

DATE:

LAB REPORT NO: 9 SET: B

TITLE OF THE PROGRAM: **Pointer Operations In C**

OBJECTIVES

- To understand pointer operations in C.
- To learn how to manipulate variables using pointers.
- To practice arithmetic operations on pointers.
- To demonstrate the assignment of pointer values.

REQUIREMENTS

1. C Compiler (e.g., GCC)
2. Computer System
3. IDE or Text Editor
4. OS compatible with the software

THEORY

Pointers in C are variables that store the memory address of another variable. Pointer operations allow efficient manipulation and accessing of memory. Common operations include increment, decrement, addition, and subtraction of pointers, as well as assignment between pointers.

PROCEDURE (Program Code, Comment, and Output)

1. Changing the Value of a Variable Using a Pointer

```
#include <stdio.h>

int main() {
    int a;
    a = 10;

    int *p = &a; // declaring and initializing the pointer

    // prints the value of 'a'
    printf("%d\n", a);
```

```

    *p = 34; // change the value stored in variable a

    // prints the value of 'a'


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

    return 0;

}

```

OUTPUT

 Copy code

```

10
34

```

Explanation

- This program demonstrates how to change the value of a variable using a pointer. The pointer p is used to modify the value of a.

2. Increment (++) and Decrement (--) Operation

```

#include <stdio.h>

int main() {
    // Initializing integer variable
    int a = 34;
    // Declaring pointer variable
    int* ptr_a;
    // Initializing pointer variable
    ptr_a = &a;
    // Value of a before increment
    printf("Increment:\n");
    printf("Before increment a = %d\n", *ptr_a);
    // Unary increment operation
    (*ptr_a)++; // Increases the value of a by 1
    // Value of a after increment
    printf("After increment a = %d", *ptr_a);
}

```

```

// Value before decrement
printf("\n\nDecrement:\n");

printf("Before decrement a = %d\n", *ptr_a);

// Unary decrement operation
(*ptr_a)--; // Decreases the value of a by 1
// Value after decrement
printf("After decrement a = %d", *ptr_a);
return 0;
}

```

OUTPUT

```

Increment:
Before increment a = 34
After increment a = 35

Decrement:
Before decrement a = 35
After decrement a = 34

```

Explanation

This program illustrates the use of increment and decrement operations on a variable through a pointer.

3. Addition of an Integer to a Pointer Variable

```

#include <stdio.h>

int main() {
    int a = 10;
    int *b = &a; // declaring and initializing the pointer
    printf("Memory address before addition of an integer to pointer = %u",
b);

    b = b + 4; // addition of an integer to a pointer variable
    printf("\nAfter adding 4 to the memory address of a");
}

```

```
printf("\nMemory address after addition of an integer to pointer = %u",
b);
return 0;
}
```

Output

```
Memory address before addition of an integer to pointer = 6684180
After adding 4 to the memory address of a
Memory address after addition of an integer to pointer = 6684196
```

Explanation

This program shows how to add an integer value to a pointer, demonstrating pointer arithmetic.

4. Subtraction of an Integer from a Pointer Variable

```
#include <stdio.h>
```

```
int main() {
    int a = 10;
    int *b = &a; // declaring and initializing the pointer
    printf("Memory address before subtraction of an integer to pointer =
%u", b);
    b = b - 4; // subtraction of an integer from a pointer variable
    printf("\nAfter subtracting 4 from the memory address of a");
    printf("\nMemory address after subtraction of an integer to pointer =
%u", b);
    return 0;
}
```

OUTPUT

```
Memory address before subtraction of an integer to pointer = 6684180
After subtracting 4 from the memory address of a
Memory address after subtraction of an integer to pointer = 6684164
```

5. Assigning the Value of One Pointer to Another Pointer

```
#include <stdio.h>
```

```
int main() {  
    int a;  
    a = 10;  
    int *ptr1, *ptr2;  
    ptr1 = &a;  
    ptr2 = ptr1;  
    printf("Values at ptr1 and ptr2 %d %d\n", *ptr1, *ptr2);  
    printf("Address pointed to by ptr1 and ptr2: %u %u", ptr1, ptr2);  
    return 0;  
}
```

OUTPUT

```
Values at ptr1 and ptr2 10 10  
Address pointed to by ptr1 and ptr2: 6684172 6684172
```

Explanation:

This program assigns the value of one pointer to another pointer and prints both the values and addresses.

CONCLUSION

Through these programs, we have explored various pointer operations in C. We learned how to change variable values, perform arithmetic operations on pointers, and assign pointer values. Understanding these operations is crucial for efficient memory management and manipulation in C programming.