Практическая работа №1

«Изучение алгоритмов обнаружения и исправления ошибок при передаче данных по каналам связи»

Введение

У каналов передачи данных большой диапазон различных характеристик. Например, при применении оптического волокна, вероятность ошибки крайне низка, поэтому потеря данных происходит редко. Однако, в беспроводных сетях их появление считается нормой и потери информации происходят регулярно. Но для того чтобы полностью исключить ошибки при передаче данных, потребуется применить довольно сложные методы, которые скажутся на производительности вычислительной сети [1].

Существует две основные стратегии для борьбы с ошибками, но каждая из них основывается на добавлении к передаваемым данным некоторой избыточной информации. В одном случае этой информации должно быть достаточно, чтобы принимающая сторона могла выявить, какие данные должны были прийти. В другом случае избыточной информации должно быть достаточно только для того, чтобы получатель понял, что произошла ошибка, и запросил повторную передачу. Первая стратегия использует коды, называющиеся корректирующими или кодами с исправлением ошибок (error-correcting codes, ECC). Вторая – коды с обнаружением ошибок (error-detecting codes, EDC). Использование кода с исправлением ошибок часто называют прямым исправлением ошибок (Forward Error Correction – FEC) [1].

В высоконадёжных каналах, таких как оптоволокно, дешевле использовать код с обнаружением ошибок и просто заново передавать случайные повреждённые блоки. А, например, беспроводные соединения, в которых может возникать множество ошибок, чаще используют коды с избыточностью, достаточной для того, чтобы приёмник мог определить, какие данные должны были прийти. Прямое исправление ошибок применяется в шумных каналах, так как вероятность ошибки при повторной передаче так же велика, как и при первой [1].

Ни ECC, ни EDC не позволят справиться со всеми возможными ошибками, поскольку лишние биты, передаваемые для повышения надёжности, также могут быть повреждены в пути. Для того чтобы избежать необнаруженных ошибок, необходимо использовать достаточно надёжные коды [1].

Для того чтобы определить, какой метод лучше подойдёт в конкретной ситуации, нужно понять, какой тип ошибок более вероятен: чаще возникают однобитные или многобитные ошибки. Объясняется это физическими процессами, вызывающими неполадки, такими как глубокое замирание беспроводного канала или временная электрическая помеха в кабельном канале [1].

Последовательность ошибок может быть лучше одиночных. Компьютер всегда отправляет данные блоками. Например, размер блока равен 1000 бит, а вероятность ошибки равна 0,001 на один бит. Если бы ошибки были независимыми, то почти в каждом

блоке обнаруживалась бы ошибка. Однако если возникнет целая последовательность ошибок, то в среднем из ста блоков только один будет повреждён. С другой стороны, последовательность ошибок исправить намного сложнее, чем изолированные ошибки. Существуют и другие типы ошибок. Иногда местоположение ошибки известно [1].

Коды с исправлением ошибок и коды с обнаружением ошибок используются весьма широко, так как вопрос надёжности важен. Коды исправления ошибок можно встретить на физическом уровне, особенно когда речь идёт о зашумлённых каналах, и на более высоких уровнях, особенно при рассылке мультимедийной информации в режиме реального времени. Коды обнаружения ошибок применяются на канальном, сетевом и транспортном уровнях [1].

Методы с обнаружением ошибок

Избыточные коды

При избыточном кодировании исходный двоичный код представляется в виде последовательностей нескольких битов – символами (порциями), каждая из которых заменяется новой последовательностью, содержащей большее количество бит, чем исходная [2,3].

К методам избыточного кодирования относятся: 4B/5B, 5B/6B, 8B/10B, 64B/66B [3].

Код 4B/5B (рисунок 1), используется в технологии Fast Ethernet. Исходные символы длиной 4 бита заменяются символами длиной 5 бит. В коде 4B/5B результирующие символы могут содержать 32 битовые комбинации (2^5=32). Таким образом, количество избыточных (запрещённых) кодов: 32-16=16. Появление запрещённых символов означает ошибку в передаваемых данных [2,3].

Data		4B5B code	Data		4B5B code
(Hex)	(Binary)		(Hex)	(Binary)	
0	0000	11110	8	1000	10010
1	0001	01001	9	1001	10011
2	0010	10100	A	1010	10110
3	0011	10101	B	1011	10111
4	0100	01010	C	1100	11010
5	0101	01011	D	1101	11011
6	0110	01110	E	1110	11100
7	0111	01111	F	1111	11101

Рисунок 1 – Код 4B/5B

Контроль по паритету

Контроль чётности (посимвольный контроль чётности или поперечный) представляет собой наиболее простой метод и наименее мощный алгоритм контроля целостности данных, так как с его помощью можно обнаружить только одиночные ошибки в проверяемых данных [4].

Метод заключается в суммировании по модулю 2 всех битов контролируемой информации. Для информации, состоящей из нечётного числа единиц, контрольная сумма всегда равна 1, а при чётном числе единиц – 0. Например, для данных 11011100 результатом контрольного суммирования будет значение 1.

Результат суммирования представляет собой один дополнительный бит данных, который пересылается вместе с контролируемой информацией. При искажении в процессе пересылки любого бита исходных данных (или контрольного разряда) результат суммирования будет отличаться от принятого контрольного разряда, что говорит об ошибке. Однако двойная ошибка, например 110010001, будет неверно принята за корректные данные. Поэтому контроль по паритету применяется к небольшим порциям данных, как правило, к каждому байту, что даёт коэффициент избыточности для этого метода 1/8. Метод редко применяется в компьютерных сетях из-за значительной избыточности и невысоких диагностических способностей [4].

Пример расчёта контрольной суммы данным методом приведён в таблице 1.

Таблица 1 – Пример расчёта контрольной суммы методом «контроль по паритету»

Биты																																																**КС**
1	0	0	0	0	0	1	0	1	0	0	1	0	0	0	1	1	0	0	1	0	0	0	1	1	0	1	0	1	0	0	0	1	0	0	1	0	0	1	0	1	0	0	0	1	0	1	0	1

Вертикальный и горизонтальный контроль по паритету

Вертикальный и горизонтальный контроль чётности (поблочный контроль чётности или продольный) представляет собой модификацию метода контроля по паритету. Его отличие состоит в том, что исходные данные рассматриваются в виде матрицы, строки которой составляют байты данных. Контрольный разряд подсчитывается отдельно для каждой строки и для каждого столбца матрицы. Этот метод обнаруживает значительную часть двойных ошибок, однако обладает ещё большей избыточностью. Кроме того, он позволяет восстанавливать информацию при одиночных ошибках и в некоторых случаях при двойных (при условии появления ошибок в разных столбцах и строках). Метод сейчас почти не применяется при передаче информации по сети [4].

Для использования данного метода, источник и приёмник информации должны заранее «договориться», какое число передаваемых символов будет рассматриваться ими как единый блок данных. Биты чётности добавляются в виде обычного символа в конец блока.

Пример расчёта контрольной суммы данным методом приведён в таблице 2.

Таблица 2 – Пример расчёта контрольной суммы методом «вертикальный и горизонтальный контроль по паритету»

Байты	**Биты**								
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**КС**
1	1	0	0	0	0	0	1	0	0
2	1	0	0	1	0	0	0	1	1
3	1	0	0	1	0	0	0	1	1
4	1	0	1	0	1	0	0	0	1
5	1	0	0	1	0	0	1	0	1
6	1	0	0	0	1	0	1	0	1
КС	0	0	1	1	0	0	1	0	

Циклический избыточный контроль (Cyclic Redundancy Check — CRC)

В настоящее время CRC является наиболее популярным методом контроля целостности данных в вычислительных сетях (и не только в сетях; в частности, этот метод широко применяется при записи данных на гибкие и жёсткие диски). Метод основан на рассмотрении исходных данных в виде одного многоразрядного двоичного числа. Например, кадр стандарта Ethernet, состоящий из 1024 байт, будет рассматриваться как одно число, состоящее из 8192 бит. Контрольной информацией считается остаток от деления этого числа на известный делитель *R*. Обычно в качестве делителя выбирается семнадцати- или тридцатитрехразрядное число, чтобы остаток от деления имел длину 16 разрядов (2 байта) или 32 разряда (4 байта). При получении кадра данных снова вычисляется остаток от деления на тот же делитель *R*, но при этом к данным кадра добавляется и содержащаяся в нем контрольная сумма. Если остаток от деления на *R* равен нулю, то делается вывод об отсутствии ошибок в полученном кадре, в противном случае кадр считается искажённым [4].

Принцип вычисления контрольной суммы:

- Исходная информация (бинарный или текстовый файл, несколько символов и т. д. и т. п.) представляется единой последовательностью битов.
- Последовательность делится на некоторое фиксированное двоичное число *R* (англ. generator polinomial – генераторный полином, полином, CRC-полином).
- Контрольным значением будет являться остаток от деления.
- Получатель информации таким же образом выполняет деление и сравнивает полученный остаток с переданным отправителем.
- Если они равны, то считается, что исходное сообщение не повреждено.

Для вычисления контрольной суммы используют специальную полиномиальную арифметику: делитель, делимое (исходного сообщения), частное и остаток вместо положительных целых чисел, представляется в виде полиномов с двоичными

коэффициентами или в виде строки бит, каждый из которых является коэффициентом полинома. Исходное сообщение и делитель *R* могут быть представлены в виде полиномов, с которыми можно выполнять любые арифметические действия.

Например. Кадр из *k* бит рассматривается как список коэффициентов многочлена степени *k – 1*, состоящего из *k* членов от x^{k-1} до x^0. Старший (самый левый) бит кадра соответствует коэффициенту при x^{k-1}, следующий бит – коэффициенту при x^{k-2} и т. д. Например, число 110001 состоит из 6 бит и, следовательно, представляется в виде многочлена пятой степени (степенью полинома называют позицию самого старшего единичного бита) с коэффициентами 1, 1, 0, 0, 0 и 1: $x^5 + x^4 + x^0$ [1].

С данными многочленами осуществляются арифметические действия по модулю 2 в соответствии с алгебраической теорией поля. При этом перенос при сложении и заём при вычитании не производится. И сложение, и вычитание эквивалентны исключающему ИЛИ (XOR), что продемонстрировано на рисунке 2 [1].

$$\begin{array}{rrrr} 10011011 & 00110011 & 1110000 & 01010101 \\ \pm\ \underline{11001010} & +\ \underline{11001101} & -\ \underline{10100110} & -\ \underline{10101111} \\ 01010001 & 11111110 & 01010110 & 11111010 \end{array}$$

Рисунок 2 – Арифметические действия с многочленами по модулю 2

Деление чисел осуществляется в точности так же, как и деление обычных двоичных чисел, с той разницей, что вычитание производится снова по модулю 2 [1].

При использовании циклического кода отправитель и получатель должны сначала договориться насчёт образующего многочлена, *G(x)*. Старший и младший биты образующего многочлена должны быть равны 1. Для вычисления CRC для некоторого кадра из *m* бит, соответствующего полиному *M(x)*, необходимо, чтобы этот кадр был длиннее образующего многочлена. Идея состоит в добавлении CRC в конец кадра таким образом, чтобы получившийся многочлен делился на образующийся многочлен *G(x)* без остатка. Получатель, приняв кадр, содержащий контрольную сумму, пытается разделить его на *G(x)*. Ненулевой остаток от деления означает ошибку [1].

Алгоритм вычисления CRC при этом может быть следующим:

1 Пусть *r* – степень многочлена *G(x)*. Необходимо добавить *r* нулевых бит в конец кадра так, чтобы он содержал $m + r$ бит и соответствовал многочлену $x^rM(x)$.

2 Битовая строка, соответствующая многочлену $x^rM(x)$, делится по модулю 2 на битовую строку, соответствующую образующему многочлену *G(x)*.

3 Затем, вычитается по модулю 2 остаток от деления (он должен быть не более *r* бит) из битовой строки, соответствующей многочлену $x^rM(x)$. В результате будет получен многочлен *T(x)*, который и будет передаваемым кадром [1].

На рисунке 3 показаны вычисления для кадра 1101011111 и образующего многочлена $G(x) = x^4 + x + 1$ [1].

Многочлен из примера показанного на рисунке 3 *T(x)* делится (по модулю 2) на *G(x)* без остатка. Если уменьшить делимое на остаток, то результат должен делиться без остатка. Например, в десятичной системе счисления, если разделить 210.278 на 10.941, то остаток от деления будет равен 2.399. Если вычесть 2.399 из 210.278, то результат (207.879) будет делиться на 10.941 без остатка [1].

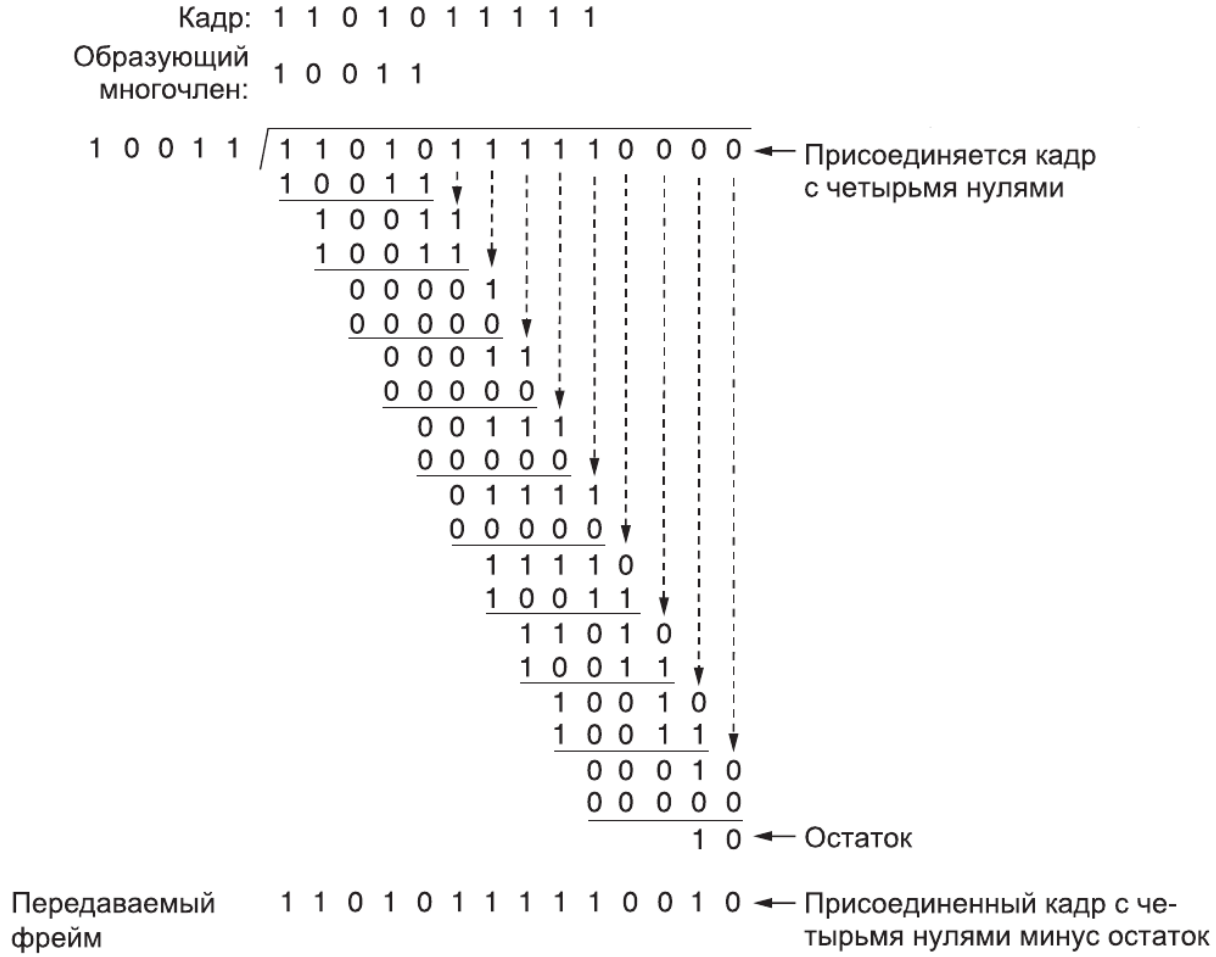

Рисунок 3 – Пример вычисления CRC

Некоторые образующие многочлены стали международными стандартами. Например, полином, использующийся в IEEE 802 (основан на многочлене, который первоначально предлагался для стандартов Ethernet) [1]:

$$x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x^1 + 1.$$

Хотя алгоритм вычисления CRC может показаться сложным, Питерсон (Peterson) и Браун в 1961 году показали, что может быть создана простая схема для аппаратной проверки и подсчёта CRC на основе сдвигового регистра. Эта схема до сих пор повсеместно применяется на практике. Десятки сетевых стандартов работают на основе кодов CRC, включая почти все локальные сети (такие как Ethernet, 802.11) [1].

Алгоритм вычисления контрольной суммы CRC с помощью сдвигового регистра:

1 Создаётся массив (регистр), заполненный нулями, равный по длине разрядности (степени *r*) полинома.

2 Исходное сообщение дополняется нулями в младших разрядах, в количестве, равном числу *r* разрядов полинома.

3 В младший разряд регистра заносится один старший бит сообщения, а из старшего разряда регистра выдвигается один бит.

4 Если выдвинутый бит равен «1», то производится операция исключающего ИЛИ (XOR) регистра и полинома.

5 Если в сообщении ещё есть биты – переход к шагу 3.

6 Когда все биты сообщения поступили в регистр и были обработаны этим алгоритмом, в регистре остаётся остаток от деления, который и является контрольной суммой.

Получатель может сделать одно из возможных действий:

1 Выделить текст исходного сообщения, вычислить для него контрольную сумму (не забыв при этом дополнить сообщение *r* битами), и сравнить её с переданной.

2 Вычислить контрольную сумму для всего переданного сообщения (без добавления нулей), и посмотреть, получится ли в результате нулевой остаток.

Для реализации второго подхода, получатель должен в пункте 2 описанного алгоритма вместо дополнения исходного сообщения нулями дополнить его битами рассчитанной контрольной суммы, а остальное оставить как есть.

Пример программной реализации алгоритма вычисления контрольной суммы CRC с помощью сдвигового регистра. Пусть имеется полином с *r*=4 и его двоичное представление имеет следующий вид: 10011. Тогда для выполнения деления потребуется 4-битный регистр, представленный на рисунке 4.

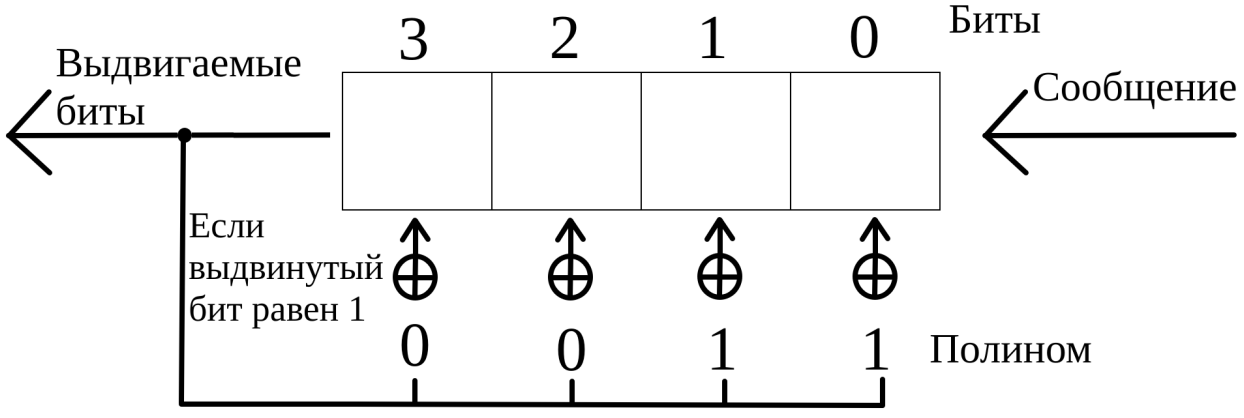

Рисунок 4 – Схематичное представление вычисления CRC с применением сдвигового регистра на примере деления на многочлен $x^4 + x + 1$

На псевдокоде работа функции вычисления контрольной суммы с применением сдвигового регистра будет иметь следующий вид:

```
Загрузить регистр нулевыми битами
Дополнить хвостовую часть сообщения r нулевыми битами
WHILE Есть необработанные биты
    Сдвинуть регистр на 1 бит влево
    Поместить в крайний правый бит регистра необработанный бит сообщения
    IF Из регистра был выдвинут бит со значением «1»
        Регистр = Регистр XOR Полином
    ENDIF
ENDWHILE
Контрольная сумма равна значению оставшемуся в регистре
```

Реализация CRC на основе сдвигового регистра имеет ряд недостатков – требует большого числа машинных операций и выполняется достаточно медленно, что существенно ограничивает её применение в современных системах связи. Для ускорения вычисления расчёта был предложен 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 двойные слова (32 бита) и т.д.

Для табличного алгоритма создаётся таблица (пример представлен в таблице 3) чисел размером $2^n xn$, где n – разрядность контрольной суммы.

Таблица 3 – Табличный метод вычисления контрольной суммы

Адрес в таблице	**Данные в таблице (числа)**
…	…
…	…
Адрес *m*	Остаток от деления числа 1 0000 0000 на полином
Адрес *m+1*	Остаток от деления числа 10 0000 0000 на полином
Адрес *m+2*	Остаток от деления числа 11 0000 0000 на полином
Адрес *m+3*	Остаток от деления числа 100 0000 0000 на полином
Адрес *m+4*	Остаток от деления числа 101 0000 0000 на полином
…	…
Адрес *n*	Остаток от деления числа 1111 1111 0000 0000 на полином

Числа являются остатком от деления по модулю 2 числа с n конечными нулями (например $n = 8$) и с n начальными разрядами, равными номеру числа (его адресу) в таблице. Выполняется деление на 9-разрядный образующий полином. Вычисление таблицы производится однократно, и она хранится в ОЗУ или ПЗУ.

Алгоритм вычисления контрольного кода при помощи данной таблицы следующий:

- выбирается первый байт массива, который рассматривается как адрес в таблице (номер числа);
- находится в таблице число с данным номером – получается остаток O_1;
- выбирается второй байт информационного массива и суммируется по модулю 2 с остатком O_1;
- находится в таблице число с данным номером – получается остаток O_2;
- выбирается третий байт массива, суммируется по модулю 2 с остатком O_2;
- находится в таблице число с данным номером – получается остаток O_3;
- …

Практическая работа

Цель: изучение алгоритмов обнаружения и исправления ошибок при передаче данных по каналам связи.

Задачи первой части работы:

- аналитический обзор алгоритмов обнаружения и исправления ошибок при передаче данных по каналам связи;
- практическое знакомство с алгоритмами «контроль по паритету», «вертикальный и горизонтальный контроль по паритету», «циклический избыточный контроль» путём выполнения практического задания без применения вычислительной техники;
- закрепление полученных в ходе выполнения первой части работы знаний, умений и навыков.

Задачи второй части работы:

- практическое знакомство с алгоритмами «контроль по паритету», «вертикальный и горизонтальный контроль по паритету», «циклический избыточный контроль» путём написания программных функций, которые реализуют изучаемые алгоритмы на выбранном языке программирования;
- реализация программного обеспечения, для подсчёта контрольных сумм файлов с применением ранее написанных функций; программное обеспечение должно обеспечивать подсчёт и проверку контрольных сумм;
- закрепление полученных в ходе выполнения второй части работы знаний, умений и навыков.

Варианты индивидуальных заданий

Практическая работа состоит из двух частей. По каждой из них необходимо составить отчёт (имеется возможность объединения материалов всей практической работы в один документ).

Для закрепления материала по теме практической работы и выполнения задания её первой части, каждому обучающемуся необходимо рассчитать вручную контрольные суммы алгоритмами «контроль по паритету» (каждая буква передаётся отдельно), «вертикальный и горизонтальный контроль по паритету» (все буквы передаются одним пакетом), «циклический избыточный контроль» (все буквы передаются одним пакетом, полином согласно варианту таблицы 5).

В качестве исходного сообщения, подлежащего передаче, используется фамилия обучающегося выполняющего задание. Для перевода фамилии в численное представление используется таблица ASCII (с учётом регистра, приложение А).

Получившееся для передачи сообщение необходимо представить в шестнадцатеричном и двоичном кодах (пример в таблице 4).

Таблица 4 – Пример создания исходного сообщения

Фамилия в виде текста	Иванов
Фамилия в виде шестнадцатеричного кода	88 A2 A0 AD AE A2
Фамилия в виде двоичного кода	10001000 10100010 10100000 10101101 10101110 10100010

Таблица 5 – Варианты заданий первой части работы

Номер варианта	**Название**	**Полином**	**Применение**
1	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)
2	CRC-8-Dallas/Maxim	$x^8+x^5+x^4+1$	1-Wire bus
3	CRC-5-USB	x^5+x^2+1	USB token packets
4	CRC-4-ITU	x^4+x+1	ITU G.704
5	CRC-8	$x^8+x^7+x^6+x^4+x^2+1$	ETSI EN 302307 3), 5.1.4
6	CRC-5-ITU	$x^5+x^4+x^2+1$	ITU G.704
7	CRC-5-EPC	x^5+x^3+1	Gen 2 RFID
8	CRC-6-ITU	x^6+x+1	ITU G.704
9	CRC-7	x^7+x^3+1	системы телекоммуникации, ITU-T G.707, ITU-T, G.832, MMC, SD
10	CRC-8-SAE J1850	$x^8+x^4+x^3+x^2+1$	AES3, OBD

Для выполнения второй части работы, каждому обучающемуся необходимо реализовать программное обеспечение, которое имеет возможность подсчёта и проверки контрольных сумм указанного пользователем файла. Значения контрольных сумм сохраняются в файле в директории исходного файла, с аналогичным именем, но с расширением «.ccs». Программное обеспечение должно включать в себя отдельные функции, реализующие алгоритмы расчёта контрольных сумм:

- «контроль по паритету» – контрольная сумма для каждого байта сообщения;
- «вертикальный и горизонтальный контроль по паритету» – пакет данных равен 8 байтам, последний пакет, при нехватке данных, дополнять нулями;
- «циклический избыточный контроль» – полином согласно варианту таблицы 6 [5-8]. Реализация либо табличным методом, либо при помощи сдвигового регистра (обязательно применение побитовых операций). Полином задаётся переменной типа «беззнаковое целое» в шестнадцатеричной системе счисления. Для проверки правильности подсчёта контрольной суммы использовать следующие два вектора (последовательности цифр в кодировке ASCII): «123456789» и «987654321».

Таблица 6 – Варианты заданий второй части работы

Номер варианта	**Название**	**Полином в HEX виде**	**Начальное состояние регистра**	**Отражение входных данных***	**Отражение выходных данных****	**Сложение результата по модулю 2*****	**Проверка 1**	**Проверка 2**	**Применение**
1	CRC-12 DECT	0x80F	0x000	Нет	Нет	0x000	0xF5B	0x9D6	системы телекоммуникации
2	CRC-15-CAN	0x4599	0x0000	Нет	Нет	0x0000	0x59E	0x6FB	сеть контроллеров
3	CRC-32Q	0x814141AB	0x00000000	Нет	Нет	0x00000000	0x3010BF7F	0x3172BE3A	aviation, AIXM
4	CRC-32 MPEG-2	0x04C11DB7	0xFFFFFFFF	Нет	Нет	0x00000000	0x0376E6E7	0x56FD8C5B	V.42, MPEG-2, PNG, POSIX cksum
5	CRC-16-DNP	0x3D65	0x0000	Да	Да	0xFFFF	0xEA82	0x51A6	DNP, IEC 870, M-Bus
6	CRC-64-ECMA-182	0x42F0E1EBA9EA3693	0x0000000000000000	Нет	Нет	0x0000000000000000	0x6C40DF5F0B497347	0xA3CA689185CAED8E	ECMA-182, XZ Utils
7	CRC-64-ISO	0x000000000000001B	0xFFFFFFFFFFFFFFFF	Да	Да	0xFFFFFFFFFFFFFFFF	0xB90956C775A41001	0x3CAFB00415A776CF	SO 3309 (HDLC), Swiss-Prot/TrEMBL; considered weak for hashing
8	CRC-11 FLEXRAY	0x385	0x01A	Нет	Нет	0x000	0x5A3	0x1FD	FlexRay
9	CRC-32C (Castagnoli)	0x1EDC6F41	0xFFFFFFFF	Да	Да	0xFFFFFFFF	0xE3069283	0xE46BD790	iSCSI, G.hn payload
10	CRC-30	0x2030B9C7	0x3FFFFFFF	Нет	Нет	0x3FFFFFFF	0x04C34ABF	0x39374ABA	CDMA
11	CRC-16-CCITT ZERO	0x1021	0x0000	Нет	Нет	0x0000	0x31C3	0x9CAD	X.25, HDLC, XMODEM, Bluetooth, SD идр.
12	CRC-24 LTE-A	0x864CFB	0x000000	Нет	Нет	0x000000	0xCDE703	0xDF1D1E	OpenPGP, RTCM104v3, LTE
13	CRC-13-BBC	0x1CF5	0x0000	Нет	Нет	0x0000	0x04FA	0xAB5	Time signal, Radio teleswitch
14	CRC-14-DARC	0x0805	0x0000	Да	Да	0x0000	0x082D	0x1F74	Data Radio Channel
15	CRC-17 CAN-FD	0x1685B	0x00000	Нет	Нет	0x00000	0x04F03	0x12619	CAN FD
16	CRC-40 GSM	0x0004820009	0x0000000000	Нет	Нет	0xFFFFFFFFFF	0xD4164FC646	0x88CFF7FF7D	GSM

Номер варианта	Название	Полином в HEX виде	Начальное состояние регистра	Отражение входных данных*	Отражение выходных данных**	Сложение результата по модулю 2***	Проверка 1	Проверка 2	Применение
17	CRC-16 KERMIT	0x1021	0x0000	Да	Да	0x0000	0x2189	0x349F	KL10 Error-Free Reciprocal Microprocessor Interchange over TTY lines
18	CRC-64 REDIS	0xAD93D23594C935A9	0x0000000000000000	Да	Да	0x0000000000000000	0xE9C6D914C4B8D9CA	0x2AB0F240E815BB43	Redis
19	CRC-14 GSM	0x202D	0x0000	Нет	Нет	0x3FFF	0x30AE	0x1486	GSM
20	CRC-15 MPT1327	0x6815	0x0000	Нет	Нет	0x0001	0x2566	0x0869	MPT1327

* Если «Да», то каждый входной байт отражается перед использованием в расчёте. Отражённый означает, что биты входного байта используются в обратном порядке. Пример с байтом 0x82 = b10000010, после отражения: b01000001 = 0x41.
** Если «Да», то перед возвратом отражается конечное значение КС. Отражение выполняется по всему значению КС.
*** Сложение по модулю 2 (XOR) значения из таблицы с результатом расчёта КС перед возвратом. Это делается после шага «Отражение выходных данных».

Состав отчёта по практической работе

В отчёт по работе должно обязательно входить следующее:

1. титульный лист (название министерства и ВУЗа, направление и направленности подготовки, номера курса, номер группы, название дисциплины, номер работы, ФИО обучающегося и преподавателя, город и год);
2. цели и задачи работы;
3. аналитический обзор;
4. ход выполнения работы;
5. выводы по работе (отразить достижение целей и задач работы, в выводах необходимо использовать ключевые слова: изучено, проанализировано, пришли к заключению и т. д.);
6. использованная литература;
7. приложение А «Листинг программы».

Использованная литература

1 Таненбаум, Э. Компьютерные сети / Э. Таненбаум, Д. Уэзеролл – 5-е изд. — Санкт-Петербург : Питер, 2012. – 960 с. – ISBN 978-5-459-00342-0.

2 Олифер, В. Компьютерные сети. Принципы, технологии, протоколы : Юбилейное издание / В. Олифер, Н. Олифер. – 6-е изд. – Санкт-Петербург : Питер, 2020. – 1008 с. – ISBN 978-5-4461-1426-9.

3 Алиев, Т.И. Компьютерные сети и телекоммуникации: задания и тесты : учебно-методическое пособие / Т. И. Алиев, В. В. Соснин, Д. Н. Шинкарук ; Минобрнауки России, федеральное государственное автономное образовательное учреждение высшего образования «Национальный исследовательский университет ИТМО» : Санкт-Петербург : Университет ИТМО, 2018. – 112 с.

4 Пролетарский А.В. Беспроводные сети Wi-Fi / А.В. Пролетарский, И.В. Баскаков, Р.А. Федотов, А.В. Бобков, Д.Н. Чирков, В.А. Платонов. – Москва : Национальный Открытый Университет ИНТУИТ, 2016. – 284 с. – ISBN 978-5-94774-737-9. – URL: https://intuit.ru/studies/courses/1004/202/info (дата обращения: 06.10.2020). – Текст: электронный.

5 Catalogue of parametrised CRC algorithms : электронно-справочная система : сайт. – 2022 – . – URL: https://reveng.sourceforge.io/crc-catalogue/legend.htm (дата обращения 10.10.2022).

6 CRC Calculator (Javascript) : электронно-справочная система : сайт. – 2022 – . – URL: http://www.sunshine2k.de/coding/javascript/crc/crc_js.html (дата обращения 10.10.2022).

7 Online CRC-8 CRC-16 CRC-32 Calculator : электронно-справочная система : сайт. – 2022 – . – URL: https://crccalc.com/ (дата обращения 10.10.2022).

8 CRC calculation : электронно-справочная система : сайт. – 2022 – . – URL: http://www.zorc.breitbandkatze.de/crc.html (дата обращения 10.10.2022).

ПРИЛОЖЕНИЕ А

(справочное)

Таблица ASCII

Таблица 7 – Символы ASCII с кодами от 00h до 7Fh

Hex	Char	Cmd	Hex	Char	Cmd	Hex	Char	Cmd	Hex	Char	Cmd
00		NUL	20		(sp)	40	@		60	`	
01	☺	SOH	21	!		41	A		61	a	
02	☻	STX	22	"		42	B		62	b	
03	♥	ETX	23	#		43	C		63	c	
04	♦	EOT	24	$		44	D		64	d	
05	♣	ENQ	25	%		45	E		65	e	
06	♠	ACK	26	&		46	F		66	f	
07	•	BEL	27	'		47	G		67	g	
08	◘	BS	28	(		48	H		68	h	
09	○	TAB	29	)		49	I		69	i	
0A	◙	LF	2A	*		4A	J		6A	j	
0B	♂	VT	2B	+		4B	K		6B	k	
0C	♀	FF	2C	,		4C	L		6C	l	
0D	♪	CR	2D	-		4D	M		6D	m	
0E	♫	SO	2E	.		4E	N		6E	n	
0F	☼	SI	2F	/		4F	O		6F	o	
10	►	DLE	30	0		50	P		70	p	
11	◄	DC1	31	1		51	Q		71	q	
12	↕	DC2	32	2		52	R		72	r	
13	‼	DC3	33	3		53	S		73	s	
14	¶	DC4	34	4		54	T		74	t	
15	§	NAK	35	5		55	U		75	u	
16	▬	SYN	36	6		56	V		76	v	
17	↨	ETB	37	7		57	W		77	w	
18	↑	CAN	38	8		58	X		78	x	
19	↓	EM	39	9		59	Y		79	y	
1A	→	SUB	3A	:		5A	Z		7A	z	
1B	←	ESC	3B	;		5B	[		7B	{	
1C	∟	FS	3C	<		5C	\		7C	|	
1D	↔	GS	3D	=		5D	]		7D	}	
1E	▲	RS	3E	>		5E	^		7E	~	
1F	▼	US	3F	?		5F	_		7F	⌂	DEL

Таблица 8 – Символы ASCII с кодами от 80h до FFh (кодировка IBM CP866)

Hex	Char	Hex	Char	Hex	Char	Hex	Char
80	А	A0	а	C0	└	E0	р
81	Б	A1	б	C1	┴	E1	с
82	В	A2	в	C2	┬	E2	т
83	Г	A3	г	C3	├	E3	у
84	Д	A4	д	C4	─	E4	ф
85	Е	A5	е	C5	┼	E5	х
86	Ж	A6	ж	C6	╞	E6	ц
87	З	A7	з	C7	╟	E7	ч
88	И	A8	и	C8	╚	E8	ш
89	Й	A9	й	C9	╔	E9	щ
8A	К	AA	к	CA	╩	EA	ъ
8B	Л	AB	л	CB	╦	EB	ы
8C	М	AC	м	CC	╠	EC	ь
8D	Н	AD	н	CD	═	ED	э
8E	О	AE	о	CE	╬	EE	ю
8F	П	AF	п	CF	╧	EF	я
90	Р	B0	░	D0	╨	F0	Ё
91	С	B1	▒	D1	╤	F1	ё
92	Т	B2	▓	D2	╥	F2	Є
93	У	B3	│	D3	╙	F3	є
94	Ф	B4	┤	D4	╘	F4	Ї
95	Х	B5	╡	D5	╒	F5	ї
96	Ц	B6	╢	D6	╓	F6	Ў
97	Ч	B7	╖	D7	╫	F7	ў
98	Ш	B8	╕	D8	╪	F8	°
99	Щ	B9	╣	D9	┘	F9	∙
9A	Ъ	BA	║	DA	┌	FA	·
9B	Ы	BB	╗	DB	█	FB	√
9C	Ь	BC	╝	DC	▄	FC	№
9D	Э	BD	╜	DD	▌	FD	¤
9E	Ю	BE	╛	DE	▐	FE	■
9F	Я	BF	┐	DF	▀	FF	