

EXPERIMENT : 8POINTERS

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

```
#include <stdio.h>
int main () {
    int a = 10 ;
    float b = 25.5 ;
    char c = 'A' ;

    int *ptr1 = &a ;
    float *ptr2 = &b ;
    char *ptr3 = &c ;
```

```
printf ("values of variables :\n");
printf ("a = %d\n", a);
printf ("b = %.2f\n", b);
printf ("c = %c\n", c);
```

```
printf ("Values using pointers :\n");
printf ("*ptr1 = %d\n", *ptr1);
printf ("*ptr2 = %.2f\n", *ptr2);
```

```
printf (" * ptr 3 = %c\n\n", * ptr 3);
```

```
printf (" Addresses stored in pointers : \n");
```

```
printf (" * ptr 1 (address of a) = %p\n", ptr 1);
```

```
printf (" * ptr 2 (address of b) = %p\n", ptr 2);
```

```
printf (" * ptr 3 (address of c) = %p\n", ptr 3);
```

```
printf (" Addresses of the pointers themselves : \n");
```

```
printf (" & ptr 1 = %p\n", (void *) & ptr 1);
```

```
printf (" & ptr 2 = %p\n", (void *) & ptr 2);
```

```
printf (" & ptr 3 = %p\n", (void *) & ptr 3);
```

```
return 0;
```

```
}
```

Output :- Values of variables :

$a = 10$, $b = 25.50$, $c = A$

Values using pointers :

$*ptr 1 = 10$, $*ptr 2 = 25.50$, $*ptr 3 = A$

Addresses stored in pointers :

$ptr 1 (address of a) = 0x7ffe13e9b7fc$

$ptr 2 (address of b) = 0x7ffe13e9b7f8$

$ptr 3 (address of c) = 0x7ffe13e9b7f7$

Addresses of the pointers themselves :

$\&ptr 1 = 0x7ffe13e9b7e8$

$\&ptr 2 = 0x7ffe13e9b7e0$

$\&ptr 3 = 0x7ffe13e9b7d8$



```
1 #include <stdio.h>
2 int main() {
3     int a = 10;
4     float b = 25.5;
5     char c = 'A';
6     int *ptr1 = &a;
7     float *ptr2 = &b;
8     char *ptr3 = &c;
9
10    printf("Values of variables:\n");
11    printf("a = %d\n", a);
12    printf("b = %.2f\n", b);
13    printf("c = %c\n\n", c);
14
15    printf("Values using pointers:\n");
16    printf("*ptr1 = %d\n", *ptr1);
17    printf("*ptr2 = %.2f\n", *ptr2);
18    printf("*ptr3 = %c\n\n", *ptr3);
19
20    printf("Addresses stored in pointers:\n");
21    printf("ptr1 (address of a) = %p\n", ptr1);
22    printf("ptr2 (address of b) = %p\n", ptr2);
23    printf("ptr3 (address of c) = %p\n\n", ptr3);
24
25    printf("Addresses of the pointers themselves:\n");
26    printf("&ptr1 = %p\n", (void*)&ptr1);
27    printf("&ptr2 = %p\n", (void*)&ptr2);
28    printf("&ptr3 = %p\n", (void*)&ptr3);
29
30 }
```

Values of variables:

a = 10
b = 25.50
c = A

Values using pointers:

*ptr1 = 10
*ptr2 = 25.50
*ptr3 = A

Addresses stored in pointers:

ptr1 (address of a) = 0x7ffe13e9b7fc
ptr2 (address of b) = 0x7ffe13e9b7f8
ptr3 (address of c) = 0x7ffe13e9b7f7

Addresses of the pointers themselves:

&ptr1 = 0x7ffe13e9b7e8
&ptr2 = 0x7ffe13e9b7e0
&ptr3 = 0x7ffe13e9b7d8

== Code Execution Successful ==



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

```
#include <stdio.h>
int main () {
    int a[3] = {10, 20, 30};
    float b[3] = {1.1, 2.2, 3.3};
    char c[3] = {'A', 'B', 'C'};

    int *pInt = a;
    float *pFloat = b;
    char *pChar = c;

    printf ("Initial addresses and values: \n");
    printf ("pInt = %p, *pInt = %d\n", (void *) pInt, *pInt);
    printf ("pFloat = %p, *pFloat = %.1f\n", (void *) pFloat, *pFloat);
    printf ("pChar = %p, *pChar = %c\n", (void *) pChar, *pChar);

    pInt++;
    pFloat++;
    pChar++;

    printf ("After incrementing pointers: \n");
```

Experiment No. _____ Name: _____

```
printf ("pInt = %p pInt = %d\n", (void *)pInt,
       *pInt);
```

```
printf ("pFloat = %p pFloat = %.1f\n", (void *)pFloat,
        *pFloat);
```

```
printf ("pChar = %p pChar = %c\n\n", (void *)pChar,
       *pChar);
```

pInt --;

pFloat --;

pChar --;

```
printf ("after decrementing pointers (back to
original positions) : \n");
```

```
printf ("pInt = %p, *pInt = %d\n", (void *)
       pInt, *pInt);
```

```
printf ("pFloat = %p, *pFloat = %.1f\n", (void *)
       pFloat, *pFloat);
```

```
printf ("pChar = %p, *pChar = %c\n\n", (void *)
       pChar, *pChar);
```

return 0;

}

Output :- Initial addresses and values :

pInt = 0x7ffdfc9fe23c , *pInt = 10

pFloat = 0x7ffdfc9fe230 , *pFloat = 1.1

pChar = 0x7ffdfc9fe22d , *pChar = A

After incrementing pointers :

pInt = 0x7ffdfc9fe240 , *pInt = 20

pFloat = 0x7ffdfc9fe234 , *pFloat = 2.2

pChar = 0x7ffdfc9fe22e , *pChar = B

After decrementing pointers (back to original positions) :

pInt = 0x7ffdfc9fe23c , *pInt = 10

pFloat = 0x7ffdfc9fe230 , *pFloat = 1.1

pChar = 0x7ffdfc9fe22d , *pChar = A



```
1 #include <stdio.h>
2 int main() {
3     int a[3] = {10, 20, 30};
4     float b[3] = {1.1, 2.2, 3.3};
5     char c[3] = {'A', 'B', 'C'};
6
7     int *pInt = a;
8     float *pFloat = b;
9     char *pChar = c;
10
11    printf("Initial addresses and values:\n");
12    printf("pInt = %p, *pInt = %d\n", (void*)pInt, *pInt);
13    printf("pFloat = %p, *pFloat = %.1f\n", (void*)pFloat, *pFloat);
14    printf("pChar = %p, *pChar = %c\n\n", (void*)pChar, *pChar);
15    pInt++;
16    pFloat++;
17    pChar++;
18
19    printf("After incrementing pointers:\n");
20    printf("pInt = %p, *pInt = %d\n", (void*)pInt, *pInt);
21    printf("pFloat = %p, *pFloat = %.1f\n", (void*)pFloat, *pFloat);
22    printf("pChar = %p, *pChar = %c\n\n", (void*)pChar, *pChar);
23    pInt--;
24    pFloat--;
25    pChar--;
26
27    printf("After decrementing pointers (back to original positions):\n");
28    printf("pInt = %p, *pInt = %d\n", (void*)pInt, *pInt);
29    printf("pFloat = %p, *pFloat = %.1f\n", (void*)pFloat, *pFloat);
30    printf("pChar = %p, *pChar = %c\n", (void*)pChar, *pChar);
31    return 0;
32 }
```

Initial addresses and values:
pInt = 0x7ffd9fc9fe23c, *pInt = 10
pFloat = 0x7ffd9fc9fe230, *pFloat = 1.1
pChar = 0x7ffd9fc9fe22d, *pChar = A

After incrementing pointers:
pInt = 0x7ffd9fc9fe240, *pInt = 20
pFloat = 0x7ffd9fc9fe234, *pFloat = 2.2
pChar = 0x7ffd9fc9fe22e, *pChar = B

After decrementing pointers (back to original positions):
pInt = 0x7ffd9fc9fe23c, *pInt = 10
pFloat = 0x7ffd9fc9fe230, *pFloat = 1.1
pChar = 0x7ffd9fc9fe22d, *pChar = A

==== Code Execution Successful ===

3). Write a function that accept pointers as parameters . Pass variables by reference using pointers and modify their values within the function

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

```
void modifyValues (int *x, float *y, char *z) {
```

```
*x = *x + 10;
```

```
*y = *y + 2;
```

```
*z = *z + 1;
```

```
}
```

```
int main () {
```

```
int a = 5;
```

```
float b = 3.5;
```

```
char c = 'A';
```

```
printf ("Before modification : \n");
```

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

```
printf ("b = %.2f\n", b);
```

```
printf ("c = %c\n", c);
```

```
modifyValues (&a, &b, &c);
```

```
printf ("After modification (inside function) : \n");
```

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

```
printf ("b = %.2f\n", b);
```

```
printf ("c = %c\n", c);
```

Experiment No. _____ Name: _____

PAGE NO.:

DATE : / /

return 0;
3

Output :- Before modification :

$$a = 5$$

$$b = 3.50$$

$$c = A$$

After modification (inside function) :

$$a = 15$$

$$b = 7.00$$

$$c = B$$

Teacher's Signature : _____

