Библиографический список

1. ***Вагин, В. Н.*** Кому и зачем нужен интернет-банкинг? / В. Н. Вагин // Мир интернет. – 1999. – № 9. – С. 20–22.
2. ***Веб-сервис***: XML, WSDL, SOAP и UDDI : пер с англ. – М. : Питер, 2003. – 256 с.
3. ***Основы Windows*** Communication Foundation: Пер. с англ. – М. : Русская Редакция ; БХВ-Петербург, 2008. – 384 с.
4. ***Программное*** обеспечение промежуточного слоя для обработки сообщений. – Режим доступа: http://www.cio-world.ru/infrastructure/system/29702/print.html, свободный. – Заглавие с экрана. – Яз. рус.
5. ***Система*** «Банк – Интернет – Клиент». – Режим доступа: http://www.moscow-bank.ru/ClientBank.htm, свободный. – Заглавие с экрана. – Яз. рус.

УДК 004.428.4

ПРОГРАММНЫЙ КОМПЛЕКС ДЛЯ ДЕМОНСТРАЦИИ ПРИМЕНЕНИЯ КОРРЕКТИРУЮЩЕГО КОДИРОВАНИЯ

М.О. Смирнова

В статье представлен программный продукт, который демонстрирует основные этапы применения корректирующих кодов. Дано описание основных компонентов программного продукта и возможностей использования при изучении теории информации и кодирования.

Ключевые слова*: информация, передача информации, демонстрационная программа, корректирующие коды, программный код.*

Key word: *information, transmission of information, the demonstration program, corrective code, the program code.*

В процессе хранения данных и передачи информации по сетям связи неизбежно возникают ошибки. Контроль целостности данных и исправление ошибок – важные задачи на многих уровнях работы с информацией (в частности, физическом, канальном, транспортном уровнях модели OSI).

Корректирующие коды, помехоустойчивые коды, коды обнаружения и исправления ошибки, применяются при передаче и обработке информации в вычислительной технике, телеграфии, телемеханике и технике связи, где возможны искажения сигнала в результате действия различного рода помех. Кодовые слова корректирующих кодов содержат информационные и проверочные разряды (символы) [5].

В процессе кодирования при передаче информации из информационных разрядов в соответствии с определенными для каждого корректирующего кода правилами формируются дополнительные символы – проверочные разряды. При декодировании из принятых кодовых слов по тем же правилам вновь формируют проверочные разряды и сравнивают их с принятыми; если они не совпадают, при передаче произошла ошибка. Существуют коды, обнаруживающие факт искажения сообщения, и коды, исправляющие ошибки, т.е. такие, с помощью которых можно восстановить первичную информацию [8].

Данный программный продукт состоит из пяти окон (форм): титульного окна, окна выбора задач, окна для кодов Хемминга, окна для циклических кодов и окна для теоретического материала и справки.

Программа «Корректирующие коды» написана на языке Visual Basic, который входит в комплект Microsoft Visual Studio 2005. Этот язык позволяет быстро и эффективно создавать программы, способные работать в операционной системе Windows [1, 3].

Программа «Корректирующие коды» обладает следующими возможностями:

- позволяет осуществлять ввод сообщения, состоящего из нулей и единиц, указанной длины;
- производит кодирование информационного сообщения с помощью кода Хемминга;
- производит кодирование введенного сообщения с помощью циклического кода;
- позволяет вносить в закодированное сообщение одну ошибку;
- производит восстановление закодированного сообщения;
- предоставляет возможность познакомиться с теоретическими основами указанных кодов;
- обладает инструкцией по использованию данного программного продукта.

На титульном окне располагается единственная кнопка, процедура-обработчик щелчка на которой приводит к показу следующего окна и сокрытию текущего титульного.

```
Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles Button1.Click
    Me.Hide()
    Form2.Show()
End Sub
```

Так как в окне выбора находятся только три кнопки, то модуль этого окна содержит только три подпрограммы, обрабатывающие щелчки по этим кнопкам. Обработчики для первых двух кнопок выполняют идентичные действия: скрывают текущее (второе) окно и показывают новое.

```
Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button1.Click
    Me.Hide()
    Form3.Show()
End Sub
```

```
Private Sub Button2_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button2.Click
    Me.Hide()
    Form4.Show()
End Sub
```

Обработчик третьей кнопки содержит один оператор End, который завершает работу программы.

Если сработал обработчик первой кнопки, то появляется окно для задачи кодирования информации кодом Хемминга. Данное окно содержит много элементов, большинство из которых имеют свои обработчики событий. Так, объект TextBox1, предназначенный для ввода длины информационного сообщения, имеет обработчик изменения значения текстового поля. При условии, что введено число в пределах 11, переменная k, хранящая длину сообщения, получает это значение после его преобразования из строкового представления в числовое. При попытке ввести число больше 11 или нецифровые символы выдаются сообщения об ошибках, и пользователь вынужден повторить ввод.

Следующий объект, подвергающийся обработке, текстовый редактор TextBox2. Он принимает информационное сообщение, которое должно состоять из нулей и единиц и соответствовать указанной в поле текста объекта TextBox1 длине. В противном случае выдаются сообщения об ошибках.

Еще одним обработчиком, имеющим важное значение, является обработчик щелчка по кнопке Button1. Он и запускает процесс кодирования полученной информации. Прежде всего происходит заполнение информационного массива mess нулями и единицами из поля редактора TextBox2. Это происходит с помощью функции Val, преобразующей каждый символ строки с введенной иинформацией в цифру.

```
ReDim mess(st.Length)
For i = 0 To st.Length – 1
   mess(i + 1) = Val(st.Chars(i))
Next i
```

При этом следует отметить, что массив mess предварительно переопределяется в соответствии с длиной информационного сообщения, так как он был объявлен динамическим: mess() As Integer.

Далее вычисляется количество дополнительных символов в коде и общая длина закодированного сообщения: n = k + r, на основе полученных значений производится подготовка массивов, необходимых для кодирования: ReDim cod(n), ReDim sindrom(r), ReDim H(r, n) и ReDim G(k, n). Первый массив cod нужен для хранения закодированного сообщения, второй массив sindrom – для синдрома, по которому в дальнейшем будет определяться ошибочный разряд закодированного сообщения, третий массив H – для проверочной матрицы, а четвертый массив G – для порождающей матрицы. После их переопределения массивы H и G заполняются нулями.

После подготовки массивов начинается их заполнение. Первой формируется проверочная матрица на основе массива, хранящего инфор-мационное сообщение. Если длина информационного сообщения больше количества единиц в нем, то первые k столбцов задаются посредством кольцевого сдвига информационного сообщения. Оставшиеся r столбцов матрицы H образуют единичную матрицу.

```
For i = 1 To r
    For j = i To k
       H(i, j – i + 1) = mess(j)
    Next j
    For j = 1 To i – 1
       H(i, k – i + j + 1) = mess(j)
    Next j
 Next i
 For i = 1 To r
    H(i, k + i) = 1
 Next i
```

Если информационное сообщение полностью состоит из единиц, то первые k столбцов заполняются единицами, а потом диагональ обнуляется. Оставшаяся часть опять заполняется как единичная матрица. За этим следует вывод полученной матрицы в поле объекта RichTextBox2.

```
For i = 1 To r
   For j = 1 To n
      RichTextBox2.Text = RichTextBox2.Text + CStr(H(i, j)) + " "
   Next j
Next i
```

На основе проверочной матрицы H строится проверочная матрица G. Первая часть, являющаяся подматрицей размером k × k, заполняется как единичная матрица, а оставшаяся часть заполняется элементами транспо-нированной матрицы H. Сформированная порождающая матрица выводится в поле текстового редактора RichTextBox1.

```
For i = 1 To k
        G(i, i) = 1
    Next i
    For i = 1 To r
        For j = 1 To k
            G(j, k + i) = H(i, j)
        Next j
    Next i
    For i = 1 To k
        For j = 1 To n
            RichTextBox1.Text = RichTextBox1.Text + CStr(G(i, j)) + " "
        Next j
    Next i
```

Последнее, что делает данный обработчик, – это кодирование информационного сообщения. Массив cod, предназначенный для закодированного сообщения, получается в результате умножения массива сообщения mess на порождающую матрицу G с последующим делением по модулю 2.

```
For i = 1 To n
        cod(i) = 0
        For j = 1 To k
            cod(i) = cod(i) + mess(j) * G(j, i)
        Next j
        cod(i) = cod(i) Mod 2
        TextBox3.Text = TextBox3.Text + CStr(cod(i)) + " "
    Next i
```

Параллельно с вычислением элементов массива cod происходит их вывод в поле редактора текста TextBox3.

В случае, когда сработал обработчик второй кнопки окна выбора задач, появляется окно для задачи кодирования информации циклическим кодом. Данное окно также содержит много элементов, которые имеют свои обработчики событий.

Следующим должен сработать обработчик кнопки Button4. Прежде всего, вычисляется степень порождающего многочлена, значение которого выводится в поле текстового редактора TextBox3. Далее, так же как в первой задаче, производится вычисление длины закодированного сообщения, переопределение величины массива информационного сообщения и его заполнение нулями и единицами из поля текстового редактора TextBox2.

```
n = k + r
ReDim mess(k)
For i = 1 To k
    mess(i) = Val(st.Chars(i – 1))
Next i
```

Следующим важным шагом является выбор неприводимого (элементарного) многочлена, называемого порождающим. Для этой цели служит объект ComboBox1, являющийся комбинированным списком. Его обработчик осуществляет процесс выбора небходимого многочлена из списка многочленов. Данный список создается на этапе закрузки окна задачи. Все коэффициенты многочленов хранятся в текстовом файле polinom.txt. В каждой строке файла находятся коэффициенты одного многочлена, поэтому файл считывается построчно и каждая строка вставляется в список ComboBox1.

Выбранная в списке строка коэффициентов помещается в переменную s и проверяется ее соответствие требуемой длине. Если такого соответствия нет, выдается сообщение об

ошибке и требуется повторный выбор. Если выбран многочлен соответствующей длины, то его коэффициенты при помощи функции Val помещаются в массив polinom, который перед этим был переопределен.

Так как порождающий многочлен должен делить по модулю 2 без остатка многочлен $x^k + 1$, следует сделать проверку правильности выбора порождающего многочлена. Это осуществляет обработчик щелчка по кнопке Button5.

Сначала производится переопределение массива pol2, который имеет только первый и последний элементы, равные единице, а остальные элементы заполнены нулями.

```
ReDim pol2(k + 1)
pol2(1) = 1
        pol2(k + 1) = 1
        For i = 2 To k
          pol2(i) = 0
        Next i
```

Далее производится деление по модулю 2 многочлена, хранимого в pol2, на порождающий многочлен, находящийся в массиве polinom. Результаты деления помещаются опять в массив pol2. В случае, когда все его элементы равны нулю, в поле редактора TextBox9 выводится сообщение о правильности выбора порождающего многочлена, в противном случае сообщается о неправильности выбора. Во втором слечае придется порождающий многочлен выбрать заново.

После выбора порождающего многочлена строится закодированное сообщение. Для этой цели используются два массива cod и pol1. Первый должен хранить результат кодирования, а второй – результат деления произведения информационного многочлена на одночлен степени r на порождающий многочлен.

Прежде всего, указанные массивы переопределяются. Затем в массив cod копируется массив сообщения mess.

```
        For i = 1 To k
          cod(i) = mess(i)
        Next i
        For i = k + 1 To r
          cod(i) = 0
        Next i
```

Второй цикл соответствует умножению многочлена сообщения на одночлен степени r. Теперь требуется определить остаток от деления по модулю 2 полученного произведения на порождающий многочлен. Для этого копируется массив cod в массив pol1.

```
        For i = 1 To n
          pol1(i) = cod(i)
        Next i
```

И только сейчас производится деление. Для данной операции применяется цикл while по параметру i, так как изменение параметра может быть неравномерным.

```
 i = 1
        Do While i <= k
          For j = 1 To r + 1
            pol1(i + j – 1) = (pol1(i + j – 1) + polinom(j)) Mod 2
          Next j
          Do While pol1(i) = 0 And i < n
            i = i + 1
          Loop
        Loop
```

Внутри этого цикла используются еще два цикла: for-next и while. Первый цикл осуществляет деление соответствующих элементов массива pol1 (коэффициентов многочлена, полученного в результате умножения, описанного выше) на элементы массива polinom, хранящего коэффициенты порождающего многочлена. Второй цикл подготавливает следующий шаг деления в случае, когда нужные элементы (коэффициенты) являются нулевыми.

Далее строится закодированное сообщение как сумма по модулю 2 (исключающее ИЛИ) элементов массива cod и массива pol1. Готовые компоненты сообщения выводятся в поле редактора TextBox6. А в поле текстового редактора TextBox7 выводится его длина.

```
For i = 1 To n
    cod(i) = (cod(i) + pol1(i)) Mod 2
    TextBox6.Text = TextBox6.Text + CStr(cod(i))
Next i
TextBox7.Text = CStr(n)
```

Все рассмотренные операции осуществляются обработчиком кнопки Button1.

Для демонстрации возможностей циклического кодирования по исправлению ошибки в один из разрядов закодированного сообщения следует внести изменения. Данной задачей занимается обработчик щелчка по кнопке Button2.

Чтобы данное изменение было возможным, поле редактора TextBox4 должно содердать номер разряда, в который вносится ошибка. Когда разряд не задан, никаких действий не происходит. А в случае задания номера разряда, не входящего в сообщение, выдается сообщение об ошибке и требуется новый ввод. Когда все задано правильно, в соответствующий разряд вносится ошибка. Это делается путем сложения элемента массива cod с единицей по модулю 2 (исключающее ИЛИ).

```
l = CInt(TextBox4.Text)
If l <= n Then
   cod(l) = (cod(l) + 1) Mod 2
   For i = 1 To n
      TextBox5.Text = TextBox5.Text + CStr(cod(i))
   Next i
```

«Испорченное» сообщение выводится в поле редактора текста TextBox5.

Теперь задача определить ошибочный разряд и его исправить. Это осуществляет обработчик кнопки Button3. Так как при этом будут производиться манипуляции с закодированным сообщением, то вводятся два дополнительных массива pol2 и cod2, что позволяет сохранить массивы сообщений без изменений. В данные массивы копируется массив с закодированным сообщением. Далее организуется процесс нахождения остатка от деления по модулю 2 массива pol1 на порождающий многочлен. Он продолжается до тех пор, пока не получается остаток больше единицы.

```
Do While i <= k
    For j = 1 To r + 1
       pol1(i + j – 1) = (pol1(i + j – 1) + polinom(j)) Mod 2
    Next j
    Do While pol1(i) = 0 And i < n
       i = i + 1
    Loop
Loop
```

Если на шаге цикла остаток имеет вес больше нуля, то производится сдвиг элементов массива pol1 влево на одну позицию. Для этого используется массив cod2, который хранит

исходную последовательность элементов. После осуществления сдвига процесс деления и нахождения остатка повторяется.

Когда на шаге цикла остаток перестанет превышать единицу, производится восстановление сдвига массива предыдущего шага. Это делается с массивом pol2, потому что массив pol1 в этот момент содержит остаток от деления. Затем к получившимуся массиву прибавляется полученный остаток от деления.

```
For j = 1 To n – 1
        pol2(j) = cod2(j + l)
    Next j
    For j = 1 To l
        pol2(n – l + j) = cod2(j)
    Next j
    For j = 1 To n
        pol2(j) = (pol1(j) + pol2(j)) Mod 2
Next j
```

Полученный массив подвергается теперь сдвигу вправо на количество левых сдвигов. Результат получается в массиве cod2, который и выводится в качестве исправленного закодированного сообщения в поле редактора TextBox8.

```
For j = l + 1 To n
        cod2(j) = pol2(j – l)
    Next j
    For j = 1 To l
        cod2(j) = pol2(n – l + j)
    Next j
    For i = 1 To n
        TextBox8.Text = TextBox8.Text + CStr(cod2(i))
    Next i
```

В заключение следует отметить, что в каждом окне задач имеется меню. В реализации обработчиков пунктов меню много общего, особенно это относится к пунктам, связанным с вызовом окна теоретического материала и справки.

Разработанный программный продукт может быть использован при подготовке по специальностям «Прикладная информатика в экономике» и «Организация и технология защиты информации» при изучении дисциплины «Теория информации и кодирования».

Библиографический список

1. ***Ананьев, А. И.*** Самоучитель Visual Basic 6.0 / А. И. Ананьев, А. Ф. Федоров. – СПб. : БХВ-Петербург, 2006. – 302 с.

2. ***Блейхут, Р.*** Теория и практика кодов, контролирующих ошибки / Р. Блейхут. – М. : Мир, 1986. – 234 с.

3. ***Браун, С.*** Visual Basic 6. Учебный курс / С. Браун. – СПб. : Питер, 2006. – 308 с.

4. ***Вернер, М.*** Основы кодирования / М. Вернер. – М. : Техносфера, 2004. – 345 с.

5. ***Золотарев, В. В.*** Помехоустойчивое кодирование. Методы и алгоритмы / В. В. Золотарев, Г. В. Овечкин. – М. : Телеком, 2004. – 126 с.

6. ***Кассами, Т.*** Теория кодирования / Т. Кассами, Н. Токура, Е. Ивадири, Я. Инагаки. – М. : Мир, 1978. – 325 с.

7. ***Лапонина, О. Р.*** Основы сетевой безопасности: криптографические алгоритмы и протоколы взаимодействия / О. Р. Лапонина. – М. : ИНТУИТ.ру, 2005. – 366 с.

8. ***Мак-Вильямс, Дж.*** Теория кодов, исправляющих ошибки / Дж. Мак-Вильямс, Дж. Слоэн. – М. : Радио и связь, 1979. – 275 с.

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