АЛГОРИТМ СЖАТИЯ ИНФОРМАЦИИ ЛЕМПЕЛА-ЗИВА-ВЕЛЧА

Дашкевич Р.М.
Кафедра вычислительных методов и программирования
Научный руководитель: Колосов С.В., профессор, д-р физ.-мат. наук, доцент
e-mail: kolosov@bsuir.by

***Аннотация* — В работе рассмотрен вариант реализации алгоритма сжатия информации Лемпеля — Зива — Велча в среде Delphi. Это алгоритм сжатия без потерь информации, он используется во многих приложениях, таких как – ARG, ZIP, RAR и др. В докладе проанализирована степень сжатия различных типов файлов: текстовых, графических, исполняемых и др.**

Ключевые слова: алгоритмы сжатия информации, кодирование, декодирование.

Алгоритм Лемпеля — Зива — Велча (Lempel-Ziv-Welch, LZW) — это универсальный алгоритм сжатия данных без потерь, созданный Абрахамом Лемпелем (*Abraham Lempel*), Якобом Зивом (*Jacob Ziv*) и Терри Велчем (*Terry Welch*). Он был опубликован Велчем в 1984 году [1], в качестве улучшенной реализации алгоритма LZ78, опубликованного Лемпелем и Зивом в 1978 году.

Алгоритм на удивление прост. Если в двух словах, то LZW-сжатие заменяет строки символов некоторыми кодами. Это делается без какого-либо анализа входного текста. Вместо этого при добавлении каждой новой строки символов просматривается таблица строк. Сжатие происходит, когда код заменяет строку символов. Коды, генерируемые LZW-алгоритмом, могут быть любой длины, но они должны содержать больше бит, чем единичный символ. Первые 256 кодов (когда используются 8-битные символы) по умолчанию соответствуют стандартному набору символов. Остальные коды соответствуют обрабатываемым алгоритмом строкам.

Алгоритм LZW построен вокруг таблицы фраз (словаря), которая отображает строки символов сжимаемого сообщения в коды фиксированной длины (обычно 12-битовые). Таблица обладает так называемым свойством предшествования, то есть для каждой фразы словаря, состоящей из некоторой фразы w и символа K, фраза w тоже содержится в словаре.

Алгоритм кодирования.

Инициируем словарь односимвольными фразами (обычно это 256 ASCII символов).

Читаем первый символ сообщения в текущую фразу W.

Шаг алгоритма:

```
        Читаем очередной символ сообщения K;
        Если это конец сообщения, то
            Выдаем код w;
            Выход;
        Если фраза wK уже есть в словаре, то
            Заменяем w на код фразы wK;
            Повторяем Шаг алгоритма;
        Иначе
            Выдаем код w;
    Добавляем wK в словарь;
    Повторить Шаг алгоритма;
    Конец;
```

Описанный алгоритм не пытается оптимально выбирать фразы для добавления в словарь, однако он может быть эффективно реализован.

Алгоритм декодирования.

Декодер LZW использует тот же словарь, что и кодировщик. Каждый считываемый код разбивается с помощью словаря на предшествующую фразу w и символ K.

```
Код=читаем первый код сообщения;
Предыдущий код= Код;
Выдаем символ K, у которого код(K)==Код;
Следующий код:
    Код=читаем очередной код сообщения;
    Входной Код=Код;
    Если Конец Сообщения, то Выход;
Следующий символ:
 Если Код==Код(wK), то
   Выдать K;
   Код=Код(w);
   Повторить следующий символ;
   Иначе Если Код==код(K), то
   Выдать K;
   Добавить в словарь (Предыдущий Код, K).
        Предыдущий Код= Входной Код;
   Повторить Следующий Код;
   Конец;
```

Алгоритм LZW хорошо сжимает текстовые файлы, базы данных, картинки в формате *.Bmp. Но его не рекомендуем для сжатия файлов типа *.exe, *.gif, *.jpg, *.tif и других, где уже были использованы алгоритмы сжатия информации. Для таких файлов этот алгоритм уже не приведет к их сжатию, а только увеличит размер файлов.

[1] Terry Welch, Technique for High-Performance Data Compression // Computer, June, 1984.