# Лабораторна робота №10

**Конструювання програм з БД за допомогою технології ADO.NET та наборами в середовищі програмування Visual C#**

***Мета роботи:*** *засвоїти методику та виробити практичні навички роботи з ADO.NET та наборами в середовищі програмування Visual C# 2010 Express Edition.*

* 1. **Теоретичні відомості.**

ADO.NET (ActiveX Data Objects .NET) — це набір бібліотек, що поставляється з Microsoft .NET Framework і призначений для взаємодії з різними сховищами даних з .NET-додатків. Бібліотеки ADO.NET включають класи для приєднання до джерела даних, виконання запитів і обробки їхніх результатів. Крім того, ADO.NET можна використовувати в якості надійного, ієрархічно організованого, відокремленого кешу даних для автономної роботи з даними.

ADO.NET була розроблена компанією Microsoft, для вирішення проблем, які виникали при роботі з ADO та попередніми технологіями, такими як: Data Access Objects(DAO), Remote Data Objects (RDO). Основною перевагою ADO.NET додатків є гнучкість та розвинута підтримка XML.

## Основні переваги

1. Підтримка XML

ADO.NET створювався з врахуванням XML, тому набагато ефективніше обробляє XML-дані, ніж ADO.

1. Простота модифікації

Протягом терміну служби системи в неї можна вносити незначні зміни, однак спроби провести архітектурні зміни трапляються рідко, через виняткову складність завдання. На жаль, при природному розвитку подій такі зміни іноді виявляються необхідними.

1. Простота програмування

Компоненти даних ADO.NET в Visual Studio інкапсулюють функціональні можливості доступу до даних різними способами, що допомагає розробляти програмні продукти значно швидше і з меншою кількістю помилок.

1. Продуктивність

Для непідключених додатків набори даних ADO.NET дають виграш в продуктивності у порівнянні із непідключеними наборами записів ADO. Передача непідключеного набору записів між рівнями за допомогою COM-упаковки може призвести до великої витрати обчислювальних ресурсів, тому що значення в наборі записів перетворюються до типів даних, відомих COM. У ADO.NET таке перетворення типів даних не потрібно.

## DataSet

Модель об'єктів ADO.NET дуже велика, але в її основі лежить досить простий набір класів. Найбільш важливим з них вважається DataSet, який представляє собою відображення використовуваної бази даних, перенесене на машину користувача. При цьому немає необхідності в постійному з'єднанні з сервером бази даних для модифікації даних. Лише іноді з'єднується DataSet з його батьківською базою даних і модифікує її внесеними змінами.

DataSet складається з об'єктів типу DataTable та об'єктів DataRelation. До них можна звертатися як до властивостей об'єкта DataSet. Властивість Tables повертає об'єкт типу DataTableCollection, який містить всі об'єкти DataTable використовуваної бази.

## Таблиці і поля (об'єкти DataTable і DataColumn)

Об'єкт типу DataTable представляє собою таблицю бази даних. Такий об'єкт може бути створений програмно або шляхом запиту до бази даних. Об'єкт DataTable складається з рядків і стовпців. Рядки представляють собою окремі записи таблиці, стовпці – відповідні поля.

Для отримання сукупності стовпців об'єкт DataSet має властивість Columns, яка повертає DataColumnCollection, яке в свою чергу складається з об'єктів типу DataColumn. Кожен об'єкт DataColumn представляє собою окремий стовпець таблиці, з якого можна отримати будь-який запис.

## Зв'язки між таблицями (об'єкт DataRelation)

Крім набору таблиць DataSet має властивість Relations, що повертає об'єкт типу DataRelationCollection, що складається з об'єктів DataRelation.

Кожен DataRelation об'єкт зберігає дані про зв'язки між двома таблицями за допомогою об'єктів DataColumn. Наприклад, в базі даних Northwind таблиця Customers має зв'язок з таблицею Orders допомогою стовпця CustomerlD. Таке відношення називається мовою баз даних один до багатьох (one-to-many). Для будь-якого замовлення може бути тільки один замовник, але один замовник може мати скільки завгодно замовлень.

## Рядки (об'єкт DataRow)

Властивість Rows об'єкта DataTable повертає сукупність всіх рядків таблиці – DataRowCollection. Цю властивість слід застосовувати для того, щоб користуватися результатами запитів до бази даних. У ADO.NET немає необхідності в ітераційному обході DataSet для отримання даних. Ви можете звертатись до записів таблиці як до елементів простого масиву. Це значно спрощує процес доступу до елементів бази.

// створення DataTable.

DataTable table = new DataTable("Product");

// створення DataColumn і встановлення властивостей.

DataColumn column = new DataColumn();

column.DataType = System.Type.GetType("System.Decimal");

column.AllowDBNull = false;

column.Caption = "Price";

column.ColumnName = "Price";

column.DefaultValue = 25;

//Додавання колонки в таблиць.

table.Columns.Add(column);

// додавання 10 рядків.

DataRow row;

for(int i = 0; i < 10; i++)

{

row = table.NewRow();

row["Price"] = i + 1;

// Додавання рядка в таблицю

table.Rows.Add(row);

}

## DataAdapter

DataSet – це образ реляційної бази даних. ADO.NET використовує об'єкт типу DataAdapter як міст між DataSet і джерелом даних, що є основною базою даних. DataAdapter містить метод Fill() для оновлення даних з бази і заповнення DataSet.

## DBCommand і DBConnection

Об'єкт DBConnection представляє собою засіб для з'єднання DataSet з джерелом даних. З'єднання може бути доступно за допомогою різних командних об'єктів. Наприклад, об'єкт DBCommand дозволяє послати команду (зазвичай це SQL запит або збережена процедура) до бази даних. Часто командні об'єкти створюються неявно, під час формування об'єкта DataSet. Але ADO.NET дозволяє явно звертатися до таких об'єктів.

Якщо потрібний об’єкт DBCommand, який не повертає результат у вигляді таблиці, слід використовувати метод ExecuteNonQuery.

comm = conn.CreateCommand();

//INSERT IN TO DATABASE

comm.CommandText = "INSERT INTO books (Autor, Name, Vudannj, Rik)VALUES

('"+tempBook.autor+"','"+tempBook.Name+"','"+tempBook.Vudannj+"',"+tempBook.rik+");";

int rows = comm.ExecuteNonQuery();

return rows;

Деколи потрібно виконати запит, відповідь на який буде включати єдине поле одного стовпця. Використовують команду ExecuteScalar.

int count;

comm = conn.CreateCommand();

comm.CommandText = "Select count(\*) from books";

count = (int)comm.ExecuteScalar();

return count;

Метод ExecuteReader вертає екземпляр DBDataReader, який містить результат запиту

DbDataReader dr;

DataTable temp = new DataTable();

comm = conn.CreateCommand();

comm.CommandText = "Select \* from books";

dr = comm.ExecuteReader();

temp.Load(dr);

return temp;

## Провайдери даних .NET

У ADO.NET використовуються так звані провайдери даних (Data Providers) .NET. Вони забезпечують доступ до відповідних джерел даних і містять чотири ключових об'єкти (Connection, Command, DataReader і DataAdapter). На даний час з ADO.NET надаються провайдери:

* SQL Server .NET Data Provider. Призначений для роботи з базами даних Microsoft SQL Server 7.0 і більш пізніх версій. Оптимізований для доступу до SQL Server і взаємодіє з ним безпосередньо по «рідному» протоколу передачі даних SQL Server. Варто користуватися цим провайдером при роботі з SQL Server 7.0 або SQL Server 2000.
* OLE DB .NET Data Provider. Керований провайдер для джерел даних OLE DB. Трохи поступається за ефективністю SQL Server .NET Data Provider, так як взаємодіє з базою даних через рівень OLE DB. Цим провейдером не підтримується провайдер OLE DB для ODBC (Open Database Connectivity). Для джерел даних ODBC використовують ODBC .NET Data Provider, описаний нижче.

Решта провайдерів даних .NET в даний час знаходяться в стані бета-тестування.

* ODBC .NET Data Provider. У даний момент доступна для завантаження перша бета-версія. Цей провайдер забезпечує «рідний» доступ до ODBC-драйверів так само, як і OLE DB .NET Data Provider до «рідних» провайдерам OLE DB.
* Керований провайдер для зчитування XML з SQL Server 2000. XML for SQL Server Web update 2 (у даний момент проходить бета-тестування) включає, крім усього іншого, керований провайдер, призначений спеціально для зчитування XML з SQL Server 2000.

## Структура просторів імен

Типи (класи, структури, списки і т.д.), пов'язані з кожним із провайдерів даних .NET, знаходяться у власних просторах імен:

* System.Data.SqlClient. Містить типи SQL Server .NET Data Provider, Керівництво по архітектурі доступу до даних на платформі .NET
* System.Data.OleDb. Містить типи OLE DB .NET Data Provider.
* System.Data.Оdbc. Містить типи ODBC .NET Data Provider.
* System.Data. Містить типи, незалежні від провайдерів, наприклад DataSet і DataTable.

Для кожного з провайдерів у його просторі імен містяться реалізації об'єктів Connection, Command, DataReader і DataAdapter. Імена реалізацій об'єктів із простору імен SqlClient починаються з префікса "Sql", а імена реалізацій з простору імен OleDb – з префікса «OleDb». Наприклад, реалізація об'єкта Connection з простору імен SqlClient називається SqlConnection, тоді як її еквівалент з OleDb – OleDbConnection. Відповідні реалізації об'єкта DataAdapter називаються SqlDataAdapter і OleDbDataAdapter.

### Створення об’єкта SqlConnection

SqlConnection є об'єктом, як і будь-який інший об'єкт C#. Оголошення об’єкта SqlConnection відбувається наступним чином:

SqlConnection conn = new SqlConnection("Data Source=(local);Initial Catalog=DatabaseName;Integrated Security=SSPI");

Описане вище оголошення об’єкта SqlConnection використовує конструктор з єдиним параметром – стрічкою з’єднання (Connection String).

|  |  |
| --- | --- |
| **Параметр стрічки з'єднання** | **Опис** |
| Data Source | Визначає сервер. Це може бути локальний комп’ютер, доменне ім’я іншого комп’ютера або IP-адреса |
| Initial Catalog | Ім’я БД |
| Integrated Security | Оберіть SSPI для з’єднання з логіном вашого користувача Windows |
| User ID | Ім’я користувача в SQL Server. |
| Password | Пароль для користувача SQL Server. |

Для використання з'єднання із вказанням користувача і паролю використовуються оголошення наступного вигляду:

SqlConnection conn = new SqlConnection("Data Source=DatabaseServer; Initial Catalog=DatabaseName;  
User ID=YourUserID;Password=YourPassword");

### Створення об’єкта OleDbConnection і OleDbCommand

Ole-об’єкти зручно використовувати для з’єднання із локальною базою даних, наприклад створеною в MS Access. Для цього використовується наступна стрічка з’єднання:

@"Provider=Microsoft.Jet.OLEDB.4.0;Data Source= ...\\...\\db4.mdb"

де ...\\...\\db4.mdb – це шлях до файлу БД. «...\\...\\» означає, що файл лежить у тій же папці, де і ехе-файл.

using System;

using System.Data;

using System.Data.OleDb;

class OleDbAdapterDemo

{

static void Main()

{

string conString = @"Provider=Microsoft.Jet.OLEDB.4.0;Data Source=...\\...\\db4.mdb";

// ініціалізація з'єднання

OleDbConnection conn = new OleDbConnection(conString);

try

{

conn.Open();

//створити data adapter

OleDbDataAdapter dataAdapter = new OleDbDataAdapter("SELECT name,status FROM Vykladachi Order by name", conn);

//створити DataSet

DataSet dataSet = new DataSet();

//назвати DataSet

dataSet.DataSetName = "Vykladachi";

//заповнити DataSet, використовуючи DataAdapter

dataAdapter.Fill(dataSet);

//виведення на екран кожного запису БД

foreach (DataRow dataRow in dataSet.Tables[0].Rows)

{

Console.WriteLine(dataRow[0] + "\t\t" + dataRow[1]);

}

//створити рядок з новии даними

DataTable dt = dataSet.Tables[0];

DataRow dr=dt.NewRow();

dr[0] = "Бойчук А.А.";

dr[1] = "викладач";

//додати рядок до dataSet

dataSet.Tables[0].Rows.Add(dr);

Console.WriteLine("\n---\n");

//виведення на екран кожного запису БД

foreach (DataRow dataRow in dataSet.Tables[0].Rows)

{

Console.WriteLine(dataRow[0] + "\t\t" + dataRow[1]);

}

}

finally

{

//закрити з'єднання

if (conn != null)

{

conn.Close();

}

}

}

}

* 1. **Завдання**

Спроектувати та розробити базу даних інформаційної системи на Microsoft Access. Створити програму, яка буде працювати з БД (виводити кількість записів у таблиці, виводити всі записи, додавати та видаляти записи з таблиці, реалізувати фільтри). Розробити UML діаграми.

* 1. **Приклад виконання лабораторної роботи**

Приклад на основі інформаційної системи про книги з полями в базі даних: Id, Автор, Назва, Видання, Рік.

В Microsoft Access за допомогою конструктора створюємо таблицю з назвою Books із даними полями, на рисунку 1 зображено конструктор таблиці та на рисунку 2 зображено створену таблицю із тестовими даними.

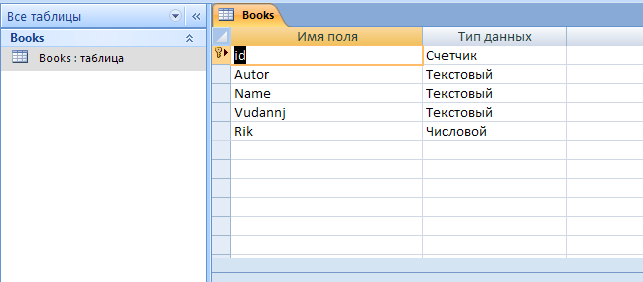


Рис. 1. Конструктор створення таблиці books в Microsoft Access.

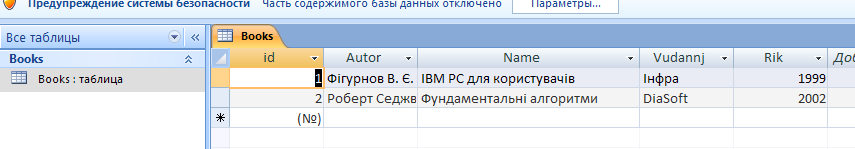


Рис. 2.Створена таблиця Books із тестовими даними.

Додаємо два класи та називаємо їх Book та UseDB. Клас Book відображає в інформаційній системі інформацію про об’єкт, в нашому випадку про книжку а у класі UseDB реалізовуються методи для роботи з базою даних.

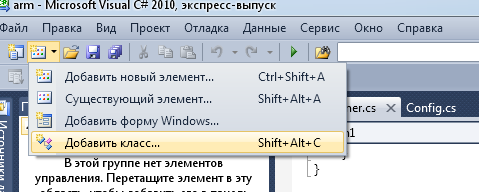


Рис. 3. Додавання класів в Visual Studio 2010.

* 1. **Код програми**

Клас UseBD в якому реалізовані методи для роботи з базою даних:

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Data.OleDb;

using System.Data;

using System.Data.Common;

namespace lab1

{

class UseDB

{

OleDbConnection conn;

DataSet ds = new DataSet();

DbCommand comm;

public UseDB(string conString)

{

conn = new OleDbConnection(conString);

}

//метод підєднання до БД

public void OpenCon()

{

conn.Open();

}

//Метод визначення кількості записів в БД

public int Cout()

{

int count;

comm = conn.CreateCommand();

comm.CommandText = "Select count(\*) from books";

count = (int)comm.ExecuteScalar();

return count;

}

//Метод який вибирає всі елементи з БД

public DataTable AllBooks()

{

DbDataReader dr;

DataTable temp = new DataTable();

comm = conn.CreateCommand();

comm.CommandText = "Select \* from books";

dr = comm.ExecuteReader();

temp.Load(dr);

return temp;

}

//метод пошуку елементів по ІД

public DataTable SearchByID(int id)

{

DbDataReader dr;

DataTable temp = new DataTable();

comm = conn.CreateCommand();

comm.CommandText = "Select \* from books where id="+id;

dr = comm.ExecuteReader();

temp.Load(dr);

return temp;

}

//Метод запису елементів в БД

public int InsertBook(Book tempBook)

{

comm = conn.CreateCommand();

//INSERT IN TO DATABASE

comm.CommandText = "INSERT INTO books (Autor, Name, Vudannj, Rik)VALUES ('"+tempBook.autor+"','"+tempBook.Name+"','"+tempBook.Vudannj+"',"+tempBook.rik+");";

int rows = comm.ExecuteNonQuery();

return rows;

}

//метод знищення елементу по ІД

public int DeleteBook(int id)

{

comm = conn.CreateCommand();

//Delete IN TO DATABASE

comm.CommandText = "DELETE FROM books WHERE id="+id;

int rows = comm.ExecuteNonQuery();

return rows;

}

//метод Відєднання від БД

public void CloseCon()

{

conn.Close();

}

}

}

Клас Program – головний:

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Data;

namespace lab1

{

class Program

{

static void Main(string[] args)

{

int id;

Book myBook=new Book();

DataTable table = new DataTable();

//Стрічка підєднання до БД Access

string conString = "Provider=Microsoft.ACE.OLEDB.12.0;Data Source=lab2.accdb;Persist Security Info=False;";

//Підєднання до БД

UseDB uDB = new UseDB(conString);

uDB.OpenCon();

//Вивід кількості записів у БД

Console.WriteLine("В БД "+uDB.Cout()+" записів\n\n");

//Вивід всіх записів з БД

Console.WriteLine("| ID | Autor | Name | Vudannj | Rik |\n");

Console.WriteLine("|===========================================================================|\n");

table = uDB.AllBooks();

foreach (DataRow dataRow in table.Rows)

{

Console.WriteLine("| "+String.Format("{0,-7}", dataRow[0].ToString()) + "| " +String.Format("{0,-15}", dataRow[1]) + "| " + String.Format("{0,-28}", dataRow[2]) + "| " + String.Format("{0,-10}",dataRow[3])+ "| " + String.Format("{0,-7}",dataRow[4])+"|");

}

//Пошук літератури по ID

Console.WriteLine("\n\n Пошук по ID\n Введіть ID:");

id =Convert.ToInt16(Console.ReadLine());

table = uDB.SearchByID(id);

foreach (DataRow dataRow in table.Rows)

{

Console.WriteLine(String.Format("{0,-5}", dataRow[0].ToString()) + String.Format("{0,-15}", dataRow[1]) + String.Format("{0,-28}", dataRow[2]) + String.Format("{0,-10}", dataRow[3]) + String.Format("{0,-7}", dataRow[4]) );

}

//Запис в БД

Console.WriteLine("\n\nЗапис в БД\nВведфть ім'я Автора:");

myBook.autor = Console.ReadLine();

Console.WriteLine("Введіть назву книжки:");

myBook.Name = Console.ReadLine();

Console.WriteLine("Введіть Видання:");

myBook.Vudannj = Console.ReadLine();

Console.WriteLine("Введіть рік видання:");

myBook.rik =Convert.ToInt16(Console.ReadLine());

uDB.InsertBook(myBook);

//Вивід кількості записів в БД

Console.WriteLine("В БД " + uDB.Cout() + " записів\n\n");

//Вивід всіх записів з БД

Console.WriteLine("| ID | Autor | Name | Vudannj | Rik |\n"); Console.WriteLine("|===========================================================================|\n");

table = uDB.AllBooks();

foreach (DataRow dataRow in table.Rows)

{

Console.WriteLine("| " + String.Format("{0,-7}", dataRow[0].ToString()) + "| " + String.Format("{0,-15}", dataRow[1]) + "| " + String.Format("{0,-28}", dataRow[2]) + "| " + String.Format("{0,-10}", dataRow[3]) + "| " + String.Format("{0,-7}", dataRow[4]) + "|");

}

//Пошук літератури по ID

Console.WriteLine("\n\n Знищення по ID\n Введіть ID:");

id = Convert.ToInt16(Console.ReadLine());

uDB.DeleteBook(id);

//Вивід всіх записів з БД

Console.WriteLine("| ID | Autor | Name | Vudannj | Rik |\n"); Console.WriteLine("|===========================================================================|\n");

table = uDB.AllBooks();

foreach (DataRow dataRow in table.Rows)

{

Console.WriteLine("| " + String.Format("{0,-7}", dataRow[0].ToString()) + "| " + String.Format("{0,-15}", dataRow[1]) + "| " + String.Format("{0,-28}", dataRow[2]) + "| " + String.Format("{0,-10}", dataRow[3]) + "| " + String.Format("{0,-7}", dataRow[4]) + "|");

}

uDB.CloseCon();

Console.ReadKey();

}

}

}

Клас Book відображає в інформаційній системі інформацію про об’єкт, в нашому випадку про книжку:

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

namespace lab1

{

class Book

{

public int id;

public String autor;

public String Name;

public String Vudannj;

public int rik;

}

}

* 1. **Результат виконання програми**

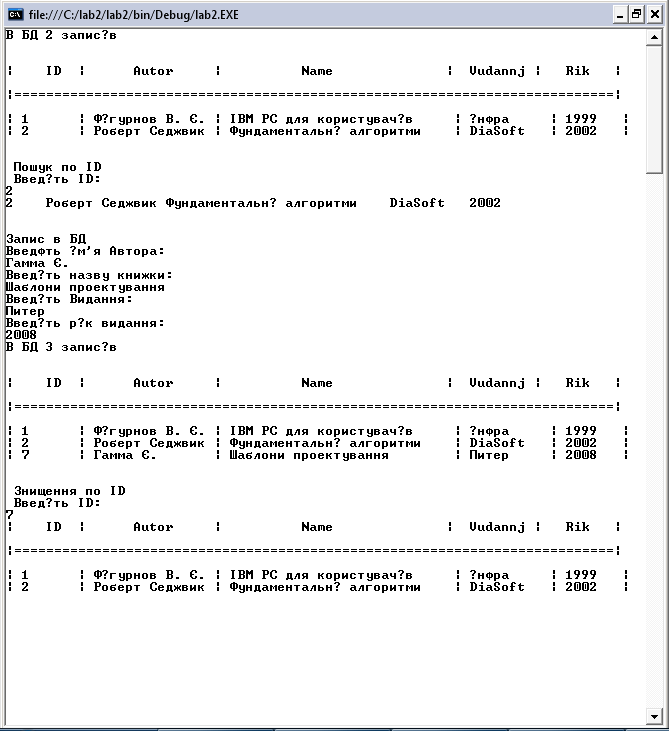
****

Рис. 4. Результат виконання програми.

* 1. **Uml-діаграма класів**

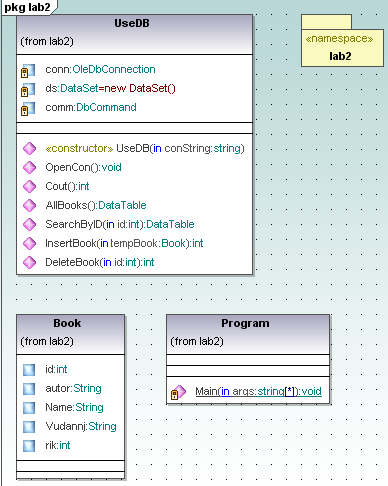


Рис. 5. UML-діаграма класів програми.

* 1. **Хід роботи**

1. Ознайомитися з теоретичними відомостями та прикладом виконання лабораторної роботи.
2. За номером у журналі вибрати власний варіант.
3. Розробити блок-схему програми.
4. Запустити Visual Studio 2003/2005/2008. Вибрати в меню **Файл/Создать/Проект,** у вікні, що відкриється вибрати тип проекту (рисунок 6), шаблон консольної програми і ввести назву, яка відповідає прізвищу студента, його варіанту і номеру лабораторної роботи.

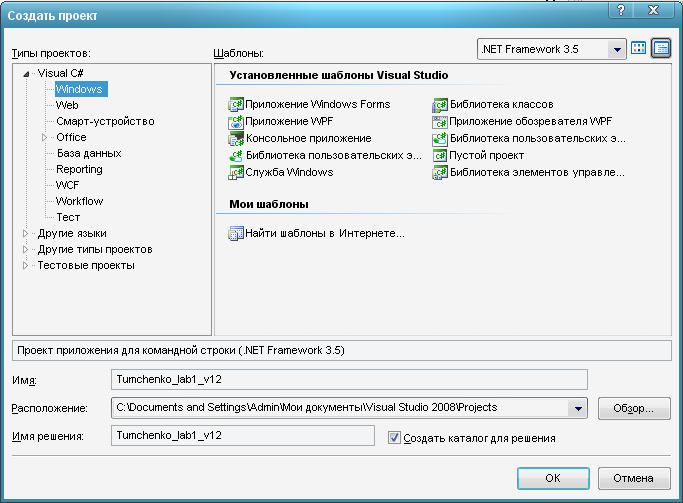


Рис. 6. Діалогове вікно створення проектів у Visual Studio 2010

1. Додати в проект новий клас.
2. Створити БД в Microsoft Access.
3. Реалізувати програму.
4. Розробити звіт.
5. Захистити лабораторну роботу.

# Варіанти завдань до лабораторних робіт

|  |  |  |
| --- | --- | --- |
| Варіант | Назва інформаційної БД | Поля |
| 1 | Автомобілі | Id, марка автомобіля, рік випуску, тип двигуна, об’єм двигуна, розхід палива на 100 км. |
| 2 | Комп’ютери | Id, материнська плата, ОП, тип ОП, об’єм вінчестера, відеокарта |
| 3 | Кондиціонери | Id, назва, площа кондиціонування, потужність охолодження, потужність обігріву, колір |
| 4 | Фільми | Id, назва, жанр, кіностудія, якість, тривалість |
| 5 | Музичні альбоми | Id, виконавець, назва альбому, жанр, кількість пісень, тривалість |
| 6 | Телевізори | Id, виробник, діагональ, тип телевізора, потужність |
| 7 | Носії інформації  (Flach) | Id, фірма, об’єм, тип, швидкість запису, швидкість читання |
| 8 | Комп’ютерні ігри | Id, назва, жанр, необхідна операційна система, мова інтерфейсу, розробник |
| 9 | Принтери | Id, фірма, модель, тип, швидкість друку, резерв сторінок |
| 10 | Мобільні телефони | Id, модель, дисплей, карта пам’яті, вага, колір |
| 11 | Підприємства | Id, назва, сфера діяльності, адреса, № телефону, кількість працівників |
| 12 | Планшетні ПК | Id, процесор, ОП, пам'ять, дисплей, вага, колір |
| 13 | Програмне забезпечення | Id, назва, версія, фірма виготовлення, призначення, платформа |
| 14 | Фотоапарати | Id, назва, матриця, зум, дисплей, вага |
| 15 | Велосипеди | Id, назва, тип, розмір рами, кількість швидкостей, діаметр коліс |
| 16 | Літаки | Id, назва, тип, швидкість, пасажироміскість, дальність польоту |
| 17 | Локомотиви | Id, назва, тип, швидкість, потужність, робоча маса |
| 18 | Готелі | Id, назва, кількість зірок, ціна за добу, адреса |
| 19 | Зброя | Id, назва, тип, калібр, вага, термін гарантії |
| 20 | Житло | Id, район, кількість кімнат, ціна за квадратний метр, площа |
| 21 | Ресторани | Id, назва, кухня, адреса, години роботи, кількість працівників |
| 22 | Футбол | Id, назва команди, країна, група, ліга, очки |
| 23 | Посуд | Id, тип, матеріал, об’єм, виробник, ціна |
| 24 | Канцтовари | Id, назва канцтоварів, група канцтоварів, ціна, виробник, |
| 25 | Косметика | Id, назва косметичного товару, ціна, фірма, вид косметики |
| 26 | Гелікоптери | Id, назва, тип, швидкість, пасажироміскість, дальність польоту |
| 27 | Відпочинкові бази | Id, назва бази, місто, ціна кімнати за добу, відстань до моря, |
| 28 | Навчальні заклади | Id, назва, рівень акредитації, кількість працівників, кількість учнів |
| 29 | Напої | Id, вид, назва, об’єм, ціна, інгредієнти |
| 30 | Взуття | Id, тип, назва, ціна, виробник, матеріал, колір |