4. Pointers and Arrays

Arrays and pointers have a very close relationship in C. In many contexts, an array name acts as a pointer to its first element.

4.1 Array Name as Pointer

```
#include <stdio.h>

int main() {
    int arr[5] = {10, 20, 30, 40, 50};

    // Array name is a pointer to first element
    printf("Address of arr: %p\n", (void*)arr);
    printf("Address of arr[0]: %p\n", (void*)&arr[0]);
    printf("These addresses are the same!\n\n");

    // Access elements using pointer notation
    printf("arr[0] = %d, *arr = %d\n", arr[0], *arr);
    printf("arr[1] = %d, *(arr+1) = %d\n", arr[1], *(arr + 1));
    printf("arr[2] = %d, *(arr+2) = %d\n", arr[2], *(arr + 2));

    return 0;
}
```

4.2 Relationship Between Arrays and Pointers

Key Equivalences:

```
int arr[5] = {10, 20, 30, 40, 50};
int *ptr = arr;

// These are equivalent:
arr[i]  ≡  *(arr + i)
arr[i]  ≡  ptr[i]
arr[i]  ≡  *(ptr + i)
&arr[i] ≡  (arr + i)
```

```
&arr[i] ≡  (ptr + i)
```

Visual Representation:

```
Array: arr[5] = {10, 20, 30, 40, 50}

Index:     0    1    2    3    4
          ┌────┬────┬────┬────┬────┐
arr -->   | 10 | 20 | 30 | 40 | 50 |
          └────┴────┴────┴────┴────┘

           ↑
         arr, &arr[0], arr+0, *(arr+0)
               ↑
           arr+1, &arr[1], *(arr+1)
                    ↑
               arr+2, &arr[2], *(arr+2)
```

4.3 Traversing Arrays with Pointers

Method 1: Using array indexing

```
int arr[5] = {10, 20, 30, 40, 50};

for (int i = 0; i < 5; i++) {
    printf("%d ", arr[i]);
}
```

Method 2: Using pointer arithmetic

```
int arr[5] = {10, 20, 30, 40, 50};
int *ptr = arr;

for (int i = 0; i < 5; i++) {
    printf("%d ", *(ptr + i));
}
```

Method 3: Incrementing pointer

```
int arr[5] = {10, 20, 30, 40, 50};
int *ptr = arr;
```

```
int *end = arr + 5;

while (ptr < end) {
    printf("%d ", *ptr);
    ptr++;
}
```

4.4 Important Difference: Array vs Pointer

```
int arr[5] = {1, 2, 3, 4, 5};
int *ptr = arr;

// This is VALID:
ptr = ptr + 1;  // ptr can be modified
ptr++;          // ptr can be incremented

// This is INVALID:
arr = arr + 1;  // ERROR! Array name is a constant pointer
arr++;          // ERROR! Cannot modify array name

// However, this is valid:
int *ptr2 = arr + 1;  // Create new pointer pointing to arr[1]
```

Key Difference:

- `arr` is a **constant pointer** (cannot be reassigned)
- `ptr` is a **pointer variable** (can be modified)

Revision #1
Created 2025-10-01 03:26:48 UTC by DS
Updated 2025-10-01 03:27:09 UTC by DS