Міністерство освіти і науки України

Національний технічний університет України «Київський політехнічний

інститут імені Ігоря Сікорського"

Факультет інформатики та обчислювальної техніки

Кафедра інформатики та програмної інженерії

Звіт

з лабораторної роботи №1.2 з дисципліни

«Основи програмування 2. Модульне програмування»

«Бінарні файли»

Варіант 26

Виконав студент: ІП-15 Поліщук Валерій Олександрович

(шифр, прізвище, ім'я, по батькові)

Перевірила: Вечерковська Анастасія Сергіївна

(прізвище, ім'я, по батькові)

Київ 2022

Лабораторна робота № 1.2

Бінарні файли

Варіант 26

Мета — вивчити особливості створення і обробки бінарних файлів.

Постановка задачі

26. Створити файл із списком клієнтів, обслужених менеджером протягом дня: прізвище клієнта, час приходу та час закінчення обслуговування клієнта. При введенні даних перевіряти їх допустимість (чи не перетинаються клієнти з наявними). Визначити, чи мав менеджер протягом робочого дня (з 9:00 до 17:00) вільний час (проміжки), і якщо так, то скільки і в якій половині дня.

Розв’язання

Python

main.py

```python
from funcs import *

ClearOrContinue()
Populate()
PrintRecordings()
FindAndPrintGaps()
```

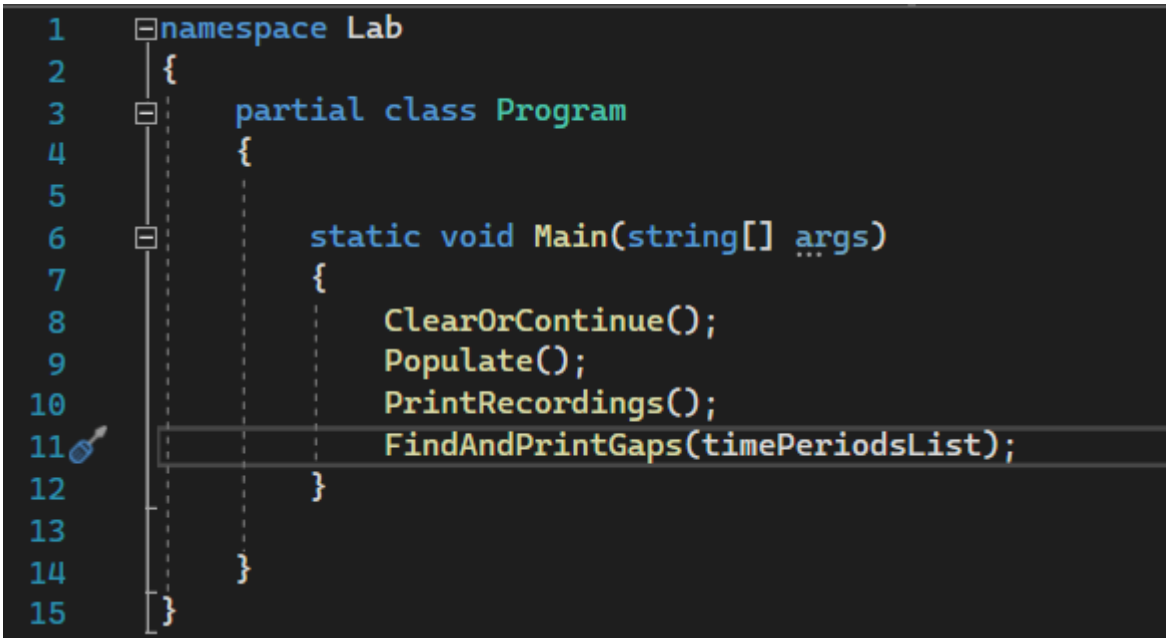
funcs.py

```python
import pickle
from datetime import time
from os.path import exists
import os


CONST_PATH = "records.dat"
dayStart = time(9,0)
dayEnd = time(17,0)
addInfoAboutFinish = False
timePeriodsList = []

class TimePeriod :
    def __init__(self, tStart, tEnd):
        self.tStart = tStart
        self.tEnd = tEnd


def ClearOrContinue():
    if (exists(CONST_PATH)):
        print("records.bat already exists. Clear it (1) or continue populating (2) ?   (1/2) ")
        ans = int(input())
        print()
        if (ans==1):
            os.remove(CONST_PATH)
        else:
            global addInfoAboutFinish
            addInfoAboutFinish = True
            AddExistingRecordsToList()


def AddExistingRecordsToList():
    with open(CONST_PATH, 'rb') as file:
        while (True):
            try:
                surname = pickle.load(file)
            except Exception:
                break

            tStart = pickle.load(file)
            # tStart = time(int(stStart[0:2]), int(stStart[3:]))

            tEnd = pickle.load(file)
            # tEnd = time(int(stEnd[0:2]), int(stEnd[3:]))

            timePeriodsList.append(TimePeriod(tStart,tEnd))

def PrintRecordings():
    print("All records :")
    print()
    with open(CONST_PATH, 'rb') as file:
        while (True):
            try:
                surname = pickle.load(file)
            except Exception:
                break
            tStart = pickle.load(file)
            tEnd = pickle.load(file)

            print("Surname: " + surname)
            print("Service start time: " + str(tStart.strftime("%H:%M")))
            print("Service end time: " + str(tEnd.strftime("%H:%M")))
            print()


def MakeRecord(surname, tStart, tEnd):
    with open(CONST_PATH,'ab') as file:
        pickle.dump(surname,file)
        pickle.dump(tStart, file)
        pickle.dump(tEnd, file)
```

```
def Populate() :
    while (True):
        overlaping = False
        global addInfoAboutFinish
        if(addInfoAboutFinish):
            print("Enter a surname of your client (type \"finish\" to finish) :")
        else:
            print("Enter a surname of your client :")
        surname = input()
        if (surname=="finish"):
            break
        print()

        print("Enter a customer arrival time (in format hh:mm) :")
        stStart = input()
        try:
            tStart = time(int(stStart[0:2]), int(stStart[3:]))
        except Exception:
            print()
            print("Incorrect time format. Record could not be added!")
            print()
            continue
        print()

        print("Enter a customer service end time (in format hh:mm) :")
        stEnd = input()
        try:
            tEnd = time(int(stEnd[0:2]), int(stEnd[3:]))
        except Exception:
            print()
            print("Incorrect time format. Record could not be added!")
            print()
            continue
        print()

        if (tEnd<=tStart):
            print("Start time cannot be greater then or equal to end time! Record could not be added!")
            print()
            continue

        if (tStart<dayStart or tEnd>dayEnd):
            print("Working day starts at 09:00 and ends at 17:00. Record could not be added! ")
            print()
            continue

        for span in timePeriodsList:
            if (tStart< span.tEnd and span.tStart < tEnd):
                print("This time period overlaps the existing one. Record could not be added!")
                overlaping = True
                break

        if (overlaping):
            continue

        timePeriodsList.append(TimePeriod(tStart, tEnd))
        MakeRecord(surname, tStart, tEnd)
        addInfoAboutFinish = True

def FindAndPrintGaps():
    print()
    print("Time when manager was free :")
    print()
    sortedList = timePeriodsList
    sortedList.sort(key= lambda TimePeriod: TimePeriod.tStart)
    freeTime = False
    count = 0
    position = dayStart
    for period in sortedList:
        if (position < period.tStart):
            count+=1
            print(str(position.strftime("%H:%M")) + " - " + str(period.tStart.strftime("%H:%M")))
        position = period.tEnd

    if (sortedList[-1].tEnd < dayEnd):
        count+=1
        print(str(sortedList[-1].tEnd.strftime("%H:%M")) + " - " + str(dayEnd.strftime("%H:%M")))

    print()
    print("A total amout of gaps : " + str(count))
```

C#

Program.cs



```
namespace Lab
{
    partial class Program
    {

        static void Main(string[] args)
        {
            ClearOrContinue();
            Populate();
            PrintRecordings();
            FindAndPrintGaps(timePeriodsList);
        }

    }
}
```

Funcs.cs

```
using System;
namespace Lab
{
    partial class Program
    {
        static List<TimePeriod> timePeriodsList = new List<TimePeriod>();
        const string path = "records.dat";
        static TimeOnly dayStart = new TimeOnly(9, 0);
        static TimeOnly dayEnd = new TimeOnly(17, 0);
        static bool addInfoAboutFinish = false;

        struct TimePeriod
        {
            public TimeOnly tStart;
            public TimeOnly tEnd;
        }


        // FULL PROCESS OF PUPULATING BINARY FILE
        private static void Populate()
        {

            ConsoleKeyInfo key;
            bool overlaping = false;
            BinaryWriter writer = new BinaryWriter(File.Open(path, FileMode.Append));

            while (true)
            {
                TimeOnly tStart;
                TimeOnly tEnd;
                overlaping = false;

                if (addInfoAboutFinish)
                {
                    Console.WriteLine(@"Enter a surname of your client (press ""Home"" button to finish) :");
                }
                else
                {
                    Console.WriteLine("Enter a surname of your client :");
                }
                key = Console.ReadKey();
                if (key.Key == ConsoleKey.Home)
                {
                    break;
                }

                string surname = key.KeyChar.ToString() + Console.ReadLine();
                Console.WriteLine();


```

```
            Console.WriteLine("Enter a customer arrival time (in format hh:mm) :");
            string? startTime = Console.ReadLine();
            try
            {
                tStart = new TimeOnly(Convert.ToInt32(startTime[0..2]), Convert.ToInt32(startTime[3..5]));
            }
            catch (Exception)
            {
                Console.WriteLine();
                Console.WriteLine("Incorrect time format. Record could not be added!");
                Console.WriteLine();
                continue;
            }
            Console.WriteLine();



            Console.WriteLine("Enter a customer service end time (in format hh:mm) :");
            string? endTime = Console.ReadLine();
            try
            {
                tEnd = new TimeOnly(Convert.ToInt32(endTime[0..2]), Convert.ToInt32(endTime[3..5]));
            }
            catch (Exception)
            {
                Console.WriteLine();
                Console.WriteLine("Incorrect time format. Record could not be added!");
                Console.WriteLine();
                continue;
            }
            Console.WriteLine();



            if (tEnd <= tStart)
            {
                Console.WriteLine("Start time cannot be greater then or equal to end time! Record could not be added!");
                Console.WriteLine();
                continue;
            }

            if (tStart < dayStart || tEnd > dayEnd)
            {
                Console.WriteLine("Working day starts at 09:00 and ends at 17:00. Record could not be added! ");
                Console.WriteLine();
                continue;
            }
```

```
            foreach (var span in timePeriodsList)
            {
                if (tStart < span.tEnd && span.tStart < tEnd)
                {
                    Console.WriteLine("This time period overlaps the existing one. Record could not be added!");
                    Console.WriteLine();
                    overlaping = true;
                    break;
                }

            }

            if (overlaping)
            {
                continue;
            }

            timePeriodsList.Add(new TimePeriod { tEnd = tEnd, tStart = tStart });
            MakeRecord(surname, tStart, tEnd, writer);
            addInfoAboutFinish = true;

        }
        writer.Close();

    }


    // ASK USER (CLEAR FILE OR CONTINUE) IF FILE ALREADY EXIST
    private static void ClearOrContinue()
    {
        if (File.Exists(path))
        {
            Console.WriteLine("records.bat already exists. Clear it (1) or continue populating (2) ?   (1/2) ");
            int ans = Convert.ToInt32(Console.ReadLine());
            Console.WriteLine();
            if (ans == 1)
            {
                File.Delete(path);
            }
            else
            {
                addInfoAboutFinish = true;
                AddExistingRecordsToList();
            }
            Console.Clear();
        }
    }
```

```
148         // WRITE A RECORD IN A BINARY FILE
149         private static void MakeRecord(string surname, TimeOnly tStart, TimeOnly tEnd, BinaryWriter writer)
150         {
151
152             writer.Write(surname);
153             writer.Write(tStart.ToString("HH:mm"));
154             writer.Write(tEnd.ToString("HH:mm"));
155
156         }
157
158
159         //ADDS EXISTING RECORDS IF USER CHOSE TO COUNTINUE EDITING FILE
160         private static void AddExistingRecordsToList()
161         {
162             using (BinaryReader reader = new BinaryReader(File.Open("records.dat", FileMode.Open)))
163             {
164
165                 while (reader.PeekChar() > -1)
166                 {
167                     string surname = reader.ReadString();
168                     string startTime = reader.ReadString();
169                     string endTime = reader.ReadString();
170
171                     TimeOnly tEnd = new TimeOnly(Convert.ToInt32(endTime[0..2]), Convert.ToInt32(endTime[3..5]));
172                     TimeOnly tStart = new TimeOnly(Convert.ToInt32(startTime[0..2]), Convert.ToInt32(startTime[3..5]));
173
174                     timePeriodsList.Add(new TimePeriod { tEnd = tEnd, tStart = tStart });
175
176                 }
177             }
178         }
```

```
181         // PRINT ALL RECORDINGS FROM A BINARY FILE IN CONSOLE
182         private static void PrintRecordings()
183         {
184             Console.Clear();
185             Console.WriteLine("All records :");
186             Console.WriteLine();
187             using (BinaryReader reader = new BinaryReader(File.Open("records.dat", FileMode.Open)))
188             {
189                 // пока не достигнут конец файла
190                 // считываем каждое значение из файла
191                 while (reader.PeekChar() > -1)
192                 {
193                     string surname = reader.ReadString();
194                     string startTime = reader.ReadString();
195                     string endTime = reader.ReadString();
196                     //TimeOnly tEnd = new TimeOnly(Convert.ToInt32(endTime[0..2]), Convert.ToInt32(endTime[3..5]));
197                     //TimeOnly tStart = new TimeOnly(Convert.ToInt32(startTime[0..2]), Convert.ToInt32(startTime[3..5]));
198                     //timePeriodsList.Add(new TimePeriod { tEnd = tEnd, tStart = tStart });
199                     Console.WriteLine("Surname: " + surname);
200                     Console.WriteLine("Service start time: " + startTime);
201                     Console.WriteLine("Service end time: " + endTime);
202                     Console.WriteLine();
203
204                 }
205             }
206         }
207
208
209         //FINDS AND PRINTS ALL TIME PERIODS WHEN MANAGER WAS FREE
210         private static void FindAndPrintGaps(List<TimePeriod> timePeriodsList)
211         {
212             Console.WriteLine();
213             Console.WriteLine("Time when manager was free :");
214             Console.WriteLine();
215             List<TimePeriod> sorted = timePeriodsList.OrderBy(x => x.tStart).ToList();
216             bool freeTime = false;
217             int count = 0;
218             TimeOnly position = dayStart;
219             foreach (var period in sorted)
220             {
221                 if (position < period.tStart)
222                 {
223                     count++;
224                     Console.WriteLine(position + " - " + period.tStart);
225                 }
226                 position = period.tEnd;
227
228             }
229             if (sorted.Last().tEnd < dayEnd)
230             {
231                 count++;
232                 Console.WriteLine(sorted.Last().tEnd + " - " + dayEnd);
233             }
234             Console.WriteLine();
235             Console.WriteLine("A total amout of gaps : " + count);
236         }
237     }
238 }
```

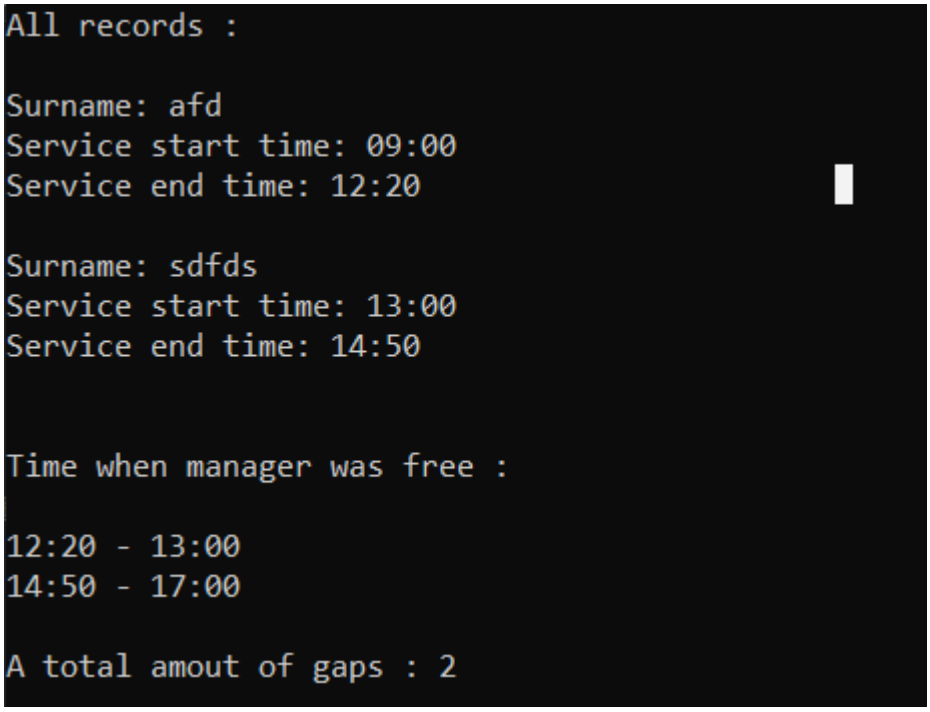
Тестування програми

Python

```
records.bat already exists. Clear it (1) or continue populating (2) ?   (1/2)
1

Enter a surname of your client :
user

Enter a customer arrival time (in format hh:mm) :
12:22

Enter a customer service end time (in format hh:mm) :
14:00

Enter a surname of your client (type "finish" to finish) :
user2

Enter a customer arrival time (in format hh:mm) :
15:00

Enter a customer service end time (in format hh:mm) :
17:00

Enter a surname of your client (type "finish" to finish) :
finish
All records :

Surname: user
Service start time: 12:22
Service end time: 14:00

Surname: user2
Service start time: 15:00
Service end time: 17:00


Time when manager was free :

09:00 - 12:22
14:00 - 15:00

A total amout of gaps : 2

Process finished with exit code 0
```

C#

```
All records :

Surname: dddd
Service start time: 09:00
Service end time: 17:00


Time when manager was free :


A total amout of gaps : 0
```



Висновок

Я вивчив особливості створення та обробки бінарних файлів.