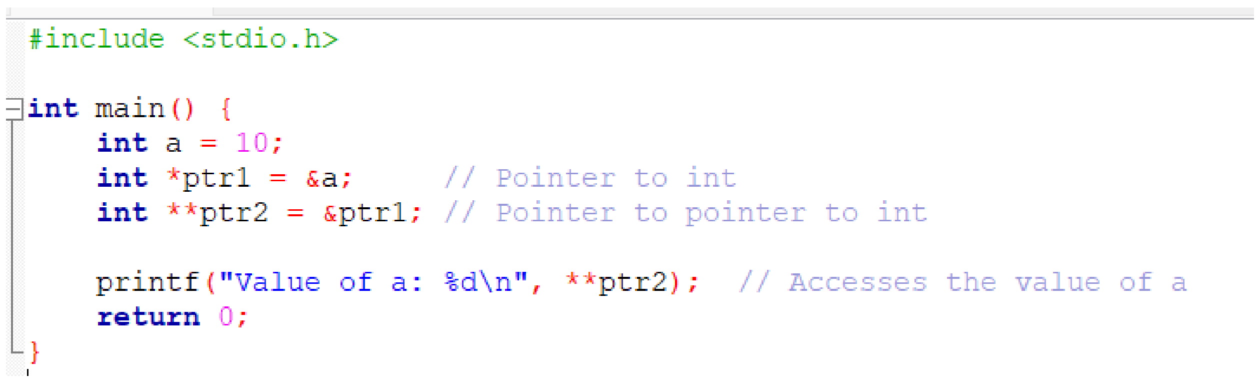
1. Purpose of Double Pointers:

A **double pointer** (pointer to a pointer) is a pointer that stores the address of another pointer. Double pointers are used in situations where we need to work with multi-level indirection or manipulate the address of a pointer.

- **Applications of Double Pointers:**
    - **Dynamic Memory Allocation:** When we use functions like malloc() to allocate memory, double pointers are used to modify the pointer itself in functions.
    - **2D Arrays:** Double pointers are useful for handling 2D arrays dynamically, where each element in a row can be treated as a pointer to the first element of the row.
    - **Passing by Reference:** Double pointers are used to modify a pointer passed to a function.
- Example:



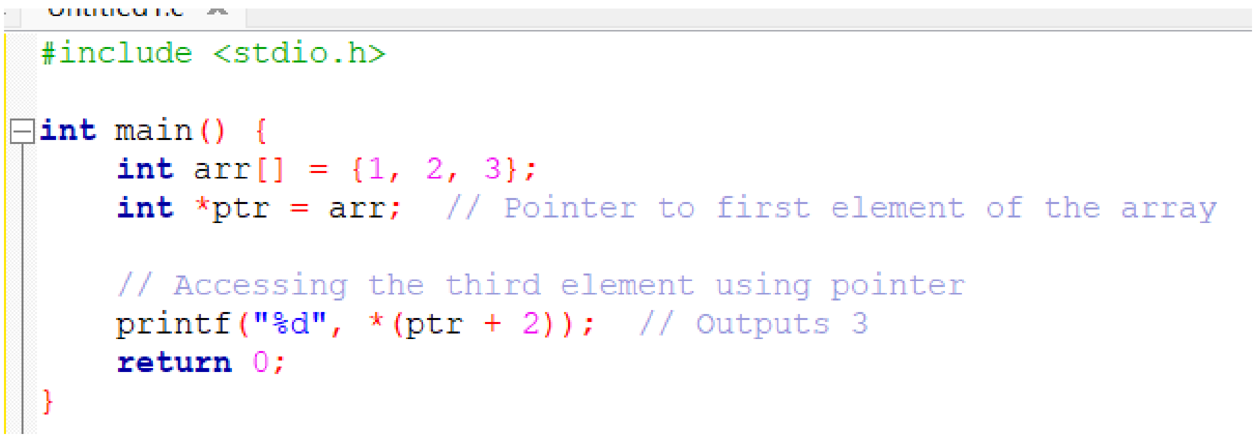
```c
#include <stdio.h>

int main() {
    int a = 10;
    int *ptr1 = &a;      // Pointer to int
    int **ptr2 = &ptr1; // Pointer to pointer to int

    printf("Value of a: %d\n", **ptr2);  // Accesses the value of a
    return 0;
}
```

- In this example, ptr2 is a double pointer that holds the address of ptr1. By dereferencing ptr2 twice, we access the value of a.

2. Relation Between Pointers, Arrays, and Strings:

Pointers are closely related to arrays and strings in C. In fact, arrays and strings can be thought of as pointers in certain contexts:

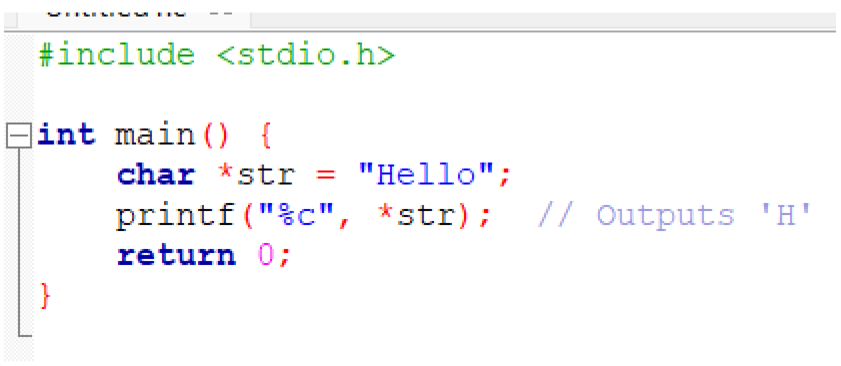
- **Arrays and Pointers:** The name of an array represents a pointer to its first element. Arrays are stored in contiguous memory locations, and pointers are often used to access the elements of an array by incrementing the pointer.
- Example for Arrays:



```
#include <stdio.h>

int main() {
    int arr[] = {1, 2, 3};
    int *ptr = arr;  // Pointer to first element of the array

    // Accessing the third element using pointer
    printf("%d", *(ptr + 2));  // Outputs 3
    return 0;
}
```

Here, ptr points to the first element of the array arr, and by incrementing the pointer, we can access other elements.

- **Strings and Pointers:** In C, strings are simply arrays of characters. A string is stored in memory as a contiguous block of characters, and a pointer to the first character of the string can be used to access it.
- Example for Strings:



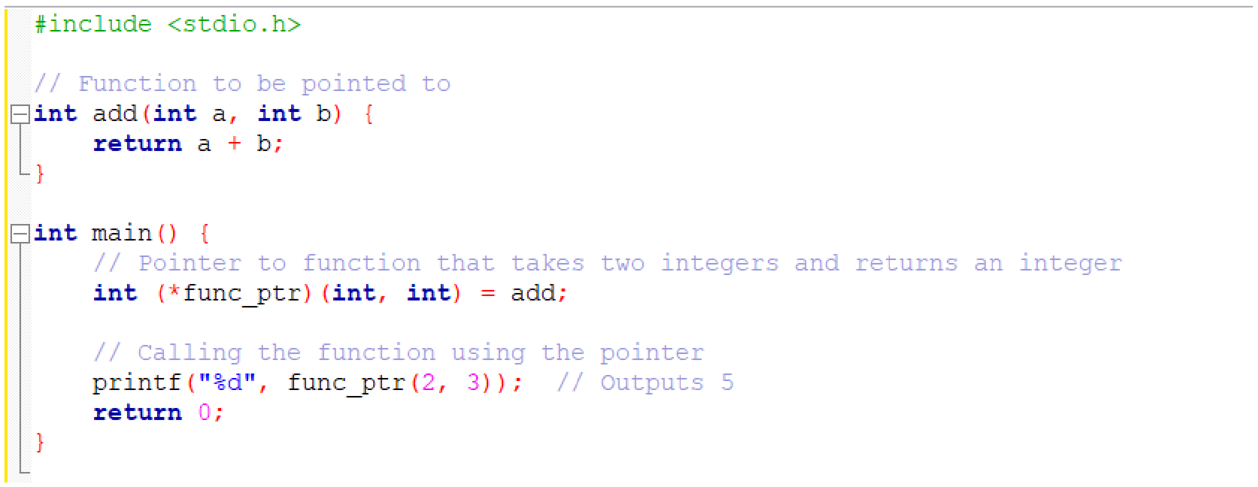
```
#include <stdio.h>

int main() {
    char *str = "Hello";
    printf("%c", *str);  // Outputs 'H'
    return 0;
}
```

Here, str is a pointer to the first character of the string "Hello", and dereferencing str gives the first character.

3. Purpose of Pointer to Function:

A **pointer to a function** is a pointer that holds the address of a function. This allows us to call a function indirectly through the pointer, enabling dynamic function selection or passing a function as an argument to another function.

- **Applications of Pointer to Function:**
  - **Callback Functions:** Pointers to functions are often used in callback mechanisms, where a function is passed as an argument to another function to be executed later.
  - **Dynamic Function Selection:** Pointers to functions allow us to select a function to call at runtime.
- Example:



```c
#include <stdio.h>

// Function to be pointed to
int add(int a, int b) {
    return a + b;
}

int main() {
    // Pointer to function that takes two integers and returns an integer
    int (*func_ptr)(int, int) = add;

    // Calling the function using the pointer
    printf("%d", func_ptr(2, 3));  // Outputs 5
    return 0;
}
```

In this example, func_ptr is a pointer to the add function. The function is called through the pointer, demonstrating how function pointers can be used.