Experiment 1: Basic Declaration of Pointer

Problem Statement

Write a C program to demonstrate the basic declaration and usage of pointers.

Objective

To understand and implement pointer operations in C programming.

Algorithm

1. Start the program.
2. Declare variables and pointers as required.
3. Initialize or take user input.
4. Perform operations using pointers.
5. Display results.
6. Stop the program.

Program Code

```
#include <stdio.h>
int main(void)
{
    int m = 10, n, o;
    int *z = &m;
    printf(" Here is m=10, n and o are two integer variable and *z is an
integer");
    printf("\n\n z stores the address of m  = %p\n", z);
    printf("\n *z stores the value of m = %i\n", *z);
    printf("\n &m is the address of m = %p\n", &m);
    printf("\n &n stores the address of n = %p\n", &n);
    printf("\n &o  stores the address of o = %p\n", &o);
    printf("\n &z stores the address of z = %p\n\n", &z);
}
```

Sample Output

Output:
Displays address and values of variables using pointer operations.

Experiment 2: Addition of Two Numbers Using Pointers

Problem Statement

Write a C program to add two numbers using pointers.

Objective

To understand and implement pointer operations in C programming.

Algorithm

1. Start the program.
2. Declare variables and pointers as required.
3. Initialize or take user input.
4. Perform operations using pointers.
5. Display results.
6. Stop the program.

Program Code

```c
#include <stdio.h>
int main() {
    int fno, sno, *ptr, *qtr, sum;
    printf(" Input the first number : ");
    scanf("%d", &fno);
    printf(" Input the second number : ");
    scanf("%d", &sno);
    ptr = &fno;
    qtr = &sno;
    sum = *ptr + *qtr;
    printf(" The sum of the entered numbers is : %d\n\n", sum);
    return 0;
}
```

Sample Output

Input: 5, 7

Output: The sum of the entered numbers is : 12

Experiment 3: Maximum of Two Numbers Using Pointers

Problem Statement

Write a C program to find the maximum of two numbers using pointers.

Objective

To understand and implement pointer operations in C programming.

Algorithm

1. Start the program.
2. Declare variables and pointers as required.
3. Initialize or take user input.
4. Perform operations using pointers.
5. Display results.
6. Stop the program.

Program Code

```c
#include <stdio.h>
#include <stdlib.h>
int main() {
    int fno, sno, *ptr1 = &fno, *ptr2 = &sno;
    printf(" Input the first number : ");
    scanf("%d", ptr1);
    printf(" Input the second number : ");
    scanf("%d", ptr2);
    if (*ptr1 > *ptr2) {
        printf("\n\n %d is the maximum number.\n\n", *ptr1);
    } else {
        printf("\n\n %d is the maximum number.\n\n", *ptr2);
    }
    return 0;
}
```

Sample Output

Input: 12, 18

Output: 18 is the maximum number.

Experiment 4: Factorial Using Pointers

Problem Statement

Write a C program to find the factorial of a number using pointers.

Objective

To understand and implement pointer operations in C programming.

Algorithm

1. Start the program.
2. Declare variables and pointers as required.
3. Initialize or take user input.
4. Perform operations using pointers.
5. Display results.
6. Stop the program.

Program Code

```
#include <stdio.h>
void findFact(int, int*);
int main() {
    int fact;
    int num1;
    printf("\n\n Pointer : Find the factorial of a given number :\n");
    printf("-------------------------------------------------------\n");
    printf(" Input a number : ");
    scanf("%d", &num1);
    findFact(num1, &fact);
    printf(" The Factorial of %d is : %d \n\n", num1, fact);
    return 0;
}
void findFact(int n, int *f) {
    int i;
    *f = 1;
    for (i = 1; i <= n; i++) {
        *f = *f * i;
    }
}
```

Sample Output

Input: 5
Output: The Factorial of 5 is : 120

Experiment 5: Array Using Pointers

Problem Statement

Write a C program to access array elements using pointers.

Objective

To understand and implement pointer operations in C programming.

Algorithm

1. Start the program.
2. Declare variables and pointers as required.
3. Initialize or take user input.
4. Perform operations using pointers.
5. Display results.
6. Stop the program.

Program Code

```
#include <stdio.h>
int main() {
  int myNumbers[4] = {25, 50, 75, 100};
  int *ptr = myNumbers;
  int i;
  for (i = 0; i < 4; i++) {
    printf("%d\n", *(ptr + i));
  }
  return 0;
}
```

Sample Output

Output:
25
50
75
100

Experiment 6: Addition of Array Elements Using Pointers

Problem Statement

Write a C program to calculate the sum of all elements in an array using pointers.

Objective

To understand and implement pointer operations in C programming.

Algorithm

1. Start the program.
2. Declare variables and pointers as required.
3. Initialize or take user input.
4. Perform operations using pointers.
5. Display results.
6. Stop the program.

Program Code

```
#include <stdio.h>
int main() {
    int arr1[10];
    int i, n, sum = 0;
    int *pt;
    printf("\n\n Pointer : Sum of all elements in an array :\n");
    printf("-------------------------------------------------\n");
    printf(" Input the number of elements to store in the array (max 10) : ");
    scanf("%d", &n);
    printf(" Input %d number of elements in the array : \n", n);
    for (i = 0; i < n; i++) {
        printf(" element - %d : ", i + 1);
        scanf("%d", &arr1[i]);
    }
    pt = arr1;
    for (i = 0; i < n; i++) {
        sum = sum + *pt;
        pt++;
    }
    printf(" The sum of array is : %d\n\n", sum);
}
```

Sample Output

Input: 4 elements [1, 2, 3, 4]
Output: The sum of array is : 10

Experiment 7: Call by Value (Swap)

Problem Statement

Write a C program to demonstrate call by value using swap function.

Objective

To understand and implement pointer operations in C programming.

Algorithm

1. Start the program.
2. Declare variables and pointers as required.
3. Initialize or take user input.
4. Perform operations using pointers.
5. Display results.
6. Stop the program.

Program Code

```
#include <stdio.h>
void swapx(int x, int y) {
    int t;
    t = x;
    x = y;
    y = t;
    printf("Inside swapx: x = %d y = %d\n", x, y);
}
int main() {
    int a = 10, b = 20;
    swapx(a, b);
    printf("Inside main: a = %d b = %d", a, b);
    return 0;
}
```

Sample Output

Output:
Inside swapx: x = 20 y = 10
Inside main: a = 10 b = 20

Experiment 8: Call by Reference (Swap)

Problem Statement

Write a C program to demonstrate call by reference using swap function.

Objective

To understand and implement pointer operations in C programming.

Algorithm

1. Start the program.
2. Declare variables and pointers as required.
3. Initialize or take user input.
4. Perform operations using pointers.
5. Display results.
6. Stop the program.

Program Code

```
#include <stdio.h>
void swapx(int* x, int* y) {
    int t;
    t = *x;
    *x = *y;
    *y = t;
    printf("Inside swapx: x = %d y = %d\n", *x, *y);
}
int main() {
    int a = 10, b = 20;
    swapx(&a, &b);
    printf("Inside main: a = %d b = %d", a, b);
    return 0;
}
```

Sample Output

Output:
Inside swapx: x = 20 y = 10
Inside main: a = 20 b = 10