Міністерство освіти і науки України

Національний технічний університет України “Київський політехнічний інститут імені Ігоря Сікорського”

Факультет інформатики та обчислювальної техніки

Кафедра інформатика та програмної інженерії

Звіт

З лабораторної роботи № 2 з дисципліни “Основи програмування - 2” “Бінарні файли”

Варіант 2

Виконав: ІП-13 Бабашев Олексій

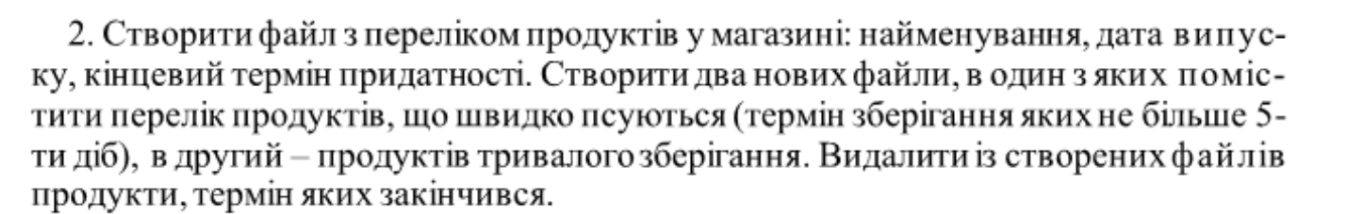
Перевірила: Вєчерковська А. С.

Київ - 2022

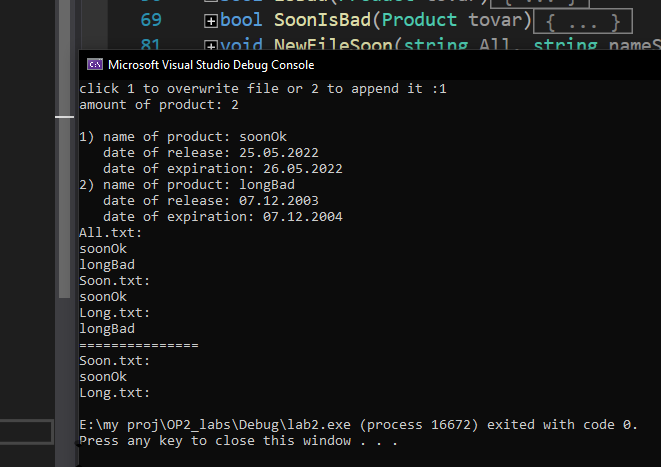
**Лабораторна робота 2**

**Бінарні Файли**

**Індивідуальне завдання:**



**Виконання програми в терміналі та код на С++:**

****

**Вихідний код:**

**lab2.cpp**

#include "Function.h"

int main()

{

string nameAll = "All.txt";

string nameSoon = "Soon.txt";

string nameLong = "Long.txt";

char filemode = ChooseFilemode();

createFile(nameAll, filemode);

displayFile(nameAll);

NewFileSoon(nameAll, nameSoon, filemode);

displayFile(nameSoon);

NewFileLong(nameAll, nameLong, filemode);

displayFile(nameLong);

cout << "===============\n";

deleteExpired(nameSoon);

displayFile(nameSoon);

deleteExpired(nameLong);

displayFile(nameLong);

}

**Function.cpp**

#include "Function.h"

char ChooseFilemode() {

char res;

do

{

cout << "click 1 to overwrite file or 2 to append it: "; cin >> res;

} while (res!='1' && res != '2');

return res;

}

void createFile(string name,char filemode) {

ofstream File;

if (filemode=='1')

{

File.open(name, ios::binary);

}

else {

File.open(name, ios::binary | ios::app);

}

if (!File.is\_open())

{

cout << "fail\n";

return;

}

Product tovar;

size\_t amount;

cout << "amount of product: "; cin >> amount; cout << endl;

for (size\_t i = 0; i < amount; i++) {

cout << i + 1 << ") " << "name of product: "; cin >> tovar.name;

cout << " " << "date of release: ";

string date;

getline(cin, date, '.'); tovar.start.day = stoi(date);

getline(cin, date, '.'); tovar.start.month = stoi(date);

getline(cin, date); tovar.start.year = stoi(date);

cout <<" " << "date of expiration: ";

getline(cin, date, '.'); tovar.end.day = stoi(date);

getline(cin, date, '.'); tovar.end.month = stoi(date);

getline(cin, date); tovar.end.year = stoi(date);

File.write((char\*)&tovar, sizeof(Product));

}

File.close();

}

bool isBad(Product tovar) {

bool res = false;

CTime current = CTime::GetCurrentTime();

CTime end(tovar.end.year, tovar.end.month, tovar.end.day, 0, 0, 0);

CTimeSpan diff = end - current;

int days = diff.GetDays();

if (days<0)

{

res = true;

}

return res;

}

bool SoonIsBad(Product tovar) {

bool res = false;

CTime start(tovar.start.year, tovar.start.month, tovar.start.day, 0, 0, 0);

CTime end(tovar.end.year, tovar.end.month, tovar.end.day, 0, 0, 0);

CTimeSpan diff = end - start;

int days = diff.GetDays();

if (days<=5)

{

res = true;

}

return res;

}

void NewFileSoon(string All, string nameSoon,char filemode) {

ifstream FileAll(All, ios::binary);

if (!FileAll.is\_open())

{

cout << "fail\n";

return;

}

ofstream FileSoon;

if (filemode == '1')

{

FileSoon.open(nameSoon, ios::binary);

}

else {

FileSoon.open(nameSoon, ios::binary | ios::app);

}

if (!FileSoon.is\_open())

{

cout << "fail\n";

return;

}

Product tovar;

while (FileAll.read((char\*)&tovar, sizeof(Product)))

{

if (SoonIsBad(tovar))

{

FileSoon.write((char\*)&tovar, sizeof(Product));

}

}

FileSoon.close();

FileAll.close();

}

void NewFileLong(string All,string nameLong,char filemode) {

ifstream FileAll(All, ios::binary);

if (!FileAll.is\_open())

{

cout << "fail\n";

return;

}

ofstream FileLong;

if (filemode == '1')

{

FileLong.open(nameLong, ios::binary);

}

else {

FileLong.open(nameLong, ios::binary | ios::app);

}

if (!FileLong.is\_open())

{

cout << "fail\n";

return;

}

Product tovar;

while (FileAll.read((char\*)&tovar, sizeof(Product)))

{

if (!SoonIsBad(tovar))

{

FileLong.write((char\*)&tovar, sizeof(Product));

}

}

FileLong.close();

FileAll.close();

}

void displayFile(string name) {

ifstream File(name, ios::binary);

Product tovar;

if (!File.is\_open())

{

cout << "fail\n";

return;

}

cout << name << ':' << '\n';

while (File.read((char\*)&tovar, sizeof(Product))) {

cout << tovar.name<<'\n';

}

}

void deleteExpired(string name) {

ifstream FileIn(name, ios::binary);

vector <Product> buff;

Product tovar;

while (FileIn.read((char\*)&tovar, sizeof(Product)))

{

if (isBad(tovar)==false) {

buff.push\_back(tovar);

}

}

FileIn.close();

ofstream FileOut(name, ios::binary);

for (auto& buffered\_tovar: buff)//size\_t i = 0; i<buff.size();i++

{

FileOut.write((char\*)&buffered\_tovar, sizeof(Product)); //buff[i]

}

FileOut.close();

}

**Function.h**

#pragma once

#include <iostream>

#include<vector>

#include<string>

#include<fstream>

#include "Product.h"

#include<atltime.h>

using namespace std;

void createFile(string name,char filemode);

char ChooseFilemode();

void NewFileSoon(string All, string nameSoon,char filemode);

void NewFileLong(string All, string nameLong,char filemode);

bool isBad(Product tovar);

bool SoonIsBad(Product tovar);

void displayFile(string name);

void deleteExpired(string name);

**Product.h**

#pragma once

struct Date

{

int day;

int month;

int year;

};

struct Product

{

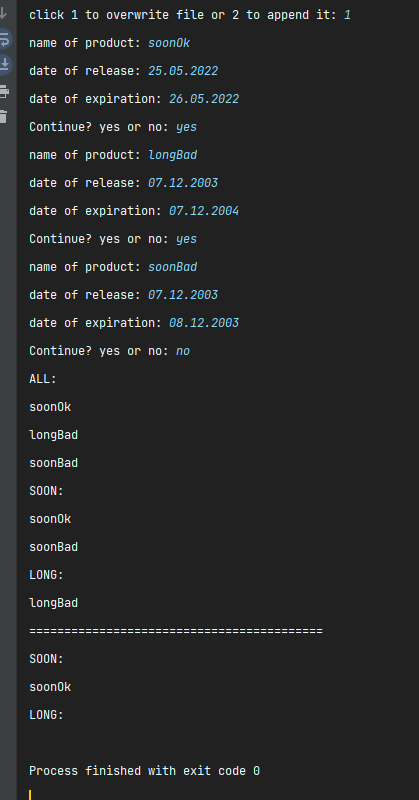
char name[15];

Date start;

Date end;

};

# Виконання програми в терміналі та код на Python:



# main.py

*from* module *import* \*  
fileAll = "All.txt"  
fileSoon = "Soon.txt"  
fileLong = "Long.txt"  
  
WantToClear(fileAll, fileSoon, fileLong)  
products = NewFile(fileAll)  
Diffs = ListOfDiffs(products)  
  
CreatingSoonLong(fileSoon, fileLong, products, Diffs)  
  
*print*("ALL:")  
displayFile(fileAll)  
*print*("SOON:")  
displayFile(fileSoon)  
*print*("LONG:")  
displayFile(fileLong)  
  
*print*("==========================================")  
  
deleteExpired(fileSoon, fileLong, products, Diffs)  
  
*print*("SOON:")  
displayFile(fileSoon)  
*print*("LONG:")  
displayFile(fileLong)

# module.py

*import* pickle  
*from* datetime *import* \*  
  
*def* NewFile(fileName):  
 *with open*(fileName, 'ab') *as* file:  
 products = []  
 *while True*:  
 product = {  
 "name":*str*(*input*("name of product: ")),  
 "start":*str*(*input*("date of release: ")),  
 "end":*str*(*input*("date of expiration: "))  
 }  
 products.append(product)  
 res = *str*(*input*("Continue? yes or no: "))  
 *while* res != 'yes' *and* res != 'no':  
 res = *str*(*input*("Press yes or no: "))  
 *if* res == "no":  
 *break* pickle.dump(products,file)  
 *return* products  
  
*def* WantToClear(file1, file2, file3):  
 res = *input*("click 1 to overwrite file or 2 to append it: ")  
 *while* res != '1' *and* res != '2':  
 res = *str*(*input*("click 1 to overwrite file or 2 to append it: "))  
 *if* res == "1":  
 clearFile(file1)  
 clearFile(file2)  
 clearFile(file3)  
  
*def* isOutOfDate(product):  
 DateOfToday = date.today()  
 startdate = DateOfToday.strftime("%d.%m.%Y")  
 lastdate = product["end"]  
 start = datetime.strptime(startdate, "%d.%m.%Y")  
 end = datetime.strptime(lastdate, "%d.%m.%Y")  
 day = end - start  
 Days = day.days  
 *if* Days < 0:  
 *return True  
 else*:  
 *return False  
  
def* daysDiff(product):  
 firstdate = product["start"]  
 lastdate = product["end"]  
 t1 = datetime.strptime(firstdate, "%d.%m.%Y")  
 t2 = datetime.strptime(lastdate, "%d.%m.%Y")  
 day = t2-t1  
 *return* day.days  
  
*def* ListOfDiffs(products):  
 diffs = []  
 *for* product *in* products:  
 day = daysDiff(product)  
 diffs.append(day)  
 *return* diffs  
  
*def* CreatingSoonLong(fileSoon, fileLong, products, diffs):  
 file1 = *open*(fileSoon, 'ab')  
 file2 = *open*(fileLong, 'ab')  
 soonProd = []  
 longProd = []  
 *for* i *in range*(0, *len*(diffs)):  
 *if* diffs[i] <=5:  
 soonProd.append(products[i])  
 *else*:  
 longProd.append(products[i])  
 pickle.dump(soonProd, file1)  
 pickle.dump(longProd, file2)  
 file1.close()  
 file2.close()  
  
  
*def* deleteExpired(fileSoon, fileLong, products, diffs):  
 file1 = *open*(fileSoon, 'wb')  
 file2 = *open*(fileLong, 'wb')  
 soonProd = []  
 longProd = []  
 *for* i *in range*(0, *len*(diffs)):  
 *if* isOutOfDate(products[i]):  
 *continue  
 else*:  
 *if* diffs[i] <=5:  
 soonProd.append(products[i])  
 *else*:  
 longProd.append(products[i])  
  
 pickle.dump(soonProd, file1)  
 pickle.dump(longProd, file2)  
 file2.close()  
 file1.close()  
*def* readPickle(fileName):  
 *with open*(fileName, 'rb') *as* file:  
 *while True*:  
 *try*:  
 *yield* pickle.load(file)  
 *except EOFError*:  
 *break  
  
def* clearFile(file):  
 *with open*(file,'wb') *as* clear\_file:  
 *pass  
  
def* displayFile(fileName):  
 *for* product *in* readPickle(fileName):  
 *for* item *in* product:  
 *print*(item["name"])

**Висновок:** Вивчив особливості створення і обробки бінарних файлів. Застосував отриманні знання на практиці.