struct stuInfo｛｝； 存儲一個學生的所有信息

struct hashField ｛｝；hashTable中的每一個 item。存儲一個學生的所有信息，以及以該學生學號算出來的hashKey

讀取二進制文檔：

vector<StuInfo>& ReadBinaryFile(vector<StuInfo> &stuInfoVector ,string outPutFileName)

該函數將二進制文檔讀取，並存入結構體StuInfo中，在讀取時，使用了ifstream進行讀取，以結構體為單位逐個讀取使用的函數是read，將結構體傳入read中，讓其自動讀取並填充。

任務二上下文函數

Procedure(string fileName)

讀入用戶輸入的文件名，而後進行Mission2的上下文操作。創建此函數的目的是更方便地合併到任务一的 main（）函數中，解耦。

判斷一個數字是否為素數

bool JudgeIfPrime（int judgeNumber）

傳入一個 integer，用數學方法判斷其是否為素數。是返回 true，否則返回 false

創建hashTable函數

void CreateHashTable（vector<StuInfo> stuInfoVector，string fileName）

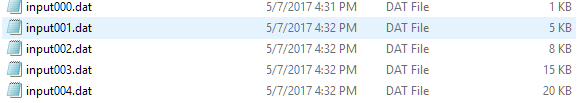
傳入讀取的所有學生信息vector，根據每一個學生信息的sid號，存放到對應的hashTable位置。

計算學號各個字母Ascii值的乘積

double CalculateSidAscii（char\* sid）；

傳入一個數字字母組成的字符串，計算每個字母對應的Ascii值的乘積。

我自行組裝了5個2進位格式的學生資料檔



Input000.dat共16筆學生資料，

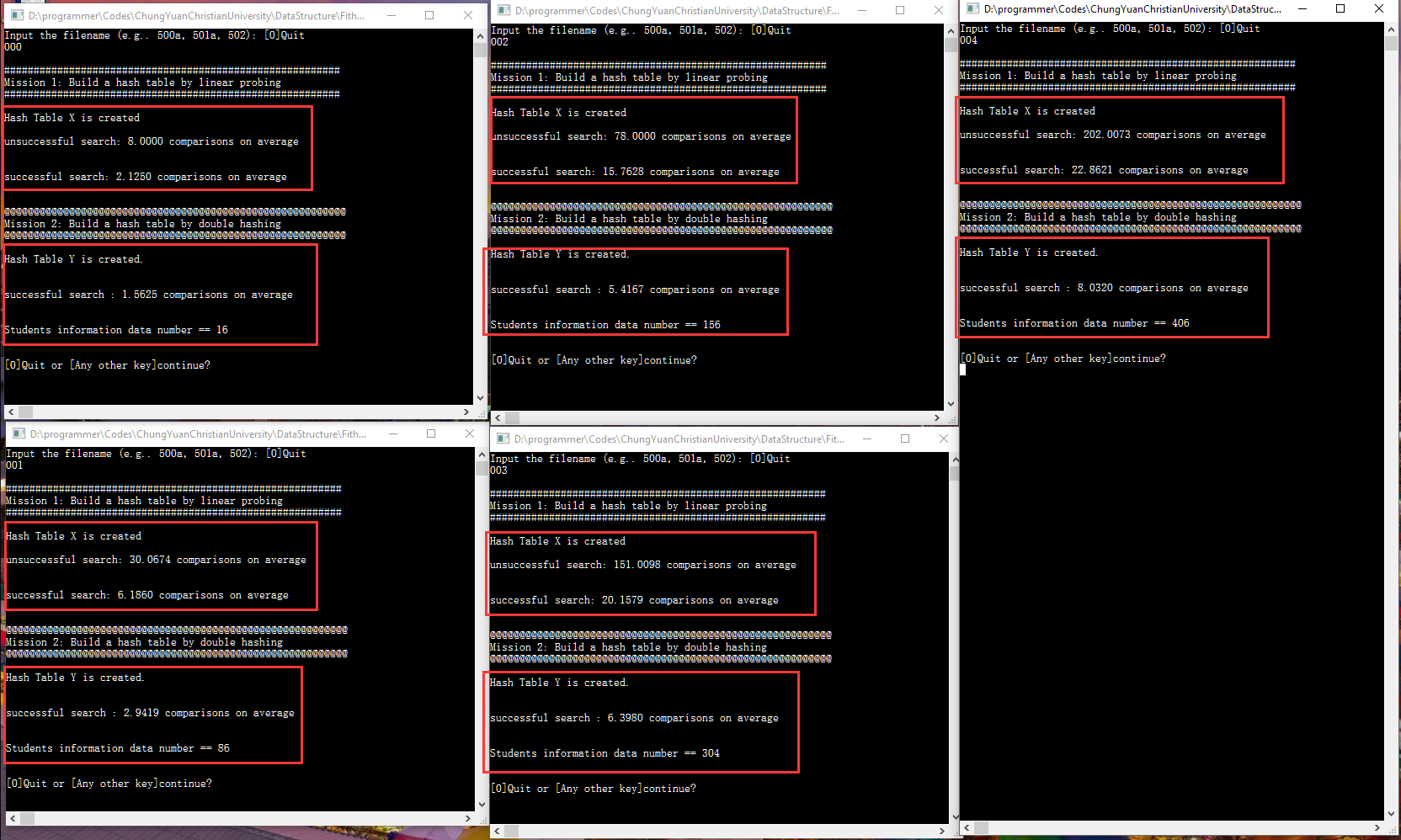
Input001.dat共86筆學生資料，

Input002.dat共156筆學生資料，

Input003.dat共304筆學生資料，

Input004.dat共406筆學生資料

運行結果如圖所示



**建表如下**

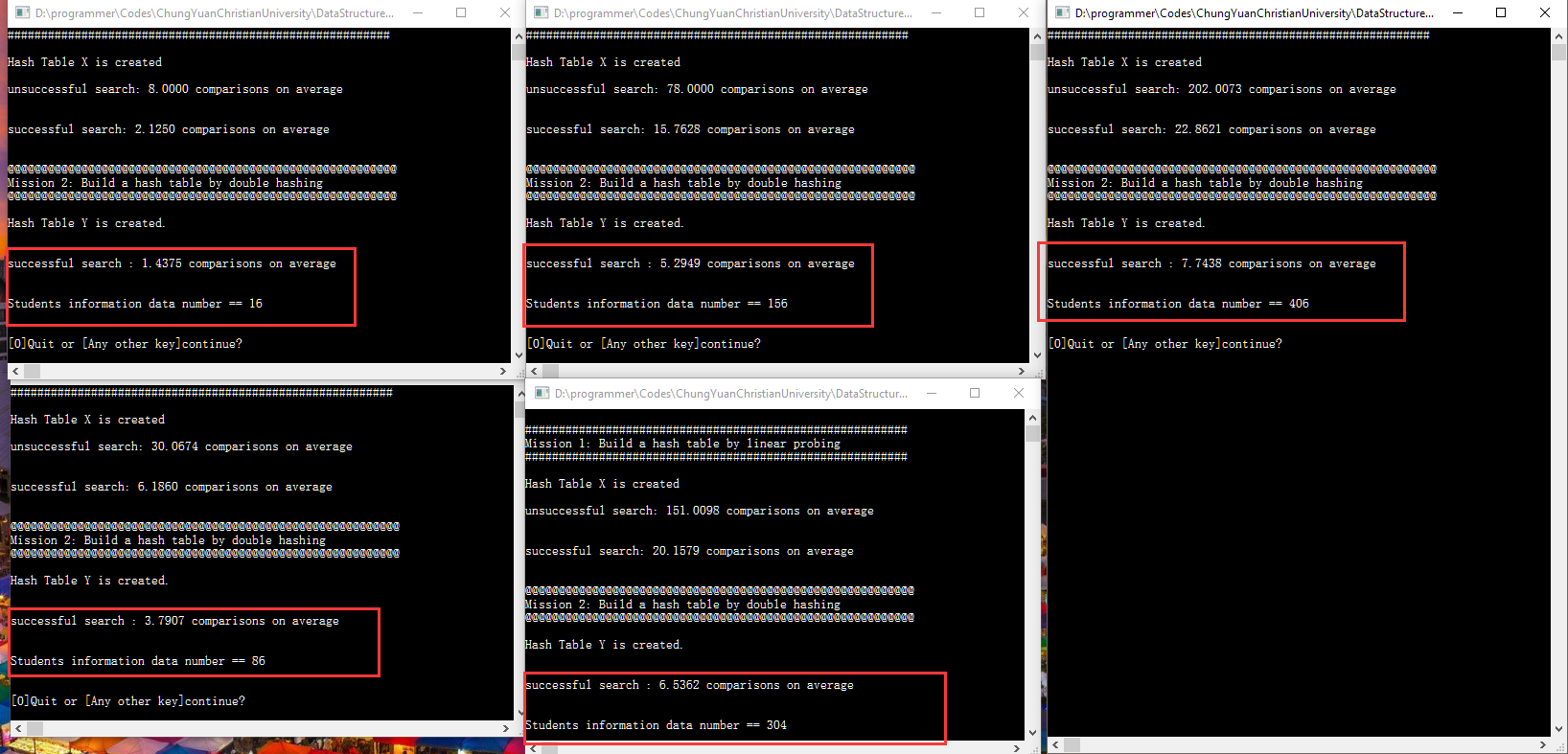
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Data Number** | **Linear Probing** | | **Double Hashing** |
|  |  | **Unexisted Search** | **Existed Search** | **Existed Search** |
| Input000.dat | 16 | 8.0000 | 2.1250 | 1.5625 |
| Input001.dat | 86 | 30.0674 | 6.1860 | 2.9419 |
| Input002.dat | 156 | 78.0000 | 15.7628 | 5.4167 |
| Input003.dat | 304 | 151.0098 | 20.1579 | 6.3980 |
| Input004.dat | 406 | 202.0073 | 22.8621 | 8.0320 |

依表可知，在 Existed Search 上，Double Hashing要明顯優於 Linear Probing。

**更改stepSize函數為**

stepSize(key) = 最高步階 - （（學號每個字母對應數字的和） 除以 最高步階 取餘數）

運行結果如下



**建表如下**

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Data Number** | **Double Hashing** | **Double Hashing（change step function）** |
|  |  | **Existed Search** | **Existed Search** |
| Input000.dat | 16 | 1.5625 | 1.4375 |
| Input001.dat | 86 | 2.9419 | 3.7907 |
| Input002.dat | 156 | 5.4167 | 5.2949 |
| Input003.dat | 304 | 6.3980 | 6.5362 |
| Input004.dat | 406 | 8.0320 | 7.7438 |

從表格中可以看出，使用更改後的 step（key）函數，在資料量為 16、156、406時，表現好過使用原來的 step（key）函數。

但是由於對比資料太少，此結論僅供參考，沒有足夠的理論依據。