МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждения образования

«БЕЛОРУССКИЙ

ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

**ОТЧЁТ**

**По лабораторной работе №10:**

**ТЕМА «**СЖАТИЕ/РАСПАКОВКА ДАННЫХ МЕТОДОМ ЛЕМПЕЛЯ − ЗИВА**»**

Минск 2022

**Цель**: приобретение практических навыков использования метод Лемпеля − Зива (Lempel-Ziv) для сжатия/распаковки данных.

**Задачи:**

1. Закрепить теоретические знания по алгебраическому описанию и использованию методов сжатия/распаковки (архивации/ разархивации) данных на основе метода Лемпеля − Зива.

2. Разработать приложение для реализации метода Лемпеля − Зива.

3. Результаты выполнения лабораторной работы оформить в виде описания разработанного приложения, методики выполнения экспериментов с использованием приложения и результатов эксперимента.

**Теоретические сведения**

Классический алгоритм Лемпеля − Зива – LZ77, названный так по году представления метода, формулируется следующим образом: «если в проанализированном (сжатом) ранее выходном потоке уже встречалась подобная последовательность байт, причем запись о ее длине и смещении от текущей позиции короче, чем сама эта последовательность, то в выходной файл записывается ссылка (смещение, длина), а не сама последовательность».

Суть метода LZ77 (как и последующих его модификаций) состоит в следующем: упаковщик постоянно хранит некоторое количество последних обработанных символов в буфере. По мере обработки входного потока вновь поступившие символы попадают в конец буфера, сдвигая предшествующие символы и вытесняя самые старые. Размеры этого буфера, называемого также скользящим словарем (англ. sliding dictionary), варьируются в разных реализациях систем сжатия. Скользящее окно имеет длину n, т. е. в него помещается n символов, и состоит из двух частей:

• последовательности длины n1 = n − n2 уже закодированных символов (словарь);

• упреждающего буфера (буфера предварительного просмотра, lookahead) длиной n2 – буфера кодирования.

*Нужно найти самое длинное совпадение между строкой буфера кодирования, начинающейся с символа St + 1, и всеми фразами словаря.*

Длина кодовой комбинации (триады – p, q, s) на каждом шаге определяется соотношением:

******

После каждого шага окно смещается на q + 1 символов вправо и осуществляется переход к новому циклу кодирования. Величина сдвига объясняется тем, что мы реально закодировали именно q + 1 символов: q – с помощью указателя и 1 − с помощью тривиального копирования. Передача одного символа в явном виде (s) позволяет разрешить проблему обработки еще ни разу не встречавшихся символов, но существенно увеличивает размер сжатого блока.

Практическое задание

У нас есть текстовый файл, который содержит имя и фамилию рис.1.

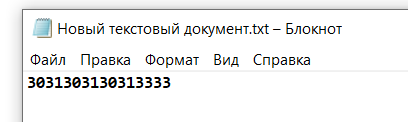


Рис.1 Содержимое текстового документа

Для открытия и считывания информации из файла, в приложении создается объект класса *OpenFileDialog,* с помощью которого у нас открывается окно в котором мы можем выбрать нужный нам файл, в нашем случае text.txt рис.2.

Для созданного объекта применяем метод GetCurrentDirectory(). С помощью которого мы получаем путь к текущей папке. С помощью метода ShowDialog(), открывается диалоговое окно рис.2.

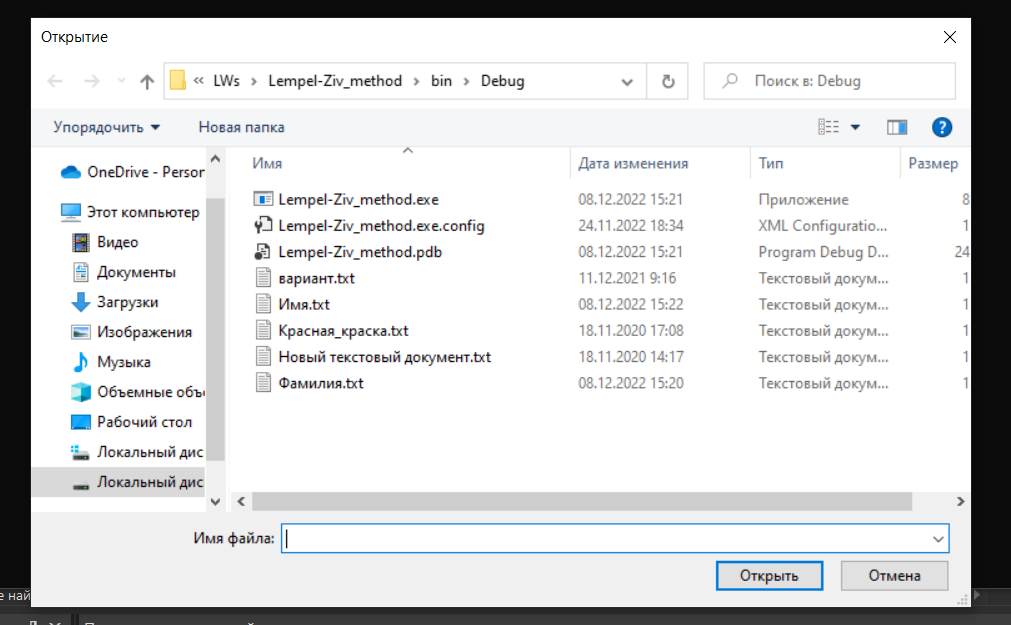


Рис.2 Открытие диалогового окна

В приложении есть переменная pahtFile, которая содержит имя выбранного нами файла в диалоговом окне. В переменную text записывается информация считанная из файла, с помощью метода ReadAllText, а затем считанная информация выводится в консоль, код представлен на рис.3.

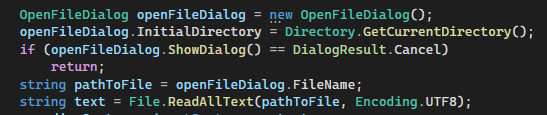


Рис.3  Код открытия и считывания информации из текстового файла

В начале алгоритма Лемпеля – Зива следует установить длину словаря, длину буфера данных и систему счисления, в данном алгоритме принято вычислять в четверичной системе счисления рис.4.

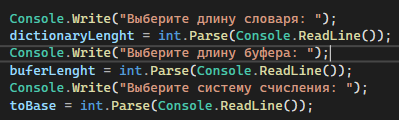


Рис. 4 Код для введения констант

В данном алгоритме принято вычислять в четверичной системе исчисления, результат кода представленном на рис.4 показан на рис.5.

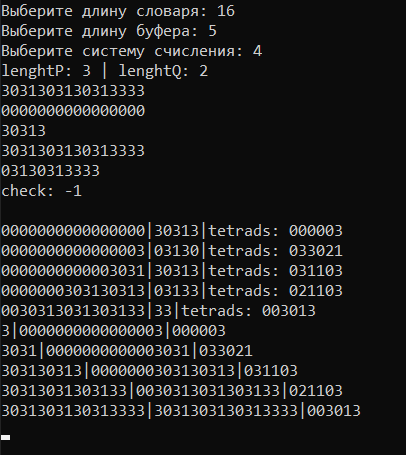


Рис.5 Результат кода

Далее применяются константы p и q, где p — номер позиции в которой мы нашли соответствие, q — длина соответствия интервала. S — следующий символ.

Затем мы можем посмотреть наглядно наше сообщение считанное из файла рис.6:

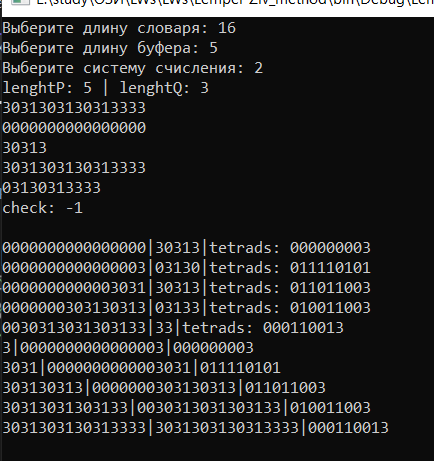


Рис.6 Сообщение считанное из файла

Словарь, заполненный нулями, с помощью кода рис.7:



Рис.7 Код заполнения словаря нулями

Результат кода заполнения словаря нулями представлен на рис.8:

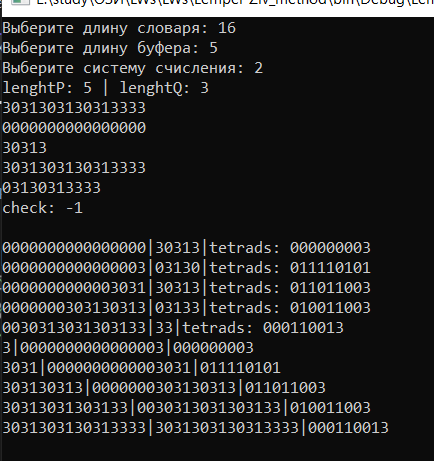


Рис.8 Словарь

Далее у нас заполняется буфер данных, длинной указанной ранее(в нашем примере 5 символов), код вывода буфера данных представлен на рис.9:



Рис.9 Код заполнения буфера

Результат буфера данных рис.10:

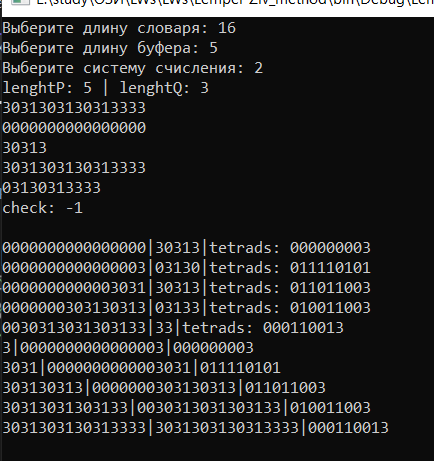


Рис.10 Буфер данных

Далее определяем нашу входную последовательность без первых n символов которые находятся в буфере рис.11. В данном коде с помощью метода Remove c первой позиции удаляются символы переменной buferLength.



Рис.11 Определение последовательности

Результат последовательности представлен на рис.12:

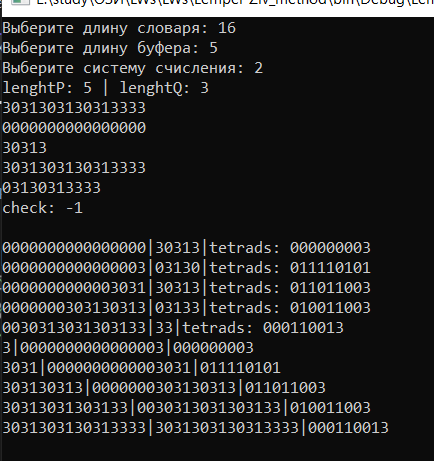


Рис.12 Последовательность

**Прямое преобразование**

Шаг 1: Из буфера данных (таблица 1) берем первый слева символ «3» и осуществляем поиск данного символа в буфере словаря. Символ в словаре отсутствует. Формируем триаду с началом повторяющейся последовательности (р), равным 0, длиной повторяющихся символов (q), равной 0, и следующими за повторяющейся последовательностью символом (c) «03».

| Буфер словаря | Буфер данных | Сообщение | Код |
| --- | --- | --- | --- |
| 00000000000000 | 30313 | 0313031333 | (00 00 03)4 |

Таблица 1

Сдвигаем окна на q + 1 позицию вправо (Таблица 2).

| Буфер словаря | Буфер данных | Сообщение | Код |
| --- | --- | --- | --- |
| 00000000000003 | 03130 | 313031333 | (03 30 21)4 |

Таблица 2

Шаг 2: Далее в буфере словаря ищем последовательность 03, она имеется, производим снова сдвиг вправо и формируем триаду (Таблица 3.)

| Буфер словаря | Буфер данных | Сообщение | Код |
| --- | --- | --- | --- |
| 00000000000003031 | 30313 | 3031333 | (03 11 03)4 |

Таблица 3

Следуя выше описонному алгоритму получаем следующие таблицы(таблица 4, таблица 5):

| Буфер словаря | Буфер данных | Сообщение | Код |
| --- | --- | --- | --- |
| 0000000303130313 | 03133 | 3031333 | (02 11 03 )4 |

Таблица 4

| Буфер словаря | Буфер данных | Сообщение | Код |
| --- | --- | --- | --- |
| 0000303130313133 | 33 | 333 | (03 30 21)4 |

Таблица 5

Для сжатия данных методом Лемпеля-Зива был разработан код представленный на рис. 13.1 и рис. 13.2, который выполняется только при условии, что буфер данных не равен нулю. Далее с помощью раннее кодом определенных переменных p, q формируются триады и происходят сдвиги.

|  |  |
| --- | --- |

Рис.13.1 Первая часть кода сжатия Рис.13.2 Вторая часть кода сжатия

Конечный результат сжатия будет набор полученных триад рис.14:

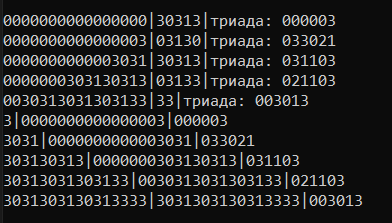


Рис.14 Результат сжатия

**Обратное преобразование**

В процессе восстановления используются только набор триад (кодов) и буфер. Принимаем размер буфера и его начальное содержимое такое же, как и при сжатии.

Шаг 1:

| Сообщение (распакованное) | Буфер словаря | Код |
| --- | --- | --- |
| 3 | 000000000000000003 | 00 00 03 |

Таблица 6

Анализируется триада (00,00,03) таблица 6. Начало повторяющейся последовательности (p) и ее длина (q) равны 0 – повторений нет. Добавляем к словарю символ «1» и сдвигаем окно словаря на q + 1 позицию вправо.

Шаг 2:

| Сообщение (распакованное) | Буфер словаря | Код |
| --- | --- | --- |
| 3031 | 0000000000000003031 | 03 30 21 |

Таблица 7

Анализируется триада (03,30,21) таблица 7. Повторяющаяся последовательность равна 3. Добавляем к словарю повторяющуюся последовательность «031» и символ «0». Сдвигаем окно словаря на q + 1 позицию вправо.

Шаг 3:

| Сообщение (распакованное) | Буфер словаря | Код |
| --- | --- | --- |
| 303130313 | 0000000000303130313 | 03 11 03 |

Таблица 8

Анализируется триада (03,11,03) таблица 8. Повторяющаяся последовательность равна 5. Добавляем к словарю повторяющуюся последовательность «01» и символ «1». Сдвигаем окно словаря на q + 1 позицию вправо. И т. д.

Делая данный алгоритм мы получим следующий результат рис. 15

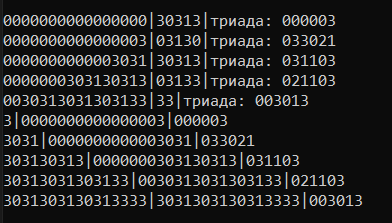


Рис. 15 Результат распаковки

Код распаковки сообщения представлен на рис.16. Данный код описывает распаковку сообщения с помощью триад, составленных при сжатии сообщения. Каждая триада разбивается на отдельные символы и анализируется кодом.

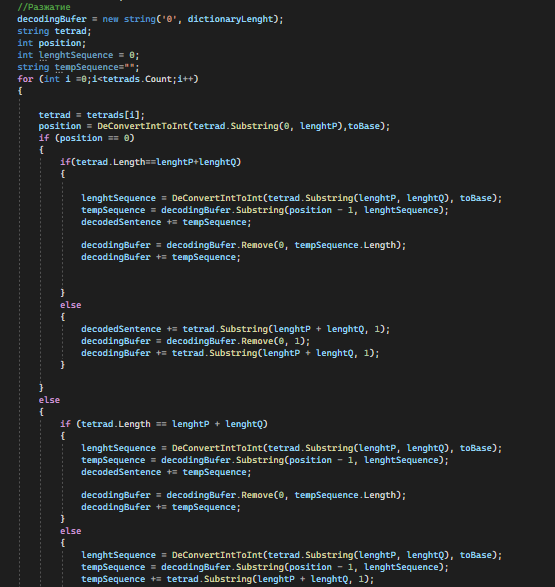


Рис.16 Код распаковки данных

Для измерения времени потраченного на сжатие данных был использован объект stopwatch1 класса Stopwatch. И вызван метод Start() для начала измерения времени рис.17.



Рис.17 Начало измерения времени запаковки данных

Для остановки измерения времени, вызывался метод Stop() рис.18.



Рис.18 Остановка измерения времени запаковки данных

Результат времени потраченный на упаковку данных представлен на рис.19.

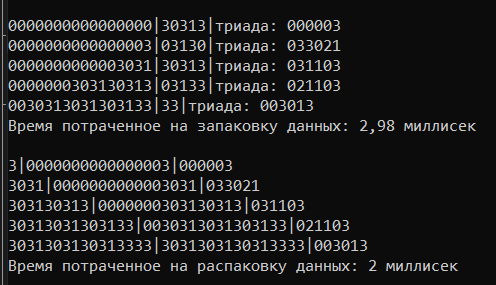


Рис.19 Результат времени запаковки данных

Такой же метод использовался для измерения времени на распаковку данных, результат которого представлен на рис. 20.

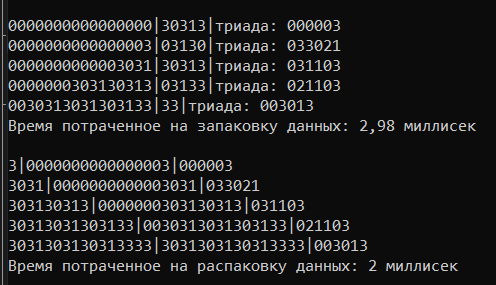


Рис.20 Результат времени распаковки данных

Вывод: приобретены теоретические знания по использованию метода Лемпеля − Зива (Lempel-Ziv) для сжатия/распаковки данных. Разработано приложение для сжатия/распаковки данных по методу Лемпеля − Зива (Lempel-Ziv). Измерено время сжатия/распаковки данных. Приведен пример в отчете запаковки и распаковки данных.

**Ответы на контрольные вопросы:**

1. *К какому классу методов сжатия относится метод LZ и почему?*

Метод *LZ* относится к словарным методам сжатия.

Так как сжатие данных осуществляется за счет замены записей соответствующими кодами из словаря.

1. *В чем заключается, на Ваш взгляд, оптимизация алгоритма LZ?*

Словарные методы предпологает решения оптимизационной задачи связанные с выбором размеров или длины словаря и буфера: чем длиннее эти элементы, тем большую эффективность сжатия можно достичь, однако при этом возрастает время анализа(сжатия).

1. *Какие модификации метода LZ77 Вам известны? В чем зключается особенность модификаций?*

Суть метода LZ77 (как и последующих его модификаций) состоит в следующем: упаковщик постоянно хранит некоторое количество последних обработанных символов в буфере. По мере обработки входного потока вновь поступившие символы попадают в конец буфера, сдвигая предшествующие символы и вытесняя самые старые. Размеры этого буфера, называемого также скользящим словарем (англ. sliding dictionary), варьируются в разных реализациях систем сжатия.

1. *Пояснить физический смысл соотношения (10.1).*

**

Длина кодовой комбинации (триады – p, q, s) на каждом шаге определяется соотношением описанным выше.

1. *Как будет выглядеть сообщение, состоящее из символа моноалфавита при определенных значениях n1, n2?*

Моноалфавит — это алфавит состоящий из одного символа.

Будет использовано наше сообщение, а недостающие символы заполняется нулями.