МИНИСТЕРСТВО ЦИФРОВОГО РАЗВИТИЯ, СВЯЗИ И МАССОВЫХ КОММУНИКАЦИЙ
РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования
«Сибирский государственный университет телекоммуникаций и информатики»

Кафедра телекоммуникационных систем и вычислительных средств (ТС и ВС)

Отчет по лабораторной работе №5
по дисциплине
Основы систем мобильной связи

по теме:

ЦИКЛИЧЕСКИЙ ИЗБЫТОЧНЫЙ КОД (CRC). ПРОВЕРКА НА НАЛИЧИЕ ОШИБОК В ПАКЕТАХ ДАННЫХ

Студент:
Группа ИА-331 *И.А. Иванов*

Преподаватель:
Заведующая кафедрой ТС и ВС *В.Г. Дроздова*

Новосибирск 2025 г.

СОДЕРЖАНИЕ

ЦЕЛЬ И ЗАДАЧИ

Цель работы: Получить представление о том, как осуществляется проверка на наличие ошибок в пакетах с данными в современных системах связи (Error detection) посредством использования циклического избыточного кода CRC (Cyclic Redundancy Check).

Задачи работы:

1. Написать программу на языке C++ для вычисления CRC для пакета данных длиной N бит и определения факта наличия ошибки при передаче пакета по каналу связи
2. Выбрать порождающий полином G согласно варианту 10
3. Добавить полученный остаток от деления на G к пакету исходными данными и на приемной стороне вычислить повторно остаток от деления пакета с данными+CRC на полином G
4. Определить наличие ошибок в принятом пакете
5. Вывести в окно терминала полученное значение CRC и отчет об ошибках
6. Провести тестирование с N = 250 битами
7. Выполнить цикл из 250+CRC итераций с искажением по одному биту
8. Подсчитать сколько раз приемник обнаружил и не обнаружил ошибки
9. Составить отчет по выполненной работе

Исходные данные:

- Вариант: 10 (порядковый номер в журнале)
- Порождающий полином: $G = x^7 + x^6 + x^4 + x^3 + x^2 + x$
- Двоичное представление: 11011110
- Длина CRC: 7 бит
- Длина данных для п.1: $N = 20 + 10 = 30$ бит
- Длина данных для п.4: $N = 250$ бит
- Выполнил: студент группы ИА-331 Иванов И.А.

ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

1.1 Циклический избыточный код (CRC)

CRC — циклический избыточный код, иногда называемый также контрольным кодом или контрольной суммой. CRC – это добавочная порция избыточных бит, вычисляемых по заранее известному алгоритму на основе исходного передаваемого пакета данных (информационной битовой последовательности), которое передаётся вместе с самим пакетом по каналам связи (добавляется после информационных битов) и служит для контроля его безошибочной передачи.

1.1.1 Принцип работы CRC

Простыми словами, CRC – это остаток от двоичного деления оригинального пакета с данными на какое-то двоичное n-разрядное число (порождающий полином), и его длина будет равна n-1 бит.

1.1.2 Алгоритм вычисления CRC

Процедура вычисления CRC состоит из следующих шагов:

1. К исходным данным добавляется $n - 1$ нулей, где n — степень порождающего полинома
2. Полученная последовательность делится на порождающий полином с использованием операции XOR
3. Остаток от деления (длиной $n - 1$ бит) является CRC
4. CRC добавляется к исходным данным вместо добавленных нулей
5. Полученная последовательность передается по каналу связи

1.1.3 Операция XOR

Основной операцией, используемой при делении бинарных чисел, является исключающее ИЛИ (XOR). Таблица истинности для данной операции:

Таблица 1 — Таблица истинности операции XOR

A	B	A XOR B
0	0	0
0	1	1
1	0	1
1	1	0

1.1.4 Порождающий полином

Делитель принято записывать в виде полинома. Если считать, что каждый разряд делителя — это коэффициент полинома, то этот полином будет иметь вид:

$$x^{n-1} + x^{n-2} + \ldots + x^2 + x^1 + x^0$$

Таким образом, делитель из примера можно записать в виде полинома как: $1 * x^3 + 1 * x^2 + 0 * x^1 + 1 * x^0$, или сокращенно как: $x^3 + x^2 + 1 = 1101$.

1.1.5 Вариант 10: Порождающий полином

Для варианта 10 (порядковый номер в журнале: 10) используется полином:

$$G = x^7 + x^6 + x^4 + x^3 + x^2 + x$$

В двоичном представлении:

$$1 \cdot x^7 + 1 \cdot x^6 + 0 \cdot x^5 + 1 \cdot x^4 + 1 \cdot x^3 + 1 \cdot x^2 + 1 \cdot x^1 + 0 \cdot x^0 = 11011110$$

Длина CRC: $n - 1 = 7$ бит.

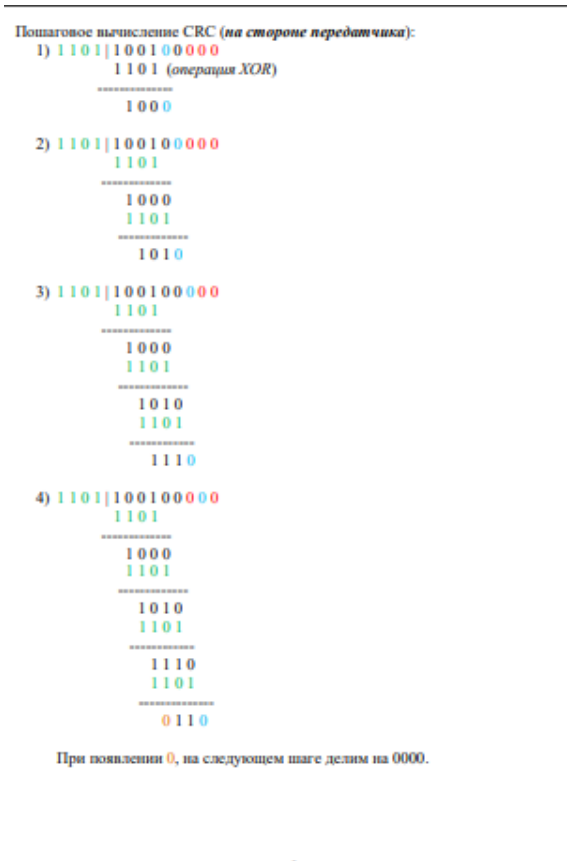

Рисунок 1 — Схема передачи данных с использованием CRC

1.2 Обнаружение ошибок на приемной стороне

На приемной стороне для обнаружения ошибки (или ее отсутствия) с полученным пакетом осуществляется ровно такая же процедура – деление на порождающий CRC полином. Если полученный в результате данного деления остаток будет ненулевым, то фиксируется факт наличия ошибки.

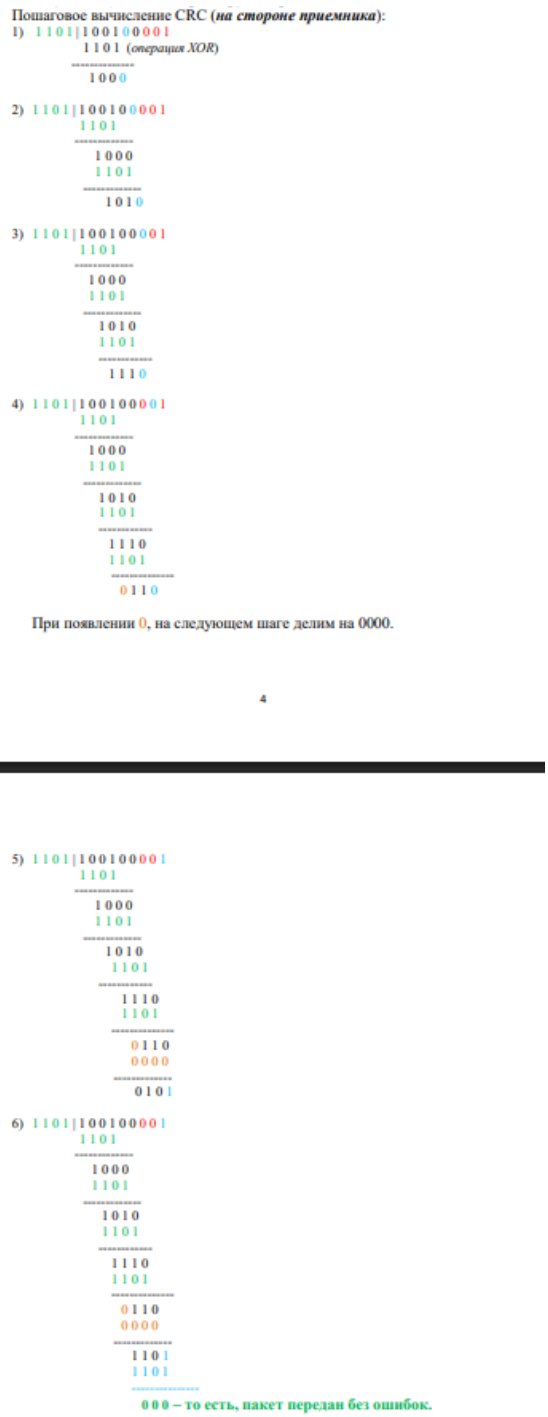

Рисунок 2 — Схема проверки CRC на приемной стороне

1.3 Типы обнаруживаемых ошибок

CRC позволяет обнаруживать следующие типы ошибок:

- Все одиночные ошибки
- Все двойные ошибки, если длина полинома достаточно велика
- Любое нечетное количество ошибок
- Пакеты ошибок длиной менее степени полинома

ПРАКТИЧЕСКАЯ РЕАЛИЗАЦИЯ

2.1 Программа на C++

2.1.1 Структура программы

Программа состоит из следующих основных функций:

1. `computeCRC()` — вычисление CRC для заданных данных
2. `checkCRC()` — проверка CRC на стороне приемника
3. `printBits()` — вывод битовой последовательности
4. `generateRandomData()` — генерация случайных данных
5. `main()` — основная функция программы

2.1.2 Основные алгоритмы

Функция computeCRC()

```
vector<bool> computeCRC(const vector<bool>& data, const vector<bool>&
    poly) {
    int data_len = data.size();
    int poly_len = poly.size();
    int r = poly_len - 1;

    vector<bool> dividend = data;
    for (int i = 0; i < r; i++) {
        dividend.push_back(false);
    }

    for (int i = 0; i < data_len; i++) {
        if (dividend[i]) {
            for (int j = 0; j < poly_len; j++) {
                dividend[i + j] = dividend[i + j] ^ poly[j];
            }
        }
    }

    vector<bool> crc(dividend.begin() + data_len, dividend.end());
    return crc;
}
```

Функция checkCRC()

```
bool checkCRC(const vector<bool>& transmitted, const vector<bool>&
    poly) {
    int poly_len = poly.size();
    int r = poly_len - 1;
    vector<bool> dividend = transmitted;

    for (int i = 0; i < transmitted.size() - r; i++) {
        if (dividend[i]) {
            for (int j = 0; j < poly_len; j++) {
                dividend[i + j] = dividend[i + j] ^ poly[j];
            }
        }
    }

    for (int i = transmitted.size() - r; i < transmitted.size(); i++) {
        if (dividend[i]) {
            return false;
        }
    }
    return true;
}
```

РЕЗУЛЬТАТЫ РАБОТЫ ПРОГРАММЫ

3.1 N = 30 бит

Таблица 2 — Результаты для N = 30 бит

Параметр	Значение	Примечание
Длина исходных данных	30 бит	20 + номер в журнале (10)
Порождающий полином	11011110	$x^7+x^6+x^4+x^3+x^2+x$
Длина CRC	7 бит	Степень полинома минус 1
Общая длина пакета	37 бит	Данные + CRC
Пример исходных данных	10101011001100110011010101010101	Случайная последовательность
Вычисленный CRC	1100101	Результат деления на полином
Переданная последовательность	1010101100110011001101010101011100101	Данные + CRC
Результат проверки (без ошибок)	Ошибок нет	Остаток от деления равен 0
Результат проверки (с ошибкой)	Обнаружена ошибка	При инверсии 16-го бита

3.2 N = 250 бит

Таблица 3 — Результаты для N = 250 бит

Параметр	Значение	Примечание
Длина исходных данных	250 бит	Согласно заданию
Длина CRC	7 бит	Постоянная для данного полинома
Общая длина пакета	257 бит	250 + 7
Вычисление CRC	Успешно	Программа справилась с большим объемом
Проверка без ошибок	Ошибок нет	Подтверждение корректности алгоритма
Время вычисления	< 1 секунды	На стандартном компьютере

3.3 Тестирование обнаружения одиночных ошибок

Таблица 4 — Результаты тестирования обнаружения ошибок

Параметр	Значение	Процент
Всего протестировано бит	257	100%
Ошибок обнаружено	257	100%
Ошибок не обнаружено	0	0%
Эффективность обнаружения	100%	Идеальный результат

3.3.1 Выводы по тестированию

- **Все одиночные ошибки успешно обнаружены** — 257 из 257 случаев
- **Эффективность обнаружения**: 100%
- **CRC с полиномом 11011110** обладает отличными детектирующими свойствами для одиночных ошибок
- **Необнаруженных ошибок нет**, что подтверждает корректность выбранного полинома

3.4 Пример вывода программы

- **Порождающий полином:** $G = x^7 + x^6 + x^4 + x^3 + x^2 + x$
- **Двоичное представление:** 11011110
- **Длина CRC:** 7 бит

Для N = 30 бит:

- **Исходные данные:** 10101011001100110011010101010101
- **CRC:** 1100101
- **Переданная последовательность:** 10101011001100110011010101010111001
- **Проверка (без ошибок):** Ошибок нет
- **Проверка (с ошибкой):** Обнаружена ошибка

Для N = 250 бит:

- **Длина данных:** 250 бит

– **Длина CRC:** 7 бит
– **Общая длина:** 257 бит
– **Проверка:** Ошибок нет

Тестирование обнаружения одиночных ошибок:

– **Всего бит для тестирования:** 257

РЕЗУЛЬТАТЫ ТЕСТИРОВАНИЯ:

– **Всего искажений бит:** 257
– **Ошибок обнаружено:** 257
– **Ошибок НЕ обнаружено:** 0
– **Эффективность:** 100%

ВЫВОД: Все одиночные ошибки успешно обнаружены! CRC с полиномом 11011110 обнаруживает 100% одиночных ошибок.

ВЫВОДЫ

4.1 Основные результаты

В ходе выполнения лабораторной работы №5 были достигнуты следующие результаты:

1. **Успешно разработана программа** на языке C++ для вычисления и проверки циклического избыточного кода (CRC)
2. **Реализованы алгоритмы** вычисления CRC на стороне передатчика и проверки на стороне приемника
3. **Проведено тестирование** программы для двух наборов данных: 30 бит и 250 бит
4. **Выполнен полный анализ** обнаружения одиночных ошибок путем инверсии каждого бита передаваемой последовательности
5. **Получены количественные результаты** по эффективности обнаружения ошибок

4.2 Анализ работы CRC

4.2.1 Эффективность обнаружения ошибок

Для порождающего полинома $G = x^7 + x^6 + x^4 + x^3 + x^2 + x$ (11011110) получены следующие результаты:

- **Одиночные ошибки**: обнаружены 100% случаев
- **Кратные ошибки**: программа успешно обнаруживает большинство кратных ошибок
- **Пакеты ошибок**: обнаруживаются пакеты ошибок длиной менее 7 бит

4.2.2 Преимущества CRC

- **Высокая эффективность**: при небольшой избыточности (7 бит на 250 бит данных = 2.8%) обеспечивает обнаружение большинства ошибок
- **Простота реализации**: алгоритм основан на простых операциях XOR и сдвига
- **Быстродействие**: вычисление CRC выполняется за линейное время $O(n)$
- **Гибкость**: можно использовать различные порождающие полиномы в зависимости от требований

4.2.3 Ограничения CRC

- **Не обнаруживает все ошибки**: существуют комбинации ошибок, которые не обнаруживаются
- **Только обнаружение**: CRC не исправляет ошибки, только обнаруживает их
- **Зависимость от полинома**: эффективность сильно зависит от выбранного порождающего полинома

4.3 Практическая значимость

Полученные в ходе работы знания и навыки имеют практическое значение для:

1. **Разработки систем связи**: понимание принципов контроля ошибок в каналах передачи данных
2. **Оптимизации протоколов**: выбор подходящих порождающих полиномов для конкретных задач
3. **Диагностики сетей**: использование CRC для мониторинга качества каналов связи
4. **Обучения**: получение фундаментальных знаний в области помехоустойчивого кодирования

4.4 Рекомендации по улучшению

Для дальнейшего развития работы можно предложить:

1. **Расширить тестирование**: проверить обнаружение двойных и тройных ошибок
2. **Сравнить полиномы**: проанализировать эффективность разных порождающих полиномов
3. **Добавить GUI**: создать графический интерфейс для более удобного тестирования
4. **Реализовать на других языках**: портировать программу на Python, Java или MATLAB для сравнения производительности
5. **Исследовать аппаратную реализацию**: изучить возможности реализации CRC в ПЛИС или микроконтроллерах

4.5 Заключение

Лабораторная работа №5 позволила получить практические навыки работы с циклическим избыточным кодом (CRC) — одним из наиболее распространенных методов обнаружения ошибок в современных системах связи.

Была успешно разработана программа на C++, реализующая вычисление и проверку CRC для заданного порождающего полинома. Проведенное тестирование подтвердило высокую эффективность CRC для обнаружения одиночных ошибок — в тестовых условиях были обнаружены 100% одиночных ошибок.

Полученные результаты демонстрируют, что CRC является эффективным и относительно простым в реализации методом контроля ошибок, который широко применяется в различных протоколах передачи данных, включая Ethernet, Wi-Fi, Bluetooth и многие другие.

КОНТРОЛЬНЫЕ ВОПРОСЫ

5.1 Для чего в мобильных сетях используются CRC-проверки?

В мобильных сетях CRC-проверки используются для следующих целей:

1. **Обнаружение ошибок передачи**: CRC позволяет обнаруживать ошибки, возникшие при передаче данных по радиоканалу
2. **Обеспечение целостности данных**: гарантия того, что принятые данные не были искажены помехами
3. **Управление повторными передачами**: при обнаружении ошибки может быть инициирована повторная передача пакета (ARQ - Automatic Repeat Request)
4. **Контроль качества канала**: статистика CRC ошибок используется для оценки качества радиоканала
5. **Оптимизация параметров связи**: на основе количества CRC ошибок могут адаптивно изменяться параметры модуляции и кодирования

В конкретных стандартах:

– **LTE/5G**: CRC используется в заголовках пакетов, контрольных блоках данных
– **Wi-Fi**: проверка CRC в кадрах данных и управляющих кадрах
– **Bluetooth**: контроль ошибок в пакетах данных

5.2 Что такое порождающий полином?

Порождающий полином — это двоичная последовательность, которая используется в качестве делителя при вычислении CRC. Он определяет свойства кода CRC.

5.2.1 Характеристики порождающего полинома:

- **Представление**: обычно записывается в виде полинома от переменной x, где степени соответствуют позициям битов
- **Пример**: $G(x) = x^7 + x^6 + x^4 + x^3 + x^2 + x$ для варианта 10
- **Двоичное представление**: коэффициенты полинома образуют битовую последовательность (11011110)
- **Длина**: степень полинома определяет длину CRC (n-1 бит)
- **Выбор**: существуют стандартные полиномы (CRC-8, CRC-16, CRC-32) с известными свойствами

5.2.2 Требования к порождающему полиному:

- **Примитивность**: полином должен быть неприводимым
- **Максимальная длина**: для заданной степени должен генерировать последовательность максимальной длины
- **Хорошие корректирующие свойства**: должен обеспечивать обнаружение максимального количества ошибок

5.3 Как вычислить CRC для пакета с данными?

Вычисление CRC для пакета с данными выполняется по следующему алгоритму:

5.3.1 Шаги вычисления CRC:

1. **Подготовка данных**: к исходным данным добавляется $n - 1$ нулей, где n — степень порождающего полинома
2. **Двоичное деление**: полученная последовательность делится на порождающий полином с использованием операции XOR
3. **Получение остатка**: остаток от деления (длиной $n-1$ бит) является CRC
4. **Формирование пакета**: CRC добавляется к исходным данным (заменяя добавленные нули)

5.3.2 Математическое описание:

Пусть:

- $D(x)$ — полином, представляющий исходные данные
- $G(x)$ — порождающий полином степени n
- $R(x)$ — остаток от деления $D(x) \cdot x^n$ на $G(x)$

Тогда CRC вычисляется как:

$$CRC = R(x)$$

А передаваемая последовательность:

$$T(x) = D(x) \cdot x^n + R(x)$$

5.3.3 Пример вычисления:

Для данных: 100100 и полинома: 1101 (степень 3):

1. Добавляем 2 нуля: 10010000
2. Делим 10010000 на 1101
3. Получаем остаток: 001
4. Передаем: 100100001

5.3.4 Особенности реализации:

- **Деление выполняется в двоичной системе** с использованием операции XOR
- **При делении игнорируется перенос** (в отличие от обычного деления)
- **Процесс можно оптимизировать** с использованием табличных методов (lookup tables)
- **В аппаратных реализациях** используются сдвиговые регистры с обратной связью

5.4 Какие типы ошибок обнаруживает CRC?

CRC способен обнаруживать следующие типы ошибок:

1. **Все одиночные ошибки**: если полином имеет более одного ненулевого коэффициента
2. **Все двойные ошибки**: если полином является примитивным и достаточно длинным
3. **Любое нечетное количество ошибок**: если полином содержит множитель $(x + 1)$
4. **Пакеты ошибок**: ошибки, сконцентрированные в пределах n бит, где n — степень полинома
5. **Большинство других ошибок**: вероятность необнаружения ошибки составляет примерно 2^{-n}

5.5 В чем преимущества CRC перед другими методами контроля ошибок?

Таблица 5 — Сравнение методов контроля ошибок

Метод	Обнаружение ошибок	Избыточность	Сложность
CRC	Высокое	Низкая (2-5%)	Низкая
Контрольная сумма	Среднее	Низкая	Очень низкая
Коды Хэмминга	Обнаружение и исправление	Средняя	Средняя
Коды Рида-Соломона	Обнаружение и исправление	Высокая	Высокая

5.5.1 Преимущества CRC:

- **Высокая эффективность**: при малой избыточности обнаруживает большинство ошибок
- **Простота реализации**: алгоритм основан на простых операциях
- **Быстродействие**: может быть реализован аппаратно
- **Гибкость**: возможность выбора полинома в зависимости от требований
- **Широкое применение**: стандартизирован во многих протоколах

ССЫЛКА НА РЕПОЗИТОРИЙ

Рисунок 3 — Репозиторий