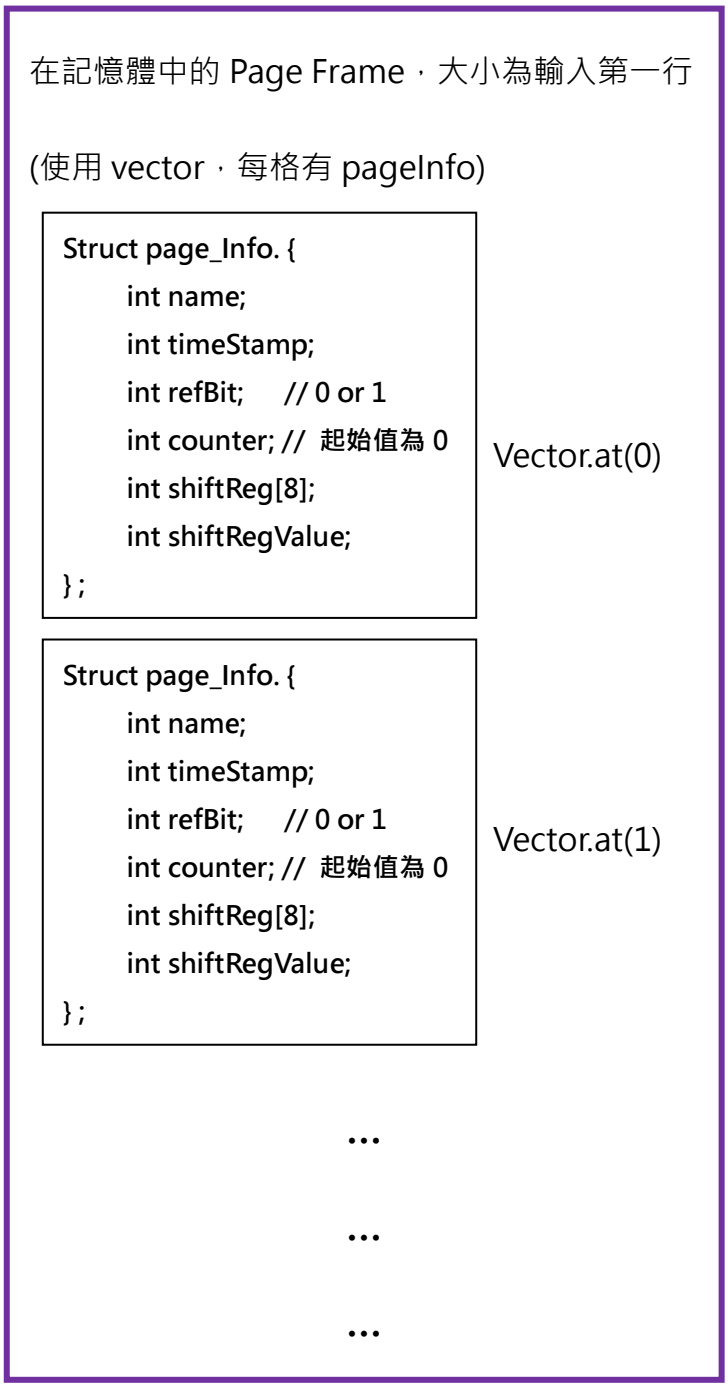
10427138 OS_程式作業 3 說明文件

1. **開發環境** : Windows10

2. **開發平台** : Dev-C++ 5.11 、Visual Studio Community 2017

3. **資料結構** :

   運算模擬結構 :



輸出結構 ：

存結果，Vector 結構

```
struct reSult {   // 存上半部結果 步驟輸出
      int ref;
      vector<pageInfo> currentFrame;
      //每次 reference，
      //都把當前 Page Frame 整個存進來
      char pageFault;   // 空或'F'
} reSult;
```

Vector.at(0)

```
struct reSult {   // 存上半部結果 步驟輸出
      int ref;
      vector<pageInfo> currentFrame;
      //每次 reference，
      //都把當前 Page Frame 整個存進來
      char pageFault;   // 空或'F'
} reSult;
```

Vector.at(1)

…

…

…

存結果，單一 struct

存放該作法的 ：

Page Faults、

Page Replace、

Page Frames 。

4. 程式功能、流程 ：

將 page reference 的順序讀入並保存後，進入各個做法的 function 開始做處理，期間，各功能都可以直接將結果存入分別的結果結構中，並且累加頁錯誤跟頁置換到輸出結構 2 中。

功能性 function :

int isInFrame(int No) ;

// 輸入指定的頁編號，回傳其在 Page Frame 中的位置

int hasSpace() ;

// 檢查記憶體中的 Page Frame 還有沒有空間，若有，回傳那個位置

void Shhhift() ;　void CalcuteShiftValue() ;

//作暫存器的位移　　// 計算各暫存器的值

void PageShift() ;

//將 Frame 中的所有 page 往左位移一個，替換掉最左邊的 page

void Put_samllest_to_the_leftMost() ;

//將某個特別的頁移至左邊準備被替換

在 FIFO 中，我將每次 ref.的頁標上 time stamp，每次有頁被參考，就用泡沫排序法整理出最小的 stamp，並將頁替換掉。

在 LRU 中，類似於 FIFO 的作法，唯獨每次有頁被參考，就加上新的 time stamp。

在 Additional Reference Bits 中，每次有頁被參考，就令其的 refBit 為 1，並且在下頁 ref 進來前，所有人的 register 位移一次，並將所有人的 refBit 作 reset； 發現 Page Replace 時，將最小 register 值的 page 移到最左邊，並將整個 Page Frame 往左位移一次，新頁加至最右邊。

在 Second Chance 中，用 refBit 來記錄這個 page 是否擁有第二次機會，若有，回到 Page Frame 最右邊，並失去這個機會，若無，則是可以置換的對象，在發現 page replace 時，我們需要不斷執行上述的機會運算，直到有 page 能被替換掉為止。

在 Least Frequently Used 和 Most Frequently Used 中，每個 page frame 會適時的計算 counter，page frame 作左位移前，先將最小/最大的 counter 值的 page 移到最左邊，再左位移替換掉，新頁放在最右邊。

5. 未完成功能 ： 無