Національний університет “Львівська політехніка”

Кафедра програмного забезпечення

**КУРСОВА РОБОТА**

**з дисципліни «Об’єктно-орієнтоване програмування»**

**На тему:***«Робота з списком бійців»*

Студента групи ПЗ-23

спеціальності 6.121

“Інженерія ПЗ”

Борковича А. М.

Керівник: доцент кафедри ПЗ,

к.т.н., доцент Коротєєва Т. О.

Національна шкала \_\_\_\_\_\_\_\_\_\_\_\_\_

Кількість балів\_\_\_ Оцінка ECTS\_\_\_

Члени комісії          \_\_\_\_\_\_\_\_\_\_\_  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Львів 2022

ЗАВДАННЯ

на курсову роботу з дисципліни «Об’єктно-орієнтоване програмування»

студента групи ПЗ-23 Борковича Андрія

**Тема: «*Робота з списком бійців*»**

**Завдання**: (Варіант 3)

Створити таблицю у візуальному середовищі:

№ | Прізвище | Ім‘я | Вік | Група крові | Звання | Боєкомплект |

1) Порахувати кількість одиниць озброєння виданого бійцям, згрупувавши бійців по званню.

2) Визначити середній вік бійців для кожного звання і загалом по батальйону.

3) Вивести бійців які мають більше однієї одиниці зброї, бійців які мають АК-47 і гранатомет, які мають пістолет і будь-який автомат.

4) Вивести бійців які мають першу групу крові резус мінус і четверту групу резус плюс віком до 25 років.

5) Вивести рядових (солдат) віком до 20 років без боєкомплекту.

6) Визначити донорів для кожної групи крові.

Для класу створити:

1) Конструктор за замовчуванням;

2) Конструктор з параметрами;

3) конструктор копій;

4) перевизначити операції >>, << (- ,+) для зчитування та запису у файл.

Для демонстрації роботи програми використати засоби візуального середовища програмування.

**Зміст завдання та календарний план його виконання**

|  |  |  |
| --- | --- | --- |
| **№ з/п** | **Зміст завдання** | **Дата** |
| 1 | Здійснити аналітичний огляд літератури за заданою темою та обгрунтувати вибір інструментальних засобів реалізації. | 15.09 |
| 2 | Побудова UML діаграм | 23.09 |
| 3 | Розробка алгоритмів реалізації | 28.09 |
| 4 | Реалізація завдання (кодування) | 2.10 |
| 5 | Формування інструкції користувача | 19.10 |
| 6 | Оформлення звіту до курсової роботи згідно з вимогами Міжнародних стандартів, дотримуючись такої структури:   * зміст; * алгоритм розв‘язку задачі у покроковому представленні; * діаграми UML класів, прецедентів, послідовності виконання; * код розробленої програми з коментарями; * протокол роботи програми для кожного пункту завдання * інструкція користувача та системні вимоги; * опис виняткових ситуацій; * структура файлу вхідних даних; * висновки; * список використаних джерел. | 21.10 |

Завдання прийнято до виконання: \_\_\_\_\_\_\_\_\_БАМ\_\_\_\_\_\_\_\_\_\_ (Боркович А. М.) 22.08.2022 р.

Керівник роботи: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/ Коротєєва Т. О./

**Зміст**

1. [Алгоритм розв’язку задачі у покроковому представленні](#алгоритми)………………..5
2. [Діаграми UML класів, прецедентів, послідовності виконання](#діаграми)…………...9
3. [Код розробленої програми](#код)…………………………………………………11
4. [Протокол роботи програми](#протокол)…………………...…………………………...24
5. [Інструкція користувача та системні вимоги](#інструкція)……………………………...32
6. [Опис виняткових ситуацій](#винятковіситуації)…………………………………………………39
7. [Структура файлу вхідних даних](#файл)……………………………………...…...42

[Висновки](#висновок)………………………………………………………………………..43

[Список використаної літератури](#література)……………………………………...………44

**Алгоритм розв‘язку задачі у покроковому представленні**

Алгоритм розв'язку задачі загалом:

STEP1. Зчитати дані з файлу або форми.  
STEP2. Виконати необхідні дії відповідно завдання.

STEP3. Зберегти роботу в тому самому або в новому файлі.

Алгоритм для підрахунку озброєння для бійців певного звання (загальний випадок незалежно від звання):

**Алгоритм CW**

Задано Warriors – список бійців у батальйоні, n – кількість бійців у списку, ammountOfAmmunition – кількість озброєння, warriorAmmunitionCount – кількість озброєння в окремого бійця, і – індекс проходження у циклі.

CW1. Цикл за індексом проходження. Повторювати кроки CW2-CW3 допоки i=1…n;

CW2. Сумуємо warriorAmmunitionCount до ammountOfAmmunition для Warriors[i];

CW3. i = i+1;

CW4. Кінець.

Алгоритм для підрахунку середнього віку для бійців певного звання (загальний випадок незалежно від звання):

**Алгоритм CA**

Задано Warriors – список бійців у батальйоні, n – кількість бійців у списку, warriorAge – вік окемого бійця, ammountOfYears – загальне число років, averageAge – середній вік, і – індекс проходження у циклі.

CA1. Цикл за індексом проходження. Повторювати кроки CA2-CA3 допоки i=1…n;

CA2. Додавання warriorAge до ammountOfYears для Warriors[i];

CA3. i=i+1;

CA4. Вихід з циклу, averageAge = amountOfYears/n;

CA5. Кінець.

Алгоритм для визначення і виводу списку бійців, які мають більше однієї одиниці зброї:

**Алгоритм MOW**

Задано Warriors – список бійців у батальйоні, n – кількість бійців у списку, warriorAmmunitionCount – кількість озброєння в окремого бійця, NewWarriors – новий список бійців за ознакою, і – індекс проходження в циклі.

MOW1. Цикл за індексом проходження. Виконувати кроки MOW2-MOW3 допоки i=1…n;

WMO2. Якщо warriorAmmunitionCount > 1, то додаємо Warriors[i] до NewWarriors;

MOW3. і=і+1;

MOW4. Кінець. Вивести новий список на екран.

Алгоритм для визначення і виводу списку бійців, які мають АК-47 і гранатомет:

**Алгоритм AK47GG**

Задано Warriors – список бійців у батальйоні, n – кількість бійців у списку, warriorAmmunitionList – список озброєння в окремого бійця, NewWarriors – новий список бійців за ознакою, і – індекс проходження в циклі.

AK47GG1. Цикл за індексом проходження. Виконувати кроки AK47GG2-AK47GG3 допоки i=1…n;

AK47GG2. Якщо warriorAmmunitionList містить в собі АК-47 і гранатомет одночасно, то додаємо Warriors[i] до NewWarriors;

AK47GG3. і=і+1;

AK47GG4. Кінець. Вивести новий список на екран.

Алгоритм для визначення і виводу списку бійців, які мають пістолет і будь-який автомат (АК-47, АКМ, РПК, АК-74,АК-101, АК-103, АК-105, ПКМ, ПКТ, М16, М4):

**Алгоритм GR**

Задано Warriors – список бійців у батальйоні, n – кількість бійців у списку, warriorAmmunitionList – список озброєння в окремого бійця, NewWarriors – новий список бійців за ознакою, і – індекс проходження в циклі.

GR1. Цикл за індексом проходження. Виконувати кроки GR2-GR3 допоки i=1…n;

GR2. Якщо warriorAmmunitionList містить один з видів автоматів (див. [тут](#автомати)) і гранатомет одночасно, то додаємо Warriors[i] до NewWarriors;

GR3. і=і+1;

GR4. Кінець. Вивести новий список на екран.

Алгоритм для визначення і виводу списку бійців які мають першу групу крові резус мінус і четверту групу резус плюс віком до 25 років:

**Алгоритм BGA**

Задано Warriors – список бійців у батальйоні, n – кількість бійців у списку, warriorAge – вік окремого бійця, warriorBloodGroup – група крові окремого бійця, NewWarriors – новий список бійців за ознакою, і – індекс проходження в циклі.

BGA1. Цикл за індексом проходження. Виконувати кроки BGA2-BGA3 допоки i=1…n;

BGA2. Якщо warriorAge < 25 і warriorBloodGroup = О- (1 група резус мінус) або warriorBloodGroup=AB+ (4 група резус плюс), то додаємо Warriors[i] до NewWarriors;

BGA3. і=і+1;

BGA4. Кінець. Вивести новий список на екран.

Алгоритм для визначення і виводу списку солдат віком до 20 років без боєкомплекту:

**Алгоритм SNW**

Задано Warriors – список бійців у батальйоні, n – кількість бійців у списку, warriorRank – звання окремого бійця, warriorAge – вік бійця, warriorAmmunitionCount – кількість озброєння в окремого бійця, NewWarriors – новий список бійців за ознакою, і – індекс проходження в циклі.

SNW1. Цикл за індексом проходження. Виконувати кроки SNW2- SNW3 допоки i=1…n;

SNW2. Якщо warriorRank є значення «Солдат», warriorAge < 20 та warriorAmmunitionCount=0, то додаємо Warriors[i] до NewWarriors;

SNW3. і=і+1;

SNW4. Кінець. Вивести новий список на екран.

Алгоритм для визначення і виводу донорів для кожної групи крові.

**Алгоритм FD**

Задано Warriors – список бійців у батальйоні, n – кількість бійців у списку,

warriorBloodGroup – група крові в окремого бійця, donorsForFirstGroup – донори для 1 групи крові, donorsForSecondGroup – для 2 групи, donorsForThirdGroup – для 3, donorsForFourthGroup – для 4, і - індекс проходження в циклі.

FD1. Цикл за індексом проходження. Виконувати кроки FD2-FD6 допоки i=1…n;

FD2. Якщо warriorBloodGroup дорівнює 1+ або 1-, то додаємо Warriors[i] до donorsForFirstGroup;

FD3. Якщо warriorBloodGroup дорівнює 2+ або 2-, то додаємо Warriors[i] до donorsForSecondGroup;

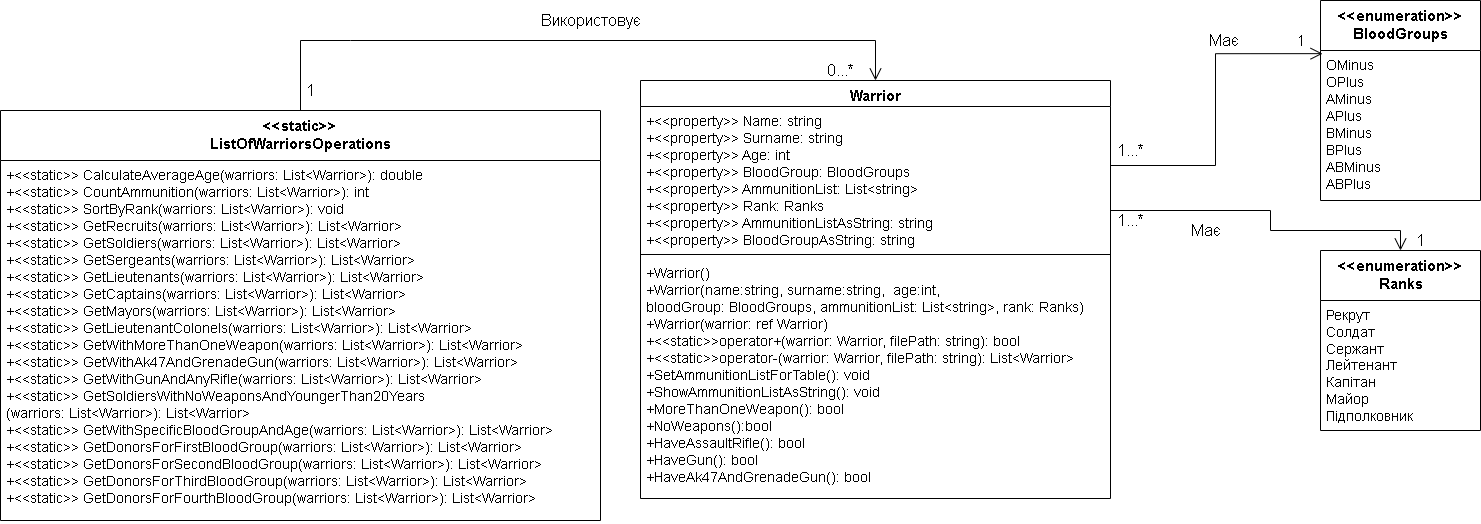
FD4. Якщо warriorBloodGroup дорівнює 3+ або 3-, то додаємо Warriors[i] до donorsForThirdGroup;

FD5. Якщо warriorBloodGroup дорівнює 4+ або 4-, то додаємо Warriors[i] до donorsForFourthGroup;

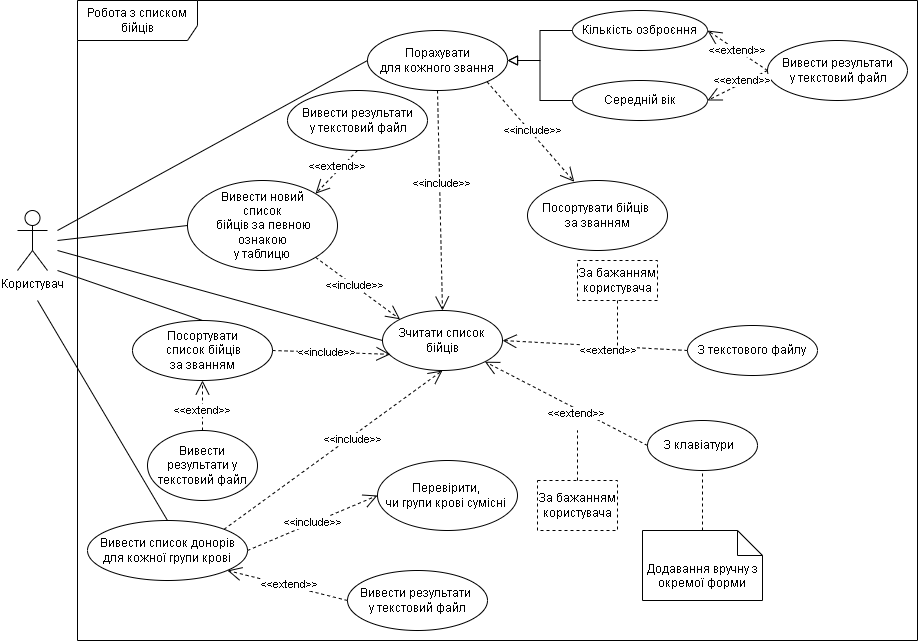
FD6. i = i+1;

FD6. Кінець. Вивести сформовані списки на екран.

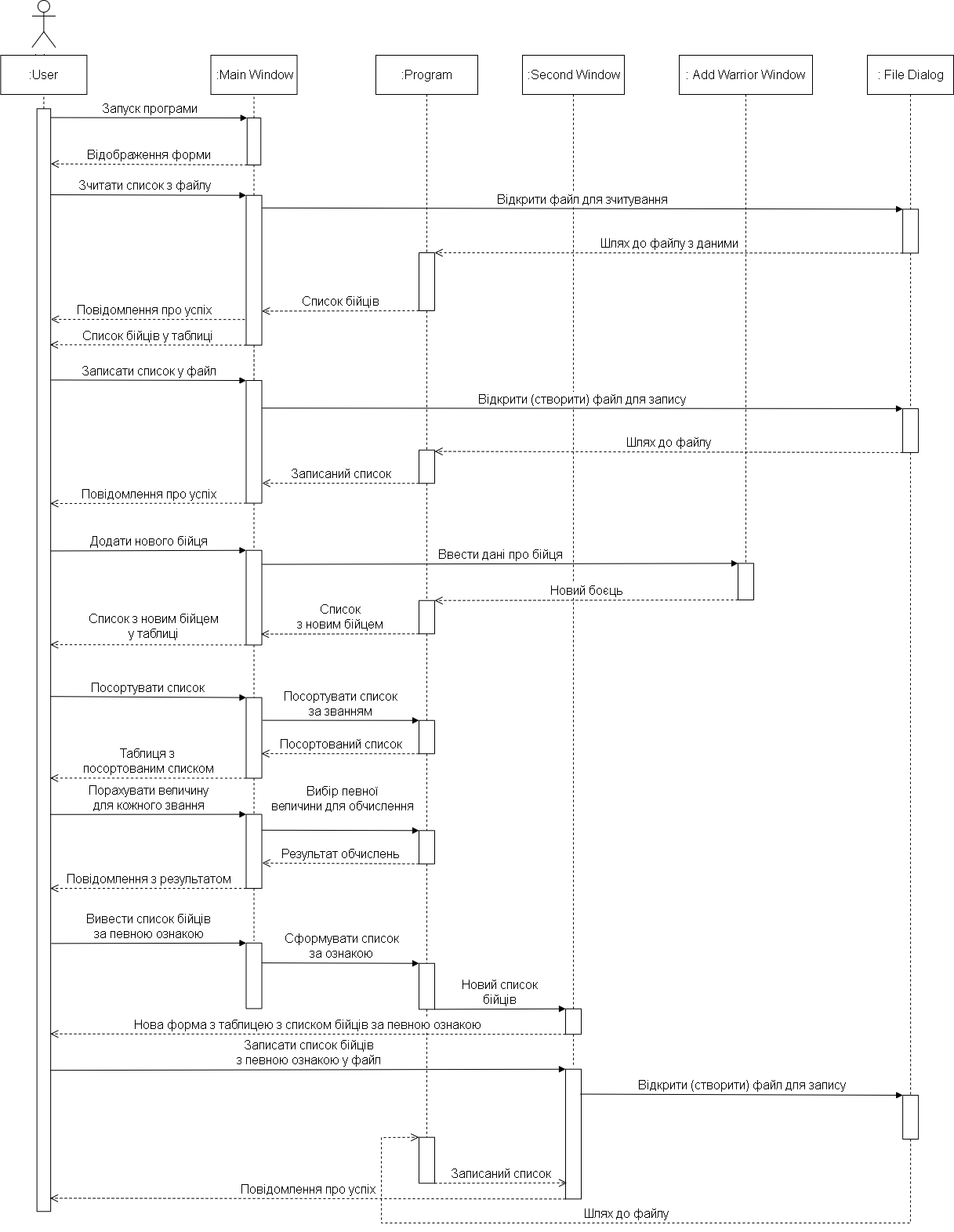
**Діаграми UML класів, прецедентів, послідовності виконання**



*Рис. 2.1. Діаграма класів*



*Рис.2.2. Діаграма прецедентів*



*Рис. 2.3. Діаграма послідовності виконання*

**Код розробленої програми**

**Warrior.cs**

using System;

using System.Collections.Generic;

using System.IO;

using System.Linq;

using System.ComponentModel.DataAnnotations;

namespace CourseWork

{

//перелічення зі званнями в батальйоні

public enum Ranks

{

Рекрут = 1,

Солдат = 2,

Сержант = 3,

Лейтенант = 4,

Капітан = 5,

Майор = 6,

Підполковник = 7

}

//перелічення з групами крові (разом з резус-факторами)

public enum BloodGroups

{

//1 група

[Display(Name = "O-")] OMinus = -1,

[Display(Name = "O+")] OPlus = +1,

//2 група

[Display(Name = "A-")] AMinus = -2,

[Display(Name = "A+")] APlus = +2,

//3 група

[Display(Name = "B-")] BMinus = -3,

[Display(Name = "B+")] BPlus = +3,

//4 група

[Display(Name = "AB-")] AbMinus = -4,

[Display(Name = "AB+")] AbPlus = +4

}

//клас Боєць

class Warrior

{

public string Name { get; set; } //ім'я бійця

public string Surname { get; set; } //прізвище бійця

public int Age { get; set; } //вік бійця

public BloodGroups BloodGroup { get; set; } //група крові

public List<string> AmmunitionList { get; set; } //боєкомплект (список зброї)

public Ranks Rank { get; set; } //звання бійця в батальйоні

public string AmmunitionListAsString { get; set; } //боєкомплект у вигляді стрічки (для коректного відображення у таблиці)

public string BloodGroupAsString { get; set; } //група крові у вигляді стрічки (для коректного відображення у таблиці)

//конструктор за замовчуванням

public Warrior()

{

Name = string.Empty;

Surname = string.Empty;

Age = 18;

BloodGroup = BloodGroups.OPlus;

AmmunitionList = new List<string>();

Rank = Ranks.Рекрут;

}

//конструктор з параметрами

public Warrior(string name, string surname, int age,

BloodGroups bloodGroup, List<string> ammunitionList, Ranks rank)

{

Name = name;

Surname = surname;

Age = age;

BloodGroup = bloodGroup;

DisplayAttribute bloodGroupAttribute = BloodGroup.GetDisplayAttributesFrom(typeof(BloodGroups));

BloodGroupAsString = bloodGroupAttribute.Name;

AmmunitionList = ammunitionList;

SetAmmunitionListForTable();

Rank = rank;

}

//конструктор копій

public Warrior(ref Warrior warrior)

{

Name = warrior.Name;

Surname = warrior.Surname;

Age = warrior.Age;

BloodGroup = warrior.BloodGroup;

DisplayAttribute bloodGroupAttribute = BloodGroup.GetDisplayAttributesFrom(typeof(BloodGroups));

BloodGroupAsString = bloodGroupAttribute.Name;

AmmunitionList = warrior.AmmunitionList;

SetAmmunitionListForTable();

Rank = warrior.Rank;

}

//формуємо стрічку з боєкомлекту

public void SetAmmunitionListForTable()

{

AmmunitionListAsString = string.Join(", ", AmmunitionList.ToArray());//розділяємо кожен вид зброї з списку боєкомлекту комою

}

//перевірка чи боєць має більше 1 одиниці в своєму боєкомплекті

public bool MoreThanOneWeapon()

{

return AmmunitionList.Count > 1;

}

//перевірка чи боєць не має взагалі зброї в своєму боєкомплекті

public bool NoWeapons()

{

return AmmunitionList.Count == 0;

}

//перевірка наявності у бійця будь-якого автомата

public bool HaveAssaultRifle()

{

//масив можливих автоматів

string[] rifles =

{

"АК-47", "АКМ", "РПК", "АК-74",

"АК-101", "АК-103", "АК-105", "ПКМ", "ПКТ", "М16", "М4"

};

bool isRifle = false;

foreach (var rifle in rifles)

if (AmmunitionList.Contains(rifle))//перевіряємо чи в боєкомплекті бійця є один з зазначених вище автоматів

isRifle = true;

return isRifle;

}

//перевірка наявності у бійця пістолета

public bool HaveGun()

{

return AmmunitionList.Contains("Пістолет");

}

//перевірка наявності у бійця і АК-47 і гранатомета

public bool HaveAk47AndGrenadeGun()

{

return AmmunitionList.Contains("АК-47") && AmmunitionList.Contains("Гранатомет");

}

//перевантаження оператора для виводу у файл 1 бійця

public static bool operator +(Warrior warrior, string filePath)

{

string warriorLine = string.Empty;//стрічка (рядок), яка зберігатиме дані про бійця

warriorLine += $"{warrior.Surname,14}\t {warrior.Name,12}\t {warrior.Age,2}\t{warrior.BloodGroupAsString,4}\t{warrior.Rank,16}\t{warrior.AmmunitionListAsString,25}\n";

File.AppendAllText(filePath, warriorLine);

return true;

}

//перевантаження оператора для зчитування з файлу списку бійців

public static List<Warrior> operator -(Warrior warrior, string filePath)

{

List<Warrior> warriors = new List<Warrior>();

List<string> lines = File.ReadAllLines(filePath).ToList();//формуємо список зчитаних рядків

int lineCounter = 0;

if (lines.Count == 0)

throw new Exception("Файл пустий!\nВиберіть інший файл!");

foreach (var line in lines)

{

Warrior newWarrior = new Warrior();

string[] entries = line.Split(' '); //формуємо масив стрічок в рядку, які розділені пробілом

if (!entries[0].All(char.IsLetter) && !entries[0].Contains("'"))

throw new Exception($"Прізвище в рядку {lineCounter + 1} містить заборонені символи!\nБудь ласка, відредагуйте цей файл або виберіть інший");

if ((!entries[1].All(char.IsLetter)) && !entries[1].Contains("'"))

throw new Exception($"Ім'я в рядку {lineCounter + 1} містить заборонені символи!\nБудь ласка, відредагуйте цей файл або виберіть інший");

newWarrior.Surname = entries[0];

newWarrior.Name = entries[1];

newWarrior.Age = Convert.ToInt32(entries[2]);

if (newWarrior.Age < 18 || newWarrior.Age > 65)

throw new Exception($"Вік бійця {newWarrior.Surname} {newWarrior.Name} не придатний для служби!");

newWarrior.BloodGroup = (BloodGroups)Enum.Parse(typeof(BloodGroups), entries[3]);

DisplayAttribute bloodGroup = newWarrior.BloodGroup.GetDisplayAttributesFrom(typeof(BloodGroups));//доступаємось до атрибуту Name для поточної групи крові

newWarrior.BloodGroupAsString = bloodGroup.Name;

newWarrior.Rank = (Ranks)Enum.Parse(typeof(Ranks), entries[4]);

if (entries[5] == "")

newWarrior.AmmunitionList.Capacity = 0;

else

{

newWarrior.AmmunitionList = entries[5].Split(',').ToList();

newWarrior.SetAmmunitionListForTable();

}

warriors.Add(new Warrior(ref newWarrior));

lineCounter++;

}

return warriors;

}

}

}

**ListOfWarriorsOperations.cs**

using System;

using System.Collections.Generic;

using System.Linq;

namespace CourseWork

{

//статичний клас для проведення різних операцій з списком бійців (об'єктів класу Warrior)

static class ListOfWarriorsOperations

{

//обчислення середнього віку серед бійців

public static double CalculateAverageAge(List<Warrior> warriors)

{

double averageAge = 0;

foreach (var warrior in warriors) averageAge += warrior.Age;

return Math.Round((averageAge / warriors.Count), 2);

}

//обчислення кількості озброєння для бійців

public static int CountAmmunition(List<Warrior> warriors)

{

int amountOfAmmunition = 0;

foreach (var warrior in warriors) amountOfAmmunition += warrior.AmmunitionList.Count;

return amountOfAmmunition;

}

//посортувати бійців за званням

public static void SortByRank(List<Warrior> warriors)

{

int n = warriors.Count;

for (int i = 0; i < n - 1; i++)

for (int j = i + 1; j < n; j++)

if (warriors[j].Rank < warriors[i].Rank)

(warriors[j], warriors[i]) = (warriors[i], warriors[j]);

}

//отримати список рекрутів

public static List<Warrior> GetRecruits(List<Warrior> warriors)

{

return warriors.Where(warrior => warrior.Rank.Equals(Ranks.Рекрут)).ToList();

}

//отримати список солдатів

public static List<Warrior> GetSoldiers(List<Warrior> warriors)

{

return warriors.Where(warrior => warrior.Rank.Equals(Ranks.Солдат)).ToList();

}

//отримати список сержантів

public static List<Warrior> GetSergeants(List<Warrior> warriors)

{

return warriors.Where(warrior => warrior.Rank.Equals(Ranks.Сержант)).ToList();

}

//отримати список лейтинантів

public static List<Warrior> GetLieutenants(List<Warrior> warriors)

{

return warriors.Where(warrior => warrior.Rank.Equals(Ranks.Лейтенант)).ToList();

}

//отримати список капітанів

public static List<Warrior> GetCaptains(List<Warrior> warriors)

{

return warriors.Where(warrior=> warrior.Rank.Equals(Ranks.Капітан)).ToList();

}

//отримати список майорів

public static List<Warrior> GetMayors(List<Warrior> warriors)

{

return warriors.Where(warrior => warrior.Rank.Equals(Ranks.Майор)).ToList();

}

//отримати список підполковників

public static List<Warrior> GetLieutenantColonels(List<Warrior> warriors)

{

return warriors.Where(warrior => warrior.Rank.Equals(Ranks.Підполковник)).ToList();

}

//визначення бійців, які мають більше однієї одиниці зброї

public static List<Warrior> GetWithMoreThanOneWeapon(List<Warrior> warriors)

{

return warriors.Where(warrior => warrior.MoreThanOneWeapon()).ToList();

}

//визначення бійців, які мають АК-47 і гранатомет

public static List<Warrior> GetWithAk47AndGrenadeGun(List<Warrior> warriors)

{

return warriors.Where(warrior => warrior.HaveAk47AndGrenadeGun()).ToList();

}

//визначення бійців, які мають пістолет і будь-який автомат

public static List<Warrior> GetWithGunAndAnyRifle(List<Warrior> warriors)

{

return warriors.Where(warrior => warrior.HaveAssaultRifle() && warrior.HaveGun()).ToList();

}

//визначення солдат, які не мають озброєння віком до 20 років

public static List<Warrior> GetSoldiersWithNoWeaponsAndYoungerThan20Years(List<Warrior> warriors)

{

return warriors.Where(warrior => warrior.Age < 20 && warrior.NoWeapons() && warrior.Rank.Equals(Ranks.Солдат)).ToList();

}

//визначення бійців, які мають першу групу крові резус мінус і четверту групу резус плюс віком до 25 років.

public static List<Warrior> GetWithSpecificBloodGroupAndAge(List<Warrior> warriors)

{

return warriors.Where(warrior => warrior.Age < 25 && (warrior.BloodGroup.Equals(BloodGroups.OMinus) || warrior.BloodGroup.Equals(BloodGroups.AbPlus))).ToList();

}

//визначити донорів для 1 групи крові

public static List<Warrior> GetDonorsForFirstBloodGroup(List<Warrior> warriors)

{

return warriors.Where(warrior => warrior.BloodGroup.Equals(BloodGroups.OMinus) || warrior.BloodGroup.Equals(BloodGroups.OPlus)).ToList();

}

//визначити донорів для 2 групи крові

public static List<Warrior> GetDonorsForSecondBloodGroup(List<Warrior> warriors)

{

return warriors.Where(warrior => warrior.BloodGroup.Equals(BloodGroups.AMinus) || warrior.BloodGroup.Equals(BloodGroups.APlus)).ToList();

}

//визначити донорів для 3 групи крові

public static List<Warrior> GetDonorsForThirdBloodGroup(List<Warrior> warriors)

{

return warriors.Where(warrior => warrior.BloodGroup.Equals(BloodGroups.BMinus) || warrior.BloodGroup.Equals(BloodGroups.BPlus)).ToList();

}

//визначити донорів для 4 групи крові

public static List<Warrior> GetDonorsForFourthBloodGroup(List<Warrior> warriors)

{

return warriors.Where(warrior => warrior.BloodGroup.Equals(BloodGroups.AbMinus) || warrior.BloodGroup.Equals(BloodGroups.AbPlus)).ToList();

}

}

}

**MainWindow.xaml.cs**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.IO;

using System.Windows;

using System.Windows.Controls;

using Microsoft.Win32;

namespace CourseWork

{

public partial class MainWindow

{

public MainWindow()

{

InitializeComponent();

Warriors = new List<Warrior>();

IsListSaved = true;

}

internal List<Warrior> Warriors;

internal string FilePathToWrite;

internal bool IsListSaved;//змінна, яка визначає чи список було збережено у файл

//обробник події закриття вікна

private void MainWindow\_OnClosing(object sender, CancelEventArgs e)

{

if (!IsListSaved || Warriors.Count != 0)

{

MessageBoxResult clickResult = MessageBox.Show("Список не записаний у файл! Дійсно бажаєте вийти?", "Увага!",

MessageBoxButton.YesNo, MessageBoxImage.Information);

if (clickResult == MessageBoxResult.No) e.Cancel = true;

}

}

//зчитати список бійців з файлу

private void ReadFromFile\_OnClick(object sender, RoutedEventArgs e)

{

OpenFileDialog openFileDialog = new OpenFileDialog

{

Filter = "Text files (\*.txt)|\*.txt|All files (\*.\*)|\*.\*",

Multiselect = true,

CheckFileExists = true,

Title = "Вибір списку бійців"

};

try

{

string filePathToRead;

if (openFileDialog.ShowDialog() == true)

{

filePathToRead = openFileDialog.FileName;

if (!filePathToRead.Contains("txt"))

throw new Exception(

"Вибрано не текстовий файл\nБудь ласка, виберіть файл з потрібним розширенням - txt");

}

else

throw new Exception("Не вдалося відкрити файл!\nСпробуйте ще раз");

Warrior warrior = new();

Warriors.AddRange(warrior - filePathToRead);

table.ItemsSource = Warriors;

table.Items.Refresh();

statusBar.Text = "Список бійців успішно зчитано з файлу!";

MessageBox.Show(this, "Список бійців було успішно зчитано з файлу!", "Вітання!", MessageBoxButton.OKCancel,

MessageBoxImage.Information);

IsListSaved = false;

}

catch (Exception exception)

{

MessageBox.Show(exception.Message, "Помилка", MessageBoxButton.OKCancel, MessageBoxImage.Error);

}

}

//вивід списку бійців у файл

private void MiWriteToFile\_OnClick(object sender, RoutedEventArgs e)

{

if (Warriors.Count == 0)

{

MessageBox.Show(this, "Список бійців пустий!\nНема що записувати у файл!", "Увага!",

MessageBoxButton.OKCancel, MessageBoxImage.Warning);

return;

}

SaveFileDialog saveFileDialog = new()

{

Filter = "Text files (\*.txt)|\*.txt|All files (\*.\*)|\*.\*",

Title = "Зберегти список бійців"

};

try

{

if (saveFileDialog.ShowDialog() == true)

{

FilePathToWrite = saveFileDialog.FileName;

if (!FilePathToWrite.Contains("txt"))

throw new Exception(

"Вибрано не текстовий файл\nБудь ласка, виберіть файл з потрібним розширенням - txt");

var result = false;

string columnsTitles = "| Прізвище | Ім‘я |Вік|Група крові| Звання | Боєкомплект | \n";

File.AppendAllText(FilePathToWrite, columnsTitles);

foreach (var warrior in Warriors) result = warrior + FilePathToWrite;

IsListSaved = result;

statusBar.Text = "Список бійців було успішно записано у файл!";

MessageBox.Show("Список бійців було успішно записано до файлу!", "Вітання", MessageBoxButton.OKCancel,

MessageBoxImage.Information);

}

else

throw new Exception(

"Не вдалося створити файл для збереження списку!\nБудь ласка, спробуйте ще раз");

}

catch (Exception exception)

{

MessageBox.Show(exception.Message, "Помилка", MessageBoxButton.OKCancel, MessageBoxImage.Error);

}

}

//сортування бійців за званням

private void MiSortByRank\_OnClick(object sender, RoutedEventArgs e)

{

if (Warriors.Count == 0)

{

MessageBox.Show("Список пустий!\nБудь ласка, наповніть його!", "Увага!", MessageBoxButton.OK,

MessageBoxImage.Warning);

return;

}

ListOfWarriorsOperations.SortByRank(Warriors);

MessageBox.Show(this, "Список бійців було успішно посортовано!", "Вітання!", MessageBoxButton.OKCancel,

MessageBoxImage.Information);

table.Items.Refresh();

table.ItemsSource = Warriors;

statusBar.Text = "Список бійців успішно посортовано за званням!";

if (!string.IsNullOrEmpty(FilePathToWrite))

{

File.AppendAllText(FilePathToWrite, "Список, посортований за званнями:\n");

string columnsTitles = "| Прізвище | Ім‘я |Вік|Група крові| Звання | Боєкомплект | \n";

File.AppendAllText(FilePathToWrite, columnsTitles);

bool result = false;

foreach (var warrior in Warriors) result = warrior + FilePathToWrite;

IsListSaved = result;

}

}

//додавання нового бійця до списку

private void MiAddWarrior\_OnClick(object sender, RoutedEventArgs e)

{

try

{

AddWarriorWindow w = new();

w.Show();

}

catch (Exception exception)

{

MessageBox.Show(exception.Message, "Помилка", MessageBoxButton.OKCancel, MessageBoxImage.Error);

}

}

//підрахувати кількість озброєння виданого бійцям (для кожного звання)

private void CountAmmunition\_OnClick(object sender, RoutedEventArgs e)

{

if (Warriors.Count == 0)

{

MessageBox.Show("Список пустий!\nБудь ласка, наповніть його!", "Увага!", MessageBoxButton.OK,

MessageBoxImage.Warning);

return;

}

int recruitsWeapons = ListOfWarriorsOperations.CountAmmunition(ListOfWarriorsOperations.GetRecruits(Warriors)),

soldiersWeapons = ListOfWarriorsOperations.CountAmmunition(ListOfWarriorsOperations.GetSoldiers(Warriors)),

sergeantsWeapons = ListOfWarriorsOperations.CountAmmunition(ListOfWarriorsOperations.GetSergeants(Warriors)),

lieutenantsWeapons = ListOfWarriorsOperations.CountAmmunition(ListOfWarriorsOperations.GetLieutenants(Warriors)),

captainsWeapons = ListOfWarriorsOperations.CountAmmunition(ListOfWarriorsOperations.GetCaptains(Warriors)),

mayorsWeapons = ListOfWarriorsOperations.CountAmmunition(ListOfWarriorsOperations.GetMayors(Warriors)),

lieutenantsColonelsWeapons = ListOfWarriorsOperations.CountAmmunition(ListOfWarriorsOperations.GetLieutenantColonels(Warriors)),

allWarriorsWeapons = ListOfWarriorsOperations.CountAmmunition(Warriors);

var text = $"Одиниць озброєння:\nРекрути: {recruitsWeapons} од.\nСолдати: {soldiersWeapons} од.\nСержанти: {sergeantsWeapons} од.\n" +

$"Лейтенанти: {lieutenantsWeapons} од.\nКапітани: {captainsWeapons} од.\nМайори: {mayorsWeapons} од.\n" +

$"Підполковники: {lieutenantsColonelsWeapons} од.\nЗагалом: {allWarriorsWeapons} од.\n";

if (!string.IsNullOrEmpty(FilePathToWrite))

File.AppendAllText(FilePathToWrite, text);

MessageBox.Show(text, "Батальйон", MessageBoxButton.OK, MessageBoxImage.Information);

}

//підрахувати середній вік для кожного звання і загалом для батальйону

private void CountAverageAge\_OnClick(object sender, RoutedEventArgs e)

{

if (Warriors.Count == 0)

{

MessageBox.Show("Список пустий!\nБудь ласка, наповніть його!", "Увага!", MessageBoxButton.OK,

MessageBoxImage.Warning);

return;

}

double recruitsAverageAge = ListOfWarriorsOperations.CalculateAverageAge(ListOfWarriorsOperations.GetRecruits(Warriors)),

soldiersAverageAge = ListOfWarriorsOperations.CalculateAverageAge(ListOfWarriorsOperations.GetSoldiers(Warriors)),

sergeantsAverageAge = ListOfWarriorsOperations.CalculateAverageAge(ListOfWarriorsOperations.GetSergeants(Warriors)),

lieutenantsAverageAge = ListOfWarriorsOperations.CalculateAverageAge(ListOfWarriorsOperations.GetLieutenants(Warriors)),

captainsAverageAge = ListOfWarriorsOperations.CalculateAverageAge(ListOfWarriorsOperations.GetCaptains(Warriors)),

mayorsAverageAge = ListOfWarriorsOperations.CalculateAverageAge(ListOfWarriorsOperations.GetMayors(Warriors)),

lieutenantsColonelAverageAge = ListOfWarriorsOperations.CalculateAverageAge(ListOfWarriorsOperations.GetLieutenantColonels(Warriors)),

allWarriorsAverageAge = ListOfWarriorsOperations.CalculateAverageAge(Warriors);

var text = $"Середній вік для:\nРекрутів: {recruitsAverageAge} р.\nСолдатів: {soldiersAverageAge} р.\nСержантів: {sergeantsAverageAge} р.\n" +

$"Лейтенантів: {lieutenantsAverageAge} р.\nКапітанів: {captainsAverageAge} р.\nМайорів: {mayorsAverageAge} р.\n" +

$"Підполковників: {lieutenantsColonelAverageAge} р.\nВсіх бійців: {allWarriorsAverageAge} р.\n";

if(!string.IsNullOrEmpty(FilePathToWrite))

File.AppendAllText(FilePathToWrite, text);

MessageBox.Show(this, text, "Батальйон", MessageBoxButton.OK, MessageBoxImage.Information);

}

//Бійці, які мають більше однієї одиниці зброї

private void Mi1\_OnClick(object sender, RoutedEventArgs e)

{

if (Warriors.Count == 0)

{

MessageBox.Show("Список пустий!\nБудь ласка, наповніть його!", "Увага!", MessageBoxButton.OK,

MessageBoxImage.Warning);

return;

}

if (ListOfWarriorsOperations.GetWithMoreThanOneWeapon(Warriors).Count == 0)

{

MessageBox.Show(this, "Бійців за даною ознакою немає", "Увага!", MessageBoxButton.OKCancel,

MessageBoxImage.Exclamation);

return;

}

SecondWindow warriorsWindow = new SecondWindow

{

labelTitle =

{

Content = "Бійці, які мають більше однієї одиниці зброї"

},

warriorsTable =

{

ItemsSource = ListOfWarriorsOperations.GetWithMoreThanOneWeapon(Warriors)

}

};

warriorsWindow.Show();

}

//Бійці, які мають АК-47 і гранатомет

private void Mi2\_OnClick(object sender, RoutedEventArgs e)

{

if (Warriors.Count == 0)

{

MessageBox.Show("Список пустий!\nБудь ласка, наповніть його!", "Увага!", MessageBoxButton.OK,

MessageBoxImage.Warning);

return;

}

if (ListOfWarriorsOperations.GetWithAk47AndGrenadeGun(Warriors).Count == 0)

{

MessageBox.Show(this, "Бійців за даною ознакою немає", "Увага!", MessageBoxButton.OKCancel,

MessageBoxImage.Exclamation);

return;

}

SecondWindow warriorsWindow = new()

{

warriorsTable =

{

ItemsSource = ListOfWarriorsOperations.GetWithAk47AndGrenadeGun(Warriors)

},

labelTitle =

{

Content = "Бійці, які мають АК-47 і гранатомет"

}

};

warriorsWindow.Show();

}

//Бійці, які мають пістолет і будь-який автомат

private void Mi3\_OnClick(object sender, RoutedEventArgs e)

{

if (Warriors.Count == 0)

{

MessageBox.Show("Список пустий!\nБудь ласка, наповніть його!", "Увага!", MessageBoxButton.OK,

MessageBoxImage.Warning);

return;

}

if (ListOfWarriorsOperations.GetWithGunAndAnyRifle(Warriors).Count == 0)

{

MessageBox.Show("Бійців за даною ознакою немає", "Увага!", MessageBoxButton.OKCancel,

MessageBoxImage.Exclamation);

return;

}

SecondWindow warriorsWindow = new()

{

labelTitle =

{

Content = "Бійці, які мають пістолет і будь-який автомат"

},

warriorsTable =

{

ItemsSource = ListOfWarriorsOperations.GetWithGunAndAnyRifle(Warriors)

}

};

warriorsWindow.Show();

}

//Бійці, які мають першу групу крові резус мінус і четверту групу резус плюс віком до 25 років

private void Mi4\_OnClick(object sender, RoutedEventArgs e)

{

if (Warriors.Count == 0)

{

MessageBox.Show("Список пустий!\nБудь ласка, наповніть його!", "Увага!", MessageBoxButton.OK,

MessageBoxImage.Warning);

return;

}

if (ListOfWarriorsOperations.GetWithSpecificBloodGroupAndAge(Warriors).Count == 0)

{

MessageBox.Show("Бійців за даною ознакою немає", "Увага!", MessageBoxButton.OKCancel,

MessageBoxImage.Exclamation);

return;

}

SecondWindow warriorsWindow = new()

{

labelTitle =

{

Content = "Бійці, які мають першу групу крові резус мінус або четверту групу резус плюс віком до 25 років"

},

warriorsTable =

{

ItemsSource = ListOfWarriorsOperations.GetWithSpecificBloodGroupAndAge(Warriors)

}

};

warriorsWindow.Show();

}

//Солдати віком до 20 років без боєкомплекту

private void Mi5\_OnClick(object sender, RoutedEventArgs e)

{

if (Warriors.Count == 0)

{

MessageBox.Show("Список пустий!\nБудь ласка, наповніть його!", "Увага!", MessageBoxButton.OK,

MessageBoxImage.Warning);

return;

}

if (ListOfWarriorsOperations.GetSoldiersWithNoWeaponsAndYoungerThan20Years(Warriors).Count == 0)

{

MessageBox.Show("Бійців за даною ознакою немає", "Увага!", MessageBoxButton.OKCancel,

MessageBoxImage.Exclamation);

return;

}

SecondWindow warriorsWindow = new()

{

labelTitle =

{

Content = "Cолдати віком до 20 років без боєкомплекту"

},

warriorsTable =

{

ItemsSource = ListOfWarriorsOperations.GetSoldiersWithNoWeaponsAndYoungerThan20Years(Warriors)

}

};

warriorsWindow.Show();

}

//Донори для кожної групи крові

private void Mi6\_OnClick(object sender, RoutedEventArgs e)

{

if (Warriors.Count == 0)

MessageBox.Show("Список пустий!\nБудь ласка, наповніть його!", "Увага!", MessageBoxButton.OK,

MessageBoxImage.Warning);

else

{

List<Warrior>[] donors = new List<Warrior>[4];

donors[0] = ListOfWarriorsOperations.GetDonorsForFirstBloodGroup(Warriors);

donors[1] = ListOfWarriorsOperations.GetDonorsForSecondBloodGroup(Warriors);

donors[2] = ListOfWarriorsOperations.GetDonorsForThirdBloodGroup(Warriors);

donors[3] = ListOfWarriorsOperations.GetDonorsForFourthBloodGroup(Warriors);

SecondWindow[] windows = new SecondWindow[4];

for (int i = 0; i < 4; i++)

{

if (donors[i].Count == 0)

{

MessageBox.Show($"На жаль, донорів для {i + 1} групи крові немає", "Увага!", MessageBoxButton.OKCancel,

MessageBoxImage.Warning);

}

else

{

windows[i] = new SecondWindow

{

labelTitle =

{

Content = $"Донори для {i + 1} групи крові"

},

warriorsTable =

{

ItemsSource = donors[i]

}

};

windows[i].Show();

}

}

}

}

//виклик вікна з загальними відомостями для програми

private void MiInstruction\_OnClick(object sender, RoutedEventArgs e)

{

MessageBox.Show("Звання в батальйоні:\nРекрут\nСолдат\nСержант\nЛейтенант\nКапітан\nМайор\nПідполковник\n" +

"Групи крові: I - O, II - A, III - B, IV - AB\nРезус фактор: + або -\n","Пояснення",

MessageBoxButton.OK, MessageBoxImage.Information);

}

//виклик вікна, яке містить інформацію про автора

private void MiAboutCreator\_OnClick(object sender, RoutedEventArgs e)

{

MessageBox.Show("Цю програму було створено студентом групи ПЗ-23 Борковичем Андрієм\nНУЛП\t2022 р.",

"Про розробника", MessageBoxButton.OK, MessageBoxImage.Information);

}

//завантаження номеру поточного рядка

private void Table\_OnLoadingRow(object sender, DataGridRowEventArgs e)

{

e.Row.Header = (e.Row.GetIndex() + 1).ToString();

}

}

}

**SecondWindow.xaml.cs**

using System;

using System.ComponentModel;

using System.IO;

using System.Linq;

using System.Windows;

using System.Windows.Controls;

using Microsoft.Win32;

namespace CourseWork

{

public partial class SecondWindow : Window

{

public SecondWindow()

{

InitializeComponent();

isListSaved = false;

}

private bool isListSaved;//змінна, яка визначає чи список було збережено у файл

//зберігаємо список у текстовий файл

private void WriteToFileButton\_OnClick(object sender, RoutedEventArgs e)

{

var warriors = warriorsTable.ItemsSource.Cast<Warrior>().ToList();//конвертуємо список з таблиці в змінну

SaveFileDialog saveFileDialog = new()

{

Filter = "Text files (\*.txt)|\*.txt|All files (\*.\*)|\*.\*"

};

try

{

if (saveFileDialog.ShowDialog() == true)

{

var filePath = saveFileDialog.FileName;

if (!filePath.Contains("txt"))

throw new Exception("Вибрано не текстовий файл!\nБудь ласка, виберіть файл з потрібним розширенням - txt");

bool result = false;

File.AppendAllText(filePath, $"{labelTitle.Content}\n");

string columnsTitles = "| Прізвище | Ім‘я |Вік|Група крові| Звання | Боєкомплект | \n";

File.AppendAllText(filePath, columnsTitles);

foreach (var warrior in warriors) result = warrior + filePath;

isListSaved = result;

MessageBox.Show("Список бійців було успішно записано до файлу!", "Вітання!", MessageBoxButton.OK,

MessageBoxImage.Information);

}

else

throw new Exception(

"Не вдалося створити файл для збереження списку!\nБудь ласка, спробуйте ще раз");

}

catch(Exception ex)

{

MessageBox.Show(this, ex.Message, "Помилка", MessageBoxButton.OKCancel, MessageBoxImage.Error);

}

}

//обробник події закриття вікна

private void SecondWindow\_OnClosing(object sender, CancelEventArgs e)

{

if (!isListSaved)

{

MessageBoxResult clickResult = MessageBox.Show("Список не записаний у файл! Дійсно бажаєте вийти?", "Увага!",

MessageBoxButton.YesNo, MessageBoxImage.Information);

if (clickResult == MessageBoxResult.No) e.Cancel = true;

}

}

//завантаження номеру поточного рядка

private void WarriorsTable\_OnLoadingRow(object sender, DataGridRowEventArgs e)

{

e.Row.Header = (e.Row.GetIndex() + 1).ToString();

}

}

}

**AddWarriorWindow.xaml.cs**

using System;

using System.ComponentModel;

using System.IO;

using System.Linq;

using System.Windows;

using System.Windows.Controls;

using Microsoft.Win32;

namespace CourseWork

{

public partial class SecondWindow : Window

{

public SecondWindow()

{

InitializeComponent();

isListSaved = false;

}

private bool isListSaved;//змінна, яка визначає чи список було збережено у файл

//зберігаємо список у текстовий файл

private void WriteToFileButton\_OnClick(object sender, RoutedEventArgs e)

{

var warriors = warriorsTable.ItemsSource.Cast<Warrior>().ToList();//конвертуємо список з таблиці в змінну

SaveFileDialog saveFileDialog = new()

{

Filter = "Text files (\*.txt)|\*.txt|All files (\*.\*)|\*.\*"

};

try

{

if (saveFileDialog.ShowDialog() == true)

{

var filePath = saveFileDialog.FileName;

if (!filePath.Contains("txt"))

throw new Exception("Вибрано не текстовий файл!\nБудь ласка, виберіть файл з потрібним розширенням - txt");

bool result = false;

File.AppendAllText(filePath, $"{labelTitle.Content}\n");

string columnsTitles = "| Прізвище | Ім‘я |Вік|Група крові| Звання | Боєкомплект | \n";

File.AppendAllText(filePath, columnsTitles);

foreach (var warrior in warriors) result = warrior + filePath;

isListSaved = result;

MessageBox.Show("Список бійців було успішно записано до файлу!", "Вітання!", MessageBoxButton.OK,

MessageBoxImage.Information);

}

else

throw new Exception(

"Не вдалося створити файл для збереження списку!\nБудь ласка, спробуйте ще раз");

}

catch(Exception ex)

{

MessageBox.Show(this, ex.Message, "Помилка", MessageBoxButton.OKCancel, MessageBoxImage.Error);

}

}

//обробник події закриття вікна

private void SecondWindow\_OnClosing(object sender, CancelEventArgs e)

{

if (!isListSaved)

{

MessageBoxResult clickResult = MessageBox.Show("Список не записаний у файл! Дійсно бажаєте вийти?", "Увага!",

MessageBoxButton.YesNo, MessageBoxImage.Information);

if (clickResult == MessageBoxResult.No) e.Cancel = true;

}

}

//завантаження номеру поточного рядка

private void WarriorsTable\_OnLoadingRow(object sender, DataGridRowEventArgs e)

{

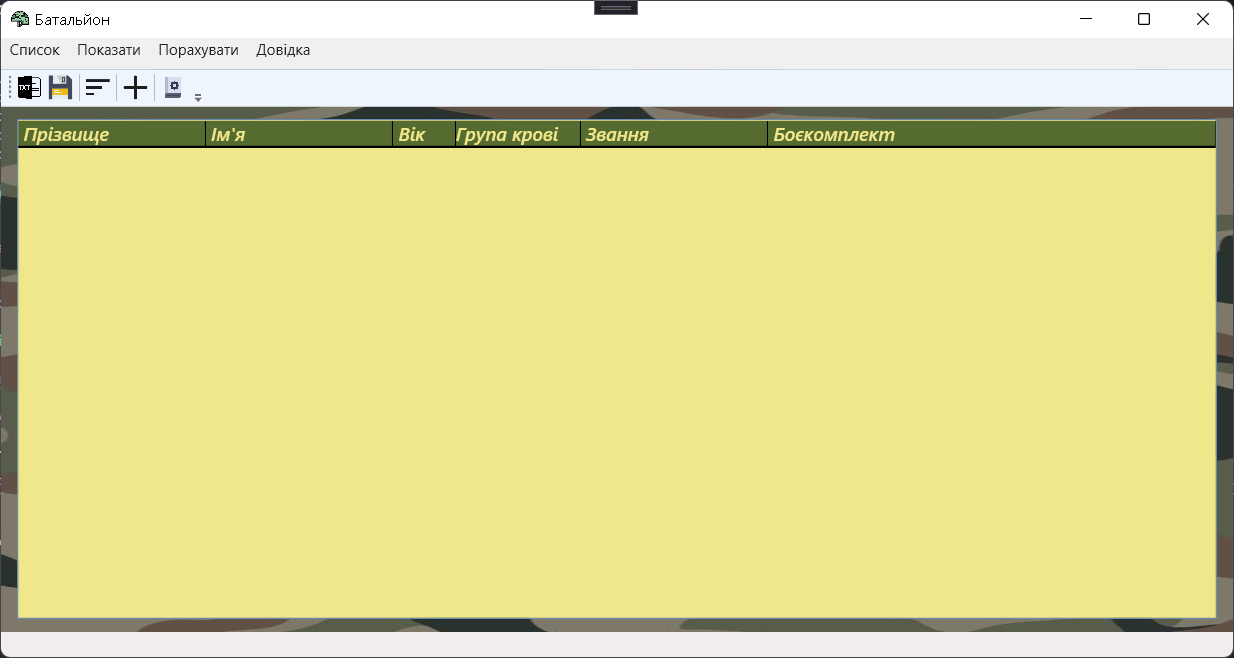
e.Row.Header = (e.Row.GetIndex() + 1).ToString();

}

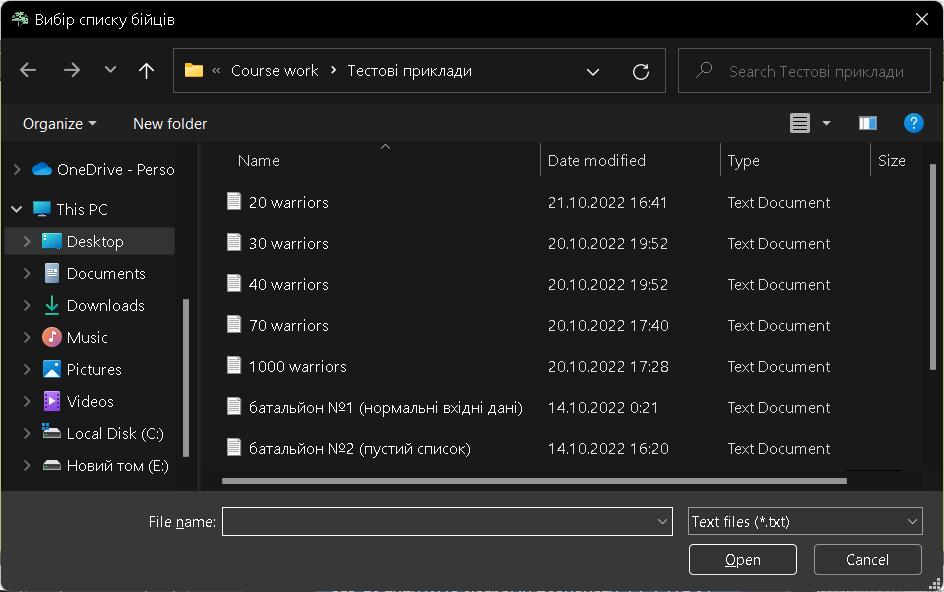
}

}

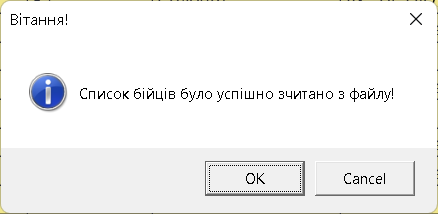
**Протокол роботи програми**



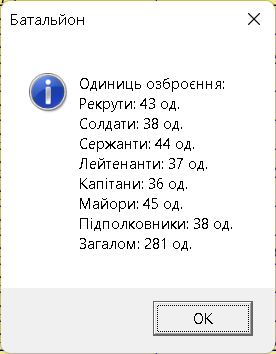
*Рис. 4.1. Головне вікно програми*



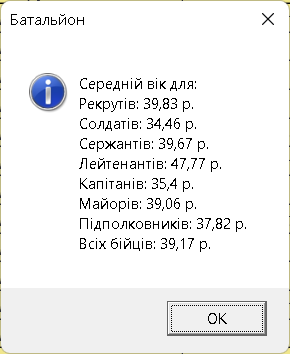
*Рис. 4.2. Вікно для вибору файлу для зчитування списку бійців*



*Рис. 4.3. Повідомлення про успішне зчитування списку*



*Рис. 4.4. Вікно з результатами обчислень кількості озброєння для кожного звання і загалом по батальйону*



*Рис. 4.5****.*** *Вікно з результатами обчислень середнього віку бійців для кожного звання і загалом по батальйону*



*Рис. 4.6. Вигляд програми після вибору файлу*



*Рис. 4.7. Таблиця з списком бійців, посортованих за званням*



*Рис. 4.8. Результат, який відображається у новому вікні, пошуку бійців, які мають більше однієї одиниці зброї*



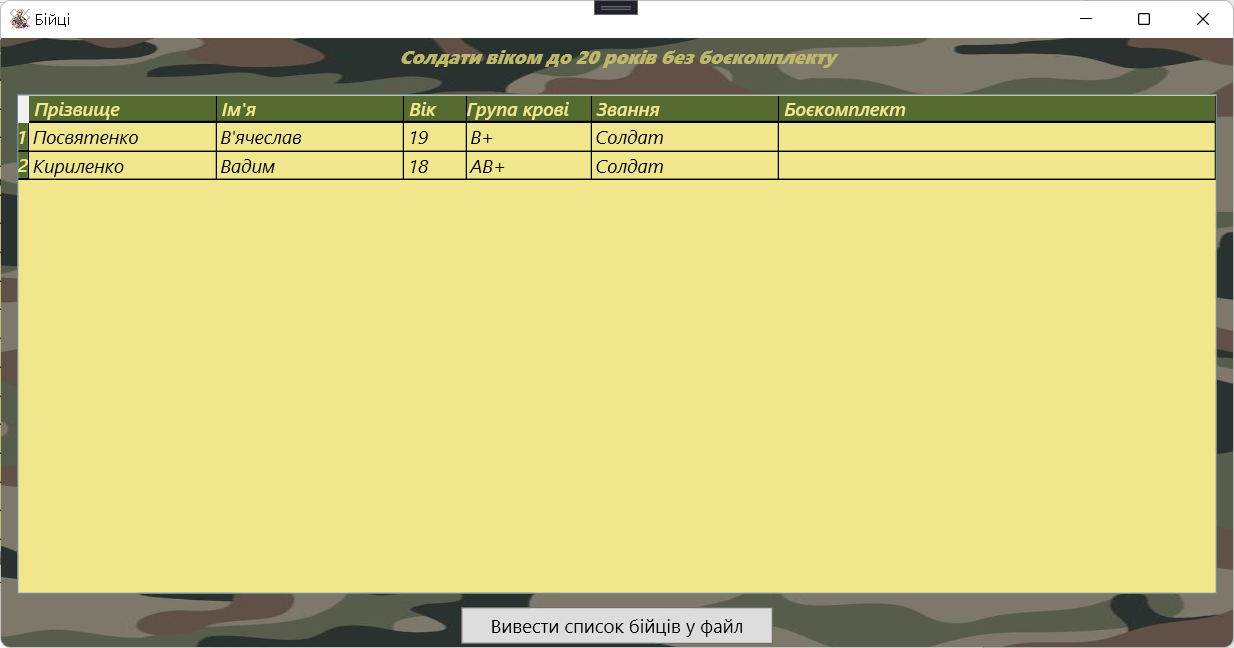
*Рис. 4.9. Результат пошуку бійців, які мають АК-47 і гранатомет, який відображається у новому вікні*



*Рис. 4.10. Список бійців, які мають пістолет і будь-який автомат*



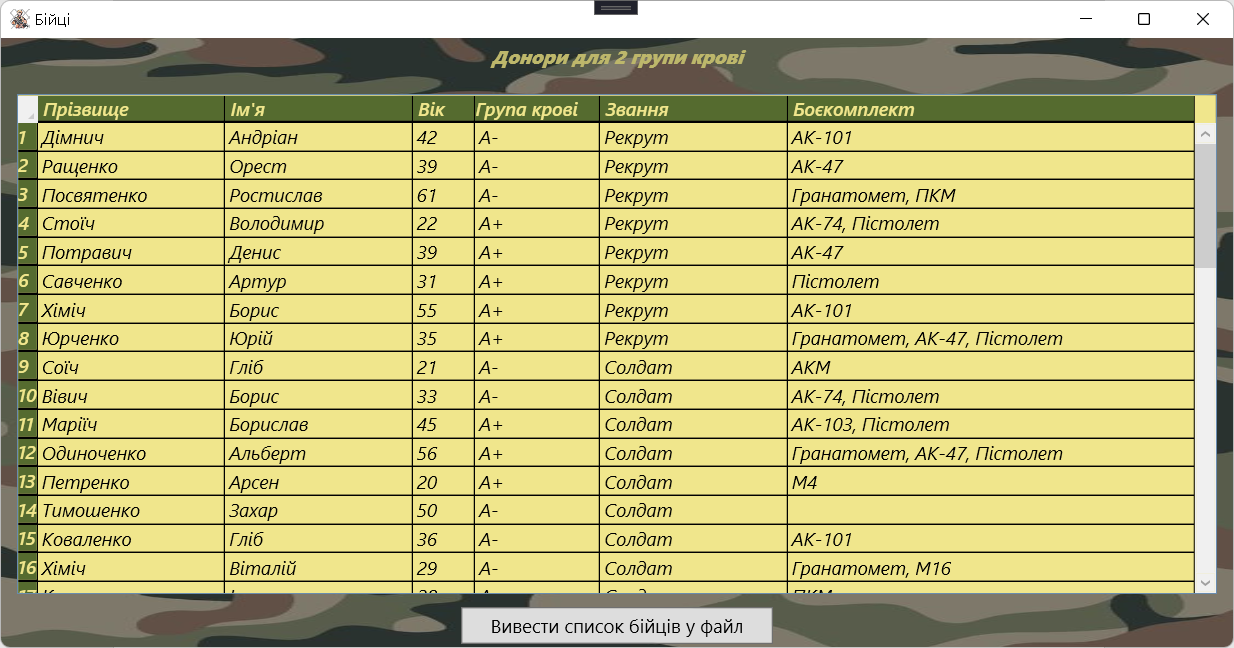
*Рис. 4.11. Список бійців, які мають першу групу резус мінус або четверту групу резус плюс віком до 25 років, поданий у таблиці*



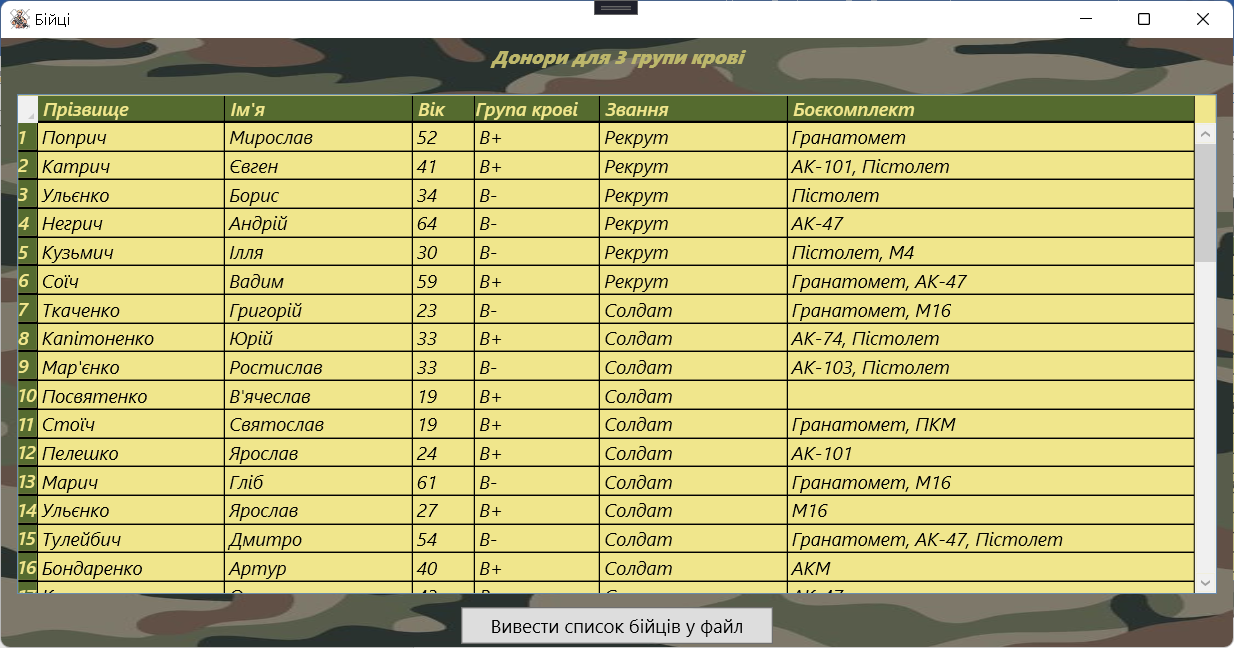
*Рис. 4.12. Список солдат, які не мають боєкомплекту, віком до 20 років*



*Рис. 4.13. Список бійців, які є донорами для 1 групи крові*



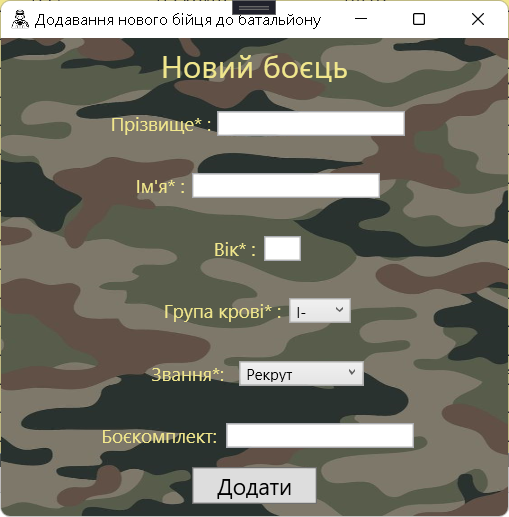
*Рис. 4.14. Список бійців, які є донорами для 2 групи крові*



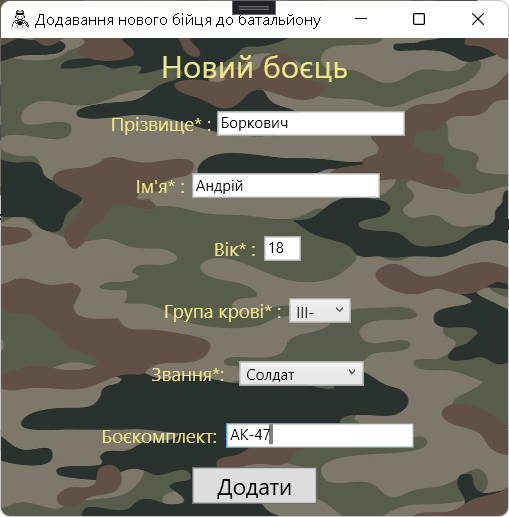
*Рис. 4.15.**Список бійців, які є донорами для 3 групи крові*



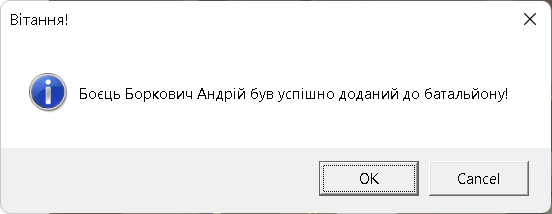
*Рис. 4.16. Список бійців, які є донорами для 4 групи крові*



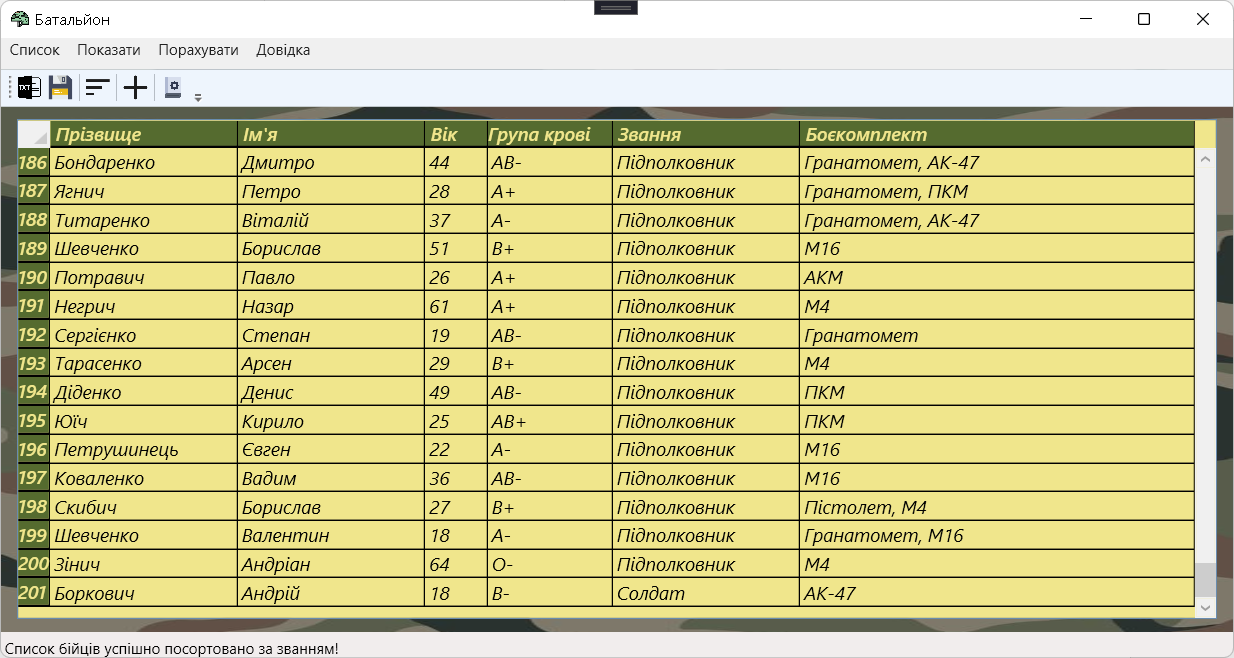
*Рис. 4.17. Форма для додавання нового бійця до батальйону*



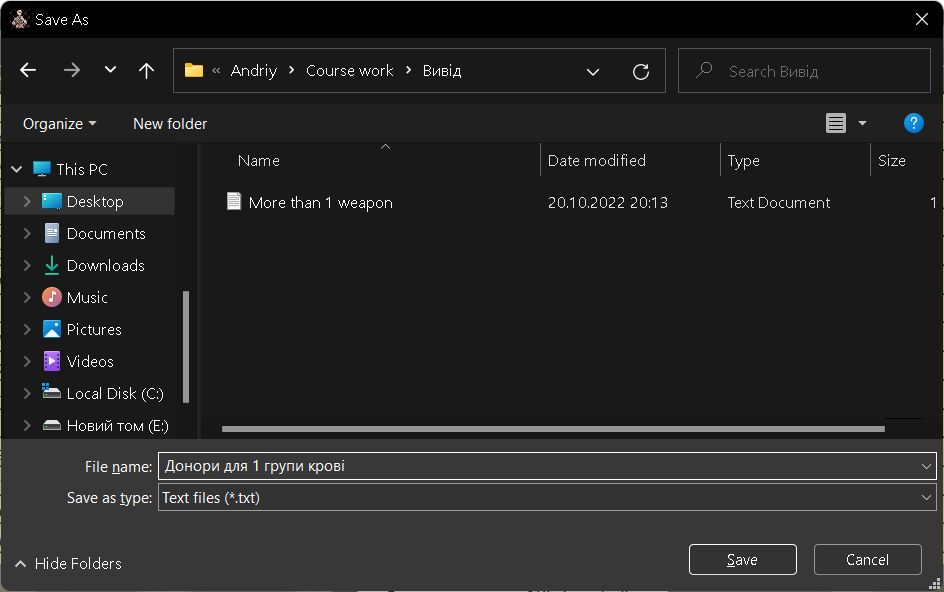
*Рис. 4.18. Заповнення форми даними про бійця*



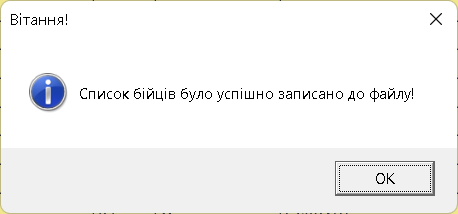
*Рис. 4.19. Повідомлення про успішне додавання бійця до батальйону*



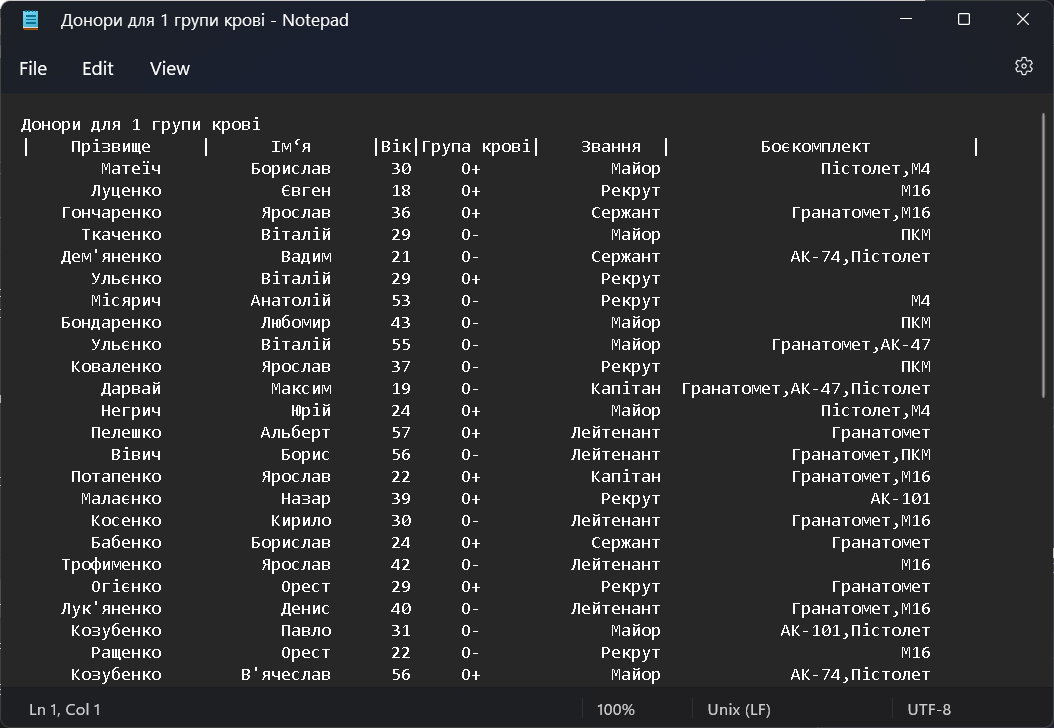
*Рис. 4.20. Результат додавання бійця до списку*



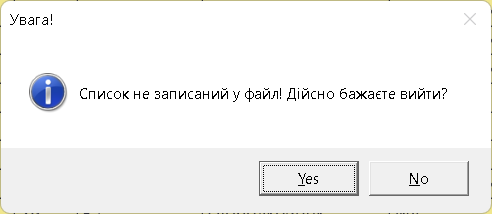
*Рис. 4.21. Вікно для запису списку бійців у текстовий файл*



*Рис. 4.22. Повідомлення про успішний запис до файлу*



*Рис. 4.23. Результат виводу у файл*



*Рис. 4.24. Попередження користувача, що список не збережено у файл, перед закриттям вікна*

**Інструкція користувача та системні вимоги**

1. **Компоненти ПЗ**

Пакет розроблено на мові програмування С# за стандартом С# 10.0 у середовищі розробки Microsoft Visual Studio 2022 з використанням графічної підсистеми WPF (Windows Presentation Foundation), яка в свою чергу є частиною .NET Framework 4.8, яке може експлуатуватися під управлінням сімейства операційних систем Windows. Під час програмування підсистем використовувався об’єктно орієнтований підхід. Всі класи документувались інформаційно і семантично.

Для коректної роботи пакету необхідна користувацька машина з процесором, тактова частота якого не менше 400 MHz, оперативною пам’яттю не менше 512 Mb.

**Набір файлів для коректної роботи програми «CourseWork»**

|  |  |  |  |
| --- | --- | --- | --- |
| № | Файл | Призначення | Належить проекту |
| 1 | warriors.txt | Текстовий файл, що містить дані про бійців, оформлений відповідно до ТЗ | Може бути розміщений у будь-якому місці пристрою |
| 2 | CourseWork.exe | Виконавчий файл | CourseWork |

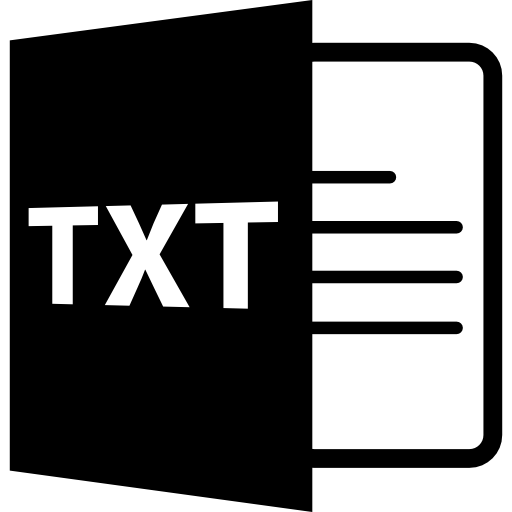
1. **Встановлення ПЗ**

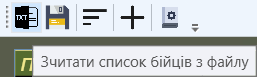
Для коректної роботи пакету потрібно підготувати текстовий файл відповідно до вимог та запустити виконавчий файл.

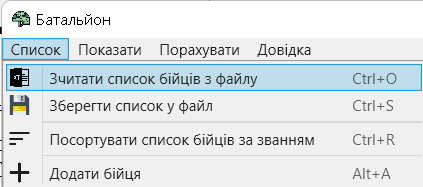
1. **Налаштування ПЗ**

Налаштовувати дане ПЗ немає потреби, воно відразу є готовим до взаємодії з користувачем.

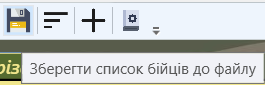
1. **Базові функції ПЗ**

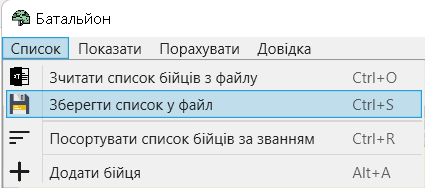
* Щоб зчитати список бійців з файлу, на панелі інструментів виберіть іконку  , або в меню **«Список»** вибрати пункт **«Зчитати список бійців з файлу»**. Швидким варіантом є комбінація гарячих клавіш *Ctrl+O*. Потім у відкритому вікні вибору (див. [рис. 4.2](#рисунок2)) оберіть файл, з якого хочете зчитати дані.



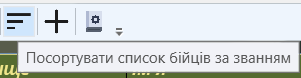
****

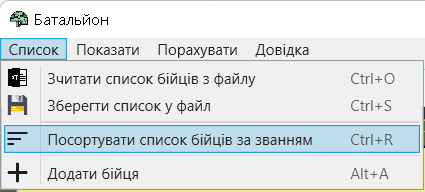
* Щоб записати список бійців у файл, потрібно на панелі інструментів вибрати іконку  , або в меню **«Список»** вибрати пункт **«Зберегти список у файл»**. Також можна скористатись комбінацією гарячих клавіш *Ctrl+S*. Після цього потрібно вибрати файл (або створити) (див. [рис. 4.21](#рисунок21)), в який буде записано список бійців.

****

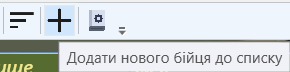
****

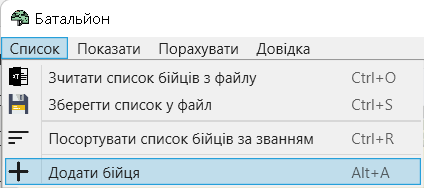
Для того, щоб посортувати бійців за їхніми званнями (від найнижчого звання до найвищого), потрібно на панелі інструментів вибрати іконку https://pics.freeicons.io/uploads/icons/png/3755052181551940363-512.png , або в меню **«Список»** вибрати пункт **«Посортувати список бійців за званням»**. Також можна скористатися комбінацією клавіш *Ctrl+R*.. Після цього на екрані з’явиться посортований список бійців. Також в програмі передбачене сортування за прізвищем або ім’ям, або віком. Для цього потрібно натиснути на заголовок відповідного стовпця (один клік – сортування за зростанням, два – за спаданням)

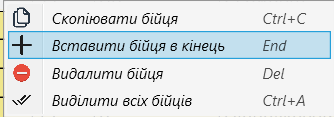
******

******

* Для того, щоб додати нового бійця до списку, потрібно на панелі інструментів вибрати іконку https://pics.freeicons.io/uploads/icons/png/4413922431556277115-512.png, або в меню «Список» вибрати пункт «Додати бійця». Також це можна зробити, натиснувши клавіші Alt+A (або End). Альтернативний варіант: правий клік мишки по таблиці, потому відкривається **контекстне меню** таблиці, на якому треба вибрати пункт **«Вставити бійця в кінець»**. Потім відкриється нове вікно з полями, які потрібно заповнити (див. [рис. 4.17](#рисунок17)). Після заповнення обов’язкових полів (позначені «\*»), потрібно натиснути кнопку «Додати». Якщо всі дані були введені правильно, то з’явиться повідомлення про успіх і боєць буде доданий в кінець списку.

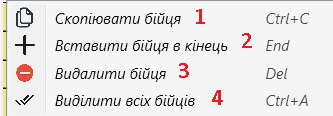




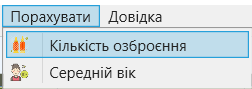




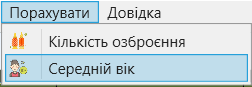
* Щоб скопіювати дані про окремого бійця, потрібно вибрати рядок в якому міститься інформація про нього і натиснути правою кнопкою по ньому. Після чого з’явиться **контекстне меню**, у якому потрібно вибрати пункт під номером **1**. За аналогію для додавання бійця в кінець списку вибираємо пункт **2**, видалення виділеного бійця з списку – цифра **3**, виділити всіх бійців у таблиці – цифра **4**.



* Щоб порахувати кількість озброєння, виданого кожному званню в батальйоні і загалом, потрібно в меню «Порахувати» вибрати пункт «Кількість озброєння». Після чого з’явиться повідомлення з результатами (див. [рис. 4.4](#рисунок4)).



* Щоб порахувати середній вік для кожного звання і загалом у батальйоні, треба в меню «Порахувати» вибрати пункт «Середній вік». Після чого з’явиться повідомлення з результатами (див. [рис. 4.5](#рисунок5)).

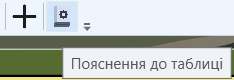


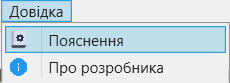
* Щоб побачити список бійців, які мають більше однієї одиниці зброї, мають АК-47 і гранатомет, мають пістолет і будь-який автомат потрібно в меню **«Показати»** вибрати пункти позначені цифрами **1, 2, 3** відповідно. Для визначення бійців, які мають першу групу крові резус мінус і четверту групу резус плюс віком до 25 років, потрібно вибрати пункт під цифрою **4**. Щоб показати солдат віком до 20 років без боєкомплекту, треба вибрати пункт під номером **5**. Для визначення донорів для кожної групи крові вибирається пункт під цифрою **6**. Незалежно від вибору користувача відкриється нова форма (форми), на якій (яких) буде таблиця з бійцями за даною ознакою (наприклад, як на [рис. 4.10](#рисунок10)). Користувачеві надається можливість вивести сформований список у файл. Для цього потрібно натиснути внизу кнопку **«Вивести список бійців у файл»**. Після цього потрібно вибрати файл (або створити) (див. [рис. 4.21](#рисунок21)), в який буде записано список бійців. Якщо користувач спробує закрити дане вікно без збереження, то з’явиться повідомлення (див. [рис. 4.24](#рисунок24)), яке перепитає чи справді користувач не бажає зберегти список до файлу.



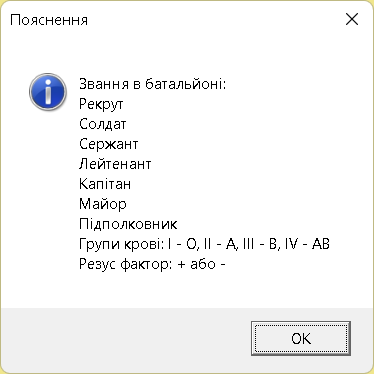


* Щоб побачити пояснення до груп крові та звань в даному батальйоні, потрібно в меню **«Довідка»** вибрати пункт **«Пояснення»**, або на панелі інструментів натиснути на іконку .

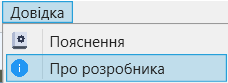




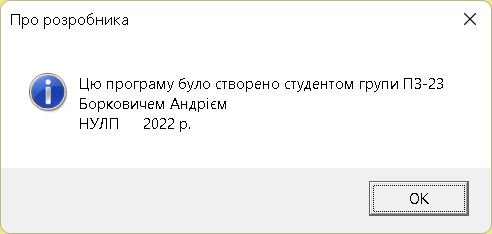
Після цього з’явиться наступне повідомлення:



* Щоб побачити інформацію про розробника, потрібно в меню «Довідка» вибрати пункт «Про розробника».



Після цього з’явиться наступне повідомлення:



1. **Аналіз можливих помилок**

У випадку помилки при зчитуванні даних потрібно перевірити правильність введених даних і їх відповідність вимогам. Лише після внесення поправок, можна зробити спробу зчитування.

У випадку помилки при зчитуванні пустого файлу, цей файл потрібно заповнити відповідно до вимог і повторити спробу, або вибрати інший, вже заповнений, файл.

У випадку помилки виконання будь-якої операції без зчитаних даних потрібно вибрати файл для зчитування або додати бійців з клавіатури.

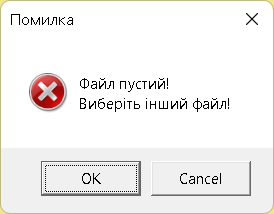
Детальний аналіз виняткових ситуацій наведено у розділі 6.

1. **Системні вимоги**

* Процесор: 32-розрядний або 64-розрядний процесор із тактовою частотою 400 МГц або швидший;
* ОЗП: 512 МБ ОЗП (в т.ч. 100 МБ вільних, решта - для підтримки операційної системи) або більше;
* Інше обладнання: під’єднана клавіатура, або інший метод стандартного вводу, наявність монітора або іншого методу стандартного виводу.
* Графічний пристрій із підтримкою DirectX 9 і драйвером WDDM 1.0 або новішим.
* Рекомендована ОС: Windows 7 і вище.
* Спеціальне програмне забезпечення: не вимагається.

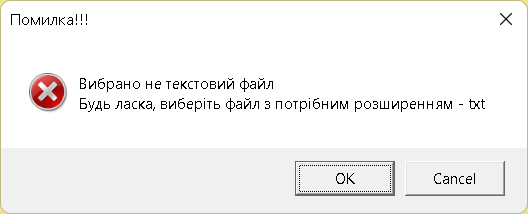
**Опис виняткових ситуацій**

Якщо користувач вибрав для зчитування пустий файл, то з’явиться відповідне повідомлення з помилкою, в якому користувача попросять вибрати заповнений файл:



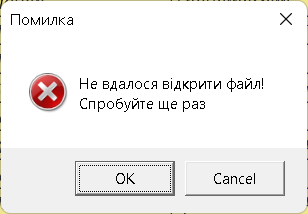
*Рис. 6.1. Спроба відкрити пустий файл*

Якщо користувач вибрав для зчитування не текстовий файл, а інший (випадково вибрав картинку), то з’явиться повідомлення з відповідною помилкою. На додачу, користувача попросять вибрати файл з правильним розширенням (.txt):



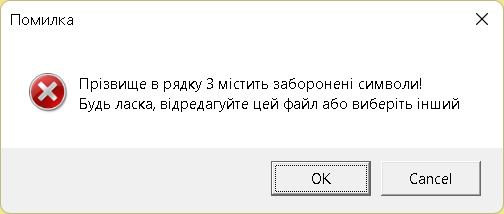
*Рис. 6.2. Спроба відкрити не текстовий файл*

Якщо виникла помилка при виборі файлу (його не було вибрано), то з’явиться відповідне повідомлення:

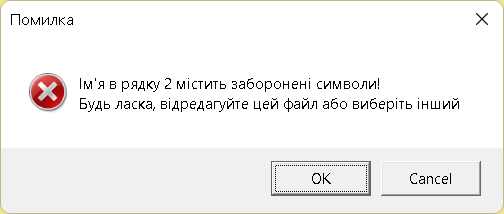


*Рис. 6.3. Відсутність вибору файлу*

Якщо у вибраному користувачем файлі для зчитування інформації у прізвищі або імені бійця міститься символ, який не є літерою, то з’явиться відповідне повідомлення, в якому ще буде вказано в якому саме рядку знаходиться помилка. Також користувача попросять виправити цю помилку або вибрати інший файл:

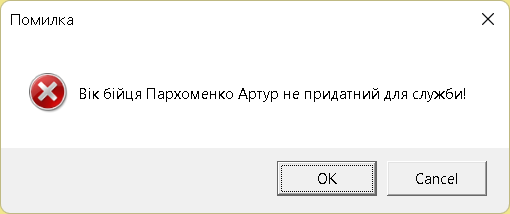


*Рис. 6.4. Спроба зчитування неправильно оформлених даних*



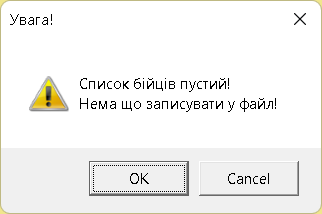
*Рис. 6.5. Спроба зчитування неправильно оформлених даних*

Якщо у вибраному користувачем файлі для зчитування інформації вік бійця є непридатним для військової служби (придатним вважається вік від 18 до 65 років), тоді на екрані з’явиться відповідне повідомлення, де буде вказано прізвище та ім’я такого бійця:



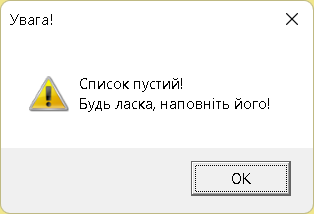
*Рис. 6.6. Спроба зчитати інформацію про непридатного бійця*

Якщо користувач спробує записати у файл пустий список бійців, то на екрані з’явиться відповідне повідомлення:



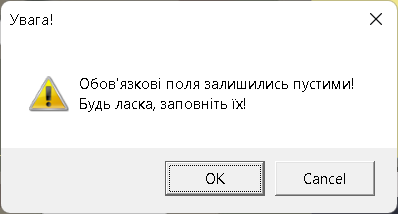
*Рис. 6.7. Спроба записати у файл пустий список*

Якщо користувач спробує виконати будь-яку функцію програми без попередньо зчитаних даних, то його попередять про це і попросять сформувати список бійців:



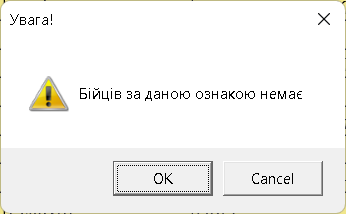
*Рис. 6.8. Спроба виконати операцію програми з пустим списком*

Якщо при додаванні бійця з клавіатури користувач не заповнив обов’язкові поля, то з'явиться попередження і його попросять заповнити ці поля:



*Рис. 6.9. Незаповнені поля у формі для додавання бійця*

Якщо в списку немає бійців за деякою з ознакою, то на екрані з’явиться відповідне повідомлення:



*Рис 6.10. Повідомлення про відсутність бійців за деякою ознакою*

**Структура файлу вхідних даних**

Дані у файлі з списком бійців повинні бути подані у такому порядку:

Прізвище, ім’я, вік, група крові, звання, боєкомплект.

Значення розділяються пробілом. Якщо в боєкомплекті є більше ніж 1 одиниця озброєння, то ті види зброї розділяються комою без пробілу. Допускається, що боєкомплект може бути пустим, тоді після звання ставиться пробіл і лише потім починається перехід на новий рядок.

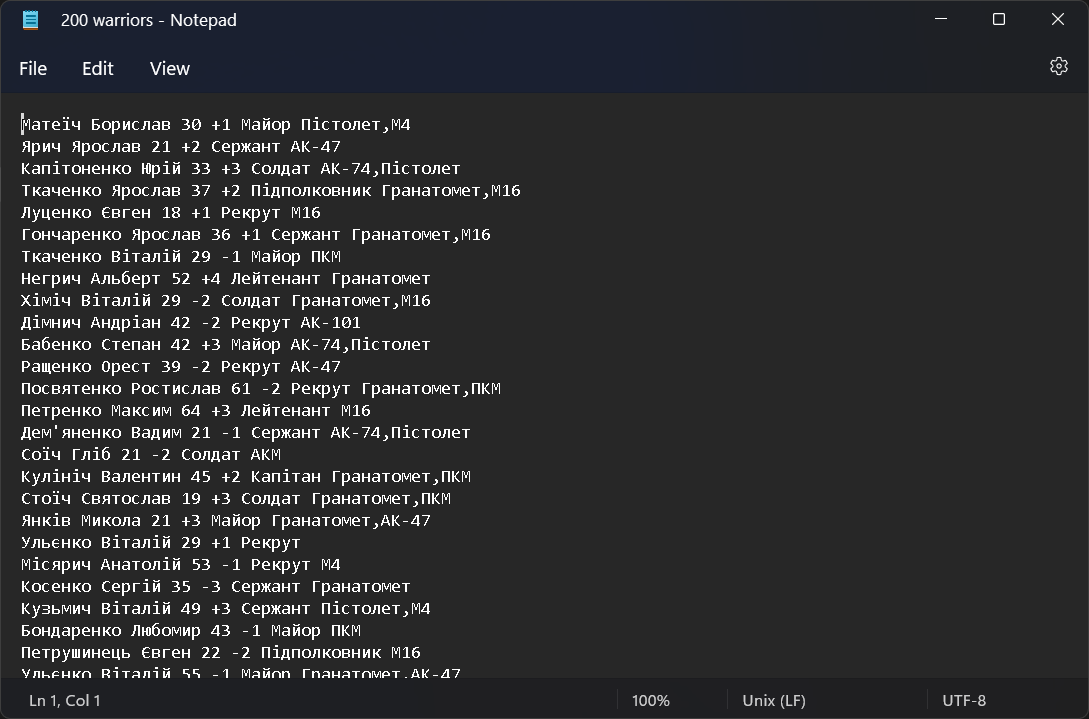
Прізвище та ім’я – лише кириличні літери, починаються з великої літери, можуть містити апостроф.

Вік – ціле додатне число (в межах від 18 до 65).

Група крові – подається у вигляді цілого числа в межах від 1 до 4. Містить спочатку знак, який означає резус фактор. Наприклад, -2 – ІІ (B) група крові резус мінус.

Звання – лише кириличні літери, починається з великої літери. Можливі значення (від найнижчого до найвищого): Рекрут, Солдат, Сержант, Лейтенант, Капітан, Майор, Підполковник.

Боєкомплект – лише кириличні літери, кожен вид зброї починається з великої літери. Можливі такі види зброї: пістолет, гранатомет, будь-який з автоматів. У загальному випадку, обмежень на види зброї немає.



*Рис. 7.1. Приклад вхідного файлу*

**Висновки**

У результаті виконання даної курсової роботи я закріпив практичні навички об’єктно орієнтованого програмування, реалізувавши мовою С# в середовищі програмування Visual Studio 2022 програмне забезпечення для роботи з списком бійців. Розроблена мною програма проводить різні маніпуляції зі списком, а саме: формування списку на основі заповненого текстового файлу або користувацького вводу з клавіатури, сортує список за званнями, виводить нові списки за різноманітними ознаками, обчислює статистику на основі даних про бійців. Під час написання курсової я використовував такі прийоми мови С# як авто-властивості, статичні методи та класи, перевантаження операторів, колекції тощо. Також я реалізував конструктори за замовчуванням, з параметрами та копій. Було реалізовано керування винятковими ситуаціями, для запобігання некоректної роботи програми, та роботу з файлами засобами мови С#. Також для спрощення деяких базових конструкцій для зручності мною було використано мову запитів до джерела даних LINQ. Задля забезпечення логіки та якості програми, я детально ознайомився з предметними областями «Групи крові», «Зброя» та «Військові звання».

**Список використаної літератури**

1. Левус Є., Мельник Н. Вступ до інженерії програмного забезпечення : навч. посіб. Львів : Видавництво Національного університету «Львівська політехніка», 2018. 246 с.
2. Об’єктно-орієнтоване програмування: методичні вказівки до виконання робіт студентів напряму 6.121 «Інженерія програмного забезпечення» /Укл. Коротєєва Т.О., Дяконюк Л.М. – Львів: Видавництво Національного університету «Львівська політехніка», 2020. – с.
3. Jeffrey Richter. CLR via C#. 4-те вид. Microsoft Press, 2012. 896 с.
4. C# documentation [Електронний ресурс]: [Веб-сайт]. – Режим доступу: URL : <https://docs.microsoft.com/en-us/dotnet/csharp/>
5. Andrew Troelsen, Phil Japikse. Pro C# 10 with .NET 6: Foundational Principles and Practices in Programming. 11-те вид. Apress, 2022. 1705 c.
6. Windows Presentation Platform documentation [Електронний ресурс] : [Веб-сайт]. – Режим доступу : URL :

<https://learn.microsoft.com/enus/dotnet/desktop/wpf/?view=netdesktop-6.0>