**1. Program Selected:**

#include <stdio.h>

#include <unistd.h> // For sleep()

void displaySignal(char \*signal, int duration) {

printf("%s light ON for %d seconds.\n", signal, duration);

sleep(duration);

printf("%s light OFF.\n", signal);

}

void controlTraffic(int cycles) {

int count = 0;

while (count < cycles) {

printf("\nCycle %d starting...\n", count + 1);

displaySignal("RED", 5);

displaySignal("YELLOW", 2);

displaySignal("GREEN", 5);

count++;

}

}

int main() {

int totalCycles;

printf("Enter number of signal cycles: ");

scanf("%d", &totalCycles);

if (totalCycles > 0)

controlTraffic(totalCycles);

else

printf("Invalid number of cycles.\n");

printf("\nTraffic Control Program Ended.\n");

return 0;

}

**2. Structured Constructs Identified:**

| **Construct** | **Example** | **Description** |
| --- | --- | --- |
| **Sequence** | printf(), sleep() statements | Execute in order |
| **Selection** | if (totalCycles > 0) | Decision-making |
| **Iteration** | while (count < cycles) | Repetition for signal cycles |

Each function performs a **single, clear task**, perfectly fitting structured programming principles.

**3. Violation Check:**

* ✅ No goto used.
* ✅ Loops and conditions properly structured.
* ✅ Clear entry and exit points.

Hence, no violation of structured programming rules.

**4. Observation:**

If we draw boxes around each loop and condition, they fit **neatly nested** — showing a clear control hierarchy.  
This makes the logic readable, reusable, and easy to modify (e.g., changing signal durations).

**5. Conclusion:**

The Smart Traffic Light Controller follows **structured programming** completely.  
Each operation (sequence, selection, iteration) is independent and well-organized — improving maintainability and reducing complexity.