МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

ГЛАВНОЕ УПРАВЛЕНИЕ ПО ОБРАЗОВАНИЮ

МОГИЛЕВСКОГО ОБЛАСТНОГО ИСПОЛНИТЕЛЬНОГО КОМИТЕТА

УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ

«МОГИЛЕВСКИЙ ГОСУДАРСТВЕННЫЙ ПОЛИТЕХНИЧЕСКИЙ КОЛЛЕДЖ»

**БАЗЫ ДАННЫХ И**

**СИСТЕМЫ УПРАВЛЕНИЯ**

**БАЗАМИ ДАННЫХ**

ДОМАШНЯЯ КОНТРОЛЬНАЯ РАБОТА №2

Выполнил Учащийся группы ПО-455

О.И. Любаль

Шифр 14

2023

**16 Опишите операторы цикла.**

Циклы являются управляющими конструкциями, позволяя в зависимости от определенных условий выполнять некоторое действие множество раз. В C# имеются следующие виды циклов:

* for
* foreach
* while
* do...while

**Цикл for**

Цикл for имеет следующее формальное определение:

|  |  |
| --- | --- |
|  | for ([действия\_до\_выполнения\_цикла]; [условие]; [действия\_после\_выполнения])  {      // действия  } |

Объявление цикла for состоит из трех частей. Первая часть объявления цикла - некоторые действия, которые выполняются один раз до выполнения цикла. Обычно здесь определяются переменные, которые будут использоваться в цикле.

Вторая часть - условие, при котором будет выполняться цикл. Пока условие равно true, будет выполняться цикл.

И третья часть - некоторые действия, которые выполняются после завершения блока цикла. Эти действия выполняются каждый раз при завершении блока цикла.

После объявления цикла в фигурных скобках помещаются сами действия цикла.

Рассмотрим стандартный цикл for:

|  |  |
| --- | --- |
|  | for (int i = 1; i < 4; i++)  {      Console.WriteLine(i);  } |

Здесь первая часть объявления цикла - int i = 1 - создает и инициализирует переменную i.

Вторая часть - условие i < 4. То есть пока переменная i меньше 4, будет выполняться цикл.

И третья часть - действия, выполняемые после завершения действий из блока цикла - увеличение переменной i на единицу.

Весь процесс цикла можно представить следующим образом:

1. Определяется переменная int i = 1
2. Проверяется условие i < 4. Оно истинно (так как 1 меньше 4), поэтому выполняется блок цикла, а именно инструкция Console.WriteLine(i), которая выводит на консоль значение переменной i
3. Блок цикла закончил выполнение, поэтому выполняется треться часть объявления цикла - i++. После этого переменная i будет равна 2.
4. Снова проверяется условие i < 4. Оно истинно (так как 2 меньше 4), поэтому опять выполняется блок цикла - Console.WriteLine(i)
5. Блок цикла закончил выполнение, поэтому снова выполняется выражение i++. После этого переменная i будет равна 3.
6. Снова проверяется условие i < 4. Оно истинно (так как 3 меньше 4), поэтому опять выполняется блок цикла - Console.WriteLine(i)
7. Блок цикла закончил выполнение, поэтому снова выполняется выражение i++. После этого переменная i будет равна 4.
8. Снова проверяется условие i < 4. Теперь оно возвражает false, так как значение переменной i НЕ меньше 4, поэтому цикл завершает выполнение. Далее уже выполняется остальная часть программы, которая идет после цикла

В итоге блок цикла сработает 3 раза, пока значение i не станет равным 4. И каждый раз это значение будет увеличиваться на 1. Однократное выполнение блока цикла называется итерацией. Таким образом, здесь цикл выполнит три итерации. Результат работы программы:

1

2

3

Если блок цикла for содержит одну инструкцию, то мы можем его сократить, убрав фигурные свобки:

|  |  |
| --- | --- |
|  | for (int i = 1; i < 4; i++)      Console.WriteLine(i);    // или так  for (int i = 1; i < 4; i++) Console.WriteLine(i); |

При этом необязательно именно в первой части цикла объявлять переменную, а в третий части изменять ее значение - это могут быть любые действия. Например:

|  |  |
| --- | --- |
|  | var i = 1;    for (Console.WriteLine("Начало выполнения цикла"); i < 4; Console.WriteLine($"i = {i}"))  {      i++;  } |

Здесь опять же цикл срабатывает, пока переменная i меньше 4, только приращение переменной i происходит в блоке цикла. Консольный вывод данной программы:

Начало выполнения цикла

i = 2

i = 3

i = 4

Нам необязательно указывать все условия при объявлении цикла. Например, мы можем написать так:

|  |  |
| --- | --- |
|  | int i = 1;  for (; ;)  {      Console.WriteLine($"i = {i}");      i++;  } |

Формально определение цикла осталось тем же, только теперь блоки в определении у нас пустые: for (; ;). У нас нет инициализированной переменной, нет условия, поэтому цикл будет работать вечно - бесконечный цикл.

Мы также можем опустить ряд блоков:

|  |  |
| --- | --- |
|  | int i = 1;  for (; i<4;)  {      Console.WriteLine($"i = {i}");      i++;  } |

Этот пример по сути эквивалентен первому примеру: у нас также есть переменная-счетчик, только определена она вне цикла. У нас есть условие выполнения цикла. И есть приращение переменной уже в самом блоке for.

Также стоит отметить, что можно определять несколько переменных в объявлении цикла:

|  |  |
| --- | --- |
|  | for (int i = 1, j = 1; i < 10; i++, j++)      Console.WriteLine($"{i \* j}"); |

Здесь в первой части объявления цикла определяются две переменных: i и j. Цикл выполняется, пока i не будет равна 10. После каждой итерации переменые i и j увеличиваются на единицу. Консольный вывод программы:

1

4

9

16

25

36

49

64

81

**Цикл do..while**

В цикле do сначала выполняется код цикла, а потом происходит проверка условия в инструкции while. И пока это условие истинно, цикл повторяется.

|  |  |
| --- | --- |
|  | do  {      действия цикла  }  while (условие) |

Например:

|  |  |
| --- | --- |
|  | int i = 6;  do  {      Console.WriteLine(i);      i--;  }  while (i > 0); |

Здесь код цикла сработает 6 раз, пока i не станет равным нулю. Но важно отметить, что цикл do гарантирует хотя бы единократное выполнение действий, даже если условие в инструкции while не будет истинно. То есть мы можем написать:

|  |  |
| --- | --- |
|  | int i = -1;  do  {      Console.WriteLine(i);      i--;  }  while (i > 0); |

Хотя у нас переменная i меньше 0, цикл все равно один раз выполнится.

**Цикл while**

В отличие от цикла do цикл while сразу проверяет истинность некоторого условия, и если условие истинно, то код цикла выполняется:

|  |  |
| --- | --- |
|  | while (условие)  {      действия цикла  } |

Например:

|  |  |
| --- | --- |
|  | int i = 6;  while (i > 0)  {      Console.WriteLine(i);      i--;  } |

**Цикл foreach**

Цикл foreach предназначен для перебора набора или коллекции элементов. Его общее определение:

|  |  |
| --- | --- |
|  | foreach(тип\_данных переменная in коллекция)  {      // действия цикла  } |

После оператора foreach в скобках сначала идет определение переменной. Затем ключевое слово in и далее коллекция, элементы которой надо перебрать.

При выполнении цикл последовательно перебирает элементы коллекции и помещает их в переменную, и таким образом в блоке цикла мы можем выполнить с ними некоторые действия.

Например, возьмем строку. Строка по сути - это коллекция символов. И .NET позволяет перебрать все элементы строки - ее символы с помощью цикла foreach.

|  |  |
| --- | --- |
|  | foreach(char c in "Tom")  {      Console.WriteLine(c);  } |

Здесь цикл foreach пробегается по всем символам строки "Tom" и каждый символ помещает в символьную переменную c. В блоке цикла значение переменной c выводится на консоль. Поскольку в строке "Tom" три символа, то цикл выполнится три раза. Консольный вывод программы:

T

o

m

Стоит отметить, что переменная, которая определяется в объявлении цикла, должна по типу соответствовать типу элементов перебираемой коллекции. Так, элементы строки - значения типа char - символы. Поэтому переменная c имеет тип char. Однако в реальности не всегда бывает очевидно, какой тип представляют элементы коллекции. В этом случае мы можем определить переменную с помощью оператора var:

|  |  |
| --- | --- |
|  | foreach(var c in "Tom")  {      Console.WriteLine(c);  } |

В дальнейшем мы подробнее рассмотрим, что представляют собой коллекции в .NET и какие именно коллекции можно перебирать с помощью цикла foreach.

**Операторы continue и break**

Иногда возникает ситуация, когда требуется выйти из цикла, не дожидаясь его завершения. В этом случае мы можем воспользоваться оператором break.

Например:

|  |  |
| --- | --- |
|  | for (int i = 0; i < 9; i++)  {      if (i == 5)          break;      Console.WriteLine(i);  } |

Хотя в условии цикла сказано, что цикл будет выполняться, пока счетчик i не достигнет значения 9, в реальности цикл сработает 5 раз. Так как при достижении счетчиком i значения 5, сработает оператор break, и цикл завершится.

0

1

2

3

4

Теперь поставим себе другую задачу. А что если мы хотим, чтобы при проверке цикл не завершался, а просто пропускал текущую итерацию. Для этого мы можем воспользоваться оператором continue:

|  |  |
| --- | --- |
|  | for (int i = 0; i < 9; i++)  {      if (i == 5)          continue;      Console.WriteLine(i);  } |

В этом случае цикл, когда дойдет до числа 5, которое не удовлетворяет условию проверки, просто пропустит это число и перейдет к следующей итерации:

0

1

2

3

4

6

7

8

Стоит отметить, что операторы break и continue можно применять в любом типе циклов.

**Вложенные циклы**

Одни циклы могут быть вложенными в другие. Например:

|  |  |
| --- | --- |
|  | for (int i = 1; i < 10; i++)  {      for (int j = 1; j < 10; j++)      {          Console.Write($"{i \* j} \t");      }      Console.WriteLine();  } |

В данном случае цикл for (int i = 1; i < 10; i++) выполняется 9 раз, то есть имеет 9 итераций. Но в рамках каждой итерации выполняется девять раз вложенный цикл for (int j = 1; j < 10; j++). В итоге данная программа выведет таблицу умножения.

**32 Изложите процесс подключения клиент-серверной БД к среде Delphi.**

**Delphi** поставляется с более чем 40 готовых компонентов базы данных. Обеспечивает визуальную среду программирования, которая включает в себя встроенный редактор кода, базы данных форм мастер, который ускоряет шаги по созданию доступных для просмотра данных формы. Для облегчения работы используется также модуль конструктор, который можно использовать для обмена данными между несколькими формами.

Некоторые другие базы данных специализированных инструментов также предоставляется в Delphi, чтобы помочь программировать быстрее и легче.

Страницы доступа к данным компоненты Палитра содержит компоненты, которые используются для подключения к источнику данных. На странице **Data Controls**, даны компоненты, которые (после подключения к базе данных) можно использовать — извлечь и отправить данные в базу или из базы данных. Компоненты на вкладке ADO ActiveX Data Objects используются для доступа к информации из базы данных через OLEDB. Компоненты на странице доступа к базе данных InterBase используются напрямую.

В Дельфи, мы можем подключиться к различным типам баз данных: локальная или клиент-сервер (удаленный сервер) базы данных. Местные базы данных хранятся на вашем локальном диске или в локальной сети. Серверы удаленного базы данных обычно находятся на удаленном компьютере. Виды местных баз данных Paradox, DBase и MS Access. Виды клиент / сервер баз данных MS SQL Server, Interbase или Oracle.

Компонент базы данных типа **Database** автоматически включается в любое приложении **Delphi**, работающее с базами данных при помощи BDE. Компонент решает следующие задачи:

* Создание соединения с базами данных;
* Регистрация пользователя при первом обращении к серверу;
* Создание локальных псевдонимов приложений;
* Управление транзакциями;
* Регулирование одновременных транзакций к одним и тем же таблицам.

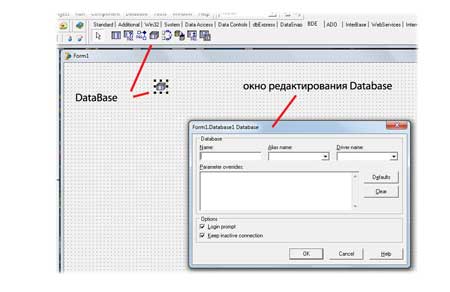
[](http://delphi-box.ru/wp-content/uploads/2013/04/16.jpg)

Рисунок 1. Подключение к базе данных.

**Database** связывается с компонентами набором данных **Table, Query** и другими через имя базы данных, к которой он подключается. Это имя задается в свойстве **DataBaseName**. Можно задать псевдоним базы данных или полный путь к ней. При задавании псевдонима **BDE**, можно опустить свойства **AliasName, DriveName и Params**.

Свойство **Connected** совместно с **KeepConnection** управляет процессом соединения компонентов с базой. При значении **true** соединение с базой постоянное даже при отсутствии наборов данных. Если же **false**, то при каждом обращении к таблице нужно устанавливать **Connected** в **true**.

**Свойства Database (Properties Database)**

**AliasName —** псевдоним BDE базы данных.

**Connected —** указывает активно ли соединение с базой.

**DatabaseName** **—** указывает имя базы данных, с которой связан компонент.

**DataSetCount –** число активных наборов данных, связанных с компонентом.

**DataSets** – индексированный массив активных наборов данных.

**Directory** – рабочий каталог для баз Paradox и qBase.

**DriverName** – имя используемого драйвера BDE.

**Exclusive** – запрещает или разрешает доступ к базе данных другими приложениями.

**Locale** – драйвер языка BDE. Только для чтения.

**LoginPrompt** – будет ли появляться окно ввода имени пользователя и пароля при открытии базы данных.

**Session** – указывает компонент Session.

**SessionName** – имя компонента сеанса сетевого соединения Session.

**ReadOnly** – соответственно, только для чтения. Значения **Boolean**.

**Основные методы**

**ApplyUpdates** –перенос кэшированных данных в базу.

**Close** – закрывает соединение.

**CloseDataSets** – закрывает все наборы данных.

**Commit** – фиксирует в базе данных все изменения, произведенные транзакцией, и завершает транзакцию.

**Open** – открывает соединение.

**RollBack** – откат назад, анализируя все изменения.

**StartTransaction** – начинает новую транзакцию.

**События на вкладке (Events):**

**OnLogin** – наступает при соединении приложения с базой.

**AfterConnect** – после соединения.

**AfterDisconnect** – после закрытия соединения.

**BeforeConnect** – перед установкой соединения.

**BeforeDisconnect** – перед закрытием соединения.

**76 «Салон автомобилей»**

1) Для создания информационной системы спроектируйте базу данных в Microsoft SQL Server по № варианта (задание 61-90), содержащую не менее трех таблиц. Установите отношения между таблицами.

2) Заполните таблицы данными (не менее 10 записей).

3) На языке SQL напишите запросы:

- на вывод некоторых полей из двух таблиц;

- на вывод данных по условию, представляющему выражение:

* 1. типа сравнения;
  2. с логическим оператором «И»;
  3. с логическим оператором «ИЛИ»;

- с вычислениями над полями БД;

- параметрический.

4) Разработать форму на языке C#, содержащую все созданные объекты БД.

5) Реализовать добавление, удаление изменение данных в таблице, а также поиск и фильтрацию данных.

**Решение**

Создадим форму и разместим на ней компонент TabControl в котором разместим вкладки, по одной для каждой таблицы и итоговую для выполнения заданий. Внешний вид формы представлен на рисунке 2. На каждой из них разместим соответствующие элементы управления: поля ввода данных и кнопки для добавления, редактирования и удаления записи, а на итоговой вкладке кнопки для запуска заданий.

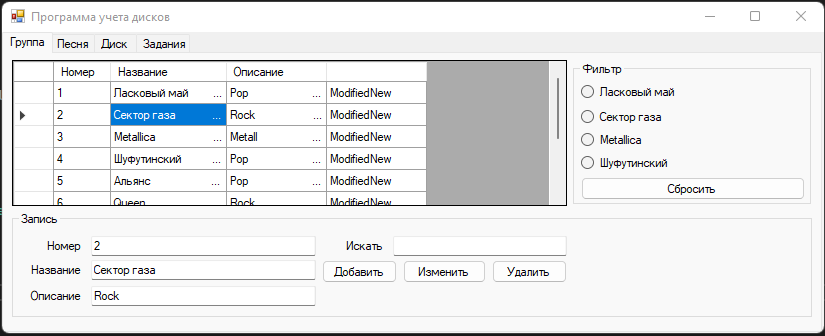


Рисунок 2 Внешний вид основной формы

Для примера добавления строки в таблицу внешней формой создадим вторую форму и разместим на ней элементы управления (рисунок 3)

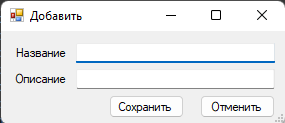


Рисунок 3 Внешний вид формы 2

Заполним таблицы данными, у нас это будут таблицы: Диск, Песня и Группа. Внешний вид приложения со вкладками Песня и Диск представлен на рисунках 4 и 5.

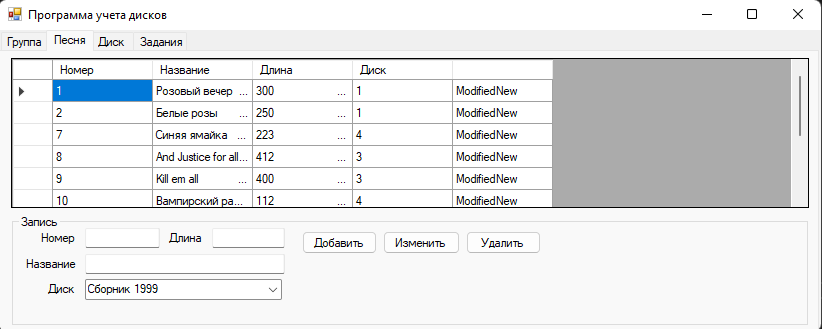


Рисунок 4 Приложение вкладка Песня

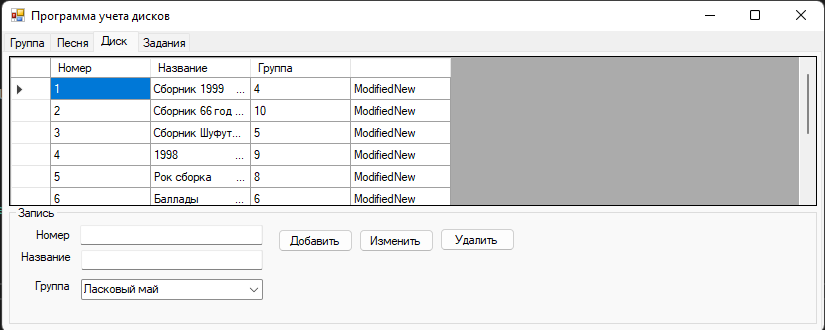


Рисунок 5 Приложение, вкладка Диск

На рисунке 6 представлен внешний вид приложения с открытой вкладкой для выполнения заданий. Выполнено задание вывода данных из двух таблиц.

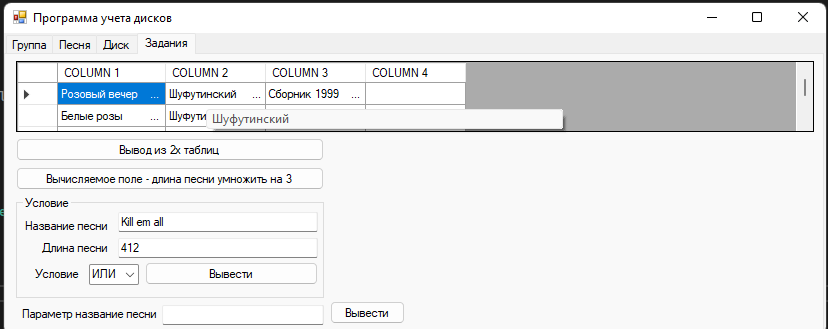


Рисунок 6 Приложение вкладка выполнения заданий

На рисунке 7 представлен результат выполнения действий над полями таблицы, длина песни умножается на 3 и выводится в третьем столбце.

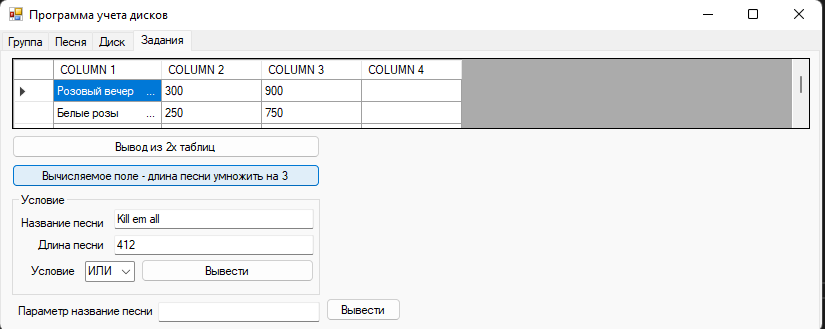


Рисунок 7 Результат выполнения задания вычисляемых полей

На рисунке 8 представлен вид приложения с выполненным заданием выбора песни по названию, либо по длине.

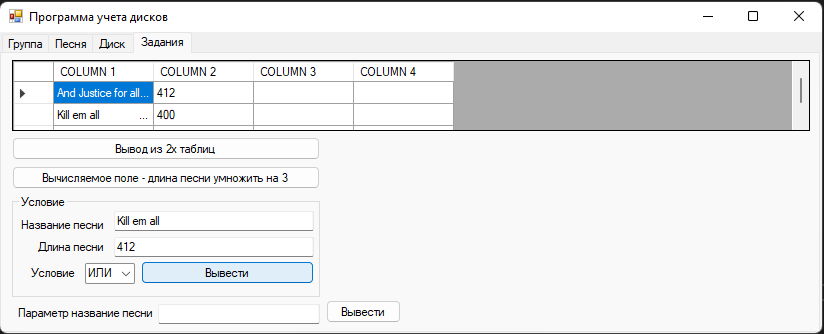


Рисунок 8 Результат выполнения задания Условие

На рисунке 9 представлен вывод задания с параметром, в качестве параметра название песни

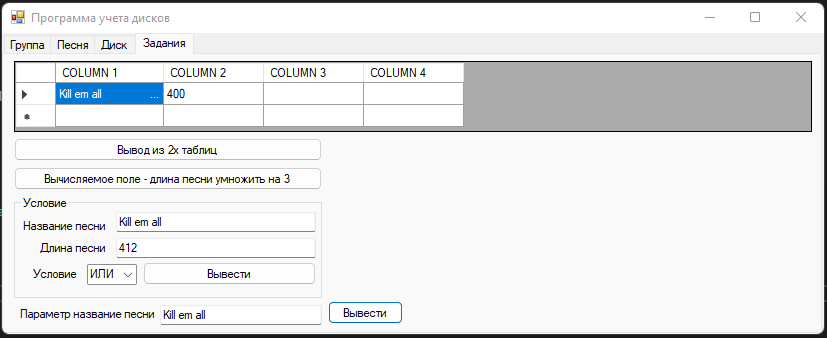
****

Рисунок 9 Результат выполнения задания Параметр

На рисунке 10 представлен результат выполнения задания Фильтр

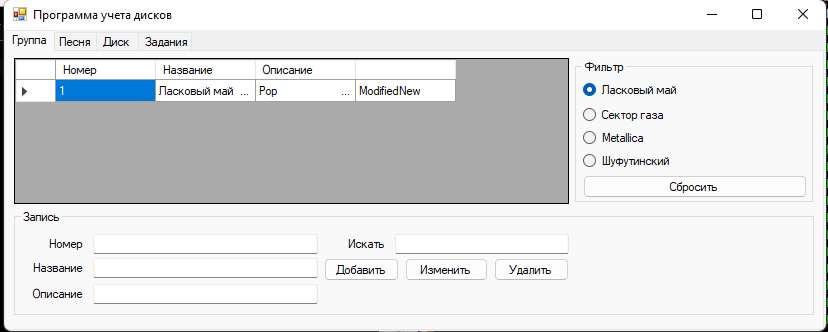
****

Рисунок 10результат выполнения задания Фильтр

На рисунке 11 представлен результат выполнения задания Поиск

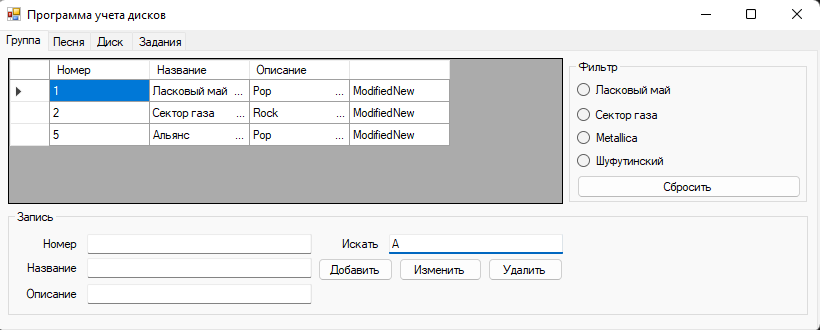


Рисунок 11результат выполнения задания Поиск

**Код программы, Форма 1**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Data.SqlClient;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using static System.Windows.Forms.VisualStyles.VisualStyleElement;

namespace Avto

{

enum rowState

{

Existed,

New,

Modified,

ModifiedNew,

Deleted

}

public partial class Form1 : Form

{

DataBase database = new DataBase();

int selectedRow;

public Form1()

{

InitializeComponent();

CreateColumns();

refreshDG(dataGridView1);

refreshDG2(dataGridView2);

refreshDG3(dataGridView3);

refreshDG4(dataGridView4,$"select '1' as Col1, '2' as col2, '3' as col3, '4' as col4 from [Park]");

}

private void Form1\_Load(object sender, EventArgs e)

{

CreateColumns();

refreshDG(dataGridView1);

refreshDG2(dataGridView2);

refreshDG3(dataGridView3);

//refreshDG4(dataGridView4,$"select '1' as Col1, '2' as col2, '3' as col3, '4' as col4 from [Park]");

}

private void CreateColumns()

{

dataGridView1.Columns.Add("ID", "Номер");

dataGridView1.Columns.Add("Name", "Название");

dataGridView1.Columns.Add("Adr", "Адрес");

dataGridView1.Columns.Add("IsNew", String.Empty);

dataGridView2.Columns.Add("ID", "Номер");

dataGridView2.Columns.Add("FIO", "ФИО");

dataGridView2.Columns.Add("Age", "Возраст");

dataGridView2.Columns.Add("IDAvto", "Машина");

dataGridView2.Columns.Add("IsNew", String.Empty);

dataGridView3.Columns.Add("ID", "Номер");

dataGridView3.Columns.Add("Name", "Название");

dataGridView3.Columns.Add("IDPark", "Парк");

dataGridView3.Columns.Add("IsNew", String.Empty);

dataGridView4.Columns.Add("Col1", String.Empty);

dataGridView4.Columns.Add("Col2", String.Empty);

dataGridView4.Columns.Add("Col3", String.Empty);

dataGridView4.Columns.Add("Col4", String.Empty);

}

private void ReadSinglRow(DataGridView dgw, IDataRecord record)

{

dgw.Rows.Add(record.GetInt32(0), record.GetString(1), record.GetString(2), rowState.ModifiedNew);

}

private void ReadSinglRow2(DataGridView dgw, IDataRecord record)

{

dgw.Rows.Add(record.GetInt32(0), record.GetString(1), record.GetString(2), record.GetInt32(3), rowState.ModifiedNew);

}

private void ReadSinglRow3(DataGridView dgw, IDataRecord record)

{

dgw.Rows.Add(record.GetInt32(0), record.GetString(1), record.GetInt32(2), rowState.ModifiedNew);

}

private void ReadSinglRow4(DataGridView dgw, IDataRecord record)

{

dgw.Rows.Add(record.GetString(0), record.GetString(1), record.GetString(2), record.GetString(3));

}

private void refreshDG(DataGridView dgw)

{

dgw.Rows.Clear();

string querystr = $"Select \* from [Park]";

SqlCommand command = new SqlCommand(querystr, database.getConnection());

database.openConnection();

SqlDataReader reader = command.ExecuteReader();

while (reader.Read())

{

ReadSinglRow(dgw, reader);

}

reader.Close();

}

private void refreshDG2(DataGridView dgw)

{

dgw.Rows.Clear();

string querystr = $"Select \* from [Voditel]";

SqlCommand command = new SqlCommand(querystr, database.getConnection());

database.openConnection();

SqlDataReader reader = command.ExecuteReader();

while (reader.Read())

{

ReadSinglRow2(dgw, reader);

}

reader.Close();

}

private void refreshDG3(DataGridView dgw)

{

dgw.Rows.Clear();

string querystr = $"Select \* from [Avto]";

SqlCommand command = new SqlCommand(querystr, database.getConnection());

database.openConnection();

SqlDataReader reader = command.ExecuteReader();

while (reader.Read())

{

ReadSinglRow3(dgw, reader);

}

reader.Close();

}

private void refreshDG4(DataGridView dgw, string s)

{

dgw.Rows.Clear();

string querystr = s;

SqlCommand command = new SqlCommand(querystr, database.getConnection());

database.openConnection();

SqlDataReader reader = command.ExecuteReader();

while (reader.Read())

{

ReadSinglRow4(dgw, reader);

}

reader.Close();

}

private void button1\_Click(object sender, EventArgs e)

{

}

private void button2\_Click(object sender, EventArgs e)

{

}

private void dataGridView1\_CellContentClick(object sender, DataGridViewCellEventArgs e)

{

}

private void dataGridView1\_CellClick(object sender, DataGridViewCellEventArgs e)

{

selectedRow = e.RowIndex;

if (e.RowIndex >= 0)

{

DataGridViewRow row = dataGridView1.Rows[selectedRow];

textBox1.Text = row.Cells[0].Value.ToString();

textBox2.Text = row.Cells[1].Value.ToString();

textBox3.Text = row.Cells[2].Value.ToString();

}

}

private void button3\_Click(object sender, EventArgs e)

{

Form2 f\_add = new Form2();

f\_add.ShowDialog();

refreshDG(dataGridView1);

}

private void textBox4\_TextChanged(object sender, EventArgs e)

{

DataGridView dwg = dataGridView1;

dwg.Rows.Clear();

string querystr = $" select \* from [Park] Where concat(Name,Adr) like '%" + textBox4.Text + "%'";

SqlCommand com = new SqlCommand(querystr, database.getConnection());

database.openConnection();

SqlDataReader reader = com.ExecuteReader();

while (reader.Read())

{

ReadSinglRow(dwg, reader);

}

reader.Close();

}

private void radioButton1\_CheckedChanged(object sender, EventArgs e)

{

}

private void radioButton1\_Click(object sender, EventArgs e)

{

DataGridView dwg = dataGridView1;

dwg.Rows.Clear();

string querystr = $" select \* from [Park] Where Name = 'Автобаза №4'";

SqlCommand com = new SqlCommand(querystr, database.getConnection());

database.openConnection();

SqlDataReader reader = com.ExecuteReader();

while (reader.Read())

{

ReadSinglRow(dwg, reader);

}

reader.Close();

}

private void button4\_Click(object sender, EventArgs e)

{

refreshDG(dataGridView1);

}

private void button3\_Click\_1(object sender, EventArgs e)

{

Form2 f\_add = new Form2();

f\_add.ShowDialog();

refreshDG(dataGridView1);

}

private void dataGridView1\_CellClick\_1(object sender, DataGridViewCellEventArgs e)

{

selectedRow = e.RowIndex;

if (e.RowIndex >= 0)

{

DataGridViewRow row = dataGridView1.Rows[selectedRow];

textBox1.Text = row.Cells[0].Value.ToString();

textBox2.Text = row.Cells[1].Value.ToString();

textBox3.Text = row.Cells[2].Value.ToString();

}

}

private void radioButton2\_Click(object sender, EventArgs e)

{

DataGridView dwg = dataGridView1;

dwg.Rows.Clear();

string querystr = $" select \* from [Park] Where Name = 'Автопарк №10'";

SqlCommand com = new SqlCommand(querystr, database.getConnection());

database.openConnection();

SqlDataReader reader = com.ExecuteReader();

while (reader.Read())

{

ReadSinglRow(dwg, reader);

}

reader.Close();

}

private void radioButton4\_Click(object sender, EventArgs e)

{

DataGridView dwg = dataGridView1;

dwg.Rows.Clear();

string querystr = $" select \* from [Park] Where Name = 'Городской автопарк №1'";

SqlCommand com = new SqlCommand(querystr, database.getConnection());

database.openConnection();

SqlDataReader reader = com.ExecuteReader();

while (reader.Read())

{

ReadSinglRow(dwg, reader);

}

reader.Close();

}

private void radioButton3\_Click(object sender, EventArgs e)

{

DataGridView dwg = dataGridView1;

dwg.Rows.Clear();

string querystr = $" select \* from [Park] Where Name = 'Автобусный парк №2'";

SqlCommand com = new SqlCommand(querystr, database.getConnection());

database.openConnection();

SqlDataReader reader = com.ExecuteReader();

while (reader.Read())

{

ReadSinglRow(dwg, reader);

}

reader.Close();

}

private void button2\_Click\_1(object sender, EventArgs e)

{

var addQuery = $"delete from [Park] where ID={textBox1.Text}";

var command = new SqlCommand(addQuery, database.getConnection());

command.ExecuteNonQuery();

refreshDG(dataGridView1);

}

private void button1\_Click\_1(object sender, EventArgs e)

{

var addQuery = $"update [Park] set Name='{textBox2.Text}' , Adr = '{textBox3.Text}' where ID = {textBox1.Text}";

var command = new SqlCommand(addQuery, database.getConnection());

command.ExecuteNonQuery();

refreshDG(dataGridView1);

}

private void Form1\_Load\_1(object sender, EventArgs e)

{

}

private void dataGridView2\_CellClick(object sender, DataGridViewCellEventArgs e)

{

selectedRow = e.RowIndex;

if (e.RowIndex >= 0)

{

DataGridViewRow row = dataGridView2.Rows[selectedRow];

textBox5.Text = row.Cells[0].Value.ToString();

textBox6.Text = row.Cells[1].Value.ToString();

textBox7.Text = row.Cells[2].Value.ToString();

comboBox1.SelectedIndex = Int32.Parse(row.Cells[3].Value.ToString())-1;

}

}

private void button5\_Click(object sender, EventArgs e)

{

var addQuery = $"insert into [Voditel] (Fio,Age, IdAvto) values ('{textBox6.Text}' , '{textBox7.Text}',{comboBox1.SelectedIndex+1})";

var command = new SqlCommand(addQuery, database.getConnection());

command.ExecuteNonQuery();

refreshDG2(dataGridView2);

}

private void button7\_Click(object sender, EventArgs e)

{

var addQuery = $"delete from [Voditel] where ID={textBox5.Text}";

var command = new SqlCommand(addQuery, database.getConnection());

command.ExecuteNonQuery();

refreshDG2(dataGridView2);

}

private void dataGridView3\_CellClick(object sender, DataGridViewCellEventArgs e)

{

selectedRow = e.RowIndex;

if (e.RowIndex >= 0)

{

DataGridViewRow row = dataGridView3.Rows[selectedRow];

textBox8.Text = row.Cells[0].Value.ToString();

textBox9.Text = row.Cells[1].Value.ToString();

comboBox2.SelectedIndex = Int32.Parse(row.Cells[2].Value.ToString())-1;

}

}

private void button8\_Click(object sender, EventArgs e)

{

var addQuery = $"insert into [Avto] (Name,IdPark) values ('{textBox9.Text}' ,{comboBox2.SelectedIndex + 1})";

var command = new SqlCommand(addQuery, database.getConnection());

command.ExecuteNonQuery();

refreshDG3(dataGridView3);

}

private void button10\_Click(object sender, EventArgs e)

{

var addQuery = $"delete from [Avto] where ID={textBox8.Text}";

var command = new SqlCommand(addQuery, database.getConnection());

command.ExecuteNonQuery();

refreshDG3(dataGridView3);

}

private void button6\_Click(object sender, EventArgs e)

{

var addQuery = $"update [Voditel] set Fio='{textBox6.Text}' , Age = '{textBox7.Text}', IdAvto = '{comboBox1.SelectedIndex+1}' where ID = {textBox5.Text}";

var command = new SqlCommand(addQuery, database.getConnection());

command.ExecuteNonQuery();

refreshDG2(dataGridView2);

}

private void button9\_Click(object sender, EventArgs e)

{

var addQuery = $"update [Avto] set Name='{textBox9.Text}' , IdPark = '{comboBox2.SelectedIndex+1}' where ID = {textBox8.Text}";

var command = new SqlCommand(addQuery, database.getConnection());

command.ExecuteNonQuery();

refreshDG3(dataGridView3);

}

private void button11\_Click(object sender, EventArgs e)

{

dataGridView4.Rows.Clear();

string querystr = $"select s.name as col1, g.name as col2, d.name as col3,'' as col4\r\nfrom [Voditel] d, [Avto] s, [Park] g\r\nwhere d.idAvto=s.id and s.idPark=g.id";

SqlCommand command = new SqlCommand(querystr, database.getConnection());

database.openConnection();

SqlDataReader reader = command.ExecuteReader();

while (reader.Read())

{

ReadSinglRow4(dataGridView4, reader);

}

reader.Close();

}

private void button12\_Click(object sender, EventArgs e)

{

dataGridView4.Rows.Clear();

string querystr = $"select FIO as col1, CAST(Age AS varchar(5)) as col2, CAST(Age+20 AS varchar(5)) as col3, '' as col4 from [Voditel]";

SqlCommand command = new SqlCommand(querystr, database.getConnection());

database.openConnection();

SqlDataReader reader = command.ExecuteReader();

while (reader.Read())

{

ReadSinglRow4(dataGridView4, reader);

}

reader.Close();

}

private void button13\_Click(object sender, EventArgs e)

{// условие

dataGridView4.Rows.Clear();

string s = "OR";

if (radioButton5.Checked) {

s = "AND"; }

string n = textBox10.Text;

string l = textBox11.Text;

string querystr = $"select fio as col1, CAST(Age AS varchar(5)) as col2, '' as col3, '' as col4 from [Voditel] s where Fio='{n}' {s} Age='{l}'";

SqlCommand command = new SqlCommand(querystr, database.getConnection());

database.openConnection();

SqlDataReader reader = command.ExecuteReader();

while (reader.Read())

{

ReadSinglRow4(dataGridView4, reader);

}

reader.Close();

}

private void button14\_Click(object sender, EventArgs e)

{

dataGridView4.Rows.Clear();

string n = textBox12.Text;

string querystr = $"select Fio as col1, CAST(Age AS varchar(5)) as col2, '' as col3, '' as col4 from [Voditel] s where Fio='{n}'";

SqlCommand command = new SqlCommand(querystr, database.getConnection());

database.openConnection();

SqlDataReader reader = command.ExecuteReader();

while (reader.Read())

{

ReadSinglRow4(dataGridView4, reader);

}

reader.Close();

}

}

class DataBase

{

SqlConnection sqlConnection = new SqlConnection(@"Data Source=localhost\SQLEXPRESS;Initial Catalog=Avto;Integrated Security=True");

public void openConnection()

{

if (sqlConnection.State == System.Data.ConnectionState.Closed)

{

sqlConnection.Open();

}

}

public void closeConnection()

{

if (sqlConnection.State == System.Data.ConnectionState.Open)

{

sqlConnection.Close();

}

}

public SqlConnection getConnection()

{

return sqlConnection;

}

}

}

**Форма 2**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Data.SqlClient;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using static System.Windows.Forms.VisualStyles.VisualStyleElement;

namespace Avto

{

public partial class Form2 : Form

{

DataBase dataBase = new DataBase();

public Form2()

{

InitializeComponent();

}

private void button1\_Click\_1(object sender, EventArgs e)

{

dataBase.openConnection();

var addQuery = $"insert into [Park] (Name,Adr) values ('{textBox1.Text}' , '{textBox2.Text}')";

var command = new SqlCommand(addQuery, dataBase.getConnection());

command.ExecuteNonQuery();

dataBase.closeConnection();

this.Close();

}

private void button2\_Click\_1(object sender, EventArgs e)

{

this.Close();

}

}

}

**Список использованных источников**

1. Дейт, К.Дж. Введение в системы баз данных/ К.Дж.Дейт. – М.: Вильямс, 2018
2. Дюбуа, Поль. MySQI / Поль Дюбуа. – 3-е изд. – М.: Издательский дом "Вильямс", 2007. - 816 с.: ил.
3. Колисниченко, Д.Н. Профессиональное программирование на PНP / Д.Н. Колисниченко. – Санкт-Петербург: БХВ-Петербург, 2007. –416 с.
4. Лазицкас, Е.А. Базы данных и системы управления базами данных / Е.А. Лазицкас. – Минск:РИПО, 2016
5. Роб, П. Системы баз данных: проектирование, реализация и управление / П.Роб, К.Коронел. – 5-е изд., перераб. и доп.: пер. с англ. – Санкт-Петербург: БХВ-Петербург, 2004. - 1040 с.: ил
6. Фаронов, В.В. Программирование баз данных в Delphi / В.В.Фаронов. – 2-е изд. – СанктПетербург.: Питер, 2004. – 459 с.
7. Хернандес, М.Дж. SQL-запросы для простых смертных: практическое руководство по манипулированию данными в SQL / М.Дж.Хернандес, Дж.Л.Вьескас. – Москва: Лори, 2000. - 473 с.