Міністерство освіти і науки України

Львівський національний університет імені Івана Франка

Факультет прикладної математики та інформатики

кафедра теорії

оптимальних процесів

Курсова робота з курсу програмного забезпечення

“Розробка привітальної листівки”

Роботу виконав:

**Граневич Денис Олександрович,**

студент групи ПМА-31

Науковий керівник:

**Мельничин Андрій Володимирович,**

доцент кафедри теорії оптимальних процесів

Львів-2013

Зміст

[Вступ 3](#_Toc312870496)

[Розділ 1. Теоретичні положення 4](#_Toc312870497)

[1.1 Мова програмування C# 4](#_Toc312870499)

[1.2 Windows Presentation Foundation та допоміжні бібліотеки 5](#_Toc312870499)

[Розділ 2. Привітальна листівка 7](#_Toc312870500)

[2.1 Постановка задачі 8](#_Toc312870501)

[2.2 Підходи щодо вирішення поставленої задачі 8](#_Toc312870502)

[Розділ 3. Структура програми 9](#_Toc312870503)

[3.1 Об'єктно орієнтований принцип створення програми 10](#_Toc312870504)

[3.1 Класи і методи, які використані у програмі 13](#_Toc312870507)

[3.3 Візуальне офомлення проекту 13](#_Toc312870507)

[Висновок 18](#_Toc312870508)

[Список використаної літератури 19](#_Toc312870509)

# Вступ

З розвитком інформаційних технологій, змінилися і певні стереотипи. Зокрема люди вже менше і менше використовують друковані матеріали замінюючи їх електронними документами, комп’ютерними фотографіями і т.п. Відповідно з’явилось велика кількість програмного забезпечення для їх створення, редагування, конвертування.

Розглянемо зокрема сферу електронних зображень. Існує багато потужних графічних редакторі, та іноді набагато важливіша простота певної групи операцій ніж їх різноманіття.

Найбільш поширеними форматами зображень є: .gif, .png, .jpg, .bmp. Кожний з них має свої переваги і недоліки, певні спеціалізації.

**Розділ 1. Теоретичні положення**

## 1.1. Мова програмування C#

C# — об’єктно-орієнтована мова програмування з безпечною системою типізації для платформи .NET . Розроблена Андерсом Гейлсбергом, Скотом Вілтамутом  та Пітером Гольде  під егідою  Microsoft Research (при фірмі  Microsoft).

Синтаксис C# близький до С++ і Java. Мова має строгу статичну типізацію, підтримує поліморфізм, перевантаження операторів, вказівники на функції- члени класів, атрибути, події, властивості, винятки, коментарі у форматі XML. Перейнявши багато що від своїх попередників — мов С++, Delphi, Модула і Smalltalk — С#, спираючись на практику їхньо-го використання, виключає деякі моделі, що зарекомендували себе як проблематичні при розробці програмних систем, наприклад множинне спадкування класів (на відміну від C++).

Крім того, творці С# хотіли приховати від розробника якомога більше незначних технічних деталей, включаючи операції по упаковці/розпаковуванню типів, ініціалізації змінних і збірці сміття. Завдяки цьому програміст, що пише на C#, може краще концентруватися на змістовній частині завдання. В процесі рішення цієї задачі проектувальники C# намагалися врахувати уроки реалізації Visual Basic'а, який достатньо успішний в утаєнні деталей реалізації, але недостатньо ефективний для написання крупних промислових систем: творці C# декларують, що нова мова володіє потужністю С++ і в той же час простотою Visual Basic'а.

Ще одна перевага створення нової мови програмування в порівнянні з розширенням тих, що існують полягає в тому, що при створенні нової мови немає необхідності піклуватися про проблеми зворотної сумісності, які зазвичай помітно утрудняють виправлення застарілих проблем і навіть внесення нових властивостей в стандарт мови (докладний опис труднощів, що виникають при розширенні старої мови програмування, можна прочитати в книзі Б. Страуструпа "Дизайн і еволюція мови С++", М.: ДМК, 2000).

Таким чином, C# є новою мовою програмування, орієнтованою на розробку для платформи .NET і придатний як для швидкого прототіпірованія застосувань, так і для розробки великомасштабних застосувань.

## 1.2Технологія Windows Presentation Foundation та допоміжні бібліотеки

WPF (Windows Presentation Foundation) -. Один з компонентів ядра NET Framework, який дозволяє розробникам створювати багаті, різноманітні ОС Windows програми. WPF 4 містить важливі поліпшення в продуктивності, елементів управління, XAML, тексту, графіки, інтеграції в Windows 7 (мультитач, інтеграція в панель завдань, і т.п.), основних принципів роботи, розгортання .

WPF це перше реальне оновлення технологічного середовища призначеного для користувача інтерфейсу з часу випуску Windows 95. Воно включає нове ядро, яке повинне замінити GDI і GDI+, використовувані на нинішній Windows-платформі. WPF є високорівневим об'єктно-орієнтованим функціональним шаром (англ. framework), що дозволяє створювати двовимірні та тривимірні інтерфейси.

В основі WPF лежить використання мови XAML (Extensible Application Markup Language). XAML є мовою розмітки, яку використовують для створення екземплярів об'єктів .NET. Хоча мова XAML — це технологія, що може бути застосовна до багатьох різних предметних областей, її головне призначення — конструювання інтерфейсів користувачів WPF

Перевагою технології WPF полягає в тому, що Windows Forms не підтримає повність функцію прозорості, а лише певним чином маскує. Для кількості елементів більше 2 виникають дефекти. Технологія WPF повністю реалізую функцію прозорості за допомогою властивості Opacity і змішення через альфа-канал. Також суттєво відрізняються схема кольорів

При розробці інтерфейсу користувача відбувається упорядкування елементів управління по розташуванню і розміром для формування структури . Основною вимогою будь-якої структури є адаптованість до змін розміру вікна і параметрам відображення . Щоб не довелося створювати код для адаптації структури в таких обставинах , WPF надає першокласну розширювану систему структури .

Основою системи структури є відносне позиціонування , що збільшує здатність адаптації до змінюваному вікна і умовам відображення . Крім того , система структури управляє узгодженням між елементами управління для визначення структури . Таке узгодження складається з двох етапів: спочатку елемент управління повідомляє батьківського елементу , яке розташування і розмір потрібно; потім батьківський елемент повідомляє елементу управління , який простір він може зайняти.

WPF полегшує розміщення елементів за допомогою панель компонування: StackPanel, WrapPanel, DockPanel, Grid, UniformGrid, Canvas . Особливо використовуються StackPanel ( Розміщає елементи в горизонтальний або вертикальний стек) та Canvas (Дозволяє елементам позиціонуватися абсолютно — за фіксованими координатами).

.NET дозволяє вікну «володіти» іншими вікнами. Вікна, що мають вікно-власника, зручно застосовувати для плаваючих вікон панелей інструментів і вікон команд. Одним із прикладів такого вікна є вікно Find and Replace (Знайти й замінити) в Microsoft Word. Коли вікно-власник згортається, вікно, яким воно володіє, теж автоматично згортається. Коли вікно, що має власника, перекриває вікно, яке ним володіє, воно завжди відображається зверху.

Для підтримки володіння вікна клас Window пропонує дві властивості: властивість Owner і властивість OwnedWindows. Властивість Owner являє собою посилання, що вказує на вікно, що володіє поточним вікном (якщо таке є), а властивість OwnedWindows — колекцію всіх вікон, якими володіє поточне вікно (знову ж, якщо такі є).

Також корисними є такі бібліотеки:

* WpfAnimatedGif – надає інструменти для роботи з анімованими зображеннями (gif).
* Xceed, ColorFont – забезпечують різноманітні діалогові компоненти, зокрема ColorPicker та ColorFontDialog для вибору шрифтів та кольорів.

**Розділ 2. Привітальна листівка**

## 2.1 Постановка задачі

Розробити програму для складання привітальної листівки з нагоди свята Нового року, яка повинна містити текст(різної форми і кольору) і рухомі картинки (гірлянда, яка мигає;салют, свічка, яка горить і т.п.). Перегляд підготовлених варіантів має бути в окремому вікні для кожного варіанту.

## 2.2 Підходи щодо вирішення поставленої задачі

Для виконання даної задачі потрібно створити програму з зручним і

зрозумілим для користувача інтерфейсом. Основними сутностями в нас є

картинка та текст, які налаштовуються відповідними елементами керування.

Програма повинна бути зручна в роботі.

З точки зору користувача дана програма повинна забезпечити зручне редагування даних. Інтерфейс проекту повинен відповідати сучасним вимогам. Програма повинна бути стійкою до помилок і будь-які дії користувача, навіть некоректні, не можуть привести до втрати працездатності програми.

З точки зору програміста дана програма повинна собою являти узгоджену ієрархію класів, методів, які б забезпечували взаємодію з користувачем. Ця програма створена з метою швидкої і простої розробки вітальної листівки , збереження її у зручному форматі для подальшого надсилання різними засобами.

Застосунок має підтримувати як додавання власних картинок різних форматів так і пропонувати невелику ієрархію готових зображень(рухомих і статичних). Програмі необхідно передбачає функцію створення декількох листівок одночасно в окремих вкладках.

Окремою темою можна винести панель керування, адже тут є простір для фантазії і винахідливості програміста в тому плані що можна різну використати різну систему керування: залежність від комп’ютерної мишкі, управляння клавішами, гарячі клавіші. Потрібно продумати елементи управління з точки зору “дружнього інтерфейсу” та оптимальної швидкодії.

# Розділ 3. Структура програми

## 3.1 Об'єктно орієнтований принцип створення програми

Дуже часто термін *клас* вживають у зв’язку з розглядом питань об’єктно-орієнтованого програмування (ООП). Пригадаємо собі, що таке ООП. Це особливий спосіб програмного моделювання реального світу: об’єкти програми (екземпляри класів) поєднують у собі здатність зберігати стан та демонструвати поведінку, тому є найбільш адекватними моделями реальних сутностей. Це особливий спосіб структурування програм за допомогою побудови та використання ієрархії об’єктних типів – класів. Це особливий спосіб обмеження доступу та приховування даних за допомогою інкапсуляції їх в класі та «обгортання» оболонкою методів доступу. Це ще й особли­вий спосіб повторного використання коду та удосконалення, розвитку функціональності класів.

Звичайно, всі ці аспекти враховано розробниками С++, є відповідні інструменти мови для реалізації тих чи інших потреб ООП. Та все ж *клас для С++* – це, в першу чергу, досконалий *інструмент оголошення типів користувача*. Типів, що якнайкраще задовольняють особливі потреби користувача, і є такими ж зручними у використанні, як і вбудовані типи: одні і ті ж правила для об’єкта та для цілого числа (наприклад). А що ж таке ТИП? Тип – це, по-перше, певна множина значень і набір пов’язаних з ними операцій, по-друге, реалізація типу визначає певні правила кодування значень цієї множини (зокрема, довжину коду – розмір необхідної пам’яті). Таку інформацію про вбудовані типи зберігає компілятор. Множину значень, набір операцій для нового типу описує програміст за допомогою визначення класу. Компілятор «навчений» будувати для нього спосіб кодування та долучати новий тип до компанії вже визначених.

Саме з такої точки зору – клас як інструмент створення нового типу – варто знайомитися з класами С++. Під час знайомства доведеться звернути увагу на такі важливі моменти: визначення типу (множини значень, операцій, поведінки тощо); оголошення та визначення об’єктів, час життя об’єкта (зокрема, з огляду на клас пам’яті); присвоєння об’єктові, автоматичне перетворення типу.

Розмова буде довгою і не завжди простою, тому спочатку узгодимо термінологію. На жаль, автори вживають у публікаціях, присвячених різним мовам програмування, різні терміни для тих самих понять.

*Клас* – об’єктний тип; тип, оголошений користувачем, що поєднує дані та методи опра­цювання, відкритий для наслідування, для створення на його базі нових типів, є шаблоном для створення екземплярів класу, є «фабрикою» екземплярів.

*Об’єкт* – екземпляр класу, що має власний набір даних і поділяє методи з іншими екземплярами, змінна об’єктного типу, модель реальної сутності.

*Повідомлення* – особливий спосіб «звертання» до об’єкта. Опрацюванням повідомлення є відшукання і виконання відповідного методу. У С++ часто говорять про *виклик член-функції*.

*Поле даних (екземпляра)* – змінна, оголошена в межах класу. В С++ – *член класу, елемент даних*. Поля даних оголошують в класі, але існують вони в екземплярах: кожен об’єкт отримує власний набір полів і використовує їх для зберігання інформації про свій стан.

*Змінна класу – статичний член класу* в С++. Особливе поле даних, що існує в самому класі, поза будь-яким екземпляром, може зберігати інформацію, спільну для всіх екземплярів класу.

*Метод (екземпляра) – член-функція класу* в С++, *елемент-функція*. Функція, оголошена в межах класу. Описує поведінку об’єкта. Надає програмний інтерфейс для доступу до даних об’єкта. Зберігається в словнику методів класу, використовується спільно всіма екземплярами класу. За допомогою методів об’єкт опрацьовує отримані повідомлення. Таким чином, доступ до методів – через надсилання повідомлень об’єктам.

*Метод класу – статична член-функція* в С++. Метод, виклик якого виконують виключно через ім’я класу. Методи класу працюють зі змінними класу і не можуть звертатися до полів екземпляра.

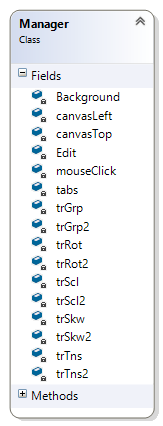
*Конструктор* – метод класу, відповідальний за створення та ініціалізацію екземплярів.

*Деструктор* – метод класу, відповідальний за знищення екземплярів.

Будь-який тип, особливо клас, є абстракцією деякої сутності.

## 3.2 Класи і методи, які використані у програмі

При створені програми “ Розробка привітальної листівки (Christmas Postcard)” було використано такий клас:

 *Перегляд класів*

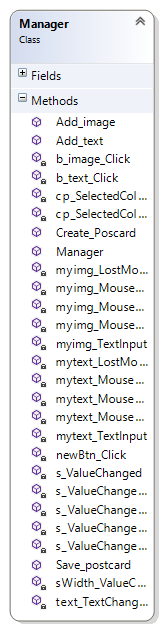
Manager

Клас Manager призначений для керування об’єктами. Ніхто немає доступу до елементів окрім нього .

Принцип поліморфізму приховує усі деталі реалізації керування. Manager містить усі необхідні об’єкти для зміни даних(картинок, тексту). Конструктор класу приймає три параметра: public Manager(TabControl tabs, StackPanel Background, StackPanel Edit). Один з них є елементом управління вкладками а інші два надають інтерфейс для редагування елементів.

Відкрити методами класу є лише чотири:

Create\_Poscard(),Add\_text(),Save\_postcard(),Add\_image().

Відповідно вони реалізують створення нової вкладки, тексту, картинки і збереження в файл.

В більшості випадків функціонал програми забезпечується через опрацювання подій, що дозволяє програмі бути доволі простою в користуванні.

private Point mouseClick;

private double canvasLeft;

private double canvasTop;

Три приватних поля які забезпечують рух картинок та тексту. Для зберігання зображень використовуємо клас Image, а тексту – TextBlock. Всі користувацькі елементи та елементи управління створюються динамічно.

private TransformGroup trGrp;

private TransformGroup trGrp2;

private SkewTransform trSkw;

private RotateTransform trRot;

private TranslateTransform trTns;

private ScaleTransform trScl;

private SkewTransform trSkw2;

private RotateTransform trRot2;

private TranslateTransform trTns2;

private ScaleTransform trScl2;

Ці класи забезпечують трансформацію картинок. Елемент потрібно прив’язати до TransformGroup яка певний час буде відповідати за зміни. Кожен елемент має своє індивідуальне ім’я яке використовується для ідентифікації елементів.

text.Name = "textBox" + canva.Children.Count.ToString();

Усі користувацькі елементи зберігаються в Canvas.

Пошук елементів відбувається за допомогою конструкції:

Canvas canva = ((tabs.SelectedItem as TabItem).Content as StackPanel).Children[0] as Canvas;

foreach (Image a in canva.Children.OfType<Image>())

{

if (a.Name == (sender as Slider).Name)

{

Edit.Children.Clear();

canva.Children.Remove(a);

break; }

}

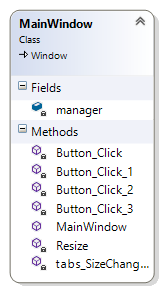
В головній програмі міститься один екземпляр класу Manager і опрацьвується декілька подій.

Дизайн програми розроблявся за допомогою XAML, зразо коду:

<Button Content="Create Postcard" HorizontalAlignment="Right" Margin="0,10,17,0" VerticalAlignment="Top" Width="87" Click="Button\_Click" RenderTransformOrigin="0.5,0.5"/>

<Button Content="Add image" HorizontalAlignment="Right" Margin="0,35,17,0" VerticalAlignment="Top" Width="87" Click="Button\_Click\_1"/>

<Button Content="Add text" HorizontalAlignment="Right" Margin="0,60,16,0" VerticalAlignment="Top" Width="88" Click="Button\_Click\_3"/>

 <StackPanel x:Name="Edit" HorizontalAlignment="Right" Height="319" Margin="0,169,10,-22" VerticalAlignment="Top" Width="94">

***3.2 Візуальне офрмлення проекту***

Вимоги:

Інтерфейс проекту має відповідати сучасним вимогам – повинен містити: меню, спливаючі вікна, панель інструментів, рядок стану, діалогові вікна.

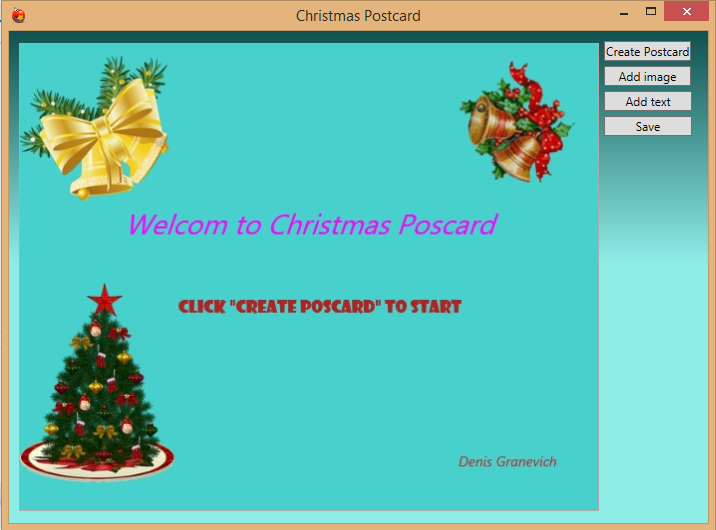
Запустивши програму користувач бачить привітальне зображення з простими інструкціями для початку та зразком листівки розроблений даною програмою.

рис.1

Після запуску програми користувач бачить перед собою чотири елементи управління(кнопки) за допомогою яких відбувається зовнішній зв’язок програми. “Create Postcard” дозволяє створити нові вкладки з листівками. Після утворення першої вкладки з’являється елемент управління кольором фону листівки(рис.2).

рис.2

Вибирається фон за замовчуванням і надається можливість змінити його.

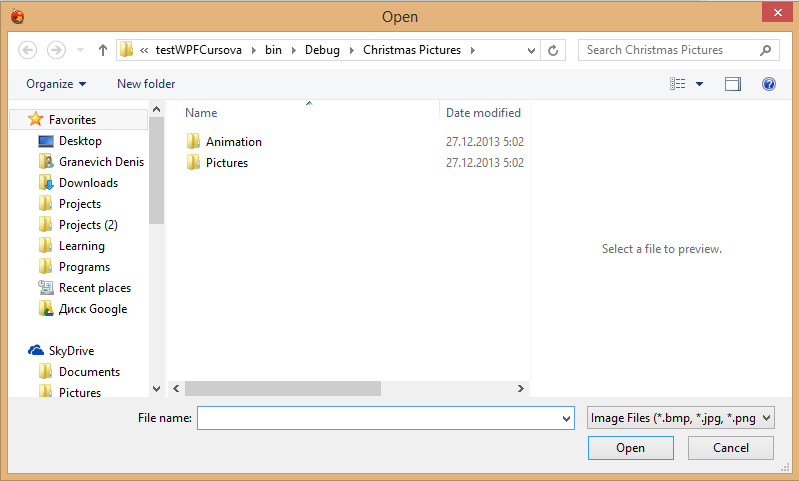
Додати картинки можна вибравши через dialogBox вказавши свій шлях, або скористуватись готовою невеликою ієрархією пропонованих зображень(рис.3).

рис.3

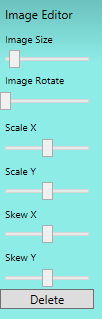
При виділенні картинки відкривається меню редагувань “Image Editor”(рис.4) з відповідними налаштуваннями розміру, повороту, і різноманітних трансформацій. Відзначимо що картинки можна пересовувати за допомогою миші. Текст має подібну структуру але відрізняється в редакторі, зокрема з’являється поле вибору кольору і шрифту(рис.5). В будь-якій момент можна видалити окремий елемент. “Save” відповідає за збереження листівки. Користувачу надається вибрати зручний для нього формат зображення і також вказати шлях зберігання(рис.6).

рис.4

рис.5

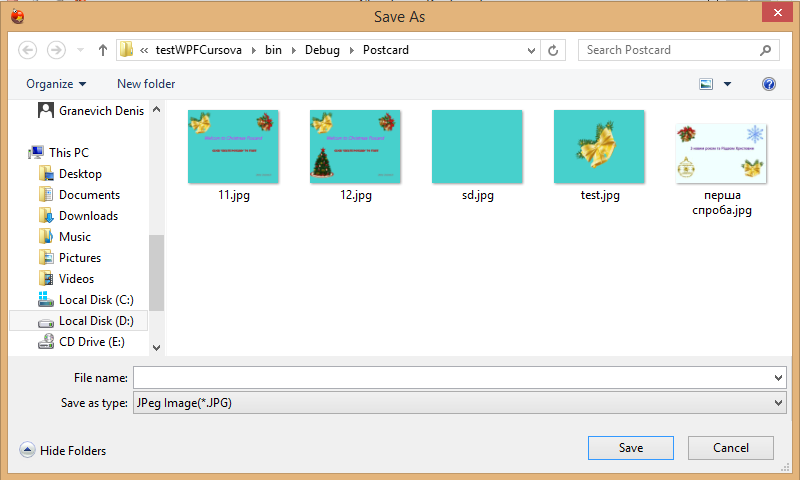


рис.6

**Висновок**

Створена програма «Розробка привітальної листівки», відповідає поставленим задачам та вимогам. Під час проектування найбільша увага приділялася простоті інтерфейсу і візуальному оформленню, одне з головних завдань було створення програми, що відповідає сучасному інтерфейсу і є легкою в користуванні. З написанням проекту, я дізнався багато нового про мову програмування С# і особливо з технологією WPF, що буде корисним у подальшому навчанні і для здобуття професії.

**Список використаної літератури**

* 1. Прата Стивен Язык программирования С++. Лекции и упражнения.
  2. Брюс Эккель - Философия C++. Введение в стандартный С++

3. Роберт Лафоре "Объектно-ориентированное программирование в С++ "

4. Максим Динман. С++. Освой на примерах

5. [http://habrahabr.ru](http://habrahabr.ru/)

6. [http://uk.wikipedia.org](http://uk.wikipedia.org/)