

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.  
Методології програмування»

«Бінарні файли»

Варіант 13

Виконав студент ПІ-13 Жмайло Дмитро Олександрович  
(шифр, прізвище, ім'я, по батькові)

Перевірив Вечерковська 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 20:00) коштує 1,5 грн., а вночі -- 0,90 грн. Видалити з файлу дані про розмови тривалістю менше 3 хв.

### Код програми

C#

#### 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)
{

```

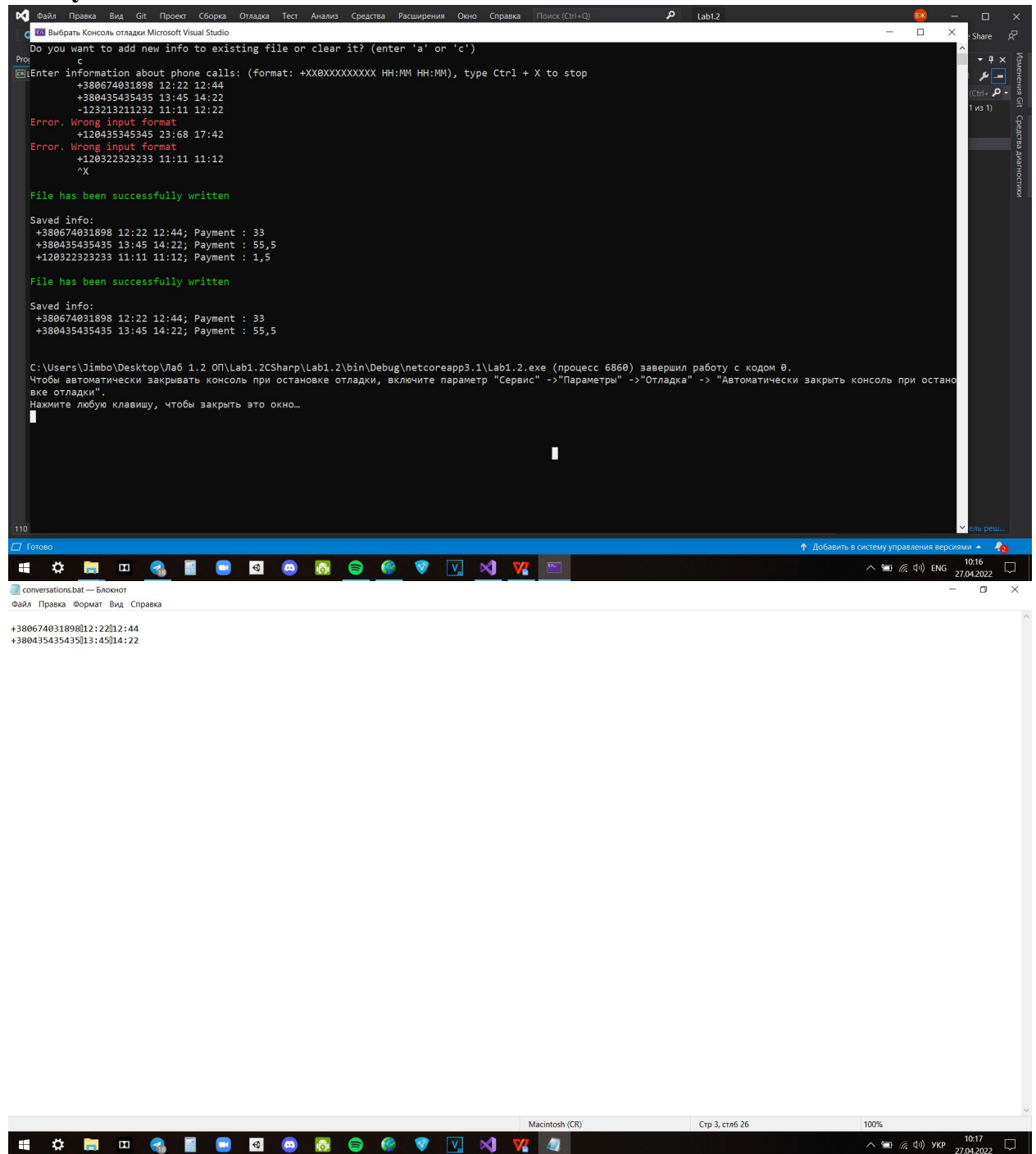
```

        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:



The screenshot displays a Windows desktop environment. The primary window is Microsoft Visual Studio Code, titled 'Lab1.2'. The console pane at the bottom shows the following output:

```
Do you want to add new info to existing file or clear it? (enter 'a' or 'c')
Enter information about phone calls: (format: +XX0XXXXXXXXX HH:MM HH:MM), type Ctrl + X to stop
+380674031898 12:22 12:44
+380435435435 13:45 14:22
-123213211232 11:11 12:22
Error. Wrong input format
+120435345345 23:68 17:42
Error. Wrong input format
+120322323233 11:11 11:12
^X

File has been successfully written

Saved info:
+380674031898 12:22 12:44; Payment : 33
+380435435435 13:45 14:22; Payment : 55,5
+120322323233 11:11 11:12; Payment : 1,5

File has been successfully written

Saved info:
+380674031898 12:22 12:44; Payment : 33
+380435435435 13:45 14:22; Payment : 55,5

C:\Users\Jimbo\Desktop\Лаб 1.2 ОП\Lab1.2CSharp\Lab1.2\bin\Debug\netcoreapp3.1\Lab1.2.exe (процесс 6860) завершил работу с кодом 0.
Чтобы автоматически закрывать консоль при остановке отладки, включите параметр "Сервис" -> "Параметры" -> "Отладка" -> "Автоматически закрыть консоль при остано
вке отладки".
Нажмите любую клавишу, чтобы закрыть это окно..
```

Below the Visual Studio Code window, a Notepad window titled 'conversations.bat — Блокнот' is open, displaying the same log data as the console:

```
+38067403189812:2212:44
+38043543543513:4514:22
```

The Windows taskbar at the bottom shows the system clock as 10:16 on 27.04.2022. The bottom-most bar of the image shows a Macintosh (CR) interface with a system clock of 10:17 on 27.04.2022.

# 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: +XX0XXXXXXXXXX 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"]}; '
              fPrice 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] != ':'):

```



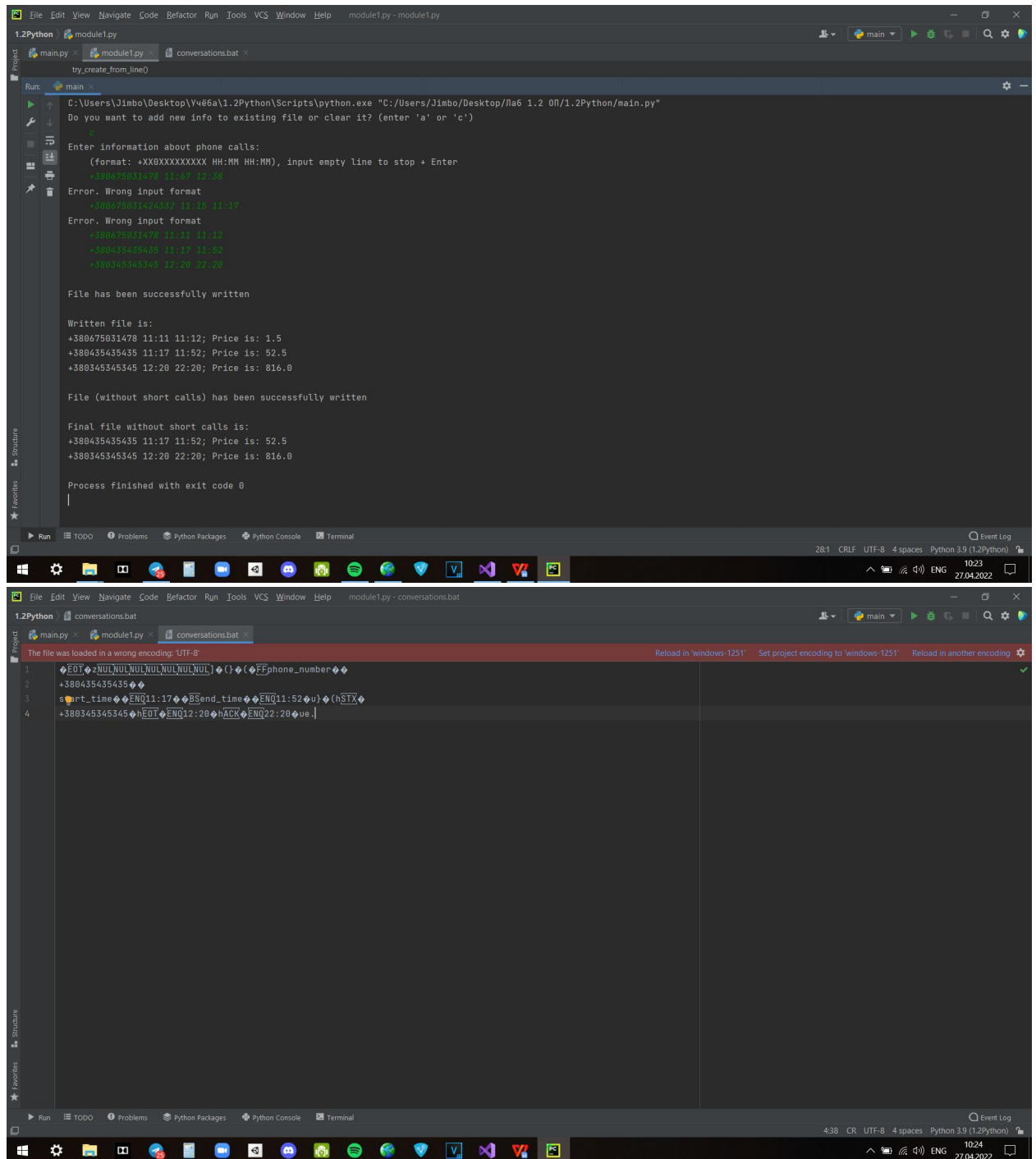
```
    return False

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 даних на двох мовах програмування та побачив відмінності в їх реалізації.