Họ và tên : Trần Đình Khang

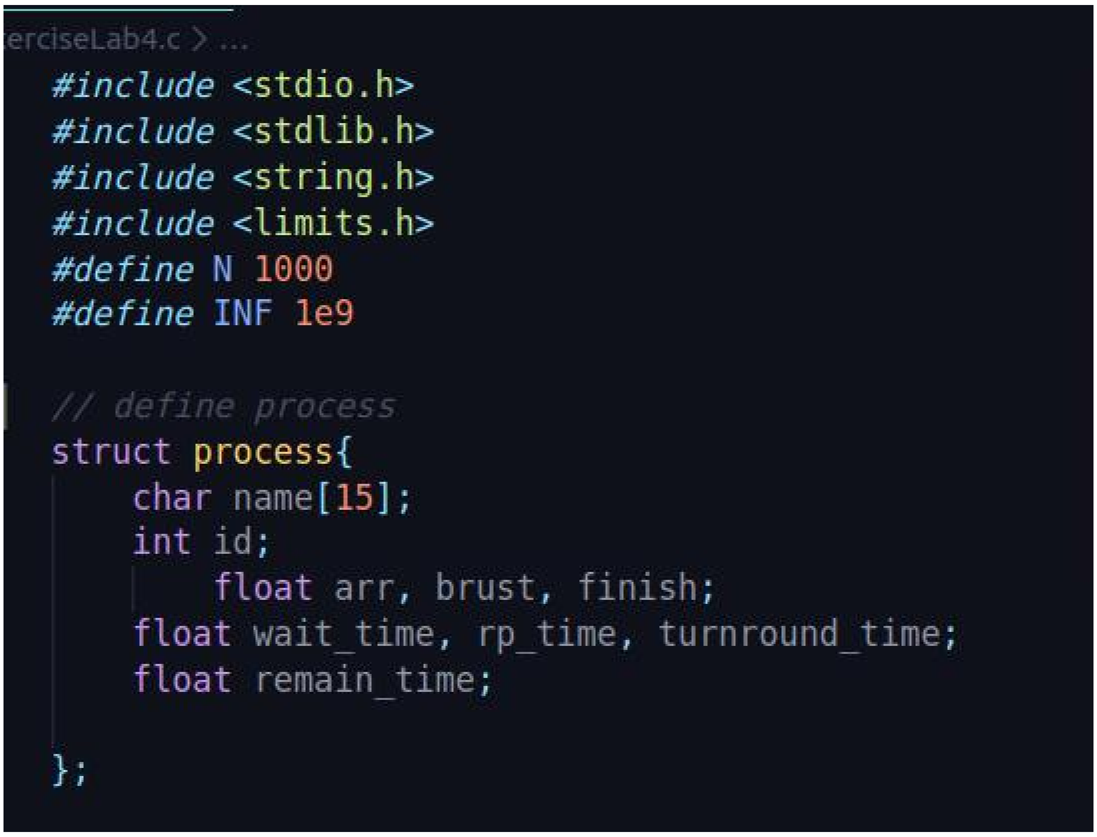
MSSV: 18520072

Mã Lớp: IT007.K21.KHTN

# Bài thực hành Lab04

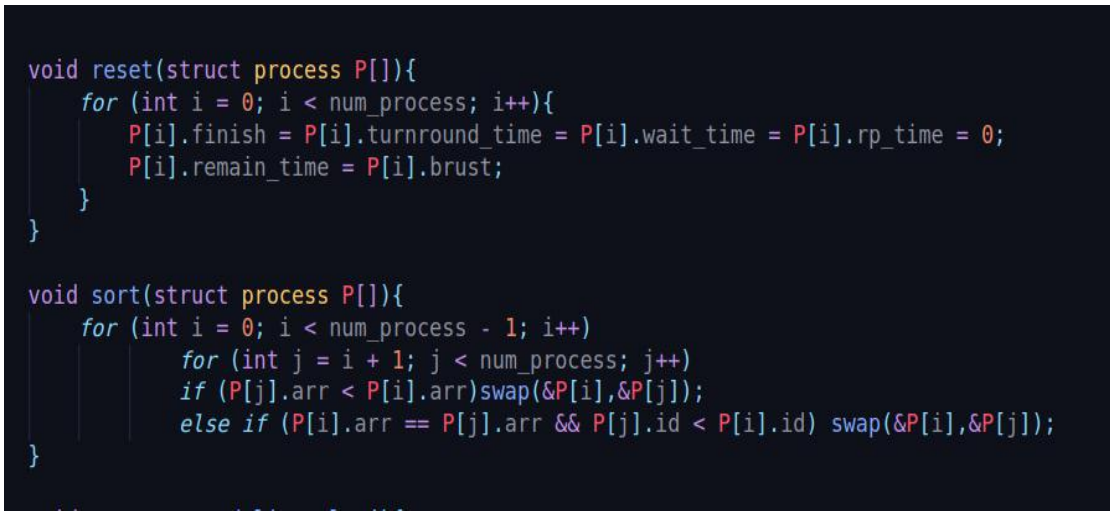
## 1. Code:

**- Tạo struct process:**



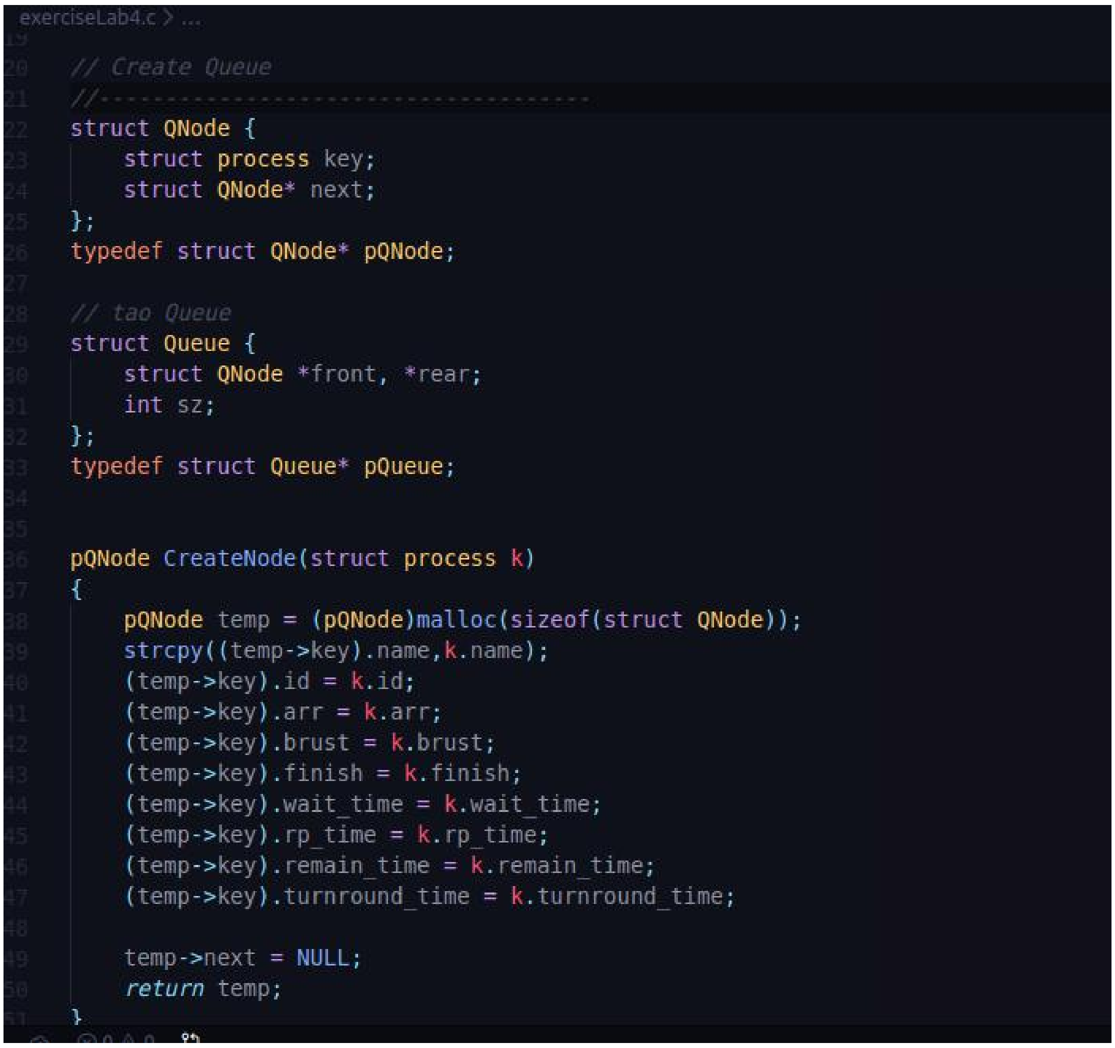
```
erciseLab4.c > ...
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <limits.h>
#define N 1000
#define INF 1e9

// define process
struct process{
    char name[15];
    int id;
        float arr, brust, finish;
    float wait_time, rp_time, turnround_time;
    float remain_time;

};
```

**- Tạo hàm sort các process theo arrive time và hàm reset để khởi tạo lại các tiến trình:**



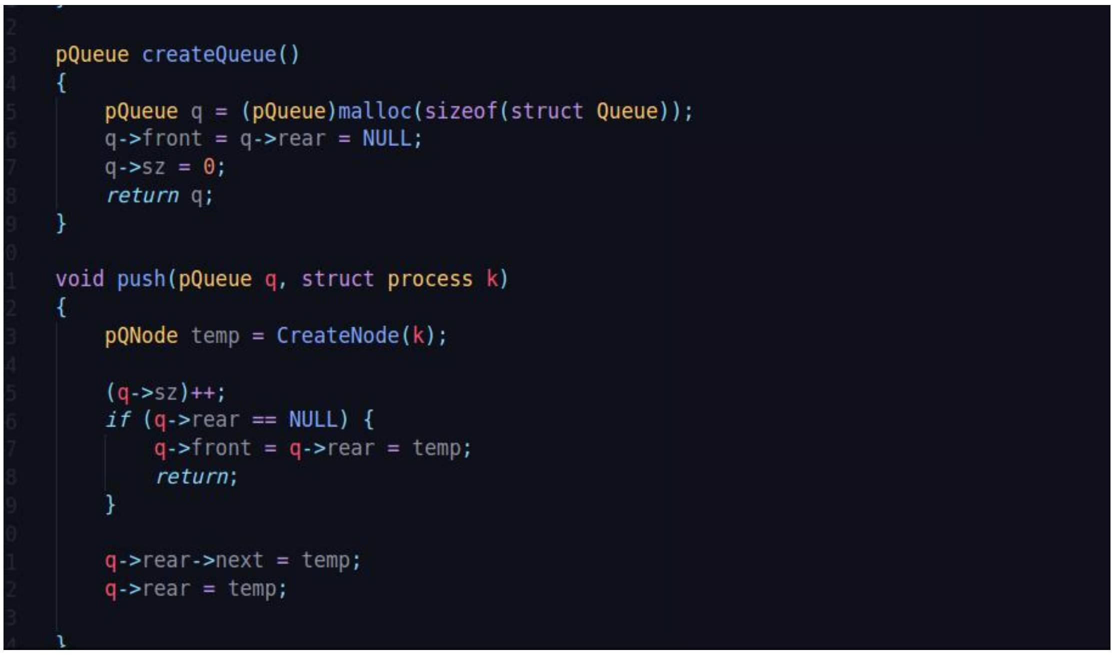
```c
void reset(struct process P[]){
    for (int i = 0; i < num_process; i++){
        P[i].finish = P[i].turnround_time = P[i].wait_time = P[i].rp_time = 0;
        P[i].remain_time = P[i].brust;
    }
}

void sort(struct process P[]){
    for (int i = 0; i < num_process - 1; i++)
        for (int j = i + 1; j < num_process; j++)
        if (P[j].arr < P[i].arr)swap(&P[i],&P[j]);
        else if (P[i].arr == P[j].arr && P[j].id < P[i].id) swap(&P[i],&P[j]);
}
```

**- Tạo Queue và các hàm phương thức dùng trong thuật toán RR:**

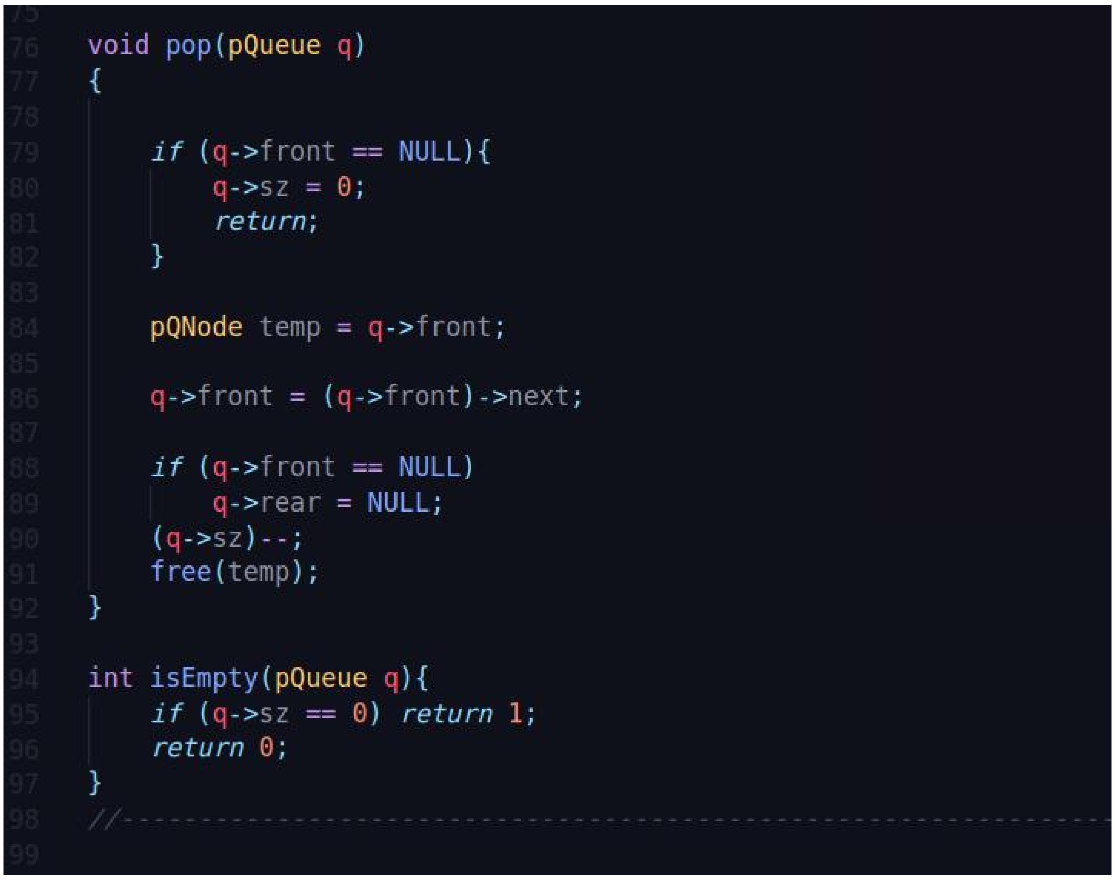


exerciseLab4.c > ...

```c
// Create Queue
//-----------------------------------
struct QNode {
    struct process key;
    struct QNode* next;
};
typedef struct QNode* pQNode;

// tao Queue
struct Queue {
    struct QNode *front, *rear;
    int sz;
};
typedef struct Queue* pQueue;


pQNode CreateNode(struct process k)
{
    pQNode temp = (pQNode)malloc(sizeof(struct QNode));
    strcpy((temp->key).name,k.name);
    (temp->key).id = k.id;
    (temp->key).arr = k.arr;
    (temp->key).brust = k.brust;
    (temp->key).finish = k.finish;
    (temp->key).wait_time = k.wait_time;
    (temp->key).rp_time = k.rp_time;
    (temp->key).remain_time = k.remain_time;
    (temp->key).turnround_time = k.turnround_time;

    temp->next = NULL;
    return temp;
}
```

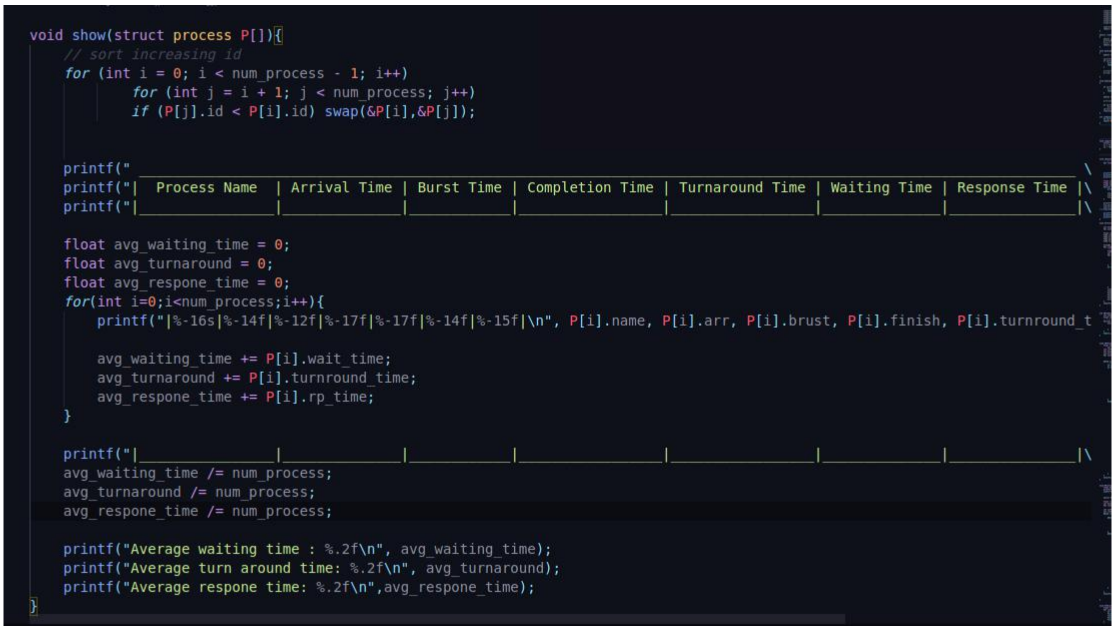


```
pQueue createQueue()
{
    pQueue q = (pQueue)malloc(sizeof(struct Queue));
    q->front = q->rear = NULL;
    q->sz = 0;
    return q;
}

void push(pQueue q, struct process k)
{
    pQNode temp = CreateNode(k);

    (q->sz)++;
    if (q->rear == NULL) {
        q->front = q->rear = temp;
        return;
    }

    q->rear->next = temp;
    q->rear = temp;

}
```



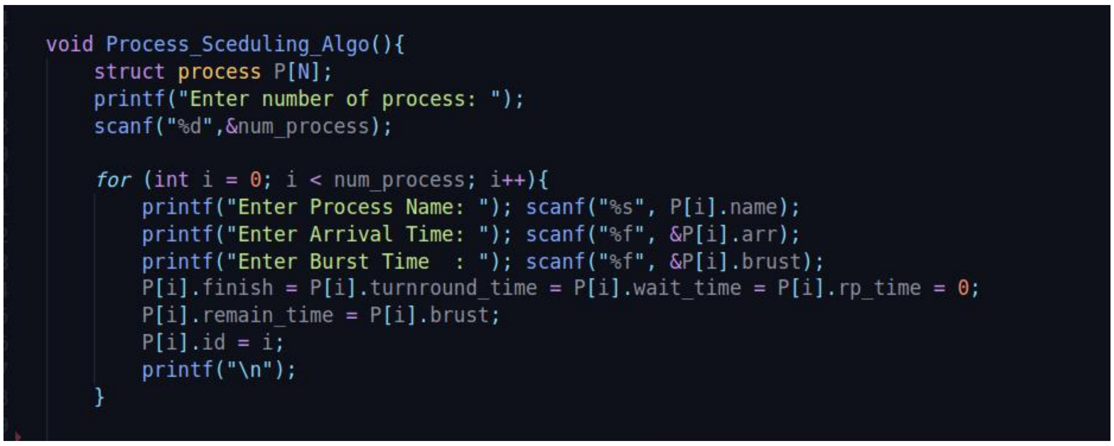
```
void pop(pQueue q)
{

    if (q->front == NULL){
        q->sz = 0;
        return;
    }

    pQNode temp = q->front;

    q->front = (q->front)->next;

    if (q->front == NULL)
        q->rear = NULL;
    (q->sz)--;
    free(temp);
}

int isEmpty(pQueue q){
    if (q->sz == 0) return 1;
    return 0;
}
//-----------------------------------------------------------------
```

**- Hàm Show() để in ra kết quả:**

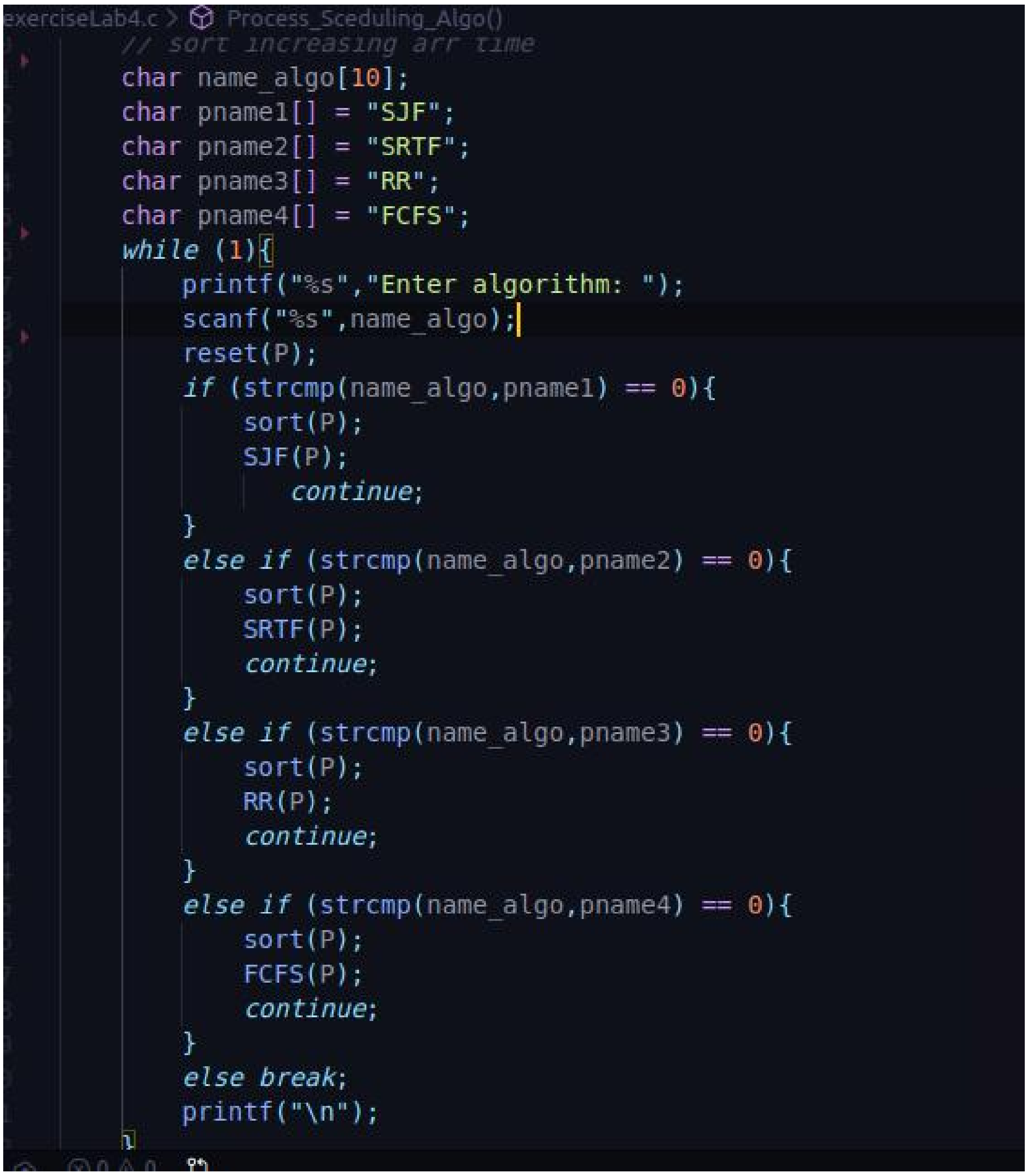


```c
void show(struct process P[]){
    // sort increasing id
    for (int i = 0; i < num_process - 1; i++)
            for (int j = i + 1; j < num_process; j++)
            if (P[j].id < P[i].id) swap(&P[i],&P[j]);


    printf(" _____________________________________________________________________________________________________________ \
    printf("|  Process Name   | Arrival Time | Burst Time | Completion Time | Turnaround Time | Waiting Time | Response Time |\
    printf("|_________________|______________|____________|_________________|_________________|______________|_______________|\

    float avg_waiting_time = 0;
    float avg_turnaround = 0;
    float avg_respone_time = 0;
    for(int i=0;i<num_process;i++){
        printf("|%-16s|%-14f|%-12f|%-17f|%-17f|%-14f|%-15f|\n", P[i].name, P[i].arr, P[i].brust, P[i].finish, P[i].turnround_t

        avg_waiting_time += P[i].wait_time;
        avg_turnaround += P[i].turnround_time;
        avg_respone_time += P[i].rp_time;
    }

    printf("|_________________|______________|____________|_________________|_________________|______________|_______________|\
    avg_waiting_time /= num_process;
    avg_turnaround /= num_process;
    avg_respone_time /= num_process;

    printf("Average waiting time : %.2f\n", avg_waiting_time);
    printf("Average turn around time: %.2f\n", avg_turnaround);
    printf("Average respone time: %.2f\n",avg_respone_time);
}
```

**- Hàm Process_Sceduling_Algo() để nhập dữ liệu và chọn các thuật toán thực thi:**

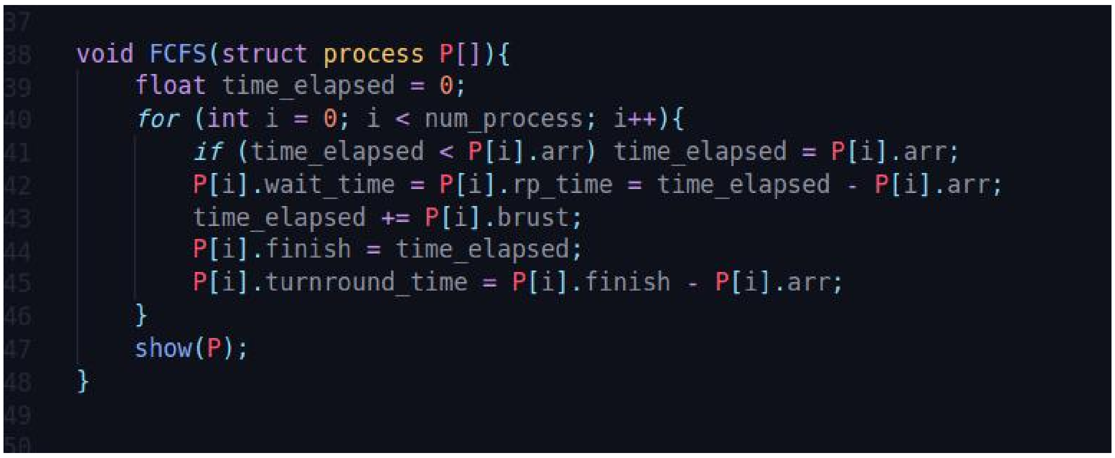


```c
void Process_Sceduling_Algo(){
    struct process P[N];
    printf("Enter number of process: ");
    scanf("%d",&num_process);

    for (int i = 0; i < num_process; i++){
        printf("Enter Process Name: "); scanf("%s", P[i].name);
        printf("Enter Arrival Time: "); scanf("%f", &P[i].arr);
        printf("Enter Burst Time  : "); scanf("%f", &P[i].brust);
        P[i].finish = P[i].turnround_time = P[i].wait_time = P[i].rp_time = 0;
        P[i].remain_time = P[i].brust;
        P[i].id = i;
        printf("\n");
    }
```



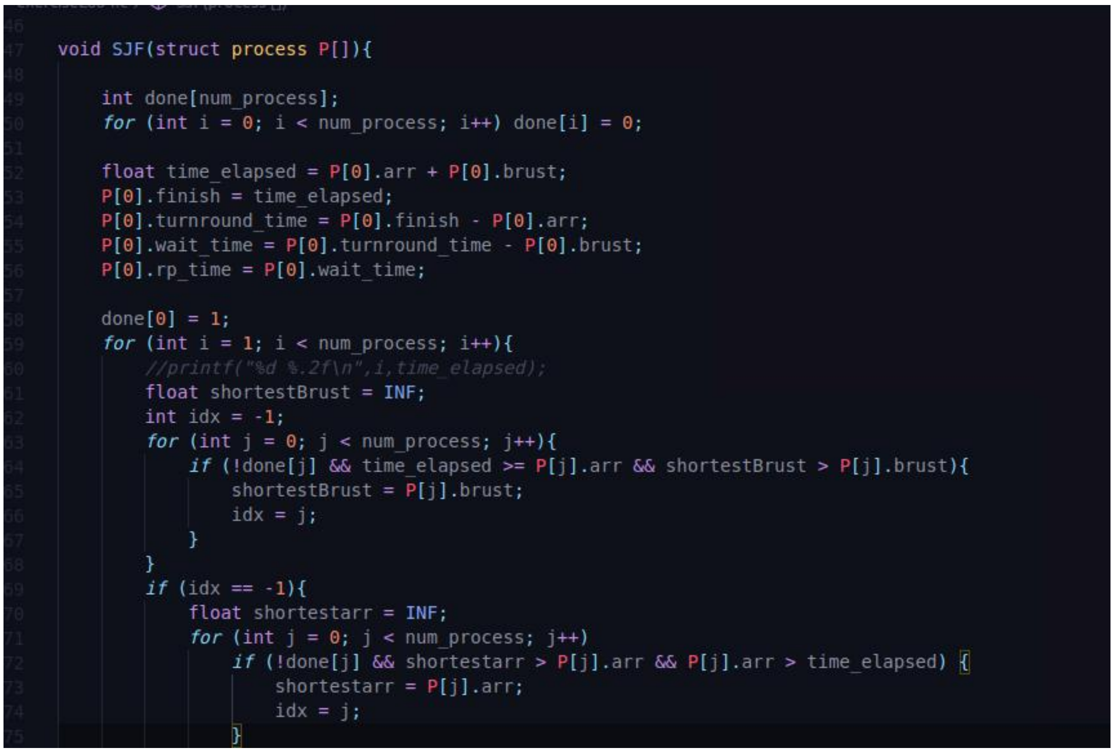
```
    // sort increasing arr time
    char name_algo[10];
    char pname1[] = "SJF";
    char pname2[] = "SRTF";
    char pname3[] = "RR";
    char pname4[] = "FCFS";
    while (1){
        printf("%s","Enter algorithm: ");
        scanf("%s",name_algo);
        reset(P);
        if (strcmp(name_algo,pname1) == 0){
            sort(P);
            SJF(P);
                continue;
        }
        else if (strcmp(name_algo,pname2) == 0){
            sort(P);
            SRTF(P);
            continue;
        }
        else if (strcmp(name_algo,pname3) == 0){
            sort(P);
            RR(P);
            continue;
        }
        else if (strcmp(name_algo,pname4) == 0){
            sort(P);
            FCFS(P);
            continue;
        }
        else break;
        printf("\n");
    }
```

**- Giải thuật FCFS:**

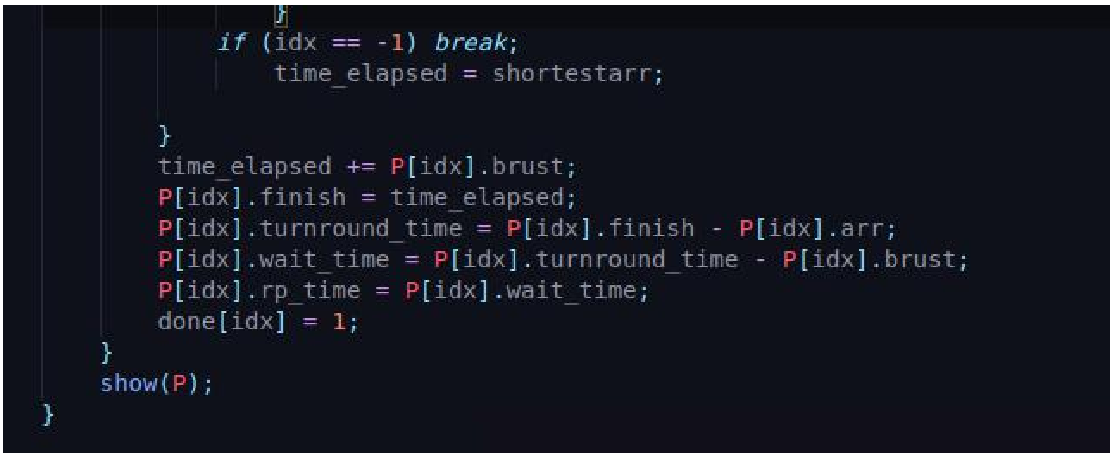


```c
void FCFS(struct process P[]){
    float time_elapsed = 0;
    for (int i = 0; i < num_process; i++){
        if (time_elapsed < P[i].arr) time_elapsed = P[i].arr;
        P[i].wait_time = P[i].rp_time = time_elapsed - P[i].arr;
        time_elapsed += P[i].brust;
        P[i].finish = time_elapsed;
        P[i].turnround_time = P[i].finish - P[i].arr;
    }
    show(P);
}
```

**- Giải thuật SJF:**

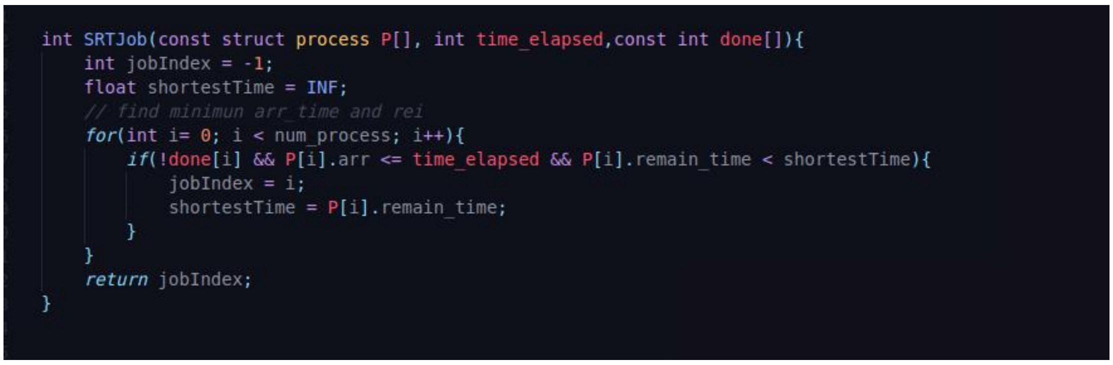


```c
void SJF(struct process P[]){

    int done[num_process];
    for (int i = 0; i < num_process; i++) done[i] = 0;

    float time_elapsed = P[0].arr + P[0].brust;
    P[0].finish = time_elapsed;
    P[0].turnround_time = P[0].finish - P[0].arr;
    P[0].wait_time = P[0].turnround_time - P[0].brust;
    P[0].rp_time = P[0].wait_time;

    done[0] = 1;
    for (int i = 1; i < num_process; i++){
        //printf("%d %.2f\n",i,time_elapsed);
        float shortestBrust = INF;
        int idx = -1;
        for (int j = 0; j < num_process; j++){
            if (!done[j] && time_elapsed >= P[j].arr && shortestBrust > P[j].brust){
                shortestBrust = P[j].brust;
                idx = j;
            }
        }
        if (idx == -1){
            float shortestarr = INF;
            for (int j = 0; j < num_process; j++)
                if (!done[j] && shortestarr > P[j].arr && P[j].arr > time_elapsed) {
                    shortestarr = P[j].arr;
                    idx = j;
                }
```

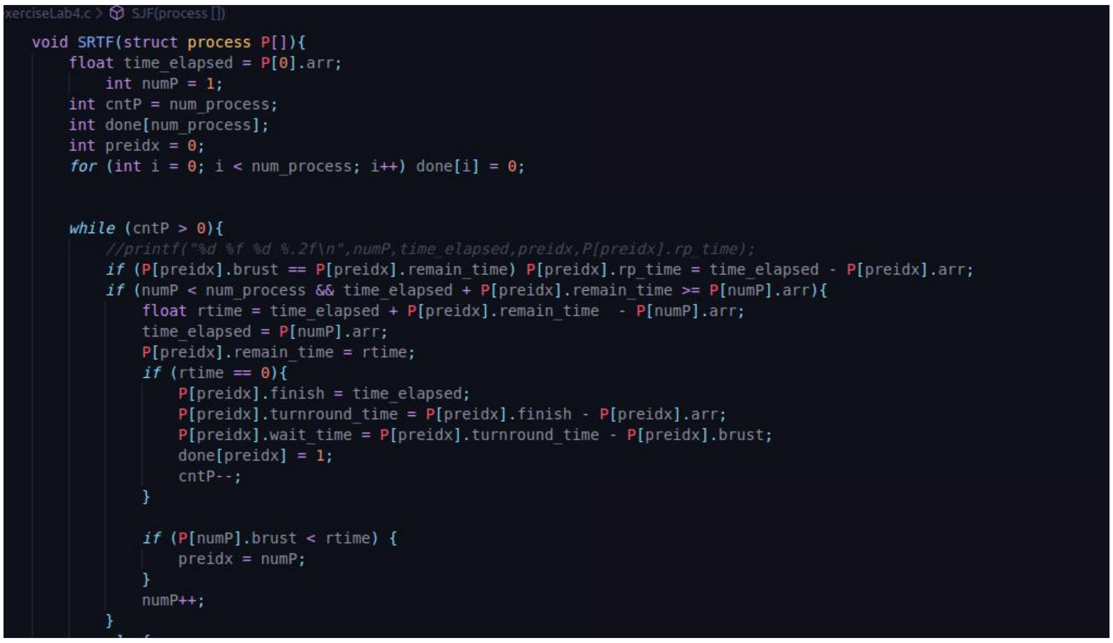


```c
                }
            if (idx == -1) break;
                time_elapsed = shortestarr;

        }
        time_elapsed += P[idx].brust;
        P[idx].finish = time_elapsed;
        P[idx].turnround_time = P[idx].finish - P[idx].arr;
        P[idx].wait_time = P[idx].turnround_time - P[idx].brust;
        P[idx].rp_time = P[idx].wait_time;
        done[idx] = 1;
    }
    show(P);
}
```

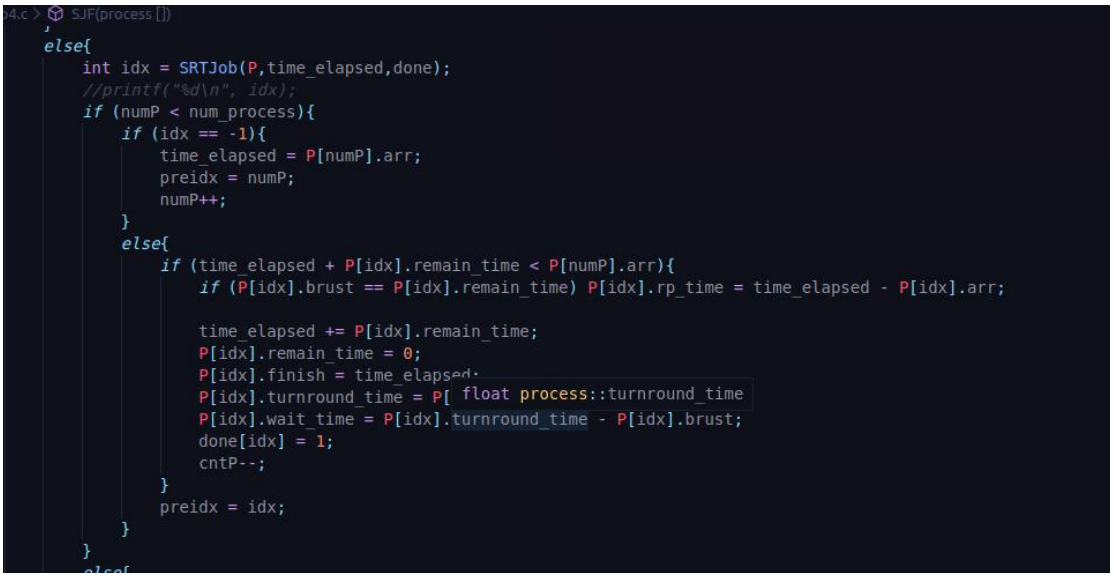
## - Giải thuật SRTF:



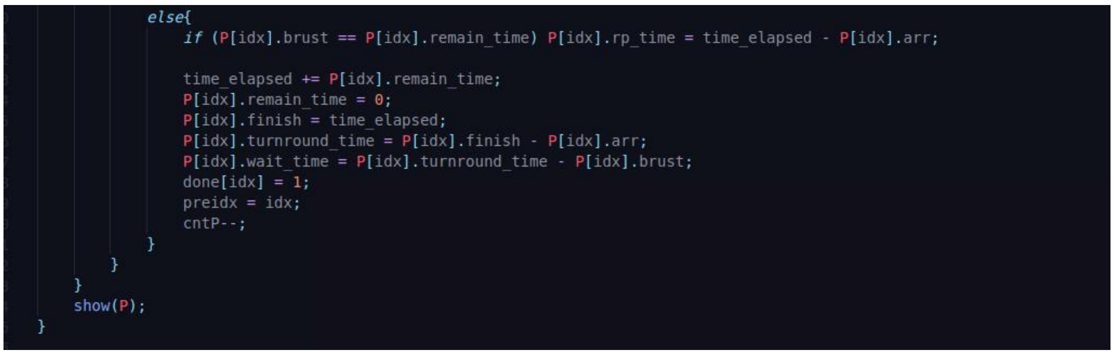
```c
int SRTJob(const struct process P[], int time_elapsed,const int done[]){
    int jobIndex = -1;
    float shortestTime = INF;
    // find minimun arr_time and rei
    for(int i= 0; i < num_process; i++){
        if(!done[i] && P[i].arr <= time_elapsed && P[i].remain_time < shortestTime){
            jobIndex = i;
            shortestTime = P[i].remain_time;
        }
    }
    return jobIndex;
}
```



```c
void SRTF(struct process P[]){
    float time_elapsed = P[0].arr;
        int numP = 1;
    int cntP = num_process;
    int done[num_process];
    int preidx = 0;
    for (int i = 0; i < num_process; i++) done[i] = 0;


    while (cntP > 0){
        //printf("%d %f %d %.2f\n",numP,time_elapsed,preidx,P[preidx].rp_time);
        if (P[preidx].brust == P[preidx].remain_time) P[preidx].rp_time = time_elapsed - P[preidx].arr;
        if (numP < num_process && time_elapsed + P[preidx].remain_time >= P[numP].arr){
            float rtime = time_elapsed + P[preidx].remain_time  - P[numP].arr;
            time_elapsed = P[numP].arr;
            P[preidx].remain_time = rtime;
            if (rtime == 0){
                P[preidx].finish = time_elapsed;
                P[preidx].turnround_time = P[preidx].finish - P[preidx].arr;
                P[preidx].wait_time = P[preidx].turnround_time - P[preidx].brust;
                done[preidx] = 1;
                cntP--;
            }

            if (P[numP].brust < rtime) {
                preidx = numP;
            }
            numP++;
        }
```

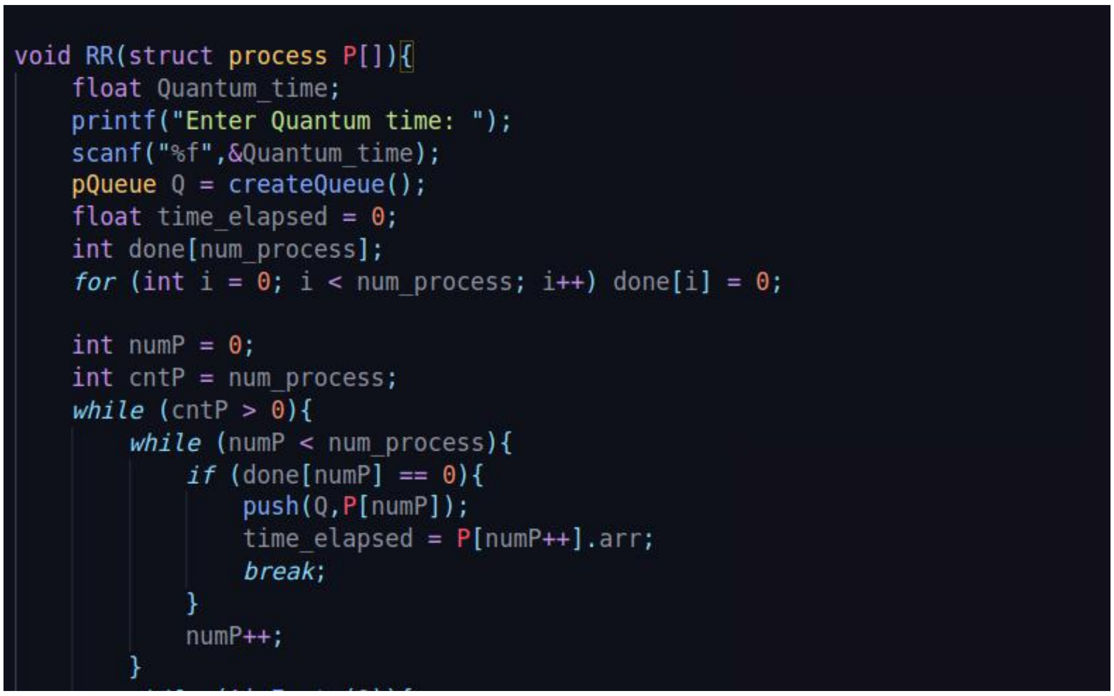


```c
    else{
        int idx = SRTJob(P,time_elapsed,done);
        //printf("%d\n", idx);
        if (numP < num_process){
            if (idx == -1){
                time_elapsed = P[numP].arr;
                preidx = numP;
                numP++;
            }
            else{
                if (time_elapsed + P[idx].remain_time < P[numP].arr){
                    if (P[idx].brust == P[idx].remain_time) P[idx].rp_time = time_elapsed - P[idx].arr;

                    time_elapsed += P[idx].remain_time;
                    P[idx].remain_time = 0;
                    P[idx].finish = time_elapsed;
                    P[idx].turnround_time = P[ float process::turnround_time
                    P[idx].wait_time = P[idx].turnround_time - P[idx].brust;
                    done[idx] = 1;
                    cntP--;
                }
                preidx = idx;
            }
        }
```

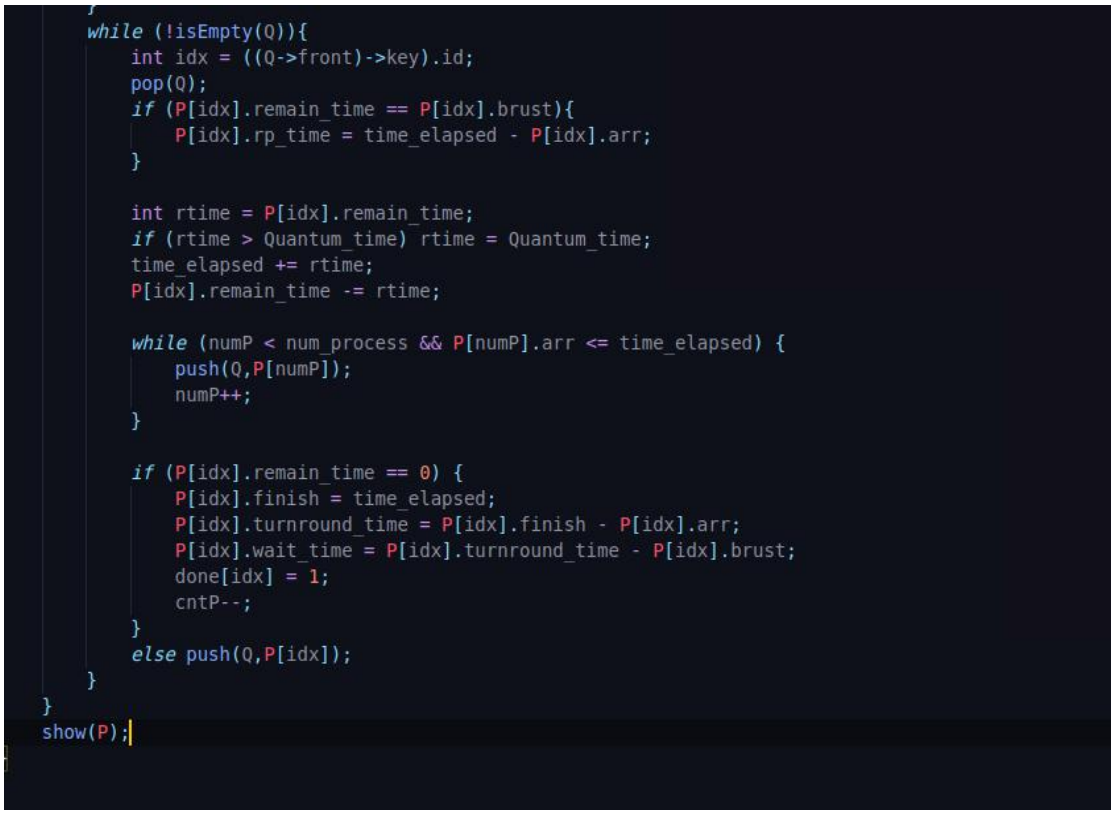


```
                else{
                    if (P[idx].brust == P[idx].remain_time) P[idx].rp_time = time_elapsed - P[idx].arr;

                    time_elapsed += P[idx].remain_time;
                    P[idx].remain_time = 0;
                    P[idx].finish = time_elapsed;
                    P[idx].turnround_time = P[idx].finish - P[idx].arr;
                    P[idx].wait_time = P[idx].turnround_time - P[idx].brust;
                    done[idx] = 1;
                    preidx = idx;
                    cntP--;
                }
            }
        }
        show(P);
    }
```

**- Giải thuật RR:**



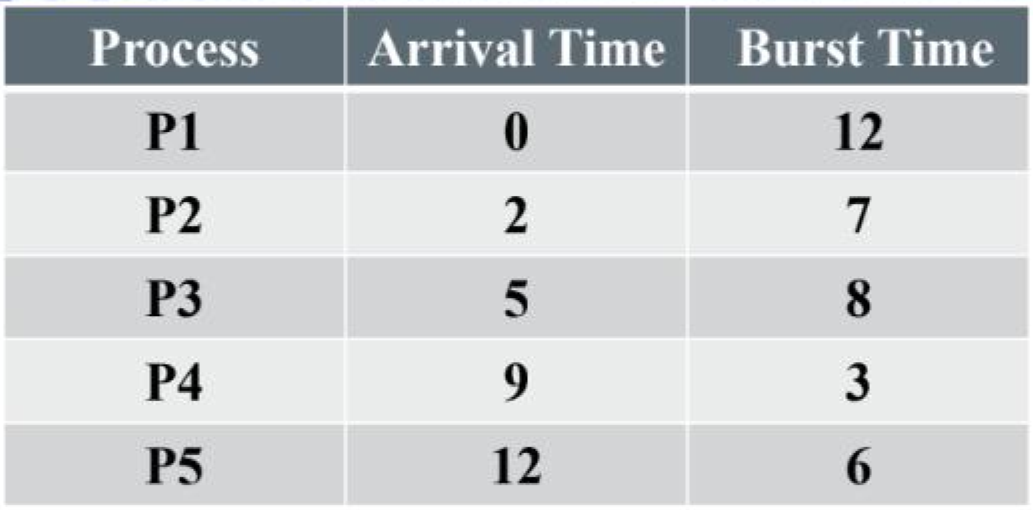
```
void RR(struct process P[]){
    float Quantum_time;
    printf("Enter Quantum time: ");
    scanf("%f",&Quantum_time);
    pQueue Q = createQueue();
    float time_elapsed = 0;
    int done[num_process];
    for (int i = 0; i < num_process; i++) done[i] = 0;

    int numP = 0;
    int cntP = num_process;
    while (cntP > 0){
        while (numP < num_process){
            if (done[numP] == 0){
                push(Q,P[numP]);
                time_elapsed = P[numP++].arr;
                break;
            }
            numP++;
        }
```



```
    while (!isEmpty(Q)){
        int idx = ((Q->front)->key).id;
        pop(Q);
        if (P[idx].remain_time == P[idx].brust){
            P[idx].rp_time = time_elapsed - P[idx].arr;
        }

        int rtime = P[idx].remain_time;
        if (rtime > Quantum_time) rtime = Quantum_time;
        time_elapsed += rtime;
        P[idx].remain_time -= rtime;

        while (numP < num_process && P[numP].arr <= time_elapsed) {
            push(Q,P[numP]);
            numP++;
        }

        if (P[idx].remain_time == 0) {
            P[idx].finish = time_elapsed;
            P[idx].turnround_time = P[idx].finish - P[idx].arr;
            P[idx].wait_time = P[idx].turnround_time - P[idx].brust;
            done[idx] = 1;
            cntP--;
        }
        else push(Q,P[idx]);
    }
}
show(P);
```

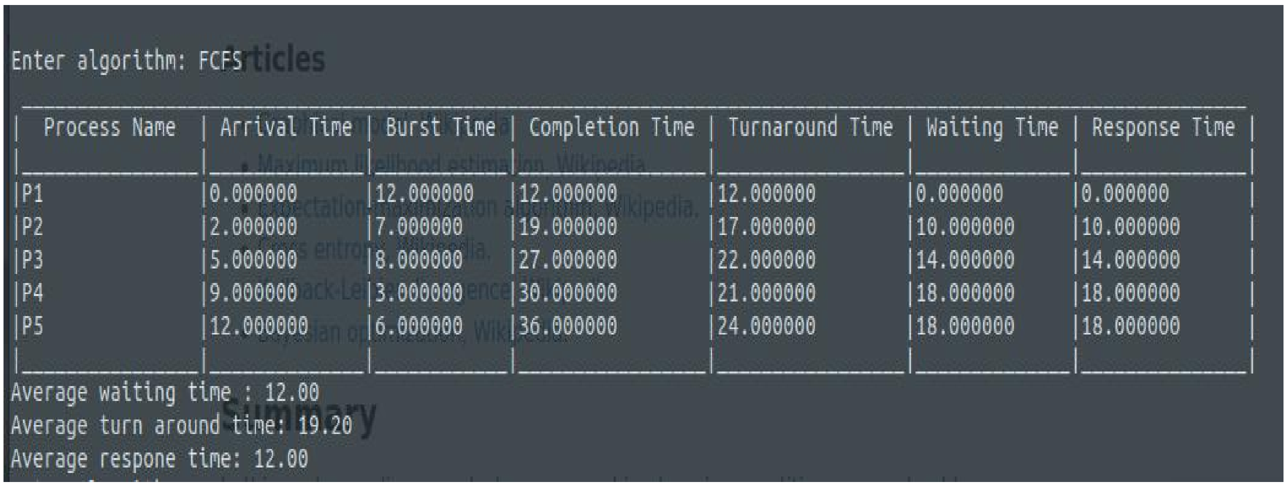
*** Link Full Code:**
https://github.com/KhangTran2503/IT007.K21.KHTN/blob/master/Lab4/exerciseLab4.c

## 2. Test:



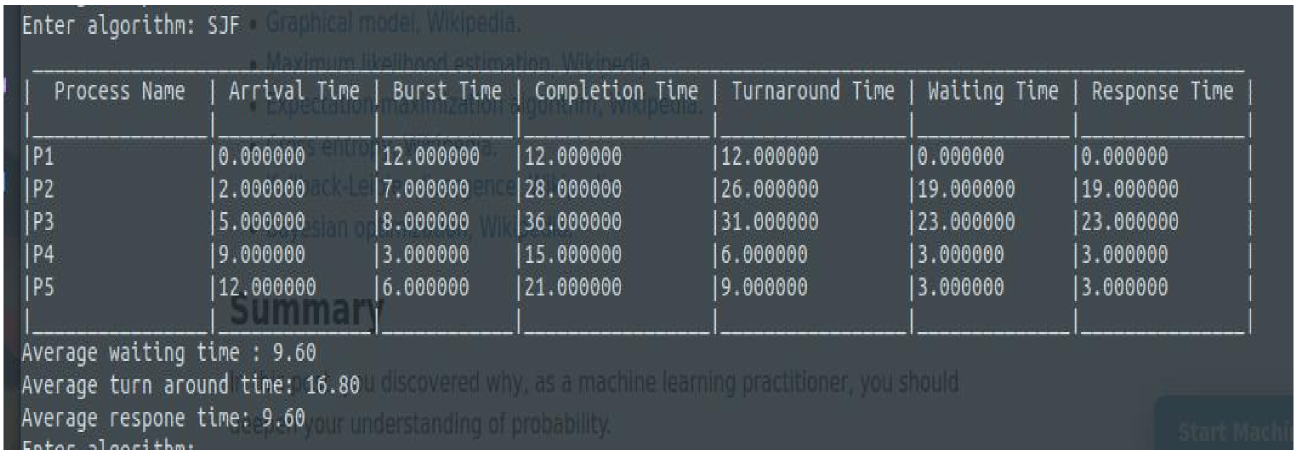
| Process | Arrival Time | Burst Time |
|---|---|---|
| P1 | 0 | 12 |
| P2 | 2 | 7 |
| P3 | 5 | 8 |
| P4 | 9 | 3 |
| P5 | 12 | 6 |

## Câu 1: Giải thuật FCFS(First Come First Severed)



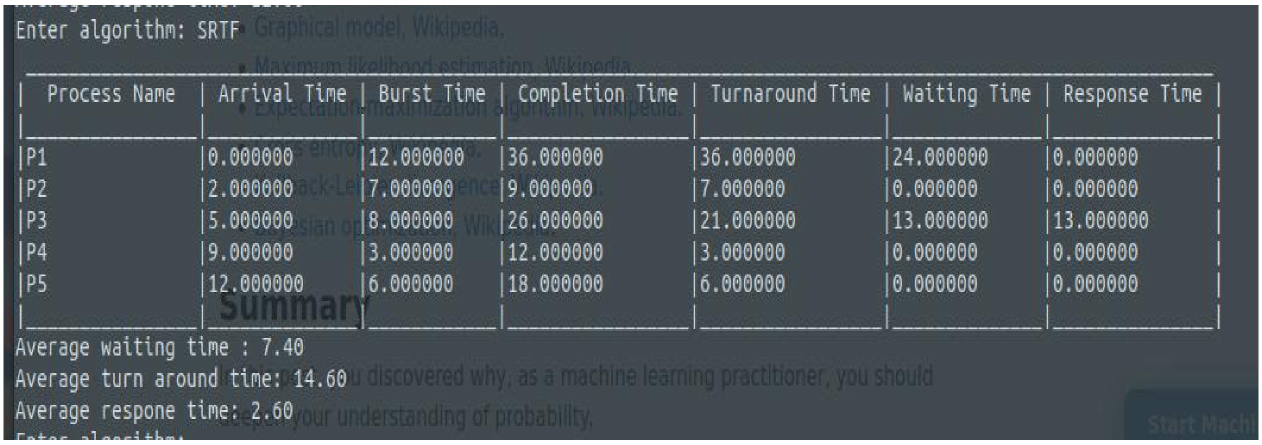
```
Enter algorithm: FCFS
 ________________________________________________________________________________________________________________________________
|  Process Name  | Arrival Time | Burst Time | Completion Time | Turnaround Time | Waiting Time | Response Time |
|________________|______________|____________|_________________|_________________|______________|_______________|
|P1              |0.000000      |12.000000   |12.000000        |12.000000        |0.000000      |0.000000       |
|P2              |2.000000      |7.000000    |19.000000        |17.000000        |10.000000     |10.000000      |
|P3              |5.000000      |8.000000    |27.000000        |22.000000        |14.000000     |14.000000      |
|P4              |9.000000      |3.000000    |30.000000        |21.000000        |18.000000     |18.000000      |
|P5              |12.000000     |6.000000    |36.000000        |24.000000        |18.000000     |18.000000      |
|________________|______________|____________|_________________|_________________|______________|_______________|
Average waiting time : 12.00
Average turn around time: 19.20
Average respone time: 12.00
```

## Câu 2: Giải thuật SJF



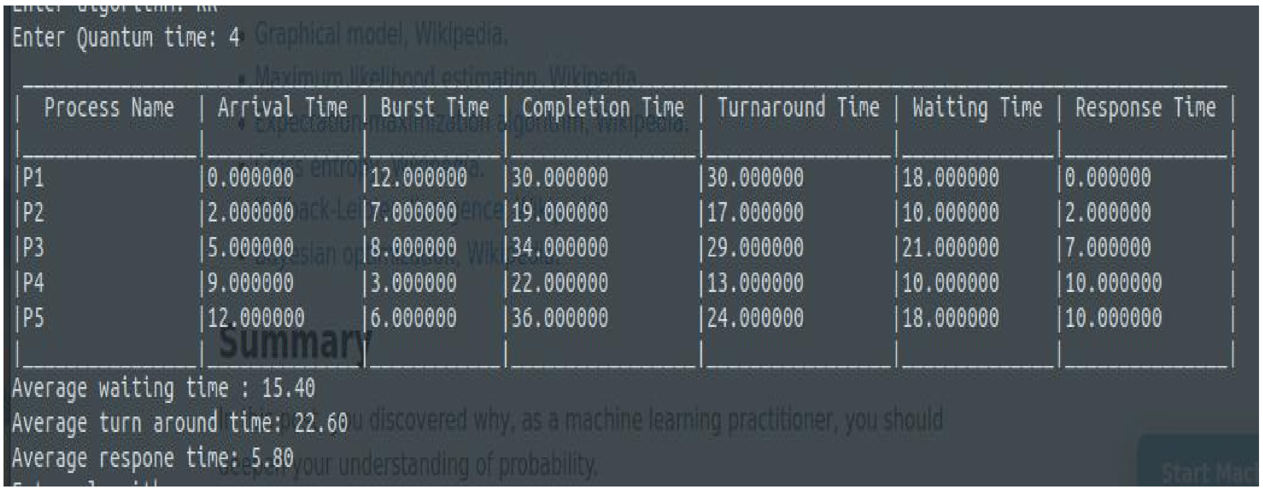
```
Enter algorithm: SJF
 ________________________________________________________________________________________________________________________________
|  Process Name  | Arrival Time | Burst Time | Completion Time | Turnaround Time | Waiting Time | Response Time |
|________________|______________|____________|_________________|_________________|______________|_______________|
|P1              |0.000000      |12.000000   |12.000000        |12.000000        |0.000000      |0.000000       |
|P2              |2.000000      |7.000000    |28.000000        |26.000000        |19.000000     |19.000000      |
|P3              |5.000000      |8.000000    |36.000000        |31.000000        |23.000000     |23.000000      |
|P4              |9.000000      |3.000000    |15.000000        |6.000000         |3.000000      |3.000000       |
|P5              |12.000000     |6.000000    |21.000000        |9.000000         |3.000000      |3.000000       |
|________________|______________|____________|_________________|_________________|______________|_______________|
Average waiting time : 9.60
Average turn around time: 16.80
Average respone time: 9.60
```

## Câu 3: Giải thuật SRTF



```
Enter algorithm: SRTF
 ________________________________________________________________________________________________________________________________
|  Process Name  | Arrival Time | Burst Time | Completion Time | Turnaround Time | Waiting Time | Response Time |
|________________|______________|____________|_________________|_________________|______________|_______________|
|P1              |0.000000      |12.000000   |36.000000        |36.000000        |24.000000     |0.000000       |
|P2              |2.000000      |7.000000    |9.000000         |7.000000         |0.000000      |0.000000       |
|P3              |5.000000      |8.000000    |26.000000        |21.000000        |13.000000     |13.000000      |
|P4              |9.000000      |3.000000    |12.000000        |3.000000         |0.000000      |0.000000       |
|P5              |12.000000     |6.000000    |18.000000        |6.000000         |0.000000      |0.000000       |
|________________|______________|____________|_________________|_________________|______________|_______________|
Average waiting time : 7.40
Average turn around time: 14.60
Average respone time: 2.60
```

Câu 4: Giải thuật RR(Quantum time = 4)

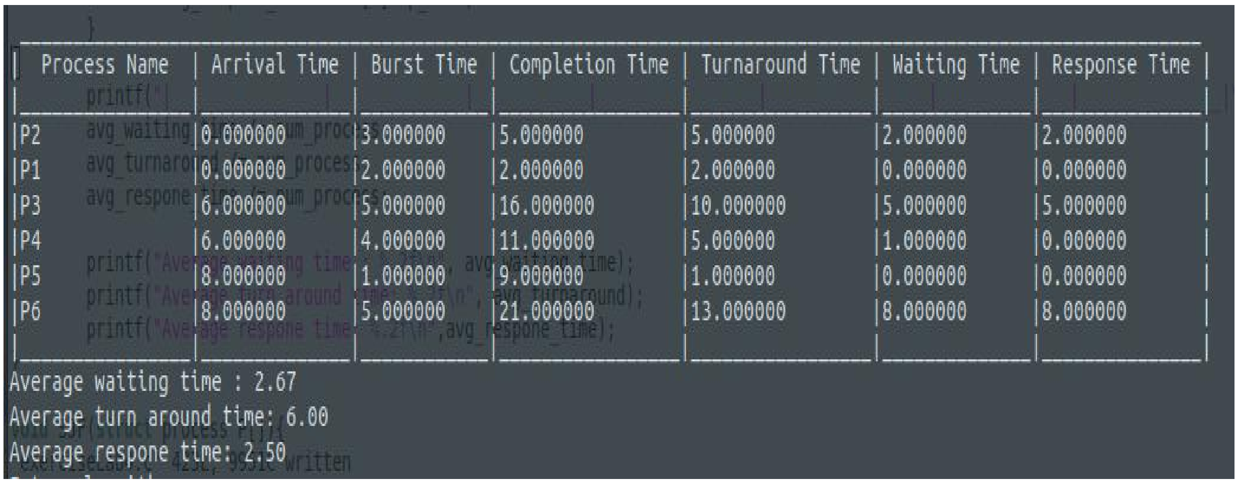


```
Enter Quantum time: 4

 ____________________________________________________________________________________________________________
|  Process Name  | Arrival Time | Burst Time | Completion Time | Turnaround Time | Waiting Time | Response Time |
|________________|______________|____________|_________________|_________________|______________|_______________|
|P1              |0.000000      |12.000000   |30.000000        |30.000000        |18.000000     |0.000000       |
|P2              |2.000000      |7.000000    |19.000000        |17.000000        |10.000000     |2.000000       |
|P3              |5.000000      |8.000000    |34.000000        |29.000000        |21.000000     |7.000000       |
|P4              |9.000000      |3.000000    |22.000000        |13.000000        |10.000000     |10.000000      |
|P5              |12.000000     |6.000000    |36.000000        |24.000000        |18.000000     |10.000000      |
|________________|______________|____________|_________________|_________________|______________|_______________|
Average waiting time : 15.40
Average turn around time: 22.60
Average respone time: 5.80
```

Test Thầy:

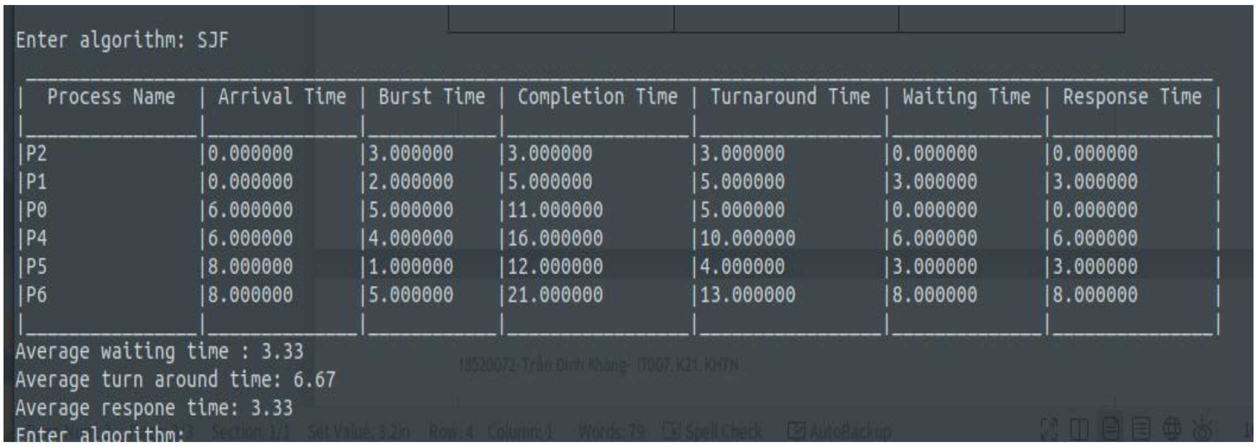
| Process | Arrival Time | Brust Time |
|---|---|---|
| P2 | 0 | 3 |
| P1 | 0 | 2 |
| P0 | 6 | 5 |
| P4 | 6 | 4 |
| P5 | 8 | 1 |
| P6 | 8 | 5 |

+ Giải thuật SRTF:



```
 ___________________________________________________________________________________________________________________
|  Process Name  | Arrival Time | Burst Time | Completion Time | Turnaround Time | Waiting Time | Response Time |
|________________|______________|____________|_________________|_________________|______________|_______________|
|P2              |0.000000      |3.000000    |5.000000         |5.000000         |2.000000      |2.000000       |
|P1              |0.000000      |2.000000    |2.000000         |2.000000         |0.000000      |0.000000       |
|P3              |6.000000      |5.000000    |16.000000        |10.000000        |5.000000      |5.000000       |
|P4              |6.000000      |4.000000    |11.000000        |5.000000         |1.000000      |0.000000       |
|P5              |8.000000      |1.000000    |9.000000         |1.000000         |0.000000      |0.000000       |
|P6              |8.000000      |5.000000    |21.000000        |13.000000        |8.000000      |8.000000       |
|________________|______________|____________|_________________|_________________|______________|_______________|
Average waiting time : 2.67
Average turn around time: 6.00
Average respone time: 2.50
```

+ Giải thuật SJF:



```
Enter algorithm: SJF
 ___________________________________________________________________________________________________________________
|  Process Name  | Arrival Time | Burst Time | Completion Time | Turnaround Time | Waiting Time | Response Time |
|________________|______________|____________|_________________|_________________|______________|_______________|
|P2              |0.000000      |3.000000    |3.000000         |3.000000         |0.000000      |0.000000       |
|P1              |0.000000      |2.000000    |5.000000         |5.000000         |3.000000      |3.000000       |
|P0              |6.000000      |5.000000    |11.000000        |5.000000         |0.000000      |0.000000       |
|P4              |6.000000      |4.000000    |16.000000        |10.000000        |6.000000      |6.000000       |
|P5              |8.000000      |1.000000    |12.000000        |4.000000         |3.000000      |3.000000       |
|P6              |8.000000      |5.000000    |21.000000        |13.000000        |8.000000      |8.000000       |
|________________|______________|____________|_________________|_________________|______________|_______________|
Average waiting time : 3.33
Average turn around time: 6.67
Average respone time: 3.33
Enter algorithm:
```