EEN318 Lab 2 – Mergesort

Nate Paternoster

9/27/13

#include <iostream>

#include <fstream>

#include <string>

using namespace std;

class person {

public:

string SSN, fname, lname, dob;

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

virtual void print();

bool bigger(person\* p);

};

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

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

{ }

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->fname) return true;

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

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

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

else return false;

}

}

}

void read\_file(string filename, int size, person\*\* & A) {

A = new person\* [size];

ifstream fin(filename.c\_str());

if (fin.fail()) {

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

exit(1);

}

string S, F, L, D;

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

fin >> S >> F >> L >> D;

if (fin.fail()) break;

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

}

fin.close();

return;

}

void split(person\*\* & a, person\*\* & b, person\*\* & c, int & size, int & bsize, int & csize) {

b = new person\* [size/2];

c = new person\* [size-size/2];

bool flip = false;

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

if (flip) {

b[i] = a[k];

i++;

bsize++;

}

else {

c[j] = a[k];

j++;

csize++;

}

flip = !flip;

}

}

void merge(person\*\* & b, person\*\* & c, person\*\* & a, int & bsize, int & csize) {

int acounter = 0, bcounter = 0, ccounter = 0;

while (bcounter<bsize && ccounter<csize) {

if (b[bcounter]->bigger(c[ccounter])) {

a[acounter] = c[ccounter];

acounter++; ccounter++;

}

else if (b[bcounter]->bigger(c[ccounter])==false) {

a[acounter] = b[bcounter];

acounter++; bcounter++;

}

}

while (bcounter < bsize) {

a[acounter] = b[bcounter];

acounter++; bcounter++;

}

while (ccounter < csize) {

a[acounter] = c[ccounter];

acounter++; ccounter++;

}

delete [] b;

delete [] c;

b = NULL;

c = NULL;

}

void mergesort(person\*\* & a, int size) {

if (size <= 1) return;

int bsize = 0, csize = 0;

person\* \*b = NULL;

person\* \*c = NULL;

split(a, b, c, size, bsize, csize);

mergesort(b, bsize);

mergesort(c, csize);

merge(b, c, a, bsize, csize);

}

void main() {

person\* \*database = NULL;

int size;

string filename;

cout << "Size of database: ";

cin >> 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("/home/www/class/een118/database100.txt", size, database);

mergesort(database,size);

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

database[i]->print();

}

}

}

