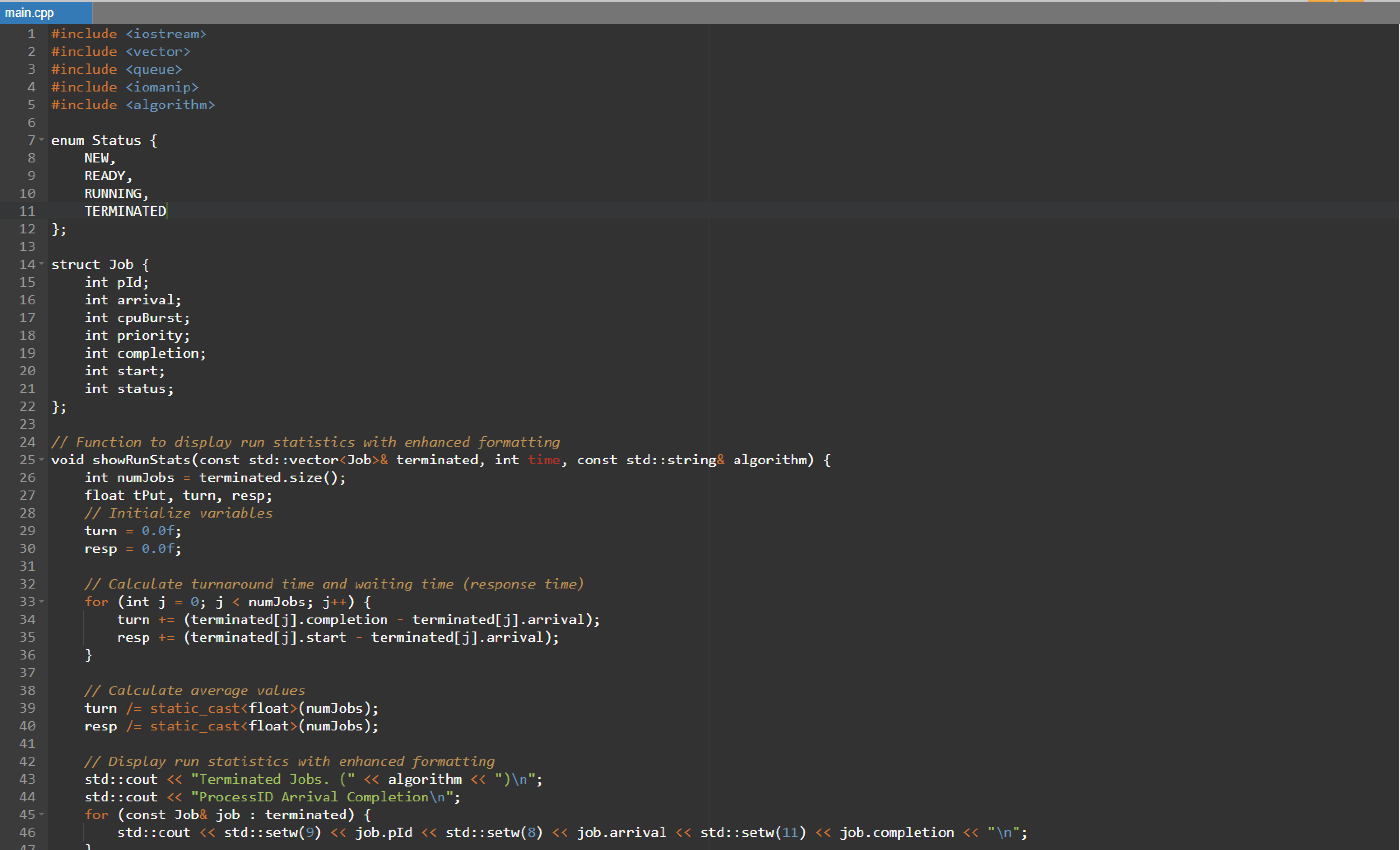
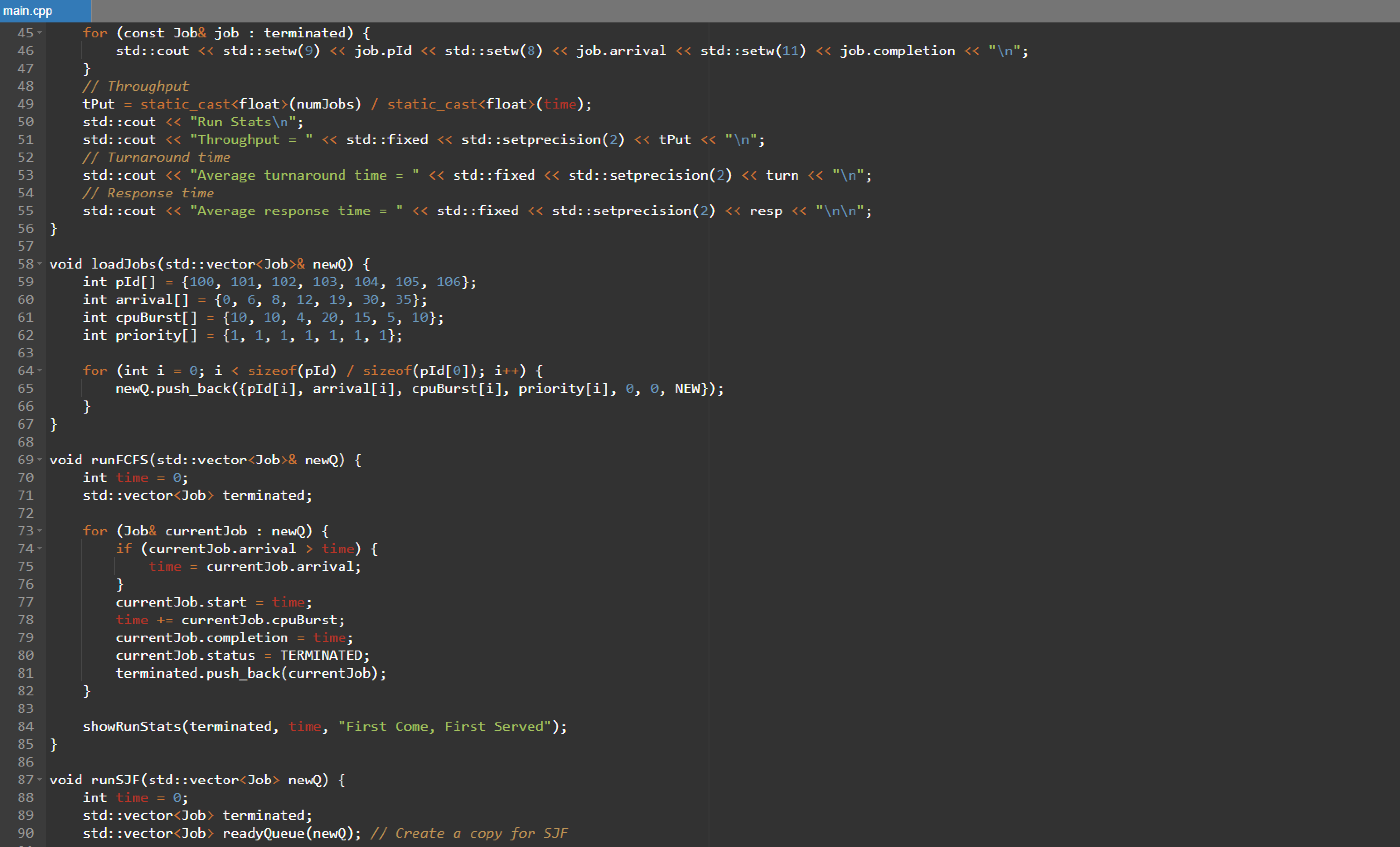
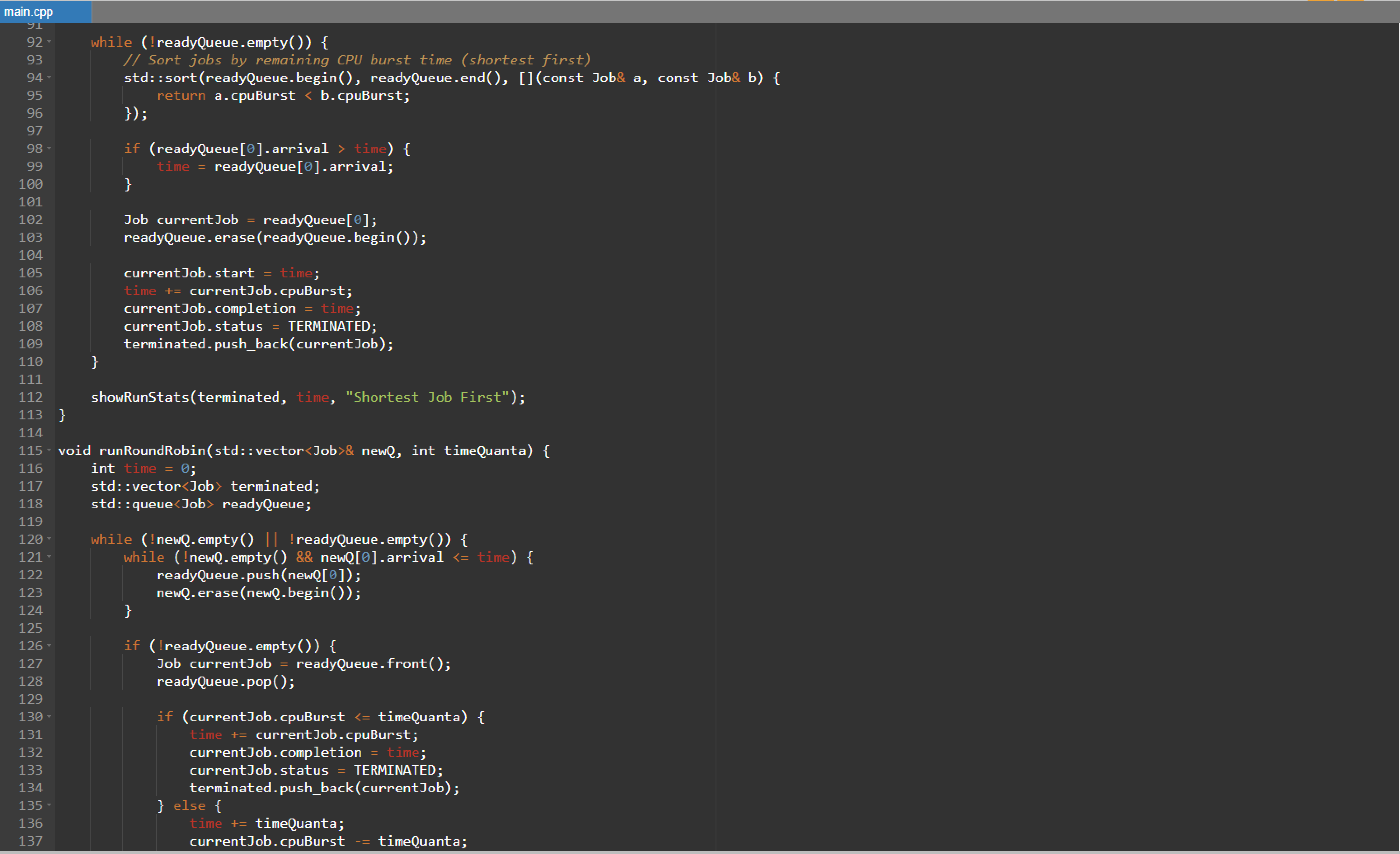


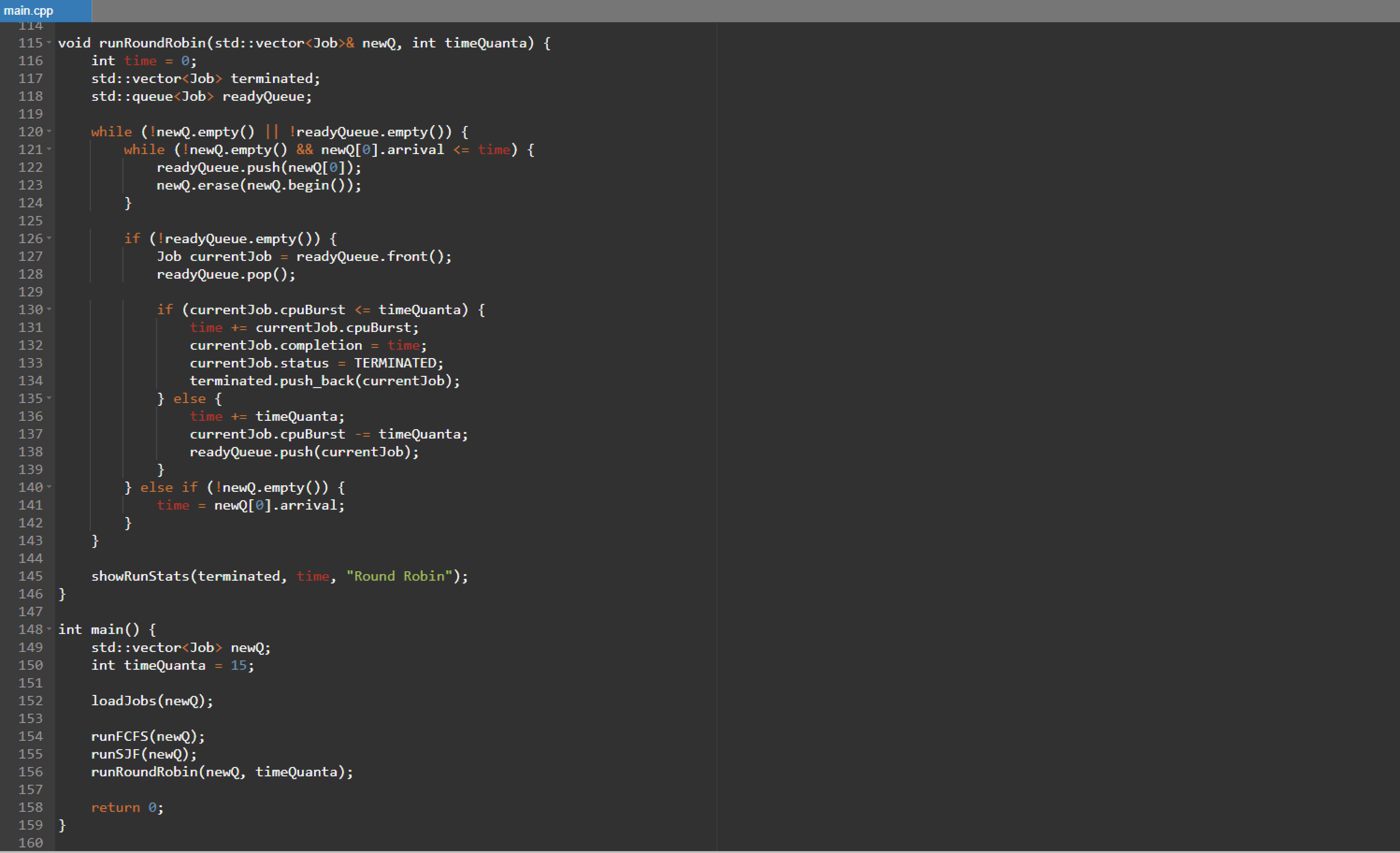
```cpp
#include <iostream>
#include <vector>
#include <queue>
#include <iomanip>
#include <algorithm>

enum Status {
    NEW,
    READY,
    RUNNING,
    TERMINATED
};

struct Job {
    int pId;
    int arrival;
    int cpuBurst;
    int priority;
    int completion;
    int start;
    int status;
};

// Function to display run statistics with enhanced formatting
void showRunStats(const std::vector<Job>& terminated, int time, const std::string& algorithm) {
    int numJobs = terminated.size();
    float tPut, turn, resp;
    // Initialize variables
    turn = 0.0f;
    resp = 0.0f;

    // Calculate turnaround time and waiting time (response time)
    for (int j = 0; j < numJobs; j++) {
        turn += (terminated[j].completion - terminated[j].arrival);
        resp += (terminated[j].start - terminated[j].arrival);
    }

    // Calculate average values
    turn /= static_cast<float>(numJobs);
    resp /= static_cast<float>(numJobs);

    // Display run statistics with enhanced formatting
    std::cout << "Terminated Jobs. (" << algorithm << ")\n";
    std::cout << "ProcessID Arrival Completion\n";
    for (const Job& job : terminated) {
        std::cout << std::setw(9) << job.pId << std::setw(8) << job.arrival << std::setw(11) << job.completion << "\n";
    }
```



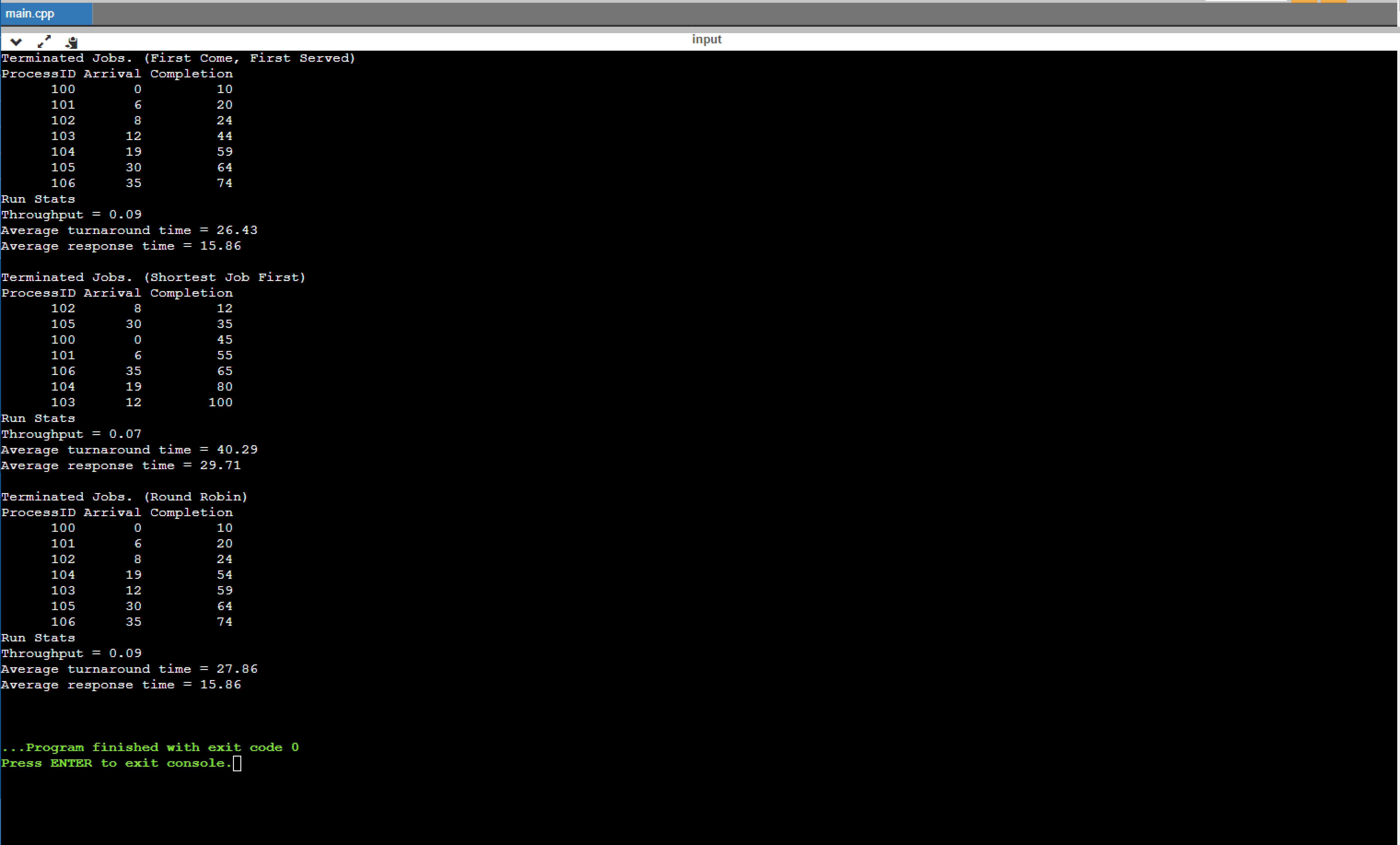
```cpp
    for (const Job& job : terminated) {
        std::cout << std::setw(9) << job.pId << std::setw(8) << job.arrival << std::setw(11) << job.completion << "\n";
    }
    // Throughput
    tPut = static_cast<float>(numJobs) / static_cast<float>(time);
    std::cout << "Run Stats\n";
    std::cout << "Throughput = " << std::fixed << std::setprecision(2) << tPut << "\n";
    // Turnaround time
    std::cout << "Average turnaround time = " << std::fixed << std::setprecision(2) << turn << "\n";
    // Response time
    std::cout << "Average response time = " << std::fixed << std::setprecision(2) << resp << "\n\n";
}

void loadJobs(std::vector<Job>& newQ) {
    int pId[] = {100, 101, 102, 103, 104, 105, 106};
    int arrival[] = {0, 6, 8, 12, 19, 30, 35};
    int cpuBurst[] = {10, 10, 4, 20, 15, 5, 10};
    int priority[] = {1, 1, 1, 1, 1, 1, 1};

    for (int i = 0; i < sizeof(pId) / sizeof(pId[0]); i++) {
        newQ.push_back({pId[i], arrival[i], cpuBurst[i], priority[i], 0, 0, NEW});
    }
}

void runFCFS(std::vector<Job>& newQ) {
    int time = 0;
    std::vector<Job> terminated;

    for (Job& currentJob : newQ) {
        if (currentJob.arrival > time) {
            time = currentJob.arrival;
        }
        currentJob.start = time;
        time += currentJob.cpuBurst;
        currentJob.completion = time;
        currentJob.status = TERMINATED;
        terminated.push_back(currentJob);
    }

    showRunStats(terminated, time, "First Come, First Served");
}

void runSJF(std::vector<Job> newQ) {
    int time = 0;
    std::vector<Job> terminated;
    std::vector<Job> readyQueue(newQ); // Create a copy for SJF
```



```cpp
    while (!readyQueue.empty()) {
        // Sort jobs by remaining CPU burst time (shortest first)
        std::sort(readyQueue.begin(), readyQueue.end(), [](const Job& a, const Job& b) {
            return a.cpuBurst < b.cpuBurst;
        });

        if (readyQueue[0].arrival > time) {
            time = readyQueue[0].arrival;
        }

        Job currentJob = readyQueue[0];
        readyQueue.erase(readyQueue.begin());

        currentJob.start = time;
        time += currentJob.cpuBurst;
        currentJob.completion = time;
        currentJob.status = TERMINATED;
        terminated.push_back(currentJob);
    }

    showRunStats(terminated, time, "Shortest Job First");
}

void runRoundRobin(std::vector<Job>& newQ, int timeQuanta) {
    int time = 0;
    std::vector<Job> terminated;
    std::queue<Job> readyQueue;

    while (!newQ.empty() || !readyQueue.empty()) {
        while (!newQ.empty() && newQ[0].arrival <= time) {
            readyQueue.push(newQ[0]);
            newQ.erase(newQ.begin());
        }

        if (!readyQueue.empty()) {
            Job currentJob = readyQueue.front();
            readyQueue.pop();

            if (currentJob.cpuBurst <= timeQuanta) {
                time += currentJob.cpuBurst;
                currentJob.completion = time;
                currentJob.status = TERMINATED;
                terminated.push_back(currentJob);
            } else {
                time += timeQuanta;
                currentJob.cpuBurst -= timeQuanta;
```


main.cpp

```cpp
void runRoundRobin(std::vector<Job>& newQ, int timeQuanta) {
    int time = 0;
    std::vector<Job> terminated;
    std::queue<Job> readyQueue;

    while (!newQ.empty() || !readyQueue.empty()) {
        while (!newQ.empty() && newQ[0].arrival <= time) {
            readyQueue.push(newQ[0]);
            newQ.erase(newQ.begin());
        }

        if (!readyQueue.empty()) {
            Job currentJob = readyQueue.front();
            readyQueue.pop();

            if (currentJob.cpuBurst <= timeQuanta) {
                time += currentJob.cpuBurst;
                currentJob.completion = time;
                currentJob.status = TERMINATED;
                terminated.push_back(currentJob);
            } else {
                time += timeQuanta;
                currentJob.cpuBurst -= timeQuanta;
                readyQueue.push(currentJob);
            }
        } else if (!newQ.empty()) {
            time = newQ[0].arrival;
        }
    }

    showRunStats(terminated, time, "Round Robin");
}

int main() {
    std::vector<Job> newQ;
    int timeQuanta = 15;

    loadJobs(newQ);

    runFCFS(newQ);
    runSJF(newQ);
    runRoundRobin(newQ, timeQuanta);

    return 0;
}
```



```
Terminated Jobs. (First Come, First Served)
ProcessID Arrival Completion
      100       0         10
      101       6         20
      102       8         24
      103      12         44
      104      19         59
      105      30         64
      106      35         74
Run Stats
Throughput = 0.09
Average turnaround time = 26.43
Average response time = 15.86

Terminated Jobs. (Shortest Job First)
ProcessID Arrival Completion
      102       8         12
      105      30         35
      100       0         45
      101       6         55
      106      35         65
      104      19         80
      103      12        100
Run Stats
Throughput = 0.07
Average turnaround time = 40.29
Average response time = 29.71

Terminated Jobs. (Round Robin)
ProcessID Arrival Completion
      100       0         10
      101       6         20
      102       8         24
      104      19         54
      103      12         59
      105      30         64
      106      35         74
Run Stats
Throughput = 0.09
Average turnaround time = 27.86
Average response time = 15.86



...Program finished with exit code 0
Press ENTER to exit console.
```