一、開發環境 ：

Window10，DevC++，語言使用 C++

二、程式設計 ：

功能 ：

依 input 的 page frame 數量與 page reference 順序來進執行 6 種 page replacement 的方法。

方法介紹 ：

1. First In First Out(FIFO) : 先進先出頁置換法，每當 page 被載入主記憶體內時，會以時間標記來記錄時間(已存在在主記憶體內的 page 則不會更新其時間標記)，若遇到需要做 page replace 時(page frame 滿了)，先從時間標記最小的 page 來做置換。
2. Least Recently Used(LRU) : 最近罕用頁置換法，每當要載入或使用 page 時，都會重新標記其時間標記(若已在主記憶體內的 page 也會更新)，當要 page replacement 時，先將過去最久沒被使用到的 page 做置換。
3. Least Frequently Used(LFU) + FIFO : 最不常使用頁置換法，在每個 page frame 設置 counter，計算其使用次數，當要做 page replacement 時，將 counter 數值最小的 page 做置換，若 counter 值相同則依照 FIFO 來做置換。
4. Most Frequently Used (MFU) + FIFO : 最常使用頁置換法，，在每個 page frame 設置 counter，計算其使用次數，當要做 page replacement 時，將 counter 數值最大的 page 做置換，若 counter 值相同則依照 FIFO 來做置換。
5. Least Frequently Used(LFU) + LRU : 最不常使用頁置換法，在每個 page frame 設置 counter，計算其使用次數，當要做 page replacement 時，將 counter 數值最小的 page 做置換，若 counter 值相同則依照 LRU 來做置換。
6. Most Frequently Used (MFU) + LRU : 最常使用頁置換法，，在每個 page frame 設置 counter，計算其使用次數，當要做 page replacement 時，將 counter 數值最大的 page 做置換，若 counter 值相同則依照 LRU 來做置換。

使用的資料結構 ：

我使用 type 為 int 的 vector 來存讀入的 page reference 順序，還有兩個 type 為 int 的 vector，來存每個方法所發生的 page fault 與 page replacement 次數，以及一個 int 來存 page frame 數量，並宣告了一個 struct Frame，裡面有 int 來存 page number、string 來存是否有發生 page fault、有 type 為 int 的 vector 來存 page frame。每個方法都有一個 type 為 Frame 的 vector 來存結果。在需要 counter 的方法中，我宣告了另一個 struct List，裡面有兩個

int，用來存 page 使用次數以及 page number。

流程 ：

1. FIFO :
   (1) 宣告一個 type 為 int 的 vector 來當作 page frame。
   (2) 當有 page 要被載入時，先檢查是否有存在在 page frame。
   (3) 若有就將目前的 page frame 以及 page number 存入 type 為 Frame 的 vector(fifo)中，並記錄其無 page fault 也無 page replacement 發生。
   (4) 若不在 page frame 內，則紀錄其發生 page fault，並看 page frame 是否已滿。
   (5) 若未滿就將此 page number 加到 page frame 中，並將目前的 page frame 以及 page number 存入 fifo 中。
   (6) 若已滿就將 page frame 的第一筆資料刪除，再將此時的 page number 將入 page frame 中，且將目前的 page frame 以及 page number 存入 fifo 中，並記錄其發生 page replacement。
   (7) 最後將此次發生 page fault 與 page replacement 的次數存到 vector 中。
2. LRU :
   (1) 宣告一個 type 為 int 的 vector 來當作 page frame。
   (2) 當有 page 要被載入時，先檢查是否有存在在 page frame。
   (3) 若有就將此筆資料刪除，再將 page number 重新加入 page frame 內，並將目前的 page frame 以及 page number 存入 type 為 Frame 的 vector(lru)中，並記錄其無 page fault 也無 page replacement 發生。
   (4) 若不在 page frame 內，則紀錄其發生 page fault，並看 page frame 是否已滿。
   (5) 若未滿就將此 page number 加到 page frame 中，並將目前的 page frame 以及 page number 存入 lru 中。
   (6) 若已滿就將 page frame 的第一筆資料刪除，再將此時的 page number 將入 page frame 中，且將目前的 page frame 以及 page number 存入 lru 中，並記錄其發生 page replacement。
   (7) 最後將此次發生 page fault 與 page replacement 的次數存到 vector 中。
3. LFU + FIFO :
   (1) 宣告一個 type 為 List 的 vector 作為 page frame。
   (2) 當有 page 要被載入時，先檢查是否有存在在 page frame。
   (3) 若有就將目前的 page frame 以及 page number 存入 type 為 Frame 的 vector(lfu)中，並將其 page frame 內的 counter 數加一，記錄其

無 page fault 也無 page replacement 發生。

(4) 若不在 page frame 內，則紀錄其發生 page fault，並看 page frame 是否已滿。

(5) 若未滿就將 counter 數設為 0，並把 page number 與 counter 加到 page frame 中，並將目前的 page frame 以及 page number 存入 lfu 中。

(6) 若已滿就將 page frame 內 counter 數最小的 page 刪除，若 counter 數相同則將 index 靠前的 page 刪除，再將此時的 page number 將入 page frame 中，且將目前的 page frame 以及 page number 存入 lfu 中，並記錄其發生 page replacement。

(7) 最後將此次發生 page fault 與 page replacement 的次數存到 vector 中。

4. MFU + FIFO :

(1) 宣告一個 type 為 List 的 vector 作為 page frame。

(2) 當有 page 要被載入時，先檢查是否有存在在 page frame。

(3) 若有就將目前的 page frame 以及 page number 存入 type 為 Frame 的 vector(mfu)中，並將其 page frame 內的 counter 數加一，記錄其無 page fault 也無 page replacement 發生。

(4) 若不在 page frame 內，則紀錄其發生 page fault，並看 page frame 是否已滿。

(5) 若未滿就將 counter 數設為 0，並把 page number 與 counter 加到 page frame 中，並將目前的 page frame 以及 page number 存入 mfu 中。

(6) 若已滿就將 page frame 內 counter 數最大的 page 刪除，若 counter 數相同則將 index 靠前的 page 刪除，再將此時的 page number 將入 page frame 中，且將目前的 page frame 以及 page number 存入 mfu 中，並記錄其發生 page replacement。

(7) 最後將此次發生 page fault 與 page replacement 的次數存到 vector 中。

5. LFU + LRU :

(1) 宣告一個 type 為 List 的 vector 作為 page frame。

(2) 當有 page 要被載入時，先檢查是否有存在在 page frame。

(3) 若有就將此筆資料刪除，並將 counter 加一與 page number 一起重新加入 page frame 內，且將目前的 page frame 以及 page number 存入 type 為 Frame 的 vector(lfu_lru)中，記錄其無 page fault 也無 page replacement 發生。

(4) 若不在 page frame 內，則紀錄其發生 page fault，並看 page frame 是否已滿。

(5) 若未滿就將 counter 數設為 0，並把 page number 與 counter 加到 page frame 中，並將目前的 page frame 以及 page number 存入 lfu_lru 中。

(6) 若已滿就將 page frame 內 counter 數最小的 page 刪除，若 counter 數相同則將 index 靠前的 page 刪除，再將此時的 page number 將入 page frame 中，且將目前的 page frame 以及 page number 存入 lfu_lru 中，並記錄其發生 page replacement。

(7) 最後將此次發生 page fault 與 page replacement 的次數存到 vector 中。

6. MFU + LRU :

(1) 宣告一個 type 為 List 的 vector 作為 page frame。

(2) 當有 page 要被載入時，先檢查是否有存在在 page frame。

(3) 若有就將此筆資料刪除，並將 counter 加一與 page number 一起重新加入 page frame 內，且將目前的 page frame 以及 page number 存入 type 為 Frame 的 vector(mfu_lru)中，記錄其無 page fault 也無 page replacement 發生。

(4) 若不在 page frame 內，則紀錄其發生 page fault，並看 page frame 是否已滿。

(5) 若未滿就將 counter 數設為 0，並把 page number 與 counter 加到 page frame 中，並將目前的 page frame 以及 page number 存入 mfu_lru 中。

(6) 若已滿就將 page frame 內 counter 數最大的 page 刪除，若 counter 數相同則將 index 靠前的 page 刪除，再將此時的 page number 將入 page frame 中，且將目前的 page frame 以及 page number 存入 mfu_lru 中，並記錄其發生 page replacement。

(7) 最後將此次發生 page fault 與 page replacement 的次數存到 vector 中。

三、不同方法之間的比較:

Page frame = 3

input1	FIFO	LRU	LFU+FIFO	MFU+FIFO	LFU+LRU	MFU+LRU
Page Fault	9	10	10	9	10	9
Page Replacement	6	7	7	6	7	6

Page frame = 3

Input2	FIFO	LRU	LFU+FIFO	MFU+FIFO	LFU+LRU	MFU+LRU
Page Fault	15	12	13	15	11	12
Page Replacement	12	9	10	12	8	9

FIFO : FIFO 會將在 memory 內存在最久的 page 置換掉，但是會在 memory 內待很久的 page 有可能會是重要的 page 或是很常用到的 page，所以有可能會造成 page fault 與 page replacement 次數增加，但是若將 page frame 增加也不一定能降低 page fault 與 page replacement 的產生，反而有可能會增加，這就是所謂的畢雷笛反例。

LRU : LRU 是將"最近"較少被用到的 page 做置換。由於最近較少用到此 page，所以我們可以推斷最近應該不太會使用到此 page，故可先將此 page 置換成目前較常用到的 page，以此來減少 page fault 與 page replacement 的產生。且由上表可知，使用 LRU 所發生的 page fault 與 page replacement 次數相較於其他的方法而言，發生次數較少。

LFU + FIFO : 此方法是將 memory 內使用次數較少的 page 做置換。由於使用次數較少，所以我們可以推斷此 page 應該是不常用到的 page，我們便可以將其置換成其他較常用到的 page，來減少 page fault 與 page replacement 的次數。但此方法對新加入的 page 較不友善，因為新加入的 page 的 counter 值一定較少，很容易被置換出去。
搭配 FIFO : 當使用次數相同時，則是將待在 memory 的時間較長的 page 做置換，故也有可能將未來可能會更常用到的 page 置換出，但是由於會先看 counter 數值，故可降低 page fault 與 page replacement 的次數。
搭配 LRU : 當使用次數相同時，則是將最近較少用到的 page 做置換，由於 LRU 會以最近的 page 使用情況作為判斷依據，故可以有效降低 page fault 與 page replacement 的次數。

MFU + FIFO : 此方法是將 memory 內使用次數較多的 page 做置換。由於使用次數較多，我們可以推測其應該在 memory 內待很久了，所以我們可以推斷此 page 應該已經做完事情了，我們便可以將其置換成其他 page。但是在 memory 內存在很久的 page 有可能很重要，若將其置換出去，有可能很快的又要在置換回來，反而造成 page fault 與 page replacement 次數增加。
搭配 FIFO : 當使用次數相同時，則是將待在 memory 的時間較長的 page 做置換，但是資深的 page 有可能很重要，故有可能還是要將其置換回來，反而會增加 page fault 與 page replacement 次數。
搭配 LRU : 當使用次數相同時，則是將最近較少用到的 page 做置換，由於 LRU 會以最近的 page 使用情況作為判斷依據，故可以有效降低 page fault 與 page replacement 的次數。

四、未完成的功能:
無