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. Метидології програмування»

«Бінарні файли»

Варіант 13

Виконав студент ІП-13 Жмайло Дмитро Олександрович

(шифр, прізвище, ім'я, по батькові)

Перевірив Вєчерковська Анастасія Сергіївна

(прізвище, ім'я, по батькові)

Київ 2022

**Лабораторна робота 1**

**Текстові файли**

**Варіант 13**

Створити файл з інформацією про телефонні переговори: номер телефону,

початок та кінець переговорів (за шаблоном - ГГ.ХХ). Розрахувати оплату за переговори, вважаючи, що хвилина розмови вдень (з 9:00 до 201 00) коштує 1,5 грн.,а в вночі -- 0,90 грн. Видалити з файлу дані про розмови тривалістю менше 3 хв.

**Код програми**

**С#**

**Program.cs**using System;

using System.Collections.Generic;

namespace Lab1.\_2

{

class Program

{

static void Main(string[] args)

{

const string filePath = "conversations.bat";

bool appendOrNot = Operations.ChooseAppendOrNot(filePath);

List<CallInfo> callList = Operations.InputInfo();

Operations.SaveInfo(filePath, callList, appendOrNot);

List<CallInfo> newCallList = Operations.ReadInfo(filePath);

Operations.ShowInfo(newCallList);

newCallList = Operations.DeleteShortest(newCallList);

Operations.SaveInfo(filePath, newCallList, false);

List <CallInfo>finalCallList = Operations.ReadInfo(filePath);

Operations.ShowInfo(finalCallList);

}

}

}

**Callinfo.cs**using System;

using System.Collections.Generic;

using System.Text;

namespace Lab1.\_2

{

class CallInfo

{

private string phoneNumber;

private string startTime;

private string endTime;

public CallInfo(string phoneNumber, string startTime, string endTime)

{

this.phoneNumber = phoneNumber;

this.startTime = startTime;

this.endTime = endTime;

}

public string GetPhoneNumber()

{

return phoneNumber;

}

public string GetStartMinute()

{

return startTime;

}

public string GetEndMinute()

{

return endTime;

}

public int Duration

{

get

{

int startMinute = ConvertStringTimeToMinutes(startTime);

int endMinute = ConvertStringTimeToMinutes(endTime);

if (startMinute > endMinute)

{

return endMinute + 24 \* 60 - startMinute;

}

return endMinute - startMinute;

}

}

public float Payment

{

get

{

float price = 0;

int totalDurationNight = 0;

int totalDurationDay = 0;

//int duration = Duration;

float priceDay = 1.5f;

float priceNight = 0.9f;

const int zeroMinuteOfTheDay = 0;

const int lastMinuteOfTheDay = 24 \* 60;

int startT = ConvertStringTimeToMinutes(startTime);

int endT = ConvertStringTimeToMinutes(endTime);

if (startT <= endT)

{

GetNightAndDayDuration(startT, endT, out totalDurationNight, out totalDurationDay);

}

else

{

GetNightAndDayDuration(startT, lastMinuteOfTheDay, out int firstDurationNight, out int firstDurationDay);

GetNightAndDayDuration(zeroMinuteOfTheDay, endT, out int secondDurationNight, out int secondDurationDay);

totalDurationDay = firstDurationDay + secondDurationDay;

totalDurationNight = firstDurationNight + secondDurationNight;

}

price = totalDurationNight \* priceNight + totalDurationDay \* priceDay;

return price;

}

}

public static bool TryCreateFromString(string line, out CallInfo result)

{

string[] elements = line.Split(' ');

if (elements.Length != 3)

{

result = null;

return false;

}

else if (!IsPhoneNumberValid(elements[0]))

{

result = null;

return false;

}

else if (!IsTimeValid(elements[1]) || !IsTimeValid(elements[2]))

{

result = null;

return false;

}

result = new CallInfo(elements[0], elements[1], elements[2]);

return true;

}

private static bool IsPhoneNumberValid(string number)

{

if (number.Length != 13)

{

return false;

}

if (number[0] != '+')

{

return false;

}

if (number[3] != '0')

{

return false;

}

for (int i = 1; i < number.Length; i++)

{

if (!Char.IsDigit(number[i]))

{

return false;

}

}

return true;

}

private static bool IsTimeValid(string time)

{

if (time.Length != 5 || time[2] != ':')

{

return false;

}

for (int i = 0; i < time.Length; i++)

{

if (i != 2 && !char.IsDigit(time[i]))

{

return false;

}

}

string[] digits = time.Split(':');

int hours = Convert.ToInt32(digits[0]);

int minutes = Convert.ToInt32(digits[1]);

if (hours > 23 || minutes > 59)

{

return false;

}

return true;

}

private static int ConvertStringTimeToMinutes(string time)

{

string[] digits = time.Split(':');

int hours = Convert.ToInt32(digits[0]);

int minutes = Convert.ToInt32(digits[1]);

return hours \* 60 + minutes;

}

private static void GetNightAndDayDuration(int startT, int endT, out int durationNight, out int durationDay)

{

durationNight = 0;

durationDay = 0;

int nightTime = 20 \* 60;

int dayTime = 9 \* 60;

if (startT == endT)

{

durationNight = 0;

durationDay = 0;

}

else if (startT < dayTime && endT < dayTime)

{

durationNight = endT - startT;

}

else if (startT < dayTime && endT > dayTime && endT < nightTime)

{

durationNight = dayTime - startT;

durationDay = endT - dayTime;

}

else if (startT < dayTime && endT >= nightTime)

{

durationNight = dayTime - startT + endT - nightTime;

durationDay = nightTime - dayTime;

}

else if (startT >= dayTime && startT < nightTime && endT < nightTime)

{

durationDay = endT - startT;

}

else if (startT >= dayTime && startT < nightTime && endT >= nightTime)

{

durationDay = nightTime - startT;

durationNight = endT - nightTime;

}

else if (startT >= nightTime)

{

durationNight = endT - startT;

}

}

}

}

**Operations.cs**

using System;

using System.Collections.Generic;

using System.IO;

using System.Text;

namespace Lab1.\_2

{

class Operations

{

public static List<CallInfo> InputInfo()

{

List<CallInfo> infoList = new List<CallInfo>();

Console.WriteLine("Enter information about phone calls: (format: +XX0XXXXXXXXX HH:MM HH:MM), type Ctrl + X to stop");

string exitLine = "\u0018";

while (true)

{

Console.Write("\t");

string line = Console.ReadLine();

if (line == exitLine)

{

Console.WriteLine();

return infoList;

}

else if (CallInfo.TryCreateFromString(line, out CallInfo result))

{

infoList.Add(result);

}

else

{

Console.ForegroundColor = ConsoleColor.Red;

Console.WriteLine("Error. Wrong input format");

Console.ForegroundColor = ConsoleColor.Gray;

}

}

}

public static bool ChooseAppendOrNot(string path)

{

if (File.Exists(path))

{

while (true)

{

Console.WriteLine("Do you want to add new info to existing file or clear it? (enter 'a' or 'c')");

Console.Write("\t");

string input = Console.ReadLine();

if (input == "a")

{

return true;

}

else if (input == "c")

{

return false;

}

else

{

Console.ForegroundColor = ConsoleColor.Red;

Console.WriteLine("Wrong symbol. Try again");

Console.ForegroundColor = ConsoleColor.Gray;

}

}

}

return true;

}

public static void SaveInfo(string path, List<CallInfo> infoList, bool appendOrNot)

{

if (!appendOrNot)

{

File.Delete(path);

}

using (BinaryWriter writer = new BinaryWriter(File.Open(path, FileMode.Append)))

{

foreach (CallInfo callInfo in infoList)

{

writer.Write(callInfo.GetPhoneNumber());

writer.Write(callInfo.GetStartMinute());

writer.Write(callInfo.GetEndMinute());

}

}

Console.ForegroundColor = ConsoleColor.Green;

Console.WriteLine("File has been successfully written");

Console.ForegroundColor = ConsoleColor.Gray;

Console.WriteLine();

}

public static List<CallInfo> ReadInfo(string path)

{

List<CallInfo> infoList = new List<CallInfo>();

using (BinaryReader reader = new BinaryReader(File.Open(path, FileMode.Open)))

{

while (reader.PeekChar() > -1)

{

string number = reader.ReadString();

string startTime = reader.ReadString();

string endTime = reader.ReadString();

infoList.Add(new CallInfo(number, startTime, endTime));

}

}

return infoList;

}

public static void ShowInfo(List<CallInfo>info)

{

Console.WriteLine("Saved info: ");

for (int i = 0; i < info.Count; i++)

{

Console.WriteLine($" {info[i].GetPhoneNumber()} {info[i].GetStartMinute()} {info[i].GetEndMinute()}; Payment : {info[i].Payment}");

}

Console.WriteLine();

}

public static List<CallInfo> DeleteShortest(List<CallInfo> info)

{

for (int i = 0; i < info.Count; i++)

{

if (info[i].Duration < 3)

{

info.Remove(info[i]);

i--;

}

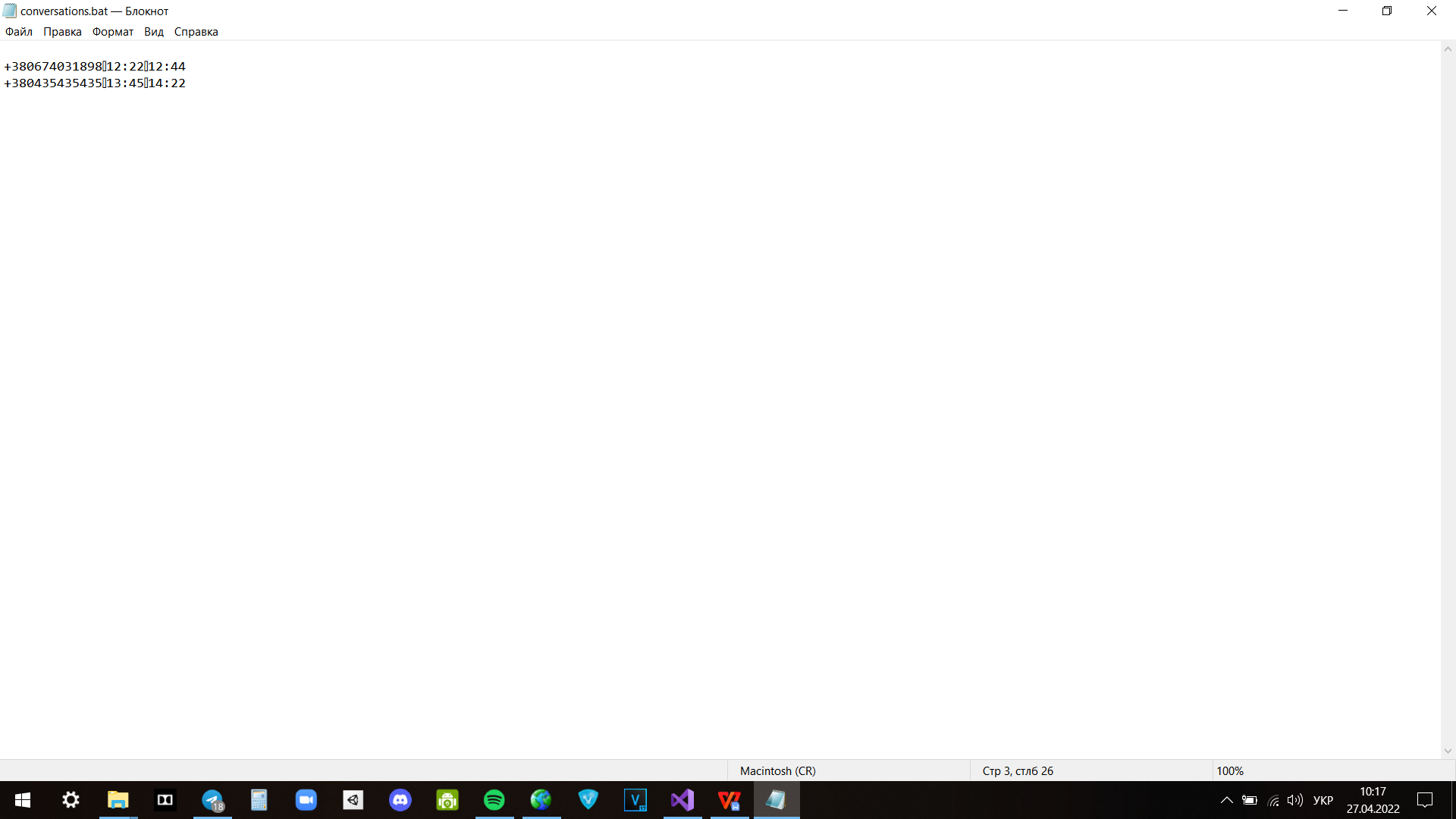
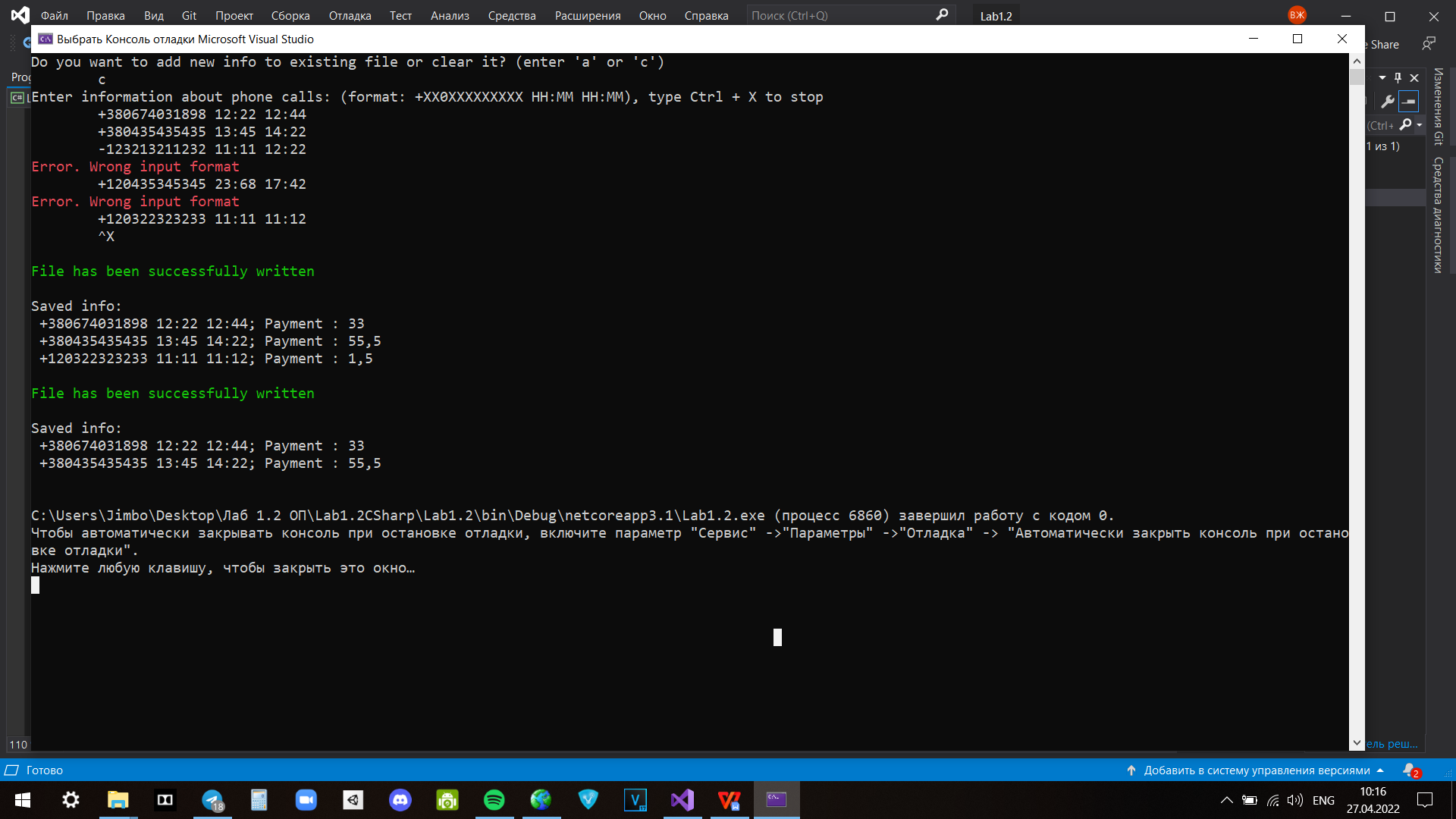
}

return info;

}

}

}

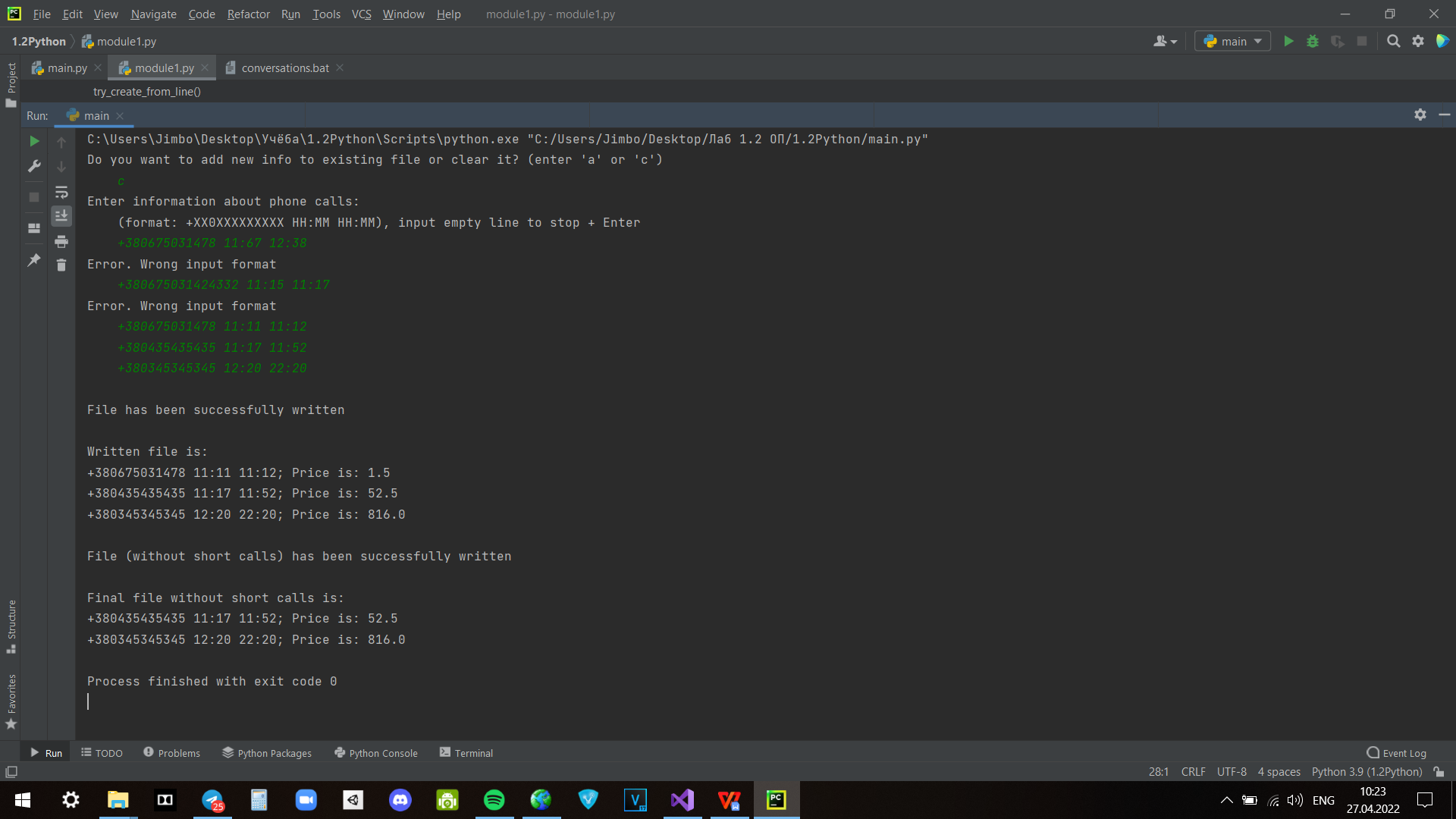
**Тестування:**

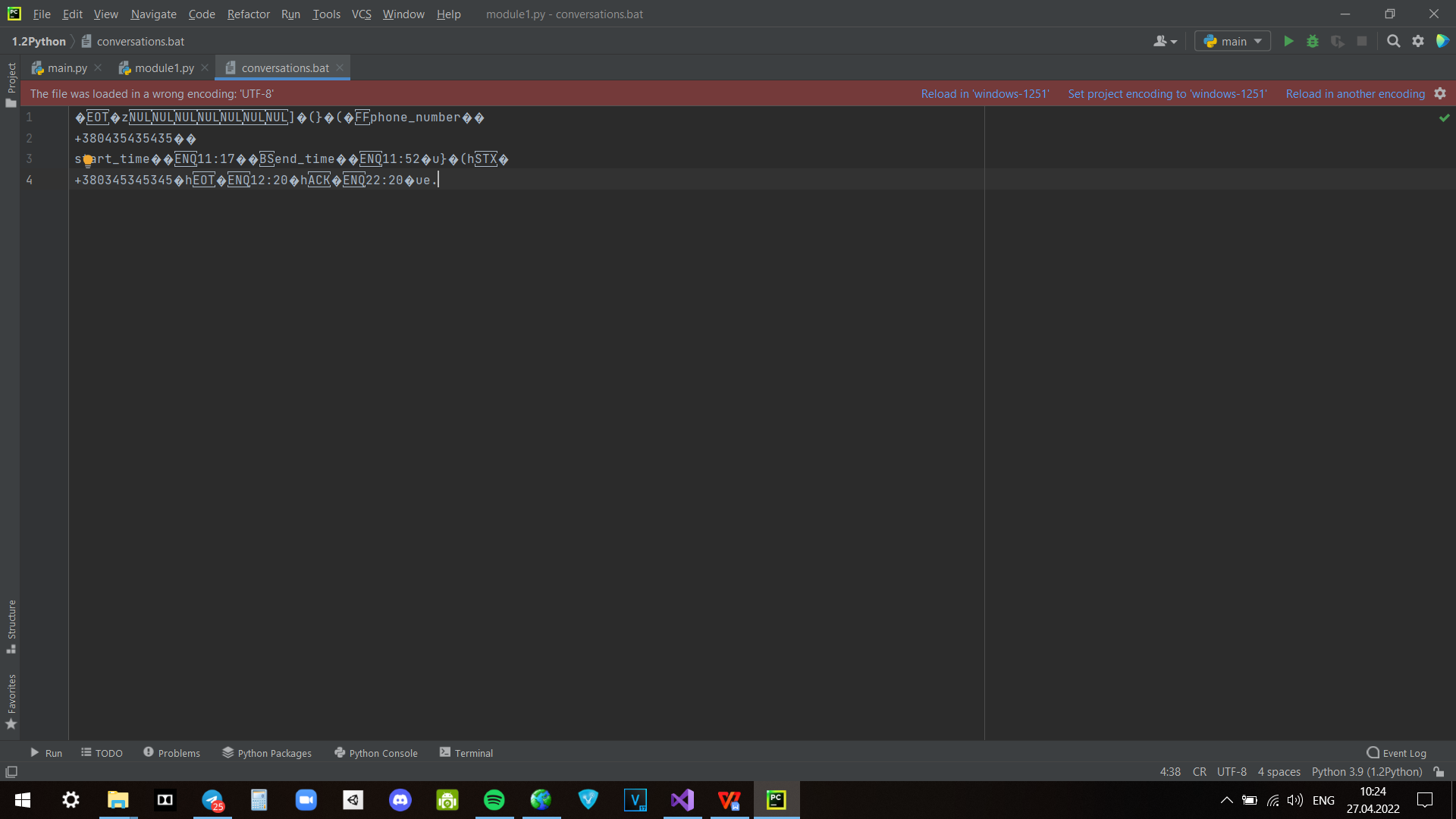
**Python  
  
main.py**from module1 import \*  
  
path = "conversations.bat"  
choose\_append\_or\_not(path)  
list\_of\_calls = input\_info()  
  
write\_info(path, list\_of\_calls)  
new\_list\_of\_calls = read\_info(path)  
  
print("Written file is: ")  
show\_info(new\_list\_of\_calls)  
delete\_shortest(new\_list\_of\_calls)  
  
write\_without\_shortest(path, new\_list\_of\_calls)  
final\_list\_of\_calls = read\_info(path)  
print("Final file without short calls is: ")  
show\_info(final\_list\_of\_calls)

**module1.py**

import pickle  
  
  
def choose\_append\_or\_not(path):  
 input\_mode = str(input("Do you want to add new info to existing file or clear it? (enter 'a' or 'c') \n \t"))  
 while True:  
 if input\_mode == "a":  
 break  
 elif input\_mode == "c":  
 open(path, "wb").close()  
 break  
 else:  
 input\_mode = str(input("Wrong symbol. Try again: \n \t"))  
  
  
def input\_info():  
 info = []  
 line = input('''Enter information about phone calls:   
 (format: +XX0XXXXXXXXX HH:MM HH:MM), input empty line to stop + Enter \n \t''')  
 while line:  
 if try\_create\_from\_line(line):  
 info.append(line)  
 else:  
 print("Error. Wrong input format")  
 line = input("\t")  
 return info  
  
  
def try\_create\_from\_line(line):  
 elements = line.split()  
 if len(elements) != 3:  
 return False  
 elif not is\_phone\_number\_valid(elements[0]):  
 return False  
 elif not (is\_time\_valid(elements[1]) & is\_time\_valid(elements[2])):  
 return False  
  
 return True  
  
  
def write\_info(path, info):  
 saved\_info = []  
 try:  
 with open(path, "rb") as file:  
 saved\_info = pickle.load(file)  
 except:  
 pass  
 for call in info:  
 elements = call.split()  
 call\_info = {  
 "phone\_number": elements[0],  
 "start\_time": elements[1],  
 "end\_time": elements[2]  
 }  
 saved\_info.append(call\_info)  
 with open(path, "wb") as file:  
 pickle.dump(saved\_info, file)  
 print("File has been successfully written \n")  
  
  
def write\_without\_shortest(path, info):  
 with open(path, "wb") as file:  
 pickle.dump(info, file)  
 print("\nFile (without short calls) has been successfully written \n")  
  
  
def read\_info(path):  
 with open(path, "rb") as file:  
 info = pickle.load(file)  
 return info  
  
  
def show\_info(info):  
 for calls in info:  
 print(f'{calls["phone\_number"]} {calls["start\_time"]} {calls["end\_time"]}; '  
 f'Price is: {get\_price(calls["start\_time"], calls["end\_time"])}')  
  
  
def get\_price(start\_time, end\_time):  
 price = 0  
 total\_duration\_night = 0  
 total\_duration\_day = 0  
 price\_day = 1.5  
 price\_night = 0.9  
  
 zero\_minute\_of\_the\_day = 0  
 last\_minute\_of\_the\_day = 24\*60  
  
 start\_t = convert\_time\_into\_minutes(start\_time)  
 end\_t = convert\_time\_into\_minutes(end\_time)  
  
 if start\_t <= end\_t:  
 total\_duration\_day, total\_duration\_night = get\_night\_and\_day\_duration(start\_t, end\_t)  
 else:  
 first\_duration\_day, first\_duration\_night = get\_night\_and\_day\_duration(start\_t, last\_minute\_of\_the\_day)  
 second\_duration\_day, second\_duration\_night = get\_night\_and\_day\_duration(zero\_minute\_of\_the\_day, end\_t)  
 total\_duration\_day = first\_duration\_day + second\_duration\_day  
 total\_duration\_night = first\_duration\_night + second\_duration\_night  
  
 price = total\_duration\_night \* price\_night + total\_duration\_day \* price\_day  
 return price  
  
  
def get\_night\_and\_day\_duration(start\_t, end\_t):  
 duration\_night = 0  
 duration\_day = 0  
  
 night\_time = 20 \* 60  
 day\_time = 9 \* 60  
 if start\_t == end\_t:  
 duration\_night = 0  
 duration\_day = 0  
 elif start\_t < end\_t < day\_time:  
 duration\_night = end\_t - start\_t  
 elif (start\_t < day\_time) & (day\_time < end\_t < night\_time):  
 duration\_night = day\_time - start\_t  
 elif (start\_t < day\_time) & (end\_t >= night\_time):  
 duration\_night = day\_time - start\_t + end\_t - night\_time  
 duration\_day = night\_time - day\_time  
 elif (start\_t >= day\_time) & (start\_t < end\_t < night\_time):  
 duration\_day = end\_t - start\_t  
 elif (day\_time <= start\_t < night\_time) & (end\_t >= night\_time):  
 duration\_day = night\_time - start\_t  
 duration\_night = end\_t - night\_time  
 elif start\_t >= night\_time:  
 duration\_night = end\_t - start\_t  
  
 return duration\_day, duration\_night  
  
  
def delete\_shortest(info):  
 for calls in info:  
 if get\_duration(calls["start\_time"], calls["end\_time"]) < 3:  
 info.remove(calls)  
  
  
def get\_duration(start\_time, end\_time):  
 start\_t = convert\_time\_into\_minutes(start\_time)  
 end\_t = convert\_time\_into\_minutes(end\_time)  
 if start\_t > end\_t:  
 return end\_t + 24 \* 60 - start\_t  
 return end\_t - start\_t  
  
  
def convert\_time\_into\_minutes(line):  
 digits = line.split(":")  
 return int(digits[0]) \* 60 + int(digits[1])  
  
  
def is\_phone\_number\_valid(line):  
 if len(line) != 13:  
 return False  
 if line[3] != '0':  
 return False  
  
 edited\_line = line.replace('+', '1')  
 if not edited\_line.isdigit():  
 return False  
  
 return True  
  
  
def is\_time\_valid(time):  
 if (len(time) != 5) | (time[2] != ':'):  
 return False  
  
 edited\_time = time.replace(':', '1')  
 if not edited\_time.isdigit():  
 return False  
  
 digits = time.split(':')  
 if (int(digits[0]) > 23) | (int(digits[1]) > 59):  
 return False  
  
 return True

**Тестування:**





**Висновки:**

На цій лабораторній роботі я застосував на практиці знання щодо створення та обробки бінарних файлів данних на двох мовах програмування та побачив відмінності в їх реалізації.