5. Variable Scope and Lifetime

5.1 Local Variables

Variables declared inside a function are **local** to that function:

- **Scope:** Only accessible within the function where they're declared
- **Lifetime:** Created when function is called, destroyed when function returns
- **Storage:** Usually on the stack

```
#include <stdio.h>

void function_a() {
    int local_var = 10;  // Local to function_a
    printf("In function_a: %d\n", local_var);
}

void function_b() {
    int local_var = 20;  // Different variable, local to function_b
    printf("In function_b: %d\n", local_var);
}

int main() {
    int local_var = 5;   // Local to main
    printf("In main: %d\n", local_var);

    function_a();  // Output: In function_a: 10
    function_b();  // Output: In function_b: 20

    printf("Back in main: %d\n", local_var);  // Still 5
    return 0;
}
```

5.2 Global Variables

Variables declared outside all functions are **global**:

- **Scope:** Accessible from any function in the program
- **Lifetime:** Exist for the entire program execution
- **Storage:** In static memory area

```
#include <stdio.h>

// Global variables
int global_counter = 0;
float global_sum = 0.0;

void increment_counter() {
    global_counter++;  // Can access global variable
    printf("Counter: %d\n", global_counter);
}

void add_to_sum(float value) {
    global_sum += value;  // Can modify global variable
    printf("Sum: %.2f\n", global_sum);
}

int main() {
    increment_counter();  // Counter: 1
    increment_counter();  // Counter: 2

    add_to_sum(10.5);     // Sum: 10.50
    add_to_sum(7.3);      // Sum: 17.80

    printf("Final values - Counter: %d, Sum: %.2f\n", global_counter, global_sum);
    return 0;
}
```

5.3 Variable Shadowing

When a local variable has the same name as a global variable, the local variable "shadows" the global one:

```
#include <stdio.h>

int value = 100;  // Global variable
```

```
void test_shadowing() {
    int value = 50;  // Local variable shadows the global one
    printf("Local value: %d\n", value);  // Prints 50
}

int main() {
    printf("Global value: %d\n", value);  // Prints 100
    test_shadowing();
    printf("Global value after function: %d\n", value);  // Still 100
    return 0;
}
```

5.4 Best Practices for Variable Scope

1. **Minimize global variables:** Use them sparingly
2. **Prefer local variables:** Keep data close to where it's used
3. **Use meaningful names:** Avoid naming conflicts
4. **Initialize variables:** Always initialize before use

```
// Good practice
int calculate_area(int length, int width) {
    int area = length * width;  // Local variable, clearly named
    return area;
}

// Avoid this
int x, y, z;  // Global variables - hard to track
int calc(int a, int b) {
    z = a * b;  // Modifying global state
    return z;
}
```

Revision #1
Created 2025-09-06 10:19:28 UTC by DS
Updated 2025-09-06 10:20:07 UTC by DS