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

**ĐẠI HỌC KHOA HỌC TỰ NHIÊN**

**KHOA CÔNG NGHỆ THÔNG TIN**

**------------------------------------**



**ĐẶNG PHƯƠNG NAM**

**1612406 | 16CTT3**

**tP.HCM, 11/12/2017**

BÁO CÁO KỸ THUẬT P2 MÔN: CTDL & GT

**GVHD: Ths. nguyễn tri tuấn**

**MỤC LỤC**

[PHỤ LỤC HÌNH ẢNH 2](#_Toc501054209)

[LƯU Ý 2](#_Toc501054210)

[I. YÊU CẦU ĐỀ[1] 3](#_Toc501054211)

[1. Nội dung 3](#_Toc501054212)

[2. Chương trình 3](#_Toc501054213)

[II. KỸ THUẬT 4](#_Toc501054214)

[1. Cấu trúc file nén 4](#_Toc501054215)

[2. Các cấu trúc dữ liệu quan trọng[2] 6](#_Toc501054216)

[a. Thuật toán Huffman 6](#_Toc501054217)

[b. Dữ liệu nén 7](#_Toc501054218)

[3. Sơ đồ lớp 8](#_Toc501054219)

[a. Class PRIORITY\_QUEUE 8](#_Toc501054220)

[b. Class H\_NODE 8](#_Toc501054221)

[c. Class HUFFMAN\_TREE 9](#_Toc501054222)

[d. Class STATIC\_HUFFMAN 10](#_Toc501054223)

[e. Class PROGRAM 10](#_Toc501054224)

[4. Các hàm chính 11](#_Toc501054225)

[a. Hàm tự code[3] 11](#_Toc501054226)

[b. Hàm hỗ trợ[5] 15](#_Toc501054227)

[III. SẢN PHẨM 16](#_Toc501054228)

[IV. TÀI LIỆU THAM KHẢO 18](#_Toc501054229)

# PHỤ LỤC HÌNH ẢNH

[Hình 1. Sơ đồ lớp 8](#_Toc501054240)

[Hình 2. Bật bit i của một số nguyên 15](#_Toc501054241)

[Hình 3. Lấy bit i của một số nguyên 15](#_Toc501054242)

[Hình 4. Menu chính 16](#_Toc501054243)

[Hình 5. Menu 1 16](#_Toc501054244)

[Hình 6. Menu 2 17](#_Toc501054245)

[Hình 7. Menu 3 17](#_Toc501054246)

[Hình 8. Menu 4 17](#_Toc501054247)

# LƯU Ý

Toàn bộ Source code được viết bằng Visual Studio Community 2015

# YÊU CẦU ĐỀ[[1](#Thamkhao1)]

## Nội dung

Cho phép nén tất cả các file trong một folder (cho trước) thành một tập tin nén có cấu trúc xác định; không yêu cầu nén (các) folder con; cho phép chọn lựa 1 (hay nhiều) file để giải nén; có kiểm tra checksum kích thước file khi giải nén; báo cáo kỹ thuật (dạng Word/PDF).

## Chương trình

1. Nén các file trong folder
2. Xem nội dung file đã nén
3. Giải nén tất cả các file trong tập tin nén
4. Chọn và giải nén các file riêng lẻ

|  |  |
| --- | --- |
| **Menu** | **Ý nghĩa** |
| 1 | * Yêu cầu user nhập vào folder chứa các file cần nén (folder\_in) và nhập đường dẫn+tên file nén (file\_out). * Chương trình sẽ nén (encode) tất cả các file trong folder\_in và lưu kết quả nén vào file\_out. |
| 2 | * Chương trình hiển thị danh sách tập tin chứa trong file nén file\_out, theo mẫu sau:   STT Tên file Size trước nén Size sau nén  1. xxxxxxxxxxxx xxxxx xxxx  2. xxxxxxxxxxxx xxxxx xxxx  … |
| 3 | * Yêu cầu user nhập vào folder chứa các file giải nén (folder\_out). * Chương trình giải nén (decode) tất cả các file trong tập tin file\_out và lưu vào thư mục folder\_out. * Khi giải nén, kiểm tra checksum bằng cách so sánh kích thước file trước nén và sau khi giải nén, nếu sai  báo lỗi. |
| 4 | * Yêu cầu user nhập vào folder chứa các file giải nén (folder\_out) và thứ tự các file muốn giải nén. VD: 1,3,6 * Giải nén (decode) các file 1,3,6 và lưu vào thư mục folder\_out. * Khi giải nén, kiểm tra checksum bằng cách so sánh kích thước file trước nén và sau khi giải nén, nếu sai → báo lỗi. |

# KỸ THUẬT

## Cấu trúc file nén

|  |  |
| --- | --- |
| HEADER  FILE | Mã nhận dạng file //dpnam |
| Số lượng file nén |
| Dữ liệu tạo các cây huffman  *Số lượng cây Số lượng ký tự có tần số khác 0 (cây 1, cây 2,…)*  *Bảng tần số của cây 1*  *Bảng tần số của cây 2*  *…….* |
| **STT Tên file Size trước nén Size sau nén Số lượng bits cuối**  **có nghĩa**  int | string | unsigned long long | unsigned long long | unsigned short  1 xxxxxxxx xxxxx xxxxx x  2 xxxxxxx xxxxx xxxxx x  ……..  Vd:  1|P2.pdf|12345|12222|6  2|main.cpp|4245|4300|8  ……... |
| DATA  NÉN | Dữ liệu nén theo thứ tự  Data 1  Data 2  …… |

***File minh họa:***

|  |  |
| --- | --- |
| HEADER  FILE |  |
|  |
|  |
|  |
| DATA  NÉN |  |

## Các cấu trúc dữ liệu quan trọng[[2](#Thamkhao2)]

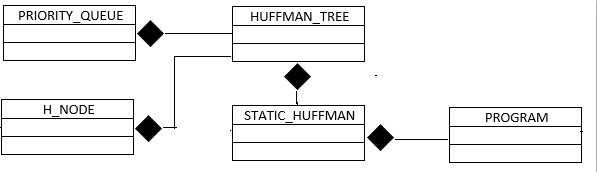
### Thuật toán Huffman

|  |  |
| --- | --- |
| Hàng đợi ưu tiên | template <class T> class PRIORITY\_QUEUE {  private:  T \*items;  int rear;  int maxsize;  void heapify(int);  public:  PRIORITY\_QUEUE();  PRIORITY\_QUEUE(int);  PRIORITY\_QUEUE(const PRIORITY\_QUEUE&);  ~PRIORITY\_QUEUE();  bool isEmpty();  bool enQueue(T);  bool deQueue(T&);  bool minValue(T&);  }; |
| Cây Huffman | class HUFFMAN\_TREE {  private:  H\_NODE HuffTree[MAX\_NODES];  short int root;  private:  void traverseTreeEncode(CODE\_TABLE\_ITEM\*, short int, char\*, int  public:  HUFFMAN\_TREE();  ~HUFFMAN\_TREE();  void buildTree();  bool countChar(string, unsigned int\*, unsigned long long&);  void createBitcode(CODE\_TABLE\_ITEM\*);  void loadFreq(unsigned int\*);  void traverseTreeDecode(string, string}; |
| Struct lưu trữ chuỗi char bits ứng cho mỗi loại ký tự | struct CODE\_TABLE\_ITEM {  char \*bits; // Mã bit của ký tự  unsigned char nBitLen; // số bit thật sự sử dụng  }; |

### Dữ liệu nén

|  |  |
| --- | --- |
| Struct lưu trữ dữ liệu nén | struct ENCODE {  string idFile;//mã nhận dạng file  string nameInput;//tên file ban đầu  unsigned int freq[256];  unsigned long long sizeInput;//kích thước file ban đầu  unsigned long long sizeEncode;//kích thước file nén  unsigned short bitsFinal;//số bit có nghĩa trong byte cuối  string data;//dữ liệu nén  }; |

## Sơ đồ lớp



Hình 1. Sơ đồ lớp

### Class PRIORITY\_QUEUE

|  |
| --- |
| template <class T> class PRIORITY\_QUEUE {  private:  T \*items;//mảng chứa các giá trị  int rear;//số lượng phần tự hiện có  int maxsize;//số lượng phần tử tối đa  void heapify(int);//hiệu chỉnh lại vị trí theo điều kiện min heap  public:  PRIORITY\_QUEUE();//constructor default  PRIORITY\_QUEUE(int); //constructor default  PRIORITY\_QUEUE(const PRIORITY\_QUEUE&);//constructor copy  ~PRIORITY\_QUEUE();//destructor  bool isEmpty();//kiểm tra rỗng  bool enQueue(T);//thêm 1 phần tử  bool deQueue(T&);//xóa phần tử min  bool minValue(T&);//lấy giá trị phần tử min  }; |

### Class H\_NODE

|  |
| --- |
| class H\_NODE {  private:  unsigned char c;//ký tự c  unsigned int freq;//tần số xuất hiện  short int left;//nhánh trái  short int right;//nhánh phải  unsigned short pos;//vị trí trong bảng các phần tử của cây huffman  public:  H\_NODE();//constructor  ~H\_NODE();//destructor  //Getter  const unsigned char& getChar()const;  const unsigned int& getFreq() const;  const short int& getLeft() const;  const short int& getRight() const;  const unsigned short& getPos() const;  //Setter  void setChar(unsigned char);  void setPos(unsigned short);  //tạo 1 node trong cây huffman: ký tự, tần số, con trái, con phải  void initNode(unsigned char, unsigned int, short int, short int);  void increaseFreq(unsigned int k = 1);//tăng tần số xuất hiện  //các operator so sánh  bool operator== (const H\_NODE&);  bool operator!= (const H\_NODE&);  bool operator> (const H\_NODE&);  bool operator< (const H\_NODE&);  bool operator>= (const H\_NODE&);  bool operator<= (const H\_NODE&);  }; |

### Class HUFFMAN\_TREE

|  |
| --- |
| class HUFFMAN\_TREE {  private:  H\_NODE HuffTree[MAX\_NODES];//511 node  short int root;  private:  //duyệt cây để tạo bảng bit code phục vụ cho việc nén  void traverseTreeEncode(CODE\_TABLE\_ITEM\*, short int, char\*, int);  public:  HUFFMAN\_TREE();//constructor  ~HUFFMAN\_TREE();//destructor  void buildTree();//xây dựng cây huffman  //một số hàm chỉ dùng cho việc nén  bool countChar(string, unsigned int\*, unsigned long long&);//thống kê tần số xuất hiện của mỗi ký tự từ file input  void createBitcode(CODE\_TABLE\_ITEM\*);//Tạo ra bảng bit code  //một số hàm chỉ dùng cho việc giải nén  void loadFreq(unsigned int\*);  void traverseTreeDecode(string, string&);//duyệt cây để giải nén  }; |

### Class STATIC\_HUFFMAN

|  |
| --- |
| class STATIC\_HUFFMAN {  private:  HUFFMAN\_TREE HTree\_Encode;//cây phục vụ cho việc nén  HUFFMAN\_TREE HTree\_DeCode;//cây phục vụ cho việc giải nén  //hàm chuyển đổi dữ liệu vào thành chuỗi bits: linkInput, bảng bits, số lượng bytes cuối có nghĩa  string enCodeToStringBits(string, CODE\_TABLE\_ITEM \*, unsigned short&);  public:  STATIC\_HUFFMAN();//constructor default  ~STATIC\_HUFFMAN();//destructor  bool encode(string, ENCODE&);//hàm nén: linkInput, struct dữ liệu nén  bool decode(ENCODE, ofstream&);//struct dữ liệu nén, file ra  }; |

### Class PROGRAM

|  |
| --- |
| class PROGRAM {  private:  ENCODE \*EN;//mảng chứa các struct dữ liệu nén  unsigned int numberFile = 0;//số lượng dữ liệu nén  STATIC\_HUFFMAN SH;//thuật toán nén huffman tĩnh  void menuMain();//menu chính  bool enCode\_FullFiles();//nén tất cả  bool printInfor\_FilesDeCode();//in thông tin dữ liệu nén  bool deCode\_FullFiles();//giải nén tất cả các file  bool deCode\_Files();//giải nén một số file theo yêu cầu  bool loadDataEnCode(string);//load data nén từ file nén  public:  PROGRAM();//constructor default  ~PROGRAM();//destructor  void run();//chương trình chạy chính  }; |

## Các hàm chính

### Hàm tự code[[3](#Thamkhao3)]

|  |
| --- |
| #define MAX\_TREE\_NODES 511 // số nút max trong cây  #define MAX\_CODETABLE\_ITEMS 256// số phần tử max trong bảng mã bit  #define MAX\_BIT\_LEN 256// c.dài max của mã bit 256 bits # 32 bytes |

|  |
| --- |
| //Hàm bool countChar(..)  //Ý nghĩa: thống kê số phần tử xuất hiện trong file  //Input: tên file cần thống kê, bảng tần số gồm 256 ký tự, kích thước //file  //Output: bảng tần số xuất hiện của 256 ký tự, kích thước file  bool HUFFMAN\_TREE::countChar(string filename, unsigned int \*freq, unsigned long long &sizeInput) {  //Mở file  //Lấy từng ký tự trong file để thống kê tần số xuất hiện vào bảng gồm các node huffman  //Xác định kích thước file ban đầu  //Đóng file  } |

|  |
| --- |
| //Hàm void buildTree()  //Ý nghĩa: Xây dựng cây huffman  //Input: Bảng tần số xuất hiện của 256 ký tự  //Output: Cây huffman hoàn chỉnh được lưu trong mảng tối đa 511 node  void HUFFMAN\_TREE::buildTree() {  //Khởi tạo hang đợi ưu tiên gồm tối đa 256 huffman node  //Lặp i từ 0 đến 255  //Nếu tần số của node thứ I khác 0 thì thêm vào hàng đợi ưu tiên  //Hết lặp  //Lặp i = 0  //Lấy trong hàng đợi ưu tiên ra node min thứ nhất  //Nếu trong hàng đợi chỉ còn đúng 1 node min thì đó là node root, thoát lặp  //Ngược lại thì lấy node min đó ra làm node min thứ hai  //Hai node min được lấy ra sẽ tạo thành 1 node min mới và được thêm vào hàng đợi ưu tiên  //Tăng i 1 đơn vị  //Hết lặp  } |

|  |
| --- |
| //Hàm void traveseTreeEncode(…)  //Ý nghĩa: duyệt cây để phụ vụ cho việc nén  //Input: Bảng bits bằng chuỗi char\* cho ký tự có tần số xuất hiện khác 0, vị trí node cần duyệt, chuỗi bits tạm cho 1 ký tự, số lượng bits của chuỗi bits tạm  //Output: Bảng bits bằng chuỗi char\* ứng với cập nhật cho một ký tự  void HUFFMAN\_TREE::traverseTreeEncode(CODE\_TABLE\_ITEM \*HuffCodeTable, short int currNode, char \*bitCode, int bitCodeLen) {  //Nếu vị trí node là -1 thì kết thúc  //Nếu lá node lá thì chuyển toàn bộ chuỗi bits tạm sang chuỗi bits chính trong bảng mã bit  //Thêm bit 0 vào cuối chuỗi bits tạm, gọi đệ quy cho cây duyệt sang trái  //Thêm bit 1 vào cuối chuỗi bits tạm, gọi đệ quy cho cây duyệt sang phải  } |

|  |
| --- |
| //Hàm void traveseTreeDecode (…)  //Ý nghĩa: duyệt cây để phụ vụ cho việc giải nén  //Input: chuỗi bits 0100…., chuỗi lưu data giải nén  //Output: chuỗi đã giải nén  void HUFFMAN\_TREE::traverseTreeDecode(string BitsCode, string &dataDecode) {  //Lấy vị trí duyệt cây tại root  //Lặp từ 0 đến size của chuỗi BitsCode  //Nếu là node lá thì tiến hành thêm ký tự của node vào chuỗi data giải nén  //Nếu BitsCode[i] == ‘0’ thì duyệt sang trái  //Nếu BitsCode[i] == ‘1’ thì duyệt sang phải  //Hết lặp  } |

|  |
| --- |
| //Hàm void createBitcode (…)  //Ý nghĩa: tạo bảng bit code cho những ký tự có tần số khác 0  //Input: bảng bit code rỗng  //Output: bảng bit code đã hình thành  void HUFFMAN\_TREE::createBitcode(CODE\_TABLE\_ITEM \*HuffCodeTable) {  //Khởi tạo bảng bits ứng với 256 ký tự có chiều dài 0  //Khởi tạo chuỗi bitCode tạm gồm 256 bits tối đa, với chiều dài thực ban đầu là 0  //Duyệt cây để tạo bảng bits tương ứng  } |

|  |
| --- |
| //Hàm void enCodeToStringBits (…)  //Ý nghĩa: Chuyển chuỗi data vào thành chuỗi bit chứa data nén  //Input: data vào, bảng bits code, số lượng bit cuối có nghĩa  //Output: chuỗi bits lưu trữ data nén  string STATIC\_HUFFMAN::enCodeToStringBits(string input, CODE\_TABLE\_ITEM \*HuffCodeTable, unsigned short &bitsFinal) {  //Mở file  //Đọc từng ký tự trong file chuyển thành chuỗi char bits ứng vơi nó nhờ vào bảng HuffCodeTable  //Đóng file  //Xác định số phần tử cuối cùng có nghĩa  //Trả về chuỗi bits  } |

|  |
| --- |
| //Hàm void encode (…)  //Ý nghĩa: nén data của file ban đầu  //Input: đường dẫn file ban đầu, struct lưu lại dữ liệu nén  //Output: struct lưu lại dữ liệu nén  bool STATIC\_HUFFMAN::encode(string linkInput, ENCODE &EN)  {  //Bước 1: Duyệt file, lập bảng thống kê tần số xuất hiện của mỗi loại ký tự  //Bước 2: Phát sinh cây Huffman dựa vào bảng thống kê  //Bước 3: Từ cây Huffman phát sinh bảng mã bit cho từng loại ký tự  //Bước 4: Duyệt file, thay thế các ký tự bằng mã bit tương ứng  //Bước 5: Lưu lại bảng tần số xuất hiện các ký tự  } |

|  |
| --- |
| //Hàm void decode (…)  //Ý nghĩa: giải nén data  //Input: struct lưu dữ liệu nén, file lưu dữ liệu giải nén  //Output: file lưu dữ liệu giải nén  bool STATIC\_HUFFMAN::decode(ENCODE EN, ofstream &f)  {  //Kiểm tra idFile có hợp lệ không  //Load bảng tần số lên cây Huffman dành cho việc giải nén  //Xây dựng cấy Huffman  //Tiến hành chuyển data nén thành chuỗi bits bằng thao tác bật tất cả các bít trong chuỗi data nén  //Loại bỏ những bits không cần thiết được thêm vào khi thực hiện nén  //Duyệt cây Huffman để chuyển chuỗi bits thành chuỗi ký tự giải nén  //Check sum xem kích thước chuỗi nén có bằng chuỗi ban đầu hay không  //Nếu bằng thì ghi toàn bộ chuỗi giải nén được ra file  } |

**Tham khảo [**[4](#Thamkhao4)**]**

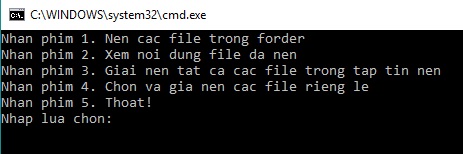
|  |
| --- |
| Hình 2. Bật bit i của một số nguyên |

|  |
| --- |
| Hình 3. Lấy bit i của một số nguyên |

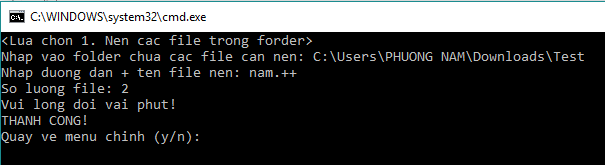
### Hàm hỗ trợ[[5](#Thamkhao5)]

|  |  |
| --- | --- |
| HANDLE WINAPI FindFirstFile(  \_In\_  LPCTSTR    lpFileName,  \_Out\_ LPWIN32\_FIND\_DATA lpFindFileData  ); | //Dùng để xác định file đầu tiên trong folder |
| BOOL WINAPI FindNextFile(  \_In\_  HANDLE            hFindFile,  \_Out\_ LPWIN32\_FIND\_DATA lpFindFileData  ); | //Dùng để xác định file kế tiếp trong folder bắt đầu từ file trước đó |
| DWORD WINAPI GetCurrentDirectory(  \_In\_  DWORD  nBufferLength,  \_Out\_ LPTSTR lpBuffer  ); | //Lấy địa chỉ ban của folder |
| BOOL WINAPI SetCurrentDirectory(  \_In\_ LPCTSTR lpPathName  ); | //Cố định địa chỉ thao tác tại folder |
| BOOL WINAPI CreateDirectory(  \_In\_     LPCTSTR         lpPathName,  \_In\_opt\_ LPSECURITY\_ATTRIBUTES lpSecurityAttributes  ); | //Khởi tạo một folder mới |

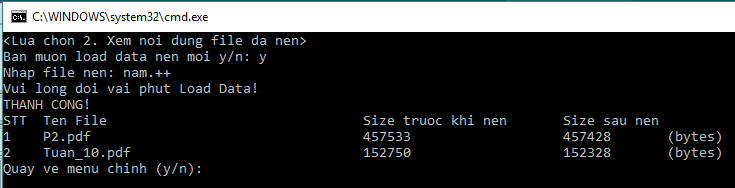
# SẢN PHẨM



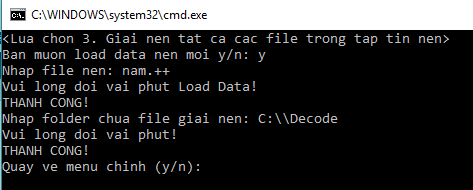
Hình 4. Menu chính



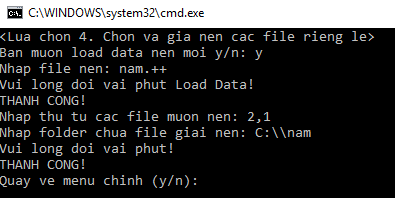
Hình 5. Menu 1



Hình 6. Menu 2



Hình 7. Menu 3



Hình 8. Menu 4

# TÀI LIỆU THAM KHẢO

[1]. [*Ths. Nguyễn Tri Tuấn, Đề bài P2, Khoa CNTT, Đại học KHTN - ĐHQG TP.HCM*](https://courses.fit.hcmus.edu.vn/pluginfile.php/88883/mod_resource/content/1/P2.pdf)

[2]. [*Ths. Nguyễn Tri Tuấn, slide Các giải thuật nén, Khoa CNTT, Đại học KHTN - ĐHQG TP.HCM*](https://www.dropbox.com/sh/d2o9grm8c1kdbgx/AABSfD7QyVsD6LE-F5zLAevxa?dl=0&preview=15.+Chuong+6+-+Cac+thuat+toan+nen.pdf)

[3]. [*Ths. Nguyễn Tri Tuấn, Hướng dẫn Static Huffman, Khoa CNTT, Đại học KHTN - ĐHQG TP.HCM*](https://drive.google.com/file/d/1TongqzBirq8WtTTYMGPl3jARp8hpLObZ/view?usp=sharing)

[4].[*CTDL-Lab06-Nen\_Huffman, Khoa CNTT, Đại học KHTN - ĐHQG TP.HCM*](https://drive.google.com/file/d/11urho8XzI60OLRq5q3340rp92mW0chVx/view?usp=sharing)

[5]. [*Trang web microsoft*](https://msdn.microsoft.com/en-us/library/windows/desktop/aa364418(v=vs.85).aspx)