C++ Object Oriented Programming Project

Digital Calender Project

Code:

```
#include <iostream>
#include <string>
#include <ctime>
#include <fstream>
#include <vector>
using namespace std;
class Calendar
{
        private:
                static int dayNumber(int day, int month, int year);
                static string getMonthName(int monthNumber);
                static int numberOfDays(int monthNumber, int year);
                static bool isLeapYear(int year);
                static void printTask(const string& task);

        public:
                static void printCalendar(int year, int month = -1);
                static void printCurrentDate();
                static int calculateAge(int birthYear);
                static bool checkLeapYear(int year);
                static void addTask();
                static void displayTasks();
                static void editTask();
                static void setReminder();
                static void removeTask();
                static void searchTasksByDate();
                static string tasks[100];
```

```
};

string Calendar::tasks[100];

int Calendar::dayNumber(int day, int month, int year)
{
    static int t[] = {0, 3, 2, 5, 0, 3, 5, 1, 4, 6, 2, 4};
    year -= month < 3;
    return (year + year / 4 - year / 100 + year / 400 + t[month - 1] + day) % 7;
}

string Calendar::getMonthName(int monthNumber)
{
    string month;
    switch (monthNumber)
            {
                case 0:
                    month = "January";
                    break;
                case 1:
                    month = "February";
                    break;
                case 2:
                    month = "March";
                    break;
                case 3:
                    month = "April";
                    break;
                case 4:
                    month = "May";
                    break;
                case 5:
                    month = "June";
                    break;
                case 6:
                    month = "July";
                    break;
                case 7:
                    month = "August";
                    break;
                case 8:
                    month = "September";
                    break;
                case 9:
                    month = "October";
                    break;
                case 10:
                    month = "November";
                    break;
                case 11:
                    month = "December";
                    break;
    }
    return month;
}


int Calendar::numberOfDays(int monthNumber, int year)
{
    if (monthNumber == 0 || monthNumber == 2 || monthNumber == 4 || monthNumber == 6 ||
monthNumber == 7 || monthNumber == 9 || monthNumber == 11)
        return 31;
    else if (monthNumber == 3 || monthNumber == 5 || monthNumber == 8 || monthNumber == 10)
```

```
        return 30;
    else if (monthNumber == 1)
        return (isLeapYear(year)) ? 29 : 28;
    return 0;
}

bool Calendar::isLeapYear(int year)
{
    return (year % 4 == 0 && year % 100 != 0) || (year % 400 == 0);
}

void Calendar::printTask(const string& task)
{
    cout << "- " << task << endl;
}

void Calendar::printCalendar(int year, int month)
{
    cout << "\n\n        Calendar - " << year << "\n\n";
    int days, current;

    if (month >= 0 && month < 12)
            {
        days = numberOfDays(month, year);
        cout << "\n ------------" << getMonthName(month) << "-------------\n";
        cout << " Sun  Mon  Tue  Wed  Thu  Fri  Sat\n";
        current = dayNumber(1, month + 1, year);
        for (int k = 0; k < current; k++)
            cout << "     ";
        for (int j = 1; j <= days; j++)
                    {
```

```
        cout.width(5);
        cout << j;
        if (++current > 6)
                                {
          current = 0;
          cout << endl;
        }
      }
      if (current)
        cout << endl;
    }
         else
         {
      for (int i = 0; i < 12; i++)
                  {
        days = numberOfDays(i, year);
        cout << "\n ------------" << getMonthName(i) << "-------------\n";
        cout << " Sun  Mon  Tue  Wed  Thu  Fri  Sat\n";
        current = dayNumber(1, i + 1, year);
        for (int k = 0; k < current; k++)
          cout << "     ";
        for (int j = 1; j <= days; j++)
                                {
          cout.width(5);
          cout << j;
          if (++current > 6)
                                        {
            current = 0;
            cout << endl;
          }
        }
```

```
            if (current)
                cout << endl;
        }
    }
}

void Calendar::printCurrentDate()
{
    time_t now = time(0);
    tm* currentDate = localtime(&now);
    cout << "Today's date: " << currentDate->tm_mday << "/" << currentDate->tm_mon + 1 << "/" <<
currentDate->tm_year + 1900 << endl;
}

int Calendar::calculateAge(int birthYear)
{
    time_t now = time(0);
    tm* currentDate = localtime(&now);
    int currentYear = currentDate->tm_year + 1900;
    return currentYear - birthYear;
}

bool Calendar::checkLeapYear(int year)
{
    return isLeapYear(year);
}

void Calendar::addTask()
{
    ofstream taskFile("tasks.txt", ios_base::app);
    if (!taskFile.is_open())
```

```
            {
        cout << "Error: Unable to open tasks file!" << endl;
        return;
    }

    string task;
    string date;

    cout << "Enter the task: ";
    cin.ignore();
    getline(cin, task);

    cout << "Enter the date for the task (DD/MM/YYYY): ";
    getline(cin, date);

    if (date.size() != 10 || date[2] != '/' || date[5] != '/')
            {
        cout << "Invalid date format! Please use DD/MM/YYYY format." << endl;
        return;
    }

    taskFile << date << ": " << task << endl;
    cout << "Task added successfully!" << endl;
    taskFile.close();
}

void Calendar::displayTasks()
{
    ifstream taskFile("tasks.txt");
    if (!taskFile.is_open())
            {
```

```
        cout << "Error: Unable to open tasks file!" << endl;
        return;
    }
    string task;
    cout << "Tasks:" << endl;
    while (getline(taskFile, task))
            {
        printTask(task);
    }
    taskFile.close();
}

void Calendar::removeTask()
{
    ifstream inFile("tasks.txt");
    if (!inFile.is_open())
            {
        cout << "Error: Unable to open tasks file!" << endl;
        return;
    }

    string taskLine;
    int index = 1;

    cout << "Tasks :" << endl;
    while (getline(inFile, taskLine))
            {
        cout << index << ". ";
        printTask(taskLine);
        tasks[index - 1] = taskLine;
        ++index;
```

```
}
inFile.close();

int taskIndex;
cout << "Enter the index of the task to remove: ";
cin >> taskIndex;
if (taskIndex < 1 || taskIndex > 100 || tasks[taskIndex - 1].empty())
        {
  cout << "Invalid index!" << endl;
  return;
}

tasks[taskIndex - 1].clear();

ofstream outFile("tasks.txt");
if (!outFile.is_open())
        {
  cout << "Error: Unable to open tasks file!" << endl;
  return;
}

for (const auto& task : tasks)
        {
  if (!task.empty())
                {
    outFile << task << endl;
  }
}
outFile.close();

cout << "Task removed successfully!" << endl;
```

```
}

void Calendar::editTask() {
    ifstream inFile("tasks.txt");
    if (!inFile.is_open()) {
        cout << "Error: Unable to open tasks file!" << endl;
        return;
    }

    vector<string> tasks;
    string taskLine;
    int index = 1;

    cout << "Tasks :-" << endl;
    while (getline(inFile, taskLine)) {
        cout << index << ". " << taskLine << endl;
        tasks.push_back(taskLine);
        ++index;
    }
    inFile.close();

    int taskIndex;
    cout << "Enter the index of the task to edit :- ";
    cin >> taskIndex;
    if (taskIndex < 1 || taskIndex > tasks.size())
             {
        cout << "Invalid index!" << endl;
        return;
    }

    string newTask;
    cout << "Enter the new task description :- ";
    cin.ignore();
    getline(cin, newTask);

    string newDate;
    cout << "Enter the new date for the task (DD/MM/YYYY) :- ";
    getline(cin, newDate);

    if (newDate.size() != 10 || newDate[2] != '/' || newDate[5] != '/')
            {
        cout << "Invalid date format! Please use DD/MM/YYYY format." << endl;
        return;
    }

    tasks[taskIndex - 1] = newDate + ": " + newTask;

    ofstream outFile("tasks.txt");
    if (!outFile.is_open())
            {
        cout << "Error: Unable to open tasks file!" << endl;
        return;
    }

    for (const auto &task : tasks)
            {
        outFile << task << endl;
    }
    outFile.close();

    cout << "Task edited successfully!" << endl;
}
```

```
void Calendar::setReminder()
{
    ifstream taskFile("tasks.txt");
    if (!taskFile.is_open())
            {
        cout << "Error: Unable to open tasks file!" << endl;
        return;
    }

    vector<pair<string, string>> tasks;
    string taskLine;
    while (getline(taskFile, taskLine))
            {
        size_t pos = taskLine.find(":");
        if (pos != string::npos) {
            string date = taskLine.substr(0, pos);
            string task = taskLine.substr(pos + 2);
            tasks.push_back({date, task});
        }
    }
    taskFile.close();

    cout << "Checking reminders..." << endl;
    time_t now = time(0);
    tm* currentDate = localtime(&now);
    string currentDateStr = (currentDate->tm_mday < 10 ? "0" : "") + to_string(currentDate-
>tm_mday) + "/" + (currentDate->tm_mon + 1 < 10 ? "0" : "") + to_string(currentDate->tm_mon + 1)
+ "/" + to_string(currentDate->tm_year + 1900);

    bool taskFound = false;
    for (const auto &task : tasks)
```

```
            {
        if (task.first == currentDateStr)
                        {
            cout << "Reminder: " << task.second << endl;
            taskFound = true;
        }
    }

    if (!taskFound) {
        cout << "No tasks for today." << endl;
    }
}

void Calendar::searchTasksByDate()
{
    ifstream taskFile("tasks.txt");
    if (!taskFile.is_open()) {
        cout << "Error: Unable to open tasks file!" << endl;
        return;
    }

    string date;
    cout << "Enter the date to search for tasks (DD/MM/YYYY) :- ";
    cin.ignore();
    getline(cin, date);

     if (date.size() != 10 || date[2] != '/' || date[5] != '/')
              {
        cout << "Invalid date format! Please use DD/MM/YYYY format." << endl;
        return;
    }
    string taskLine;

    bool found = false;

    while (getline(taskFile, taskLine))

            {

        if (taskLine.find(date) != string::npos)

                        {

            cout << taskLine << endl;

            found = true;

        }

    }


    if (!found)

            {

        cout << "No tasks found for " << date << endl;

    }


    taskFile.close();

}


int main() {

    cout << "\n\t\t      Welcome To Digital Calendar \n\n";

    int age = 0;

    while (true) {

        cout << "\n\t\t  ===================================";

        int choice;

        cout << "\n\t\t|\t1. Print Calendar            |\n\t\t|\t2. Check Today's Date         |\n\t\t|\t3.
Calculate Age             |\n\t\t|\t4. Add/Display/Remove Tasks   |\n\t\t|\t5. Check for Leap Year
|\n\t\t|\t6. To Search Task            |\n\t\t|\t7. Reminder                 |\n\t\t|\t8. Exit                     |";

        cout << "\n\t\t  ===================================\n";

        cout << "\nEnter Your Choice: ";

        cin >> choice;
        switch (choice) {
        case 1:
            int subChoice;
            cout << "\n\t\t  ===========================================";
            cout << "\n\t\t|\t 1. To Print Full Year Calendar      |\n\t\t|\t 2. To Print Particular Month
|";
            cout << "\n\t\t  ===========================================\n";
            cout << "\nEnter Your Choice :- ";
            cin >> subChoice;
            if (subChoice == 1) {
                int year;
                cout << "Enter year :- ";
                cin >> year;
                Calendar::printCalendar(year);
            }
                                    else if (subChoice == 2)
                                    {
                int year, month;
                cout << "Enter Year :- ";
                cin >> year;
                cout << "Enter Month (1-12) :- ";
                cin >> month;
                Calendar::printCalendar(year, month - 1);
            }
                                    else
                                    {
                cout << "Invalid choice.\n";
            }
            break;
        case 2:
            Calendar::printCurrentDate();
```

```
            break;
        case 3:
            int birthYear;
            cout << "Enter Your Birth Year: ";
            cin >> birthYear;
            age = Calendar::calculateAge(birthYear);
            cout << "Your Age: " << age << endl;
            break;
        case 4:
            int taskChoice;
            cout << "\n\t\t  ===================================";
            cout << "\n\t\t|\t1. Add New Task              |\n\t\t|\t2. Display All Tasks          |\n\t\t|\t3.
Remove Task               |\n\t\t|\t4. Edit Task                 |";
            cout << "\n\t\t  ===================================\n";
            cout << "\nEnter Your Choice: ";
            cin >> taskChoice;
            switch (taskChoice)
                                {
            case 1:
                Calendar::addTask();
                break;
            case 2:
                Calendar::displayTasks();
                break;
            case 3:
                Calendar::removeTask();
                break;
            case 4:
                Calendar::editTask();
                break;
            default:
                cout << "Invalid Choice!";
            }
            break;
        case 5:
            int leapYear;
            cout << "Enter Year to Check: ";
            cin >> leapYear;
            if (Calendar::checkLeapYear(leapYear))
                cout << leapYear << " is a Leap year." << endl;
            else
                cout << leapYear << " is not a Leap year." << endl;
            break;
        case 6:
            Calendar::searchTasksByDate();
            break;
        case 7:
            Calendar::setReminder();
            break;
        case 8:
            cout << "Exiting...";
            exit(0);
        default:
            cout << "Invalid Choice!";
        }
    }
    return 0;
}
```