

ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH

ĐẠI HỌC KHOA HỌC TỰ NHIÊN

KHOA CÔNG NGHỆ THÔNG TIN



BÁO CÁO KỸ THUẬT P2

MÔN: CTDL & GT

GVHD: THS. NGUYỄN TRI TUẤN

ĐẶNG PHƯƠNG NAM

1612406 | 16CTT3

TP.HCM, 11/12/2017

MỤC LỤC

PHỤ LỤC HÌNH ẢNH	2
LƯU Ý	2
I. YÊU CẦU ĐỀ^[1]	3
1. Nội dung	3
2. Chương trình.....	3
II. KỸ THUẬT	4
1. Cấu trúc file nén	4
2. Các cấu trúc dữ liệu quan trọng ^[2]	6
a. Thuật toán Huffman	6
b. Dữ liệu nén	7
3. Sơ đồ lớp.....	8
a. Class PRIORITY_QUEUE	8
b. Class H_NODE	8
c. Class HUFFMAN_TREE.....	9
d. Class STATIC_HUFFMAN.....	10
e. Class PROGRAM.....	10
4. Các hàm chính	11
a. Hàm tự code ^[3]	11
b. Hàm hỗ trợ ^[5]	15
III. SẢN PHẨM	16
IV. TÀI LIỆU THAM KHẢO.....	18

PHỤ LỤC HÌNH ẢNH

<i>Hình 1. Sơ đồ lớp</i>	<i>8</i>
<i>Hình 2. Bật bit i của một số nguyên.....</i>	<i>15</i>
<i>Hình 3. Lấy bit i của một số nguyên</i>	<i>15</i>
<i>Hình 4. Menu chính.....</i>	<i>16</i>
<i>Hình 5. Menu 1.....</i>	<i>16</i>
<i>Hình 6. Menu 2.....</i>	<i>17</i>
<i>Hình 7. Menu 3.....</i>	<i>17</i>
<i>Hình 8. Menu 4.....</i>	<i>17</i>

LƯU Ý

Toàn bộ Source code được viết bằng Visual Studio Community 2015

I. YÊU CẦU ĐỀ^[1]

1. 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).

2. 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

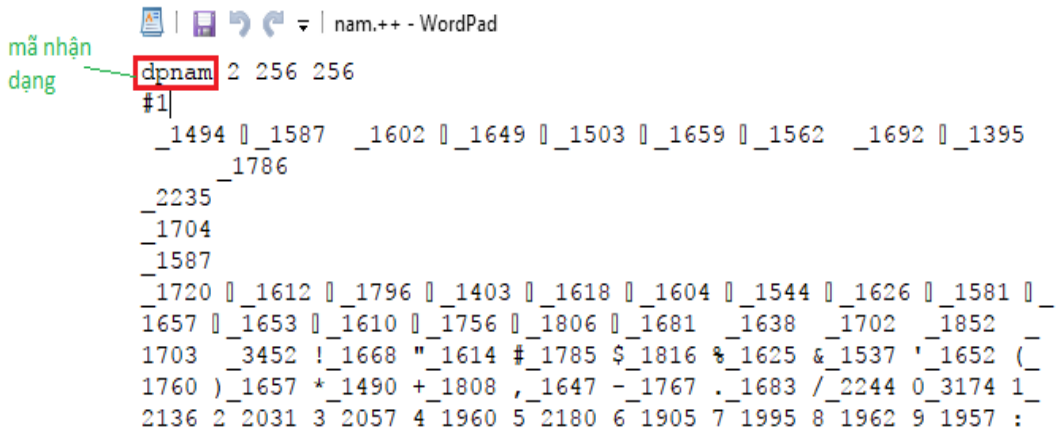
Menu	Ý nghĩa																
1	<ul style="list-style-type: none">• Yêu cầu user nhập vào folder chứa các file cần nén (folder_in) và nhập đường dẫn+tên file nén (file_out).• Chương trình sẽ nén (encode) tất cả các file trong folder_in và lưu kết quả nén vào file_out.																
2	<ul style="list-style-type: none">• Chương trình hiển thị danh sách tập tin chứa trong file nén file_out, theo mẫu sau: <table><tr><th>STT</th><th>Tên file</th><th>Size trước nén</th><th>Size sau nén</th></tr><tr><td>1.</td><td>xxxxxxxxxxxxx</td><td>xxxxx</td><td>xxxx</td></tr><tr><td>2.</td><td>xxxxxxxxxxxxx</td><td>xxxxx</td><td>xxxx</td></tr><tr><td>...</td><td></td><td></td><td></td></tr></table>	STT	Tên file	Size trước nén	Size sau nén	1.	xxxxxxxxxxxxx	xxxxx	xxxx	2.	xxxxxxxxxxxxx	xxxxx	xxxx	...			
STT	Tên file	Size trước nén	Size sau nén														
1.	xxxxxxxxxxxxx	xxxxx	xxxx														
2.	xxxxxxxxxxxxx	xxxxx	xxxx														
...																	
3	<ul style="list-style-type: none">• 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	<ul style="list-style-type: none">• 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.																

II. KỸ THUẬT

1. Cấu trúc file nén

HEADER FILE	Mã nhận dạng file //d p nam																											
	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																											
	<table border="1"> <thead> <tr> <th>STT</th><th>Tên file</th><th>Size trước nén</th><th>Size sau nén</th><th>Số lượng bits cuối có nghĩa</th></tr> </thead> <tbody> <tr> <td>int</td><td>string</td><td>unsigned long long</td><td>unsigned long long</td><td>unsigned short</td></tr> <tr> <td>1</td><td>xxxxxxx</td><td>xxxxx</td><td>xxxxx</td><td>x</td></tr> <tr> <td>2</td><td>xxxxxxx</td><td>xxxxx</td><td>xxxxx</td><td>x</td></tr> <tr> <td>.....</td><td></td><td></td><td></td><td></td></tr> </tbody> </table> Vd: 1 P2.pdf 12345 12222 6 2 main.cpp 4245 4300 8				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	xxxxxxx	xxxxx	xxxxx	x	2	xxxxxxx	xxxxx	xxxxx	x			
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	xxxxxxx	xxxxx	xxxxx	x																								
2	xxxxxxx	xxxxx	xxxxx	x																								
.....																												
DATA NÉN	Dữ liệu nén theo thứ tự Data 1 Data 2																											

File minh họa:

HEADER FILE	 <pre> mã nhận dạng dnam 2 256 256 #1 _1494 _1587 _1602 _1649 _1503 _1659 _1562 _1692 _1395 _1786 _2235 _1704 _1587 _1720 _1612 _1796 _1403 _1618 _1604 _1544 _1626 _1581 _ 1657 _1653 _1610 _1756 _1806 _1681 _1638 _1702 _1852 _ 1703 _3452 !_1668 "_1614 #_1785 \$_1816 %_1625 &_amp;_1537 ' _1652 (_ 1760)_1657 *_1490 +_1808 ,_1647 -_1767 ._1683 /_2244 0_3174 1_ 2136 2_2031 3_2057 4_1960 5_2180 6_1905 7_1995 8_1962 9_1957 :_ </pre>
----------------	--

1741 ; 1812 < 2304 = 1886 > 2398 ? 1776 @ 1492 A 1674 B 1641 C

- Phần data tạo cây Huffman 2

2 | Tuan 10.pdf | 152750 | 152328 | 4

DATA NÉN	<pre> {æ%δGéδδãf%YÖÖÇÄ+I?iç-†-Q·Çiç?ÖÖÇÇ³1,þ! >!Pyþ!çM³?)ùþ! >9% gδ5yñ÷÷ñóí ...ãûãîç™ Èöi÷-ëY÷ãã ÷BÄÈ)Y-@Y ÇÈ-VÇë?÷ / Ü\o iizß/ Ü\o... « ù "Gn ý%: 'ÜÖë- 6ã36iøw<" ÄÛ ý KSü- n"r;#t 3/oDsdî ò5[ÿ#HÉû Þ48çç4;JÄ-hèîÆCz S : ØÚ6 @G#»δ "ç³»é4N-tvì Ä~yÁ11! È= Ä µ sr i,!, °*÷÷1' ìiô S;- äâéóYWDZ.% eú÷^Ä...δ«;{1°Kÿ s , 'O;"δ@' >éäš Ç? ñ >±i è íz0 / hæ- úñòq`ÜÄ~çÖ,?; ó÷...ufgÜ;Ð» È'ÇÄ! øàÖPß OµÜ^ -Xm _ 0ÿçé@ æÑB ~-o•d BÈ[oç-•Üo8δ-O~8u^w÷Piã±qý²² Ço3"j7µ =Yç~n f° %2;F5Êi 4ÄN/ :E~ "-È«°r6%ù g)O...(? I >omDc"GèÜzÿ ÜÜ%Äò =;ðùné»×;ß×%³`k ôÜ4^~îs¹Èßó°iF `ú@δ»8 ^i#-G Þ2@~È>ØNÄ÷Ý²BûYs7àÜN@ó-øùò pó»/...7ÄÖ`R râ ÞizSàEu&uiué× 7 -úç<«I,, "àzÄzÜ"âIQùv÷B5c>ò`iX×^δ //..... >lã•(-Ü) ç`X`ãã' "G*·;× -ÖdB?äu¹±i=ç>,±±(a ?;çî6SÍÄ n+; J°oçIiósê±z% Ä>ñi²bm"çYœî #çš ü-Äsò šJ 'Ç B43Ú 3a^ê ;ÖþB³DÄ Té 4ãô*~ µ çEÖ³» "ÄÄä' ú µbo"ú Lxb5Pä...ØZœÜÈÈ ýß?B~ S; '+ÄññäXÜ[>äßrl Ošùt¹ æ 'hmq Óú ç`^~_] ž IççbB-â"yR ¶ñrlâ föÜ øi ,t vÈšœ89{d "±'YÖ ...ÍàiYœ #pÈ-,€ 8ø\$... :°-ÖÖbmçüñ È\$÷÷ò=óýö`U1 //..... </pre>	<p>Dữ liệu nén của file 1</p> <p>Dữ liệu nén của file 2</p>
-------------	--	---

2. Các cấu trúc dữ liệu quan trọng^[2]

a. Thuật toán Huffman

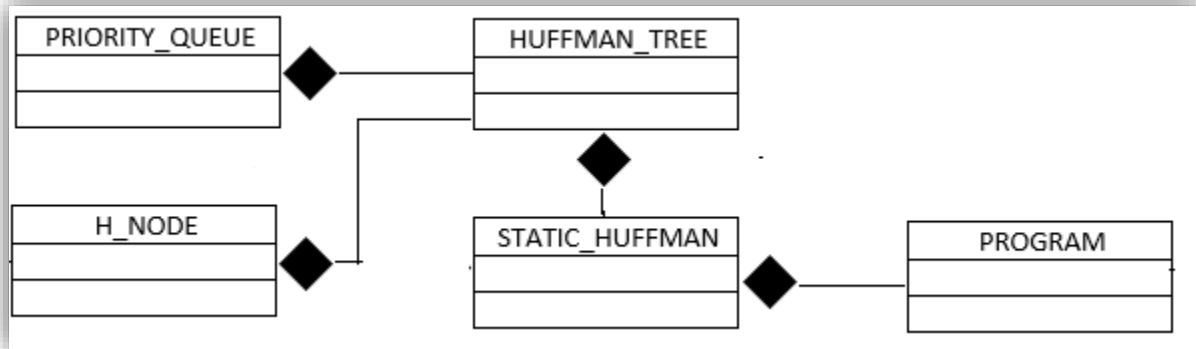
Hàng đợi ưu tiên	<pre> template <class T> class PRIORITY_QUEUE { private: T *items; int rear; int maxsize; void heapify(int); public: PRIORITY_QUEUE(); PRIORITY_QUEUE(int); PRIORITY_QUEUE(const PRIORITY_QUEUE&); ~PRIORITY_QUEUE(); bool isEmpty(); bool enqueue(T); bool dequeue(T&); bool minValue(T&); }; </pre>
------------------	---

Cây Huffman	<pre> 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); </pre>
Struct lưu trữ chuỗi char bits ứng cho mỗi loại ký tự	<pre> struct CODE_TABLE_ITEM { char *bits; // Mã bit của ký tự unsigned char nBitLen; // số bit thật sự sử dụng }; </pre>

b. Dữ liệu nén

Struct lưu trữ dữ liệu nén	<pre> 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 }; </pre>
----------------------------	---

3. Sơ đồ lớp



Hình 1. Sơ đồ lớp

a. Class PRIORITY_QUEUE

```
template <class T> class PRIORITY_QUEUE {
private:
    T *items; //mảng chứa các giá trị
    int rear; //số lượng phần tử 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
};
```

b. 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&);
};

```

c. Class HUFFMAN_TREE

```

class HUFFMAN_TREE {
private:
    H_NODE HuffTree[MAX_NODES];//511 node
    short int root;
private:
    //duyet 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&);//duyet cây để giải
nén
    };

```

d. 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
};

```

e. 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
};

```

4. Các hàm chính

a. Hàm tự code^[3]

```
#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 hàng 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 traverseTreeEncode(...)
//Ý nghĩa: duyệt cây để phục 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 traverseTreeDecode (...)
//Ý nghĩa: duyệt cây để phục 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
    với 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 cây Huffman

    //Tiến hành chuyển data nén thành chuỗi bits bằng thao tác bật
    tắt cả các bit 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]

- **Bật bit i của số nguyên out**

`out = out | (1 << i);`

Ví dụ: bật bit 2 của biến out

	7	6	5	4	3	2	1	0
Out	0	0	1	0	0	0	0	0
$1 << 2$	0	0	0	0	0	1	0	0
$out (1 << 2)$	0	0	1	0	0	1	0	0

Hình 2. Bật bit i của một số nguyên

- **Lấy bit i của số nguyên out**

`(out >> i) & 1`

Ví dụ: lấy bit 2 của biến out

	7	6	5	4	3	2	1	0
Out	0	0	1	0	0	1	0	0
$out >> 2$	0	0	0	0	1	0	0	1
1	0	0	0	0	0	0	0	1
$(out >> 2) \& 1$	0	0	0	0	0	0	0	1

Hình 3. Lấy bit i của một số nguyên

b. Hàm hỗ trợ^[5]

<pre>HANDLE WINAPI FindFirstFile(_In_ LPCTSTR lpFileName, _Out_ LPWIN32_FIND_DATA lpFindFileData);</pre>	//Dùng để xác định file đầu tiên trong folder
<pre>BOOL WINAPI FindNextFile(_In_ HANDLE hFindFile, _Out_ LPWIN32_FIND_DATA lpFindFileData);</pre>	//Dùng để xác định file kế tiếp trong folder bắt đầu từ file trước đó

<pre> DWORD WINAPI GetCurrentDirectory(_In_ DWORD nBufferLength, _Out_ LPTSTR lpBuffer); </pre>	//Lấy địa chỉ ban của folder
<pre> BOOL WINAPI SetCurrentDirectory(_In_ LPCTSTR lpPathName); </pre>	//Cố định địa chỉ thao tác tại folder
<pre> BOOL WINAPI CreateDirectory(_In_ LPCTSTR lpPathName, _In_opt_ LPSECURITY_ATTRIBUTES lpSecurityAttributes); </pre>	//Khởi tạo một folder mới

III. SẢN PHẨM

```

C:\WINDOWS\system32\cmd.exe
Nhan phim 1. Nen cac file trong forder
Nhan phim 2. Xem noi dung file da nen
Nhan phim 3. Giai nen tat ca cac file trong tap tin nen
Nhan phim 4. Chon va gia nen cac file rieng le
Nhan phim 5. Thoat!
Nhap lua chon:

```

Hình 4. Menu chính

```

C:\WINDOWS\system32\cmd.exe
<Lua chon 1. Nen cac file trong forder>
Nhap vao folder chua cac file can nen: C:\Users\PHUONG NAM\Downloads\Test
Nhap duong dan + ten file nen: nam.++
So luong file: 2
Vui long doi vai phut!
THANH CONG!
Quay ve menu chinh (y/n):

```

Hình 5. Menu 1

```
C:\WINDOWS\system32\cmd.exe
<Lua chon 2. Xem noi dung file da nen>
Ban muon load data nen moi y/n: y
Nhap file nen: nam.++
Vui long doi vai phut Load Data!
THANH CONG!
STT  Ten File                               Size truoc khi nen      Size sau nen
1    P2.pdf                                457533                  457428      (bytes)
2    Tuan_10.pdf                           152750                  152328      (bytes)
Quay ve menu chinh (y/n):
```

Hình 6. Menu 2

```
C:\WINDOWS\system32\cmd.exe
<Lua chon 3. Giai nen tat ca cac file trong tap tin nen>
Ban muon load data nen moi y/n: y
Nhap file nen: nam.++
Vui long doi vai phut Load Data!
THANH CONG!
Nhap folder chua file giai nen: C:\\Decode
Vui long doi vai phut!
THANH CONG!
Quay ve menu chinh (y/n):
```

Hình 7. Menu 3

```
C:\WINDOWS\system32\cmd.exe
<Lua chon 4. Chon va gia nen cac file rieng le>
Ban muon load data nen moi y/n: y
Nhap file nen: nam.++
Vui long doi vai phut Load Data!
THANH CONG!
Nhap thu tu cac file muon nen: 2,1
Nhap folder chua file giai nen: C:\\nam
Vui long doi vai phut!
THANH CONG!
Quay ve menu chinh (y/n):
```

Hình 8. Menu 4

IV. TÀI LIỆU THAM KHẢO

- [1]. *Ths. Nguyễn Tri Tuấn, Đề bài P2, Khoa CNTT, Đại học KHTN - ĐHQG TP.HCM*
- [2]. *Ths. Nguyễn Tri Tuấn, slide Các giải thuật nén, Khoa CNTT, Đại học KHTN - ĐHQG TP.HCM*
- [3]. *Ths. Nguyễn Tri Tuấn, Hướng dẫn Static Huffman, Khoa CNTT, Đại học KHTN - ĐHQG TP.HCM*
- [4]. *CTDL-Lab06-Nen Huffman, Khoa CNTT, Đại học KHTN - ĐHQG TP.HCM*
- [5]. *Trang web microsoft*