

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»  
«Робота з бінарними файлами»  
Варіант 28

Виконав студент ІП-13, Петров Ігор Ярославович  
(шифр, прізвище, ім'я, по батькові)

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

Київ 2022  
**Лабораторна робота № 2 Тема:**  
**Робота з бінарними файлами**  
**Варіант 28**

28. Створити файл із списком автомобілів автосалону: назва, дата випуску, дата надходження у продаж. Створити список автомобілів, що надійшли у продаж за останній місяць. Вивести інформацію про вживані автомобілі (які були випущені більш ніж за рік до надходження у продаж).

### Код C++

#### main.cpp

```
#include "func.h"

int main()
{
    string first_path = "Allcars.bin";
    string second_path = "Recentcars.bin";
    string mode = enter_mode();
    creating_first_file(first_path, mode);
    creating_second_file(first_path, second_path);
    cout << "\nAll cars:\n";
    read_cars(first_path);
    cout << "\nRecent cars:\n";
    read_cars(second_path);
    cout << "\nUsed cars: \n";
    output_used_cars(first_path);
}
```

#### func.cpp

```
#include "func.h"

string enter_mode() {
    string file_mode;
    cout << "Enter 'w' to overwrite file or 'a' to append information to file: " << endl;
    cin >> file_mode;
    while (file_mode != "w" and file_mode != "a") {
        cout << "Incorrect input. Enter 'w' or 'a'.Enter 'w' to overwrite file or 'a' to append information to file : " << endl;
        cin >> file_mode;
    }
    return file_mode;
}

void creating_first_file(string path, string mode) {
    ofstream file;
    if (mode == "w") {
        file.open(path, ios::binary);
    }
    else {
        file.open(path, ios::binary | ios::app);
    }
    string line;
    cout << "Enter cars in format: name DD.MM.YYYY(release date) DD.MM.YYYY(sell date).\n"
        << "To finish entering go to a new line and press <Ctrl + S>:\n";
    getline(cin, line);
    while (line[0] != char(19)) {
        if (line.length() > 0) {
            Car car;
            vector<string> splitted = split(line);
            string name;
            for (size_t i = 0; i < splitted.size() - 2; i++)
            {
                name += splitted[i] + ' ';
            }
        }
    }
}
```

```

    }
    name.pop_back();
    strcpy_s(car.name, name.c_str());
    strcpy_s(car.release_date, splitted[splitted.size() - 2].c_str());
    strcpy_s(car.sell_date, splitted[splitted.size() - 1].c_str());
    file.write((char*)&car, sizeof(Car));
}
getline(cin, line);
}
}

void creating_second_file(string first_path, string second_path) {
    ifstream first_file(first_path, ios::binary);
    if (first_file.is_open()) {
        ofstream second_file(second_path, ios::binary);
        if (second_file.is_open()) {
            Car car;
            time_t t = time(0);
            tm* now = localtime(&t);
            const int current_year = now->tm_year + 1900;
            const int current_month = now->tm_mon + 1;
            while (first_file.read((char*)&car, sizeof(Car))) {
                if (current_month == stoi(split(car.sell_date, '.')[1]) and current_year ==
stoi(split(car.sell_date, '.')[2])) {
                    second_file.write((char*)&car, sizeof(Car));
                }
            }
        }
        else {
            cerr << "Fatal error";
        }
        second_file.close();
    }
    else {
        cerr << "Fatal error";
    }
    first_file.close();
}

void read_cars(string path) {
    ifstream file(path, ios::binary);
    Car car;
    while (file.read((char*)&car, sizeof(Car))) {
        cout << "Name: " << car.name << ", release date: " << car.release_date << ", sell date: " <<
car.sell_date << endl;
    }
    file.close();
}

void output_used_cars(string file_path) {
    ifstream file(file_path, ios::binary);
    if (file.is_open()) {
        Car car;
        vector<Car> car_list;
        while (file.read((char*)&car, sizeof(Car))) {
            car_list.push_back(car);
        }
        int s_day, s_month, s_years, r_day, r_month, r_years, delta_year, delta_month, delta_day;
        string string_sell, string_release;
        for (size_t i = 0; i < car_list.size(); i++)
        {
            string_sell = string(car_list[i].sell_date);
            vector<string> splitted_sell = split(string_sell, '.');
            s_day = stoi(splitted_sell[0]);
            s_month = stoi(splitted_sell[1]);
            s_years = stoi(splitted_sell[2]);
            string_release = string(car_list[i].release_date);
            vector<string> splitted_release = split(string_release, '.');
        }
    }
}

```

```
    r_day = stoi(splitted_release[0]);
    r_month = stoi(splitted_release[1]);
    r_years = stoi(splitted_release[2]);
    delta_year = s_years - r_years;
    delta_month = s_month - r_month;
    delta_day = s_day - r_day;
    if (delta_year > 1 or (delta_year == 1 and delta_month > 0) or (delta_year == 1 and
delta_month == 0 and delta_day > 0)) {
        cout << "Name: " << car_list[i].name << ", release date: " << car_list[i].release_date <<
", sell date: " << car_list[i].sell_date << endl;
    }
}
}
else {
    cout << "Fatal error";
}
file.close();
}

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

### Header.h

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

```
using namespace std;
```

```
void write_file(string);
void read_file(string);
void format_file(string, string, int);
```

```
mode = enter_mode()
creating_first_file(first_path, mode)
creating_second_file(first_path, second_path)
print("\nAll cars:")
read_cars(first_path)
print("\nRecent cars:")
read_cars(second_path)
output_used_cars(first_path)
```

### Код python

#### main.py

```
from func import enter_mode,
creating_first_file, creating_second_file,
read_cars, output_used_cars
```

```
def main():
    first_path = "Allcars.bin"
    second_path = "Recentcars.bin"
```

```
if __name__ == "__main__":
    main()
```

### func.py

```
import pickle
from datetime import datetime

class Car:
    def __init__(self, name: str, release_date: str, sell_date: str):
        self.name = name
        self.release_date = release_date
        self.sell_date = sell_date

    def __str__(self):
        return "Name: {}, release date: {}, sell date: {}".format(self.name, self.release_date,
self.sell_date)

def enter_mode():
    file_mode = input("Enter 'w' to overwrite file or 'a' to append information to file: ")
    while file_mode != 'w' and file_mode != 'a':
        file_mode = input("Incorrect input. Enter 'w' to overwrite file or 'a' to append information to
file: ")
    return file_mode + "b"

def creating_first_file(path : str, mode: str):
    line = input("Enter cars in format: name DD.MM.YYYY(release date) DD.MM.YYYY(sell date).\n"
        "To finish entering go to a new line and press '&&':\n")
    with open(path, mode) as file:
        while line != "&&":
            if line:
                splitted = line.split()
                name = str()
                for info in splitted[:len(splitted)-2]:
                    name += info + " "
                name = name.rstrip(name[-1])
                release_date = splitted[len(splitted)-2]
                sell_date = splitted[-1]
                car = Car(name, release_date, sell_date)
                pickle.dump(car, file)
            line = input()

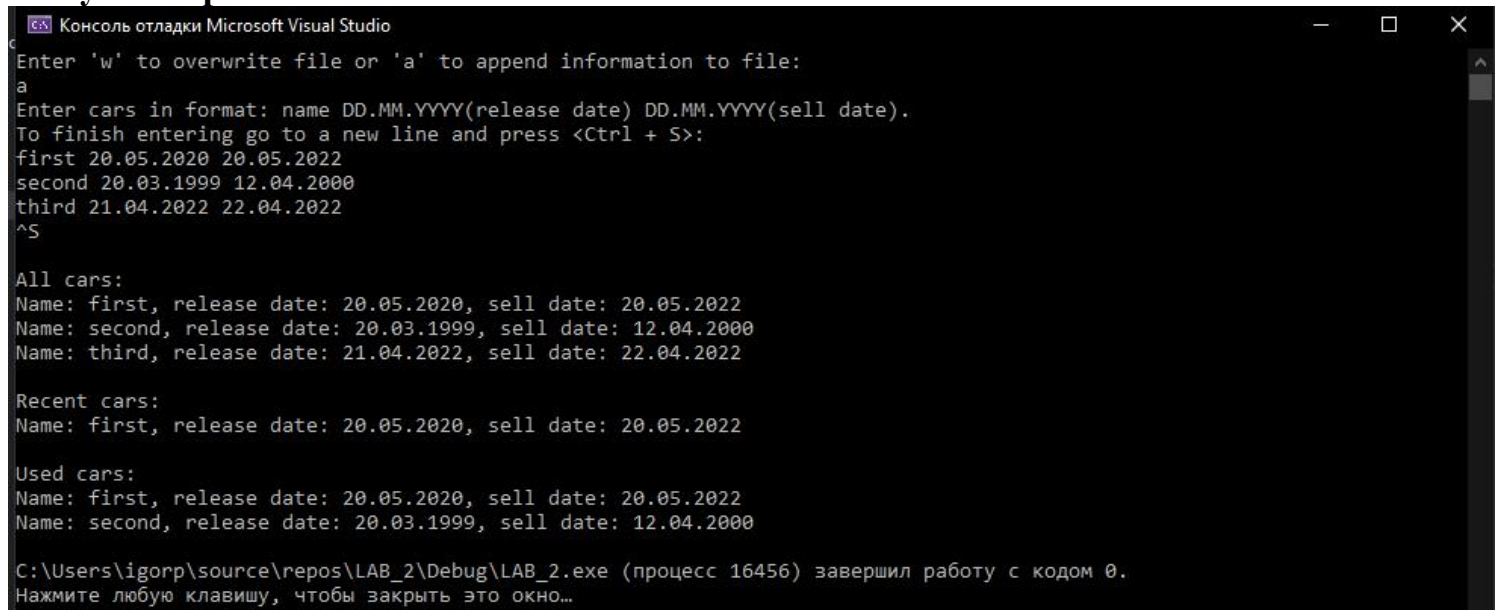
def creating_second_file(first_path: str, second_path: str):
    with open(first_path, 'rb') as first_file:
        with open(second_path, "wb") as second_file:
            while True:
                try:
                    car = pickle.load(first_file)
                    if datetime.now().month == int(car.sell_date.split('.')[1]) and datetime.now().year ==
int(car.sell_date.split('.')[2]):
                        pickle.dump(car, second_file)
                except EOFError:
                    break

def read_cars(path: str):
    with open(path, 'rb') as file:
        while True:
            try:
                print(pickle.load(file))
            except EOFError:
                break

def output_used_cars(file_path: str):
    car_list = list()
    with open(file_path, "rb") as file:
```

```
while True:
    try:
        car_list.append(pickle.load(file))
    except EOFError:
        break
print("\nUsed cars:")
for car in car_list:
    splitted_sell = car.sell_date.split(".")
    s_day = int(splitted_sell[0])
    s_month = int(splitted_sell[1])
    s_years = int(splitted_sell[2])
    splitted_release = car.release_date.split(".")
    r_day = int(splitted_release[0])
    r_month = int(splitted_release[1])
    r_years = int(splitted_release[2])
    delta_year = s_years - r_years
    delta_month = s_month - r_month
    delta_day = s_day - r_day
    if delta_year > 1 or (delta_year == 1 and delta_month > 0) or (delta_year == 1 and delta_month ==
0 and delta_day > 0):
        print(car)
```

### Результат роботи



```
Консоль отладки Microsoft Visual Studio
Enter 'w' to overwrite file or 'a' to append information to file:
a
Enter cars in format: name DD.MM.YYYY(release date) DD.MM.YYYY(sell date).
To finish entering go to a new line and press <Ctrl + S>:
first 20.05.2020 20.05.2022
second 20.03.1999 12.04.2000
third 21.04.2022 22.04.2022
^S

All cars:
Name: first, release date: 20.05.2020, sell date: 20.05.2022
Name: second, release date: 20.03.1999, sell date: 12.04.2000
Name: third, release date: 21.04.2022, sell date: 22.04.2022

Recent cars:
Name: first, release date: 20.05.2020, sell date: 20.05.2022

Used cars:
Name: first, release date: 20.05.2020, sell date: 20.05.2022
Name: second, release date: 20.03.1999, sell date: 12.04.2000

C:\Users\igor\source\repos\LAB_2\Debug\LAB_2.exe (процесс 16456) завершил работу с кодом 0.
Нажмите любую клавишу, чтобы закрыть это окно...
```