Day : Pointers (9-8-2025)

1. Write a program to print the address of a variable using pointer.

   IPO

- **Input**: Integer value.
- **Process**: Store its address in a pointer, print it.
- **Output**: Address of the variable.

Program

```
#include <stdio.h>
int main()
{
   int a;
   int *p;
   scanf("%d", &a);
   p = &a;
   printf("Address of a = %p\n", p);
   return 0;
}
```

Output

```
Output
25
Address of a = 0x7fff76fab1c4
```

2. Write a program to access array elements using pointers.

   IPO

- **Input**: Elements of an array.
- **Process**: Use pointer to display elements.
- **Output**: Elements printed.

Program

```
#include <stdio.h>

int main()

{

   int arr[5], i;

   int *p = arr;

  for(i = 0; i < 5; i++)

      scanf("%d", p + i);

   printf("Array elements:\n");

   for(i = 0; i < 5; i++)

      printf("%d ", *(p + i));

   return 0;

}
```

Output

```
Array elements:
12 12 12 34 56
```

3.Write a program to swap two numbers using pointers.

IPO

- Input: Two numbers.
- Process: Swap using pointer variables.
- Output: Swapped numbers.

Program

```
#include <stdio.h>
int main()
 {
   int a, b, temp;
   int *p1 = &a, *p2 = &b;
   printf("Enter two numbers: ");
   scanf("%d %d", p1, p2);
   temp = *p1;
   *p1 = *p2;
   *p2 = temp;
   printf("After swap: a=%d b=%d\n", a, b);
   return 0;
}
```

Output

```
5
10
After swap: a=10 b=5
```

4.Write a program to add two numbers using pointers.

IPO

- Input: Two integers.
- Process: Add values using pointers.
- Output: Sum.

Program

```
#include <stdio.h>
int main() {
   int a, b, sum;
   int *p1 = &a, *p2 = &b;
   printf("Enter two numbers: ");
   scanf("%d %d", p1, p2);
   sum = *p1 + *p2;
   printf("Sum = %d\n", sum);
   return 0;
}
```

Output

```
Output
Enter two numbers: 3
5
Sum = 8
```

5.Write a program to find the length of a string using pointers.

IPO

- **Input**: String.
- **Process**: Move pointer till '\0' to count length.
- **Output**: Length of string.

Program

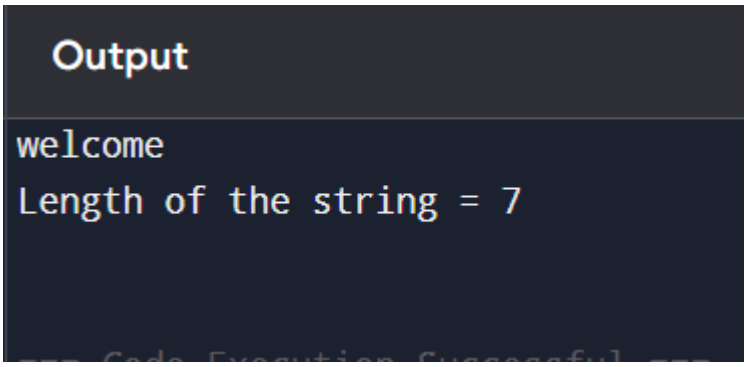
```
#include <stdio.h>

int main()
```

```
{
    char str[100];
    char *p;
    int length = 0;
    scanf("%[^\n]", str);
    p = str;
    while (*p != '\0')
    {
        length++;
        p++;
    }
    printf("Length of the string = %d\n", length);

    return 0;
}
```

Output



6.Write a program to reverse a string using pointers.

IPO

- Input: String.
- Process: Use two pointers to reverse characters.
- Output: Reversed string

Program

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

```
int main() {
    char str[50], *p, *q, temp;
    printf("Enter a string: ");
    fgets(str, sizeof(str), stdin);

    // Remove newline if fgets stored it
    str[strcspn(str, "\n")] = '\0';

    p = str;
    q = str + strlen(str) - 1;

    while (p < q) {
        temp = *p;
        *p = *q;
        *q = temp;
        p++;
        q--;
    }

    printf("Reversed string: %s\n", str);
    return 0;
}
```

Output

```
Output
Enter a string: welcome
Reversed string: emoclew
```

7,Write a program to count vowels using pointer.

IPO

- **Input**: String.
- **Process**: Traverse string via pointer, count vowels.
- **Output**: Vowel count.

Program

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

int main()
 {
   char str[100], *p;
   int count = 0;

   printf("Enter a string: ");
   fgets(str, sizeof(str), stdin);

   str[strcspn(str, "\n")] = '\0';

   p = str;
   while (*p) {
      char ch = tolower(*p);
      if (ch == 'a' || ch == 'e' || ch == 'i' || ch == 'o' || ch == 'u')
         count++;
      p++;
   }

   printf("Total vowels: %d\n", count);
   return 0;
}
```

Output

```
Output
Enter a string: programming in c
Total vowels: 4
```

8.Write a program to demonstrate pointer to pointer.

IPO

- **Input**: Integer value.
- **Process**: Use pointer to pointer to access value.
- **Output**: Value displayed.

Program

```
#include <stdio.h>
int main()
{
   int a = 10;
   int *p = &a;
   int **pp = &p;
   printf("Value of a = %d\n", **pp);
   return 0;
}
```

Output

```
Output
Value of a = 10
```

9.Write a program to allocate memory using malloc() and free it.

IPO

- **Input**: Size and values.
- **Process**: Allocate memory dynamically, store values, free memory.
- **Output**: Values printed.

Program

```
#include <stdio.h>

#include <stdlib.h>

int main()

{
```

```
    int n, i, *p;
    printf("Enter number of elements: ");
    scanf("%d", &n);
    p = (int*)malloc(n * sizeof(int));
    if(p == NULL) {
        printf("Memory not allocated.\n");
        return 1;
    }
    printf("Enter elements:\n");
    for(i = 0; i < n; i++)
        scanf("%d", p + i);
    printf("You entered:\n");
    for(i = 0; i < n; i++)
        printf("%d ", *(p + i));
    free(p);
    return 0;
}
```

Output

```
Enter number of elements: 3
Enter elements:
1
2
3
1 2 3
```

10.Write a program to sort an array using pointer notation.

IPO

- **Input**: Array elements.
- **Process**: Use pointer arithmetic to sort elements.
- **Output**: Sorted array.

Program

```
#include <stdio.h>
int main() {
```

```
    int arr[5], i, j, temp;
    int *p = arr;
    printf("Enter 5 elements:\n");
    for(i = 0; i < 5; i++)
        scanf("%d", p + i);
    for(i = 0; i < 5; i++) {
        for(j = i + 1; j < 5; j++) {
            if(*(p + i) > *(p + j)) {
                temp = *(p + i);
                *(p + i) = *(p + j);
                *(p + j) = temp;
            }
        }
    }
    printf("Sorted array:\n");
    for(i = 0; i < 5; i++)
        printf("%d ", *(p + i));
    return 0;
}
```

Output

```
Sorted array:
2 3 4 5 12
```