Учреждение образования

«БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Дисциплина «Защита информации и надёжность информационных систем»

**Лабораторная работа №9**

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

Выполнил:

Студент 3 курса 7 группы ФИТ

Тимошенко Д. В.   
 Проверила:   
 асс. Николайчук А. Н.

Минск 2024

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

**Задание на лабораторную работу**

1. Разработать авторское приложение в соответствии с целью лабораторной работы. При этом предусмотреть возможность оперативного изменения размеров окон (n1, n2).

2. С помощью приложения выполнить прямое и обратное преобразования произвольного текста длиной несколько килобайт.

Формат представления параметров p и q выбрать по указанию преподавателя.

3. Изменяя размеры окон, оценить скорость и эффективность (используя соотношения на с. 76) выполнения операций сжатия/распаковки.

4. Результаты оформить в виде отчета по установленным правилам.

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, 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.

Полученная в результате поиска фраза S t− (р− 1) ,S t − (р− 1) + 1, S t− (р− 1) + (q− 1) кодируется с помощью двух чисел:

1) смещения (англ. offset) от начала буфера p;

2) длины соответствия, или совпадения (англ. match length) q.

Ссылки (p и q − указатели) однозначно определяют фразу. Дополнительно в выходной поток записывается символ s, следующий за совпавшей строкой буфера.

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

l(с ) = logN n1 +logN n2 + 1, (10.1)

где N – мощность алфавита.

Приложение, для выполнения задания, написано на языке программирования C#.NET. Ниже представлены листинг функций, используемых во всех задания, листинг 1.1.

|  |
| --- |
| void FindChars(string dict, string bufWord, out int indexInArray, out int Length, out char lastElement)  {  indexInArray = 0;  Length = 0;  lastElement = bufWord[0];  while (dict.Contains(bufWord.Substring(0, (Length + 1))))  {  indexInArray = dict.IndexOf(bufWord.Substring(0, (Length + 1))) + 1;  Length++;  if (bufWord.Length == Length)  {  lastElement = '|';  break;  }  else  lastElement = bufWord[Length];  }  }  void SendChars(ref string firstBuf, ref string secondByf, int charsCount)  {  charsCount = charsCount > secondByf.Length ? secondByf.Length : charsCount;  if (charsCount > 0)  {  firstBuf += secondByf.Substring(0, charsCount);  secondByf = secondByf.Substring(charsCount, secondByf.Length - charsCount);  }  }  void Minimize(ref string byf, int size)  {  if (byf.Length > size)  byf = byf.Substring((byf.Length - size), byf.Length - (byf.Length - size));  } |

Листинг 1.1 – Вспомогательные функции

Код выполнения самих заданий лабораторной работы, листинг 1.2.

|  |
| --- |
| static void Main(string[] args)  {  string dictionary;  string word;  string message;  string kodWord = "";  int dictionaryLength;  int wordLength;  string inputMessage = "Тимошенко Дмитрий Валерьевич";  string resultMessage = "";  dictionaryLength = 8;  wordLength = 8;  word = inputMessage.Substring(0, wordLength);  message = inputMessage.Substring(wordLength, inputMessage.Length - wordLength);  int p = 0, q = 0;  char c;  dictionary = new string('0', dictionaryLength);  while (word != "")  {  p = 0;  q = 0;  FindChars(dictionary, word, out p, out q, out c);  SendChars(ref dictionary, ref word, q + 1);  SendChars(ref word, ref message, q + 1);  Minimize(ref dictionary, dictionaryLength);  Minimize(ref word, wordLength);  kodWord += p.ToString() + q.ToString() + c.ToString();  Console.WriteLine("Словарь: " + dictionary);  Console.WriteLine("Слово(буфер данных): " + word);  Console.WriteLine("Кодовое слово: " + kodWord);  Console.WriteLine("-------------------------------");  }  string str = "";  dictionary = new string('0', dictionaryLength);  for (int i = 0; i < kodWord.Length / 3; i++)  {  p = int.Parse(kodWord[i \* 3].ToString());  q = int.Parse(kodWord[i \* 3 + 1].ToString());  c = kodWord[i \* 3 + 2];  if (p == 0 && q == 0)  {  resultMessage += c;  dictionary += c;  }  else  {  str = dictionary.Substring((p - 1), q) + c;  resultMessage += str;  dictionary += str;  }  Minimize(ref dictionary, dictionaryLength);  Console.WriteLine("Код: " + p + " " + q + " " + c);  Console.WriteLine("Результат: " + resultMessage);  Console.WriteLine("Словарь: " + dictionary);  Console.WriteLine("-------------------------------");  } |

Листинг 1.2 – выполнение задания лабораторной работы

Вывод представленного кода, рисунок 1.1.

|  |  |
| --- | --- |
| а | б |

Рисунок 1.1 Вывод кода: а – кодирование; б - декодирование

**Вывод:** в ходе выполнения лабораторной работы был изучен алгоритм сжатия Лемпеля/Зива. Были приобретены практические навыки сжатия и распаковки с этим алгоритма.