|  |
| --- |
| [Type the company name] |
| Program 7: Airport Ticker |
| No Shipping Cost! |
|  |
| **Brett Kriz** |
| **4/8/2014** |

|  |
| --- |
| [Type the abstract of the document here. The abstract is typically a short summary of the contents of the document. Type the abstract of the document here. The abstract is typically a short summary of the contents of the document.] |

// Brett Kriz

// Program 7: Airport Ticker

//

// Includes, just incase

#include <algorithm> // Algorithms >.>

#include <cmath> // Basic Math Operations

#include <cstdlib> // control of stdlib

#include <cstring> // Cstrings

#include <ctime> // Time stuff

#include <ctype.h> // typeing

#include <fstream> // File stream (I/O)

#include <iomanip> // Manipulation of I/O

#include <iterator>

#include <iostream> // Input\Output

#include <sstream> // String stream

#include <stdlib.h> // Standard Library

#include <string> // String Stuff

#include <string.h> // String Stuff

#include <time.h> // More time stuff

//#define \_CRT\_SECURE\_NO\_WARNINGS

using namespace std;

// Standard...

// Early Proto

//void file\_read();

string INTtoSTR(int a);

void sleep(unsigned int mseconds);

// Variables

const int ARRAY\_SIZE = 250;

bool I2B(int a);

struct Port{

char code[4]; // 3 char + TERM

string full;

string city;

};

// PortList\* ports = new PortList;

struct PortList{

int size;

Port ports[ARRAY\_SIZE];

};

class Plane{

private:

string date;

string code;

string dist;

string fl; // Needs basic processing

string ID;

//string Location;

public:

// Gets and sets

string getdate();

string getcode();

string getdist();

string getfl();

string getID();

void setdist(string s);

void setcode(string s);

void setfl(string s);

void setID(string s);

void setdate(string s);

// funcs

Plane();

Plane (string Date, string Code, string Dist, string Fl, string id);

string getTime();

string getdateprint(); // for format

void print(PortList l);

string getLoc(PortList l);

string getFL();

string getAirport(PortList l);

};

Plane::Plane(){

date = "";

code = "";

dist = "";

fl = "";

ID = "";

}

Plane::Plane (string Date, string Code, string Dist, string Fl, string id){

date = Date;

code = Code;

dist = Dist;

fl = Fl;

ID = id;

}

// Gets & Sets

string Plane::getdate() { return date; }

string Plane::getcode() { return code; }

string Plane::getdist() { return dist; }

string Plane::getfl() { return fl; }

string Plane::getID() {

string ans = ID;

int x=0;

while(x<ans.size()){

if (!isalpha( ans[x] )) {

ans = ans.insert(x," ");// all for a space

return ans;

}

x++;

}

return ans;

}

void Plane::setdate(string s) { date = s; }

void Plane::setcode(string s) { code = s; }

void Plane::setdist(string s) { dist = s; }

void Plane::setfl(string s) { fl = s; }

void Plane::setID(string s) { ID = s; }

void Plane::print(PortList l){

cout << "\n\tFlight: " << getID()

<< "\n\tDate: " << getdateprint()

<< "\n\tTime: " << getTime()

<< "\n\tLocation: " << getLoc(l)

<< "\n\tAltitude: " << getFL()

<< endl;

}

string Plane::getLoc(PortList l){

string ans, dir, addin = "", airport = "";

int x = 0, size = dist.size();

dir = dist.substr(size-2, size);

airport = getAirport(l);

// Evaluate for bools

bool s = (dist.find("S") != dist.npos), w = (dist.find("W") != dist.npos), e = (dist.find("E") != dist.npos), n = I2B(dist.find("N") != dist.npos);

// Concat the str

if (s)

addin += "South";

if (n)

addin += "North";

if (w)

addin += "West";

if (e)

addin += "East";

if (!(n||e||s||w)) {cout << "\n! NO DIRECTION !\n";}

// in original, replace the end 2, with this

ans = dist.replace(size-2,size, " miles " + addin + " of " + airport );

return ans;

}

string Plane::getFL(){

string arg = fl, comma = "";

int l = -1;

arg = arg.substr(2);

l = atoi(arg.c\_str())\*100;

if (l > 1000) {

comma = ",";

}

arg = INTtoSTR(l);

arg = arg.insert(arg.size()-3, comma) + " feet";

return arg;

}

string Plane::getdateprint(){

string ans = "";

unsigned int x = 0;

ans = date.substr(0,date.find\_last\_of("-"));

while(x < ans.size()){

if (ans[x] == '-'){

ans = ans.replace(x,1,"/");

}

x++;

}

return ans;

}

// noitice the scope resolution operator

string Plane::getTime(){

string ans = "", addin = "AM";

string madd = "";

int pos, Temp;

pos = date.find\_last\_of("-");

ans = date.substr(pos+1); // @@@GETTIME

Temp = atoi(ans.c\_str());

int hh,mm, mc; // the ints of hour and minute

hh = int(Temp/100); // just add algebra and trunk.

mm = Temp- (hh\*100) ;

// 60/60 = 1

mc = int(mm/60); // either 1 or 0

hh += mc;

mm = abs(mc-1)\*mm; // either 60 or 0

if ( mm < 10){

madd = "0";

}

if ( hh == 0 ){ // Fix for midnight flights

hh = 12;

cout << endl;

}

int h = (hh\*100)+mm; // corrected for errors

if ( h >= 1200 ){

addin = "PM"; // AT noon (1200 hours) it becomes PM.

if ( h >= 1300 ){ // check for 1pm before subtracting 12

hh -= 12;

}

}

string k = INTtoSTR(hh), j = INTtoSTR(mm);

ans = "";

ans += k + ":" + j + madd + " " + addin;

// Done

return ans;

}

string Plane::getAirport(PortList l){

int x=0, size = l.size ;

string ans = "Unknown Port";

//char pc[] = code.c\_str();

//look for them

while(x<size){

int var = code.compare( string(l.ports[x].code) );

bool y = var == 0;

if (y){

return l.ports[x].full;

}

x++;

}// endloop

return ans;

}

struct List{

int size;

PortList ports;

Plane plane[ARRAY\_SIZE];

};

//Other PROTOS

void setup();

void DisplayList(List data, PortList l);

bool Read\_IN(List& data, string des = "flights.txt"); // Read in

bool Read\_IN(PortList& data, string des = "miairports.txt"); // Read in

string token\_burn(string& line, string den, int lookat = 0);

/\*--------------------------------------------------------

Function: main()

Disc: What? You know...

--------------------------------------------------------\*/

int main(){

// Main

setup(); // Yup

// Create Main Variables

ifstream IN;

int x = 0;

string des;

List data;

PortList ports;// = new PortList;

data.size = 0;

// start functions

bool go = true;

go = Read\_IN(ports);

if (!go) {system("Pause"); return 404;} // Make sure were good to go ;D

data.ports = ports;

go = Read\_IN(data); // LOAD IN flights.txt and miairports.txt

if (!go) {system("Pause"); return 404;} // Make sure were good to go ;D

// Shipping cost???

// Customer Name??

DisplayList(data, ports);

system("Pause");

return 0;

}

/\*--------------------------------------------------------

Function: Setup

Disc: Setup the window quick

See 'onStart'

--------------------------------------------------------\*/

void setup(){

// Setup

system ("Title Program 7 - Airport Ticker");

system ("Color 0A");

system("cls");

}

/\*--------------------------------------------------------

Function: Sleep

Disc: Sleeps for x many seconds

--------------------------------------------------------\*/

void sleep(unsigned int mseconds){

// Sleep...

clock\_t goal = mseconds + clock();

while (goal > clock());

}

void DisplayList(List data, PortList l){ // the d controlls the sleep

// NO SORTING HERE!

int size = data.size;

int i = 0;

while (i<size){// note data.size not size

Plane z = data.plane[i];

cout << "\n #" << i;

z.print(l);

i++;

}

cout << endl;

}

string INTtoSTR(int a){

char buff [33] = "";

\_itoa\_s(a, buff, 10); // have to return the buffer

// Look into triming

return buff;

}

bool I2B(int a){

bool z = false;

if (a > 0)

z = true;

return z;

}

bool Read\_IN(PortList& data, string des){

system("title Program 7: Read in Port List");

ifstream IN;

int x = 0;

bool good = true;

IN.open(des); // Modularity >:3

//cout << endl;

if (IN.is\_open()){

system("color 0F");

while(!IN.eof()){

Port z;

string s = ",", line = "", den = " ";

int t3s = 47;

char rawline[256];

IN >> z.code; // Snag the first one

IN.getline(rawline, 256 ); // snag the line in STR

line = string(rawline);

line = line.substr(4);// 4 to end is start of token 2 & 3

string fp = line.substr( 0, t3s - 1);// grabbing token 2 substr

// Find the trimed length/pos

int c = fp.length()-1;

bool a = (c>1 && (fp[c] == ' ' || fp[c] == '\t' || fp[c] == '\0')); // Check terminator just incase

// Look for not space char, from the end for trim

do{

a = (c>1 && (fp[c-1] == ' ' || fp[c-1] == '\t') && a);

if (a) {c--;} // go down if a, arive at the prev. checked value

}while(a); // check the bool var a;

z.full = fp.substr( 0, c );

z.city = line.substr( t3s );

// Checks

//cout << fp <<s<< c <<"::\n"<< z.full << "::\n" << z.city<< "\n"; // Debug space

cout <<"#"<< x <<s<< z.code <<s<< z.full <<s<< z.city << "\n";

data.ports[x] = z;

x++; // yup increment

//system("pause");

sleep(10);

} // end loop

cout << "\n Airport read in complete!\n\n";

data.size = x; // Store size

//system("pause");

sleep(950);

setup();

// not open?

}else{

cout << "\n\nInput File is empty!\n\t Check path!\a\n";

good = false;

}// end if

IN.close();// Better close that...

return good;

}// end func

// FOR LIST NOW

bool Read\_IN(List& data, string des){

ifstream IN;

int x = 0;

bool good = true;

system("title Program 7: Read in Aircraft List");

IN.open(des); // Modularity >:3

//cout << endl;

if (IN.is\_open()){

system("color 0F");

while(!IN.eof()){

Plane z;

string s = ",", line = "", den = ",";

char rawline[256];

IN.getline(rawline, 256 ); // snag all

line = string(rawline);

// Token\_Burn was such a good idea

z.setdate( token\_burn(line,den,15) );

z.setcode( token\_burn(line,den,2) );

z.setdist( token\_burn(line,den,1) );

z.setfl( token\_burn(line,den,4) );

z.setID( line ); // should be chewed down that far

cout << "\n RAW #" << x << " ~~~~~~~~~~~~~~~~~~~~"

<< "\n Date: " << z.getdate() << "<"

<< "\n Code: " << z.getcode() << "<"

<< "\n Dist: " << z.getdist() << "<"

<< "\n FL: " << z.getfl() << "<"

<< "\n ID: " << z.getID() << "<"

<< endl;

// Save it!

data.plane[x] = z;

x++; // yup increment

sleep(0050);

} // end loop

cout << "\n Airplane read in complete!\n\n";

data.size = x; // Store size

//system("pause");

sleep(850);

setup();

//size = x;

// not open?

}else{

cout << "\n\nInput File is empty!\n\t Check path!\a\n";

good = false;

}// end if

IN.close();// Better close that...

return good;

}// end func

string token\_burn(string& line, string den, int lookat){

int pos = line.find( den, lookat ), x = 0;

string z = "";

while(x < pos){z += " "; x++;}

z += "^";// HERE

//cout << line << "\n" << z << "\n";

string token = line.substr( 0 , pos );

line = line.substr( pos + 1 ); // till end

//cout << line << "\n\"" << token << "\""

//<< endl;

sleep(20);

return token;

}

Output:

