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 № 2 з дисципліни
«Основи програмування – 2. Методології програмування»

«Бінарні файли»

Варіант 22

Виконав студент ІП-13, Музичук Віталій Андрійович
(шифр, прізвище, ім'я, по батькові)

Перевірила Вечерковська Анастасія Сергіївна
(прізвище, ім'я, по батькові)

Київ 2022

Лабораторна робота 2

Бінарні файли

Мета – вивчити особливості створення і обробки бінарних файлів даних.

Варіант 22

Завдання:

Створити файл зі списком справ на поточний день: умовна назва, час початку, передбачувана тривалість. Визначити, яка справа за списком наступна (найближча до поточного часу). Створити файл із інформацією про вільний час у другій половині дня (після 13:00): початок та закінчення тимчасового проміжку та його тривалість (розрахувати).

1. Виконання завдання на мові C++:

// Lab_1.cpp

```
#include "func.h"

int main()
{
	ofstream file_out;
	string name_out = create_file(file_out);
	input_file(file_out);
	file_output(name_out);
	the_nearest_case(name_out);
	file_out.close();

	string name_in = free_time(name_out);
	b_file_output(name_in);

	return 0;
}
```

// lib.cpp

```cpp
#include "func.h"

string create_file(ofstream& file) {
	string file_name;
	cout << "Enter a file name: ";
	getline(cin, file_name);
	cout << "How do you want to add text\n1) append to existing file \n2) create " <<
		"new file\n(Write 1 or 2)" << endl;
	while (true)
	{
		int howOpen; cin >> howOpen;
		if (howOpen == 1)
		{
			file.open("Files/" + file_name, ios::binary | ios::app);
			break;
		}
		else {
			if (howOpen == 2) {
				file.open("Files/" + file_name, ios::binary);
				break;
			}
			else {
				cout << "Incorrect input. Try again" << endl;
			}
		}
	}
	if (!file.is_open())
	{
		cerr << "Couldn't open the file";
		exit(0);
	}
	return "Files/" + file_name;
}

void input_file(ofstream& file) {
	TManager my_case;
	int num;
	cout << "How many cases do you have today: ";
	cin >> num; cin.ignore(32767, '\n');
	if (num <= 0) {
		cout << "Today you have a chill day. Have a rest :)";
		exit(0);
	}
	for (int i = 0; i < num; i++)
	{
		string time;
		cout << "Name: "; cin.getline(my_case.name, 50);
		cout << "Start hour (HH:MM): "; getline(cin, time);
		get_time(time, my_case.start_hours, my_case.start_minutes);
		cout << "Duration (HH:MM): "; getline(cin, time);
		get_time(time, my_case.duration_hours, my_case.duration_minutes);
		file.write((char*)&my_case, sizeof(TManager));
	}
	file.close();
}

void get_time(string time, int& hours, int& minutes) {
	if (time.empty())
```

```
	{
		cerr << "Incorrectly entered data";
		exit(0);
	}
	int find_symb;
	find_symb = time.find(":");
	if (find_symb == -1)
	{
		cerr << "Incorrectly entered data";
		exit(0);
	}
	hours = stoi(time.substr(0, find_symb));
	minutes = stoi(time.substr(find_symb + 1, 2));
	if (minutes >= 60 || hours >= 24 || minutes < 0 || hours < 0)
	{
		cerr << "Incorrectly entered time";
		exit(0);
	}
}

void file_output(string name) {
	ifstream file(name, ios::binary);
	TManager my_case;
	cout << "\n==========////==========\n\n" << name << '\n';
	while (file.read((char*)&my_case, sizeof(TManager)))
	{
		cout << "\nName of occasion: " << my_case.name << endl;

		time_out("Starts at", my_case.start_hours, my_case.start_minutes);
		time_out("Duration", my_case.duration_hours, my_case.duration_minutes);

		int end_hours = my_case.start_hours + my_case.duration_hours;
		int end_minutes = my_case.start_minutes + my_case.duration_minutes;

		if (end_minutes >= 60) {
			end_minutes -= 60;
			end_hours++;
		}
		if (end_hours >= 24) {
			end_hours %= 24;
		}
		time_out("End time", end_hours, end_minutes);
	}
	cout << "\n==========////==========\n\n";
	file.close();
}

void time_out(string occasion, int hours, int minutes) {
	cout << occasion << ((hours < 10) ? ": 0" : ": ") << hours;
	cout << ((minutes < 10) ? ":0" : ":") << minutes << endl;
}

void the_nearest_case(string file_name) {

	struct tm current_time;
	time_t t = time(0);
	localtime_s(&current_time, &t);

	ifstream file(file_name, ios::binary);
	TManager my_case;
```

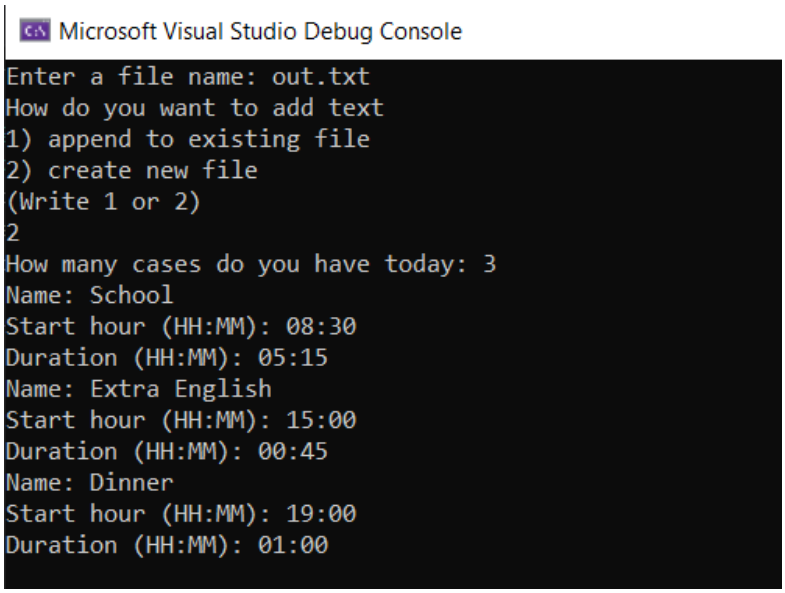
```
	char nearest_case[50];
	int current_minutes = current_time.tm_hour * 60 + current_time.tm_min;
	int nearest_minutes = 1440; // максимальна кількість хвилин в добі

	while (file.read((char*)&my_case, sizeof(TManager)))
	{
		int temp_minutes = my_case.start_hours * 60 + my_case.start_minutes;
		if (current_minutes <= temp_minutes && nearest_minutes >= temp_minutes) {
			nearest_minutes = temp_minutes;
			strcpy_s(nearest_case, my_case.name);
		}
	}

	if (nearest_minutes != 1440) {
		cout << "Your next occasion is - " << nearest_case;
		time_out(" and it starts at", nearest_minutes / 60, nearest_minutes % 60);
	}
	else
		cout << "All your occasions is over. Have a rest!\n" << endl;
	file.close();
}

string free_time(string name_out) {
	ofstream file_in;
	string name_in = create_file(file_in);
	ifstream file_out(name_out, ios::binary);
	recursion(file_in, file_out, 780, 1440, name_out);
	file_in.close();
	return name_in;
}


void recursion(ofstream& file_in, ifstream& file_out, int upper_border, int lower_border,
string name_out) {
	TManager case_out;
	if (!file_out.read((char*)&case_out, sizeof(TManager)))
	{
		file_out.close();
		TFreeTime t;
		t.start_minutes = upper_border;
		t.end_minutes = lower_border;
		t.duration = lower_border - upper_border;
		file_in.write((char*)&t, sizeof(TFreeTime));
		return;
	}
	else {
		int start_time = case_out.start_hours * 60 + case_out.start_minutes;
		int end_time = start_time + case_out.duration_hours * 60 +
case_out.duration_minutes;
		if (start_time <= upper_border && end_time < lower_border && end_time >
upper_border)
		{
			upper_border = end_time;
			recursion(file_in, file_out, upper_border, lower_border, name_out);
		}
		else if (start_time > upper_border && end_time >= lower_border && start_time
< lower_border)
		{
			lower_border = start_time;
			recursion(file_in, file_out, upper_border, lower_border, name_out);
		}
```

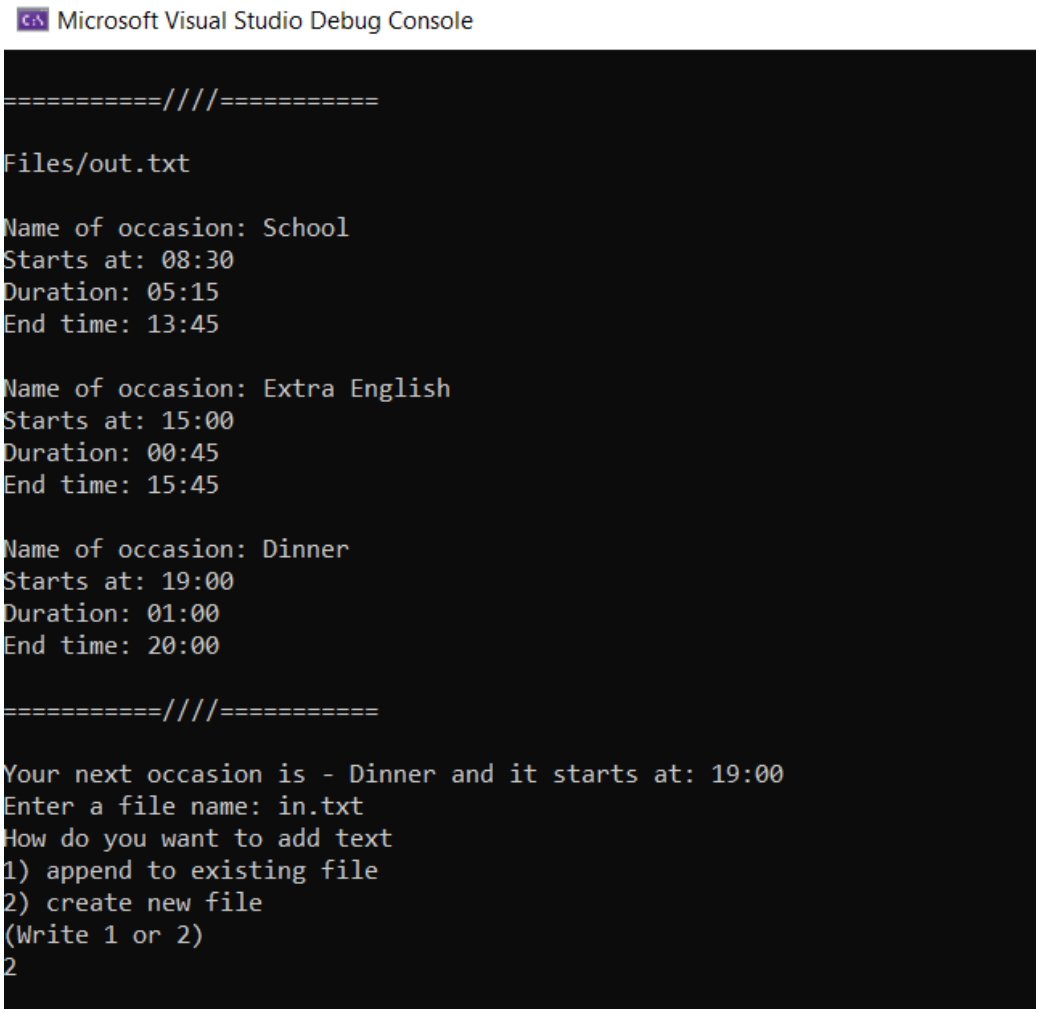
```
		else if (start_time > upper_border && end_time < lower_border)
		{
			int position = file_out.tellg();
			recursion(file_in, file_out, upper_border, start_time, name_out);
			ifstream file_out2(name_out, ios::binary);
			file_out2.seekg(position, ios::beg);
			recursion(file_in, file_out2, end_time, lower_border, name_out);
		}
		else if ((start_time < upper_border && end_time <= upper_border) ||
(start_time >= lower_border && end_time > lower_border))
		{
			recursion(file_in, file_out, upper_border, lower_border, name_out);
		}
		else if (start_time <= upper_border && end_time >= lower_border)
			return;
		else {
			cout << "Something went wrong :(\nCheck your input data" << endl;
			exit(0);
		}
	}
}

void b_file_output(string name) {
	ifstream file(name, ios::binary);
	TFreeTime time;
	cout << "\n==========////==========\n\n" << name << '\n';
	while (file.read((char*)&time, sizeof(TFreeTime)))
	{
		time_out("\nFree time starts at", time.start_minutes / 60, time.start_minutes
% 60);
		time_out("Ends at", time.end_minutes / 60, time.end_minutes % 60);
		time_out("Duration", time.duration / 60, time.duration % 60);
	}
	cout << "\n==========////==========\n\n";
	file.close();
}
```

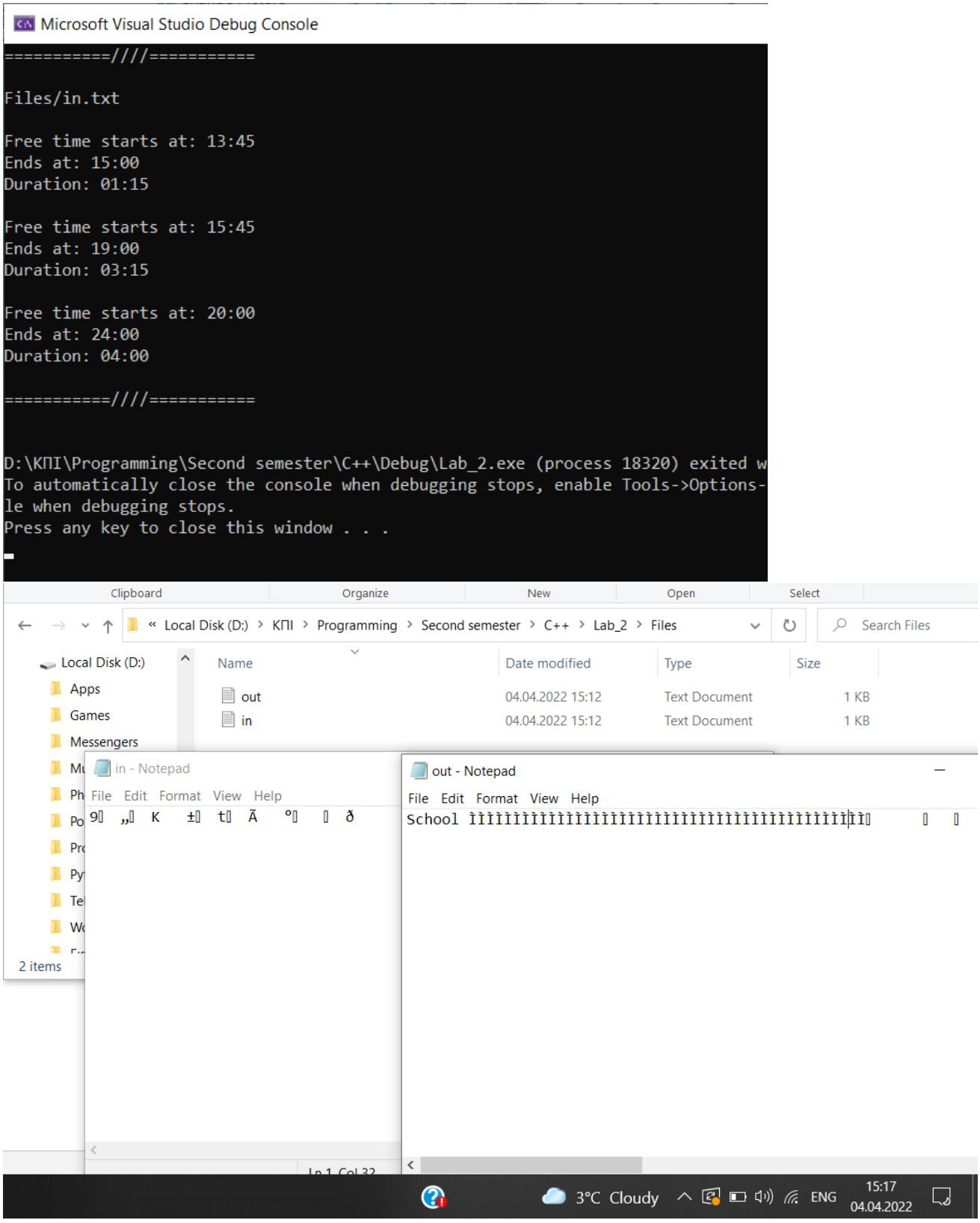
// lib.h

```
#ifndef FUNC_H
#define FUNC_H

#include <iostream>
#include <iomanip>
#include <string>
#include <fstream>
#include <stdlib.h>
#include <stdio.h>
#include <time.h>
using namespace std;

struct TManager {
	char name[50];  // назва події
	int start_hours;    // година початку
	int start_minutes;  // хвилини початку
	int duration_hours;   // скільки триває годин
	int duration_minutes;   // скільки триває хвилин
};

struct TFreeTime {
	int start_minutes;
	int end_minutes;
	int duration;
};

void input_file(ofstream& file);
void file_output(string name);
void b_file_output(string name);
void get_time(string time, int& hours, int& minutes);
void time_out(string occasion, int hours, int minutes);
void the_nearest_case(string file_name);
string create_file(ofstream& file);
string free_time(string name_out);
void recursion(ofstream& file_in, ifstream& file_out, int upper_border, int lower_border,
string name_out);

#endif
```

Тестування програми:






Microsoft Visual Studio Debug Console
==========////==========
Files/in.txt
Free time starts at: 13:45
Ends at: 15:00
Duration: 01:15
Free time starts at: 15:45
Ends at: 19:00
Duration: 03:15
Free time starts at: 20:00
Ends at: 24:00
Duration: 04:00
==========////==========
D:\КПІ\Programming\Second semester\C++\Debug\Lab_2.exe (process 18320) exited w
To automatically close the console when debugging stops, enable Tools->Options-
le when debugging stops.
Press any key to close this window . . .
Clipboard
Organize
New
Open
Select
Local Disk (D:) › КПІ › Programming › Second semester › C++ › Lab_2 › Files
Search Files
Local Disk (D:)
Apps
Games
Messengers
Name
Date modified
Type
Size
out
04.04.2022 15:12
Text Document
1 KB
in
04.04.2022 15:12
Text Document
1 KB
2 items
in - Notepad
File Edit Format View Help
out - Notepad
File Edit Format View Help
School
3°C Cloudy
ENG
15:17
04.04.2022

2. Виконання завдання на мові Python:

// Lab_1.py

```
from lib import *

name_out = "Files/" + input("Enter a file name: ")

output_file = create_file(name_out)
input_file(output_file)
file_output(name_out)
the_nearest_case(name_out)

name_in = "Files/" + input("Enter a file name: ")
free_time(name_out, name_in)
b_file_output(name_in)
```

// lib.py

```
import pickle
from datetime import datetime


def create_file(file_name):
    print("How do you want to add information\n1) append to existing file \n2) create new
file")
    while (True):
        how_open = int(input("Choose 1 or 2: "))
        if how_open == 1:
            file = open(file_name, "ab")
            break
        elif how_open == 2:
            file = open(file_name, "wb")
            break
        else:
            print("Incorrect input. Try again")
    return file

def get_time(string):
    if (not string):
        print("Incorrectly entered data")
        exit(0)
    find_symb = string.find(":")
    if find_symb == -1:
        print("Incorrectly entered data")
        exit(0)
    hours = int(string[0:find_symb])
    minutes = int(string[find_symb + 1:])
    if (minutes >= 60 or hours >= 24 or minutes < 0 or hours < 0):
        print("Incorrectly entered data")
        exit(0)
    return (hours * 60 + minutes)


def input_file(file):
    times = int(input("How many cases do you have today: "))
    if times < 0:
        print("Today you have a chill day. Have a rest :)")
```

```
            exit(0)
        for i in range(times):
            name = input("Name: ")
            start = input("Start at (HH:MM): ")
            start_minutes = get_time(start)
            duration = input("Duration (HH:MM): ")
            duration_minutes = get_time(duration)
            case = {
                "Name": name,
                "Start": start_minutes,
                "Duration": duration_minutes
            }
            pickle.dump(case, file)
        file.close()

    def time_out(string, hours, minutes):
        print(string, ": 0" if (hours < 10) else ": ", hours, end = '', sep = '')
        print(":0" if (minutes < 10) else ":", minutes, sep = '')

    def file_output(file_name):
        print('\n', "-" * 20, sep = '')
        print('\n', file_name, '\n', sep = '')

        with open(file_name, 'rb') as file:
            size_file = file.seek(0, 2)
            file.seek(0)

            while file.tell() < size_file:
                case = pickle.load(file)
                print("Name of occasion:", case['Name'])
                time_out("Starts at", case['Start'] // 60, case['Start'] % 60)
                time_out("Duration", case['Duration'] // 60, case['Duration'] % 60)
                end_time = case['Start'] + case['Duration']
                time_out("End time", end_time // 60, end_time % 60)
                print()

        print("-" * 20, '\n')

    def the_nearest_case(file_name):
        current_time = datetime.now()
        current_minutes = current_time.hour * 60 + current_time.minute
        nearest_minutes = 1440
        nearest_case = "All your occasions is over. Have a rest!\n"

        with open(file_name, 'rb') as file:
            size_file = file.seek(0, 2)
            file.seek(0)

            while file.tell() < size_file:
                new_case = pickle.load(file)
                temp_minutes = new_case['Start']

                if current_minutes <= temp_minutes and nearest_minutes >= temp_minutes:
                    nearest_minutes = temp_minutes
                    nearest_case = new_case['Name']
            if nearest_minutes != 1440:
                print("Your next occasion is -", nearest_case, end = '')
                time_out(" and it starts at", nearest_minutes // 60, nearest_minutes % 60)
                print()
            else:
                print(nearest_case)
```

```
def free_time(name_out, name_in):
    file_out = open(name_out, 'rb')
    size_file = file_out.seek(0, 2)
    file_out.seek(0)
    file_in = create_file(name_in)
    recursion(file_out, file_in, 780, 1440, 0, size_file)
    file_in.close()
    file_out.close()


def recursion(file_out, file_in, upper_border, lower_border, temp_byte, size_file):
    if temp_byte >= size_file:
        time = {
            'Start': upper_border,
            'End': lower_border,
            'Duration': lower_border - upper_border
        }
        pickle.dump(time, file_in)
    else:
        case = pickle.load(file_out)
        temp_byte = file_out.tell()
        start_time = case['Start']
        end_time = case['Start'] + case['Duration']

        if start_time <= upper_border and end_time < lower_border and end_time >
upper_border:
            upper_border = end_time
            recursion(file_out, file_in, upper_border, lower_border, temp_byte, size_file)

        elif start_time > upper_border and end_time >= lower_border and start_time <
lower_border:
            lower_border = start_time
            recursion(file_out, file_in, upper_border, lower_border, temp_byte, size_file)

        elif start_time > upper_border and end_time < lower_border:
            position = file_out.tell()
            recursion(file_out, file_in, upper_border, start_time, temp_byte, size_file)
            file_out.seek(position)
            recursion(file_out, file_in, end_time, lower_border, temp_byte, size_file)

        elif (start_time < upper_border and end_time <= upper_border) or (start_time >=
lower_border and end_time > lower_border):
            recursion(file_out, file_in, upper_border, lower_border, temp_byte, size_file)

        elif start_time <= upper_border and end_time >= lower_border:
            return

        else:
            print("Something went wrong :(\nCheck your input data")
            exit(0)

def b_file_output(file_name):
    print('\n', "-" * 20, sep = '')
    print('\n', file_name, '\n', sep = '')

    with open(file_name, 'rb') as file:
        size_file = file.seek(0, 2)
        file.seek(0)

        while file.tell() < size_file:
```

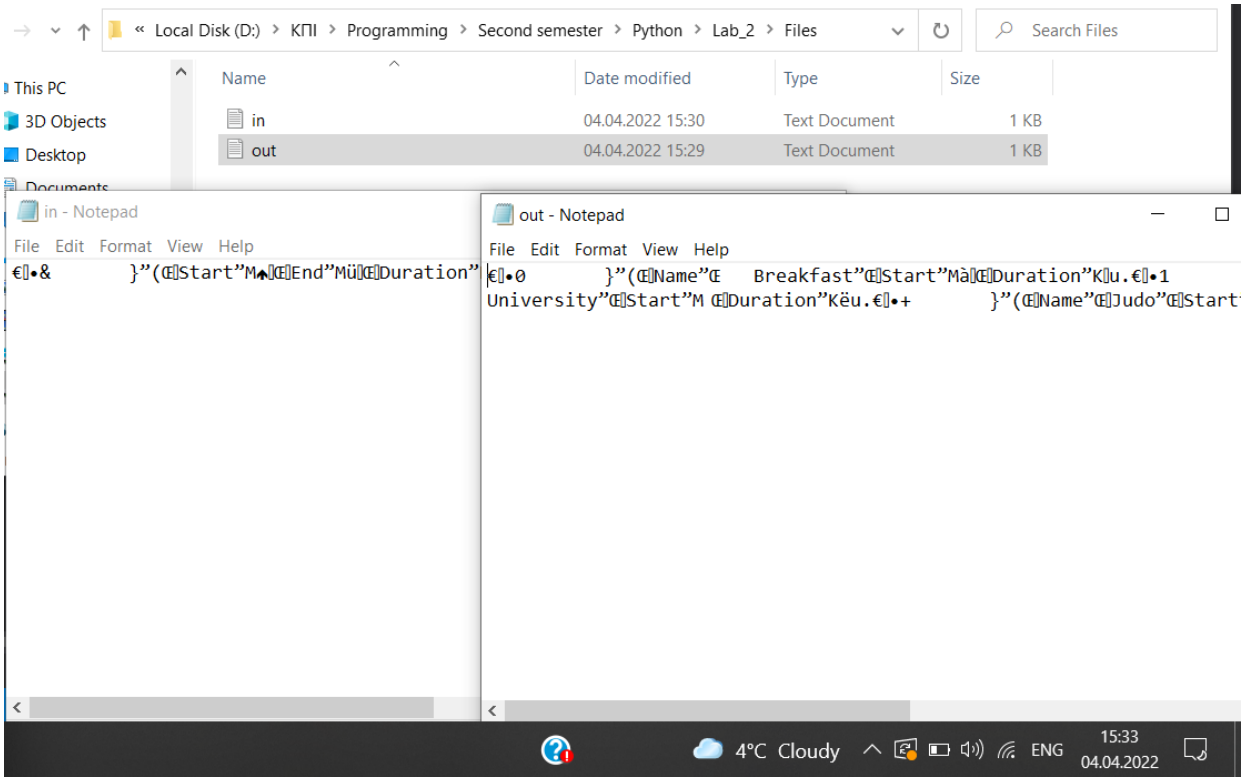
```
            time = pickle.load(file)
            time_out("Free time starts at", time['Start'] // 60, time['Start'] % 60)
            time_out("Ends at", time['End'] // 60, time['End'] % 60)
            time_out("Duration", time['Duration'] // 60, time['Duration'] % 60)
            print()

    print("-" * 20, '\n')
```

Тестування програми:

```
D:\Python\python.exe
Enter a file name: out.txt
How do you want to add information
1) append to existing file
2) create new file
Choose 1 or 2: 2
How many cases do you have today: 4
Name: Breakfast
Start at (HH:MM): 08:00
Duration (HH:MM): 00:15
Name: University
Start at (HH:MM): 09:00
Duration (HH:MM): 03:55
Name: Judo
Start at (HH:MM): 17:00
Duration (HH:MM): 01:30
Name: Dinner
Start at (HH:MM): 20:00
Duration (HH:MM): 00:30

--------------------

Files/out.txt

Name of occasion: Breakfast
Starts at: 08:00
Duration: 00:15
End time: 08:15

Name of occasion: University
Starts at: 09:00
Duration: 03:55
End time: 12:55

Name of occasion: Judo
Starts at: 17:00
Duration: 01:30
End time: 18:30

Name of occasion: Dinner
Starts at: 20:00
Duration: 00:30
End time: 20:30

--------------------

Your next occasion is - Judo and it starts at: 17:00
```

D:\Python\python.exe

```
Enter a file name: in.txt
How do you want to add information
1) append to existing file
2) create new file
Choose 1 or 2: 2

--------------------

Files/in.txt

Free time starts at: 13:00
Ends at: 17:00
Duration: 04:00

Free time starts at: 18:30
Ends at: 20:00
Duration: 01:30

Free time starts at: 20:30
Ends at: 24:00
Duration: 03:30

--------------------

Press any key to continue . . .
```



Висновок: Під час виконання лабораторної роботи я вивчив особливості створення і обробки бінарних файлів даних на прикладі мов C++ та Python. Результатом виконання лабораторної роботи є програми, написані на вищевказаних мовах, основним завданням яких є створення графіку робочого дня і запису вільного часу до файлу. Після тестування програм можна зробити висновок, що вони справляються із поставленою задачею.