一、開發環境 ：

1. Windows11

2. IDE : dev c++

二、實作方法與流程

1. 資料結構:

```
struct ready{ // Ready Queue
    int index ;
    int pid ;
    int priority ;
};

struct process{
    int pid ;
    int cpu_burst_ori ; // 原本的 cpu burst time (計算turnaroundTime用)
    int cpu_burst ; // 剩餘的 cpu burst time
    int arrival ;
    int priority ;
    int turnaroundTime ;
    double responseRatio ; // (cpu_burst + wait) / cpu_burst
    bool done ; // process 是否完成
    bool arrivalOK ;
    string output ; // Gantt Chart

};
```

```
int gmethod = 0 ;
int gtimeslice = 0 ;
int now = 0 ; // 目前時間
vector<process> vec ; // input data
vector<string> output ; // 每個method自己的 Gantt Chart
vector<ready> readyQ ; // RR 的 ready Queue
vector<ready> PPRRQ ; // PPRR 的 ready Queue
vector<int> arrival ; // 所有process的 arrivaltime
vector<vector<process>> alloutput ; // 所有method的輸出data
```

2. 流程 ：

I. First Come First Served (FCFS) :

(1) 將檔案讀入 vector 後依照 Arrival Time 排序，若 Arrival time 相同則進行 Process ID 的排序。

(2) 每次取 vector 內第一個未完成的 process 執行，直到每個 process 執行完畢。

(3) 將結果輸出檔案。

II. Round Robin (RR) :

(1) 將檔案讀入 vector 後依照 Arrival Time 排序，若 Arrival Time 相同則進行 Process ID 的排序。

(2) 將 Arrival time 比目前時間小且未執行過的 process 放入 Queue，每次取 Queue 中的第一個 process 執行。

(3) 當 process Time Out 時，被換下的 process 放入 Queue 重新排隊，當 process 還沒用完 time slice 就執行完時，就換下一個 process 執行，且有完整的 time slice。

(4) 重複(2)、(3)直到每個 process 皆執行完畢，再將結果輸出檔案。

III. Shortest Remaining Time First (SRTF)：

(1) 將檔案讀入 vector。

(2) 依照 CPU Burst Time 排序，若 CPU Burst Time 相同則進行 Arrival Time 的排序，若 Arrival Time 也相同則進行 Process ID 的排序。

(3) 每次取 vector 內的第一個未完成且 Arrival Time 小於目前時間的 process 執行，並檢查每個時間點是否有新的 process 抵達，若有新的 process 抵達則回到(2)。

(4) 重複(2)、(3)直到每個 process 皆執行完畢，再將結果輸出檔案。

IV. Preemptive Priority + Round Robin (PPRR)：

(1) 將檔案讀入 vector 後依照 Arrival time 排序，若 Arrival time 相同則進行 Process ID 的排序。

(2) 將 Arrival time 比目前時間小且未執行過的 process 放入 Queue，並在 Queue 中依照 priority 值由小到大做排序。

(3) 每次取 Queue 中的第一個 process 執行

- 若 Queue 中有和 process 相同 priority 的其他 process 則採用 RR 原則進行排程，並檢查每個時間點是否有優先權 ≥ 目前 process 的其他 process 抵達，若有則回到(2)。
- 若 Queue 中沒有和 process 相同 priority 的其他 process 則不需採 RR 進行排程，但仍需檢查每個時間點是否有優先權 ≥ 目前 process 的其他 process 抵達，若有則回到(2)。

(4) 重複(2)、(3)直到每個 process 皆執行完畢，將結果輸出檔案。

V. Highest Response Ratio Next (HRRN)：

(1) 將檔案讀入 vector。

(2) 計算未完成 process 的 Response Ratio 並依照 Response Ratio 排序，若 Response Ratio 相同則進行 Arrival Time 的排序，若 Arrival Time 相同則進行 Process ID 的排序。

(3) 每次取 vector 內第一個未完成且 Arrival Time 小於目前時間的 process 執行。

(4) 重複(2)、(3)直到每個 process 皆執行完畢，再將結果輸出檔案。

VI. ALL：

將 FCFS、RR、SRTF、PPRR、HRRN 都實作一次，並輸出檔案。

3. **不同排程法的比較：**

平均等待時間（Average Waiting Time）：

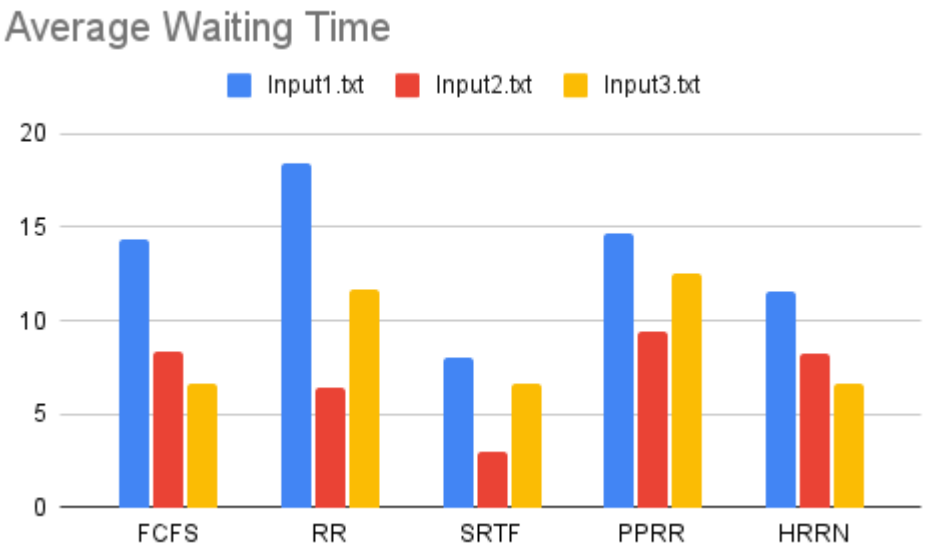
	Input1.txt	Input2.txt	Input3.txt
FCFS	14.333	8.4	6.666
RR	18.4	6.4	11.666
SRTF	8.066	3	6.666
PPRR	14.666	9.4	12.5
HRRN	11.6	8.2	6.666

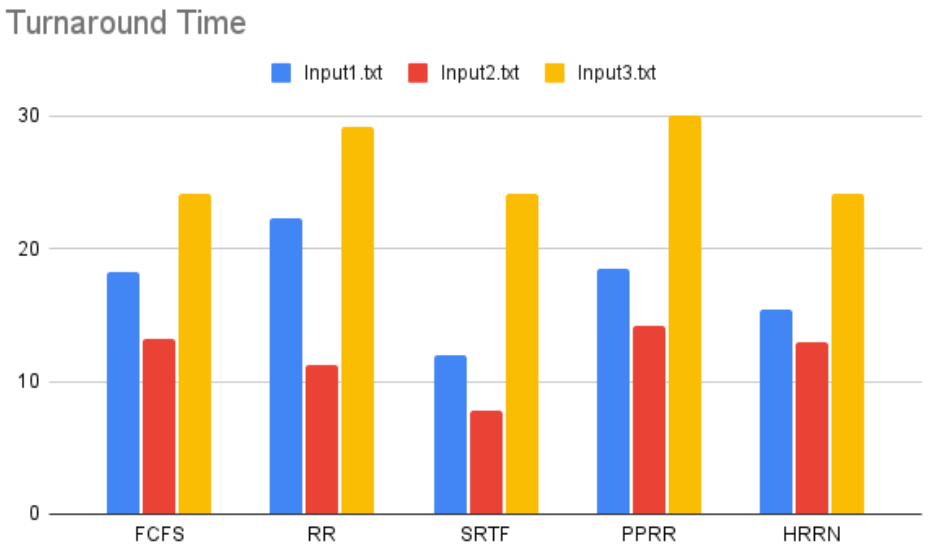
工作往返時間（Turnaround Time）

	Input1.txt	Input2.txt	Input3.txt
FCFS	18.2	13.2	24.16
RR	22.26	11.2	29.16
SRTF	11.93	7.8	24.16
PPRR	18.53	14.2	30
HRRN	15.46	13	24.16

4. **結果與討論：**

(1) 由圖表結果得知，在 input1、input2 中用 SRTF 排程法的平均等待時間最短，而在 input3 中 FCFS、SRTF、HRRN 的平均等待時間皆為最短。





(2) 其他比較: 將 input2 的 time slice 改為 1、5、15，比較 Average Waiting Time 的差異

Average Waiting Time :

	time slice=1	time slice=3	time slice=5	time slice=15
FCFS	8.4	8.4	8.4	8.4
RR	6.6	6.4	7.2	8.4
SRTF	3	3	3	3
PPRR	9.6	9.4	9.4	9.4
HRRN	8.2	8.2	8.2	8.2

a. 若 time slice 過大則 RR 會變成 FCFS，排班效能不佳。
b. 若 time slice 太小，除了 context switching 太過頻繁外，cpu performance 也不好。