1. 開發環境 :

作業系統 : windows

使用語言 : C++

測試環境 : Dev C++

1. 程式設計 :

Data structure :

全域變數:

int page\_frame ; // 儲存讀入的page frame

vector<int> page\_ref ; // 儲存讀入的page

區域變數:

int fault, replaces = 0 ; // 用來存各自的page fault、page replaces

vector<int> buffer ; // 用來存目前正被使用的page

bool f ; // 判斷是否有page fault，有的話要印出 ‘F’

bool stored ; // 用來判斷目前加入的page是否有在buffer 中，若有的話，在LRU中會有另外做處理

vector<int> counter ; // 在LFU、MFU、LFU+LRU、MFU+LRU中使 用，用來計算網頁的使用次數

function:

int main() // 在成功讀檔後執行六種方法

bool Read() // 讀檔

void FIFO() // 從第一個page開始，先判斷是否有在buffer中，如 果沒有的話( !stored )，將f 設為 true、fault+1，如果 buffer的大小小於page\_frame，就將此page放到buffer 的最前面；若buffer的大小等於page\_frame，就將 buffer的最後一個移除，把此page放到buffer的第一 個。印出目前的page、buffer的所有page，如果f =true 則印出F，重複下一個page，直到所有page都執行完， 最後再印出page fault、replaces。

void LRU() // 與FIFO幾乎相同，唯一的差別在於當目前的page有 在buffer中的話 (store == true)，要將此page移到buffer 中的第一個，其他步驟街與FIFO相同。

vector<int> MakeCounter() // 建立一個大小為page\_ref中最大值得 vector，且全部預設為0。

void LFU() // 與FIFO相似，差別在於，當buffer的大小等於 page\_frame時，是將buffer中的page在counter中最小 的移除，若相同則取最早加入的，再將目前的page放到 buffer的最前面，並將目前page在counter中的位置設 為0。當此page的處理結束後，會將此page在counter 的位置的值加一。其餘步驟皆與FIFO相同。

void MFU() // 與LFU相同，只差在要移除時是取buffer在counter中最大的。

void LFU\_LRU() // 與LFU幾乎相同，唯一的差別在於當目前的page有 在buffer中的話 (store == true)，要將此page移到buffer 中的第一個，其他步驟皆與LFU相同。

void MFU\_LRU() // 與LFU\_LRU相同，只差在要移除時是取buffer在 counter中最大的。