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 № 1.2 з дисципліни «Основи програмування - 2. Методологія програмування»

«Бінарні файли»

**Варіант 10**

Виконав студент ІП-13 Дейнега Владислав Миколайович

Перевірив Вечерковська Анастасія Сергієвна

Київ 2022

# Лабораторна №1.2

## Бінарні файли

**Мета -** вивчити особливості створення і обробки бінарних файлів даних.

## Завдання

Створити файл із списком пацієнтів, записаних на прийом до лікаря: прізвище пацієнта, дата попереднього відвідування лікаря та час, на який пацієнт записаний. Видалити з файлу записи пацієнтів, час відвідування яких минув. Створити два нових файлів: в один занести відомості про вторинних пацієнтів(попереднє відвідування яких було протягом 10-ти останніх днів), а в другий - про решту пацієнтів.

# C++

## main

```
#include "CppHeader.h"


using namespace std;


int main()

{

	string Path = "AllCustomers.txt";

	string SecPath = "SecondCustomers.txt";

	string AnotherPath = "AnotherCustomers.txt";



	int choice;

	cout << "You want to edit or overwrite the file?\n1 - edit\n2 - 
owerwrite\n";

	cin >> choice;
```

```
	while (choice < 1 && choice > 2)
	{
		cout << "You enter the wrong number! Please enter 1 or 2";
		cin >> choice;
	}
	if (choice == 1) edit_file(Path);
	else create_file(Path);

	out_file_data(Path);
	cout << endl;

	delet_old_customers(Path);
	cout << "Customers list without old customers" << endl;
	out_file_data(Path);
	cout << endl;

	make_second_customers_list(Path, SecPath, AnotherPath);
	cout << "Second customers list" << endl;
	out_file_data(SecPath);
	cout << endl;

	cout << "Another customers list" << endl;
	out_file_data(AnotherPath);
	cout << endl;
}
```

## CppHeader.h

```
#pragma once
#include <iostream>
```

```
#include <fstream>
#include <string>
#include <vector>
#include <windows.h>

using namespace std;

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

struct Customer
{
	char secondname[40];
	Date lastVisit;
	Date NextVisit;
};

void edit_file(string);
void create_file(string);
vector<Customer>write_file_data_in_vector(string);
void out_file_data(string);
void delet_old_customers(string);
void make_second_customers_list(string, string, string);
```

```
Date date_fill();
void date_out(Date);
Customer customer_fill();
vector<Customer> vec_customer_fill();
void customer_out(Customer);
bool is_old_cust(Customer);
vector<Customer> vector_without_old_customers(string);
bool is_second_cust(Customer);
```

## CppFileWork

```
#include "CppHeader.h"


void edit_file(string path)
{
	ofstream writeFile(path, ios::binary || ios::app);
	if (!writeFile)
	{
		cout << "Error!";
	}
	else
	{
		vector<Customer> customers;
		customers = vec_customer_fill();
		for (int i = 0; i < customers.size(); i++)
		{
			writeFile.write((char*)&customers, sizeof(Customer));
		}
	}
```

```
	writeFile.close();
}

void create_file(string path)
{
	ofstream writeFile(path, ios::binary);

	if (!writeFile)
	{
		cout << "Error!";
	}
	else
	{
		vector<Customer> customers;
		customers = vec_customer_fill();
		for (int i = 0; i < customers.size(); i++)
		{
			writeFile.write((char*)&customers[i], sizeof(Customer));
		}
	}

	writeFile.close();
}

vector<Customer> write_file_data_in_vector(string path)
{
	vector<Customer> customers;
	Customer cust;
```

```
	ifstream readFile(path, ios::binary);
	if (!readFile)
	{
		cout << "Error!";
	}
	else
	{
		while (readFile.read((char*)&cust, sizeof(Customer)))
		{
			customers.push_back(cust);
		}
	}
	readFile.close();

	return customers;
}

void out_file_data(string path)
{
	vector<Customer> customers = write_file_data_in_vector(path);
	for (int i = 0; i < customers.size(); i++)
	{
		customer_out(customers[i]);
	}
}

void make_second_customers_list(string path, string secPath, string
anotherPath)
{
```

```
	ofstream writeSecFile(secPath, ios::binary);
	ofstream writeAnotherFile(anotherPath, ios::binary);

	vector<Customer> customers = write_file_data_in_vector(path);
	for (int i = 0; i < customers.size(); i++)
	{
		if (is_second_cust(customers[i]))
		{
			writeSecFile.write((char*)&customers[i],
sizeof(Customer));
		}
		else
		{
			writeAnotherFile.write((char*)&customers[i],
sizeof(Customer));
		}
	}
	writeAnotherFile.close();
	writeSecFile.close();
}
```

## CppCustomersWork.cpp

```
#include "CppHeader.h"

Customer customer_fill()
{
	Customer cust;
	cout << "Write secondname of castomer" << endl;
	cin >> cust.secondname;
	cout << "Write date of last visit like dd.mm.yyyy" << endl;
```

```
	cust.lastVisit = date_fill();
	cout << "Write date of next visit like dd.mm.yyyy" << endl;
	cust.NextVisit = date_fill();
	return cust;
}


Date date_fill()
{
	Date date;
	cin >> date.day;
	cin.ignore();
	cin >> date.month;
	cin.ignore();
	cin >> date.year;
	cin.ignore();
	while (date.day < 1 || date.day > 31 || date.month < 1 || date.month
> 12 || date.year < 0)
	{
		cout << "You write incorect date! Please, check your date and
write again." << endl;
		cin >> date.day;
		cin.ignore();
		cin >> date.month;
		cin.ignore();
		cin >> date.year;
		cin.ignore();
	}
	return date;
}
```

```
vector<Customer>  vector_without_old_customers(string path)
{
	ifstream readFile(path, ios::binary);
	vector<Customer> customers;
	customers = write_file_data_in_vector(path);
	if (!readFile)
	{
		cout << "Error!";
	}
	else
	{
		for (int i = 0; i < customers.size(); i++)
		{
			if (is_old_cust(customers[i]))
			{
				customers.erase(customers.begin() + i);
			}
		}
	}
	readFile.close();
	return customers;
}

void delet_old_customers(string path)
{
	vector<Customer> customers = vector_without_old_customers(path);
	ofstream writeFile(path, ios::binary);
```

```
	if (!writeFile)
	{
		cout << "Error!";
	}
	else
	{
		for (int i = 0; i < customers.size(); i++)
		{
			writeFile.write((const char*)&customers[i], 
sizeof(Customer));
		}
	}

	writeFile.close();
}

bool is_old_cust(Customer cust)
{
	SYSTEMTIME time;
	GetLocalTime(&time);
	bool chek = false;
	if (cust.NextVisit.year < time.wYear)
	{
		chek = true;
	}
	else
	{
		if (cust.NextVisit.year == time.wYear)
```

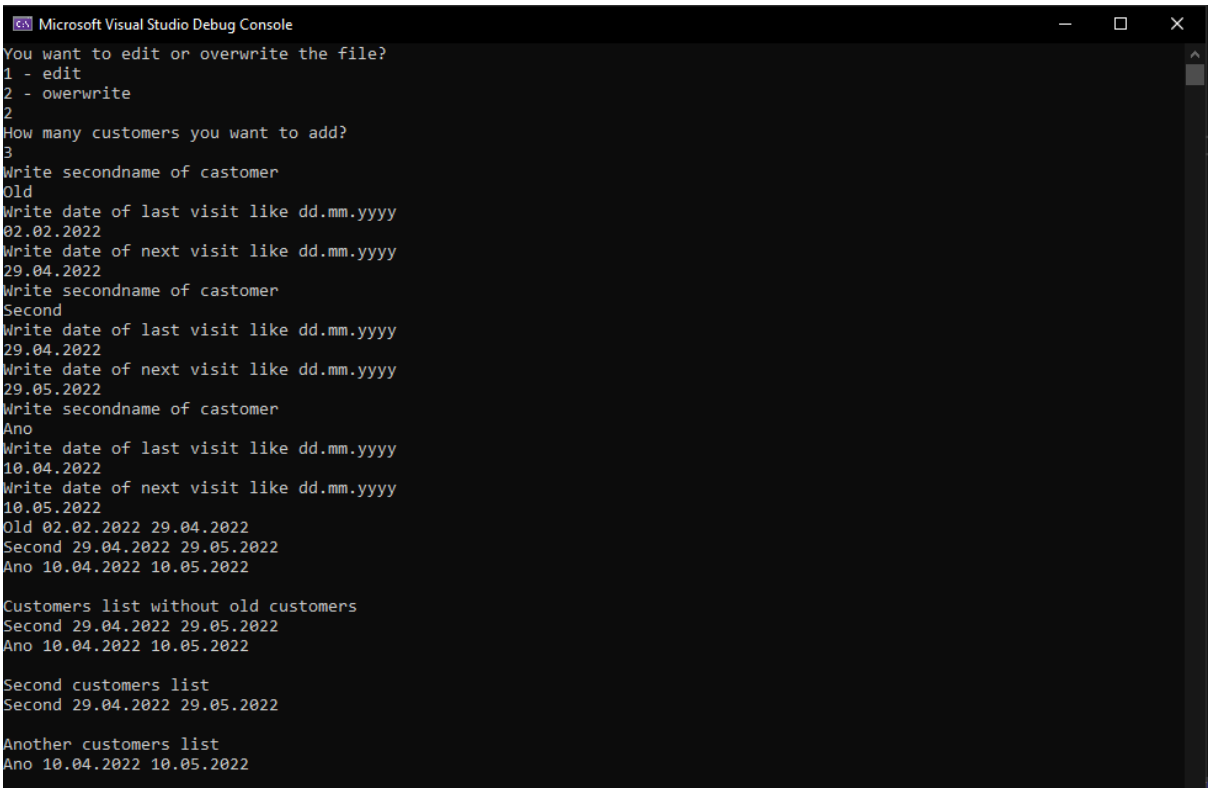
```
            {
                  if (cust.NextVisit.month < time.wMonth)
                  {
                        chek = true;
                  }
                  else
                  {
                        if (cust.NextVisit.month == time.wMonth)
                        {
                              if (cust.NextVisit.day < time.wDay)
                              {
                                    chek = true;
                              }
                        }
                  }
            }
      }
      return chek;
}


void date_out(Date date)
{
      if (date.day >= 1 && date.day < 10) cout << '0' << date.day << '.';
      else cout << date.day << '.';
      if (date.month >= 1 && date.month < 10) cout << '0' << date.month <<
'.';
      else cout << date.month << '.';
      cout << date.year;
}
```

```
void customer_out(Customer cust)
{
	cout << cust.secondname  << " ";
	date_out(cust.lastVisit);
	cout << " ";
	date_out(cust.NextVisit);
	cout << endl;
}

vector<Customer> vec_customer_fill()
{
	int count_cust;
	cout << "How many customers you want to add?\n";
	cin >> count_cust;

	vector<Customer> customers;
	for (int i = 0; i < count_cust; i++)
	{
		customers.push_back(customer_fill());
	}

	return customers;
}

bool is_second_cust(Customer cust)
{
	SYSTEMTIME time;
```

```
	GetLocalTime(&time);
	bool chek = false;
	if (cust.lastVisit.month == time.wMonth)
	{
		if (time.wDay - cust.lastVisit.day <= 10)
		{
			chek = true;
		}
	}
	else
	{
		if (time.wMonth - cust.lastVisit.month == 1)
		{
			if (cust.lastVisit.month == 2)
			{
				if (28 - cust.lastVisit.day + time.wDay <= 10)
				{
					chek = true;
				}
			}
			else if(cust.lastVisit.month == 1 ||
cust.lastVisit.month == 3 || cust.lastVisit.month == 5 ||
cust.lastVisit.month == 7 || cust.lastVisit.month == 8 ||
cust.lastVisit.month == 10 || cust.lastVisit.month == 12)
			{
				if (31 - cust.lastVisit.day + time.wDay <= 10)
				{
					chek = true;
				}
```

```
                }
                else
                {
                        if (30 - cust.lastVisit.day + time.wDay <= 10)
                        {
                                chek = true;
                        }
                }
        }
    }
    return chek;
}
```

# Тестування



# Python

## main

```
from PYFileWork import*


path = "Path.txt"
SecPath = "SecondCustomers.txt"
AnotherPath = "AnotherCustomers.txt"

print("You want to edit or overwrite the file?\n1 - edit\n2 - owerwrite")
choice = int(input())
while choice < 1 or choice > 2:
    print("You enter the wrong number! Please enter 1 or 2")
    choice = int(input())
if choice == 1:
    edit_file(path)
else:
    create_file(path)
out_file_data(path)
print()

delet_old_customers(path)
print("Customers list without old customers")
out_file_data(path)
print()

make_second_customers_list(path,SecPath,AnotherPath)
print("Second customers list")
out_file_data(SecPath)
print()
print("Another customers list")
out_file_data(AnotherPath)
print()
```

## PYFileWork.py

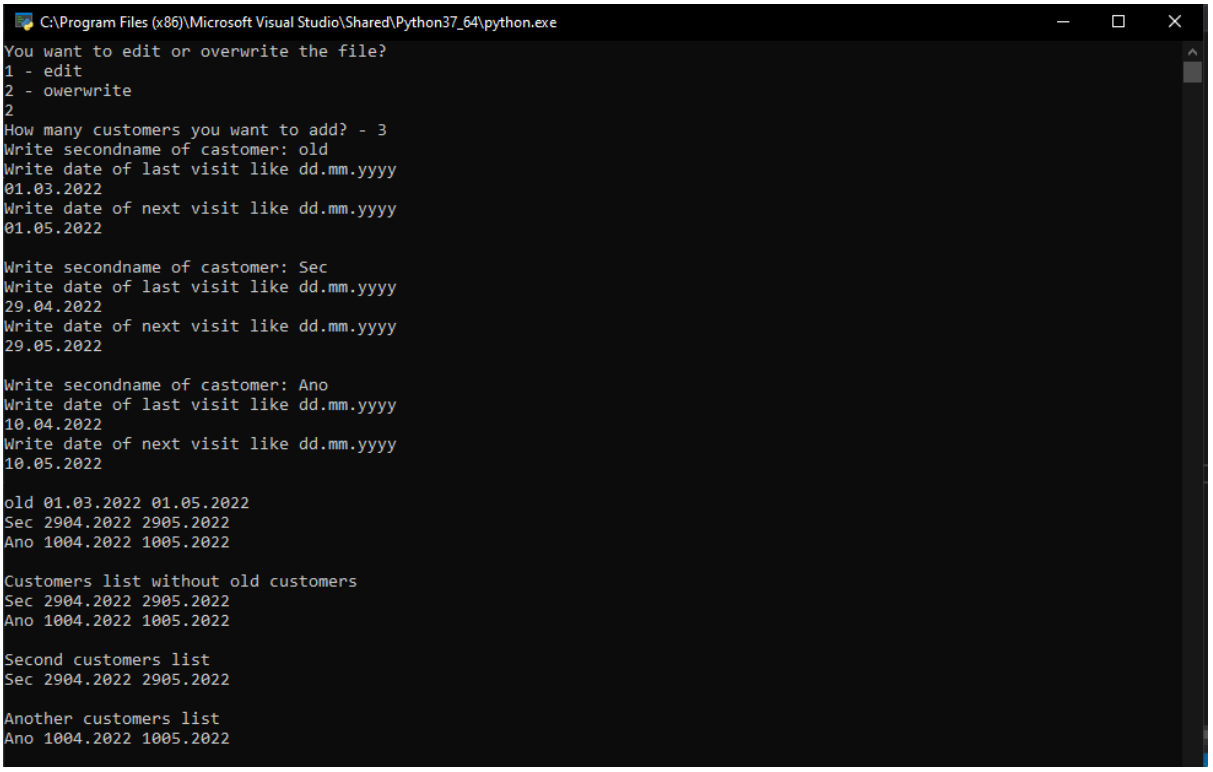
```
import pickle
from datetime import date

def edit_file(path):
    with open(path, 'ab') as writeFile:
        customers = []
        numb = int(input("How many customers you want to add? - "))
        for i in range(numb):
            cust = customer_fill()
            customers.append(cust)
```

```
            print()
        for cust in customers:
            pickle.dump(cust, writeFile)

def create_file(path):
    with open(path, 'wb') as writeFile:
        customers = []
        numb = int(input("How many customers you want to add? - "))
        for i in range(numb):
            cust = customer_fill()
            customers.append(cust)
            print()
        for cust in customers:
            pickle.dump(cust, writeFile)

def write_file_data_in_list(path):
    customers = []
    with open(path, 'rb') as readFile:
        readFile.seek(0, 2)
        end = readFile.tell()
        readFile.seek(0, 0)
        while readFile.tell() != end:
            cust = pickle.load(readFile)
            customers.append(cust)
    return customers

def out_file_data(path):
    customers = write_file_data_in_list(path)
    for cust in customers:
        customer_out(cust)

def make_second_customers_list(path, secPath, anotherPath):
    customers =  write_file_data_in_list(path)
    with open(secPath, 'wb') as writeSecFile:
        with open(anotherPath, 'wb') as writeAnotherFile:
            for cust in customers:
                if is_second_cust(cust):
                    pickle.dump(cust, writeSecFile)
                else:
                    pickle.dump(cust, writeAnotherFile)

def delet_old_customers(path):
    customers = []
    customers = list_without_old_cust(path)
    with open(path, 'wb') as writeFile:
        for cust in customers:
            pickle.dump(cust, writeFile)
```

```
def customer_fill():
    customer ={'secondname' : input("Write secondname of castomer: "),
               'lastVisit' : date_fill("last visit"),
               'nextVisit' : date_fill("next visit")}
    return customer

def date_fill(dateName):
    print("Write date of", dateName , "like dd.mm.yyyy")
    str = input()
    str = str.split('.')
    date ={'day' : int(str[0]),
           'month' : int(str[1]),
           'year' : int(str[2])}
    while date['day'] < 1 or date['day'] > 31 or date['month'] < 1 or
date['month'] > 12 or date['year'] < 1:
        print("You write incorect date! Please, check your date and write
again.")
        str = input("Write date of", dateName , "like dd.mm.yyyy")
        str = str.split('.')
        date['day'] = str[0]
        date['month'] = str[1]
        date['year'] = str[2]
    return date

def date_out(date):
    if date['day'] >= 1 and date['day'] < 10:
        print('0' + str(date['day']) + '.', end = '')
    else:
        print(date['day'], end = '')
    if date['month'] >= 1 and date['month'] < 10:
        print('0' + str(date['month']) + '.', end = '')
    else:
       print(date['month'], end = '')
    print(date['year'], end=' ')

def customer_out(cust):
    print(cust['secondname'], end =' '),
    date_out(cust['lastVisit'])
    date_out(cust['nextVisit'])
    print()

def is_second_cust(cust):
```

```
    local_time = date.today()
    chek = False
    if cust['lastVisit']['month'] == local_time.month:
        if local_time.day - cust['lastVisit']['day'] <= 10:
            chek = True
    elif local_time.month - cust['lastVisit']['month'] == 1:
        if cust['lastVisit']['month'] == 2:
            if 28 - cust['lastVisit']['day'] + local_time.day <= 10:
                chek = True
        elif cust['lastVisit']['month'] == 1 or cust['lastVisit']['month']
== 3 or cust['lastVisit']['month'] == 5 or cust['lastVisit']['month'] == 7
or cust['lastVisit']['month'] == 8 or cust['lastVisit']['month'] == 10 or
cust['lastVisit']['month'] == 12:
            if 31 - cust['lastVisit']['day'] + local_time.day <= 10:
                chek = True
        else:
            if 30 - cust['lastVisit']['day'] + local_time.day <= 10:
                chek = True
    return chek

def is_old_cust(cust):
    local_time = date.today()
    chek = False
    if cust['nextVisit']['year'] < local_time.year:
        chek = True
    elif cust['nextVisit']['year'] == local_time.year:
        if cust['nextVisit']['month'] < local_time.month:
            chek = True
        elif cust['nextVisit']['month'] == local_time.month:
            if cust['nextVisit']['day'] < local_time.day:
                chek = True
    return chek

def list_without_old_cust(path):
    customers = []
    customers = write_file_data_in_list(path)
    i = 0
    for cust in customers:
        if is_old_cust(cust):
            customers.pop(i)
        else:
            i += 1
    return customers
```

# Тестування




## Висновок

Під час виконання лабораторної роботи я навчився створювати та обробляти бінарні файли на мовах Python та C++, та особливості роботи з ними.