Mục lục

[Bảng phân công 1](#_Toc89631171)

[Bảng đánh giá 1](#_Toc89631172)

[Câu 2B các hạn chế: 1](#_Toc89631173)

[1. Sector 2](#_Toc89631174)

[a) Xác định các thông số quan trọng. 2](#_Toc89631175)

[b / Viết chương trình đưa vào (thư mục gốc) N tập tin F0.Dat, F1.Dat,… F<N>.Dat với nội dung của F<K>.Dat là các dòng văn bản mà mỗi dòng đều là giá trị K và chiếm (2 - K%2) cluster. 4](#_Toc89631176)

[c) Dự đoán (có lý giải) số cluster của RDET khi: 4](#_Toc89631177)

[d) Xóa toàn bộ các file F<K>.Dat với K chẵn 4](#_Toc89631178)

[# Cứu file F0.dat 5](#_Toc89631179)

[f) Chương trình phục hồi tất cả file 6](#_Toc89631180)

[Tài liệu tham khảo: 6](#_Toc89631181)

[CÂU 2: A/ 7](#_Toc89631182)

[2.1 Yêu cầu hệ thống 8](#_Toc89631183)

[2.2 Yêu cầu người dùng 8](#_Toc89631184)

[2.3 Thiết kế cluster 8](#_Toc89631185)

[2.4 Bảng quản lí cluster: 2 bảng giống nhau 8](#_Toc89631186)

[2.5 RDET 9](#_Toc89631187)

[2.6 ReadBlock/WriteBlock 9](#_Toc89631188)

[2.7 Cách ghi tập tin vào các khối Block 10](#_Toc89631189)

[2.8 Chức năng chống phân mảnh 10](#_Toc89631190)

[2B) Chọn thiết kế dễ hơn để làm 11](#_Toc89631191)

[2.1 Boot sector 11](#_Toc89631192)

[2.3 Bảng RDET 11](#_Toc89631193)

[2.4 Các hàm chính 11](#_Toc89631194)

[2.5 Chạy thử: 12](#_Toc89631195)

# Bảng phân công

|  |  |  |
| --- | --- | --- |
| MSSV | Họ tên | Nhiệm vụ |
| 19120637 | Đỗ Xuân Sang | Câu 1 |
| 19120652 | Nguyễn Trọng Thái | Câu 2 |

# Bảng đánh giá

|  |  |  |
| --- | --- | --- |
| Câu | Hoàn thành yêu cầu hàm | Hoàn thành yêu cầu chức năng |
| 1 | 100% | 100% |
| 2A | 100% | 100% |
| 2B | 100% | 80% |

# Câu 2B các hạn chế:

* Việc ghi rất chậm, do mỗi lần cập nhật đều đọc hoặc ghi toàn bộ MYFS.dat => do vậy chỉ thử nghiệm được với dữ liệu nhỏ
* Việc nhìn vào file đọc dữ liệu rất dễ do ghi bằng text
* Có cài mật khẩu nhưng cũng không mã hóa tập tin
* Chỉ làm trên thư mục gốc
* Hàm Outport: tức hàm đọc file ghi ra file khác. Đọc được nhưng file mới lại thừa dữ liệu. Không biết xử lý như thế nào, dù sử dụng hàm length để đo độ dài ví dụ: 1000 mà ghi vào file nó lại nhiều hơn.
* 1 hạn chế rất lớn nữa là: ghi vào file theo ký tự chứ không phải theo bit: ví dụ ghi số 255 chỉ tốn 1 byte mà phải tốn tới 3 bytes để ghi ‘2’,’5’,’5’

# 1. Sector

## a) Xác định các thông số quan trọng.

Ảnh có chứa văn bản

Mô tả được tạo tự động

* Số byte của một sector: 512

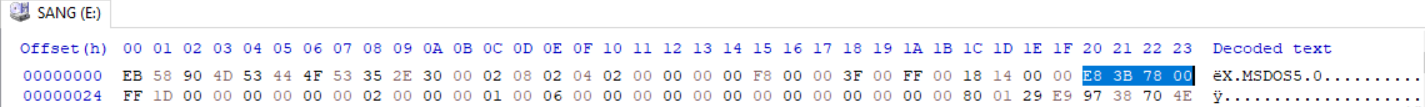
(2 byte tại offset 0B là: 00 02)

Ảnh có chứa văn bản

Mô tả được tạo tự động

* Kích thước volume: 4 034 MegaByte và 3 847 MebiByte

(4 byte tại offset 20 là: E8 3B 78 00)



* Số sector trước FAT: 1026

(2 byte tại offset 0E là: 02 04)

Ảnh có chứa văn bản

Mô tả được tạo tự động

* Số sector của một bảng FAT: 7679

(4 byte tại offset 24 là: FF 1D 00 00)

Ảnh có chứa văn bản

Mô tả được tạo tự động

* Số bảng FAT: 2

(1 byte tại offset 10 là: 02)

Ảnh có chứa văn bản

Mô tả được tạo tự động

* Số entry (hiện tại) của RDET: (16448-16384)\*512/32=1024
* Số sector của một cluster: 8 (\*)

(1 byte tại offset 0D là: 08)

Ảnh có chứa văn bản

Mô tả được tạo tự động

* Số sector của volume: 7 879 656 (\*\*)

1. byte tại offset 20 là: E8 3B 78 00)

Ảnh có chứa văn bản

Mô tả được tạo tự động

* Số cluster của volume: 7 879 656 / 8 = 984 957

Trừ đi 16384 sectors của reserved sector, fats and root cluster.

Bằng với 16384 / 8 = 2046 cluster

* Số cluster sử dụng của volume: 984 957 – 2046 = 982 911
* Công thức tính vị trí của cluster (theo sector):

SectorofCluster = (N - 16384)/8 ;

Với 16384 là sector bắt đầu.

## b / Viết chương trình đưa vào (thư mục gốc) N tập tin F0.Dat, F1.Dat,… F<N>.Dat với nội dung của F<K>.Dat là các dòng văn bản mà mỗi dòng đều là giá trị K và chiếm (2 - K%2) cluster.

* Khởi tạo với tham số N được nhập từ cmd

@echo off

for /l %%i in (0 2 %1) do fsutil file createnew f%%i.dat 512

for /l %%i in (1 2 %1) do fsutil file createnew f%%i.dat 1024

for /l %%i in (0 1 %1) do echo %%i > f%%i.dat

for /l %%i in (0 1 100) do ( for /l %%i in (0 1 %1) do echo %%i >> f%%i.dat )

* Khởi tạo với tham số N được chọn sẵn trong file.cmd

@echo off

set /a count=5

for /l %%i in (0 2 %count%) do fsutil file createnew f%%i.dat 512

for /l %%i in (1 2 %count%) do fsutil file createnew f%%i.dat 1024

for /l %%i in (0 1 %count%) do echo %%i > f%%i.dat

for /l %%i in (0 1 100) do ( for /l %%i in (0 1 %count%) do echo %%i >> f%%i.dat )

## c) Dự đoán (có lý giải) số cluster của RDET khi:

* N=11
* Dễ thấy, với K là số chẵn sẽ chiếm 2 cluster, còn K là số lẻ thì chiếm 1 cluster.
* Một cặp 2 tập tin liền kề nhau (1 chẵn, 1 lẻ) sẽ chiếm 3 cluster.
* Với N=11, ta thấy có 6 cặp như thế. Suy ra có 6\*3=18 cluster.
* 18 cluster
* N=2021
* Tương tự như trên, với N=2021, sẽ có 1011 cặp tập tin.
* Suy ra sẽ chiếm 1011\*3 = 3033 cluster.
* 3033 cluster

## d) Xóa toàn bộ các file F<K>.Dat với K chẵn

@echo off

set /a count=0

for %%x in (\*) do set /a count+=1

set /a count=count-3

for /l %%i in (0 2 %count%) do del f%%i.dat

* Cũng có thể xóa từng file F<K> với K được nhập từ cmd

@echo off

del f%1.dat

* Hoặc xóa những file F<K> chẵn, từ file F<X> đến file F<Y>, X, Y được nhập từ cmd (X<Y)

@echo off

for /l %%i in (%1 2 %2) do del f%%i.dat

## # Cứu file F0.dat

* Tính toán sector bắt đầu: 2 fat (mỗi fat 7679 sector) có 15358 sector, cộng thêm 1026 sector trước fat. 7679\*2+1026=16384 sector.
* Chuyển đến sector bắt đầu, sector 16384. Cứu lại tên của F0. Sửa E5 thành 46.

Ảnh có chứa bàn

Mô tả được tạo tự động

* Chuyển đến bản fat thứ nhất nắm ở sector 1026. Phục hồi lại nó.

A picture containing graphical user interface

Description automatically generated

A picture containing calendar

Description automatically generated

* Tiếp tục đến bản fat thứ hai nắm ở sector 8705 và phục hồi nó.

Ảnh có chứa văn bản

Mô tả được tạo tự động

Calendar

Description automatically generated with low confidence

* Vậy là chúng ta đã phục hồi xong file F0.dat

Ảnh có chứa văn bản

Mô tả được tạo tự động

## f) Chương trình phục hồi tất cả file

* Chương trình cmd.

chkdsk E/f

attrib –h –r –s /s /d \*.\*]

* Chương trình python.

Đính kèm trong thư mục Source.

## Tài liệu tham khảo:

Trang wikiHow.vn

https://www.wikihow.vn/T%E1%BA%A1o-v%C3%A0-x%C3%B3a-t%E1%BA%ADp-tin-ho%E1%BA%B7c-th%C6%B0-m%E1%BB%A5c-t%E1%BB%AB-Windows-Command-Prompt

Trang completejavascript.com

https://completejavascript.com/cac-cau-lenh-thuong-dung-tren-cmd-windows/

Trang github.com

https://github.com/amooose/NTFS-File-Recovery-Flagger

Trang youtube TechkyLover

https://www.youtube.com/watch?v=o4JQyi6gNp4

Trang prgwonders.blogspot.com

https://prgwonders.blogspot.com/2016/07/python-script-to-recover-deleted-files.html

# CÂU 2: A/

Xây dựng một mô hình và thiết kế kiến trúc tổ chức cho một hệ thống tập tin được chứa trong file MyFS.Dat có sẵn trên HĐH (tương tự như 1 file .ISO / .ZIP / .RAR). Các tiêu chí thiết kế như sau:

1/ Việc bảo mật thông tin (tránh bị lộ nội dung 1 số tập tin quan trọng) được xem là thiết yếu nhất.

2/ Việc an toàn dữ liệu (tránh hư hỏng /mất mát nội dung các tập tin quan trọng) cũng rất cần thiết.

3/ Tốc độ truy xuất tập tin cũng cần đạt mức không thấp hơn quá nhiều so với tốc độ truy xuất các tập tin bên ngoài của HĐH (và cũng ưu tiên cho việc đọc ra hơn là ghi vào).

4/ Số tập tin cần tổ chức trong MyFS.Dat có thể đủ nhiều /đa dạng để phải tổ chức hệ thống thư mục phân cấp.

5/ Kích thước tập tin trong MyFS không quá lớn (100MB) sẽ là không quan trọng và không đòi hỏi phải đáp ứng tốt 3 tiêu chí đầu.

6/ Nội dung các byte bên trong file MyFS.Dat cần được truy xuất theo từng cụm 512 byte giống như sector trên các volume của HĐH hoặc các đĩa của máy tính. (tức cần thiết kế & xây dựng các hàm ReadBlock / WriteBlock giống các hàm ReadSector / WriteSector của HĐH & máy tính)

|  |  |
| --- | --- |
| Phân vùng | Gồm |
| Reserved sectors | Boot sector |
| Chương trình nạp HĐH |
| … |
| Bảng quản lý cluster | Bảng FAT1 |
| Bảng FAT2 |
| RDET | Các entry |
| Data |  |

Các vấn đề sẽ thiết kế:

-Yêu cầu hệ thống

-Yêu cầu người dùng

-Thiết kế cluster

-Bảng cluster

-Directory file

-ReadBlock/WriteBlock

## 2.1 Yêu cầu hệ thống

* **Để đáp ứng yêu cầu 1,2,3**

1. Có mật khẩu và phân quyền cho các đối tượng. Khi cài đặt mật khẩu thì cả thông tin mật khẩu sẽ được mã hóa để cho người khác không thể đọc từ quản lý volume, đồng thời tất cả thông tin liên quan sẽ được mã hóa tương ứng với mật khẩu
2. Thiết kế 2 bảng quản lý cluster để có thể khôi phục dữ liệu
3. Đưa bảng quản lý cluster lên trên RAM. Bảng quản lý cluster quản lý dưới dạng xâu kết hợp chỉ mục nên khi truy cập cluster bắt đầu thì sẽ biết ngay cluster tiếp theo để tiếp tục truy cập. 1 cluster = 8 sector nên số lần đọc ít hơn.

## 2.2 Yêu cầu người dùng

- Chống phân mảnh: khi người dùng muốn lưu dữ liệu tại các Block liên tục. Có thể chọn tập tin mong muốn, hoặc chống phân mảnh trên tất cả các tập tin

- Liệt kê danh sách các tập tin: các tập tin sẽ hiển thị dưới dạng cây với các thông số: tên+phần mở rộng, vị trí lưu, kích thước

- Đổi mật khẩu tập tin: khi người dùng muốn đổi mật khẩu phải nhập đúng mật khẩu cũ và nhập mật khẩu mới 2 lần.

- Phân quyền cho từng loại người dùng: khi tạo file or chia sẻ file sẽ cài đặt các chế độ tùy theo người dùng như: đọc, ghi, chỉnh sửa, bảo mật.

- Xóa và khôi phục : xóa tập tin, xóa thư mục. Xóa nhưng không mất, xóa bảo mật (xóa mất hẳn luôn)

- Đọc, ghi, chỉnh sửa

- Xem thông tin chi tiết,…..

## 2.3 Thiết kế cluster

- Quy định 1 cluster = 8 sector = 8x512 = 4096 bytes. Do 1 file không quá 4GB nên sử dụng mô hình giống FAT32 sẽ tốt hơn

- Cluster bắt đầu bằng cluster 0

- Mỗi cluster chỉ chứa 1 tập tin

## 2.4 Bảng quản lí cluster: 2 bảng giống nhau

- cluster 0 và 1 không lưu tập tin

- Quản lý các cluster của tập tin bằng xâu kết hợp chỉ mục, mỗi phần tử lưu vị trí cluster tương ứng, có N cluster sẽ có N phần tử lưu với giá trị:

+ FREE/BAD: nếu không được lưu dữ liệu thì sẽ lưu 1 trong 2 giá trị này tương ứng với cluster trống và cluster bị hư

+ Một tập tin khi lưu: vị trí cluster bắt đầu. Tại vị cluster bắt đầu chứa số nguyên là cluster tiếp theo tương tự vậy cho đến khi gặp kí hiệu kết thúc là EOF với EOF là dấu hiệu kết thúc của 1 tập tin được lưu

+ Nếu tập tin chỉ có 1 tập tin thì phần tử quản lí chứa chỉ số bắt đầu = kí hiệu kết thúc = EOF

- Chúng ta sẽ sao chép để có 2 bảng quản lý cluster giống nhau để có thể phục hồi nếu 1 trong 2 bảng bị hư

## 2.5 RDET

- Là tập tin đặc biệt để ghi các thông số của thư mục gốc

- Các thuộc tính của file:

+ name: tên chính

+ phần mở rộng

+ is directory: xem là thư mục hay tập tin

+ created: ngày tạo gồm thời gian và ngày

+ last modified: lần cuối chỉnh sửa gồm thời gian và ngày

+ owner: người sở hữu

+ mode: các quyền đối với tập tin

+ size: độ dài dữ liệu chứa nội dung tập tin

+ Password: nếu ở phần mode chúng ta cài đặt cần mật khẩu thì sẽ lưu mật khẩu 8 ký tự(8B)

- Các thuộc tính của bảng RDET: tổng 32 bytes(COPY FAT32)

|  |  |  |
| --- | --- | --- |
| offset | Size (bytes) | Tên |
| 0 | 8 | Tên tập tin |
| 8 | 3 | Phần mở rộng |
| 11 | 1 | Thuộc tính(mode) |
| 12 | 10 | Để trống |
| 22 | 2 | Created |
| 24 | 2 | last modified |
| 26 | 2 | Block bắt đầu |
| 28 | 4 | Size file |

- Khi các file có thay đổi ta phải cật nhật cả bảng RDET

## 2.6 ReadBlock/WriteBlock

- Giả sử có hàm **bool ReadSector (int vol\_ , int sec\_ , int num\_ , char \* buf\_ )** đọc <num\_>  sector  trên volume <vol\_>  bắt đầu tại sector <sec\_> vào vùng nhớ <buf\_>

bool ReadBlock(int vol\_, int Block\_, int num\_, char\* buf){//Số sector cần đọc= số Block\*8

int sec\_ = Block\_ \* 8;

if (!ReadSector(vol\_, sec\_, num\_ \* 8, buf))

return false;

return true

}

+ Trong hệ điều hành có hỗ trợ hàm đọc sector, khi muốn đọc 1 khối block ta chỉ cần đọc 8 sector liên tiếp tương ứng

+ Vị trí sector bắt đầu đọc = <Block>\*8 vì 1 Block = 8 sector

+ Số lượng sector cần đọc = <Số lượng Block muốn đọc> \* 8

-Hàm WriteBlock():

+ Chúng ta sẽ dựa bảng quản lí cluster để biết tình trạng của các cluster để ghi vào.

+ Hệ điều hành có sẵn hàm đọc sector nên khi muốn ghi vào 1 khối block ta chỉ cần ghi vào 8 sector liên tiếp tương ứng.

+ Vị trí của sector bắt đầu ghi là: <Vị trí khối block> \* 8 vì 1 block =8 sector

## 2.7 Cách ghi tập tin vào các khối Block

- Dựa vào bảng quản lý cluster để biết cluster nào đang trống để chọn 1 cluster sau đó ghi dữ liệu vào và cật nhật cluster đó trên bảng quản lý cluster

- Tiếp tục tìm cluster trống và ghi dữ liệu vào sau đó cật nhật các phần tử quản lý cluster tương ứng để khi đọc biết đâu là khối tiếp theo

- Thực hiện như vậy cho đến khi hết dữ phần tử quản lý cluster cuối cùng của tập tin này lưu với giá trị eof

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | 0 | 0 | BAD | 0 | EOF | 0 | 0 | 0 |
| Block 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |

-VD lưu tập tin A: có kích thước KB=48/32 = 1.5 cluster. Vậy cần 2 cluster để lưu A. Giả sử chúng ta có bảng quản lý cluster như trên

- Đầu tiên chọn ra phần tử trống bỏ vào đánh dấu EOF, sau đó xét tập tin còn lại là 0.5 tiếp tục chọn 1 khối trống bỏ vào. Giả sử cluster 3. Phần tử quản lí cluster 3 có giá trị EOF

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | 0 | EOF | BAD | 0 | EOF | 0 | 0 | 0 |
| Block 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |

- Cật nhật phần tử trước đó là vị trí cluster hiện tại và phần tử hiện tại là EOF. Giả sử cluster hiện tại là 8

Phần tử quản lí cluster giá trị là 8, phần tử quản lí cluster 8 giá trị là EOF. Kiểm tra phần còn lại của tập tin, trống nên kết thúc

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | 0 | 8 | BAD | 0 | EOF | 0 | EOF | 0 |
| Block 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |

## 2.8 Chức năng chống phân mảnh

**-** Ban đầu các khối block lưu cùng dữ liệu của tập tin A là không liên tiếp. Sau đó chúng ta sẽ sao chép các khối dữ liệu sao cho các khối Block của A cho chúng liên tiếp nhau. Ví dụ:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| File A | File B | File C |  |  |  |  |  |  |

+ Sau khi thêm dữ liệu vào file A

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| File A | File B | File C | File A | File A |  |  |  |

+ Tiếp tục chúng ta sẽ sao chép dữ liệu sao cho các khối Block của file A gần nhau. Kết quả

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| File A | File A | File A | File B | File C |  |  |  |

# 2B) Chọn thiết kế dễ hơn để làm

## 2.1 Boot sector

- Phần này chứa 1 số dữ liệu cơ bản 30B

|  |  |  |
| --- | --- | --- |
| offset | Size(B) | Tên |
| 0 | 4 | Kích thước của sector(512/1024) |
| 4 | 8 | Tổng số sector |
| 12 | 1 | Số sector của cluster. Trường hợp này là 2 |
| 13 | 8 | Số khối block(cluster) |
| 21 | 1 | Chế độ có hoặc không có mật khẩu |
| 22 | 8 | Mật khẩu phải là 8 ký tự |

**2.2 Bảng FAT**

- Chiếm 2 cluster

- Mỗi phần tử quản lý vị trí cluster tương ứng: gồm có trống hoặc không trống

- Quản lý theo kiểu 1 file sẽ được lưu ở các cluster kế tiếp nhau

## 2.3 Bảng RDET

- Chiếm 4 cluster

- 8 bytes đầu chứa tổng số entry hiện tại

- Mỗi entry chiếm 52B

|  |  |  |
| --- | --- | --- |
| Offset(xét theo 52B) | Size(B) | Tên |
| 0 | 8 | Số thứ tự của entry đó |
| 8 | 16 | Tên tập tin |
| 24 | 3 | Phần mở rộng phải là 3B |
| 27 | 8 | Kích thước tập tin |
| 35 | 1 | Chế độ có hoặc không mật khẩu |
| 36 | 8 | Khối block bắt đầu |
| 44 | 8 | Mật khẩu phải là 8 ký tự |

## 2.4 Các hàm chính

- void formatVolume(int size\_volume, string password = ""): nếu PW để trống thì xem như không có PW

- string readSector(int nsect, int lsect, char\* x): trả về kiểu chuỗi

- string readBlock(int nblock, int lblock, char\* x)

- void writeSector(string buf, int nsect, int lsect)

- void writeBlock(string buf, int nblock, int lblock)

- string readFAT();//chiem 2 cluster

- void updateFAT(int nblock, int lblock, char giaTri): cập nhật lại bảng FAT giá trị là ‘0’ hoặc ‘1’

- int changePW(): Hàm này để cài đặt hoặc đổi mật khẩu cho MYFS.dat

- int get\_size\_RDET(): lấy tổng số entry hiện tại

- TapTin\* readRDET(): đọc danh sách các entry

- void addFile\_RDET(TapTin add): cập nhật lại bảng RDET

- void listFile(): xuất ra danh sách các tập tin gồm:

+ Số thứ tự, Tên tập tin”.”Phần mở rộng,size, mode, block bắt đầu

-class TapTin:

int stt;//8B

string TenTapTin;//16B

string PhanMoRong;//3B

int size;//8B

int mode;//1B

int block\_bd;//8B

string password;//8B

- bool CheckName(string name, int& vi\_tri): kiểm tra xem tên file mới có trùng tên với các file đã có không

- bool Import(string name): name là tên tập tin muốn thêm

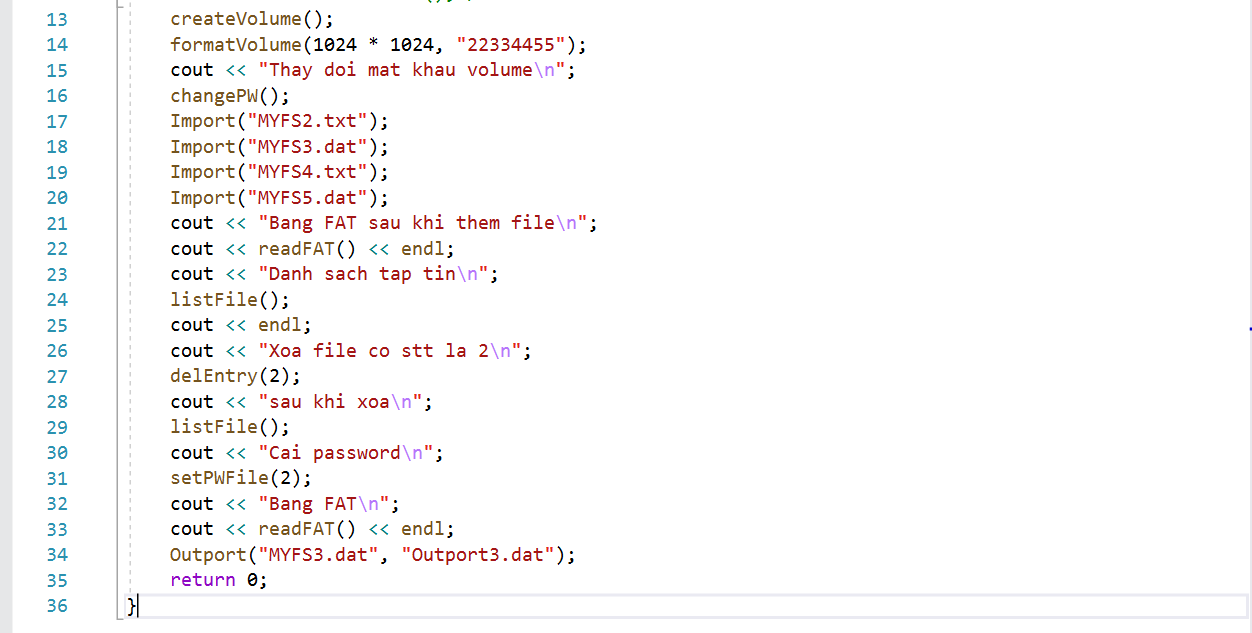
- bool Outport(string name, string name\_new): name là tên file muốn ghi vào file mới, name\_new phải trùng phần mở rộng với name

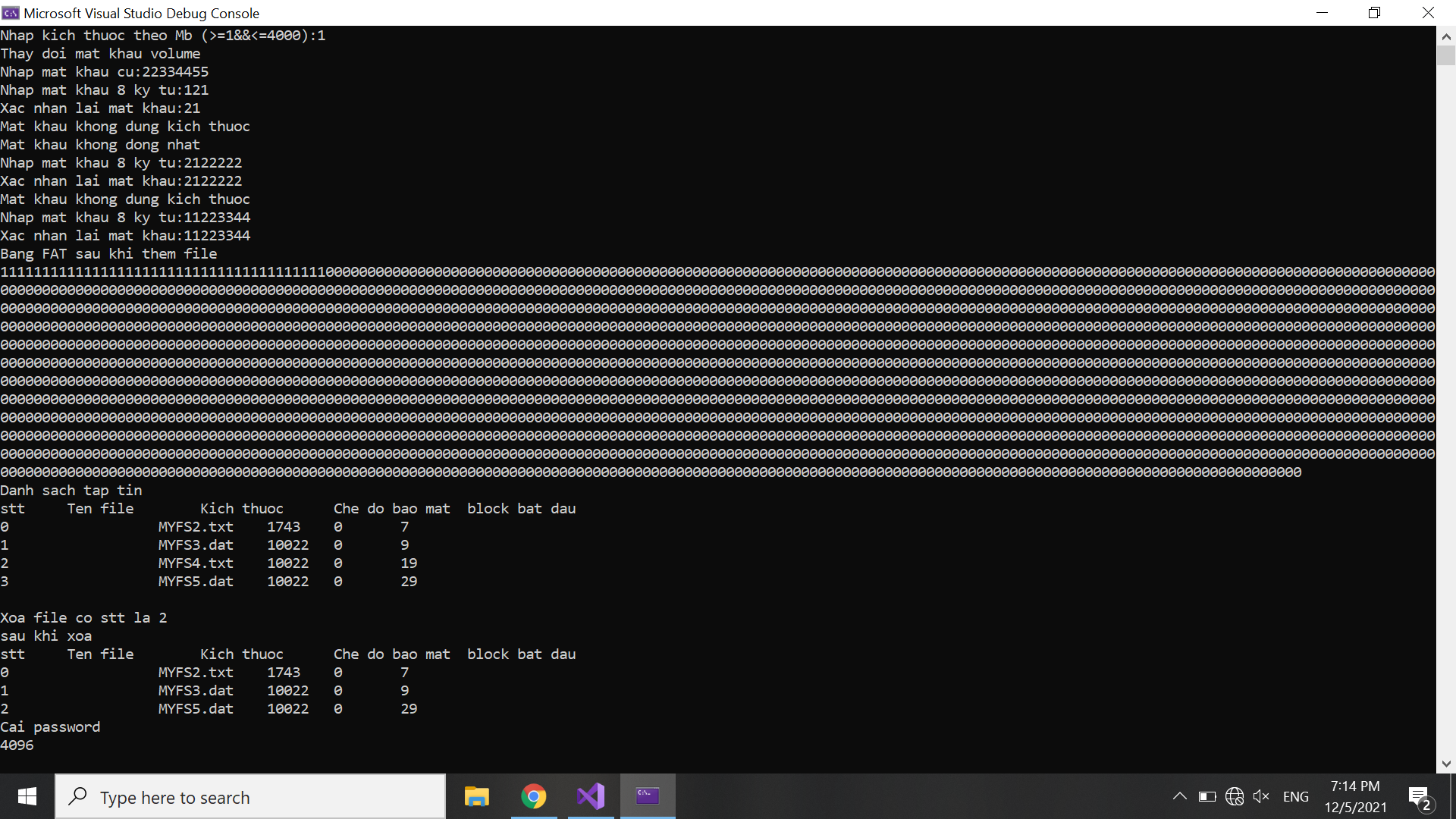
- bool delEntry(int stt):xóa tập tin tại số thứ tự

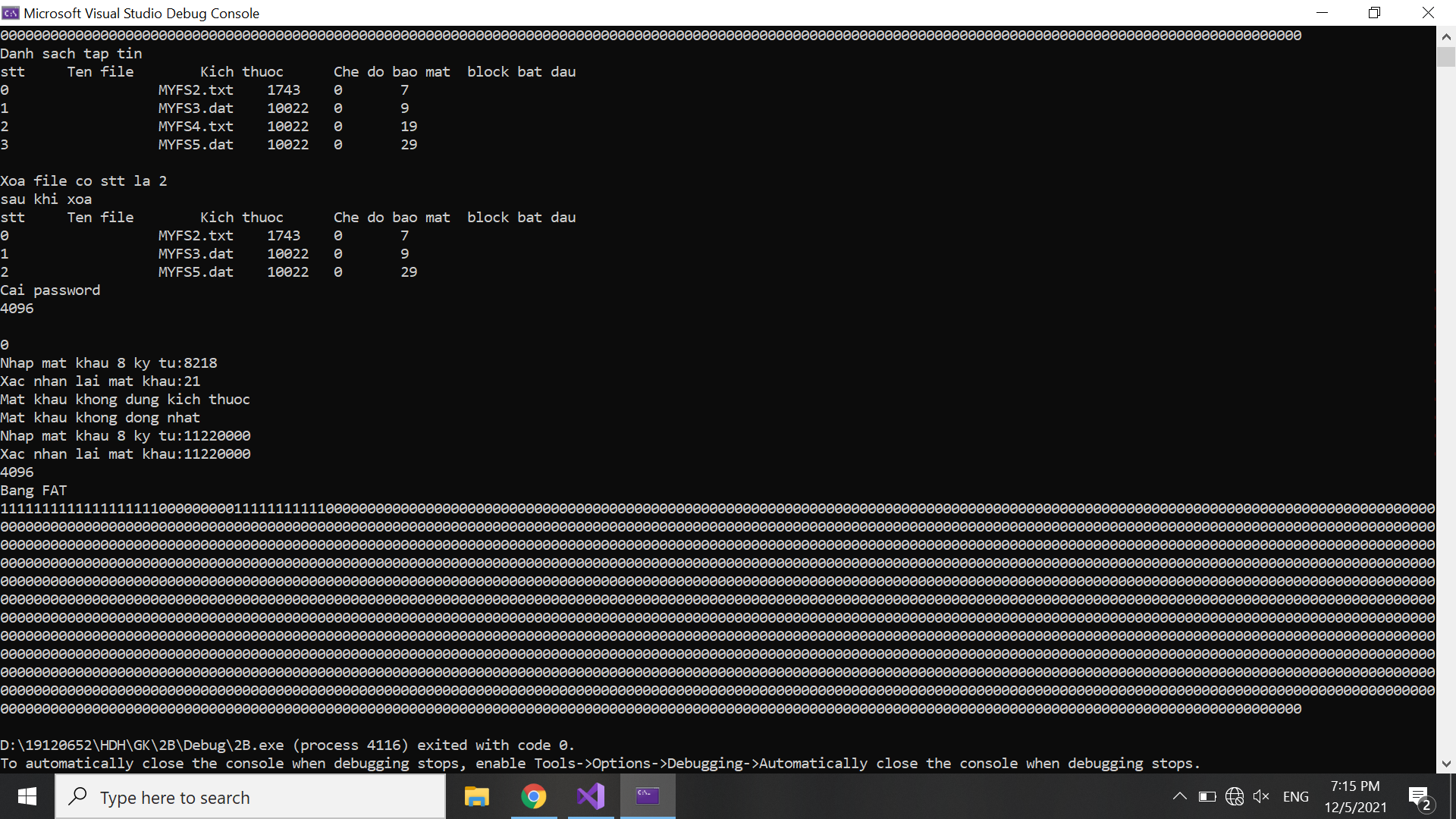
- bool setPWFile(int stt): cài đặt hoặc thay đổi mật khẩu cho tập tin tại số thứ tự

## 2.5 Chạy thử:

- Nên nhập 1MB tại vì chương trình chạy từng chức năng riêng rẻ với đọc ghi và xử lý rất nhiều nên rất lâu

****

****

****

**Tài liệu tham khảo**

1. Web SlidePlayer. Bài viết: quản lý hệ thống file

<https://slideplayer.com/slide/14644085/>

1. Kênh youtube: Học Mãi Channel. Vào danh sách phát: Tổng quan về hệ điều hành máy tính 7-16

<https://www.youtube.com/watch?v=AlfMFt57ADE&list=PLUBFHW_0UXVXBM--AP7OM28U5A89Z4bn6&index=8>

1. Web: Quantrimang. Bài viết: đọc ghi file c++

<https://quantrimang.com/doc-ghi-file-trong-c-fstream-trong-cplusplus-156307>

1. Web: Lập trình không khó. Bài viết: các hàm trong thư viện string.h

<https://nguyenvanhieu.vn/cac-ham-trong-thu-vien-string-h/>

1. Web: Hoclaptrinh.vn. Bài viết: Đọc/ghi file trong c++

<https://hoclaptrinh.vn/tutorial/hoc-c-plusplus/doc-ghi-file-trong-cplusplus-fstream-trong-cplusplus>