Name Priyadarshi Prabhakar SAP ID 590029237

EXPERIMENT 8 : POINTERS

Activity 1: *Declare different types of pointers (int, float, char) and initialize them with the addresses of variables. Print the values of both the pointers and the variables they point to.*

ALGORITHM:

STEP 1: START
STEP 2: Declare variables of types **int**, **float**, and **char**.
STEP 3: Declare pointers of types **int***, **float***, and **char***
STEP 4: Initialize each pointer with the address of its corresponding variable.
STEP 5: Print the value of each variable.
STEP 6: Print the address stored in each pointer.
STEP 7: Print the value pointed to by each pointer.
STEP 8: END

PSEUDOCODE :

START

DECLARE integer a

DECLARE float b

DECLARE character c

SET a = 10

SET b = 3.14

SET c = 'X'

DECLARE integer pointer p1 = address of a

DECLARE float pointer p2 = address of b

DECLARE character pointer p3 = address of c

PRINT "Value of a:", a

PRINT "Value of b:", b

PRINT "Value of c:", c

PRINT "Address stored in p1:", p1

PRINT "Address stored in p2:", p2

PRINT "Address stored in p3:", p3

PRINT "Value pointed by p1:", *p1

PRINT "Value pointed by p2:", *p2

PRINT "Value pointed by p3:", *p3

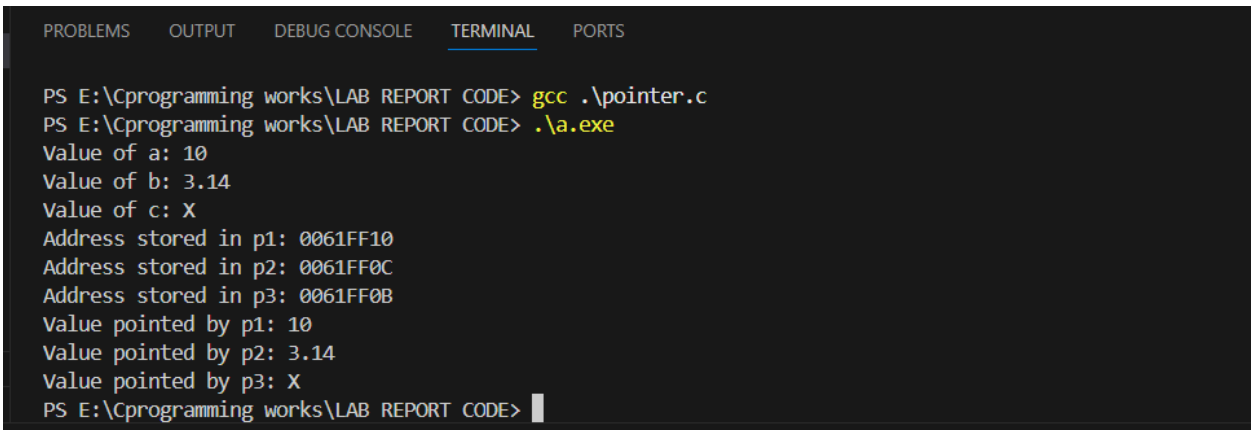
END

CODE :

```
#include <stdio.h>

int main() {

int a = 10;

float b = 3.14;

char c = 'X';

int *p1 = &a;
```

```
float *p2 = &b;

char *p3 = &c;

// Printing variables

printf("Value of a: %d\n", a);

printf("Value of b: %.2f\n", b);

printf("Value of c: %c\n", c);

// Printing pointer values (addresses)

printf("Address stored in p1: %p\n", p1);

printf("Address stored in p2: %p\n", p2);

printf("Address stored in p3: %p\n", p3);

// Printing values using pointers

printf("Value pointed by p1: %d\n", *p1);

printf("Value pointed by p2: %.2f\n", *p2);

printf("Value pointed by p3: %c\n", *p3);

return 0;

}
```

OUTPUT :



Activity 2 : *Perform pointer arithmetic (increment and decrement) on pointers of different data types. Observe how the memory addresses change and the effects on data access.*

ALGORITHM:

STEP 1: Start
STEP 2: Declare variables **a, b, c** of data types: **int, float, char**
STEP 3: Declare pointers for each variable and store their addresses.
STEP 4: Print the initial pointer addresses.
STEP 5: Increment each pointer (**p++**).
STEP 6: Print the new addresses after increment.
STEP 7: Decrement each pointer (**p--**).
STEP 8: Print the addresses again after decrement.
STEP 9: End

PSEUDOCODE :

START

DECLARE int a

DECLARE float b

DECLARE char c

*DECLARE int *p1 = &a*

*DECLARE float *p2 = &b*

*DECLARE char *p3 = &c*

PRINT "Initial address in p1:", p1

PRINT "Initial address in p2:", p2

PRINT "Initial address in p3:", p3

p1++

p2++

p3++

PRINT "Address after increment p1:", p1

PRINT "Address after increment p2:", p2

PRINT "Address after increment p3:", p3

p1 --

p2 --

p3 --

PRINT "Address after decrement p1:", p1

PRINT "Address after decrement p2:", p2

PRINT "Address after decrement p3:", p3

END

CODE :

```
#include <stdio.h>
int main() {
int a = 10;
float b = 3.14;
char c = 'X';
int *p1 = &a;
float *p2 = &b;
```

```
char *p3 = &c;
printf("Initial Addresses:\n");
printf("p1 = %p\n", p1);
printf("p2 = %p\n", p2);
printf("p3 = %p\n", p3);
p1++;
p2++;
p3++;
printf("\nAfter Increment:\n");
printf("p1 = %p\n", p1);
printf("p2 = %p\n", p2);
printf("p3 = %p\n", p3);
p1--;
p2--;
p3--;
printf("\nAfter Decrement:\n");
printf("p1 = %p\n", p1);
printf("p2 = %p\n", p2);
printf("p3 = %p\n", p3);
return 0;
}
```

OUTPUT :

```
PS E:\Cprogramming works\LAB REPORT CODE> gcc .\increment.c
PS E:\Cprogramming works\LAB REPORT CODE> .\a.exe
Initial Addresses:
p1 = 0061FF10
p2 = 0061FF0C
p3 = 0061FF0B

After Increment:
p1 = 0061FF14
p2 = 0061FF10
p3 = 0061FF0C

After Decrement:
p1 = 0061FF10
p2 = 0061FF0C
p3 = 0061FF0B
PS E:\Cprogramming works\LAB REPORT CODE>
```

Activity 3 : *Write a function that accepts pointers as parameters. Pass variables by reference using pointers and modify their values within the function.*

ALGORITHM:

STEP 1: START

STEP 2: Declare two integer variables **a** & **b**.

STEP 3: Read input values for **a** and **b** from the user.

STEP 4: Call the function modifyValues(**&a**, **&b**) and pass the addresses of **a** & **b**.

STEP 5: Inside **modifyValues**, access the variables using pointers.

STEP 6: Modify the values.

STEP 7: Return to the main program after modification.

STEP 8: Display the updated values of **a** & **b** in the main function.

STEP 9: END

PSEUDOCODE :

START

DECLARE int a

DECLARE int b

READ a

READ b

DECLARE function modifyValues(int *p1, int *p2)

CALL modifyValues(&a, &b)

PRINT "Updated value of a:", a

PRINT "Updated value of b:", b

END

FUNCTION modifyValues(int *p1, int *p2)

****p1 = *p1 + 10***

****p2 = *p2 + 10***

END FUNCTION

CODE :

```
#include <stdio.h>
void modifyValues(int *x, int *y) {
*x = *x + 10; // modify a
*y = *y + 20; // modify b
```

```
}
int main() {
int a, b;
printf("Enter value of a: ");
scanf("%d", &a);
printf("Enter value of b: ");
scanf("%d", &b);
modifyValues(&a, &b);
printf("\nAfter modification:\n");
printf("a = %d\n", a);
printf("b = %d\n", b);
return 0;
}
```

OUTPUT :

```
PS E:\Cprogramming works\LAB REPORT CODE> gcc .\modify.c
PS E:\Cprogramming works\LAB REPORT CODE> .\a.exe
Enter value of a: 5
Enter value of b: 2

After modification:
a = 15
b = 22
PS E:\Cprogramming works\LAB REPORT CODE>
```