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 №5 з дисципліни
«Основи програмування – 2. Методології програмування»

«Успадкування та поліморфізм»

Варіант 22

Виконав студент ІП-13, Музичук Віталій Андрійович
(шифр, прізвище, ім'я, по батькові)

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

Київ 2022

Лабораторна робота 4

Перевантаження операторів

Мета – вивчити механізми створення класів з використанням класів та об’єктів.

Варіант 22

Завдання:

Створити клас “Товар”, який містить назву, дату виготовлення, ціну, кількість одиниць, а також метод обчислення сумарної вартості товару. На його основі створити класи-нащадки “Промисловий товар”, що додатково зберігає умови транспортування, місце знаходження товару (на складі, в торговому залі) та “Харчовий продукт”, який додатково містить термін зберігання продукту. Створити n номенклатур промислових товарів та m номенклатур харчових продуктів. Визначити загальну вартість харчових продуктів термін зберігання яких закінчився, і загальну вартість промислових товарів, які знаходяться на складі.

1. Виконання завдання на мові C++:

// Lab_5.cpp

```
#include "Goods.h"

int main()
{
	cout << "How many industrial products do you want create: ";
	int numInd; cin >> numInd;
	IndustrialGoods* indGoods = createInd(numInd);

	cout << "How many eating products do you want create: ";
	int numEat; cin >> numEat;
	EatingGoods* eatGoods = createEat(numEat);

	struct tm current_time;
	time_t t = time(0);
	localtime_s(&current_time, &t);
	TDate currentTime(current_time.tm_mday, current_time.tm_mon + 1,
1900 + current_time.tm_year);

	int full_price = 0;
```

```
	for (int i = 0; i < numEat; i++) {
		if (eatGoods[i].isSpoiledGood(currentTime)) {
			full_price += eatGoods[i].get_sum();
		}
	}
	cout << "The total price of all spoiled products is: " <<
full_price << endl;

	full_price = 0;
	for (int i = 0; i < numInd; i++) {
		if (indGoods[i].get_conditions() == IN_STOCK) {
			full_price += indGoods[i].get_sum();
		}
	}

	cout << "The total price of all products that store IN_STOCK: " <<
full_price << endl;

	return 0;
}
```

//Goods.cpp

```
#include "Goods.h"


// Реалізація класу для збереження дати


int TDate::monthDate[12] = { 31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30,
31 };


TDate::TDate(int day1 = 1, int month1 = 1, int year1 = 1901) {
	if (year1 < 1900 || year1 > 2022) {
		cerr << "Incorrect enter year";
		exit(0);
	}
	if (month1 <= 0 || month1 > 12) {
		cerr << "Invalid enter month";
		exit(0);
	}
	if (day1 <= 0 || day1 > monthDate[month1 - 1]) {
		cerr << "Invalid entered day";
		exit(0);
	}
	this->day = day1;
	this->month = month1;
	this->year = year1;
}


void TDate::addDay(int day) {
	if (day < 0) {
		cerr << "Incorrected day";
		exit(0);
```

```
	}
	this->day += day;
	while (true) {
		if (this->day > monthDate[this->month - 1]) {
			this->day -= monthDate[this->month - 1];
			this->month += 1;
		}
		else break;
	}
}

// Реалізація базового класу

Goods::Goods() {
	name = "noName";
	date;
	price = 0.0;
	number = 0;
}

Goods::Goods(string name1, TDate date1, float price1, int number1) :
name(name1), date(date1) {
	if (price1 < 0) {
		cerr << "Incorrect entered price";
		exit(0);
	}
	if (number1 <= 0) {
		cerr << "Invalid number of products";
		exit(0);
	}
```

```
	this->price = price1;
	this->number = number1;
}

float Goods::get_sum() {
	return this->price * this->number;
}

// Реалізація похідного класу для збереження промислових товарів

IndustrialGoods::IndustrialGoods(): Goods(), conditions(IN_STOCK) {}

IndustrialGoods::IndustrialGoods(string name1, TDate date1, float
price1, int number1, Transport conditions1) :
	Goods(name1, date1, price1, number1), conditions(conditions1) {}

Transport IndustrialGoods::get_conditions() {
	return this->conditions;
}

// Реалізація похідного класу для збереження харчових товарів

EatingGoods::EatingGoods() : Goods(), expiration(0) {}

EatingGoods::EatingGoods(string name1, TDate date1, float price1, int
number1, int expiration1) :
	Goods(name1, date1, price1, number1){
	if (expiration1 < 0) {
		cerr << "Incorrected expiration date";
		exit(0);
```

```
	}
	this->expiration = expiration1;
}

bool EatingGoods::isSpoiledGood(TDate& currentDate) {
	TDate expirationDate = this->date;
	expirationDate.addDay(this->expiration);

	int cuurentDays = currentDate.year * 365 + (currentDate.month - 1)
* 31 + currentDate.day;
	int expirationDays = expirationDate.year * 365 +
(expirationDate.month - 1) * 31 + expirationDate.day;

	if (cuurentDays > expirationDays) {
		return true;
	}
	else if (expirationDays >= cuurentDays) {
		return false;
	}
}

// Допоміжні функції

IndustrialGoods* createInd(int numInd) {
	IndustrialGoods* indGoods = new IndustrialGoods[numInd];
	cout << "Add information about industrial product" << endl;
	string name, dateLine; float price; int num, cond; int dates[3];
	for (int i = 0; i < numInd; i++) {
		cout << "Name product: ";
		cin.ignore(3200, '\n');
```

```
            getline(cin, name);

            cout << "Enter date in format[dd.mm.yyyy]: ";
            getline(cin, dateLine);
            for (int j = 0; j < 3; j++)
            {
                  if (j == 2)
                        dates[j] = stoi(dateLine);
                  else {
                        int pos = dateLine.find('.');
                        dates[j] = stoi(dateLine.substr(0, pos));
                        dateLine.erase(0, pos + 1);
                  }
            }

            cout << "Enter price: ";
            cin >> price;

            cout << "Enter number of products: ";
            cin >> num;

            cout << "Enter a conditions of transoprt(0 - IN_STOCK, 1 -
IN_TRADING_HALL): ";
            cin >> cond;

            indGoods[i] = IndustrialGoods(name, TDate(dates[0], dates[1],
dates[2]), price, num, static_cast<Transport>(cond));
            cout << endl;
      }
      return indGoods;
```

```
}


EatingGoods* createEat(int numEat) {
	EatingGoods* eatGoods = new EatingGoods[numEat];
	cout << "Add information about eating product" << endl;
	string name, dateLine; float price; int num, expiriation; int dates[3];
	for (int i = 0; i < numEat; i++) {
		cout << "Name product: ";
		cin.ignore(3200, '\n');
		getline(cin, name);

		cout << "Enter date in format[dd.mm.yyyy]: ";
		getline(cin, dateLine);
		for (int j = 0; j < 3; j++)
		{
			if (j == 2)
				dates[j] = stoi(dateLine);
			else {
				int pos = dateLine.find('.');
				dates[j] = stoi(dateLine.substr(0, pos));
				dateLine.erase(0, pos + 1);
			}
		}

		cout << "Enter price: ";
		cin >> price;

		cout << "Enter number of products: ";
		cin >> num;
```

```
            cout << "Enter a expiration date: ";
            cin >> expiriation;

            eatGoods[i] = EatingGoods(name, TDate(dates[0], dates[1],
dates[2]), price, num, expiriation);
            cout << endl;
      }
      return eatGoods;
}
```

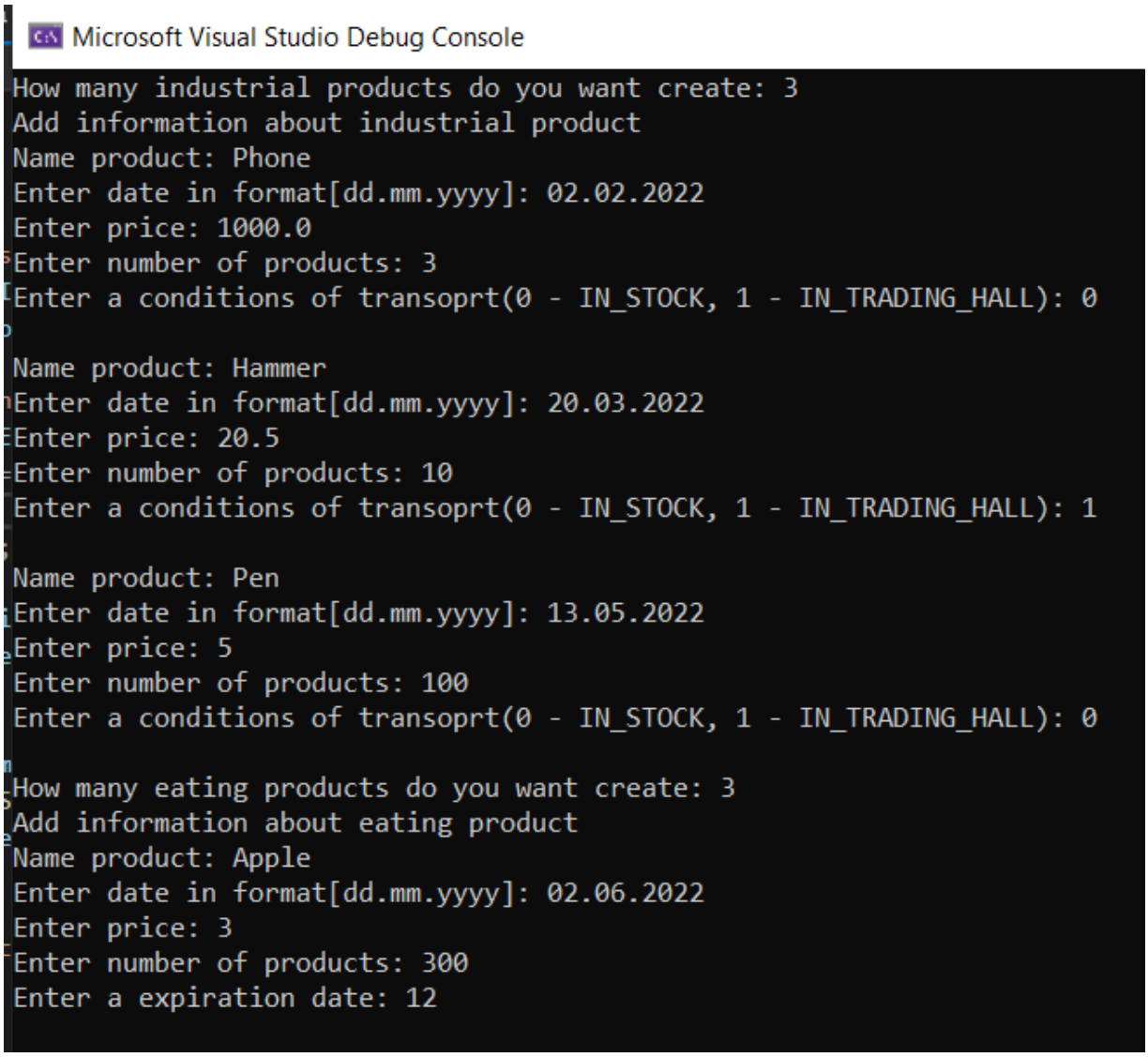
// vector.h

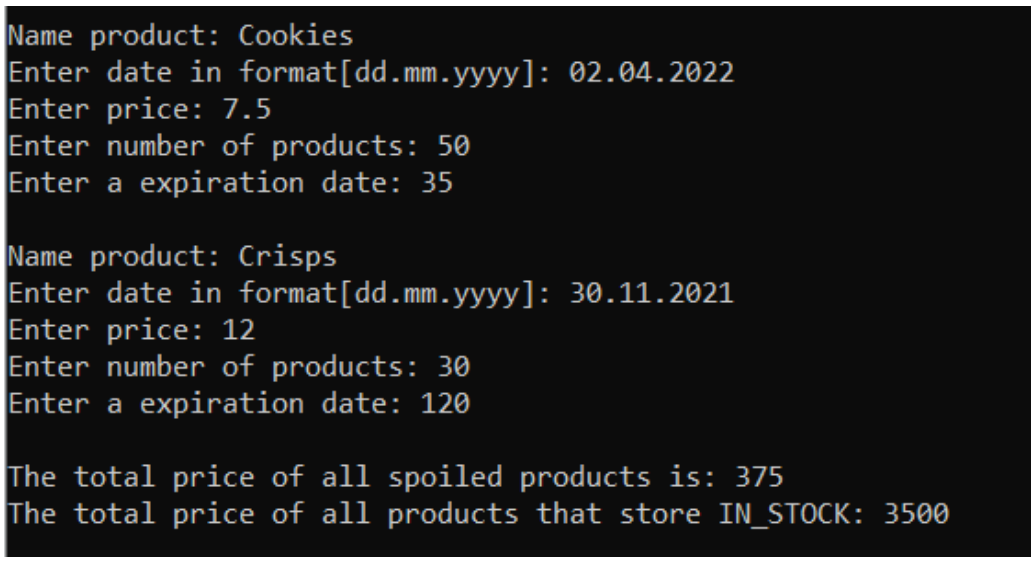
```
#pragma once

#include <iostream>
#include <string>
using namespace std;

class TDate {
public:
      int day;
      int month;
      int year;
      static int monthDate[12];
      TDate(int day1, int month1, int year1);
      void addDay(int);
};

enum Transport {
```

```
	IN_STOCK,
	IN_TRADING_HALL
};

class Goods {
protected:
	string name;
	TDate date;
	float price;
	int number;
	Goods();
	Goods(string, TDate, float, int);
public:
	float get_sum();
};

class IndustrialGoods : public Goods {
	Transport conditions;
public:
	IndustrialGoods();
	IndustrialGoods(string, TDate, float, int, Transport);
	Transport get_conditions();
};

class EatingGoods : public Goods {
	int expiration;
public:
	EatingGoods();
	EatingGoods(string, TDate, float, int, int);
```

```
      bool isSpoiledGood(TDate&);
};

IndustrialGoods* createInd(int numInd);
EatingGoods* createEat(int numEat);
```

Тестування програми:





2. Виконання завдання на мові Python:

// Lab_5.py

```
from lib import *

num_ind_goods = int(input("How many industrial products do you want create: "))
ind_goods = create_ind_goods(num_ind_goods)

num_eating_goods = int(input("How many eating products do you want create: "))
eating_goods = create_eating_goods(num_eating_goods)

time = datetime.now()
current_time = TDate(time.day, time.month, time.year)

full_price = 0
for i in range(num_eating_goods):
    if eating_goods[i].is_spoiled_good(current_time):
        full_price += eating_goods[i].get_sum()

print("The total price of all spoiled products is:", full_price)

full_price = 0
for i in range(num_ind_goods):
    if ind_goods[i].get_conditions() == "IN_STOCK":
        full_price += ind_goods[i].get_sum()

print("The total price of all products that store IN_STOCK:", full_price)
```

// lib.py

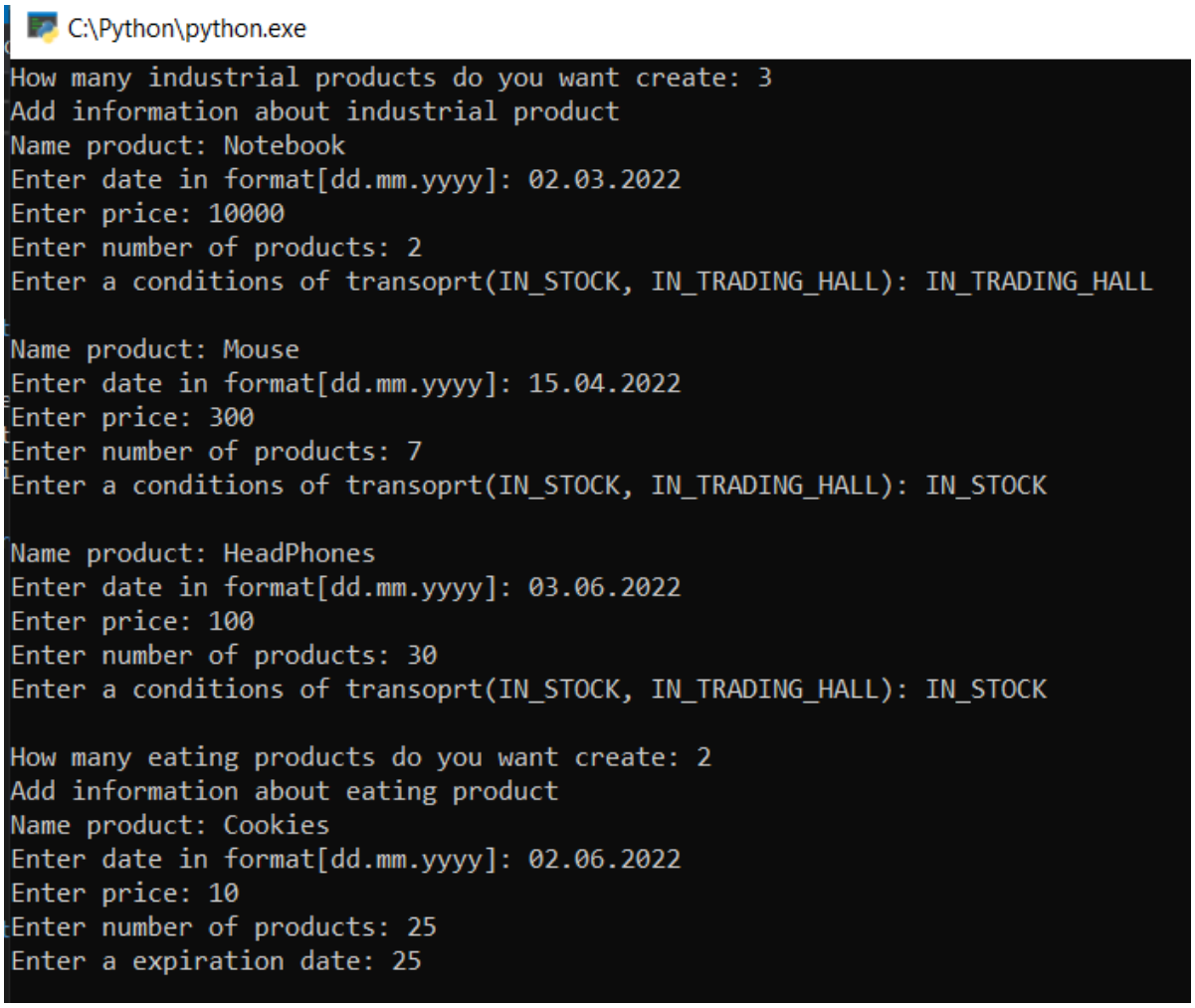
```
from datetime import datetime
MONTH_DATE = [31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31]

class TDate:
    def __init__(self, day, month, year):
        if year < 1900 or year > 2022:
            print("Incorrect enter year")
            exit(0)
        if month <= 0 or month > 12:
            print("Invalid enter month")
            exit(0)
        if day <= 0 or day > MONTH_DATE[month - 1]:
            print("Invalid entered day")
            exit(0)
        self.day = day
        self.month = month
        self.year = year

    def add_day(self, day):
        self.day += day
        while True:
            if self.day > MONTH_DATE[self.month - 1]:
                self.day -= MONTH_DATE[self.month - 1]
                self.month += 1
            else:
                break

class Goods:
    def __init__(self, name, date, price, number):
        self._name = name
        self._date = date
```

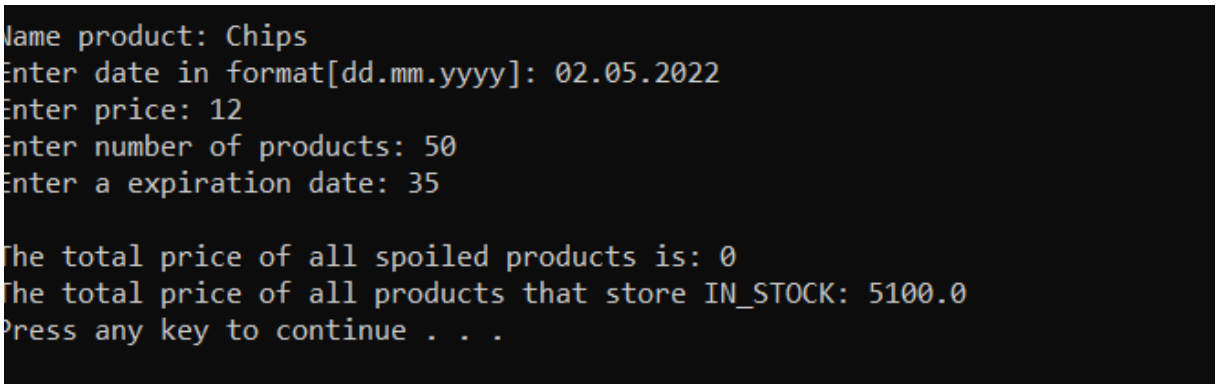
```
        if price < 0:
            print("Incorrect entered price")
            exit(0)
        if number <= 0:
            print("Invalid number of products")
            exit(0)
        self._price = price
        self._number = number

    def get_sum(self):
        return self._price * self._number


class Industrial_Goods(Goods):
    def __init__(self, name, date, price, number, conditions):
        super().__init__(name, date, price, number)
        self.__conditions = conditions

    def get_conditions(self):
        return self.__conditions

class Eating_Goods(Goods):
    def __init__(self, name, date, price, number, expiration):
        super().__init__(name, date, price, number)
        if expiration < 0:
            print("Incorrected expiration date")
            exit(0)
        self.__expiration = expiration
    def is_spoiled_good(self, current_date):
        expiration_date = self._date
        expiration_date.add_day(self.__expiration)
```

```
          current_days = current_date.year * 365 + (current_date.month -
1) * 31 + current_date.day
                    expiration_days  =  expiration_date.year  *  365  +
(expiration_date.month - 1) * 31 + expiration_date.day

        if current_days > expiration_days:
            return True
        elif expiration_days >= current_days:
            return False

def create_ind_goods(num_ind_goods):
    ind_goods = []
    print("Add information about industrial product")

    for i in range(num_ind_goods):
        name = input("Name product: ")
        date = input("Enter date in format[dd.mm.yyyy]: ")
        date = date.split('.')
        for i in range(3):
            date[i] = int(date[i])
        price = float(input("Enter price: "))
        number = int(input("Enter number of products: "))
          conditions = input("Enter a conditions of transoprt(IN_STOCK,
IN_TRADING_HALL): ")
         ind_goods.append(Industrial_Goods(name, TDate(date[0], date[1],
date[2]), price, number, conditions))
        print()
    return ind_goods

def create_eating_goods(num_eating_goods):
    eating_goods = []
    print("Add information about eating product")
```

```
    for i in range(num_eating_goods):
        name = input("Name product: ")
        date = input("Enter date in format[dd.mm.yyyy]: ")
        date = date.split('.')
        for i in range(3):
            date[i] = int(date[i])
        price = float(input("Enter price: "))
        number = int(input("Enter number of products: "))
        expiriation = int(input("Enter a expiration date: "))
          eating_goods.append(Eating_Goods(name, TDate(date[0], date[1],
date[2]), price, number, expiriation))
        print()
    return eating_goods
```

Тестування програми:





Висновок: Під час виконання лабораторної роботи я вивчив особливості створення класів з використанням перевантажених операторів (операцій) на прикладі мови C++. Результатом виконання лабораторної роботи є програма, основним завданням якої є створення класу вектору та ініціалізація трьох його об’єктів. Після тестування програм можна зробити висновок, що вони справляються із поставленою задачею.