Завдання №1: Реалізація алгоритмів побайтного перетворення інформації

1 Мета роботи

Опанувати поняття коду, дослідити різні види перетворень даних, представлених двійковими словами (байтами).

2 Завдання роботи

В залежності від варіанту, вам необхідно реалізувати один з двох алгоритмів перетворення даних:

- кодування даних за `Base64`;
- стискання даних методом RLE.

3 `Base64`-кодування

3.1 Загальні положення

Перетворення `Base64` призначене для представлення потоку двійкових данних у шестибітовому алфавіті (64 символи). Таке представлення було необхідне тому, що в традиційному восьмибітовому алфавіті деякі байти мали значення, які інтерпретувались системами зв'язку як команди, і тому їх не можна було передавати безпосередньо. `Base64` використовує байтове представлення для символів, але в алфавіті фігурують лише так звані *текстові* символи: літери латинської абетки, цифри та деякі знаки пунктуації.

У рамках даного завдання вам необхідно реалізувати класичний `Base64`-кодер, який використовує абетку

```
ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+\
```

та два спецсимволи:
'=' — заповнювач для паддінгу наприкінці закодованого повідомлення;
'-' — символ початку коментаря.

Кодування відбувається таким чином:

- вхідний потік байтів розбивається на фрагменти по три послідовні байти, тобто 24 біти;
- кожна 24-бітова послідовність перерозбивається на чотири шестибітові фрагменти;

- кожен шестибітовий фрагмент трактується як номер в абетці кодера та замінюється на відповідний символ абетки; таким чином, вхідна послідовність з трьох байтів замінюється на закодовану послідовність з чотирьох байтів.

Останній фрагмент вхідного потоку може складатись не з трьох, а з двох чи одного байту. У такому випадку вхідні байти теж перерозбиваються на шестибітові фрагменти із доповненням нульовими бітами (два байти перетворюються на три фрагменти, один байт — на два фрагменти), а до вихідної послідовності додається один чи два спецсимволи '=', щоб вихідна послідовність завжди була з чотирьох символів.

3.2 Вимоги до кодера

`Base64`-кодер повинен бути реалізований як окрема програма або як модуль в межах більш функціональної програми, який викликається спеціальною командою (з інтерфейсу програми, з меню чи з командного рядку консолі).

1. Кодер повинен приймати на вхід ім'я файла для кодування та (опціонально) ім'я файла для зберігання закодованого результату.

2. Якщо користувач не надав ім'я файла для зберігання закодованого результату, кодер повинен створити такий файл сам, додавши до імені оригінального файлу розширення '`.base64`'.

3. Вхідний файл повинен трактуватись як файл двійкових даних (потік байтів).

4. Результат кодування повинен записуватись у файл виходу як у текстовий файл, рядками по 76 символів (окрім, можливо, більш короткого останнього рядку).

5. Кодер може вставляли у файл виходу довільні коментарі (наприклад, метадані для читача). Кожен такий рядок повинен починатись з символу '`-`' та не повинен перевищувати 76 символів разом із початковим символом.

3.3 Вимоги до декодера

`Base64`-декодер повинен бути реалізований як окрема програма або як модуль в межах більш функціональної програми, який викликається спеціальною командою (з інтерфейсу програми, з меню чи з командного рядку консолі).

1. Декодер повинен приймати на вхід ім'я закодованого файлу та ім'я файла для зберігання декодованого результату.

2. (Опціонально) Декодер може запропонувати користувачу варіант імені для декодованого файлу: наприклад, якщо закодований файл має розширення '`.base64`' — відкинувши таке розширення; або, якщо закодований файл у метаданих зберігає ім'я оригінального файлу, — запропонувати збережене ім'я. Але, в будь-якому випадку, користувач повинен або сам зафіксувати ім'я файлу виходу, або свідомо погодитись із пропозицією такого імені.

3. Файл виходу повинен трактуватись як файл двійкових даних (потік байтів).

4. Вхідний файл повинен трактуватись як текстовий файл, що містить результати `Base64`-кодування та рядки-коментарі.

5. Усі рядки, які починаються з символу '-', повинні ігноруватись декодером. У всіх інших рядках наявність символу '-' не на першому місті повинна трактуватись як помилка вхідних даних «*Некоректний вхідний символ*».

6. Усі рядки файлу (окрім, можливо, останнього) повинні мати довжину 76 символів. Якщо рядок має іншу довжину, декодер повинен повернути помилку «*Некоректна довжина рядку*».

7. Якщо в рядку, який декодується, знайдено символ, який не належить алфавіту `Base64`, декодер повинен повернути помилку «*Некоректний вхідний символ*».

8. Символи '=', якщо вони є у закодованому повідомленні, означають завершення закодованого повідомлення. Якщо декодер знайшов символ '=', який не є останнім символом останнього закодованого рядку, він повинен повернути помилку «*Неправильне використання паддінгу*».

9. Якщо декодер зміг ідентифікувати останній рядок[1], але після нього ще йдуть якісь рядки, які не є коментарями, декодер повинен повернути попередження «*Наявні дані після кінця повідомлення*» і не декодовувати такі дані. При цьому результат декодування правильної частини вхідного файлу повинен бути записаний у файл виходу.

При виникненні будь-якої помилки декодер повинен зупинити роботу. Повідомлення про помилки «*Некоректний вхідний символ*» та «*Неправильне використання паддінгу*» повинні супроводжуватись номером рядку та позицією символу, в якому була виявлена помилка; повідомлення про помилку «*Некоректна довжина рядку*» повинно супроводжуватись номером рядку, в якому виникла помилка, та довжиною даного рядку:

```
Рядок 20, символ 02: Некоректний вхідний символ ('*')
Рядок 25, символ 44: Неправильне використання паддінгу
Рядок 36: Некоректна довжина рядку (75)
```

4 Алгоритм стискання RLE

4.1 Загальні положення

Алгоритм RLE (від англ. *Run Length Encoding*) — це алгоритм, який стискає дані шляхом заміни послідовностей однакових байтів на спеціальні записи виду «(довжина послідовності, значення байту)».

Вхідний потік байтів кодується таким чином:

- послідовність з ℓ однакових байтів "`a`, `a`, ..., `a`" (де $2 \le \ell \le 129$) замінюється на послідовність з двох байтів "L, `a`", де $L = 128 + \ell - 2$; іншими словами, байт L у двійковому записі має вид $\texttt{1}\|\texttt{xxxxxxx}$, де `xxxxxxx` — семибітове двійкове зображення числа $\ell - 2$;

- послідовність з ℓ різних байтів "`a`, `b`, ..., `z`" (де $1 \le \ell \le 128$) замінюється на послідовність з байтів "L, `a`, `b`, ..., `z`", де $L = \ell - 1$; іншими словами, байт L у двійковому записі має вид $\texttt{0}\|\texttt{xxxxxxx}$, де `xxxxxxx` — семибітове двійкове зображення числа $\ell - 1$.

[1]Зауважимо, що декодер не завжди може виявити останній рядок закодованого повідомлення. Явними ознаками такого рядку є довжина менше 76 символів та/або використання символів паддінгу. Однак повідомлення з 57 байтів буде закодоване у один рядок з 76 символів без паддінгу. У даному випадку єдине, що вкаже декодеру на завершення закодованого повідомлення — це раптовий кінець файлу

4.2 Вимоги до кодера

RLE-кодер (програма стискання даних) повинен бути реалізований як окрема програма або як модуль в межах більш функціональної програми, який викликається спеціальною командою (з інтерфейсу програми, з меню чи з командного рядку консолі).

1. Кодер повинен приймати на вхід ім'я файла для кодування та (опціонально) ім'я файла для зберігання закодованого результату.
2. Якщо користувач не надав ім'я файла для зберігання закодованого результату, кодер повинен створити такий файл сам, додавши до імені оригінального файлу розширення '`.rle`'.
3. Вхідний файл повинен трактуватись як файл двійкових даних (потік байтів).
4. Стиснені дані повинні записуватись у вихідний файл як у файл двійкових даних (потік байтів).
5. Кодер може вставляли у вихідний файл довільні необхідні йому метадані (наприклад, заголовки чи якісь службові команди, які ви передбачаєте своєю реалізацією).

Алгоритм RLE не є однозначним в тому сенсі, що заданим вхідним даним у багатьох випадках може відповідати декілька стиснених образів. У своїх реалізаціях ви можете передбачити деякі евристичні оптимізації, які покращують якість стиснення — наприклад, відслідковувати короткі серії однакових байтів, які погано стискаються, і замінювати їх на нестиснену послідовність.

4.3 Вимоги до декодера

RLE-декодер (програма, яка розпаковує дані в оригінальний нестиснений вид) повинен бути реалізований як окрема програма або як модуль в межах більш функціональної програми, який викликається спеціальною командою (з інтерфейсу програми, з меню чи з командного рядку консолі).

1. Декодер повинен приймати на вхід ім'я стисненого файлу та ім'я файла для зберігання розпакованого результату.
2. (Опціонально) Декодер може запропонувати користувачу варіант імені для декодованого файлу: наприклад, якщо закодований файл має розширення '`.rle`' — відкинувши таке розширення; або, якщо закодований файл у метаданих зберігає ім'я оригінального файлу, — запропонувати збережене ім'я. Але, в будь-якому випадку, користувач повинен або сам зафіксувати ім'я файлу виходу, або свідомо погодитись із пропозицією такого імені.
3. Файли входу та виходу повинні трактуватись як файли двійкових даних (потоки байтів).
4. Якщо ваша реалізація передбачає якісь метадані у стиснутому файлі, декодер повинен перевірити їх коректність та, у разі їх некоректності, вивести відповідне повідомлення про помилку.
5. Якщо декодер виявив неправильну команду алгоритму стиснення у вхідному файлі (команда каже зчитати послідовність байтів, яка виходить за кінець файлу), декодер повинен вивести відповідне повідомлення про помилку.
6. У випадку виникнення помилки декодування декодер може як повністю зупинити свою роботу, так і спробувати декодувати те, що зможе. Вибір поведінки залишається на ваш розсуд.

5 Вимоги до звіту

Результати вашої роботи подаються у вигляді коротенького звіту, який повинен включати опис та особливості вашої реалізації заданого алгоритму, а також приклади кодування, декодування та декодування спотворених даних.

Ваш програмний код традиційно повинен надаватись у вигляді посилання на GitHub-репозиторій.