Лабораторна робота №3
Дослідження циклічного надлишкового коду (CRC)

Мета роботи: здобуття навичок використання циклічного надлишкового коду (CRC).

Зміст:

Короткі теоретичні відомості

При передачі сигналу через будь-який канал зв'язку можливе виникнення помилок, які можуть призводити до спотворення інформації, що переноситься. Існує багато методів для виправлення подібних помилок, але, перш ніж виправляти, необхідно ці помилки виявити. Для цього також існують певні методи, засновані на надмірності інформації, що передається, що дозволяє не тільки виявляти наявність факту спотворення інформації, але і в ряді випадків усувати ці спотворення.

Найбільш відомими з методів виявлення помилок передачі даних є:

- *Посимвольний контроль парності* , званий також поперечним, має на увазі передачу з кожним байтом додаткового біта, що приймає одиничне значення по парному чи непарному кількості одиничних бітів у контрольованому байті. Посимвольний контроль парності простий як і програмної, і у апаратної реалізації, та його навряд можна назвати ефективним методом виявлення помилок, оскільки спотворення більше біта вихідної послідовності різко знижує ймовірність виявлення помилки передачі. Цей вид контролю зазвичай реалізується апаратно у пристроях зв'язку.
- *Побічний контроль парності* , званий поздовжнім. Схема даного контролю передбачає, що з джерела і приймача інформації заздалегідь відомо, скільки символів, що передаються, буде розглядатися ними як єдиний блок даних. У цій схемі контролю кожної позиції розрядів у символах блоку (поперек блоку) розраховуються свої біти парності, які додаються у вигляді звичайного символу в кінець блоку. У порівнянні з посимвольним контролем парності побічний контроль парності має великі можливості щодо виявлення і навіть коригування помилок передачі, але все одно йому не вдається виявляти певні типи помилок.
- *Обчислення контрольних сум.* На відміну від попередніх методів для методу контрольних сум, немає чіткого визначення алгоритму. Кожен розробник трактує поняття контрольної суми по-своєму. У найпростішому вигляді контрольна сума — арифметична сума двійкових значень контрольованого блоку символів. Але цей метод має практично ті ж недоліки, що й попередні, найголовніший з яких — нечутливість контрольної суми до парного числа помилок в одній колонці і самому порядку символів у блоці.
- *Контроль циклічно надлишковим кодом* - CRC (Cyclical Redundancy Check). Це набагато потужніший і широко використовуваний метод виявлення помилок передачі. Він забезпечує виявлення помилок із високою ймовірністю. Крім того, цей метод має низку інших корисних моментів, які можуть знайти своє втілення у практичних завданнях.

Циклічний надлишковий код (англ. Cyclic redundancy code , CRC) — алгоритм обчислення контрольної суми, призначений для перевірки цілісності даних, що передаються. Алгоритм CRC виявляє всі поодинокі помилки, подвійні помилки та помилки в непарному числі бітів. Поняття

циклічних кодів досить широке, проте на практиці його зазвичай використовують для позначення лише одного різновиду, що використовує циклічний контроль (перевірку) надмірності. У зв'язку з цим в англомовній літературі CRC часто розшифровується як Cyclic Redundancy Check .

CRC деякої послідовності обчислюється на підставі іншої (вихідної) бітової послідовності. Головна особливість (і практичне значення) значення CRC полягає в тому, що воно однозначно ідентифікує вихідну бітову послідовність і тому використовується в різних протоколах зв'язку, а також для перевірки цілісності блоків даних, що передаються різними пристроями. Завдяки відносній простоті алгоритм обчислення CRC часто реалізується на апаратному рівні.

Основна ідея обчислення CRC полягає у наступному. Вихідна послідовність бітів, якою може бути і великий файл, і текст розміром кілька слів і навіть символів, є єдиною послідовністю бітів. Ця послідовність ділиться на деяке фіксоване двійкове число (поліном, CRC-поліном, генераторний поліном, англ. generator polinomial). Інтерес представляє залишок від цього поділу, який є значенням CRC. Все, що тепер потрібно, — це певним чином запам'ятати його і передати разом із вихідною послідовністю. Приймач даної інформації завжди може так само виконати поділ і порівняти його залишок з вихідним значенням CRC. Якщо вони рівні, то вважається, що вихідне повідомлення не пошкоджене, і т.д.

Ступенем CRC-полінома *W* називають позицію найстаршого одиничного біта. Наприклад, ступенем полінома 10011_2 дорівнює 4.

Для обчислення CRC використовують спеціальну т.зв. поліноміальну математику. Замість подання дільника, поділеного (повідомлення), приватного та залишку у вигляді позитивних цілих чисел, можна уявити їх у вигляді поліномів з двійковими коефіцієнтами або у вигляді рядка біт, кожен з яких є коефіцієнтом полінома.

Наприклад, десяткове число 23 у шістнадцятковій системі та двійкових системах матиме вигляд: $23_{10} = 17_{16} = 10111_2$, що збігається з поліномом: $1 \cdot x^4 + 0 \cdot x^3 + 1 \cdot x^2 + 1 \cdot x^1 + 1 \cdot x^0$ або спрощено $x^4 + x^2 + x^1 + x^0$.

І повідомлення, і дільник можуть бути представлені у вигляді поліномів, з якими можна виконувати будь-які арифметичні дії.

Припустимо, що треба перемножити числа 1101_2 і 1011_2. Це можна виконати як множення поліномів: $\left(x^3 + x^2 + x^0\right) \cdot \left(x^3 + x^1 + x^0\right) = \left(x^6 + x^5 + x^3\right) + \left(x^4 + x^3 + x^1\right) + \left(x^3 + x^2 + x^0\right) = 1 \cdot x^6 + 1 \cdot x^5 + 1 \cdot x^4 + 3 \cdot x^3 + 1 \cdot x^2 + 1 \cdot x^1 + 1 \cdot x^0$.

Оскільки перемножувалися двійкові числа ($x = 2$), то виразі $3 \cdot x^3$ можливий перенесення біта від цього члена суми у старший розряд ($3 \cdot x^3 = x^4 + x^3$), тому остаточне вираз матиме вид: $x^7 + x^3 + x^2 + x^1 + x^0$.

Якщо розглядати коефіцієнти полінома як ізольовані друг від друга й у обчислення кожного їх використовувати свої власні правила, можна отримати різні види поліноміальної арифметики. Зокрема, широко використовується т.зв. «поліноміальна арифметика за модулем 2», коли коефіцієнти складаються за модулем 2 без перенесення - тобто коефіцієнти можуть мати значення лише 0 або 1.

Тоді вищерозглянутий приклад у поліноміальній арифметиці за модулем 2 матиме вигляд: $\left(x^3 + x^2 + x^0\right) \cdot \left(x^3 + x^1 + x^0\right) = 1 \cdot x^6 + 1 \cdot x^5 + 1 \cdot x^4 + 3 \cdot x^3 + 1 \cdot x^2 + 1 \cdot x^1 + 1 \cdot x^0 = x^6 + x^5 + x^4 + x^3 + x^2 + x^1 + x^0$.

Правила поліноміальної арифметики за модулем 2 можна використовувати і за звичайної арифметики, оскільки дії, що виконуються під час обчислення CRC, є арифметичними операціями без урахування переносів.

Додавання двох чисел в CRC арифметиці цілком аналогічно звичайному арифметичному дії крім відсутності переносів з розряду в розряд. Це означає, що кожна пара бітів визначає результат свого розряду незалежно від інших пар. Наприклад:

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

Тобто. і при складанні і при відніманні над кожним бітом окремо виконується операція, аналогічна операції *XOR* , тому в CRC арифметиці дві арифметичні операції (складення та віднімання) замінюються операцією *XOR* .

Множення, як і звичайній арифметиці, вважається сумою значень першого співмножника, зрушених відповідно до значення другого співмножника. Причому при підсумовуванні також використовується CRC додавання. Наприклад:

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

Розподіл у CRC арифметиці визначається аналогічно з урахуванням того, що віднімання виконується за правилами CRC арифметики. Наприклад:

```
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
```

У CRC арифметиці вважається що число *A* ділиться на число *У* якщо його можна отримати з нуля шляхом деякого числа додавань зсунутого числа B. Наприклад, нехай A=0111010110 $_2$ і B=11 $_2$, тоді

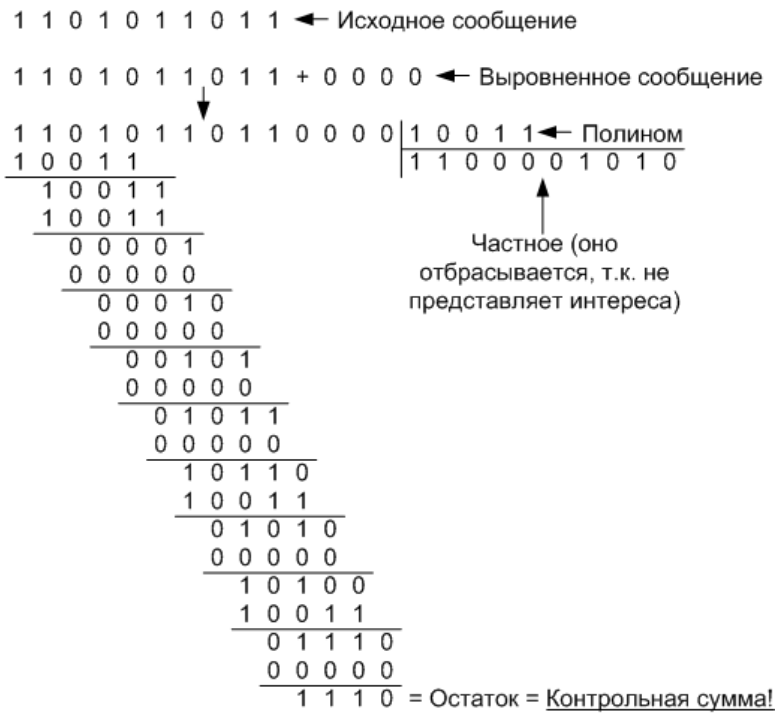
```
= 0 1 1 1 0 1 0 1 1 0
+ . . . . . . . 1 1 .
+ . . . . 1 1 . . . .
+ . . . 1 1 . . . . .
  . 1 1 . . . . . . .
```

Перед початком обчислення CRC вихідне повідомлення слід доповнити *W* нулями праворуч та виконати поділ за правилами CRC-арифметики. Розглянемо приклад:

Вихідне повідомлення: 1101011011

Генераторний поліном: 10011

Повідомлення, доповнене *W* бітами: 11010110110000



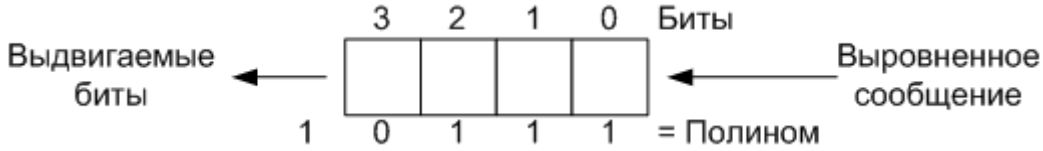
Як видно, контрольна сума (CRC) у цьому прикладі дорівнює 1110. Як правило, контрольна сума додається до вихідного повідомлення (у нашому прикладі повідомлення, що передається, буде дорівнює 11010110111110) і отримане розширене повідомлення передається через канал зв'язку.

На іншому кінці каналу приймач може зробити одну з можливих дій (обидва варіанти абсолютно рівноправні):

1. Виділити текст власне повідомлення, обчислити йому контрольну суму (не забувши у своїй доповнити повідомлення *W* бітами), і порівняти її з переданої.
2. Обчислити контрольну суму для всього переданого повідомлення (без додавання нулів) і подивитися, чи вийде в результаті нульовий залишок.

Оскільки вихідне повідомлення може бути дуже великим (до декількох МБайтів) і так само через те, що для отримання CRC використовується CRC-арифметика, використовувати звичайну комп'ютерну операцію поділу не можна.

Розглянемо з прикладу як найпростіше програмно реалізувати алгоритм обчислення контрольної суми. Нехай поліном з *W* =4 дорівнює 10111. Тоді для виконання поділу потрібно 4-бітний регістр:



та алгоритм отримання контрольної суми матиме вигляд:

```
Завантажимо регістр нульовими бітами
Доповнимо хвостову частину повідомлення W нульовими бітами
While ( поки ще є необроблені біти )
  Begin
Посунемо регістр на 1 біт вліво і помістимо черговий
    ще не оброблений біт із повідомлення в 0 позицію регістра.
    If ( З регістру був висунутий біт зі значенням "1" )
Реєстр = Реєстр XOR Поліном
  End
Тепер у регістрі міститься залишок
```

Така реалізація найпростіша і зрозуміліша для розуміння, однак, вона має низку недоліків – вона вимагає великої кількості машинних операцій і виконується досить повільно, що суттєво

обмежує її застосування у сучасних системах зв'язку. До того ж побітова реалізація є досить трудомісткою мовами високого рівня. Для прискорення обчислення розрахунку було запропоновано т.зв. табличний алгоритм, який працює не з окремими бітами, а з блоками бітів. Такими блоками можуть бути напівбайти (4 біти), байти, (8 біт), слова (16 біт) та довгі слова (32 біти), і навіть довші фрагменти, якщо дозволяють ресурси системи.

Завдання для виконання

1. Написати функцію, що реалізує обчислення CRC згідно з варіантом. Як вхідні параметри використовувати:
   a. вхідну послідовність 0 та 1,
   b. довжину послідовності K,
2. Написати функцію, яка генерувала б випадкову послідовностей 0 і 1 довжиною $K=1000$ (або взяти з попередніх робіт).
3. Перевірити правильність функціонування раніше написаних функцій обчислення CRC . Для цього спочатку обчислити CRC та додати його в кінці послідовності. Після цього обчислити CRC отриманої послідовності і переконатися, що вона дорівнюватиме нулю.

Номер варіанта	Назва	Утворює поліном	Примітка
1	CRC-16-IBM	$x^{16}+x^{15}+x^2+1$	Bisync , Modbus , USB , ANSI X 3.28, багато інших; також відомий як CRC -16 і CRC - 16 - ANSI
2	CRC-8-CCITT	x^8+x^2+x+1	(ATM HEC), ISDN Header Error Control і 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)]

Номер варіанта	Назва	Утворює поліном	Примітка
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,Bluetooth, SD і ін .
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 і ін .
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	$x^7 + x^3 + 1$	Системи телекомунікації, 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	$x^{12} + x^{11} + x^3 + x^2 + x + 1$	системи телекомунікації [7]
30	CRC-64-ECMA-182	$x^{64} + x^{62} + x^{57} + x^{55} + x^{54} + x^{53} + + x^{52} + x^{47} + x^{46} + x^{45} + x^{40} + + x^{39} + x^{38} + x^{37} + x^{35} + x^{33} + x^{32} + x^{31} + x^{29} + x^{27} + x^{24} + x^{23} + + x^{22} + x^{21} + x^{19} + x^{17} + x^{13} + x^{12} + x^{10} + x^9 + x^7 + x^4 + x + 1$	

Вимоги до оформлення звіту

1. Протокол оформляється кожним студентом групи окремо.
2. Протокол повинен містити:
    a. Титульна сторінка.
    b. Завдання згідно з варіантом.
    c. Лістинги програми.
    d. Результати роботи програми.
    e. Висновки про виконану роботу.
3. Захист роботи проводиться кожним студентом персонально.

Контрольні питання

1. Як розшифровується та перекладається CRC?
2. З якою метою зазвичай використовується CRC?
3. Дайте визначення CRC.
4. У чому важлива особливість CRC-арифметики?
5. Опишіть найпростіший алгоритм обчислення CRC.

Література

1. Росс N. Williams. Елементарний посібник з CRC алгоритмів виявлення помилок.
2. Synchronous frame structures використовувалися на 1544, 6312, 2048, 8448 і 44 736 кбіт/с ієрархічних рівнів.
3. EN 302 307. Digital Video Broadcasting (DVB); Завтра generation framing structure, канал coding і modulation systems для Broadcasting, Interactive Services, News Gathering та інші broadband satellite applications (DVB-S2).
4. FlexRay Protocol Specification version 2.1 Revision A. - 22 грудня 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. TV Ramabadran, SS Gaitonde. A tutorial on CRC computations // IEEE Micro. -1988. - Т. 8. - № 4. - С. 62-75. - DOI: 10.1109/40.7773.