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»

Варіант 25

Виконав студент ІП-14 Радзівіло Валерія Артемівна

(шифр, прізвище, ім'я, по батькові)

Перевірив

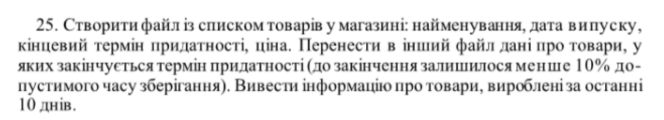
( прізвище, ім'я, по батькові)

Київ 2022

**Лабораторна робота №2**

**Тема роботи:** Бінарні файли

**Мета роботи:** Вивчити особливості створення й обробки бінарних файлів даних.



**Вирішення задачі С++:**

**main:**

#include<iostream>  
#include<fstream>  
#include <ctime>  
#include "header.cpp"  
using namespace std;  
  
  
int main() {  
 int work = 0;  
  
 do {  
 time\_t ttime = time(0);  
 tm \*local\_time = localtime(&ttime);  
 string today = "0" + to\_string(local\_time->tm\_mday) + ".0" + to\_string(1 + local\_time->tm\_mon) + "." +  
 to\_string(1900 + local\_time->tm\_year);// today's date as string  
 Pack Goods;  
 int goods\_amount = 200;  
 string names[goods\_amount], date\_mades[goods\_amount], use\_till\_dates[goods\_amount];  
 int prices[goods\_amount];  
 //get input from user  
 get\_info(goods\_amount, names, date\_mades, use\_till\_dates, prices);  
 //create and fill first file with product list  
 make\_first\_file(Goods, goods\_amount, names, date\_mades, use\_till\_dates, prices, "product\_list.dat");  
 //read the file  
 read\_file("product\_list.dat", goods\_amount);  
 //make new file according to the task  
 find\_data(goods\_amount, "end\_of\_expiration.dat", names, date\_mades, use\_till\_dates, prices, today);  
 cout << endl;  
 // print products that were made 10 days or less ago  
 in\_the\_last\_ten(goods\_amount, names, date\_mades, use\_till\_dates, prices, today);  
 cout << "Do you want to do the program again?(print 0 if yes)"<< endl;  
 //to re-run the program  
 cin >> work;  
 cin.ignore();  
 }while(work == 0);  
 return 0;  
}

}

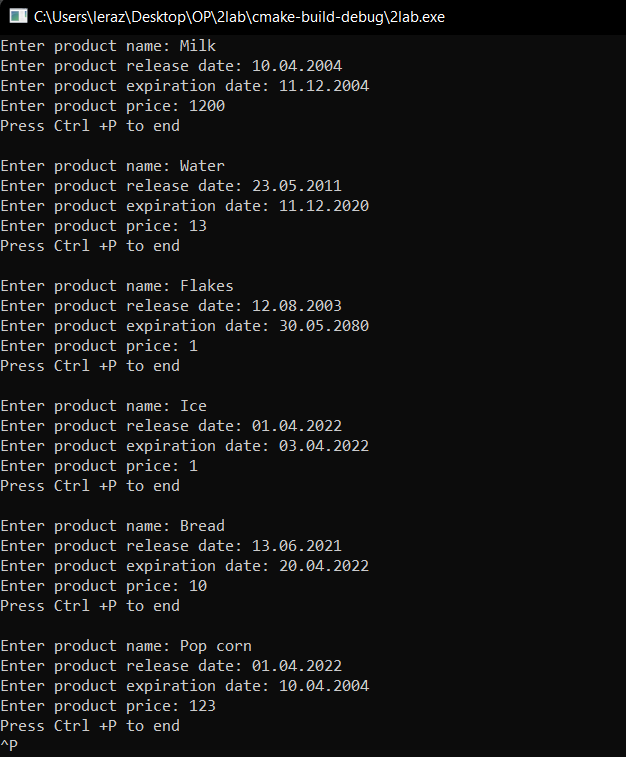
**header.h:**

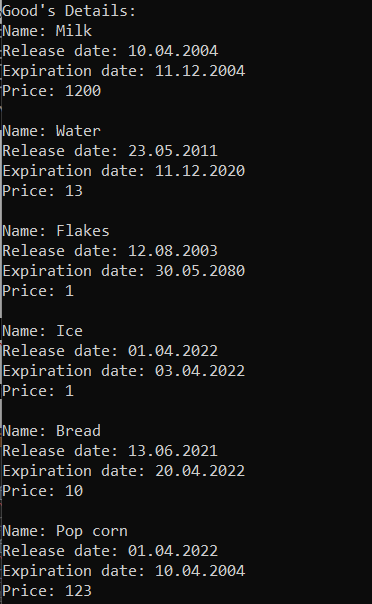
//  
// Created by leraz on 26-03-22.  
//  
  
#ifndef INC\_2LAB\_HEADER\_H  
#define INC\_2LAB\_HEADER\_H  
#include <iostream>  
#include <fstream>  
#include <filesystem>  
#include <cstring>  
  
using namespace std;  
typedef struct Goods {  
 string name;  
 string date\_made;  
 string use\_till\_date;  
 int price;  
}Pack;  
  
void get\_info(int &goods\_amount, string \*prod\_names,string \*prod\_date\_mades,string \*prod\_use\_till\_dates, int \*prices);  
int make\_first\_file(struct Goods, int goods\_amount, string \*prod\_names,string \*prod\_date\_mades,string \*prod\_use\_till\_dates, int \*prices, string first\_file);  
int date\_difference(string date\_one, string date\_two);  
int read\_file(string filename, int goods\_amount);  
void find\_data(int goods\_amount,string second\_file, string \*prod\_names,string \*prod\_date\_mades,string \*prod\_use\_till\_dates, int \*prices, string today);  
void in\_the\_last\_ten(int goods\_amount, string \*prod\_names,string \*prod\_date\_mades,string \*prod\_use\_till\_dates, int \*prices, string today);  
#endif //INC\_2LAB\_HEADER\_H

**header.cpp:**

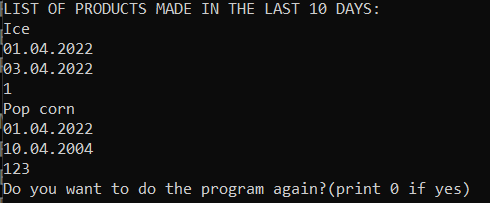
#include "header.h"  
  
  
void get\_info(int &goods\_amount, string \*prod\_names,string \*prod\_date\_mades,string \*prod\_use\_till\_dates, int \*prices)  
{  
 string to\_end;  
  
 int i = 0;  
 do {  
 string prod\_name;  
 cout << "Enter product name: ";  
 getline(cin, prod\_name);  
 prod\_names[i] = prod\_name;  
 string prod\_date\_made;  
 do {  
 cout << "Enter product release date: ";  
 getline(cin, prod\_date\_made);  
 } while (prod\_date\_made.length() != 10);  
 prod\_date\_mades[i] = prod\_date\_made;  
 string prod\_use\_till\_date;  
 do {  
 cout << "Enter product expiration date: ";  
 getline(cin, prod\_use\_till\_date);  
 } while (prod\_use\_till\_date.length() != 10);  
 prod\_use\_till\_dates[i] = prod\_use\_till\_date;  
 int prod\_price;  
 cout << "Enter product price: ";  
 cin >> prod\_price;  
 prices[i] = prod\_price;  
 cout <<"Press Ctrl +P to end"<< endl;  
 cin.ignore();  
 getline(cin, to\_end);  
 i++;  
 }while(to\_end.find(16)==string::npos);  
 goods\_amount = i;  
}  
int make\_first\_file(struct Goods, int goods\_amount, string \*prod\_names,string \*prod\_date\_mades,string \*prod\_use\_till\_dates, int \*prices, string first\_file)  
{  
 ofstream wf(first\_file, ios::out | ios::binary | ios::app);  
 if(!wf) {  
 cout << "Cannot open file!" << endl;  
 return 1;  
 }  
 Goods prod[goods\_amount];  
 for(int i =0; i < goods\_amount; i++)  
 {  
 prod[i].name = prod\_names[i];  
 prod[i].date\_made = prod\_date\_mades[i];  
 prod[i].use\_till\_date = prod\_use\_till\_dates[i];  
 prod[i].price = prices[i];  
  
 }  
  
 for(int i = 0; i < goods\_amount; i++)  
 wf.write((char \*) &prod[i], sizeof(Goods));  
 wf.close();  
 if(!wf.good()) {  
 cout << "Error occurred at writing time!" << endl;  
 return 1;  
 }  
  
}  
  
int read\_file(string filename, int goods\_amount)  
{  
 ifstream rf(filename, ios::in | ios::binary);  
 if(!rf) {  
 cout << "Cannot open file!" << endl;  
 return 1;  
 }  
 Goods rprod[goods\_amount];  
 for(int i = 0; i < goods\_amount; i++)  
 rf.read((char \*) &rprod[i], sizeof(rprod[i]));  
 rf.close();  
 if(!rf.good()) {  
 cout << "Error occurred at reading time!" << endl;  
 return 1;  
 }  
 cout<<"Good's Details:"<<endl;  
 for(int i=0; i < goods\_amount; i++) {  
 cout << "Name: " << rprod[i].name << endl;  
 cout << "Release date: " <<rprod[i].date\_made << endl;  
 cout << "Expiration date: " << rprod[i].use\_till\_date << endl;  
 cout << "Price: " << rprod[i].price << endl;  
 cout << endl;  
 }  
}  
  
void find\_data(int goods\_amount,string second\_file, string \*prod\_names,string \*prod\_date\_mades,string \*prod\_use\_till\_dates, int \*prices, string today)  
{  
 Goods prod[goods\_amount];  
 ofstream file\_b2(second\_file, ios::binary);  
 int j =0;  
 for(int i =0; i< goods\_amount; i++)  
 {  
 string release = prod\_date\_mades[i];  
 string expir = prod\_use\_till\_dates[i];  
 int can\_use = date\_difference(release,expir) / 10;  
 int till\_today = abs(date\_difference(expir, today));  
 if (can\_use > till\_today) {  
 prod[j].name = prod\_names[i];  
 prod[j].date\_made = prod\_date\_mades[i];  
 prod[j].use\_till\_date = prod\_use\_till\_dates[i];  
 prod[j].price = prices[i];  
 j++;  
 }  
 }  
 for(int i = 0; i < j; i++)  
 file\_b2.write((char \*) &prod[i], sizeof(Goods));  
 file\_b2.close();  
 Goods pr;  
 ifstream file\_b(second\_file, ios::binary);  
 cout << "NEW FILE:"<< endl;  
 while(file\_b.read((char\*)&pr, sizeof (Goods)))  
 {  
 cout << pr.name << " - " << pr.date\_made << " - "<<pr.use\_till\_date << " - "<<pr.price << endl;  
 }  
}  
  
  
int date\_difference(string date\_one, string date\_two)  
{  
 int day\_one = stoi(date\_one.substr(0,2));  
 int month\_one = stoi(date\_one.substr(3,2));  
 int year\_one = stoi(date\_one.substr(6,4));  
 int day\_two = stoi(date\_two.substr(0,2));  
 int month\_two = stoi(date\_two.substr(3,2));  
 int year\_two = stoi(date\_two.substr(6,4));  
  
 //calc  
 int year\_dif;  
 int month\_dif;  
 int day\_dif;  
 year\_dif=year\_two-year\_one;  
 month\_dif=month\_two-month\_one;  
 day\_dif=day\_two-day\_one;  
  
 //case if dd is negative i.e. current dd is less than first date  
 if(day\_dif<0)  
 {  
 month\_dif--;  
 day\_dif+=30;  
 }  
  
 //case if mm is negative i.e. current mm is less than first date  
 if (month\_dif<0)  
 {  
 year\_dif--;  
 month\_dif+=12;  
 }  
 int result;  
 result = year\_dif \* 365 + month\_dif \* 30 + day\_dif;  
 return result;  
}  
  
void in\_the\_last\_ten(int goods\_amount, string \*prod\_names,string \*prod\_date\_mades,string \*prod\_use\_till\_dates, int \*prices, string today) {  
 Goods prod;  
 cout <<"LIST OF PRODUCTS MADE IN THE LAST 10 DAYS:" << endl;  
 for (int i = 0; i < goods\_amount; i++) {  
 string release = prod\_date\_mades[i];  
 int made = date\_difference(release, today);  
 if (made <= 10) {  
 cout << prod\_names[i] << endl;  
 cout << prod\_date\_mades[i] << endl;  
 cout << prod\_use\_till\_dates[i] <<endl;  
 cout << prices[i] << endl;  
 }  
 }  
}

**Результат:**









**Python:**

**main:**

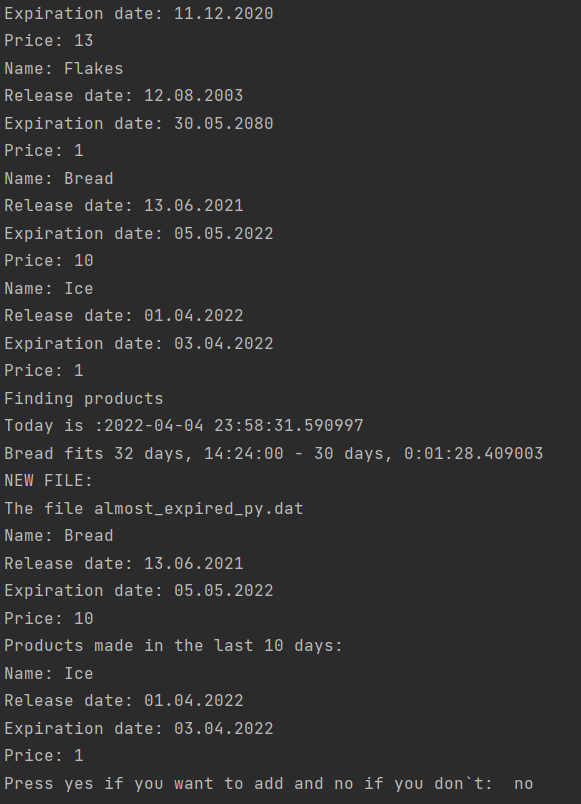
import funcs  
  
funcs.make\_product\_list("product\_list\_py.dat") # make first file with product list  
funcs.read\_file("product\_list\_py.dat") # to read first file  
print("Finding products")  
# find products to the rule (10% of the time it can be used)  
funcs.transfer\_data\_expiration("product\_list\_py.dat", "almost\_expired\_py.dat")  
print("NEW FILE:")  
# making new file with these products  
funcs.read\_file("almost\_expired\_py.dat")  
# products that have less than 10 days  
funcs.info\_ten\_days("product\_list\_py.dat")  
funcs.dopys() # do you want to re-run the file

**funcs:**

import pickle as pk  
from datetime import date, datetime, timedelta  
  
  
def make\_product\_list(filename):  
 goods\_amount = int(input("How much products do you want to add?"))  
 product = []  
 product\_info = []  
 for i in range(goods\_amount):  
 name = input("Name: ")  
 release\_date = input("Release date: ")  
 expirat\_date = input("Expiration date: ")  
 price = input("Price: ")  
 product = [name, release\_date, expirat\_date, price]  
 product\_info.append(product)  
 file = open(filename, 'wb')  
 pk.dump(product\_info, file)  
 file.close()  
  
  
def read\_file(filename):  
 print("The file " + filename)  
 f = open(filename, 'rb')  
 products\_data = pk.load(f)  
 f.close  
 for product in products\_data:  
 print("Name: " + str(product[0]))  
 print("Release date: " + str(product[1]))  
 print("Expiration date: " + str(product[2]))  
 print("Price: " + str(product[3]))  
  
  
def transfer\_data\_expiration(from\_file, to\_file):  
 d1 = datetime.today()  
 print("Today is :" + str(d1))  
 f = open(from\_file, 'rb')  
 products\_data = pk.load(f)  
 f.close  
 suitable\_prod = []  
 suitable\_prod\_info = []  
 for product in products\_data:  
 release = product[1]  
 expire = product[2]  
 dt\_start = datetime.strptime(release, '%d.%m.%Y')  
 dt\_end = datetime.strptime(expire, '%d.%m.%Y')  
 allowed\_time = dt\_end - dt\_start  
 if d1 > dt\_end:  
 substr = d1 - dt\_end  
 else:  
 substr = dt\_end - d1  
 if allowed\_time / 10 > substr:  
 print(product[0] + " fits " + str(allowed\_time / 10) + " - " + str(substr))  
 suitable\_prod = [product[0], product[1], product[2], product[3]]  
 suitable\_prod\_info.append(suitable\_prod)  
 file = open(to\_file, 'wb')  
 pk.dump(suitable\_prod\_info, file)  
 file.close()  
  
  
def info\_ten\_days(filename):  
 print("Products made in the last 10 days: ")  
 f = open(filename, 'rb')  
 products\_data = pk.load(f)  
 f.close  
 d1 = datetime.today()  
 for product in products\_data:  
 release = product[1]  
 dt\_start = datetime.strptime(release, '%d.%m.%Y')  
 days\_made = d1 - dt\_start  
 if days\_made <= timedelta(days=10):  
 print("Name: " + str(product[0]))  
 print("Release date: " + str(product[1]))  
 print("Expiration date: " + str(product[2]))  
 print("Price: " + str(product[3]))  
  
def dopys(): #to re-run the program  
 while True:  
 answ = input("Press yes if you want to add and no if you don`t: ")  
 if answ == "yes":  
 make\_product\_list("product\_list\_py.dat")  
 read\_file("product\_list\_py.dat")  
 print("Finding products")  
 transfer\_data\_expiration("product\_list\_py.dat", "almost\_expired\_py.dat")  
 print("NEW FILE:")  
 read\_file("almost\_expired\_py.dat")  
 info\_ten\_days("product\_list\_py.dat")  
 if answ != "yes":  
 break

**Результат:**





**Висновок**: У цій лабораторній роботі було вивчено особливості створення і обробки бінарних файлів. Була постановлена задача, в якій визначався список продуктів, введений користувачем. У алгоритмі були уточнені обмеження з використанням умов при яких задача буде виконана правильно. За допомоги цієї лабораторної роботи можливо складання списку продуктів, перевірка у яких продуктів завершується термін придатності та які були виготовлені менш ніж 10 днів тому.