**ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH**

**TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN, ĐHQG-HCM**

**KHOA CÔNG NGHỆ THÔNG TIN**

---o0o---

Logo

Description automatically generated with medium confidence

**BÁO CÁO**

**PROJECT 1:**

**HỆ THỐNG QUẢN LÝ TẬP TIN TRÊN WINDOWS**

**Thành viên nhóm MSSV**

Đỗ Phan Tuấn Đạt 22127057

Phạm Thành Đạt 22127064

Đỗ Đình Hải 22127095

Lê Hồ Phi Hoàng 22127123

Trần Nguyễn Minh Hoàng 22127131

**Lớp:** 22CLC02

**Môn học:** Hệ điều hành

**Học kỳ:** 2

**Năm học:** 2023-2024

# Mục lục

[I. Mục lục 2](#_Toc161044033)

[II. Môi trường làm việc 4](#_Toc161044034)

[1. Môi trường làm việc: 4](#_Toc161044035)

[2. Ngôn ngữ: 4](#_Toc161044036)

[III. Danh sách file trong source code 4](#_Toc161044037)

[IV. Phân tích, giải thích source code 4](#_Toc161044038)

[1. fatTable.h + fatTable.cpp 4](#_Toc161044039)

[a) Constructor 5](#_Toc161044040)

[b) Operator = 5](#_Toc161044041)

[c) getClusters: 6](#_Toc161044042)

[d) listClustersOfEntry 6](#_Toc161044043)

[2. headerAttribute.h + headerAttribute.cpp 6](#_Toc161044044)

[a) Constructor 8](#_Toc161044045)

[b) printInfo 9](#_Toc161044046)

[c) getTypeID, getAttributeSize, getFlafNonResident, getNameLength, getPositionOfAttributeName, getFlag, getAttributeID 9](#_Toc161044047)

[3. ATTRIBUTE\_LIST + ATTRIBUTE\_LIST.cpp 9](#_Toc161044048)

[a) Constructor 12](#_Toc161044049)

[b) print 13](#_Toc161044050)

[c) printInfo 14](#_Toc161044051)

[d) Các hàm getHeader, getRecords, getAttributeSize 14](#_Toc161044052)

[4. DATA.h + DATA.cpp 15](#_Toc161044053)

[a) Constructor: 15](#_Toc161044054)

[b) print 17](#_Toc161044055)

[c) printInfo 17](#_Toc161044056)

[d) Các hàm getter, getHeader, getDataSize và getData 17](#_Toc161044057)

[V. Demo sử dụng chương trình 18](#_Toc161044058)

[VI. Đóng góp 18](#_Toc161044059)

[VII. Tài liệu tham khảo 18](#_Toc161044060)

# Môi trường làm việc

## Môi trường làm việc:

Visual Studio Code

## Ngôn ngữ:

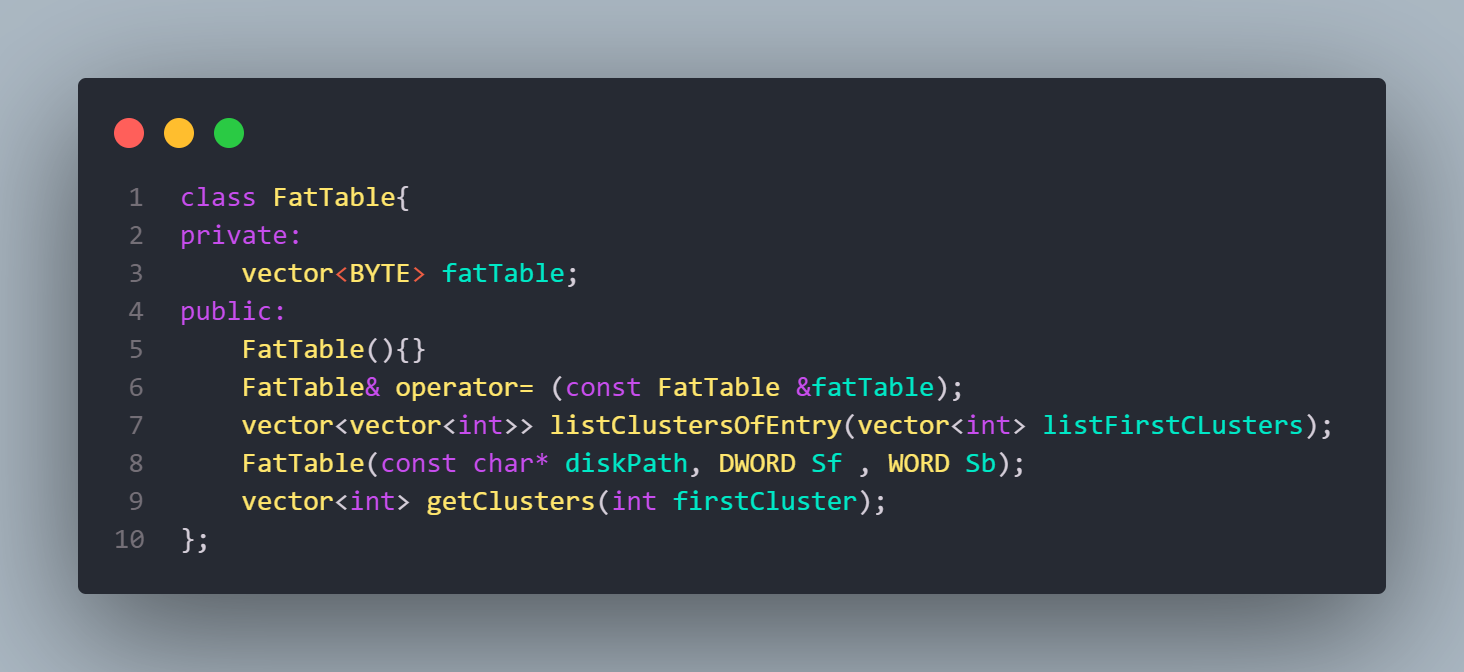
C++

# Danh sách file trong source code

* fatTable.h + fatTable.cpp
* headerAttribute.h + headerAttribute.cpp
* ATTRIBUTE\_LIST.h + ATTRIBUTE\_LIST.cpp
* DATA.h + DATA.cpp

# Phân tích, giải thích source code

## fatTable.h + fatTable.cpp



* Class FatTable có 1 thuộc tính chính là vector chứa byte nội dung của bảng FAT.
* Class FatTable có 4 chức năng:

### Constructor



* + Có 3 tham số:
    - Disk path: tên của ổ đĩa có định dang FAT32
    - Sf: số sector của bảng FAT.
    - Sb: số sector trước bảng FAT.
  + Logic:
    - Đầu tiên tạo 1 mảng có kích thước bằng số sector của bảng FAT với 512 byte 1 sector, Sf \* 512.
    - Sử dụng hàm readSector để đọc các sector này, bỏ qua số byte bằng số sector trước bảng FAT \* 512.

### Operator =

* + So sánh 2 bảng FAT của 2 object FatTable.

### getClusters:



* + Logic: Sử dụng tham số là số thứ tự của cluster đầu tiên và 1 mảng chứa các cluster hư, rỗng hoặc kết thúc. Nếu cluster hiện tại thuộc mảng chứa các byte kết thúc, trả về các cluster đã đọc được. Nếu không, thêm cluster hiện tại vào mảng cluster. Tiếp theo, di chuyển đến cluster tiếp theo và tiếp tục vòng lặp.

### listClustersOfEntry

* + Sử dụng hàm getClusters để lấy tất cả cluster thuộc entry

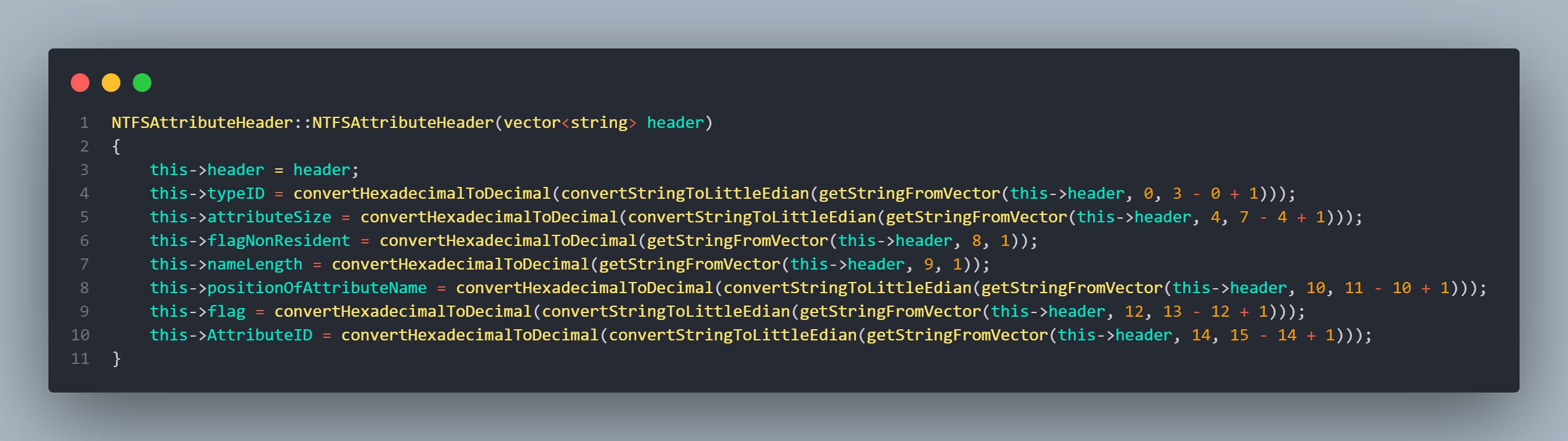
## headerAttribute.h + headerAttribute.cpp

* Chứa class NTFSAttributeHeader, tượng trung cho header của các attribute trong MFT entry



* Class bao gồm các dữ liệu:
  + header: vector lưu lại dữ liệu dạng byte của header
  + Type ID - Mã loại của attribute.
  + Attribute size - Kích thước attribute: số byte mà attribute này chiếm giữ, chiếm 4 byte 4-7.
  + Flag non-resident - Cờ báo loại tệp tin: 1 byte 0x008 báo hiệu loại nội dung là non-resident hay resident.
  + Attribute's name length - Chiều dài tên attribute: 1 byte 0x009 cho biết độ dài của tên attribute.
  + Position of Attribute's name - Vị trí tên của attribute: 2 byte 0x00A và 0X00B cho biết offset của tên chứa attribute
  + Flags - Các cờ báo: 2 byte 0x00C và 0X00D cho biết các cờ báo cảu attribute.
  + AttributeID - Mã định danh của Attribute: cho biết mã định danh độc nhất của attribute trên toàn entry.
* Class còn bao gồm các hàm:

### Constructor



* + Khởi tạo một đối tượng NTFSAttributeHeader từ dữ liệu header được cung cấp dưới dạng một vector string.
  + this->typeID: Đọc 4 byte đầu tiên từ vector header, chuyển đổi thành dạng little-endian và sau đó chuyển đổi từ hệ thập lục phân sang thập phân để lưu trữ type ID của attribute.
  + this->attributeSize: Đọc 4 byte tiếp theo từ vector header, chuyển đổi thành dạng little-endian và sau đó chuyển đổi từ hệ thập lục phân sang thập phân để lưu trữ kích thước của attribute.
  + this->flagNonResident: Đọc byte tiếp theo từ vector header và chuyển đổi từ hệ thập lục phân sang thập phân để lưu trữ giá trị cờ báo non-resident.
  + this->nameLength: Đọc byte tiếp theo từ vector header và chuyển đổi từ hệ thập lục phân sang thập phân để lưu trữ độ dài của tên attribute.
  + this->positionOfAttributeName: Đọc 2 byte tiếp theo từ vector header, chuyển đổi thành dạng little-endian và sau đó chuyển đổi từ hệ thập lục phân sang thập phân để lưu trữ vị trí bắt đầu của tên attribute trong dữ liệu.
  + this->flag: Đọc byte tiếp theo từ vector header và chuyển đổi từ hệ thập lục phân sang thập phân để lưu trữ giá trị cờ báo của attribute.
  + this->AttributeID: Đọc 2 byte cuối cùng từ vector header, chuyển đổi thành dạng little-endian và sau đó chuyển đổi từ hệ thập lục phân sang thập phân để lưu trữ ID dộc nhất của attribute.

### printInfo



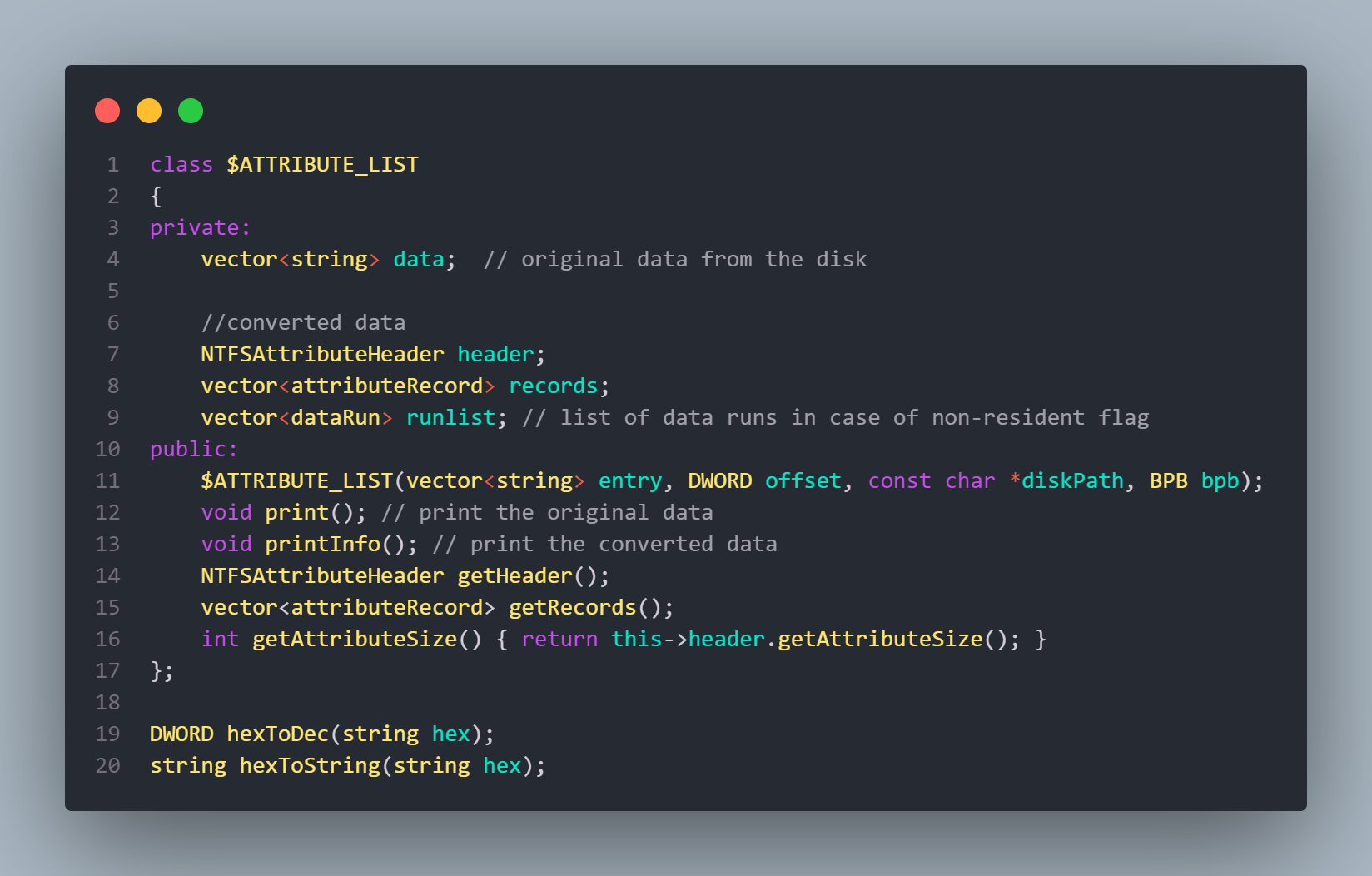
* + In ra dữ liệu đã chuyển đổi của header theo thứ tự: type ID 🡪 attribute size 🡪 non-resident flag 🡪 độ dài tên 🡪 offset đến tên 🡪 cờ báo 🡪 attribute ID

### getTypeID, getAttributeSize, getFlafNonResident, getNameLength, getPositionOfAttributeName, getFlag, getAttributeID

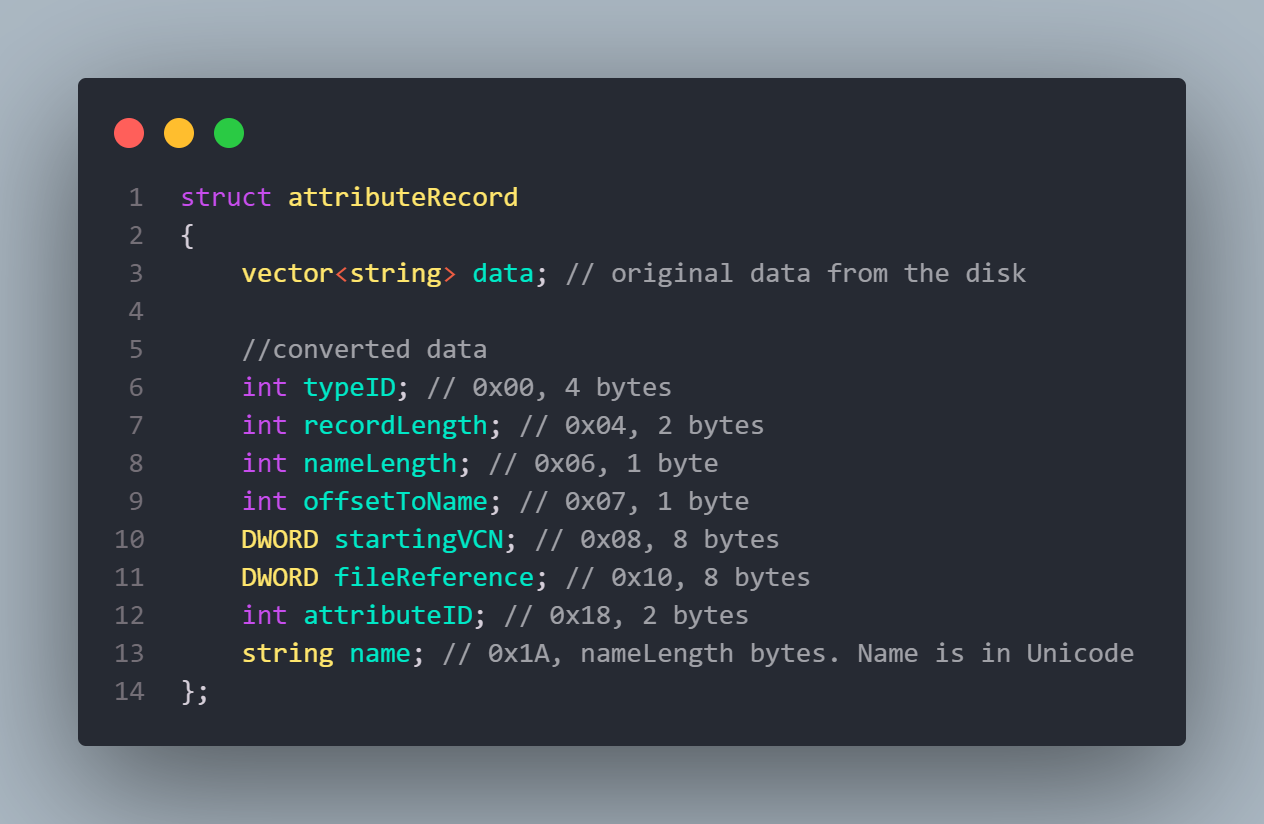
* + Trả về các giá trị tương ứng trong header

## ATTRIBUTE\_LIST + ATTRIBUTE\_LIST.cpp

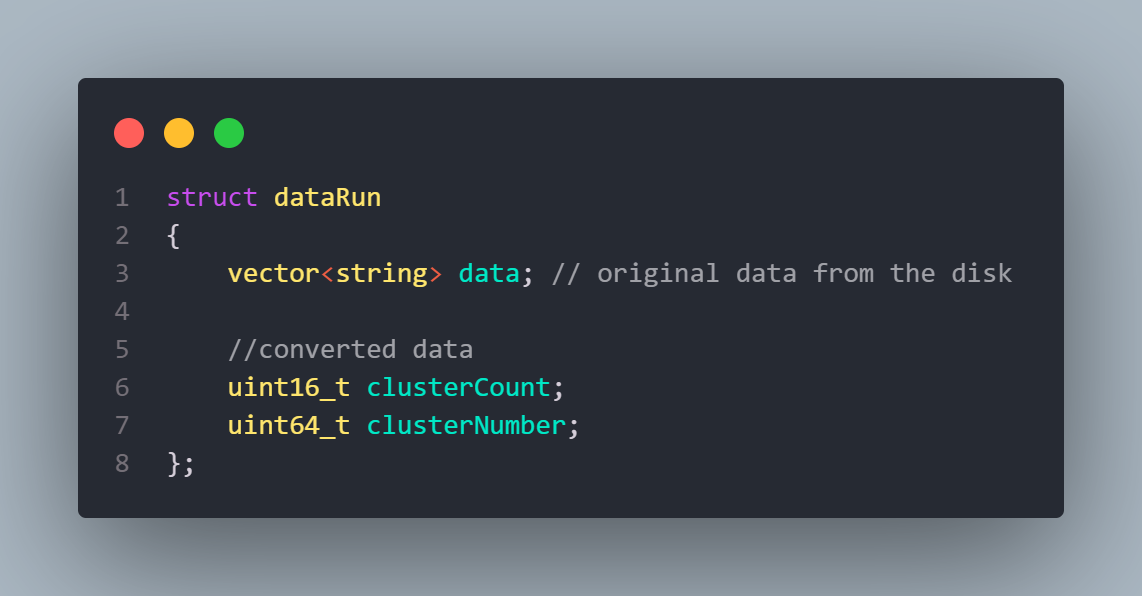
* Chứa class $ATTRIBUTE\_LIST tượng trưng cho attribute $ATTRIBUTE\_LIST trong một entry



* Dữ liệu trong class bao gồm:
  + Dữ liệu gốc dạng byte từ đĩa
  + Header của NTFS attribute
  + Danh sách các thuộc tính theo format:

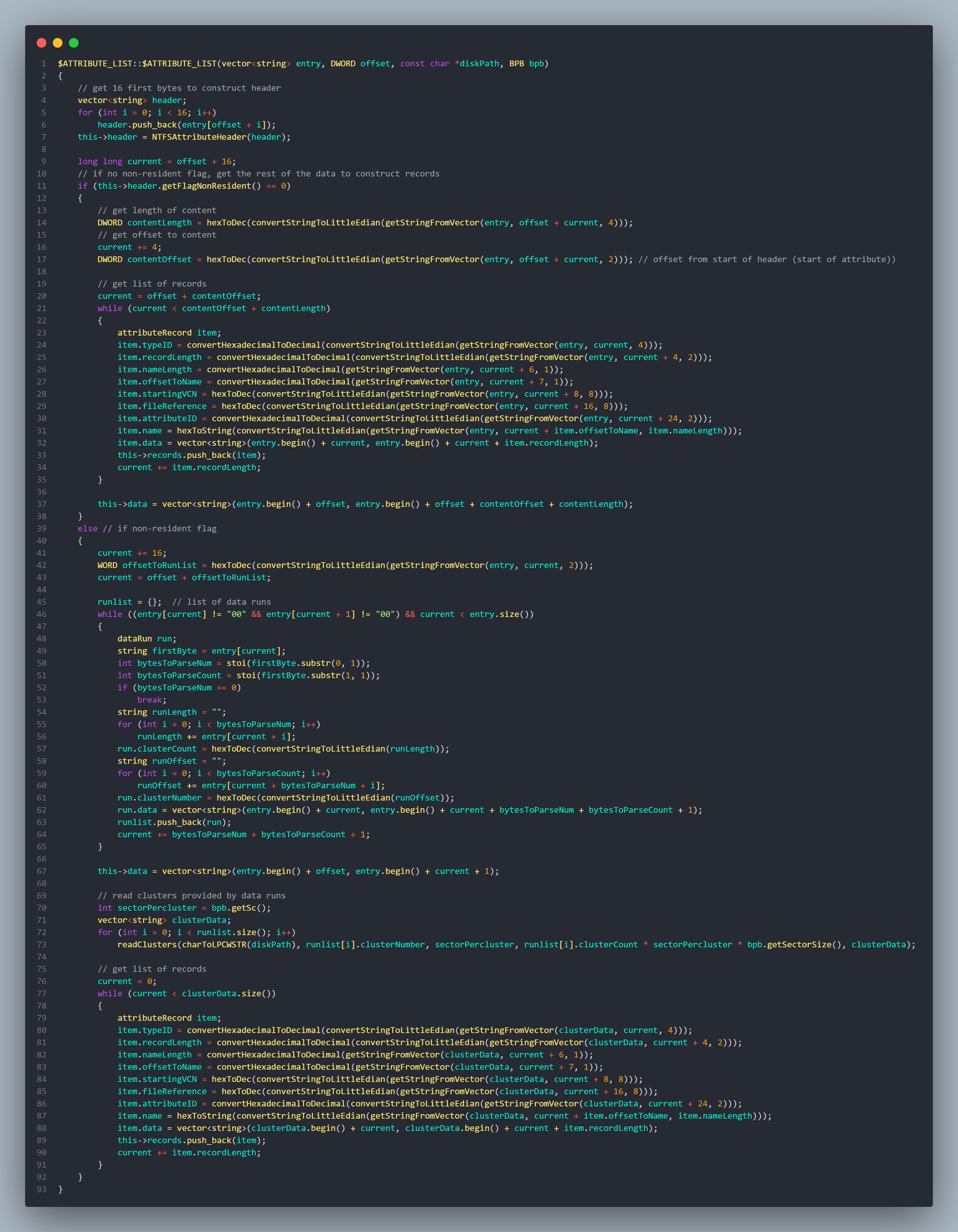


* + Danh sách các data run trong trường hợp có non-resident flag, mỗi data run có format:



* Trong class còn có các function:

### Constructor



* + Dòng 4-7: Khởi tạo một vector header để lưu trữ 16 byte đầu tiên từ dữ liệu đọc được, xây dựng tiêu đề NTFSAttributeHeader.
  + Dòng 11-34: Nếu flagNonResident trong header là 0 (attribute thuộc kiểu resident), chỉ đọc phần còn lại của dữ liệu để xây dựng danh sách các attribute (attributeRecord)
    - Dòng 14-17: Đọc độ dài của nội dung attribute và vị trí của nội dung trong dữ liệu đọc được.
    - Dòng 20-34: Duyệt qua dữ liệu từ vị trí bắt đầu đến kết thúc nội dung thuộc tính, xây dựng các bản ghi attribute và lưu trữ chúng trong vector records.
  + Dòng 39-91: Nếu flagNonResident là 1 (attribute thuộc kiểu non-resident), xây dựng danh sách các data run:
    - Dòng 41-43: Đọc vị trí của danh sách các data run.
    - Dòng 46-65: Duyệt qua danh sách các run dữ liệu và xây dựng các đối tượng dataRun, lưu trữ chúng trong vector runlist bằng vòng lặp:
      * Đọc byte đầu tiên, chia ra thành 2 phần, nửa đầu của byte là số byte cần đọc để lấy chỉ số của cluster cần đến, nửa sau là số byte để lấy số lượng cluster cần đọc
      * Xử lý các byte vừa đọc và lưu vào giá trị tương ứng
      * Điều chỉnh offset để đọc data run tiếp theo, nếu chưa thấy 2 byte liên tiếp có giá trị “00” và chưa hết entry thì quay lại đầu vòng lặp
    - Dòng 70-73: Đọc dữ liệu từ các cluster được cung cấp bởi các data run.
    - Dòng 76-91: Duyệt qua dữ liệu đã đọc từ các cluster và xây dựng các bản ghi attribute, lưu trữ chúng trong vector records.

### print

* + In ra dữ liệu của attribute (dạng byte)

### printInfo



* + In ra dữ liệu đã chuyển đổi của attribute $ATTRIBUTE\_LIST, bao gồm vòng lặp để in ra danh sách các attribute, và danh sách các data run nếu attribute thuộc loại non-resident

### Các hàm getHeader, getRecords, getAttributeSize

* + Để trả về header, danh sách các attribute, và size của attribute $ATTRIBUTE\_LIST
* Trong file còn có các hàm hexToDec và hexToString để hỗ trợ việc xử lý dữ liệu

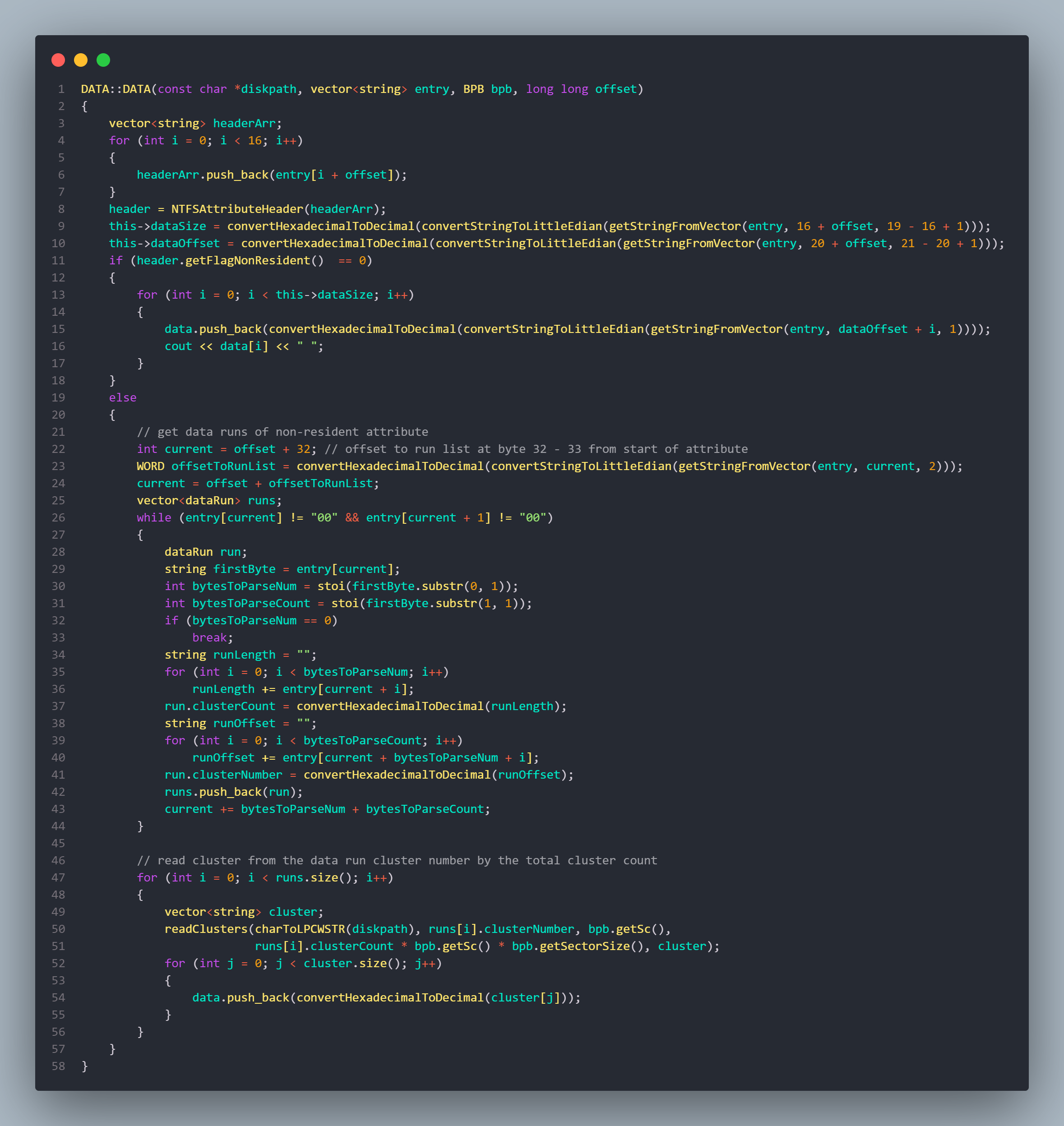
## DATA.h + DATA.cpp



* Class Data (dùng để biểu diễn attribute $DATA) có 4 thuộc tính chính:
  + Data size: một số nguyên lưu trữ số byte là dữ liệu của tập tin được attribute $DATA lưu trữ. Data size thường nằm ở byte số 16 đến byte số 19 tính từ đầu attribute $DATA.
  + Data offset: là một số nguyên chứa vị trí bắt đầu của phần nội dung. Nếu header của attribute $DATA có cờ resident, data offset sẽ cho biết thứ tự byte bắt đầu của phần nội dung ngay sau attribute $DATA. Ngược lại, nếu có cờ non-resident, data offset sẽ cho biết thứ tự byte bắt đầu của phần nội dung trong runlist. Nếu có là resident, data offset sẽ là 2 byte 20 và 21, còn nếu là non-resident, data offset sẽ là 2 byte 32 và 33.
  + Header: header của attribute $DATA chiếm 16 byte.
  + Data: Một vector byte chứa nội dung đọc được.
* Class Data có 6 chức năng:

### Constructor:

* Tạo 1 attribute $DATA và đọc nội dung của attribute đó



* + 4 tham số được truyền vào là:
    - Diskpath: một chuỗi chứa tên ổ đĩa chứ tệp tin được đọc.
    - Entry: một vector chuỗi chứa nội dung của toàn entry chứa attribute $DATA được đọc.
    - BPB: BIOS PARAMETER CLOCK, một object thuộc class BPB, chứa thông tin mô tả về tổ chức của đĩa logic và hệ thống quản lý tập tin. BPB chứa Sc là số sector của 1 cluster, sẽ dùng trong việc tính toán.
    - Offset: offset từ đầu entry đến vị trí của attribute $DATA.
  + Logic:
    - Đầu tiên, lấy header của attribute $DATA bằng việc lấy 16 byte đầu tiên tính từ offset.
    - Tiếp theo, lấy data size tại byte 16, 17, 18, 19 và data offset tại byte 20, 21.
    - 2 trường hợp loại dữ liệu có thể xảy ra:
      * Nếu dữ liệu thuộc loại resident: Sử dụng 1 vòng lặp có độ dài bằng data size, thu thập số byte bắt đầu từ data offset tính từ đầu attribute $DATA.
      * Nếu dữ liệu thuộc loại non-resident:
        + Cần lấy được runlist. Di con trỏ đến byte 32 và 33 để lấy offset đến runlist.
        + Sau đó di con trỏ đến vị trí bắt đầu runlist
        + Sử dụng vòng lặp while đến khi không có 2 byte liên tiếp nào bằng 0, có nghĩa là runlist vẫn còn dữ liệu
        + Sử dụng cấu trúc dataRun, có thuộc tính data là dữ liệu được lấy, clusterCount là số cluster thuộc runlist và clusterNumber là số thứ tự của cluster.
        + Lấy byte đầu tiên của runlist, chia làm 2 phần

Phần đầu là số byte chứa số thứ tự của cluster.

Phần sau là số byte chứa số cluster cần lấy.

* + - * + Với số byte chứa số thứ tự của cluster và số byte chứa số cluster cần lấy sẽ được số thứ tự và số cluster cần lấy.
        + Sử dụng hàm readCluster để đọc nội dung cluster vào 1 vector string.
        + Cuối cùng, thu thập số byte từ vector cluster đó bằng đúng data size.

### print

* + In header và vector data dưới dạng BYTE.

### printInfo

* + In header và vector data dưới dạng ký tự ASCII.

### Các hàm getter, getHeader, getDataSize và getData

* + Lấy thuộc tính header, size của attribute và data từ class DATA

# Demo sử dụng chương trình

# Đóng góp

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Thành viên | MSSV | Công việc | Tỉ lệ (%) hoàn thành | Tổng đóng góp cho nhóm |
| Trần Nguyễn Minh Hoàng | 22127131 |  | 100% |  |
| Lê Hồ Phi Hoàng | 22127123 |  | 100% |  |
| Đỗ Đình Hải | 22127095 |  | 100% |  |
| Phạm Thành Đạt | 22127064 |  | 100% |  |
| Đỗ Phan Tuấn Đạt | 22127057 |  | 100% |  |

# Tài liệu tham khảo

* Source code Read Sector C++ cung cấp bởi thầy Lê Viết Long
* <http://inform.pucp.edu.pe/~inf232/Ntfs/ntfs_doc_v0.5/attributes/attribute_list.html>
* <https://www.kcall.co.uk/ntfs/index.html?fbclid=IwAR0_ve_jzXeksnhJD3_q7JAAr6PuazBDYy277tIfBtB-LX53UwiMADvLcWU>
* <https://www.youtube.com/watch?v=6WFUM5eViIk>