#include<iostream>

#include<math.h>

#include<string>

#include<fstream>

#include<iomanip>//设置字符串长度及其对齐方式

using namespace std;

string keyword\_table[73];

int ASCII\_table[73];

const int M = 10000;

struct keyword\_information {

string data;

int code;

int count;

};

struct user\_identifier {

string data;

int code;

int count;

};

keyword\_information key\_table[73];

user\_identifier user\_table[100];

void SetHash() {//通过读取文件，建立哈希表

ifstream ifile("keywords.txt", ios::in);

if (!ifile)

cerr << "fail to open" << endl;

string key;

int i = 0;

while (getline(ifile, key)) {//逐行读入

keyword\_table[i] = key;

ASCII\_table[i] = 0;

int j = key.size();

for (int k = 0; k < j; k++)

ASCII\_table[i] = ASCII\_table[i] + int(key[k]);//将每个关键字的ASCII码总和暂存于ASCII\_table中

i++;

}

ifile.close();

}

void InputKT() {//将关键字及其ASCII码总和存于key\_table中

for (int i = 0; i < 73; i++)

key\_table[i].code = -1;

for (int i = 0; i < 73; i++) {

int j = ASCII\_table[i] % 73;

if (key\_table[j].code == -1) {

key\_table[j].code = ASCII\_table[i];

key\_table[j].data = keyword\_table[i];

}

else {

while (key\_table[j].code != -1) {

j = (j + 1) % 73;

if (key\_table[j].code == -1) {

key\_table[j].code = ASCII\_table[i];

key\_table[j].data = keyword\_table[i];

break;

}

}

}

}

}

int FilterFile(string name1, string name2) {//读取要比较的文件，并将其中的注释部分忽略，然后输入进另一文件

ifstream ifile(name1, ios::in);

while (!ifile) {

cerr << "fail to open" << endl;

exit(0);

}

char ch;

string word = "";

while (ifile.get(ch)) {

if (ch == '/') {

ifile.get(ch);

if (ch == '/') {

ifile.get(ch);

while (ch != '\n')

ifile.get(ch);

word = word + ' ';

}

else if (ch == '\*') {

for (;;) {

ifile.get(ch);

if (ch == '\*') {

ifile.get(ch);

if (ch == '/') {

word = word + ' ';

break;

}

}

}

}

}

else if ((ch >= 'A'&&ch <= 'Z') || (ch >= 'a'&&ch <= 'z') || ch == '\_' || (ch >= '0'&&ch <= '9'))

word = word + ch;

else

word = word + ' ';

}

ifile.close();

ofstream ofile(name2, ios::out);

ofile << word;

ofile.close();

}

int InputArray(string name, string word[M]) {

ifstream ifile(name, ios::in);

char ch;

int j = 0, k = 0;

string input;

string str = "";

while (ifile.get(ch)) {//将上一函数中文件中的内容全部读入input字符串中

input = input + ch;

k++;

}

ifile.close();

for (int i = 0; i < k; i++) {//对input字符串进行分词，并录入word字符串数组中

if (input[i] != ' ')

str = str + input[i];

else if (input[i] == ' '&&input[i + 1] != ' '&&i != 0) {

word[j] = str;

j++;

str = "";

}

}

ofstream ofile(name, ios::out);//将字符串数组录入文件

for (int k = 0; k < j; k++)

ofile << word[k] << " ";

ofile.close();

return j;//返回的是word中的单词数量

}

int UserStatisticMixed(string word[M], int m, string u[], int v[], int w[]) {

string user\_key[100];

int j = 0;

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

if (word[i] == "int"&&word[i + 1] != "main") {

user\_key[j] = word[i + 1];

j++;

}

else if (word[i] == "short") {

user\_key[j] = word[i + 1];

j++;

}

else if (word[i] == "long") {

user\_key[j] = word[i + 1];

j++;

}

else if (word[i] == "double") {

user\_key[j] = word[i + 1];

j++;

}

else if (word[i] == "float") {

user\_key[j] = word[i + 1];

j++;

}

else if (word[i] == "char") {

user\_key[j] = word[i + 1];

j++;

}

else if (word[i] == "void") {

user\_key[j] = word[i + 1];

j++;

}

else if (word[i] == "class") {

user\_key[j] = word[i + 1];

j++;

}

else if (word[i] == "struct") {

user\_key[j] = word[i + 1];

j++;

}

}//统计用户标识符,并存入user\_key数组中

for (int i = 0; i < j - 1; i++) {

for (int a = i + 1; a < j; a++) {

if (user\_key[i] == user\_key[a]) {

for (int b = a; b < j; b++)

user\_key[b] = user\_key[b + 1];

j--;

}

}

}

for (int i = 0; i < j; i++)

user\_table[i].code = -1;

int \*code = new int[j];

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

string str = "";

str = user\_key[i];

int k = str.size();

code[i] = 0;

for (int j = 0; j < k; j++)

code[i] = code[i] + int(str[j]);

}//将用户标识符转换成ASCII码形式，一一对应

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

int k = code[i] % j;

if (user\_table[k].code == -1) {

user\_table[k].code = code[i];

user\_table[k].data = user\_key[i];

}

else {

while (user\_table[k].code != -1) {

k = (k + 1) % j;

if (user\_table[k].code == -1) {

user\_table[k].code = code[i];

user\_table[k].data = user\_key[i];

break;

}

}

}

}

for (int i = 0; i < j; i++)

user\_table[i].count = 0;

int \*code1 = new int[m];

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

string str = "";

str = word[i];

int k = str.size();

code1[i] = 0;

for (int j = 0; j < k; j++)

code1[i] = code1[i] + int(str[j]);

}

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

int o = code1[i] % j;

if (user\_table[o].code == code1[i])

user\_table[o].count++;

else {

int u = o;

o = o + 1;

while (o % j != u) {

int n = o % j;

if (user\_table[n].code != -1 && user\_table[n].code == code1[i]) {

user\_table[n].count++;

break;

}

else o++;

}

}

}

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

if (user\_table[i].count == 0) {

for (int k = i; k < j - 1; k++)

user\_table[k] = user\_table[k + 1];

j--;

}

}

cout << "\t" << "序号" << "\t\t" << "关键字" << "\t\t\t\t" << "ASCII值" << "\t\t" << "频度" << endl;

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

cout << "\t" << i << "\t\t";

cout << setiosflags(ios::left) << setw(20) << user\_table[i].data;

cout << "\t\t" << user\_table[i].code << "\t\t" << user\_table[i].count << endl;

}

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

u[i] = user\_table[i].data;

v[i] = user\_table[i].code;

w[i] = user\_table[i].count;

}

return j;

}

void TransformCode(string word[], int j, int code[]) {

string str = "";

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

str = word[i];

int k = str.size();

code[i] = 0;

for (int j = 0; j < k; j++)

code[i] = code[i] + int(str[j]);

str = "";

}

}

void Statistics(int x[], int word[], int len) {

for (int i = 0; i < 73; i++)

key\_table[i].count = 0;

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

int m = word[i] % 73;

if (key\_table[m].code == word[i])

key\_table[m].count++;

else {

int u = m;

m = m + 1;

while (m % 73 != u) {

int n = m % 73;

if (key\_table[n].code != -1 && key\_table[n].code == word[i]) {

key\_table[n].count++;

break;

}

else m++;

}

}

}

for (int i = 0; i < 73; i++)

x[i] = key\_table[i].count;

}

double similarity(int a[], int b[]) {

double sum = 0;

for (int i = 0; i < 73; i++)

sum = sum + (a[i] - b[i])\*(a[i] - b[i]);

double sim = sqrt(sum);

return sim;

}

int main() {

SetHash();

InputKT();

char ch;

cout << "按任意键开始比较，按‘E’键结束程序" << endl;

while (cin >> ch && ch != 'E') {

string word1[M], word2[M];

string file1, file2;

cout << "请输入第一个要比较的文件的文件名" << endl;

cin >> file1;

FilterFile(file1, file1);

cout << "请输入第二个要比较的文件的文件名" << endl;

cin >> file2;

FilterFile(file2, file2);

int len1 = InputArray(file1, word1);

int len2 = InputArray(file2, word2);

int w1[M], w2[M];

TransformCode(word1, len1, w1);//将word字符串数组中的字符，转化成ASCII码之和，存入w的整型数组中

TransformCode(word2, len2, w2);

int x1[73], x2[73];

Statistics(x1, w1, len1);

Statistics(x2, w2, len2);

string u1[50], u2[50];

int v1[50], v2[50];

int w3[50], w4[50];

cout << "文件1的用户标识符统计如下：" << endl;

int p = UserStatisticMixed(word1, len1, u1, v1, w3);

cout << endl;

cout << "文件2的用户标识符统计如下：" << endl;

int q = UserStatisticMixed(word2, len2, u2, v2, w4);

cout << endl;

int z = p + q;

int \*r = new int[z];

int \*t = new int[z];

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

r[i] = 0;

t[i] = 0;

}

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

for (int j = 0; j < q; j++) {

if (u1[i] == u2[j]) {

t[i] = w4[j];

z--;

for (int k = j; k < q; k++) {

u2[k] = u2[k + 1];

v2[k] = v2[k + 1];

w4[k] = w4[k + 1];

}

q--;

break;

}

}

}

string \*r2 = new string[z];

for (int i = 0; i < p; i++)

r2[i] = u1[i];

int j1 = 0;

for (int i = p; i < z; i++) {

r2[i] = u2[j1];

j1++;

}

for (int i = 0; i < p; i++)

r[i] = w3[i];

int j2 = 0;

for (int i = p; i < z; i++) {

t[i] = w4[j2];

j2++;

}

int \*r3 = new int[z];

for (int i = 0; i < p; i++)

r3[i] = v1[i];

int j3 = 0;

for (int i = p; i < z; i++) {

r3[i] = v2[j3];

j3++;

}

cout << "文件1与文件2的用户标识符整合：" << endl;

cout << "\t" << "序号" << "\t\t" << "关键字" << "\t\t\t\t" << "ASCII值" << "\t\t" << "频度1" << "\t\t" << "频度2" << endl;

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

cout << "\t" << i << "\t\t";

cout << setiosflags(ios::left)/\*左对齐方式输出\*/ << setw(20)/\*设域宽为20个字符\*/ << r2[i];

cout << "\t\t" << r3[i] << "\t\t" << r[i] << "\t\t" << t[i] << endl;

}

int sum1 = 0;

double sim1 = 0;

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

sum1 = sum1 + (r[i] - t[i])\*(r[i] - t[i]);

sim1 = sqrt(sum1);

}

cout << "以用户标识符为基准的相对距离为：" << sim1 << endl;

cout << endl;

cout << "文件1与文件2的关键字统计如下：" << endl;

double s = similarity(x1, x2);

cout << "\t" << "序号" << "\t\t" << "关键字" << "\t\t\t\t" << "ASCII值" << "\t\t" << "频度1" << "\t\t" << "频度2" << endl;

for (int i = 0; i < 73; i++) {

cout << "\t" << i << "\t\t";

cout << setiosflags(ios::left)/\*左对齐方式输出\*/ << setw(20)/\*设域宽为20个字符\*/ << key\_table[i].data;

cout << "\t\t" << key\_table[i].code << "\t\t" << x1[i] << "\t\t" << x2[i] << endl;

}

cout << endl;

cout << "这两个源代码的相对距离为：" << s << endl;

if (s > 20)

cout << "这两个代码基本不可能抄袭" << endl;

else if (sim1 >= 10 && (s >= 10 && s <= 20))

cout << "这两个代码可能存在小部分抄袭" << endl;

else if ((s >= 10 && s <= 20 && sim1 < 10) || (s < 10 && sim1 >= 10))

cout << "这两个代码可能存在大部分抄袭" << endl;

else

cout << "这两个代码可能为全部抄袭" << endl;

cout << "按任意键进行下一轮对比，按‘E’键结束程序" << endl;

delete r, t, r2, r3;

}

return 0;

}