Міністерство освіти і науки України

Харківський національний університет радіоелектроніки

Кафедра Програмної інженерії

КУРСОВА РОБОТА

ПОЯСНЮВАЛЬНА ЗАПИСКА

з дисципліни “Об’єктно-орієнтоване програмування”

КРИЛАТІ ФРАЗИ

Керівник , асист. каф. ПІ Ляпота В.М.

Студент гр. ПІ-15-2 Байдак В.Є.

Комісія: проф. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Дудар З.В.

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

Харків 2016

**ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ РАДІОЕЛЕКТРОНІКИ**

Кафедра: ***Програмної інженерії***

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

Спеціальність: ***Програмна інженерія***

Курс 1 .Група *ПІ-15-2* . Семестр 2 .

***ЗАВДАННЯ***

***на курсовий проект студента***

***Байдака Вадима Євгеновича***

1. Тема проекту: Крилаті фрази.
2. Термін здачі студентом закінченого проекту: *“28” - травня - 2016 р*.
3. Вихідні дані до проекту: *специфікація програми, методичні вказівки до виконання курсової роботи*.
4. Зміст розрахунково-пояснювальної записки: *вступ, програмна специфікація, проектна специфікація,* інструкція користувача, висновки.
5. Перелік графічного матеріалу: екранні форми, *схема об’єктної моделі, діаграми класів*.

**КАЛЕНДАРНИЙ ПЛАН**

|  |  |  |
| --- | --- | --- |
| *№* | *Назва етапу* | *Термін виконання* |
| 1 | Видача теми, узгодження і затвердження теми | 01-03-2016 р. |
| 2 | Аналіз предметної області | 02-03-2016 – 12-03-2016 р. |
| 3 | Розробка постановки задачі | 12-03-2016 – 22-03-2016 р. |
| 4 | Розробка об’єктної моделі | 22-03-2016 – 12-04-2016 р. |
| 5 | Кодування програмної системи | 12-04-2016 – 12-05-2016 р. |
| 6 | Тестування і доопрацювання розробленої програмної системи. | 12-05-2016 – 27-05-2016 р. |
| 7 | Оформлення пояснювальної записки | 15-05-2016 – 27-05-2016 р. |
| 8 | Публічний захист проекту перед комісією | 28-05-2016 р. |

Студент \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Байдак В.Є.

Керівник \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Ляпота В.М.

« 1 » березня 2016р.

РЕФЕРАТ

Пояснювальна записка до курсової роботи: 26 с., 27 рис., 1 додаток,

5 джерел.

Метою роботи є розробка програмної системи під назвою “Крилаті фрази” на засадах об'єктно-орієнтованого програмування.

Методи розробки базуються на використанні середи розробки Microsoft Visual Studio 2015, Windows Forms, платформи .NET Framework 4.5.2, мови програмування C#.

В результаті отримана програма під назвою “Крилаті фрази”, яка дозволяє роботати з інформаційною базою даних, чию роль виконують dat-файли. Розроблена програма дає швидкий та зручний доступ до крилатих фраз, інформації про авторів та джерела, які існують в базі. Також є можливість додавання нових авторів, джерел, крилатих фраз, їх видалення або редагування, якщо існує така необхідність.

ПРОГРАМА, ОБ’ЕКТНО-ОРІЄНТОВАНЕ ПРОГРАМУВАННЯ, КЛАС, МОВА ПРОГРАМУВАННЯ C#, ПЛАТФОРМА .NET, КРИЛАТА ФРАЗА, АВТОР, ДЖЕРЕЛО, ДОВІДНИК.

ЗМІСТ

|  |  |
| --- | --- |
| Вступ ………………………………………………………………………. | 6 |
| 1 Специфікація програми …………………………………………............ | 8 |
| 1.1 Головне вікно, додавання автора, джерела, крилатої фрази, тем ....... | 8 |
| 1.2 Вікно авторів, джерел ………………………………………………… | 11 |
| 1.3 Робота з крилатими фразами (видалення та редагування)…………. | 13 |
| 2 Проектна специфікація …..……………………………………………... | 15 |
| 2.1 Об’єктна модель програми …..……………………………………….. | 15 |
| 2.2 Реалізація функцій програми ..……………………………………….. | 19 |
| 3 Інструкція користувача ..………………………………………………... | 23 |
| ВИСНОВКИ …..…………………………………………………………... | 25 |
| ПЕРЕЛІК ПОСИЛАНЬ …..……………………………………………….. | 26 |
| Додаток А Частинний код програми ...…………………………………... | 27 |

ВСТУП

Крилата фраза — короткий влучний оригінальний вислів, що зробився усталеним; яка-небудь узагальнена думка, висловлена стисло в дуже виразній, легкій для запам'ятовування формі, яка згодом неодноразово відтворюється іншими людьми. В афоризмі досягається найвища концентрація безпосереднього повідомлення і того контексту, в якому думка сприймається слухачами або читачем [1].

Щоб продемонструвати свою освіченість в компанії незнайомих людей або у колі друзів достатньо лише процитувати вдало підібрану крилату фразу. Цитування слів видатних особистостей завжди було модно, адже це говорить про ерудицію людини, яка відтворила цей вислів.

Крилаті фрази визначаються своєю простотою і доступністю. Їх легко запам’ятовувати завдяки їх лаконічності. Дуже важливо відтворити фразу дослівно, щоб вона не втратила свого сенсу. Цитувати афоризми може будь-яка людини: від слюсаря до лікаря.

Отже, крилаті фрази – це простий та доступний кожному засіб продемонструвати свою освіченість.

Метою даної курсової роботи є розробка програми-довідника, за допомогою якою можна отримувати доступ до крилатих фраз, додавати та редагувати вислови, інформацію про авторів і джерела цих висловів, здійснювати пошук за різними критеріями по наявній базі даних. Цю програму зможе використовувати будь-яка людина, яка хоче систематизувати набір своїх улюблених крилатих фраз і розширювати його. За основу розробки обраний об’єктно-орієнтований підхід.

Завдання виконання роботи:

1. вивчення предметної галузі з метою визначення головних принципів даної сфери;
2. проектування ієрархії класів, інтерфейсів, взаємодії компонентів на основі виділених принципів;
3. використання вбудованих елементів середовища розробки задля структуризації класів і оптимізації коду;
4. застосування принципа інкапсуляції до класів;
5. реалізація програмної взаємодії з базою даних.

Об’єктно-орієнтований підхід потребує повного розуміння основних концепцій, на яких він базується. В об’єктно-орієнтованому підході основними концепціями є об’єкти і класи [3].

На сьогоднішній день об’єктно-орієнтоване програмування є лідируючим в галузі прикладного програмування. Використання цього підходу надає розробнику всі необхідні інструменти для забезпечення функціональності і для супроводження проекту [2].

Розробка базується на використанні середовища розробки Microsoft Visual Studio 2015. В якості мови розробки була обрана мова С#.

1 СПЕЦИФІКАЦІЯ ПРОГРАМИ

* 1. Головне вікно, додавання автора, джерела, крилатої фрази, тем

Починаючи роботу з програмою, користувач зустрінеться з головним вікном (рис. 1.1). У нижній більшій частині панелі вікна користувач може побачити усі крилаті фрази, які знаходяться у базі даних. Також можна виконувати пошук по фразам, обираючи різні критерії (за автором, за джерелом, за темою, за текстом або за роком написання) з випадаючого списку. Пошук виконується автоматично по введенню користувачем даних в поле пошуку. Результати пошуку відображаються в тій же області, де буди усі крилаті фрази.

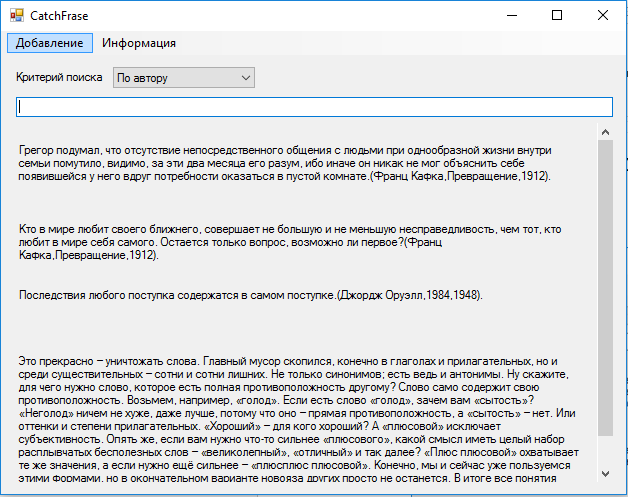
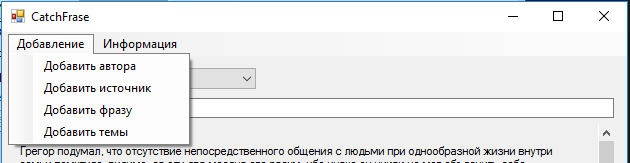


Рисунок 1.1 – Головне вікно програми

Користувач може додавати нових авторів, джерела, теми фраз та самі крилаті фрази. Для цього потрібно натиснути вкладку меню «Добавление» та обрати відповідний варіант (рис. 1.2).

Рисунок 1.2 – Вкладка «Добавление»



В залежності від вибору з’явиться одне з наступних вікон:

* «Добавить автора» (рис. 1.3);
* «Добавить источник» (рис. 1.4);
* «Добавить фразу» (рис. 1.5);
* «Добавление темы» (рис 1.6).

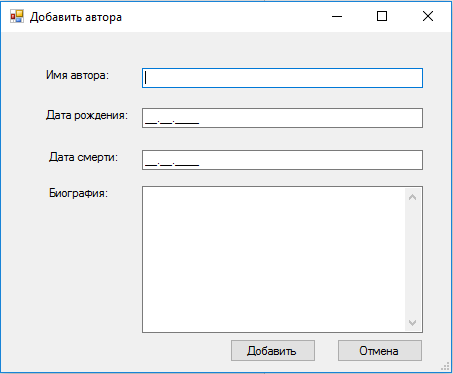


Рисунок 1.3 – Вікно «Добавить автора»

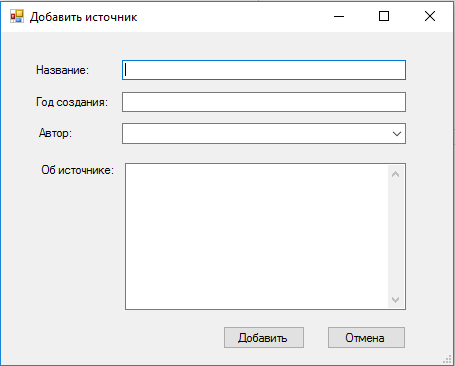


Рисунок 1.4 – Вікно «Добавить источник»

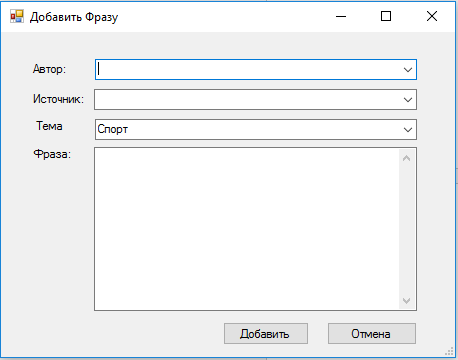


Рисунок 1.5 – Вікно «Добавить фразу»

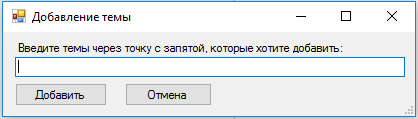


Рисунок 1.6 – Вікно «Добавление темы»

Після коректного заповнення усіх полів відповідної форми і натискання кнопки «Добавить», елемент/и буде додано до бази даних.

* 1. Вікно авторів, джерел

Користувач може переглядати інформацію про авторів або джерела наявних в базі даних. Для цього потрібно натиснути вкладку меню «Информация» та обрати відповідний варіант (рис. 1.7).

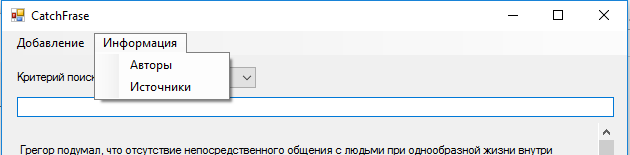


Рисунок 1.7 – Вкладка «Информация»

В залежності від вибору з’явиться одне з наступних вікон:

* «Авторы» (рис. 1.8);
* «Источники» (рис. 1.9).

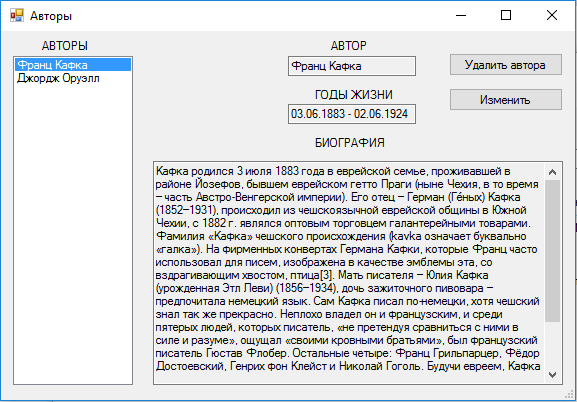


Рисунок 1.8 – Вікно «Авторы»

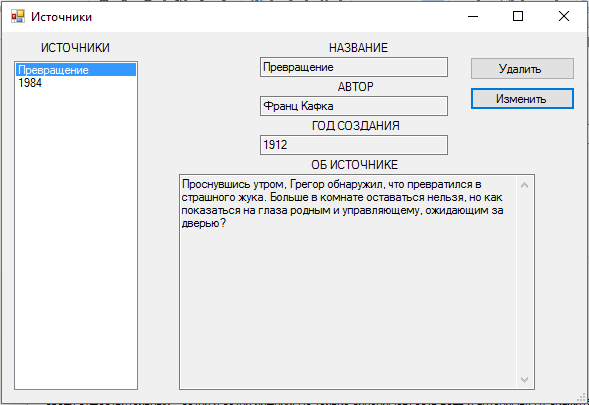


Рисунок 1.9 – Вікно «Источники»

У лівій частині панелі цих вікон є список, за допомогою якого користувач може обрати автора (джерело), інформацію про якого він хоче подивитись. Інформація відображується в правій більшій частині панелі. Також в цих вікнах користувач має можливість видалити обраного автора (джерело) натисканням кнопки «Удалить» або змінити інформацію про автора (джерело) натисканням кнопки «Изменить». Після натискання кнопки «Изменить» з’явиться вікно «Изменить информацию об авторе (источнике)» (рис. 1.10, рис. 1.11).

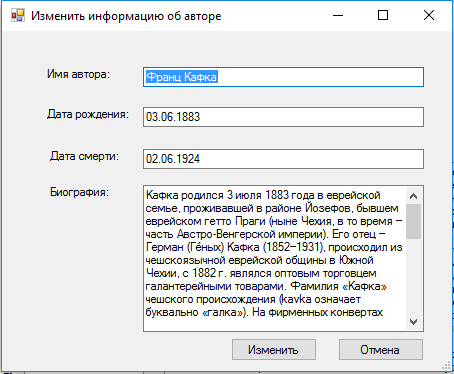


Рисунок 1.10 – Вікно «Изменить информацию об авторе»

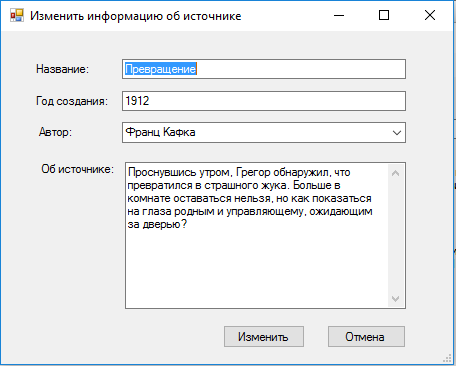


Рисунок 1.11 – Вікно «Изменить информацию об источнике»

Після коректного заповнення полів форми і натискання кнопки «Изменить» інформація про автора (джерело) буде змінена.

* 1. Робота з крилатими фразами (видалення та редагування)

Користувач має можливість видаляти та змінювати крилаті фрази. Для цього необхідно у головному вікні програми натиснути правою кнопкою миші на афоризм та вибрати з підменю, яке з’явилося (рис. 1.12), відповідний варіант.

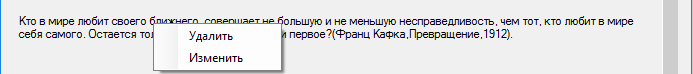


Рисунок 1.12 – Підменю крилатої фрази

При натисканні кнопки підменю «Удалить», афоризм буде видалено. При натисканні кнопки підменю «Изменить» з’явиться вікно «Изменить фразу» (рис. 1.13).

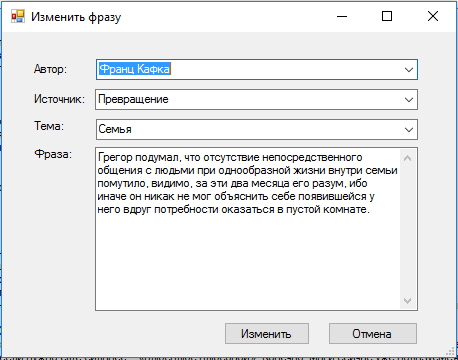


Рисунок 1.13 – Вікно «Изменить фразу»

Після коректного заповнення полей форми якого і натискання кнопки «Изменить», афоризм буде змінено.

1. ПРОЕКТНА СПЕЦИФІКАЦІЯ
   1. Об’єктна модель програми

Основним конструктивним елементом в об’єктно орієнтованих мовах і відповідно у реалізованій програмі є модуль, що представляє собою логічно пов'язану сукупність класів і об'єктів, а не підпрограм, як в більш ранніх мовах. Інакше кажучи, "якщо процедури і функції - це дієслова, а дані - це іменники, то в основі процедурно-орієнтованих програм лежать дієслова, а в основі об'єктно-орієнтованих - іменники" [4].

Графічний інтерфейс розробленої програми містить набір форм додавання і редагування даних. Відповідно в проекті реалізовані методи , які зчитують данні, заповнюють ними колекції для подальшої роботи, записують данні. Методами зчитування і запису даних є десеріалізація і серіалізація.

В об’єктній моделі програми (рис. 2.1) реалізовані наступні класи:

* Author (рис. 2.2) – базовий клас, який реалізує автора крилатої фрази. Має такі основні властивості як: Name (ім’я автора), Born (дата народження автора), Die (дата смерті автора), Bio (біографія автора), Id (айді автора у списку авторів). Також клас має конструктор;
* Source (рис. 2.3) – базовий клас, який реалізує джерело крилатої фрази. Має такі основні властивості як: Name (назва джерела), Author (автор джерела), Year (рік написання джерела), Info (інформація про джерело), Id (айді джерела у списку джерел). Також клас має конструктор;
* Phrase (рис 2.4) – базовий клас, який реалізує саму крилату фразу. Має такі основні властивості як: Author (автор крилатої фрази), Source (джерело крилатої фрази), Theme (тема крилатої фрази), Content (зміст крилатої фрази). Також клас має конструктор. В класі перекритий метод ToString();
* AList (рис 2.5) – клас, який реалізує колекцію авторів і призначен для роботи з інформацією про авторів. Колекція складається з об’єктів класу Author. Клас має конструктор, індексатор, властивість Length (довжина колекції авторів). Також клас має методи серіалізації, десеріалізації, додавання автора в колекцію, видалення автора із колекції по номеру, метод перевіряючий на наявність автора в колекції, метод перевірки дати на коректність та метод, який прибирає зайві пропуски зі строки;
* SList (рис 2.6) – клас, який реалізує колекцію джерел і призначен для роботи з інформацією про джерела. Колекція складається з об’єктів класу Source. Клас має конструктор, індексатор, властивість Length (довжина колекції джерел). Також клас має методи серіалізації, десеріалізації, додавання джерела в колекцію, видалення джерела із колекції, видалення джерела із колекції по номеру, метод, який прибирає зайві пропуски зі строки та метод який повертає колекцію джерел автора;
* PList (рис 2.7) – клас, який реалізує колекцію крилатих фраз і призначен для роботи з інформацією про крилаті фрази. Колекція складається з об’єктів класу Phrase. Клас має конструктор, індексатор, властивість Length (довжина колекції джерел). Також клас має методи серіалізації, десеріалізації, додавання фрази в колекцію, видалення фрази із колекції, видалення фрази із колекції по номеру, метод, який прибирає зайві пропуски зі строки, перекритий метод ToString(), метод, який повертає строкову колекцію фраз та метод пошуку;
* TList (рис 2.8) – клас, який реалізує колекцію тем і призначен для роботи з темами. Клас має конструктор, індексатор, властивість Length (довжина колекції тем). Також клас має методи серіалізації, десеріалізації, додавання теми в колекцію, метод, який прибирає зайві пропуски зі строки та метод, перевіряючий на наявність теми в колекції.

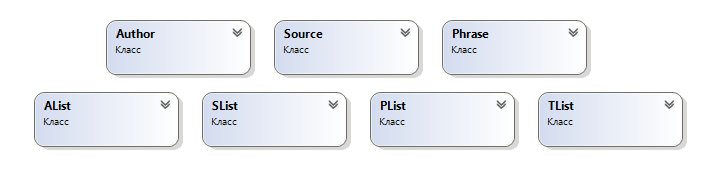


Рисунок 2.1 – Об’єктна модель класів

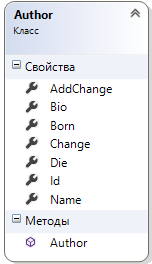


Рисунок 2.2 – Клас Author

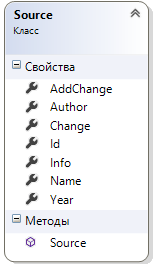


Рисунок 2.3 – Клас Source

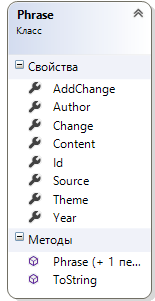


Рисунок 2.4 – Клас Phrase

|  |  |
| --- | --- |
| Рисунок 2.5 – Клас AList | Рисунок 2.6 – Клас SList |

|  |  |
| --- | --- |
| Рисунок 2.7 – Клас PList | Рисунок 2.8 – Клас TList |

* 1. Реалізація функцій програми

У цьому розділі будуть розглянуть осноіні функції реалізовані в програмі. Повний набір функцій можна знайти у коді програми (додаток А).

Одними з найважливіших методів програми є методи зчитування та запису даних. В програмі вони реалізовані за допомогою серіалізації да десеріалізації об’єктів. Оскільки ці методи наявні в декількох класах і відрізняються лише файлами запису/зчитування, розглянемо їх на прикладі класу AList (рис. 2.9, рис. 2.10):

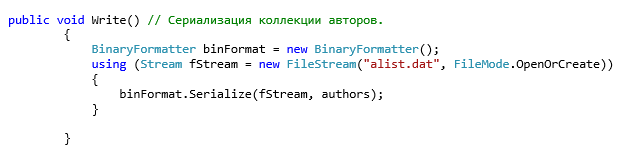


Рисунок 2.9 – Серіалізація

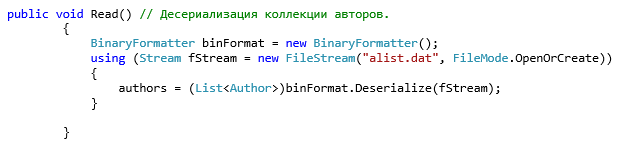


Рисунок 2.10 – Десеріалізація

Також дуже важливим методом для повноцінного функціонування програми є метод пошуку крилатої фрази (рис 2.11). Пошук може здійснюватися в залежності від введеного аргумента за декількома критеріями: за автором, за джерелом, за темою, за текстом, за роком написання.

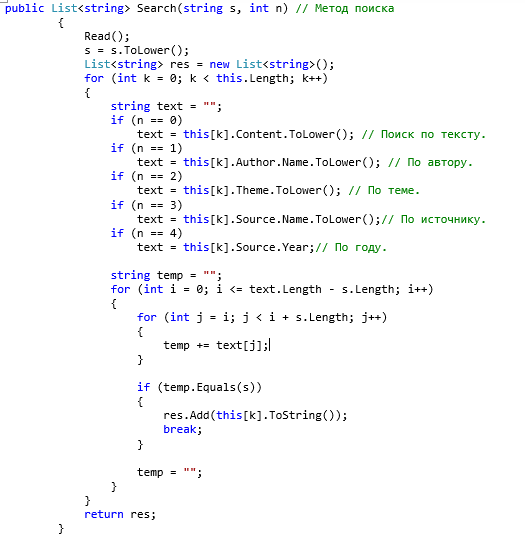
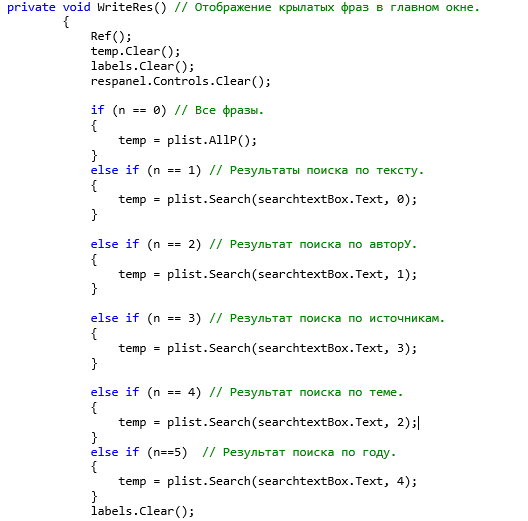


Рисунок 2.11 – Метод пошуку крилатої фрази

Ще один не менш важливий метод – це метод відображення відповідних крилатих фраз в головному вікні програми (рис. 2.12). При зміні інформації про джерело, автора, рік написання фрази або при зміні її змісту , фраза змінюється у головному вікні програми. Крім того при пошуку у вікні відображаються лише ті фрази, які відповідають критеріям пошуку введеним користувачем.



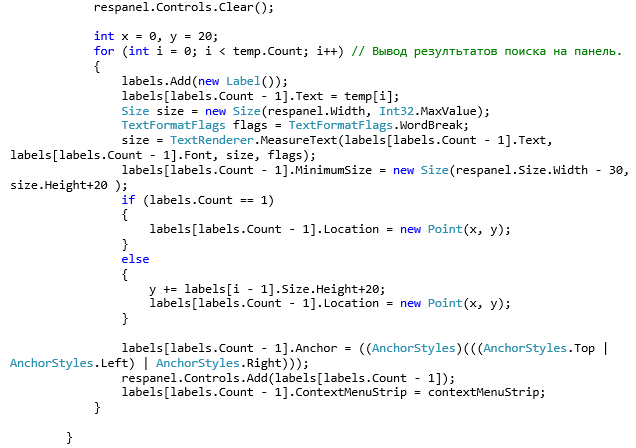


Рисунок 2.12 – Метод відображення крилатих фраз

В загальному випадку об’єкти додаються до колекції методом Add.Так як цей метод однаковий для всіх об’єктів, розглянемо його на прикладі Author (рис. 2.13), а видаляються методом RemoveAt, який видаляє об’єкт за номером елемента (рис. 2.14).

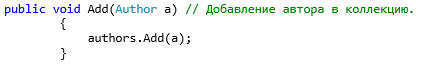


Рисунок 2.13 – Метод додавання Add

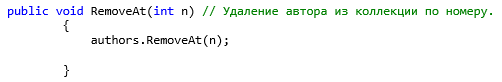


Рисунок 2.14 – Метод видалення RemoveAt

Ці методи реалізовані у відповідних класах (AList, SList, Plist, TList).

1. ІНСТРУКЦІЯ КОРИСТУВАЧА

Для пошуку крилатих фраз в головному вікні програми необхідно обрати критерій пошуку з випадаючого списку та почати вводити символи у поле пошуку. Результати пошуку будуть відображатися динамічно.

Для додавання автора потрібно натиснути вкладку меню «Добавление», вибрати «Добавить автора». В вікні додавання автора, яке з’явилося необхідно коректно заповнити всі поля форми:

* дата народження і дата смерті повинні бути введені коректно;
* дата народження не може бути більшою за дату смерті;
* неможливо дати вже існуючого в базі даних автора;
* усі поля обов’язкові до заповнення.

Для додавання джерела потрібно натиснути вкладку меню «Добавление», вибрати «Добавить источник». В вікні додавання джерела, яке з’явилося необхідно коректно заповнити всі поля форми:

* дата створення не може бути більшою за дату смерті автора і меншою ніж дата народження автора;
* автора джерела необхідно вибрати з випадаючого списку вже занесених в базу даних авторів;
* усі поля обов’язкові до заповнення.

Для додавання теми потрібно натиснути вкладку меню «Добавление», вибрати «Добавить темы». В вікні додавання тем, яке з’явилося необхідно заповнити поле, вводячи теми, які потрібно додати, через крапку з комою.

Для додавання крилатої фрази потрібно натиснути вкладку меню «Добавление», вибрати «Добавить фразу». В вікні додавання фрази, яке з’явилося необхідно коректно заповнити всі поля форми:

* автора фрази необхідно вибрати з випадаючого списку вже занесених в базу даних авторів;
* джерело фрази необхідно вибрати з випадаючого списку вже занесених в базу даних джерел, які відповідають обраному автору;
* тему фрази необхідно вибрати з випадаючого списку вже занесених в базу даних тем;
* усі поля обов’язкові до заповнення.

Щоб подивитися інформацію про існуючих в базі даних авторів, необхідно натиснути вкладку меню «Інформація» та обрати «Авторы». У вікні, яке відкрилося, для перегляду інформації про конкретного автора необхідно обрати цього автора у списку зліва. Також у користувача є можливість видалити обраного автора натисканням кнопки «Удалить». Видалення можливе тільки якщо за автором не закріплене жодне джерело та жодна крилата фраза. Крім того користувач може змінити інформацію про обраного автора. Для цього необхідно натиснути кнопку «Изменить». Вікно зміни інформації про автора відповідає тим же вимогам що і вікно додавання.

Щоб подивитися інформацію про існуючі в базі даних джерела, необхідно натиснути вкладку меню «Інформація» та обрати «Источники». У вікні, яке відкрилося, для перегляду інформації про конкретне джерело необхідно обрати це джерело у списку зліва. Також у користувача є можливість видалити обране джерело натисканням кнопки «Удалить». Видалення можливе тільки якщо за джерелом не закріплена жодна крилата фраза. Крім того користувач може змінити інформацію про обране джерело. Для цього необхідно натиснути кнопку «Изменить». Вікно зміни інформації про джерело відповідає тим же вимогам що і вікно додавання.

Крилаті фрази розміщенні під полем пошуку у головному вікні програми. При натисканні правою кнопкою миші на відповідній крилатій фразі, з’являється підменю, за допомогою якого можна видалити крилату фразу або змінити інформацію про неї. Вікно зміни інформації про крилату фразу відповідає тим же вимогам що і вікно додавання.

ВИСНОВКИ

В результаті курсової роботи була отримана готова програма – програма-довідник «Крилаті фрази». За допомогою цієї програми можна зручно шукати крилаті фрази та додавати нові.

Інтерфейс програми простий, зрозумілий та зручний в використанні. Цією програмою без зусиль зможе користуватися навіть недосвідчений користувач.

Ця програма створена для тих, хто хоче систематизувати набір своїх улюблених крилатих фраз або дізнатися нові. За допомого цієї програми можна легко заносити та систематизувати нові крилаті фрази з прив’язкою до авторів та джерел та шукати ці фрази за різноманітними критеріями.

В курсовій роботі реалізований об’єктно-орієнтований підхід в створенні програмного забезпечення інформаційного характеру, який дає можливість на перших етапах розробки визначити необхідний набір функцій майбутньої програми, її склад і структуру баз даних необхідну для реалізації визначених функцій. Все це захищає від майбутньої необхідності в виправленні вже створених компонентів програми.

Шляхи вдосконалення та розвитку програми: використання більш розширеної бази даних; можливість додавання мультимедійного матеріалу; більш просунутий пошук для знаходження необхідних користувачу крилатих фраз.

ПЕРЕЛІК ПОСИЛАНЬ

1. Афоризм. Матеріал з Вікіпедії — вільної енциклопедії. [Електронний ресурс] – Режим доступу: www/URL: https://uk.wikipedia.org/wiki/Афоризм
2. Бондарев В. М. «Объектно-ориентированное программирование на С#» [Текст]: учеб. Пособ / В. М. Бондарев. – Х.: Смит, 2013. – 224 с.;
3. Троелсен Эндрю Язык программирования C# (Си Шарп) и платформа .NET 4.2 – 3-е изд. ./ Эндрю Троелсен. – М. : «Вильямс», 2011. – 1168 c.
4. Шилдт Г. C# 4.0.: Полное руководство [Текст]: пер. с англ./ Г. Шилдт – М.: Вильямс, 2011 – 1056 с
5. C Sharp. Матеріал з Вікіпедії — вільної енциклопедії. [Електронний ресурс] – Режим доступу: http://uk.wikipedia.org/wiki/C\_Sharp – Назва з титул. екрану.

Додаток А

Частинний код програми

Клас Author

[Serializable]

class Author

{

public string Name { get; set; }

public string Born { get; set; }

public string Die { get; set; }

public string Bio { get; set; }

public double Id { get; set; }

//public bool ShowA { get; set; }

public bool Change { get; set; }

public bool AddChange { get; set; }

public Author(string n, string b, string d, string bio,double i)

{

Name = n;

Born = b;

Die = d;

Bio = bio;

Id = i;

}

}

Клас Source

[Serializable]

class Source

{

public string Name { get; set; }

public Author Author { get; set; }

public string Year { get; set; }

public string Info { get; set; }

public double Id { get; set; }

public bool Change { get; set; }

public bool AddChange { get; set; }

public Source(string n, Author a, string y,string i,double id)

{

Name = n;

Author = a;

Year = y;

Info = i;

Id = id;

}

}

Клас Phrase

[Serializable]

class Phrase

{

public string Content { get; set; }

public string Year { get; set; }

public Author Author { get; set; }

public Source Source { get; set; }

public string Theme { get; set; }

public double Id { get; set; }

public bool Change { get; set; }

public bool AddChange { get; set; }

public Phrase()

{

Content = "";

Year = "";

Author = null;

Source = null;

Theme = "";

Id = -1;

}

public Phrase(string c, string y, Author a, Source s,string t,double i)

{

Content = c;

Year = y;

Author = a;

Source = s;

Theme = t;

Id = i;

}

public override string ToString() // Переопределение метода ToString().

{

return String.Format("{0}({1},{2},{3}).", Content, Author.Name, Source.Name, Year);

}

Клас AList

[Serializable]

class AList

{

List<Author> authors = new List<Author>();

public List<Author> Authors

{

get { return authors; }

set { authors = value; }

}

public Author this[int i] // Индексатор для коллекции авторов.

{

get { return authors[i]; }

set { authors[i] = value; }

}

public int Length // Свойство - длина коллекции авторов.

{

get { return authors.Count; }

}

public void Write() // Сериализация коллекции авторов.

{

BinaryFormatter binFormat = new BinaryFormatter();

using (Stream fStream = new FileStream("alist.dat", FileMode.OpenOrCreate))

{

binFormat.Serialize(fStream, authors);

}

}

public void Read() // Десериализация коллекции авторов.

{

BinaryFormatter binFormat = new BinaryFormatter();

using (Stream fStream = new FileStream("alist.dat", FileMode.OpenOrCreate))

{

authors = (List<Author>)binFormat.Deserialize(fStream);

}

}

public void Add(Author a) // Добавление автора в коллекцию.

{

authors.Add(a);

}

public void RemoveAt(int n) // Удаление автора из коллекции по номеру.

{

authors.RemoveAt(n);

}

public bool Exist(Author a) // Метод, проверяющий наличие автора в базе.

{

foreach (Author x in authors)

{

if (x.Name == a.Name && x.Born == a.Born && x.Die == a.Die && a.Bio == x.Bio)

return true;

}

return false;

}

public bool checkDate(string s) // Прверка даты на корректность.

{

string[] s1 = s.Split('.');

if (s1[0] == "" && s1[1] == "" && s1[2] == "")

return false;

int[] temp = { Convert.ToInt32(s1[0]), Convert.ToInt32(s1[1]), Convert.ToInt32(s1[2]) };

if (temp[1] <= 0 || temp[1] > 12)

return false;

else if (temp[1] == 1 || temp[1] == 3 || temp[1] == 5 || temp[1] == 7 || temp[1] == 8 || temp[1] == 10 || temp[1] == 12)

{

if (temp[0] <= 0 || temp[0] > 31)

return false;

}

else if (temp[1] == 4 || temp[1] == 6 || temp[1] == 9 || temp[1] == 11)

{

if (temp[0] <= 0 || temp[0] > 30)

return false;

}

else if (temp[1] == 2)

{

if (temp[0] <= 0 || temp[0] > 29)

return false;

}

if (temp[2] > 2016)

return false;

return true;

}

public string RemoveSpaces(string s) // Убирает лишние пробелы из строки.

{

string[] s1 = s.Split(' ');

string r = "";

for (int i = 0; i < s1.Length; i++)

{

if (s1[i] != "")

{

r += s1[i];

if (i != s1.Length - 1)

{

r += " ";

}

}

}

return r;

}

}

Клас AList

[Serializable]

class PList

{

List<Phrase> phrases = new List<Phrase>();

public List<Phrase> Phrases

{

get { return phrases; }

set { phrases = value; }

}

public Phrase this[int i] // Индексатор для коллекции фраз.

{

get { return phrases[i]; }

set { phrases[i] = value; }

}

public int Length // Свойство - длина коллекции фраз.

{

get { return phrases.Count; }

}

public void Write() // Сериализация коллекции фраз.

{

BinaryFormatter binFormat = new BinaryFormatter();

try

{

using (Stream fStream = new FileStream("plist.dat", FileMode.OpenOrCreate))

{

binFormat.Serialize(fStream, phrases);

}

}

catch { }

}

public void Read() // Десериализация коллекции фраз.

{

BinaryFormatter binFormat = new BinaryFormatter();

try

{

using (Stream fStream = new FileStream("plist.dat", FileMode.Open))

{

phrases = (List<Phrase>)binFormat.Deserialize(fStream);

}

}

catch { }

}

public void Add(Phrase q) // Добавление фразы в коллекцию.

{

phrases.Add(q);

}

public void Remove(Phrase q) // Удаление фразы из коллекции.

{

phrases.Remove(q);

}

public void RemoveAt(int i) // Удаление фразы из коллекции по номеру.

{

phrases.RemoveAt(i);

}

public override string ToString() // Переопределение метода ToString().

{

Read();

string r = "";

foreach (Phrase q in phrases)

{

r += q.ToString();

}

return r;

}

public string RemoveSpaces(string s) // Убирает лишние пробелы.

{

string[] s1 = s.Split(' ');

string r = "";

for (int i = 0; i < s1.Length; i++)

{

if (s1[i] != "")

{

r += s1[i];

if (i != s1.Length - 1)

{

r += " ";

}

}

}

return r;

}

public List<string> AllP() // Возвращает строковую коллекцию фраз.

{

Read();

List<string> res = new List<string>();

foreach (Phrase p in phrases)

{

res.Add(p.ToString());

}

return res;

}

public List<string> Search(string s, int n) // Метод поиска

{

Read();

s = s.ToLower();

List<string> res = new List<string>();

for (int k = 0; k < this.Length; k++)

{

string text = "";

if (n == 0)

text = this[k].Content.ToLower(); // Поиск по тексту.

if (n == 1)

text = this[k].Author.Name.ToLower(); // По автору.

if (n == 2)

text = this[k].Theme.ToLower(); // По теме.

if (n == 3)

text = this[k].Source.Name.ToLower();// По источнику.

if (n == 4)

text = this[k].Source.Year;// По году.

string temp = "";

for (int i = 0; i <= text.Length - s.Length; i++)

{

for (int j = i; j < i + s.Length; j++)

{

temp += text[j];

}

if (temp.Equals(s))

{

res.Add(this[k].ToString());

break;

}

temp = "";

}

}

return res;

}

}

Форма AddAuthor

public partial class AddAuthor : Form

{

AList alist = new AList();

private double id;

private bool add;

Author temp;

public AddAuthor()

{

InitializeComponent();

}

private bool Check() // Проверка - заполнены ли все поля.

{

if (String.IsNullOrWhiteSpace(ntextBox.Text)) return false;

if (String.IsNullOrWhiteSpace(btextBox.Text)) return false;

if (String.IsNullOrWhiteSpace(dtextBox.Text)) return false;

if (String.IsNullOrWhiteSpace(biotextBox.Text)) return false;

return true;

}

private void AddAuthor\_Load(object sender, EventArgs e) // Загрузка формы.

{

alist.Read();

if (alist.Length == 0 || alist[0].AddChange) // В случае добавления автора.

{

add = true;

AddButton.Text = "Добавить";

Text = "Добавить автора";

}

else // В случае изменения информации.

{

add = false;

AddButton.Text = "Изменить";

Text = "Изменить информацию об авторе";

for (int i = 0; i < alist.Length; i++)

{

if (alist[i].Change)

{

ntextBox.Text = alist[i].Name;

btextBox.Text = alist[i].Born;

dtextBox.Text = alist[i].Die;

biotextBox.Text = alist[i].Bio;

break;

}

}

}

alist[0].AddChange = false;

}

private void Clear() // Очистить все поля.

{

ntextBox.Text = "";

btextBox.Text = "";

dtextBox.Text = "";

dtextBox.Mask = "00/00/0000";

btextBox.Mask = "00/00/0000";

biotextBox.Text = "";

}

private void AddButton\_Click(object sender, EventArgs e) // Кнопка добавления автора(изменения информации об авторе).

{

if (Check())

{

string BornYear = "";

string DeadYear = "";

try {

if (!alist.checkDate(btextBox.Text))

{

MessageBox.Show("Неверная дата");

return;

}

if (!alist.checkDate(dtextBox.Text))

{

MessageBox.Show("Неверная дата");

return;

}

}

catch

{

MessageBox.Show("Неверная дата");

return;

}

BornYear = btextBox.Text.Split('.')[2];

DeadYear = dtextBox.Text.Split('.')[2];

if (Convert.ToInt32(DeadYear) < Convert.ToInt32(BornYear))

{

MessageBox.Show("Неверная дата");

return;

}

if (add == true) // В случае добавления автора.

{

try

{

id = 0;

if (alist.Length == 0)

{

temp = new Author(alist.RemoveSpaces(ntextBox.Text), btextBox.Text, dtextBox.Text, alist.RemoveSpaces(biotextBox.Text), id);

}

else

{

id = alist[alist.Length - 1].Id + 1;

temp = new Author(alist.RemoveSpaces(ntextBox.Text), btextBox.Text, dtextBox.Text, alist.RemoveSpaces(biotextBox.Text), id);

}

}

catch

{

MessageBox.Show("Неверно заполнены поля");

return;

}

if (alist.Exist(temp))

{

MessageBox.Show("Этот автор уже существует");

return;

}

alist.Add(temp);

DialogResult = DialogResult.OK;

alist.Write();

}

else // В случае изменения информации об авторе.

{

PList plist = new PList();

plist.Read();

alist.Read();

for (int i = 0; i < alist.Length; i++)

{

if (alist[i].Change)

{

alist[i].Name = alist.RemoveSpaces(ntextBox.Text);

alist[i].Born = btextBox.Text;

alist[i].Die = dtextBox.Text;

alist[i].Bio = alist.RemoveSpaces(biotextBox.Text);

alist[i].Change = false;

for (int j = 0; j < plist.Length; j++)

{

if (plist[j].Author.Id == alist[i].Id)

{

plist[j].Author = alist[i];

}

}

break;

}

}

alist.Write();

plist.Write();

}

Clear();

Close();

}

else

{

MessageBox.Show("Не все поля заполнены");

}

}

private void CancelButton\_Click(object sender, EventArgs e) // Кнопка отмены.

{

Clear();

Close();

}

}