資訊三甲 10827102 沈柏融

開發環境 ：

Dev-c++ 5.11 version

實作方法和流程 ：

依照使用者鍵入的檔案名稱讀取資料，並儲存進特定 **list** 中，完成指定的排程方法後將結果輸出至新的**.txt** 檔中

FCFS :

依照 **Arrival_Time** 的先後次序，對原先讀入的 **list** 進行 **bubble sort** ，如遇到相同的 **Arrival_Time** 則比較其 **ID** 大小，每完成一個 **ID** 紀錄其完成時間，並算出 **Turnaround_Time** 和 **Waiting_Time** 儲存至 **vector** 中

RR:

依照 **Arrival_Time** 的先後次序，若 **Arrival_Time** 相同則比較其 **ID** 大小，將已到達的 **Process** 放進 **queue** 中，每一個 **Process** 的 **Time_Slice** 用完時就必須回到 **queue** 的最後面，若此時剛好有新的 **Process** 到達，則必須讓新的 **Process** 排在前面，直到所有的 **Process** 執行完畢，且每完成一個 **ID** 紀錄其完成時間，並算出 **Turnaround_Time** 和 **Waiting_Time** 儲存至 **vector** 中

SJF:

依照 **CPU_Brust** 的大小排序，若剩餘 **CPU_Brust** 相同則依照 **Arrival_Time** 的先後次序，若 **Arrival_Time** 相同則比較其 **ID** 大小，

執行完一 **Process** 再重新依照 **CPU_Brust** 的大小排序，且每完成一個 **ID** 紀錄其完成時間，並算出 **Turnaround_Time** 和 **Waiting_Time** 儲存至 **vector** 中

SRTF:

依照 **CPU_Brust** 的大小排序，若剩餘 **CPU_Brust** 相同則依照 **Arrival_Time** 的先後次序，若 **Arrival_Time** 相同則比較其 **ID** 大小，決定下一個時間單位要執行哪個 **Process**，且每完成一個 **ID** 紀錄其完成時間，並算出 **Turnaround_Time** 和 **Waiting_Time** 儲存至 **vector** 中

PPRR:

依照 **Priority** 的大小排序，若有多個 **Priority** 的大小相同，則採用 **RR** 的原則將多個 **Process** 放進 **queue** 中，若有更小於之前所有的 **Process** 出現，則優先處理 **Priority** 最小的排序，且每完成一個 **ID** 紀錄其完成時間，並算出 **Turnaround_Time** 和 **Waiting_Time** 儲存至 **vector**

HRRN

依照反應時間比率的大小排序，若反應時間比率的大小相同則依照 **Arrival_Time** 的先後次序，若 **Arrival_Time** 相同則比較其 **ID** 大小，決定下一個時間單位要執行哪個 **Process**，且每完成一個 **ID** 紀錄其完成時間，並算出 **Turnaround_Time** 和 **Waiting_Time** 儲存至 **vector** 中

不同排程法的比較 ：

以範例 **input2.txt** 來比較

FCFS
Average waiting times : 8.4
Average turnaround times: 13.2
RR
Average waiting times : 6.4
Average turnaround times: 11.2
SJF
Average waiting times : 8.2
Average turnaround times: 13
SRTF
Average waiting times : 3
Average turnaround times: 7.8
PPRR
Average waiting times : 9.4
Average turnaround times: 14.2
HRRN
Average waiting times : 8.2
Average turnaround times: 13

Waiting Time

ID	FCFS	RR	SJF	SRTF	HRRN	PPRR
1	0	13	0	13	0	0
2	10	2	10	0	10	21
3	10	2	12	0	12	8
4	11	6	8	1	8	9
5	11	9	11	1	11	9

Turnaround Time

ID	FCFS	RR	SJF	SRTF	HRRN	PPRR
1	11	24	11	24	11	11
2	12	4	12	2	12	23
3	13	5	15	3	15	11
4	13	8	10	3	10	11
5	17	15	17	7	17	15

解果與討論 ：

FCFS

因為一定要處理完當前程序才會讓出 **CPU**，所以 **I/O** 繁忙的環境非常不利，比較有利於長時間的作業

RR

因為必須要執行完一個時間片段，因此非常看重時間片段的長度設定，過長的話跟 **FCFS** 沒什麼區別，過短的話因為要多個時間片段才能完成，響應時間會被拉長，像是範例的 **input1.txt**

SJF

SJF 單純討論誰最快就先跑誰，雖然有了更好的排班效益，**2** 但嚴重偏好 **short　job**，對 **long-burst-time** 的排程非常不公平，需要等待非常久的時間

SRTF

比起 **SJF**，他要求的是剩下的時間，如果有 **Process** 更快，將會允許他插隊，雖然有了更好的排班效益，但會付出更多的 **contest swich** 的代價

PPRR

比起 **RR**，多了 **Priority** 這一個參考值，但我目前不知道其存在的原因

HRRN

每當要進行作業排程的時候，都會計算一次反應時間比率，也因為如此，就算進行長時間作業，隨著時間增加，後面的排程也有機會獲得排程執行，但也因為每次都多了計算反應時間比率，增加了系統的開銷