318 - Assignment 1

Nate Paternoster

**Code:**

#include <iostream>

#include <fstream>

#include <string>

using namespace std;

class person {

public:

string SSN, dob, fname, lname;

person(string s, string d, string f, string l);

virtual void print();

bool bigger(person\*);

bool equal(person\*);

};

person::person(string s, string d, string f, string l):

SSN(s), dob(d), fname(f), lname(l)

{ }

void person::print() {

cout << SSN << " " << fname << " " << lname << " " << dob << "\n";

}

bool person::bigger(person\* p) {

if (lname>p->lname) return true;

else if (lname<p->lname) return false;

else if (lname==p->lname) {

if (fname>p->lname) return true;

else if (fname<p->lname) return false;

else if (fname==p->fname) {

if (dob>p->dob) return true;

else return false;

}

}

}

bool person::equal(person\* p) {

if (lname == p->lname) {

if (fname == p->fname) {

if (dob == p->dob) {

if (SSN == p->SSN) return true;

}

}

}

else return false;

}

void read\_file(string file, int size, person\* A[]) {

ifstream f(file.c\_str());

if (f.fail()) {

cout << "Failed to open file\n";

exit(1);

}

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

string S, D, F, L;

f >> S >> F >> L >> D;

if (f.fail()) break;

A[i] = new person(S, D, F, L);

}

f.close();

}

void swap(person\* &a, person\* &b) {

person\* temp = a;

a = b;

b = temp;

}

int partition(person\* A[], int begin, int end) {

int median = end/2;

int pivpos;

if (A[median]->lname >= A[begin]->lname) {

if (A[median]->lname <= A[end]->lname) pivpos = median;

else if (A[end]->lname >= A[begin]->lname) pivpos = end;

else pivpos = begin;

}

else if (A[median]->lname >= A[end]->lname) {

if (A[median]->lname <= A[begin]->lname) pivpos = median;

else if (A[end]->lname >= A[begin]->lname) pivpos = end;

else pivpos = begin;

}

else {

if (A[begin]->lname >= A[end]->lname) pivpos = end;

else pivpos = begin;

}

int L = begin, R = end;

while (L <= R) {

while (L <= end) {

if (A[L]->equal(A[pivpos])) break;

if (A[L]->bigger(A[pivpos])) break; //if A[L]>A[pivpos] break

L++; //otherwise increment L

}

while (R >= begin) {

if (A[R]->equal(A[pivpos])) break;

if (A[pivpos]->bigger(A[R])) break; //if A[R]<A[pivpos] break

R--; //otherwise decrement R

}

if (L <= R) {

swap(A[L],A[R]);

L++;

R--;

}

}

if (L < pivpos && A[L]->bigger(A[pivpos])) { //if L<pivpos and A[L]>A[pivpos]

swap(A[L],A[pivpos]);

int temp = L;

L = pivpos;

pivpos = temp; //updating L to pivot's pos

L++;

}

else if (R > pivpos && A[pivpos]->bigger(A[R])) {

swap(A[R],A[pivpos]);

int temp = R;

R = pivpos;

pivpos = temp;

R--;

}

return pivpos;

}

void quicksort(person\* A[], int begin, int end) {

// cout << "Quicksort, Begin= " << begin << " end = " << end << "\n";

if (end > begin) {

int pivot = partition(A,begin,end);

quicksort(A,begin,pivot);

quicksort(A,pivot+1,end);

}

return;

}

void main() {

int size;

string filename;

// cout << "Size of database: ";

// cin >> size;

size = 20; //this is temporary

filename = "/home/www/class/een118/database1.txt"; //this is temporary

person\* database[size];

/\* if (size=1000) filename = "/home/www/class/een118/database1.txt";

else if (size=2000) filename = "/home/www/class/een118/database2.txt";

else if (size=3000) filename = "/home/www/class/een118/database3.txt";

else if (size=5000) filename = "/home/www/class/een118/database5.txt";

else if (size=10000) filename = "/home/www/class/een118/database10.txt";

else if (size=20000) filename = "/home/www/class/een118/database20.txt";

else if (size=30000) filename = "/home/www/class/een118/database30.txt";

else if (size=50000) filename = "/home/www/class/een118/database50.txt";

else if (size=100000) filename = "/home/www/class/een118/database100.txt";

else {

cout << "Invalid file size!\n";

filename = "NULL";

}\*/

if (filename != "NULL") {

read\_file(filename, size, database);

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

database[i]->print();

}

quicksort(database,0,size-1);

cout << "\n";

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

database[i]->print();

}

}

}

