Code:

```
#include <iostream>
#include <vector>

using namespace std;

struct Process{
   int id;
   int burstTime;
   double waitingTime;
   double turnaroundTime;
};

void fcfsScheduling(vector<Process> &processes){
   int n = processes.size();
   int totalTime = 0;
   double totalWaitingTime = 0;
   double totalTurnaroundTime = 0;

   cout << "Process"<< "\t" << "Burst Time" << "\t" << "Waiting Time" << "\t" << "Turnaround
Time" << endl;

   for (int i = 0; i < n; i++){
      totalTime += processes[i].burstTime;
      processes[i].turnaroundTime = totalTime;
      processes[i].waitingTime = processes[i].turnaroundTime - processes[i].burstTime;

      cout << "P" << processes[i].id << "\t\t" << processes[i].burstTime << "\t\t\t" <<
processes[i].waitingTime << "\t\t\t\t" << processes[i].turnaroundTime << endl;

      totalWaitingTime += processes[i].waitingTime;
      totalTurnaroundTime += processes[i].turnaroundTime;
   }

   double averageWaitingTime = totalWaitingTime / n;
   double averageTurnaroundTime = totalTurnaroundTime / n;

   cout << "\nAverage Waiting Time: " << averageWaitingTime << endl;
   cout << "Average Turnaround Time: " << averageTurnaroundTime << endl;
}

int main(){
   int n;
   cout << "Enter the number of processes: ";
   cin >> n;
```

```
    vector<Process> processes(n);

    for (int i = 0; i < n; i++){
        processes[i].id = i + 1;
        cout << "Enter burst time for Process P" << processes[i].id << ": ";
        cin >> processes[i].burstTime;
    }

    fcfsScheduling(processes);

    return 0;
}
```

Output:

```
Enter the number of processes: 5
Enter burst time for Process P1: 3
Enter burst time for Process P2: 5
Enter burst time for Process P3: 2
Enter burst time for Process P4: 7
Enter burst time for Process P5: 4
Process Burst Time  Waiting Time    Turnaround Time
P1      3           0               3
P2      5           3               8
P3      2           8               10
P4      7           10              17
P5      4           17              21

Average Waiting Time: 7.6
Average Turnaround Time: 11.8
```

Code:

```
#include <bits/stdc++.h>

using namespace std;

struct Process {
    int id;
    int arrival_time;
    int burst_time;
    int remaining_time;
    int turnaround_time;
    int waiting_time;
};

void sjf(vector<Process>& processes) {
    int n = processes.size();
    int current_time = 0;
    int completed = 0;
    double total_turnaround_time = 0;
    double total_waiting_time = 0;

    while (completed < n) {
        int shortest_job = -1;
        int min_burst_time = INT_MAX;

        for (int i = 0; i < n; i++) {
            if (processes[i].arrival_time <= current_time &&
processes[i].remaining_time < min_burst_time && processes[i].remaining_time >
0) {
                shortest_job = i;
                min_burst_time = processes[i].remaining_time;
            }
        }

        if (shortest_job == -1) {
            current_time++;
        }
        else {
            processes[shortest_job].remaining_time--;
            current_time++;

            if (processes[shortest_job].remaining_time == 0) {
                completed++;
                processes[shortest_job].turnaround_time = current_time -
processes[shortest_job].arrival_time;
                processes[shortest_job].waiting_time =
processes[shortest_job].turnaround_time - processes[shortest_job].burst_time;
```

```
                total_turnaround_time +=
processes[shortest_job].turnaround_time;
                total_waiting_time += processes[shortest_job].waiting_time;
            }
        }
    }

    cout << "Process\tArrival Time\tBurst Time\tTurnaround Time\tWaiting Time"
<< endl;
    for (const Process& p : processes) {
        cout << p.id << "\t" << p.arrival_time << "\t\t" << p.burst_time <<
"\t\t" << p.turnaround_time << "\t\t" << p.waiting_time << endl;
    }

    double avg_turnaround_time = total_turnaround_time / n;
    double avg_waiting_time = total_waiting_time / n;
    cout << "Average Turnaround Time: " << avg_turnaround_time << endl;
    cout << "Average Waiting Time: " << avg_waiting_time << endl;
}

// Preemptive SJF
void srtf(vector<Process>& processes) {
    int n = processes.size();
    int current_time = 0;
    int completed = 0;
    double total_turnaround_time = 0;
    double total_waiting_time = 0;

    while (completed < n) {
        int shortest_job = -1;
        int min_remaining_time = INT_MAX;

        for (int i = 0; i < n; i++) {
            if (processes[i].arrival_time <= current_time &&
processes[i].remaining_time < min_remaining_time &&
processes[i].remaining_time > 0) {
                shortest_job = i;
                min_remaining_time = processes[i].remaining_time;
            }
        }

        if (shortest_job == -1) {
            current_time++;
        }
        else {
            processes[shortest_job].remaining_time--;
            current_time++;
```

```
            if (processes[shortest_job].remaining_time == 0) {
                completed++;
                processes[shortest_job].turnaround_time = current_time -
processes[shortest_job].arrival_time;
                processes[shortest_job].waiting_time =
processes[shortest_job].turnaround_time - processes[shortest_job].burst_time;
                total_turnaround_time +=
processes[shortest_job].turnaround_time;
                total_waiting_time += processes[shortest_job].waiting_time;
            }
        }
    }

    cout << "Process\tArrival Time\tBurst Time\tTurnaround Time\tWaiting Time"
<< endl;
    for (const Process& p : processes) {
        cout << p.id << "\t" << p.arrival_time << "\t\t" << p.burst_time <<
"\t\t" << p.turnaround_time << "\t\t" << p.waiting_time << endl;
    }

    double avg_turnaround_time = total_turnaround_time / n;
    double avg_waiting_time = total_waiting_time / n;
    cout << "Average Turnaround Time: " << avg_turnaround_time << endl;
    cout << "Average Waiting Time: " << avg_waiting_time << endl;
}

int main() {
    int n;
    cout << "Enter the number of processes: ";
    cin >> n;

    vector<Process> processes(n);

    for (int i = 0; i < n; i++) {
        processes[i].id = i + 1;
        cout << "Enter arrival time for Process " << i + 1 << ": ";
        cin >> processes[i].arrival_time;
        cout << "Enter burst time for Process " << i + 1 << ": ";
        cin >> processes[i].burst_time;
        processes[i].remaining_time = processes[i].burst_time;
    }

    sort(processes.begin(), processes.end(), [](const Process& a, const
Process& b) {
        return a.arrival_time < b.arrival_time;
    });

    cout << "\nSJF Scheduling:\n";
```

```
    sjf(processes);

    cout << "\nSRTF Scheduling:\n";
    srtf(processes);

    return 0;
}
```

Output:

```
Enter the number of processes: 4
Enter arrival time for Process 1: 0
Enter burst time for Process 1: 3
Enter arrival time for Process 2: 1
Enter burst time for Process 2: 2
Enter arrival time for Process 3: 2
Enter burst time for Process 3: 4
Enter arrival time for Process 4: 3
Enter burst time for Process 4: 1
SJF Scheduling:
Process Arrival Time    Burst Time  Turnaround Time Waiting Time
1       0               3           3               0
2       1               2           5               3
3       2               4           8               4
4       3               1           1               0
Average Turnaround Time: 4.25
Average Waiting Time: 1.75

SRTF Scheduling:
Process Arrival Time    Burst Time  Turnaround Time Waiting Time
1       0               3           3               0
2       1               2           5               3
3       2               4           8               4
4       3               1           1               0
Average Turnaround Time: 4.25
Average Waiting Time: 1.75
```