# 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»

Варіант 13

Виконав студент Котков Тимур Максимович

(шифр, прізвище, ім'я, по батькові)

Перевірив Вітковська Ірина Іванівна

( прізвище, ім'я, по батькові)

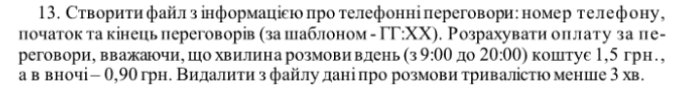
Київ 2022

**Лабораторна робота №2**

**Бінарні файли**

**Мета –** вивчити особливості створення і обробки бінарних файлів даних.

**Індивідуальне завдання:**



**Постановка задачі**

Створимо бінарний файл з інформацією про телефонні переговори, додаючи до часу початку переговорів по хвилині, доки час не буде рівним кінцевому, порахуємо вартість переговорів з урахуванням тарифів. Розмови коротші за 3хв видалимо з файлу.

**Код на С++ :**

**Main:**

#include <iostream>

#include <fstream>

#include <vector>

#include "Header.h"

using namespace std;

int main() {

string first\_f = "calls.txt", second\_f = "changed\_calls.txt";

int num\_of\_calls = 0;

create\_list(first\_f, num\_of\_calls);

cout << "\nResult of writing first edition of file is: " << endl;

show\_res(first\_f, num\_of\_calls);

delete\_cols(first\_f, second\_f, num\_of\_calls);

cout << "\nResult of changing in file is: " << endl;

show\_res(second\_f, num\_of\_calls);

}

**Source:**

#include <iostream>

#include <fstream>

#include <vector>

#include <cmath>

#include <sstream>

#include "Header.h"

using namespace std;

struct cl {

char tel\_number[10], start\_time[6], end\_time[6];

double price;

};

void show\_res(const string& file\_name, int num\_of\_calls) {

cl c;

ifstream file(file\_name, ios::binary);

for (int i = 0; i < num\_of\_calls; i++)

{

file.read((char\*)&c, sizeof(cl));

cout << "\nPhone number: +380" << c.tel\_number << endl;

cout << "Call started: " << c.start\_time << endl;

cout << "Call ended: " << c.end\_time << endl;

cout << "Price of call: " << c.price << endl;

}

}

double price\_for\_call(string start\_time, string end\_time) {

int hours\_s, mins\_s, hours\_e, mins\_e;

char extra;

std::stringstream ss\_start, ss\_end;

ss\_start << start\_time;

ss\_start >> hours\_s >> extra >> mins\_s >> extra;

ss\_end << end\_time;

ss\_end >> hours\_e >> extra >> mins\_e >> extra;

double diff = (hours\_e \* 60 + mins\_e) - (hours\_s \* 60 + mins\_s);

if (hours\_s >= 9 && hours\_s <= 20) {

return diff \* 1.5;

}

else {

return diff \* 0.9;

}

}

void delete\_cols(const string& file\_name\_1, const string& file\_name\_2, int& num\_of\_calls) {

ifstream file(file\_name\_1, ios::binary);

ofstream file2(file\_name\_2, ios::binary);

vector<cl> cls;

cl c;

int count = 0, iter = num\_of\_calls;

for (int i = 0; i < iter; i++)

{

file.read((char\*)&c, sizeof(cl));

count++;

int hours\_s, mins\_s, hours\_e, mins\_e;

char extra;

string st(c.start\_time, 5), et(c.end\_time, 5);

std::stringstream ss\_start, ss\_end;

ss\_start << st;

ss\_start >> hours\_s >> extra >> mins\_s >> extra;

ss\_end << et;

ss\_end >> hours\_e >> extra >> mins\_e >> extra;

double diff = (hours\_e \* 60 + mins\_e) - (hours\_s \* 60 + mins\_s);

if (diff <= 3) {

num\_of\_calls--;

continue;

}

else {

cls.push\_back(c);

}

}

for (int i = 0; i < cls.size(); i++)

{

c = cls[i];

file2.write((char\*)&c, sizeof(cl));

}

}

void create\_list(string file\_name, int& num\_of\_calls) {

vector<cl> calls;

ofstream file(file\_name, ios::binary);

string buf = "y";

bool is\_ok = true;

while (buf == "y") {

is\_ok = true;

num\_of\_calls++;

cl c;

cout << "Client " << num\_of\_calls << endl;

cout << "Telephone number +380";

cin >> c.tel\_number;

cin.ignore();

cout << "Start time (HH:MM): ";

cin >> c.start\_time;

cin.ignore();

cout << "End time (HH:MM): ";

cin >> c.end\_time;

cin.ignore();

string st(c.start\_time, 5), et(c.end\_time, 5);

c.price = price\_for\_call(st, et);

if (c.price < 0) {

cout << "Mismatch in time!" << endl;

is\_ok = false;

num\_of\_calls--;

}

if (is\_ok) {

file.write((char\*)&c, sizeof(cl));

calls.push\_back(c);

}

cout << "Add another client? y/n: ";

cin >> buf;

cin.ignore();

}

file.close();

}

**Header:**

#pragma once

void create\_list(std::string file\_name, int& num\_of\_calls);

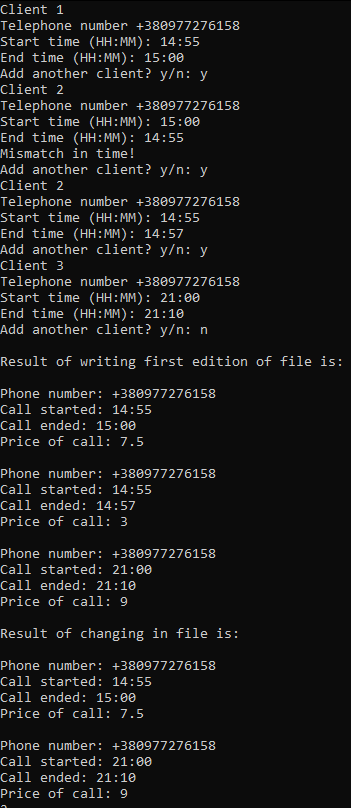
double price\_for\_call(std::string start\_time, std::string end\_time);

void delete\_cols(const std::string& file\_name\_1, const std::string& file\_name\_2, int& num\_of\_calls);

void show\_res(const std::string& file\_name, int num\_of\_calls);

struct cl;

**Результати у С++:**

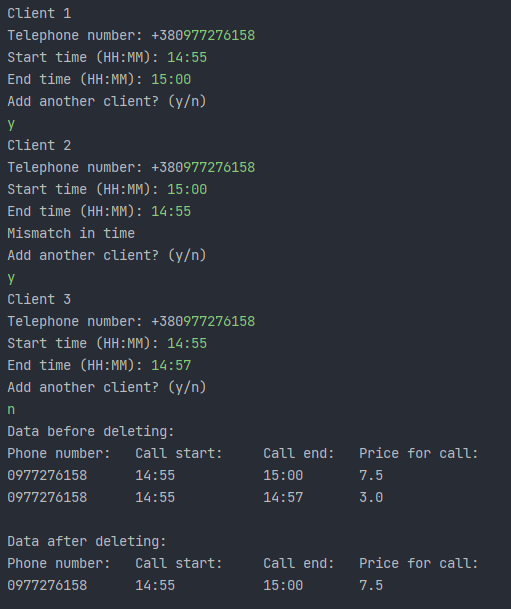
****

**}**

**Код на Python :**

*import* pickle  
  
  
*class* Call:  
 *def* \_\_init\_\_(*self*, tel\_number, start\_time, end\_time, price):  
 *self*.tel\_number = tel\_number  
 *self*.start\_time = start\_time  
 *self*.end\_time = end\_time  
 *self*.price = price  
  
  
*def* show\_res():  
 print('{:15}\t{:12}\t{:10}\t{:10}\t'.format('Phone number:', 'Call start:', 'Call end:', 'Price for call:'))  
 *with* open('calls.txt', 'rb') *as* file:  
 bin\_data = file.read()  
 clients = pickle.loads(bin\_data)['calls']  
 *for* client *in* clients:  
 print('{:15}\t{:12}\t{:10}\t{:<10}\t'  
 .format(f'0{client.tel\_number}', client.start\_time, client.end\_time, client.price))  
  
  
*def* price\_for\_call(start, end, price\_day, price\_night):  
 buff = [start.split(':'), end.split(':')]  
 start\_min = int(buff[0][0]) \* 60 + int(buff[0][1])  
 end\_min = int(buff[1][0]) \* 60 + int(buff[1][1])  
 diff = end\_min - start\_min  
 *if* 9 < int(buff[0][0]) < 20:  
 *return* diff \* price\_day  
 *else*:  
 *return* diff \* price\_night  
  
  
*def* delete\_cols():  
 *with* open('calls.txt', 'rb') *as* file:  
 bin\_data = file.read()  
 calls = pickle.loads(bin\_data)['calls']  
 buff = calls[::]  
 *for* i *in* range(len(calls)):  
 temp = [calls[i].start\_time.split(':'), calls[i].end\_time.split(':')]  
 start\_min = int(temp[0][0]) \* 60 + int(temp[0][1])  
 end\_min = int(temp[1][0]) \* 60 + int(temp[1][1])  
 diff = end\_min - start\_min  
 *if* diff <= 3:  
 buff.pop(i)  
  
 *with* open('calls.txt', 'wb') *as* file:  
 pickle.dump({'calls': buff}, file)  
  
  
*def* create\_list():  
 calls = []  
 *with* open('calls.txt', 'wb') *as* file:  
 buf = 'y'  
 count = 1  
 *while* buf == 'y':  
 is\_ok = *True* print(f'Client {count}')  
 print('Telephone number: +380', end='')  
 tel\_number = input()  
 print('Start time (HH:MM): ', end='')  
 start\_time = input()  
 print('End time (HH:MM): ', end='')  
 end\_time = input()  
 price = price\_for\_call(start\_time, end\_time, 1.5, 0.9)  
 c = Call(tel\_number, start\_time, end\_time, price)  
 *if* price <= 0:  
 is\_ok = *False* print('Mismatch in time')  
 *if* is\_ok:  
 calls.append(c)  
 print('Add another client? (y/n)')  
 buf = input()  
 count += 1  
 pickle.dump({'calls': calls}, file)  
  
  
*if* \_\_name\_\_ == '\_\_main\_\_':  
 create\_list()  
 print('Data before deleting:')  
 show\_res()  
 print('\nData after deleting:')  
 delete\_cols()  
 show\_res()

**Скріншот результатів програми на Python :**

****

**Висновок:**

Під час лабораторної роботи ми вивчили особливості створення і обробки бінарних файлів даних на двох мовах програмування: C++ та Python. Оскільки програма правильно створює та виводить бінарні файли, обчислює ціну і довжину переговорів, видаляє з файлу дані про розмови коротші за 3хв, а також має вірний вивід, то вона працює правильно.