CSC 245 - Lab 2

# names.cpp

#include <string>

#include <iostream>

using namespace std;

void BreakDown(string name, string& first, string& last, string& mi);

int main()

{

string name, first, last, mi;

cout << "Name? <Last, First MI.> ";

getline(cin, name);

BreakDown(name, first, mi, last);

cout << "First Name Entered : " << first << endl;

cout << "Last Name Entered : " << last << endl;

cout << "Middle Initial Entered : " << mi << endl;

return 0;

}

void BreakDown(string name, string& first, string& mi, string& last)

{

// pre : name is initialized with a full name

// post : first, mi, and last contain the individual components

// of that name

int commaPosition = name.find(',');

last = name.substr(0, commaPosition);

mi = name.substr(name.find('.')-1, 1);

first = name.substr(commaPosition + 2, name.length() - commaPosition - 4);

}

# myStruct.cpp

#include <iostream>

#include <string>

using namespace std;

const int SIZE = 100;

struct DateType

{

string month;

int day;

int year;

};

// Insert EventType definition HERE

struct EventType{

string place;

DateType date;

};

void PrintStruct(DateType Holiday);

void AssignMonth(EventType Dates[]);

int main()

{

DateType Holiday = { "September", 5, 2018 };

EventType Dates[SIZE];

PrintStruct(Holiday);

AssignMonth(Dates);

return 0;

}

void PrintStruct(DateType Holiday)

{

// pre : Holiday is initialized with three field values

// post : Function prints date of Holiday in form mm dd, yy

cout << Holiday.month + " " << Holiday.day << ", " << Holiday.year << endl;

}

void AssignMonth(EventType Dates[])

{

// pre : none

// post : The month field of every date record in array is assigned a month

// an enhanced for loop could also be used, if the size of the array wasn't a constant (SIZE)

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

Dates[i].date.month = "January";

};

}

# testTime.cpp (Client Program)

#include <iostream>

#include "time.h"

using namespace std;

int main() {

// 9:30 AM

Time myTime(9, 30, 0);

myTime.Write();

myTime.WriteAmPm();

// 8:00 AM and Breakfast time

myTime.Set(8, 0, 0);

myTime.WriteAmPm();

myTime.Mealtime();

// array of 10 Time objects

Time Schedules[10];

// sizeof is used to determine the length of the array

// the size of the whole array divided by the size of one

// object in the array gives the number of objects in the array

for (int i = 0; i < sizeof(Schedules)/sizeof(Schedules[0]); i++) {

Schedules[i].Set(11, 0, 0);

Schedules[i].WriteAmPm();

}

//Explain number of destructor calls below

//there are 11 destructor calls because the myTime variable is destructed

//and the schedules array of 10 time objects will all also be destructed

//after main is done executing

return 0;

}

# Time.cpp (Implementation File)

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// IMPLEMENTATION FILE (time.cpp)

// This file implements the Time member functions

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

#include <iostream>

using namespace std;

#include "time.h"

// Private members of class:

// int hrs;

// int mins;

// int secs;

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Time::Time( /\* in \*/ int initHrs,

/\* in \*/ int initMins,

/\* in \*/ int initSecs)

// Constructor

// Precondition:

// 0 <= initHrs <= 23 && 0 <= initMins <= 59

// && 0 <= initSecs <= 59

// Postcondition:

// hrs == initHrs && mins == initMins && secs == initSecs

{

hrs = initHrs;

mins = initMins;

secs = initSecs;

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Time::Time()

// Default constructor

// Postcondition:

// hrs == 0 && mins == 0 && secs == 0

{

hrs = 0;

mins = 0;

secs = 0;

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

void Time::Set( /\* in \*/ int hours,

/\* in \*/ int minutes,

/\* in \*/ int seconds)

// Precondition:

// 0 <= hours <= 23 && 0 <= minutes <= 59

// && 0 <= seconds <= 59

// Postcondition:

// hrs == hours && mins == minutes && secs == seconds

{

hrs = hours;

mins = minutes;

secs = seconds;

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

void Time::Increment()

// Postcondition:

// Time has been advanced by one second, with

// 23:59:59 wrapping around to 0:0:0

{

secs++;

if (secs > 59)

{

secs = 0;

mins++;

if (mins > 59)

{

mins = 0;

hrs++;

if (hrs > 23)

hrs = 0;

}

}

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

void Time::Write() const

// Postcondition:

// Time has been output in the form HH:MM:SS

{

if (hrs < 10)

cout << '0';

cout << hrs << ':';

if (mins < 10)

cout << '0';

cout << mins << ':';

if (secs < 10)

cout << '0';

cout << secs << endl;

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

bool Time::Equal( /\* in \*/ Time otherTime) const

// Postcondition:

// Function value == TRUE, if this time equals otherTime

// == FALSE, otherwise

{

return (hrs == otherTime.hrs && mins == otherTime.mins &&

secs == otherTime.secs);

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

bool Time::LessThan( /\* in \*/ Time otherTime) const

// Precondition:

// This time and otherTime represent times in the

// same day

// Postcondition:

// Function value == TRUE, if this time is earlier

// in the day than otherTime

// == FALSE, otherwise

{

return (hrs < otherTime.hrs ||

hrs == otherTime.hrs && mins < otherTime.mins ||

hrs == otherTime.hrs && mins == otherTime.mins

&& secs < otherTime.secs);

}

void Time::WriteAmPm() const

{

bool am;

int tempHrs;

am = (hrs <= 11);

if (hrs == 0)

tempHrs = 12;

else if (hrs >= 13)

tempHrs = hrs - 12;

else

tempHrs = hrs;

if (tempHrs < 10)

cout << '0';

cout << tempHrs << ':';

if (mins < 10)

cout << '0';

cout << mins << ':';

if (secs < 10)

cout << '0';

cout << secs;

if (am)

cout << " AM";

else

cout << " PM";

cout << endl;

}

Time::~Time()

{

cout << "Destructor Called" << endl;

}

void Time::Mealtime() const

{

if (mins == 0 & secs == 0) {

if (hrs == 8)

cout << "Breakfast" << endl;

if (hrs == 12)

cout << "Lunch" << endl;

if (hrs == 19)

cout << "Dinner" << endl;

}

}