

Реализация алгоритма матричного кодирования с помощью языка C++

Ленкин Алексей Викторович

Приамурский государственный университет имени Шолом-Алейхема.

Студент

Лучанинов Дмитрий Васильевич

Приамурский государственный университет имени Шолом-Алейхема

старший преподаватель кафедры информационных систем, математики и методик преподаваний

Аннотация

В данной статье рассмотрен алгоритм матричного кодирования на примере кода Хеммина. Описана реализация данного алгоритма с помощью языка программирования C++.

Ключевые слова: C++, код Хемминга, матричное кодирование, шифрование

The algorithm implementation with C++ language

Lenkin Aleksei Viktorovich

Sholom-Aleichem Priamursky State University

student

Luchaninov Dmitry Vasilyevich

Sholom-Aleichem Priamursky State University

Senior lecturer of the Department of Information Systems, Mathematics and training methodic

Abstract

In the article the Hamming code matrix encoding algorithm is discussed. The realization of this algorithm using C ++ programming language is described.

Keywords: C ++, Hamming code, matrix encoding, encrypting.

На сегодняшний день проблема защиты информации стала наиболее актуальной в информационной среде. Одним из способов предотвращения кражи данных стало шифрование. На данный момент известно огромное число алгоритмов кодирования информации, и с каждым днём их число растёт. Но так, как при передаче данные могут исказиться, наиболее полезными считаются самоконтролирующиеся и самокорректирующиеся коды.

Алгоритм матричного кодирования был рассмотрен в статье В.В.Сапожникова, где было проведено исследование свойств кодов Хэмминга и их модификаций в системах функционального контроля [1].

Также, для объяснения работы кода Хемминга на простом примере, на сайте habrahabr была опубликована статья «Код Хэмминга. Пример работы алгоритма» [2].

Целью исследования является первый из кодов реализации матричного кодирования – код Хэмминга.

Кодирование на примере одной буквы [2]:

1. Предположим, что нужно закодировать букву 'h'. Первым делом надо привести её к двоичному представлению по таблице ASCII. Таким образом, она примет вид '01101000'.

2. Следующим шагом, который надо сделать, это вставить в получившуюся последовательность контрольных битов со значением '0' в позиции 2^I , где I принимает значения от 0 и увеличивается на 1, пока 2^I меньше длины слова. Последовательность после операции будет выглядеть так: 000011001000 (контрольные биты подчёркнуты).

3. Последний шаг – это посчитать значения контрольных битов, которое зависит от их информационных битов под его контролем. Контрольный бит под номером N контролирует все последующие N бит через каждые N бит, начиная с позиции N . Для того, чтобы узнать его значение, нужно взять его информационные биты и посмотреть сколько среди них '1', если полученное число чётное, то остаётся 0, в противном случае ставится '1'. Шаги 2-3 более удобно представить в таблице (табл. 1).

4.

Таблица 1 – Алгоритм создания кода Хэмминга

<u>0</u>	<u>0</u>	0	<u>0</u>	1	1	0	<u>0</u>	1	0	0	0	
X		X		X		X		X		X		1
	X	X			X	X			X	X		2
			X	X	X	X					X	3
							X	X	X	X	X	4

В итоге символ 'h' будет закодирован в следующую последовательность: 100011001000.

Для реализации кода были разработаны следующие функции алгоритма кодирования:

1. strsum.

2. strheming

Функция strsum переводит одиночный символ в его двоичное представление в таблице ASCII. Для этого средствами C++ получается номер символа в таблице ASCII и полученное число переводится в двоичную систему счисления.

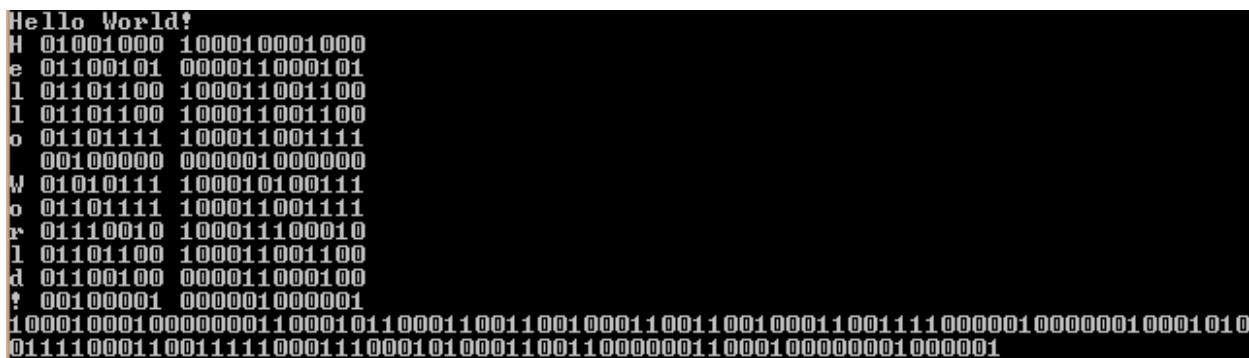
```
string strsum (string str)
{
    string sum;
    int a=(int)str[0];
    while (a!=0)
```

```
{
    sum+=char(a%2)+48;
    a/=2;
}
while (sum.length()!=8) sum+='0';
reverse(sum.begin(),sum.end());
return sum;
}
```

Функция `strheming` реализует алгоритм Хэмминга. В первом цикле двоичное представление символа заполняется контрольными битами со значением '0'. Во втором цикле осуществляется подсчёт значений контрольных битов. На вывод функции уходит уже закодированная последовательность.

```
string strheming (string str)
{
    int k=1;
    int d,sum;
    for (int i=0;i<str.length();i++)
    {
        if (k-1==i)
        {
            str.insert(i,"0");
            k*=2;
        }
    }
    k=1;
    for (int i=0;i<str.length();i++)
    {
        sum=0;
        if (k-1==i)
        {
            d=i;
            while (d<str.length())
            {
                for (int j=d;j<d+k;j++)
                {
                    if (str[j]=='1') sum++;
                }
                d+=k+1;
            }
            If (sum/2==0) str[i]='0'; else str[i]='1';
        }
    }
    return str;
}
```

Пример работы алгоритма на примере фразы «Hello World!» (рис. 1):



```
Hello World!  
H 01001000 100010001000  
e 01100101 000011000101  
l 01101100 100011001100  
l 01101100 100011001100  
o 01101111 100011001111  
o 01101111 100011001111  
W 01010111 100010100111  
o 01101111 100011001111  
r 01110010 100011100010  
l 01101100 100011001100  
d 01100100 000011000100  
! 00100001 000001000001  
10001000100000001100010110001100110010001100110010001100111100000100000010001010  
0111100011001111100011100010100011001100000011000100000001000001
```

Рисунок 1 – Пример вывода программы

Таким образом, реализация кода Хэмминга на языке C++ возможна и реализуется с помощью небольшого объема кода. Закодированный набор символов обладает достаточной криптостойкостью шифра для хранения информации, требующей защиты.

Библиографический список

1. Сапожников В.В. Ефанов Д.В. Исследование свойств кодов Хэмминга и их модификаций в системах функционального контроля // Автоматика на транспорте. 2015. №3 С. 311-337.
2. Код Хэмминга. Пример работы алгоритма [Электронный ресурс]. URL: <https://habrahabr.ru/post/140611/> (дата обращения 05.12.16).
3. Код Хэмминга [Электронный ресурс]. URL: https://ru.wikipedia.org/wiki/Код_Хэмминга (дата обращения 05.12.16).
4. Код Хэмминга. Пример работы алгоритма [Электронный ресурс]. URL: http://tpkrosreserv.ru/public/ucheb_mater/oti8.pdf (дата обращения 05.12.16).