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. Модульне програмування»

«Файли даних. Бінарні файли»

Варіант 17

Виконав студент ІП-11 Куценко Артемій Ілліч

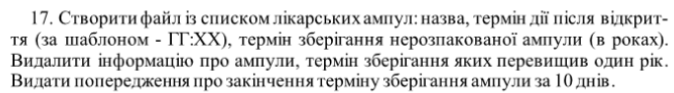
(шифр, прізвище, ім'я, по батькові)

Перевірив Вітковська Ірина Іванівна

(прізвище, ім'я, по батькові)

Київ 2022

***Умова задачі:***



***Код на С++:***

**MAIN.CPP**

#include "pch.h"

int main(){

const char\* drugsFile1 = "drugs.bin";

const char\* drugsFile2 = "drugsEdit.bin";

vector<ampula> ampuls;

writeInFile(drugsFile1);

readOutFile(drugsFile1);

writeInVector(drugsFile1, ampuls);

del(drugsFile2, ampuls);

readOutFile(drugsFile2);

warning(ampuls);

system("pause>0");

}

**PCH.H**

#pragma once

#include "structs.h"

#include <string>

#include <iostream>

#include <iomanip>

#include <fstream>

#include <vector>

#include <cstdio>

#include <stdio.h>

#include <ctime>

using namespace std;

string toLower(string &str);

void writeInFile(const char\* fileName);

void readOutFile(const char\* fileName);

void writeInVector(const char\* fileName, vector<ampula>& ampuls);

void del(const char\* fileName, vector<ampula>& ampuls);

void warning(vector<ampula>& ampuls);

**PCH.CPP**

#include "pch.h"

string toLower(string& str) {

string buffer;

for (int i = 0; i < str.length(); i++) {

if (isupper(str[i])) {

buffer += tolower(str[i]);

}

else {

buffer += str[i];

}

}

return str = buffer;

}

void writeInVector(const char\* fileName, vector<ampula>& ampuls) {

ifstream outFile(fileName, ios::binary);

ampula ampula0;

while (outFile.read((char\*)&ampula0, sizeof(ampula0))) {

ampuls.push\_back(ampula0);

}

}

void writeInFile(const char\* fileName) {

int answer;

cout << "What do you want?\n\t(1)create a new file or\n\t(2)add to an existing file\n";

cin >> answer;

ofstream inFile;

do {

if (answer == 1) {

inFile.open(fileName, ios::binary);

}

else if (answer == 2) {

inFile.open(fileName, ios::binary | ios::app);

}

else {

cout << "Invalid response, please try again: ";

cin >> answer;

}

} while (answer != 1 && answer != 2);

ampula ampula0;

string res;

if (!inFile.is\_open()) {

cout << fileName << " cannot be opened\n";

} else {

cout << ">>> " << fileName << " to WRITE was opened successfully\n";

cout << " Input ampuls parameters:\n\n";

while (toLower(res) != "no") {

if (res == "yes" || res=="") {

cout << "Name: ";

cin >> ampula0.name;

cout << "Date of production(DD MM YYYY) : ";

cin >> ampula0.productionDay >> ampula0.productionMonth >> ampula0.productionYear;

cout << "How long can be used after opening(HH:MM): ";

cin >> ampula0.hoursAfterOpening;

cin >> ampula0.minsAfterOpening;

cout << "ShelfTime: ";

cin >> ampula0.shelfTime;

cout << "\n";

inFile.write((char\*)&ampula0, sizeof(ampula0));

cout << "Do you want to add another ampula(yes or no)? -";

cin >> res;

} else{

cout << "Invalid response, try again please -";

cin >> res;

}

}

inFile.close();

cout << "\n<<< " << fileName << " was closed successfully\n\n\n";

}

}

void readOutFile(const char\* fileName) {

ifstream outFile(fileName, ios::binary);

ampula ampula0;

if (!outFile.is\_open()) {

cout << fileName << " cannot be opened\n";

}

else {

cout << ">>> " << fileName << " to READ was opened successfully\n";

cout << " Your ampuls: \n\n";

while(outFile.read((char\*)&ampula0, sizeof(ampula0))) {

cout << "Name: " << ampula0.name << '\n';

cout << "Date of production : ";

cout.fill('0');

cout.width(2);

cout << ampula0.productionDay << ".";

cout.fill('0');

cout.width(2);

cout << ampula0.productionMonth << ".";

cout << ampula0.productionYear << "\n";

cout << "How long can be used after opening(HH:MM): ";

cout.fill('0');

cout.width(2);

cout << ampula0.hoursAfterOpening << ':';

cout.fill('0');

cout.width(2);

cout << ampula0.minsAfterOpening << "\n";

cout << "ShelfTime: " << ampula0.shelfTime << "\n\n";

}

cout << "<<< " << fileName << " was closed successfully\n\n\n";

}

outFile.close();

}

void del(const char\* fileName, vector<ampula>& ampuls) {

ofstream inFile(fileName, ios::binary);

if (!inFile.is\_open()) {

cout << fileName << " cannot be opened\n";

} else {

for (int i = ampuls.size()-1; i >= 0; i--) {

if (ampuls[i].shelfTime > 1) {

ampuls.erase(ampuls.begin() + i);

}

}

for (int i = 0; i < ampuls.size(); i++) {

inFile.write((char\*)&ampuls[i], sizeof(ampuls[i]));

}

cout << "----------------------------------------------\n";

cout << "--ALL AMPULS WITH SHELFTIME > 1 WERE DELETED--\n";

cout << "----------------------------------------------\n\n";

}

inFile.close();

}

void warning(vector<ampula>& ampuls) {

int months[12] = {31,28,31,30,31,30,31,31,30,31,30,31};

int prodDay;

int prodMonth;

int prodYear;

int shelfDays = 0;

int daysPassed;

int thisYear;

int timeLeft;

time\_t ttime = time(0);

tm\* local\_time = localtime(&ttime);

for (int i = 0; i < ampuls.size(); i++) {

prodDay = ampuls[i].productionDay;

prodMonth = ampuls[i].productionMonth;

prodYear = ampuls[i].productionYear;

for (int j = 0; j < prodMonth - 1; j++) {

prodDay += months[j];

}

thisYear = local\_time->tm\_year + 1900;

shelfDays = round(ampuls[i].shelfTime \* 365);

daysPassed = -prodDay + (thisYear - prodYear) \* 365 + local\_time->tm\_yday + 1;

timeLeft = shelfDays - daysPassed;

if (timeLeft <= 0) {

cout << "!!! " << ampuls[i].name << " is expired!!!\n";

} else if(timeLeft <= 10) {

cout << "!!! " << ampuls[i].name << " is expiring in " << timeLeft << " days!!!\n";

}

}

cout << '\n';

}

**STRUCTS.H**

#pragma once

struct ampula {

char name[30];

int productionDay;

int productionMonth;

int productionYear;

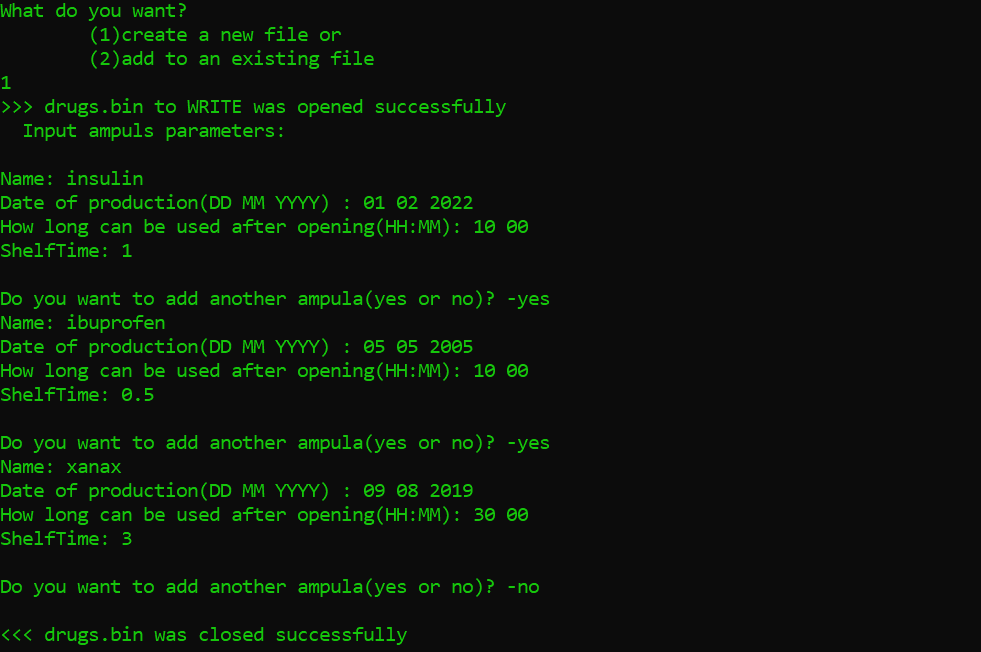
int hoursAfterOpening;

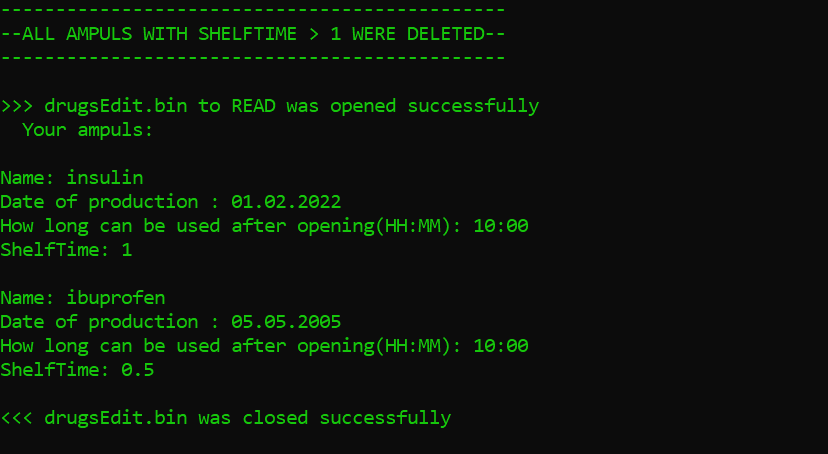
int minsAfterOpening;

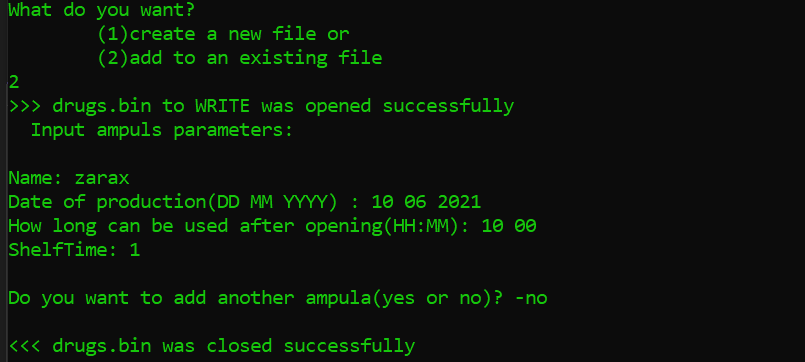
float shelfTime;

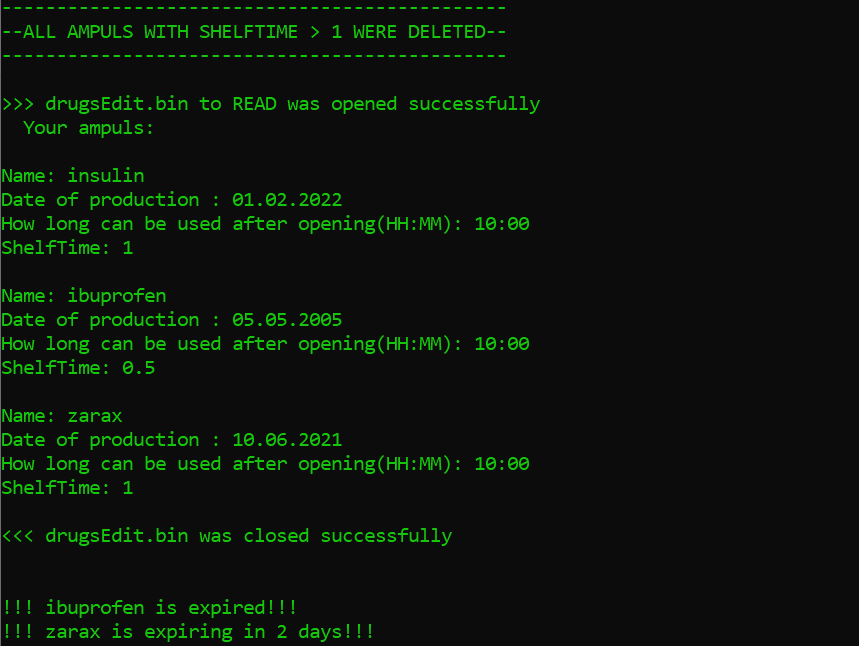
};

**Скріни на С++:**









***Код на Python:***

**opSem2Lab1PY.py**

from funcs import \*

fileName1 = 'drugs.bin'

fileName2 = 'drugsEdit.bin'

writeInFile(fileName1)

readOutFile(fileName1)

delMoreThan1Year(fileName1, fileName2)

readOutFile(fileName2)

warning(fileName2)

**FUNCS.PY**

import pickle

from datetime import datetime

def writeInFile(fileName: str):

answer = int(input("What do you want?\n\t(1)create a new file or\n\t(2)add to an existing file\n"))

b = True

while(b):

if answer==1:

inFile = open(fileName, 'wb')

b = False

elif answer==2:

inFile = open(fileName, 'ab')

b = False

else:

answer = int(input("Invalid response, please try again: "))

print(">>> " + fileName + " to WRITE was opened successfully\n\n")

print("--Input ampuls parameters:\n")

res = ""

ampulList=[]

while( res.lower()!="no" ):

if res == "yes" or res == "":

name = input("Name: ")

date = input("Date of production: ")

day = int(date[0:2])

month = int(date[3:5])

year = int(date[6:10])

time = input("How long can be used after opening(HH:MM): ")

shelfTime = float(input("ShelfTime: "))

ampula = { 'name' : name,

'date' : date,

'day' : day,

'month' : month,

'year' : year,

'time' : time,

'shelfTime' : shelfTime}

ampulList.append(ampula)

res = input("\nDo you want to add another ampula(yes or no)? -")

print("")

else:

res = input("\nInvalid response, try again please -")

print("")

pickle.dump(ampulList, inFile)

inFile.close()

print("<<< " + fileName + " to WRITE was closed successfully\n\n\n")

################################

def readOutFile(fileName: str):

with open(fileName, 'rb') as outFile:

ampulList = pickle.load(outFile)

print(">>> " + fileName + " to READ was opened successfully\n")

i=0

while(i<len(ampulList)):

print("Name: " + ampulList[i]['name'])

print("Date of production(DD.MM.YYYY): " + ampulList[i]['date'])

print("How long can be used after opening(HH:MM): " + ampulList[i]['time'])

print("ShelfTime: " + str(ampulList[i]['shelfTime']))

print('')

i+=1

outFile.close()

print("<<< " + fileName + " to READ was closed successfully\n\n")

#####

def delMoreThan1Year(fileName1: str, fileName2: str):

with open(fileName1, 'rb') as outFile:

ampulList = pickle.load(outFile)

inFile = open(fileName2, 'wb')

ampulList2=[]

i=0

while(i<len(ampulList)):

if(ampulList[i]['shelfTime']<=1):

ampulList2.append(ampulList[i])

i+=1

else:

i+=1

pickle.dump(ampulList2, inFile)

outFile.close()

inFile.close()

#############

def warning(fileName: str):

months = [31,28,31,30,31,30,31,31,30,31,30,31]

with open(fileName, 'rb') as outFile:

ampulList = pickle.load(outFile)

i=0

while(i<len(ampulList)):

prodDay = ampulList[i]['day']

prodMonth = ampulList[i]['month']

prodYear = ampulList[i]['year']

k=0

while(k<prodMonth - 1):

prodDay+=months[k]

k+=1

TIME0 = datetime.now()

thisDay=TIME0.day

thisMonth=TIME0.month

thisYear=TIME0.year

l=0

while(l<thisMonth - 1):

thisDay+=months[l]

l+=1

shelfDays = round(ampulList[i]['shelfTime'] \* 365)

daysPassed = -prodDay + (thisYear - prodYear) \* 365 + thisDay + 1

timeLeft = shelfDays - daysPassed

if(timeLeft<=0):

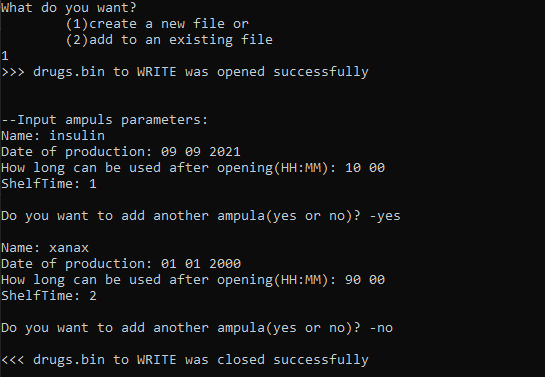
print("!!!" + ampulList[i]['name'] + " is expired!!!")

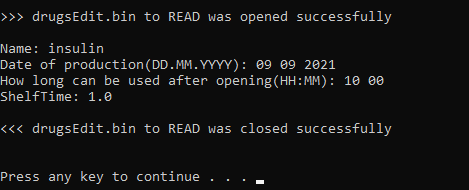
elif(timeLeft<=10):

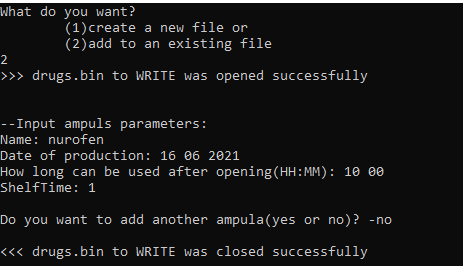
print("!!!" + ampulList[i]['name'] + " is expiring in " + str(timeLeft + 1) + " days !!!")

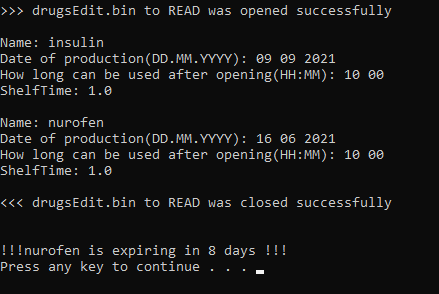
i+=1

**Скріни на Пітоні:**









**Висновок: П**ід час виконання даної лабораторної роботи Я успішно розробив програму для вводу та виводу з файла в двох режимах: створення нового файлу та допису в існуючий, навчився навичок обробки бінарних файлів.