c -o function_pointer
```

```
student@student-virtual-machine:/25SUB4508_LSP/25SUB4508_56133/ClassWork/day12$ cd Task_1G
student@student-virtual-machine:~/25SUB4508_LSP/25SUB4508_56133/ClassWork/day12/Task_1G$ ll
total 12
drwxrwxr-x 2 student student 4096 Jan  1 02:38 .
drwxrwxr-x 7 student student 4096 Jan  1 02:38 ../
-rw-rw-r-- 1 student student 486 Jan  1 02:38 function_pointer.c
student@student-virtual-machine:/25SUB4508_LSP/25SUB4508_56133/ClassWork/day12/Task_1G$ gcc function_pointer.c -o function_pointer
student@student-virtual-machine:~/25SUB4508_LSP/25SUB4508_56133/ClassWork/day12/Task_1G$ ./function_pointer
Result = 15
Result = 5
Result = 50
Result = 2
student@student-virtual-machine:/25SUB4508_LSP/25SUB4508_56133/ClassWork/day12/Task_1G$ █
```

## 3. Program Output

```
Result = 15
Result = 5
Result = 50
Result = 2
```

## 4. Explanation

1. A function pointer stores the address of a function.
2. The operation() function accepts a function pointer and two integers.
3. Different arithmetic functions are passed to operation().
4. The function pointer invokes the correct function dynamically.

## 5. Conclusion

This task demonstrates how function pointers enable dynamic function calls in C. They are widely used in callbacks, event handlers, and modular programming.