

Задача 1. HDLC-фреймер.

Для передачи данных по синхронным каналам связи часто используется HDLC-подобная структура пересылаемых пакетов, называемых кадрами. Поток кадров является битовым потоком, поскольку передаваемые данные интерпретируются на уровне отдельных битов.

Условимся о следующих терминах: *данные* – это информация, предназначенная для передачи между двумя точками и разделённая на логические блоки, размер которых кратен восьми битам; *кадр* – совокупность блока данных и его контрольной суммы; *пакет* – кадр, над которым произведена операция стаффинга.

HDLC-поток устроен следующим образом:

- Данные передаются блоками, кратными восьми битам, т.е. размер в байтах блока данных является целым числом.
- Младший (самый правый в двоичной записи) бит отдельного байта идёт в потоке первым, старший (самый левый) – последним.
- Пакеты отделяются друг от друга восьмибитовой последовательностью 0111 1110 (0x7E). Эта же последовательность передаётся в случае отсутствия полезных данных для передачи.
- Поскольку в самом потоке данных может встретиться последовательность 0111 1110, то в процессе формирования пакета производится операция, называемая стаффингом (stuffing): после передачи пяти единичных бит подряд в поток добавляется нулевой бит. Таким образом, из-за процедуры стаффинга размер передаваемого пакета может становиться некратным восьми битам.
- На приёмнике производится обратная операция – дестаффинг (destuffing): нулевой бит, идущий после пяти единичных, игнорируется. Благодаря этим двум операциям внутри передаваемого пакета не может встретиться байт 0x7E.
- Для контроля целостности данных, то есть обнаружения неправильно переданных пакетов, к данным добавляется контрольная сумма CRC16-СCITT, порождаяемая полиномом $x^{16} + x^{12} + x^5 + 1$, правда, с некоторой модификацией, которую мы рассмотрим ниже.

Контрольная сумма передаётся после данных пакета, а её вычисление происходит до операции стаффинга. Ещё раз хотелось бы обратить ваше внимание на то, что передаваемые данные рассматриваются как битовый поток, группировки в байты не происходит и размер передаваемых пакетов не кратен байту.

На передатчике сначала производится расчёт контрольной суммы потока данных, затем она дописывается к потоку данных. В результате этого из блока данных образуется кадр. Над кадром происходит операция стаффинга: кадр превращается в пакет. В линии пакеты отделяются друг от друга одним или более байтами 0x7E.

Приёмник делит битовый поток из линии на пакеты, производит операцию дестаффинга, получая кадры. Затем разделяет кадр на блок данных и переданную контрольную сумму, рассчитывает контрольную сумму блока данных и осуществляет сравнение двух контрольных сумм.

Подсчёт контрольной суммы

Поток данных рассматривается как полином: если какой-то бит равен нулю, то коэффициент перед соответствующей степенью в полиноме – 0, если бит равен единице, то коэффициент – единица. Например, поток из 17 битов 10001000000100001 эквивалентен порождающему полиному $x^{16} + x^{12} + x^5 + 1$. Подсчёт контрольной суммы (CRC, Cyclical Redundancy Code) основан на нахождении остатка от деления полинома данных на порождающий полином. Остаток – это полином, который, будучи рассмотрен как битовая последовательность, и является контрольной суммой. Деление полиномов происходит по модулю два, то есть обычная операция вычитания заменяется двоичным вычитанием (или сложением – это всё равно) без переноса или заёма.

Контрольная сумма данных получается так:

- Передаваемые данные записываются по порядку их передачи в линию: самый первый бит слева, самый последний – справа. К этой последовательности мы добавляем справа 16 нулей, что эквивалентно умножению соответствующего полинома на x^{16} , и делим получившийся полином на порождающий. Получаем первый остаток от деления.
- Записываем 16 единиц и дописываем справа столько нулей, сколько бит занимают передаваемые данные. Делим получившееся на порождающий полином, получая второй остаток.
- Складываем по модулю 2 оба остатка и число 0xFFFF. Результат и является контрольной суммой пакета, он занимает 16 бит и дописывается в конец передаваемого пакета.

Пример

Пусть мы хотим передать блок данных, состоящий из байта 0x30 (это ASCII-код символа 0). В битовом виде пакет выглядит как 0000 1100. Посчитаем контрольную сумму: допишем 16 нулей – 0000 1100 0000 0000 0000 0000 и поделим на порождающий полином:

```
0000 1100 0000 0000 0000 0000
1000 1000 0001 0000 1
 100 0100 0000 1000 01
   10 0010 0000 0100 001
    1 0001 0000 0010 0001
     1000 1000 0001 0000 1
0000 0100 1000 0001 0000 1000
     100 0100 0000 1000 01
0000 0000 1100 0001 1000 1100
     10 0010 0000 0100 001
      1 0001 0000 0010 0001
-----
0000 0000 1100 0001 1000 1100
```

Остаток от деления равен 1100 0001 1000 1100. Теперь возьмём 16 единиц и 8 нулей (длина данных) – 1111 1111 1111 1111 0000 0000. Второй остаток равен 1110 0001 1111 0000. Результат сложения по модулю два остатков и числа 0xFFFF равен 1101 1111 1000 0011. Это и есть контрольная сумма, которая равна 0xC1FB, если вернуться к привычному порядку битов.

С учётом контрольной суммы кадр получается следующим: 0x30, 0xFB, 0xC1. Или в двоичном виде (самый первый передаваемый бит слева): 0000 1100 1101 **1111** 1000 0011. Жирным выделена последовательность из пяти единичных битов, к которой после проведения стаффинга добавится нулевой бит. Следовательно, пакет выглядит так: 0000 1100 1101 1111 0100 0001 1. Полностью оформленный битовый поток будет иметь следующий вид: 0111 1110 0000 1100 1101 1111 0100 0001 1011 1111 0. Это пример ещё раз демонстрирует, что размер передаваемых данных не кратен одному байту.

Задание

Требуется написать три программы: фреймер, дефреймер и эмулятор физической линии. Фреймер преобразует поток данных, подаваемый ему из некоторого файла, в HDLC-поток, сохраняемый в другом файле. Длина блоков данных, на которые делится исходный поток, задаётся из командной строки двумя параметрами: максимальным и минимальным размером блока. При этом размер конкретного блока должен являться случайной величиной, находящейся в заданном диапазоне. Пакеты должны разделяться произвольным количеством байтов-разделителей.

Дефреймер принимает битовый поток, состоящий из HDLC-пакетов, и превращает его в пакеты данных. Битовый поток читается из одного файла; данные записываются в другой файл. Битовый поток не обязательно начинается с последовательности 0111 1110 – её ещё необходимо найти (синхронизоваться с потоком); в потоке может находиться произвольное количество пакетов; никаких предположений о длине каждого пакета не делается. Необходимо выделить из потока каждый пакет, подсчитать его контрольную сумму, сравнить её с передаваемой вместе с самим пакетом. Если сумма правильная и размер пакета кратен одному байту, то содержимое пакета добавляется к выходному файлу. Если же сумма неправильная или размер пакета не кратен байту, то печатаем сообщение об ошибке и содержимое неправильного пакета. Затем переходим к обработке следующего пакета.

Эмулятор физической линии читает из одного файла HDLC-поток, с некоторой вероятностью инвертирует произвольные биты потока (симулирует помехи в линиях передачи), и записывает получившийся поток в выходной файл. Эмулятор портит биты, выбранные случайным образом; процент инвертированных битов является вещественным числом, находящимся на отрезке $[0, 100]$, и задаётся параметром командной строки эмулятора.

Замечание: *в реальных линиях процент инвертированных битов, при котором ещё возможна синхронизация, – это величина порядка одного процента. Поэтому не следует ожидать, что при более высоком проценте ошибок в линии вы сможете синхронизоваться с потоком и восстановить данные. Однако, это не означает что эмулятору всегда будут передаваться числа меньше единицы.*

Задача 2. Генератор японских кроссвордов.

Существует множество различных типов кроссвордов: обыкновенные, финские, японские и т.д. В отличие от обычных кроссвордов, где требуется отгадывать слова, в японском кроссворде зашифровано изображение. Вам предлагается реализовать программу, строящую по известному изображению японский кроссворд.

Сам кроссворд представляет из себя прямоугольную область, состоящую из квадратов (клеточек), которые могут быть закрашены различными цветами. Сверху каждого столбца и слева от каждой строки расположены по порядку несколько чисел. Возможно, что эти числа выкрашены в разные цвета, тогда кроссворд называется цветным. Если цвет чисел один, то кроссворд чёрно-белый. Сами числа являются длинами горизонтальных (для чисел слева от строки) и вертикальных (для чисел сверху от столбца) блоков из квадратов, закрашенных в соответствующий цвет. Среди всех цветов, присутствующих в кроссворде, один называется цветом фона; вертикальные или горизонтальные блоки данного цвета не учитываются при составлении кроссворда. Суть кроссворда состоит в том, чтобы угадать где находятся блоки цвета фона и какой у них размер.

Для чёрно-белого кроссворда между блоков чёрного цвета всегда присутствует блок цвета фона. Для цветных кроссвордов такого ограничения нет. Тем не менее, между блоками одного цвета всегда должен находиться блок другого цвета (быть может цвета фона). Блоки разного цвета могут примыкать друг к другу вплотную.

Числа, находящиеся в строках и столбцах, не должны противоречить друг другу, то есть картинка должна восстанавливаться из этих чисел некоторым образом.

Задание

Нужно написать программу, которой на вход подаётся картинка в формате BMP. Картинка может быть чёрно-белой или 16-цветной. Ваша программа должна строить по данной картинке **решаемый** японский кроссворд, если его возможно построить непротиворечивым образом. Если картинка двухцветная, то цвет фона белый, а числа красятся в чёрный цвет. Если же картинка имеет 16 цветов, то в программе должна быть предусмотрена возможность выбора цвета фона.

В случае невозможности построения непротиворечивого решаемого кроссворда программа должна информировать об этом пользователя. Если кроссворд строится, то он должен быть визуализирован.

Ещё раз обращаем ваше внимание: кроссворд должен решаться.