CS Data Structure

Fall 2023

Program Exercise #3

In-class Demo: 2023/10/26 (Thu.) 13:10-16:00

* Submit Due Date: 2023/11/1 (Wed.) 23:59:00

Problem: 用 Queue 來模擬銀行的運作情形

客戶(Customer)來到銀行需要排隊(Queue)等待處理業務。排在前面(front)的客戶可以先處理業務，後來的客戶則從後面(rear)繼續排下去。若出納員(teller)沒有在處理其他客戶的業務(像是剛處理完一個客戶的業務時)，則在該窗口排隊等待的客戶中，排最前面的，可到該出納員的服務窗口開始處理業務。

同一時間會有數個窗口同時在處理業務。當有一客戶到達銀行時，會選擇排隊人數最少的窗口去排隊。(若有人數相同的情形，則先排窗口號碼較小的窗口)

此外，銀行的出納員會在特定的時間增加或是減少，所以當出納員人數改變時，客戶便會重新依下列規則重新排隊:

出納員增加: 原排隊客戶不會改變，只影響出納員增加後才到達的客戶

出納員減少: 關閉此窗戶，出納員不再接受客戶排隊，原先排在第一順位的客戶不受影響，可以待在該窗口持續接受服務；其他排在後面的客戶會依照排隊順序重新選擇排隊人數較少的窗口排隊。

每位客戶都只會固定排在一開始所選擇的窗口，只有在該窗口被關閉才會重新排隊。

請依照測試檔案所提供的客戶資料，算出每一位客戶離開銀行的時間。

每個客戶有以下資料：
id: 用來辨別客戶的代號
arrival time: 客戶到達銀行的時間
service time: 處理該客戶業務所需時間

假設客戶(A)到銀行的時間是 t1，處理客戶(A)的業務要花的時間是 t2，在銀行有 n 個出納員可以提供服務(代表有 n 個服務窗口)的情況下，有兩種情況要考慮：

Case 1:

若客戶(A)到銀行時，有一個以上的出納員在等待客戶來處理業務，則客 戶(A)可以馬上開始和出納員接洽，並在 t1 + t2 時處理完業務並離開銀行。

Case 2:

若客戶(A)到銀行時，所有的出納員都正在處理其他客戶的業務，則客戶(A)則從 n 個服務窗口中，選排隊人數最少的並從後面接著排下去。客戶(A)會在 t1 + waiting time + t2 時處理完業務。

Input File Format:

```
2
John 3 9
Mr._Brown 4 12
Alice 6 20
Jack 9 18
# 11 1
God 18 22
@ 22 1
GiGi 23 10
```

第一行的數字代表總共有幾個出納員。範例中為“2”,代表共有0 號～1 號兩個出納員(窗口)，這兩個窗口在一開始都是可以接受排隊的。在本題目中，至多會有 10 個出納員，至少會有 1 個。

“# 11 1”則是代表“在時間 11 的時候關閉1 號窗口”；
“@ 22 1”則是代表 “在時間 22 的時候重新開放1 號窗口”。

請注意，窗口0 永遠都不會被關閉。 同一時間只會有一個窗口被打開或是關閉。其他行則是代表客戶的代號或名字，客戶的代號或名字裡面不會有空格，最長不超過20。接著出現的數字分別代表客戶到達銀行時間、處理該客戶業務所需時間。這些數字都會是 “整數”，用int 或long 處理即可。Input File 最後將以EOF(End-of-File)結束。

Output File Format:

請輸出每個客戶的Leaving Time(每個客戶離開銀行的時間)，以及該客戶選擇的服務窗口。Leaving Time = 到達時間 + 等待時間 + 所需的處理時間。顯示資訊的順序，同客戶移出queue 的順序。對於同一時間移出queue 的客戶，先處理出納員號碼較小的queue；對於同一queue 中，同時要移出queue 的客戶，則依照移出順序處理。

以本題為例，Output File 應如下所示：

John 12 0 Mr._Brown 16 1 Alice 32 0 GiGi 33 1 Jack 50 0 God 62 0

說明如下:

時間	動作	窗口 0 狀態	窗口 1 狀態
0	--	(Open)	(Open)
3	John 排入窗口 0	(Open) John	(Open)
4	Mr._Brown 排入窗口 1	(Open) John	(Open) Mr._Brown
6	Alice 排入窗口 0	(Open) John, Alice	(Open) Mr._Brown
9	Jack 排入窗口 1	(Open) John, Alice	(Open) Mr._Brown, Jack
11	關閉窗口 1，Jack 重新排隊	(Open) John, Alice, Jack	(Close) Mr._Brown
12	John 離開	(Open) Alice, Jack	(Close) Mr._Brown
16	Mr._Brown 離開	(Open) Alice, Jack	(Close)
18	God 排入窗口 0	(Open) Alice, Jack, God	(Close)
22	重新開放窗口 1	(Open) Alice, Jack, God	(Open)
23	GiGi 排入窗口 1	(Open) Alice, Jack, God	(Open) GiGi
32	Alice 離開	(Open) Jack, God	(Open) GiGi
33	GiGi 離開	(Open) Jack, God	(Open)
50	Jack 離開	(Open) God	(Open)
62	God 離開	(Open)	(Open)

其他注意事項：

1. input file 中，每個時間點至多會有一個客戶到銀行，且排列是依照到達銀行的順序。
2. 若有排隊人數最少的服務窗口不只一個(ex: 編號i、i+2、i+4 的窗口排隊人數一樣少，則選編號 i 的窗口排隊(選編號數字最小的)
3. Max Queue Size = 10 (必須使用 Circular Queue，以Array 模擬；Output File 中所要求的index 即為Array 的index)
4. Input：要求使用者輸入input file 的檔名。
   Output：在螢幕上 show 出結果。

5. Demo時，助教會使用測試檔測試結果。
6. 程式的基本 data structure 和程式架構可參考後面的APPENDIX。
7. 請確定您的程式可以在Dev-C 環境下編譯執行。

Submission:

Filename format:

學號_PE#.c

例如: M06455505_PE3.c

- In-class demo (11/1 Tue. 13:10-16:00)
- iLearn 2.0 submission (Due at 10/31 Mon. 23:59)

* 用自己的學號建立資料夾，並將 source code 放入資料夾，壓縮上傳 iLearn 2.0

Grading:

Correctness	50%
Program structure	20%
Comments	10%
Header block	5%
Variable dictionary	10%
Procedures and functions	5%

Special notice:

請勿抄襲別人程式(助教會當場進行測問、判定)，或是遲交作業，否則一律 0 分計算。請一律使用 C 語言來撰寫程式，且必須保證你的程式能夠在 Dev-C++ / Code::Blocks 軟體上成功編譯與執行，使用其他程式語言一律不予計分

請依照題目給的輸入格式，否則不計分

==========

APPENDIX: Data Type

```
#define MAX_QUEUE_SIZE 10
int global_clock;

typedef struct Customer {
    char id[20];            // 客戶 id
    int arr_time;           // 客戶到達銀行時間
    int ser_time;           // 客戶完成業務要花多少時間
}Customer;

typedef struct TellerQueue {
    int status;                 // status, 0: close, 1:只剩下一位客戶，即將close, 2: open
    int front;                  // front pointer，指向第一個element 的index
    int rear;                   // rear pointer，指向最後一個element 的下一個index
    int count;                  // queue 中在排隊的人數
    int current_served_time;    // 目前在處理業務的客人已經花的時間
    Customer queue[MAX_QUEUE_SIZE];  // 以 array 型式模擬Circular Queue。
}TellerQueue;

// P.S. 一開始，front = 0, rear = 0
有 10 個服務窗口，所以要有 10 個Queue 來紀錄每個窗口目前的排隊狀態，
TellerQueue teller_queue[10];
```

Ex:

TellerQueue[0]

	B	D	K	H															

TellerQueue[0].front=0 TellerQueue.rear[0]=4 TellerQueue[0].count=4

TellerQueue[1]

	C	F																	

TellerQueue[1].front=0 TellerQueue.rear[1]=2 TellerQueue[1].count=2

TellerQueue[2]

		T	F	L															

TellerQueue[2].front=1 TellerQueue.rear[1]=4 TellerQueue[1].count=3

APPENDIX: Basic Program Flow

一開始 global_clock 設為 0。
所有的 TellerQueue 的 status 均設為 2。

```
main()
{
    while not end of file
    {
        ReadOneRecord();
        UpdateQueueStatus( int global_clock, int new_arr_time, ……)
        If (增加窗口)
        {
            OpenQueue (QueueId);
        }
        else if (關閉窗口)
            {
                CloseQueue (QueueId);
                FromTheSecondElementInCloseQueue(QueueId)
                {
                    GetTheShortestQueueId();
                    InsertCustomer( int index, struct customer NewCustomer, …..)
                }
        else
            {
                GetTheShortestQueueId ( int TellerQueueCount[3] )
                InsertNewCustomer( int index, struct customer NewCustomer, …..)
            }
    }
    Process the remaining customers in queues
}
```

UpdateQueueStatus(int global_clock, int new_arr_time, …)
每次讀入一行命令時, 要先 update 每個 Queue 目前的狀態，Check 每個 Queue 在 global_clock 和 new_arr_time 之間有沒有客戶完成業務，如果有客戶完成業務(一個 Queue 可能不只一位客戶完成業務)，把他們的離開時間輸出，並更新相關變數。
P.S : 在 new_arr_time 這個時間點, 每個 Queue 進行業務的客戶有可能改變,且每

個服務窗口進行的進度也會改變.

GetTheShortestQueueId ()
傳回 index 指出哪個窗口的排隊人數最短，此時客戶就到排隊人數最少的窗口排隊之後 update “TellerQueue.count”，“TellerQueue.rear”，“TellerQueue.status”…
InsertCustomer(int index, struct customer NewCustomer, …) 將客戶資料加到 Queue, 並更新相關資料