|  |
| --- |
|  |
| (повна назва вищого навчального закладу) |
|  |
| (повна назва кафедри) |

КУРСОВИЙ ПРОЕКТ

|  |  |
| --- | --- |
| з курсу: | Прикладна математика |
| на тему: | Розробка програми на С# з використанням технології WinForms |
|  | Додаток “Archiever” |
|  |  |
|  |  |
|  |  |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | Студента |  | | | курсу | |  | групи |
|  | напряму підготовки | | | |  | | | |
|  | спеціальності | | Прикладна математика | | | | | |
|  |  | | | | | | | |
|  | (Прізвище і ініціали) | | | | | | | |
|  | Керівник |  | | | | | | |
|  |  | | | | | | | |
|  | (посада, звання, прізвище та ініціали) | | | | | | | |
|  | Національна шкала | | | |  | | | |
|  | Кількість балів: | | | 100 | | Оцінка: ECTS | |  |
|  |  | | | | | | | |

РЕФЕРАТ

Записка пояснювальна до курсової роботи: 6 розділи, 20 cт., 5 рис., 4 використаних літературних джерела.

Об'єкт дослідження – програма «Archiever».

Мета роботи – розробка програми, яка буде архівувати файли.

Метод дослідження – вивчення літератури, складання і налагодження текстової програми на комп'ютері.

Фінальний продукт – програма – зручна та корисна у використанні для середньостатистичної людини.

Програма складена на мові С# в середовищі Visual Studio 2017 з використанням технології Windows Forms.

Зміст

[КУРСОВИЙ ПРОЕКТ 1](#_Toc515834341)

[ВСТУП 4](#_Toc515834342)

[ПОСТАНОВКА ЗАДАЧІ ТА СФЕРА ВИКОРИСТАННЯ 6](#_Toc515834343)

[Алгоритм розв’язання задачі та UML-діаграми 8](#_Toc515834344)

[Текст програми 10](#_Toc515834345)

[Описання форм, які є в проекті 13](#_Toc515834346)

[Тестовий приклад використання програми 15](#_Toc515834347)

[Інструкція користувача 18](#_Toc515834348)

[Висновки 19](#_Toc515834349)

[Список використаних джерел 20](#_Toc515834350)

# ВСТУП

В сучасному світі нас оточує величезна кількість інформації, яку ми сприймаємо, аналізуємо та передаємо один одному. З моменту зростання кількості даних до таких неймовірних обсягів почали використовуватися різні алгоритми стиснення (архівування).

Під архівацією розуміють стиснення даних. Стиснення даних — це процедура перекодування даних, яка проводиться з метою зменшення їх об'єму, розміру, обсягу. Стиснення буває без втрат (коли можливе відновлення вихідних даних без спотворень) або з втратами (відновлення можливе з незначними спотвореннями). Стиснення без втрат використовується при обробці та збереженні комп'ютерних програм і даних. Стиснення з втратами зазвичай застосовується для зменшення об'єму звукової, фото, та відеоінформації. І, як показує практика, стиснення з втратами для такого роду інформації є набагато дієвішим. Стиснення базується на видалені надлишку інформації, яка міститься у вихідних даних. Прикладом надлишку є повторення в тексті фрагментів (наприклад, слів природної або машинної мови). Подібний надлишок зазвичай усувається заміною повторюваних послідовностей більш коротким значенням (кодом). Інший вид надлишку пов'язаний з тим, що деякі значення в даних зустрічаються частіше інших. Тому, можна замінити дані, що часто зустрічаються більш короткими кодами, а ті, що рідше зустрічаються - більш довгими (ймовірнісне стиснення). Стиснення даних, які не мають властивості надлишку (наприклад випадковий сигнал чи шум), неможливе, зазвичай неможливо стиснути і зашифровану інформацію.

Архів — файл, що містить у собі один або декілька файлів та метадані. Файли можуть бути як стиснені (без втрат), так і мати початковий розмір та структуру, але першочергове завдання архіву тримати у собі саме стиснуті файли. Метедані можуть містити інформацію про початковий розмір файлів, інформацію про формат файлів, структуру директорій, коментарі до файлів, інформацію для відновлення архіву і т.д. Архіви файлів створюються за допомогою спеціалізованих програм — архіваторів, які можуть бути як окремими програмами, так і частиною інших програм. Далі буде розглянуто декілька з цих програм. Різновиди архівів поділяються на ті, які складається з одного або декількох файлів і метаданих та на ті, що містять рівно один стиснутий файл. Деякі архіватори та формати архівів об'єднують ці дві функції в довільному порядку — наприклад, 7-Zip, ARJ, ZIP. У таких випадках, якщо стиснення проводиться після об'єднання, архів називається «безперервним». Це дозволяє зменшити розмір отриманого архіву, але ускладнює відновлення при пошкодженні даних. Сам архів може складатися з декількох файлів для полегшення зберігання і перенесення великої кількості даних при обмеження на розмір однієї частини — наприклад, носія даних, або повідомлення e-mail. Такий архів називається «багатотомним».

# ПОСТАНОВКА ЗАДАЧІ ТА СФЕРА ВИКОРИСТАННЯ

На даний момент світу вже відомий певний набір найбільш відомих архіваторів. Ця курсова робота буде зорієнтована на найбільш відомий з них – ZIP. Формат ZIP був спочатку створений Філом Кацом, засновником компанії PKWARE, у відповідь на правове переслідування компанією Software Enhancement Associates (SEA), що захищала свій винахід — формат архівування ARC. Кац скопіював ARC і змінив частину коду, написаного на Сі, оптимізованим кодом на ассемблері, тим самим зробивши програму значно швидшою. Спочатку SEA спробувала ліцензувати архіватор PKARC, зроблений Кацем, але той відмовився. Тоді вони подали позов за порушення прав правовласника й виграли процес. Кац продовжив розробку і незабаром представив власний формат архівації файлів PKZIP, який набагато ефективніше стискав дані, ніж ARC. Після випуску PKZIP багато користувачів перейшли на його бік через кращий алгоритм стискання, що приносив вигоду у часі та у розмірі, а також тому що Кац зумів успішно переконати, що він «гарний хлопець», якого «використовувала» погана корпорація. На даний момент існує безліч алгоритмів компресії, які виграють у ZIP і у швидкості, і у компресії, і у кількості додаткових можливостей. Незважаючи на це, ZIP надалі є популярним методом стискання даних, резервного копіювання й обміну даними.

На даний час ZIP використовує такі методи компресії: Shrunk, Reduced (методи 1-4), Imploded, Tokenizing, Deflated, Deflate64, BZIP2, LZMA (EFS), WavPack, PPMd.

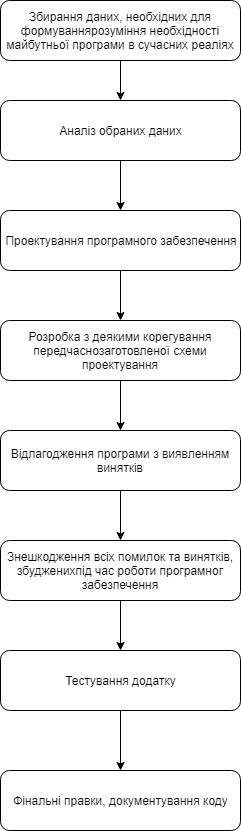
Мій власний архіватор буде використовувати вбудовану бібліотеку System.IO.Compression.ZipFile. Доцільність її використання полягає в тому, що в ній реалізовані найбільш швидкі, надійні та зручні для використання алгоритми стиснення даних.

Також при розробці програми не варто використовувати заскладний інтерфейс. Все має бути досконало та просто. Архіватор не має бути перевантажений додатковим функціоналом, але мати найкорисніші та найнеобхідніші функції. Тому, головною перевагою мого додатку буде його легкість та простота (зручність) у використанні. Такий підхід пояснюється, в першу чергу, аудиторією – простими людьми, які не є айті-спеціалістами, та які не хочуть витрачати свій дорогоцінний час для того, щоб розібратися в перевантажному функціоналі громіздкого архіватора.

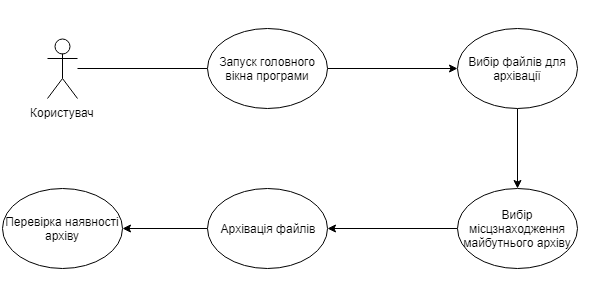
Отже, інтерфейс має бути максимально простим та інтуїтивно зрозумілим.

Вимоги до самої програми – по-перше, архівувати файли. По-друге, мати надійну стійкість до різного роду помилок та виключень, які можуть виникати по ходу роботи програми.

# Алгоритм розв’язання задачі та UML-діаграми



Діаграма прецедентів:



# Текст програми

Середовище розробки: Microsoft Visual Studio 2017

Мова програмування: C# 4.0

Технологія: ASP.NET

Версія .NET Framework: 4.0

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.IO;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using System.IO.Compression;

namespace Archiever

{

public partial class MainForm : Form

{

List<string> selectedFiles;

public MainForm()

{

InitializeComponent();

selectedFiles = new List<string>();

}

private void BUT\_OpenFile\_Click(object sender, EventArgs e)

{

OpenFileDialog choofdlog = new OpenFileDialog();

choofdlog.Filter = "All Files (\*.\*)|\*.\*";

choofdlog.FilterIndex = 1;

choofdlog.Multiselect = true;

if (choofdlog.ShowDialog() == DialogResult.OK)

selectedFiles.AddRange(choofdlog.FileNames);

if (selectedFiles.Count > 0)

{

foreach (var item in this.selectedFiles)

{

if (!LB\_SelectedItems.Items.Contains(item))

{

this.LB\_SelectedItems.Items.Add(item);

}

}

}

}

private void BUT\_Archive\_Click(object sender, EventArgs e)

{

string defaultPathOfNewFolder = @"c:\example\start";

string defaultPathOfZippedFile = @"c:\example\result.zip";

using (var fbd = new FolderBrowserDialog())

{

DialogResult result = fbd.ShowDialog();

if (result == DialogResult.OK && !string.IsNullOrEmpty(fbd.SelectedPath))

{

defaultPathOfNewFolder = fbd.SelectedPath;

}

defaultPathOfZippedFile = Path.Combine(defaultPathOfNewFolder, "result.zip");

defaultPathOfNewFolder = Path.Combine(defaultPathOfNewFolder, "start");

}

try

{

Directory.CreateDirectory(defaultPathOfNewFolder);

foreach (var file in this.selectedFiles)

{

File.Copy(file, Path.Combine(defaultPathOfNewFolder, file.Split('\\').Last()));

}

ZipFile.CreateFromDirectory(defaultPathOfNewFolder, defaultPathOfZippedFile);

MessageBox.Show("Zip archieve was created successfuly");

this.LB\_SelectedItems.Items.Clear();

}

catch (Exception ex)

{

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

}

}

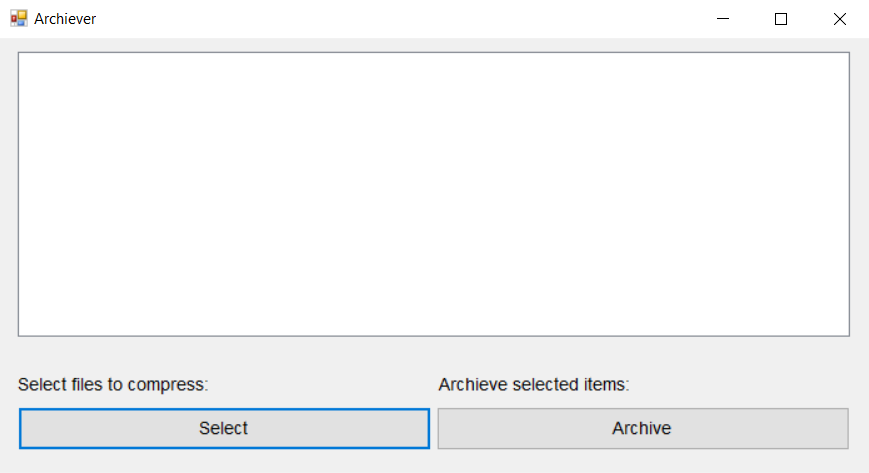
}

}

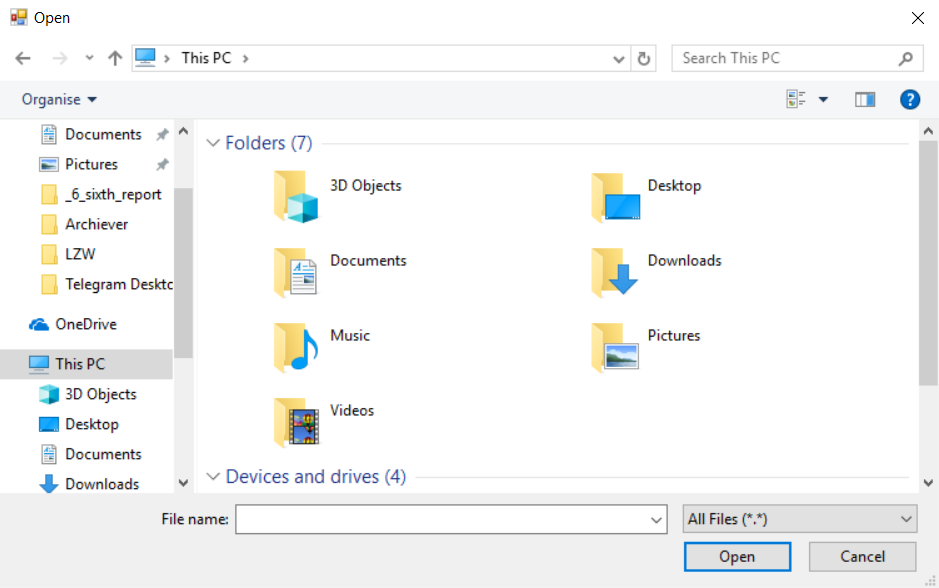
Під час написання коду програми за мету ставилася максимальна зрозумілість та читабельність коду. За непотрібне вважалося надмірне нагромадження коду, створення незрозумілих класів та додаткових (непотрібних) форм та функцій.

# Описання форм, які є в проекті

1. Головна форма.

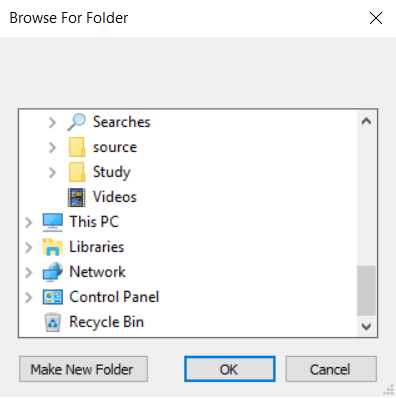
Містить інформаційну панель та дві кнопки. Вся необхідна користувачу інформація міститься на самій формі.

1. Допоміжна форма (вибір файлів для архівації).

Ця форма є звичайним (зручним у використанні) файловим менеджером. Вона дозволить користувачу обрати необхідні для стиснення файли.

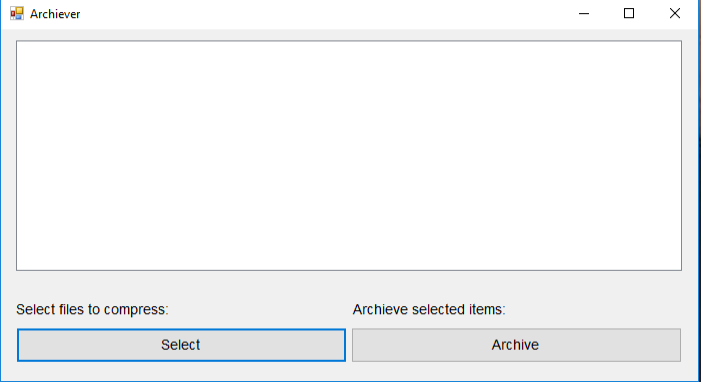
1. Допоміжна форма (вибір директорії архівації).

Ця форма дозволяє користувачу обрати зручне для нього розташування майбутнього архіву з завчасно обраними для стиснення файлами.

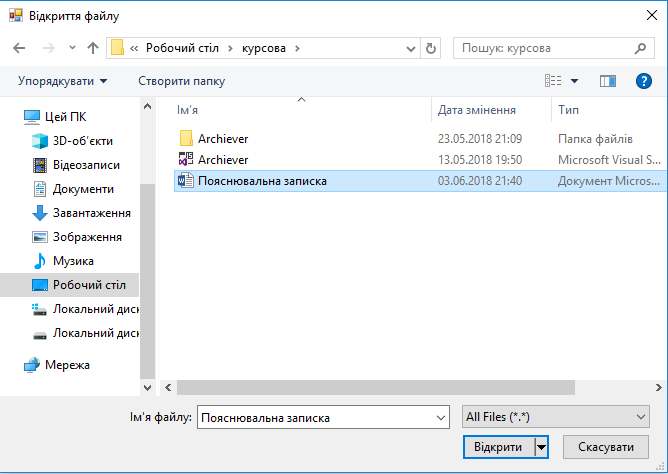


# Тестовий приклад використання програми

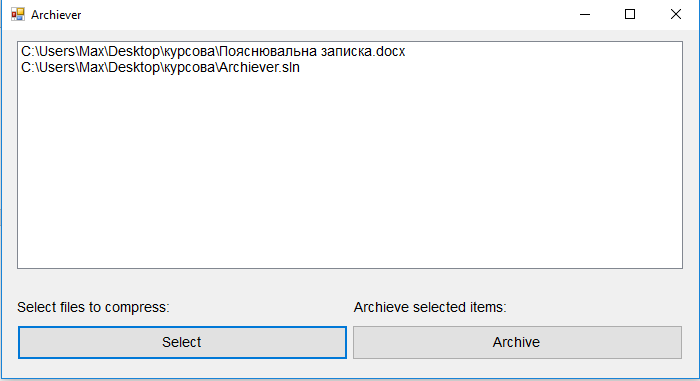
1. Запускаємо головне вікно програми.



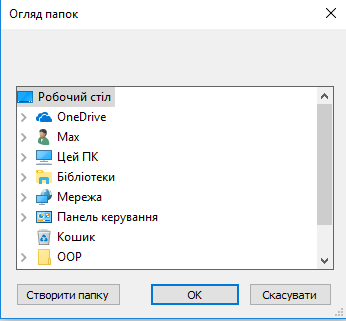
1. Натискаємо кнопку “Select”.



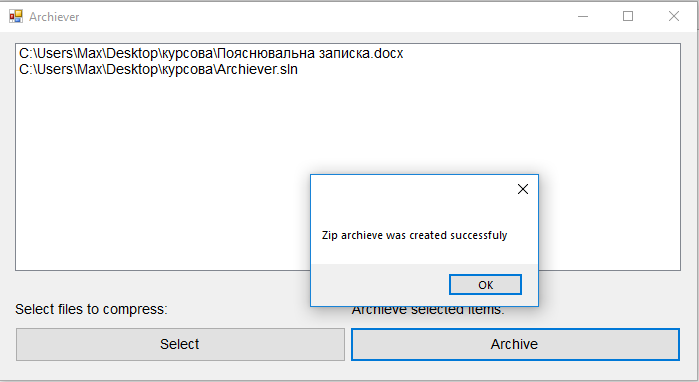
1. Обираємо файли для архівації.



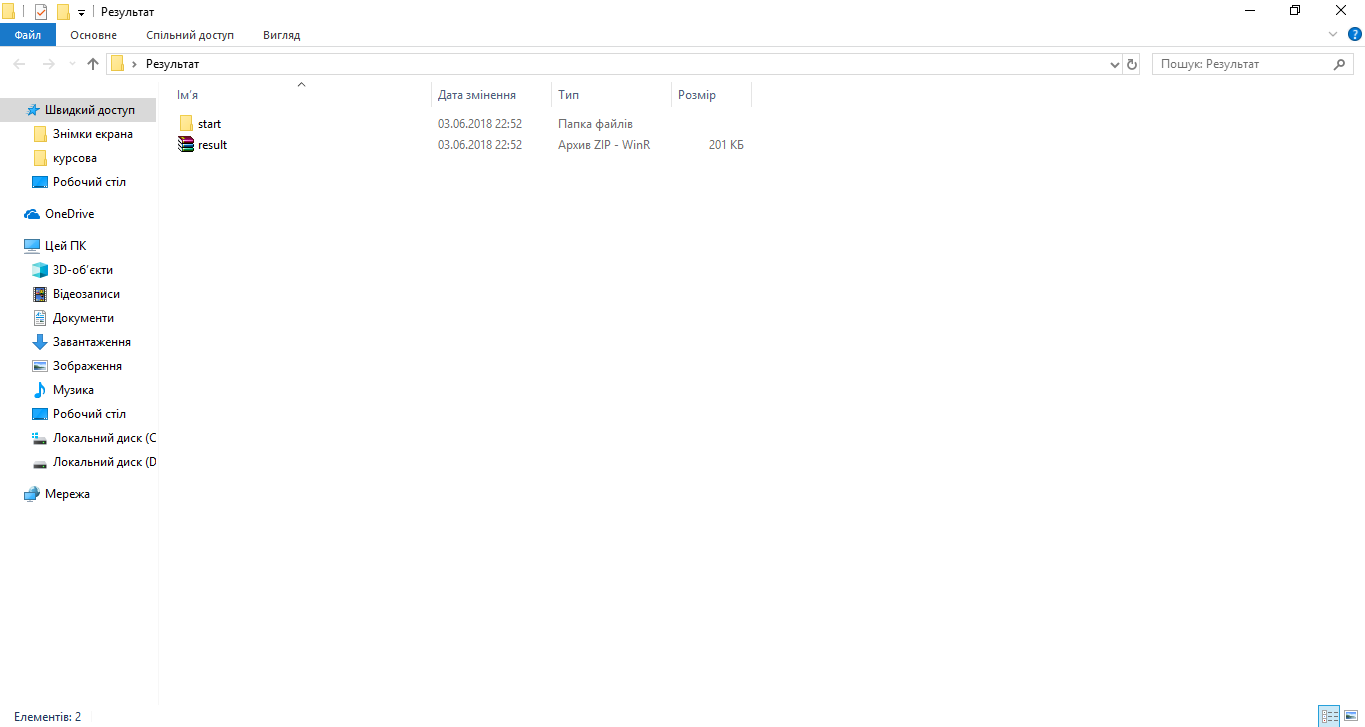
1. Натискаємо кнопку “Archive”.



1. Обираємо місцезнаходження майбутнього архіву.



1. Перевіряємо наявність архіву в обраній аудиторії.



# Інструкція користувачу

Програма відрізняється своїм простим інтерфейсом, тому вона не потребує великої інструкції.

Основні правила користування програмою:

1. Використовуйте кнопки “Select” та ”Archive” для управління архівом, який ви створюєте.
2. Поле з інформацією по обраних файлах очищається автоматично.
3. Фінальний архів буде збережений у вказану вами папку.
4. Для виходу з програми натисніть хрестик в правому верхньому кутку екрана.

# Висновки

Отже, виконавши цю роботу, я впевнився у тому, що архіви зараз є дуже широко використовуваними. Також я зміг розібратись в нюансах архівації. На мою думку, сама ідея архівації є дуже актуальною та геніально простою. Принаймні, у своїй реалізації в мові C# та на платформі .NET вона є досить простою для розуміння та не громіздкою для реалізації.

Стиснення даних зараз дуже широко використовується у будь-якій айті-сфері. Від програмування атомних реакторів до програмування пральних машин. Дані є всюди, так само всюди є потреба в їх стисненні, та, як результат, в зменшенні об’єму пам’яті, який ці дані займають.

# Список використаних джерел

1. Г. Шилдт. C#: полное руководство. Москва: BHV Санкт-Петербург, 2008. 980 c
2. Бек Л. Введение в системное программирование. М.: Мир, 1998. 440 с.
3. Джозеф Албахари, Бен Албахари. C# 6.0 in a Nutshell: The Definitive Reference. Вильямс, 2017. 1040 с.
4. <https://msdn.microsoft.com/ru-ru/>
5. <http://www.programmer-lib.ru/csharp.php>
6. https://tproger.ru/tag/c-sharp/