**Introduction to Pointers Assignments**

1. Refer the code snippet below. int main()

{

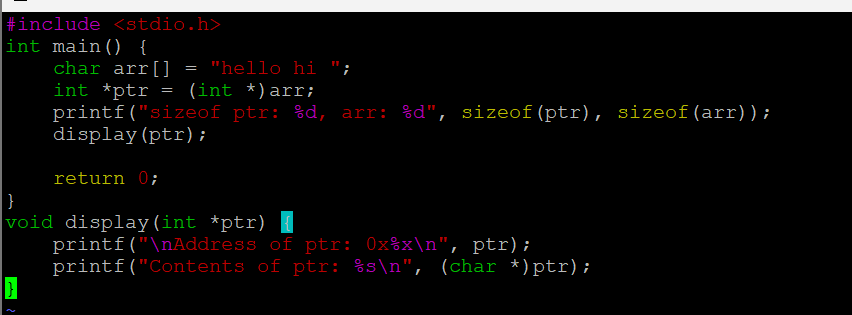
char arr=”hello hi “;

int \*ptr = arr;

printf(“sizeof ptr:%d, arr:%d”, sizeof(ptr), sizeof(arr));

display(ptr); // display the address in hex and contents using pointer

}

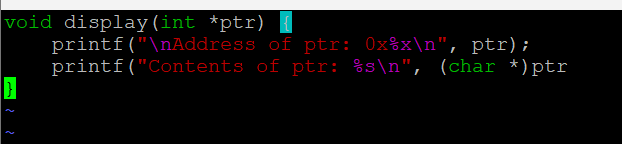


A black background with white text

Description automatically generated

Perform the following.

1. Implement the display() function (Use the “0x%x” formatting specifier to print addresses in hexadecimal.)



1. comment on the sizeof(ptr) and sizeof(arr)

A) sizeof(ptr): The pointer ptr is of type int\*, so sizeof(ptr) will give the size of a pointer to int. On most systems, the size of a pointer is 4 or 8 bytes (depending on whether the system is 32-bit or 64-bit).

sizeof(arr): arr is a character array, and the size of arr is determined by the length of the string "hello hi " which has 10 characters. Since sizeof() returns the total size of the array in bytes, sizeof(arr) will return the total size of the array, which is 11 bytes (including the null terminator).

2. Refer the code snippet below. int main()

#define MAX 100

#define SUCCESS 0

#define FAILURE 1

int main()

{

char arr[MAX] = “Learning C“;

char\*ptr = arr;

char appendstr[3]= “in my org”;

printf(“Address of ptr:%x”, ptr);

int ret = append(ptr, appendstr);// append the string

printf(“Address of ptr:%x”, ptr);

if (ret == SUCCESS)

{

display(ptr); // display the address in hex and contents using pointer

}

}

Perform the following.

a. Implement the append() function to append the contents of the appendstr[] to arr using pointer.

[Note: append() should only use its content and not manipulate it. Contents should be retained even after the call]

3. Refer the code in “pointer\_prg.c”. The functions swap\_nums() and swap\_pointers() are expected to swap the numbers and pointers respectively. But swap\_pointers() is currently not giving the expected results. Analyse and the fix the issue.

A) Pass pointers to pointers: In swap\_pointers(), we change the function signature to accept char \*\*x and char \*\*y. This way, the function can modify the original pointers.

Dereference to swap: Inside swap\_pointers(), we dereference x and y to swap the actual pointers.

Pass the addresses of the pointers: In test\_swap\_char(), we pass &s1 and &s2 to swap\_pointers() so that the function can modify the original pointers.

