9. ***Морелос-Сарагоса, Р.*** Искусство помехоустойчивого кодирования. Методы, алгоритмы, применение / Р. Морелос-Сарагоса. – М. : Техносфера, 2005. – 287 с.

10. ***Питерсон, В.*** Коды, исправляющие ошибки / В. Питерсон, Ф. Уэлдон. – М. : Мир, 1976. – 236 с.

11. ***Питерсон, У.*** Коды, исправляющие ошибки / У. Питерсон. – М. : Мир, 1964. – 313 с.

12. ***Хаулет, Т.*** Защитные средства с открытыми исходными текстами БИНОМ / Т. Хаулет. – М. : ИНТУИТ.ру, 2007 – 315 с.

УДК 004.428.4

ПРОГРАММНЫЙ ПРОДУКТ ДЛЯ ДЕМОНСТРАЦИИ ПРИМЕНЕНИЯ ОПТИМАЛЬНОГО КОДИРОВАНИЯ НА ПРИМЕРЕ АЛГОРИТМОВ ШЕННОНА-ФАНО И ХАФФМАНА

М.О. Смирнова, А.П. Смирнов

В статье представлен программный продукт, с помощью которого можно продемонстрировать применение оптимальных кодов. Дано описание компонентов программного продукта и возможностей использования при изучении разделов, связанных с информацией, ее кодированием и передачей.

Ключевые слова: *информация, энтропия, оптимальное кодирование, демонстрационная программа, программный код.*

Key words: *information, entropy, optimal coding, the demonstration program, the program code.*

Теория информации является одним из курсов при подготовке инженеров, специализирующихся в области автоматизированных систем управления и обработки информации. Функционирование таких систем существенным образом связано с получением, подготовкой, передачей, хранением и обработкой информации, поскольку без осуществления этих этапов невозможно принять правильное решение и осуществить требуемое управляющее воздействие, которое является конечной целью функционирования любой системы.

Разработанная демонстрационная программа обладает следующими возможностями:

- позволяет получить первоначальные сведения о кодировании дискретной информации, о проблеме оптимального кодирования, об алгоритмах Шеннона-Фано и алгоритме Хаффмана;
- предоставляет возможность ввода кодируемого сообщения двумя способами: с клавиатуры и из заранее приготовленного текстового файла, и выбора длины этого сообщения;
- производит кодирование введенного сообщения с помощью алгоритмов Шеннона-Фано и Хаффмана;
- вычисляет дополнительные параметры, связанные с сообщением и процессом кодирования: энтропию, среднюю длину элементарного кода, избыточность кода и коэффициент эффективности;
- справочная система программы позволяет ознакомиться с основными принципами работы данного программного продукта.

Данный программный продукт написан на языке Visual Basic – этот язык программирования довольно прост в усвоении, позволяет относительно просто создавать приложения для работы с операционной системой Windows и предназначен для демонстрации применения алгоритмов кодирования Шеннона-Фано и Хаффмана. Он состоит из трех форм, каждой

из которых соответствует свой программный модуль, и дополнительного модуля, обслуживающего основную форму программы.

В модуле первой формы (титульной) находится только один обработчик для кнопки Command1.

```
Private Sub Command1_Click()
  Title.Hide
  Codir.Show
End Sub,
```

в котором использованы методы форм. Метод Hide скрывает титульную форму, а метод Show, наоборот, делает видимой вторую форму, несущую основную функциональную нагрузку.

При загрузке второй формы запускается соответствующий обработчик, в котором задаются начальные значения некоторых параметров: свойства Value для переключателей Option1 и Option2 и переменная n, используемая для хранения количества символов кодируемого сообщения.

```
Option1.Value = False
Option2.Value = False
n = 0
```

Основным параметром, который используется в дальнейшем, является переменная n, значение которой вводится с помощью поля редактора Text2, поэтому следующий обработчик – это обработчик ввода в Text2. Прежде всего, в нем проверяется – был ли ввод данных; если был, то они преобразуются к целому и помещаются в переменную n. Кроме того, обработчик выполняет ряд дополнительных действий: очищает поле ввода элемента Text1, чтобы можно было осуществить новый ввод, очищает две таблицы Shennon и Khaffman и выводит их строки-заголовки.

Следует заметить, что параллельно с событием ввода данных генерируется событие, без которого затруднительно вводить данные, – нажатие клавиш на клавиатуре. Обработчик этого события следит за тем, чтобы введеными символами были только цифры (допускается нажатие клавиши «Back Space»).

```
Private Sub Text2_KeyPress(KeyAscii As Integer)
 If Not (KeyAscii >= Asc("0") And KeyAscii <= Asc("9") Or KeyAscii = 8) Then KeyAscii = 0
End Sub
```

Для этого проверяется ASCII код введенного символа. Он должен лежать в диапазоне символов от «0» до «9». Если символ не подходит, то код обнуляется.

Следующим шагом является ввод сообщения, которое должно быть закодировано. Чтобы это осуществить, необходимо выбрать способ ввода этого сообщения – с клавиатуры или загрузка тестового файла. Для такого выбора в программе присутствуют следующие элементы: пункты меню и переключатели. Событие выбора обрабатывается только у переключателей, а у пунктов меню таких обработчиков нет. Выбор пункта меню приводит к вызову соответствующего обработчика переключателя

```
Private Sub DataFile_Click()
  Option2_Click
End Sub
```

```
Private Sub DataRteboard_Click()
  Option1_Click
End Sub
```

Эти обработчики имеют похожую структуру. В основу положена проверка наличия ввода длины кодируемого сообщения. Если длина сообщения в поле редактора Text2 не была введена, то выдается сообщение об этом, выбор переключателя аннулируется, а фокус ввода передается редактору Text2.

В случае наличия к данному моменту в переменной n значения, соответствующего длине сообщения, редактор Text1 очищается и получает фокус ввода, фиксируется выбор первого переключателя, «обнуляются» все значения массивов xcode1 и xcode2, которые

предназначены для хранения кодов символов сообщения. Так как элемент Option1 отвечает за выбор ввода с клавиатуры, то в рассматриваемом обработчике переменная i получает значение 1, она будет выполнять роль счетчика вводимых символов сообщения.

Обработчик второго переключателя отличается от предыдущего только тем, что он не готовит редактор Text1 к вводу сообщения.

```
OpenFile
st = Left(st, n)
Text1.Text = st
```

В нем вызывается функция OpenFile, которая позволяет считать строку их текстового файла, взять из нее только n первых символов в качестве сообщения и вставить их в поле редактора Text1.

Функция OpenFile использует элемент ComDialog1 в режиме ShowOpen, позволяющем осуществлять выбор файла для открытия.

```
Sub OpenFile()
Dim fnumber As Integer
 ComDialog1.ShowOpen
 If ComDialog1.FileName <> "" Then
  fnumber = FreeFile
  Open ComDialog1.FileName For Input As fnumber
    Line Input #fnumber, st
    Close #fnumber
 End If
End Sub
```

Когда файл выбран, тогда значение свойства FileName общего диалога ComDialog1 будет содержать имя этого файла. Файл открывается как свободный, что позволяет не следить за нумерацией открываемых файлов. Далее файл с именем FileName открывается командой Open для чтения информации, считывается из него строка и помещается в переменную st строкового типа и файл закрывается командой Close.

Основные действия выполняет обработчик элемента Command2. Прежде всего, выполняются общие вычисления. Они начинаются с подсчета количества неповторяющихся символов в сообщении.

```
kx = 0
 For i = 1 To n
  l = 1
  For j = 1 To i – 1 'meet before
   ch1 = Mid(st, i, 1)
   ch2 = Mid(st, j, 1)
   If ch1 = ch2 Then l = 0
  Next j
  If l = 1 Then  'символ не рассматривался
    kx = kx + 1
    ch1 = Mid(st, i, 1)
    x(kx) = ch1
    xn(kx) = 0
    For j = i To n
     ch1 = Mid(st, i, 1)
     ch2 = Mid(st, j, 1)
     If ch1 = ch2 Then xn(kx) = xn(kx) + 1
    Next j
```

```
 End If
Next i
```

Переменная kx является в данном фрагменте кода счетчиком для неповторяющихся символов, а переменная l играет роль флага – он имеет значение 1, если символ ранее не рассматривался, в противном случае он принимает значение 0. Для работы с символами сообщения используются переменные ch1 и ch2. В них помещаются символы, вырезанные из строки сообщения функциями Mid(st, i, 1) и Mid(st, j, 1). Если взятый символ в сообщении ранее не встречался, то счетчик kx увеличивает свое значение на 1. (В дальнейшем эта переменная используется как хранитель количества различных символов во введенном сообщении.) Сам символ помещается в массив x в элемент с индексом, соответствующим значению счетчика kx, а элемент массива xn с тем же индексом используется для подсчета числа повторений в сообщении рассматриваемого символа.

Когда все предварительные подсчеты выполнены, по числу неповторяющихся символов задается количество строк в таблицах Shennon и Khaffman.

```
Shennon.Rows = kx + 1
Khaffman.Rows = kx + 1
```

Строк в таблицах на 1 больше из-за строки-заголовка. Следом заполняется массив xw относительными частотами символов сообщения (частота повторяемости символа делится на общее число символов в сообщении). С помощью данных относительных частот рассчитывается такой важный параметр, как энтропия, для нее отведена переменная hx.

```
hx = 0
For i = 1 To kx
 xw(i) = xn(i) / n
 hx = hx – xw(i) * Log(xw(i)) / Log(2)
Next i
```

Список различных символов сообщения с их частотами теперь необходимо отсортировать в порядке убывания частот повторяемости символов. В программе применена сортировка методом прямого выбора (выбор пал на простую сортировку, так как количество элементов в массиве символов невелико). Как видно из приведенного кода, сортировке подвергаются сразу три массива: массив с относительными частотами, массив с частотами и массив с символами.

Далее осуществляется кодирование символов сообщения. Сначала применяется алгоритм Шеннона-Фано, в котором, по сути, применен метод половинного деления массива относительных частот (сумма относительных частот в одной половине примерно равна сумме в другой). Параллельно с разбиением массива относительных частот заполняется массив xcode1, предназначенный для хранения кодов символов. К каждому элементу из первой половины массива прописывается «0», а к каждому элементу из второй – «1». Процесс деления пополам повторяется до тех пор, пока в каждой половинке не будет только по одному элементу. В итоге каждый элемент массива xcode1 будет содержать код соответствующего символа сообщения.

Когда кодирование закончилось, проводятся дополнительные расчеты: вычисляются средняя длина элементарного кода l_shen и коэффициент относительной эффективности кодирования nu_shen.

```
l_shen = 0
 For l = 1 To kx
  l_shen = l_shen + xw(l) * Len(xcode1(l))
 Next l
 Text6.Text = CStr(l_shen)
 nu_shen = hx / l_shen
```

```
Text7.Text = CStr(nu_shen)
```

Первая величина определяется как сумма произведений относительных частот (вероятностей появления в сообщении) символов и соответствующих длин кодов символов, а вторая величина рассчитывается как отношение энтропии к средней длине элементарного кода. Полученные значения помещаются, соответственно, в поля редакторов Text6 и Text7.

Следующим алгоритмом кодирования является алгоритм Хаффмана. Для его применения введен специальный тип elem

```
Public Type elem
    ver_sum As Single
    ind_in(1 To 50) As Integer
    code(1 To 50) As String
    kol As Integer
End Type,
```

который определен во вспомогательном модуле программы. В данном типе сосредоточены элементы для хранения кодов символов, их суммарной относительной частоты, их индексов и их количества. Данный тип определился, исходя из алгоритма Хаффмана: в нем придется объединять символы в группы, а введенный тип позволит не только их объединять, но и сохранять их индивидуальную информацию, которая понадобится на завершающем этапе кодирования.

Первоначально все символы сообщения хранятся отдельно друг от друга в элементах массива tree(1 To 50) As elem.

Для этого массива заполняются поля значениями, которые были вычислены ранее:

```
n = kx
 For i = 1 To n
  tree(i).ver_sum = xw(i)
  tree(i).kol = 1
  tree(i).ind_in(1) = i
  tree(i).code(1) = ""
 Next i,
```

только поля кода символа заполняются пустыми строками. Пока в массиве рассматривается более одного элемента, производятся действия в соответствии с алгоритмом кодирования Хаффмана: объединяются два элемента с самыми маленькими относительными частотами.

```
  tree(n – 1).ver_sum = tree(n – 1).ver_sum + tree(n).ver_sum
  For i = 1 To tree(n).kol
    tree(n – 1).ind_in(tree(n – 1).kol + i) = tree(n).ind_in(i)
    tree(n – 1).code(tree(n – 1).kol + i) = tree(n).code(i)
  Next i
```

Первый из объединенных элементов присоединяет накопленное кодовое слово к символу «0», а второй – к символу «1».

```
  For i = 1 To tree(n – 1).kol
    tree(n – 1).code(i) = "0" & tree(n – 1).code(i)
  Next i
  For i = 1 To tree(n – 1).kol
    tree(n – 1).code(tree(n – 1).kol + i) = "1" & tree(n – 1).code(tree(n – 1).kol + i)
  Next i
  tree(n – 1).kol = tree(n – 1).kol + tree(n).kol
  n = n – 1
```

Количество рассматриваемых элементов массива уменьшается на единицу. Так как получился новый элемент с новой относительной частотой, то необходимо произвести пере-

сортировку полученного массива. Для этой цели из вспомогательного модуля вызывается функция sort. Она производит сортировку методом прямого выбора по полю ver_sum. Для обмена элементов массива значениями используется функция swap_elem(i, max) из того же вспомогательного модуля.

После того, как массив tree превратился в один элемент, поле кода этого элемента, состоящее из кодов символов сообщения, распределяется по элементам массива xcode2.

```
For i = 1 To kx
  For j = 1 To kx
   If tree(1).ind_in(j) = i Then xcode2(i) = tree(1).code(j)
  Next j
 Next i.
```

Аналогично методу Шеннона-Фано после кодирования вычисляются средняя длина элементарного кода l_ khaf и коэффициент относительной эффективности кодирования nu_ khaf по тем же формулам. Результаты выводятся в поля редакторов Text8 и Text9 соответственно.

Символы сообщения, их относительная частота и соответствующие коды выводятся в таблицу Shennon и таблицу Khaffman. А в поле редактора Text3 выводится значение энтропии hx, которая была вычислена ранее.

Кроме энтропии, вычисляется и другая информация, связанная с кодированием. Так, для сообщения длиной kx вычисляется средняя длина кодового слова и выводится в поле редактора Text4.

```
If Log(kx) / Log(2) > Int(Log(kx) / Log(2)) Or kx = 1 Then
  Text4.Text = Int(Log(kx) / Log(2)) + 1
Else
  Text4.Text = 1
End If
```

Далее вычисляется полная информационная избыточность d. Для этого сначала вычисляется избыточность dp, обусловленная неравномерным распределением символов сообщения.

```
If kx > 1 Then dp = 1 – hx / Log(kx) * Log(2) Else dp = 0,
```

а затем вычисляется избыточность ds, обусловленная статистической связью между символами сообщения.

```
hy_x = Два_символа()
If hx <> 0 Then ds = 1 – hy_x / hx Else ds = 0.
```

Для вычисления ds требуется вычислить условную энтропию hy_x. Она вычисляется с помощью функции Два_символа(), объявленной во вспомогательном модуле. На основе полученных значений вычисляется избыточность d и результат помещается в поле редактора Text5.

```
d = dp + ds – dp * ds
Text5.Text = CStr(d)
```

На этом действия обработчика для кнопки Command2 заканчиваются. В последнем пункте меню собраны команды вызова справочного материала.

Разработанная демонстрационная программа может быть использована в учебном процессе высших учебных заведений при изучении разделов, связанных с информацией, ее кодированием и передачей, для подготовки инженеров, специализирующихся в области автоматизированных систем управления и обработки информации.

Библиографический список

1. ***Вернер, М.*** Основы кодирования : учеб. для вузов / М. Вернер. – М. : Техносфера, 2004. – 275 с.

2. ***Игнатов, В. А.*** Теория информации и передачи сигналов : учеб. для вузов. – 2-е изд., перераб. и доп. – М. : Радио и связь, 1991 – 259 с.

3. ***Кричевский, Р. Е.*** Сжатие и поиск информации / Р. Е. Кричевский. – М. : Радио и связь, 1989. – 230 с.

4. ***Куликовский, Л.Ф.*** Теоретические основы информационных процессов / Л. Ф. Куликовский. – М. : Высшая школа, 1987. – 344 с.

5. ***Лидовский, В. В.*** Теория информации : учеб. пос. / В. В. Лидовский. – М. : Компания Спутник+, 2004. – 293 с.

6. ***Мастрюков, Д. В.*** Алгоритмы сжатия информации. Ч. 3. Алгоритмы группы LZ / Д. В. Мастрюков // Монитор. – 1994. – № 3. – С. 8–11.

7. ***Питерсон, У.*** Коды, исправляющие ошибки / У. Питерсон, Э. Уэлдон. – М. : Мир, 1976. – 378 с.

8. ***Самсонов, Б. Б.*** Теория информации и кодирование / Б. Б. Самсонов, Е. М. Плохов, А. И. Филоненков, Т. В. Кречет. – Ростов н/Д. : Феникс, 2002. – 266 с.

9. ***Семенюк, В. В.*** Экономное кодирование дискретной информации / В. В. Семенюк. – СПб. : СПб ГИТМО (ТУ), 2001. – 388 с.

10. ***Хаффман, Д. А.*** Метод построения кодов с минимальной избыточностью : пер. с англ. / Д. А. Хаффман // Кибернетический сборник. – М. : ИЛ, 1961. – Вып. 3. – С. 79–87.