

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ  
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ  
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ  
ІМЕНІ ІГОРЯ СІКОРСЬКОГО»

Кафедра прикладної математики

**КУРСОВА РОБОТА**  
з дисципліни «Програмування»

на тему:  
Шифрування і дешифрування тексту

Виконав:  
студент I курсу групи КМ-31,  
спеціальність 113  
Прикладна математика  
Медар Кирило Максимович

Оцінка: \_\_\_\_\_

Кількість балів: \_\_\_\_\_

## ЗМІСТ

ВСТУП.....	3
1 ПОСТАНОВКА ЗАДАЧІ.....	4
2 РОЗРОБКА ПРОГРАМНОГО ПРОДУКТУ.....	5
2.1 Структура програми.....	5
2.2 Опис використаних алгоритмів.....	6
3 ВІДЛАГОДЖЕННЯ.....	10
3.1 Перевірка виконання програми на витік пам'яті.....	10
4 ВИКОРИСТАННЯ ПРОГРАМИ.....	11
4.1 Взаємодія користувача із програмою.....	11
4.2 Тестові приклади.....	11
5 РЕЗУЛЬТАТИ РОБОТИ. КОНТРОЛЬНІ ПРИКЛАДИ.....	14
5.1 Тестування функціоналу для інших студентів.....	14
5.2 Код-рев'ю для інших студентів.....	14
5.3 Порівняння продуктивності із аналогічною програмою на Python.....	14
ВИСНОВКИ.....	18
ПЕРЕЛІК ПОСИЛАНЬ.....	19
ДОДАТОК.....	20

## ВСТУП

Проблема захисту інформації шляхом її перетворення, що виключає її прочитання сторонньою особою, хвилювала людський розум з давніх часів. Історія криптографії ровесниця історії людського мовлення та спілкування. Як приклад, спочатку писемність сама по собі була криптографічною технологією, оскільки в древніх суспільствах нею оволоділи лише певні версти населення.

В сучасному світі, в епоху цифрових технологій, обмін інформацією значно спростився. Але разом з цим з'явилась й купа нових способів заволодіти цією інформацією, тому все більше людей починають боятись за безпеку своїх особистих даних, таємницю листування, тощо. Шифрування даних є одним зі способів вирішити цю проблему, оскільки тоді зломисник не зможе отримати жодної користі з викрадених даних.

Особливо гостро проблема безпеки даних постає у воєнний період, коли для ефективного ведення бойових необхідна постійна, безперервна та швидка комунікація між різними підрозділами. Якщо супротивник отримає доступ до цих даних, це може надати йому неймовірної переваги на тактичному рівні, як приклад – взламвання союзниками німецької шифрувальної машини Enigma часів Другої світової війни.

Оскільки будь яку інформацію можна представити в текстовому вигляді або звести до нього, то задачу шифрування даних можна звести до задачі шифрування тексту. Тоді користувачем відповідної програми може бути будь яка людина, що хоче зашифрувати або дешифрувати певну інформацію.

Відповідно метою даної курсової роботи є створення власного алгоритму шифрування тексту. Метою даної програми є реалізація алгоритму шифрування та надання можливості людям додатково захистити свої дані для їх подальшого збереження, передачі тощо.

## 1 ПОСТАНОВКА ЗАДАЧІ

Формат вхідних та вихідних даних:

1. В режимі шифрування програма приймає лише символи стандарту ASCII. Вивід складається з чисел.
2. В режимі дешифрування для введення ключа лише символи стандарту ASCII, для введення шифру лише цифри. Вивід може складатись лише з символів стандарту ASCII.

Шлях введення вхідних даних: Через текстовий ввід у консоль.

Шлях виведення вихідних даних: Через текстове повідомлення в консолі.

## 2 РОЗРОБКА ПРОГРАМНОГО ПРОДУКТУ

### 2.1 Структура програми

Програма складається з основної функції `main`, в якій і відбуваються процеси шифрування та дешифрування, та кількох допоміжних функцій: `input_int`, `input_str`, `max_in_array`, `len_of_num`, `check_digit`.

#### 1) Функція `main`

На початку функції константою задається максимальна кількість символів для вводу та запускається основний цикл виконання програми. В цьому циклі на кожній ітерації виводиться головне меню програми, після чого користувачу пропонується зашифрувати текст, розшифрувати текст або вийти з програми. Далі, в залежності від вибору користувача, програма виконує відповідний блок функції, та, якщо було обрано не вихід з програми, повторює ітерацію.

Якщо було обрано зашифрувати текст, програма просить ввести ключ для шифрування. Далі запускається цикл під час якого користувач вводить текст який, якщо це не “END”, буде зашифровано за алгоритмом VeLKoM із врахуванням введеного ключа. За кожен ітерацію шифрується один введений в консоль рядок, після чого програма виводить зашифровану версію цього тексту і цикл починається спочатку. Так продовжується доти, доки користувач не введе “END”, після чого цикл закінчується, а разом з ним і блок шифрування тексту.

Якщо було обрано розшифрувати текст, програма просить ввести ключ для дешифрування. Далі запускається цикл під час якого користувач вводить текст який, якщо це не “END”, буде дешифровано за алгоритмом VeLKoM із врахуванням введеного ключа. За кожен ітерацію дешифровано один введений в консоль рядок, після чого програма виводить дешифровану версію цього шифру і цикл починається спочатку. Так продовжується доти,

доки користувач не введе “END”, після чого цикл закінчується, а разом з ним і блок шифрування тексту.

Якщо ж було обрано вихід, то цикл зупиняється, і з відповідним повідомленням програма завершує роботу.

#### 2) Функція `input_int`

Функція для вводу цілого числа у змінну, використовується в програмі для вибору завдання – розшифрувати, зашифрувати чи завершити роботу.

#### 3) Функція `input_str`

Функція для вводу рядкового типу даних, що складається з символів таблиці ASCII в масив, використовується в програмі для введення ключів та для введення тексту для шифрування.

#### 4) Функція `max_in_array`

Функція для пошуку найбільшого елемента за таблицею ASCII в масиві, використовується в програмі для пошуку найбільшого елемента в ключах, що необхідно для обчислення максимальної довжини шифру, що відповідає одному вхідному символу тексту.

#### 5) Функція `len_of_num`

Функція для обчислення кількості цифр у числі, використовується в програмі для обчислення кількості цифр.

#### 6) Функція `check_digit`

Функція для перевірки чи є всі елементи масиву числами, використовується в програмі для перевірки вводу шифру для дешифрування.

## 2.2 Опис використаних алгоритмів

Розглянемо алгоритми шифрування та дешифрування, для зручності назвемо їх  $F$  та  $F^{-1}$  відповідно. Спільною частиною цих алгоритмів є обробка ключа яка відбувається один раз перед всіма іншими операціями, тож спочатку опишемо цю частину, далі опишемо  $F$  та  $F^{-1}$  окремо.

Перед шифруванням та дешифруванням визначаємо довжину кожного числа у шифрі. Спочатку аналізуємо введений ключ та серед всіх його елементів обираємо найбільший і записуємо його кодування. Далі до кількості символів у кодуванні помноженої на 3 додаємо найбільше кодування ключа помножене на 2, оскільки code може використовуватись максимум тричі, а ключ двічі за одну ітерацію. Отримане число є найбільшим можливим числом в шифрі при даному ключі. Підраховуємо кількість цифр в ньому, та записуємо в змінну length.

code є внутрішньою константою програми, що використовується в якості «маски» для шифру, тобто за її допомогою кожен, відповідний для оригінального тексту, блок шифру має однакову кількість символів. Технічно code можна використовувати як додатковий ключ при шифруванні, оскільки при різних значеннях цієї константи, один і той же текст при одному й тому ж ключі буде шифруватись по різному. Але в такому разі code має специфічний діапазон значень при яких програма б працювала коректно, тому користувачу не надається можливість її змінювати.

Алгоритм функції  $F$  при шифруванні:

1. За розташуванням у ключі беремо два символи, переводимо їх у числа відповідно до обраного кодування, та позначаємо їх як key1 та key2 відповідно, та додаємо до обох code.
2. За розташуванням у вхідному тексті беремо два символи, переводимо їх у числа відповідно до обраного кодування, та позначаємо їх як a та b відповідно.
3. Обчислюємо перший елемент шифротексту k за формулою:  $k=(a+b)/2$ . Якщо k не ціле, округляємо до меншого цілого.
4. Обчислюємо другий елемент шифротексту m за формулою:  $((b-a)/2)+key2+code$ . Якщо m не ціле, округляємо до меншого цілого.
5. Перевіряємо a та b на однакову парність, для цього значення m після округлення порівнюємо зі значенням до округлення. Якщо вони

співпадають, то  $a$  та  $b$  на однакову парність, інакше потрібно це позначити. Для цього до  $m$  додаємо  $\text{code} * 2$ , тоді значення  $m$  при непарному  $a+b$  буде значно більшим ніж при парному, що можна побачити при розшифруванні. ( $\text{code} * 2$  додається для повного уникнення проблем колізії)

6. Додаємо до  $k$  та  $m$  значення  $\text{key}_1$  та  $\text{key}_2$  відповідно, та переводимо у символи за кодуванням.
7. Якщо довжина  $k$  або  $m$  менша за  $\text{length}$ , дописуємо перед відповідним числом 0 в кількості якої недостатньо для рівності з  $\text{length}$ .

Повторюємо операцію  $f$  для всіх інших пар символів вхідного тексту.

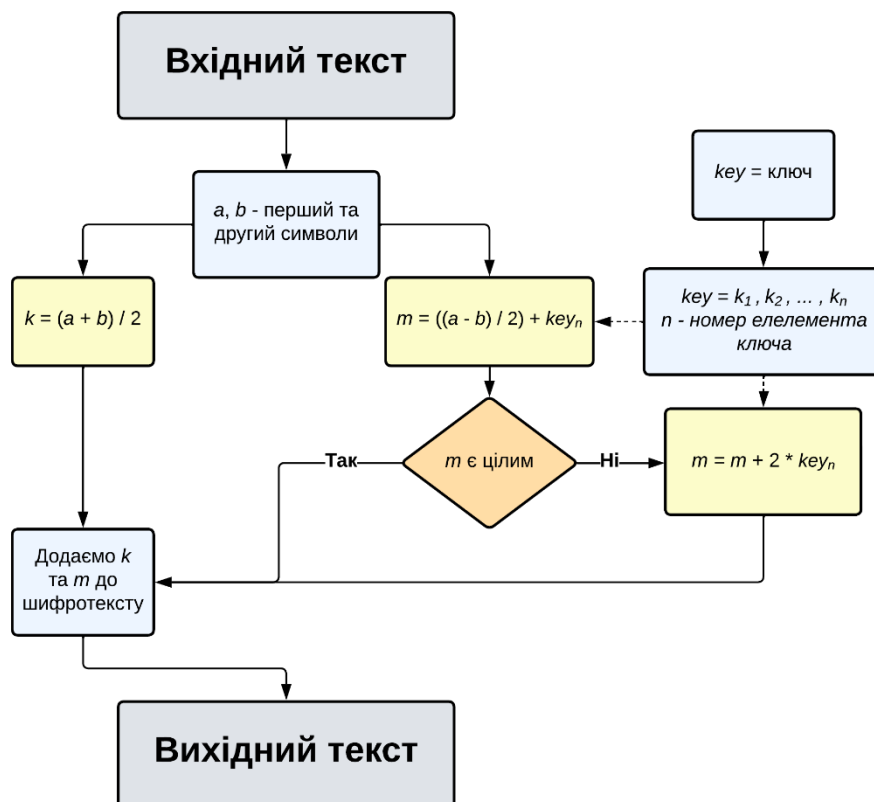


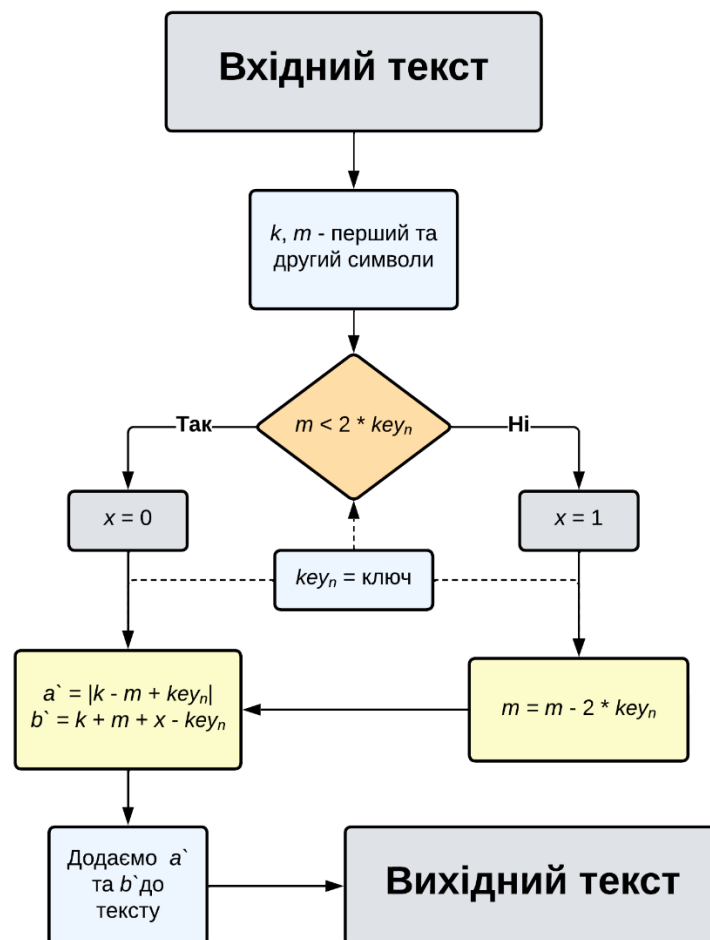
Рисунок 1 – Алгоритм  $F$

Алгоритм функції  $F'$  при розшифруванні:

1. За розташуванням у ключі беремо два символи, переводимо їх у числа відповідно до обраного кодування, та позначаємо їх як  $\text{key}_1$  та  $\text{key}_2$  відповідно, та додаємо до обох  $\text{code}$ .

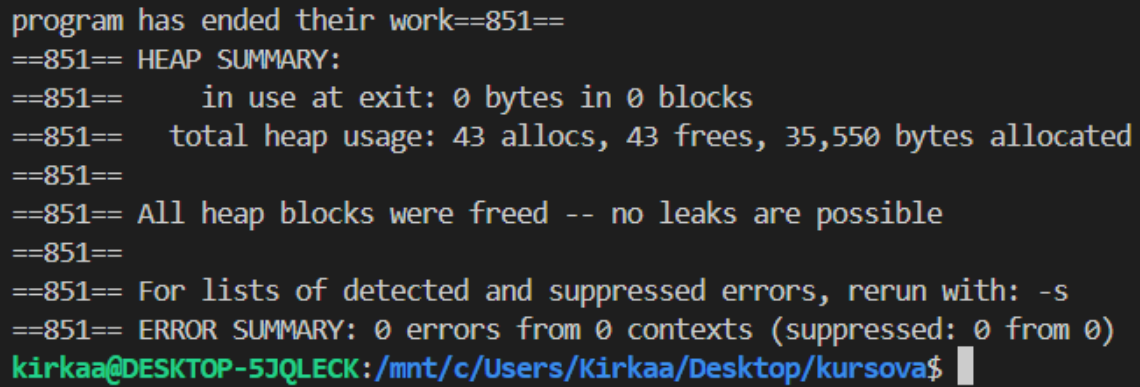


2. Розбиваємо шифротекст на числа довжини  $\text{length}$ , позначаємо їх як  $k$  та  $m$  відповідно, прибираємо зайві 0. Віднімаємо  $\text{key1}$  та  $\text{key2}$  від  $k$  та  $m$  відповідно. Від  $m$  додатково віднімається  $\text{code}$ .
3. Якщо  $m$  менше або рівне  $\text{code} * 2$  - позначаємо  $x=0$  та переходимо до наступного пункту, інакше від  $m$  віднімаємо  $\text{code} * 2$ , та позначаємо  $x=1$ .
4. Обчислюємо перший елемент вихідного тексту  $a'$  за формулою:  
 $a' = |k - m + \text{key2}|$ .
5. Обчислюємо другий елемент вихідного тексту  $b'$  за формулою:  
 $b' = k + m + x - \text{key2}$ .
6. Переводимо  $a'$  та  $b'$  у символи за кодуванням, та записуємо до вихідного тексту.
7. Повторюємо операцію  $F'$  для всіх інших пар символів шифротексту.

Рисунок 2 – Алгоритм  $F'$

## 3 ВІДЛАГОДЖЕННЯ

### 3.1 Перевірка виконання програми на витік пам'яті



```
program has ended their work==851==
==851== HEAP SUMMARY:
==851==    in use at exit: 0 bytes in 0 blocks
==851== total heap usage: 43 allocs, 43 frees, 35,550 bytes allocated
==851==
==851== All heap blocks were freed -- no leaks are possible
==851==
==851== For lists of detected and suppressed errors, rerun with: -s
==851== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
kirkaa@DESKTOP-5JQLECK:/mnt/c/Users/Kirkaa/Desktop/kursova$
```

Рисунок 3 – Перевірка на витік пам'яті

За один запуск програми було зашифровано два повідомлення та потім, за цю ж сесію їх було дешифровано та обрано вихід з програми. Перевірка на витік пам'яті за допомогою Valgrind показала результати, що наведені на Рисунку 3.

## 4 ВИКОРИСТАННЯ ПРОГРАМИ

### 4.1 Взаємодія користувача із програмою

Після запуску програми, виводиться головне меню, згідно з яким можна обрати шифрування, дешифрування або вийти з програми, ввівши 1, 2 або 0 відповідно.

Якщо введено 1, необхідно ввести ключ, який складається лише з символів ASCII. Далі відбувається ввід тексту, який програма одразу шифрує та виводить відповідний шифр. Це відбувається доки користувач не введе END, після чого він повернеться до головного меню. Якщо користувач захоче шифрувати іншим ключем, йому необхідно вийти в головне меню і ще раз обрати шифрування, після чого ввести новий ключ.

Якщо введено 2, необхідно ввести ключ, який складається лише з символів ASCII. Далі відбувається ввід шифру, який програма одразу дешифрує та виводить відповідний текст. Це відбувається доки користувач не введе END, після чого він повернеться до головного меню. Якщо користувач захоче дешифрувати за іншим ключем, йому необхідно вийти в головне меню і ще раз обрати дешифрування, після чого ввести новий ключ.

Якщо введено 0, програма завершує роботу і виводить відповідне повідомлення.

### 4.2 Тестові приклади

Тестові приклади шифрування та дешифрування програми повідомлень наведено на рисунках 4 – 5.

```

1 - to encrypt
2 - to decrypt
0 - exit from program

input num: 1

Your input: 1

input key:
Input text (you can write ASCII symbols): love <3

Your input: love <3

input text to encrypt [END to exit]
Input text (you can write ASCII symbols): its some text with #${}[]().?/\,><

Your input: its some text with #${}[]().?/\,><

The text for sifer: 111228255605471112255556048011122093336310111222033364041112249333647011122755560393111219833362521112284333
6407111229233363881112129556044111216533364221112253333647311122733336239111219055603951112215556054411122373336386111216433362
8811121763336407

input text to encrypt [END to exit]
Input text (you can write ASCII symbols): and another 123456789

Your input: and another 123456789

The text for sifer: 111227555605481112248333636011121995560446111222855605381112277556055411122383336215111217355604221112223556
0542111223555605221112151556044011121595560523

```

Рисунок 4 – Шифрування кількох повідомлень

```

input num: 2

Your input: 2

input key:
Input text (you can write ASCII symbols): love <3

Your input: love <3

input text to decrypt [END to exit]
Input text (you can write ASCII symbols): 11122825560547111225555604801112209333631011122203336404111224933364701112275556039311
1219833362521112284333640711122923336388111212955604411121653336422111225333364731112273333623911121905560395111221555605441112
23733363861112164333628811121763336407

Your input: 11122825560547111225555604801112209333631011122203336404111224933364701112275556039311121983336252111228433364071112
2923336388111212955604411121653336422111225333364731112273333623911121905560395111221555605441112237333638611121643336288111217
63336407

The decipher: its some text with #${}[]().?/\,><

input text to decrypt [END to exit]
Input text (you can write ASCII symbols): 11122755560548111224833363601112199556044611122285560538111227755605541112238333621511
12173556042211122235560542111223555605221112151556044011121595560523

Your input: 11122755560548111224833363601112199556044611122285560538111227755605541112238333621511121735560422111222355605421112
23555605221112151556044011121595560523

The decipher: and another 123456789

```

Рисунок 5 – Дешифрування кількох повідомлень

Окремо спробуємо зашифрувати повідомлення одним шифром, а розшифрувати іншим, тоді при дешифруванні маємо отримати «випадковий» набір символів, результати на рисунках 6 – 7.

```

input key:
Input text (you can write ASCII symbols): right

Your input: right

input text to encrypt [END to exit]
Input text (you can write ASCII symbols): Hello world

Your input: Hello world

The text for sifer: 111226455605441112275333640011122515560508111228433363941112279333642111122443336368

```

Рисунок 6 – Шифрування повідомлення ключем right

```

input key:
Input text (you can write ASCII symbols): wrong

Your input: wrong

input text to decrypt [END to exit]
Input text (you can write ASCII symbols): 111226455605441112275333640011122515560508111228433363941112279333642111122443336368

Your input: 111226455605441112275333640011122515560508111228433363941112279333642111122443336368

The decipher: UNpX#~VRq

```

Рисунок 7 – Спроба розшифрувати повідомлення не правильним ключем

Програма намагається розшифрувати текст і виводить певний результат, але він, як і очікувалось, є хибним. Слід відмітити, що в хибному результаті з'явилися спеціальні символи, яких не було у вхідному тексті. Тобто при спробі розшифрувати повідомлення хибним ключем, можуть з'являтися будь-які символи таблиці ASCII, що потенційно додає стійкості шифру при спробі їх дешифрувати сторонніми особами.

## 5 РЕЗУЛЬТАТИ РОБОТИ. КОНТРОЛЬНІ ПРИКЛАДИ

### 5.1 Тестування функціоналу для інших студентів.

Мною було проведено тестування роботи на тему «Задача про n ферзів» Дорошенко Романа. В ході тестування було виявлено, що при певній комбінації вхідних даних, у програмі відбувався непередбачуваний збій, через що вона припиняла свою роботу.

### 5.2 Код-рев'ю для інших студентів

Мною було проведено код-рев'ю роботи на тему «Задача про n ферзів» Дорошенко Романа. В ході чого було запропоновано кілька рекомендацій щодо загальної читабельності коду, та запропоновано змінити функцію для вводу даних, через недоліки функції `scanf`.

### 5.3 Порівняння продуктивності із аналогічною програмою на Python

Алгоритм було реалізовано мовою Python 3.9, код наведено в додатку, рисунок А.

Тестування розділимо на кілька етапів. Спочатку зробимо однакові вводи даних та порівняємо результати роботи програм. Другим тестом зашифруємо дані програмою на мові C, та спробуємо розшифрувати на Python. Третім тестом зашифруємо дані програмою на мові Python, та спробуємо розшифрувати на C.

Перше тестування

Ключ: Some key

Перший текст: Hello world

Другий текст: Some text!!!\$#%15454

Третій текст:  $2 + 2 = 4$

Шифри та дешифровки мовою C представлено на рисунках 8 та 9 відповідно:

```

Your input: Hello world

The text for sifer: 111223355605561112281333639411121675560494111228033364301112258333641111122393336360

input text to encrypt [END to exit]
Input text (you can write ASCII symbols): Some text!!!$#%15454

Your input: Some text!!!$#%15454

The text for sifer: 11122443336428111227833363901112170333644811122755560571112221556050011122063336394111213155605331112208333
64401112199556054111122255560521

input text to encrypt [END to exit]
Input text (you can write ASCII symbols): 2 + 2 = 4

Your input: 2 + 2 = 4

The text for sifer: 1112188333640511122105560516111213733363971112211556054711121893336404

```

Рисунок 8 – Шифри мовою С

```

input text to decrypt [END to exit]
Input text (you can write ASCII symbols): 111223355605561112281333639411121675560494111228033364301112258333641111122393336360

Your input: 111223355605561112281333639411121675560494111228033364301112258333641111122393336360

The decipher: Hello world

input text to decrypt [END to exit]
Input text (you can write ASCII symbols): 1112244333642811122783336390111217033364481112275556057111222155605001112206333639411
121315560533111220833364401112199556054111122255560521

Your input: 1112244333642811122783336390111217033364481112275556057111222155605001112206333639411121315560533111220833364401112
19955605411122255560521

The decipher: Some text!!!$#%15454

input text to decrypt [END to exit]
Input text (you can write ASCII symbols): 1112188333640511122105560516111213733363971112211556054711121893336404

Your input: 1112188333640511122105560516111213733363971112211556054711121893336404

The decipher: 2 + 2 = 4

```

Рисунок 9 – Дешифрування мовою С

Шифри та дешифровки мовою Python представлено на рисунках 10 та 11 відповідно:

```

Hello world
Результат:
111223355605561112281333639411121675560494111228033364301112258333641111122393336360
Some text!!!$#%15454
Результат:
111224433364281112278333639011121703336448111227555605711122215560500111220633363941112131556053311122083336440111219955605411122255560521
2 + 2 = 4
Результат:
1112188333640511122105560516111213733363971112211556054711121893336404

```

Рисунок 10 – Шифри мовою Python

```

Введіть ключ: Some key
Введіть текст для розшифрування:
111223355605561112281333639411121675560494111228033364301112258333641111122393336360
Результат:
Hello world
111224433364281112278333639011121703336448111227555605711122215560500111220633363941112131556053311122083336440111219955605411122255560521
Результат:
Some text!!!$#%15454
1112188333640511122105560516111213733363971112211556054711121893336404
Результат:
2 + 2 = 4

```

Рисунок 11 – Дешифрування мовою Python

Тестування успішне.

Друге тестування

Ключ: Key for C!

Текст: Hi, i`m python and I can decrypt this)

Шифр мовою C представлено на рисунку 12:

```
Your input: Hi, i`m python and I can decrypt this)

The text for sifer: 11122275560538111222333625011122665560537111224855603451112247556039011122493336388111229555603831112230556
057411122833336251111218355604061112204556055511122885560390111223233364481112278333362551112248556038911122533336396111225933362
981112270556054211122563336219
```

Рисунок 12– Шифр для Python

Дешифрування мовою Python представлено на рисунку 13:

```
Введіть ключ: Key for C!
Введіть текст для розшифрування:
111222755605381112223336250111226655605371112248556034511122475560390111224933363881112295556038311122305560574111228333362511122183556040611122045560555112288556039011122
323336448111227833362551112248556038911122533336396111225933362981112270556054211122563336219
Результат:
Hi, i`m python and I can decrypt this)
```

Рисунок 13 – Дешифрування мовою Python повідомлення від C

Тестування успішне.

Третє тестування

Ключ: its #52?

Текст: The rose is red, the violet's blue, The honey's sweet, and so are you.

Thou are my love and I am thine; I drew thee to my Valentine: The lot was cast  
and then I drew, And Fortune said it shou'd be you.

Шифр мовою Python представлено на рисунку 14:

```
Введіть ключ: its #52?
Введіть текст для шифрування:
The rose is red, the violet's blue, The honey's sweet, and so are you. Thou are my love and I am thine; I drew thee to my Valentine: The lot was cast and then I drew, And
Fortune said it shou'd be you.
Результат:
111226333643411122455560349111221155604241112223336311112237556058811122525560342111220655604191112218633362901112243333646611122815560382111221743336341112222333632111222,
7355605481116345556443411122172556038411122173336323111227833364161112217333625011121933336380811121805560411112276556055511228455603791112628334034611121875560404111228633,
364261112280333625611121793336262111217855604781112274333641911122525560425111217055603861112219556045411122355560517111229533362511121805560390111217233363441112276556055,
511122535560341112204556043411121805560411112284333643011122493336294111221355604291112180556041111227255605581112245333622211121515560405111221733363241112243336466111,
228355603841112204556042111215955604321112215560531112286333626311122093336307111218833363601112271556055011122455560349111221255604231112184556048411122455560507112270,
556038911122035560422111227333632111227655605411122585560362111215733363421112216556046411122393336462111229255603861112174556046911122203336327111223455605851122853336,
265111217333625611122175560452111223533363901112289333625011122045560430111216655604661112235333645811122865560377111218055603881112162556046211122743336419111223033362751,
112211556042711122305560446111227455605471112252556042511122083336392111218033362841112279556055711122525560425111220655604291116281334036811122353363901112278556038511121,
755560470111222833363211122083336417
```

Рисунок 14 – Шифр для C

Дешифрування мовою C представлено на рисунку 15:



```

Your input: 1112263336434111224556034911122115560424111222333631111223755605881112252556034211122065560419111218633362901112
243333646611122815560382111217433363411122223336321112273556054811163455564434111217255603841112217333632311122783336416111221
733362501112193333630811121805560411112276556055511228455603791116268334034611121875560404111228633364261112280333625611121793
3362621112178556047811122743336419111225255604251112170556038611122195560454111223555605171112295333625111218055603901112172333
63441112276556055511122535560341112204556043411121805560411112284333643011122493336294111221355604291112180556041111227255605
5811122453336222111215155604051112217333632411122433336466111228355603841112204556042111215955604321112221556053111122863336263
11122093336307111218833363601112271556055011122455560349111221255604231112184556048411122455560507111227055603891112203556042211
1222733363211122765560554111225855603621112157333632411122165560444111223933364621112292556038611121745560469111222033363271112
234556058511122853336265111217333625611122175560452111223533363901112289333625011122045560430111216655604661112235336458111228
65560377111218055603881112162556046211122743336419111223033362751112211556042711122305560446111227455605471112252556042511122003
33630211121803336284111227955605571112252556042511122065560429111628133403681112235333639011122785560385111217555604701112228333
63211122083336417

The decipher: The rose is red, the violets blue, The honeys sweet, and so are you. Thou are my love and I am thine; I drew the
e to my Valentine: The lot was cast and then I drew, And Fortune said it shoud be you.

```

Рисунок 15 – Дешифрування мовою С повідомлення від Python

Тестування успішне.

Отже, незважаючи на різні мови програмування, програми є повністю сумісними та однаково виконують поставлену задачу.

Відмінності реалізації алгоритмів.

Попри різні мови програмування, глобально реалізація алгоритму обома мовами була ідентичною. Якщо опустити технічні відмінності типу відсутності garbage collector в С, через що для виклику динамічного масиву потрібно писати кілька сторінок коду, там де в Python це займає одну, а дивитись лише на алгоритм, то різниця повністю відсутня, що також видно й по результатам роботи. Ледь не єдиною відмінністю може стати більша швидкодія мови програмування С, і, відповідно алгоритм цією мовою буде швидшим.

Висновок: Обидві програми однаково правильно виконують поставлену задачу, проте алгоритм реалізований мовою програмування С має більшу швидкість, тому є кращим, за аналогічний алгоритм мовою Python.

## ВИСНОВКИ

Отже було наведено та реалізовано мовою C вславний алгоритм шифрування. Головними складнощами стала погана сумсність мови C із кодуванням utf-8, яке за першоїдеєю мало головним, замість ASCII.

Наведений алгоритм шифрування може працювати з будь-якою таблицею кодувань. Єдине, що потрібно буде замінити в кодї – функцію для переведення символу у відповідне число за кодуванням. У випадку з Python тут ніяких проблем немає ось у випадку з C все не так просто. Наприклад деякі символу мають тип `char`, а деякі – `wchar`, і відповідно вже з'являється нова задача: написати функцію, яка зможе обробляти обидва ці типи даних одночасно. Додатково можна додати, що C в цілому за замовченням працює саме з ASCII, де кількість можливих символів для шифрування значно менша, ніж в тому ж самому utf-8. В цілому базовою перевагою Python є наявність `garbage collector` та динамічна типізація даних, чого немає в C, і через що довелося писати «зайві» відносно Python строки коду.

Результати тестувань показали стабільну роботу програми при правильно введених даних.

Для покращення програми можна збільшити можливу кількість символів, що може обробити програму. Для цього можна замінити ASCII на ту ж саму utf-8, або навіть написати власну таблицю символів. В останньому варіанті додатково збільшиться й стійкість шифру, оскільки, наприклад якщо намагатись взламати такий шифр методом грубої сили, до кожної можливої розшифровки доведеться застосовувати частотний аналіз, що додатково збільшить необхідні обчислення. Збільшення таблиці можливих символів буде зручним також і для користувача, оскільки це наприклад дасть можливість шифрувати кількома мовами одночасно.

Перевагою програми є відносна простота обчислень і в той же час стійкість шифру до частотного аналізу. Недоліком можна назвати використання таблиці ASCII.

## ПЕРЕЛІК ПОСИЛАНЬ

1. Дробиш Р. С. Аналіз існуючих методів захисту та шифрування інформації / Р. С. Дробиш, М. В. Люта // Інноватика в освіті, науці та бізнесі: виклики та можливості : матеріали II Всеукраїнської конференції здобувачів вищої освіти і молодих учених, м. Київ, 18 листопада 2021 року. – Т. 1. – Київ : КНУТД, 2021. 250-253 с. URL:  
<https://er.knutd.edu.ua/handle/123456789/19525>
2. Худякова К. Ю. Захист конфіденційної інформації в офісній мережі із застосуванням криптографічного методу : магістерська дис. : 171 Електроніка. КПІ ім. Ігоря Сікорського. Київ, 2019. 87 с. URL:  
<https://ela.kpi.ua/handle/123456789/30851>
3. Шифрування методом заміни. URL:  
[http://ni.biz.ua/11/11\\_4/11\\_46236\\_shifrovanie-metodom-zameni-podstanovki.html](http://ni.biz.ua/11/11_4/11_46236_shifrovanie-metodom-zameni-podstanovki.html)

## ДОДАТОК

```

while True:
    print("1 - to encrypt \n2 - to decrypt \n0 - exit from program")
    task = input("input num: ")
    if int(task) == 1:
        keys = list(input('Введіть ключ: '))
        print('Введіть текст для шифрування: ' '')
        code = 1112064
        length = len((str(code * 3 + 2 * ord(sorted(list(keys))[-1])))) # Задання довжини числа в ключі
        while True:
            i = 0
            inp = list(input())
            out = ''
            if inp == ['E', 'N', 'D']:
                break
            if len(keys) < len(inp):
                keys = keys + keys * int(len(inp) / len(keys)) # Перевірка та побудова ключа
            if len(inp) % 2 == 1: # Перевірка повідомлення на парну кількість символів
                inp += ' ' # Якщо непарна - в кінець додаємо "Пробіл"
            while True:
                if i + 1 >= len(inp):
                    break
                key1 = ord(keys[i]) + code # Пункт 1
                key2 = ord(keys[i + 1]) + code
                a = int(ord(inp[i])) # Пункт 2
                b = int(ord(inp[i + 1]))
                k = int((a + b) / 2) # Обчислення за пунктами 3 та 4
                m = int(code + key2 + ((b - a) / 2))
                if m != code + key2 + ((b - a) / 2): # Пункт 5
                    m += code * 2
                k = str(k + key1) # Пункт 6
                m = str(m + key2)
                out += "".join(["0" for _ in range(int(length) - len(k))]) + k # Пункт 7
                out += "".join(["0" for _ in range(int(length) - len(m))]) + m
                i += 2
            print('Результат:')
            print(out)
        elif int(task) == 2:
            keys = list(input('Введіть ключ: '))
            print('Введіть текст для розшифрування: ' '')
            code = 1112064
            length = len((str(code * 3 + 2 * ord(sorted(list(keys))[-1])))) # Задання довжини числа в ключі
            while True:
                i = 0
                _ = 0
                inp = str(input())
                if inp == 'END':
                    break
                inp = tuple(map("".join, zip(*[iter(inp)] * length))) # Розбиття шифротексту на числа
                out = ''
                x = 0
                if len(keys) < len(inp):
                    keys = keys + keys * int(len(inp) / len(keys)) # Перевірка та побудова ключа
                while True:
                    if i + 1 >= len(inp):
                        break
                    key1 = ord(keys[i]) + code # Пункт 1
                    key2 = ord(keys[i + 1]) + code
                    k = int(inp[i]) - key1 # Пункт 2
                    m = int(inp[i + 1]) - key2 - code
                    m = int(inp[i + 1]) - key2 - code
                    x = 0
                    if m > code * 2: # Пункт 3
                        m -= code * 2
                        x = 1
                    a = abs(int(k - m + key2)) # Обчислення за пунктами 4 та 5
                    b = int(k + m + x - key2)
                    out += str(chr(a)) # Пункт 6
                    out += str(chr(b))
                    i += 2
                print('Результат:')
                print(out)
            elif int(task) == 0:
                break
            else:
                print("invalid input")
        print("program has ended")

```

Рисунок А