9. Common Errors and Debugging

9.1 Function Declaration Errors

Error 1: Missing Function Prototype

```
// ERROR: Function used before declaration
int main() {
    int result = add_numbers(5, 3);  // Error: 'add_numbers' not declared
    return 0;
}

int add_numbers(int a, int b) {
    return a + b;
}
```

Solution:

```
// CORRECT: Add function prototype
int add_numbers(int a, int b);  // Function prototype

int main() {
    int result = add_numbers(5, 3);  // Now it works
    return 0;
}

int add_numbers(int a, int b) {
    return a + b;
}
```

9.2 Return Type Mismatches

Error 2: Wrong Return Type

```
// ERROR: Function declared to return int but returns float
int divide(int a, int b) {
    return a / b;  // Integer division, loses decimal part
}

int main() {
    printf("Result: %d\n", divide(7, 2));  // Output: 3 (not 3.5)
    return 0;
}
```

Solution:

```
// CORRECT: Use appropriate return type
float divide(int a, int b) {
    return (float)a / b;  // Cast to float for proper division
}

int main() {
    printf("Result: %.2f\n", divide(7, 2));  // Output: 3.50
    return 0;
}
```

9.3 Parameter Issues

Error 3: Expecting Changes to Original Variables

```
void swap(int a, int b) {
    int temp = a;
    a = b;
    b = temp;
    printf("Inside function: a=%d, b=%d\n", a, b);  // Values swapped
}

int main() {
    int x = 5, y = 10;
    swap(x, y);
    printf("In main: x=%d, y=%d\n", x, y);  // x=5, y=10 (unchanged!)
    return 0;
}
```

Understanding: This behavior is correct in C due to pass by value. To actually swap values, you would need pointers (weill be covered in module about "Pointer").

9.4 Debugging Techniques for Functions

1. Add Debug Prints:

```
int factorial(int n) {
    printf("DEBUG: factorial(%d) called\n", n);  // Debug output

    if (n == 0 || n == 1) {
        printf("DEBUG: base case reached, returning 1\n");
        return 1;
    } else {
        int result = n * factorial(n - 1);
        printf("DEBUG: factorial(%d) = %d\n", n, result);
        return result;
    }
}
```

2. Test Functions Independently:

```
// Test individual functions with known inputs
int main() {
    // Test factorial function
    assert(factorial(0) == 1);
    assert(factorial(1) == 1);
    assert(factorial(5) == 120);
    printf("All factorial tests passed!\n");

    return 0;
}
```

3. Check Function Signatures:

- Verify prototype matches definition
- Check parameter types and count
- Ensure return type is correct

Revision #2
Created 2025-09-06 10:30:00 UTC by DS
Updated 2025-09-06 10:30:47 UTC by DS