Assignment 7- cs 246

Yu Sheng

04/01/2018

//Makefile

a7: a7.o date.o flight.o

gcc -o a7 a7.o date.o flight.o

a7.o: flight.h a7.c

gcc -c a7.c

data.o: date.h date.c

gcc -c data.c

flight.o: flight.h flight.c

gcc -c flight.c

clear:

rm a7 \*.o

Output

[jsheng@rukbat a7]$ rm a7\*.o

[jsheng@rukbat a7]$ make a7

gcc -c a7.c

gcc -o a7 a7.o date.o flight.o

--------------------------------------------------Code--------------------------------------------------

//Assignment 7

//cs 246

//Yu Sheng

//All modified code are highlighted

// File: a7.c

// PURPOSE: To read in a flight database as described in A7

// and answer some queries about flights

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <time.h>

#include "flight.h"

void readData(char \*file, Flight list[], int \*n); // read database

int main(int argc, char \*argv[]) {

srand(time(NULL));

Flight flights[200000];

// NOTE: this array of Flight is being STATICALLY ALLOCATED

// There are a little less than 200,000 flights!

// The data is a little less than 10MB (!!)

// Not a good idea to allocate such a large static array

// In fact, the operating system by default will not allow this.

// You will bneed to increase your stack size allocation using

// the command:

// ulimit -s 20000

// That is 20 megabytes (which should be enough)

// More on this in class and in a later lab we'll se how to use

// dynamic memory allocation to do this.

int nF = 0; // # of flights in database

if (argc != 2) {

printf("Cannot proceed: Missing file name.\n");

exit(EXIT\_FAILURE);

}

readData(argv[1], flights, &nF);

// Database is read...now just show some flights

showFlight(flights[0]); // First flight

showFlight(flights[rand()%nF]); // Some random flight

showFlight(flights[nF-1]); // Last flight

showFlightDetails(flights[36367]); //show details of flight[36367]

printf("Avg. departure delay at PHL: %d minutes.\n", avgDepDelay(flights, nF));

printf("Avg. arrival delay at PHL: %d minutes.\n", avgArrDelay(flights, nF));

Date d = makeDate(4,1,2008);

printf("On %s there were %d flights that departed from PHL.\n", toString(d), dayDepFlights(flights, nF, d));

printf("On %s there were %d flights that arrived in PHL.\n", toString(d), dayArrFlights(flights, nF, d));

return(0);

} // main()

void readData(char \*file, Flight list[], int \*n) {

FILE \*fp;

int i = 0;

// Open the datafile

printf("Opening file: %s\n", file);

fp = fopen(file, "r");

if (fp == NULL) {

printf("Unable to open file %s.\n", file);

exit(EXIT\_FAILURE);

}

printf("Opened file %s.\n", file);

printf("Reading...");

// Read from data file, one line at a time and parse each line

char line[180];

fgets(line, sizeof(line), fp); // ignore first line

while ( (fgets(line, sizeof(line), fp)) != NULL) {

int l = strlen(line);

line[l-1] = '\0';

list[i] = parseFlight(line);

i++;

}

\*n = i;

fclose(fp);

printf("...done. [%d flights]\n", i);

} // readData()

// File: date.c

// Purpose: Defines a basic Date ADT functions (see date.h)

// Written By: Deepak Kumar

//Modified By: Yu Sheng

// Date: March 20, 2018

#include <stdio.h>

#include <stdlib.h>

#include "date.h"

Date makeDate(int m, int d, int y) { // Constructor

Date date = {m, d, y};

return date;

} // makeDate()

void showDate(Date d) { // print function

printf("%d/%d/%d", d.month, d.day, d.year);

} // showDate()

void showDate2(Date d) { // print function

printf("%d/%d/%d", d.year, d.month, d.day);

} // showDate()

char \*toString(Date d) { // Convert a date to string "mm/dd/yyyy"

char \*result;

// A date requires 11 chars: "mm/dd/yyyy" (including 1 for '\0')

// The command below does dynamic memory allocation. Why?

// We will discuss this in class this week!

result = (char \*)(malloc(11));

sprintf(result, "%d/%d/%d", d.month, d.day, d.year);

return result;

} // toString()

// File: date.h

// Purpose: Defines a basic Date ADT: month, day, year

// Written By: Deepak Kumar

// Date: March 20, 2018

typedef struct {

int month; // std date fields

int day;

int year;

} Date;

Date makeDate(int m, int d, int y); // Constructor

void showDate(Date d); // Print function

void showDate2(Date d); // Print function

char \*toString(Date d); // returns date string

// File: flight.c

// Purpose: Defines a flight using the following information:

// date (m/d/y), carrier, number, origin, dest, distance,

// arrTime, depTime, arrDelay, depDelay, cancelled

// Implements the functions defined in flight.h

// Written By: Deepak Kumar

//Modified By: Yu Sheng

// Date: March 20, 2018

#include <stdio.h>

#include <string.h>

#include <stdlib.h>

#include "flight.h"

Flight parseFlight(char \* str){ // Parses line and returns a Flight

Flight f;

Date d; // Parse the date

char \*token;

token = strtok(str, ",");

d.year = atoi(token);

token = strtok(NULL, ",");

d.month = atoi(token);

token = strtok(NULL, ",");

d.day = atoi(token);

f.date = d; // Date is parsed

token = strtok(NULL, ","); // depTime

f.depTime = atoi(token);

token = strtok(NULL, ","); // arrTime

f.arrTime = atoi(token);

token = strtok(NULL, ",");

strcpy(f.carrier, token); // carrier

token = strtok(NULL, ",");

strcpy(f.number, token); // (flight) number

token = strtok(NULL, ",");

f.arrDelay = atoi(token); // arrDelay

token = strtok(NULL, ",");

f.depDelay = atoi(token); // depDelay

token = strtok(NULL, ",");

strcpy(f.origin, token); // origin

token = strtok(NULL, ",");

strcpy(f.dest, token); // dest

token = strtok(NULL, ",");

f.distance = atoi(token); // distance (miles)

token = strtok(NULL, ",");

f.cancelled = atoi(token); // cancelled

return f; // Done!!

} // parseFlight()

void showFlight(Flight f) { // Display some flight info

showDate(f.date);

printf(" %s %s %s to %s\n",f.carrier,f.number,f.origin,f.dest);

} // showFlight()

int arrDelay(Flight f) { // Accessor example

return f.arrDelay;

} // arrDelay()

int depDelay(Flight f) { // Accessor example

return f.depDelay;

} // depDelay()

int cancelled(Flight f) { // Accessor example

return f.cancelled;

} // cancelled()

int showmin(int origin,int delay){

int h,m,result;

h=delay/60;

result=h\*100+origin/100\*100;

m=delay%60+origin%100;

if(m>=60){

result=result+100;

m=m%60;

}else if(m<0){

m=m+60;

result=result-100;

}

result=result+m;

if(result>=0)

result=result%2400;

if(result<0)

result=result+2400;

return result;

}

void showFlightDetails(Flight f) {

showDate2(f.date);

printf(" %s %s %s to %s %d miles\n",f.carrier,f.number,f.origin,f.dest,f.distance);

int ddelay,adelay;

ddelay=showmin(f.depTime,f.depDelay);

adelay=showmin(f.arrTime,f.arrDelay);

printf("Scheduled: Dep %d Arr %d\n",f.depTime,f.arrTime);

printf("Actual : Dep %d Arr %d\n",ddelay,adelay);

}

int avgDepDelay(Flight f[], int n){

int result=0;

int count=0;

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

if((!f[i].cancelled) &&(strcmp(f[i].origin,"PHL")==0)){

count++;

if(f[i].depDelay>0)

result+=f[i].depDelay;

}

}

result=result/count;

return result;

}

int avgArrDelay(Flight f[], int n){

int result=0;

int count=0;

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

if((!f[i].cancelled)&&(strcmp(f[i].dest,"PHL")==0)){

count++;

if(f[i].arrDelay>0)

result+=f[i].arrDelay;

}

}

result=result/count;

return result;

}

int dayDepFlights(Flight f[], int n, Date d){

int count=0;

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

if((!f[i].cancelled)&&(strcmp(toString(f[i].date),toString(d))==0)&&(strcmp(f[i].origin,"PHL")==0))

count++;

}

return count;

}

int dayArrFlights(Flight f[], int n, Date d){

int count=0;

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

if((!f[i].cancelled)&&(strcmp(toString(f[i].date),toString(d))==0)&&(strcmp(f[i].dest,"PHL")==0))

count++;

}

return count;

}

// File: flight.h

// Purpose: Defines a flight using the following information:

// date (m/d/y), carrier, number, origin, dest, distance,

// arrTime, depTime, arrDelay, depDelay, cancelled

// Written By: Deepak Kumar

// Date: March 20, 2018

#include "date.h"

#define MAX\_CHARS 5

typedef struct {

Date date; // Consult your assignment handout for details

char carrier[MAX\_CHARS];

char number[MAX\_CHARS];

char origin[MAX\_CHARS];

char dest[MAX\_CHARS];

int distance;

int depTime;

int arrTime;

int depDelay;

int arrDelay;

int cancelled;

} Flight;

Flight parseFlight(char \* line);

void showFlight(Flight f);

int arrDelay(Flight f);

int depDelay(Flight f);

int cancelled(Flight f);

int showmin(int origin,int delay); //show the adjusted time of origin according to the delay time

void showFlightDetails(Flight f); //show details of the fligt including actual departure and arrival time

int avgDepDelay(Flight f[], int n); //average delay time for flights departure dfrom PHL

int avgArrDelay(Flight f[], int n); //average delay time for flights arrived at PHL

int dayDepFlights(Flight f[], int n, Date d); //number of flights departured from PHL on a day d

int dayArrFlights(Flight f[], int n, Date d); // //number of flights arrived at PHL on a day d

------------------------------------------------Output------------------------------------------------

[jsheng@rukbat a7]$ ./a7 ~dkumar/CMSC246/A7/2008PHL.csv

Opening file: /home/dkumar/CMSC246/A7/2008PHL.csv

Opened file /home/dkumar/CMSC246/A7/2008PHL.csv.

Reading......done. [196431 flights]

1/3/2008 WN 23 JAX to PHL

10/18/2008 UA 366 DEN to PHL

12/13/2008 DL 1187 PHL to SLC

2008/3/1 US 1592 ATL to PHL 665 miles

Scheduled: Dep 631 Arr 822

Actual : Dep 627 Arr 804

Avg. departure delay at PHL: 15 minutes.

Avg. arrival delay at PHL: 16 minutes.

On 4/1/2008 there were 277 flights that departed from PHL.

On 4/1/2008 there were 282 flights that arrived in PHL.

-------------------------------------------------------Reflection---------------------------------------------------------

In this assignment, we are required to write several functions like showFlightDetails(Flight f) and avgDepDelay(Flight f[], int n). I find there are many traps in writing those functions. For example, when I was writing showFlightDetails(Flight f) functions, we needed to calculate the actual departure and arrival time by ourselves. When doing this, we need to tranfer the delay time to hours and minitues, and add them seperatly to the hour and minitues of actual time. But sometimes addition will cause minitue digits to be more than 60 or hours to be more than 24, thus we need to mod the minitues by 60 and mod the hours by 2400. However, since some of the delay time is negative, in other words, some flights departured/arrived earlier than scheduled, we actually need to consider an additional situation, for example, the scheduled departure time is 0010 but it departure earlier for 20 minitues, then the actual time becomes 2350, which is 0010-20+2400. Thus in this rare case, we need to add 2400 to the actual time.

Also in avgDepDelay(Flight f[], int n) function, we need to exculde the situation of flight cancelled. And we need to count all flights departured from PHL in the total number of flights, but we should only add up the delay time of those flights who have positive delay time to the total delay time. And divide two to calculated the average departure delay time.