✅ Day : Pointers (9-8-2025)

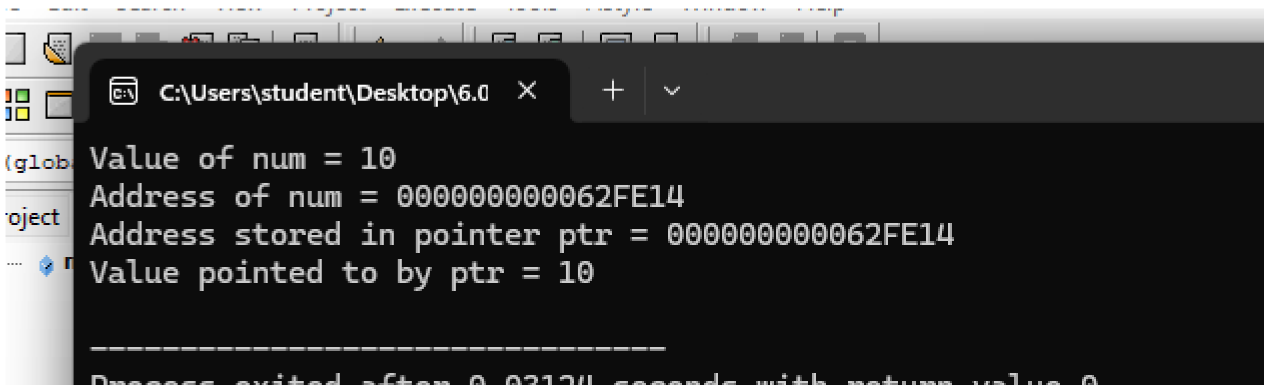
1. Write a program to print the address of a variable using pointer.

☐ **Input: A variable `num` initialized with a value.**

☐ **Process: Store the address of `num` in a pointer and print it.**

☐ **Output: Address of the variable `num`.**

```
#include <stdio.h>
int main()
{
   int num = 10;
   int *ptr;
   ptr = &num;
   printf("Value of num = %d\n", num);
   printf("Address of num = %p\n", &num);
   printf("Address stored in pointer ptr = %p\n", ptr);
   printf("Value pointed to by ptr = %d\n", *ptr);
   return 0;
}
```

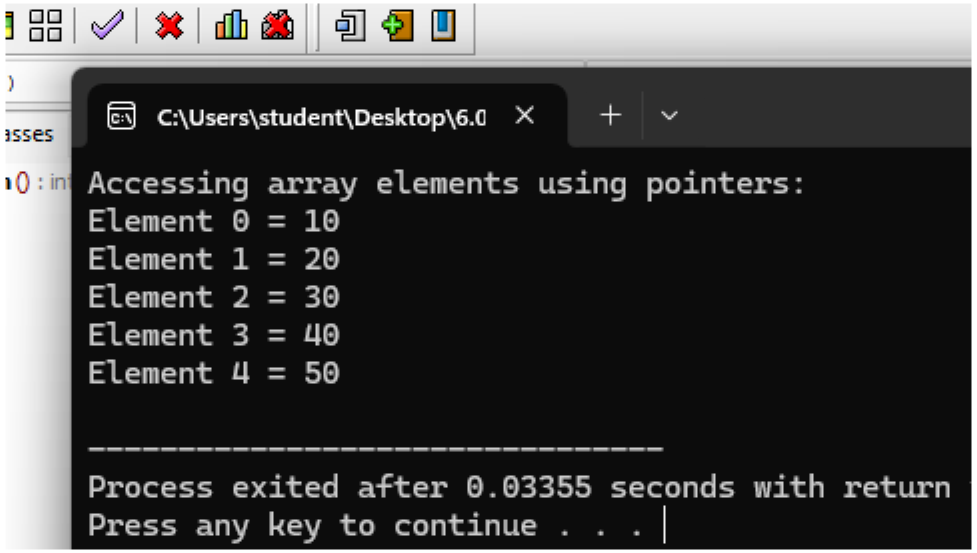


```
Value of num = 10
Address of num = 000000000062FE14
Address stored in pointer ptr = 000000000062FE14
Value pointed to by ptr = 10

--------------------------------
Process exited after 0.03124 seconds with return value 0
```

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

- **Input: An array of integers `{10, 20, 30, 40, 50}`**
- **Process: Use a pointer to access each element using pointer arithmetic `*(ptr + i)`**
- **Output: Print each element of the array using the pointer**

```
#include <stdio.h>
int main()
{
   int arr[5] = {10, 20, 30, 40, 50};
   int *ptr;
   int i;

   ptr = arr;
   printf("Accessing array elements using pointers:\n");
   for (i = 0; i < 5; i++) {
      printf("Element %d = %d\n", i, *(ptr + i));
   }
   return 0;
}
```

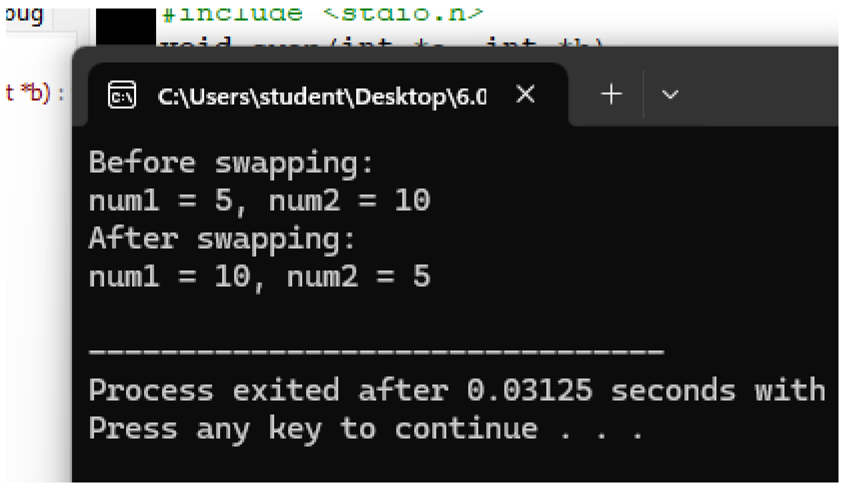


```
Accessing array elements using pointers:
Element 0 = 10
Element 1 = 20
Element 2 = 30
Element 3 = 40
Element 4 = 50

--------------------------------
Process exited after 0.03355 seconds with return
Press any key to continue . . .
```

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

- **Input: Two integers `num1 = 5`, `num2 = 10`**
- **Process: Swap the values using pointer references**
- **Output: Values of `num1` and `num2` after swapping**

```
#include <stdio.h>
 void swap(int *a, int *b)
 {
int temp; temp = *a; *a = *b;
*b = temp;
 }
int main()
 {
int num1 = 5, num2 = 10;
 printf("Before swapping:\n");
 printf("num1 = %d, num2 = %d\n", num1, num2);
swap(&num1, &num2);
printf("After swapping:\n");
printf("num1 = %d, num2 = %d\n", num1, num2);
return 0;
}
```

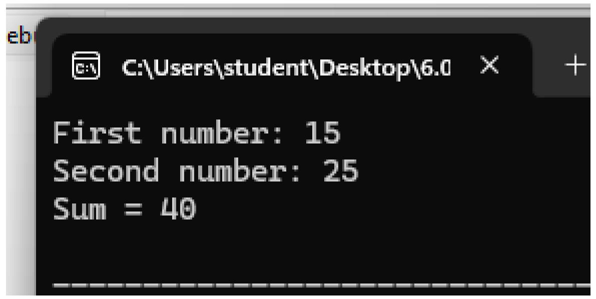


```
Before swapping:
num1 = 5, num2 = 10
After swapping:
num1 = 10, num2 = 5

--------------------------------
Process exited after 0.03125 seconds with
Press any key to continue . . .
```

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

- **Input: Two integers num1 = 15, num2 = 25**
- **Process: Use pointers to access the numbers and add them**
- **Output: Display the sum of the two numbers**

```
#include <stdio.h>
int main()
{
int num1 = 15, num2 = 25, sum; int *ptr1, *ptr2;ptr1 = &num1;
ptr2 = &num2;
sum = *ptr1 + *ptr2;
printf("First number: %d\n", *ptr1);
printf("Second number: %d\n", *ptr2);
printf("Sum = %d\n", sum);
return 0;
}
```

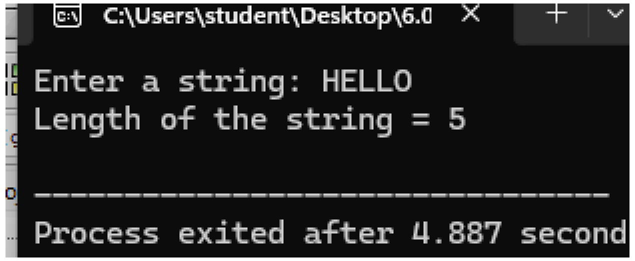

C:\Users\student\Desktop\6.0

```
First number: 15
Second number: 25
Sum = 40

------------------------------
```

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

- **Input: A string entered by the user**
- **Process: Use a pointer to iterate through the string until the null character `\0` is reached, counting each character**
- **Output: Print the length of the string**

```
#include <stdio.h>
int main()
{
   char str[100];
   char *ptr;
   int length = 0;
   printf("Enter a string: ");
   scanf("%s", str);
   ptr = str;
   while (*ptr != '\0')
{
      length++;
      ptr++;
   }
   printf("Length of the string = %d\n", length);
   return 0;
}
```

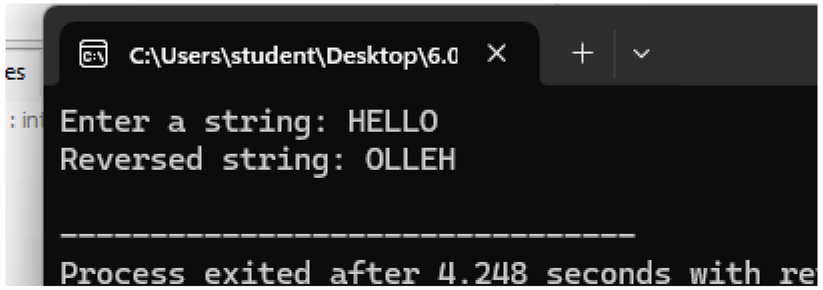


```
Enter a string: HELLO
Length of the string = 5

--------------------------------
Process exited after 4.887 second
```

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

- **Input: A string entered by the user**
- **Process: Use two pointers (`start` and end) to swap characters from front and back until the middle is reached**
- **Output: Display the reversed string**

```
#include <stdio.h>
 #include <string.h>
int main()
{
 char str[100], temp; char *start, *end; int len, i;
printf("Enter a string: ");
scanf("%s", str);
len = strlen(str);
start = str;
end = str + len - 1;
while (start < end)
{
   temp = *start;
   *start = *end;
   *end = temp;
   start++;
   end--;
}
printf("Reversed string: %s\n", str);
return 0;
}
```



```
Enter a string: HELLO
Reversed string: OLLEH

--------------------------------
Process exited after 4.248 seconds with re
```

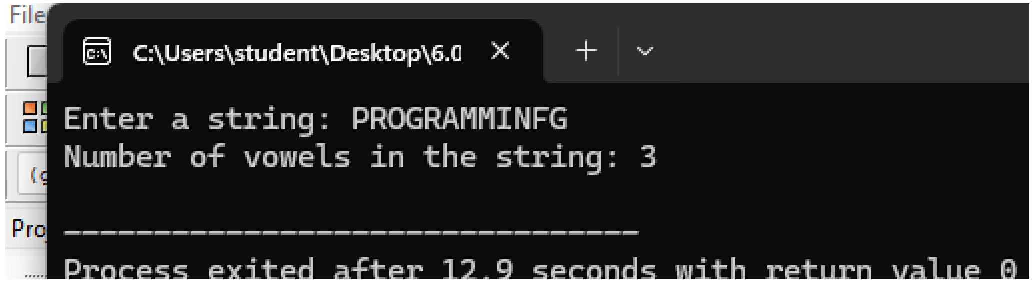
7. Write a program to count vowels using pointer.

- **Input: A string entered by the user**

- **Process: Use a pointer to check each character for a vowel and count them**

- **Output: Display the total number of vowels in the string**

```
#include <stdio.h>
int main()
{
 char str[100]; char *ptr; int count = 0;
printf("Enter a string: ");
scanf("%s", str);
ptr = str;
while (*ptr != '\0') {
   if (*ptr == 'a' || *ptr == 'e' || *ptr == 'i' || *ptr == 'o' || *ptr == 'u' ||
       *ptr == 'A' || *ptr == 'E' || *ptr == 'I' || *ptr == 'O' || *ptr == 'U')
{
       count++;
   }
   ptr++;
}
printf("Number of vowels in the string: %d\n", count);
return 0;
}
```

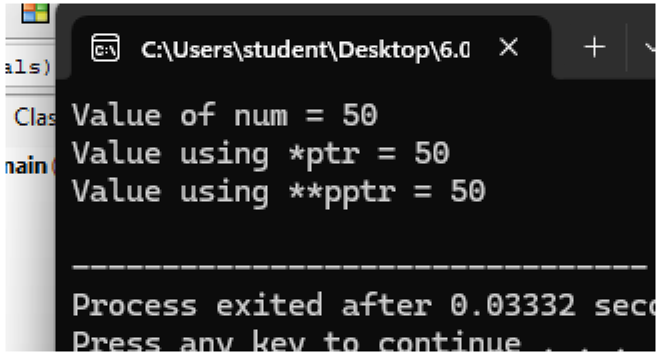


```
Enter a string: PROGRAMMINFG
Number of vowels in the string: 3

--------------------------------
Process exited after 12.9 seconds with return value 0
```

8. Write a program to demonstrate pointer to pointer.

- **Input: An integer variable num = 50**
- **Process: Create a pointer to the variable, and another pointer to that pointer. Use dereferencing to access the value.**
- **Output: Display the value using num, *ptr, and **pptr**

```
#include <stdio.h>
int main()
{
int num = 50; int *ptr; int **pptr;ptr = &num;
pptr = &ptr;    r
printf("Value of num = %d\n", num);
printf("Value using *ptr = %d\n", *ptr);
printf("Value using **pptr = %d\n", **pptr);
return 0;

}
```

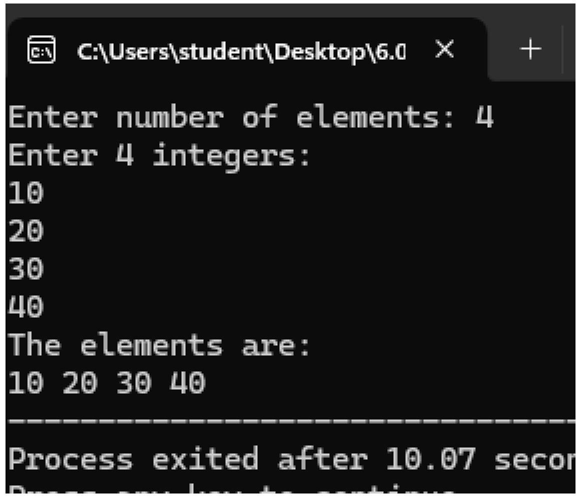


```
Value of num = 50
Value using *ptr = 50
Value using **pptr = 50

--------------------------------
Process exited after 0.03332 sec
Press any key to continue . . .
```

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

- **Input: Number of elements and the values of the elements**
- **Process: Dynamically allocate memory using `malloc()`, store and display values, then free memory using `free()`**
- **Output: Display entered integers, then release memory**

```
#include <stdio.h>
#include <stdlib.h>
int main()
{
int *ptr; int n, i;
printf("Enter number of elements: ");
scanf("%d", &n);
ptr = (int *)malloc(n * sizeof(int));
if (ptr == NULL)
{
   printf("Memory not allocated!\n");
   return 1;
}
printf("Enter %d integers:\n", n);
for (i = 0; i < n; i++) {
   scanf("%d", &ptr[i]);
}

// Display elements
printf("The elements are:\n");
for (i = 0; i < n; i++) {
   printf("%d ", ptr[i]);
}

// Free the allocated memory
free(ptr);
return 0;

}
```

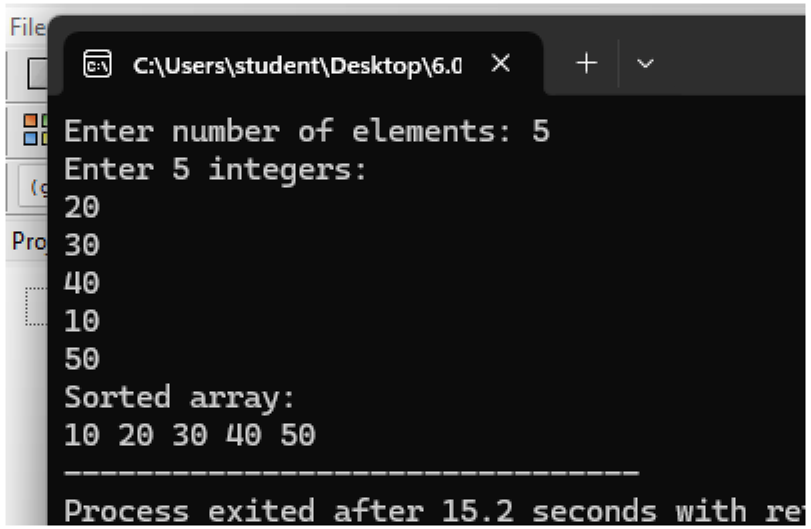


```
Enter number of elements: 4
Enter 4 integers:
10
20
30
40
The elements are:
10 20 30 40
--------------------------------
Process exited after 10.07 secon
```

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

- **Input: Number of elements and their values**
- **Process: Sort the array using pointer arithmetic (`*(arr + i)`)**
- **Output: Display the sorted array in ascending order**

```c
#include <stdio.h>
void sort(int *arr, int n)
{
 int i, j, temp;for (i = 0; i < n - 1; i++)
{
   for (j = i + 1; j < n; j++)
{
       if (*(arr + i) > *(arr + j))
{
           temp = *(arr + i);
           *(arr + i) = *(arr + j);
           *(arr + j) = temp;
       }
   }
}
int main()
{
 int arr[100], n, i;
printf("Enter number of elements: ");
scanf("%d", &n);
printf("Enter %d integers:\n", n);
for (i = 0; i < n; i++)
{
   scanf("%d", (arr + i));
}
sort(arr, n);
printf("Sorted array:\n");
for (i = 0; i < n; i++)
{
   printf("%d ", *(arr + i));
}
return 0;
}
```



```
Enter number of elements: 5
Enter 5 integers:
20
30
40
10
50
Sorted array:
10 20 30 40 50
--------------------------------
Process exited after 15.2 seconds with re
```