

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 24

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

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

Лабораторна робота 2

Бінарні файли

Варіант 24

Створити файл зі списком працівників підприємства: прізвище, дата народження, дата прийому на роботу. Вивести список працівників, у яких день народження у поточному місяці та які пропрацювали на підприємстві не менше 5-ти років. Створити новий файл з інформацією про співробітників, які оформилися на роботу на дане підприємство у віці не старше 25-ти років та пропрацювали на ньому не менше 10-ти років.

Код програми

C++

lab_cpp_2.cpp

```
#include "Header.h"

int main()
{
    string first_file_path = "first file.txt";
    string second_file_path = "second file.txt";
    input_file(first_file_path);

    cout << "Birthday in this month AND work experience >= 5 years: " << endl;
    birthday_in_this_month(first_file_path);
    cout << endl;

    create_second_file(first_file_path, second_file_path);

    cout << "Start career in <= 25 yo AND work experience >= 10 years (second file):" <<
endl;
    output_file(second_file_path);

    return 0;
}
```

Header.h

```
#define _CRT_SECURE_NO_WARNINGS
#pragma once
#include <iostream>
#include <fstream>
#include <string>
#include <vector>
#include <ctime>

using namespace std;

struct date;
struct employee;

void input_file(string);
void output_file(string);
void birthday_in_this_month(string);
```

```

void create_second_file(string, string);
template <typename T1, typename T2>
int get_years_between_dates(T1, T2);
date get_system_date();
vector<string> split(string, char);

```

Functions.cpp

```

#include "Header.h"

struct date {
    int day;
    int month;
    int year;

    string get_format() {
        string str_day = to_string(day);
        string str_month = to_string(month);
        if (day < 10) {
            str_day = '0' + to_string(day);
        }
        if (month < 10) {
            str_month = '0' + to_string(month);
        }
        return str_day + '.' + str_month + '.' + to_string(year);
    }
};

struct employee {

    struct date {
        int day;
        int month;
        int year;

        bool save(string line) {
            vector<string> dmy = split(line, '.');
            day = stoi(dmy[0]);
            month = stoi(dmy[1]);
            year = stoi(dmy[2]);
            if (day > 31 || month > 12) {
                cout << "Entered invalid date" << endl;
                return 0;
            }
            return 1;
        }

        string get_format() {
            string str_day = to_string(day);
            string str_month = to_string(month);
            if (day < 10) {
                str_day = '0' + to_string(day);
            }
            if (month < 10) {
                str_month = '0' + to_string(month);
            }
            return str_day + '.' + str_month + '.' + to_string(year);
        }
    };

    char surname[50];
    date birthday;
    date start_career;

```

```

        void print() {
            cout << "Surname: " << surname << "\t\t";
            cout << "Birthday: " << birthday.get_format() << "\t\t";
            cout << "Start career: " << start_career.get_format() << endl;
        }
};

void input_file(string file_path)
{
    cout << "Choose input mode:" << endl << "1) append info in the file" << endl <<
    "2) create new file" << endl;
    string input_mode;
    getline(cin, input_mode);

    ofstream fileout;
    while (true) {
        if (!input_mode.compare("1"))
        {
            fileout.open(file_path, ios::app);
            break;
        }
        else if (!input_mode.compare("2"))
        {
            fileout.open(file_path);
            break;
        }
        else {
            cout << "Invalid input mode";
        }
    }

    if (!fileout.is_open()) {
        cout << "ERROR: Could not open";
    }
    else {
        string line;
        employee person;
        cout << "Enter information about the employees in format [surname
dd.mm.yyyy dd.mm.yyyy] (send empty line to finish):\n";
        getline(cin, line);
        while (!line.empty()) {
            vector<string> words = split(line, ' ');
            strcpy_s(person.surname, words[0].c_str());
            if (person.birthday.save(words[1]) &&
(person.start_career.save(words[2]))) {
                fileout.write((char*)&person, sizeof(employee));
            }
            getline(cin, line);
        }

        fileout.close();
    }
}

void output_file(string file_path)
{
    ifstream filein;
    filein.open(file_path);

    if (!filein.is_open()) {
        cout << "ERROR: Could not open";
    }
    else {
        employee person;

```

```

        while (filein.read((char*)&person, sizeof(employee)))
        {
            person.print();
        }
    }
    filein.close();
}

void birthday_in_this_month(string file_path)
{
    ifstream filein;
    filein.open(file_path);

    if (!filein.is_open()) {
        cout << "ERROR: Could not open";
    }
    else {
        employee person;
        date sys_date = get_system_date();
        while (filein.read((char*)&person, sizeof(employee)))
        {
            int work_experience = get_years_between_dates(person.start_career,
sys_date);

            if (person.birthday.month == sys_date.month && work_experience >= 5)
            {
                person.print();
            }
        }
        filein.close();
    }
}

void create_second_file(string filein_name, string fileout_name)
{
    ifstream filein;
    ofstream fileout;
    filein.open(filein_name);
    fileout.open(fileout_name);

    if (!filein.is_open() || !fileout.is_open()) {
        cout << "ERROR: Could not open";
    }
    else {
        employee person;
        date sys_date = get_system_date();
        while (filein.read((char*)&person, sizeof(employee)))
        {
            int start_career_age = get_years_between_dates(person.birthday,
person.start_career);
            int work_experience = get_years_between_dates(person.start_career,
sys_date);

            if (start_career_age <= 25 && work_experience >= 10)
            {
                fileout.write((char*)&person, sizeof(employee));
            }
        }

        filein.close();
        fileout.close();
    }
}

template <typename T1, typename T2>
int get_years_between_dates(T1 start_date, T2 end_date)

```

```

{
    int years = end_date.year - start_date.year;
    if (start_date.month > end_date.month || (start_date.month == end_date.month &&
start_date.day > end_date.day))
    {
        years -= 1;
    }
    return years;
}

date get_system_date()
{
    date res;
    time_t theTime = time(NULL);
    struct tm* aTime = localtime(&theTime);

    res.day = aTime->tm_mday;
    res.month = aTime->tm_mon + 1;
    res.year = aTime->tm_year + 1900;

    return res;
}

vector<string> split(string str, char separator)
{
    vector<string> res;
    string slice = "";
    str += " ";
    for (int i = 0; i < str.length(); i++) {
        if (str[i] == separator)
        {
            if (slice.length() > 0) {
                res.push_back(slice);
                slice = "";
            }
        }
        else {
            slice += str[i];
        }
    }
    res.push_back(slice);
    return res;
}

```

Тестування:

```
Microsoft Visual Studio Debug Console

Choose input mode:
1) append info in the file
2) create new file
2
Enter information about the employees in format [surname dd.mm.yyyy dd.mm.yyyy] (send empty line to finish):
Ivanov 02.12.1988 19.09.2000
Petrov 21.02.2001 01.01.2017

Birthday in this month AND work experience >= 5 years:
Surname: Petrov      Birthday: 21.02.2001      Start career: 01.01.2017

Start career in <= 25 yo AND work experience >= 10 years (second file):
Surname: Ivanov      Birthday: 02.12.1988      Start career: 19.09.2000

D:\Study\OP\2\labs_cpp\Debug\lab_cpp_2.exe (process 5720) exited with code 0.
To automatically close the console when debugging stops, enable Tools->Options->Debugging->Automatically close
when debugging stops.
Press any key to close this window . . .
```

Python

lab_py_2.py

```
from lab_py_2_functions import *

first_file_path = "first file.txt"
second_file_path = "second file.txt"
input_file(first_file_path)

print("Birthday in this month AND work experience >= 5 years: ")
birthday_in_this_month(first_file_path)
print()

create_second_file(first_file_path, second_file_path)
print("Start career in <= 25 yo AND work experience >= 10 years (second
file):")
output_file(second_file_path)
```

lab_py_2_functions.py

```
import pickle
from datetime import datetime

def input_file(filename):
    print("Choose input mode:\n1) append info in the file\n2) create new
file")
    input_mode = int(input())
    while True:
        if input_mode == 1:
            file = open(filename, "ab")
            break
        elif input_mode == 2:
            file = open(filename, "wb")
            break
```

```

        else:
            print("Invalid input mode")

    line = input("Enter information about the employees in format [surname
dd.mm.yyyy dd.mm.yyyy] "
                "(send empty line to finish):\n")
    while line != "":
        words = line.split()
        birthday = words[1].split('.')
        start_career = words[2].split('.')
        employee = {
            "surname": words[0],
            "birthday": {"day": int(birthday[0]),
                        "month": int(birthday[1]),
                        "year": int(birthday[2])},
            "start_career": {"day": int(start_career[0]),
                            "month": int(start_career[1]),
                            "year": int(start_career[2])}
        }
        pickle.dump(employee, file)
        line = input()

    file.close()

def output_file(filename):
    with open(filename, 'rb') as file:
        size = file.seek(0, 2)
        file.seek(0)

        while file.tell() < size:
            employee = pickle.load(file)
            print_employee(employee)

def birthday_in_this_month(filename):
    with open(filename, 'rb') as file:
        sys_date = get_system_date()
        size = file.seek(0, 2)
        file.seek(0)

        while file.tell() < size:
            employee = pickle.load(file)
            work_experience =
get_years_between_dates(employee["start_career"], sys_date)
            if employee["birthday"]["month"] == sys_date["month"] and
work_experience >= 5:
                print_employee(employee)

def create_second_file(filein_name, fileout_name):
    with open(filein_name, 'rb') as filein:
        with open(fileout_name, 'wb') as fileout:
            sys_date = get_system_date()
            size = filein.seek(0, 2)
            filein.seek(0)

            while filein.tell() < size:
                employee = pickle.load(filein)
                start_career_age =
get_years_between_dates(employee["birthday"], employee["start_career"])
                work_experience =
get_years_between_dates(employee["start_career"], sys_date)
                if start_career_age <= 25 and work_experience >= 10:

```



```

        pickle.dump(employee, fileout)

def get_years_between_dates(start_date, end_date):
    years = end_date["year"] - start_date["year"]
    if start_date["month"] > end_date["month"] or (start_date["month"] ==
end_date["month"] and
                                                    start_date["day"] >
end_date["day"]):
        years -= 1
    return years

def get_format(date):
    str_day = str(date["day"])
    str_month = str(date["month"])
    if date["day"] < 10:
        str_day = '0' + str_day

    if date["month"] < 10:
        str_month = '0' + str_month

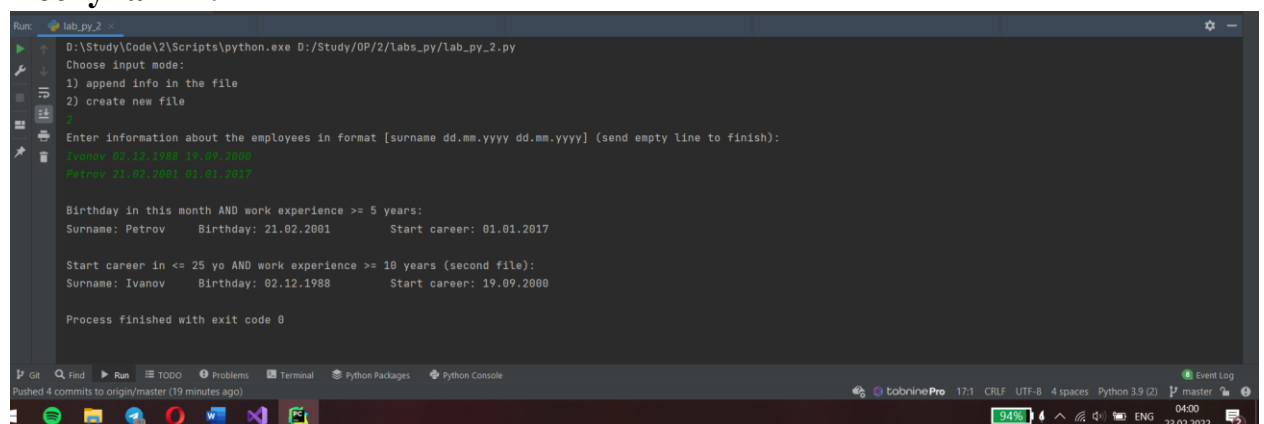
    return str_day + '.' + str_month + '.' + str(date["year"])

def get_system_date():
    current_datetime = datetime.now()
    sys_date = {
        "day": current_datetime.day,
        "month": current_datetime.month,
        "year": current_datetime.year
    }
    return sys_date

def print_employee(employee):
    print("Surname: " + employee["surname"] + "\t \tBirthday: " +
get_format(employee["birthday"])
        + "\t \tStart career: " + get_format(employee["start_career"]))

```

Тестування:



```

Run: lab_py_2
D:\Study\Code\2\Scripts\python.exe D:/Study/0P/2/labs_py/lab_py_2.py
Choose input mode:
1) append info in the file
2) create new file
3)
Enter information about the employees in format [surname dd.mm.yyyy dd.mm.yyyy] (send empty line to finish):
Ivanov 02.12.1988 19.09.2000
Petrov 21.02.2001 01.01.2017

Birthday in this month AND work experience >= 5 years:
Surname: Petrov    Birthday: 21.02.2001    Start career: 01.01.2017

Start career in <= 25 yo AND work experience >= 10 years (second file):
Surname: Ivanov    Birthday: 02.12.1988    Start career: 19.09.2000

Process finished with exit code 0

```

Висновки:

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