Day 6: Functions(9-8-2025)

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

IPO:

Input: a variable. Int a.

Process: Use a pointer to store the address of the variable and print it using %p format specifier.

Output: The memory address of the variable.

Code:

```
#include <stdio.h>
Void main()
{
 int num = 42;
 int *ptr = &num;
 printf("Value of num: %d\n", num);
 printf("Address of num: %p\n", (void*)ptr);
}
```

Output

```
Value of num: 42
Address of num: 0x7fff6e5f5964


=== Code Execution Successful ===
```

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

IPO:

Input:An array of elements (e.g.,int arr[5]={1,2,3,4,5,}).

Process:Use a pointer to traverse the array and access each element using pointer arithmetic (*(ptr+i)).

Output:The elements of the array printed one by one using the pointer.

Code:

```
#include <stdio.h>
int main()
{
 int arr[] = {10, 20, 30, 40, 50};
 int *ptr = arr;
 printf("Array elements:\n");
 for (int i = 0; i < 5; i++)
 {
 printf("%d ", *(ptr + i));
 }
 printf("\n");
 return 0;
}
```

Output Clear

```
Array elements:
10 20 30 40 50


=== Code Exited With Errors ===
```

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

IPO:

Input:Two integer values (e.g.,a=5,b=10).

Process:Use pointers to access and swap the values of the two variables using a temporary variable.

Output:The values of a and b after swapping (a=10,b=5).

Code:

```
#include <stdio.h>
void swap(int *x, int *y)
{
 int t=*x;
 *x=*y;
 *y=t;
}
int main()
{
 int a=5,b=10;
 printf("Before swap: a=%d,b=%d\n", a, b);
 swap(&a, &b);
 printf("After swap: a=%d, b=%d\n",a,b);
 return 0;
}
```

Output Clear

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


=== Code Execution Successful ===
```

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

IPO:

Input:Two integer values (e.g.,a=4,b=6).

Process:Use pointers to access the values and perform addition (*p+*q).

Output:The sum of the two numbers (e.g.,sum=10).

Code:

```
#include <stdio.h>
void main()
{
 int x=7,y= 3, s;
 int *p1=&x,*p2=&y;
 s=*p1+*p2;
 printf("Sum: %d\n", s);
}
```

Output Clear

```
Sum: 10


=== Code Exited With Errors ===
```

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

IPO:

Input:A string (e.g., “welcome”)

Process:Traverse the string using a pointer until the null character ‘\0’ is reached; count the characters.

Output:Length of the string (e.g.,5).

Code:

```
#include <stdio.h>
void main()
{
 char str[] = "welcome";
 char *pt=str;
 int length = 0;
 while (*pt!='\0')
 {
 length++;
 pt++;
 }
 printf("Length of string: %d\n", length);
}
```

Output Clear

```
Length of string: 7


=== Code Exited With Errors ===
```

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

IPO:

Input:A string (e.g.,” hello”).

Process:Use two pointers — one at the start and one at the end— and swap characters while moving toward the center.

Output:Reversed string (e.g.,”olleh”).

Code:

```
#include <stdio.h>
void main()
{
 char str[] = “ welcome”;
 char *start = str;
 char *end = str;
 while (*end != '\0')
 {
 end++;
 }
 end--;
 char temp;
 while (start<end)
 {
 temp = *start;
 *start = *end;
 *end = temp;
 start++;
 end--;
 }
 printf("Reversed string: %s\n", str);
}
```

```
Output                                    Clear
Reversed string: emoclew


=== Code Exited With Errors ===
```

7. Write a program to count vowels using pointer.

IPO:

Input:A string (e.g., “ apple”).

Process:Use a pointer to traverse the string and check each character.If it is a vowel (a,e,i,o,u in both cases), increment the count.

Output:Number of vowels in the string (e.g.,2).

Code:

```
#include <stdio.h>
int isVowel(char ch)
{
 if (ch == 'a' || ch == 'A' ||
 ch == 'e' || ch == 'E' ||
 ch == 'i' || ch == 'I' ||
 ch == 'o' || ch == 'O' ||
 ch == 'u' || ch == 'U') {
 return 1;
 }
 return 0;
}
```

```
int main()
{
 char str[] = "welcome to c Programming";
 char *ptr = str;
 int count = 0;
 while (*ptr != '\0')
 {
 if (isVowel(*ptr))
 {
 count++;
 }
 ptr++;
 }
 printf("Number of vowels: %d\n", count);
 return 0;
}
```

Output Clear

```
Number of vowels: 7


=== Code Execution Successful ===
```

8. Write a program to demonstrate pointer to pointer.

IPO:

Input:A variable with a value (e.g.,int x=10).

Process:Create a pointer to store the address of the variable, then create another pointer to store the address of the first pointer, and access the value using both levels of pointers.

Output:Value of the variable accessed through a pointer to pointer.

Code:

```
#include <stdio.h>
void main()
{
 int num = 70;
 int *ptr = &num;
 int **pptr = &ptr;
 printf("Value of num: %d\n", num);
 printf("Value via ptr: %d\n", *ptr);
 printf("Value via pptr: %d\n", **pptr);
}
```

Output Clear

```
Value of num: 70
Value via ptr: 70
Value via pptr: 70


=== Code Exited With Errors ===
```

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

IPO:

Input:Size of memory to allocate and data to store in it.

Process:Use malloc( ) to dynamically allocate memory,

store values in the allocated memory, display them, and release memory using free( ).

Output:Values stored in dynamically allocated memory.

Code:

```
#include <stdio.h>
#include <stdlib.h>
void main()
{
 int *arr;
 int n = 10;
 arr = (int *)malloc(n * sizeof(int));
 if (arr == NULL)
 {
 printf("Memory allocation failed\n");
 }
 for (int i = 0; i < n; i++)
 {
 arr[i] = i + 1;
 }
 printf("Array elements: ");
 for (int i = 0; i < n; i++)
 {
 printf("%d ", arr[i]);
 }
 printf("\n");
 free(arr);
}
```

```
Output                                    Clear
Array elements: 1 2 3 4 5 6 7 8 9 10


=== Code Execution Successful ===
```

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

IPO:

Input:Number of elements and the array elements.

Process:Use pointer notation to compare and swap elements, arrange the array in ascending (or descending) order.

Output:Sorted array elements.

Code:

```
#include <stdio.h>
void sort(int *arr, int n)
{
 for (int i = 0; i < n - 1; i++)
 {
 for (int j = i + 1; j < n; j++)
 {
 if (*(arr + j) < *(arr + i))
 {
 int temp = *(arr + i);
 *(arr + i) = *(arr + j);
 *(arr + j) = temp;
 }
 }
```

```
 }
}
int main()
{
 int arr[] = {5, 2, 9, 1, 3};
 int n = 5;
 sort(arr, n);
 printf("Sorted array: ");
 for (int i = 0; i < n; i++)
 {
 printf("%d ", *(arr + i));
 }
 printf("\n");
 return 0;
}
```

Output Clear

```
Sorted array: 1 2 3 5 9


=== Code Execution Successful ===
```