Основи програмування – 1. Алгоритми та структури даних

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»

«Текстові файли»

Варіант 29

Виконав студент ІП-13 Романюк Діана Олексіївна
(шифр, прізвище, ім'я, по батькові)

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

Київ 2022

# Лабораторна робота 2

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

## Варіант 29

## Постановка задачі:

29. Створити файл із списком покупців, які придбали товари зі знижкою на день акції: прізвище, стать, дата народження, кількість одиниць товару. Передбачається, що вартість однієї одиниці товару - 100 грн., знижка на товар дорівнює віку особи. Пенсіонерам (з 60-ти років) надається додаткова знижка 5%. Визначити виторг магазину за день. Створити новий файл з інформацією про покупців, які придбали товару більш ніж на 250 грн.

## Код програми на C++:

## Header.h

```
#pragma once
#include <iostream>
#include <fstream>
#include <vector>
#include <string>
#include <windows.h>

using namespace std;

class Client {
public:
	char sec_name[255];
	char sex;
	char date_of_birth[255];
	int amount;
	int spent;
	Client(string = "none n 01:01:1970 0");
	int GetAge();
};

int StoreProfit(string);
void outFile(string);
void inFile(string, bool);
void inNewFile(string, string);
 vector<string> split(string, char sep = ' ');
```

## Lab2.cpp

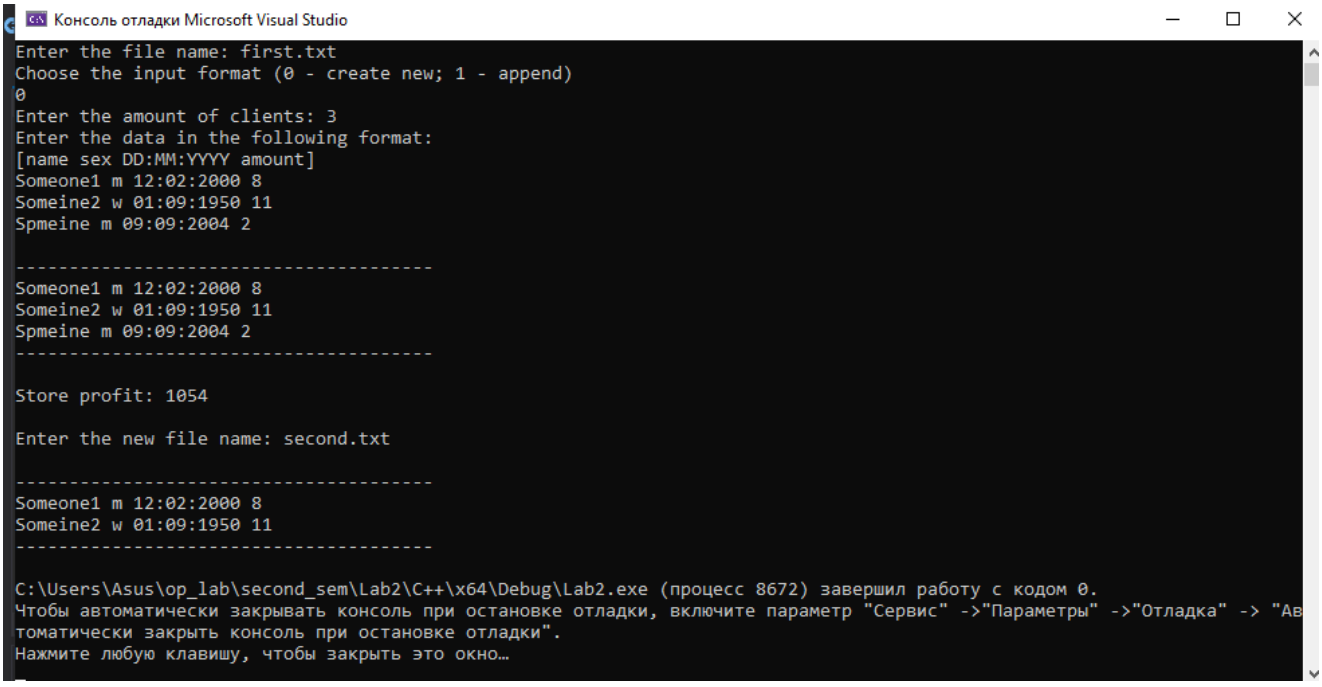
```
#include "Header.h"

int main() {
	string path1, path2;
	bool format;
	cout << "Enter the file name: "; cin >> path1;
	cout << "Choose the input format (0 - create new; 1 - append)" << endl; cin >>
format;
	inFile(path1, format);
	outFile(path1);
	cout << '\n' << "Store profit: " << StoreProfit(path1) << '\n';
	cout << '\n' << "Enter the new file name: "; cin >> path2;
	inNewFile(path1, path2);
	outFile(path2);
```

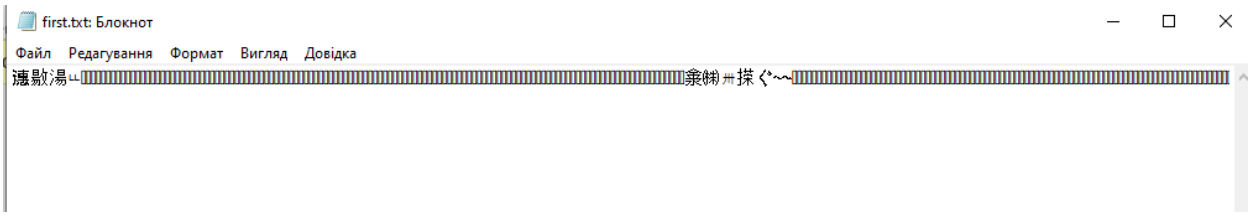
```
}
```

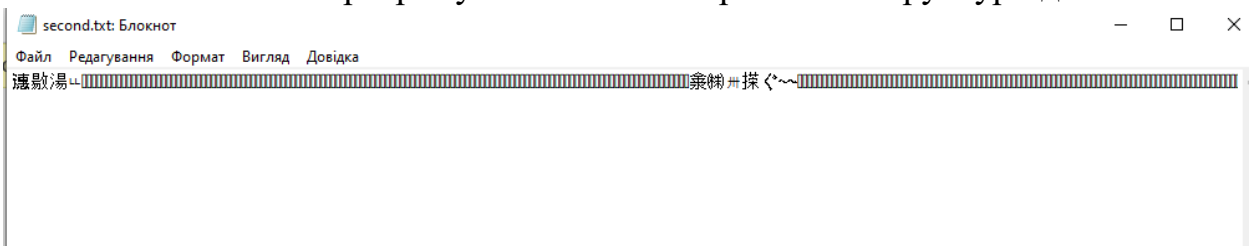
## Header.cpp

```
#include "Header.h"

Client::Client(string line) {
    strcpy_s(sec_name, split(line)[0].c_str());
    sex = split(line)[1][0];
    strcpy_s(date_of_birth, split(line)[2].c_str());
    amount = stoi(split(line)[3]);
    spent = amount * (100 - (this->GetAge() >= 60 ? this->GetAge() + 5 : this->GetAge()));
}

int Client::GetAge() {
    SYSTEMTIME current_date;
    GetLocalTime(&current_date);
    int day = stoi(split(date_of_birth, ':')[0]);
    int month = stoi(split(date_of_birth, ':')[1]);
    int year = stoi(split(date_of_birth, ':')[2]);
    return ((current_date.wMonth > month || current_date.wMonth == month &&
current_date.wDay >= day) ?
        (current_date.wYear - year) : (current_date.wYear - year - 1));
}

void inFile(string path, bool format) {
    ofstream ouf(path, ios::binary | (format ? ios::app : ios::trunc));
    cout << "Enter the amount of clients: ";
    int n; cin >> n;
    cin.ignore();
    string client;
    cout << "Enter the data in the following format:\n[name sex DD:MM:YYYY amount]\n";
    for (size_t i = 0; i < n; i++) {
        getline(cin, client);
        Client temp(client);
        ouf.write((char*)&temp, sizeof(Client));
    }
    ouf.close();
}

void inNewFile(string SourcePath, string DestinationPath) {
    ifstream inf(SourcePath, ios::binary);
    ofstream ouf(DestinationPath, ios::binary);
    Client temp;
    while (inf.read((char*)&temp, sizeof(Client))) {
        if (temp.spent > 250) {
            ouf.write((char*)&temp, sizeof(Client));
        }
    }
    inf.close();
    ouf.close();
}

void outFile(string path) {
    ifstream inf(path, ios::binary);
    Client temp;
    cout << endl << "----------------------------------------" << endl;
    while (inf.read((char*)&temp, sizeof(Client))) {
        cout << temp.sec_name << ' ' << temp.sex << ' ' << temp.date_of_birth << ' ' <<
temp.amount << endl;
    }
    inf.close();
    cout << "----------------------------------------" << endl;
}

int StoreProfit(string path) {
    ifstream inf(path, ios::binary);
    vector <Client> base;
```

```
    Client temp;
    while (inf.read((char*)&temp, sizeof(Client))) {
        base.push_back(temp);
    }
    int sum = 0;
    for (auto i : base) {
        sum += i.spent;
    }
    return sum;
}

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;
}
```





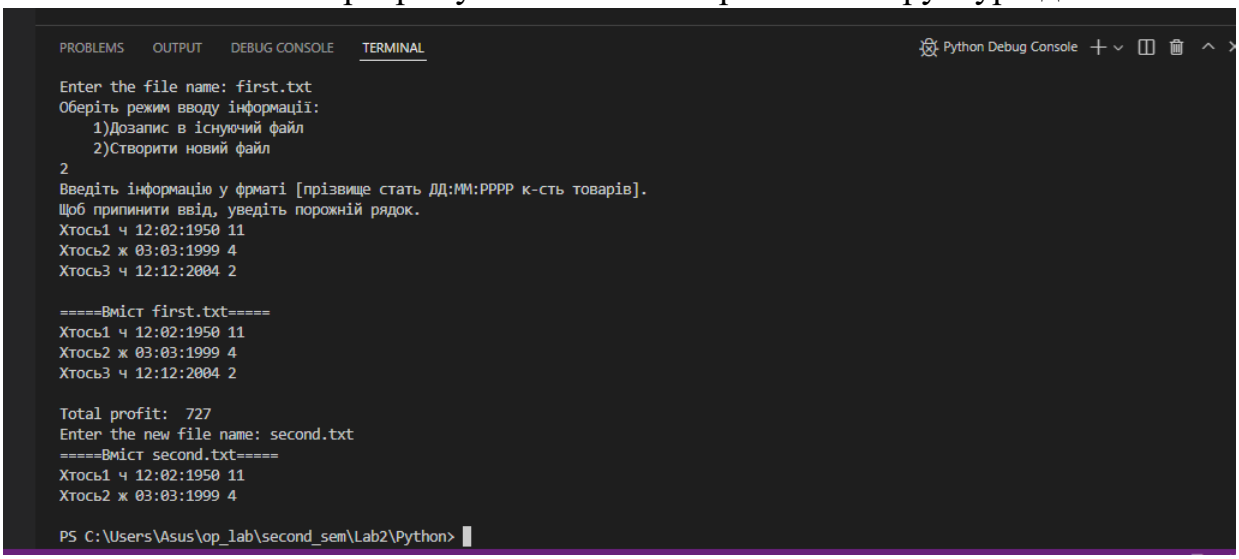

**main.py**

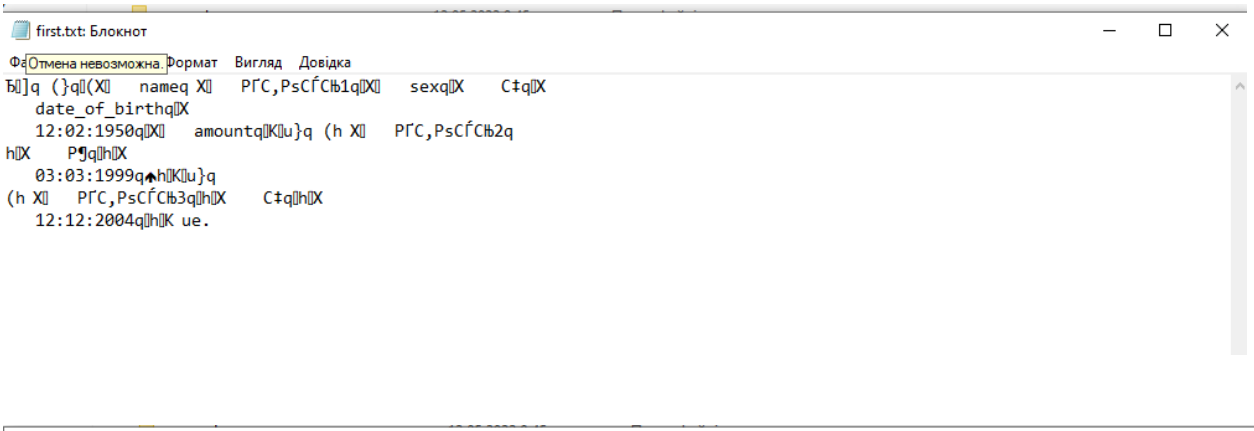
```
from Foo import *

file_name1 = input("Enter the file name: ")
create_file(file_name1)
print_file(file_name1)
print("Total profit: ", count_profit(file_name1))
file_name2 = input("Enter the new file name: ")
create_new_file(file_name1, file_name2)
print_file(file_name2)
```

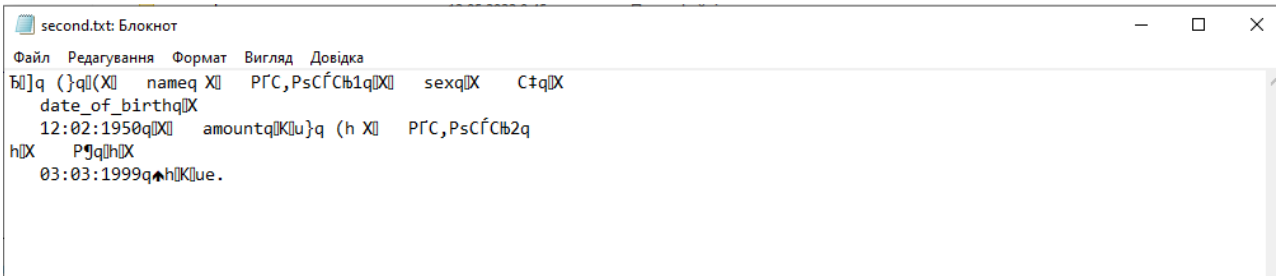
**Foo.py**

```
import pickle
import pickle
from datetime import date


def create_file(file_name: str):
    input_mode = int(input('''Оберіть режим вводу інформації:
    1)Дозапис в існуючий файл
    2)Створити новий файл\n'''))
    lst = list()
    if input_mode == 1:
        with open(file_name, "rb") as inf:
            lst = pickle.load(inf)
    with open(file_name, "wb") as ouf:
        line = input('''Введіть інформацію у фрматі [прізвище стать ДД:ММ:РРРР к-
сть товарів].
Щоб припинити ввід, уведіть порожній рядок.\n''').split()
        while (line):
            client = {
                'name': line[0],
                'sex': line[1],
                'date_of_birth': line[2],
                'amount': int(line[3]),
            }
            lst.append(client)
            line = input().split()
        pickle.dump(lst, ouf)


def create_new_file(source_file_name: str, destination_file_name: str):
    base = list()
    with open(source_file_name, "rb") as inf:
```

```
        base = pickle.load(inf)
    lst = list()
    for i in base:
        if spent(i['amount'], get_age(i['date_of_birth'])) > 250:
            lst.append(i)
    with open(destination_file_name, "wb") as ouf:
        pickle.dump(lst, ouf)


def print_file(file_name: str):
    try:
        with open(file_name, "rb") as file:
            print(f"=====Вміст {file_name}=====")
            text = pickle.load(file)
            for i in text:
                for values in i.values():
                    print(values, end=' ')
                print()
            print()
    except:
        print('Помилка при відкритті файлу!')


def count_profit(file_name: str):
    profit = 0
    with open(file_name, "rb") as inf:
        lst = pickle.load(inf)
        for i in lst:
            profit += spent(int(i['amount']), get_age(i['date_of_birth']))
    return profit


def get_age(date_of_birth: str):
    current_date = date.today().strftime("%d:%m:%Y").split(':')
    if (int(current_date[1]) > int(date_of_birth.split(':')[1])
            or int(current_date[1]) == int(date_of_birth.split(':')[1])
            and int(current_date[0]) >= int(date_of_birth.split(':')[0])):
        return int(current_date[2]) - int(date_of_birth.split(':')[2])
    else:
        return int(current_date[2]) - int(date_of_birth.split(':')[2]) - 1


def spent(amount: int, age: int):
    if age >= 60:
        return amount * (100 - age - 5)
    else:
        return amount * (100 - age)
```




**Висновки:** під час лабораторної роботи ми вивчили особливості створення і обробки текстових файлів даних.