

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 21

Виконав студент ІП-13 Макарчук Лідія Олександрівна  
(шифр, прізвище, ім'я, по батькові)

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

Київ 2022

## Лабораторна робота №2

### Бінарні файли

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

Варіант 21

### Завдання:

21. Створити файл із списком автомобілів автосалону: назва, дата випуску, дата надходження у продаж. Створити список нових автомобілів (які надійшли у продаж не більш як через 2 місяці після випуску). Вивести інформацію про автомобілі, які були випущені не раніше вказаного року.

1. Виконання завдання мовою C++

### Код:

#### //main

```
#include <iostream>
#include <string>
#include "File_functions.h"

using namespace std;

int main()
{
    string pathAutoList = "automobiles\\List of automobiles.txt";
    string newPathAutoList = "automobiles\\New automobiles.txt";
    write_automobiles_into_file(pathAutoList);
    cout << "\nThere is information in file:\n\n";
    display_file_information(pathAutoList);
    write_new_file_of_automobile(pathAutoList, newPathAutoList);
    cout << "-----\n\n";
    cout << "There is a list of automobiles that have been sold within 2 months after
production:\n";
    display_file_information(newPathAutoList);
    cout << "-----\n\n";
    display_automobiles_released_not_earlier_than_year(pathAutoList);
}
```

#### //Automobile.h

```
#pragma once
#include <vector>

struct Date
{
    int day;
    int month;
    int year;
};

struct Automobile
{
    char name[30];
    Date releaseDate;
    Date saleDate;
};

Automobile init_automobile();
```

```

void print_automobile(Automobile automobile);
Date init_date(std::string);
void check_sale_date_is_not_smaller_than_release_date(Automobile& automobile);
void print_date(Date date);
bool is_less_than_two_month_between_release_and_sale(Automobile automobile);

```

## //Automobile.cpp

```

#include <iostream>
#include <vector>
#include "Automobile.h"

using namespace std;
Automobile init_automobile()
{
    Automobile automobile;
    cout << "Enter automobile's name: ";
    cin.getline(automobile.name, sizeof(automobile.name));
    automobile.releaseDate = init_date("Release");
    automobile.saleDate = init_date("Sale");
    check_sale_date_is_not_smaller_than_release_date(automobile);
    return automobile;
}

void print_automobile(Automobile automobile)
{
    cout << "Automobile's name: " << automobile.name << endl;
    cout << "Release date: ";
    print_date(automobile.releaseDate);
    cout << endl << "Sale date: ";
    print_date(automobile.saleDate);
    cout << endl;
}

Date init_date(string dateType)
{
    Date date;
    const int MINDATE = 1;
    const int MAXMONTH = 12;
    int MAXDAY;
    do
    {
        cout << dateType << " date dd.mm.yyyy: ";
        cin >> date.day; cin.ignore();
        cin >> date.month; cin.ignore();
        cin >> date.year; cin.ignore();
        if (date.month == 4 || date.month == 6 || date.month == 9 || date.month == 11)
            MAXDAY = 30;
        else if (date.month == 2)
            MAXDAY = (date.year % 4) == 0 ? 29 : 28;
        else
            MAXDAY = 31;
    } while (date.day < MINDATE || date.day > MAXDAY || date.month < MINDATE || date.month >
MAXMONTH || date.year < MINDATE);
    return date;
}

void check_sale_date_is_not_smaller_than_release_date(Automobile& automobile)
{
    const int YFactor = 360;
    const int MFACTOR = 31;
    int release_days = automobile.releaseDate.day + automobile.releaseDate.month *
MFACTOR + automobile.releaseDate.year * YFactor;
    int sale_days = automobile.saleDate.day + automobile.saleDate.month * MFACTOR +
automobile.saleDate.year * YFactor;
    while (sale_days < release_days)
    {

```

```

        cout << "Sale date can't be smaller than release date. Please, enter correct
dates.\n";
        automobile.releaseDate = init_date("Release");
        automobile.saleDate = init_date("Sale");
        release_days = automobile.releaseDate.day + automobile.releaseDate.month *
MFACTOR + automobile.releaseDate.year * YFactor;
        sale_days = automobile.saleDate.day + automobile.saleDate.month * MFACTOR +
automobile.saleDate.year * YFactor;
    }
}

void print_date(Date date)
{
    cout.fill('0');
    cout.width(2); cout << date.day << ".";
    cout.width(2); cout << date.month << ".";
    cout.width(4); cout << date.year;
    cout.fill(' ');
}

bool is_less_than_two_month_between_release_and_sale(Automobile automobile)
{
    bool isLess = false;
    int yearDifference = automobile.saleDate.year - automobile.releaseDate.year;
    int monthDifference = 3;
    if (yearDifference == 1)
        monthDifference = 12 - automobile.releaseDate.month +
automobile.saleDate.month;
    else if (yearDifference == 0)
        monthDifference = automobile.saleDate.month - automobile.releaseDate.month;
    if (monthDifference == 2)
    {
        if (automobile.saleDate.day <= automobile.releaseDate.day)
            isLess = true;
    }
    else if (monthDifference < 2)
        isLess = true;
    return isLess;
}

```

## //File\_functions.h

```

#pragma once
#include <string>
#include "Automobile.h"

char choose_file_mode();
void write_automobiles_into_file(std::string path);
std::vector<Automobile> read_file_into_list(std::string path);
void display_file_information(std::string path);
void write_new_file_of_automobile(std::string pathOld, std::string pathNew);
void display_automobiles_released_not_earlier_than_year(std::string path);

```

## //File\_functions.cpp

```

#include <iostream>
#include <string>
#include <fstream>
#include "Automobile.h"

using namespace std;

char choose_file_mode()
{
    cout << "Do you want to create a new file or just to add new information?\nTo create
a new file enter 'n', to add information enter 'a': ";
}

```

```

        char answer; cin >> answer;
        while (answer != 'a' && answer != 'n')
        {
            cout << "Wrong letter! Try again! ";
            cin >> answer;
        }
        return answer;
    }

void write_automobiles_into_file(string path)
{
    char answer = choose_file_mode();
    ofstream outFile;
    if (answer == 'n')
        outFile.open(path, ios::binary);
    else
        outFile.open(path, ios::binary|ios::app);
    if (!outFile.is_open())
        cout << "Cannot open the file!\n";
    else
    {
        int n;
        vector<Automobile> automobileList;
        cout << "Enter number of the automobiles: ";
        cin >> n; cin.ignore(); cout << endl;
        for (int i = 0; i < n; i++)
        {
            Automobile automobile = init_automobile();
            automobileList.push_back(automobile);
            cout << endl;
        }
        for (int i = 0; i < automobileList.size(); i++)
            outFile.write((char*)&automobileList[i], sizeof(Automobile));
    }
    outFile.close();
}

vector<Automobile> read_file_into_list(string path)
{
    vector<Automobile> automobileList;
    ifstream inFile(path, ios::binary);
    if (!inFile.is_open())
        cout << "Cannot open the file!\n";
    else
    {
        Automobile automobile;
        while (inFile.read((char*)&automobile, sizeof(Automobile)))
            automobileList.push_back(automobile);
    }
    inFile.close();
    return automobileList;
}

void display_file_information(string path)
{
    vector<Automobile> automobileList = read_file_into_list(path);
    for (int i = 0; i < automobileList.size(); i++)
    {
        print_automobile(automobileList[i]);
        cout << endl;
    }
}

void write_new_file_of_automobile(string pathOld, string pathNew)
{
    vector<Automobile> newList;
    ifstream inFile(pathOld, ios::binary);
    if (!inFile.is_open())

```

```

        cout << "Cannot open the file!\n";
else
{
    Automobile automobile;
    while (inFile.read((char*)&automobile, sizeof(Automobile)))
        if (is_less_than_two_month_between_release_and_sale(automobile))
            newtList.push_back(automobile);
}
inFile.close();
ofstream outFile(pathNew, ios::binary);
if (!outFile.is_open())
    cout << "Cannot open the file!\n";
else
{
    for (int i = 0; i < newtList.size(); i++)
        outFile.write((char*)&newtList[i], sizeof(Automobile));
}
outFile.close();
}

void display_automobiles_released_not_earlier_than_year(string path)
{
    int year;
    cout << "Enter the year to see automobiles which have not been released before this
year: ";
    cin >> year; cin.ignore();
    vector <Automobile> automobileList = read_file_into_list(path);
    for (int i = 0; i < automobileList.size(); i++)
    {
        if (automobileList[i].releaseDate.year >= year)
        {
            print_automobile(automobileList[i]);
            cout << endl;
        }
    }
}

```

**Тестування програми:**

```
Do you want to create a new file or just to add new information?
To create a new file enter 'n', to add information enter 'a': a
Enter number of the automobiles: 2
```

```
Enter automobile's name: name number 2
Enter release date in this format dd.mm.yyyy: 06.08.2002
Enter sale date in this format dd.mm.yyyy: 07.09.2002
```

```
Enter automobile's name: name number 3
Enter release date in this format dd.mm.yyyy: 14.09.1999
Enter sale date in this format dd.mm.yyyy: 14.09.1999
```

```
-----
```

```
There is information in file:
```

```
Automobile's name: name number 1
Release date: 02.07.1993
Sale date: 03.05.1994
```

```
Automobile's name: name number 2
Release date: 06.08.2002
Sale date: 07.09.2002
```

```
Automobile's name: name number 3
Release date: 14.09.1999
Sale date: 14.09.1999
```

```
-----
```

```
There is a list of automobiles that have been sold within 2 months after production:
```

```
Automobile's name: name number 2
Release date: 06.08.2002
Sale date: 07.09.2002
```

```
Automobile's name: name number 3
Release date: 14.09.1999
Sale date: 14.09.1999
```

```
-----
```

```
Enter the year to see automobiles which have not been released before this year: 1999
```

```
Automobile's name: name number 2
Release date: 06.08.2002
Sale date: 07.09.2002
```

```
Automobile's name: name number 3
Release date: 14.09.1999
Sale date: 14.09.1999
```

```
C:\Users\USER1\source\repos\s21aba1cpp\Debug\laba2cpp.exe (process 17320) exited with code 0.
Press any key to close this window . . .
```

## 2. Виконання завдання мовою Python

### Код:

#### #main

```
from File_functions import write_automobiles_into_file, display_file_information,
write_new_file_of_automobile, display_automobiles_released_not_earlier_than_year

path_automobile_list = "automobiles\\List of automobiles.txt"
new_path_automobile_list = "automobiles\\New automobiles.txt"
write_automobiles_into_file(path_automobile_list)
print("There is information in file:\n")
display_file_information(path_automobile_list)
print("-----\n")
write_new_file_of_automobile(path_automobile_list, new_path_automobile_list)
print("There is a list of automobiles that have been sold within 2 months after
production:")
display_file_information(new_path_automobile_list)
print("-----\n")
display_automobiles_released_not_earlier_than_year(path_automobile_list)
```

#### #Automobile.py

```
def init_automobile():
    automobile = {
        'name' : input("Enter automobile's name: "),
        'release_date' : init_date("release"),
        'sale_date' : init_date("sale")
    }
    automobile = check_sale_date_is_not_smaller_than_release_date(automobile)
    return automobile

def print_automobile(automobile):
    print(f"Automobile's name: {automobile['name']}")
    print("Release date: ", end = '')
    print_date(automobile['release_date'])
    print("Sale date: ", end = '')
    print_date(automobile['sale_date'])

def calc_max_day(month, year):
    MAXDAY = 31
    if month == 4 or month == 6 or month == 9 or month == 11:
        MAXDAY = 30
    elif month == 2:
        if year % 4 == 0:
            MAXDAY = 29
        else:
            MAXDAY = 28
    return MAXDAY

def init_date(type_of_data):
    str = input(f"Enter {type_of_data} date in this format dd.mm.yyyy: ")
    str = str.split('.')
    date = {
        'day' : int(str[0]),
        'month' : int(str[1]),
        'year' : int(str[2])
    }
    MINDATE = 1
    MAXMONTH = 12
    MAXDAY = calc_max_day(date['month'], date['year'])
```



```

    while date['day'] < MINDATE or date['day'] > MAXDAY or date['month'] < MINDATE or
date['month'] > MAXMONTH or date['year'] < MINDATE:
        str = input("Your date is incorrect! Try again! Enter date in format dd.mm.yyyy: ")
        str = str.split('.')
        date['day'] = int(str[0])
        date['month'] = int(str[1])
        date['year'] = int(str[2])
        MAXDAY = calc_max_day(date['month'], date['year'])
    return date

def check_sale_date_is_not_smaller_than_release_date(automobile):
    y_factor = 380
    m_factor = 31
    release_days = automobile['release_date']['day'] +
automobile['release_date']['month']*m_factor + automobile['release_date']['year']*y_factor
    sale_days = automobile['sale_date']['day'] + automobile['sale_date']['month']*m_factor +
automobile['sale_date']['year']*y_factor
    while sale_days < release_days:
        print("Sale date can't be smaller than release date. Please, enter correct dates.")
        automobile['release_date'] = init_date("release")
        automobile['sale_date'] = init_date("sale")
        release_days = automobile['release_date']['day'] +
automobile['release_date']['month']*m_factor + automobile['release_date']['year']*y_factor
        sale_days = automobile['sale_date']['day'] +
automobile['sale_date']['month']*m_factor + automobile['sale_date']['year']*y_factor
    return automobile

def print_date(date):
    print(str(date['day']).zfill(2) + '.', end = '')
    print(str(date['month']).zfill(2) + '.', end = '')
    print(str(date['year']).zfill(4) + '.')

def is_less_than_two_month_between_release_and_sale(automobile):
    is_less = False
    year_difference = automobile['sale_date']['year'] - automobile['release_date']['year']
    month_difference = 3
    if year_difference == 1:
        month_difference = 12 - automobile['release_date']['month'] +
automobile['sale_date']['month']
    elif year_difference == 0:
        month_difference = automobile['sale_date']['month'] -
automobile['release_date']['month']
    if month_difference == 2:
        if automobile['sale_date']['day'] <= automobile['release_date']['day']:
            is_less = True
    elif month_difference < 2:
        is_less = True
    return is_less

```

## #File\_functions.py

```

import pickle
from Automobile import init_automobile, print_automobile,
is_less_than_two_month_between_release_and_sale

def choose_file_mode():
    print("Do you want to create a new file or just to add new information?")
    answer = input("To create a new file enter 'n', to add information enter 'a': ")
    while answer != 'a' and answer != 'n':
        answer = input("Wrong letter! Try again!")
    return answer

def write_automobiles_into_file(path):
    answer = choose_file_mode()

```

```

if answer == 'n':
    out_file = open(path, 'wb')
else:
    out_file = open(path, 'ab')
n = int(input("Enter number of the automobiles: "))
print()
automobile_list = []
for i in range(n):
    automobile = init_automobile()
    automobile_list.append(automobile)
    print()
for automobile in automobile_list:
    pickle.dump(automobile, out_file)
out_file.close()

def read_file_into_list(path):
    automobile_list = []
    with open(path, 'rb') as in_file:
        in_file.seek(0, 2)
        end_of_file = in_file.tell()
        in_file.seek(0, 0)
        while in_file.tell() != end_of_file:
            automobile = pickle.load(in_file)
            automobile_list.append(automobile)
    return automobile_list

def display_file_information(path):
    automobile_list = read_file_into_list(path)
    for automobile in automobile_list:
        print_automobile(automobile)
    print()

def write_new_file_of_automobile(path_old, path_new):
    new_list = []
    with open(path_old, 'rb') as in_file:
        in_file.seek(0, 2)
        end_of_file = in_file.tell()
        in_file.seek(0, 0)
        while in_file.tell() != end_of_file:
            automobile = pickle.load(in_file)
            if is_less_than_two_month_between_release_and_sale(automobile):
                new_list.append(automobile)
    with open(path_new, 'wb') as out_file:
        for automobile in new_list:
            pickle.dump(automobile, out_file)

def display_automobiles_released_not_earlier_than_year(path):
    year = int(input("Enter the year to see automobiles which have not been released before this year: "))
    automobile_list = read_file_into_list(path)
    for automobile in automobile_list:
        if automobile['release_date']['year'] >= year:
            print_automobile(automobile)
            print()

```

## Тестування програми:

C:\WINDOWS\system32\cmd.exe

```
Do you want to create a new file or just to add new information?
To create a new file enter 'n', to add information enter 'a': n
Enter number of the automobiles: 4

Enter automobile's name: first name1
Enter release date in this format dd.mm.yyyy: 12.13.1999
Your date is incorrect! Try again! Enter date in format dd.mm.yyyy: 12.12.1999
Enter sale date in this format dd.mm.yyyy: 01.02.2000

Enter automobile's name: second name2
Enter release date in this format dd.mm.yyyy: 13.07.2008
Enter sale date in this format dd.mm.yyyy: 11.09.2008

Enter automobile's name: name3
Enter release date in this format dd.mm.yyyy: 11.11.2004
Enter sale date in this format dd.mm.yyyy: 11.12.2003
Sale date can't be smaller than release date. Please, enter correct dates.
Enter release date in this format dd.mm.yyyy: 11.11.2004
Enter sale date in this format dd.mm.yyyy: 11.12.2005

Enter automobile's name: name4
Enter release date in this format dd.mm.yyyy: 03.04.1978
Enter sale date in this format dd.mm.yyyy: 23.06.1978

-----

There is information in file:

Automobile's name: first name1
Release date: 12.12.1999
Sale date: 01.02.2000

Automobile's name: second name2
Release date: 13.07.2008
Sale date: 11.09.2008

Automobile's name: name3
Release date: 11.11.2004
Sale date: 11.12.2005
```

```
Automobile's name: name4
Release date: 03.04.1978
Sale date: 23.06.1978

-----

There is a list of automobiles that have been sold within 2 months after production:
Automobile's name: first name1
Release date: 12.12.1999
Sale date: 01.02.2000

Automobile's name: second name2
Release date: 13.07.2008
Sale date: 11.09.2008

-----

Enter the year to see automobiles which have not been released before this year: 2000
Automobile's name: second name2
Release date: 13.07.2008
Sale date: 11.09.2008

Automobile's name: name3
Release date: 11.11.2004
Sale date: 11.12.2005

Press any key to continue . . . █
```

Висновок: Під час виконання лабораторної роботи я вивчила 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 2-х файлів зі списком автомобілів. Створення другого файлу було виконано на основі даних першого файлу та перевірки, чи є різниця в часі між датою випуску та датою продажу більшою, ніж 2 місяці. Також було реалізовано виведення інформації про автомобілі, що були випущені не раніше заданого року. Після тестування програм можна зробити висновок, що вони справляються із поставленою задачею.