Họ và tên: Nguyễn Thị Minh Châu

MSSV: 21520645

Lớp: IT007.N25

BÁO CÁO THỰC HÀNH LAB 4 HỆ ĐIỀU HÀNH

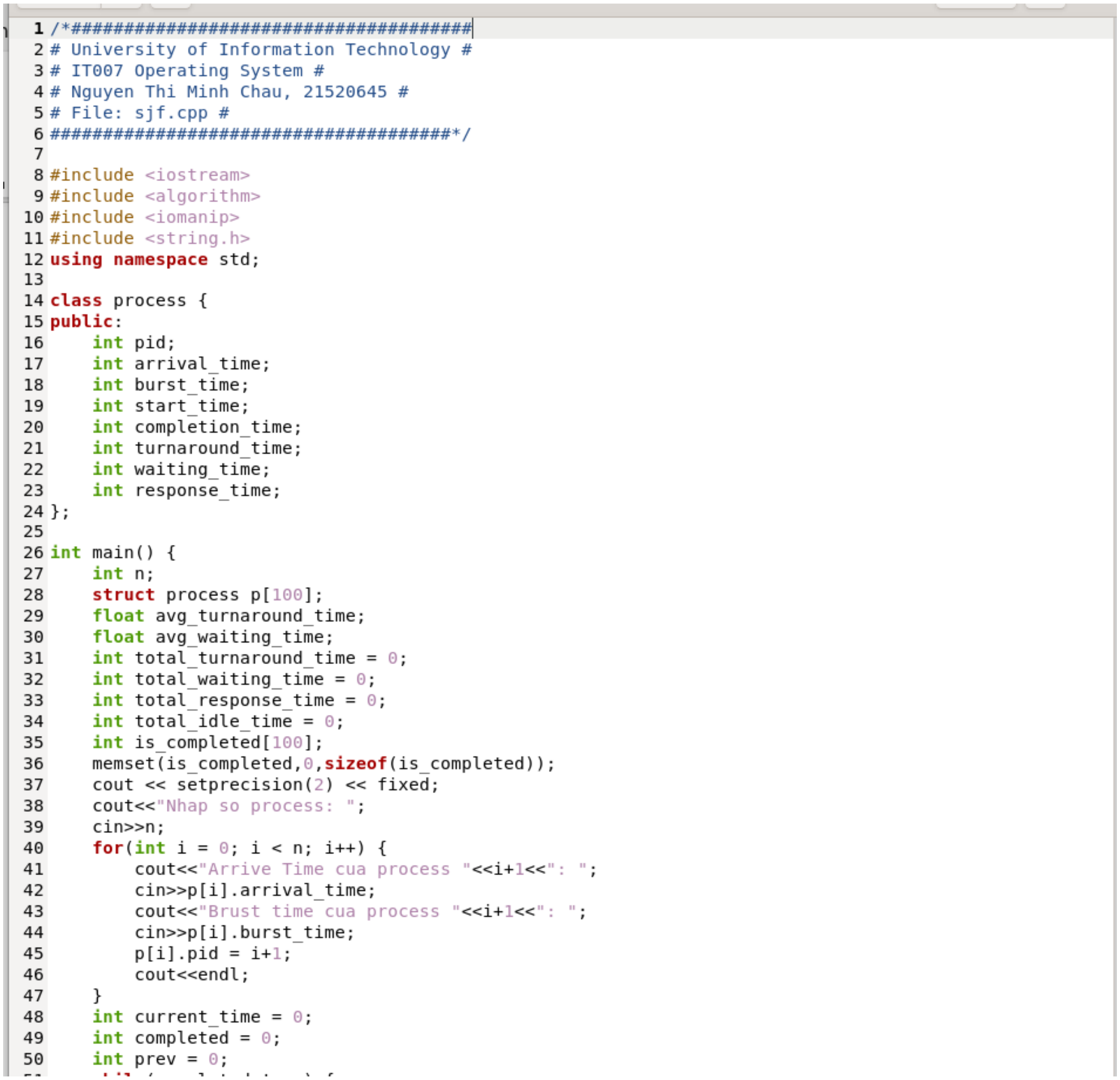
Bài 1:

Viết chương trình mô phỏng giải thuật SJF với các yêu cầu sau:

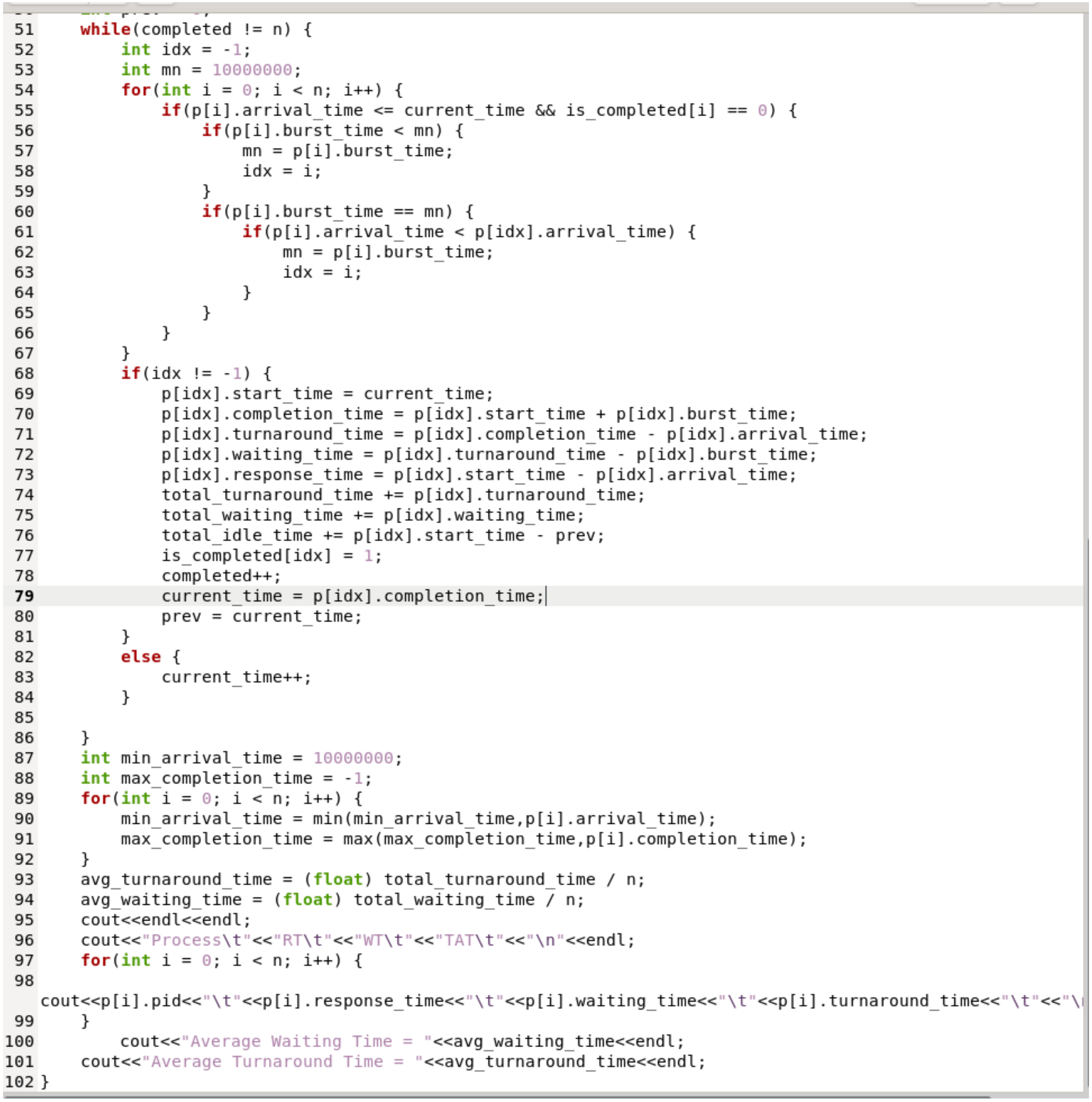
❖ Nhập số lượng process

❖ Nhập process name, arrival time, burst time

❖ In ra Process name, response time, waiting time, turnaround time, average waiting time, average turnaround time



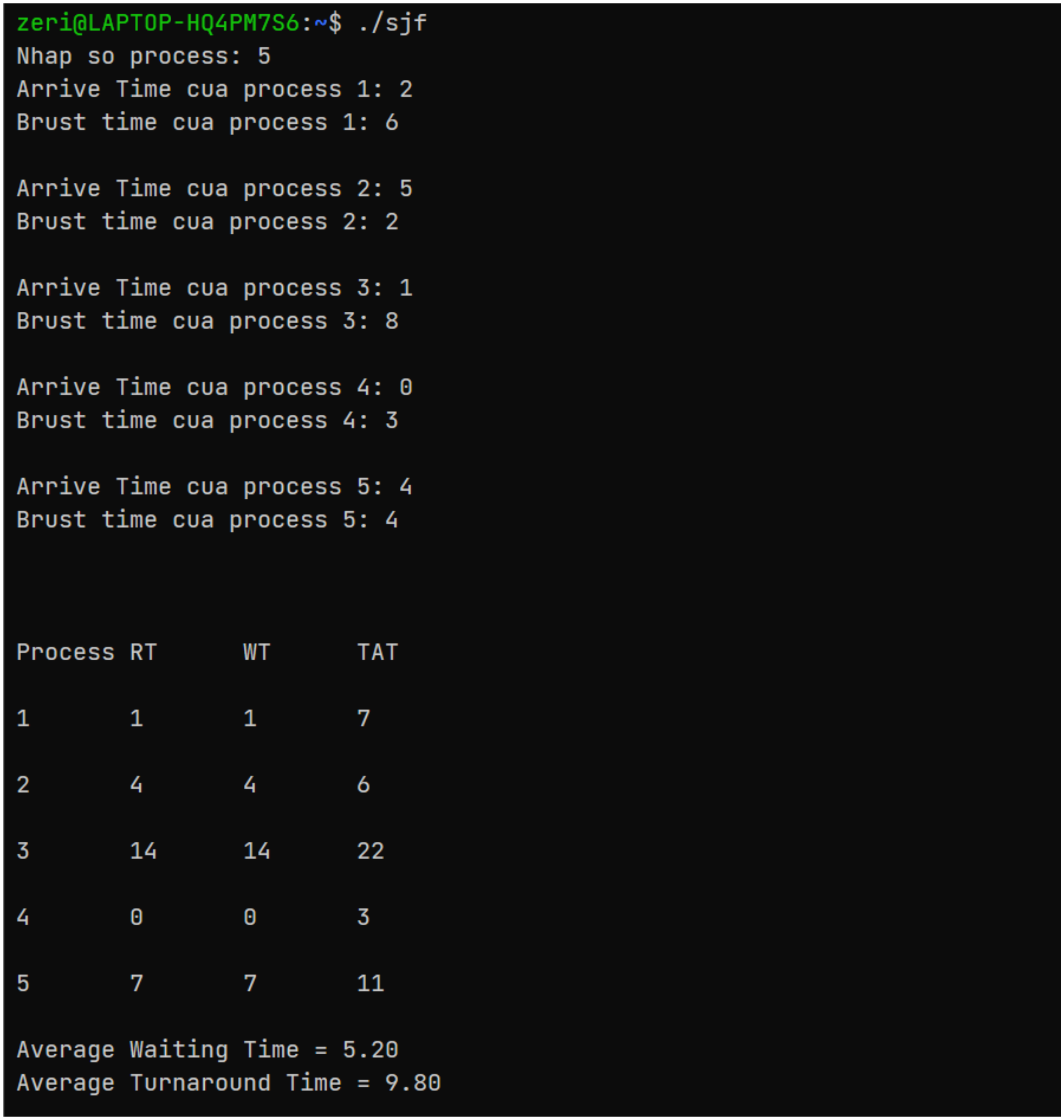
```cpp
/*######################################
# University of Information Technology #
# IT007 Operating System #
# Nguyen Thi Minh Chau, 21520645 #
# File: sjf.cpp #
######################################*/

#include <iostream>
#include <algorithm>
#include <iomanip>
#include <string.h>
using namespace std;

class process {
public:
    int pid;
    int arrival_time;
    int burst_time;
    int start_time;
    int completion_time;
    int turnaround_time;
    int waiting_time;
    int response_time;
};

int main() {
    int n;
    struct process p[100];
    float avg_turnaround_time;
    float avg_waiting_time;
    int total_turnaround_time = 0;
    int total_waiting_time = 0;
    int total_response_time = 0;
    int total_idle_time = 0;
    int is_completed[100];
    memset(is_completed,0,sizeof(is_completed));
    cout << setprecision(2) << fixed;
    cout<<"Nhap so process: ";
    cin>>n;
    for(int i = 0; i < n; i++) {
        cout<<"Arrive Time cua process "<<i+1<<": ";
        cin>>p[i].arrival_time;
        cout<<"Brust time cua process "<<i+1<<": ";
        cin>>p[i].burst_time;
        p[i].pid = i+1;
        cout<<endl;
    }
    int current_time = 0;
    int completed = 0;
    int prev = 0;
```



```cpp
    while(completed != n) {
        int idx = -1;
        int mn = 10000000;
        for(int i = 0; i < n; i++) {
            if(p[i].arrival_time <= current_time && is_completed[i] == 0) {
                if(p[i].burst_time < mn) {
                    mn = p[i].burst_time;
                    idx = i;
                }
                if(p[i].burst_time == mn) {
                    if(p[i].arrival_time < p[idx].arrival_time) {
                        mn = p[i].burst_time;
                        idx = i;
                    }
                }
            }
        }
        if(idx != -1) {
            p[idx].start_time = current_time;
            p[idx].completion_time = p[idx].start_time + p[idx].burst_time;
            p[idx].turnaround_time = p[idx].completion_time - p[idx].arrival_time;
            p[idx].waiting_time = p[idx].turnaround_time - p[idx].burst_time;
            p[idx].response_time = p[idx].start_time - p[idx].arrival_time;
            total_turnaround_time += p[idx].turnaround_time;
            total_waiting_time += p[idx].waiting_time;
            total_idle_time += p[idx].start_time - prev;
            is_completed[idx] = 1;
            completed++;
            current_time = p[idx].completion_time;
            prev = current_time;
        }
        else {
            current_time++;
        }

    }
    int min_arrival_time = 10000000;
    int max_completion_time = -1;
    for(int i = 0; i < n; i++) {
        min_arrival_time = min(min_arrival_time,p[i].arrival_time);
        max_completion_time = max(max_completion_time,p[i].completion_time);
    }
    avg_turnaround_time = (float) total_turnaround_time / n;
    avg_waiting_time = (float) total_waiting_time / n;
    cout<<endl<<endl;
    cout<<"Process\t"<<"RT\t"<<"WT\t"<<"TAT\t"<<"\n"<<endl;
    for(int i = 0; i < n; i++) {
cout<<p[i].pid<<"\t"<<p[i].response_time<<"\t"<<p[i].waiting_time<<"\t"<<p[i].turnaround_time<<"\t"<<"\
    }
        cout<<"Average Waiting Time = "<<avg_waiting_time<<endl;
    cout<<"Average Turnaround Time = "<<avg_turnaround_time<<endl;
}
```

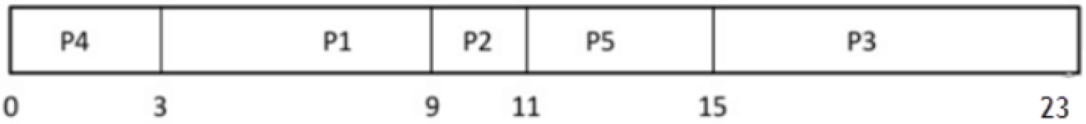
Hình 1: Code của chương trình sjf.cpp

Dưới đây là kết quả chạy:



```
zeri@LAPTOP-HQ4PM7S6:~$ ./sjf
Nhap so process: 5
Arrive Time cua process 1: 2
Brust time cua process 1: 6

Arrive Time cua process 2: 5
Brust time cua process 2: 2

Arrive Time cua process 3: 1
Brust time cua process 3: 8

Arrive Time cua process 4: 0
Brust time cua process 4: 3

Arrive Time cua process 5: 4
Brust time cua process 5: 4



Process RT      WT      TAT

1       1       1       7

2       4       4       6

3       14      14      22

4       0       0       3

5       7       7       11

Average Waiting Time = 5.20
Average Turnaround Time = 9.80
```

Thử lại bằng tay ta có biểu đồ như bên dưới:



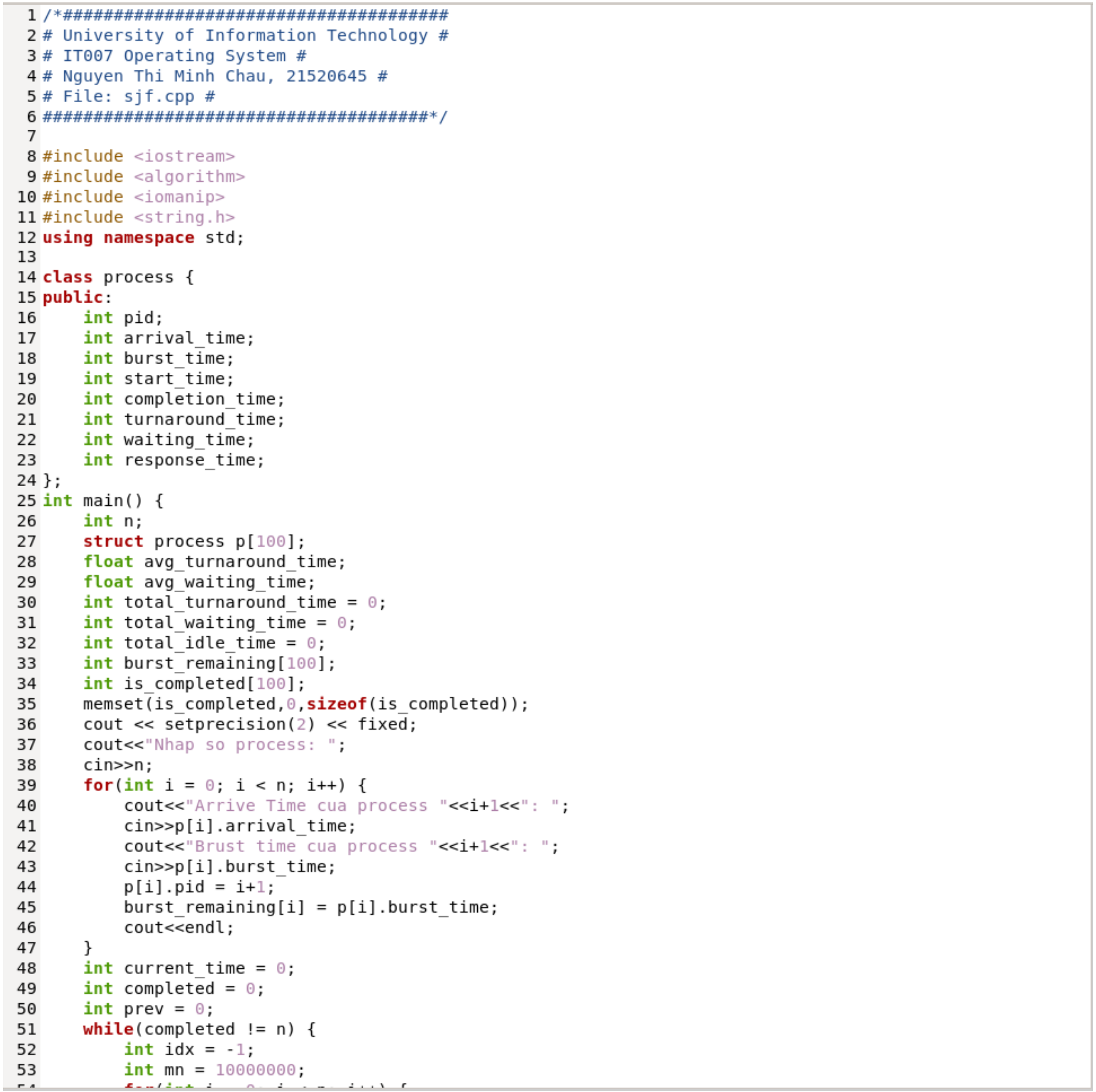
Bài 2:

Viết chương trình mô phỏng giải thuật SRT với các yêu cầu sau:

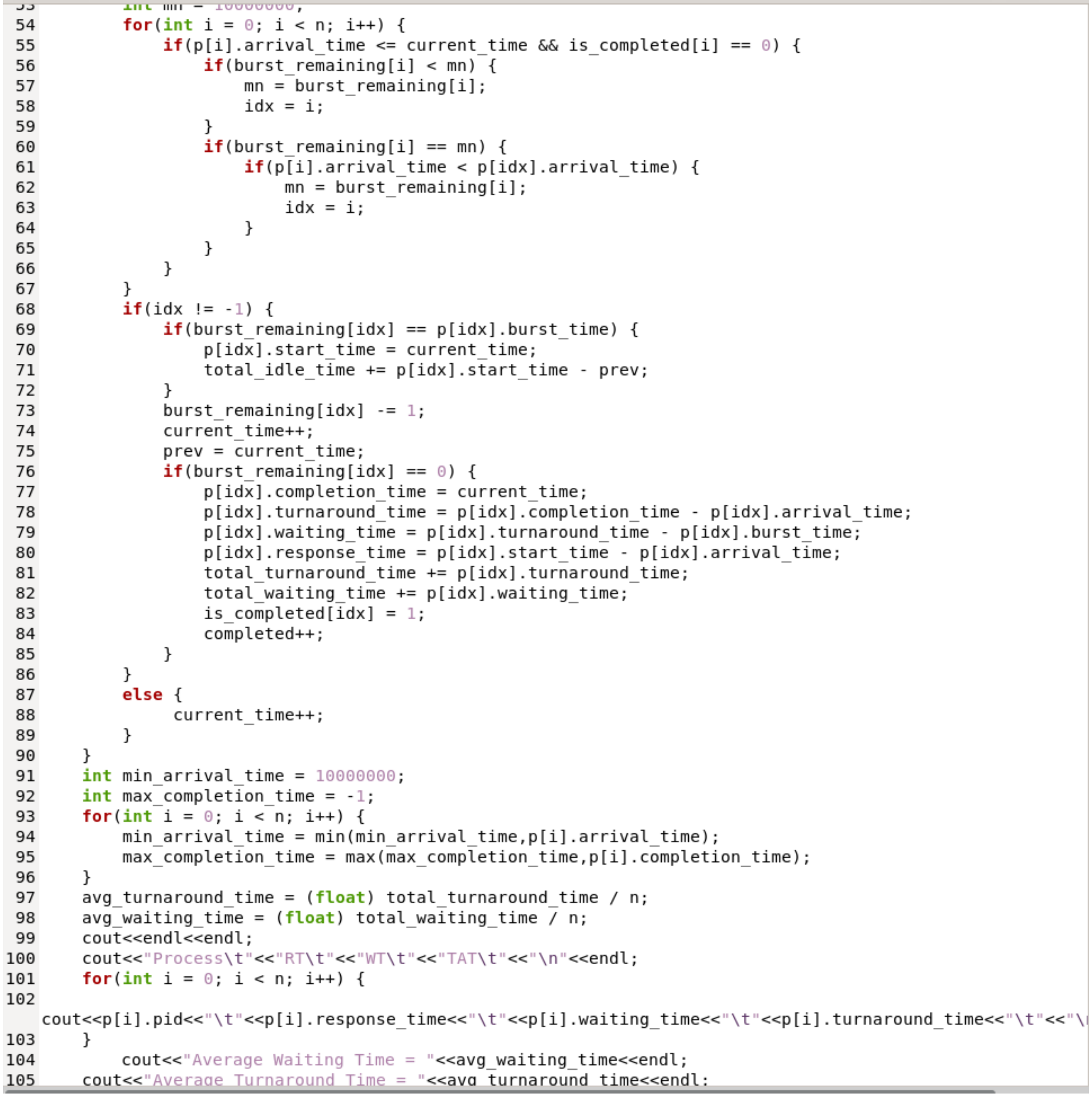
❖ Nhập số lượng process

❖ Nhập process name, arrival time, burst time 13

❖ In ra Process name, response time, waiting time, turnaround time, average waiting time, average turnaround time



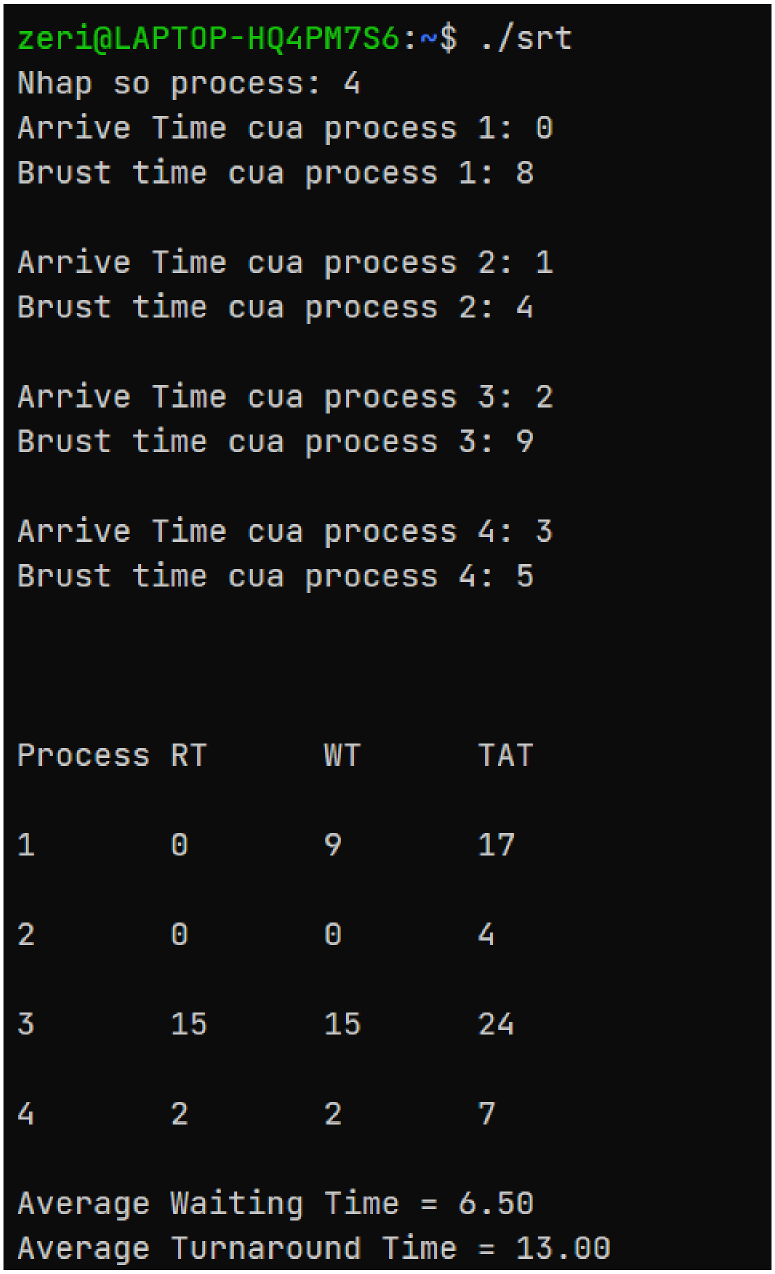
```
/*#######################################
# University of Information Technology #
# IT007 Operating System #
# Nguyen Thi Minh Chau, 21520645 #
# File: sjf.cpp #
#######################################*/

#include <iostream>
#include <algorithm>
#include <iomanip>
#include <string.h>
using namespace std;

class process {
public:
    int pid;
    int arrival_time;
    int burst_time;
    int start_time;
    int completion_time;
    int turnaround_time;
    int waiting_time;
    int response_time;
};
int main() {
    int n;
    struct process p[100];
    float avg_turnaround_time;
    float avg_waiting_time;
    int total_turnaround_time = 0;
    int total_waiting_time = 0;
    int total_idle_time = 0;
    int burst_remaining[100];
    int is_completed[100];
    memset(is_completed,0,sizeof(is_completed));
    cout << setprecision(2) << fixed;
    cout<<"Nhap so process: ";
    cin>>n;
    for(int i = 0; i < n; i++) {
        cout<<"Arrive Time cua process "<<i+1<<": ";
        cin>>p[i].arrival_time;
        cout<<"Brust time cua process "<<i+1<<": ";
        cin>>p[i].burst_time;
        p[i].pid = i+1;
        burst_remaining[i] = p[i].burst_time;
        cout<<endl;
    }
    int current_time = 0;
    int completed = 0;
    int prev = 0;
    while(completed != n) {
        int idx = -1;
        int mn = 10000000;
        for(int i = 0; i < n; i++) {
```



```
        int mn = 10000000;
        for(int i = 0; i < n; i++) {
            if(p[i].arrival_time <= current_time && is_completed[i] == 0) {
                if(burst_remaining[i] < mn) {
                    mn = burst_remaining[i];
                    idx = i;
                }
                if(burst_remaining[i] == mn) {
                    if(p[i].arrival_time < p[idx].arrival_time) {
                        mn = burst_remaining[i];
                        idx = i;
                    }
                }
            }
        }
        if(idx != -1) {
            if(burst_remaining[idx] == p[idx].burst_time) {
                p[idx].start_time = current_time;
                total_idle_time += p[idx].start_time - prev;
            }
            burst_remaining[idx] -= 1;
            current_time++;
            prev = current_time;
            if(burst_remaining[idx] == 0) {
                p[idx].completion_time = current_time;
                p[idx].turnaround_time = p[idx].completion_time - p[idx].arrival_time;
                p[idx].waiting_time = p[idx].turnaround_time - p[idx].burst_time;
                p[idx].response_time = p[idx].start_time - p[idx].arrival_time;
                total_turnaround_time += p[idx].turnaround_time;
                total_waiting_time += p[idx].waiting_time;
                is_completed[idx] = 1;
                completed++;
            }
        }
        else {
             current_time++;
        }
    }
    int min_arrival_time = 10000000;
    int max_completion_time = -1;
    for(int i = 0; i < n; i++) {
        min_arrival_time = min(min_arrival_time,p[i].arrival_time);
        max_completion_time = max(max_completion_time,p[i].completion_time);
    }
    avg_turnaround_time = (float) total_turnaround_time / n;
    avg_waiting_time = (float) total_waiting_time / n;
    cout<<endl<<endl;
    cout<<"Process\t"<<"RT\t"<<"WT\t"<<"TAT\t"<<"\n"<<endl;
    for(int i = 0; i < n; i++) {

cout<<p[i].pid<<"\t"<<p[i].response_time<<"\t"<<p[i].waiting_time<<"\t"<<p[i].turnaround_time<<"\t"<<"\
    }
        cout<<"Average Waiting Time = "<<avg_waiting_time<<endl;
    cout<<"Average Turnaround Time = "<<avg_turnaround_time<<endl;
```

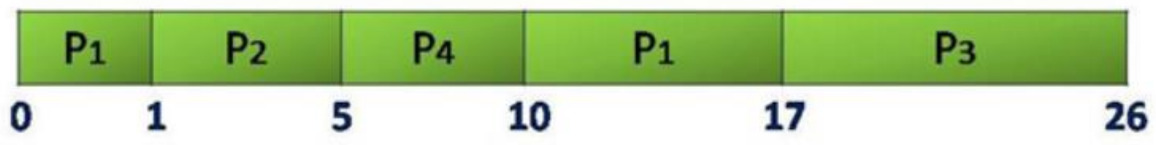
Hình 2: Code của chương trình srt.cpp

Kết quả sau khi chạy chương trình:



```
zeri@LAPTOP-HQ4PM7S6:~$ ./srt
Nhap so process: 4
Arrive Time cua process 1: 0
Brust time cua process 1: 8

Arrive Time cua process 2: 1
Brust time cua process 2: 4

Arrive Time cua process 3: 2
Brust time cua process 3: 9

Arrive Time cua process 4: 3
Brust time cua process 4: 5



Process RT      WT      TAT

1       0       9       17

2       0       0       4

3       15      15      24

4       2       2       7

Average Waiting Time = 6.50
Average Turnaround Time = 13.00
```

Thử lại kết quả bằng tay ta có:



Bài 3:

Viết chương trình mô phỏng giải thuật RR với các yêu cầu sau (giả sử tất cả các tiến trình đều có arrival time là 0):

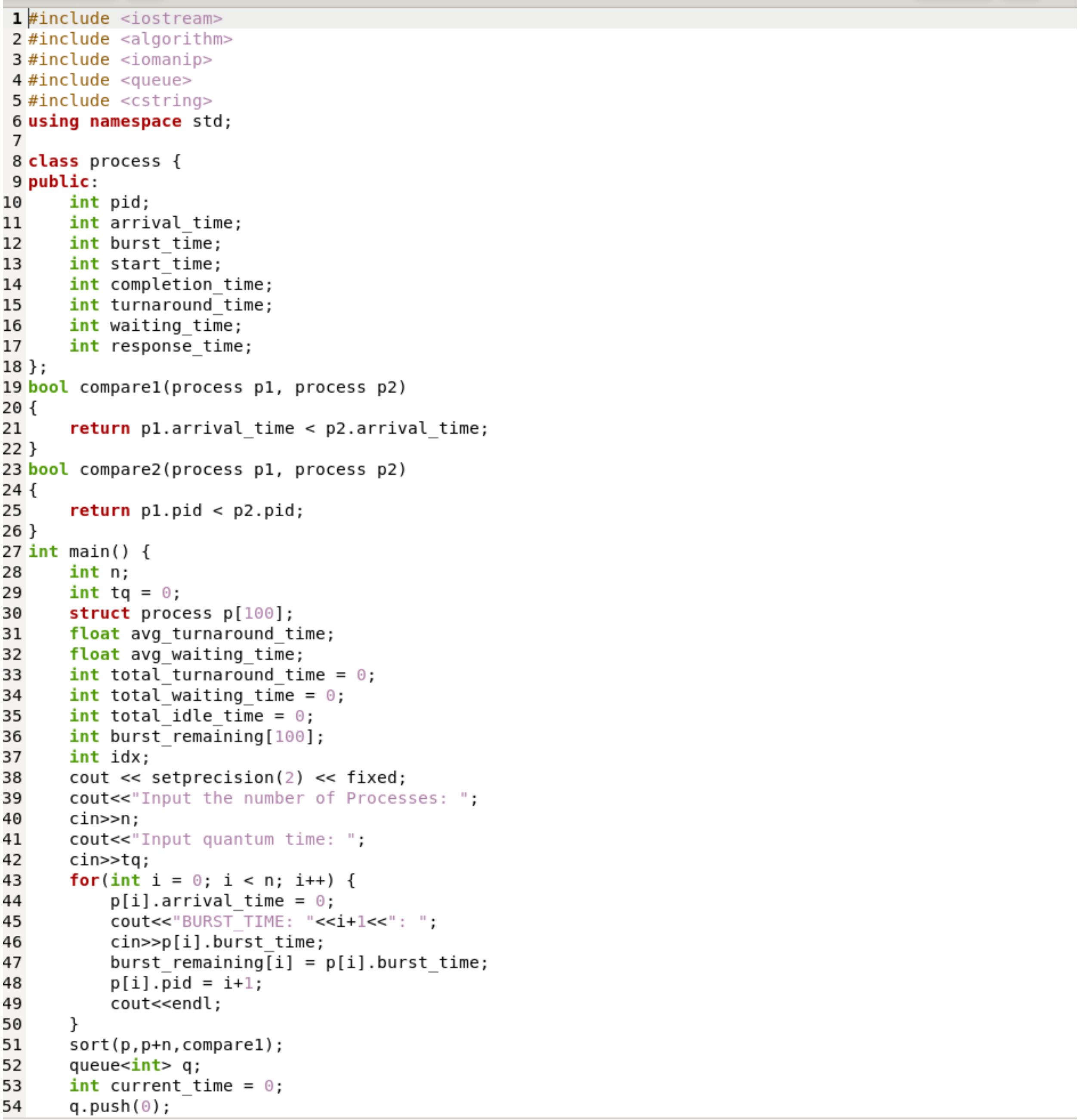
❖ Nhập số process

❖ Nhập quantum time

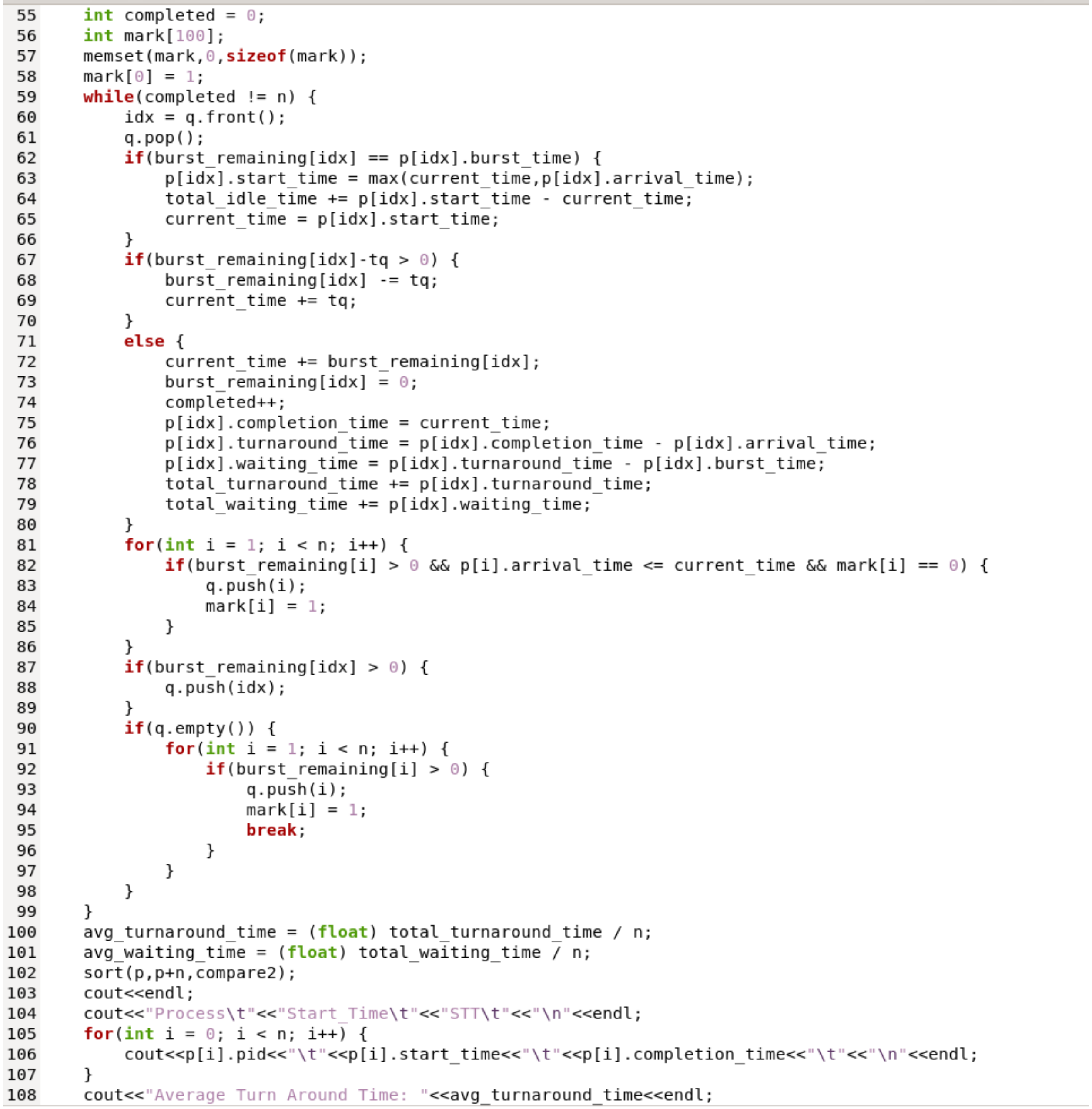
❖ Nhập process name, burst time

❖ In ra Gantt chart với các thông số: process name, start processor time, stop processor time

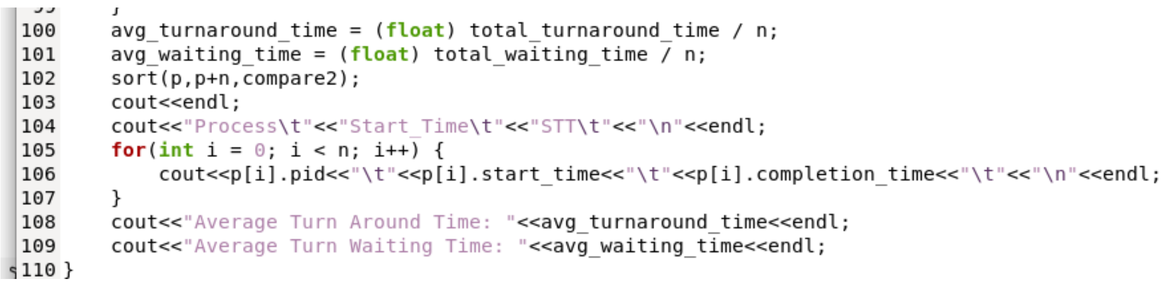
❖ In ra average waiting time và average turnaround time



```cpp
#include <iostream>
#include <algorithm>
#include <iomanip>
#include <queue>
#include <cstring>
using namespace std;

class process {
public:
    int pid;
    int arrival_time;
    int burst_time;
    int start_time;
    int completion_time;
    int turnaround_time;
    int waiting_time;
    int response_time;
};
bool compare1(process p1, process p2)
{
    return p1.arrival_time < p2.arrival_time;
}
bool compare2(process p1, process p2)
{
    return p1.pid < p2.pid;
}
int main() {
    int n;
    int tq = 0;
    struct process p[100];
    float avg_turnaround_time;
    float avg_waiting_time;
    int total_turnaround_time = 0;
    int total_waiting_time = 0;
    int total_idle_time = 0;
    int burst_remaining[100];
    int idx;
    cout << setprecision(2) << fixed;
    cout<<"Input the number of Processes: ";
    cin>>n;
    cout<<"Input quantum time: ";
    cin>>tq;
    for(int i = 0; i < n; i++) {
        p[i].arrival_time = 0;
        cout<<"BURST_TIME: "<<i+1<<": ";
        cin>>p[i].burst_time;
        burst_remaining[i] = p[i].burst_time;
        p[i].pid = i+1;
        cout<<endl;
    }
    sort(p,p+n,compare1);
    queue<int> q;
    int current_time = 0;
    q.push(0);
```



```
    int completed = 0;
    int mark[100];
    memset(mark,0,sizeof(mark));
    mark[0] = 1;
    while(completed != n) {
        idx = q.front();
        q.pop();
        if(burst_remaining[idx] == p[idx].burst_time) {
            p[idx].start_time = max(current_time,p[idx].arrival_time);
            total_idle_time += p[idx].start_time - current_time;
            current_time = p[idx].start_time;
        }
        if(burst_remaining[idx]-tq > 0) {
            burst_remaining[idx] -= tq;
            current_time += tq;
        }
        else {
            current_time += burst_remaining[idx];
            burst_remaining[idx] = 0;
            completed++;
            p[idx].completion_time = current_time;
            p[idx].turnaround_time = p[idx].completion_time - p[idx].arrival_time;
            p[idx].waiting_time = p[idx].turnaround_time - p[idx].burst_time;
            total_turnaround_time += p[idx].turnaround_time;
            total_waiting_time += p[idx].waiting_time;
        }
        for(int i = 1; i < n; i++) {
            if(burst_remaining[i] > 0 && p[i].arrival_time <= current_time && mark[i] == 0) {
                q.push(i);
                mark[i] = 1;
            }
        }
        if(burst_remaining[idx] > 0) {
            q.push(idx);
        }
        if(q.empty()) {
            for(int i = 1; i < n; i++) {
                if(burst_remaining[i] > 0) {
                    q.push(i);
                    mark[i] = 1;
                    break;
                }
            }
        }
    }
    avg_turnaround_time = (float) total_turnaround_time / n;
    avg_waiting_time = (float) total_waiting_time / n;
    sort(p,p+n,compare2);
    cout<<endl;
    cout<<"Process\t"<<"Start_Time\t"<<"STT\t"<<"\n"<<endl;
    for(int i = 0; i < n; i++) {
        cout<<p[i].pid<<"\t"<<p[i].start_time<<"\t"<<p[i].completion_time<<"\t"<<"\n"<<endl;
    }
    cout<<"Average Turn Around Time: "<<avg_turnaround_time<<endl;
```

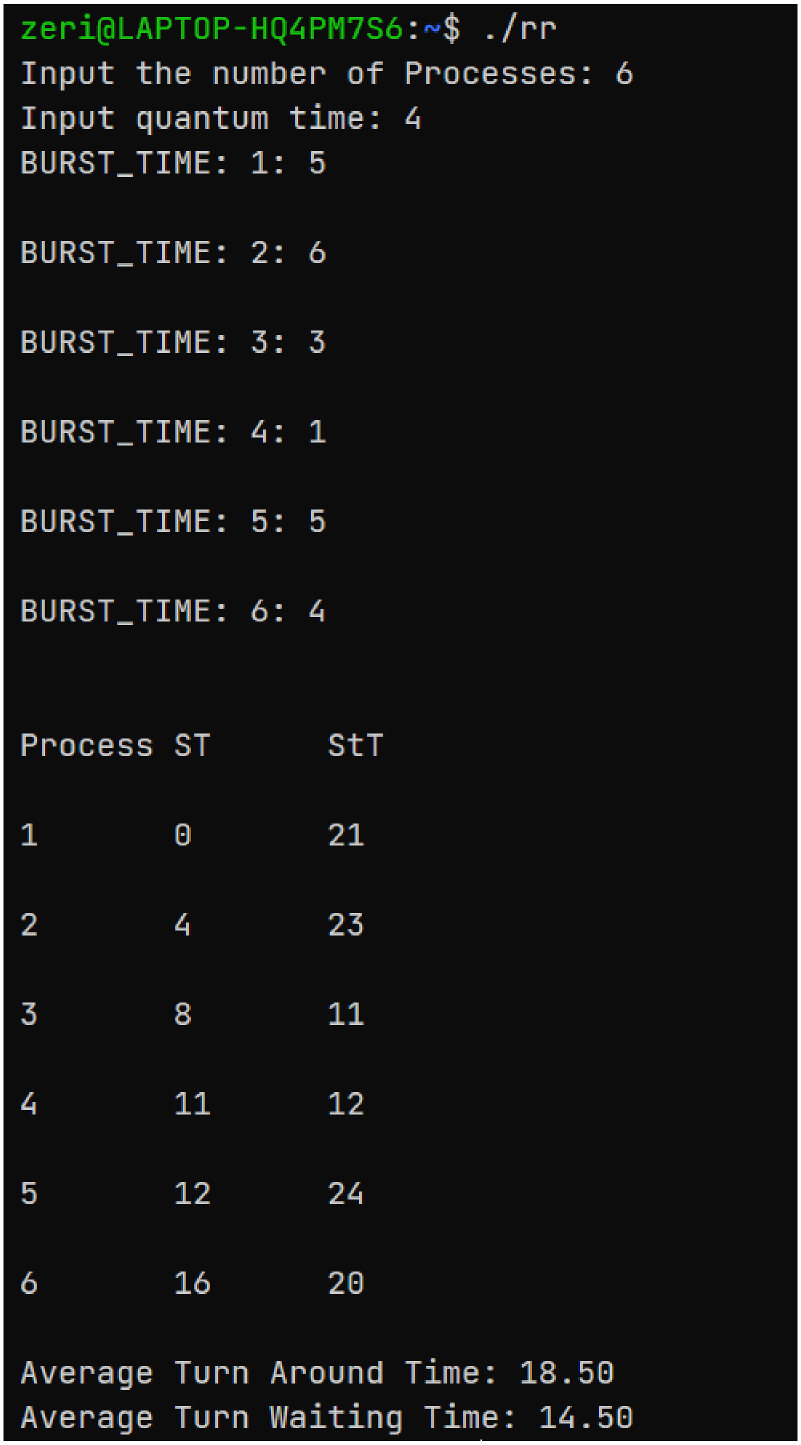


```
    avg_turnaround_time = (float) total_turnaround_time / n;
    avg_waiting_time = (float) total_waiting_time / n;
    sort(p,p+n,compare2);
    cout<<endl;
    cout<<"Process\t"<<"Start_Time\t"<<"STT\t"<<"\n"<<endl;
    for(int i = 0; i < n; i++) {
        cout<<p[i].pid<<"\t"<<p[i].start_time<<"\t"<<p[i].completion_time<<"\t"<<"\n"<<endl;
    }
    cout<<"Average Turn Around Time: "<<avg_turnaround_time<<endl;
```

```
    cout<<"Average Turn Waiting Time: "<<avg_waiting_time<<endl;
}
```

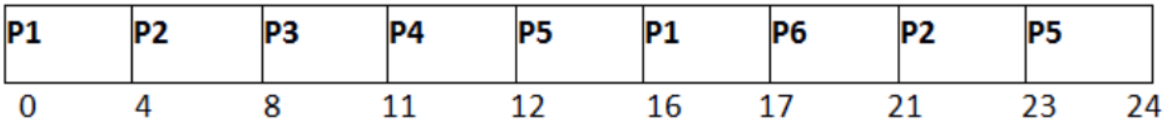
Hình 3: Code của chương trình rr.cpp

Đây là kết quả chạy chương trình:



```
zeri@LAPTOP-HQ4PM7S6:~$ ./rr
Input the number of Processes: 6
Input quantum time: 4
BURST_TIME: 1: 5

BURST_TIME: 2: 6

BURST_TIME: 3: 3

BURST_TIME: 4: 1

BURST_TIME: 5: 5

BURST_TIME: 6: 4


Process ST      StT

1       0       21

2       4       23

3       8       11

4       11      12

5       12      24

6       16      20

Average Turn Around Time: 18.50
Average Turn Waiting Time: 14.50
```

Thử lại kết quả bằng tay:


P1
P2
P3
P4
P5
P1
P6
P2
P5
0
4
8
11
12
16
17
21
23
24