МІНІСТЕРСТВО ОСВІТИ ТА НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ім. ІГОРЯ СІКОРСЬКОГО”

«ОСНОВИ ПРОГРАМУВАННЯ - 3»

Курсова робота на тему

« Розробка програми «Текстовий Редактор» »

Дата “___”_____

Виконав: студент 1 курсу ТЕФ
Кафедра АПЕПС
Гр. ТВ-61
Артамонов Олексій Юрійович

Захищена з оцінкою:

Київ – 2017

Національний технічний університет України
“Київський політехнічний інститут”

Факультет “Теплоенергетичний”
(повна назва)

Кафедра “Автоматизації проектування енергетичних процесів та систем”
(повна назва)

Освітньо-кваліфікаційний рівень “бакалавр”

Напрям підготовки 6.050103 „Програмна інженерія”
(шифр і назва)

Спеціальність “Програмне забезпечення розподільних систем”
(шифр і назва)

Завдання

на курсову роботу по “Основи програмування - 3”:
студенту групи ТВ-61 Артамонову Олексію Юрійовичу
(номер групи) (прізвище ім’я по батькові)

Вихідні дані: _____
_____

Перелік розробляємих питань:
1. Аналіз завдання на курсову роботу та робота з літературою;
2. Формування макету програмного продукту;
3. Проектування та тестування програмного продукту;
4. Оформлення пояснювальної записки та додаткових матеріалів.

Перелік представляємих до захисту матеріалів:
1. Пояснювальна записка;
2. Компакт диск з програмним продуктом та електронною версією пояснювальної записки.

Основна література:
_____
_____
_____

Дата видачі завдання: “_____” _____
Строк захисту курсової роботи: “_____” _____
Керівник: _____ (Крячок О.С.)

Графік виконання курсової роботи

Календарний план

виконання курсової роботи по дисципліні “Основи програмування - 3”

студент _____ група _____

(прізвище ім’я по батькові) (номер групи)

№ п/п	Аналіз курсової роботи та опрацювання першоджерел	Строк виконання етапів проекту (роботи)	Примітки
1	Аналіз курсової роботи та опрацювання першоджерел	*28.03.2017*	
2	Формування макета програми	*10.04.2017*	
3	Проектування та тестування	*20.04.2017*	
4	Оформлення пояснювальної записки	*10.05.2017*	
5	Здача записки на перевірку	*19.05.2017*	
6	Захист курсової роботи	*25.04.2017*	

Студент _____ _____

(підпис) (прізвище ім’я по батькові)

Узгоджено з керівником роботи: “_____” _____

_____ (Крячок О.С.)

ЗМІСТ

АНОТАЦІЯ

Була поставлена задача розробки Текстового редактора. Сам Текстовий редактор – це комп'ютерна програма-застосунок призначена для створення й зміни текстових файлів (вставки, видалення та копіювання тексту, заміни змісту), а також їх перегляду на моніторі, пошуку фрагментів тексту тощо. Результатом курсової роботи є робоча програма «Текстовий редактор». Програма «Текстовий редактор» розроблена з використанням компілятора Microsoft Visual Studio. Вихідний текст програми написаний на мові C++. Вивчено класи які містять методи для змінення тексту. Приведені різні функції для обробки текстових файлів. Виконана програмна реалізація Текстового Редактора.

SUMMARY

Was tasked with developing text editor. The very text editor - a computer program application designed to create and change text files (insert, delete and copy text, replacing contents), and view them on a monitor, search, text fragments like. The result is of course working program "Text Editor". The "Text Editor" compiler developed using Microsoft Visual Studio. Source code written in C ++. Studied classes containing methods for modification. Resulted various functions for processing text files. Software implementation executed a text editor.

ВСТУП

Метою даного курсового проекту є створення Текстового редактору, який і буде об'єктом дослідження. Текстовий редактор — програма, яка забезпечується програмно керованим електронним пристроєм — комп'ютером. Деякі текстові редактори забезпечують також розширену функціональність: підсвічування синтаксису, сортування рядків, шаблони, конвертацію кодування символів тощо. Така функціональність часто характерна для *редакторів коду*, призначених для написання сирцевого комп'ютерних програм.

Інші текстові редактори мають розширені функції форматування тексту, впровадження в нього графіків, формул, таблиць та об'єктів. Такі редактори часто називають текстовими процесорами й призначені вони для створення різного роду документів — від особистих листів до офіційних паперів. Класичні приклади — Microsoft Word і Libre Office.

Ще один клас програм цієї групи — текстові середовища — по суті, повноцінні робочі середовища, в яких можна вирішувати найрізноманітніші завдання: за допомогою надбудов вони дозволяють писати й читати листи, веб-канали, працювати в вікі й Вебі, вести щоденник, керувати списками адрес і завдань. Представники цього класу — Emacs, Archy, Vim та Acme з операційної системи Plan 9. Такі програми можуть служити середовищами розробки програмного забезпечення, в кожному разі, завжди містять текстовий редактор як необхідний інструмент програмування. Одним із найпоширених застосувань засобів комп'ютерних інформаційних технологій є створення та опрацювання різноманітних текстових документів. Введення реферату, заповнення бланку, створення листа для відправлення електронною поштою, введення вихідного тексту програми на одній із мов програмування є створенням та опрацюванням текстового документа. Залежно від типу документа та його призначення використовуються різні програмні засоби, які містять, відповідно до типу засобу, документа, набір послуг опрацювання текстових документів.

ТЕХНІЧНЕ ЗАВДАННЯ

на курсовий проект з дисципліни "Основи програмування – 2" **Студент**

Артамонов О.Ю.

Група ТВ-51

Тема: Розробка програми «Текстовий редактор»

Загальна формулювання завдання

Необхідно розробити програму "Текстовий редактор" яка виконує основні операції по редагуванню тексту, а також демонструє реалізацію роботи з файлами в середовищі програмування C++/CLI Windows Forms.

Вимоги до графічного і призначеного для користувача інтерфейсів:

• програма повинна працювати в графічному режимі;

• програма повинна містити поле для тексту.

Вимоги до функціональних можливостей:

• реалізувати можливість зберігання(у тому числі у різних форматах), відкриття, створення текстових файлів.

• реалізувати копіювання, додавання, вирізання, виділення, видалення та інше редагування тексту.

1. АНАЛІТИЧНИЙ РОЗДІЛ

1.1. ОГЛЯД ІСНУЮЧИХ ПРОГРАМНИХ РІШЕНЬ

1)Блокнот

Блокнот(Рис.1.1) - простий Текстовий Редактор який є частиною операційних систем Microsoft Windows, починаючи з виходом в 1985 році Windows 1.0.

Блокнот використовує віконний клас EDIT. Аж до виходу у 2000 року Windows ME підтримувалися тільки самі базові функції, багато функцій були доступні тільки з меню, а максимальний розмір файлу становив 64 кілобайти (межа класу EDIT). В даний час редактор підтримує контекстну заміну, гарячі клавіші (наприклад, Ctrl-S для збереження файлу), знятий межа в 64 Кбайт і додана підтримка Юнікоду.

Крім Windows, Блокнот здатний виконуватися також в ReactOS і Wine.

Альтернативою Блокноту є текстовий редактор MS-DOS (EDIT.COM), який можна викликати з командного рядка у вигляді «edit».

Блокнот не здатний коректно працювати з файлами в текстовому форматі Unix, де символом перенесення є байт з кодом 10, на відміну від Windows і DOS, де використовуються байти 13, 10 (див. Статтю Новий рядок).

Існує безліч безкоштовних більш функціональних програм, які заміняють стандартний Блокнот, наприклад: Bred, AkelPad, PSPad, Notepad ++ і Notepad2. Вони були особливо корисні в системах Windows версій 4.x, в яких функціональність Блокнота гранично обмежена.

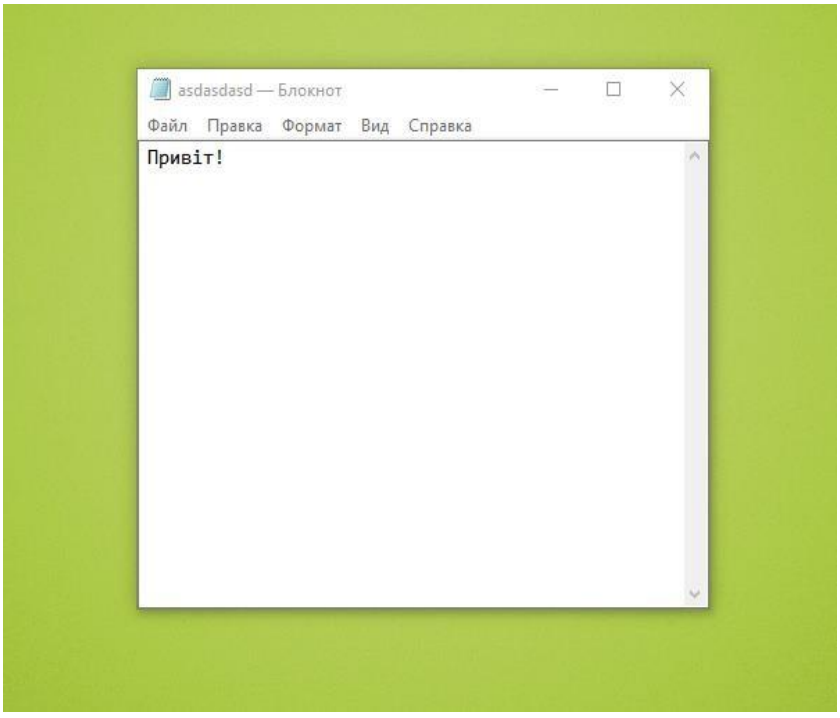

Рис.1.1- Блокнот (англ. Notepad).

2)Microsoft Word

Текстовий процесор, що випускається фірмою Майкрософт, входить до складу офісного пакету «Microsoft Office». Перша версія, «Multi Tool Word», була написана для Xenix і перенесена під DOS у 1983 році. Пізніше створено версії для Apple Macintosh (1984), Microsoft Windows (1989), SCO UNIX, OS/2.

Microsoft Word багатьом зобов'язаний Bravo — текстовому процесору з оригінальним графічним інтерфейсом, розробленому в дослідницькому центрі «Xerox PARC». Творець Bravo, Чарльз Симоні (Charles Simonyi) залишив PARC в 1981 році. Того ж літа Симоні переманив Броді, з яким разом працював над Bravo.

Перший випуск Word для MS-DOS відбувся в кінці 1983 року. Він був погано прийнятий ринком, продажі знижувала наявність конкуруючого продукту — WordPerfect. Проте версія для Макінтоша, випущена в 1985 році, набула широкого поширення. Через два роки «Word 3.01 для Macintosh» підсилив позиції (версія 3.0 рясніла помилками і швидко була замінена). Як і інше програмне забезпечення для Макінтоша, Word був повністю WYSIWYGредактором (принцип «What You See Is What You Get» — «маю те, що бачу»).

Перша версія Word для Windows, випущена в 1989 році, продавалася за ціною 500 доларів США. Вона демонструвала вибраний компанією Майкрософт шлях розвитку: як і сама Windows, вона багато що узяла від Macintosh, і використовувала стандартні клавіатурні скорочення (наприклад, CTRL-S для збереження файлу). Після випуску наступного року Windows 3.0 продажі поповзли вгору (Word 1.0 набагато краще працював з Windows 3.0, ніж зі старішими версіями Windows/386 і Windows/286), головний конкурент — WordPerfect — не зміг випустити робочу версію під Windows, що виявилося для нього смертельною помилкою. Версія 2.0 затвердила WinWord на позиції лідера ринку.

Наступні версії додавали можливостей, що виходили за рамки простого текстового процесора. Інструменти малювання дозволяли виконувати примітивні операції верстки, такі як додавання графіки в документ, хоча, звичайно, спеціалізовані програми для верстки краще справляються з цими завданнями. Впровадження об'єктів, порівняння версій документа, багатомовна підтримка і багато інших можливостей були додані за декілька років, опісля.

Microsoft Word на сьогодні є найпопулярнішим текстовим процесором у вжитку, що зробило його закритий формат документа стандартом, і багато програм конкурентів мають підтримку сумісності з даним форматом.

Розширення «.doc» на платформі IBM PC стало синонімом двійкового формату Word 97—2000. Фільтри експорту і імпорту в даний формат присутні в більшості текстових процесорів. Велика частина інформації, потрібної для роботи з даним форматом, здобувається за допомогою зворотного інжинірингу, оскільки велика її частина відсутня у відкритому доступі. Формат документа різних версій Word часто міняється, відмінності бувають досить тонкими. Форматування, що нормально виглядає в останній версії, може не відображатися в старих версіях програми, оскільки зворотна сумісність часто відсутня.

Як і решта застосунків з Microsoft Office, Word може розширювати свої можливості за допомогою використання вбудованої макромови (спочатку використовувався WordBasic, проте з версії Word 97 застосовується VBA — Visual Basic for Applications). Проте це надає широкі можливості для написання вбудовуваних в документи вірусів (так звані «макровіруси»). Найяскравішим прикладом була епідемія хробака Melissa. У зв'язку з цим, багато хто вважає розумною рекомендацію завжди виставляти найвищий рівень налаштувань безпеки при використанні Word. Також бажано використовувати антивірусне програмне забезпечення. Першим вірусом, що заражав документи Microsoft Word, був DMV, створений в грудні 1994 року Дж. Макнамара для демонстрації можливості створення макровірусів. Першим же вірусом, що потрапив в «дику природу» і викликав першу у світі епідемію макровірусів (у липні-серпні 1995), був Concept.

12 серпня 2009 року суд штату Техас заборонив продаж програми Word на території США, в зв’язку з тим що Microsoft незаконно використовує метод читання XML-файлів, патент на який належить канадській компанії i4i.

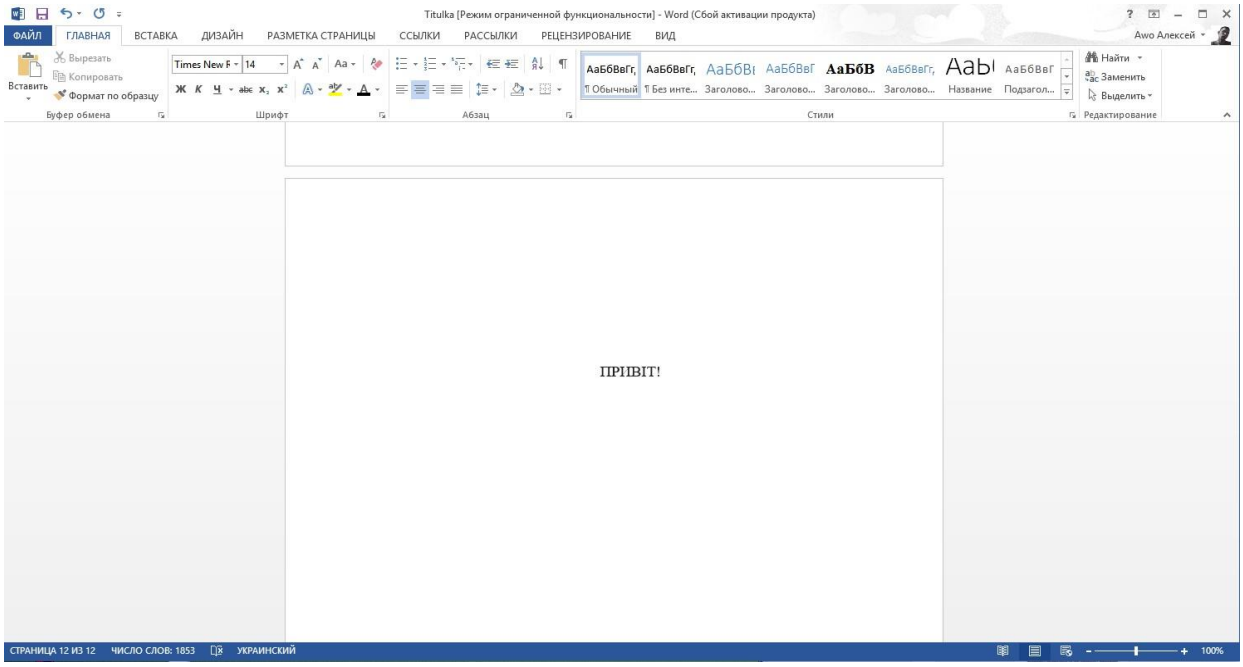

Рис.1.2- Текстовий процесор Microsoft Word.

1)Notepad++

Цей скромний текстовий редактор є незамінним інструментом для величезного числа програмістів, письменників, журналістів і самих різних людей, які потребують простому засобі для створення простих текстових заміток. Лише деякі зберігають вірність стандартному Блокноту Windows, хоча б раз спробувавши Notepad ++. У 2003 році програміст Дон Хо, вдосталь намучившись з одним з редакторів програмного коду, вирішив створити щось більш підходяще для своєї роботи. Тоді і з'явився Notepad ++ полюбився багатьом користувачам за найширші функціональні можливості, які поєднуються з простим інтерфейсом і бездоганною продуктивністю. У компанії, де працював Дон Хо, використовувався безкоштовний редактор вихідного року JEXT. Він написаний на Java і працював досить повільно. Тому майбутній творець Notepad ++ вирішив власними руками створити альтернативне рішення. До розробки він приступив у 2003 році. Прототипом програми послужила Scintilla. Але це рішення не було підтримано компанією, в якій працював Дон. Йому довелося займатися проектом в свій вільний час. Днем народження Notepad ++ можна вважати 25 листопада 2003 року, коли програма була викладена у відкритий доступ на SourceForge. Проект Notepad ++ запускався вельми скромно. Дон Хо визнається, що він не думав про створення популярного продукту. Автор поділився програмою з усіма бажаючими через SourceForge. Будь-хто міг завантажити інсталяційний файл і вихідний код. Розробник не очікував, що Notepad ++ стане такою популярною програмою. Але саме популярність і призначений для користувача відгук дозволяють постійно покращувати її

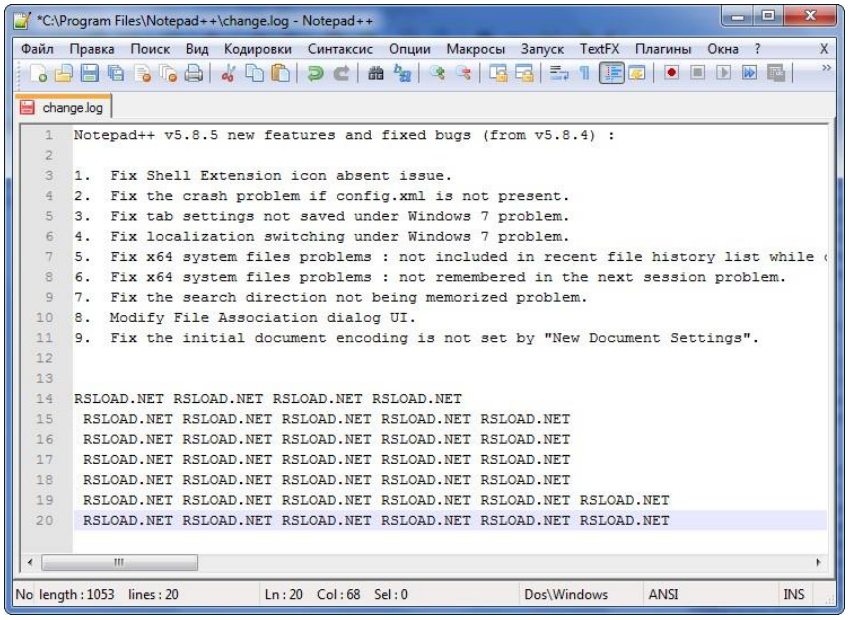

Рис.1.3- Текстовий редактор Notepad++.

1.2. ФУНКЦІОНАЛЬНІ ВИМОГИ ДО РОЗРОБЛЮВАНОЇ ПРОГРАМИ

1) Програма повинна мати простий та інтуїтивно зрозумілий інтерфейс.
2) Повинна працювати на будь-якому ПК, на якому встановлена операційна система Windows.
3) Набір тексту.
4) Можливості введення тексту декількома мовами.
5) Редагування фрагментів тексту.
6) Пошук потрібних фрагментів тексту.
7) Друкування всього тексту.
8) Створення стандартних документів (листів, резюме, записок).
9) Збереження тексту на зовнішніх носіях.

У ході виконання роботи всі вищезгадані вимоги були виконані. Програма перевірялася на кількох комп’ютерах, жодних відмінностей у її роботі не було помічено

2. КОНСТРУКТОРСЬКИЙ РОЗДІЛ

2.1. ПРОЕКТУВАННЯ ІНТЕРФЕЙСУ

Однією з найважливіших функцій будь-якої програми є введення і виведення даних. Виведені дані це те, що повідомляється користувачеві. Вхідні дані це те, що користувач повідомляє програмі. Два поля для введення данних в програмі представлені у вигляді графічного відображення вікна програми (рисунок 2.1).

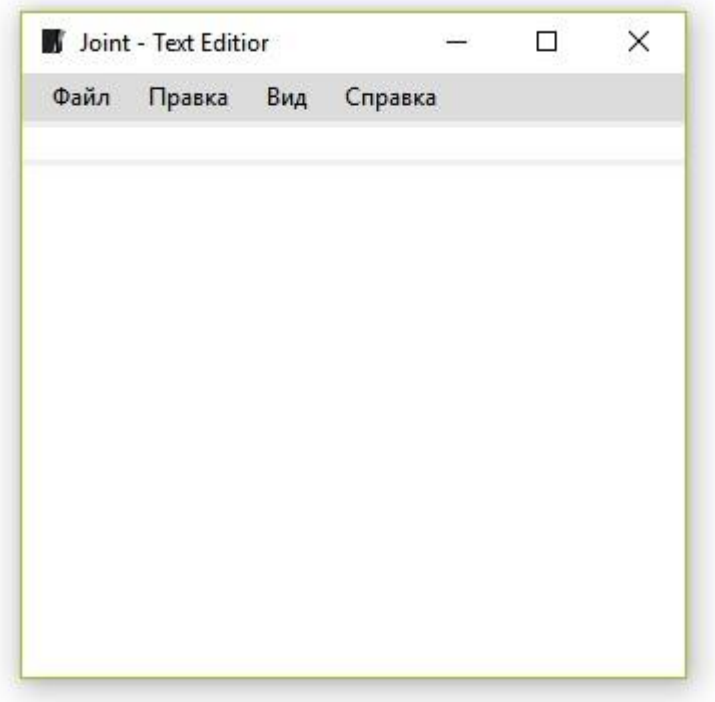

Рис.2.1- Вікно програми.

2.2. МЕТОДИ І СПОСОБИ УСУНЕННЯ ПОМИЛОК

Відладка - це комплексний процес з виявлення та виправлення дефектів в програмному забезпеченні. Самі ж дефекти, як правило, виявляється в процесі тестування ПО.

Відладка складається з наступних етапів:

• відтворення дефекту (будь-яким з доступних способів);

• аналіз дефекту (пошук причини виникнення дефекту);

• кодування (виправлення) дефекту;

• валідація виправлення;

• інтеграція виправлення в кодову базу або цільову систему;

• додаткові валідації після інтеграції.

На будь-якому етапі відладки можуть виникнути нові дефекти, які доведеться налагоджувати. Наприклад, якась частина виправлення в коді працює не так як очікується і відповідно доведеться налагоджувати цю частину в ізоляції і знову основний час йде на пункти 1 і 2 і т.д. Етап налагодження можна вважати закінченим, якщо програма правильно працює на двох-трьох наборах вхідних даних. Деякі методи налагодження ПО використовувані на даний момент в індустрії:

• запуск програми з під відладчика;

• аналіз поведінки системи;

• Блок тестування;

• аналіз коду без виконання програми;

• виконання програми (або її частини) в іншому середовищі;

• налагодження трансляцією коду.

3. ОГЛЯД РЕЗУЛЬТАТІВ

3.1 ВИЗНАЧЕННЯ СТРУКТУРИ І СКЛАДУ ПРОГРАМНОЇ СИСТЕМИ

У програмі використовуються стандартні C++/CLI Windows Forms. Елемент управління **RichTextBox** (рисунок 3.1) являє собою поле для вводу тексту або його редагування. Користувач може вводити і редагувати текст. Елемент управління також надає розширені можливості форматування, ніж стандартний елемент управління TextBox. Текст може бути завантажений в форматі RTF (RTF) або текстового файлу. Текст в елементі управління можна призначити форматування знаків і абзаців.

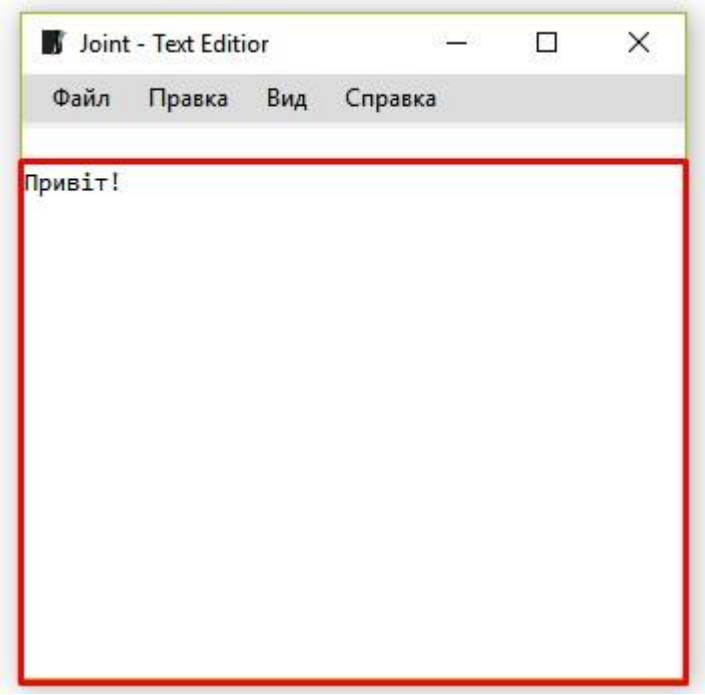

Рисунок 3.1- Приклад елемента управління **RichTextBox**.

MenuStrip(рис. 3.2) являє контейнер для структури меню. Можна додати ToolStripMenuItem об'єкти до MenuStrip які представляють окремі команди меню в структурі меню. Кожна ToolStripMenuItem може бути командою для додатка або батьківським меню для інших елементів вкладеного меню.

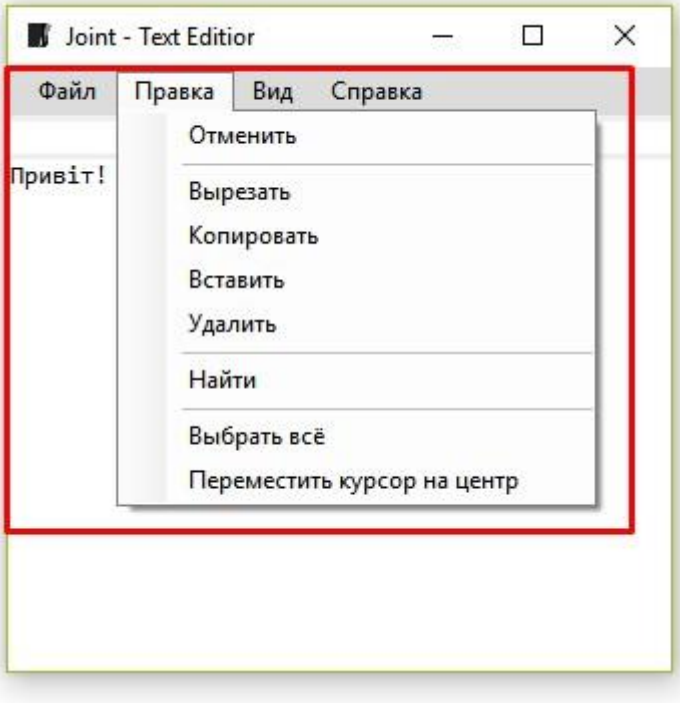

Рисунок 3.2- Приклад елемента управління **MenuStrip** .

Компонент **Label** (рисунок 3.3) призначений для виведення тексту на поверхню форми. Властивості компонента визначають вигляд і розташування тексту. | Компонент **Label** – поле виводу тексту. Компонент **Button**(рисунок 3.3) являє собою командну кнопку. Компонент **Button** – командна кнопка. В **TextBox**(рисунок 3.3), користувач може вводити текст в додатку. Цей елемент управління має додаткові функціональні можливості, відсутні в стандартних текстових полях елемента керування в Windows, включаючи редагування в багаторядковому режимі і введення знаків пароля. Як правило елемент управління **TextBox** використовується для відображення або введення одного рядка тексту. Можна використовувати Multiline і ScrollBars Властивості, щоб включити кілька рядків тексту, щоб вводити або відображати.

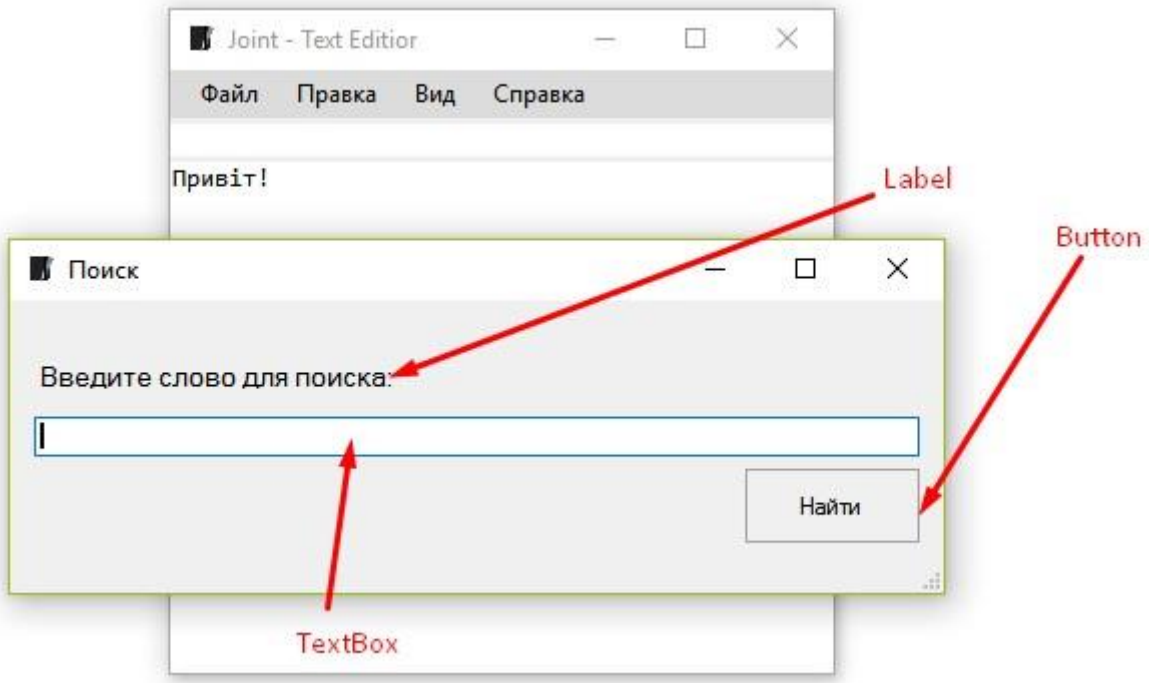

Рисунок 3.2- Приклад елементів управління **Label, Button, TextBox**.

3.2 Опис функцій

Програма була створена в такому середовищі програмування, як C++/CLI Windows Forms. Windows Forms - інтерфейс програмування додатків (API), що відповідає за графічний інтерфейс користувача і є частиною Microsoft .NET Framework. Даний інтерфейс спрощує доступ до елементів інтерфейсу Microsoft Windows за рахунок створення обгортки для існуючого Win32 API в керованому коді.

Причому керований код - класи, що реалізують API для Windows Forms, що не залежать від мови розробки. Тобто програміст однаково може використовувати Windows Forms як при написанні ПЗ на C #, C ++, так і на VB.Net, J # і ін.

З виходом .NET Framework 3.0 Microsoft випустила новий API для малювання призначених для користувача інтерфейсів: Windows Presentation Foundation, який базувався на DirectX 11 і декларативну мову опису інтерфейсів XAML. Однак, навіть незважаючи на все це, Windows Forms і WPF все ще пропонують схожу функціональність, і тому Windows Forms ні скасований на користь WPF, а продовжує використовуватися як альтернативна технологія побудови інтерфейсів поряд з WPF.

У файлі Form1.h виконана вся моя програма. На початку програми, підключаю директиви, які потрібні для використання в моїй програмі. Після цього підключенню теж потребує простір імен std System, System::ComponentModel, System::Collections, System::Windows::Forms, System::Data, System::Drawing. Після основних підключень, я починаю розробляти функціонал та оформлення програми.

Наведу приклад реалізації декількох функцій:

Функція зберігання файлу:

Ця функція реалізує створення файлу. Перша умова цієї функції передбачає змінення тексту або його наявність у формі **RichTextBox**. Далі я ввожу змінну якій потім привласнюю **MessageBox**. В ньому буде відображатись питання "Сохранить изменения в файле ?", назва файлу та три кнопки «Yes», «No», «Cancel». Потім я за допомогою конструкції switch, case, якщо користувач натиснув кнопку «Yes», задаю початкове та можливі розширення файлу при збереженні. Якщо користувач натискає кнопку «ОК», змінній path привласнюю назву яку ввів користувач у полі **saveFileDialog1**. Наступний рядок – це зберігання фалу(назви та вмісту). По завершенню показується вікно з надписом "Изменения сохранены." – що означає що змінення у файлі успішно зберіглися та очищається поле вводу тексту **RichTextBox.** Якщо ж користувач натискає кнопку «No», то просто очищається поле вводу тексту. При натисканні «Cancel» вікно **MessageBox** закривається.

```
private: System::Void cОздатьToolStripMenuItem_Click(System::Object^  sender, System::EventArgs^  e) {
  if( richTextBox1->Modified==true ) {
                        System::Windows::Forms::DialogResult res;
 res = MessageBox::Show("Сохранить изменения в файле "+((String^)gcnew String(filename.c_str()))+"? ",
"Блокнот",MessageBoxButtons::YesNoCancel);
                        switch(res)
                        {
                        case System::Windows::Forms::DialogResult::Yes:
                                saveFileDialog1->DefaultExt = "*.txt";
    saveFileDialog1->Filter = "txt files (*.txt)|*.txt|rtf files (*.rtf)|*.rtf|doc files (*.doc)|*.doc|All files (*.*)|*.*";
                        if(path == "Безимянный.txt")
                        if(saveFileDialog1->ShowDialog()==System::Windows::Forms::DialogResult::OK) {
                                path = SysToStd(saveFileDialog1->FileName);
                                richTextBox1->SaveFile(stdtosys(path), RichTextBoxStreamType::PlainText);
                                MessageBox::Show("Изменения сохранены.");richTextBox1->Clear();
                        }
                                break;
                        case System::Windows::Forms::DialogResult::No:
                                richTextBox1->Clear();
                        break;
                        case System::Windows::Forms::DialogResult::Cancel:
                        break;
                        }
                }
                        }
```

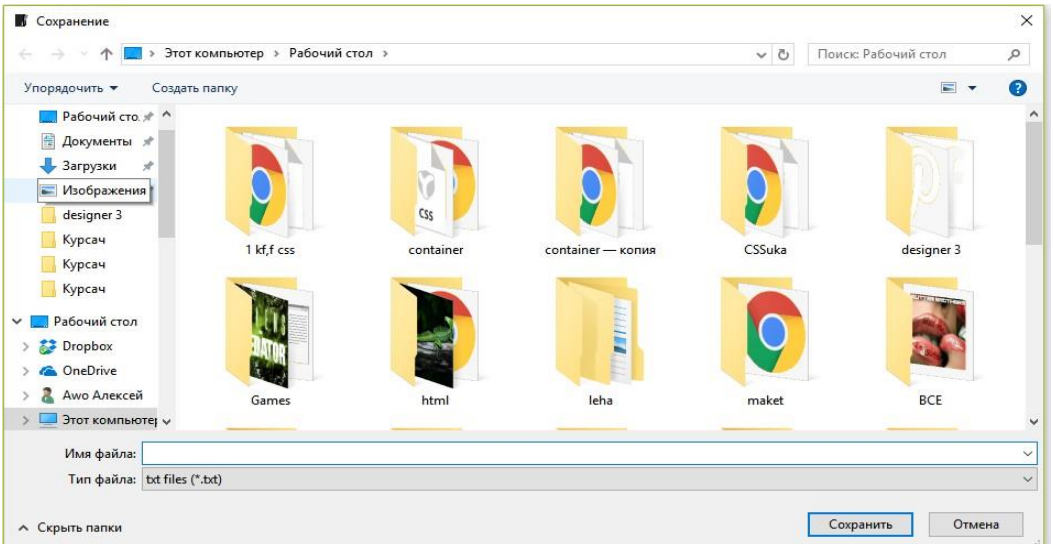

Рисунок 3.3- Приклад роботи функції **«Сохранить»**.

Функція редагування шрифту:

```
private: System::Void шрифтToolStripMenuItem_Click(System::Object^  sender,
System::EventArgs^  e) {
 FontDialog^ fontDialog1 = gcnew FontDialog;
                              fontDialog1->ShowColor = true;

     fontDialog1->Font = richTextBox1->Font;       fontDialog1->Color =
richTextBox1->ForeColor;
      if ( fontDialog1->ShowDialog() != System::Windows::Forms::DialogResult::Cancel )
     {
       richTextBox1->Font = fontDialog1->Font;          richTextBox1->ForeColor =
fontDialog1->Color;
     }
                    }
```

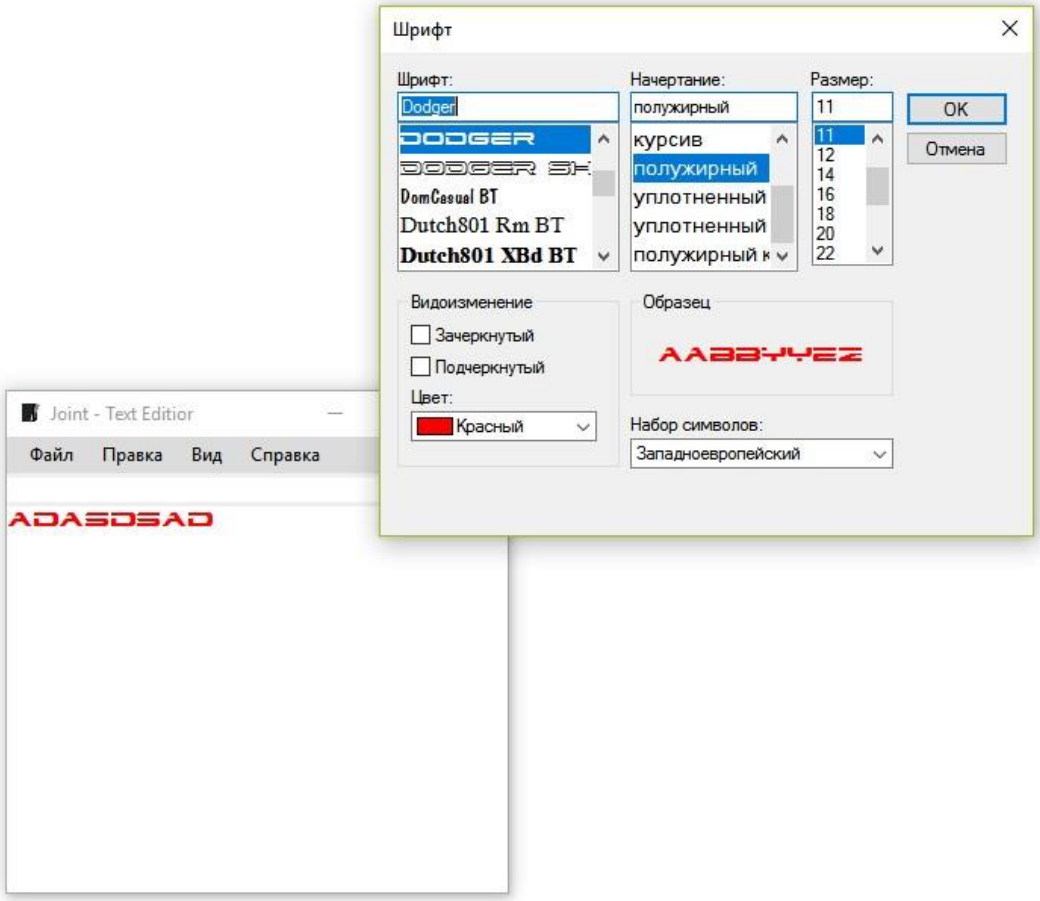

Рисунок 3.4- Приклад роботи функції **«Шрифт…»**.

Функція друкування:

```
private:          System::Void          printDocument1_PrintPage(System::Object^                    sender,
System::Drawing::Printing::PrintPageEventArgs^  e) {
 e->Graphics->DrawString(richTextBox1->Text, richTextBox1->Font, Brushes::Black,
(float)80, (float)80);

}
 private:    System::Void    печатьToolStripMenuItem_Click(System::Object^         sender,
 System::EventArgs^  e) {  printDialog1->ShowDialog();
              if (DialogResult == System::Windows::Forms::DialogResult::OK)
        {
                printDocument1->Print();
        }
                 }
```

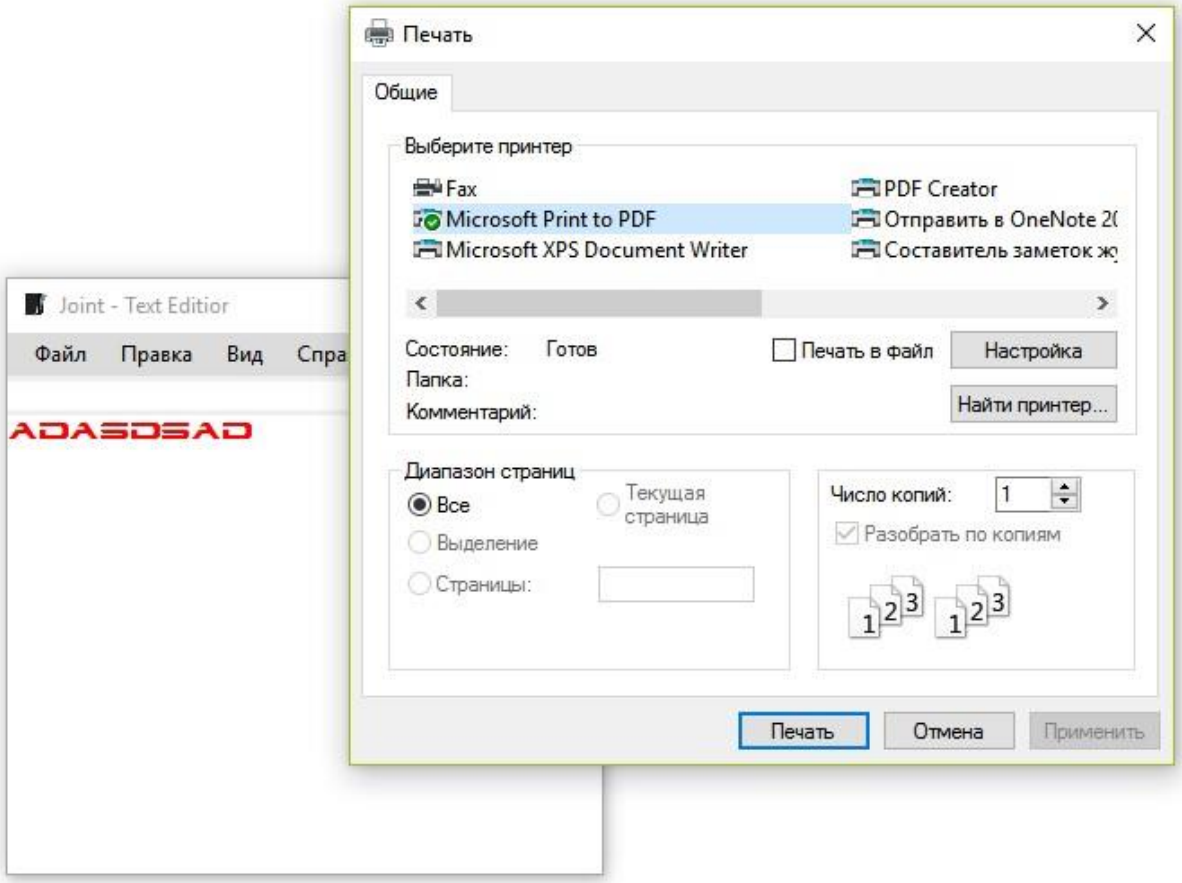

Рисунок 3.5- Приклад роботи функції **«Печать»**.

Функція пошуку:

Для пошуку я використовую метод класу **RichTextBox** який називається **Find.** Цей метод здійснює пошук тексту, зазначеного в **Search_Text->Text** і повертає розташування першого символу для пошуку рядка в елементі управління. З цією версією методу **Find**, можна вказати параметри, які дозволяють розширити або звузити область пошуку. Можна вказати параметри, які дозволяють розрізняти регістр шуканого слова або пошук слова цілком, а не частин слова. Вказавши **RichTextBoxFinds.Reverse**, можна виконати пошук тексту з кінця документа до початку.

```
MyForm^ Form2 = gcnew MyForm();
                        Form2->ShowDialog();

                        int var=1;
                        if (Search_Text->TextLength == 0)
                                MessageBox::Show("Please Enter data");
                        else
                        {
                                int index = 0;
                                var = 0;
                                String^ temp = richTextBox1->Text;
                                richTextBox1->Text = "";
                                richTextBox1->Text = temp;
                                while (index < richTextBox1->Text->LastIndexOf(Search_Text->Text))
                                {

                                        richTextBox1->Find(Search_Text->Text, index, richTextBox1-
>TextLength, RichTextBoxFinds::None);
                                        richTextBox1->SelectionBackColor = Color::Red;
                                        index = richTextBox1->Text->IndexOf(Search_Text->Text, index) +
1;

                                        var++;
                                }
                        }
                        if (!var)
                        {
                                MessageBox::Show("NO Match Found");
                                Search_Text->Clear();
                        }

        }
```

3.3 КЕРІВНИЦТВО КОРИСТУВАЧА

Вимоги до апаратури і програмного забезпечення. Так як програма виконана в середовищі програмування C++/CLI Windows Forms, то вимоги до апаратного та програмного забезпечення мінімальні. Установка програми на комп'ютер користувача полягає в копіюванні папки програми і установки ярлика на Робочий стіл. Створіть в будь-якому розділі жорсткого диска нову папку і скопіюйте в неї всі файли папки "Курсовий проект". Запускати слід файл Joint - Text Editior.exe безпосередньо з папки або за допомогою ярлика кнопкою Enter або подвійним клацанням миші. Якщо потрібна доробка програми, то необхідно мати вихідний файл Form1.h , який відкривається в середовищі програмування C++/CLI Windows Forms. Порядок роботи з програмою опишемо в керівництві користувача. Тут наводиться опис прийомів управління програмою. Порядок роботи: Запуск програми здійснюється подвійним клацанням миші на файлі Joint - Text Editior.exe або по його ярлику. Після запуску відкривається вікно програми. Далі користувач може вводити текст та редагувати його. Також зберігати та відкривати текстові файли.

Щоб здійснити пошук ви повинні:

1) Ввести у поле під меню слово, яке бажаєте знайти ;
2) Відкрити розділ меню під назвою «**Правка**»;
3) Натиснути на вкладку «**Найти**»;

Також ви можете переносити курсор на центр(наприклад для заголовків)**, для цього:**

1) Відкрийте розділ меню під назвою «**Правка**»;
2) Натисніть на вкладку «**Перенос курсора на центр**»;

Щоб перенести курсор в початкове положення зробіть те саме ще раз; **Щоб здійснити редагування шрифту та його кольору:**

1) Відкрийте розділ меню під назвою «**Вид**»;
2) Натисніть «**Шрифт…**»;

Щоб здійснити редагування шрифту та його кольору:

1) Відкрийте розділ меню під назвою «**Файл**»;
2) Натисніть «Печать»;

ВИСНОВОК

В результаті виконання даного курсового проекту було:

- Запропоновано розробити програму «Текстовий Редактор» з використанням середовища програмування C++/CLI Windows Forms.
- Встановлено вимоги до графічного інтерфейсу та до функціональних можливостей.
- Розроблено програму «Текстовий Редактор», яка дозволяє редагувати текстові файли .
- Проведено дослідження отриманої програми.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Страуструп Б. Язык программирования C++: Пер. с англ. — 3-е спец. изд. — М.: Бином, 2003. — 1025 с

2. Уэйт М., Язык Си/ М. Уэйт, С. Прата, Д. Мартин. – М.: Мир, 1994 – 112с .

3. Керніган Б. Мова С / Б. Керніган Д. Річі . – М.:Вільямс ,2009 − 272с.

4. Литвиненко Н. А. - Технология программирования на C++, – М.: Литвиненко ,2010 – 152с.

5. Андреева Е. А. - Системы счисления и компьютерная арифметика. Учебное пособие, – М.: Андреева , 2004 – 189с.

6. Лафоре Р./ Объектно-ориентированное програмирование в C++ - 4-е издание/ 922 с.

7. Основи програмування та алгоритмічні мови – 2. Спеціальні засоби мови програмування. Методичні вказівки до виконання комп’ютерних практикумів (частини 1, 2) для студентів напрямку підготовки 6.050103 – «Програмна інженерія» денної форми навчання/Укладачі: Крячок О.С., Кузьменко І.М., Гурін А.Л., Круш О.Є. – К.: НТУУ «КПІ», 2012.

ДОДАТОК

Лістинг програми

Form1

```
#pragma endregion
                static System::String^ stdtosys(string str){

                        return gcnew System::String(str.c_str());

                        }
                static const string SysToStd(System::String^ SysStr){
                                using namespace Runtime::InteropServices;
                                return(const char*)(Marshal::StringToHGlobalAnsi(SysStr)).ToPointer();
                        }

private: System::Void cОздатьToolStripMenuItem_Click(System::Object^  sender, System::EventArgs^  e) {
  if( richTextBox1->Modified==true ) {
                        System::Windows::Forms::DialogResult res;
 res = MessageBox::Show("Сохранить изменения в файле "+((String^)gcnew String(filename.c_str()))+"? ",
"Блокнот",MessageBoxButtons::YesNoCancel);
                        switch(res)
                        {
                        case System::Windows::Forms::DialogResult::Yes:
                                saveFileDialog1->DefaultExt = "*.txt";
    saveFileDialog1->Filter = "txt files (*.txt)|*.txt|rtf files (*.rtf)|*.rtf|doc files (*.doc)|*.doc|All files (*.*)|*.*";
                        if(path == "Безимянный.txt")
                        if(saveFileDialog1->ShowDialog()==System::Windows::Forms::DialogResult::OK) {
                                path = SysToStd(saveFileDialog1->FileName);
                                richTextBox1->SaveFile(stdtosys(path), RichTextBoxStreamType::PlainText);
                                MessageBox::Show("Изменения сохранены.");richTextBox1->Clear();
                        }
                                break;
                        case System::Windows::Forms::DialogResult::No:
                                richTextBox1->Clear();
                        break;
                        case System::Windows::Forms::DialogResult::Cancel:
                        break;
                        }

                }
                        }

private: System::Void открытьToolStripMenuItem_Click(System::Object^  sender, System::EventArgs^  e) {
                        if(openFileDialog1->ShowDialog()==System::Windows::Forms::DialogResult::OK) {
                                path = SysToStd(openFileDialog1->FileName);
                                filename = SysToStd(openFileDialog1->FileName;
        richTextBox1->LoadFile(stdtosys(path), RichTextBoxStreamType::PlainText);
                        }
                }

private: System::Void сохранитьToolStripMenuItem_Click(System::Object^  sender, System::EventArgs^  e) {
                        if(path == "Безимянный.txt"){
                        if(saveFileDialog1->ShowDialog()==System::Windows::Forms::DialogResult::OK) {
                                path = SysToStd(saveFileDialog1->FileName);
                                richTextBox1->SaveFile(stdtosys(path), RichTextBoxStreamType::PlainText);
```

```
                }}else richTextBox1->SaveFile(stdtosys(path), RichTextBoxStreamType::PlainText);
            }

private: System::Void выходToolStripMenuItem_Click(System::Object^  sender, System::EventArgs^  e) {
    if( richTextBox1->Modified==true ) {
                System::Windows::Forms::DialogResult res;
 res = MessageBox::Show("Сохранить изменения в файле "+((String^)gcnew String(filename.c_str()))+"? ",
"Блокнот",MessageBoxButtons::YesNoCancel);
                switch(res)
                {
                case System::Windows::Forms::DialogResult::Yes:
                        saveFileDialog1->DefaultExt = "*.txt";
   saveFileDialog1->Filter = "txt files (*.txt)|*.txt|rtf files (*.rtf)|*.rtf|doc files (*.doc)|*.doc|All files (*.*)|*.*";
                if(path == "Безимянный.txt")MessageBox::Show("Изменения сохранены.");
                if(saveFileDialog1->ShowDialog()==System::Windows::Forms::DialogResult::OK) {
                        path = SysToStd(saveFileDialog1->FileName);
                        richTextBox1->SaveFile(stdtosys(path), RichTextBoxStreamType::PlainText);

                        Application::Exit();
                }
                        break;
                case System::Windows::Forms::DialogResult::No:
                        richTextBox1->Clear();
                                Application::Exit();
                break;
                case System::Windows::Forms::DialogResult::Cancel:
                break;
                }
                 }else{Application::Exit();}
            }
private: System::Void копироватьToolStripMenuItem_Click(System::Object^  sender, System::EventArgs^  e)
{
                richTextBox1->Copy();
            }
private: System::Void вырезатьToolStripMenuItem_Click(System::Object^  sender, System::EventArgs^  e) {
                richTextBox1->Cut();
            }
private: System::Void вставитьToolStripMenuItem_Click(System::Object^  sender, System::EventArgs^  e) {
                richTextBox1->Paste();
            }
private: System::Void удалитьToolStripMenuItem_Click(System::Object^  sender, System::EventArgs^  e) {
                richTextBox1->SelectedText=" ";
            }
private: System::Void отменитьToolStripMenuItem_Click(System::Object^  sender, System::EventArgs^  e) {
                if ( richTextBox1->CanUndo == true )
    {
                richTextBox1->Undo();
                richTextBox1->ClearUndo();
    }
            }
private: System::Void выделитьВсеToolStripMenuItem_Click(System::Object^  sender, System::EventArgs^
e) {
                if ( richTextBox1->SelectionLength == 0 )
    {
      richTextBox1->SelectAll();
    }
    richTextBox1->Copy();
            }
```

```
private: System::Void шрифтToolStripMenuItem_Click(System::Object^  sender, System::EventArgs^  e) {
 FontDialog^ fontDialog1 = gcnew FontDialog;
                        fontDialog1->ShowColor = true;
     fontDialog1->Font = richTextBox1->Font;
     fontDialog1->Color = richTextBox1->ForeColor;
     if ( fontDialog1->ShowDialog() != System::Windows::Forms::DialogResult::Cancel )
     {
       richTextBox1->Font = fontDialog1->Font;
       richTextBox1->ForeColor = fontDialog1->Color;
     }
                }

private: System::Void сохранитьКакToolStripMenuItem_Click(System::Object^  sender, System::EventArgs^
e) {
 SaveFileDialog^ saveFile1 = gcnew SaveFileDialog;
     saveFile1->DefaultExt = "*.txt";
     saveFile1->Filter = "txt files (*.txt)|*.txt|rtf files (*.rtf)|*.rtf|doc files (*.doc)|*.doc|All files (*.*)|*.*";
     if ( saveFile1->ShowDialog() == System::Windows::Forms::DialogResult::OK &&
       saveFile1->FileName->Length > 0 )
     {
       richTextBox1->SaveFile( saveFile1->FileName, RichTextBoxStreamType::PlainText );
     }
                }

private: System::Void правкаToolStripMenuItem_Click(System::Object^  sender, System::EventArgs^  e) {
                        if(this->переносКурсораНаЦентрToolStripMenuItem->CheckOnClick==false){
                                this->переносКурсораНаЦентрToolStripMenuItem->Checked=true;
                                this->переносКурсораНаЦентрToolStripMenuItem->CheckOnClick=true;
                        }else{
                        this->переносКурсораНаЦентрToolStripMenuItem->Checked=false;
                        this->переносКурсораНаЦентрToolStripMenuItem->CheckOnClick=false;
                        }
                }

private: System::Void переносКурсораНаЦентрToolStripMenuItem_Click(System::Object^  sender,
System::EventArgs^  e) {
                        if(переносКурсораНаЦентрToolStripMenuItem->CheckOnClick==true){
                                richTextBox1->SelectionAlignment = HorizontalAlignment::Left;
                        }else{
                                richTextBox1->SelectionAlignment = HorizontalAlignment::Center;
                        }
                }

private: System::Void найтиToolStripMenuItem_Click_1(System::Object^  sender, System::EventArgs^  e) {
                        MyForm^ Form2 = gcnew MyForm();
                         Form2->ShowDialog();
                        int var=1;
                        if (Search_Text->TextLength == 0)
                                MessageBox::Show("Please Enter data");
                        else
                        {
                                int index = 0;
                                var = 0;
                                String^ temp = richTextBox1->Text;
                                richTextBox1->Text = "";
                                richTextBox1->Text = temp;
                                while (index < richTextBox1->Text->LastIndexOf(Search_Text->Text))
```

```
                                {
richTextBox1->Find(Search_Text->Text, index, richTextBox1->TextLength, RichTextBoxFinds::None);
                                richTextBox1->SelectionBackColor = Color::Red;
                                index = richTextBox1->Text->IndexOf(Search_Text->Text, index) + 1;
                                        var++;
                                }
                        }
                        if (!var)
                        {
                                MessageBox::Show("NO Match Found");
                                Search_Text->Clear();
                        }
                }

private: System::Void Search_Text_TextChanged(System::Object^  sender, System::EventArgs^  e) {

      /* Search_Text->Visible = false;*/
                }

        private: System::Void printDocument1_PrintPage(System::Object^  sender,
System::Drawing::Printing::PrintPageEventArgs^  e) {
        e->Graphics->DrawString(richTextBox1->Text, richTextBox1->Font, Brushes::Black, (float)80,
(float)80);

}

private: System::Void печатьToolStripMenuItem_Click(System::Object^  sender, System::EventArgs^  e) {
        printDialog1->ShowDialog();
        if (DialogResult == System::Windows::Forms::DialogResult::OK)
        {
                printDocument1->Print();
        }
                }

private: System::Void просмотретьСправкуToolStripMenuItem_Click(System::Object^  sender,
System::EventArgs^  e) {
                        MyForm1^ form1 = gcnew MyForm1;
    Button^ button1 = gcnew Button;
    Button^ button2 = gcnew Button;
    button1->Text = "OK";
    button1->Location = Point(10,10);
    form1->Text = "О Программе";
    form1->HelpButton = true;
    form1->FormBorderStyle = System::Windows::Forms::FormBorderStyle::FixedDialog;
    form1->MaximizeBox = false;
    form1->MinimizeBox = false;
    form1->CancelButton = button1;
    form1->StartPosition = FormStartPosition::CenterScreen;
    form1->Controls->Add( button1 );
    form1->ShowDialog();
                }
};
                                                        }
```