УДК 004.056.3

Кодирование методом алгоритма Лемпеля-Зива-Велча

Ленкин Алексей Викторович
Приамурский государственный университет имени Шолом-Алейхема
Студент

Аннотация
Статья посвящена разбору алгоритма Лемпеля-Зива-Велча, возможностям его применения и программной реализации на языке C++
Ключевые слова: алгоритм Лемпеля-Зива-Велча, C++, шифрование, сжатие информации, LZW

The encoding method of the algorithm Lempel-Ziv-Welch

Lenkin Aleksei Viktorovich
Sholom-Aleichem Priamursky State University
Student

Abstract
The article is devoted to parsing the algorithm Lempel-Ziv-Welch, the possibilities of its application and software implementation in the C++ language
Keywords: Lempel-Ziva-Welch algorithm, C++, encryption, data compression, LZW

Научный руководитель:
Лучанинов Дмитрий Васильевич
Приамурский государственный университет имени Шолом-Алейхема
старший преподаватель кафедры информационных систем, математики и методик преподавания

На данный момент в мире существуют и создаются терабайты различной информации ежедневно, поэтому проблема её хранения остаётся наиболее актуальной, и покупка физических носителей большого объема часто является слишком дорогим для обычных пользователей.

Для решения этих задач существует и создается огромное количество различных методов и алгоритмов сжатия данных. Многие из них могут использоваться также и для шифрования, хотя их криптоскойкость существенно меньше чем у специализированных для этого алгоритмов. Одним из самых популярных является алгоритм Лемпеля-Зива-Велча (Lempel-Ziv-Welch, LZW) — это универсальный алгоритм сжатия данных без потерь[1].

Целью исследования было рассмотреть работу алгоритма LZW, сферы его применения и создать программную реализацию.

Исследованиями в данной теме занимались следующие авторы. Т.З.Чан, В.Ш. Вуй составили обзор на алгоритмы сжатия данных[2]. М.Ж.Калдарова описала разработку программы сжатия и восстановления данных по алгоритму Лемпеля-Зива [3]. Д.С. Ковалев составил обобщение известных способов кодирования строк[4].

Алгоритм Лемпеля-Зи́ва-Велча (Lempel-Ziv-Welch, LZW) — это универсальный алгоритм сжатия данных без потерь, созданный Авраамом Лемпелем (англ. Abraham Lempel), Яаковом Зивом (англ. Jacob Ziv) и Терри Велчем (англ. Terry Welch). Он был опубликован Велчем в 1984 году в качестве улучшенной реализации алгоритма LZ78, опубликованного Лемпелем и Зивом в 1978 году. Алгоритм разработан так, чтобы его можно было быстро реализовать, но он не обязательно оптимален, поскольку он не проводит никакого анализа входных данных [5].

Сам алгоритм сжатия происходит следующим образом:

1. Формируется алфавит на основе введённой фразы и добавляется в словарь.
2. Начинается считывание посимвольно с начала строки.
3. Если данный символ есть в таблице то к нему добавляется следующий и формируется строка, так продолжается пока строка не станет уникальной для словаря.
4. Номер предыдущая итерация, найденной в словаре записывается как код, а уникальная строка добавляется в словарь.
5. Шаги с 3 по 4 продолжаются до конца кодируемого слова.

Кодирование [1]

Пусть мы сжимаем последовательность «abacabadabacabae».

Шаг 1: Тогда, согласно изложенному выше алгоритму, мы добавим к изначально пустой строке “a” и проверим, есть ли строка “a” в таблице. Поскольку мы при инициализации занесли в таблицу все строки из одного символа, то строка “a” есть в таблице.

Шаг 2: Далее мы читаем следующий символ «b» из входного потока и проверяем, есть ли строка “ab” в таблице. Такой строки в таблице пока нет.

Добавляем в таблицу <5> “ab”. В поток: <0>;

Шаг 3: “ba” — нет. В таблицу: <6> “ba”. В поток: <1>;

Шаг 4: “ac” — нет. В таблицу: <7> “ac”. В поток: <0>;

Шаг 5: “ca” — нет. В таблицу: <8> “ca”. В поток: <2>;

Шаг 6: “ab” — есть в таблице; “aba” — нет. В таблицу: <9> “aba”. В поток: <5>;

Шаг 7: “ad” — нет. В таблицу: <10> “ad”. В поток: <0>;

Шаг 8: “da” — нет. В таблицу: <11> “da”. В поток: <3>;

Шаг 9: “aba” — есть в таблице; “abac” — нет. В таблицу: <12> “abac”. В поток: <9>;

Шаг 10: “ca” — есть в таблице; “cab” — нет. В таблицу: <13> “cab”. В поток: <8>;

Шаг 11: “ba” — есть в таблице; “bae” — нет. В таблицу: <14> “bae”. В поток: <6>;

Шаг 12: И, наконец последняя строка “e”, за ней идет конец сообщения, поэтому мы просто выводим в поток <4>.

Текущая строка	Текущий символ	Следующий символ	Вывод		Словарь	
			Код	Биты		
ab	a	b	0	000	5:	ab
ba	b	a	1	001	6:	ba
ac	a	c	0	000	7:	ac
ca	c	a	2	010	8:	ca
ab	a	b	-	-	-	-
aba	b	a	5	101	9:	aba
ad	a	d	0	000	10:	ad
da	d	a	3	011	11:	da
ab	a	b	-	-	-	-
aba	b	a	-	-	-	-
abac	a	c	9	1001	12:	abac
ca	c	a	-	-	-	-
cab	a	b	8	1000	13:	cab
ba	b	a	-	-	-	-
bae	a	e	6	0110	14:	bae
e	e	-	4	0100	-	-

Итак, мы получаем закодированное сообщение «0 1 0 2 5 0 3 9 8 6 4», что на 11 бит короче.

Опишем программную реализацию данного алгоритма на языке C++. Листинг программы описан ниже:

```
#include <iostream>
#include <string>

using namespace std;
int main()
{
     string s;
     getline(cin,s);
     int code[s.length()];
     string *c=new string[s.length()*s.length()];
     int l=-1;
     bool p=true;
     string temp;

     for (int i=0;i<s.length();i++) //Формирование алфавита
     {
          for (int j=0;j<=l+1;j++)
```

```
          {
               if (s[i]==c[j][0])
                    p=false;
          }
          if (p)
          {
               l++;
               c[l]=s[i];
               p=true;
          }
          else p=true;
     }

     for (int i = 0; i <=l; i++) // Упорядочивание элементов
алфавита по возрастанию
     {
        for (int j = 0; j <= l - i - 1; j++)
          {
            if (c[j] > c[j + 1]) {
                temp = c[j];
                c[j] = c[j + 1];
                c[j + 1] = temp;
            }
        }
    }
    int i,j=0;
    int k=-1;
    int f,m;
    temp.clear();

     while (i<s.length()) // Алгоритм Лемпеля - Зива - Венча
    {
     m=0;
     k++;
     temp=s[i];
          for (j=0;j<=l;j++)
          {
               if (temp==c[j])
               {
                    m++;
                    if ((i+m)<s.length()) temp+=s[i+m];
                    f=j;
               }
          }
          if ((i+m)<s.length())
          {

               l++;
               c[l]=temp;
          }
          code[k]=f;
          i+=m;
          temp.clear();
```

```
    }

    for (i=0;i<=k;i++) cout<<code[i]<<" "; // Вывод полученного
кода
}
```

Результатом выполнения данной программы будет следующий код (рисунок 1).

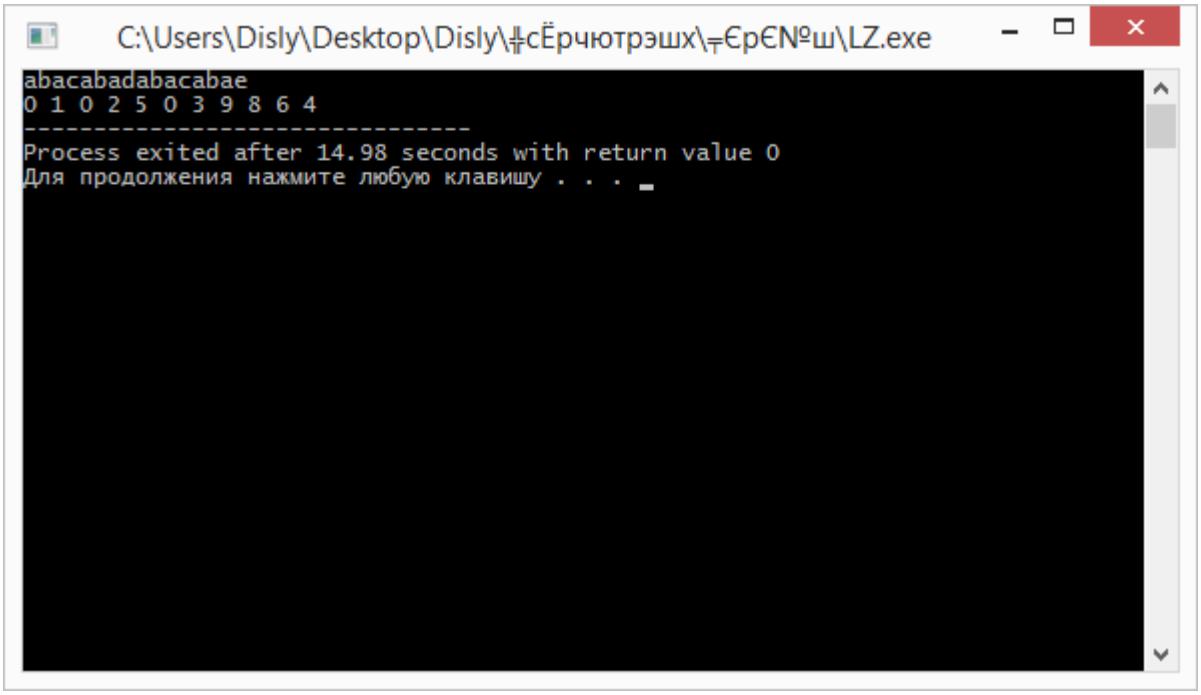

Рисунок 1. Результат выполнения кодирования по алгоритму Лемпеля-Зива - Велча

Как видно, полученная кодированная фраза такая же как у полученной при шифровании вручную.

Таким образом, можно сказать, что алгоритм Лемпеля-Зива-Велча является эффективным способом сжатия данных. Благодаря легкому алгоритму, имеется возможность его программной реализации на любом языке программирования и внедрения в проекты разработчиков, что поможет решению главной проблемы – хранение информации.

Библиографический список

1. Алгоритмы LZW, LZ77 и LZ78 [Электронный ресурс] URL: https://habrahabr.ru/post/132683/ (дата обращения 29.01.2018)
2. Чан Т.З., Вуй В.Ш. Алгоритмы сжатия данных // В сборнике: Информационные технологии в науке, управлении, социальной сфере и медицине Сборник научных трудов II Международной конференции. Национальный исследовательский Томский политехнический университет. 2015. С. 820-822.
3. Калдарова М.Ж. Разработка программы сжатия и восстановления данных

по алгоритму Лемпеля-Зива // В сборнике: Молодой исследователь: вызовы и перспективы Сборник статей по материалам XLVI международной научно-практической конференции . 2017. С. 90-96.
4. Ковалев Д.С. Обобщение известных способов кодирования строк // Вестник Новосибирского государственного университета. Серия: Информационные технологии. 2010. Т. 8. № 4. С. 5-14.
5. Прокис Д. Цифровая связь. Пер. с англ. / Под ред. Д.Д. Кловского. М.: Радио и связь. 2000. 800 с.