Лабораторна робота № 9

Тема: «Обробка виключень та робота з файлами».

Мета: вивчити основні способи роботи з виключеннями. Виключення користувача. Відкриття файлів, зчитування та запис у файл. Шляхи доступу до файлів. Функції, методи та атрибути для роботи з файлами.

Завдання:

1. Вивчити матеріал лекцій
2. Виконати індивідуальне завдання лабораторної роботи, вибране відповідно до варіанту.

Теоретичні основи:

Обробка виключень

Виключення – це повідомлення інтерпретатора, які виникають у випадку виникнення помилки в програмному коді або при настанні якої-небудь події. Якщо в коді не передбачена обробка виключення, то виконання програми переривається, і виводиться повідомлення про помилку.

Існують три типи помилок у програмі:

Синтаксичні помилки – це помилки в імені оператора або функції, відсутність закриваючих або відкриваючих лапок і т. д., тобто помилки в синтаксисі мови.

Логічні помилки – це помилки в логіці програми, які можна виявити тільки за результатами її роботи.

Помилки часу виконання – це помилки, які виникають під час роботи програми. Причиною є події, не передбачені програмістом.

Інструкція *try...except...else...finally*

Для обробки виключень призначена інструкція `try`.

Формат інструкції:

```
try:
<Блок, у якому перехоплюються виключення>
[except [<Виключення1>[ as <Об'єкт виключення>]]:
<Блок, виконуваний при виникненні виключення>
[ . . .
except [<Виключенняn>[ as <Об'єкт виключення>]]:
<Блок, виконуваний при виникненні виключення>]]
[else:
<Блок, виконуваний, якщо виключення не виникло>]
[finally:
<Блок, виконуваний у будь-якому випадку>]
```

Інструкція *with...as*

Мова Python підтримує *протокол менеджерів контексту.* Цей протокол гарантує виконання завершальних дій (наприклад, закриття файлу) незалежно від того, відбулося виключення усередині блоку коду чи ні.

Для роботи з протоколом призначена інструкція `with...as`. Інструкція має наступний формат:

```
with <Вираз1>[ as <Змінна>][,...,
     <Виразn> [as <Змінна>]]:
     <Блок, у якому перехоплюємо виключення>
```

Спочатку обчислюється `<Вираз1>`, який повинен повертати об'єкт, що підтримує протокол. Цей об'єкт повинен мати два методи: `__enter__()` і `__exit__()` . Метод `__enter__()` викликається після створення об'єкта. Значення, що повертається цим методом, присвоюється змінній, зазначеній після ключового слова `as`. Якщо змінна не зазначена, значення, що повертається, ігнорується. Формат методу `__enter__()`:

```
__enter__(self)
```

Далі виконуються інструкції усередині тіла інструкції `with`. Якщо при виконанні виникло виключення, то керування передається методу `__exit__()`. Метод має наступний формат:

```
__exit__(self, <Тип виключення>, <Значення>, <Об'єкт
traceback>)
```

Виключення користувача

Для виконання виключень користувача призначено дві інструкції: `raise i assert`.

Інструкція `raise` виконує задане виключення. Вона має кілька варіантів формату:

```
raise <Екземпляр класу>
raise <Назва класу>
raise <Екземпляр або назва класу> from <Об'єкт
виключення>
raise
```

У першому варіанті формату інструкції `raise` вказується екземпляр класу порушуваного виключення. При створенні екземпляра можна передати дані конструктору класу.

Ці дані будуть доступні через другий параметр в інструкції `except`.

Робота з файлами

Відкриття файлу

Функція відкриття має наступний формат:

```
open(<Шлях до файлу>[, mode='r') [, buffering=-1) [,
encoding=None) [,errors =None] [, newline=None) [,
closefd=True])
```

Відносний шлях буде автоматично перетворений в абсолютний шлях за допомогою функції `abspath()` з модуля `os.path`. Можливі наступні варіанти:

1. Якщо файл, що відкривається, перебуває в поточному робочому каталозі, то можна вказати тільки назву файлу.
2. Якщо файл, що відкривається, розташований у вкладеній папці, то перед назвою файлу приводять назви вкладених папок через слеш.
3. Якщо папка з файлом розташована вище рівнем, то перед назвою файлу вказують дві крапки й слеш (`"..\"`).
4. Якщо на початку шляху розташований слеш, то шлях відлічується від кореня диска. У цьому випадку місце розташування поточного робочого каталогу не має значення.

В абсолютному й відносному шляхах допускається вказувати як прямі, так і зворотні слеши. Усі вони будуть автоматично перетворені з урахуванням значення атрибута `sep` з модуля `os.path`. Значення цього атрибута залежить від використовуваної операційної системи.

Необов'язковий параметр `mode` у функції `open()` може приймати наступні значення:

`r` – тільки читання (значення за замовчуванням).
`r+` – читання й запис.
`w` – запис.
`w+` – читання й запис.
`a` – запис.
`a+` – читання й запис.
`x` – створення файлу для запису.
`x+` – створення файлу для читання й запису.
Після вказівки режиму може слідувати модифікатор:

`b` – файл буде відкритий у бінарному режимі. Файлові методи приймають і повертають об'єкти типу `bytes`;

t – файл буде відкритий у текстовому режимі (значення за замовчуванням у Windows).

У параметрі errors можна вказати рівень обробки помилок. Можливі значення:
"strict" (при помилці виконується виключення Valueerror – значення за замовчуванням),
"replace" (невідомий символ заміняється символом питання або символом з кодом \ufffd),
"ignore" (невідомі символи ігноруються),
"xmlcharrefreplace" (невідомий символ заміняється послідовністю &#xxxx;)
"backslashreplace" (невідомий символ заміняється послідовністю \uxxxx).

Параметр newline задає режим обробки символів кінця рядків. Підтримувані ним значення такі:
• None (значення за замовчуванням) – виконується стандартна обробка символів кінця рядка. Наприклад, в Windows при читанні символи \r\n перетворяться в символ \n, а при записі проводиться зворотне перетворення;
• "" (порожній рядок) – обробка символів кінця рядка не виконується;
• "<Спеціальний символ>" – зазначений спеціальний символ використовується для позначення кінця рядка, і додаткова обробка не виконується. Як спеціальний символ можна вказати лише \r\n, \r і \n.

Методи для роботи з файлами

• close() – закриває файл.

• write (<дані>) – записує рядок або послідовність байтів у файл.

• writelines (<Послідовність>) – записує послідовність у файл.
• writable() - повертає True, якщо файл підтримує запис, і False - якщо ні.

•read([<Кількість>]) – зчитує дані з файлу.
• readline([<Кількість>]) – зчитує з файлу один рядок при кожному виклику.
• readlines() – зчитує весь вміст файлу в список.
• flush() – примусово записує дані з буфера на диск;
• fileno() – повертає цілочисельний дескриптор файлу. Значення, що повертається, завжди буде більшим за число 2, оскільки число 0

закріплене за стандартним вводом `stdin`, 1 – за стандартним виводом `stdout`, а 2 – за стандартним виводом повідомлень про помилки `stderr`.

• `truncate ([<Кількість>])` – обрізає файл до зазначеної кількості символів (якщо заданий текстовий режим) або байтів (у випадку бінарного режиму).

• `tell()` – повертає позицію покажчика відносно початку файлу у вигляді цілого числа.

• `seek(<Зсув>[, <Позиція>] )` – установлює покажчик у позицію, що має зсув `<Зсув>` відносно позиції `<Позиція>`. У параметрі `<Позиція>` можуть бути зазначені наступні атрибути з модуля `io` або відповідні їм значення:

- `io.SEEK_SET` або 0 –- початок файлу (значення за замовчуванням);
- `io.SEEK_CUR` або 1 – поточна позиція покажчика. Додатне значення зсуву викликає переміщення до кінця файлу, від'ємне - до його початку;
- `io. SEEK_END` або 2 – кінець файлу.

• `seekable()` – повертає `True`, якщо покажчик файлу можна зсунути в іншу позицію, і `False` – якщо ні.

Крім методів, об'єкти файлів підтримують кілька атрибутів:
`name` – ім'я файлу;
`mode` – режим, у якому був відкритий файл;
`closed` – повертає `True`, якщо файл був закритий, і `False` – якщо ні.
`encoding` – назва кодування, яке буде використовуватися для перетворення рядків перед записом у файл або при читанні.

Стандартний вивід `stdout` також є файловим об'єктом. Атрибут `encoding` цього об'єкта завжди містить кодування пристроїв виводу, тому рядок перетвориться в послідовність байтів у правильному кодуванні.

`buffer` – дозволяє одержати доступ до буфера. Атрибут доступний тільки в текстовому режимі. За допомогою цього об'єкта можна записати послідовність байтів у текстовий потік.

Права доступу до файлів і каталогів

Права доступу позначають буквами:

`r` – файл можна читати, а вміст каталогу можна переглядати;

`w` – файл можна модифікувати, видаляти й перейменовувати, а в каталозі можна створювати або видаляти файли. Каталог можна перейменувати або вилучити;

x – файл можна виконувати, а в каталозі можна виконувати операції над файлами, у тому числі робити в ньому пошук файлів.

Таблиця. Права доступу в різних записах

Восьмерична цифра	Двійковий запис	Буквений запис	Вісімкова цифра	Двійковий запис	Буквений запис
0	000	---	4	100	r--
1	001	--x	5	101	r-x
2	010	-w-	6	110	r-w
3	011	-wx	7	111	rwx

Для визначення прав доступу до файлу або каталогу призначена функція access() з модуля os. Функція має наступний формат:

access(<Шлях>, <Режим>)

Функція повертає True, якщо перевірка пройшла успішно, або False – якщо ні. У параметрі <Режим> можуть бути зазначені наступні константи, що визначають тип перевірки:
os.F_OK – перевірка наявності шляху або файлу:
os.R_OK – перевірка на можливість читання файлу або каталогу;
os.W_OK – перевірка на можливість запису у файл або каталог;
os.X_OK – визначення, чи є файл або каталог виконуваним.
chmod()- зміна прав доступу. Функція має наступний формат: chmod(<Шлях>, <Права доступу>)

Функції для маніпулювання файлами

Для копіювання й переміщення файлів призначені наступні функції з модуля shutil:

copyfile(<Копіюємий файл>,<Куди копіюємо>)

Функція дозволяє скопіювати вміст файлу в інший файл. Ніякі метадані (наприклад, права доступу) не копіюються.

Якщо файл існує, то він буде перезаписаний. Якщо файл не вдалося скопіювати, виконується виключення OsError або одне з виключень, що є підкласом цього класу.

copy(<Копіюємий файл>, <Куди копіюємо>)
Функція дозволяє скопіювати файл разом із правами доступу.
Функція copy() як результат повертає шлях скопійованого файлу;

Copy2(<Копіюємий файл>, <Куди копіюємо>)
Функція дозволяє скопіювати файл разом з метаданими. Функція copy() як результат повертає шлях скопійованого файлу;

move(<Шлях до файлу>, <Куди переміщаємо>)

Функція переміщає файл у зазначене місце з видаленням початкового файлу. Функція move () як результат повертає шлях переміщеного файлу.

Для перейменування й видалення файлів призначені наступні функції з модуля os:

rename (<Старе ім'я>, <Нове ім'я>)

Функція перейменовує файл.

Функції для роботи з файлами

Модуль os. path містить додаткові функції, що дозволяють перевірити наявність файлу, одержати розмір файлу й ін. Опишемо ці функції:

exists (<Шлях>)

Функція перевіряє зазначений шлях на існування. Значенням функції буде True, якщо шлях існує, і False – якщо не існує.

getsize (<Шлях до файлу>)

повертає розмір файлу в байтах.

getatime (<Шлях до файлу>)

Функція служить для визначення часу останнього доступу до файлу. Як значення функція повертає кількість секунд, що пройшли з початку епохи.

getctime (<Шлях до файлу>)

Функція дозволяє довідатися дату створення файлу. Як значення функція повертає кількість секунд, що пройшли з початку епохи.

getmtime (<Шлях до файлу>)

Повертає час останньої зміни файлу. Як значення функція повертає кількість секунд, що пройшли з початку епохи.

Одержати розмір файлу й час створення, зміни й доступу до файлу, а також значення інших метаданих дозволяє функція stat() з модуля os. Як значення функція повертає об'єкт stat_result, що містить десять атрибутів: st_mode, st_ino, st_dev,st_nlink, st_uid, st_gid, st_size, st_atime, st_mtime і st_ctime.

Оновити час останнього доступу й час зміни файлу дозволяє функція utime() з модуля os.

Функція має два варіанти формату:

utime(<Шлях до файлу>, None)

utime(<Шлях до файлу>,(<Останній доступ>, <Зміна файлу>))

Як перший параметр можна вказувати не тільки шлях як рядок, але й цілочисельний дескриптор відкритого файлу, який повертає функція `open()` з модуля `os`. Якщо як другий параметр зазначене значення `None`, то час доступу й зміни файлу буде поточним. У другому варіанті формату функції `utime()` вказується кортеж з нових значень у вигляді кількості секунд, що пройшли з початку епохи.

Перетворення шляху до файлу або каталогу

Перетворити шлях до файлу або каталогу дозволяють наступні функції з модуля `os.path`:

`abspath ( <Відносний шлях>)`

Функція перетворить відносний шлях в абсолютний, враховуючи місце розташування поточного робочого каталогу.

У відносному шляху можна вказати як прямі, так і зворотні слеші. Усі вони будуть автоматично перетворені з урахуванням значення атрибуту `sep` з модуля `os.path`. Значення цього атрибута залежить від використовуваної операційної системи.

`isabs (<Шлях>)`

Функція повертає `True`, якщо шлях є абсолютним, і `False` – якщо ні.

`basename (<Шлях>)`

Функція повертає ім'я файлу без шляху до нього.

`dirname (<Шлях>)`

Функція повертає шлях до папки, де зберігається файл.

`split (<Шлях>)`

Функція повертає кортеж із двох елементів: шляху до папки, де зберігається файл, і назви файлу.

`splitdrive (<Шлях>)`

Функція розділяє шлях на ім'я диска й іншу частину шляху. Як значення повертається кортеж із двох елементів.

`splitext (<Шлях>)`

Функція повертає кортеж із двох елементів: шлях з назвою файлу, але без розширення, і розширення файлу (фрагмент після останньої крапки).

Індивідуальні завдання

Використовуючи функції та методи мови програмування Python:

1. Написати програму створення каталогу зі шляхом та назвою: «C:\lab9\»
2. Написати програму створення підкаталогу «C:\lab9\<прізвище>»

3. Завантажити в даний підкаталог файл *.txt, де * – номер Вашого варіанту лабораторної роботи та виконати з ним дії, що описані в номері Вашого варіанту. Обробити можливі виключення.

4. Навести приклади роботи та використання модулів

```
pickle
shelve
```

№	Завдання
1	Як контрольний приклад створити файл v1.txt, який складається з трьох речень, та продемонструвати на ньому виконання завдання: Зчитати файл «1.txt» та розбити файл на три файли однакового розміру, створивши файли «1part1.txt», «1part2.txt» і «1part3.txt» з кодуванням UTF-8. Вивести на друк розмір початкового файлу та розміри створених файлів. Створити файл, який починається з речення з заданим номером і містить задану кількість речень.
2	Як контрольний приклад створити файл v2.txt, який складається з трьох речень, та продемонструвати на ньому виконання завдання. Розбити файл «2.txt» на 2 файли «2part1.txt», «2part2.txt», які містять однакову кількість речень, та записати їх у кодуванні «cp1251». Вивести на друк кількість символів у початковому файлі та кількість символів у створених файлах. Створити файл, який містить непарні речення.
3.	Як контрольний приклад створити файл v3.txt, який складається з трьох речень, та продемонструвати на ньому виконання завдання. Зчитати файл «3.txt» та розбити його на 2 файли «3part1.txt», «3part2.txt», які містять непарні та парні за номером символи початкового файлу. Зберегти ці файли у кодуванні «cp1251». Зчитати дані файли та відновити з них початковий файл, зберегти цей файл у кодуванні UTF-8.
4	Як контрольний приклад створити файл v4.txt, який складається з трьох речень, та продемонструвати на ньому виконання завдання. Перетворити файл «4.txt» у файл «41.txt», який відрізняється від попереднього тим, що кожне речення у ньому повинно починатися з нового рядка. Створити новий файл «42.txt» з речень, які

	починаються з заданої літери. Зберегти цей файл у кодуванні UTF-8.
5	Як контрольний приклад створити файл v5.txt, який складається з трьох речень, та продемонструвати на ньому виконання завдання. Перетворити файл «5.txt» у файл «51.txt», який відрізняється від попереднього тим, що у ньому виконана симетрична зміна речень місцями. Створити новий файл «52.txt» з речень, які містять парну кількість символів. Зберегти цей файл у кодуванні UTF-8.
6	Як контрольний приклад створити файл v6.txt, який складається з трьох речень та продемонструвати на ньому виконання завдання. Перетворити файл «6.txt» у файл «61.txt», який складається з перших трьох слів кожного речення. Створити новий файл «62.txt» зі слів, які містять парну кількість символів. Зберегти цей файл у кодуванні UTF-8.
7	Як контрольний приклад створити файл v7.txt, який складається з трьох речень, та продемонструвати на ньому виконання завдання. Зчитати файл «7.txt» та перетворити його у файл «71.txt», який складається з речень, у яких виконана симетрична зміна порядку слів, та у файл «72.txt», який містить слова, що закінчуються заданою буквою. Зберегти цей файл у кодуванні UTF-8.
8	Як контрольний приклад створити файл v8.txt, який складається з трьох речень, та продемонструвати на ньому виконання завдання. Зчитати файл «8.txt» та перетворити його у файл «81.txt», який складається з відсортованих за довжиною речень, та у файл «82.txt», який містить слова, що починаються з заданої букви. Зберегти цей файл у кодуванні UTF-8.
9	Як контрольний приклад створити файл v9.txt, який складається з трьох речень, та продемонструвати на ньому виконання завдання. Зчитати файл «9.txt» та перетворити його у файл «91.txt», який складається з речень, що мають парну кількість слів, та у файл «92.txt», який містить слова, що закінчуються на «ать» або «ять». Зберегти цей файл у кодуванні UTF-8.
10	Як контрольний приклад створити файл v10.txt, який складається з трьох речень, та продемонструвати на ньому виконання завдання. Зчитати файл «10.txt» та перетворити його у файл «101.txt», який складається з речень, що мають непарну кількість букв, та у файл «102.txt», що містить слова, у яких більше ніж три голосні букви. Зберегти цей файл у кодуванні UTF-8.

11	Як контрольний приклад створити файл v11.txt, який складається з трьох речень, та продемонструвати на ньому виконання завдання. Зчитати файл «11.txt» та перетворити його у файл «111.txt», який складається з речень, що мають більше ніж одну велику букву, лапки або апостроф, та у файл «112.txt», що містить слова, у яких більше ніж 5 приголосних букв. Зберегти цей файл у кодуванні UTF-8.
12	Як контрольний приклад створити файл v12.txt, який складається з трьох речень, та продемонструвати на ньому виконання завдання. Зчитати файл «12.txt» та перетворити його у файл «121.txt», який складається з речень, сформованих з двох частин: у першу половину речення мають входити слова з непарного за номером речення, а у другу половину – слова з другої половини парного речення. На основі файлу «12.txt» створити також файл «122.txt», що містить слова, множина букв у яких перетинається з заданою. Зберегти цей файл у кодуванні UTF-8.
13	Як контрольний приклад створити файл v13.txt, який складається з трьох речень, та продемонструвати на ньому виконання завдання. Зчитати файл «13.txt» та перетворити його у файл «131.txt», який складається з речень, які містять непарну кількість слів непарної довжини. На основі файлу «13.txt» створити також файл «132.txt», що містить слова, букви у яких перемішані випадковим чином. Зберегти цей файл у кодуванні UTF-8.
14	Як контрольний приклад створити файл v14.txt, який складається з трьох речень, та продемонструвати на ньому виконання завдання. Зчитати файл «14.txt» та перетворити його у файл «141.txt», який складається з речень, слова у яких перемішані випадковим чином, крім першого та останнього слова. На основі файлу «14.txt» створити також файл «142.txt», кириличні букви у якому замінені відповідними за звучанням буквами або буквосполученнями латинського алфавіту.
15	Як контрольний приклад створити файл v15.txt, який складається з трьох речень, та продемонструвати на ньому виконання завдання. Зчитати файл «15.txt» та перетворити його у файл «151.txt», який складається з речень, слова яких розташовані у реверсному порядку. Перше слово має завжди писатися з великої літери. На основі файлу «15.txt» створити також файл «152.txt», кожна кирилична буква у якому замінена на букву, що йде наступною у алфавіті. М'який знак заміняємо на «а».

16	Як контрольний приклад створити файл v16.txt, який складається з трьох речень, та продемонструвати на ньому виконання завдання. Зчитати файл «16.txt» та перетворити його у файл «161.txt», який складається з речень, кількість слів у яких є найближчою до середньої арифметичної кількості у реченнях даного тексту. На основі файлу «16.txt» створити також файл «162.txt», кожна кирилична буква у якому замінена на букву, номер якої у алфавіті дорівнює номеру даної буква за умови, що відлік відбувається у реверсному порядку.
17	Як контрольний приклад створити файл v17.txt, який складається з трьох речень, та продемонструвати на ньому виконання завдання. Зчитати файл «17.txt» та перетворити його у файл «171.txt», який складається з речень, що містять слово, задане шляхом вводу з клавіатури. Кожне речення має починатися з нового рядка. На основі файлу «17.txt» створити також файл «172.txt», кожна кирилична приголосна буква у якому замінена на голосну з відповідним номером, якщо окремо лічити приголосні та голосні. Приголосні, які не мають відповідної за номером голосної, залишити без змін.
18	Як контрольний приклад створити файл v18.txt, який складається з трьох речень, та продемонструвати на ньому виконання завдання. Зчитати файл «18.txt» та перетворити його у файл «181.txt», який складається з речень, що містять слова, які знаходяться на одному зсуві у реченнях, номери яких визначаються за модулем 10. Кожне таке речення має починатися з нового рядка і перше слово писатися з великої літери. На основі файлу «18.txt» створити також файл «182.txt», кожне слово у якому замінити числом, що дорівнює сумі його букв. Одержані послідовності чисел, що відповідають реченням, упорядкувати за спаданням.
19	Як контрольний приклад створити файл v19.txt, який складається з трьох речень, та продемонструвати на ньому виконання завдання. Зчитати файл «19.txt» та перетворити його у файл «191.txt», який складається з речень, слова у яких упорядковані за сумую кодів символів у слові за кодуванням cp1251. Кожне таке речення має починатися з нового рядка і перше слово писатися з великої літери. На основі файлу «19.txt» створити також файл «192.txt», кожне слово у якому замінити числом, що дорівнює сумі його кодів у кодуванні utf-8. Одержані послідовності чисел, що відповідають реченням, упорядкувати за зростанням.

20	Як контрольний приклад створити файл v20.txt, який складається з трьох речень, та продемонструвати на ньому виконання завдання. Зчитати файл «20.txt» та перетворити його у файл «201.txt», який складається з речень, перемішаних випадковим чином з наступним видаленням тих речень, сума букв у яких перевищує середню арифметичну кількість букв, пораховану по всіх реченнях даного тексту. Кожне таке речення має починатися з нового рядка і перше слово писатися з великої літери. На основі файлу «20.txt» створити також файл «202.txt», кожну голосну букву у якому замінити випадково вибраною іншою голосною буквою.
21	Як контрольний приклад створити файл v21.txt, який складається з трьох речень, та продемонструвати на ньому виконання завдання. Зчитати файл «21.txt» та перетворити його у файл «211.txt», який складається з речень, у яких кожне парне слово замінено на випадковим чином вибране слово з наступного речення. Кожне таке речення має починатися з нового рядка і перше слово писатися з великої літери. На основі файлу «21.txt» створити також файл «212.txt», кожну приголосну букву у якому замінити випадково вибраною приголосною буквою з наступного за номером слова.
22	Як контрольний приклад створити файл v22.txt, який складається з трьох речень, та продемонструвати на ньому виконання завдання. Зчитати файл «22.txt» та перетворити його у файл «221.txt», що складається з речень, у яких букви у словах перемішані випадковим чином. Кожне таке речення має починатися з нового рядка і перше слово писатися з великої літери. На основі файлу «22.txt» створити також файл «222.txt» з словами, у яких порядок приголосних букв замінено на реверсний.
23	Як контрольний приклад створити файл v23.txt, який складається з трьох речень, та продемонструвати на ньому виконання завдання. Зчитати файл «23.txt» та перетворити його у файл «231.txt», що складається з речень, кожне з яких утворено шляхом виконання операції об'єднання множин букв у відповідних за номером словах кожного непарного та парного речення. Кожне таке речення має починатися з нового рядка і перше слово писатися з великої літери. На основі файлу «23.txt» створити також файл «232.txt», кожну приголосну букву у якому замінити випадково вибраною голосною, а кожну голосну – випадково вибраною приголосною.
24	Як контрольний приклад створити файл v24.txt, який складається з трьох речень, та продемонструвати на ньому виконання завдання.

	Зчитати файл «24.txt» та перетворити його у файл «241.txt», що складається з речень, кожне з яких утворено шляхом виконання операції об'єднання множин букв у відповідних за номером непарних слів з непарних речень та парних слів з парних речень. Кожне таке речення має починатися з нового рядка і перше слово писатися з великої літери. На основі файлу «24.txt» створити також файл «242.txt», кожну приголосну букву у якому замінити випадково вибраною голосною з наступного слова, а кожну голосну – випадково вибраною приголосною з попереднього слова.
25	Як контрольний приклад створити файл v25.txt, який складається з трьох речень, та продемонструвати на ньому виконання завдання. Зчитати файл «25.txt» та перетворити його у файл «251.txt», що складається з речень, які утворені шляхом виконання операції XOR множин букв парних і непарних слів у кожному реченні (1 з 2, 3 з 4 і т. д.). Кожне таке речення має починатися з нового рядка і перше слово писатися з великої літери. На основі файлу «25.txt» створити також файл «252.txt», у якому букви замінити символами, які утворені шляхом виконання побітової операції «&» над кодами сусідніх букв в кодуванні cp1251. Результат вивести у кодуванні utf-8.
26	Як контрольний приклад створити файл v25.txt, який складається з трьох речень, та продемонструвати на ньому виконання завдання. Зчитати файл «26.txt» та перетворити його у файл «261.txt», що складається з речень, які утворені шляхом виконання операції OR над множинами букв парних слів у кожному реченні (2 з 4, 6 з 8 і т.д.). Кожне таке речення має починатися з нового рядка і перше слово писатися з великої літери. На основі файлу «26.txt» створити також файл «262.txt», у якому букви замінити символами, які утворені шляхом виконання побітової операції зсуву вправо на дві позиції в кодуванні cp1251. Результат вивести у кодуванні utf-8.
27	Як контрольний приклад створити файл v27.txt, який складається з трьох речень та продемонструвати на ньому виконання завдання. Зчитати файл «27.txt» та розбити його на 2 файли «271.txt», «272.txt», які містять непарні та парні за номером символи початкового файлу. Виконати операцію «&» над множинами символів цих файлів. Зберегти результуючий файл «273.txt» у кодуванні «cp1251». Створити файл з символів, які присутні у файлі «27.txt», але відсутні у «273.txt». Зберегти цей файл у кодуванні UTF-8.