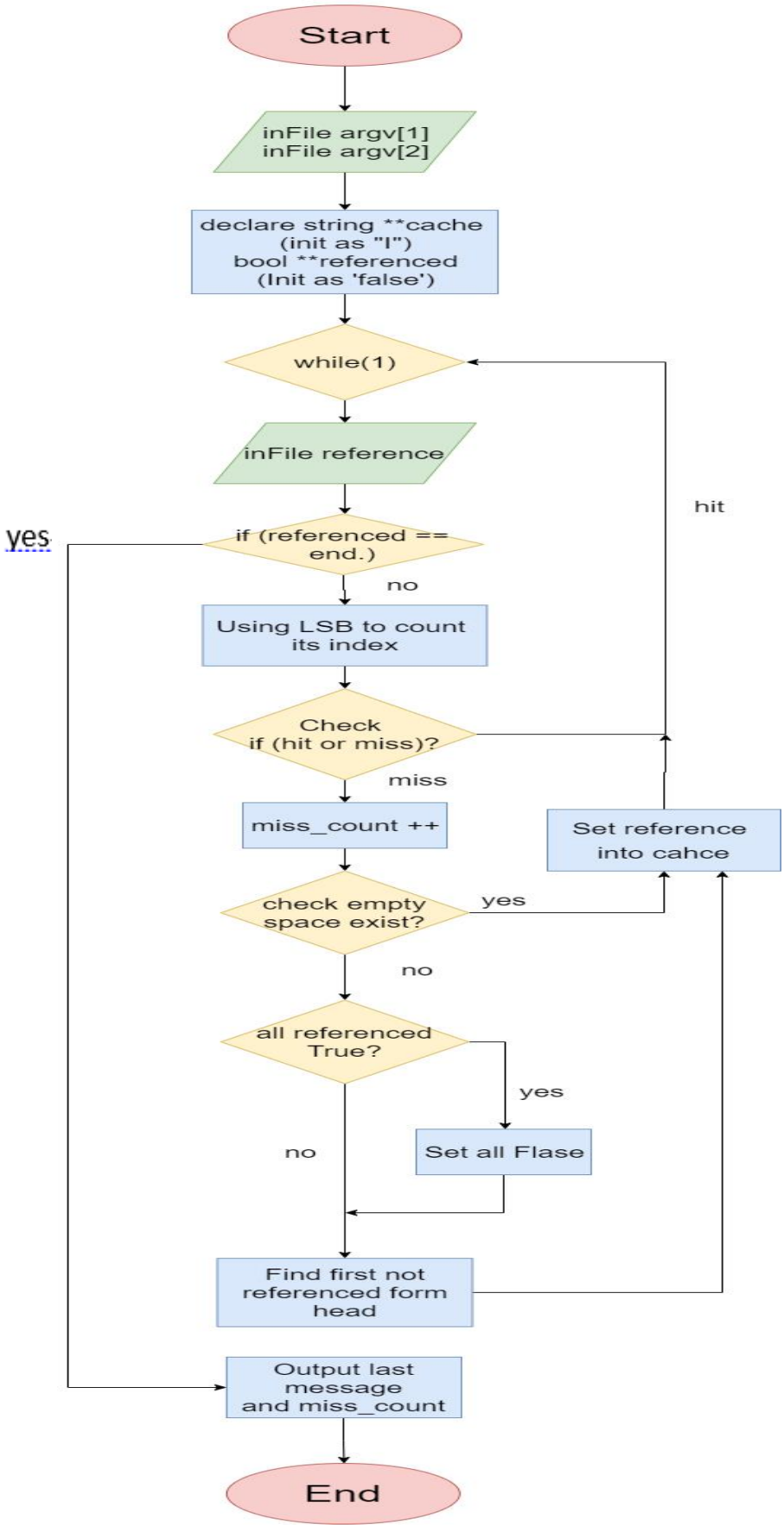
FlowChart


Start
inFile argv[1]
inFile argv[2]
declare string **cache
(init as "I")
bool **referenced
(Init as 'false')
while(1)
inFile reference
if (referenced ==
end.)
yes
no
Using LSB to count
its index
Check
if (hit or miss)?
hit
miss
miss_count ++
Set reference
into cahce
check empty
space exist?
yes
no
all referenced
True?
yes
no
Set all Flase
Find first not
referenced form
head
Output last
message
and miss_count
End

Detail description

1. 2-D array : string **cache:

Cache 二維陣列主要用來儲存近期所使用的資料，row 儲存的是 index 的數字，column 儲存的則是 data，一開始會先將所有的內容初始化為 “ I ” 代表此空間尚未被使用過。

2. 2-D array : bool **referenced:

Referenced 二維陣列主要是儲存每一筆資料所使用的狀況，根據 NRU replacement policy 會先儲存 referenced bit，而我使用的是用 bool 來儲存，如果該筆資料被參考過，就將其設為 True 否則就設為 False。

3. Variable : string reference:

Reference 變數主要為用來存取參考資料，並透過 index 計算後，與 cache index 值相同的 row 進行 tag 比對。

4. Variable : int total_cache_miss_count:

用來存取 cache miss 的次數。

Program flow description

根據 FlowChart 所繪，一開始會先將檔案名稱做為參數的 argv[1]、argv[2]讀入，再以分別將這些檔案透過以下方式開啟。

```
ifstream inFile_cache(argv[1], ios::in);
ifstream inFile_reference(argv[2], ios::in);
```

接下來進入迴圈，首先透過 reference 計算出 index 位於 cache 的哪一個 row，接下來比對 tag，判斷是否有 ”hit” ，若有 hit，就將該 referenced 位置設為 true，若為 miss，就先判斷這個 row 是否仍然存在空的空間，如果存在就以空的空間為優先放置，若依然沒有空間可以放置資料，則先搜尋這個 row 的 referenced 是否全部為 True，若全部都為 True，就將其全部重新設為 False，若執行到這，表示這個 row 的空間已經沒有不曾被放置過的空間了，接下來就從 head 開始搜尋到第一個 referenced 為 false 的，然後將他置換掉。