//PROBLEM1\_FUNCTIONS.cpp

//Andrew Ribeiro

//October 7, 2008

//To be used in: PROBLEM1.cpp

/\*File Discription & Contents

>This file containes the functions used in the file PROBLEM1.cpp

##############################Contents/Functions######################

--------------------------Functions to be called in PROBLEM1.cpp---------------------------------

1.) double safeInput(string errorMessage,string prompt,double minValue)

+This function is used to safely input a number of type double. Which means, this functions validates the input.

Also, if the user does not enter in a correct value, the function prompts the user to enter a correct value

three times. If the user fails to do so by the third time, the function returns(-1), indicating that the

program should end.

+Arguments

-string errorMessage: this string is used to display an error message.

-string prompt: this string is used to prompt the user for a valid value.

-double minValue: is used to make shure the inputed value is greater than this minimum value.

+Return type

-double: This function returns a valid inputed value of type double.

2.)void sub\_printCalendar(int firstDayOfYear,int userSelectedMonthNumber,int year)

+This function is used to print the calendar page given the data collected from the user.

+Arguments

-int firstDayOfYear: the first day of the year given by the user.

-int userSelectedMonthNumber: the month that the user wants to see a callendar page of.

-int year: the year that the user chose.

--------------------------Helper functions to be called by the functions above-----------------------------

3.) bool isLeapYear(int year)

+This function is called by the subroutine- subPrintCalendar()- in order to determine if a year

is a leap year.

+Arguments

-int year: the year the user chose to display a callendar page from.

+Return type

-bool: if the function is a leap year, the function returns true-1.

Else, the function returns false-0.

4.)int lastDayOfMonth(int daysInMonth,int startDay)

+This function is called by the subroutine - subPrintCalendar()- in order to find

the last day of the previous month, so that the begining day of the next month

can be determined.

+Arguments

-int daysInMonth: how many days are in the month that we are trying to find the

last day of.

-int startDay: the first day of the month.

+Return type

-int: returns the last day of the month.

5.)string stringOfMonthName(int monthNumber)

+This function returns the name of a month when given the month number.

+Arguments

-int monthNumber: the number of the month-gotten from the user. Ex Dec = 12.

+Return type

-string: returns a string containing the months name.

End File Discription & Contents\*/

#include <iostream>

#include <iomanip>

#include <string>

using namespace std;

//Helper function prototypes.

string stringOfMonthName(int monthNumber);

int lastDayOfMonth(int daysInMonth,int startDay);

bool isLeapYear(int year);

/\*Global Constants

>Days in the months that never change.

>Note: oct and dec are objects. So, they are named with 4 digits instead of three \*/

const int jan = 31, mar =31 , apr =30 , may=31, jun =30, jul =31 , aug =31 , sep =30,octo =31, nov =30 ,dece =31;

//#########Helper functions. Are not to be called in the main program.####

bool isLeapYear(int year)

{

/\* A leap year is one whose number is exactly divisible by four. Century years,

however, are only leap years if they are exactly divisible by 400. Hence 1900 was not a leap

year but 2000 was. Your program should not use arrays. \*/

if (year % 4 == 0)

{

if (year % 100 == 0) //If it is a century year.

{

if(year % 400 == 0)

{

return true;

}

else

{

return false;

}

}

else //If not a century year.

{

return true;

}

}

else

{

return false;

}

}

int lastDayOfMonth(int daysInMonth,int startDay)

{

/\*Add how many spaces are in front of the first day to the number of days in the month, then take the modulo of that number and subtract it from 7. This number is the number of spaces or days in that week left.\*/

//This formula was derived from observation.

int daysLeftInFirstWeek,lastDayOfMonth;

daysLeftInFirstWeek = 7 - startDay;

daysInMonth -= daysLeftInFirstWeek;

lastDayOfMonth = daysInMonth % 7;

return lastDayOfMonth;

}

string stringOfMonthName(int monthNumber)

{

//Arguments should be validated prior to usage in this function.

switch(monthNumber)

{

case 1:

return "January";

break;

case 2:

return "February";

break;

case 3:

return "March";

break;

case 4:

return "April";

break;

case 5:

return "May";

break;

case 6:

return "June";

break;

case 7:

return "July";

break;

case 8:

return "August";

break;

case 9:

return "September";

break;

case 10:

return "October";

break;

case 11:

return "November";

break;

case 12:

return "December";

break;

}

}

//#########Functions that are to be called in the main program. #########

int safeInput(string errorMessage,string prompt,int minValue,int maxValue)

{

//If maxValue is less than min value, ignore max value. Else..... add it to the checker.

int userIn = 0,errorCounter = 0;

cin>>userIn;

if(maxValue <= minValue)

{

//Loop if there is errors. Be aware of the side effects.

while(!cin.good() || userIn < minValue)

{

if (errorCounter == 2)

{

userIn = -1; //Return fail value.

break;

}

cout<<errorMessage<<endl

<<prompt;

cin.clear();

cin.ignore(80,'\n');

cin>>userIn;

errorCounter++;

}

}

else if( maxValue == minValue) // Ignore both min and max

{

while(!cin.good() )

{

if (errorCounter == 2)

{

userIn = -1; //Return fail value.

break;

}

cout<<endl<<errorMessage<<endl

<<prompt;

cin.clear();

cin.ignore(80,'\n');

cin>>userIn;

errorCounter++;

}

}

else

{

while(!cin.good() || userIn < minValue || userIn > maxValue )

{

if (errorCounter == 2)

{

userIn = -1; //Return fail value.

break;

}

cout<<errorMessage<<endl

<<prompt;

cin.clear();

cin.ignore(80,'\n');

cin>>userIn;

errorCounter++;

}

}

return userIn;

}

void sub\_printCalendar(int firstDayOfYear,int userSelectedMonthNumber,int year)

{

//Defined lastDayOfPreviousMonth and daysInUserMonth in case the user selected the first month.

int lastDayOfPreviousMonth = firstDayOfYear,feb,daysInUserSelectedMonth = jan;

bool leapYear = isLeapYear(year);

if(leapYear) //If it is a leap year

{

feb = 29;

}

else //If not a leap year.

{

feb = 28;

}

//Find out the starting day of the month the user selected.Also, find out how many days are in the\

month the user selected and assign it to daysInUserMonth.

for(int month = 1; month < userSelectedMonthNumber; month++)

{

//Finds the day of the next month, incrimentaly.

switch(month)

{

case 1:

lastDayOfPreviousMonth = lastDayOfMonth(jan,firstDayOfYear);

daysInUserSelectedMonth = feb; //Assigning the number of days in the user selected month.

break;

case 2:

lastDayOfPreviousMonth = lastDayOfMonth(feb,lastDayOfPreviousMonth);

/\* Next month is march, therefore if the loop breaks, the variable will be tuned to the user selected month.

Also, mar is a global constant.\*/

daysInUserSelectedMonth = mar;

break;

case 3:

lastDayOfPreviousMonth = lastDayOfMonth(mar,lastDayOfPreviousMonth);

daysInUserSelectedMonth = apr;

break;

case 4:

lastDayOfPreviousMonth = lastDayOfMonth(apr,lastDayOfPreviousMonth);

daysInUserSelectedMonth = may;

break;

case 5:

lastDayOfPreviousMonth = lastDayOfMonth(may,lastDayOfPreviousMonth);

daysInUserSelectedMonth = jun;

break;

case 6:

lastDayOfPreviousMonth = lastDayOfMonth(jun,lastDayOfPreviousMonth);

daysInUserSelectedMonth = jul;

break;

case 7:

lastDayOfPreviousMonth = lastDayOfMonth(jul,lastDayOfPreviousMonth);

daysInUserSelectedMonth = aug;

break;

case 8:

lastDayOfPreviousMonth = lastDayOfMonth(aug,lastDayOfPreviousMonth);

daysInUserSelectedMonth = sep;

break;

case 9:

lastDayOfPreviousMonth = lastDayOfMonth(sep,lastDayOfPreviousMonth);

daysInUserSelectedMonth = octo;

break;

case 10:

lastDayOfPreviousMonth = lastDayOfMonth(octo,lastDayOfPreviousMonth);

daysInUserSelectedMonth = nov;

break;

case 11:

lastDayOfPreviousMonth = lastDayOfMonth(nov,lastDayOfPreviousMonth);

daysInUserSelectedMonth = dece;

break;

}

} //Exit when the begining day of the user selected month is found.

/\*############ Print out the calendar ############ \*/

//Print out calendar header.

cout<<endl<<setw(23)<<stringOfMonthName(userSelectedMonthNumber)<<" "<<year<<endl

<<" \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_"<<endl

<<" Sun Mon Tus Wed Thu Fri Sat"<<endl<<setw(0); //3 spaces in between each.

/\*The ignoreDays variable is added in order to avoid more than one loop. This variable allows for the blank days to be ignored \*/

int ignoreDays = 0;

for(int day = 1 ; day <= daysInUserSelectedMonth; day++)

{

//If the month starts on sunday, do not print blanks for the first week.

if (lastDayOfPreviousMonth == 7 && day <= 7)

{

cout<<setw(6)<<(day - ignoreDays)<<setw(6);

//If the frst week is over, make a new line and make the feild width 0.

if (day % 7 == 0)

{

cout<<endl<<setw(0);

}

}

//If the month does not start on sunday, print blanks until the first day of the month.

else if(day <= lastDayOfPreviousMonth)

{

cout<<setw(6)<<" ";

ignoreDays++;

daysInUserSelectedMonth++;

}

//Start printing out the days for the week.

else

{

if (day % 7 == 0)

{

cout<<setw(6)<<(day-ignoreDays)<<endl; //End of the line

}

else

{

cout<<setw(6)<<(day - ignoreDays)<<setw(6);

}

}

}

cout<<endl<<" \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_"<<endl<<endl;

}