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 Вєчерковська 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 С++:*

**// 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**

#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);

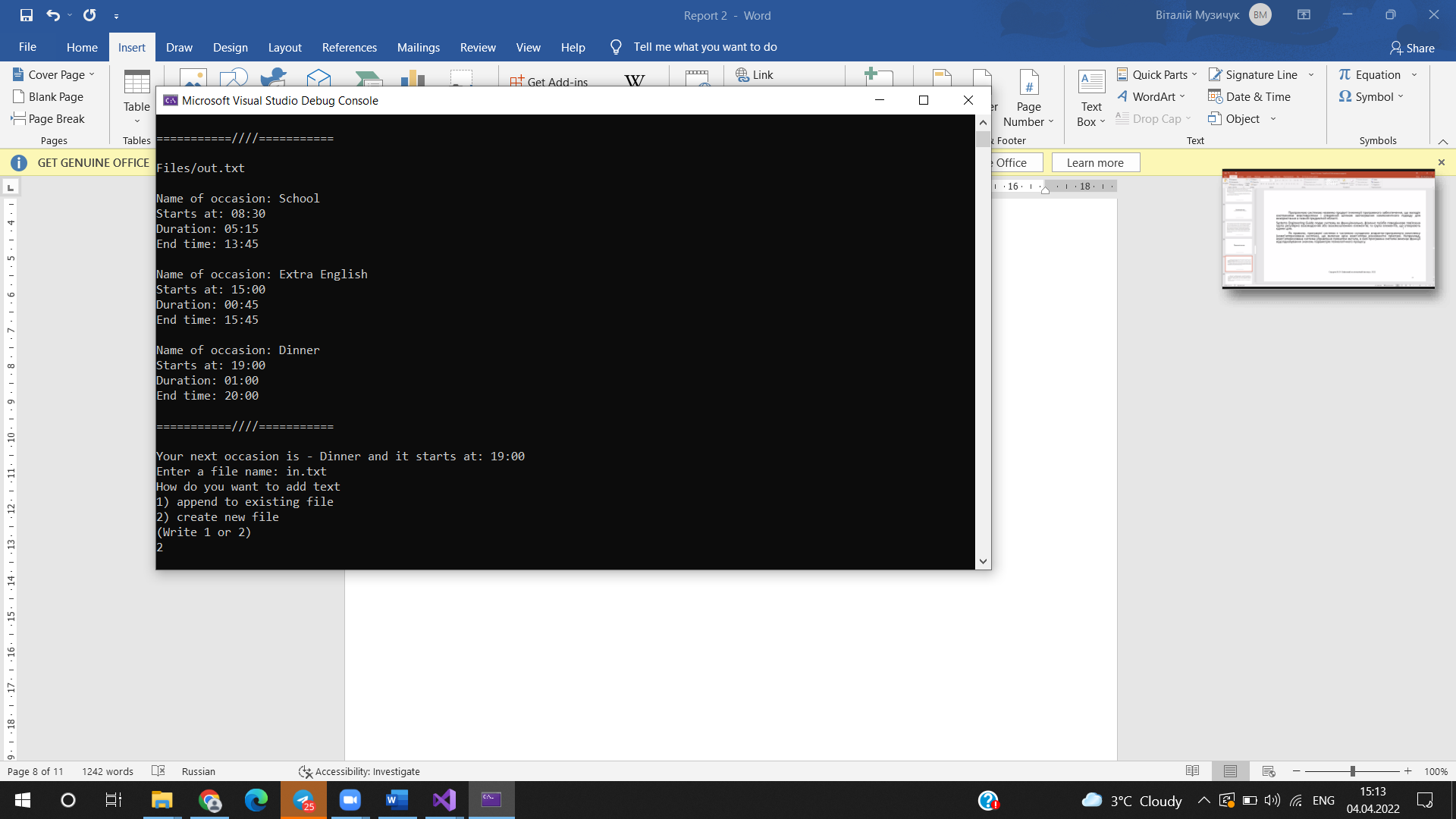
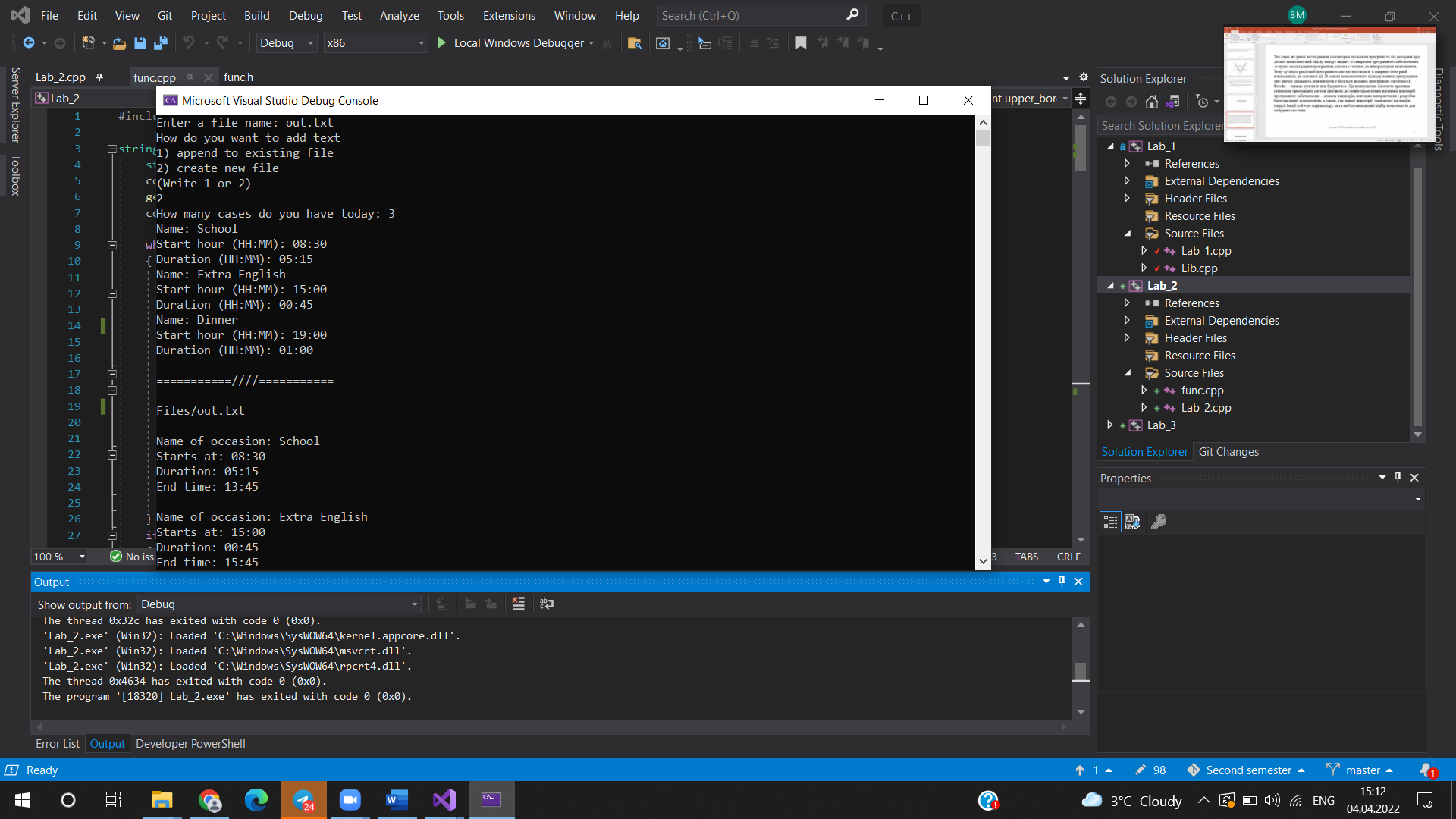
string create\_file(ofstream& file);

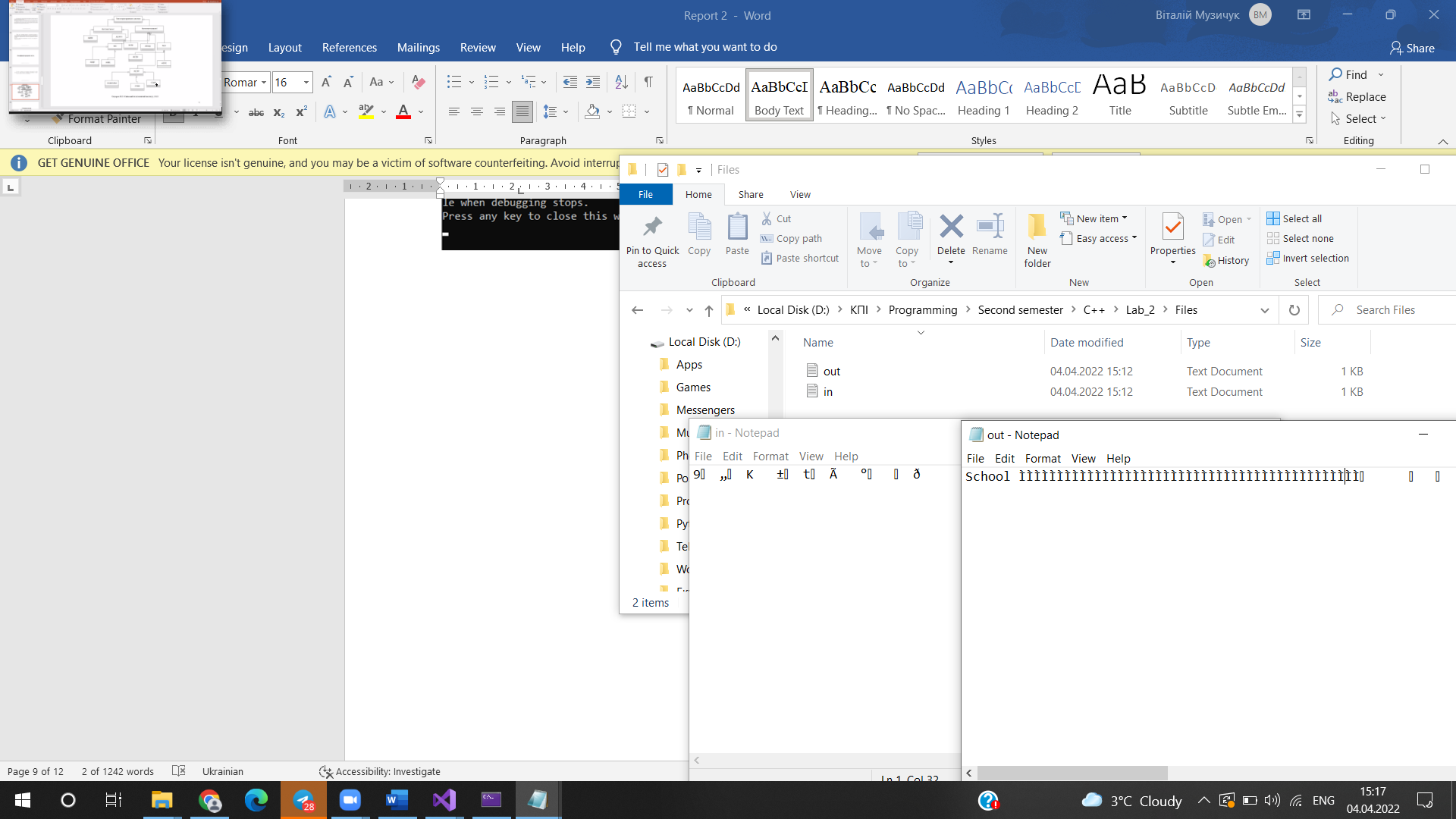
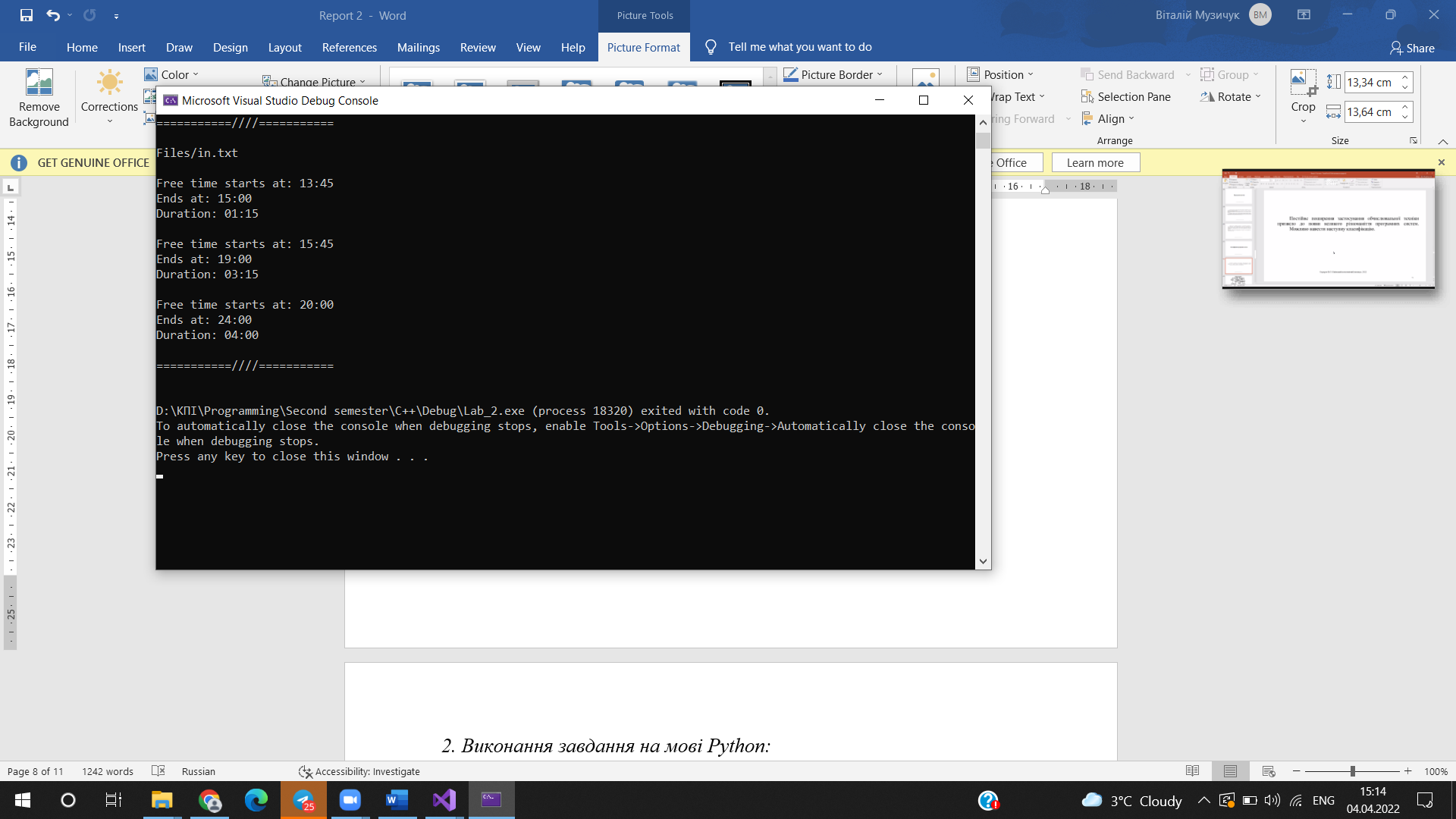
string free\_time(string name\_out);

void recursion(ofstream& file\_in, ifstream& file\_out, int upper\_border, int lower\_border, string name\_out);

#endif

**Тестування програми:**





*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)

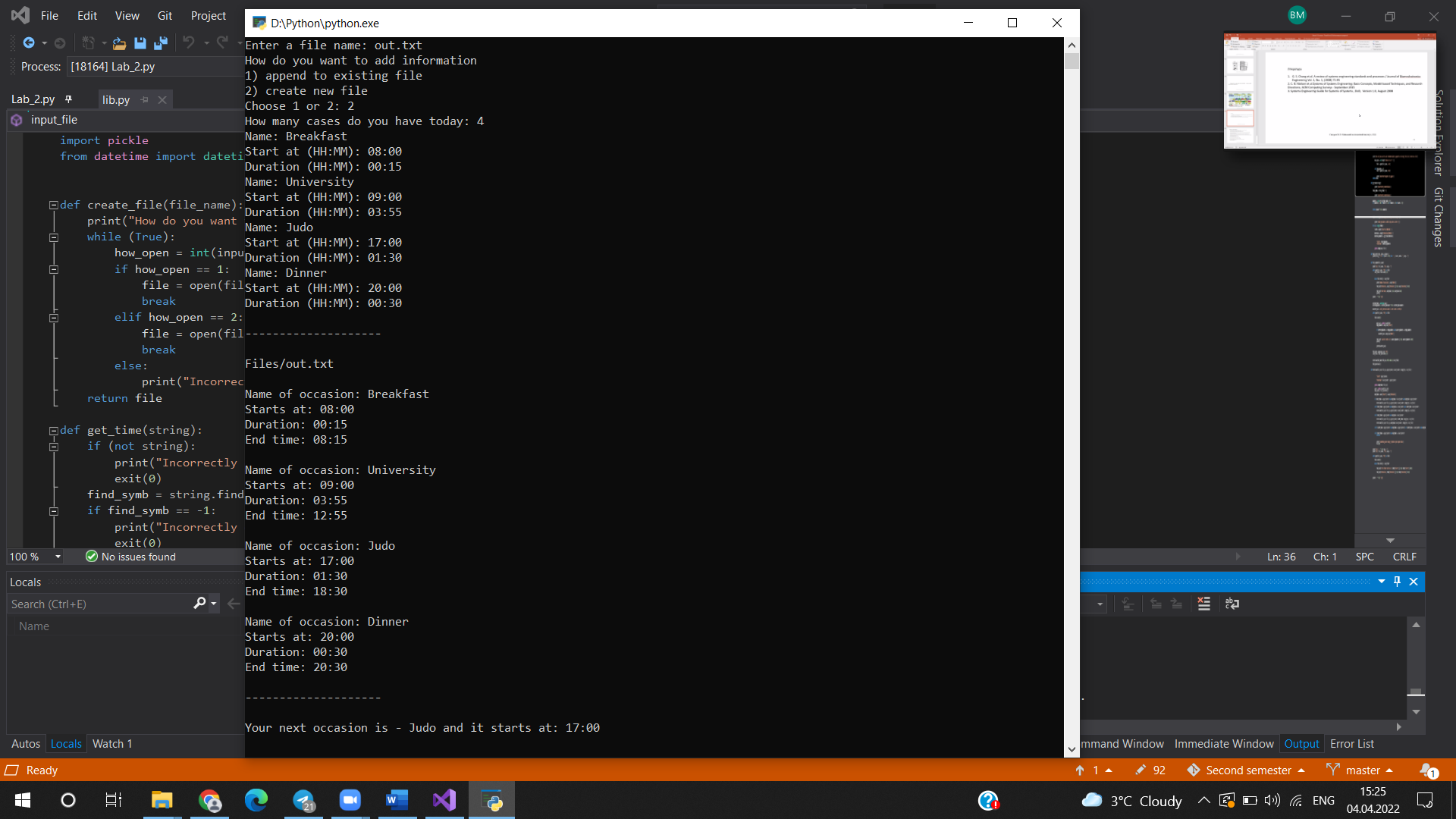
time\_out("Ends at", time['End'] // 60, time['End'] % 60)

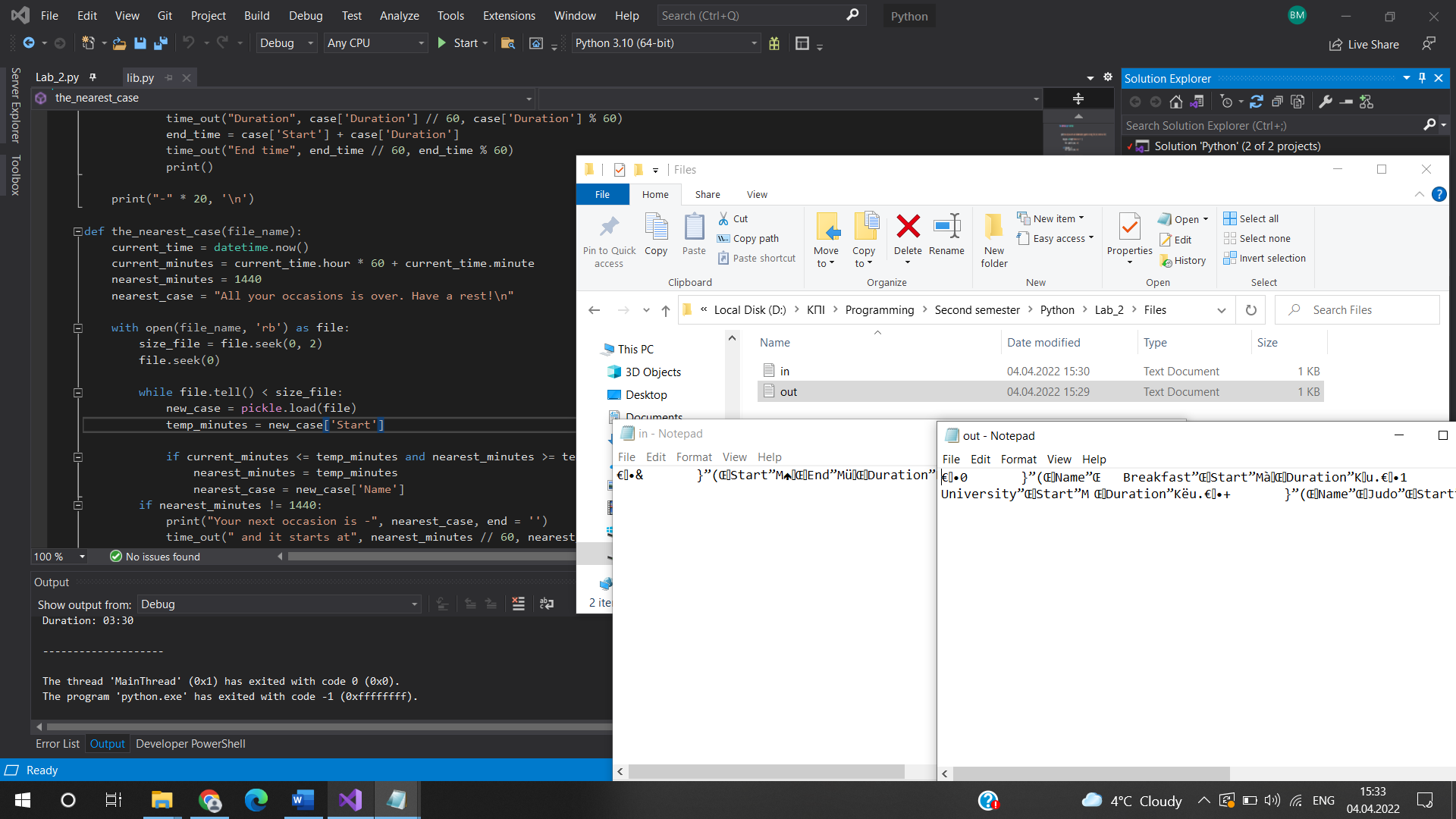
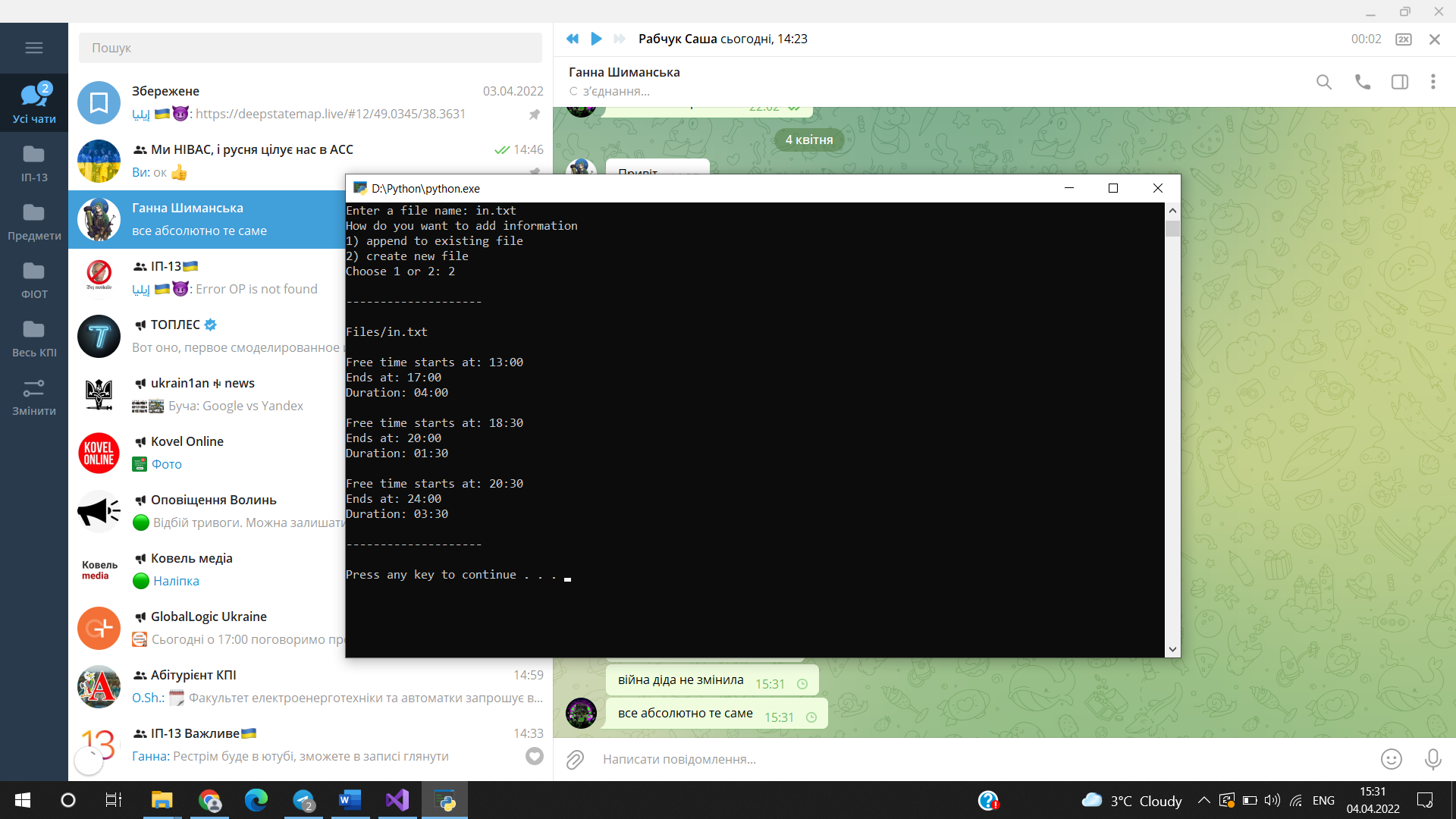
time\_out("Duration", time['Duration'] // 60, time['Duration'] % 60)

print()

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

**Тестування програми:**





**Висновок**: Під час виконання лабораторної роботи я вивчив особливості створення і обробки бінарних файлів даних на прикладі мов С++ 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.