Copilot

Sure, I'd be happy to help explain that part of the program!

The program you provided is designed to find the Greatest Common Divisor (GCD) of two positive integers (A) and (B) using the **Euclidean algorithm**. The specific part you mentioned is the core of this algorithm. Let's break it down step by step:

Specific Part Explanation

```
while (B != 0) {
  temp <- B
  B <- A %% B
  A <- temp
}
```

1. **`while (B != 0)`**: This loop continues to execute as long as (B) is not zero. The Euclidean algorithm relies on repeatedly applying the modulus operation until one of the numbers becomes zero.

2. **`temp <- B`**: This line stores the current value of (B) in a temporary variable `temp`. This is necessary because the value of (B) will be updated in the next step, and we still need the original value of (B) to update (A).

3. **`B <- A %% B`**: This line updates (B) to be the remainder of (A) divided by (B). The modulus operation (`%%`) gives the remainder when (A) is divided by (B). This step is crucial as it reduces the problem size, bringing us closer to finding the GCD.

4. **`A <- temp`**: This line updates (A) to be the value of `temp`, which is the original value of (B). This step effectively swaps the roles of (A) and (B) for the next iteration of the loop.

Context of the Whole Program

The whole program can be summarized as follows:

1. **Function Definition**: The `find_gcd` function is defined to take two arguments (A) and (B).

2. **Loop Execution**: Inside the function, the `while` loop executes the Euclidean algorithm until (B) becomes zero.

3. **Return Statement**: Once (B) is zero, the loop exits, and the function returns the value of (A), which is the GCD of the original (A) and (B).

4. **Function Call**: The function is called with (A = 56) and (B = 98).

5. **Output**: The result is printed using the `cat` function, displaying the GCD of 56 and 98.

Example Walkthrough

Let's walk through the example with (A = 56) and (B = 98):

1. Initial values: (A = 56), (B = 98)
2. First iteration:
   - `temp <- B` → `temp = 98`
   - `B <- A %% B` → `B = 56 %% 98 = 56`
   - `A <- temp` → `A = 98`
3. Second iteration:
   - `temp <- B` → `temp = 56`
   - `B <- A %% B` → `B = 98 %% 56 = 42`
   - `A <- temp` → `A = 56`

4. Third iteration:
    - `temp <- B → temp = 42`
    - `B <- A %% B → B = 56 %% 42 = 14`
    - `A <- temp → A = 42`
5. Fourth iteration:
    - `temp <- B → temp = 14`
    - `B <- A %% B → B = 42 %% 14 = 0`
    - `A <- temp → A = 14`

At this point, (B) is zero, so the loop exits, and the function returns (A = 14), which is the GCD of 56 and 98.

I hope this helps! If you have any more questions or need further clarification, feel free to ask.