Міністерство освіти і науки України

Харківський національний університет радіоелектроніки

Кафедра Програмної інженерії

КУРСОВА РОБОТА

ПОЯСНЮВАЛЬНА ЗАПИСКА

з дисципліни «Основи об’єктно-орієнтованого програмування»

ДОВІДНИК КОМАНДИРА

Керівник, ст. викл. Бондарєв В.М.

Студент гр. ПЗПІ-16-1 Півторак К.С.

Комісія:

проф. \_\_\_\_\_\_\_\_\_\_\_\_\_\_ Бондарєв В.М.

Харків 2017

**ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ РАДІОЕЛЕКТРОНІКИ**

Кафедра: ***Програмної інженерії***

Дисципліна: ***Об’єктно-орієнтоване програмування***

Напрям: ***Програмна інженерія***

Курс: 1 . Група: ПЗПІ-16-1 . Семестр: 2 .

***ЗАВДАННЯ***

***на курсовий проект студента***

***Півторак Катерини Сергіївни***

(Прізвище, Ім’я, По батькові)

1) Тема проекту: *Довідник командира*

2) Термін здачі студентом закінченого проекту: **«*31» - травня - 2017 р.***

3) Вихідні дані до проекту:

*Специфікація програми, методичні вказівки до виконання курсової роботи*

4) Зміст розрахунково-пояснювальної записки:

*Вступ, специфікація програми, проектна специфікація, інструкція користувача, висновки*

5) Перелік графічного матеріалу:

*Схема об’єктної моделі, алгоритми, приклади екранних форм*

**КАЛЕНДАРНИЙ ПЛАН**

|  |  |  |
| --- | --- | --- |
| *№* | *Назва етапу* | *Термін виконання* |
| 1 | Видача теми, узгодження і затвердження теми. | 15-02-2017 р. |
| 2 | Формулювання вимог до програми. | 15-02-2017 – 22-02-2017 р. |
| 3 | Розробка підсистеми зберігання, пошуку, видалення автобусів. | 22-02-2017 – 09-03-2017 р. |
| 4 | Розробка підсистеми зберігання, пошуку, повернення квитків. | 09-03-2017 – 23-03-2017 р. |
| 5 | Розробка інтерфейсу. | 23-03-2017 – 27-03-2017 р. |
| 6 | Розробка функцій зберігання та завантаження даних. | 27-03-2017 – 10-04-2017 р. |
| 7 | Тестування і доопрацювання розробленої програмної системи. | 10-04-2017 – 28-04-2017 р. |
| 8 | Оформлення пояснювальної записки, додатків, графічного матеріалу. | 28-04-2017 – 29-05-2017 р. |
| 9 | Захист | 29-05-2017 – 31-05-2017 р. |

Студент \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Керівник \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Бондарєв В.М.

« 15 » лютого   2017 року

РЕФЕРАТ

Пояснювальна записка до курсової роботи: 32 с., 17 рис., 3 додатки. Кількість частин: 3. Кількість джерел з переліку посилань: 5.

Мета роботи – проектування та створення системи для можливості обробки даних довідника командира.

Методи розробки базуються на використанні середи розробки Microsoft Visual Studio 2016, Windows Forms, платформи .NET Framework 4.5, мови програмування C#.

У результаті розробки отримано програму в якій можливо зберігати данні службовця (ім’я, адреса, професія, освіта, звання та дата його отримання, посада, підрозділ, форма служби, період служби, особливості характеру), формувати списки службовців заданого підрозділу та службовців із заданим званням, виконувати пошук за заданим словом або кількома словами.

ПРОГРАМА, КУРСОВА, ОБ’ЄКТНО-ОРІЄНТОВАНЕ ПРОГРАМУВАННЯ, МОВА ПРОГРАМУВАННЯ C#, ПЛАТФОРМА .NET, ДОВІДНИК КОМАНДИРА.

ЗМІСТ

Вступ……………………………………………………………………………….6

1 СПЕЦИФІКАЦІЯ ПРОГРАМИ...………………………………………………7

* 1. Основні вимоги та функції програми………………………………….7
  2. Опис інтерфейсу………………………………………………………..7

2 ПРОЕКТНА СПЕЦИФІКАЦІЯ…………………..…………………………...15

2.1 Об’єктна модель програми……………………………………………15

2.2 Реалізація функцій програми…………………………………………17

3 ІНСТРУКЦІЯ КОРИСТУВАЧА……………………………………………...20

ВИСНОВКИ...……………………………………………………………....……24

ПЕРЕЛІК ПОСИЛАНЬ..………………………………………………….……..25

Додаток А – Код основної форми……………...……………………………….26

Додаток Б – Код основної форми додавання та редагування….……………..29

Додаток С – Код форми формування списків…….……………………………31

ВСТУП

Для покращення формування списків солдат та полегшення пошуку серед усієї бази даних з’явилася ідея створення системи, яка б допомагала структурувати дані підлеглих.

Розробка програмних систем має важливе значення, адже з часом ці системи зможуть поліпшити працю людини. Надаючи перевагу системам ми зможемо зменшити кількість помилок, які залежать від людського фактору.

В даній роботі стала задача створити систему, яка зможе формувати список службовців та формувати списки за спільною ознакою. Зберігати всю інформацію локально та надавати її при необхідності.

Мета даної курсової роботи освоїти мову програмування С#, навчитися об’єктно-орієнтованому програмуванню, розвити навички роботи в Windows Forms.

Задачі роботи:

- Розробка системи зчитування та запису даних;

- Створення логіки роботи системи;

- Робота з користувачем;

- Розробка логіки додавання, пошуку та видалення інформації;

- Тестування готового продукту;

- Відпрацьовування навичок оновлення продукту;

- Створення стійкості програми.

1 СПЕЦИФІКАЦІЯ ПРОГРАМИ

1.1 Основні вимоги та функції програми

Система довідника має задачу полегшити роботу відповідальним за обробку інформації про усіх воєннослужбовців.

Серед основних функцій можна виділити наступні:

* Додавання інформації про нову людину до системи;
* Редагування вже наявного інформації про людину;
* Видалення інформації про людину;
* Пошук за заданим словом або словосполукою;
* Пошук за заданим підрозділом;
* Пошук за заданим званням;
* Запис в базу відомостей про службовців;
* Можливість зберігати всю інформацію.
  1. Опис інтерфейсу

При вході в програму перед нами з’являтиметься вікно із списком службовців, у якому можна додати мову людину, та знайти людину, у інформації якої є ключові слова пошуку.

При завантаженні перед нами з’являються (див. рис 1.1):

* Місце для виведення інформації про людину із стовпцями:

1. ФІО людини;
2. Адреса;
3. Професія;
4. Освіта;
5. Звання;
6. Дата отримання звання;
7. Посада;
8. Підрозділ;
9. Форма служби;
10. Період служби;
11. Характеристика

* Кнопки:

1. «Додати»;
2. «Видалити»;
3. «Редагувати»;
4. «Пошук» (у будь-якій графі);
5. «Пошук» (за заданим підрозділом);
6. «Пошук» (за заданим званням);

* Меню із пунктами:

1. «Відкрити»
2. «Зберегти»

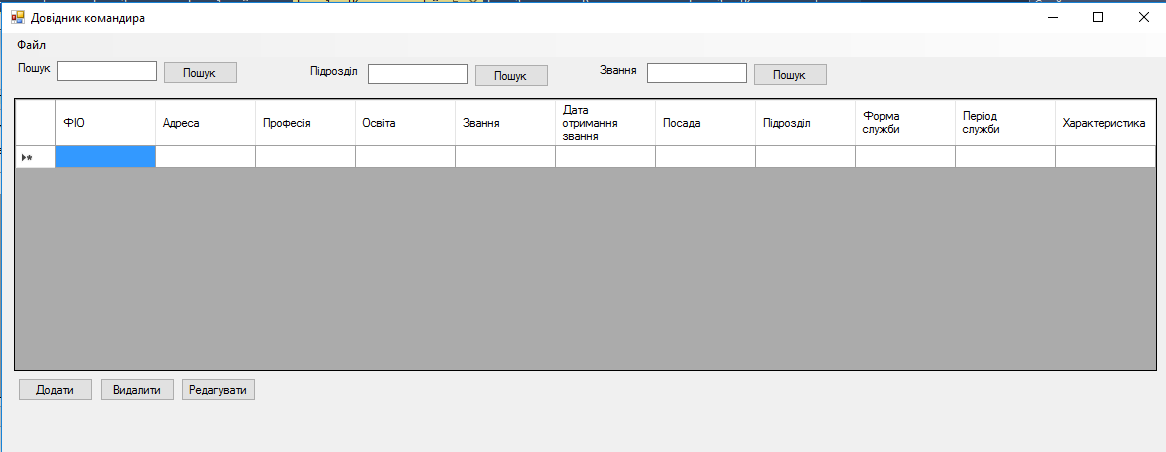


Рисунок 1.1 – Вікно відображення та маніпуляції інформацією

Поле «Файл», при натисканні на яку буде відкриватися меню із двома полями – «Відкрити» і «Зберегти», які будуть виконувати функції відкриття файлу та збереження у файл відповідно(рис. 1.2.1, рис 1.2.2).

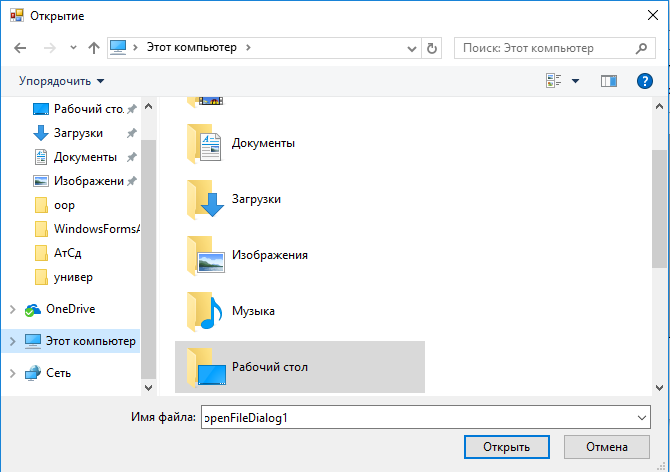


Рисунок 1.2.1 – Відкриття файлу

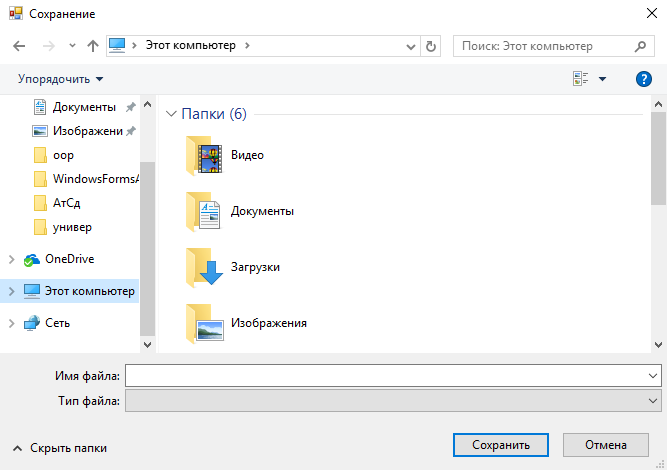


Рисунок 1.2.2 – Збереження у файл

Натиснувши кнопку «Додати» або «Редагувати» нам відкриється нове вікно(рис. 1.3.1, рис. 1.3.2), у якому ми побачимо:

* Місце для введення інформації про людину із стовпцями:

1. ФІО людини;
2. Адреси;
3. Професії;
4. Освіти;
5. Звання;
6. Дати отримання звання;
7. Посади;
8. Підрозділу;
9. Періоду служби;
10. Характеристики.

* Кнопки:

1. «Додати».

* ComboBox:

1. «Форма служби».

В залежності від того, додаємо ми або редагуємо, нам відкриється або вікно з порожніми полями, або вікно із заповненими полями даними людини, інформацію про яку ми редагуємо.

Під час використання програми користувач іноді намагається зробити дії, які можуть нашкодити дії програми, для цього система має деякий захист у вигляді попереджень (рис 1.4).

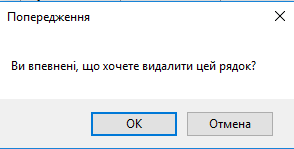
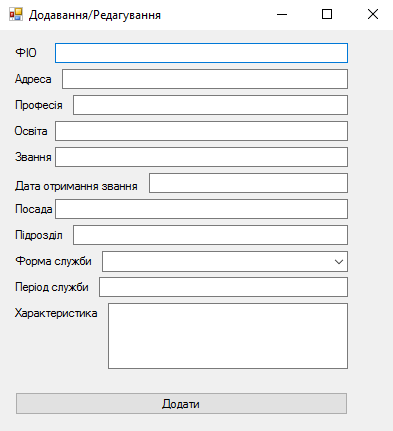


Рисунок 1.4 – Попередження перед виконанням змін



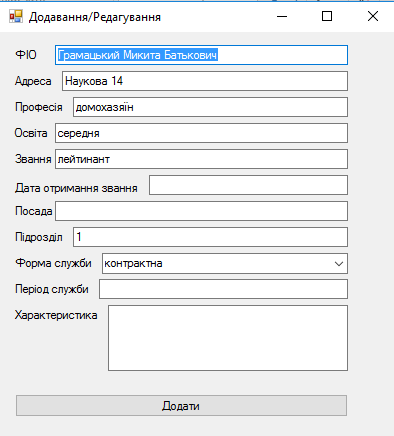
Рисунок 1.3.1 – Вікно додавання даних

Рисунок 1.3.2 – Вікно редагування даних

При натисканні на будь-яку з кнопок «Пошук», попередньо заповнивши відповідне поле текстом буде відкриватись нове вікно із списком службовців, інформація про яких задовольняє критерію пошуку(рис. 1.4)

На ньому ми побачимо:

* Місце для виведення інформації про людину із стовпцями

1. ФІО людини;
2. Адреса;
3. Професія;
4. Освіта;
5. Звання;
6. Дата отримання звання;
7. Посада;
8. Підрозділ;
9. Форма служби;
10. Період служби;
11. Характеристика.

* Кнопки:

1. «Видалити».

* Меню із пунктами:

1. Зберегти»

Кнопка «Видалити» зроблена для того, щоб редагувати список людей, якщо у нього потрапили непотрібні службовці. Меню із вкладкою «Зберегти» слугує для того, щоб можна було зберегти сформований список службовців.

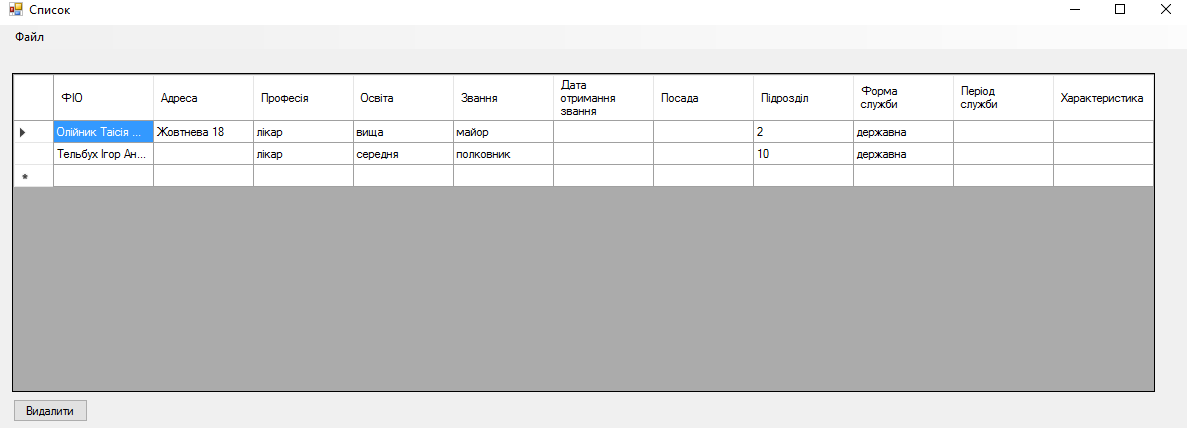


Рисунок 1.5 – Вікно із списком людей з конкретною властивістю

2 ПРОЕКТНА СПЕЦИФІКАЦІЯ

2.1 Об’єктна модель програми

Програма складається з декількох класів, вікон та файлів, які створюються під час першого запуску системи і оновлюються впродовж роботи програми. Класи взаємодіють між собою та надають нам очікуваний результат [1].

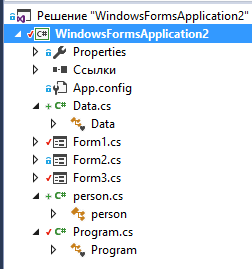
 На рисунку 2.1 зображено архітектуру проекту (всі наявні класи, службові файли та посилання).

Рисунок 2.1 – Архітектура проекту

Почнемо з класу Data, він необхідний для запису та зміни інформацій про одну конкретну людину.

На рисунку 2.2 зображено змінні, які реалізують всю необхідну інформацію людину:

* ФІО людини;
* Адреса;
* Професія;
* Освіта;
* Звання;
* Дата отримання звання;
* Посада;
* Підрозділ;
* Форма служби;
* Період служби;
* Характеристика

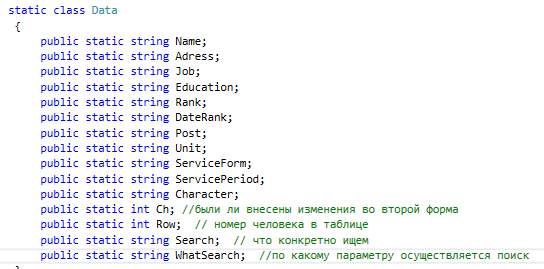
Та деякі службові функції, які допомагають запам’ятовувати чи було внесено зміни, номер людини у таблиці, ключове слово пошуку та параметр за яким здійснюється пошук.

Рисунок 2.2 – Поля класу Data

Цей клас зберігає інформацію про людину, яку щойно редагували та використовується для маніпуляції цією інформацією.

Клас person дуже схожий на клас Data, але він реалізує запис та зберігання інформації у таблиці даних. Цей клас має такі поля (див. рис 2.3):

* ФІО людини;
* Адреса;
* Професія;
* Освіта;
* Звання;
* Дата отримання звання;
* Посада;
* Підрозділ;
* Форма служби;
* Період служби;
* Характеристика

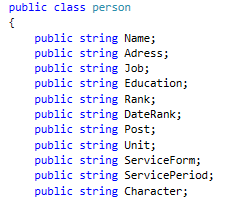


Рисунок 2.3 – Поля класу person

2.2 Реалізація функцій програми

Більш детально розглянемо принцип дій функцій, які у процесі віконує програма.

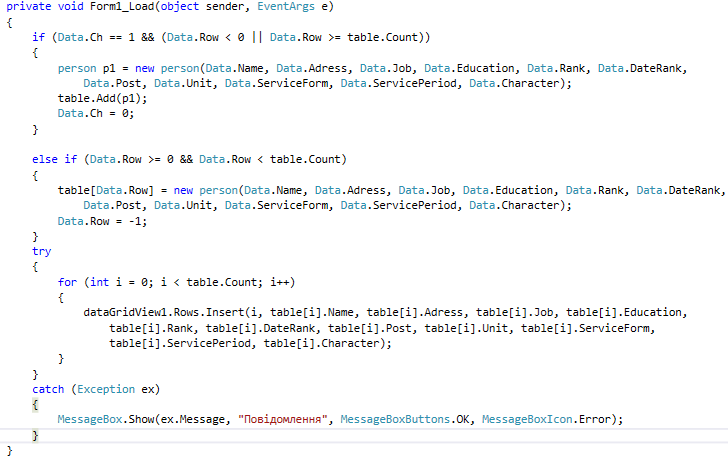
Метод для запуску форми Form1(рис. 2.4). Перед тим, як показати користувачеві вікно із інформацією, програма перевіряє, чи було внесено які-небудь зміни у таблицю і відповідно до результату виводить або попередні дані, або змінені.

Рисунок 2.3 – Метод загрузки форми Form1

При завантаженні форми додавання/редагування(рис. 2.4) перевіряється, що саме хоче зробити користувач, та відповідно до цього, генерується форма Form2 або з пустими полями, або з полями, заповненими редагуємою інформацією.

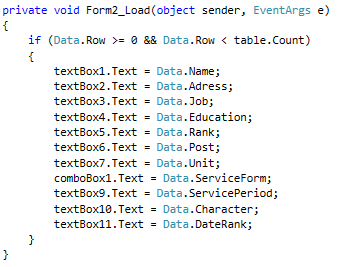


Рисунок 2.4 – Метод загрузки форми Form2

При завантаженні форми отриманого списка Form3(рис. 2.5), перевіряється, за яким критерієм(довільним, підрозділом або званням) користувач хоче винонати пошук і сформувати список. В залежності від цього перевіряється наявність ключового слова, слів або цифр у будь-якому стовбці, у підрозділі або у званні. І показується список службовців, відповідних критерію пошуку.

Рисунок 2.5 – Метод загрузки форми Form3

3 ІНСТРУКЦІЯ КОРИСТУВАЧА

Для користування програмою її необхідно завантажити на ваш комп’ютер. Після запуску, вам буде доступне вікно створення списку службовців і можливості додавання, видалення, редагування інформації і пошук за заданою ознакою (рис. 3.1).

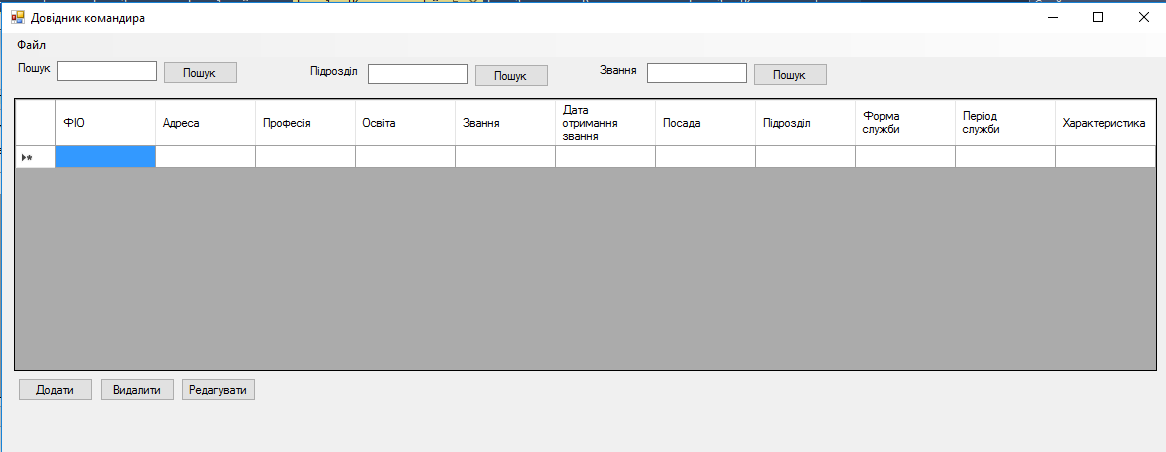


Рисунок 3.1 – Вікно створення списку службовців

Щоб додати нового офіцера потрібно натиснути кнопку «Додати». Відкриється нове вікно у якому треба буде ввести інформацію про людину. Будь-яке поле можна залишити порожнім, адже не завжди доступна уся інформація про людину

Форма служби. Форма служби може бути двох видів: державна та контрактна. Саме для цього передбачено комбінований список тільки з цими двома позиціями на вибір.

Щоб додати нову людину необхідно після заповнення інформації про неї необхідно натиснути кнопку «Додати», після чого у таблиці у головному вікні додається новий рядок із тільки но введеною інформацією (рис. 3.2).

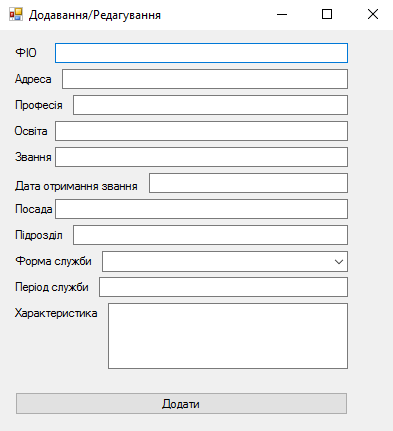


Рисунок 3.2 – Вікно додавання інформації про службовця

Для редагування, треба мишкою натиснути на потрібний нам рядок, після чого відкриється вікно як при натисканні кнопки «Додати», тільки будуть заповнені поля, інформацію яких ми обрали для редагування. Після внесення змінних потрібно натиснути кнопку «Додати» і інформація людини, яку ми обрали буде змінена на відредаговану (рис. 3.3).

Для видалення рядка треба мишкою натиснути на будь-яке поле потрібного рядка та натиснути кнопку «Видалити».

Для збереження файлу треба натиснути на поле «Файл», з випадаючого меню вибрати поле «Зберегти» після чого відкриється стандартне вікно збереження файлу, у якому потрібно вибрати місце, куди бажаємо зберегти файл і написати назву файлу.

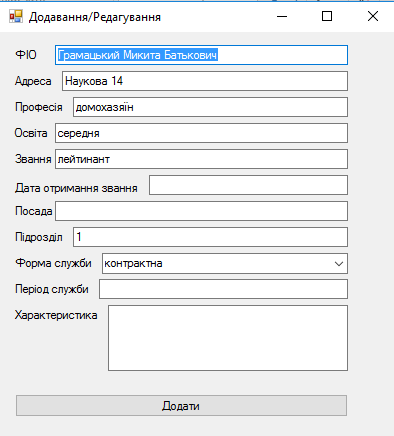
Для відкриття файлу треба натиснути на поле «Файл», з випадаючого меню вибрати поле «Відкрити» після чого відкриється стандартне вікно відкриття файлу, у якому потрібно вибрати файл, який хочемо відкрити.

Рисунок 3.3 – Вікно редагування інформації про службовця

Для пошуку ключового слова у будь-якій колонці потрібно у першому з ліва полі для вводу інформації ввести слово, за присутністю якого буде сформовано список і натиснути справа кнопку «Пошук».

Для пошуку людей з одного підрозділу потрібно у середньому полі для вводу інформації ввести потрібний підрозділ і натиснути справа кнопку «Пошук».

Для пошуку людей з конкретним званням потрібно у правому полі для вводу інформації ввести потрібний звання і натиснути справа кнопку «Пошук».

Після натискання будь-якої з кнопок «Пошук», відкриється нове вікно, у якому буде сформовано список задовольняючих службовців. У цьому вікні доступні кнопки «Видалити» та пункт меню «Файл» - «Зберегти», функції яких було описано раніше (рис. 3.4).

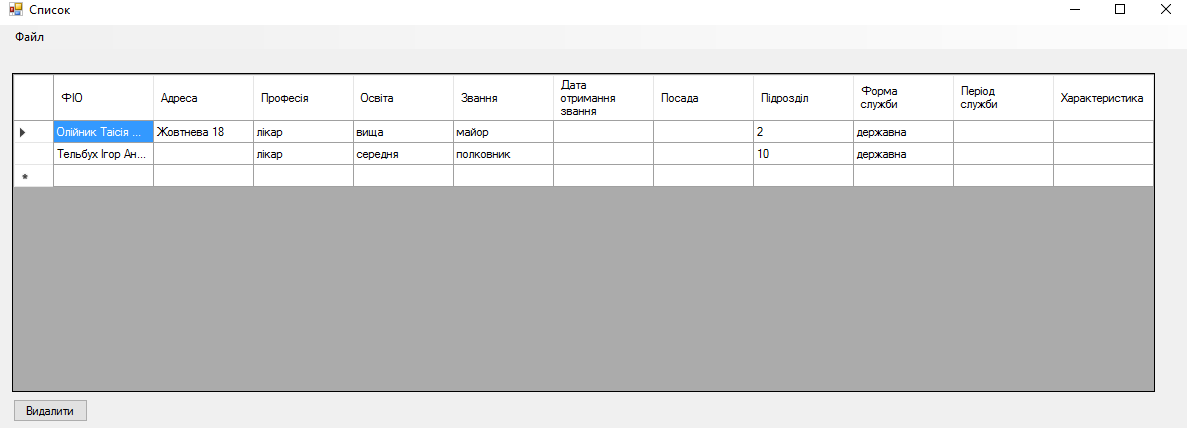


Рисунок 3.4 – Вікно для відображення списку людей за заданим параметром

ВИСНОВКИ

В ході виконання курсової роботи були освоєні методи створення програмних систем, було використано об’єктно-орієнтоване програмування.

У результаті створено програмну систему яка має певні функції які необхідні для реалізування поставленої задачі. Ця система має змогу обробляти усі запроси користувача та видавати коректні відгуки. Зберігати всю подану інформацію, мати можливість змінювати збереженні дані, додавати нові та видаляти непотрібні.

На практиці, виконуючи курсову роботу, ми познайомилися з середою розробки Microsoft Visual Studio 2016, Windows Forms, платформою .NET Framework 4.5, мовою програмування C#.

Під час створення курсової роботи придбалися навички роботи з програмною системою, створення в’язків між об’єктами.

ПЕРЕЛІК ПОСИЛАНЬ

1. Бондарев В. М. Объектно-ориентированное программирование на С# [Текст] : учеб. Пособие / В. М. Бондарев. – Харьков : СМИТ, 2009. –224 с. – Библиогр. : ISBN 978-966-2028-34-8.
2. SoloLearn. Freeprogramingcourses. C# Tutorial[Электронный ресурс] / SoloLearn –бесплатные курсы по программированию. – Режим доступа : www/URL:https://www.sololearn.com/Course/CSharp/ – 22.05.2017 г. – Загл. сэкрана.
3. Троелсен, Э. Язык программирования C# 5.0 и платформа .NET 4.5, 6-е изд. [Текст]: пер. с англ./ Э. Троелсен.- М.: Вильямс, 2013 - 1312 с. – ISBN 978-5-8459-1814-7, 978-1-43-024233-8.
4. Либерти Дж. Программирование на С# [Текст] : пер. с англ. / Дж. Либерти. – 2-е изд. – СПб. ; М. : Символ-Плюс, 2003. – 688 с.
5. Microsoft Developer Network. Библиотека MSDN. Разработка на .NET [Электронный ресурс] / MSDN – сеть разработчиков Microsoft. – Режим доступа : www/URL:https://msdn.microsoft.com/ru-ru/ - 24.05.2017 г. – Загл. с экрана.

Додаток А

Код основної форми

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.IO;

using System.Xml.Serialization;

namespace WindowsFormsApplication2

{

public partial class Form1 : Form

{

public static List<person> table = new List<person>();

public Form1()

{

InitializeComponent();

}

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

{

if (Data.Ch == 1 && (Data.Row < 0 || Data.Row >= table.Count))

{

person p1 = new person(Data.Name, Data.Adress, Data.Job, Data.Education, Data.Rank, Data.DateRank, Data.Post, Data.Unit, Data.ServiceForm, Data.ServicePeriod, Data.Character);

table.Add(p1);

Data.Ch = 0;

}

else if (Data.Row >= 0 && Data.Row < table.Count)

{

table[Data.Row] = new person(Data.Name, Data.Adress, Data.Job, Data.Education, Data.Rank, Data.DateRank, Data.Post, Data.Unit, Data.ServiceForm, Data.ServicePeriod, Data.Character);

Data.Row = -1;

}

try

{

for (int i = 0; i < table.Count; i++)

{

dataGridView1.Rows.Insert(i, table[i].Name, table[i].Adress, table[i].Job, table[i].Education, table[i].Rank, table[i].DateRank, table[i].Post, table[i].Unit, table[i].ServiceForm, table[i].ServicePeriod, table[i].Character);

}

}

catch (Exception ex)

{

MessageBox.Show(ex.Message, "Повідомлення", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

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

{

try

{

if (openFileDialog1.ShowDialog() == DialogResult.Cancel)

return;

XmlSerializer formatter = new XmlSerializer(typeof(List<person>));

using (FileStream fs = new FileStream(openFileDialog1.FileName, FileMode.OpenOrCreate))

{

table = (List<person>)formatter.Deserialize(fs);

dataGridView1.Rows.Clear();

for (int i = 0; i < table.Count; i++)

{

dataGridView1.Rows.Insert(i, table[i].Name, table[i].Adress, table[i].Job, table[i].Education,

table[i].Rank, table[i].DateRank, table[i].Post, table[i].Unit, table[i].ServiceForm,

table[i].ServicePeriod, table[i].Character);

}

}

}

catch (Exception ex)

{

MessageBox.Show(ex.Message, "Повідомлення", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

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

{

if (saveFileDialog1.ShowDialog() == DialogResult.Cancel)

{

return;

}

XmlSerializer formatter = new XmlSerializer(typeof(List<person>));

using (FileStream fs = new FileStream(saveFileDialog1.FileName, FileMode.OpenOrCreate))

{

formatter.Serialize(fs, table);

}

}

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

{

int indexRow = e.RowIndex;

if (indexRow >= 0 && indexRow < table.Count)

{

Data.Name = table[indexRow].Name;

Data.Adress = table[indexRow].Adress;

Data.Job = table[indexRow].Job;

Data.Education = table[indexRow].Education;

Data.Rank = table[indexRow].Rank;

Data.DateRank = table[indexRow].DateRank;

Data.Post = table[indexRow].Post;

Data.Unit = table[indexRow].Unit;

Data.ServiceForm = table[indexRow].ServiceForm;

Data.ServicePeriod = table[indexRow].ServicePeriod;

Data.Character = table[indexRow].Character;

}

}

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

{

Data.Row = -1;

Form2 f = new Form2();

f.Show();

Hide();

}

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

{

DialogResult res = MessageBox.Show( "Ви впевнені, що хочете видалити цей рядок?",

"Попередження", MessageBoxButtons.OKCancel);

if (res == DialogResult.OK)

{

int row = dataGridView1.CurrentCell.RowIndex;

dataGridView1.Rows.RemoveAt(row);

table.RemoveAt(row);

}

}

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

{

Data.Row = dataGridView1.CurrentCell.RowIndex;

Form2 f = new Form2();

f.Show();

Hide();

}

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

{

if (txtSearch.Text != "")

{

Data.Search = txtSearch.Text;

Data.WhatSearch = "all";

Form3 f = new Form3();

f.Show();

}

}

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

{

if (txtUnitSearch.Text != "")

{

Data.Search = txtUnitSearch.Text;

Data.WhatSearch = "unit";

Form3 f = new Form3();

f.Show();

}

}

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

{

if (txtRankSearch.Text != "")

{

Data.Search = txtRankSearch.Text;

Data.WhatSearch = "rank";

Form3 f = new Form3();

f.Show();

}

}

}

}

Додаток Б

Код основної форми додавання та редагування

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 static WindowsFormsApplication2.Form1;

namespace WindowsFormsApplication2

{

public partial class Form2 : Form

{

public Form2()

{

InitializeComponent();

}

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

{

Form1 f = new Form1();

Data.Name = textBox1.Text;

Data.Adress = textBox2.Text;

Data.Job = textBox3.Text;

Data.Education = textBox4.Text;

Data.Rank = textBox5.Text;

Data.DateRank = textBox11.Text;

Data.Post = textBox6.Text;

Data.Unit = textBox7.Text;

Data.ServiceForm = comboBox1.Text;

Data.ServicePeriod = textBox9.Text;

Data.Character = textBox10.Text;

Data.Ch = 1;

f.Show();

this.Hide();

}

private void Form2\_FormClosing(object sender, FormClosingEventArgs e)

{

Form1 f = new Form1();

f.Show();

}

private void Form2\_Load(object sender, EventArgs e)

{

if (Data.Row >= 0 && Data.Row < table.Count)

{

textBox1.Text = Data.Name;

textBox2.Text = Data.Adress;

textBox3.Text = Data.Job;

textBox4.Text = Data.Education;

textBox5.Text = Data.Rank;

textBox6.Text = Data.Post;

textBox7.Text = Data.Unit;

comboBox1.Text = Data.ServiceForm;

textBox9.Text = Data.ServicePeriod;

textBox10.Text = Data.Character;

textBox11.Text = Data.DateRank;

}

}

private void textBox1\_TextChanged(object sender, EventArgs e)

{

}

}

}

Додаток С

Код форми формування списків

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.IO;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using System.Xml.Serialization;

using static WindowsFormsApplication2.Form1;

namespace WindowsFormsApplication2

{

public partial class Form3 : Form

{

static List<person> table1 = new List<person>();

public Form3()

{

InitializeComponent();

}

private void Form3\_Load(object sender, EventArgs e)

{

if (Data.WhatSearch == "all")

{

for (int i = 0; i < table.Count; i++)

{

if (table[i].Name.Contains(Data.Search)) { dataGridView1.Rows.Add(table[i].Name, table[i].Adress, table[i].Job, table[i].Education, table[i].Rank, table[i].DateRank, table[i].Post, table[i].Unit, table[i].ServiceForm, table[i].ServicePeriod, table[i].Character); Table1Add(i); continue; }

if (table[i].Adress.Contains(Data.Search)) { dataGridView1.Rows.Add(table[i].Name, table[i].Adress, table[i].Job, table[i].Education, table[i].Rank, table[i].DateRank, table[i].Post, table[i].Unit, table[i].ServiceForm, table[i].ServicePeriod, table[i].Character); Table1Add(i); continue; }

if (table[i].Job.Contains(Data.Search)) { dataGridView1.Rows.Add(table[i].Name, table[i].Adress, table[i].Job, table[i].Education, table[i].Rank, table[i].DateRank, table[i].Post, table[i].Unit, table[i].ServiceForm, table[i].ServicePeriod, table[i].Character); Table1Add(i); continue; }

if (table[i].Education.Contains(Data.Search)) { dataGridView1.Rows.Add(table[i].Name, table[i].Adress, table[i].Job, table[i].Education, table[i].Rank, table[i].DateRank, table[i].Post, table[i].Unit, table[i].ServiceForm, table[i].ServicePeriod, table[i].Character); Table1Add(i); continue; }

if (table[i].Rank.Contains(Data.Search)) { dataGridView1.Rows.Add(table[i].Name, table[i].Adress, table[i].Job, table[i].Education, table[i].Rank, table[i].DateRank, table[i].Post, table[i].Unit, table[i].ServiceForm, table[i].ServicePeriod, table[i].Character); Table1Add(i); continue; }

if (table[i].DateRank.Contains(Data.Search)) { dataGridView1.Rows.Add(table[i].Name, table[i].Adress, table[i].Job, table[i].Education, table[i].Rank, table[i].DateRank, table[i].Post, table[i].Unit, table[i].ServiceForm, table[i].ServicePeriod, table[i].Character); Table1Add(i); continue; }

if (table[i].Post.Contains(Data.Search)) { dataGridView1.Rows.Add(table[i].Name, table[i].Adress, table[i].Job, table[i].Education, table[i].Rank, table[i].DateRank, table[i].Post, table[i].Unit, table[i].ServiceForm, table[i].ServicePeriod, table[i].Character); Table1Add(i); continue; }

if (table[i].Unit.Contains(Data.Search)) { dataGridView1.Rows.Add(table[i].Name, table[i].Adress, table[i].Job, table[i].Education, table[i].Rank, table[i].DateRank, table[i].Post, table[i].Unit, table[i].ServiceForm, table[i].ServicePeriod, table[i].Character); Table1Add(i); continue; }

if (table[i].ServiceForm.Contains(Data.Search)) { dataGridView1.Rows.Add(table[i].Name, table[i].Adress, table[i].Job, table[i].Education, table[i].Rank, table[i].DateRank, table[i].Post, table[i].Unit, table[i].ServiceForm, table[i].ServicePeriod, table[i].Character); Table1Add(i); continue; }

if (table[i].ServicePeriod.Contains(Data.Search)) { dataGridView1.Rows.Add(table[i].Name, table[i].Adress, table[i].Job, table[i].Education, table[i].Rank, table[i].DateRank, table[i].Post, table[i].Unit, table[i].ServiceForm, table[i].ServicePeriod, table[i].Character); Table1Add(i); continue; }

if (table[i].Character.Contains(Data.Search)) { dataGridView1.Rows.Add(table[i].Name, table[i].Adress, table[i].Job, table[i].Education, table[i].Rank, table[i].DateRank, table[i].Post, table[i].Unit, table[i].ServiceForm, table[i].ServicePeriod, table[i].Character); Table1Add(i); continue; }

}

}

else if (Data.WhatSearch == "unit")

{

for (int i = 0; i < table.Count; i++)

{

if (table[i].Unit == (Data.Search)) { dataGridView1.Rows.Add(table[i].Name, table[i].Adress, table[i].Job, table[i].Education, table[i].Rank, table[i].DateRank, table[i].Post, table[i].Unit, table[i].ServiceForm, table[i].ServicePeriod, table[i].Character); Table1Add(i); }

}

}

else if (Data.WhatSearch == "rank")

{

for (int i = 0; i < table.Count; i++)

{

if (table[i].Rank == Data.Search) { dataGridView1.Rows.Add(table[i].Name, table[i].Adress, table[i].Job, table[i].Education, table[i].Rank, table[i].DateRank, table[i].Post, table[i].Unit, table[i].ServiceForm, table[i].ServicePeriod, table[i].Character); Table1Add(i); }

}

}

}

private void Table1Add(int i)

{

person p1 = new person(table[i].Name, table[i].Adress, table[i].Job, table[i].Education, table[i].Rank, table[i].DateRank, table[i].Post, table[i].Unit, table[i].ServiceForm, table[i].ServicePeriod, table[i].Character);

table1.Add(p1);

}

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

{

if (saveFileDialog1.ShowDialog() == DialogResult.Cancel)

{

return;

}

XmlSerializer formatter = new XmlSerializer(typeof(List<person>));

using (FileStream fs = new FileStream(saveFileDialog1.FileName, FileMode.OpenOrCreate))

{

formatter.Serialize(fs, table1);

}

}

private void menuStrip1\_ItemClicked(object sender, ToolStripItemClickedEventArgs e)

{

}

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

{

DialogResult res = MessageBox.Show("Ви впевнені, що хочете видалити цей рядок?", "Попередження", MessageBoxButtons.OKCancel);

if (res == DialogResult.OK)

{

int row = dataGridView1.CurrentCell.RowIndex;

dataGridView1.Rows.RemoveAt(row);

table1.RemoveAt(row);

}

}

}

}