**Міністерство освіти і науки, молоді та спорту України**

**Національний університет „Львівська політехніка”**

Кафедра ЕОМ



**Курсова робота**

з предмету: «Програмування, частина 2 (Об’єктно-орієнтоване програмування)

на тему:

**«Базові принципи об’єктно-орієнтованого програмування»**

Індивідуальне завдання : «Митні склади»

**Виконав:**

ст. гр. КІ-15

Стецик О.І.

**Прийняв:**

Козак Н.Б.

Львів – 2020

# **Анотація**

Курсова робота на тему: «Базові принципи об’єктно-орієнтованого програмування».

Виконав – студент групи КІ-15 Стецик О.І.

Перевірив – асистент Козак Н.Б.

Курсова робота складається з опису виконання поставленої задачі та графічної частини. В графічній частині наведено алгоритм виконання завдання, оцінка його складності.

Опис виконання завдання включає розділи: огляд та обґрунтування вибору технологій об’єктно-орієнтованого програмування, аналіз та розробка алгоритму, програмна реалізація алгоритму, відлагодження та тестування розробленої програми, аналіз продуктивності створеної програми, висновок, лістинг програми .

В даній курсовій роботі була виконана розробка програми на мові програмування С++ з використанням об’єктно-орієнтованого програмування, яка включала в себе роботу з файлами, класами, обробку числової інформації і роботу з векторами та об’єктами.

**Зміст**

[**Анотація** 2](#_Toc515438689)

[**Завдання на курсову роботу** 4](#_Toc515438690)

[**1. Огляд та обґрунтовування вибору технологій об'єктно-орієнтованого програмування** 5](#_Toc515438691)

[**2. Аналіз та розробка алгоритму згідно індивідуального завдання** 6](#_Toc515438692)

[2.1. Опис алгоритму та оцінка його складності 6](#_Toc515438693)

[2.2. Аналіз особливостей алгоритму 8](#_Toc515438694)

[2.3. Опис алгоритму у псевдокоді 9](#_Toc515438695)

[2.4. Розробка граф-схеми алгоритму 11](#_Toc515438696)

[**3. Програмна реалізація алгоритму згідно індивідуального завдання** 13](#_Toc515438697)

[3.1. Опис послідовної програмної реалізації алгоритму 13](#_Toc515438698)

[3.1.1. Опис структур даних програми 13](#_Toc515438699)

[3.1.2. Діаграма класів 13](#_Toc515438700)

[3.1.3. Опис реалізації основних функцій/методів 15](#_Toc515438701)

[3.2. Опис програмної реалізації алгоритму 21](#_Toc515438702)

[**4. Відлагодження та тестування реалізованої програми згідно індивідуального завдання** 16](#_Toc515438703)

[**Висновок** 19](#_Toc515438704)

[**Список використаної літератури** 20](#_Toc515438705)

# **Завдання на курсову роботу**

Тема курсової роботи – «Митні склади».

Вхідні дані не були взяті з інтернету, а зроблені власноруч , беручи до уваги вимоги до курсової роботи.

Програма містить головне меню, звідки відкривається доступ до трьох похідних меню: увійти в акаунт, реєстрація та вихід.

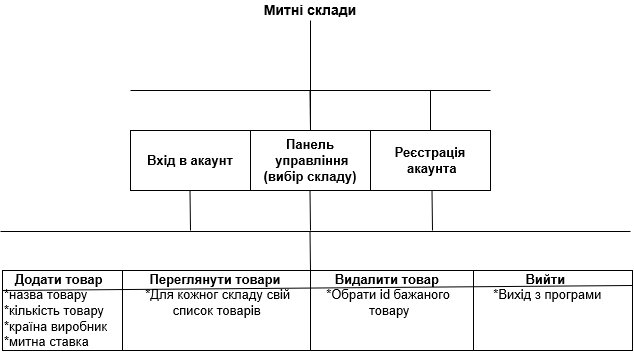


Рис. 1.1. Вигляд предметної галузі

# **1. Огляд та обґрунтовування вибору технологій об'єктно-орієнтованого програмування**

Об'єктно-орієнтоване програмування (ООП) — одна з парадигм програмування, яка розглядає програму як множину «об'єктів», що взаємодіють між собою. Основу ООП складають три основні концепції: інкапсуляція, успадкування та поліморфізм. Одною з переваг ООП є краща модульність програмного забезпечення. Успадкування (наслідування). Клас може мати «підкласи», спеціалізовані, розширені версії надкласу. Можуть навіть утворюватися цілі дерева успадкування. Підкласи успадковують атрибути та поведінку своїх батьківських класів, і можуть вводити свої власні. Успадкування може бути одиничне (один безпосередній батьківський клас) та множинне (кілька батьківських класів). Інкапсуляція. Приховування деталей про роботу класів від об'єктів, що їх використовують або надсилають їм повідомлення.

Інкапсуляція досягається шляхом вказування, які класи можуть звертатися до членів об'єкта. Як наслідок, кожен об'єкт представляє кожному іншому класу певний інтерфейс — члени, доступні іншим класам. Інкапсуляція потрібна для того, аби запобігти використанню користувачами інтерфейсу тих частин реалізації, які, швидше за все, будуть змінюватись. Це дозволить полегшити внесення змін, тобто, без потреби змінювати і користувачів інтерфейсу. Часто, члени класу позначаються як публічні (англ. public), захищені (англ. protected) та приватні (англ.private), визначаючи, чи доступні вони всім класам, підкласам, або лише до класу в якому їх визначено.

Поліморфізм. Поліморфізм означає залежність поведінки від класу, в якому ця поведінка викликається, тобто, два або більше класів можуть реагувати по-різному на однакові повідомлення. На практиці - це реалізовується шляхом реалізації ряду підпрограм (функцій, процедур, методів тощо) з однаковими іменами, але з різними параметрами. В залежності від того, що передається і вибирається відповідна підпрограма.

# **2. Аналіз та розробка алгоритму згідно індивідуального завдання**

# **2.1. Опис алгоритму та оцінка його складності**

Після запуску програми в консолі з’являється головне меню, за допомогою якого ми можемо перейти в 3 інші підменю та вибрати дію, що хочемо зробити:

1. Зареєструватися:

1) Ввести ім’я користувача

2) Ввести пароль

1. Увійти в акаунт:

1) Ввести ім’я користувача

2) Ввести пароль

->Після чого перейти до панелі керування, де ми можемо

1) Обрати один з існуючих складів

2) Додати товари

3) Переглянути існуючі товар

4) Вийти

3. Вихід

Якщо обрати операцію «Зареєструватися», то у консолі з’являється операція «Введіть ім’я користувача», а після операція «Введіть пароль». Введені дання зберігаються в середині програми й надалі.

Якщо обрати операцію «Увійти в акаунт», то у консолі з’явиться операція «Введіть ім’я користувача», а після операція «Введіть пароль». Дані для цієї операції беруться з попередньо зарезервованих в панелі ’’Зареєструватися’’

Якщо обрати операцію «Вихід», то користувач вийде з програми.

Після операції «Увійти в акаунт», з’являється панель «Виберіть митний склад».

Після операції «Виберіть митний склад», з’являється панель керування.

Якщо обрати операцію «Додати товар», то у консолі з’явиться можливість додати товар в список.

Якщо обрати операцію «Переглянути товари», то у консолі з’явиться список створених попередньо товарів з відповідними до них характеристиками.

Якщо обрати операцію «Видалити товар», то у консолі з’явиться можливість видалити товар з списку існуючих.

Якщо обрати операцію «Вихід», то користувач вийде з програми.

Всі дані, що вводяться у консолі, записуються в файл. Далі вони зчитуються з файлу та обробляються, в залежності від того, яку дію обере користувач.

# **2.2. Аналіз особливостей алгоритму**

Розглянемо гілку класів. Вона починається з класу CUI (Console user interface), DatabaseStorage (зберігання данних), DatabaseUsers (зберігання користувачів), global, та users .

Клас CUI містить наступні властивості: зареєструвати користувача (потрібно ввести своє ім’я та пароль, за якими можна буде увійти в акаунт), увійти в акаунт (тобто ввести своє ім’я і пароль, які ви попередньо зареєстрували) та вихід (здійснюється вихід з програми)

Клас DataBaseStorage містить інформацію про товари, та містить наступні властивості: додати товар і вказати його характеристики (назву, вагу та об’єм). Переглянути товар (відкривається меню з існуючими товарами) та видалити товар (дозволяє видалити товар за його ID)

Клас DataBaseStorage містить інформацію про користувача та має наступні властивості: ім’я та пароль користувача. Крім цього, даний клас містить наступні методи: загрузити з файлу дані про користувачів, зберегти у файл дані про користувачів, авторизація та реєстрація.

Клас user відповідає за головне меню, перегляд товарів та вибір митниці у програмі.

# 

# **2.3. Опис алгоритму у псевдокоді**

Головне меню

1. Увійти в акаунт

2. Зареєструватися

3. Вийти з програми.

**Випадок** 1: {

Зареєструватися

1. Ввести ім’я
2. Ввести пароль

Повернутися до головного меню

Увійти в акаунт

1. Ввести ім’я
2. Ввести пароль

**Перемикач:**

Обрати відповідний склад з 3 існуючих

**Перемикач:**

1. Додати товар

Вести назву, кількість, країну виробника та митну ставку

Повернутися в меню користувача

1. Переглянути товар
2. Видалити товар

Ввівши відповідний ID товару

1. Вийти

Відповідно вихід з самої програми

**Випадок** 2:{

Зареєструватися

1. Ввести ім’я
2. Ввести пароль

Повернутися до головного меню

Увійти в акаунт

1. Ввести неправильне ім’я
2. Ввести неправильний пароль

Наекрані виведе неправильний логін або пароль

Вийти з програми

# 

# **2.4. Розробка граф-схеми алгоритму**

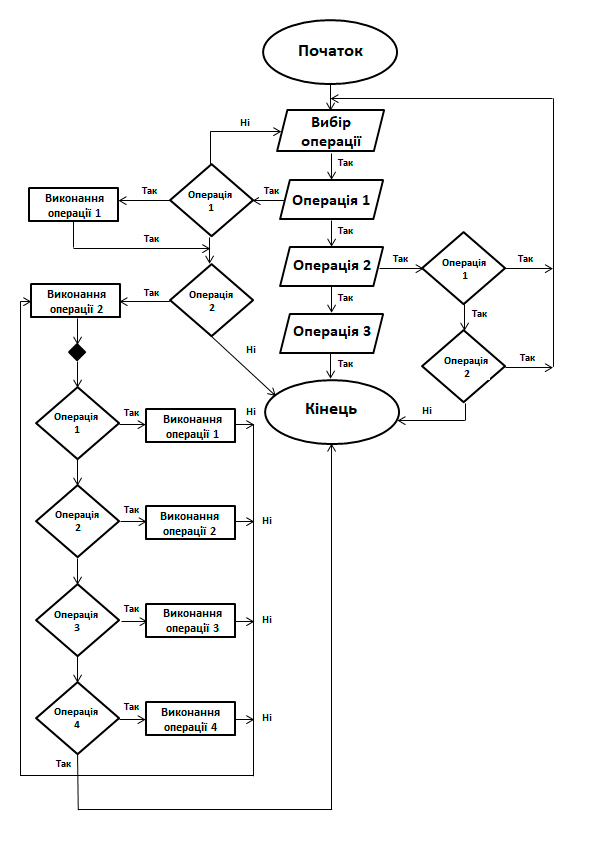


Рис. 2.4. Користувацький інтерфейс

Пояснення до блок-схеми:  
1. Вибір операції(головне меню):  
1) Операція 1 – Увійти в акаунт  
2) Операція 2 – Зареєструвати акаунт  
3) Операція 3 – Вихід

2. Виконання операцій:  
1) Операція 1: виконується Операція 1 – Ввести ім’я і пароль  
2) Операція 1: виконується Операція 2 – Обрати митний склад  
«Перехід в користувацьке меню»  
3) Операція 1: виконується Операція 1 - Додати товар  
4) Операція 1: виконується Операція 2 – Переглянути товари

«Перегляд списку товарів»  
5) Операція 1: виконується Операція 3 – Видалити товар  
6) Операція 1: виконується Операція 4 - Вихід  
«Виконується вихід з програми»  
7) Операція 2: виконується Операція 1 – Створити ім’я  
8) Операція 2: виконується Операція 4 – Створити пароль  
«Відбувається додавання нового користувача»  
9) Операція 3 – Виконується вихід з програми

# **3. Програмна реалізація алгоритму згідно індивідуального завдання**

# 

# **3.1. Опис послідовної програмної реалізації алгоритму**

# **3.1.1. Опис структур даних програми**

newUser – новий користувач;

curUser – поточний користувач;

username – ім’я користувача;

password – пароль;

bool status – стан користувача;

userGoods – товари користувача;

name – назва товару.

weight – вага товару;

volume – об’єм товару ;

goodsInfo – інформація про товар;

# **3.1.2. Діаграма класів**

Всі класи в програмі рівноправні і не наслідують один одного, але деякі класи використовують інші, що показано на рисунку.

CUI

CUI

DatabaseUsers

global

User

DatabaseStorage

Рис. 3.1.2. Ієрархія класів

Клас CUI (Console User Interface) містить такі властивості:

username – ім’я користувача;

password – пароль користувача;

curUser – поточний користувач;

newUser – новий користувач;

В класі DatabaseStorage зберігаються відомості про товари. Він містить такі властивості:

goods – товари;

file – файл;

name – назва товару;

weight – вага товару;

value – об’єм товару.

userGoods – товари користувача

В класі DatabaseUsers зберігаються відомості про користувача. Він має такі властивості:

string password – пароль;

username – ім’я користувача.

В класі user зберігаються відомості про користувача. Він має такі властивості:

curOffice – поточна митниця;

curUsername – ім’я поточного користувача;

user – користувач;

OfficeSelector – вибір митниці;

mainMenu – головне меню;

view – перегляд товарів користувача

# **3.1.3. Опис реалізації основних функцій/методів**

void StartWindow() – початкове меню;

void auth() – меню авторизації;

void newUser() – меню реєстрації;

void load() – загрузити з файлу дані про тавари;

void save() – зберегти у файл дані про товари;

vector<goodsInfo> getUserGoods(string name); – метод отримання товарів вибраного користувача;

void add(string userN, string name, date AddedDate, date storageDate, int weight, int volume, int office); – додавання товару;

bool remove(int num, string curUsername); – видалення товару;

void load(load) – завантажити з файлу дані про користувачів;

void save() – зберегти у файл дані про користувачів;

bool auth(string username, string password) – авторизація;

bool newUser(string username, string password) – реєстрація;

vector <string> split(string str, string divider); – функція розділення строки за розділювачем;

void mainMenu() – головне меню користувача;

void view() – перегляд товарів користувача;

**4. Відлагодження та тестування реалізованої програми згідно індивідуального завдання**

**4.1. Аналіз структурної складності виконання програми**

• Відлагодження та тестування окремих частин програми:

Під час тестування окремих частин програм з’являлись певні недоліки та помилки. Щоб все правильно працювало потрібно було не дати можливості користувачу змушувати програму некоректно працювати. Тож всі можливі неполадки були ізольовані і опрацьовані так, щоб з’являлось певне повідомлення з певною інформацією, звідки користувач дізнавався як слід користуватися програмою, щоб уникати таких невизначеностей.

• Відлагодження та тестування програми в цілому:

Подібні кроки були виконані і з перевіркою всієї програми. Нижче наведені скріншоти, які запевнюють в коректній роботі програми:

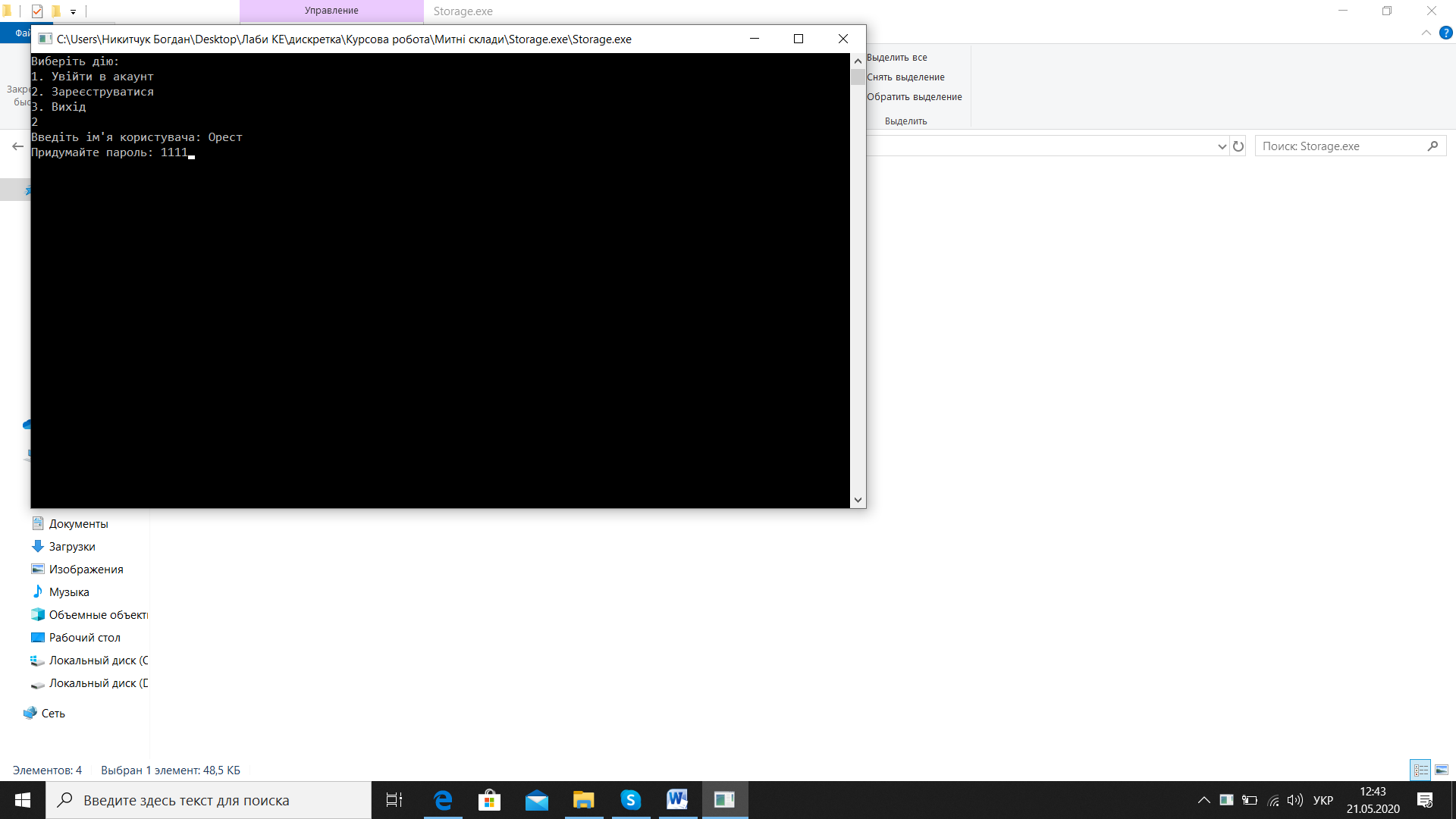


Рис. 4.1. Головне меню програми

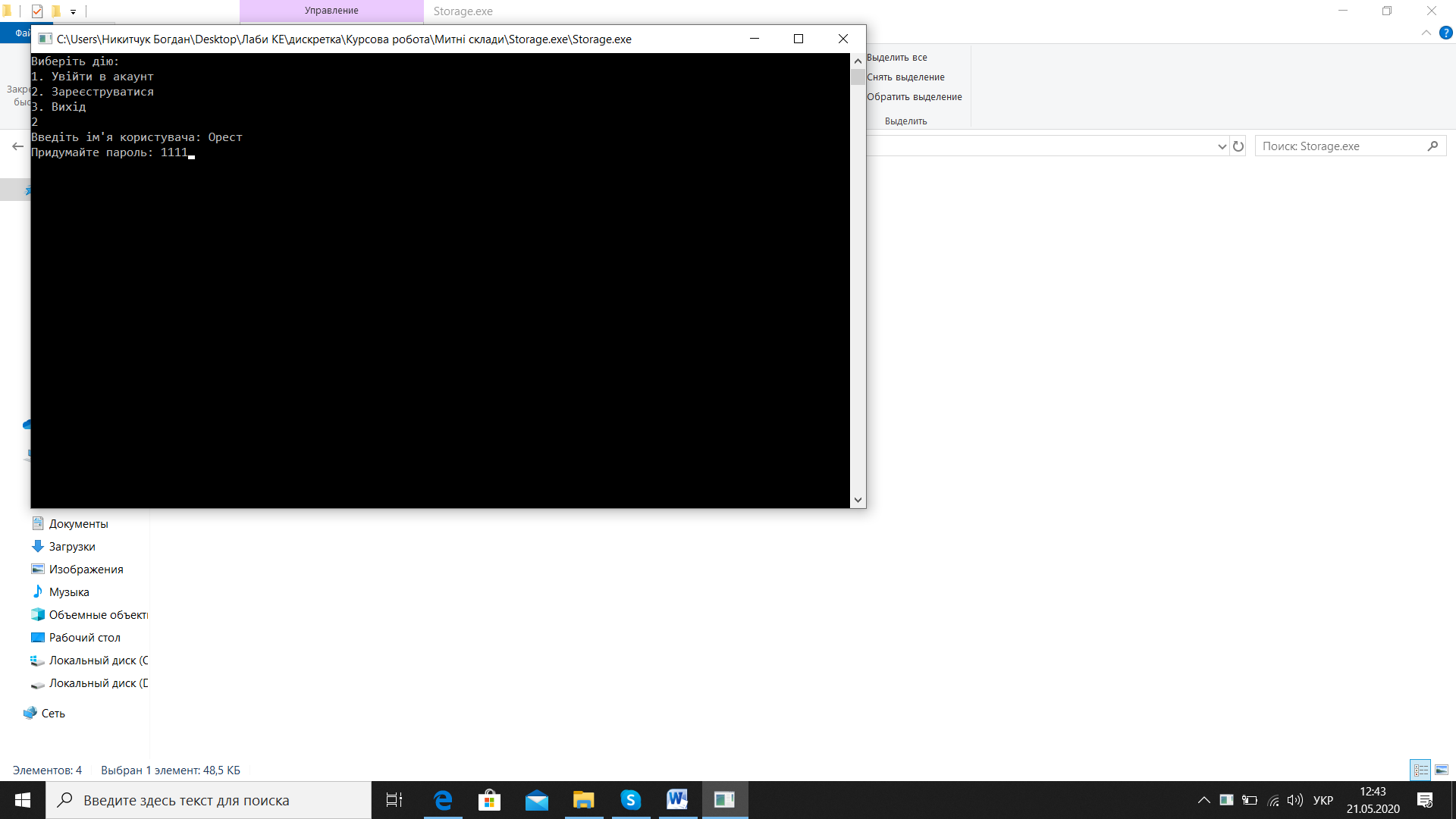


Рис. 4.2. Реєстрація акаунта

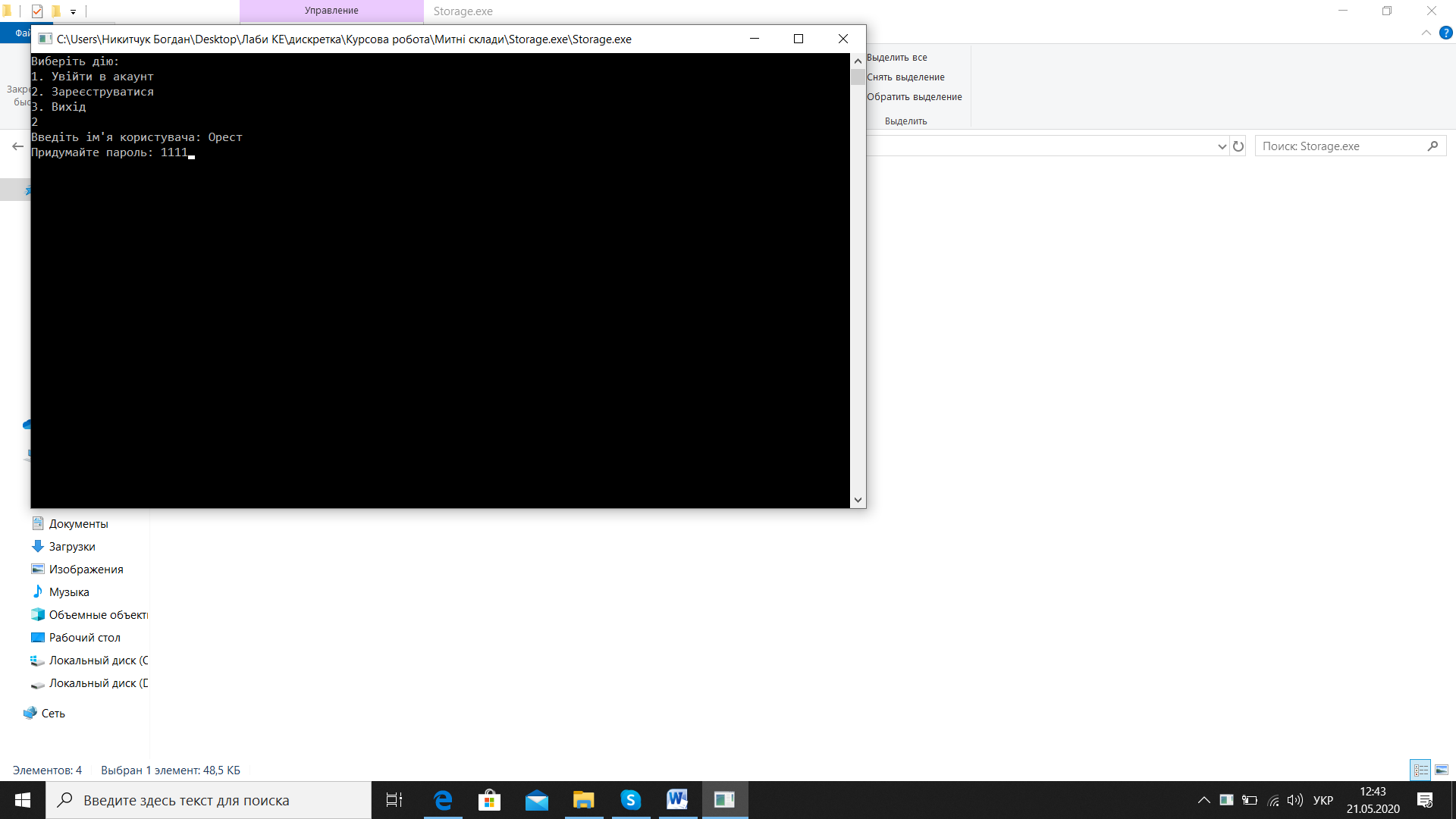


Рис. 4.3. Вхід в акаунт

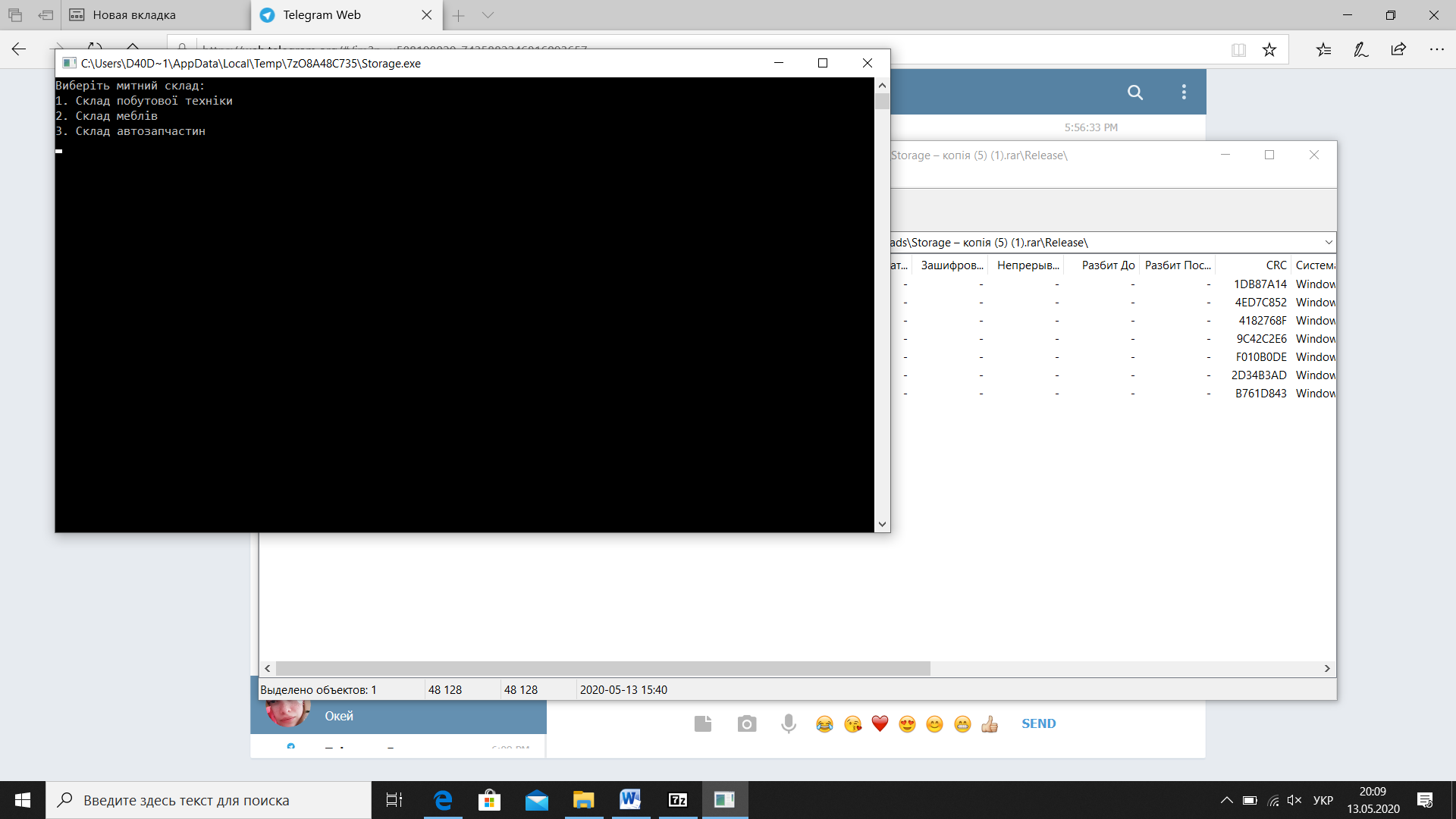


Рис. 4.4. Вибір митного пункту

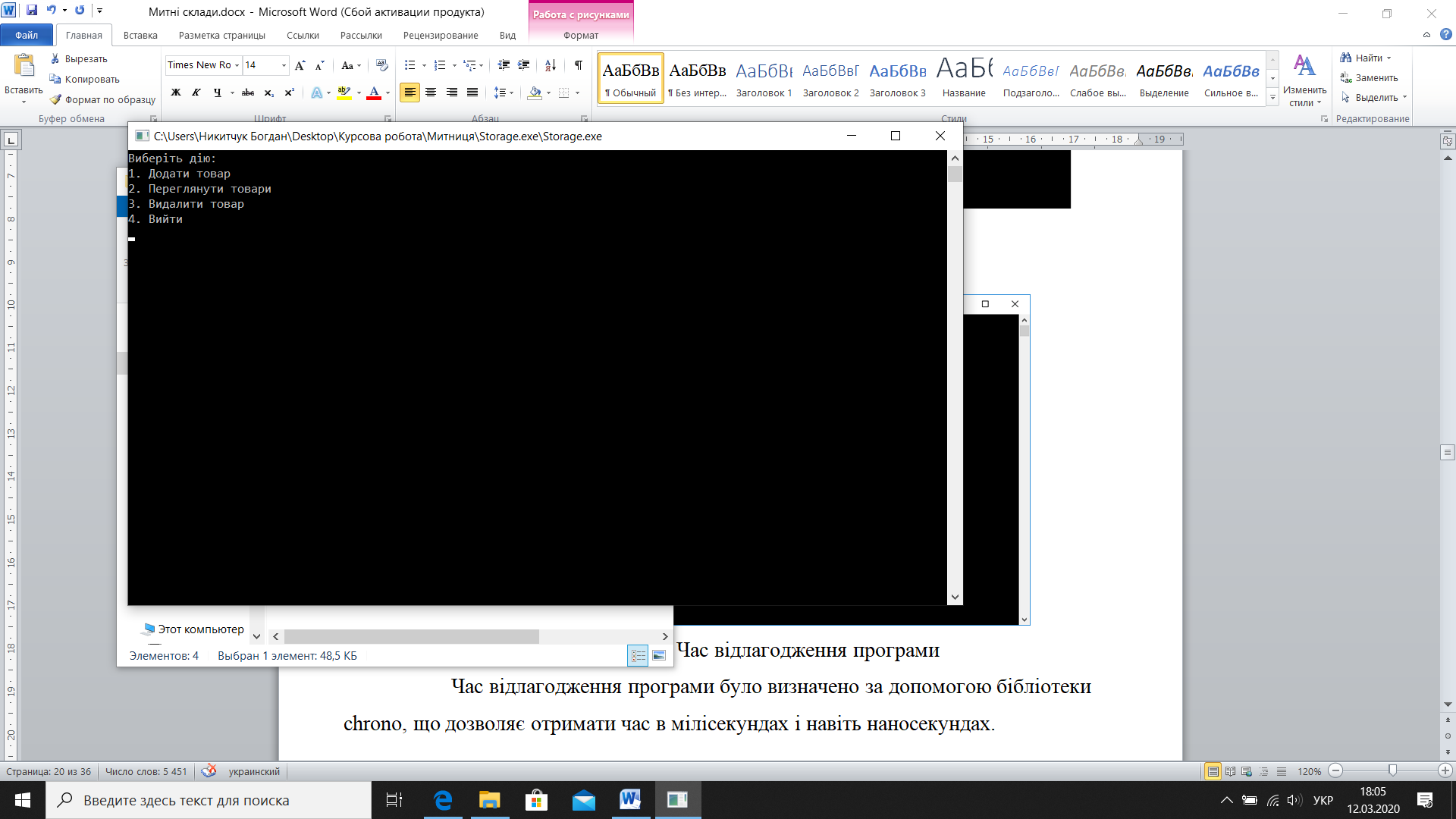


Рис. 4.5. Меню керування товарами

# **Висновок**

В ході написання курсової роботи я на практиці застосував свої знання з об’єктно-орієнтованого програмування, а також поглибив розуміння основних принципів об’єктно-орієнтованої ідеології програмування. Крім цього, я відпрацював на практиці основні методи та засоби об’єктно-орієнтованого програмування, навчився розробляти ієрархію класів та практично застосував такі поняття, як інкапсуляція, наслідування, перевантаження та перевизначення функцій.

**Список використаної літератури**

1. В.В. Подбєльский, С.С. Фомін “Програмування на мові Сі”/ Перекл. з англ. - Спб .: Питер, 2004. - 212 с.

2. М. Уєйт, С. Прата, Д. Мартін “Мова Сі”// Пер. з англ. - М .: «Видавництво Біном», СПб: «Невський діалог», 1998 р - 560 с.

3. Павловская Т.А. С/С++. Программирование на языке высокого уровня. – СПб.: Питер, 2007. – 354с.

**Додаток**

# **Опис програмної реалізації алгоритму**

CUI.h

#include "global.h"

#include "DatabaseUsers.h"

#include "user.h"

class CUI

{

private:

DatabaseUsers DatabaseSession;

public:

void StartWindow(); //початкове меню

void auth(); //меню авторизації

void newUser(); //меню реєстрації

};

CUI.cpp

#include "CUI.h"

void CUI::StartWindow()

{

system("cls"); //Очищення екрану

cout << "Виберіть дію:" << endl;

cout << "1. Увійти в акаунт" << endl;

cout << "2. Зареєструватися" << endl;

cout << "3. Вихід << endl;

int ch;

cin >> ch;

switch (ch)

{

case 1:

{

auth();

break;

}

case 2:

{

newUser();

break;

}

case 3:

{

exit(0);

break;

}

default:

StartWindow();

}

}

void CUI::auth()

{

string username, password;

bool correct = 0;

cout << "Введіть ім’я користувача: ";

cin >> username;

cout << "Введіть пароль: ";

cin >> password;

correct = DatabaseSession.auth(username, password);

if (correct == 1)

{

user curUser(username);

curUser.officeSelector(); //відкриття меню користувача

system("pause");

StartWindow();

}

if (correct == 0)

{

cout << "Неправильне ім’я користувача або пароль" << endl;

system("pause");

StartWindow();

}

}

void CUI::newUser()

{

string username, password;

cout << "Введіть ім’я користувача: ";

cin >> username;

cout << "Придумайте пароль: ";

cin >> password;

bool available = DatabaseSession.newUser(username, password);

if (available == 1) //Перевірка на наявність в базі імені користувача

{

cout << Ім’я користувача вже існує" << endl;

system("pause");

}

StartWindow();

}

DatabaseStorage.h

#include "global.h"

constexpr auto fileNameG = "goods.txt";

struct date {

int day, month, year;

};

struct goodsInfo {

int id, weight, volume, office;

string name, userN;

date dateAdded, storageDate;

};

class DatabaseStorage

{

private:

vector<goodsInfo> goods;

int curId;

void load(); //зберегти у файл дані про користувачів

public:

DatabaseStorage();

void save(); //загрузити з файла дані про користувачів

vector<goodsInfo> getUserGoods(string name); //метод отримання товарів вибраного користувача

void add(string userN, string name, date AddedDate, date storageDate, int weight, int volume, int office); //додавання товару

bool remove(int num, string curUsername); //видалення товару

};

DatabaseStorage.cpp

#include "DatabaseStorage.h"

DatabaseStorage::DatabaseStorage()

{

ifstream file("curId.txt"); //відкриття файлу

string temp;

file >> temp; //зчитування останнього id з файлу

curId = stoi(temp);

file.close(); //закриття файлу

}

void DatabaseStorage::load()

{

int i = 0;

goods.clear();

string temp;

vector <string> buff;

ifstream file(fileNameG); //відкриття файлу

while (file.eof() != 1) //читання построково поки не закінчиться файл

{

file >> temp;

buff = split(temp, div);

goods.resize(goods.size() + 1); //запис данних у вектор

goods[i].id = stoi(buff[0]);

goods[i].userN = buff[1];

goods[i].name = buff[2];

goods[i].dateAdded.day = stoi(buff[3]);

goods[i].dateAdded.month = stoi(buff[4]);

goods[i].dateAdded.year = stoi(buff[5]);

goods[i].storageDate.day = stoi(buff[6]);

goods[i].storageDate.month = stoi(buff[7]);

goods[i].storageDate.year = stoi(buff[8]);

goods[i].weight = stoi(buff[9]);

goods[i].volume = stoi(buff[10]);

goods[i].office = stoi(buff[11]);

i++;;

}

goods.pop\_back();

file.close(); //закриття файлу

}

void DatabaseStorage::save()

{

ofstream file(fileNameG); //відкриття файлу

for (unsigned int i = 0; i < goods.size(); i++)

{

file << goods[i].id << div << //запис данних у файл

goods[i].userN << div <<

goods[i].name << div <<

goods[i].dateAdded.day << div <<

goods[i].dateAdded.month << div <<

goods[i].dateAdded.year << div <<

goods[i].storageDate.day << div <<

goods[i].storageDate.month << div <<

goods[i].storageDate.year << div <<

goods[i].weight << div <<

goods[i].volume << div <<

goods[i].office << endl;

}

file.close(); //закриття файлу

goods.clear();

}

vector<goodsInfo> DatabaseStorage::getUserGoods(string name) //метод отримання товарів вибраного користувача

{

load(); //завантаження з файлу даних про товари

vector<goodsInfo> userGoods;

for (unsigned int i = 0; i < goods.size(); i++)

{

if (goods[i].userN == name) //перевірка чи товар належить даному користувачу

{

userGoods.push\_back(goods[i]);

}

}

return userGoods;

}

void DatabaseStorage::add(string userN, string name, date dateAdded, date storageDate, int weight, int volume, int office)

{

ofstream file(fileNameG, ios\_base::app); //відкриття файлу

file << curId << div << //запис в кінець файлу

userN << div <<

name << div <<

dateAdded.day << div <<

dateAdded.month << div <<

dateAdded.year << div <<

storageDate.day << div <<

storageDate.month << div <<

storageDate.year << div <<

weight << div <<

volume << div <<

office << endl;

file.close();

curId++;

ofstream fileNum("curId.txt");

fileNum << curId;

fileNum.close(); //закриття файлу

}

bool DatabaseStorage::remove(int num, string curUsername)

{

load();

for (unsigned int i = 0; i < goods.size(); i++)

{

if (goods[i].userN == curUsername && goods[i].id == num) //перевірка чи товар належить користувачу

{

goods.erase(goods.begin() + i); //видалення товару за ID

save();

goods.clear();

return 1;

}

}

return 0;

}

DatabaseUsers.h

#include "global.h"

constexpr auto fileName = "users.txt";

class DatabaseUsers

{

private:

struct userInfo

{

string username, password;

};

vector <userInfo> Data;

public:

void load(); //загрузити з файла дані про користувачів

void save(); //зберегти у файл дані про користувачів

bool auth(string username, string password); //авторизація

bool newUser(string username, string password); //реєстрація

};

DatabaseUsers.cpp

#include "DatabaseUsers.h"

void DatabaseUsers::load()

{

int i = 0;

Data.clear();

string temp;

vector <string> buff;

ifstream file(fileName); //відкриття файлу

while (file.eof() != 1) //читання построково поки не закінчиться файл

{

file >> temp;

buff = split(temp, udiv);

Data.resize(Data.size() + 1); //запис данних у вектор

Data[i].username = buff[0];

Data[i].password = buff[1];

i++;;

}

Data.pop\_back();

file.close(); //закриття файлу

}

void DatabaseUsers::save()

{

ofstream file(fileName); //відкриття файлу

for (unsigned int i = 0; i < Data.size(); i++)

{

file << Data[i].username << udiv << Data[i].password << endl; //запис данних у файл

}

file.close(); //закриття файлу

Data.clear();

}

bool DatabaseUsers::auth(string username, string password)

{

system("cls"); //очищення екрану

load();

bool correct = 0;

for (unsigned int i = 0; i < Data.size(); i++)

{

if (Data[i].username == username && Data[i].password == password)

{

correct = 1;

break;

}

}

Data.clear();

return correct;

}

bool DatabaseUsers::newUser(string username, string password)

{

system("cls"); //очищення екрану

load();

bool available = 0;

for (unsigned int i = 0; i < Data.size(); i++)

{

if (Data[i].username == username) //перевірка на наявність імені користувача

{

available = 1;

break;

}

}

Data.clear();

if (available == 0)

{

ofstream file(fileName, ios\_base::app); //відкриття файлу

file << username << udiv << password << endl;

file.close(); //закриття файлу

}

return available;

}

global.h

#ifndef \_GLOBAL\_H\_

#define \_GLOBAL\_H\_

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <iostream>

#include <iomanip>

#include <Windows.h>

#include <vector>

#include <string>

#include <fstream>

#include <iterator>

using namespace std;

#define udiv ":"

#define div ";"

vector <string> split(string str, string divider); //функція розділення строки за розділювачем

#endif

global.cpp

#include "global.h"

vector <string> split(string str, string divider) //функція розділення строки з розділювачем

{

vector <string> res;

int start = 0, end, delta = divider.length(), i = 0;

while ((end = str.find(divider, start)) != string::npos)

{

res.push\_back(str.substr(start, end - start));

start = end + delta;

}

res.push\_back(str.substr(start));

return res;

}

Storage.cpp

#include "global.h"

#include "CUI.h"

int main()

{

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

CUI session;

session.StartWindow();

return 0;

}

User.h

#include "global.h"

#include "DatabaseStorage.h"

class user

{

private:

string curUsername;

unsigned int curOffice;

vector<goodsInfo> curUserGoods;

vector<string> offices = { "Ягодин", "Рава Руська", "Ужгород", "Прикарпаття", "Пісочин", "Бачівськ", "Удрицьк", "Сновськ" };

DatabaseStorage DatabaseSession;

public:

user();

user(string curUsername);

void officeSelector();

void mainMenu(); //головне меню користувача

void view(); //перегляд товарів користувача

};

User.cpp

#include "user.h"

user::user()

{

}

user::user(string curUsername)

{

this->curUsername = curUsername;

}

void user::officeSelector()

{

system("cls"); //очищення екрану

cout << "Виберіть митний пункт:" << endl;

cout << "1. " << offices[0] << endl;

cout << "2. " << offices[1] << endl;

cout << "3. " << offices[2] << endl;

cout << "4. " << offices[3] << endl;

cout << "5. " << offices[4] << endl;

cout << "6. " << offices[5] << endl;

cout << "7. " << offices[6] << endl;

cout << "8. " << offices[7] << endl;

unsigned int k;

cin >> k;

if (k == 0 or k > 8)

{

cout << "Такого пункту не існує" << endl;

system("pause");

officeSelector();

}

else

{

curOffice = k - 1;

}

mainMenu();

}

void user::mainMenu()

{

system("cls"); //очищення екрану

cout << "Виберіть дію:" << endl;

cout << "1. Додати товар" << endl;

cout << "2. Переглянути товари" << endl;

cout << "3. Видалити товар" << endl;

cout << "4. Вийти" << endl;

int ch;

cin >> ch;

switch (ch)

{

case 1:

{

system("cls"); //очищення екрану

string name;

date dateAdded, storageDate;

int weight, volume;

time\_t t = time(0); //одержання поточної дати

tm\* now = localtime(&t);

dateAdded.year = now->tm\_year + 1900;

dateAdded.month = now->tm\_mon + 1;

dateAdded.day = now->tm\_mday;

storageDate.year = dateAdded.year + 2;

storageDate.month = dateAdded.month;

storageDate.day = dateAdded.day;

cout << "Введіть назву товару: " << endl;

cin >> name;

cout << "Введіть масу товару: " << endl;

cin >> weight;

cout << "Введіть об'єм товару: " << endl;

cin >> volume;

DatabaseSession.add(curUsername, name, dateAdded, storageDate, weight, volume, curOffice); //додавання товару

mainMenu();

break;

}

case 2:

{

system("cls"); //очищення екрану

view(); //перегляд товарів

system("pause");

mainMenu();

break;

}

case 3:

{

system("cls"); //очищення екрану

view();

cout << "Введіть ID товару якій бажаєте видалити: ";

int id;

cin >> id;

if (DatabaseSession.remove(id, curUsername) == 0) //перевірка чи товар належить користувачу

{

cout << "Це не ваш товар" << endl;

}

system("pause");

mainMenu();

break;

}

case 4:

{

exit(0);

break;

}

default:

mainMenu();

}

}

void user::view()

{

vector<goodsInfo> userGoods = DatabaseSession.getUserGoods(curUsername);

cout << setw(4) << "ID" << setw(20) << "Ім'я" << setw(20) << "Дата додавання" << setw(30) << "Дата завершення зберігання" << setw(10) << "Вага" << setw(10) << "Об'єм" << setw(15) << "Пункт" << endl; //вивід на екран даних про товари

if (userGoods.empty() == 0) //перевірка чи користувач має товари

{

for (unsigned int i = 0; i < userGoods.size(); i++)

{

cout << setw(4) << userGoods[i].id << setw(20) <<

userGoods[i].name << setw(13) <<

userGoods[i].dateAdded.day << "." <<

userGoods[i].dateAdded.month << "." <<

userGoods[i].dateAdded.year << setw(23) <<

userGoods[i].storageDate.day << "." <<

userGoods[i].storageDate.month << "." <<

userGoods[i].storageDate.year << setw(8) <<

userGoods[i].weight << "кг" << setw(9) <<

userGoods[i].volume << "л" << setw(15) <<

offices[userGoods[i].office] << endl;

}

}

}