Лабораторная работа № 3
Исследование циклического избыточного кода (CRC)

Цель работы: получение навыков использования циклического избыточного кода (CRC).

Содержание:

Краткие теоретические сведения

При передаче сигнала через любой канал связи возможно возникновение ошибок, которые могут приводить к искажению переносимой информации. Существует много методов для исправления подобных ошибок, но прежде чем исправлять, необходимо эти ошибки обнаружить. Для этого также существуют определенные методы, основанные на избыточности передаваемой информации, что позволяет не только выявлять наличие факта искажения информации, но и в ряде случаев устранять эти искажения.

Наиболее известные из методов обнаружения ошибок передачи данных являются:

- *Посимвольный контроль четности*, называемый также поперечным, подразумевает передачу с каждым байтом дополнительного бита, принимающего единичное значение по четному или нечетному количеству единичных битов в контролируемом байте. Посимвольный контроль четности прост как в программной, так и в аппаратной реализации, но его вряд ли можно назвать эффективным методом обнаружения ошибок, так как искажение более одного бита исходной последовательности резко снижает вероятность обнаружения ошибки передачи. Этот вид контроля обычно реализуется аппаратно в устройствах связи.
- *Поблочный контроль четности*, называемый продольным. Схема данного контроля подразумевает, что для источника и приемника информации заранее известно, какое число передаваемых символов будет рассматриваться ими как единый блок данных. В этой схеме контроля для каждой позиции разрядов в символах блока (поперек блока) рассчитываются свои биты четности, которые добавляются в виде обычного символа в конец блока. По сравнению с посимвольным контролем четности поблочный контроль четности обладает большими возможностями по обнаружению и даже корректировке ошибок передачи, но все равно ему не удается обнаруживать определенные типы ошибок.
- *Вычисление контрольных сумм.* В отличие от предыдущих методов для метода контрольных сумм нет четкого определения алгоритма. Каждый разработчик трактует понятие контрольной суммы по-своему. В простейшем виде контрольная сумма — это арифметическая сумма двоичных значений контролируемого блока символов. Но этот метод обладает практически теми же недостатками, что и предыдущие, самый главный из которых — нечувствительность контрольной суммы к четному числу ошибок в одной колонке и самому порядку следования символов в блоке.
- *Контроль циклически избыточным кодом* — CRC (Cyclical Redundancy Check). Это гораздо более мощный и широко используемый метод обнаружения ошибок передачи информации. Он обеспечивает обнаружение ошибок с высокой вероятностью. Кроме того, этот метод обладает рядом других полезных моментов, которые могут найти свое воплощение в практических задачах.

Циклический избыточный код (англ. Cyclic redundancy code, CRC) — алгоритм вычисления контрольной суммы, предназначенный для проверки целостности передаваемых данных. Алгоритм CRC обнаруживает все одиночные ошибки, двойные ошибки и ошибки в нечетном числе битов. Понятие циклических кодов достаточно широкое, однако на практике его обычно используют для обозначения только одной разновидности, использующей циклический контроль (проверку) избыточности. В связи с этим в англоязычной литературе CRC часто расшифровывается как Cyclic Redundancy Check.

CRC некоторой последовательности вычисляется на основании другой (исходной) битовой последовательности. Главная особенность (и практическая значимость) значения CRC состоит в том, что оно однозначно идентифицирует исходную битовую последовательность и поэтому используется в различных протоколах связи, а также для проверки целостности блоков данных, передаваемых различными устройствами. Благодаря относительной простоте алгоритм вычисления CRC часто реализуется на аппаратном уровне.

Основная идея вычисления CRC заключается в следующем. Исходная последовательность битов, которой могут быть и огромный файл, и текст размером несколько слов и даже символов, представляется единой последовательностью битов. Эта последовательность делится на некоторое фиксированное двоичное число (полином, CRC-полином, генераторный полином, англ. generator polinomial). Интерес представляет остаток от этого деления, который и является значением CRC. Все, что теперь требуется, — это некоторым образом запомнить его и передать вместе с исходной последовательностью. Приемник данной информации всегда может таким же образом выполнить деление и сравнить его остаток с исходным значением CRC. Если они равны, то считается, что исходное сообщение не повреждено, и т. д.

Степенью CRC-полинома *W* называют позицию самого старшего единичного бита. Например, степенью полинома 10011_2 равна 4.

Для вычисления CRC используют специальную т.н. полиномиальную арифметику. Вместо представления делителя, делимого (сообщения), частного и остатка в виде положительных целых чисел, можно представить их в виде полиномов с двоичными коэффициентами или в виде строки бит, каждый из которых является коэффициентом полинома.

Например, десятичное число 23 в шестнадцатеричной системе и двоичных системах будет иметь вид: $23_{10} = 17_{16} = 10111_2$, что совпадает с полиномом: $1 \cdot x^4 + 0 \cdot x^3 + 1 \cdot x^2 + 1 \cdot x^1 + 1 \cdot x^0$ или упрощенно $x^4 + x^2 + x^1 + x^0$.

И сообщение, и делитель могут быть представлены в виде полиномов, с которыми можно выполнять любые арифметические действия.

Предположим, что надо перемножить числа 1101_2 и 1011_2. Это можно выполнить, как умножение полиномов: $\left(x^3 + x^2 + x^0\right) \cdot \left(x^3 + x^1 + x^0\right)$ = $\left(x^6 + x^5 + x^3\right) + \left(x^4 + x^3 + x^1\right) + \left(x^3 + x^2 + x^0\right)$ = $1 \cdot x^6 + 1 \cdot x^5 + 1 \cdot x^4 + 3 \cdot x^3 + 1 \cdot x^2 + 1 \cdot x^1 + 1 \cdot x^0$.

Поскольку перемножались двоичные числа ($x = 2$), то в выражении $3 \cdot x^3$ возможен перенос бита от этого члена суммы в старший разряд ($3 \cdot x^3 = x^4 + x^3$), поэтому окончательное выражение будет иметь вид: $x^7 + x^3 + x^2 + x^1 + x^0$.

Если рассматривать коэффициенты полинома как изолированные друг от друга и для вычисления каждого из них использовать свои собственные правила, то можно получить различные виды полиномиальной арифметики. В частности широко используется т.н. «полиномиальная арифметика по модулю 2», когда коэффициенты складываются по модулю 2 без переноса – то есть коэффициенты могут иметь значения лишь 0 или 1.

Тогда вышерассмотренный пример в полиномиальной арифметике по модулю 2 будет иметь вид: $\left(x^3 + x^2 + x^0\right) \cdot \left(x^3 + x^1 + x^0\right)$ = $1 \cdot x^6 + 1 \cdot x^5 + 1 \cdot x^4 + 3 \cdot x^3 + 1 \cdot x^2 + 1 \cdot x^1 + 1 \cdot x^0$ = $x^6 + x^5 + x^4 + x^3 + x^2 + x^1 + x^0$.

Правила полиномиальной арифметики по модулю 2 можно использовать и при обычной арифметике, так как действия, выполняемые во время вычисления CRC, являются арифметическими операциями без учета переносов.

Сложение двух чисел в CRC арифметике полностью аналогично обычному арифметическому действия за исключением отсутствия переносов из разряда в разряд. Это означает, что каждая пара битов определяет результат своего разряда вне зависимости от результатов других пар. Например:

$$\begin{array}{r} 10011011 \\ +\ 11001010 \\ \hline 01010001 \end{array} \qquad \begin{array}{r} 10011011 \\ -\ 11001010 \\ \hline 01010001 \end{array}.$$

Т.е. и при соложении и при вычитании над каждым битом в отдельности выполняется операция, аналогичная операции *XOR*, поэтому в CRC арифметике две арифметические операции (сложение и вычитание) заменяются операцией *XOR*.

Умножение, как и в обычной арифметике, считается суммой значений первого сомножителя, сдвинутых в соответствии со значением второго сомножителя. Причем при суммировании так же используется CRC сложение. Например:

$$\begin{array}{r} 1101 \\ 1011 \\ \hline 1101 \\ 1101. \\ 0000.. \\ 1101... \\ \hline 1111111 \end{array}$$

Деление в CRC арифметике определяется аналогично с учетом того, что вычитание выполняется по правилам CRC арифметики. Например:

```
1 1 0 1 0 1 1 0 1 1 0 0 0 0 | 1 0 0 1 1
1 0 0 1 1                   | 1 1 0 0 0 0 1 0 1 0
  1 0 0 1 1
  1 0 0 1 1
    0 0 0 0 1
    0 0 0 0 0
      0 0 0 1 0
      0 0 0 0 0
        0 0 1 0 1
        0 0 0 0 0
          0 1 0 1 1
          0 0 0 0 0
            1 0 1 1 0
            1 0 0 1 1
              0 1 0 1 0
              0 0 0 0 0
                1 0 1 0 0
                1 0 0 1 1
                  0 1 1 1 0
                  0 0 0 0 0
                    1 1 1 0 = Остаток
```

В CRC арифметике считается что число *A* делится на число *B*, если его можно получить из нуля путем некоторого числа сложений сдвинутого числа B. Например, пусть A=0111010110$_2$ и B=11$_2$, тогда

```
= 0 1 1 1 0 1 0 1 1 0
+ . . . . . . . 1 1 .
+ . . . . . 1 1 . . . .
+ . . . . 1 1 . . . . .
+ . . . 1 1 . . . . . .
  . 1 1 . . . . . . . .
```

Перед началом вычисления CRC исходное сообщение следует дополнить *W* нулями справа и выполнить деление по правилам CRC-арифметики. Рассмотрим пример:

Исходное сообщение: 1101011011
Генераторный полином: 10011
Сообщение, дополненное *W* битами: 11010110110000

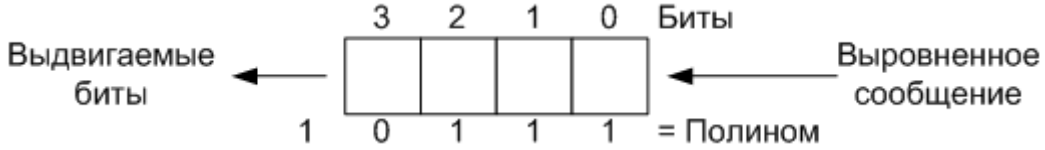
Как видно, контрольная сумма (CRC) в этом примере равна 1110. Как правило, контрольная сумма добавляется к исходному сообщению (в нашем примере передаваемое сообщение будет равно 11010110111110) и полученное расширенное сообщение передается через канал связи.

На другом конце канала приемник может сделать одно из возможных действий (оба варианта совершенно равноправны):

1. Выделить текст собственно сообщения, вычислить для него контрольную сумму (не забыв при этом дополнить сообщение *W* битами), и сравнить ее с переданной.
2. Вычислить контрольную сумму для всего переданного сообщения (без добавления нулей), и посмотреть, получится ли в результате нулевой остаток.

Поскольку исходное сообщение может быть очень большим (до нескольких МБайтов) и так же из-за того, что для получения CRC используется CRC-арифметика, использовать обычную компьютерную операцию деления нельзя.

Рассмотрим на примере как проще всего программно реализовать алгоритм вычисления контрольной суммы. Пусть полином с *W* =4 равен 10111. Тогда для выполнения деления потребуется 4-битный регистр:



и алгоритм получения контрольной суммы будет иметь вид:

```
Загрузим регистр нулевыми битами
Дополним хвостовую часть сообщения W нулевыми битами
While (пока еще есть необработанные биты)
  Begin
    Сдвинем регистр на 1 бит влево и поместим очередной
                еще не обработанный бит из сообщения в 0 позицию регистра.
    If (из регистра был выдвинут бит со значением "1")
       Регистр = Регистр XOR Полином
  End
Теперь в регистре содержится остаток
```

Такая реализация самая простая и понятная для понимания, однако, она имеет ряд недостатков – она требует большого числа машинных операций и выполняется достаточно медленно, что существенно ограничивает ее применение в современных системах связи. К тому же побитовая реализация достаточно трудоемка на языках высокого уровня. Для ускорения вычисления расчета был предложен т.н. табличный алгоритм, который работает не с отдельными битами, а с блоками бит. Такими блоками могут быть полубайты (4 бита), байты, (8 бит), слова (16 бит) и длинные слова (32 бита), и даже более длинные фрагменты, если позволяют ресурсы системы.

Задание для выполнения

1. Написать функцию, реализующую вычисление CRC согласно варианту. В качестве входных параметров использовать:
   a. входную последовательность 0 и 1,
   b. длину последовательности K,
2. Написать функцию, которая бы генерировала случайную последовательностей 0 и 1 длиной K=1000 (или взять из предыдущих работ).
3. Проверить правильность функционирования ранее написанных функций вычисления CRC. Для этого сначала вычислить CRC и добавить его в конце последовательности. После этого вычислить CRC полученной последовательности и убедиться, что она будет равна нулю.

Номер варианта	Название	Образующий полином	Примечание
1	CRC-16-IBM	$x^{16}+x^{15}+x^2+1$	Bisync, Modbus, USB, ANSI X3.28, многие другие; также известен как CRC-16 и CRC-16-ANSI
2	CRC-8-CCITT	x^8+x^2+x+1	(ATM HEC), ISDN Header Error Control and Cell Delineation ITU-T I.432.1 (02/99)
3	CRC-24	$x^{24}+x^{22}+x^{20}+x^{19}+x^{18}+x^{16}+x^{14}+x^{13}+x^{11}+x^{10}+$ $+x^8+x^7+x^6+x^3+x+1$	FlexRay
4	CRC-16-T10-DIF	$x^{16}+x^{15}+x^{11}+x^9+x^8+x^7+x^5+x^4+x^2+x+1$	SCSI DIF
5	CRC-5-USB	x^5+x^2+1	USB token packets
6	CRC-32C (Castagnoli)	$x^{32}+x^{28}+x^{27}+x^{26}+x^{25}+x^{23}+x^{22}+x^{20}+x^{19}+x^{18}+x^{14}+$ $+x^{13}+x^{11}+x^{10}+x^9+x^8+x^6+1$	iSCSI, G.hn payload
7	CRC-10	$x^{10}+x^9+x^5+x^4+x+1$	
8	CRC-24-Radix-64	$x^{24}+x^{23}+x^{18}+x^{17}+x^{14}+x^{11}+x^{10}+x^7+x^6+x^5+x^4+$ $+x^3+x+1$	OpenPGP
9	CRC-8-Dallas/Maxim	$x^8+x^5+x^4+1$	1-Wire bus
10	CRC-32K (Koopman)	$x^{32}+x^{30}+x^{29}+x^{28}+x^{26}+x^{20}+x^{19}+x^{17}+x^{16}+x^{15}+$ $+x^{11}+x^{10}+x^7+x^6+x^4+x^2+x+1$	
11	CRC-15-CAN	$x^{15}+x^{14}+x^{10}+x^8+x^7+x^4+x^3+1$	
12	CRC-4-ITU	x^4+x+1	ITU G.704 [2)]
13	CRC-32Q	$x^{32}+x^{31}+x^{24}+x^{22}+x^{16}+x^{14}+x^8+x^7+x^5+x^3+x+1$	aviation; AIXM
14	CRC-16-CCITT	$x^{16}+x^{12}+x^5+1$	X.25, HDLC, XMODEM,

Номер варианта	Название	Образующий полином	Примечание
			Bluetooth, SD и др.
15	CRC-8	$x^8 + x^7 + x^6 + x^4 + x^2 + 1$	ETSI EN 302 307 [3], 5.1.4
16	CRC-32-IEEE 802.3	$x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1$	V.42, MPEG-2, PNG, POSIX cksum
17	CRC-6-ITU	$x^6 + x + 1$	ITU G.704 [2]
18	CRC-64-ISO	$x^{64} + x^4 + x^3 + x + 1$	HDLC — ISO 3309
19	CRC-11	$x^{11} + x^9 + x^8 + x^7 + x^2 + 1$	FlexRay [4]
20	CRC-16-CCITT	$x^{16} + x^{12} + x^5 + 1$	X.25, HDLC, XMODEM, Bluetooth, SD и др.
21	CRC-5-EPC	$x^5 + x^3 + 1$	Gen 2 RFID [5]
22	CRC-6-ITU	$x^6 + x + 1$	ITU G.704 [2]
23	CRC-16-DNP	$x^{16} + x^{13} + x^{12} + x^{11} + x^{10} + x^8 + x^6 + x^5 + x^2 + 1$	DNP, IEC 870, M-Bus
24	CRC-7	$x^7 + x^3 + 1$	системы телекоммуникации, ITU-T G.707, ITU-T G.832, MMC, SD. [6]
25	CRC-8-Dallas/Maxim	$x^8 + x^5 + x^4 + 1$	1-Wire bus
26	CRC-8-SAE J1850	$x^8 + x^4 + x^3 + x^2 + 1$	
27	CRC-30	$x^{32} + x^{29} + x^{21} + x^{20} + x^{15} + x^{13} + x^{12} + x^{11} + x^8 + x^7 + x^6 + x^2 + x + 1$	CDMA
28	CRC-5-ITU	$x^5 + x^4 + x^2 + 1$	ITU G.704 [2]
29	CRC-12	$x^{12} + x^{11} + x^3 + x^2 + x + 1$	системы телекоммуникации [7]
30	CRC-64-ECMA-182	$x^{64} + x^{62} + x^{57} + x^{55} + x^{54} + x^{53} + x^{52} + x^{47} + x^{46} + x^{45} + x^{40} + x^{39} + x^{38} + x^{37} + x^{35} + x^{33} + x^{32} + x^{31} + x^{29} + x^{27} + x^{24} + x^{23} + x^{22} + x^{21} + x^{19} + x^{17} + x^{13} + x^{12} + x^{10} + x^9 + x^7 + x^4 + x + 1$	

Требования к оформлению отчета

1. Протокол оформляется каждым студентом группы в отдельности.
2. Протокол должен содержать:
    a. Титульный лист.
    b. Задание согласно варианту.
    c. Листинги программы.
    d. Результаты работы программы.
    e. Выводы о проделанной работе.
3. Защита работы проводится каждым студентом персонально.

Контрольные вопросы

1. Как расшифровывается и переводится CRC?
2. Для каких целей обычно используется CRC?
3. Дайте определение CRC.
4. В чем принципиальная особенность CRC-арифметики?
5. Опишите простейший алгоритм вычисления CRC.

Литература

1. Ross N. Williams. Элементарное руководство по CRC□алгоритмам обнаружения ошибок.
2. Synchronous frame structures used at 1544, 6312, 2048, 8448 and 44 736 kbit/s hierarchical levels.
3. EN 302 307. Digital Video Broadcasting (DVB); Second generation framing structure, channel coding and modulation systems for Broadcasting, Interactive Services, News Gathering and other broadband satellite applications (DVB-S2).
4. FlexRay Protocol Specification version 2.1 Revision A. — 22 декабря 2005. — С. 93.
5. Class-1 Generation-2 UHF RFID Protocol version.
6. G.832 : Transport of SDH elements on PDH networks — Frame and multiplexing structures.
7. T. V. Ramabadran, S. S. Gaitonde. A tutorial on CRC computations // IEEE Micro. - 1988. — Т. 8. — № 4. — С. 62—75. — DOI:10.1109/40.7773.