Introduction to programming with C

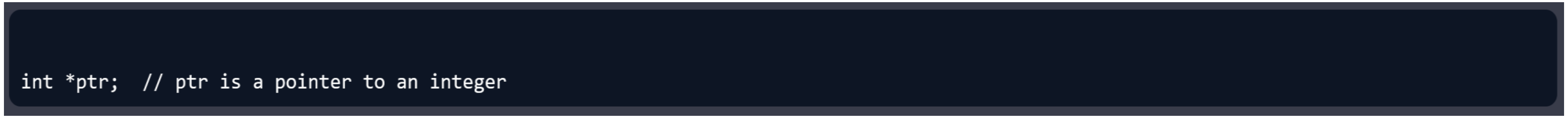
By: Chimango Nyasulu, PhD

Outline

- Declaring a pointer
- Referencing a pointer
- Pointer differences
- Arrays as pointers
- Pointers in function arguments

Pointers

- A pointer is a variable that stores the memory address of another variable.
- Pointers are a fundamental feature of the C programming language and are used for various purposes, such as dynamic memory allocation, arrays, and function arguments.
- To declare a pointer, you use the * symbol. For example:



```c
int *ptr;  // ptr is a pointer to an integer
```

Pointers

- You can initialize a pointer with the address of a variable using the address-of operator &.

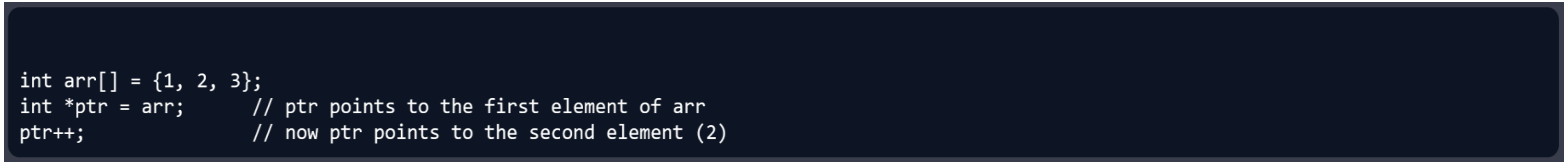
```
int var = 10;
int *ptr = &var;  // ptr now holds the address of var
```

- You can access the value at the address stored in a pointer using the dereference operator *.

```
int value = *ptr;  // value is now 10, which is the value of var
```

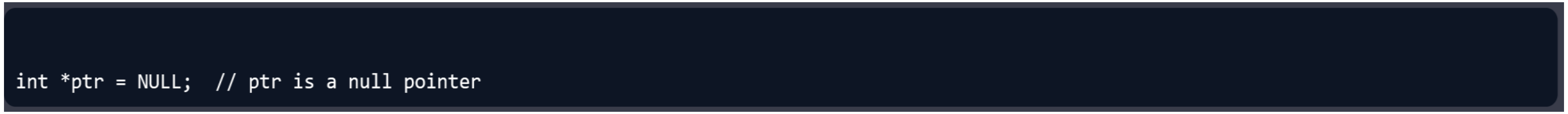
Pointers

- Pointers can be incremented or decremented, which moves the pointer to the next or previous memory location, respectively. This is particularly useful when working with arrays.



```
int arr[] = {1, 2, 3};
int *ptr = arr;      // ptr points to the first element of arr
ptr++;               // now ptr points to the second element (2)
```

- A pointer can be assigned a value of NULL, indicating that it does not point to any valid memory location.



```
int *ptr = NULL;  // ptr is a null pointer
```

Pointers

- Pointers are commonly used for dynamic memory allocation with functions like *malloc, calloc*, and *free*.

- *malloc* - allocates size bytes of memory and returns a pointer to the beginning of the allocated memory block.

- The memory allocated by malloc is uninitialized; it contains garbage values.

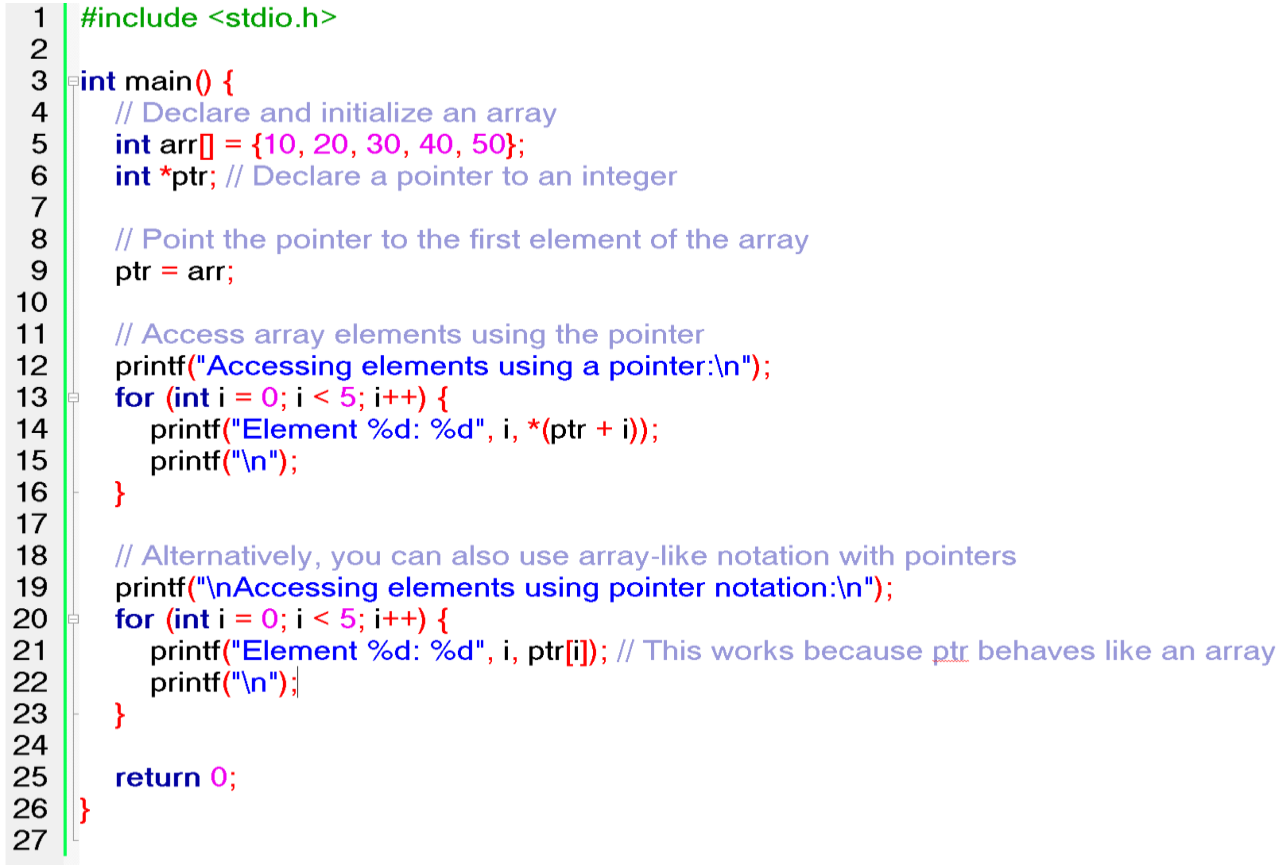
- *malloc* - allocates memory for an array of *n* elements, each of size bytes, and initializes all bytes to zero.

- Unlike malloc, calloc initializes the allocated memory to zero.

- *free* - deallocates the memory previously allocated by malloc or calloc.

ICT @ Mzuni

Pointers



```c
#include <stdio.h>

int main() {
    // Declare and initialize an array
    int arr[] = {10, 20, 30, 40, 50};
    int *ptr; // Declare a pointer to an integer

    // Point the pointer to the first element of the array
    ptr = arr;

    // Access array elements using the pointer
    printf("Accessing elements using a pointer:\n");
    for (int i = 0; i < 5; i++) {
        printf("Element %d: %d", i, *(ptr + i));
        printf("\n");
    }

    // Alternatively, you can also use array-like notation with pointers
    printf("\nAccessing elements using pointer notation:\n");
    for (int i = 0; i < 5; i++) {
        printf("Element %d: %d", i, ptr[i]); // This works because ptr behaves like an array
        printf("\n");
    }

    return 0;
}

```