**ВВЕДЕНИЕ**

В данной пояснительной записке рассматривается описание программного модуля «Справочник лесника» на основе объектно-ориентированного подхода.

При объектно-ориентированном подходе программа представляет собой описание объектов, их свойств (или атрибутов), совокупностей (или классов), отношений между ними, способов их взаимодействия и операций над объектами (или методов).

Несомненным преимуществом данного подхода является концептуальная близость к предметной области произвольной структуры и назначения. Механизм наследования атрибутов и методов позволяет строить производные понятия на основе базовых и таким образом создавать модель сколь угодно сложной предметной области с заданными свойствами.

Еще одним теоретически интересным и практически важным свойством объектно-ориентированного подхода является поддержка механизма обработки событий, которые изменяют атрибуты объектов и моделируют их взаимодействие в предметной области.

В отличие от предыдущих подходов к программированию, объектно-ориентированный подход требует глубокого понимания основных принципов, или, иначе, концепций, на которых он базируется. К числу основополагающих понятий ООП обычно относят абстракцию данных, наследование, инкапсуляцию и полиморфизм.

Объектно-ориентированное программирование в настоящее время является абсолютным лидером в области прикладного программирования.

В качестве основного инструмента разработки применяется Microsoft Visual Studio 2015. Язык программирования C#.

**ЦЕЛЬ И ЗАДАЧИ ВЫПОЛНЕНИЯ РАБОТЫ**

Целью данного курсового проекта является автоматизация деятельности лесничего на основе объектно-ориентированного подхода, а также получение навыков в реализации этого подхода, проектировании и реализации схемы данных.

Справочник лесника.

Вы работаете лесником в национальном парке, вашей задачей является следить за благосостоянием леса. Для автоматизации вашей работы вы приняли решение вести справочник.

В справочнике хранится информация о областях леса с различной флорой, а именно классификация деревьев, площадь зоны, средний возраст деревьев. Вам необходимо проводить требуемые для лучшего развития лесного фонда виды ухода, характеризуемые по следующим атрибутам: Название, описание, государственные затраты. Также в справочник вы ведете планировщик задач, где указываете в какой области леса и каким видом ухода вам нужно заняться в поставленное время.

Задачи выполнения работы:

исследование предметной области с целью выявления основных сущностей и их атрибутов;

проектирование иерархии классов и интерфейсов на основе выделенных сущностей;

использование наследования и агрегации для структуризации классов;

применение принципа инкапсуляции к классам;

использование интерфейсов, описывающих операции в данной предметной области, для обеспечения множественного наследования;

- тестирование созданных классов в клиентском приложении, автоматизирующем деятельность предметной области;

- документация разработанных классов при помощи XML комментариев.

**1 ВНЕШНЕЕ ОПИСАНИЕ**

**1.1 Анализ предметной области**

Анализ предметной области – важнейший этап разработки программного обеспечения.

Анализ работы грузовых перевозок позволяет определить ключевые понятия, выяснить задачи, требующие решения в рамках проекта, проанализировать функциональные требования будущего продукта (набор функций, предоставляемых программой) и нефункциональные (надёжность, эффективность, изучаемость, модифицируемость и т.п.).

Компонентами данной предметной области являются области леса, виды ухода и запланированные задачи. Данное приложение должно позволять пользователю создавать записи, удалять, редактировать, сохранять и открывать файлы создаваемые программой Access.

Нефункциональные требования к программному средству:

- надежность: программа должна быть автономной;

- эффективность: программа должна иметь минимальные требования к аппаратному обеспечению. Для использования данного программного средства не должна требоваться установка на компьютер. Не должно требоваться дополнительных периферийных средств (принтер, сканер, дополнительные дисководы и т.д.)

- изучаемость: программа должна быть интуитивно понятна, иметь удобный пользовательский интерфейс;

- модифицируемость: программа должна быть легко модифицируемой вследствие небольшого исходного размера и объектно-ориентированного подхода.

**1.2. Функциональная спецификация**

Минимальные технические характеристики компьютера, на котором гарантируется стабильная работа программы:

компьютер/процессор: компьютер с процессором класса Pentium II 450 МГц;

память: 64 МБ ОЗУ;

монитор: монитор VGA с разрешением 800x600 точек или более высоким, поддерживающий 256 цветов;

операционная система: операционная система Windows XP с установленным Microsoft .NET Framework 4.5.2;

наличие свободного дискового пространства на жёстком диске.

Внешние функции:

внесение данных и редактирование данных об областях леса, видах ухода и запланированных задачах;

сохранение данных в Access -файл;

считывание данных из Access -файла;

проверка вводимых данных и вывод сообщений об ошибках;

программа должна выполнять все функции за предельно малое количество времени;

Критические случаи:

отсутствие свободного места на жёстком диске. Любая операция с файловой системой должна быть отменена;

некорректный Access -файл. Программа не должна аварийно завершаться;

завершение работы Windows. В этом случае программа должна завершать свою работу, не применяя выбранные пользователем настройки.

**2 ТЕХНИЧЕСКОЕ ЗАДАНИЕ НА АВТОМАТИЗИРОВАННУЮ СИСТЕМУ**

Техническое задание определяет требования к будущей автоматизированной информационной системе со стороны основных процессов.

Система предназначена для решения перечисленных ниже задач:

изменение списка продаж (добавления, удаление, редактирование);

изменение материалов (добавления, удаление, редактирование);

изменение изделий (добавления, удаление, редактирование);

отображение информации, хранящейся в Access –файле;

сохранение информации в Access –файл.

**3 ПРОГРАММИРОВАНИЕ**

**3.1 Описание классов и интерфейсов**

В ООП главным элементом является класс, включающий множество объектов с одинаковыми свойствами, операциями и отношениями. Класс имеет внутреннее (реализацию) и внешнее представление – интерфейс. В данной программе реализованы классы KindCare, Region, Task, а также интерфейс WindowsForm.

Рассмотрим остальные классы:

Класс KidnCare описывает вид ухода за лесом. Содержит такие свойства, как название, описание и государственные затраты.

Класс Region описывает область леса и содержит такие атрибуты и параметры, как его классификация, площадь и средний возраст деревьев.

Класс Task описывает задачи, которые должен выполнить лесник и содержит такие атрибуты, как область леса, вид ухода и дата выполнения задачи.

**3.2 Схема данных**

На рисунке 1 представлена схема данных:

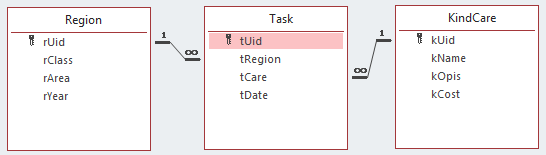


Рисунок 1 – Схема данных

**4 КЛИЕНТСКОЕ ПРИЛОЖЕНИЕ**

**4.1 Структура**

В данном ПП для продуктивной работы реализован удобный пользовательский интерфейс, похожий на большинство Windows – приложений. Данное программное средство можно разбить на модули, взаимодействующие друг с другом. На рисунке 2 представлена структура приложения:

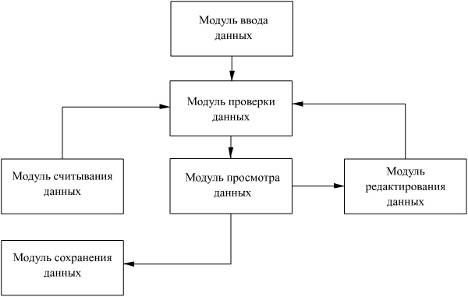


Рисунок 2 – Структура приложения

Модуль просмотра данных предоставляет возможность наглядного отображения данных.

Модуль проверки данных реализует методы:

корректность вводимых пользователем данных;

корректность считываемых из файла данных

Модуль ввода данных, реализует простой графический интерфейс для ввода необходимых значений.

Модуль сохранения данных, формирует все записи в один Access - файл.

Модуль считывания предоставляет возможность открытия Access -файла и заполнения списков в программе.

Модуль редактирования записей предоставляет простой интерфейс для изменения ранее внесенных данных, а также добавления и удаления.

**4.2 Проектирование графического интерфейса**

На рисунке 3-4 представлена форма приложения и панель, содержащей таблицы данных:

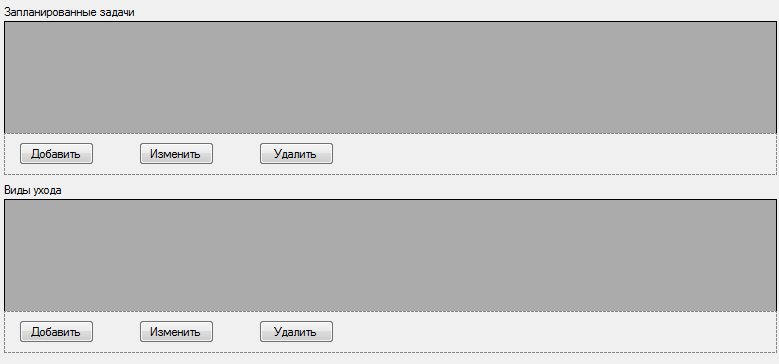


Рисунок 3 – Форма MainForm. Таблицы: Запланированные задачи и Виды ухода.

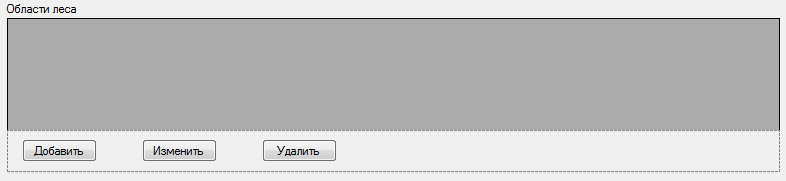


Рисунок 4 – Форма MainForm. Таблица Области леса

Для редактирования записей о продаже, изделия, материала используется отдельные формы приложения и панели для редактирования таблиц в MainForm. Для редактирования таблицы «Запланированные задачи» используется форма FmTask под названием «Редактирование задач», для таблицы «Виды ухода», форма FmKindCare с названием «Редактирование видов ухода», а для последней таблицы используется форма FmRegion, название которой «Редактирование области леса». Эти формы изображены на рисунках 5-7.

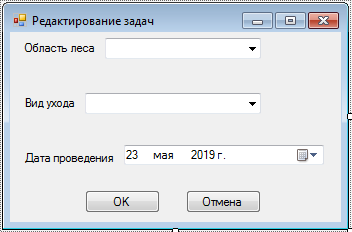


Рисунок 5 – Форма FmTask.

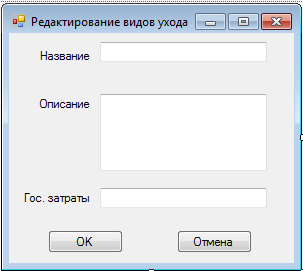


Рисунок 6 – Форма FmKindCare.

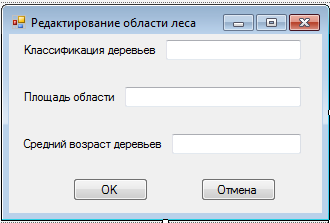


Рисунок 7 – Форма FmRegion.

Удаление области леса, видов ухода и задач, осуществляется с помощью специальной кнопки, расположенной на главной форме приложения у каждой таблицы. Нажатием кнопки «Удалить» - удаляем из списка область леса. Аналогично для других списков. Кнопки отображены на рисунках 8-10:

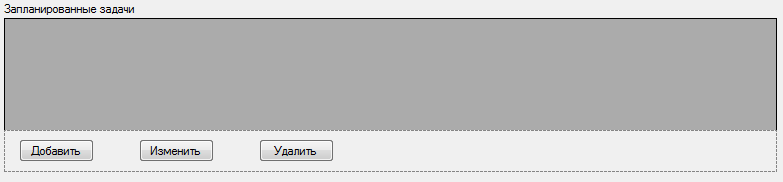


Рисунок 8 – Таблица «Запланированные задачи»

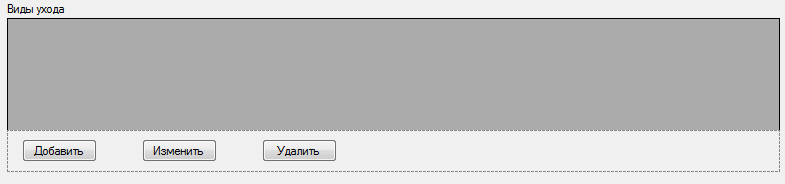


Рисунок 9 – Таблица «Виды ухода»

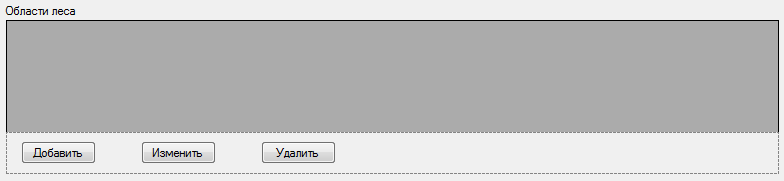


Рисунок 10 – Таблица «Области леса»**5 ТЕСТИРОВАНИЕ**

Для тестирования программы производились различные манипуляции с данными.

Проводились следующие тесты:

Тест 1 – Пустое значение в обязательном поле ввода данных, изображен на рисунке 12:

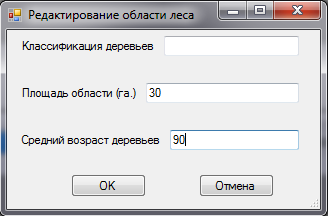


Рисунок 12 – Ввод пустого значения в поле «Классификация деревьев»

Сообщение, которое будет показано пользователю после ввода пустого значения, представлено на рисунке 13:

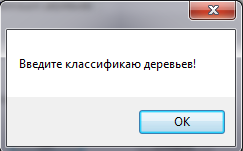


Рисунок 13 – Ошибка при вводе пустого значения.

Тест 2 – Попытка ввода текста в числовое поле, изображен на рисунке 14:

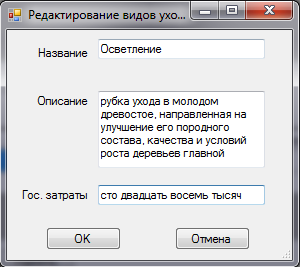


Рисунок 14 – Ввод текста в поле «Гос. Затраты»

Сообщение, которое будет показано пользователю после ввода текста вместо числового значения, представлено на рисунке 15:

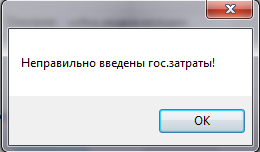


Рисунок 15 – Ввод некорректных данных.

Тест 3 –Добавление значение в поле, которого нет в базе, привязанной к этому полю. Изображено на рисунке 17:

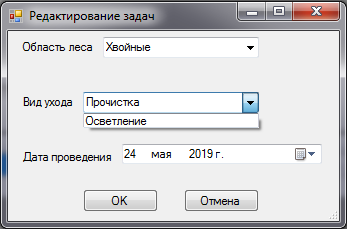


Рисунок 17 – Ввод значения которого нет в базе, привязанный к полю код вида ухода.

Сообщение, которое будет показано пользователю после ввода значения, которого нет в базе, представлено на рисунке 18:

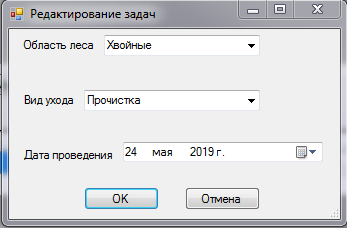


Рисунок 18 – Ошибка неверного выбора.

А так же были проведены другие операции в хаотическом порядке, призванные выявить ошибки в программе.

Тестирование выявило мелкие недочеты, которые впоследствии были устранены.

**СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

1. Э. Троелсен. С# и платформа .NET. Библиотека программиста. - СПб.: Питер, 2015. —796 с.: ил.
2. Е.А. Жоголев. «Технология программирования» - М., Научный мир, 2015, 216 с
3. Т. А. Павловская. «С#. Программирование на языке высокого уровня. Учебник для вузов» - СПб.: Питер, 2015. – 432 с.
4. Microsoft Developer Network(MSDN). [Электронный ресурс]. – Режим доступа: msdn.microsoft.com, свободный.
5. СТО ЗМК 01. – 2017. Работы студентов. Общие требования и правила оформления.

**ПРИЛОЖЕНИЕ А**

**Листинг библиотеки классов**

**Класс Db.**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Data.OleDb;

using System.Windows.Forms;

using System.Configuration;

namespace Forester

{

/// <summary>

/// класс для создания коннекции к БД

/// </summary>

public static class Db

{

//коннекция к БД

public static OleDbConnection Connection = new OleDbConnection();

/// <summary>

/// процедура соеднинения с БД

/// </summary>

/// <returns></returns>

public static bool Connect()

{

//читаем строку коннекции к БД из файла конфиг

var connectionString = ConfigurationManager.AppSettings["db"];

try

{

//присваиваем строку коннекции

Connection.ConnectionString = connectionString;

//открываем соединение с БД

Connection.Open();

return true;

}

catch (Exception e)

{

//обработка ошибки при коннекции к БД. вызываем сообщение с ошибкой

MessageBox.Show(@"Ошибка соединения с БД. " + e.Message, @"Ошибка",

MessageBoxButtons.OK,

MessageBoxIcon.Error);

return false;

}

}

}

}

**Класс Program.**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace Forester

{

static class Program

{

/// <summary>

/// Главная точка входа для приложения.

/// </summary>

[STAThread]

static void Main()

{

//пробуем соединиться к БД, если соединение не прошло то выведется сообщение, дальнейший запуск не актуален

if (!Db.Connect()) return;

Application.EnableVisualStyles();

Application.SetCompatibleTextRenderingDefault(false);

Application.Run(new MainForm());

}

}

}

**Класс KindCare.**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Forester

{

/// <summary>

/// описание вида ухода

/// </summary>

public class KindCare

{

//Идентификатор

public int Uid { get; set; }

//Название

public string Name { get; set; }

//Описание

public string Opis { get; set; }

//Государственные затраты

public float Cost { get; set; }

}

}

**Класс Region**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Forester

{

/// <summary>

/// описание области леса

/// </summary>

public class Region

{

//Идентификатор

public int Uid { get; set; }

//Классификация деревьев

public string Class { get; set; }

//Площадь области

public float Area { get; set; }

//Средний возраст деревьев

public int Year { get; set; }

}

}

**Класс Task.**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Forester

{

/// <summary>

/// описание запланированных задач

/// </summary>

public class Task

{

//Идентификатор

public int Uid { get; set; }

//Код области

public int Region { get; set; }

//Код ухода

public int Care { get; set; }

//Дата выполнения задачи

public DateTime Date { get; set; }

}

}

**ПРИЛОЖЕНИЕ Б**

**Листинг клиентского приложения**

**Форма приложения MainForm**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using System.Data.OleDb;

namespace Forester

{

/// <summary>

/// главная форма приложения

/// </summary>

public partial class MainForm : Form

{

public MainForm()

{

InitializeComponent();

//загружаем данные по областям леса

LoadRegion();

//загружаем данные по видам ухода

LoadKindCare();

//загружаем данные по запланированным задачам

LoadTask();

}

/// <summary>

/// загрузка областей леса

/// </summary>

private void LoadRegion()

{

//пишем SQL по отбору данных по областям леса, сортируем по классификации

var sql = @"select \* from Region order by rClass";

var da = new OleDbDataAdapter(sql, Db.Connection);

var ds = new DataSet();

da.Fill(ds);

//свзяываем отобанные данные с компонентом datagridview

dgvRegion.DataSource = ds;

dgvRegion.DataMember = ds.Tables[0].TableName;

//не показываем столбец с ИД

dgvRegion.Columns["rUid"].Visible = false;

//устанавдиваем ширину столбца

dgvRegion.Columns["rClass"].Width = 230;

//устанавливаем заголовок столбца

dgvRegion.Columns["rClass"].HeaderText = @"Классификация деревьев";

dgvRegion.Columns["rArea"].HeaderText = @"Площадь области";

dgvRegion.Columns["rArea"].Width = 230;

dgvRegion.Columns["rYear"].HeaderText = @"Средний возраст деревьев";

dgvRegion.Columns["rYear"].Width = 230;

}

/// <summary>

/// загрузка видов ухода

/// </summary>

private void LoadKindCare()

{

//пишем SQL по отбору данных по видам ухода, сортируем по названию

var sql = @"select \* from KindCare order by kName";

var da = new OleDbDataAdapter(sql, Db.Connection);

var ds = new DataSet();

da.Fill(ds);

//свзяываем отобанные данные с компонентом datagridview

dgvCare.DataSource = ds;

dgvCare.DataMember = ds.Tables[0].TableName;

//не показываем столбец с ИД

dgvCare.Columns["kUid"].Visible = false;

//устанавдиваем ширину столбца

dgvCare.Columns["kName"].Width = 230;

//устанавливаем заголовок столбца

dgvCare.Columns["kName"].HeaderText = @"Название";

dgvCare.Columns["kOpis"].HeaderText = @"Описание";

dgvCare.Columns["kOpis"].Width = 230;

dgvCare.Columns["kCost"].HeaderText = @"Гос. затраты";

dgvCare.Columns["kCost"].Width = 230;

}

/// <summary>

/// загрузка задач

/// </summary>

private void LoadTask()

{

//пишем SQL по отбору данных по запланированным задачам, сортируем по дате выполнения

var sql = @"select kUid, rUid, tUid, tRegion, [tCare], tDate from [Region], [Task], [KindCare] where tCare=kUid and tRegion=rUid order by tDate desc";

var da = new OleDbDataAdapter(sql, Db.Connection);

var ds = new DataSet();

da.Fill(ds);

//свзяываем отобанные данные с компонентом datagridview

dgvTask.DataSource = ds;

dgvTask.DataMember = ds.Tables[0].TableName;

//не показываем столбец с ИД

dgvTask.Columns["kUid"].Visible = false;

dgvTask.Columns["rUid"].Visible = false;

dgvTask.Columns["tUid"].Visible = false;

//устанавдиваем ширину столбца

dgvTask.Columns["tRegion"].Width = 230;

//устанавливаем заголовок столбца

dgvTask.Columns["tRegion"].HeaderText = @"Область леса";

dgvTask.Columns["tCare"].HeaderText = @"Вид ухода";

dgvTask.Columns["tCare"].Width = 230;

dgvTask.Columns["tDate"].HeaderText = @"Дата выполнения";

dgvTask.Columns["tDate"].Width = 230;

}

/// <summary>

/// добавление задачи

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

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

{

//создаем форму редактирования

var f = new FmTask();

//показываем диалог с редактированием

if (f.ShowDialog() == DialogResult.OK)

{

//если нажали кнопку ОК пишем SQL по добавлению данных о задаче

var cmd = new OleDbCommand(@"insert into Task (tRegion, tCare, tDate)

values (?,?,?)")

{

Connection = Db.Connection,

CommandType = CommandType.Text

};

//запоняем параметры для добавления данных

cmd.Parameters.AddWithValue(@"tRegion", f.Task.Region);

cmd.Parameters.AddWithValue(@"tCare", f.Task.Care);

cmd.Parameters.AddWithValue(@"tDate", f.Task.Date);

//выполняем запрос по добавлению

cmd.ExecuteNonQuery();

//обновляем данные в datagridview

LoadTask();

}

}

/// <summary>

/// редактирование задачи

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

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

{

//если нет выделенной ячейки, то выходим

if (dgvTask.CurrentCell == null) return;

//индекс выделенной ячейки

var i = dgvTask.CurrentCell.RowIndex;

//создаем форму редактирования

var f = new FmTask();

//заплняем объект класса Task данными из datagridview

f.Task.Uid = (int)dgvTask.Rows[i].Cells["tUid"].Value;

f.Task.Region = (int)(dgvTask.Rows[i].Cells["rUid"].Value);

f.Task.Care = (int)(dgvTask.Rows[i].Cells["kUid"].Value);

f.Task.Date = Convert.ToDateTime(dgvTask.Rows[i].Cells["tDate"].Value);

//показываем диалог с редактированием

if (f.ShowDialog() == DialogResult.OK)

{

//если нажали кнопку ОК пишем SQL по изменению данных о задачи

var cmd = new OleDbCommand(@"update Task set tRegion=?, tCare=?, tDate=? where tUid=?")

{

Connection = Db.Connection,

CommandType = CommandType.Text

};

//запоняем параметры для изменения данных

cmd.Parameters.AddWithValue(@"tRegion", f.Task.Region);

cmd.Parameters.AddWithValue(@"tCare", f.Task.Care);

cmd.Parameters.AddWithValue(@"tDate", f.Task.Date);

cmd.Parameters.AddWithValue(@"tUid", f.Task.Uid);

//выполняем запрос по изменению

cmd.ExecuteNonQuery();

//обновляем данные в datagridview

LoadTask();

}

}

/// <summary>

/// удаление задачи

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

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

{

//если нет выделенной ячейки, то выходим

if (dgvTask.CurrentCell == null) return;

//диалог подтверждения удаления

if (MessageBox.Show(@"Удалить?", @"Удалить", MessageBoxButtons.YesNo) != DialogResult.Yes)

return;

//индекс выделенной ячейки

var i = dgvTask.CurrentCell.RowIndex;

//идентификатор записи

var id = dgvTask.Rows[i].Cells["tUid"].Value;

//sql по удалению задачи

var cmd = new OleDbCommand(@"delete from Task where tUid=?")

{

Connection = Db.Connection,

CommandType = CommandType.Text

};

//запоняем параметры для удаления данных

cmd.Parameters.AddWithValue(@"tUid", id);

//выполняем запрос по удалению

cmd.ExecuteNonQuery();

//обновляем данные в datagridview

LoadTask();

}

/// <summary>

/// обработка двойного клика по datagridview

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

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

{

BtnTaskChange\_Click(sender, e);

}

/// <summary>

/// добавление вида ухода

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

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

{

//создаем форму редактирования

var f = new FmKindCare();

//показываем диалог с редактированием

if (f.ShowDialog() == DialogResult.OK)

{

//если нажали кнопку ОК пишем SQL по добавлению данных о виде ухода

var cmd = new OleDbCommand(@"insert into KindCare (kName, kOpis, kCost)

values (?,?,?)")

{

Connection = Db.Connection,

CommandType = CommandType.Text

};

//запоняем параметры для добавления данных

cmd.Parameters.AddWithValue(@"kName", f.KindCare.Name);

cmd.Parameters.AddWithValue(@"kOpis", f.KindCare.Opis);

cmd.Parameters.AddWithValue(@"kCost", f.KindCare.Cost);

//выполняем запрос по добавлению

cmd.ExecuteNonQuery();

//обновляем данные в datagridview

LoadKindCare();

}

}

/// <summary>

/// редактирование вида ухода

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

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

{

//если нет выделенной ячейки, то выходим

if (dgvCare.CurrentCell == null) return;

//индекс выделенной ячейки

var i = dgvCare.CurrentCell.RowIndex;

//создаем форму редактирования

var f = new FmKindCare();

//заплняем объект класса KindCare данными из datagridview

f.KindCare.Uid = (int)dgvCare.Rows[i].Cells["kUid"].Value;

f.KindCare.Name = Convert.ToString(dgvCare.Rows[i].Cells["kName"].Value);

f.KindCare.Opis = Convert.ToString(dgvCare.Rows[i].Cells["kOpis"].Value);

f.KindCare.Cost = (float)dgvCare.Rows[i].Cells["kCost"].Value;

//показываем диалог с редактированием

if (f.ShowDialog() == DialogResult.OK)

{

//если нажали кнопку ОК пишем SQL по изменению данных о виде ухода

var cmd = new OleDbCommand(@"update KindCare set kName=?, kOpis=?, kCost=?

where kUid=?")

{

Connection = Db.Connection,

CommandType = CommandType.Text

};

//запоняем параметры для изменения данных

cmd.Parameters.AddWithValue(@"kName", f.KindCare.Name);

cmd.Parameters.AddWithValue(@"kOpis", f.KindCare.Opis);

cmd.Parameters.AddWithValue(@"kCost", f.KindCare.Cost);

cmd.Parameters.AddWithValue(@"kUid", f.KindCare.Uid);

//выполняем запрос по изменению

cmd.ExecuteNonQuery();

//обновляем данные в datagridview

LoadKindCare();

}

}

/// <summary>

/// удаление вида ухода

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

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

{

//если нет выделенной ячейки, то выходим

if (dgvCare.CurrentCell == null) return;

//диалог подтверждения удаления

if (MessageBox.Show(@"Удалить?", @"Удалить", MessageBoxButtons.YesNo) != DialogResult.Yes)

return;

//индекс выделенной ячейки

var i = dgvCare.CurrentCell.RowIndex;

//идентификатор записи

var id = dgvCare.Rows[i].Cells["kUid"].Value;

//sql по удалению вида ухода

var cmd = new OleDbCommand(@"delete from KindCare where kUid=?")

{

Connection = Db.Connection,

CommandType = CommandType.Text

};

//запоняем параметры для удаления данных

cmd.Parameters.AddWithValue(@"kUid", id);

//выполняем запрос по удалению

cmd.ExecuteNonQuery();

//обновляем данные в datagridview

LoadKindCare();

}

/// <summary>

/// обработка двойного клика по datagridview

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

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

{

BtnCareChange\_Click(sender, e);

}

/// <summary>

/// добавление области леса

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

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

{

//создаем форму редактирования

var f = new FmRegion();

//показываем диалог с редактированием

if (f.ShowDialog() == DialogResult.OK)

{

//если нажали кнопку ОК пишем SQL по добавлению данных об области леса

var cmd = new OleDbCommand(@"insert into Region (rClass, rArea, rYear)

values (?,?,?)")

{

Connection = Db.Connection,

CommandType = CommandType.Text

};

//запоняем параметры для добавления данных

cmd.Parameters.AddWithValue(@"rClass", f.Region.Class);

cmd.Parameters.AddWithValue(@"rArea", f.Region.Area);

cmd.Parameters.AddWithValue(@"rYear", f.Region.Year);

//выполняем запрос по добавлению

cmd.ExecuteNonQuery();

//обновляем данные в datagridview

LoadRegion();

}

}

/// <summary>

/// редактирование области леса

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

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

{

//если нет выделенной ячейки, то выходим

if (dgvRegion.CurrentCell == null) return;

//индекс выделенной ячейки

var i = dgvRegion.CurrentCell.RowIndex;

//создаем форму редактирования

var f = new FmRegion();

//заплняем объект класса Region данными из datagridview

f.Region.Uid = (int)dgvRegion.Rows[i].Cells["rUid"].Value;

f.Region.Class = Convert.ToString(dgvRegion.Rows[i].Cells["rClass"].Value);

f.Region.Area = (float)(dgvRegion.Rows[i].Cells["rArea"].Value);

f.Region.Year = (int)dgvRegion.Rows[i].Cells["rYear"].Value;

//показываем диалог с редактированием

if (f.ShowDialog() == DialogResult.OK)

{

//если нажали кнопку ОК пишем SQL по изменению данных об области леса

var cmd = new OleDbCommand(@"update Region set rClass=?, rArea=?, rYear=? where rUid=?")

{

Connection = Db.Connection,

CommandType = CommandType.Text

};

//запоняем параметры для изменения данных

cmd.Parameters.AddWithValue(@"rClass", f.Region.Class);

cmd.Parameters.AddWithValue(@"rArea", f.Region.Area);

cmd.Parameters.AddWithValue(@"rYear", f.Region.Year);

cmd.Parameters.AddWithValue(@"rUid", f.Region.Uid);

//выполняем запрос по изменению

cmd.ExecuteNonQuery();

//обновляем данные в datagridview

LoadRegion();

}

}

/// <summary>

/// удаление области леса

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

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

{

//если нет выделенной ячейки, то выходим

if (dgvRegion.CurrentCell == null) return;

//диалог подтверждения удаления

if (MessageBox.Show(@"Удалить?", @"Удалить", MessageBoxButtons.YesNo) != DialogResult.Yes)

return;

//индекс выделенной ячейки

var i = dgvRegion.CurrentCell.RowIndex;

//идентификатор записи

var id = dgvRegion.Rows[i].Cells["rUid"].Value;

//sql по удалению области леса

var cmd = new OleDbCommand(@"delete from Region where rUid=?")

{

Connection = Db.Connection,

CommandType = CommandType.Text

};

//запоняем параметры для удаления данных

cmd.Parameters.AddWithValue(@"rUid", id);

//выполняем запрос по удалению

cmd.ExecuteNonQuery();

//обновляем данные в datagridview

LoadRegion();

}

/// <summary>

/// обработка двойного клика по datagridview

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

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

{

BtnRegionChange\_Click(sender, e);

}

}

}

**Форма приложения FmKindCare.**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace Forester

{

public partial class FmKindCare : Form

{

/// <summary>

/// форма ввода и редактирования видов ухода

/// </summary>

public FmKindCare()

{

InitializeComponent();

}

//объект класса KindCare

public KindCare KindCare = new KindCare();

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

{

//устанавливаем значения компонентов при редактировании

txtCost.Text = KindCare.Cost.ToString();

txtName.Text = KindCare.Name;

txtOpis.Text = KindCare.Opis;

}

/// <summary>

/// обработка кнопки ОК

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

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

{

//проверяем на заполненность названия

if (string.IsNullOrEmpty(txtName.Text))

{

//сообщение

MessageBox.Show(@"Введите название!");

//устанавливаем фокус

txtName.Focus();

//не закрываем форму

DialogResult = DialogResult.None;

return;

}

//проверяем на заполненность описания

if (string.IsNullOrEmpty(txtOpis.Text))

{

//сообщение

MessageBox.Show(@"Введите описание!");

//устанавливаем фокус

txtOpis.Focus();

//не закрываем форму

DialogResult = DialogResult.None;

return;

}

//проверяем правильность ввода гос. затрат

Single d = 0; if (!Single.TryParse(txtCost.Text, out d))

{

MessageBox.Show(@"Неправильно введены гос.затраты!");

txtCost.Focus();

DialogResult = DialogResult.None;

return;

}

//присваиваем данные из компонент объекту классса KindCare

KindCare.Name = txtName.Text;

KindCare.Opis = txtOpis.Text;

KindCare.Cost = Convert.ToSingle(txtCost.Text);

}

}

}

**Форма приложения FmRegion.**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace Forester

{

public partial class FmRegion : Form

{

/// <summary>

/// форма ввода и редактирования области леса

/// </summary>

public FmRegion()

{

InitializeComponent();

}

//объект класса Region

public new Region Region = new Region();

/// <summary>

/// обработка кнопки ОК

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void Button1\_Click(object sender, EventArgs e)

{

//проверяем на заполненность классификации деревьев

if (string.IsNullOrEmpty(txtClass.Text))

{

//сообщение

MessageBox.Show(@"Введите классификаю деревьев!");

//устанавливаем фокус

txtClass.Focus();

//не закрываем форму

DialogResult = DialogResult.None;

return;

}

//проверяем правильность ввода площади

Single d = 0; if (!Single.TryParse(txtArea.Text, out d))

{

MessageBox.Show(@"Неправильно введена площадь области!");

txtArea.Focus();

DialogResult = DialogResult.None;

return;

}

//проверяем правильность ввода среднего возраста

int f = 0; if (!Int32.TryParse(txtYear.Text, out f))

{

MessageBox.Show(@"Неправильно введен средний возраст деревьев!");

txtYear.Focus();

DialogResult = DialogResult.None;

return;

}

//присваиваем данные из компонент объекту классса Region

Region.Area = Convert.ToSingle(txtArea.Text);

Region.Class = txtClass.Text;

Region.Year = Convert.ToInt32(txtYear.Text);

}

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

{

//устанавливаем значения компонентов при редактировании

txtArea.Text = Region.Area.ToString();

txtClass.Text = Region.Class;

txtYear.Text = Region.Year.ToString();

}

}

}

**Форма приложения FmTask.**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using System.Data.OleDb;

namespace Forester

{

public partial class FmTask : Form

{

/// <summary>

/// форма ввода и редактирования запланированных задач

/// </summary>

public FmTask()

{

InitializeComponent();

//заполняем комбобокс видов ухода

LoadKindCare();

//заполняем комбобокс областей леса

LoadRegion();

//по-умолчанию дата выполнения - сегодняшнее число

Task.Date = DateTime.Today;

}

//объект класса Task

public Task Task = new Task();

private void LoadKindCare()

{

//пишем SQL по отбору данных по видам ухода, сортируем по названию

var sql = @"select \* from KindCare order by kName";

var da = new OleDbDataAdapter(sql, Db.Connection);

var ds = new DataSet();

da.Fill(ds);

//свзяываем отобанные данные с компонентом комбобокс

cbCare.DataSource = ds.Tables[0];

cbCare.DisplayMember = "kName";

cbCare.ValueMember = "kUid";

}

private void LoadRegion()

{

//пишем SQL по отбору данных по областям леса, сортируем по размеру площади

var sql = @"select \* from Region order by rArea";

var da = new OleDbDataAdapter(sql, Db.Connection);

var ds = new DataSet();

da.Fill(ds);

//свзяываем отобанные данные с компонентом комбобокс

cbRegion.DataSource = ds.Tables[0];

cbRegion.DisplayMember = "rClass";

cbRegion.ValueMember = "rUid";

}

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

{

//устанавливаем значения компонентов при редактировании

cbCare.SelectedValue = Task.Care;

cbRegion.SelectedValue = Task.Region;

dtpDate.Value = Task.Date;

}

/// <summary>

/// обработка кнопки ОК

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void Button1\_Click(object sender, EventArgs e)

{

//проверяем выбор вида ухода

if (cbCare.SelectedIndex < 0)

{

//сообщение

MessageBox.Show(@"Выберите вид ухода!");

//устанавливаем фокус

cbCare.Focus();

//не закрываем форму

DialogResult = DialogResult.None;

return;

}

//проверяем выбор области леса

if (cbRegion.SelectedIndex < 0)

{

MessageBox.Show(@"Выберите область!");

cbRegion.Focus();

DialogResult = DialogResult.None;

return;

}

//присваиваем данные из компонентов объекту классса Task

Task.Care = (int)cbCare.SelectedValue;

Task.Region = (int)cbRegion.SelectedValue;

Task.Date = dtpDate.Value;

}

}

}