Додаток 1

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. Метидології програмування»

«Бінарні файли»

Варіант 23

Виконав студент ІП-13 Недельчев Євген Олександрович
(шифр, прізвище, ім'я, по батькові)

Перевірив Вєчерковська Анастасія Сергіївна
(прізвище, ім'я, по батькові)

Київ 2022

Лабораторна робота 1

Варіант 23

23. Створити файл із списком клієнтів, обслужених менеджером протягом дня: прізвище, час приходу та час закінчення обслуговування. При введенні даних перевіряти їхню допустимість (чи не перетинаються клієнти з наявними). Створити новий файл, який містить інформацію про клієнтів, з якими менеджер спілкувався понад 30 хв.

Код програми

C++

main.cpp

```
#include "Header.h"

int main() {
	const char* FName1 = "abc.dat";
	const char* FName2 = "dce.dat";
	CreateFile(FName1);
	cout << FName1 << ":\n";
	PrintFile(FName1);
	CreateNewFile(FName1, FName2);
	cout << FName2 << ":\n";
	PrintFile(FName2);
}
```

header.h

```
#pragma once
#include <iostream>
#include <fstream>
#include <string>
#include <vector>

using namespace std;


void CreateNewFile(const char* FName1, const char* FName2);
void CreateFile(const char* FName);
void PrintFile(const char* FName);
vector<string> split(string line, char sep = ' ');
```

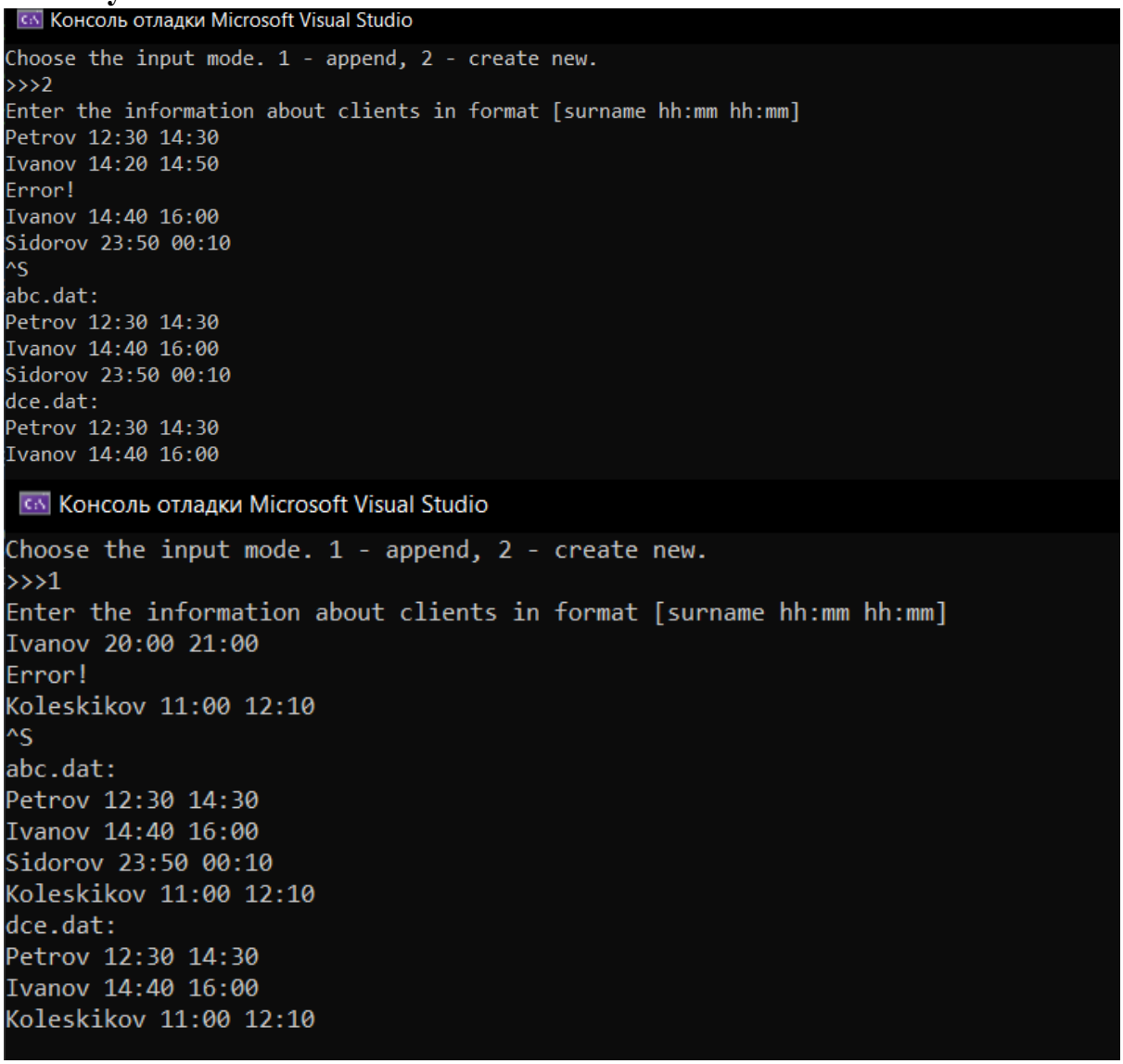
header.cpp

```
#include "header.h"

struct Client
{
    string name, str;
    int start_time, end_time, duration_time;
    Client(string line) {
        str = line + '\n';
        vector <string> words = split(line);
        name = words[0];
        start_time = stoi(split(words[1], ':')[0]) * 60 + stoi(split(words[1], ':')[1]);
        end_time = stoi(split(words[2], ':')[0]) * 60 + stoi(split(words[2], ':')[1]);
        if (end_time < start_time) {
            end_time += 1440;
        }
        duration_time = end_time - start_time;
    }
};

bool is_client_in(Client a, vector <Client> base) {
    for (int i = 0; i < base.size(); i++) {
        if (base[i].name == a.name) return true;
        else if (base[i].end_time > a.end_time && a.end_time > base[i].start_time) return
true;
        else if (a.start_time < base[i].start_time && a.end_time > base[i].end_time)
return true;
        else if (base[i].start_time < a.start_time && a.start_time < base[i].end_time)
return true;
    }
    return false;
}

void CreateFile(const char* FName) {
    cout << "Choose the input mode. 1 - append, 2 - create new.\n>>>";
    int input_mode; cin >> input_mode;
    string str = "";
    vector <Client> base;
    while (true) {
        if (input_mode == 1) {
            ifstream inf(FName, ios::binary);
            char symbol;
            while (inf.read((char*)&symbol, sizeof(char)))
            {
                if (symbol == '\n') {
                    base.push_back(Client(str));
                    str = "";
                }
                else str += symbol;
            }
            inf.close();
            break;
        }
        else if (input_mode == 2) {
            break;
        }
        else {
            cout << "Wrong input format!\n>>>";
            cin >> input_mode;
        }
    }
```

```
    ofstream ouf(FName, ios::binary);
    cout << "Enter the information about clients in format [surname hh:mm hh:mm]\n";
    cin.ignore();
    getline(cin, str);
    while (str[0] != 19) {
        Client temp(str);
        if (base.size() == 0) {
            base.push_back(temp);
            getline(cin, str);
            continue;
        }
        if (!is_client_in(temp, base)) base.push_back(temp);
        else cout << "Error!" << endl;
        getline(cin, str);
    }
    for (int i = 0; i < base.size(); i++) {
        ouf.write(base[i].str.c_str(), base[i].str.length());
    }
    ouf.close();
}

void PrintFile(const char* FName) {
       ifstream inf(FName, ios::binary);
       char symbol;
       while (inf.read((char*)&symbol, sizeof(char))) {
              cout << symbol;
       }
}

void CreateNewFile(const char* FName1, const char* FName2) {
    ifstream inf(FName1, ios::binary);
    ofstream ouf(FName2, ios::binary);
    vector <Client> base;
    char symbol;
    string line = "";
    while (inf.read((char*)&symbol, sizeof(char)))
    {
        if (symbol == '\n') {
            Client a(line);
            if ((a.end_time - a.start_time) > 30) {
                base.push_back(a);
            }
            line = "";
        }
        else line += symbol;
    }
    for (int i = 0; i < base.size(); i++) {
        ouf.write(base[i].str.c_str(), base[i].str.length());
    }
}

vector<string> split(string line, char sep) {
    vector<string> words;
    string temp_word = "";
    line += sep;
    for (int i = 0; i < line.length(); i++) {
        if (line[i] == sep) {
            if (temp_word.length() > 0) {
                words.push_back(temp_word);
            }
            temp_word = "";
        }
        else {
            temp_word += line[i];
```

```
            }
        }
        return words;
    }
```

Тестування:


Консоль отладки Microsoft Visual Studio
Choose the input mode. 1 - append, 2 - create new.
>>>2
Enter the information about clients in format [surname hh:mm hh:mm]
Petrov 12:30 14:30
Ivanov 14:20 14:50
Error!
Ivanov 14:40 16:00
Sidorov 23:50 00:10
^S
abc.dat:
Petrov 12:30 14:30
Ivanov 14:40 16:00
Sidorov 23:50 00:10
dce.dat:
Petrov 12:30 14:30
Ivanov 14:40 16:00
Консоль отладки Microsoft Visual Studio
Choose the input mode. 1 - append, 2 - create new.
>>>1
Enter the information about clients in format [surname hh:mm hh:mm]
Ivanov 20:00 21:00
Error!
Koleskikov 11:00 12:10
^S
abc.dat:
Petrov 12:30 14:30
Ivanov 14:40 16:00
Sidorov 23:50 00:10
Koleskikov 11:00 12:10
dce.dat:
Petrov 12:30 14:30
Ivanov 14:40 16:00
Koleskikov 11:00 12:10

Python

func.py

```
import pickle


def is_client_in(client: dict, base: list):
    for i in base:
        start_time1 = int(client['start_time'].split(':')[0]) * 60 + int(client['start_time'].split(':')[1])
        end_time1 = int(client['end_time'].split(':')[0]) * 60 + int(client['end_time'].split(':')[1])
        if end_time1 < start_time1: end_time1 += 1440
        start_time2 = int(i['start_time'].split(':')[0]) * 60 + int(i['start_time'].split(':')[1])
        end_time2 = int(i['end_time'].split(':')[0]) * 60 + int(i['end_time'].split(':')[1])
        if end_time2 < start_time2: end_time2 += 1440
        if client['surname'] == i['surname']:
            return True
        elif end_time2 > end_time1 > start_time2:
            return True
        elif start_time1 < start_time2 and end_time1 > end_time2:
            return True
        elif start_time2 < start_time1 < end_time2:
            return True
        else:
            return False
    return False


def create_file(file_name):
    input_mode = int(input('''Выберите режим ввода информации:
    1)Дозапись в существующий(если существует файл с таким именем)
    2)Создать новый файл\n'''))
    while True:
        if input_mode == 1:
            break
        elif input_mode == 2:
            file = open(file_name, "wb").close()
            break
        else:
            input_mode = input("Введите 1 или 2")
    lst = []
    try:
        with open(file_name, "rb") as file:  # получение данных из файла для дозаписи
            lst = pickle.load(file)
    except:
        pass
    with open(file_name, "wb") as file:
        line = input('''Вводите информацию о клиенте в формате [Фамилия ЧЧ:ММ ЧЧ:ММ].
Чтобы остановить ввод, введите пустую строку.\n''').split()
        while (line):
            client = {
                'surname': line[0],
                'start_time': line[1],
```

```
            'end_time': line[2],
        }
        if not is_client_in(client, lst):
            lst.append(client)
        else:
            print("Ошибка!")
        line = input().split()
    pickle.dump(lst, file)


def create_new_file(file_name1, file_name2):
    # try:
    with open(file_name1, "rb") as inf:
        text = pickle.load(inf)
        new_text = []
        for i in range(len(text)):
            start_time = int(text[i]['start_time'].split(':')[0]) * 60 + int(text[i]['start_time'].split(':')[1])
            end_time = int(text[i]['end_time'].split(':')[0]) * 60 + int(text[i]['end_time'].split(':')[1])
            if end_time < start_time: end_time += 1440
            if (end_time - start_time) > 30:
                new_text.append(text[i])
        with open(file_name2, "wb") as ouf:
            pickle.dump(new_text, ouf)


def print_file(file_name):
    try:
        with open(file_name, "rb") as file:
            print(f"=====Содержимое {file_name}=====")
            text = pickle.load(file)
            for i in text:
                for values in i.values():
                    print(values, end=' ')
                print()
            print()
    except:
        print('Ошибка при открытии файла!')
```

main.py

```
from func import *

FName1 = input("Введите имя файла: ")
create_file(FName1)
print_file(FName1)
FName2 = input("Введите имя нового файла: ")
create_new_file(FName1, FName2)
print_file(FName2)
```

Тестування:

```
Введите имя файла: a.bin
Выберите режим ввода информации:
    1)Дозапись в существующий(если существует файл с таким именем)
    2)Создать новый файл
2
Вводите информацию о клиенте в формате [Фамилия ЧЧ:ММ ЧЧ:ММ].
Чтобы остановить ввод, введите пустую строку.
Иванов 12:30 13:30
Сидоров 12:00 13:00
Ошибка!
Сидоров 12:00 12:30
Пупкин 08:00 08:20

=====Содержимое a.bin=====
Иванов 12:30 13:30
Сидоров 12:00 12:30
Пупкин 08:00 08:20

Введите имя нового файла: b.bin
=====Содержимое b.bin=====
Иванов 12:30 13:30
```

Висновки:

Я вивчив особливості створення і обробки бінарних файлів данних. Застосував ці навички на практиці.