Міністерство освіти і науки України
Черкаський національний університет імені Богдана Хмельницького
Факультет обчислювальної техніки, інтелектуальних та управляючих систем
Кафедра програмного забезпечення автоматизованих систем

КУРСОВА РОБОТА

з дисципліни «_Програмування та алгоритмічні мови_____»

на тему: «_Перестановочний шифр_____»

Студента(ки) 1 курсу КС-181 групи спеціальності
__121 Інженерія програмного забезпечення_____
(код Назва)
Пахаренко І.В._____
(прізвище та ініціали)

Керівник: Сердюченко В. С._____
(прізвище та ініціали)

Оцінка за шкалою:

_____
(національною, кількість балів, ECTS)

Члени комісії: _____ _____
(підпис) (прізвище та ініціали)

_____ _____
(підпис) (прізвище та ініціали)

_____ _____
(підпис) (прізвище та ініціали)

Черкаси – 2019

ЗМІСТ

ВСТУП

Потреба якимось чином захищати інформацію з’явилася у людей приблизно тоді ж, коли вони навчилися нею обмінюватися одне з одним. Поступово від примітивніших фізичних методів зберігання інформації людство переходить до її зберігання у цифровому форматі.

Зараз, після появи Інтернету, коли домашні комп’ютери та смартфони щодня передають на сервери і завантажують з них гігабайти різноманітної, зокрема конфіденційної, інформації, питання захисту інформації під час обміну і протягом зберігання постає особливо нагально. Звичайно, без якісного захисту на небезпеку наражаються не лише прості користувачі, ризикуючи дати зловмисникам доступ до листування чи навіть банківського аккаунту. Компанії можуть зазнати катастрофічних збитків чи навіть збанкрутувати, якщо допусять витік важливої інформації за межі своїх офісів. Саме тому інформаційній безпеці приділяють дуже багато уваги навіть такі цифрові гіганти, як Google, Apple, Microsoft.

Здавна для захисту даних, зокрема цифрових, використовується криптографія — наука про методи перетворення (шифрування) інформації з метою її захисту від несанкціонованих користувачів. На перший погляд може здатися, що всі доступні людям алгоритми для шифрування і відповідно злому шифрів уже створені і досягли досконалості, тому все, що залишається — намагатися комбінувати алгоритми в спробі відтягти момент злому. Проте з появою комп’ютерів можливості обробки інформації зросли на багато порядків, і тепер ми можемо використовувати значно більш просунуті алгоритми для захисту важливої інформації. Звичайно, комп’ютери так само дозволяють здійснювати злом захисту значно ефективніше. Отже, прямо зараз відбувається «війна озброєнь» у світі алгоритмів шифрування. Всі алгоритми, застосовні для використання без комп’ютера, стали неефективними. Зате алгоритмам, що передбачають використання комп’ютера, ще є куди розвиватися. Ледь не щороку створюється

новий метод шифрування, а ще за рік він стає повністю беззбройним перед хакерами і з’являється потреба в новому, ще кращому алгоритмі.

Мета цієї курсової роботи — створити додаток, що дозволяє зашифровувати і розшифровувати тестові повідомлення використовуючи перестановочний шифр

Задачі:

1. Оглянути інформаційні джерела.
2. Реалізувати алгоритм шифрування перестановками.
3. Провести тестування алгоритму та інтерфейсу програми.

РОЗДІЛ 1. ОГЛЯД ТЕХНОЛОГІЙ ДЛЯ РЕАЛІЗАЦІЇ ІНТЕРФЕЙСУ ТА ПЕРЕСТАНОВОЧНОГО АЛГОРИТМУ

1.1 Вибір мови програмування та середовища розробки

Перш за все, необхідно визначити мову програмування, що буде використовуватися для написання програми. Для мене вибір постав між двома мовами: C# та Java.

Обидві мови C-подібні (тому мають дуже схожий синтаксис), створені приблизно у той же час, компілюються не у машинний код, а код для віртуальних машин. Тому і їхня продуктивність, грубо кажучи, приблизно однакова. Однак вони мають одну вирішальну відмінність: Java вже доволі довго не використовується для створення додатків для Windows, на відміну від C#. Тому для C# буде більше готових рішень, що спростять та прискорять розробку під Windows. Оскільки створення програми для мобільних платформ не таке доцільне, як для персональних ПК, я використаю C#.

Серед двох найвідоміших середовищ розробки для C# (Visual Studio та JetBrains Rider) я обираю Visual Studio. Попри те, що суб’єктивно для мене звичніший і зручніший Rider після довгого використання Android Studio від тих же розробників, Rider, на жаль, не підтримує Windows Forms, що можна використати як бібліотеку для створення користувацького інтерфейсу.

1.2 Огляд технологій для створення інтерфейсу

Якщо розглядати графічні бібліотеки, що підійдуть для даного продукту, які можна використовувати, програмуючи мовою C#, варіантів можна знайти багато. Ось перелік відомих бібліотек з цього списку: WinForms (Windows Forms), WPF (Windows Presentation Foundation), Xamarin. Крім них, є ще значно просунутіші

бібліотеки, використовувані переважно для ігор та іншої складної графіки: OpenGL (Open Graphics Library), DirectX, Vulcan.

Windows Forms є частиною Microsoft .NET Framework. Дозволяє використання C#, C++ та ще декількох мов. Ця бібліотека застаріла, проте досі підтримується.

Windows Presentation Foundation — також частина Microsoft .NET Framework. По-суті, є прямою наступницею WinForms. Порівняно з останньою, новіша бібліотека більш гнучка, застосовує останні існуючі концепти програмування. WPF володіє потужнішими засобами для захисту інформації, кращими можливостями для дизайну інтерфейсу і працює швидше порівняно з попередницею. За все це розробникам на WPF доводиться платити помітно більшим порогом входження і збільшеною складністю використання[1].

Xamarin — технологія, що дозволяє створювати нативні додатки для Android, iOS, macOS та Windows використовуючи єдину кодову базу на C#[2]. Станом на квітень 2017 року, Xamarin використовують більш ніж 1.4 мільйони розробників по всьому світу[3]. Звичайно, написання коду з використанням Xamarin ускладнюється потребою розглядати випадки використання продукту на різних платформах.

DirectX та Vulcan — набори API, створені для спрощення завдань, пов’язаних з ігровим програмуванням. Інакше кажучи, для створення відеоігор. Вони помітно потужніші за попередні названі технології. Проте хоч DirectX та Vulcan теоретично дозволяють створити інтерфейс, потрібний для створюваного продукту, процес написання коду значно ускладниться і подовжиться порівняно з попередніми, а незначний приріст у продуктивності програми не грає ролі на сучасних комп’ютерах. Офіційна підтримка .NET у DirectX була припинена[4], а для Vulcan її не було ніколи. Для обох випадків можна використовувати бібліотеки з відкритим

кодом, що допоможуть використовувати C# з цими технологіями. Проте пошук якісної бібліотеки серед десятку — зайве, непотрібне ускладнення завдання.

Отже, зваживши всі плюси й мінуси кожної названої технології, я дійшов висновку, що варто застосувати Windows Forms, адже ця бібліотека найпростіша у використанні поміж інших; дизайн інтерфейсу цілковито базовий, бо мета завдання не у створенні довершеного UI та UX; відносно невелика ефективність виконання коду з використанням WinForms не буде критичною для такого простого додатку з мінімумом інтерфейсу та функцій.

1.3 Огляд алгоритмів шифрування.

Перед тим як розпочати аналіз конкретно заданого в завданні алгоритму шифрування — перестановочного алгоритму, потрібно прояснити центральні поняття у світі шифрування.

Шифр — метод перетворення інформації з метою її захисту від несанкціонованих користувачів.

Ключ — змінний елемент шифру, який містить в собі певні правила, за допомогою яких відбувається шифрування конкретного повідомлення.

Шифрування (зашифровування) — процес застосування шифру до інформації, яку потрібно захистити, тобто перетворення інформації, що захищається, за певними правилами, що містяться в шифрі.

Дешифрування (дешифровка) — процес, обернений до шифрування, тобто перетворення зашифрованого повідомлення назад в незахищену інформацію за допомогою певних правил, що містяться в шифрі.

Злом (розтин) шифру — процес отримання захищених даних із зашифрованого повідомлення без використання заздалегідь відомого ключа, використаного у шифрі.

Останні два важливі поняття це симетричне та асиметричне шифрування.

У випадку симетричного шифрування є лише один ключ, що використовується і для шифрування, і для дешифрування. В свою чергу, симетричні алгоритми шифрування діляться на потокові і блочні. Потокові алгоритми шифрування послідовно обробляють текст повідомлення. Блочні алгоритми співпрацюють з блоками фіксованого розміру. Зазвичай розмір такого блоку дорівнює 64 бітам. Проте розмір буває і інший: в AES, для прикладу, розмір блоку — 128 біт.

Інколи симетричні алгоритми шифрування можуть використовуватися у поєднанні з асиметричними: асиметричні алгоритми використовуються для передачі ключів ефективніших симетричних алгоритмів. В основному, симетричні алгоритми шифрування вимагають менше обчислень, ніж асиметричні. Відповідно, якісні асиметричні алгоритми в сотні або в тисячі разів повільніші за якісні симетричні алгоритми. Недоліком симетричних алгоритмів є необхідність мати секретний ключ з обох боків передачі інформації. Оскільки ключі можуть перехопити зловмисники, їх необхідно часто змінювати та передавати по безпечних каналах передачі інформаці. Найвідомішими алгоритмами симетричного шифрування є AES, DES, 3DES, IDEA.

Асиметричне шифрування (криптографічна система з відкритим ключем), передбачає застосування пари ключів: відкритого та секретного. Відкритий ключ передається по незахищеному відкритому каналі і використовується, щоб зашифрувати повідомлення. Секретний же використовується для розшифровування повідомлення. В асиметричному шифруванні ключі співпрацюють у парі, тобто коли інформація шифрується відкритим ключем, то розшифровування відбувається тільки відповідним секретним ключем та навпаки. Таке шифрування сьогодні широко застосовується в мережевих протоколах. RSA — приклад алгоритму асиметричного шифрування.

Отож, асиметричні методи шифрування безпечніші, бо дозволяють уникнути потреби передавати секретний ключ по спеціальному захищеному каналу. З іншого

боку, довжина ключа в асиметричних алгоритмів значно більша, ніж у симетричних. Окрім того, асиметричні алгоритми працюють на порядки повільніше симетричних.

1.4 Аналіз перестановочного шифру.

Перестановочний шифр — такий шифр, в якому позиції елементів відкритого тексту міняють місцями згідно з певним, встановленим ключем, порядком. Елементами тексту можуть бути окремі знаки або їхні групи (пари, трійки знаків і так далі). Криптоаналіз перестановочного шифру не є справді складним завданням, адже він симетричний і дуже старий, тому стійкий шифр не можна створити застосовуючи лише перестановки. Проте, він застосовний у поєднанні з більш потужним методом шифрування.

Є багато варіантів реалізації такого шифру. Розлянемо декілька з них.

Шифр маршрутної перестановки, як і переважна більшість шифрів перестановки, базується на занесенні елементів у певну таблицю (матрицю). У випадку цієї реалізації, розміри таблиці обговорюються заздалегідь. Ключову роль відіграє маршрут запису елементів. Ключ і визначає цей маршрут. Наприклад, маршрут може бути таким: по горизонталі, починаючи з нижнього правого кутка, згори вниз. Такий шифр використовувався в поєднанні з іншим під час Громадянської війни в США.

Шифр вертикальної перестановки. Текст записується у рядки таблиці з визначеною довжиною. Довжина рядка така ж, як і довжина ключа. Считування відбувається по колонках, причому у порядку, визначеному ключем. Зазвичай порядок читання визначається алфавітним порядком знаків у ключі.

Поєднання попередніх двох реалізацій дає третю, більш складну: шифр подвійної перестановки. Спершу елементи тексту за деяким визначеним ключем маршрутом записуються у таблицю, а потім за певним порядком, також визначеним ключем, переставляються стовпці й рядки таблички. Ключ містить у собі розмір

таблиці, маршрути вписування та виписування, а також порядок перестановки стовпців та рядків[5].

Для цієї роботи я використаю шифр вертикальної перестановки, бо він наглядний і нескладний у реалізації.

1.5. Висновок до першого розділу

У цьому розділі було розглянуто алгоритм, що буде реалізовано у програмі. Також визначено стек технологій, що будуть використовуватися під час написання програми.

РОЗДІЛ 2. ПРОЕКТУВАННЯ ПРОГРАМИ

2.1 Блок-схема методу Encipher.

Метод Encipher зашифровує текст, переданий у параметрі input. Нижче зображена його блок-схема (див. рисунок 2.1 і рисунок 2.2).

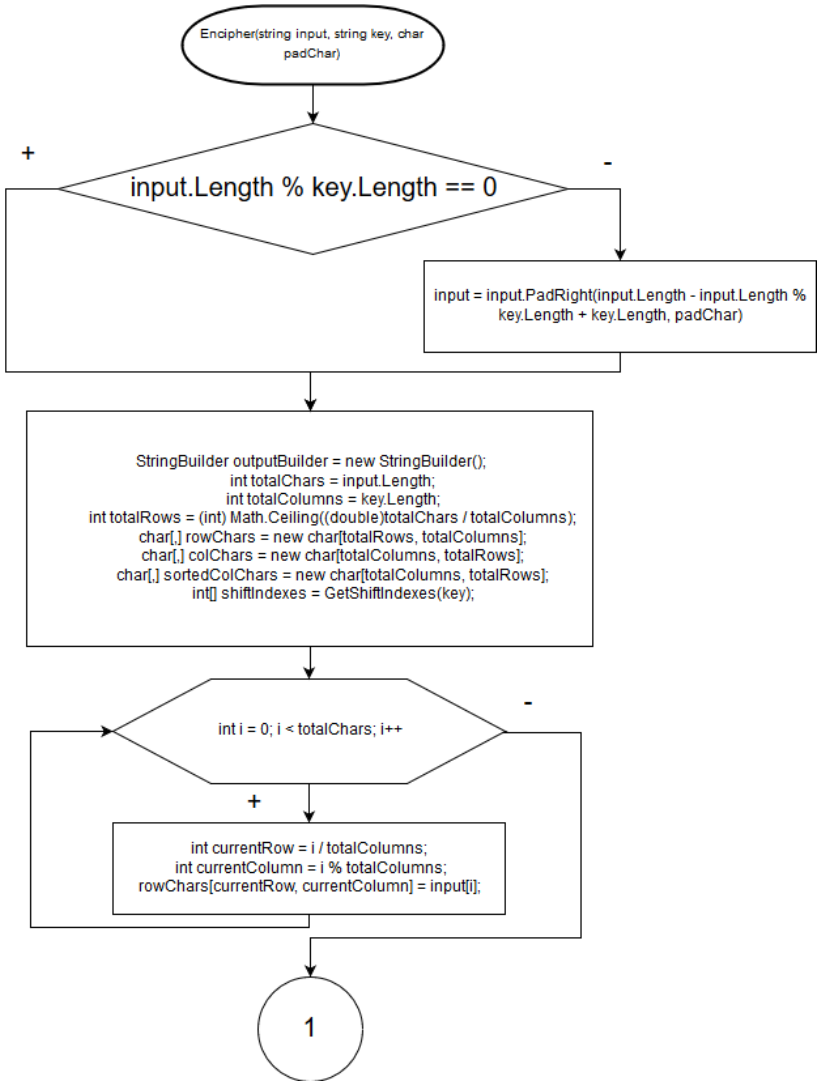

Рисунок 2.1 — блок-схема методу Encipher, частина перша

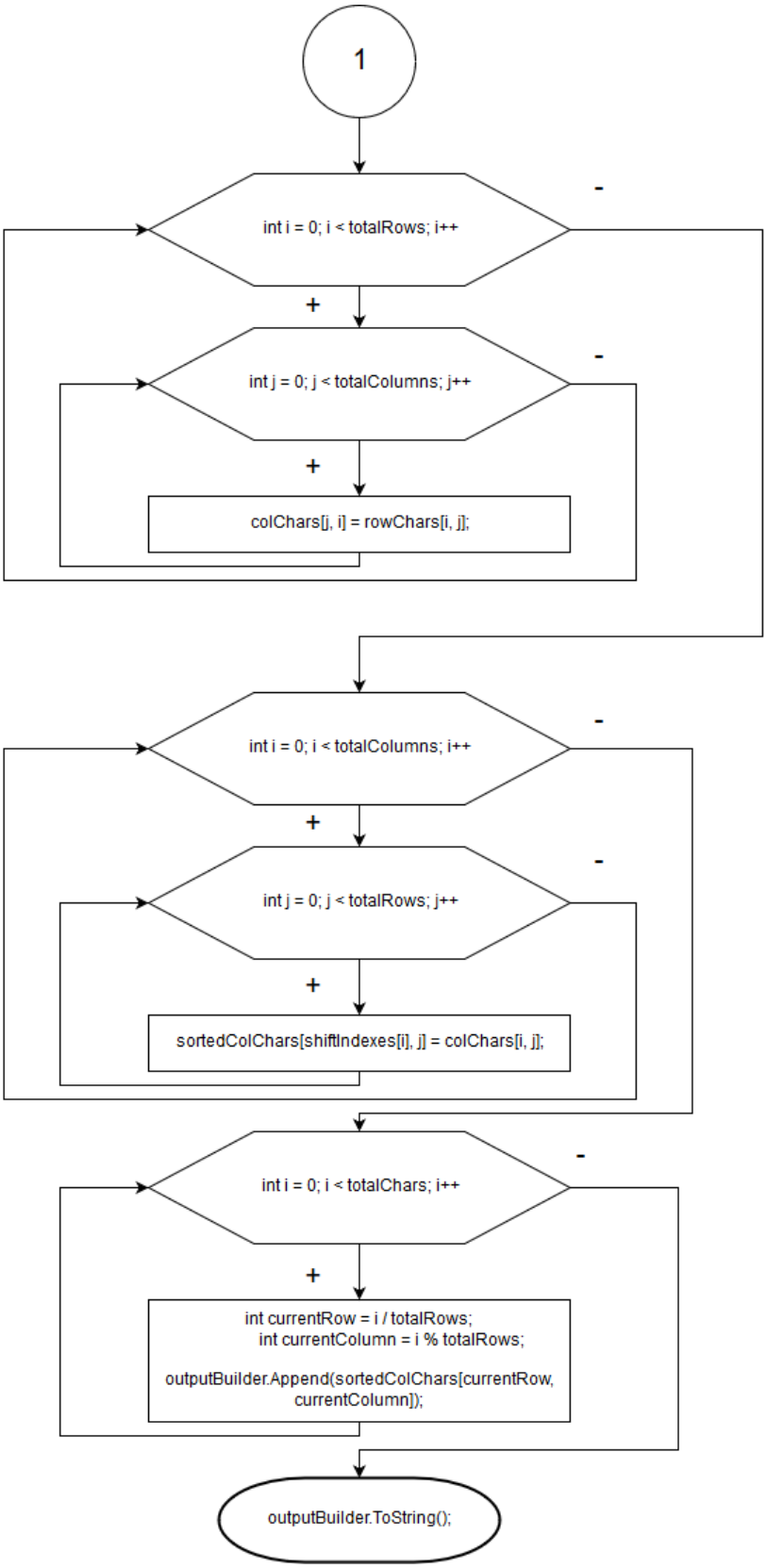

Рисунок 2.2 — блок-схема методу Encipher, частина друга

2.2 Блок-схема методу Decipher.

Алгоритм дешифровки тексту дуже схожий на алгоритм його шифрування, проте має деякі ключові відмінності. По-суті, частина логіки тут виконується у зворотньому порядку. Крім того, відсутній if-statement на початку метода, що застосовувався для «вирівнювання» тексту — додавав до кінця тексту символи, щоб загальна довжина тексту ділилася на довжину ключа. Це тому, що текст, який передається в Decipher, уже «вирівняний». Нижче зображена його блок-схема (див. рисунок 2.3 і рисунок 2.4).

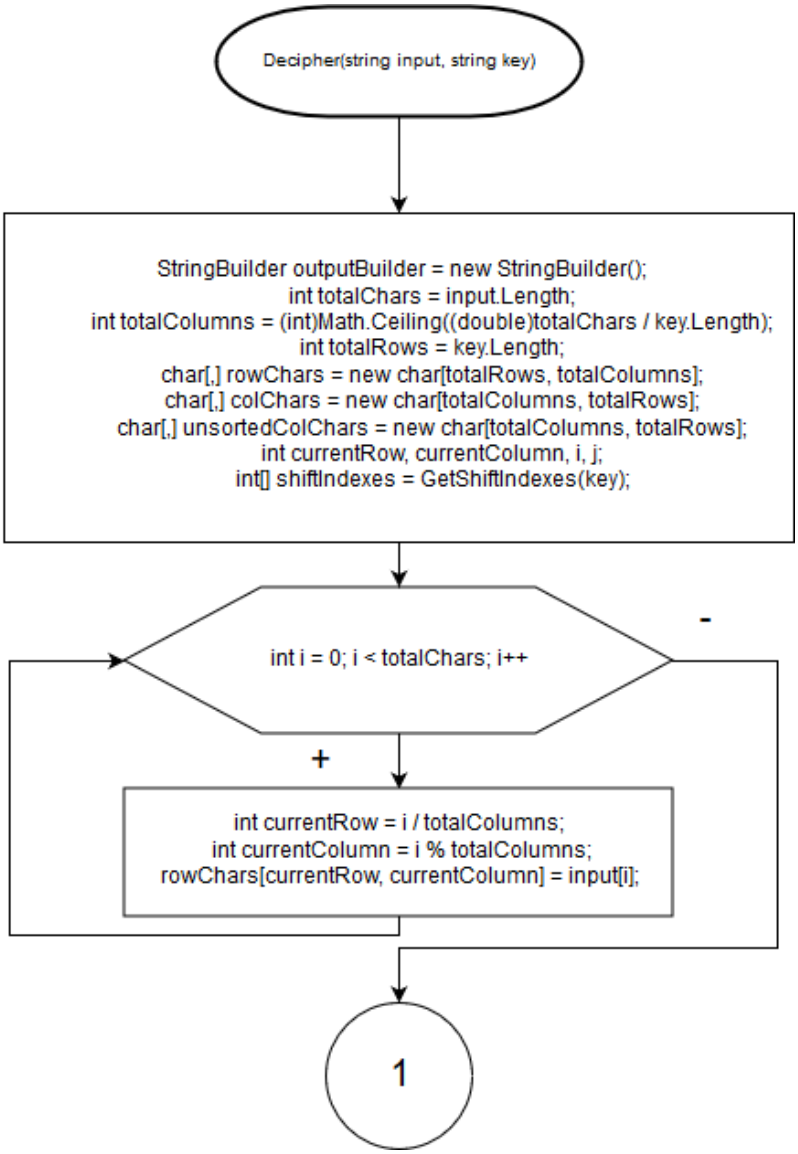

Рисунок 2.3 — блок-схема методу Decipher, частина перша

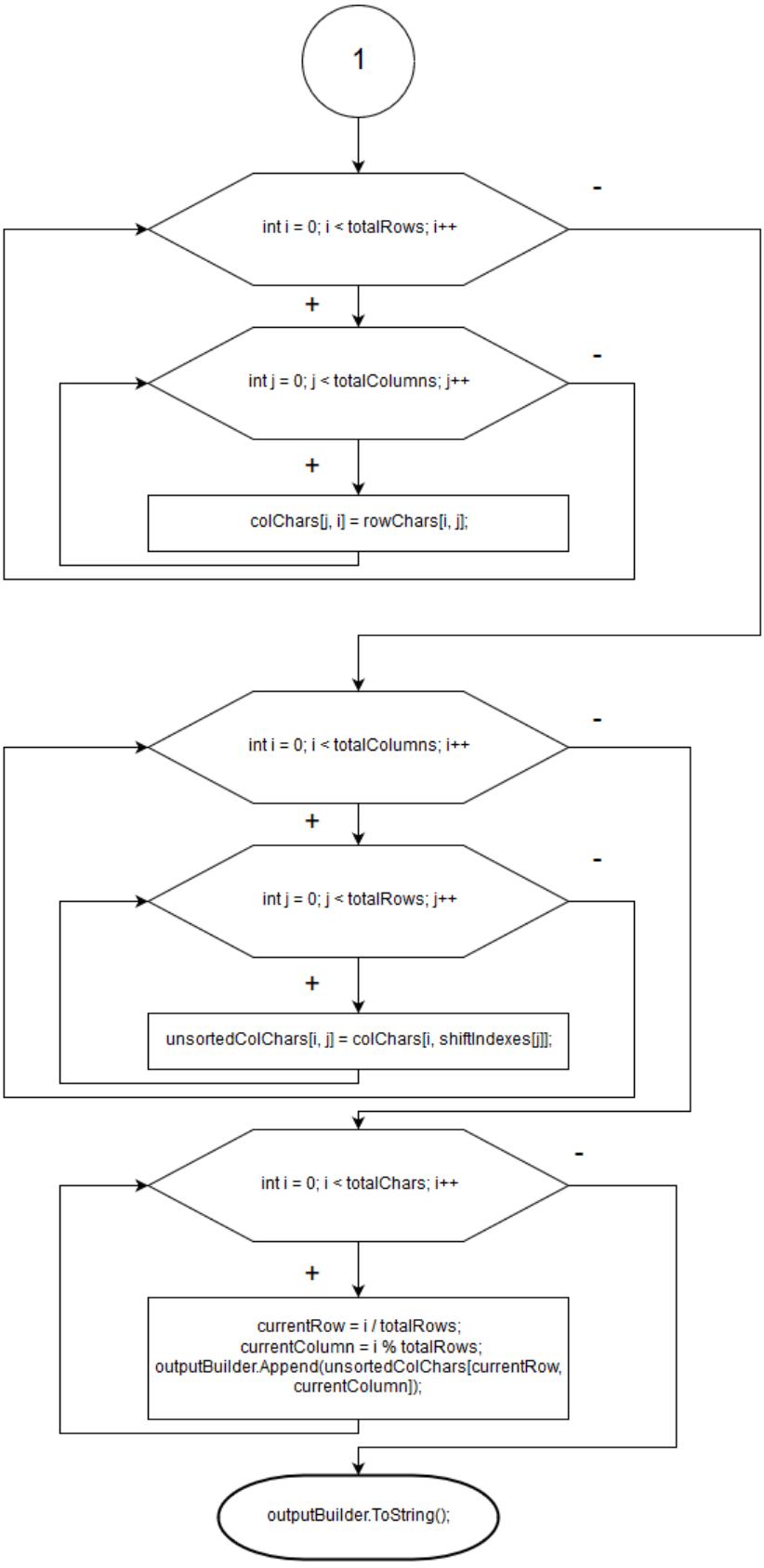

Рисунок 2.4 — блок-схема методу Decipher, частина друга

2.3 Блок-схема методу GetShiftIndexes.

В обох методах, описаних вище, застосовується третій метод класу TranspositionCipher — GetShiftIndexes. Він використовується для визначення порядку, в якому повинні читатися колонки таблички, утвореної з тексту. Справа зображена його блок-схема (див. рисунок 2.5).

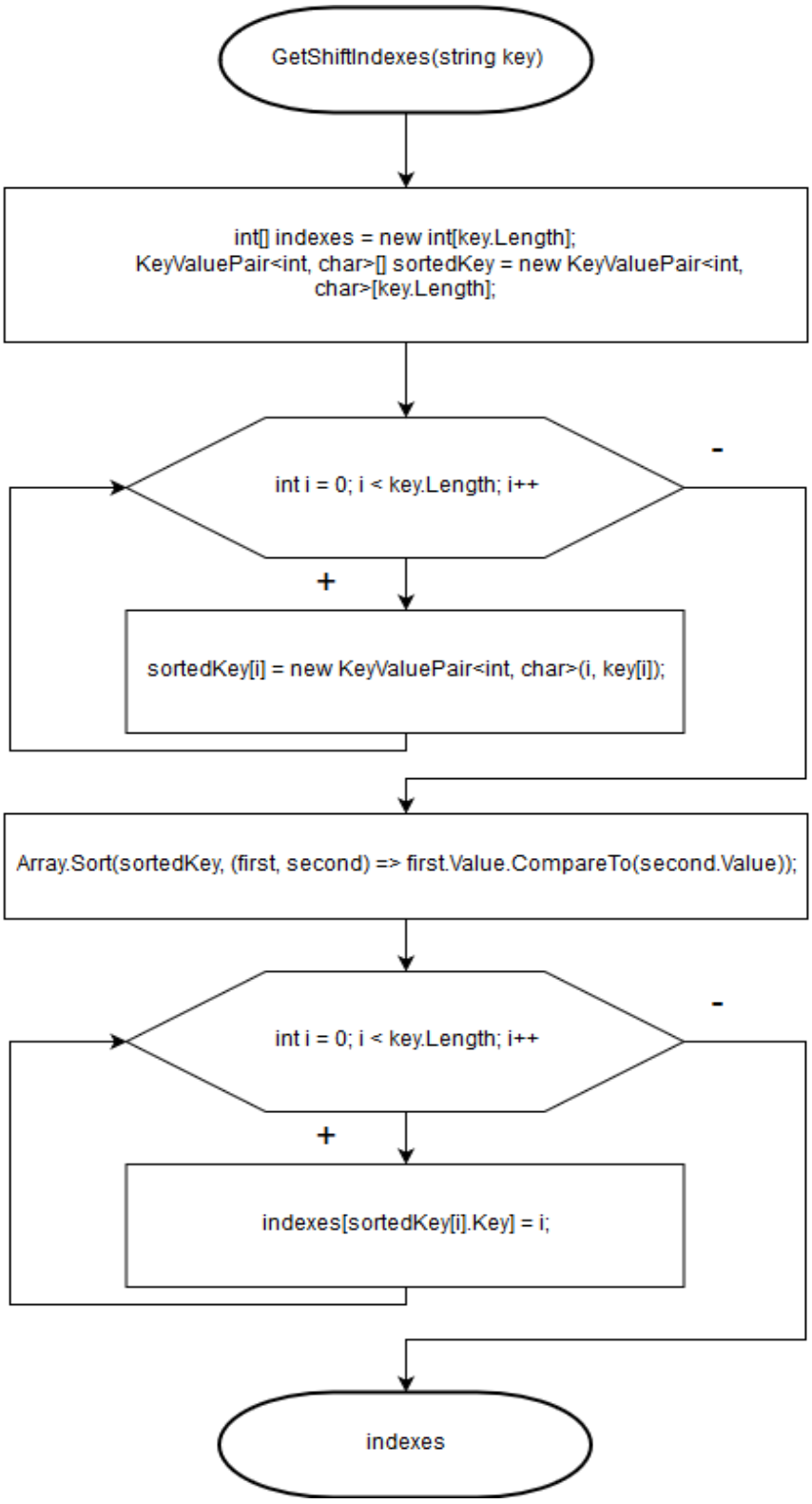

Рисунок 2.5 — блок-схема методу GetShiftIndexes

2.4. Висновок до другого розділу

У цьому розділі через блок-схеми було описано основні методи програми, що здійснюють шифрування та дешифрування тексту. Тепер можна розпочинати реалізацію алгоритму.

РОЗДІЛ 3. РЕАЛІЗАЦІЯ ТА ТЕСТУВАННЯ ПЕРЕСТАНОВОЧНОГО ШИФРУ

3.1 Реалізація користувацького інтерфейсу.

Користувацький інтерфейс умовно ділиться на дві частини: для шифрування і для розшифрування. Обидві частини містять в собі два текстових поля і одну кнопку. Зверху вниз ідуть: поле для вводу тексту; поле для вводу ключа; кнопка, що запускає процес шифрування чи розшифровування. Посередині між цими двома частинами є ще два текстові поля, що вказують куди вводити текст, а куди ключ.

Текстові поля Text і Key не мають функціонального навантаження, вони не повинні використовуватися для введення у них тексту, тож значення їхніх властивостей Enabled були змінені з True на False. Завдяки цій властивості вони стали візуально і практично неактивними, з ними не можна взаємодіяти. Див. рисунок 3.1.

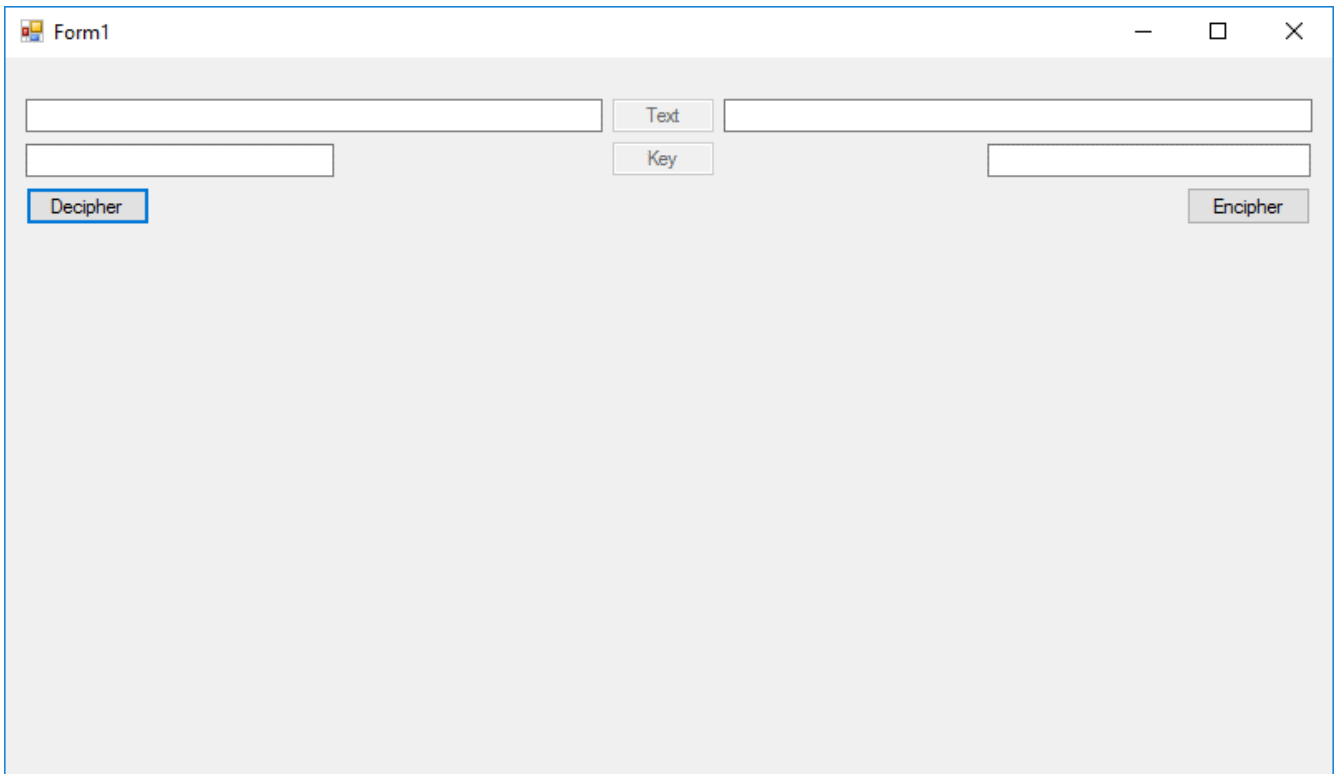

Рисунок 3.1 — Інтерфейс програми

Кожній кнопці назначено колбек на клік: ButtonDecipher_Click і ButtonEncipher_Click.

Лістинг 3.1 Колбеки на натискання кнопок

```
        private void ButtonDecipher_Click(object sender, EventArgs e)
        {
            string deciphered;
            if (TextBoxToDecipherKey.Text.Length > 0)
            {
                deciphered = TranspositionCipher.Decipher(TextBoxToDecipher.Text,
TextBoxToDecipherKey.Text);
            }
            else
            {
                deciphered = TranspositionCipher.Decipher(TextBoxToDecipher.Text);
            }

            TextBoxToEnchipher.Text = deciphered;
        }

        private void ButtonEncipher_Click(object sender, EventArgs e)
        {
            string enciphered;
            if (TextBoxToEncipherKey.Text.Length > 0)
            {
                enciphered = TranspositionCipher.Encipher(TextBoxToEnchipher.Text,
TextBoxToEncipherKey.Text);
            }
            else
            {
                enciphered = TranspositionCipher.Encipher(TextBoxToEnchipher.Text);
            }

            TextBoxToDecipher.Text = enciphered;
        }
```

По клікові на кнопку Decipher, у змінну deciphered записується значення, повернене з виклику метода Decipher класу TranspositionCipher. У разі, якщо користувач вводив свій ключ (поле для вводу ключа містить непорожній рядок), Decipher викликається з двома параметрами: текстом з TextBoxToDecipher та ключем з TextBoxToDecipherKey. Якщо коритувач ключ не вводив, у Decipher передається тільки текст із TextBoxToDecipher. Після цього властивості TextBoxToEncipher.Text присвоюється значення змінної deciphered, тобто у текстове поле TextBoxToEncipher записується розшифрований текст.

Для кліку на кнопку Encipher виконується логіка, дзеркальна до ButtonDecipher_Click. У змінну enciphered записується значення, повернене

методом Encipher. Воно, у свою чергу, відображається у текстовому полі TextBoxToDecipher.

3.2 Реалізація алгоритму шифрування. Клас TranspositionCipher.

Було створено статичний клас TranspositionCipher, що містить три статичних методи. Два з них публічні, вони використовуються для шифрування і розшифровування тексту. Ще один метод — GetShiftIndexes — приватний. Він використовується обома попередніми методами (Decipher і Encipher), але не має практичного застосування поза цими двома методами, тому його не можна викликати поза класом TranspositionCipher. Також у цьому класі є константа DefaultKey. Про її застосування далі.

Для всіх трьох методів були застосовані опціональні параметри. Так, в методі Decipher обов'язковим є лише параметр input, тобто те, що необхідно розшифрувати. Замість параметра key, якщо його не передати, буде використана константа DefaultKey зі значенням ElPsyCongroo.

В методі Encipher три параметри. Так само як у Decipher, обов'язковим є input, тобто текст, що потрібно зашифрувати. Параметр Key, якщо його не передати, так само набуде значення DefaultKey. Третій параметр: padChar. Якщо його не передати, його значення буде '-'. Це символ, який додасться в кінець input стільки разів, щоб довжина input націло ділилася на довжину key. Така вимога спричинена особливістю шифру.

У GetShiftIndexes лише один параметр — key. Якщо його не передати, буде використано DefaultKey.

Розберемо кожен метод окремо. Почнемо з найменшого і найпростішого GetShiftIndexes.

Лістинг 3.2 Метод GetShiftIndexes

```
private static int[] GetShiftIndexes(string key = DefaultKey)
{
    int[] indexes = new int[key.Length];
    KeyValuePair<int, char>[] sortedKey = new KeyValuePair<int, char>[key.Length];

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

```
        {
            sortedKey[i] = new KeyValuePair<int, char>(i, key[i]);
        }

        Array.Sort(sortedKey, (first, second) => first.Value.CompareTo(second.Value));

        for (int i = 0; i < key.Length; i++)
        {
            indexes[sortedKey[i].Key] = i;
        }

        return indexes;
    }
```

Масив indexes на початку виконання методу містить лише нулі, бо це стандартне значення для int. Далі йде масив sortedKey пар типу int char. В першому циклі sortedKey заповнюється парами (індекс символа в ключі) – (сам символ ключа). Далі елементи sortedKey сортуються за допомогою метода рядка CompareTo. Цей метод гарантує правильну роботу лише для символів латинського алфавіту, тому застосування у ключі символів з інших алфавітів може призвести до неочікуваних наслідків. Відсортований в алфавітному порядку по значенню в парі, масив sortedKey використовується у останньому циклі методу, щоб заповнити масив indexes потрібними значеннями: ключами пар із sortedKey. Тепер indexes вказує порядок, за яким слід читати елементи матриць, утворених із рядка input у інших методах класу. Масив indexes повертається і виконання методу закінчується.

Розглянемо детальніше методи Encipher та Decipher. Вони дуже схожі між собою, тож показувати я буду на прикладі Encipher і перейду до Decipher лише на одній серйозній відмінності.

Лістинг 3.3

```
        for (int i = 0; i < totalChars; i++)
        {
            int currentRow = i / totalColumns;
            int currentColumn = i % totalColumns;
            rowChars[currentRow, currentColumn] = input[i];
        }
```

Цикл зверху розбиває рядок input на декілька рядків. Довжина одного такого рядка рівна довжині ключа.

Лістинг 3.4

```
for (int i = 0; i < totalRows; i++)
{
    for (int j = 0; j < totalColumns; j++)
    {
        colChars[j, i] = rowChars[i, j];
    }
}
```

Цикл зверху виконує транспонування матриці. За матрицю приймемо табличку, на яку був розбитий input.

Лістинг 3.5

```
for (int i = 0; i < totalColumns; i++)
{
    for (int j = 0; j < totalRows; j++)
    {
        sortedColChars[shiftIndexes[i], j] = colChars[i, j];
    }
}
```

Цикл зверху переміщає колонки в транспонованій матриці згідно з порядком, отриманим з GetShiftIndexes.

Лістинг 3.6

```
for (i = 0; i < totalColumns; i++)
{
    for (j = 0; j < totalRows; j++)
    {
        unsortedColChars[i, j] = colChars[i, shiftIndexes[j]];
    }
}
```

Цикл зверху належить методу Decipher. Тут виконується зворотня операція до попереднього описаного циклу.

Лістинг 3.7

```
for (int i = 0; i < totalChars; i++)
{
    int currentRow = i / totalRows;
    int currentColumn = i % totalRows;
    outputBuilder.Append(sortedColChars[currentRow, currentColumn]);
}
```

Нарешті, цикл зверху записує кожен елемент матриці у string builder. Після цього утворений builder'ом рядок повертається і виконання методу завершується.

3.3 Тестування алгоритму шифрування

Введемо в поле для вводу тексту для зашифровки "Hello world!". Залишимо поле для ключа порожнім, щоб використати значення DefaultKey. Натиснемо на

кнопку Encipher. Як бачимо, зашифроване повідомлення з’явилося у полі вводу тексту для розшифровки. Див. рисунок 3.2

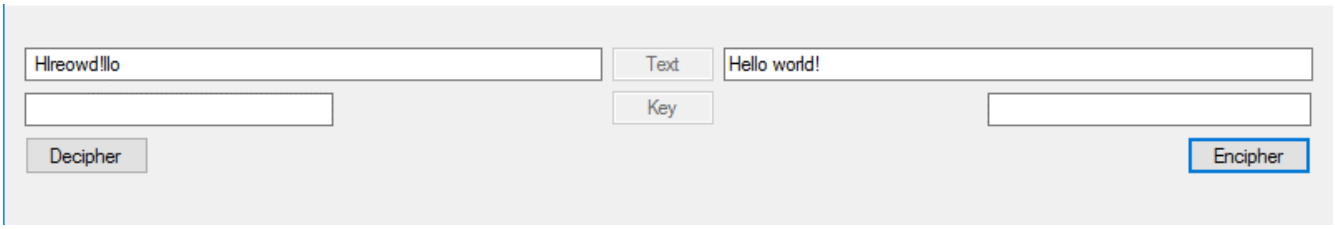

Рисунок 3.2

Зітремо “Hello world”, залишимо поле для вводу ключа знову порожнім. Див. Рис 3.3

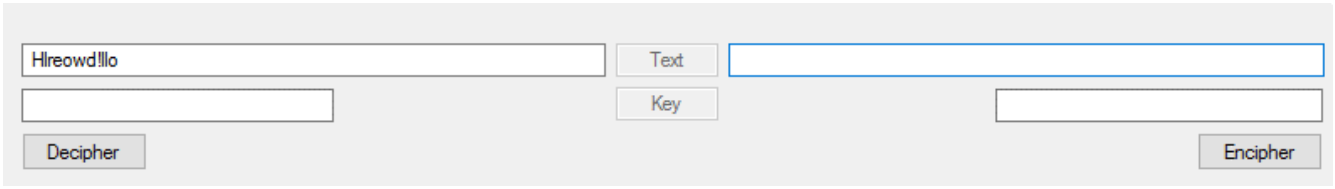

Рисунок 3.3

Натиснемо Decipher. У правому полі з’явився розшифрований текст із лівого. Див. Рис 3.4

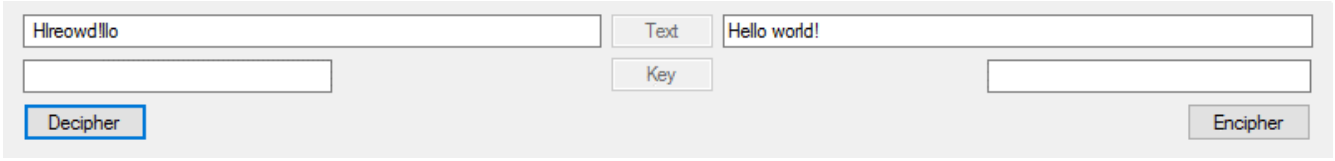

Рисунок 3.4

3.4 Висновок до третього розділу

У програмі реалізований необхідний функціонал — перестановочне шифрування.

ВИСНОВКИ

Мовою програмування високого рівня C# було створено програму, що реалізує алгоритм перестановочного шифрування інформації. Мінімально необхідний для функціонування користувацький інтерфейс було зроблено через Windows Forms.

Програма здатна зашифровувати і розшифровувати текст, використовуючи перестановочне шифрування. Для додаткової зручності користувач може оминути крок введення ключа. В такому разі і для шифрування, і для розшифровки буде використаний стандартний ключ, закладений у програму.

Оскільки застосований алгоритм шифрування дуже застарілий, програму можна використовувати у ознайомлювальних цілях. Алгоритм нескладний, тому код, що його реалізує, також легко читати. Розроблену програму можна показувати людям, що починають вивчати алгоритми шифрування, адже в разі їхньої зацікавленості вони можуть переглянути код програми, що поліпшить їхні знання і сприятиме кращому розумінню алгоритмів шифрування.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Difference between Winforms vs WPF [Електронний ресурс]. (https://www.educba.com/winforms-vs-wpf/) Переглянуто: 09.05.19
2. Xamarin App Development with Visual Studio [Електронний ресурс]. (https://visualstudio.microsoft.com/xamarin/) Переглянуто: 09.05.19
3. Xamarin — Wikipedia[Електронний ресурс]. (https://en.wikipedia.org/wiki/Xamarin) Переглянуто: 09.05.19
4. c# - How to develop DirectX applications? – Stack Overflow [Електронний ресурс]. (https://stackoverflow.com/questions/43961070/how-to-develop-directx-applications) Переглянуто: 09.05.19
5. Transposition cipher – Wikipedia [Електронний ресурс]. (https://en.wikipedia.org/wiki/Transposition_cipher) Переглянуто: 09.05.19