Міністерство науки і освіти України
Харківський національний університет радіоелектроніки

Кафедра Інформаційних управляючих систем

КУРСОВА РОБОТА

з дисципліни

ТЕХНОЛОГІЇ ОБ'ЄКТНО ОРІЄНТОВАНОГО ПРОГРАМУВАННЯ

на тему:

Розробка об'єктно-орієнтованої програмної системи «Фермерське господарство»

Роботу виконала	Керівник
ст. гр. ІТУ-20-1	_____ Чала О.В._____
Троценко Анастасія Ігорівна	До захисту
"____"_____2021__ р.	" 16 " червня 2021р.
_____ (підпис)	_____ (підпис)

Захист курсової роботи

_____ _____
(дата) (оцінка)

Голова комісії _____ _____
(підпис)

Члени комісії _____ _____
(підпис)

_____ _____
(підпис)

_____ _____
(підпис)

Харків 2021

Харківський національний університет радіоелектроніки

Факультет КН Кафедра ІУС

Спеціальність 122 Комп'ютерні науки

Освітня програма Інформаційні технології управління

ЗАВДАННЯ

на курсову роботу з дисципліни «Технології об'єктно орієнтованого програмування»

студенту Троценко Анастасії Ігорівні

групи ІТУ-20-1

1. Тема роботи розробка об'єктно-орієнтованої програмної системи «Фермерське господарство»
2. Строк захисту студентом закінченої роботи 11.06.2020
3. Вихідні дані до роботи опис предметної області, базовий клас Rabotnik, похідні класи: Fermer, Rukovoditel. Базовий клас Produkcija, похідні класи: Zerno, Miaso, Fructi, Ovocshi. Базовий клас Potreba, похідні класи: Udobrenija, SredstvaPriizvodstva, TransportnijeUslugi. Базовий клас Error, похідні класи: ProdukcijaError, FermerError, PotrebaError, FileError.
4. Зміст пояснювальної записки (перелік питань, що повинні бути розроблені) вступ, аналіз предметної області і постановка задач курсової роботи, опис вхідної та вихідної інформації програми, опис використаних у програмі класів та інших засобів мови програмування, схема алгоритму роботи програми, діаграма класів, текст програми з коментарями, приклад виконання програми, висновки.
5. Перелік графічного матеріалу (с точним зазначенням обов'язкових креслень) діаграма класів, алгоритм програми, екранні форми.
6. Дата видачі завдання 16.03.2020

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів курсової роботи	Термін виконання	Примітка
1	Отримання завдання на курсову роботу	16.03.21	
2	Аналіз предметної області	20.03.21	
3	Побудова діаграми класів	30.03.21	
4	Побудова схеми алгоритму роботи програми	10.04.21	
5	Розробка програми	10.05.21	
6	Налагодження програми	20.05.21	
7	Тестування програми	25.05.21	
8	Оформлення пояснювальної записки	29.05.21	
9	Захист курсової роботи	11.06.21	

Студент _____ А. І. Троценко
(підпис) (ініціали, прізвище)

Керівник
курсової роботи _____ О. В. Чала
(підпис) (ініціали, прізвище)

РЕФЕРАТ

Пояснювальна записка до курсової роботи містить: 60 стор., 4 рис., 3 додатків, 6 джерел.

ІНФОРМАЦІЙНА СИСТЕМА, ФЕРМЕРСЬКЕ ГОСПОДАРСТВО, ПРОДУКЦІЯ, КЛАС, С++, ДІАГРАМА КЛАСІВ.

Мета роботи – розробка математичного та програмного забезпечення для задачі управління та контролю виробництва фермерських господарств.

Було проведено опис вхідної та вихідної інформації задачі, проаналізований вибір програмних засобів розробки програмного забезпечення задачі, розроблена схема роботи задачі, діаграма класів та програма задачі управління та контролю виробництва фермерським господарством.

Програма забезпечує швидке одержання необхідної та внесення додаткової інформації, що підвищує ефективність роботи фермерства та зручний контроль для керівника.

Областю застосування проектованого програмного забезпечення є органи державної влади та органи місцевого самоврядування, а також ЦСУ (центральне статистичне управління), що здійснюють контроль за діяльністю фермерських господарств.

ЗМІСТ

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАКИ

ООП – об’єктно-орієнтоване програмування.

ЦСУ – центральне статистичне управління;

STL – Standard Template Library – стандартна шаблонна бібліотека C++.

FIFO (first in - first out)– принцип роботи черги, перший зайшов – перший вийшов.

Windows Forms – назва інтерфейсу програмування додатків, що відповідає за графічний інтерфейс користувача і є частиною Microsoft .NET Framework

ВСТУП

Сучасний розвиток фермерства набуває все більшої актуальності. Останнім часом фермерські господарства посідають важливе місце серед інших організаційно-правових форм товарного сільськогосподарського виробництва, адже зараз в Україні їх функціонує більше ніж 40 тисяч. Тому надзвичайно важливо, щоб відбувався чіткий та зручний контроль і доступ до інформації про виготовлену продукцію та фермерів.

Безперечною перевагою використання об’єктно-орієнтованого підходу у розв’язанні даної задачі є концептуальна близькість до предметної області довільної структури та призначення. Механізм спадкоємства атрибутів і методів дозволяє будувати похідні поняття на основі базових, і, таким чином, створювати модель предметної області із заданими властивостями.

Метою курсової роботи є розробка елементів прикладного інформаційного забезпечення для фермерського господарства, за допомогою можливостей мови програмування C++ та графічної бібліотеки Windows Forms. Таким чином, для виконання курсової роботи необхідно виконати наступні завдання:

- проаналізувати та дослідити принципи роботи фермерств;
- визначити зв’язки між кредитами та прибутком, виготовленою продукцією та потребами для господарств;
- встановити логіку роботи програми для досягнення максимально зручного доступу до даних;
- розробити програмну реалізацію алгоритмів внесення даних та отримання потрібної інформації.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ І ПОСТАНОВКА ЗАДАЧ КУРСОВОЇ РОБОТИ

На сьогодні фермерське господарство – це складна структура з безліччю різних інформаційних зв’язків. Воно охоплює різні види та типи сільського господарства, пов’язує між собою різних фермерів та керівників. Не менш важливою є й система виділення кредитів та розрахунку прибутків. Для того, щоб такий механізм працював справно, потрібно щоб керівництво могло зручно ним управляти та прослідковувати зміни.

Сучасний застосунок допоможе структурувати інформацію, швидко знайти потрібні дані, внести зміни, та записати звіт у текстовий документ. Його використання дозволяє не витрачати купу часу на елементарні дії, які є реалізовані в ньому, натомість у керівництва з’явиться можливість приділяти даний час оптимізації виробництва та поліпшенню внутрішніх процесів господарства. До того ж, слід звернути увагу на ряд переваг: швидку роботу, зручний і зрозумілий інтерфейс, можливість знайти потрібну інформацію в будь-який момент.

Таким чином, створивши додаток для систематизації даних про фермерське господарство, ми значно скоротили витрати часу та об’єм роботи бухгалтерів. З іншого боку, надали можливість ЦСУ швидко отримувати необхідні дані. За допомогою прикладної програми можна зручно та легко реєструвати нових фермерів, отримувати інформацію про продукцію, що виготовляється, та потреби для її виробництва, обчислювати необхідний кредит кожному фермеру, виводити його прибутки та вартість продуктів.

У програмі створені узагальнені списки: відомості про фермерів; відомості про вироблену продукцію; відомості про потреби.

Керівник області може зробити запит про такі відомості:

– яку продукцію виробляють фермери області;
– що потрібно кожному фермеру для виробництва;
– яка кількість заданої продукції виробляють фермери;
– прибуток фермерів по кожному виду продукції;
– який кредит потрібно кожному фермеру;
– яка різниця між кредитом і отриманим прибутком.

Крім того, реалізовано функції додавання та видалення елементів, очищення даних, довідка про програму, запис та зчитування з файлу, вихід з програми.

При розробці програми було виділено такі основні сутності: керівник, фермер, потреба, продукція. У класі Фермер реалізовано присвоєння імені фермеру, додавання, пошук та видалення продукції, а також задання необхідного бюджету. Сутність Керівник має змогу додавати, знаходити та видаляти фермерів. Потреби можуть отримувати назву, тип, кількість та вартість. Клас Продукція забезпечує можливість додавати та видаляти потреби, отримувати назву, тип, кількість, ціну продуктів. Для того, щоб зберігати та керувати об’єктами цих класів було створено ще й додаткові сутності. Ними виступають власні поліморфні класи-контейнери, реалізовані у вигляді черги. Також було створено класи-ітератори, що описують інтерфейс для доступу та обходу елементів класів-контейнерів.

Було розроблено зручний та зрозумілий інтерфейс програми, щоб будь-який користувач міг легко в ній розібратися. В якості графічної бібліотеки використано Windows Forms. Деякі дані користувач може вносити самостійно, а інші потрібно вибрати із запропонованих за допомогою елементу ComboBox. Оброблено всі можливі випадки виникнення виключень, створено ієрархію класів винятків, при спрацюванні яких з’являється інформаційне вікно з описом помилки.

2 ОПИС ВХІДНОЇ І ВИХІДНОЇ ІНФОРМАЦІЇ ПРОГРАМИ

Завданням програми є надання користувачеві можливості вводити дані про фермерів їх потреби і продукцію у текстові поля, деякі інші дані обирати із запропонованих. Після чого інформація оброблюється та виводиться за запитами користувача. Реалізовано можливість зчитування та запису інформації у файли.

Спочатку при запуску програми у ній відсутні будь-які дані. Користувач може ввести самостійно ім’я та бюджет фермера. Після цього слід обрати фермера в розділі «Продукція» та заповнити інформацію про назву, вид, кількість, вартість та одиниці продукції яку виготовляє саме цей фермер. В розділі «Потреби» за допомогою випадаючих списків обирається ім’я фермера, його продукція, а користувач заповнює дані про тип, назву, кількість і вартість потреб для її виготовлення.

Вся вище описана інформація може бути введена з клавіатури чи завантажена в програму із файлу, після чого є можливість доповнити інформацію самостійно та зберегти в цей же файл зі змінами або завантажити у новий.

Інформація може зчитуватися та записуватися у файл формату «.txt» . Цей файл повинен мати чітко описану структуру. Інформація у файлі подається за таким же принципом, як й інтерфейс програми розподілено на сектори: перший сектор відомості про фермера, другий – про продукцію, третій – про потреби, між секторами лексема для відділення – «---» (три знака мінус). Крім того, дані у файлі повинні бути лише англійською мовою, через особливості кодування символів різних мов.

Вихідні дані програми – інформація подана у вигляді таблиць, що формуються за запитами користувача.

Користувач може зробити запити:

– про продукцію, яку виробляють фермери області;
– що потрібно кожному фермеру для виробництва;
– яка кількість заданої продукції виробляють фермери;
– прибуток фермерів по кожному виду продукції;
– який кредит потрібно кожному фермеру;
– яка різниця між кредитом і отриманим прибутком.

Після отримання запиту, в програмі ведеться збір, обробка інформації та створюється потрібна таблиця й виводиться на екран.

3 ОПИС ВИКОРИСТАНИХ В ПРОГРАМІ КЛАСІВ ТА ІНШИХ ЗАСОБІВ МОВИ ПРОГРАМУВАННЯ

Вихідний код програми розміщений у декількох файлах, у кожному розміщено базовий класи та його нащадки, також відділено інтерфейс класів від реалізації.

Діаграми класів файлів Fermer.h, Produkciia.h, Potreba.h, Error.h представлені в додатку А.

Клас Produkcija, розміщений у файлі «Produkciia.h» розроблений для роботи з відомостями про продукцію, яку можуть виробляти фермера. Він мітить поля із захищеним(protected) доступом. Поле string classname – описує тип продукції, int kolichestvo – кількість продукції, double cenazaedenicu - ціна за одиницю, string ed – одиниці вимірювання продукції, string name – її назва. Також створено об’єкт класу PotrebaQueue potrebi, бо потреби та продукція залежать один від одного.

Опис призначень відкритих методів класу:

– virtual string getName(void) const гетер для отримання даних про назву продукції;

– virtual string getClassName(void) const гетер для отримання даних про тип продукції;

– virtual double getCenazaedenicu(void) const гетер для отримання даних про ціну продукції;

– virtual void setCenazaedenicu(const double cenazaedenicu) сетер для присваювання значення ціна за одиницю продукції;

– virtual int getKolichestvo(void) const гетер для отримання даних про кількість;

– virtual void setKolichestvo(const int kolichestvo) сетер для присваювання кількості продукції;

- virtual string getEdenici(void) const гетер для отримання даних про одиниці вимірювання кількості продукції;
- virtual void setEdenici(const string ed) сетер для присваювання даних про одиниці вимірювання кількості продукції;
- void addPotreba(Potreba* potreba) функція для додавання потреби, щоб виготовити задану продукцію;
- bool delPotreba(string potreba) видалення потреби;
- const PotrebaQueue& getPotrebiQueue(void) const гетер для отримання даних про потребу.

Також клас містить конструктор за параметрами Produkcija(string name, string classname, double cenazaedenicu, int kolichestvo, string ed) та без – Produkcija(void) і деструктор ~Produkcija() для звільнення пам’яті.

Цей клас має декілька похідних, що визначають тип продукції та успадковують всі методи, а саме класи Zerno, Miaso, Fructi, Ovocshi, які в свою чергу містять параметризовані конструктори для ініціалізації даних.

```
class Zerno : public Produkcija{
public:
Zerno(string name, double cenazaedenicu, int kolichestvo, string ed);};
class Miaso : public Produkcija
{public:
Miaso(string name, double cenazaedenicu, int kolichestvo, string ed);};
class Fructi : public Produkcija
{
public:
Fructi(string name,double cenazaedenicu, int kolichestvo, string ed);};
class Ovocshi : public Produkcija
{
public:
Ovocshi(string name, double cenazaedenicu, int kolichestvo, string ed);};
```

У файлі «Potreba.h» знаходиться клас Potreba призначений для опису та виконання дій над потребами.

Він містить такі захищені поля:

- int stoimost описує вартість потреби;
- string classname тип потреби;
- string name назва потреби;
- int kolichestvo кількість конкретної потреби.

Опис призначень методів класу:

- int getStoimost(void) const гетер для отримання даних про вартість потреби;
- void setStoimost(const int stoimost) сетер для присваювання значення вартості;
- int getKolichestvo(void) const гетер для отримання даних про кількість потреби;
- void setKolichestvo(const int kolichestvo) сетер для присваювання значення про кількість потреби;
- string getClassname(void) const гетер для отримання даних про тип потреби;
- string getName(void) const гетер для отримання даних про назву потреби;
- double stoimostPotrebnosti(void) функція для обчислення вартості заданої кількості потреби;
- Potreba() конструктор без параметрів;
- Potreba(string classname, string name, int stoimost, int kolichestvo) конструктор з параметрами для ініціалізації даних;

Цей клас має декілька похідних, що визначають тип потреби та успадковують всі методи, а саме класи Udobrenija, SredstvaProizvodstva, TransportnijeUslugi.

```
class Udobrenija : public Potreba{
public:
```

```
    Udobrenija(string name, double stoimost, int kolichestvo);};
class SredstvaProizvodstva : public Potreba
{
public:
    SredstvaProizvodstva(string name, double stoimost, int kolichestvo);};
class TransportnijeUslugi : public Potreba
{
public:
    TransportnijeUslugi(string name, double stoimost, int kolichestvo);};
```

Основним класом даної програми є клас Fermer розташований у файлі «Fermer.h», призначений для виконання дій над потребами та продукцією, він успадковується від класу Rabotnik.

Синтаксис класу Rabotnik:

```
class Rabotnik
{
protected:
    string name;// ім’я працівника
public:
    Rabotnik(string name);// параметризований конструктор

    };
```

Поля класу Fermer:

– ProdukcijaQueue prod об’єкт класу-контейнеру що працює з продукцією;
– double budjet бюджет необхідний фермеру;

Опис призначень методів класу:

– string getName(void) const метод для отримання доступу до імені;
– double getBudjet(void) const гетер для отримання даних про бюджет;
– void setBudjet(const double budjet) сетер для присваювання значення бюджету;
– void addProdukcija(Produkcija *produkcija) функція для додавання продукції;

– int delProdukcija(string produkcija, int kolichestvo) функція для видалення продукції;
– Produkcija* findProdukcija(string produkcija) метод для пошуку заданої продукції;
– const ProdukcijaQueue& getProdukcijaQueue(void) const метод що повертає значення об’єкта з черги;
– void prodToComboBox(ComboBox^ comboBox) метод що добавляє продукцію у випадаючий список;
– Fermer(void) конструктор без параметрів;
– Fermer(string name, double budjet) конструктор з параметрами;
– ~Fermer(void) деструктор для звільнення пам’яті.

У цьому файлі є ще один з основних класів Rukovoditel, який також є нащадком класу Rabotnik. Він призначений для роботи з фермерами.

Поля класу Rabotnik:

– FermerQueue fermers об’єкт класу-контейнеру для фермерів.

Опис призначень методів класу:

– void clearFermers(void) видалення усіх даних про фермерів;
– void addFermer(Fermer* fermer) додавання фермера;
– int delFermer(string name) видалення фермера по імені;
– Fermer* findFermer(string name) пошук фермера;
– const FermerQueue& getFermerQueue(void) const метод повертає фермера з черги;
– void fermersToComboBox(ComboBox^ comboBox) додавання фермера у випадаючий список;
– Rukovoditel(string name) параметризований конструктор;
– ~Rukovoditel(void) деструктор для звільнення пам’яті.

Крім того, у програмі реалізовано власні поліморфні класи-контейнери типу черги для кожного типу даних. У них продемонстровано алгоритми

вивчені в дисципліні «Теорія алгоритмів». Нижче наведено приклад одного з класів контейнерів:

Поля класу PotrebaQueue:

– PotrebaElem* back вказівник на початковий елемент;
– PotrebaElem* front вказівник на останній елемент.

Методи класу та їх призначення:

– PotrebaQueue(void) конструктор без параметрів;
– ~PotrebaQueue(void) деструктор;
– void push_back(Potreba* value) метод додавання елемента в кінець черги;
– Potreba* pop_back(void) метод видалення елементу з кінця черги;
– bool pop(string name) функція видалення заданого об’єкта черги;
– void clear(void) функція очищення;
– Potreba* find(string name) метод для пошуку елементу за заданою назвою;

Було створено ітератор, який здатний перебирати елементи контейнерного класу без необхідності користувачеві знати реалізацію певного класу–контейнеру. Після того, як ітератор відповідного типу створений, програміст може використовувати інтерфейс, що надається даним ітератором, для переміщення по елементах контейнера або доступу до його елементів, не турбуючись при цьому про те, який тип перебору елементів задіяний або яким чином в контейнері зберігаються дані. У класі-ітераторі реалізовано такі методи:

– PotrebaElem* begin(void) повертає вказівник на перший елемент черги;
– Potreba* getIter(void) повертає поточні дані;
– bool isEnd(void) превірка на кінець черги;
– void next(void) перехід до наступного елемента черги;

4 СХЕМА АЛГОРИТМУ РОБОТИ ПРОГРАМИ

При розробці програми було вирішено використати класи-контейнери типу черги , для зберігання і роботи з об’єктами інших класів. Черга - це структура даних, яка побудована за принципом FIFO (first in - first out: перший прийшов – перший вийшов). У ній реалізовано функції додавання в кінець, видалення і пошук заданого елементу та прохід по елементам.

У C ++ вже є готовий STL контейнер – queue, але в роботі було реалізовано власні не шаблонні контейнери, що мають такий же алгоритм роботи.

В курсовій роботі розроблено графічний інтерфейс програми за допомогою графічної бібліотеки Windows Forms.

Використані стандартні елементи управління:

– Text – для створення надпису на формі;
– ComboBox – випадаючий список;
– Button – кнопка;
– TextBox – поле для введення тексту;
– ToolStripMenuItem – забезпечує систему меню для форми;
– DataGridView – відображає дані таблиць;
– GroupBox – відображає рамку навколо групи елементів керування з підписом.

В меню програми користувач може обрати функції зчитування та завантаження інформації у обраний файл, очищення всіх відомостей, вихід з додатку та довідка про програму.

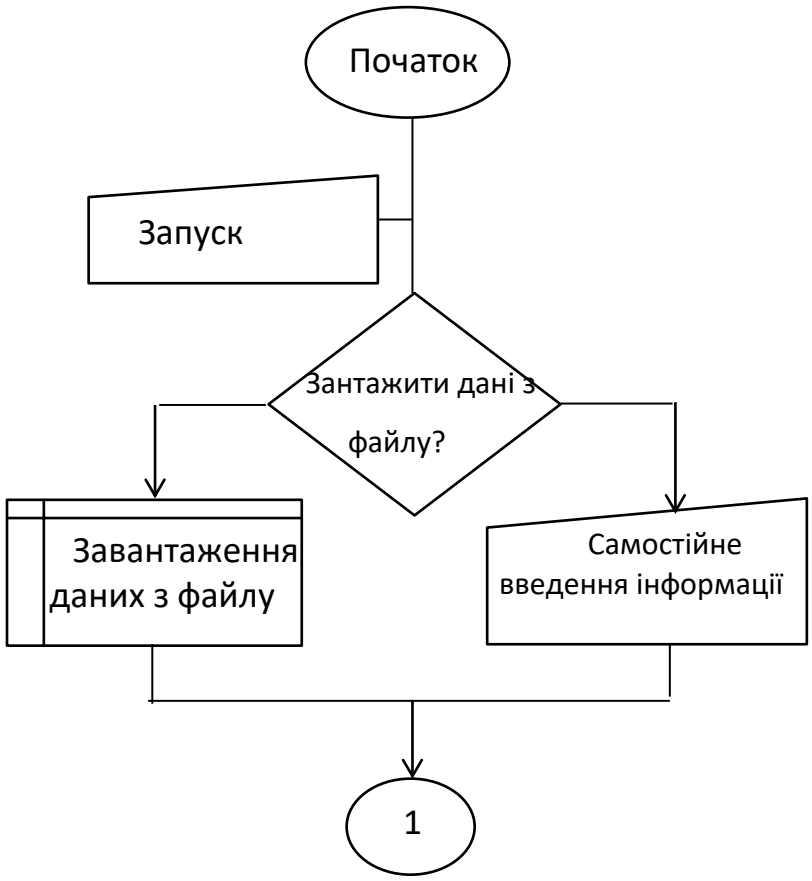
Діалогові вікна – це спеціальні елементи інтерфейсу призначені для виведення інформації або отримання відповіді від користувача. У курсовій роботі за допомогою діалогових вікон виводиться наступна інформація:

помилка в роботі програми, введення некоректних даних або успішне виконання функції.

Обробник подій – це підпрограма, яка опрацьовує матеріали, отримані з програми. Обробник певним чином реагує на події та починає виконувати дії, які згенерувала та чи інша подія. При створенні додатку було використано обробник подій Click, який спрацьовує, якщо користувач натиснув і відпустив основну клавішу миші, коли вказівник миші розташовано на об'єкті.

Основні задачі програми реалізовані в якості кнопок, які розміщені у візуальному інтерфейсі програми. При натисканні на них спрацьовує відповідна функція, прописана у головному файлі – «Fermerstvo.h».

На рисунку 4.1 приведено загальну схему роботи програми по обробці інформації.



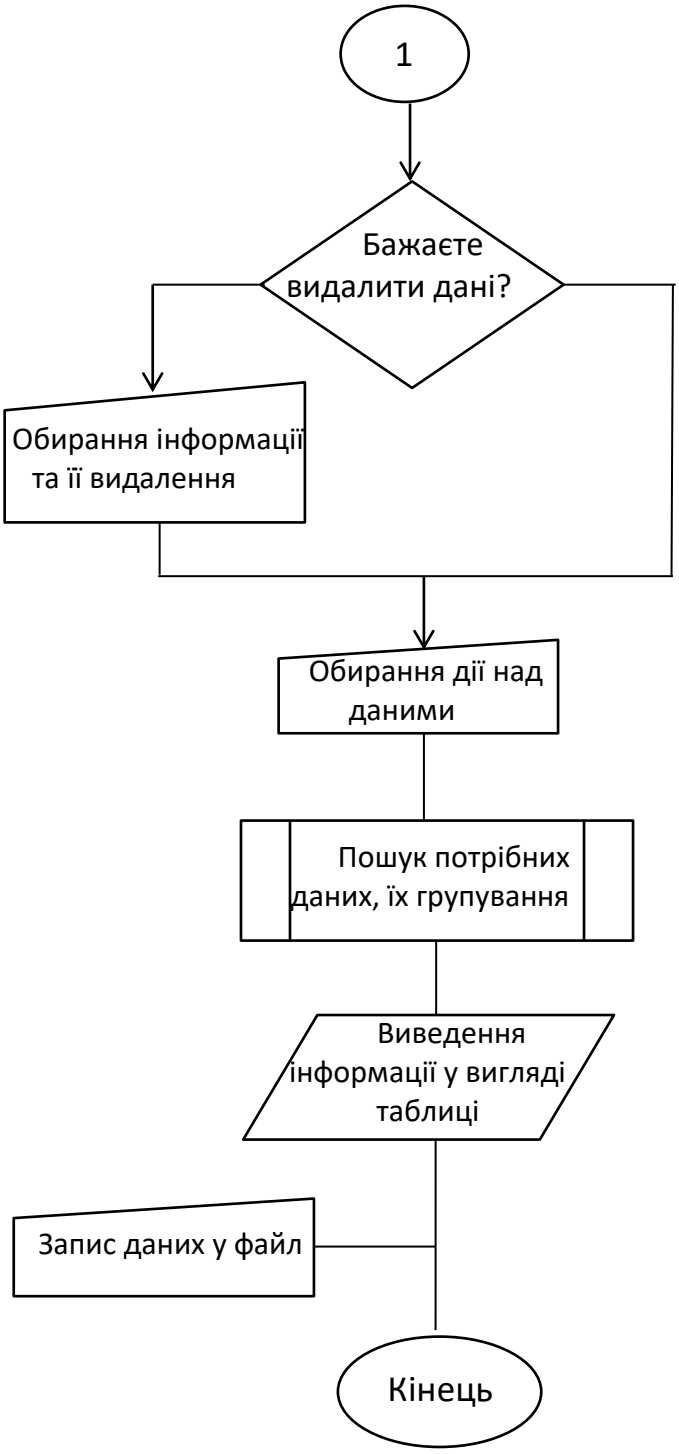


Рисунок 4.1 Схема алгоритму роботи програми

ВИСНОВОК

У курсовій роботі для розробки додатку було обрано мову програмування C++ та використано об’єктно-орієнтований підхід, який має свої переваги та недоліки. Серед переваг можна виділити такі: скорочення числа ймовірних помилок, можливість повторного використання функцій для вирішення інших завдань у даній галузі. Недоліками даного підходу є складність реалізації та необхідність введення додаткових способів подання інформації про предметну область і методів її аналізу.

У інформаційній системі реалізовано всі необхідні функції згідно із поставленою задачею, розроблено блок-схему алгоритму роботи програми та створено візуальний інтерфейс з використанням графічної бібліотеки Windows Forms. Також у роботі наведено діаграму класів, реалізовану за допомогою мови UML.

Створена прикладна система може знадобитися для керівників областей, ЦСУ і бухгалтерів для ведення та отримання звітів, моніторингу виробленої продукції. Додаток містить різні функції для обробки запитів, зокрема, додавання фермерів та виведення інформації про їх діяльність.

Таким чином, програма є дійсно корисною як для індивідуального використання, так і для великих виробництв, оскільки має ряд переваг: багатофункціональність, зручний і зрозумілий інтерфейс, можливість швидко знайти потрібну інформацію в будь-який момент, структурування даних та зберігання їх у файлах.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Грицюк Ю, Рак Т. Об’єктно-орієнтоване програмування мовою С++ : Навч. посібник. – Львів : Вид-во ЛДУ БЖД, 2011. 404 с.

2. Методичні вказівки до курсової роботи з дисципліни «Технології об’єктно-орієнтованого програмування» для студентів усіх форм навчання першого (бакалаврського) рівня вищої освіти спеціальності 122 – «Комп’ютерні науки» освітньої програми «Інформаційні технології управління» / Упоряд. Білова Т.Г., Севостьянова К.А. – Харків: ХНУРЕ, 2020. 75 с.

3. Badd T. Ob'ektno-orientirovannoe programmirovanie v deystvii. Piter, 1997. 304 str.

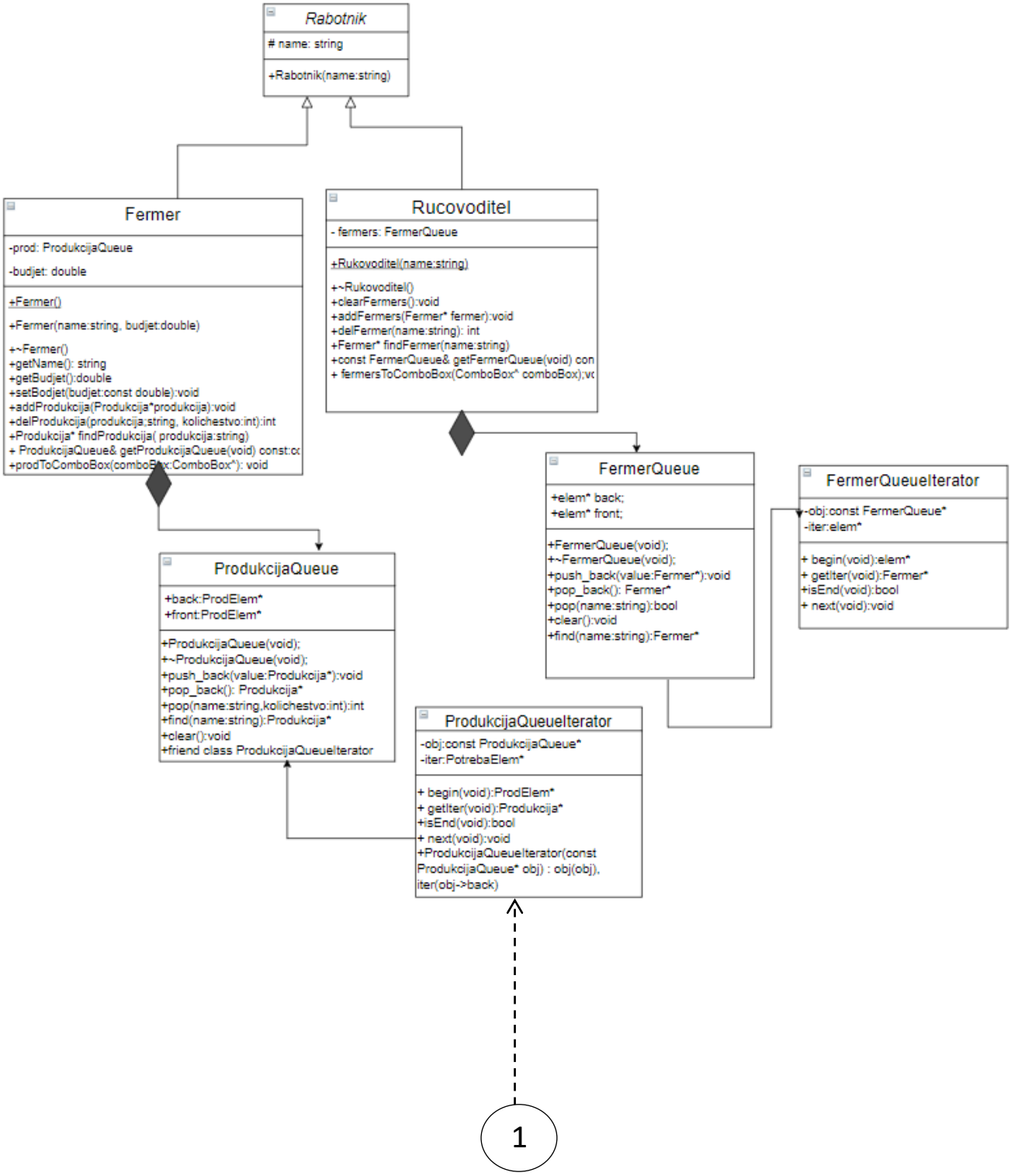
4. Bokova A.V., Morozov A. V., Laptev V. V. C++ OOP Zadachi i uprazhneniya. Piter, 2007. 288 p.

5. Kubenskiy A. A. Struktury dannykh i algoritmy obrabotki dannykh: ob"yektno-oriyentirovannyy podkhod i realizatsiya na S++ / A. A. Kubenskiy. SPb. : BVKH-Peterburg, 2004. 262 s.

6. Straustrup. B. Yazyik programmirovaniya C++. M.: «Binom», 2011. 1136 str.

ДОДАТОК А

ДІАГРАМА КЛАСІВ



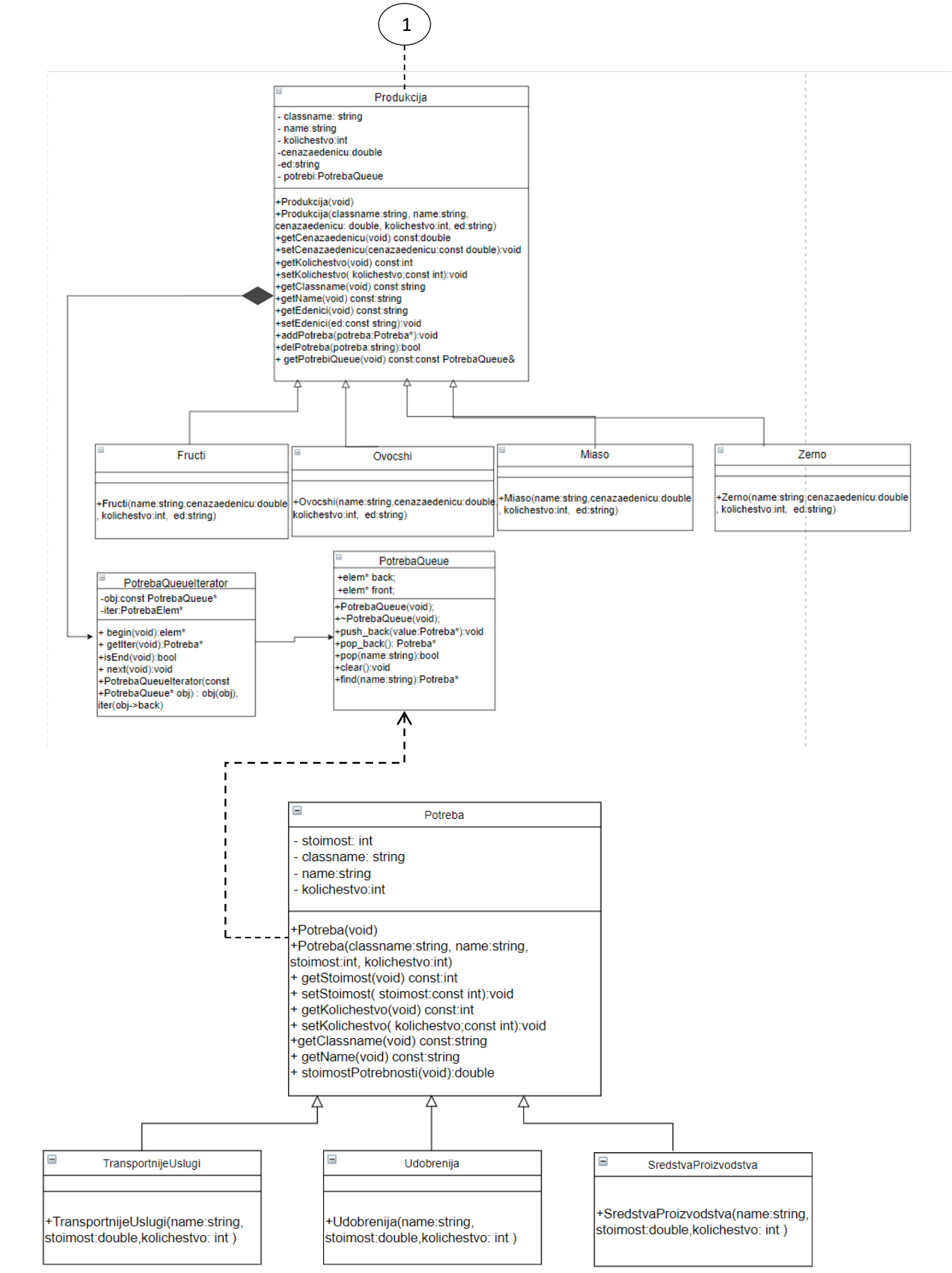


Рисунок А.1 – Діаграма класів програми

ДОДАТОК В

ТЕКСТ ПРОГРАМИ З КОМЕНТАРЯМИ

Зміст файлу "Fermer.h" програми.

```
#include "Produkciia.h"
#include "Potreba.h"
#include <vector>
#include <iostream>
#include <string>
#include <windows.h>
#include <msclr\marshal_cppstd.h
using namespace System;
using namespace System::ComponentModel;
using namespace System::Collections;
using namespace System::Windows::Forms;
using namespace System::Data;
using namespace System::Drawing;
using namespace std;
class Fermer;
struct elem
{Fermer* data;//поточні дані
        elem* prev;//попередній елемент
        elem* next;//наступний елемент};
class FermerQueue
{
private:
        elem* back;//початковий елемент
        elem* front;//останній елемент
public:
        FermerQueue(void);
        ~FermerQueue(void);
        void push_back(Fermer* value);//додавання в кінець черги
        Fermer* pop_back(void);//видалення з кінця черги
        bool pop(string name);//видалення заданого елемента черги
        void clear(void);// очищення
        Fermer* find(string name);//пошук за заданою назвою
        friend class FermerQueueIterator;};
class FermerQueueIterator
{FermerQueue* obj;
elem* iter;
pablic:
FermerQueueIterator( FermerQueue* obj) : obj(obj), iter(obj->back) {}
elem* begin(void)
{return obj->back;}
Fermer* getIter(void)
{return iter->data;}
bool isEnd(void)
```

```
44. {return iter == NULL;}
45. void next(void)
46. {iter = iter->next;};
47. //---------------------------------------------------------------------------------------
48. class Rabotnik
49. {protected:
50. string name;
51. public:
52. Rabotnik(string name);};
53. //-----------------------------------------------------------------------------------------------
54. class Fermer : public Rabotnik
55. {
56. private:
57. ProdukcijaQueue prod;// об’єкт класу - контейнеру що працює з продукцією
58. double budjet;//бюджет необхідний фермеру
59. public:
60. Fermer(void);
61. Fermer(string name, double budjet);
62. ~Fermer(void);
63. string getName(void) const;//метод для отримання доступу до імені
64. double getBudjet(void) const;//гетер для отримання даних про бюджет
65. void setBudjet(const double budjet);//сетер для присваювання значення бюджету
66. void addProdukcija(Produkcija *produkcija);//функція для додавання продукції
67. int delProdukcija(string produkcija, int kolichestvo);//функція для видалення продукції
68. Produkcija* findProdukcija(string produkcija);//метод для пошуку заданої продукції
69. ProdukcijaQueue& getProdukcijaQueue(void);//метод що повертає значення об’єкта з
    черги
70. void prodToComboBox(ComboBox^ comboBox);//метод що добавляє продукцію у
    випадаючий список};
71. //-----------------------------------------------------------------------------------------------
72. class Rukovoditel : public Rabotnik
73. {
74. private:
75. FermerQueue fermers;
76. public:
77. Rukovoditel(string name);
78. ~Rukovoditel(void);
79. void clearFermers(void);//видалення усіх даних про фермерів
80. void addFermer(Fermer* fermer);//додавання фермера
81. int delFermer(string name);//видалення фермера по імені
82. Fermer* findFermer(string name);//пошук фермера
83. FermerQueue& getFermerQueue(void);//метод повертає фермера з черги
84. void fermersToComboBox(ComboBox^ comboBox);//додавання фермера у випадаючий
    список};
```

Зміст файлу "Fermer.cpp" програми.

```
1. FermerQueue::FermerQueue(void) : back(NULL), front(NULL) {}
2. FermerQueue::~FermerQueue(void)
3. {
```

```
4.  this->clear();
5.  }
6.  void FermerQueue::push_back(Fermer* value)
7.  {
8.  elem* temp = new elem;
9.  temp->data = value;
10. temp->prev = NULL;
11. if (!this->back)
12. {
13. this->back = this->front = temp; temp->next = NULL;
14. }
15. else
16. {
17. temp->next = this->back;
18. this->back->prev = temp;
19. this->back = temp;
20. }
21. }
22. Fermer* FermerQueue::pop_back(void)
23. {
24. if (!this->back)
25. return NULL;
26. if (!this->back->next)
27. {
28. Fermer* out = this->back->data;
29. delete this->back;
30. this->back = this->front = NULL;
31. return out;
32. }
33. elem* del = this->back;
34. Fermer* out = this->back->data;
35. this->back = this->back->next;
36. delete del;
37. this->back->prev = NULL;
38. return out;}
39. bool FermerQueue::pop(string name)
40. {
41. elem* cursor = this->back;
42. while (cursor)
43. if (cursor->data->getName() == name)
44. {
45. if (!(this->back->next || this->back->prev))
46. this->back = NULL;
47. if (cursor->prev)
48. cursor->prev->next = cursor->next;
49. if (cursor->next)
50. cursor->next->prev = cursor->prev;
51. if (cursor == this->back)
52. this->back = cursor->next;
53. delete cursor->data;
54. delete cursor;
```

```
55. return true;
56. }
57. else
58. cursor = cursor->next;
59. return false;
60. }
61. void FermerQueue::clear(void)
62. {while (this->back) delete this->pop_back();}
63. Fermer* FermerQueue::find(string name)
64. {
65. elem* cursor = this->back;
66. while (cursor)
67. if (cursor->data->getName() == name)
68. return cursor->data;
69. else
70. cursor = cursor->next;
71. return NULL;
72. }
73. //-----------------------------------------------------------------------------------------
74. Rabotnik::Rabotnik(string name) : name(name) {}
75. //-----------------------------------------------------------------------------------------
76. Fermer::Fermer(void) : Rabotnik(""), budjet(0) {}
77. Fermer::Fermer(string name, double budjet) : Rabotnik(name), budjet(budjet) {}
78. Fermer::~Fermer(void)
79. {prod.clear();}
80. string Fermer::getName(void) const
81. {return name;}
82. double Fermer::getBudjet(void) const
83. {return budjet;}
84. void Fermer::setBudjet(const double budjet)
85. {this->budjet = budjet;}
86. void Fermer::addProdukcija(Produkcija *produkcija)
87. {prod.push_back(produkcija);}
88. int Fermer::delProdukcija(string produkcija, int kolichestvo)
89. {return prod.pop(produkcija, kolichestvo);}
90. Produkcija* Fermer::findProdukcija(string produkcija)
91. {return prod.find(produkcija);}
92. ProdukcijaQueue& Fermer::getProdukcijaQueue(void)
93. {return prod;}
94.  void Fermer::prodToComboBox(ComboBox^ comboBox)
95.  {
96.  comboBox->Items->Clear();
97.  for (ProdukcijaQueueIterator it(&getProdukcijaQueue()); !it.isEnd(); it.next())
98.  comboBox->Items->Add(gcnew String(it.getIter()->getName().c_str()));}
99.  //-----------------------------------------------------------------------------------------
100. Rukovoditel::Rukovoditel(string name) : Rabotnik(name) {}
101. Rukovoditel::~Rukovoditel(void)
102. {fermers.clear();}
103. void Rukovoditel::clearFermers(void)
104. {fermers.clear();}
105. void Rukovoditel::addFermer(Fermer* fermer)
```

```
106. {fermers.push_back(fermer);}
107. int Rukovoditel::delFermer(string name)
108. {return fermers.pop(name);}
109. Fermer* Rukovoditel::findFermer(string name)
110. {return fermers.find(name);}
111. FermerQueue& Rukovoditel::getFermerQueue(void)
112. {return fermers;}
113. void Rukovoditel::fermersToComboBox(ComboBox^ comboBox)
114. {comboBox->Items->Clear();
115. for (FermerQueueIterator it(&getFermerQueue()); !it.isEnd(); it.next())
comboBox->Items->Add(gcnew String(it.getIter()->getName().c_str()));}
```

Зміст файлу " Potreba.h" програми.

```
1.  #include <vector>
2.  #include <iostream>
3.  #include <string>
4.  #include <map>
5.  using namespace std;
6.
7.  class Potreba;
8.  struct PotrebaElem
9.  {
10. Potreba* data;//дані
11. PotrebaElem* prev;//попередній елемент
12. PotrebaElem* next;//наступний елемент
13. };
14. class PotrebaQueue
15. {
16. private:
17. PotrebaElem* back;//початковий елемент
18. PotrebaElem* front;//останній
19. public:
20. PotrebaQueue(void);
21. ~PotrebaQueue(void);

22. void push_back(Potreba* value);//додавання в кінець черги
23. Potreba* pop_back(void);//видалення з кінця черги
24. bool pop(string name);//видалення заданого елемента черги
25. void clear(void);// очищення
26. Potreba* find(string name);//пошук за заданою назвою
27. friend class PotrebaQueueIterator;};
28. class PotrebaQueueIterator
29. {PotrebaQueue* obj;
30. PotrebaElem* iter;
31. public:
32. PotrebaQueueIterator( PotrebaQueue* obj) : obj(obj), iter(obj->back) {}
33. PotrebaElem* begin(void)// повертає вказівник на початок черги
34. {return obj->back;}
35. Potreba* getIter(void)//поточні дані
```

```
36. {return iter->data;}
37. bool isEnd(void)//повертає елемент після останнього
38. {return iter == NULL;}
39. void next(void)//перехід до наступного
40. {iter = iter->next;}};
41. //---------------------------------------------------------------------------------------
42. class Potreba
43. {
44. protected:
45. int stoimost;
46. string classname;//тип потреби
47. string name;//назва потреби
48. int kolichestvo;//кількість
49. public:
50. Potreba(void);
51. Potreba(string classname, string name, int stoimost, int kolichestvo);
52. int getStoimost(void) const;// гетер для отримання даних про вартість потреби;
53. void setStoimost(const int stoimost);// сетер для присваювання значення вартості;
54. int getKolichestvo(void) const; //гетер для отримання даних про кількість потреби;
55. void setKolichestvo(const int kolichestvo);//сетер для присваювання значення про
    кількість потреби;
56. string getClassname(void) const;//гетер для отримання даних про тип потреби;
57. string getName(void) const;//для отримання доступу до імені
58. double stoimostPotrebnosti(void);//встановлення вартості потреби
59. };
60. //---------------------------------------------------------------------------------------
61. class Udobrenija : public Potreba
62. {
63. public:
64. Udobrenija(string name, double stoimost, int kolichestvo);
65. };
66. //---------------------------------------------------------------------------------------
67. class SredstvaProizvodstva : public Potreba
68. {
69. public:
70. SredstvaProizvodstva(string name, double stoimost, int kolichestvo);
71. };
72. //---------------------------------------------------------------------------------------
73. class TransportnijeUslugi : public Potreba
74. {
75. public:
76. TransportnijeUslugi(string name, double stoimost, int kolichestvo);
77. };
78. //---------------------------------------------------------------------------------------
79. Potreba* getPotrebaObj(string classname, string name, int kolichestvo, int
    stoimost);//функція для визначеня обраного похідного класу потреби
```

Зміст файлу " Potreba.cpp" програми.

```
1.  PotrebaQueue::PotrebaQueue(void) : back(NULL), front(NULL) {}
2.  PotrebaQueue::~PotrebaQueue(void)
3.  {this->clear();}
4.  void PotrebaQueue::push_back(Potreba* value)
5.  {PotrebaElem* temp = new PotrebaElem;
6.  temp->data = value;
7.  temp->prev = NULL;
8.  if (!this->back)//якщо не кінець
9.  {this->back = this->front = temp
10. temp->next = NULL}
11. else{
12. temp->next = this->back;
13. this->back->prev = temp;
14. this->back = temp;}
15. }
16. Potreba* PotrebaQueue::pop_back(void)
17. {
18. if (!this->back)
return NULL;
19. if (!this->back->next)//якщо  з одного елемента
1.  {
Potreba* out = this->back->data;
delete this->back;
this->back = this->front = NULL;
return out;
2.  }
3.  PotrebaElem* del = this->back;
4.  Potreba* out = this->back->data;
5.  this->back = this->back->next;
6.  delete del;
7.  this->back->prev = NULL;
8.  return out;
9.  }
10. bool PotrebaQueue::pop(string name)
11. {
12. PotrebaElem* cursor = this->back;
13. while (cursor)
if (cursor->data->getName() == name)
{
if (!(this->back->next || this->back->prev))
this->back = NULL;
if (cursor->prev)
cursor->prev->next = cursor->next;
if (cursor->next)
cursor->next->prev = cursor->prev;
if (cursor == this->back)
this->back = cursor->next;
delete cursor->data;
delete cursor;
return true;
}
```

```
else
cursor = cursor->next;
14. return false;
15. }
16. void PotrebaQueue::clear(void)
17. {while (this->back) delete this->pop_back();}
18. Potreba* PotrebaQueue::find(string name)
19. {PotrebaElem* cursor = this->back;
20. while (cursor)
if (cursor->data->getName() == name)
return cursor->data;
else
cursor = cursor->next;
21. return NULL;}
22. //---------------------------------------------------------------------------------------------
23. Potreba::Potreba(void) {
24. Potreba::Potreba(string classname, string name, int stoimost, int kolichestvo) :
25. classname(classname), name(name), stoimost(stoimost), kolichestvo(kolichestvo) {}
26. int Potreba::getStoimost(void) const
27. {return stoimost;}
28. void Potreba::setStoimost(const int stoimost)
29. {this->stoimost = stoimost;}
30. int Potreba::getKolichestvo(void) const
31. {return kolichestvo;}
32. void Potreba::setKolichestvo(const int kolichestvo)
33. {this->kolichestvo = kolichestvo;}
34. string Potreba::getName(void) const
35. {return name;}
36. string Potreba::getClassname(void) const
37. {return classname;}

38. double Potreba::stoimostPotrebnosti(void)
39. {return kolichestvo * stoimost;}
40. //---------------------------------------------------------------------------------------------
41. Udobrenija::Udobrenija(string name, double stoimost, int kolichestvo) :
42. Potreba("Udobrenija", name, stoimost, kolichestvo) {}
43. //---------------------------------------------------------------------------------------------
44. SredstvaProizvodstva::SredstvaProizvodstva(string name, double stoimost, int kolichestvo) :
45. Potreba("SredstvaPriizvodstva", name, stoimost, kolichestvo) {}
46. //---------------------------------------------------------------------------------------------
47. TransportnijeUslugi::TransportnijeUslugi(string name, double stoimost, int kolichestvo) :
48. Potreba("TransportnijeUslugi", name, stoimost, kolichestvo) {}
49. //---------------------------------------------------------------------------------------------
50. Potreba* getPotrebaObj(string classname, string name, int kolichestvo, int stoimost)
51. {
52. if (classname == "Udobrenija")
return new Udobrenija(name, stoimost, kolichestvo);
53. else if (classname == "SredstvaPriizvodstva")
return new SredstvaProizvodstva(name, stoimost, kolichestvo);
54. else if (classname == "TransportnijeUslugi")
return new TransportnijeUslugi(name, stoimost, kolichestvo);
```

```
55. else return NULL;}
```

Зміст файлу " Produkciia.h" програми.

```
1.  #include "Potreba.h"
2.  #include <vector>
3.  #include <iostream>
4.  #include <string>
5.  #include <map>
6.  using namespace std;
7.
8.  class Produkcija;
9.
10. struct ProdElem
11. {
12. Produkcija* data; // дані
13. ProdElem* prev;//попередній елемент
14. ProdElem* next;//наступний елемент};
15. class ProdukcijaQueue
16. {
17. private:
18. ProdElem* back;//початковий елемент
19. ProdElem* front;//останній
20. public:
21. ProdukcijaQueue(void);
22. ~ProdukcijaQueue(void);

23. void push_back(Produkcija* value);//додавання в кінець черги
24. Produkcija* pop_back(void);//видалення з кінця черги
25. int pop(string name, int kolichestvo);//видалення заданого елемента черги
26. void clear(void);// очищення
27. Produkcija* find(string name);//пошук за заданою назвою
28. friend class ProdukcijaQueueIterator;};
29. //---------------------------------------------------------------------------------------------
30. class ProdukcijaQueueIterator
31. {
32. private:
33. ProdukcijaQueue* obj;
34. ProdElem* iter;
35. public:
36. ProdukcijaQueueIterator( ProdukcijaQueue* obj) : obj(obj), iter(obj->back) {}
37. ProdElem* begin(void)
38. {return obj->back;}
39. Produkcija* getIter(void)
40. {return iter->data;}
41. bool isEnd(void)
42. {return iter == NULL;}
43. void next(void)
```

```
44. {iter = iter->next;}};
45. //-----------------------------------------------------------------------------------------
46. class Produkcija
47. {
48. protected:
49. string classname;//тип продукції
50. int kolichestvo;//кількість продукції
51. double cenazaedenicu;//ціна за одиницю
52. string ed;//одиниці вимірювання продукції
53. string name;//назва продукції
54. PotrebaQueue potrebi;
55. public:
56. Produkcija(void);
57. Produkcija(string name, string classname, double cenazaedenicu, int kolichestvo, string ed);
58. ~Produkcija(void);
59. virtual string getName(void) const;// гетер для отримання даних про назву продукції;
60. virtual string getClassName(void) const;//гетер для отримання даних про тип продукції;
61. virtual double getCenazaedenicu(void) const;//гетер для отримання даних про ціну
    продукції;
62. virtual void setCenazaedenicu(const double cenazaedenicu);//сетер для присваювання
    значення ціна за одиницю продукції;
63. virtual int getKolichestvo(void) const;//гетер для отримання даних про кількість;
64. virtual void setKolichestvo(const int kolichestvo);//) сетер для присваювання кількості
    продукції;
65. virtual string getEdenici(void) const;//гетер для отримання даних про одиниці
    вимірювання кількості продукції;
66. virtual void setEdenici(const string ed);//сетер для присваювання даних про одиниці
    вимірювання кількості продукції;
67. void addPotreba(Potreba* potreba);//функція для додавання потреби, щоб виготовити
    задану продукцію;
68. bool delPotreba(string potreba);//видалення потреби
69. PotrebaQueue& getPotrebiQueue(void) ;//гетер для отримання даних про потребу;};
70. //-----------------------------------------------------------------------------------------
71. class Zerno : public Produkcija
72. {
73. public:
74. Zerno(string name, double cenazaedenicu, int kolichestvo, string ed);
75. };
76. //-----------------------------------------------------------------------------------------
77. class Miaso : public Produkcija
78. {
79. public:
80. Miaso(string name, double cenazaedenicu, int kolichestvo, string ed);
81. };
82. //-----------------------------------------------------------------------------------------
83. class Fructi : public Produkcija
84. {
85. public:
86. Fructi(string name,double cenazaedenicu, int kolichestvo, string ed);
87. };
88. //-----------------------------------------------------------------------------------------
```

```
89. class Ovocshi : public Produkcija
90. {
91. public:
92. Ovocshi(string name, double cenazaedenicu, int kolichestvo, string ed);
93. };
94. //---------------------------------------------------------------------------------------
95. Produkcija* getProdukcijaObj(string name, string classname, double cenazaedenicu, int
    kolichestvo, string ed);//функція для визначеня обраного похідного класу продукції
```

Зміст файлу " Produkciia.cpp" програми.

```
1. #include "Produkciia.h"

2. ProdukcijaQueue::ProdukcijaQueue(void) : back(NULL), front(NULL) {}

3. ProdukcijaQueue::~ProdukcijaQueue(void)
4. {this->clear();}
5. void ProdukcijaQueue::push_back(Produkcija* value)
6. {ProdElem* temp = new ProdElem;
7. temp->data = value;
8. temp->prev = NULL;
9. if (!this->back)
10. {this->back = this->front = temp; temp->next = NULL;
11. }else
12. {
13. temp->next = this->back;
14. this->back->prev = temp;
15. this->back = temp;
16. }}
17. Produkcija* ProdukcijaQueue::pop_back(void)
18. {
19. if (!this->back)
20. return NULL;
21. if (!this->back->next)
22. {
23. Produkcija* out = this->back->data;
24. delete this->back;
25. this->back = this->front = NULL;
26. return out;
27. }
28. ProdElem* del = this->back;
29. Produkcija* out = this->back->data;
30. this->back = this->back->next;
31. delete del;
32. this->back->prev = NULL;
33. return out;
34. }
35. int ProdukcijaQueue::pop(string name, int kolichestvo)
36. {
```

```
37. ProdElem* cursor = this->back;
38. while (cursor)
39. if (cursor->data->getName() == name)
40. {
41. if (cursor->data->getKolichestvo() < kolichestvo)
42. return -1;
43. if (cursor->data->getKolichestvo() > kolichestvo)
44. cursor->data->setKolichestvo(cursor->data->getKolichestvo() - kolichestvo);
45. else
46. {
47. if (!(this->back->next || this->back->prev))
48. this->back = NULL;
49. if (cursor->prev)
50. cursor->prev->next = cursor->next;
51. if (cursor->next)
52. cursor->next->prev = cursor->prev;
53. if (cursor == this->back)
54. this->back = cursor->next;
55. delete cursor->data;
56. delete cursor;
57. return true;
58. }
59. return 1;
60. }
61. else
62. cursor = cursor->next;
63. return 0;
64. }
65. void ProdukcijaQueue::clear(void)
66. {
67. while (this->back) delete this->pop_back();
68. }
69. Produkcija* ProdukcijaQueue::find(string name)
70. {
71. ProdElem* cursor = this->back;
72. while (cursor)
73. if (cursor->data->getName() == name)
74. return cursor->data;
75. else
76. cursor = cursor->next;
77. return NULL;
78. }
79. //---------------------------------------------------------------------------------------------
80. Produkcija::Produkcija(void) {}
81. Produkcija::Produkcija(string name, string classname, double cenazaedenicu, int
    kolichestvo, string ed) :
82. name(name), classname(classname), cenazaedenicu(cenazaedenicu),
    kolichestvo(kolichestvo), ed(ed) {}
83. Produkcija::~Produkcija(void)
84. {potrebi.clear();}
85. string Produkcija::getClassName(void) const
```

```
{return classname;}
string Produkcija::getName(void) const
{return name;}
double Produkcija::getCenazaedenicu(void) const
{return cenazaedenicu;}
void Produkcija::setCenazaedenicu(const double cenazaedenicu)
{this->cenazaedenicu = cenazaedenicu;}
int Produkcija::getKolichestvo(void) const
{return kolichestvo;}
void Produkcija::setKolichestvo(const int kolichestvo)
{this->kolichestvo = kolichestvo;}
string Produkcija::getEdenici(void) const
{return ed;}
void Produkcija::setEdenici(const string ed)
{this->ed = ed;
}
void Produkcija::addPotreba(Potreba* potreba)
{
potrebi.push_back(potreba);
}
bool Produkcija::delPotreba(string potreba)
{return potrebi.pop(potreba);}
PotrebaQueue& Produkcija::getPotrebiQueue(void)
{return potrebi;}
//-------------------------------------------------------------------------------------------
Zerno::Zerno(string name, double cenazaedenicu, int kolichestvo, string ed) :
Produkcija(name, "Zerno", cenazaedenicu, kolichestvo, ed)
{}
//-------------------------------------------------------------------------------------------
Miaso::Miaso(string name, double cenazaedenicu, int kolichestvo, string ed) :
Produkcija(name, "Miaso", cenazaedenicu, kolichestvo, ed)
{}
//-------------------------------------------------------------------------------------------
Fructi::Fructi(string name, double cenazaedenicu, int kolichestvo, string ed) :
Produkcija(name, "Fructi", cenazaedenicu, kolichestvo, ed)
{}
//-------------------------------------------------------------------------------------------
Ovocshi::Ovocshi(string name, double cenazaedenicu, int kolichestvo, string ed) :
Produkcija(name, "Ovocshi", cenazaedenicu, kolichestvo, ed)
{}
//-------------------------------------------------------------------------------------------
Produkcija* getProdukcijaObj(string name, string classname, double cenazaedenicu, int
    kolichestvo, string ed)
{
if (classname == "Zerno")
return new Zerno(name, cenazaedenicu, kolichestvo, ed);
else if (classname == "Miaso")
return new Miaso(name, cenazaedenicu, kolichestvo, ed);
else if (classname == "Fructi")
return new Fructi(name, cenazaedenicu, kolichestvo, ed);
else if (classname == "Ovocshi")
```

```
136.return new Ovocshi(name, cenazaedenicu, kolichestvo, ed);
137.else
138.return NULL;
139.}
```

Зміст файлу " Error.h" програми.

```
1.  class Error
2.  {protected:
3.  const char* text;
4.  public:
5.  Error(const char* text);
6.  virtual const char* message(void) const;};

7.  class ProdukcijaError : public Error
8.  {public:
9.  ProdukcijaError(const char* text);};
10.
11. class FermerError : public Error
12. {public:
13. FermerError(const char* text);};
14.
15. class PotrebaError : public Error
16. {public:
17. PotrebaError(const char* text);}
18.
19. class FileError : public Error
20. {public:
21. FileError(const char* text);};
```

Зміст файлу " Error.cpp" програми.

```
1.  #include "Error.h"
2.  Error::Error(const char* text) : text(text) {}
3.
4.  const char* Error::message(void) const
5.  {return text;}
6.
7.  ProdukcijaError::ProdukcijaError(const char* text) : Error(text) {}
8.  FermerError::FermerError(const char* text) : Error(text) {}
9.  PotrebaError::PotrebaError(const char* text) : Error(text) {}
10. FileError::FileError(const char* text) : Error(text) {}
```

Зміст файлу "Fermerstvo.h" програми.

```
1.  #include <windows.h>
2.  #include <msclr\marshal_cppstd.h>
3.  #include <sstream>
```

```
#include <fstream>
#include <string>
#include "Fermer.h"
#include "Error.h"
#pragma endregion
private: System::Void textBox1_TextChanged(System::Object^ sender,
System::EventArgs^e) {}
private: System::Void вихідToolStripMenuItem_Click(System::Object^ sender,
System::EventArgs^ e) {}
// ----------додавання нового фермера----------
private: System::Void button1_Click(System::Object^ sender, System::EventArgs^ e)
{   try
{string name = msclr::interop::marshal_as<std::string>(textBox1->Text);
int budjet = System::Convert::ToDouble(textBox2->Text);
fermerstvo->addFermer(new Fermer(name, budjet));
fermerstvo->fermersToComboBox(comboBox3);
fermerstvo->fermersToComboBox(comboBox4);
MessageBox::Show("Нового фермера успішно додано.", ":)");}
catch (...) {
MessageBox::Show("Введено некоректні дані!", "Помилка!");}}
// ----------додавання нової продукції----------
private: System::Void button2_Click(System::Object^ sender, System::EventArgs^ e){
try{
string fermer = msclr::interop::marshal_as<std::string>(comboBox4->Text);
string name = msclr::interop::marshal_as<std::string>(textBox6->Text);
string vid = msclr::interop::marshal_as<std::string>(comboBox1->Text);
if (vid == "М'ясо") vid = "Miaso";
else if (vid == "Зерно") vid = "Zerno";
else if (vid == "Овочі") vid = "Ovocshi";
else if (vid == "Фрукти") vid = "Fructi";
int kolichestvo = System::Convert::ToInt16(textBox4->Text);
int cena = System::Convert::ToInt16(textBox7->Text);
string edenici = msclr::interop::marshal_as<std::string>(textBox8->Text);
Produkcija* elem = getProdukcijaObj(name, vid, cena, kolichestvo, edenici);
if (!elem)
```

```
28. throw ProdukcijaError("Такого виду продукції не існує");
29. fermerstvo->findFermer(fermer)->addProdukcija(elem);
30. MessageBox::Show("Продукцію успішно додано.", ":)");}
31. catch (ProdukcijaError error) {
32. MessageBox::Show(gcnew String(error.message()), "Помилка!");}
33. catch (...) {
34. MessageBox::Show("Введено некоректні дані!", "Помилка!");}}
35. private: System::Void button6_Click(System::Object^ sender, System::EventArgs^ e)
36. {}
37. private: System::Void button3_Click(System::Object^ sender, System::EventArgs^ e)
38. {}
39. private: System::Void button4_Click(System::Object^ sender, System::EventArgs^ e)
40. {}
41. // ----------------------------видалення продукції-----------------------------------
42. private: System::Void button5_Click(System::Object^ sender, System::EventArgs^ e)
43. {try
44. {
45. string fermer = msclr::interop::marshal_as<std::string>(comboBox4->Text);
46. string nazvanie = msclr::interop::marshal_as<std::string>(textBox6->Text);
47. int kolichestvo = System::Convert::ToInt16(textBox4->Text);
48. int result = fermerstvo->findFermer(fermer)->delProdukcija(nazvanie, kolichestvo);
49. if(!result)
50. throw ProdukcijaError("Такого виду продукції не існує");
51. else if(result == -1)
52. throw ProdukcijaError("Недостатньо кількості продукції на складі");
53. MessageBox::Show("Продукцію успішно відпущено.", ":)");}
54. catch (ProdukcijaError error) {
55. MessageBox::Show(gcnew String(error.message()), "Помилка!");}
56. catch (...) {
57. MessageBox::Show("Введено некоректні дані!", "Помилка!");}}
58. private: System::Void вихідToolStripMenuItem1_Click(System::Object^ sender,
    System::EventArgs^ e){
59. exit(0);}
60. private: System::Void button8_Click(System::Object^ sender, System::EventArgs^ e)
```

```
61. {}
62. private: System::Void button7_Click(System::Object^ sender, System::EventArgs^ e)
63. {}
64. //----------------------------------------запис у файл----------------------------------------
65. private: System::Void інструкціяToolStripMenuItem_Click(System::Object^ sender,
    System::EventArgs^ e)
66. {try
67. {saveFileDialog1->Filter = "Text files(*.txt)|*.txt|All files(*.*)|*.*";
68. saveFileDialog1->ShowDialog();
69. string filename = msclr::interop::marshal_as<std::string>(saveFileDialog1->FileName);
70. ofstream fout(filename, ios_base::out);
71. if (!fout.is_open())
72. throw FileError("Помилка відкриття файлу.");
73. fout << "Fermers\n\n";
74. for (FermerQueueIterator it(&fermerstvo->getFermerQueue()); !it.isEnd();)
75. {
76. fout << "Name: " << it.getIter()->getName() << "\n";
77. fout << "Budjet: " << it.getIter()->getBudjet();it.next();
78. fout << "\n";
79. if (!it.isEnd())
80. fout << "---";
81. fout << "\n";
82. }
83. fout << "Production\n\n";
84. for (FermerQueueIterator it(&fermerstvo->getFermerQueue()); !it.isEnd();)
85. {bool flag = false;
86. for (ProdukcijaQueueIterator it2(&it.getIter()->getProdukcijaQueue()); !it2.isEnd();)
87. {flag = true;
88. fout << "Fermer: " << it.getIter()->getName() << "\n";
89. fout << "Nazva: " << it2.getIter()->getName() << "\n";
90. fout << "Vid: " << it2.getIter()->getClassName() << "\n";
91. fout << "Kolichestvo: " << it2.getIter()->getKolichestvo() << "\n";
92. fout << "CenaZaEdenicu: " << it2.getIter()->getCenazaedenicu() << "\n";
93. fout << "Odynyci: " << it2.getIter()->getEdenici();
```

```
94. it2.next();
95. fout << "\n";
96. if (!it2.isEnd())
97. fout << "---";}
98. it.next();
99. if(!it.isEnd() && flag)
100. fout << "---";
101. if(flag)
102. fout << "\n";}
103. fout << "Potrebi\n\n";
104. for (FermerQueueIterator it(&fermerstvo->getFermerQueue()); !it.isEnd();)
105. {
106. bool flag = false;
107. for (ProdukcijaQueueIterator it2(&it.getIter()->getProdukcijaQueue()); !it2.isEnd();)
108. {for (PotrebaQueueIterator it3(&it2.getIter()->getPotrebiQueue()); !it3.isEnd();{
109. flag = true;
110. fout << "Fermer: " << it.getIter()->getName() << "\n";
111. fout << "Production: " << it2.getIter()->getName() << "\n";
112. fout << "Type: " << it3.getIter()->getClassname() << "\n";
113. fout << "Name: " << it3.getIter()->getName() << "\n";
114. fout << "Kolichestvo: " << it3.getIter()->getKolichestvo() << "\n";
115. fout << "Cena: " << it3.getIter()->getStoimost();
116. it3.next();
117. fout << "\n";
118. if (!it3.isEnd())
119. fout << "---";}
120. it2.next();
121. if (!it2.isEnd())
122. fout << "---\n";}
123. it.next();
124. if (!it.isEnd() && flag)
125. fout << "---\n";
126. else if (flag)
127. fout << "\n";}
```

```
128. MessageBox::Show("Дані успішно збережено у файл.", ":)");}
129. catch (FileError error) {
130. MessageBox::Show(gcnew String(error.message()), "Помилка!");}
131. catch (...) {
132. MessageBox::Show("Помилка обробки файлу!", "Помилка!");}}
133.//-----------------------------------зчитування з файлу----------------------------------------
     private: System::Void проПрограмуToolStripMenuItem_Click(System::Object^ sender,
     System::EventArgs^ e){
134.try{openFileDialog1->Filter = "Text files(*.txt)|*.txt|All files(*.*)|*.*";
135.openFileDialog1->ShowDialog();
136.string filename = msclr::interop::marshal_as<std::string>(openFileDialog1->FileName);
137.ifstream finp(filename, ios_base::in);
138.if (!finp.is_open())
139.throw FileError("Помилка відкриття файлу.");
140.string what;
141.finp >> what;
142.if (what == "Fermers"){
143.do {string name;
144.int budjet;
145.//читаємо з файла дані про  фермера
146.finp >> what;
147.finp >> what;
148.name += what + " ";
149.finp >> what;
150.name += what + " ";
151.finp >> what;
152.name += what;
153.finp >> what;
154.finp >> budjet;
155.fermerstvo->addFermer(new Fermer(name, budjet));
156.finp >> what;
157.} while (what == "---" && !finp.eof());
158.}else
159.throw FileError("Некоректні дані у файлі.");
```

```
160.if (what == "Production"){
161.do {//зчитуємо дані про продукцію
162.string fermer;
163.string name;
164.string vid;
165.int kolichestvo;
166.int cena;
167.string edenici;
168.finp >> what;
169.finp >> what;
170.fermer += what + " ";
171.finp >> what;
172.fermer += what + " ";
173.finp >> what;
174.fermer += what;
175.finp >> what;
176.finp >> name;
177.finp >> what;
178.finp >> vid;
179.finp >> what;
180.finp >> kolichestvo;
181.finp >> what;
182.finp >> cena;
183.finp >> what;
184.finp >> edenici;
185.Produkcija* elem = getProdukcijaObj(name, vid, cena, kolichestvo, edenici);
186.if (!elem)
187.throw ProdukcijaError("Некоректні дані у файлі.");
188.fermerstvo->findFermer(fermer)->addProdukcija(elem);
189.finp >> what;
190.} while (what == "---" && !finp.eof());
191.}
192.else
193.throw FileError("Некоректні дані у файлі.");
```

```
194.if (what == "Potrebi")
195.{
196.do {//зчитуємо дані про потреби
197.string fermer;
198.string producciia;
199.string vid;
200.string name;
201.int kolichestvo;
202.int cena;
203.finp >> what;
204.finp >> what;
205.fermer += what + " ";
206.finp >> what;
207.fermer += what + " ";
208.finp >> what;
209.fermer += what;
210.finp >> what;
211.finp >> producciia;
212.finp >> what;
213.finp >> vid;
214.finp >> what;
215.finp >> name;
216.finp >> what;
217.finp >> kolichestvo;
218.finp >> what;
219.finp >> cena;
220.Potreba* elem = getPotrebaObj(vid, name, kolichestvo, cena);
221.if (!elem)
222.throw PotrebaError("Некоректні дані у файлі.");
223.fermerstvo->findFermer(fermer)->findProdukcija(producciia)->addPotreba(elem);
224.finp >> what;
225.} while (what == "---" && !finp.eof());
226.}
227.else
```

```
228.throw FileError("Некоректні дані у файлі.");
229.fermerstvo->fermersToComboBox(comboBox3);//добавляєть фермер у випадаючі
    списки
230.fermerstvo->fermersToComboBox(comboBox4);
231.MessageBox::Show("Дані з файлу успішно завантажено в програму.", ":)");}
232.catch (FileError error) {
233.MessageBox::Show(gcnew String(error.message()), "Помилка!");}
234.catch (...) {
235.MessageBox::Show("Помилка обробки файлу!", "Помилка!");}}
236.private: System::Void menuStrip1_ItemClicked(System::Object^ sender,
    System::Windows::Forms::ToolStripItemClickedEventArgs^ e) {
237.}
238.//-------------------------------------------------видалення фермера---------------------------------
239.private: System::Void button14_Click(System::Object^ sender, System::EventArgs^ e)
240.{try
241.{
242.string name = msclr::interop::marshal_as<std::string>(textBox1->Text);
243.if(!fermerstvo->delFermer(name))
244.throw FermerError("Такого фермера немає. Видалення неможливе!");
245.fermerstvo->fermersToComboBox(comboBox3);
246.fermerstvo->fermersToComboBox(comboBox4);
247.MessageBox::Show("Вибраного фермера успішно видалено.", ":)");
248.}
249.catch (FermerError error) {
250.MessageBox::Show(gcnew String(error.message()), "Помилка!");
251.}
252.catch (...) {
253.MessageBox::Show("Введено некоректні дані!", "Помилка!");}}
254.//-----------------------------------------------------додавання потреби-----------------------------
255.private: System::Void button4_Click_1(System::Object^ sender, System::EventArgs^ e)
256.{
257.try{
258.string fermer = msclr::interop::marshal_as<std::string>(comboBox3->Text);
259.string producciia = msclr::interop::marshal_as<std::string>(comboBox5->Text);
```

```
260.string vid = msclr::interop::marshal_as<std::string>(comboBox2->Text);
261.if(vid == "Добрива") vid = "Udobrenija";
262.else if (vid == "Засоби виробництва") vid = "SredstvaPriizvodstva";
263.else if (vid == "Транспортні послуги") vid = "TransportnijeUslugi";
264.string name = msclr::interop::marshal_as<std::string>(textBox15->Text);
265.int kolichestvo = System::Convert::ToInt16(textBox9->Text);
266.int cena = System::Convert::ToInt16(textBox13->Text);
267.Potreba* elem = getPotrebaObj(vid, name, kolichestvo, cena);
268.if (!elem)
269.throw PotrebaError("Такого виду потреб не існує");
270.fermerstvo->findFermer(fermer)->findProdukcija(producciia)->addPotreba(elem);
271.MessageBox::Show("Потребу успішно додано.", ":)");}
272.catch (PotrebaError error) {
273.MessageBox::Show(gcnew String(error.message()), "Помилка!");
274.}catch (...) {
275.MessageBox::Show("Введено некоректні дані!", "Помилка!");}}
276.//-----------------------------------видалення потреби-----------------------------------------
277.private: System::Void button15_Click(System::Object^ sender, System::EventArgs^ e)
278.{try
279.string fermer = msclr::interop::marshal_as<std::string>(comboBox3->Text);
280.string producciia = msclr::interop::marshal_as<std::string>(comboBox5->Text);
281.string name = msclr::interop::marshal_as<std::string>(textBox15->Text);
282.if (!fermerstvo->findFermer(fermer)->findProdukcija(producciia)->delPotreba(name))
283.throw PotrebaError("Такої потреби для вибраної продукції не існує.");
284.MessageBox::Show("Потребу успішно видалено.", ":)");
285.}catch (PotrebaError error) {
286.MessageBox::Show(gcnew String(error.message()), "Помилка!");}
287.catch (...) {
288.MessageBox::Show("Введено некоректні дані!", "Помилка!");}}
289.//-------------------------------------------------довідка про програму-------------------------
290.private: System::Void переглядДовідкиToolStripMenuItem_Click(System::Object^ sender,
      System::EventArgs^ e){
291.MessageBox::Show("Програмний застосунок для керування фермерським
      господарством.", "Про програму");}
```

```
292. private: System::Void comboBox3_SelectedIndexChanged(System::Object^ sender,
     System::EventArgs^ e)
293. {fermerstvo->findFermer(msclr::interop::marshal_as<std::string>(comboBox3->Text))-
     >prodToComboBox(comboBox5);}
294. //--------------створенн таблиці: яку продукцію виробляють фермери області ----------
295. private: System::Void button7_Click_1(System::Object^ sender, System::EventArgs^ e)
296. {dataGridView1->Columns->Clear();
297. DataGridViewTextBoxColumn ^numser;
298. numser = gcnew DataGridViewTextBoxColumn();
299. numser->Name = "number";
300. numser->HeaderText = "№";
301. numser->Width = 30;
302. dataGridView1->Columns->Add(numser);
303. DataGridViewTextBoxColumn^ fermer;
304. fermer = gcnew DataGridViewTextBoxColumn();
305. fermer->Name = "fermer";
306. fermer->HeaderText = "Фермер";
307. fermer->Width = 175;
308. dataGridView1->Columns->Add(fermer);
309. DataGridViewTextBoxColumn^ prod;
310. prod = gcnew DataGridViewTextBoxColumn();
311. prod->Name = "prod";
312. prod->HeaderText = "Продукція";
313. prod->Width = 175;
314. dataGridView1->Columns->Add(prod);
315. int i = 0;
316. for (FermerQueueIterator it(&fermerstvo->getFermerQueue()); !it.isEnd(); it.next())
317. {dataGridView1->Rows->Add();
318. dataGridView1->Rows[i]->Cells[0]->Value = (i + 1).ToString();
319. dataGridView1->Rows[i]->Cells[1]->Value = gcnew String(it.getIter()->getName().c_str());
320. ProdukcijaQueueIterator it2(&it.getIter()->getProdukcijaQueue());
321. if (it2.isEnd())
322. dataGridView1->Rows[i]->Cells[2]->Value += gcnew String("(не виробляє нічого)");
323. else for (; !it2.isEnd(); it2.next())
```

```
dataGridView1->Rows[i]->Cells[2]->Value += gcnew String(it2.getIter()->getName().c_str()) + " ";
i++;}}
//----------------------------–Що потрібно кожному фермеру для виробництва ----------
private: System::Void button6_Click_1(System::Object^ sender, System::EventArgs^ e)
{dataGridView1->Columns->Clear();
DataGridViewTextBoxColumn^ numser;
numser = gcnew DataGridViewTextBoxColumn();
numser->Name = "number";
numser->HeaderText = "№";
numser->Width = 30;
dataGridView1->Columns->Add(numser);
DataGridViewTextBoxColumn^ fermer;
fermer = gcnew DataGridViewTextBoxColumn();
fermer->Name = "fermer";
fermer->HeaderText = "Фермер";
fermer->Width = 175;
dataGridView1->Columns->Add(fermer);
DataGridViewTextBoxColumn^ potrebi;
potrebi = gcnew DataGridViewTextBoxColumn();
potrebi->Name = "potrebi";
potrebi->HeaderText = "Потреби";
potrebi->Width = 175;
dataGridView1->Columns->Add(potrebi);
int i = 0;
for (FermerQueueIterator it(&fermerstvo->getFermerQueue()); !it.isEnd(); it.next())
{bool flag = false;
dataGridView1->Rows->Add();
dataGridView1->Rows[i]->Cells[0]->Value = (i + 1).ToString();
dataGridView1->Rows[i]->Cells[1]->Value = gcnew String(it.getIter()->getName().c_str());
for (ProdukcijaQueueIterator it2(&it.getIter()->getProdukcijaQueue()); !it2.isEnd(); it2.next())
for (PotrebaQueueIterator it3(&it2.getIter()->getPotrebiQueue()); !it3.isEnd(); it3.next())
{flag = true;
```

```
356.dataGridView1->Rows[i]->Cells[2]->Value += gcnew String(it3.getIter()
357.->getName().c_str())
358.+ ": " + System::Convert::ToString(it3.getIter()->getKolichestvo()) + " ";}
359.if (!flag)
360.dataGridView1->Rows[i]->Cells[2]->Value += gcnew String("(нічого не потрібно)");
361.i++;}}
362.//------------------створення таблиці: який кредит потрібен фермерам----------------------
363.private: System::Void button11_Click(System::Object^ sender, System::EventArgs^ e)
364.{dataGridView1->Columns->Clear();
365.DataGridViewTextBoxColumn^ numser;
366.numser = gcnew DataGridViewTextBoxColumn();
367.numser->Name = "number";
368.numser->HeaderText = "№";
369.numser->Width = 30;
370.dataGridView1->Columns->Add(numser);
371.DataGridViewTextBoxColumn^ fermer;
372.fermer = gcnew DataGridViewTextBoxColumn();
373.fermer->Name = "fermer";
374.fermer->HeaderText = "Фермер";
375.fermer->Width = 175;
376.dataGridView1->Columns->Add(fermer);
377.DataGridViewTextBoxColumn^ kredit;
378.kredit = gcnew DataGridViewTextBoxColumn();
379.kredit->Name = "potrebi";
380.kredit->HeaderText = "Необхідний кредит";
381.kredit->Width = 175;
382.dataGridView1->Columns->Add(kredit);
383.int i = 0;
384.for (FermerQueueIterator it(&fermerstvo->getFermerQueue()); !it.isEnd(); it.next())
385.{
386.dataGridView1->Rows->Add();
387.dataGridView1->Rows[i]->Cells[0]->Value = (i + 1).ToString();
388.dataGridView1->Rows[i]->Cells[1]->Value = gcnew String(it.getIter()->getName().c_str());
389.int kredit = 0;
```

```
390. for (ProdukcijaQueueIterator it2(&it.getIter()->getProdukcijaQueue()); !it2.isEnd();
     it2.next())
391. for (PotrebaQueueIterator it3(&it2.getIter()->getPotrebiQueue()); !it3.isEnd(); it3.next())
392. kredit += it3.getIter()->getStoimost();
393. if (!kredit)
394. dataGridView1->Rows[i]->Cells[2]->Value += gcnew String("(кредит не потрібен)");
395. else
396. dataGridView1->Rows[i]->Cells[2]->Value += "потрібен кредит "
397. + System::Convert::ToString(kredit) + " гривень";
398. i++;}}
399. //---------створенн таблиці: прибуток фермерів по кожному виду продукції----------
400. private: System::Void button10_Click(System::Object^ sender, System::EventArgs^ e)
401. {dataGridView1->Columns->Clear();
402. DataGridViewTextBoxColumn^ numser;
403. numser = gcnew DataGridViewTextBoxColumn();
404. numser->Name = "number";
405. numser->HeaderText = "№";
406. numser->Width = 30;
407. dataGridView1->Columns->Add(numser);
408. DataGridViewTextBoxColumn^ prod;
409. prod = gcnew DataGridViewTextBoxColumn();
410. prod->Name = "prod";
411. prod->HeaderText = "Продукція";
412. prod->Width = 175;
413. dataGridView1->Columns->Add(prod);
414. DataGridViewTextBoxColumn^ kilkist;
415. kilkist = gcnew DataGridViewTextBoxColumn();
416. kilkist->Name = "kilkist";
417. kilkist->HeaderText = "Загальний прибуток фермерів";
418. kilkist->Width = 175;
419. dataGridView1->Columns->Add(kilkist);
420. int pribilMiaso = 0;
421. int pribilZerno = 0;
422. int pribilFructi = 0;
```

```
int pribilOvocshi = 0;
dataGridView1->Rows->Add();
dataGridView1->Rows[0]->Cells[0]->Value = (1).ToString();
dataGridView1->Rows[0]->Cells[1]->Value = gcnew String("М'ясо");
dataGridView1->Rows->Add();
dataGridView1->Rows[1]->Cells[0]->Value = (2).ToString();
dataGridView1->Rows[1]->Cells[1]->Value = gcnew String("Зерно");
dataGridView1->Rows->Add();
dataGridView1->Rows[2]->Cells[0]->Value = (3).ToString();
dataGridView1->Rows[2]->Cells[1]->Value = gcnew String("Фрукти");
dataGridView1->Rows->Add();
dataGridView1->Rows[3]->Cells[0]->Value = (4).ToString();
dataGridView1->Rows[3]->Cells[1]->Value = gcnew String("Овочі");
for (FermerQueueIterator it(&fermerstvo->getFermerQueue()); !it.isEnd(); it.next())
for (ProdukcijaQueueIterator it2(&it.getIter()->getProdukcijaQueue()); !it2.isEnd();
    it2.next())
if(it2.getIter()->getClassName() == "Miaso")
pribilMiaso += it2.getIter()->getKolichestvo() * it2.getIter()->getCenazaedenicu();
else if (it2.getIter()->getClassName() == "Zerno")
pribilZerno += it2.getIter()->getKolichestvo() * it2.getIter()->getCenazaedenicu();
else if (it2.getIter()->getClassName() == "Fructi")
pribilFructi += it2.getIter()->getKolichestvo() * it2.getIter()->getCenazaedenicu();
else if (it2.getIter()->getClassName() == "Ovocshi")
pribilOvocshi += it2.getIter()->getKolichestvo() * it2.getIter()->getCenazaedenicu();
dataGridView1->Rows[0]->Cells[2]->Value = pribilMiaso + " гривень";
dataGridView1->Rows[1]->Cells[2]->Value = pribilZerno + " гривень";
dataGridView1->Rows[2]->Cells[2]->Value = pribilFructi + " гривень";
dataGridView1->Rows[3]->Cells[2]->Value = pribilOvocshi + " гривень";
}
//-----------створення таблиці: яку кількість заданої продукції виробляють----------
private: System::Void button8_Click_1(System::Object^ sender, System::EventArgs^ e)
{try
{if (textBox12->Text == "")
throw ProdukcijaError("Не вказано назву продукції.");
```

```
dataGridView1->Columns->Clear();
DataGridViewTextBoxColumn^ numser;
numser = gcnew DataGridViewTextBoxColumn();
numser->Name = "number";
numser->HeaderText = "№";
numser->Width = 30;
dataGridView1->Columns->Add(numser);
DataGridViewTextBoxColumn^ fermer;
fermer = gcnew DataGridViewTextBoxColumn();
fermer->Name = "fermer";
fermer->HeaderText = "Фермер";
fermer->Width = 175;
dataGridView1->Columns->Add(fermer);
DataGridViewTextBoxColumn^ kilkist;
kilkist = gcnew DataGridViewTextBoxColumn();
kilkist->Name = "kilkist";
kilkist->HeaderText = "Кількість продукції " + textBox12->Text;
kilkist->Width = 175;
dataGridView1->Columns->Add(kilkist);
nt i = 0;
for (FermerQueueIterator it(&fermerstvo->getFermerQueue()); !it.isEnd(); it.next())
{
dataGridView1->Rows->Add();
dataGridView1->Rows[i]->Cells[0]->Value = (i + 1).ToString();
dataGridView1->Rows[i]->Cells[1]->Value = gcnew String(it.getIter()->getName().c_str());
bool flag = false;
for (ProdukcijaQueueIterator it2(&it.getIter()->getProdukcijaQueue()); !it2.isEnd();
    it2.next())
if (gcnew String(it2.getIter()->getName().c_str()) == textBox12->Text)
{
dataGridView1->Rows[i]->Cells[2]->Value += gcnew
    String(System::Convert::ToString(it2.getIter()->getKolichestvo()))
+ " " + gcnew String(it2.getIter()->getEdenici().c_str());
flag = true;}
```

```
488.if (!flag)
489.dataGridView1->Rows[i]->Cells[2]->Value += gcnew String("(не виробляє задану
    продукцію)");
490.i++;}}
491.catch (ProdukcijaError error) {
492.MessageBox::Show(gcnew String(error.message()), "Помилка!");
493.}
494.catch (...) {
495.MessageBox::Show("Введено некоректні дані!", "Помилка!");}}
496.//----------створення таблиці: різниця між кредитом та отриманим прибутком----------
497.private: System::Void button12_Click(System::Object^ sender, System::EventArgs^ e)
498.{dataGridView1->Columns->Clear();
499.DataGridViewTextBoxColumn^ numser;
500.numser = gcnew DataGridViewTextBoxColumn();
501.numser->Name = "number";
502.numser->HeaderText = "№";
503.numser->Width = 30;
504.dataGridView1->Columns->Add(numser);
505.DataGridViewTextBoxColumn^ fermer;
506.fermer = gcnew DataGridViewTextBoxColumn();
507.fermer->Name = "fermer";
508.fermer->HeaderText = "Фермер";
509.fermer->Width = 175;
510.dataGridView1->Columns->Add(fermer);
511.DataGridViewTextBoxColumn^ raznica;
512.raznica = gcnew DataGridViewTextBoxColumn();
513.raznica->Name = "raznica";
514.raznica->HeaderText = "Різниця між кредитом та отриманим прибутком";
515.raznica->Width = 175;
516.dataGridView1->Columns->Add(raznica);
517.int i = 0;
518.for (FermerQueueIterator it(&fermerstvo->getFermerQueue()); !it.isEnd(); it.next())
519.{
520.dataGridView1->Rows->Add();
```

```
521. dataGridView1->Rows[i]->Cells[0]->Value = (i + 1).ToString();
522. dataGridView1->Rows[i]->Cells[1]->Value = gcnew String(it.getIter()->getName().c_str());
523. long pribil = 0;
524. int kredit = 0;
525. for (ProdukcijaQueueIterator it2(&it.getIter()->getProdukcijaQueue()); !it2.isEnd();
     it2.next())
526. {
527. pribil += it2.getIter()->getKolichestvo() * it2.getIter()->getCenazaedenicu();
528. for (PotrebaQueueIterator it3(&it2.getIter()->getPotrebiQueue()); !it3.isEnd(); it3.next())
529. kredit += it3.getIter()->getStoimost();
530. }
531. if(pribil == 0 && kredit == 0)
532. dataGridView1->Rows[i]->Cells[2]->Value += "фермер не виробляє ніякої продукції";
533. else if ((pribil - kredit) == 0)
534. dataGridView1->Rows[i]->Cells[2]->Value += "прибуток дорівнює суммі кредиту";
535. else
536. {
537. if ((pribil - kredit) > 0)
538. dataGridView1->Rows[i]->Cells[2]->Value += "+";
539. dataGridView1->Rows[i]->Cells[2]->Value += (pribil - kredit) + " гривень";
540. }
541. i++;}}
542. //------------------------------очищення таблиці-----------------------------------------
543. private: System::Void очиститиToolStripMenuItem_Click(System::Object^ sender,
     System::EventArgs^ e)
544. {fermerstvo->clearFermers();
545. comboBox3->Items->Clear();
546. comboBox4->Items->Clear();
547. comboBox5->Items->Clear();
548. dataGridView1->Columns->Clear();}
549. private: System::Void openFileDialog1_FileOk(System::Object^ sender,
     System::ComponentModel::CancelEventArgs^ e) {}
550. private: System::Void saveFileDialog1_FileOk(System::Object^ sender,
     System::ComponentModel::CancelEventArgs^ e) {}
```

```
551. private: System::Void comboBox4_SelectedIndexChanged(System::Object^ sender,
     System::EventArgs^ e) {}
552. private: System::Void textBox4_TextChanged(System::Object^ sender,
     System::EventArgs^e) {}
553. private: System::Void textBox7_TextChanged(System::Object^ sender,
     System::EventArgs^e) {}
554. private: System::Void textBox8_TextChanged(System::Object^ sender,
     System::EventArgs^e) {}};}
```

ДОДАТОК С

ПРИКЛАД ВИКОНАННЯ ПРОГРАМИ

Екранна форма початкового вікна приведена на рисунку С.1.

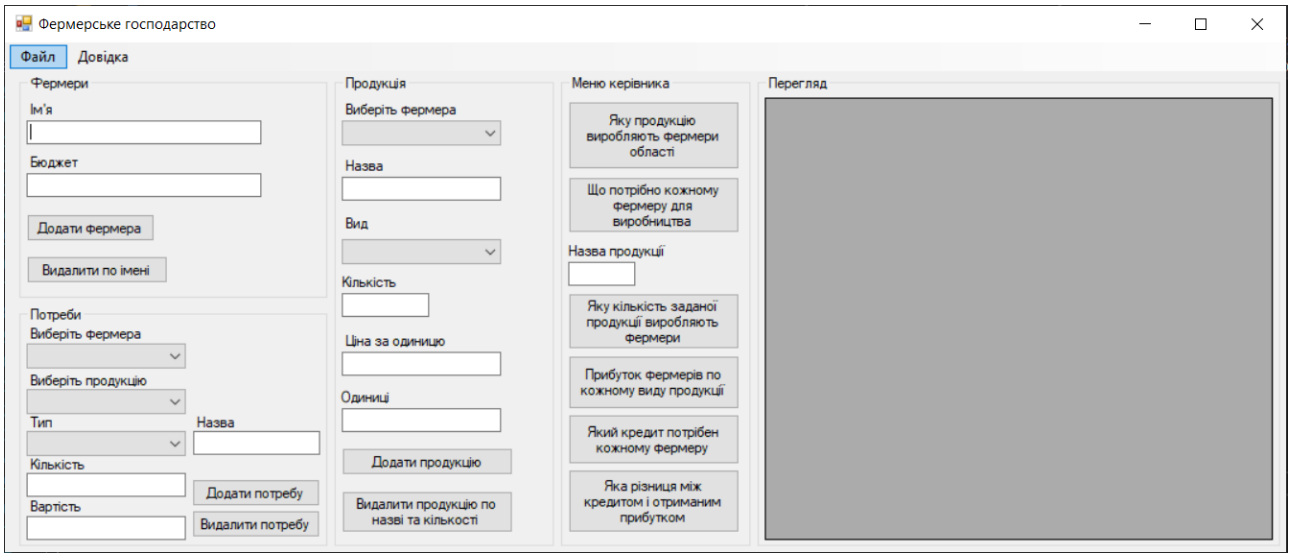


Рисунок С.1 – Вікно виконання програми

На рисунку С.2 наведено приклад самостійного заповнення даних, та виведення таблиці про продукцію яку виготовляють фермери.

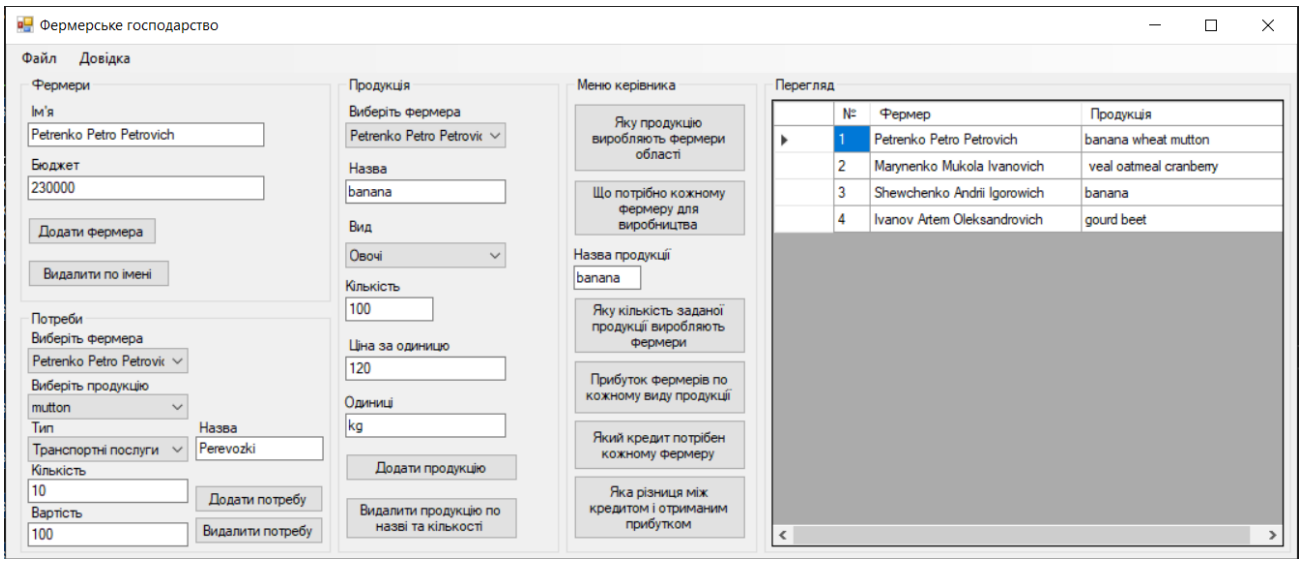


Рисунок С.2 – Вікно виконання програми

На рисунку С.3 наведено приклад появи помилки при введенні некоректних даних.

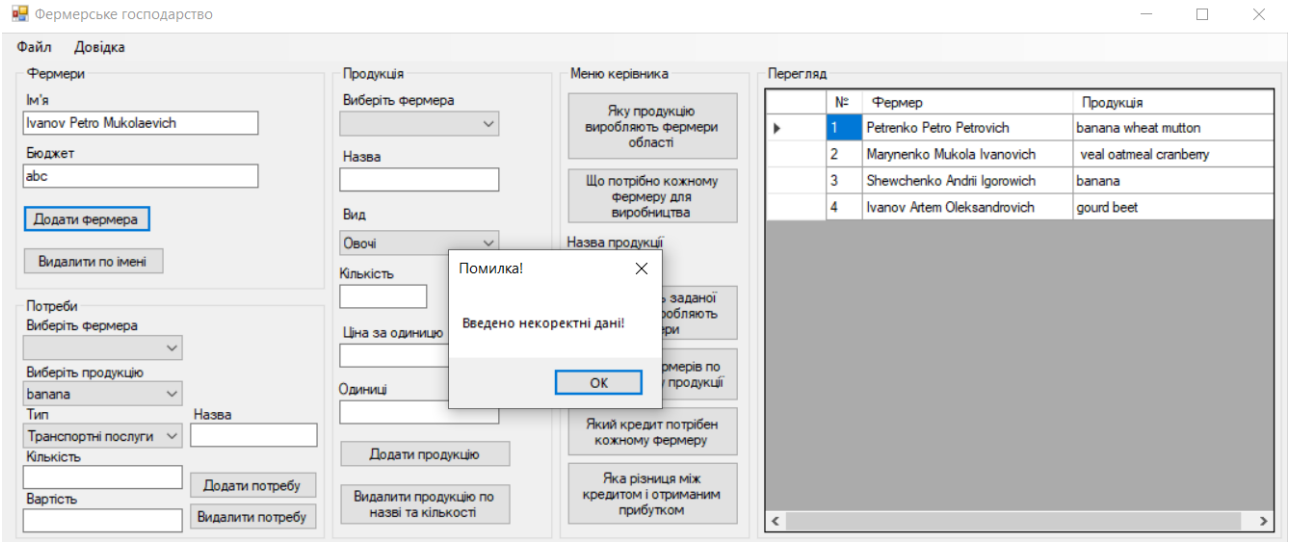


Рисунок С.3 – Вікно виконання програми

Приклад зчитування даних з файлу наведено на рисунку С.4.

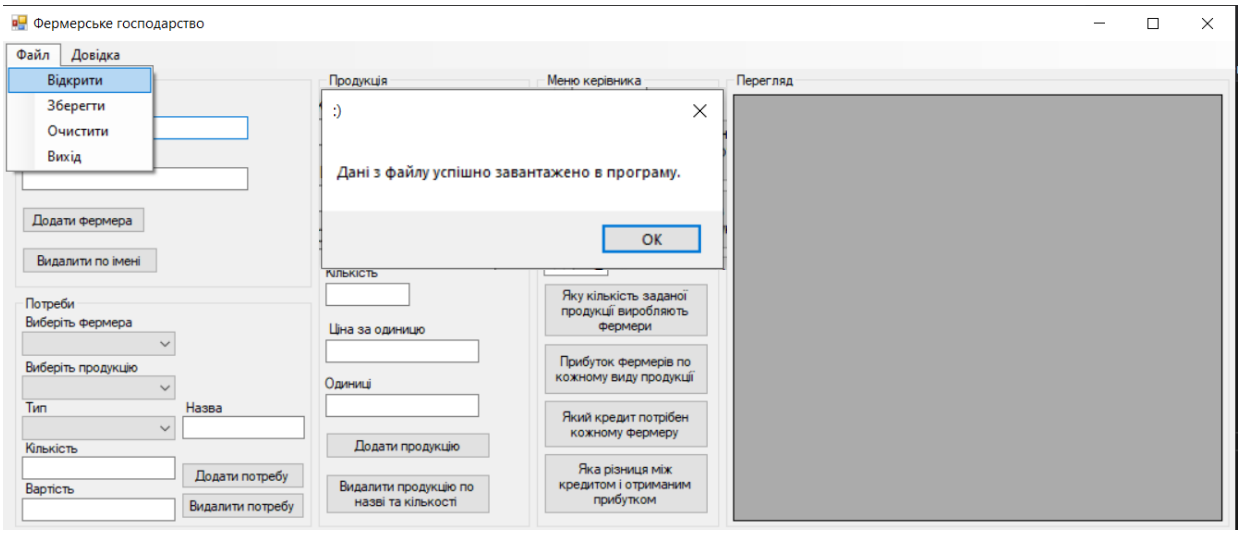


Рисунок С.4 – Вікно виконання програми

На рисунку С.5 зображено виведення таблиці даних про різницю між кредитом та отриманим прибутком для кожного фермера.

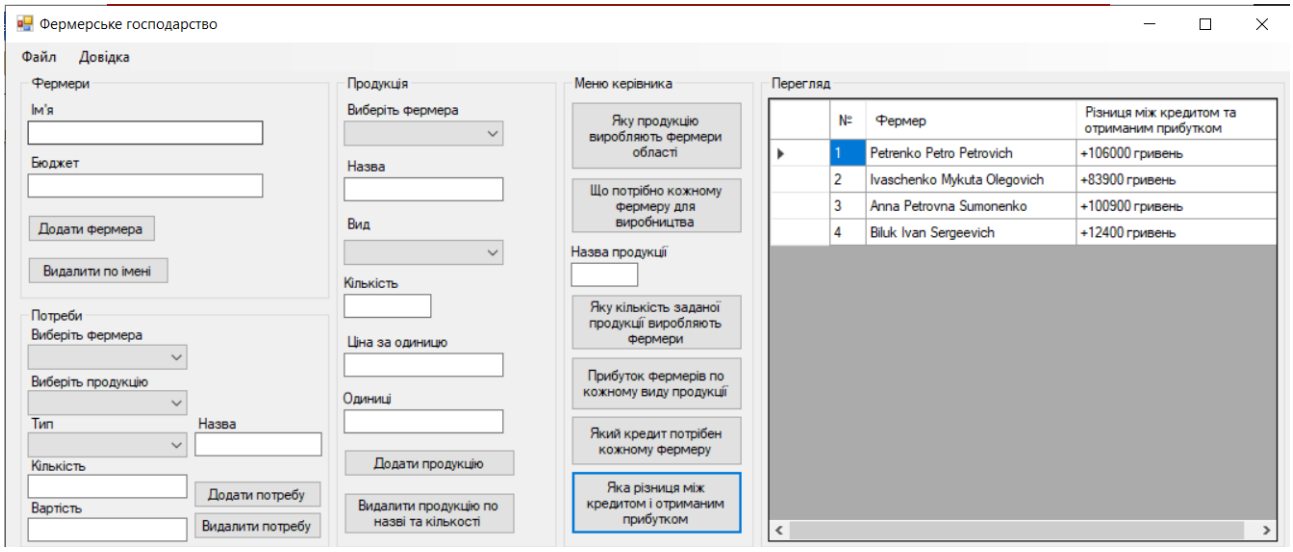


Рисунок С.5 – Вікно виконання програми