Tên: Vũ Tuấn Hải

MSSV: 17520433

Lớp: IT007.J21

**BÁO CÁO LAB 4**

1. Viết chương trình mô phỏng giải thuật SJF

a. Source code

#include <iostream>

#include <queue>

using namespace std;

struct Process {

int name;

int burst\_time;

int arrival\_time;

};

static double ave\_turnaround\_time = 0;

static double ave\_waiting\_time = 0;

void swap(Process &p1, Process &p2) {

Process tmp;

tmp = p1;

p1 = p2;

p2 = tmp;

}

void sortByArrivalTime(Process \*p, int n) {

for (int i = 0; i < n; i++)

{

for (int j = i + 1; j < n; j++)

{

if (p[i].arrival\_time < p[j].arrival\_time) {

swap(p[i], p[j]);

}

else{

if (p[i].arrival\_time == p[j].arrival\_time) {

if (p[i].burst\_time < p[j].burst\_time)

{

swap(p[i], p[j]);

}

}

}

}

}

}

void sortByBurstTime(Process \*p, int n, int time\_current) {

for (int i = 0; i < n - 1; i++)

{

for (int j = i + 1; j < n; j++)

{

if (p[i].burst\_time < p[j].burst\_time && p[i].arrival\_time <= time\_current) {

swap(p[i], p[j]);

}

}

}

}

void Input(Process \*p, int n) {

for (int i = 0; i < n; i++) {

cout << "-----------------" << endl;

cout << "Nhap ID process: "; cin >> p[i].name;

cout << "Nhap burst time: "; cin >> p[i].burst\_time;

cout << "Nhap arrival time: "; cin >> p[i].arrival\_time;

}

}

void SelectionFunction(Process \*p, int n) {

int time\_current;

int flag = 1;

// Sort theo thứ tự arrival time trước đã

sortByArrivalTime(p, n);

// Hàm lựa chọn quyết định xem process nào vào queue trước

for (int i = 0; i < n; n--)

{

//pQueue.push(p[n - 1]);

if (flag == 1) {

time\_current = p[n - 1].arrival\_time;

flag = 0;

}

time\_current += p[n - 1].burst\_time;

ave\_waiting\_time += time\_current - p[n - 1].arrival\_time - p[n - 1].burst\_time;

ave\_turnaround\_time += (time\_current - p[n - 1].arrival\_time);

cout << p[n - 1].name << " " << time\_current - p[n - 1].arrival\_time - p[n - 1].burst\_time << " " << time\_current - p[n - 1].arrival\_time - p[n - 1].burst\_time << " " << (time\_current - p[n - 1].arrival\_time) << endl;

sortByBurstTime(p, n - 1, time\_current);

}

}

int main()

{

Process \*p = new Process[100];

queue<Process> pQueue;

int n;

cout << "Nhap so luong process: "; cin >> n;

Input(p, n);

cout << "Process Response-time Waiting-time Turn around-time" << endl;

SelectionFunction(p, n);

cout << "Thoi gian dap ung trung binh: " << ave\_waiting\_time / n << endl;

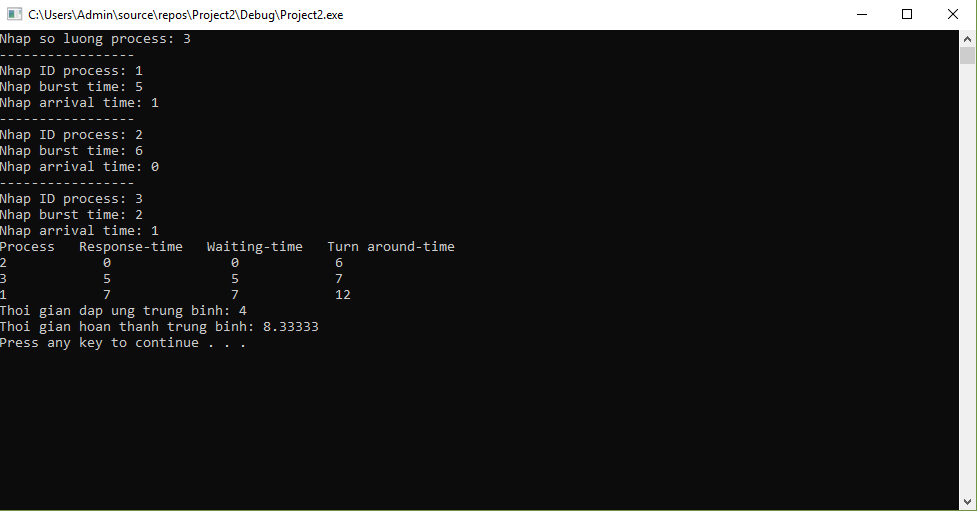
cout << "Thoi gian hoan thanh trung binh: " << ave\_turnaround\_time / n << endl;

system("pause");

return 0;

}

b. Kết quả chạy thử



2. Viết chương trình mô phỏng giải thuật SRT

a. Source code

#include <iostream>

#include <queue>

using namespace std;

struct Process {

int name;

int burst\_time;

int arrival\_time;

};

static double ave\_turnaround\_time = 0;

static double ave\_waiting\_time = 0;

void swap(Process &p1, Process &p2) {

Process tmp;

tmp = p1;

p1 = p2;

p2 = tmp;

}

int minBurstTime(Process \*p, int n) {

int min = p[0].burst\_time;

int imin = 0;

for (int i = 0; i < n; i++)

{

if (p[i].burst\_time < min) {

min = p[i].burst\_time;

imin = i;

}

}

return imin;

}

void sortByArrivalTime(Process \*p, int n) {

for (int i = 0; i < n; i++)

{

for (int j = i + 1; j < n; j++)

{

if (p[i].arrival\_time < p[j].arrival\_time) {

swap(p[i], p[j]);

}

else {

if (p[i].arrival\_time == p[j].arrival\_time) {

if (p[i].burst\_time < p[j].burst\_time)

{

swap(p[i], p[j]);

}

}

}

}

}

}

void ShortestRemainingTimeFirst(Process \*p, int n, int time\_current, int burst\_time) {

for (int i = 0; i < n - 1; i++)

{

for (int j = i + 1; j < n; j++)

{

if (p[i].burst\_time < p[j].burst\_time && p[i].arrival\_time <= time\_current && p[i].burst\_time < burst\_time) {

swap(p[i], p[j]);

}

}

}

}

void Input(Process \*p, int n) {

for (int i = 0; i < n; i++) {

cout << "-----------------" << endl;

cout << "Nhap ID process: "; cin >> p[i].name;

cout << "Nhap burst time: "; cin >> p[i].burst\_time;

cout << "Nhap arrival time: "; cin >> p[i].arrival\_time;

}

}

int check(int flag[], int n) {

for (int i = 0; i < n; i++) {

if (flag[i] == 1) return 1;

}

return 0;

}

void SelectionFunction(Process \*p, int n) {

Process \*p\_temp = new Process[100];

int time\_current = 0; // timeline chương trình

///

int flag\_c = 1; // phòng trừ trường hợp không có process nào tới tại thời điểm 0

int flag\_first\_come[100]; // đánh dấu thời điểm được thực thi lần đầu

int flag\_previous; // vị trí process vừa chạy trước đó

int flag\_current; // vị trí process hiện đang chạy

int waiting\_time[100]; // tính thời gian chờ mỗi khi bị preemtive đến lúc được thực thi lại

/// khởi tạo các mảng tạm

for (int i = 0; i < 100; i++) {

waiting\_time[i] = 0;

flag\_first\_come[i] = -1; // -1 vì nếu sử dụng 0 để đánh dấu (flag chỉ cập nhật duy nhất 1 lần

// ) thì sẽ có process đầu tiên có arrival time = 0 cập nhật 2 lần

}

sortByArrivalTime(p, n); // chọn ra process có burst time min xếp ra phía cuối

flag\_first\_come[p[n - 1].name] = p[n-1].arrival\_time;

// Duyệt từ cuối lên

while (n > 0) {

p[n - 1].burst\_time--; // Xét từ từ chậm rãi

// tăng waiting time khi process đã đến hàng đợi mà chưa được thực thi

for (int i = 0; i < n; i++) {

if (p[i].arrival\_time <= time\_current) {

waiting\_time[p[i].name]++;

}

}

// Chừa thằng đang thực thi ra

waiting\_time[p[n - 1].name]--;

// Trừ trường hợp ko có process nào đến lúc time\_current = 0

if (flag\_c == 1) {

time\_current = p[n - 1].arrival\_time;

flag\_c = 0;

}

time\_current++;

flag\_previous = p[n - 1].name; // Lưu tên process sắp rời/ có khả năng rời hàng đợi

// Nếu đã thực thi hết (không còn burst thì cout trạng thái

if (p[n - 1].burst\_time == 0) {

cout << p[n - 1].name << " " << flag\_first\_come[p[n - 1].name] - p[n-1].arrival\_time << " " << waiting\_time[p[n - 1].name] << " " << (time\_current - p[n-1].arrival\_time) << endl;

ave\_waiting\_time += waiting\_time[p[n - 1].name]; // cộng dồn waiting time đã cộng nãy giờ

ave\_turnaround\_time += time\_current - p[n - 1].arrival\_time;

n--; // thu hẹp kích thước để xét những thằng còn lại

if (n == 0) return;

}

ShortestRemainingTimeFirst(p, n, time\_current, p[n-1].burst\_time); // chọn ra thằng có burst < burst còn lại của p[flag\_current]

flag\_current = p[n - 1].name; // cập nhập lại nếu có thằng nào đó thỏa cái trên

// nếu chuyển ngữ cảnh xảy ra

if (flag\_current != flag\_previous) {

if (flag\_first\_come[p[n - 1].name] == -1) {

flag\_first\_come[p[n - 1].name] = time\_current;

}

}

}

}

int main()

{

Process \*p = new Process[100];

queue<Process> pQueue;

int n;

cout << "Nhap so luong process: "; cin >> n;

Input(p, n);

cout << "Process Response-time Waiting-time Turn around-time" << endl;

SelectionFunction(p, n);

cout << "Thoi gian dap ung trung binh: " << ave\_waiting\_time / n << endl;

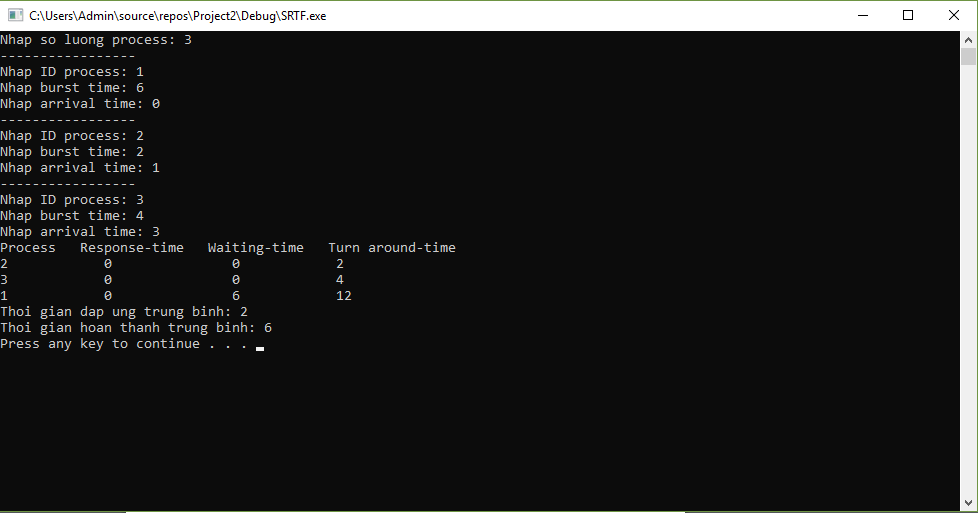
cout << "Thoi gian hoan thanh trung binh: " << ave\_turnaround\_time / n << endl;

system("pause");

return 0;

}

b. Kết quả chạy thử



3. Viết chương trình mô phỏng giải thuật RR

a. Source code

#include <iostream>

#include <queue>

using namespace std;

struct Process {

int name;

int burst\_time;

int arrival\_time;

};

static queue<int> pStart\_time;

static double ave\_turnaround\_time = 0;

static double ave\_waiting\_time = 0;

static int waiting\_time[100];

void swap(Process &p1, Process &p2) {

Process tmp;

tmp = p1;

p1 = p2;

p2 = tmp;

}

void sortByArrivalTime(Process \*p, int n) {

for (int i = 0; i < n; i++)

{

for (int j = i + 1; j < n; j++)

{

if (p[i].arrival\_time > p[j].arrival\_time) {

swap(p[i], p[j]);

}

}

}

}

void Input(Process \*p, int n) {

for (int i = 0; i < n; i++) {

cout << "-----------------" << endl;

cout << "Nhap ID process: "; cin >> p[i].name;

cout << "Nhap burst time: "; cin >> p[i].burst\_time;

cout << "Nhap arrival time: "; cin >> p[i].arrival\_time;

}

}

int check(int flag[], int n) {

for (int i = 0; i < n; i++) {

if (flag[i] == 1) return 1;

}

return 0;

}

queue<Process> SelectionFunction(Process \*p, int n, int quantum\_time) {

int time\_current = 0;

int flag\_c = 1;

queue<Process> pQueue;

int flag[100];

sortByArrivalTime(p, n);

for (int i = 0; i < n; i++) {

flag[i] = 1;

waiting\_time[i] = p[i].arrival\_time;

}

while (check(flag, n))

{

// Duyệt qua hết 1 lượt các process

for (int i = 0; i < n; i++) {

// process nào đã đến & còn burst time mới được xét

if (time\_current >= p[i].arrival\_time && flag[i] == 1) {

pQueue.push(p[i]);

if (flag\_c == 1) {

time\_current = p[i].arrival\_time;

flag\_c = 0;

}

pStart\_time.push(time\_current);

p[i].burst\_time -= quantum\_time;

if (p[i].burst\_time <= 0) {

flag[i] = 0;

ave\_turnaround\_time += time\_current + (p[i].burst\_time + quantum\_time) - p[i].arrival\_time;

}

ave\_waiting\_time += (time\_current - waiting\_time[i]);

if (p[i].burst\_time < 0) {

time\_current += p[i].burst\_time + quantum\_time;

}

else {

time\_current += quantum\_time;

}

waiting\_time[i] = time\_current;

}

}

}

return pQueue;

}

void Print(queue<Process> &pQueue) {

while (pQueue.empty() == 0) {

cout << pQueue.front().name << " -- ";

pQueue.pop();

}

cout << endl;

while (pStart\_time.empty() == 0) {

cout << pStart\_time.front() << " -- ";

pStart\_time.pop();

}

cout << endl;

}

int main()

{

Process \*p = new Process[100];

queue<Process> pQueue;

int n, quantum\_time;

cout << "Nhap so luong process: "; cin >> n;

cout << "Nhap quantum time: "; cin >> quantum\_time;

Input(p, n);

int total\_time = 0;

for (int i = 0; i < n; i++) {

total\_time += p[i].burst\_time;

}

int quantum\_step = total\_time / n;

if (quantum\_step \* n < total\_time) quantum\_step++;

pQueue = SelectionFunction(p, n, quantum\_time);

Print(pQueue);

cout << "Thoi gian cho trung binh: " << ave\_waiting\_time / n << endl;

cout << "Thoi gian hoan thanh trung binh: " << ave\_turnaround\_time / n << endl;

system("pause");

return 0;

}

b. Kết quả chạy thử

