**Зміст**

Вступ

1 Загальна частина

1.1 Постановка задачі

1.2 Дослідження й аналіз об’єкту програмування

1.3 Опис засобів програмної реалізації

1.4 Вимоги до апаратного та програмного забезпечення

2 Практична частина

2.1 Створення та налагодження програми

2.2 Опис програми та її алгоритмів

2.3 Інструкція програміста

2.4 Інструкція оператора

Висновки

Литература

Додаток А – Код програми

**Вступ**

Об’єктно-орієнтоване програмування є основною парадигмою створення програмних додатків у сучасному світі. Засвоєння його основних концепцій є важливою складовою успішності розробника програмного забезпечення.

Предметом роботи є створення бібліотеки класів для виконання запитів до даних у базах даних. Метою курсової роботи було засвоєння навичок роботи з об’єктно-орієнтованим програмуванням.

Об`єктом дослідження є створення додатку з бліотеки класів для виконання запитів до даних у базах даних.

Програма була створена за допомогою мови програмування с# та середовище Microsoft Visual Studio на платформі Windows 10.

Ця програма розрахована на користувачів, що мають працювати з базами даних. Для основи даних для БД було використано обліку відділів компанії.

**1 Загальна частина**

**1.1 Постановка задачі**

Основне завдання курсового проекту полягає розробці програмного додатку, який на основі ієрархії класів, демонструватиме роботу різних видів тракторів, враховуючи їхнє призначення та характеристики, які вводяться користувачем.

Програма має приймати вхідні дані, які вводяться користувачем у елементи керування формою на відповідному меню.

**1.2 Аналіз об'єкту програмування**

При вивченні матеріалів для розробки програми було детальнiше вивчив структуру баз данних.

1)Загальні відомості про бази даних

База даних – це засіб збирання та впорядкування інформації. Бази даних можуть зберігати відомості про людей, продукти, замовлення або будь-що інше. Багато баз даних починаються зі списку в текстовому редакторі або електронній таблиці. Оскільки список збільшується, то в даних починають з'являтися неузгоджені та невідповідні значення. У формі списку дані буде важко розібратися, а також є обмежені способи пошуку або витягування підмножини даних для перевірки. Після того як ці проблеми почнуть з'являтися, радимо перенести дані до бази даних, створеної системою керування базами даних (СУБД), наприклад Access.

1. Використовуючи бази данних, можна:

додавати нові дані до бази даних, наприклад новий елемент до запасів;

редагувати наявні дані в базі даних, наприклад змінювати поточне розташування елемента;

видаляти відомості, якщо, наприклад, елемент продано або вилучено;

упорядковувати й переглядати дані різними способами;

надавати спільний доступ до даних іншим, використовуючи звіти, повідомлення електронної пошти, інтрамережу чи Інтернет.

## 3)Компоненти бази даних

[Таблиці](https://support.microsoft.com/uk-ua/office/%D0%BE%D1%81%D0%BD%D0%BE%D0%B2%D0%BD%D1%96-%D0%B2%D1%96%D0%B4%D0%BE%D0%BC%D0%BE%D1%81%D1%82%D1%96-%D0%BF%D1%80%D0%BE-%D0%B1%D0%B0%D0%B7%D0%B8-%D0%B4%D0%B0%D0%BD%D0%B8%D1%85-a849ac16-07c7-4a31-9948-3c8c94a7c204" \l "__toc257378512)

[Форми](https://support.microsoft.com/uk-ua/office/%D0%BE%D1%81%D0%BD%D0%BE%D0%B2%D0%BD%D1%96-%D0%B2%D1%96%D0%B4%D0%BE%D0%BC%D0%BE%D1%81%D1%82%D1%96-%D0%BF%D1%80%D0%BE-%D0%B1%D0%B0%D0%B7%D0%B8-%D0%B4%D0%B0%D0%BD%D0%B8%D1%85-a849ac16-07c7-4a31-9948-3c8c94a7c204" \l "__toc257378513)

[Звіти](https://support.microsoft.com/uk-ua/office/%D0%BE%D1%81%D0%BD%D0%BE%D0%B2%D0%BD%D1%96-%D0%B2%D1%96%D0%B4%D0%BE%D0%BC%D0%BE%D1%81%D1%82%D1%96-%D0%BF%D1%80%D0%BE-%D0%B1%D0%B0%D0%B7%D0%B8-%D0%B4%D0%B0%D0%BD%D0%B8%D1%85-a849ac16-07c7-4a31-9948-3c8c94a7c204" \l "__toc257378514)

[Запити](https://support.microsoft.com/uk-ua/office/%D0%BE%D1%81%D0%BD%D0%BE%D0%B2%D0%BD%D1%96-%D0%B2%D1%96%D0%B4%D0%BE%D0%BC%D0%BE%D1%81%D1%82%D1%96-%D0%BF%D1%80%D0%BE-%D0%B1%D0%B0%D0%B7%D0%B8-%D0%B4%D0%B0%D0%BD%D0%B8%D1%85-a849ac16-07c7-4a31-9948-3c8c94a7c204" \l "__toc257378515)

[Макроси](https://support.microsoft.com/uk-ua/office/%D0%BE%D1%81%D0%BD%D0%BE%D0%B2%D0%BD%D1%96-%D0%B2%D1%96%D0%B4%D0%BE%D0%BC%D0%BE%D1%81%D1%82%D1%96-%D0%BF%D1%80%D0%BE-%D0%B1%D0%B0%D0%B7%D0%B8-%D0%B4%D0%B0%D0%BD%D0%B8%D1%85-a849ac16-07c7-4a31-9948-3c8c94a7c204" \l "__toc257378516)

[Модулі](https://support.microsoft.com/uk-ua/office/%D0%BE%D1%81%D0%BD%D0%BE%D0%B2%D0%BD%D1%96-%D0%B2%D1%96%D0%B4%D0%BE%D0%BC%D0%BE%D1%81%D1%82%D1%96-%D0%BF%D1%80%D0%BE-%D0%B1%D0%B0%D0%B7%D0%B8-%D0%B4%D0%B0%D0%BD%D0%B8%D1%85-a849ac16-07c7-4a31-9948-3c8c94a7c204" \l "__toc257378517)

Первинним призначенням бази даних є зберігання масивів даних. Але їх широко використовують і для збереження адміністративної інформації та спеціалізованих даних, наприклад, для інженерних даних чи для економічних моделей.

Види баз данних:

#### 1.Ієрархічні бази даних

В ієрархічних базах даних кожен запис має одного «батька». Це створює деревоподібну структуру, в якій записи класифікуються за їхніми стосункам з ланцюжком батьківських записів.

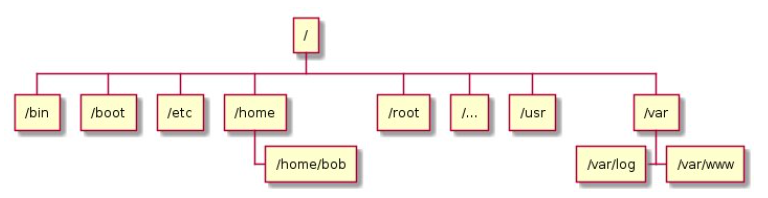


Рисунок 1.1- Приклад побудови ієрархічних зв'язків

****Наслідки:****

* інформація організована у вигляді дерева з відносинами «предок-нащадок»;
* кожен запис може мати не більше одного з батьків;
* зв'язки між записами виконані у вигляді фізичних покажчиків;
* неможливо реалізувати відносини «багатьох до багатьох».

****Приклади:****

* файлові системи
* DNS
* LDAP

#### 2. Мережеві бази даних

Мережеві бази даних розширюють функціональність ієрархічних: записи можуть мати більше одного батька. А значить, можна моделювати складні відносини.

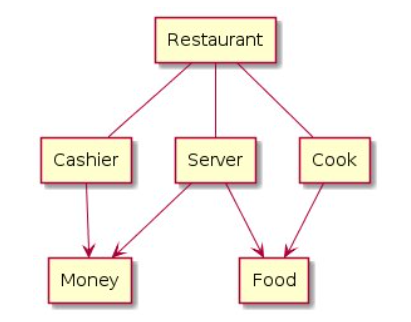


Рисунок 1.2- Приклад зв'язків в мережевій базі даних

****Наслідки:****

* мережеві бази даних подаються не деревом, а загальним графом
* обмежені тими ж шаблонами доступу, що ієрархічні БД

****Приклади:****

* IDMS

**1.3 Опис засобів програмної реалізації**

Для курсового проектування обрано середовище розробки Microsoft Visual Studio та , мову програмування С#.

C# - сучасна об'єктно-орієнтована і типобезпечна мова програмування. C# дозволяє розробникам створювати різні типи безпечних та надійних програм, що виконуються в .NET. C# відноситься до широко відомого сімейства мов C, і здасться добре знайомим будь-кому, хто працював з C, C++, Java або JavaScript. Тут представлений огляд основних компонентів мови C# 8 та ранніх версій. Якщо ви хочете вивчити мову за допомогою інтерактивних прикладів, рекомендуємо попрацювати з вступними посібниками C#.

C# — об'єктно-орієнтована, орієнтована на компоненти мова програмування. C# надає мовні конструкції безпосередньої підтримки такої концепції роботи. Завдяки цьому C# підходить для створення та застосування програмних компонентів. З моменту створення мова C# збагатилася функціями для підтримки нових робочих навантажень та сучасними рекомендаціями щодо розробки ПЗ. В основному C# - об'єктно-орієнтована мова. Ви визначаєте типи та їх поведінку.

Visual Studio — це стартовий майданчик для написання, налагодження та складання коду, а також подальшої публікації програм. Крім стандартного редактора і відладчика, які є в більшості середовищ IDE, Visual Studio включає компілятори, засоби автозавершення коду, графічні конструктори і багато інших функцій для поліпшення процесу розробки.



Рисунок 1.3- Середовище Visual Studio

На малюнку вище представлено середовище Visual Studio з відкритим проектом та підказки по основним вікнам та функціональним можливостям:

* Праворуч у верхньому куті вікна Оглядача рішень можна переглядати файли коду, переміщатися по них і керувати ними. Оглядач рішень дозволяє впорядкувати код шляхом об'єднання файлів у рішення та проекти.
* У центральному вікні редактора, з яким ви, ймовірно, працюватимете найдовше, відображається вміст файлу. У вікні редактора ви можете вносити зміни в код або розробляти інтерфейс користувача, наприклад вікно з кнопками або текстові поля.
* Вікно змін Git у нижньому куті праворуч дозволяє відстежувати робочі елементи та надавати спільний доступ до коду за допомогою Git, GitHub або інших технологій керування версіями.

**1.4 Вимоги до апаратного та програмного забезпечення**

Параметри Значення

Процесор Intel, AMD

Тактова частота процесора 1.4GHz+

Жорсткий диск 50Gb+

Об’єм оперативної пам’яті 4Gb+

Вимоги до програмного забезпечення

Операційна система

Windows 7/8/10

Linux

macOS

**2 Спеціальна частина**

**2.1 Створення та налагоджування програми**

Створення та налагодження програми виконувалося в програмі Microsoft Visual Studio.

Для початку створюємо консольну програму,як на рисунку.

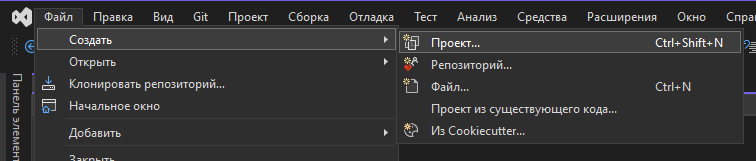


Рисунок 2.1-Створення нового проекту

Далі потрібно знайти Консольну програму і її створити, як на даному малюнку.

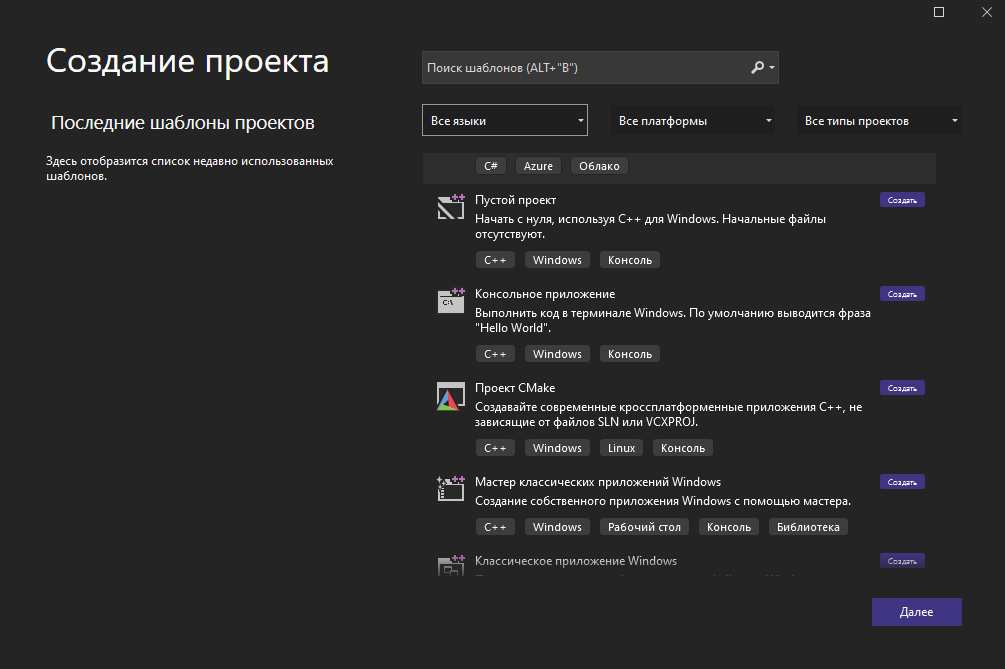


Рисунок 2.2- Створення консольного додатку

Створення програми для запиту до баз даних :

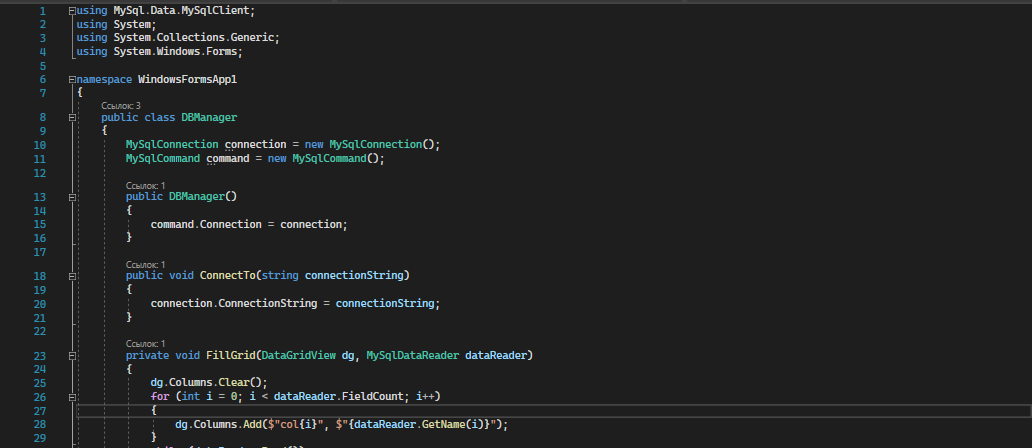


Рисунок 2.3- программа для запиту до баз даних

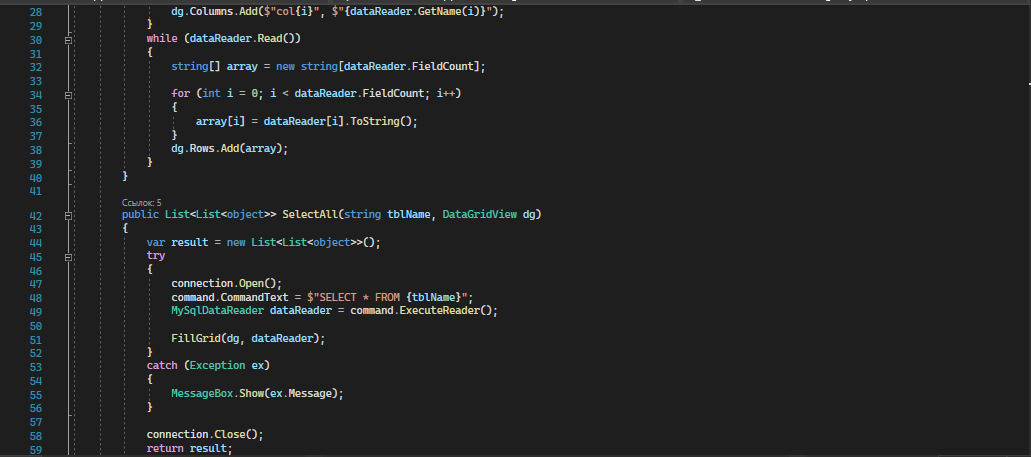


Рисунок2.4- программа для запиту до баз даних

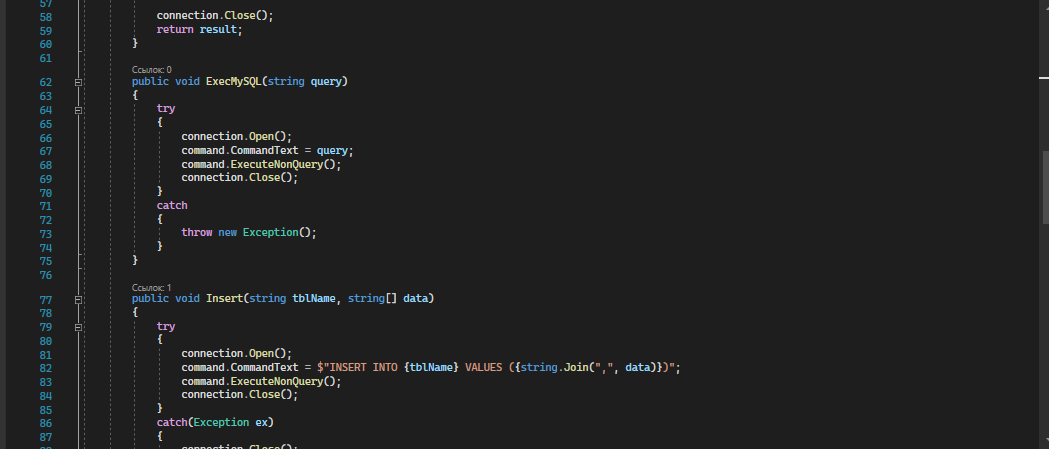


Рисунок 2.5- программа для запиту до баз даних

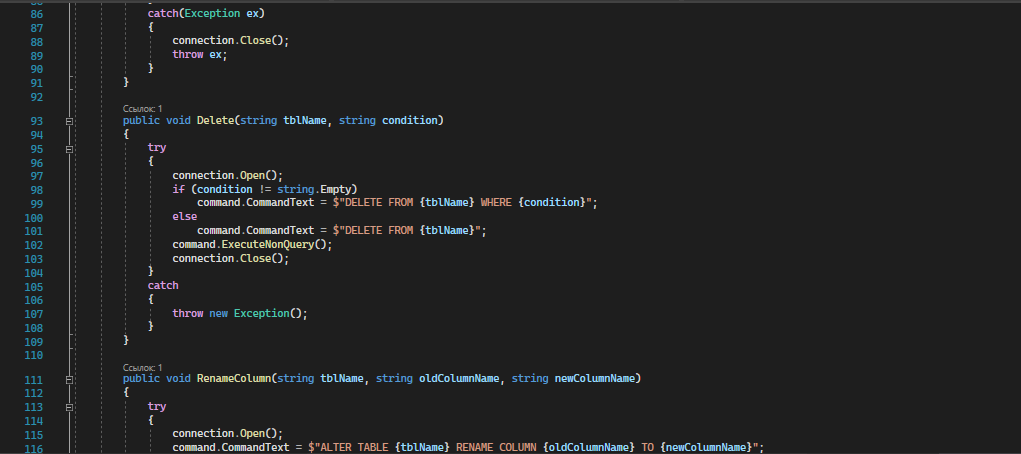


Рисунок 2.6 программа для запиту до баз даних

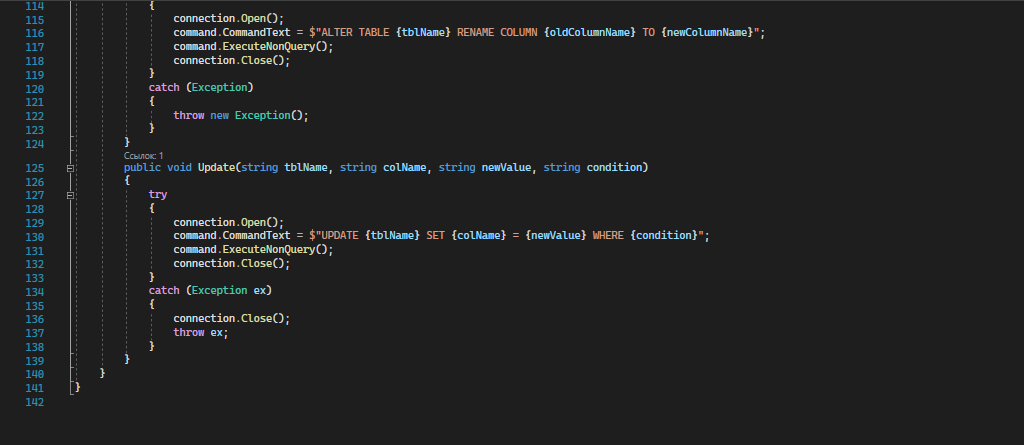


Рисунок 2.7- программа для запиту до баз даних

Після створення створення запиту до баз даних у програмі та бази даних, варто створити інтерфейс користувача:

Створення форми для входу :

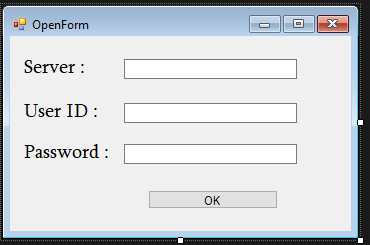


Рисунок 2.8- iнтерфейс для входу у базу даних



Рисунок 2.9 –OpenForm iнтерфейс для входу у базу даних

Створення вiзуального iнтерфейсу для відображення й взаємодій з базою даних у пррограммi:

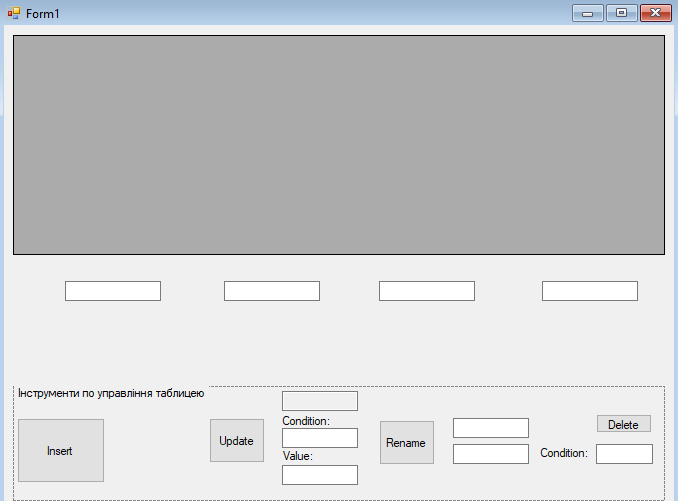


Рисунок 2.10-iнтерфейсу для відображення й взаємодій з базою даних

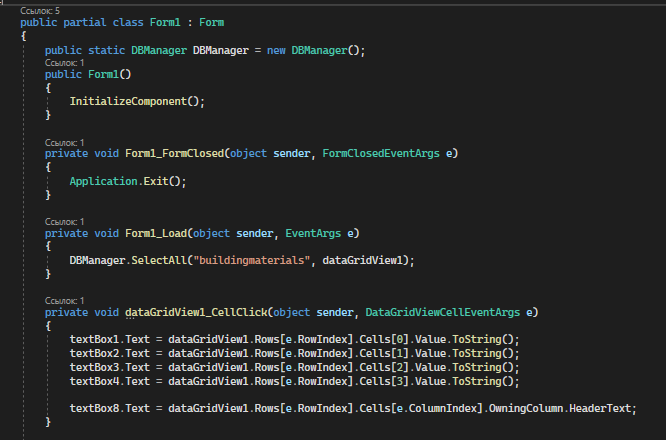


Рисунок 2.11-Код iнтерфейсу для відображення й взаємодій з базою даних

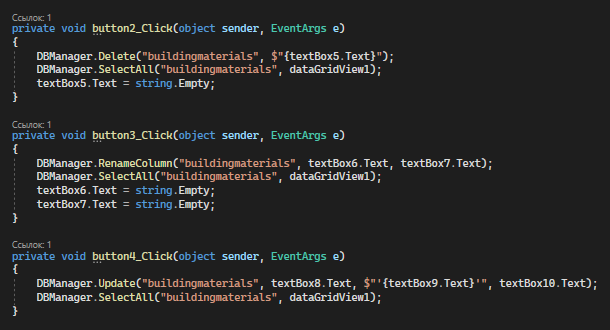


Рисунок 2.12 –Код iнтерфейсу для відображення й взаємодій з базою даних

Для перевiрки роботи програми створюємо базу даних в середовищі MySQL

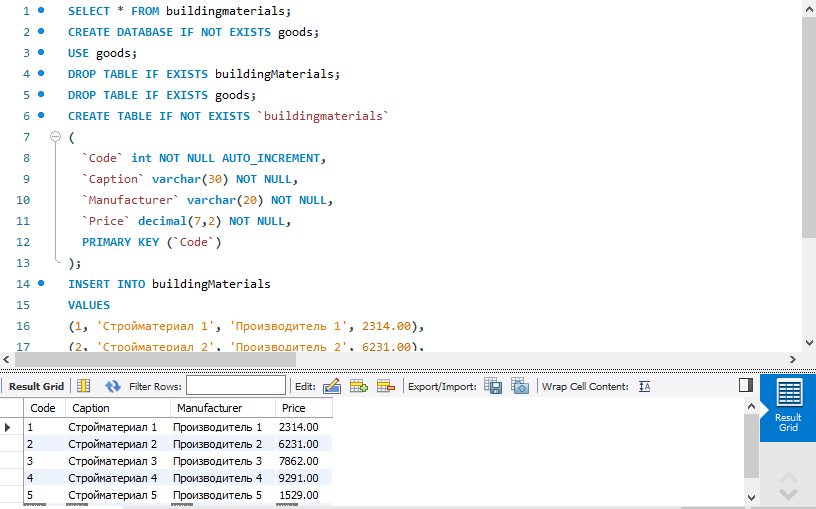


Рисунок 2.13 – База даних

Візуальна складова програми використовує елементи керування WinForms для введення даних у базу даних. Для того, щоб форма функціонувала, необхідно прописати функціонал для її елементів. Одним з таких елементів є кнопка «Insert», що зберігає змінену інформацію у таблиці. Кнопка також здійснює перевірку на коректність введених даних та не пропускає виконання програми при пустих полях чи з однаковими даними:

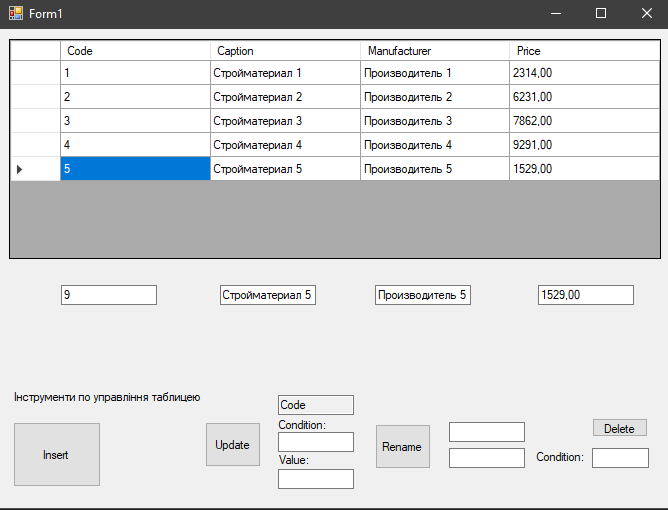


Рисунок 2.14 -Вiдображенння бази данних

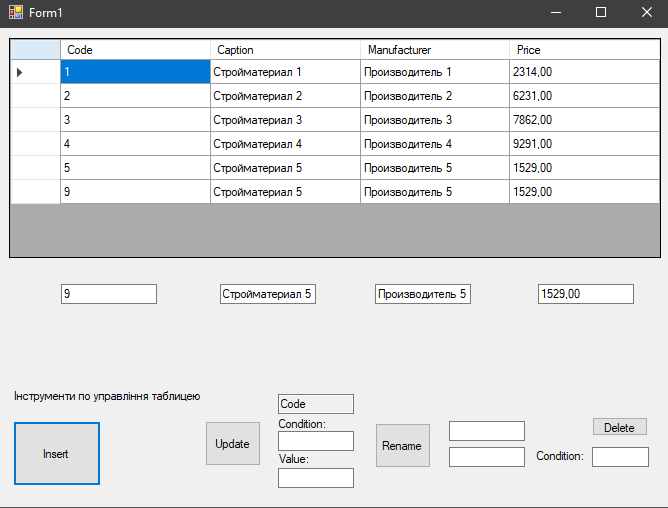


Рисунок 2.15–Взаємодія з базою даних

**2.2 Опис програми та її алгоритмів**

**Опис функцій системи.** Система розрахована на багатьх користувачив з доволі простим функціоналом. Створення *UML* діаграм допоможе краще зрозуміти логіку програмного забезпечення, як воно повинно працювати, показати усі його залежні компоненти та об’єкти, їх взаємодію та функціонал програмного забезпечення.

Суть діаграми прецедентів полягає в тому, що проектована система подається у вигляді множини сутностей чи акторів, що взаємодіють із системою за допомогою так званих варіантів використання. Варіант використання (англ. use case) використовують для описання послуг, які система надає актору. Іншими словами, кожен варіант використання визначає деякий набір дій, який виконує система під час діалогу з актором. При цьому нічого не говориться про те, яким чином буде реалізовано взаємодію акторів із системою.

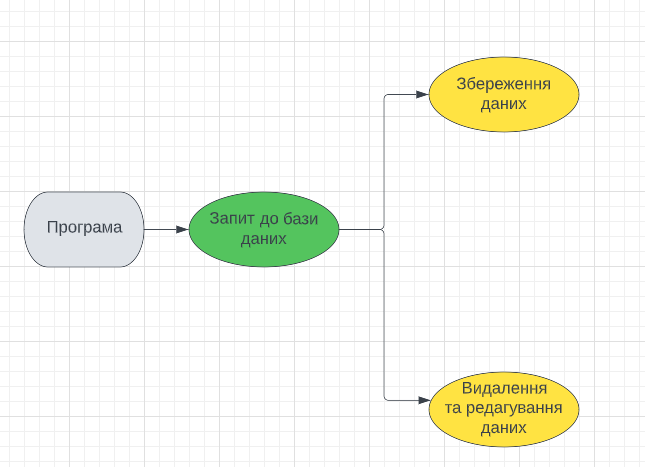


Рисунок 2.16 -UML діаграма прецедентів

У програмі буде представлено один базового класу DBManagerClass:

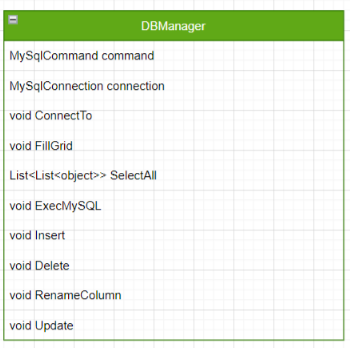


Рисунок 2.17 – Діаграма класів програми

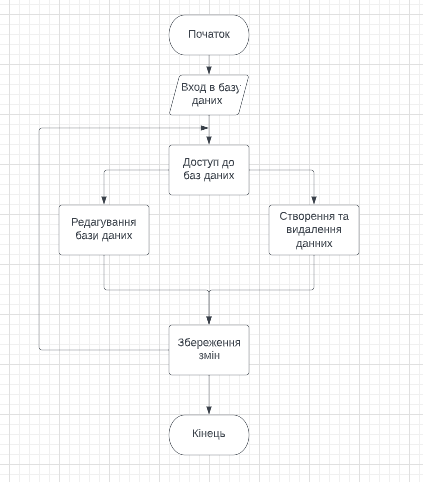


Рисунок 2.18-Блок схем роботи программи

**2.3 Інструкція програміста**

Програмний продукт допоможе користувачу, який вміє користуватися персональним комп’ютером та може працювати з базами даних. Розроблена програма дуже проста у користуванні, не потребує ніяких додаткових навчичок.

Автор: Кузбiт Антон

Мова програмування: С#.

Середовище програмування: Microsoft Visual Studio

Опараційна система- Windows 10

Оперативна пам’ять: не менше 4 Gb

Тактова частота процесору – 1.54 GHz і вище;

Об’єм вільного дискового простору – не менше 16 Гб;

Наявність клавіатури та миші.

**2.4 Інструкція оператора**

Щоб взаємодіяти з програмою, використовується миша. Для запуску програми необхідно запустити файл WindowsFormsApp1.exe. Для цього зайти в папку Kuzbit\WindowsFormsApp1\WindowsFormsApp1\bin\Debug. На рисунку 2. можна продивитися вікно з вибором файлу для завантаження програми.

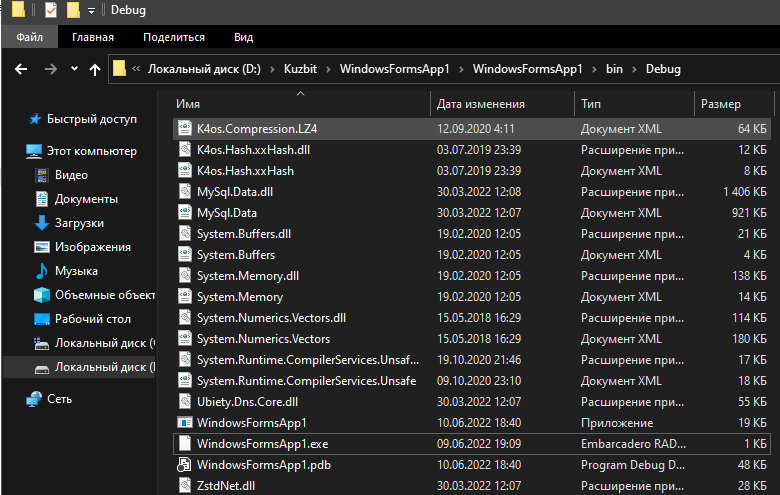


Рисунок 2.19 – Вікно з вибором файлу програми

У результаті роботи було створено додаток, який виконує запит до баз даних. При запуску програми користувач бачить наступне вікно на даному рисунку:

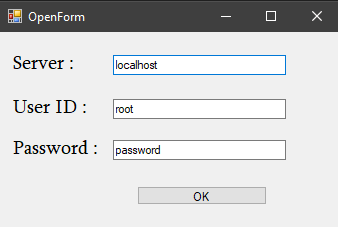


Рисунок 2.20 – Початкова форма роботи з програмою

При взаємодії з формою вам потрібно ввести дані сервера, щоб отримати доступ до бази даних:

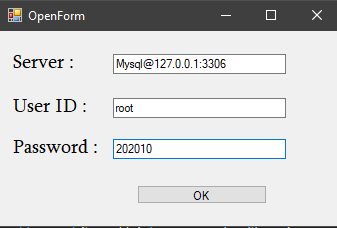


Рисунок 2.21 – Заповнена форма

Після заповнення полів вам відкриється вікно з даними в базі даних куди ви входили:

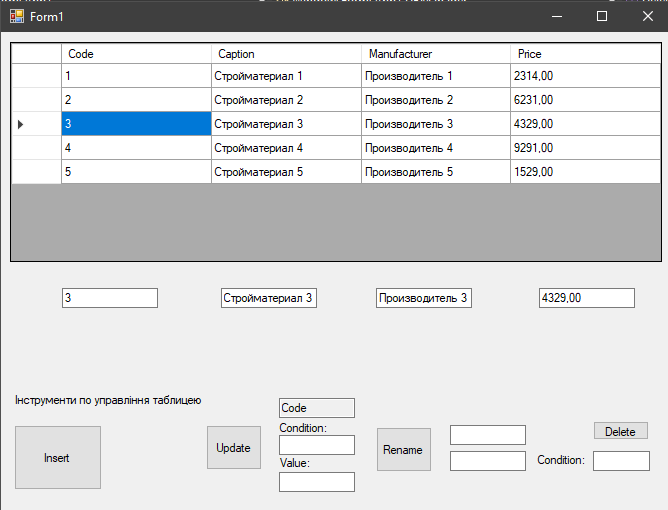


Рисунок 2.22 – Поля з даними з бази даних

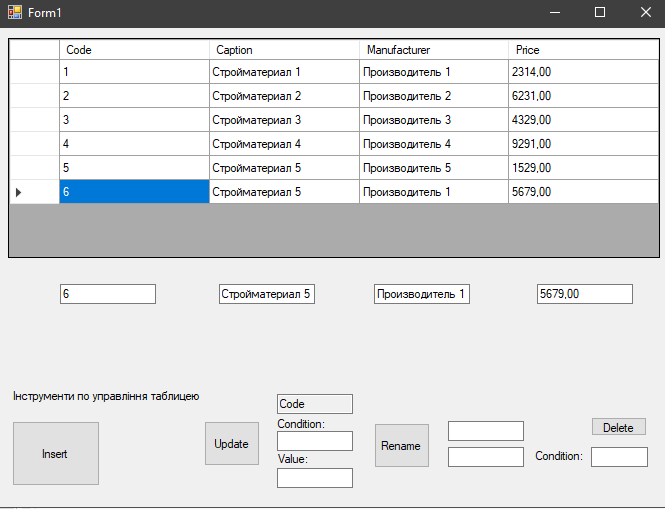


Рисунок 2.23 –Внесенi змiни у базу даних

**Висновки**

Під час написання курсової роботи була створена бібліотека класів для виконання запитів до даних у базах даних. У результаті роботи було опановано навички у об’єктно-орієнтованому програмуванні, було опановано його основні принципи – інкапсуляцію, поліморфізм, наслідування, а також було засвоєно принципи роботи з класами та об’єктами.

Програму було виконано на мові програмування C#

Також було обрано середовище програмування: Microsoft Visual Studio на платформі Windows 10.

C# - сучасна об'єктно-орієнтована і типобезпечна мова програмування. C# дозволяє розробникам створювати різні типи безпечних та надійних програм, що виконуються в .NET. C# відноситься до широко відомого сімейства мов C, і здасться добре знайомим будь-кому, хто працював з C, C++, Java або JavaScript. Тут представлений огляд основних компонентів мови C# 8 та ранніх версій. Якщо ви хочете вивчити мову за допомогою інтерактивних прикладів, рекомендуємо попрацювати з вступними посібниками C#.

C# — об'єктно-орієнтована, орієнтована на компоненти мова програмування. C# надає мовні конструкції безпосередньої підтримки такої концепції роботи. Завдяки цьому C# підходить для створення та застосування програмних компонентів. З моменту створення мова C# збагатилася функціями для підтримки нових робочих навантажень та сучасними рекомендаціями щодо розробки ПЗ. В основному C# - об'єктно-орієнтована мова. Ви визначаєте типи та їх поведінку.

# **Література**

* 1. Р. Лафоре. Обьектно-ориентированое программирование, 4-е издание.

Санкт-Петербург, 2004.

* 1. Бьерн Страуструп. Язык программирования С++ – специальное издание. Москва, 2006.
  2. Березин Б.И., Березин С.Б. Начальний курс С и С++. 2001.
  3. [www.CyberForum.ru](http://www.CyberForum.ru)

1. Айвор Хортон. Visual C++ 2010: полный курс .Visual C++ 2010. — М.: [Диалектика](https://ru.wikipedia.org/w/index.php?title=%D0%94%D0%B8%D0%B0%D0%BB%D0%B5%D0%BA%D1%82%D0%B8%D0%BA%D0%B0_(%D0%B8%D0%B7%D0%B4%D0%B0%D1%82%D0%B5%D0%BB%D1%8C%D1%81%D1%82%D0%B2%D0%BE)&action=edit&redlink=1" \o "Диалектика (издательство) (страница отсутствует)), 2010. — С. 1216.
2. Береза А.М., Основи створення інформаційних систем. Навчальний посібник.// 2-ге видання. – К.: КНЕУ,2001р., 156ст
3. Березин Б.И., Березин С.Б. Начальний курс С и С++. 2001.
4. Бьерн Страуструп. Язык программирования С++ – специальное издание. Москва, 2006.
5. C#. Матеріал із сайту про програмування – Metanit.com. [Електронний ресурс] – Режим доступу: https://metanit.com/sharp/ -
6. Головна сторiнка MySQL <https://dev.mysql.com/>/

8 Березин Б.И., Березин С.Б. Начальний курс С и С++. 2001.

[www.CyberForum.ru](http://www.CyberForum.ru)

# **Додаток А**

**Коду програми**

using MySql.Data.MySqlClient;

using System;

using System.Collections.Generic;

using System.Windows.Forms;

namespace WindowsFormsApp1

{

public class DBManager

{

MySqlConnection connection = new MySqlConnection();

MySqlCommand command = new MySqlCommand();

public DBManager()

{

command.Connection = connection;

}

public void ConnectTo(string connectionString)

{

connection.ConnectionString = connectionString;

}

private void FillGrid(DataGridView dg, MySqlDataReader dataReader)

{

dg.Columns.Clear();

for (int i = 0; i < dataReader.FieldCount; i++)

{

dg.Columns.Add($"col{i}", $"{dataReader.GetName(i)}");

}

while (dataReader.Read())

{

string[] array = new string[dataReader.FieldCount];

for (int i = 0; i < dataReader.FieldCount; i++)

{

array[i] = dataReader[i].ToString();

}

dg.Rows.Add(array);

}

}

public List<List<object>> SelectAll(string tblName, DataGridView dg)

{

var result = new List<List<object>>();

try

{

connection.Open();

command.CommandText = $"SELECT \* FROM {tblName}";

MySqlDataReader dataReader = command.ExecuteReader();

FillGrid(dg, dataReader);

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

connection.Close();

return result;

}

public void ExecMySQL(string query)

{

try

{

connection.Open();

command.CommandText = query;

command.ExecuteNonQuery();

connection.Close();

}

catch

{

throw new Exception();

}

}

public void Insert(string tblName, string[] data)

{

try

{

connection.Open();

command.CommandText = $"INSERT INTO {tblName} VALUES ({string.Join(",", data)})";

command.ExecuteNonQuery();

connection.Close();

}

catch(Exception ex)

{

connection.Close();

throw ex;

}

}

public void Delete(string tblName, string condition)

{

try

{

connection.Open();

if (condition != string.Empty)

command.CommandText = $"DELETE FROM {tblName} WHERE {condition}";

else

command.CommandText = $"DELETE FROM {tblName}";

command.ExecuteNonQuery();

connection.Close();

}

catch

{

throw new Exception();

}

}

public void RenameColumn(string tblName, string oldColumnName, string newColumnName)

{

try

{

connection.Open();

command.CommandText = $"ALTER TABLE {tblName} RENAME COLUMN {oldColumnName} TO {newColumnName}";

command.ExecuteNonQuery();

connection.Close();

}

catch (Exception)

{

throw new Exception();

}

}

public void Update(string tblName, string colName, string newValue, string condition)

{

try

{

connection.Open();

command.CommandText = $"UPDATE {tblName} SET {colName} = {newValue} WHERE {condition}";

command.ExecuteNonQuery();

connection.Close();

}

catch (Exception ex)

{

connection.Close();

throw ex;

}

}

}

}