

# Design and Implementation of a C-based Spreadsheet Program

## COP290 Project Documentation

## 1 Program Structure and Design Decisions

### 1.1 Modular Architecture

The program follows a modular design with six core components:

- **Initialization Module (init.c/h)**: Handles memory allocation and sheet initialization
- **I/O Handler (io.c/h)**: Manages input parsing and command validation
- **Process Handler (process.c/h)**: Executes commands and evaluates formulas
- **Display Manager (display.c/h)**: Controls the visual representation
- **Dependency Manager (dependent.c/h)**: Manages cell relationships and updates
- **Stack Operations (stack.c/h)**: Provides auxiliary data structure support.



## 1.2 Key Design Decisions

### 1. Cell Dependencies

- Implementation of parent-child relationships for formula tracking
- Use of adjacency lists for cycle detection
- Topological sorting for ordered updates

### 2. Command Processing

- Structured parsing using ParsedCommand data structure
- Support for arithmetic operations and cell ranges
- Implementation of statistical functions (MIN, MAX, SUM, AVG, STDEV)

### 3. Memory Management

- Dynamic allocation for sheet structure
- Careful memory cleanup to prevent leaks
- Efficient storage of formulas and dependencies

## 2 Implementation Challenges

### 2.1 Technical Challenges

#### 1. Circular Dependency Detection

- Challenge: Detecting cycles in formula references
- Solution: Implementation of depth-first search algorithm
- Prevention of infinite loops during cell updates

#### 2. Formula Evaluation

- Challenge: Parsing and evaluating complex formulas
- Solution: Stack-based evaluation system
- Handling of nested functions and arithmetic operations

#### 3. Memory Management

- Challenge: Managing dynamic memory for cells and dependencies
- Solution: Structured allocation and deallocation procedures
- Prevention of memory leaks in dependency updates

#### 4. SLEEP Function

- Challenge: Implementing time delays without blocking spreadsheet operations

- Solution: Custom function type with dependency-aware sleep mechanism
- Prevention of deadlocks in formula evaluation chains

## 3 Error Handling and Edge Cases

### 3.1 Input Validation

- Cell reference validation (e.g., "A1", "ZZ999")
- Formula syntax checking
- Range validation (e.g., "A1:B10")
- Function name and argument validation

### 3.2 Error Scenarios Handled

#### 1. Formula Errors

- Division by zero
- Invalid cell references
- Malformed expressions
- Circular dependencies

#### 2. Range Errors

- Invalid range specifications
- Out-of-bounds references
- Empty cell ranges in functions

#### 3. Memory Errors

- Allocation failures
- Memory leak prevention
- Buffer overflow protection

## 4 Test Suite Coverage

### 4.1 Unit Tests (tests)

#### 1. Input Parsing Tests

- Command type identification
- Parameter extraction

- Formula parsing
- Function argument validation

## 2. **Validation Tests**

- Cell reference validation
- Range specification validation
- Function name validation
- Numeric input validation

## 3. **Cell Conversion Tests**

- String to row/column conversion
- Boundary value testing
- Invalid input handling

# 4.2 **Edge Case Coverage**

## 1. **Formula Evaluation**

- Complex nested formulas
- Multiple cell references
- Statistical function edge cases
- Error propagation

## 2. **Dependency Management**

- Circular reference detection
- Multiple dependency chains
- Update propagation
- Dependency cleanup

## 3. **Memory Management**

- Large sheet handling
- Dynamic resizing
- Resource cleanup