#include<stdlib.h>

#include<string.h>

#include<stdio.h>

#include<fstream>

#include<string>

#include<map>

#include<iostream>

using namespace std;

#define \_CRT\_SECURE\_NO\_WARNINGS

typedef struct {

int weight;

int parent;

int lchild;

int rchild;

}HTNode,\*HuffmanTree;

typedef char\*\* HuffmanCode;

void FrequencyTable(const char \*filepath, const char \*output, map<char,int> &freq) {

string str;

ifstream in(filepath, ios::in);

while (getline(in, str)) {

for (int i = 0; i != str.size(); i++) {

if (str[i] == ' ')str[i] = '-';//'-' for space

if (freq.end() == freq.find(str[i])) {

freq.insert(make\_pair(str[i], 1));

}

else freq[str[i]]++;

}

if (freq.end() == freq.find('\n'))freq.insert(make\_pair('!', 1));//'!' for end of line

else freq['!']++;

}

freq.insert(make\_pair('+', 1));//'+' for end of message

ofstream out(output, ios::out);

out << "字符\t频率" << endl;

map<char, int>::iterator iter;

for (iter = freq.begin(); iter != freq.end(); iter++) {

out << (\*iter).first << '\t' << (\*iter).second << endl;

}

in.close();

out.close();

}

void CodeFile(const char\* output,map<char,char\*> &code,map<char,int> freq,HuffmanCode HC) {

int i = 1;

map<char, int>::iterator fiter = freq.begin();

while (fiter != freq.end()) {

code.insert(make\_pair((\*fiter).first, HC[i]));

i++;

fiter++;

}

ofstream out(output, ios::out);

out << "字符\t代码" << endl;

map<char, char\*>::iterator citer = code.begin();

for (; citer != code.end(); citer++) {

out << (\*citer).first << '\t' << (\*citer).second << endl;

}

out.close();

}

void Select(HuffmanTree HT, int n, int &s1, int &s2){

int i, min;

for (i = 1; i <= n; i++){

if (HT[i].parent == 0){

min = i;

break;

}

}

for (i = 1; i <= n; i++){

if (HT[i].parent == 0){

if (HT[i].weight<HT[min].weight)

min = i;

}

}

s1 = min;

for (i = 1; i <= n; i++){

if (HT[i].parent == 0 && i != (s1)){

min = i;

break;

}

}

for (i = 1; i <= n; i++){

if (HT[i].parent == 0 && i != (s1)){

if (HT[i].weight<HT[min].weight)

min = i;

}

}

s2 = min;

}

void HuffmanCoding(HuffmanTree &HT, HuffmanCode &HC, int \*w, int n) {

//w存放n个字符的权值，构造树HT，求出n个字符的赫夫曼编码HC

if (n <= 1)return;

int m = 2 \* n - 1;

HT = new HTNode[m+1];//0号单元未用

HuffmanTree p = NULL;

int i;

for (p = HT + 1, i = 1; i <= n; i++, p++, w++) {

\*p = { \*w,0,0,0 };

}//初始化表的前n行

for (; i <= m; i++, p++) {

\*p = { 0,0,0,0 };

}//初始化表的n+1到m行

for (i = n + 1; i <= m; i++) {

//在HT[1...i-1]选择parent为0且weight最小的两个结点，其序号分别为s1和s2

int s1, s2;

Select(HT, i - 1, s1, s2);

HT[s1].parent = i; HT[s2].parent = i;

HT[i].lchild = s1; HT[i].rchild = s2;

HT[i].weight = HT[s1].weight + HT[s2].weight;

}

//——从叶子到根逆序求每个字符的赫夫曼编码

HC = new char\*[n+1];//分配n个字符编码的头指针向量

char\* cd = new char[n];//分配求编码的工作空间

cd[n - 1] = '\0';//编码结束符

for (i = 1; i <= n; i++) {//逐个字符求赫夫曼编码

int start = n - 1;//编码结束符位置

for (int c = i, f = HT[i].parent; f != 0; c = f, f = HT[f].parent) {//从叶子到根逆向求编码

if (HT[f].lchild == c) cd[--start] = '0';

else cd[--start] = '1';

}

HC[i] = new char[n - start];

strcpy(HC[i], &cd[start]);

}

delete[] cd;

}

void Encodedmessage(const char\* infile, const char \*outfile, map<char, char\*>code) {

ifstream in(infile, ios::in);

ofstream out(outfile, ios::out | ios::binary);

string str;

int i = 0;

while (getline(in,str)) {

for (i = 0; i < str.size(); i++) {

if (str[i] == ' ') out << code['-'];

else out << code[str[i]];

out << ' ';

}

out << code['!'] << ' ';

}

out << code['+'];

in.close();

out.close();

}

void Decodemessage(const char\* infile, const char \*outfile, map<char, char\*>code) {

ifstream in(infile, ios::in | ios::binary);

ofstream out(outfile, ios::out);

char cd[100];

map<char, char\*>::iterator iter;

while (in.getline(cd, 20, ' ')) {

for (iter = code.begin(); iter != code.end(); iter++) {

if (!strcmp(cd,(\*iter).second)) {

if ((\*iter).first == '-')out << ' ';

else if ((\*iter).first == '!')out << '\n';

else if ((\*iter).first == '+')break;

else out << (\*iter).first;

}

}

}

in.close();

out.close();

}

void main() {

map<char, int> freq;

FrequencyTable("A4-message.txt", "frequency.txt", freq);

int n = freq.size();

int \*w = new int[n];

map<char, int>::iterator fiter = freq.begin();

for (int i = 0; i < n; i++) {

w[i] = (\*fiter).second;

fiter++;

}

HuffmanTree HT = NULL;

HuffmanCode HC;

HuffmanCoding(HT, HC, w, n);

map<char, char\*> code;

map<char, char\*>::iterator citer = code.begin();

CodeFile("code.txt", code, freq, HC);

Encodedmessage("A4-message.txt", "encodedmessage.bin", code);

Decodemessage("encodedmessage.bin", "decodedmessage.txt", code);

delete[] w;

code.clear(); freq.clear();

cout << "Success!" << endl;

}

心得：

本次作业用赫夫曼树进行编码，让我对二叉树的理解进一步加深，学会了建立二叉树，逆向遍历数等等。

此外，为了完成本次作业，用到了STL，使我对文件和STL的掌握加强，在学习数据结构的同时，提高了对C++语法的认识。