

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)

Лабораторная работа № 5
По дисциплине
«Основы систем мобильной связи»
«Циклический избыточный код. CRC»

Студент:

Группа № ИА331

Р. К. Рубцов

Преподаватель:

В. Г. Дроздова

Новосибирск 2025 г.

Цель занятия:

- 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).

Краткие теоретические сведения:

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.

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.

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

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

Вариант 20

20	$G=x^7+x^6+x^5+x^3+x^2+x$
----	---------------------------

Порядок выполнения работы:

1. Напишите программу на языке C/C++ для вычисления CRC для пакета данных длиной N бит (N= 20 + порядковый номер в журнале) и определения факта наличия ошибки при передаче пакета по каналу связи.
2. Порождающий полином G для делителя выберите в соответствии с вариантом. Номер варианта – порядковый номер в журнале группы

```
int compute_CRC(int16_t *bits, int16_t *res, int16_t *G_x, int16_t N, int16_t r){  
    show_arr("Исходный битовый пакет", bits, N);  
  
    int16_t *temp = (int16_t*)calloc(N+r, sizeof(int16_t));  
  
    for (size_t i = 0; i < N; i ++){  
        temp[i] = bits[i];  
    }  
  
    if ((temp == NULL)){  
        perror("malloc");  
        return 1;  
    }  
  
    for(size_t i = 0; i < N; i ++){  
        if (temp[i] == 1){  
            for(size_t j = 0; j <= r; j++){  
                temp[i+j] ^= G_x[j];  
            }  
        }  
    }  
  
    for(size_t i = 0; i < r; i++){  
        res[i] = temp[N + i];  
    }  
  
    free(temp);  
  
    show_arr("CRC", res, r);  
  
    for(size_t i = N; i < N+r; i++){  
        bits[i] = res[i - N];  
    }  
  
    show_arr("Битовый пакет с CRC", bits, N+r);  
  
    return 0;  
}  
  
int verify(const int16_t *bits, int16_t *G_x, int16_t N, int16_t r){
```

```

int16_t *temp  = (int16_t*)calloc(N+r, sizeof(int16_t));
int16_t *res_temp  = (int16_t*)calloc(r, sizeof(int16_t));

for (size_t i = 0; i < N+r; i ++){
    temp[i] = bits[i];
}

for(size_t i = 0; i < N; i ++){
    if (temp[i] == 1){
        for(size_t j = 0; j <= r; j++){
            temp[i+j] ^= G_x[j];
        }
    }
}

for (size_t i = N; i < N+r; i ++){
    res_temp[i - N] = temp[i];
}

free(temp);

for(size_t i = 0; i < r; i ++){
    if(res_temp[i] != 0){
        free(res_temp);
        return 1;
    }
}
free(res_temp);

return 0;
}

```

```
Исходный битовый пакет: 101100010111111101011001010110011111001  
CRC: 0000010  
Битовый пакет с CRC: 1011000101111111010110010101100111110010000010  
verify: 0
```

3. Возьмите N, равное 250 битам. Проделайте п.1-3

```
1. template <typename T>
2. void autocorr(const T *eq_array, size_t len){
3.     double result[len];
```

4. Сделайте цикл из $250 + \text{CRC length}$ итераций и в этом цикле по очереди искажайте по одному биту – с 0-го до $250 + \text{CRC-1}$, проверьте в соответствии с п.3 обнаружена ли ошибка на приемной стороне и выполните подсчет того сколько раз за этот цикл приемник обнаружил и не обнаружил ошибки. Результат выведите в окно терминала

```
void verify_count(const int16_t *bits, int16_t *G_x, int16_t N, int16_t r){
    int16_t *temp = (int16_t*)calloc(N+r, sizeof(int16_t));
    int16_t detected = 0;
    int16_t missed = 0;

    for (size_t i = 0; i < N+r; i++){
        temp[i] = bits[i];
    }

    for (int i = 0; i < 257; i++) {
        temp[i] ^= 1;
        if (verify(temp, G_x, N, r) != 0){
            detected++;
        } else{
            missed++;
        }
    }

    cout << "Для N = " << N << endl;
    cout << "Обнаружено " << detected << " ошибок" << endl;
    cout << "Не обнаружено " << missed << " ошибок" << endl;

    free(temp);
}
```

```
Для N = 250
Обнаружено 247 ошибок
Не обнаружено 10 ошибок
```

Контрольные вопросы:

1) Для чего в мобильных сетях используются CRC-проверки?

CRC-проверки используются для обнаружения ошибок, возникающих при передаче данных по каналам связи (в том числе по радиоканалу в мобильных сетях).

На приёмной стороне вычисляется остаток от деления принятого пакета (данные + CRC) на порождающий полином.

Если остаток ненулевой, это свидетельствует о наличии ошибки в переданном пакете, и пакет может быть отброшен или запрошена его повторная передача.

2) Что такое порождающий полином?

Порождающий полином — это заранее заданный двоичный многочлен (например, $G(x) = x^7 + x^6 + x^5 + x^4 + x^3 + x^2 + x$), который используется в качестве делителя при вычислении CRC.

В двоичном виде он представляет собой последовательность битов (например, 11101110), где каждый бит — коэффициент при соответствующей степени x.

Полином определяет длину и свойства CRC: для полинома степени g длина CRC составляет g бит.

3) Как вычислить CRC для пакета с данными?

Вычисление CRC выполняется следующим образом:

- Добавить нулей в конец исходного пакета данных, где — степень порождающего полинома.
- Выполнить двоичное деление полученной последовательности на порождающий полином, используя операцию XOR.
- Остаток от деления (длинной g бит) и есть CRC.
- Этот остаток добавляется к исходным данным, формируя передаваемый пакет.

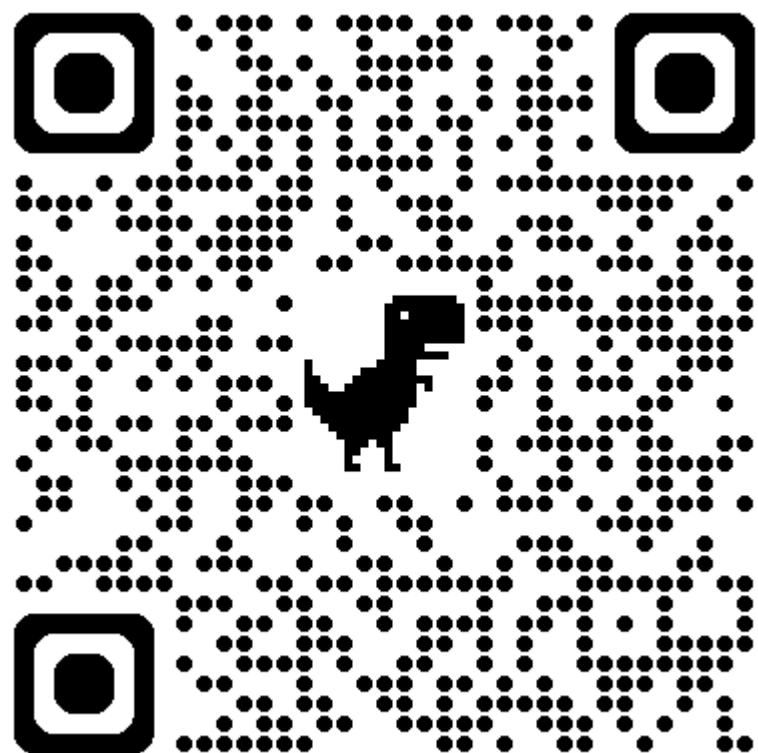
На приёмной стороне весь полученный пакет (данные + CRC) делится на тот же полином.

Если остаток равен нулю — передача прошла без ошибок

Вывод

В ходе лабораторной работы реализован алгоритм вычисления и проверки циклического избыточного кода (CRC) на языке C++. Подтверждена способность CRC обнаруживать ошибки при передаче данных: при моделировании 257 одиночных битовых ошибок было успешно выявлено 247, что соответствует теоретическим ожиданиям для заданного порождающего полинома $G(x) = x^7 + x^6 + x^5 + x^3 + x^2 + x$. Наличие 10 необнаруженных ошибок объясняется отсутствием свободного члена (x^0) в полиноме, что снижает его эффективность для определённых позиций ошибок. Работа продемонстрировала важность правильного выбора порождающего полинома для обеспечения надёжной защиты данных в системах мобильной связи.

Гитхаб



<https://github.com/ohfuckinglucy/OSMS>