

**实 验 报 告**

**课程名称：** 数据结构

**实验名称：** 赫夫曼编码

**院 （系）：** 计算机与通信工程学院

**姓 名：** 原彬贺

**学 号：** 542001020223

**专业班级：** 计算机科学与技术20-02

**指导教师：** 南姣芬 王治国

2022 **年** 5 **月** 6 **日**

实验报告正文

1. **实验目的**

掌握赫夫曼树和赫夫曼编码的基本思想和算法的程序实现。

1. **实验内容及要求**

**实验内容：**

实现文件中数据的加解密与压缩：将硬盘上的一个文本文件进行加密，比较加密文件和原始文件的大小差别；对加密文件进行解密，比较原始文件和解码文件的内容是否一致。

**实验说明：**

1．输入和输出：

（1）输入：硬盘上给定的原始文件及文件路径。

（2）输出：

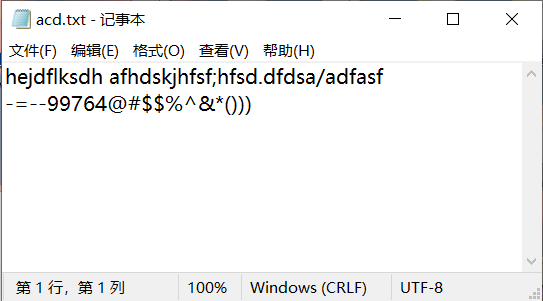
* + 硬盘上的加密文件及文件路径；
  + 硬盘上的解码文件及文件路径；
  + 原始文件和解码文件的比对结果。

2．实验要求：

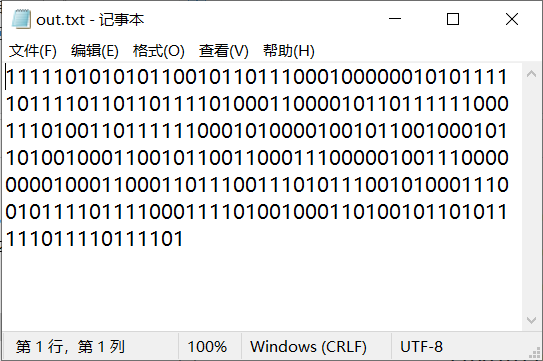
* 提取原始文件中的数据（包括中文、英文或其他字符），根据数据出现的频率为权重，构建Huffman编码表；
* 根据Huffman编码表对原始文件进行加密，得到加密文件并保存到硬盘上；
* 将加密文件进行解密，得到解码文件并保存点硬盘上；
* 比对原始文件和解码文件的一致性，得出是否一致的结论。

1. **测试**
   1. 结果

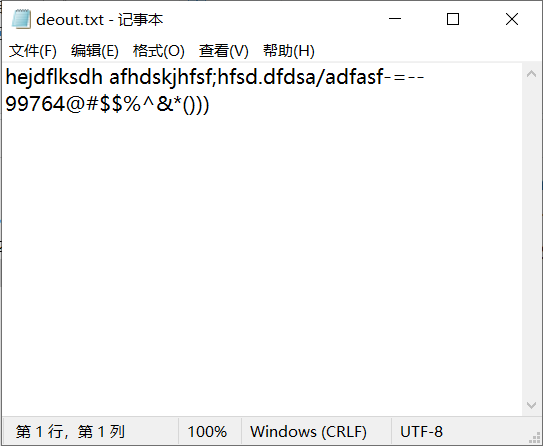
原文件：



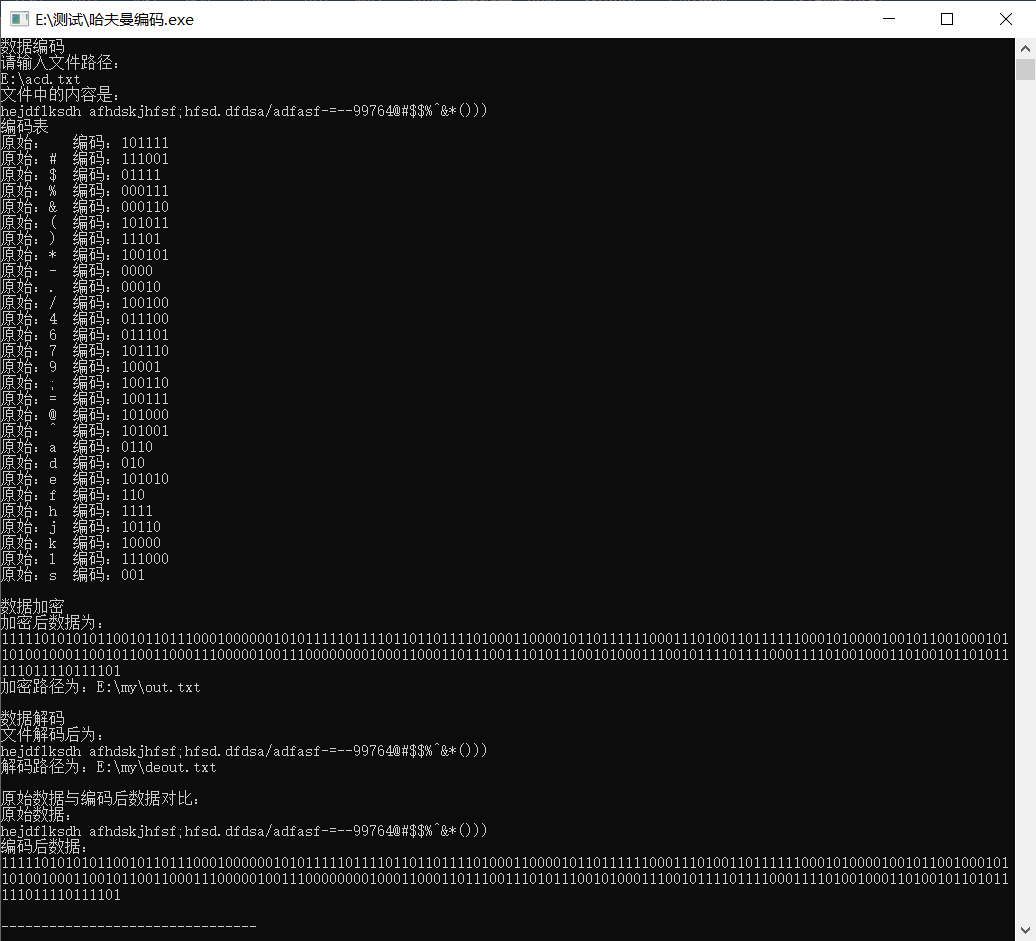
编码文件：



解码文件：



执行结果：



1. **总结与讨论**

本实验通过对目标文件的路径的输入，程序对该文件进行读取数据，并对每个字符统计其出现的频率，再根据其频率来构建该文件专属的赫夫曼树，再由该赫夫曼树生成每个字符对应的专属赫夫曼编码分别生成两个以原始字符为key和赫夫曼编码为key的map ，原始字符为key的map进行编码，和而赫夫曼编码为key的map进行数据的解码，并分别将编码与解码的文件存入系统并输出该文件路径，同时输出编码与解码的对比结果。通过这个小程序，使我对赫夫曼树有了更深刻的理解，同时对我编写代码的规范性，已经降低代码之间的耦合度帮助很大，使我收获很多。

**附：程序的源代码**

#include<iostream>

#include<stdio.h>

#include<string>

#include<vector>

#include<map>

#include <algorithm>

#include <fstream>

using namespace std;

typedef class treenote \* pnote;

class treenote{

public:

char da;

int num;

pnote left;

pnote right;

treenote(char a,int b,pnote c,pnote r):da(a),num(b),left(c),right(r){

}

treenote()

{

da=0;

num=0;

left=NULL;

right=NULL;

}

};

void input(string &Data,string &path);

void test1();

string Huffman(string &Data,map<string,char>& code,map<char,string> decode);

void decodes(string s,map<string,char>& code);

void mid(pnote s,string si,map<string,char>& code,map<char,string> &decode);

bool cm(pair<char ,int>a,pair<char,int>b);

bool cmt(treenote a,treenote b);

void compare(string s,string c);

int main()

{

test1();

return 0;

}

void test1()

{

vector<int> anum;

string Data,path;

map<string,char> code;

map<char,string> decode;

string taget;

input(Data,path);

taget=Huffman(Data,code, decode);

decodes(taget,code);

compare(Data,taget);

}

void input(string &Data,string &path)

{

cout<<"数据编码"<<endl;

cout<<"请输入文件路径："<<endl;

cin>>path;

ifstream file(path,ios::in);

if(!file.is\_open())

{

cout<<"open file error"<<endl;

return ;

}

string len;

while(getline(file,len))

{

Data+=len;

len="\n";

}

cout<<"文件中的内容是："<<endl;

cout<<Data<<endl;

file.close();

return;

}

string Huffman(string &Data,map<string,char>& code,map<char,string> decode)

{

map<char,int> Datew;

for(auto em:Data)

{

map<char,int> ::iterator itr=Datew.find(em);

if(itr==Datew.end())

{

Datew.insert(pair<char,int>(em,1));

}

else

{

itr->second+=1;

}

}

map<char,int> ::iterator itr=Datew.begin();

vector<pair<char,int >> nums(Datew.begin(),Datew.end());

sort(nums.begin(),nums.end(),cm);

vector<pair<char,int >> ::iterator itrs=nums.begin();

vector<treenote> tree;

for(;itrs!=nums.end();itrs++)

{

treenote ar(itrs->first,itrs->second,NULL,NULL);

tree.push\_back(ar);

}

while(1)

{

vector<treenote>::iterator itt1=tree.end()-1;

if(itt1==tree.begin())

{

break;

}

vector<treenote>::iterator itt2=itt1-1;

pnote p1=new treenote(itt1->da,itt1->num,itt1->left,itt1->right);

pnote p2=new treenote(itt2->da,itt2->num,itt2->left,itt2->right);

tree.erase(itt2,tree.end());

treenote tr('0',p1->num+p2->num,p1,p2);

tree.push\_back(tr);

sort(tree.begin(),tree.end(),cmt);

}

treenote sno=tree[0];

pnote tr=&tree[0];

string str="";

mid(tr,str,code,decode);

cout<<"编码表"<<endl;

for(auto em:decode)

{

cout<<"原始："<<em.first<<" 编码："<<em.second<<endl;

}

string taget="";

map<char,string> ::iterator ita;

for(auto em:Data)

{

ita=decode.find(em);

if(ita!=decode.end())

{

taget+=ita->second;

}

}

cout<<endl;

cout<<"数据加密"<<endl;

cout<<"加密后数据为："<<endl;

cout<<taget<<endl;

ofstream os;

cout<<"加密路径为：" <<"E:\\my\\out.txt"<<endl;

os.open("E:\\my\\out.txt", ios::trunc);

if (os.is\_open()) {

os << taget << endl;

os.close();}

cout<<endl;

return taget;

}

void mid(pnote s,string si,map<string,char>& code,map<char,string> &decode)

{

pnote sa=s->left;

if(sa==NULL)

{

code.insert(pair<string,char>(si,s->da));

decode.insert(pair<char,string>(s->da,si));

return;

}

si=si+"0";

mid(sa,si,code,decode);

si.erase(si.end()-1);

sa=s->right;

si=si+"1";

mid(sa,si,code,decode);

}

void decodes(string s,map<string,char>& code)

{

string str="";

string sta="";

map<string,char>::iterator its;

for(auto em:s)

{

str+=em;

its=code.find(str);

if(its!=code.end())

{

sta+=its->second;

str="";

}

}

cout<<"数据解码"<<endl;

cout<<"文件解码后为："<<endl;

cout<<sta<<endl;

ofstream os;

cout<<"解码路径为：" <<"E:\\my\\deout.txt"<<endl;

cout<<endl;

os.open("E:\\my\\deout.txt", ios::trunc);

if (os.is\_open()) {

os << sta << endl;

os.close();}

return ;

}

bool cm(pair<char ,int>a,pair<char,int>b)

{

return a.second>b.second;

}

bool cmt(treenote a,treenote b)

{

return a.num>b.num;

}

void compare(string s,string c)

{

cout<<"原始数据与编码后数据对比："<<endl;

cout<<"原始数据："<<endl;

cout<<s<<endl;

cout<<"编码后数据："<<endl;

cout<<c<<endl;

return ;

}