МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждение образования «БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ

ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет информационных технологий

Кафедра информационных систем и технологий

Специальность Информационные системы и технологии

**ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №5 НА ТЕМУ:**

**Избыточное кодирование данных в информационных системах. Итеративные коды**

Выполнил студент 3 курса 1 группы

Кашперко Василиса Сергеевна

Минск 2022

**Цель:** приобретение практических навыков кодирования / декодирования двоичных данных при использовании итеративных кодов.

Задачи:

1. Закрепить теоретические знания по использованию итеративных кодов для повышения надежности передачи и хранения в памяти компьютера двоичных данных.

2. Разработать приложение для кодирования/декодирования двоичной информации итеративным кодом с различной относительной избыточностью кодовых слов.

3. Результаты выполнения лабораторной работы оформить в виде описания разработанного приложения, методики выполнения экспериментов с использованием приложения и результатов эксперимента.

**Теоретические сведения**

**Кодом произведения** двух исходных (базовых) помехоустойчивых кодов называется такой *многомерный помехоустойчивый код*, кодовыми последовательностями которого являются все двумерные таблицы со строками кода (*k1*) и столбцами кода (*k2*).

Итеративные коды могут строиться на основе использования дву-, трёхмерных матриц (таблиц) и более высоких размерностей. Каждая из отдельных последовательностей информационных символов кодируется определенным линейным кодом (групповым или циклическим). Полученный таким образом результат является *линейным*.

Простейшим из итеративных кодов является *двумерный* код с проверкой на четность по строкам и столбцам. Итеративные коды, иногда называемые *прямоугольными* кодами, либо *композиционными*, являются одними из самых простых (с точки зрения аппаратной реализации).

Достоинства:

Простота аппаратной и программной реализации

Недостаток:

Высокая избыточность

Кодовые слова записываются в матрице в виде таблицы.

Проверочные символы вычисляются из того, что строки и столбцы должны содержать четное (нечётное) число единиц.

Пример:

При кодировании информационного слова *Xk* = 011101111 с помощью таблицы с чётностью по строкам и столбцам получим избыточные символы *Хr = Xh, Xv, Xhv* = 0010011

**Избыточные символы** называют так же **паритеты**.

Паритеты записываются сверху вниз, справа налево.

Возможен и обратный или иной порядок, НО при декодировании сообщения использовать нужно тогда аналогичный порядок следования паритетов.

**Паритет паритетов** (символ ***Xhv***) = сумме по модулю 2 символов информационного слова, а также проверочных символов *Xv* и *Xh*.

Декодирование начинают сразу, не ожидая поступления всего блока информации. Проверка соответствия избыточных символов полученного слова (*Yr = Yh, Yv, Yhv* либо *Yr = Yh, Yv*) при декодировании позволяет обнаружить любое нечетное число искаженных символов, расположенных в одной строке или в одном столбце. Формально такое декодирование осуществляется сравнением принятых (*Yh, Yv, Yhv*) и вновь вычисленных (*Yh’, Yv’, Yhv*’) для полученного слова паритетов.

Определение местоположения одиночной ошибки по строке указывает на наличие ошибки в этой строке матрицы, а проверка по столбцу – конкретный символ.

Однако этим кодом не могут быть установлены местоположения многократных ошибок, имеющих четное число искаженных символов как по строкам, так и по столбцам. Простейшая необнаруживаемая ошибка содержит четыре искаженных символа, расположенных в вершинах прямоугольника или квадрата. Это происходит из-за того, что четность (паритет) по строкам и по столбцам матрицы не нарушается

*Многомерные линейные итеративные коды*

Принято считать рассматриваемый код **многомерным**, если количество измерений, по которым вычисляются и анализируются паритеты, не менее 3. Таким образом, простейшим многомерным линейным итеративным кодом является код трехмерный.

**Ход выполнения практического задания**

Первоначально формируем произвольное двоичное представление информационного слова *Хk* (рис. 1) длиной *k* битов (9 бит) в двумерную матрицу размерностью 3x3 (рис. 2)

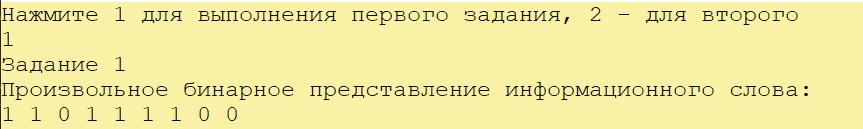


Рисунок 1 – Сгенерированное двоичное представление

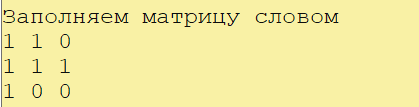


Рисунок 2 – Заполненная двумерная матрица

Далее вычисляем проверочные биты (биты паритетов) по двум, трём и четырем направлениям (рис. 3) и после этого формируем кодовое слово *Xn* присоединением избыточных символов к информационному слову (рис. 4).

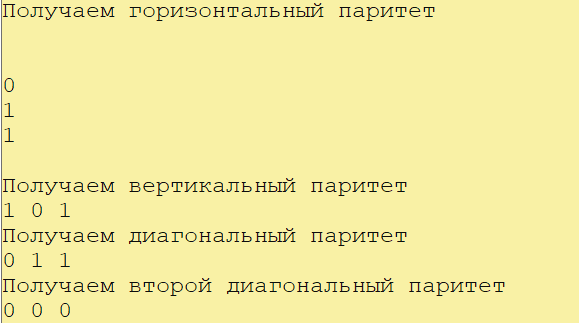


Рисунок 3 – Вычисление и вывод паритетов

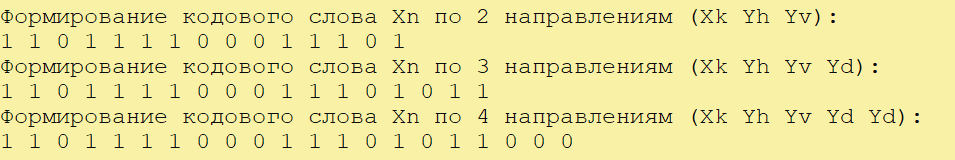


Рисунок 4 – Формирование кодового слова

Генерируем ошибку произвольной кратности, распределенную случайным образом среди символов слова *Xn* (Листинг 1), в результате чего формируется кодовое слово *Yn*.

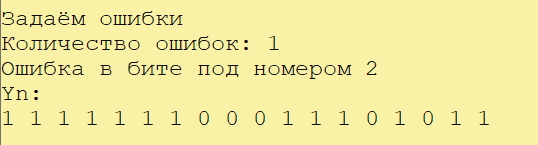
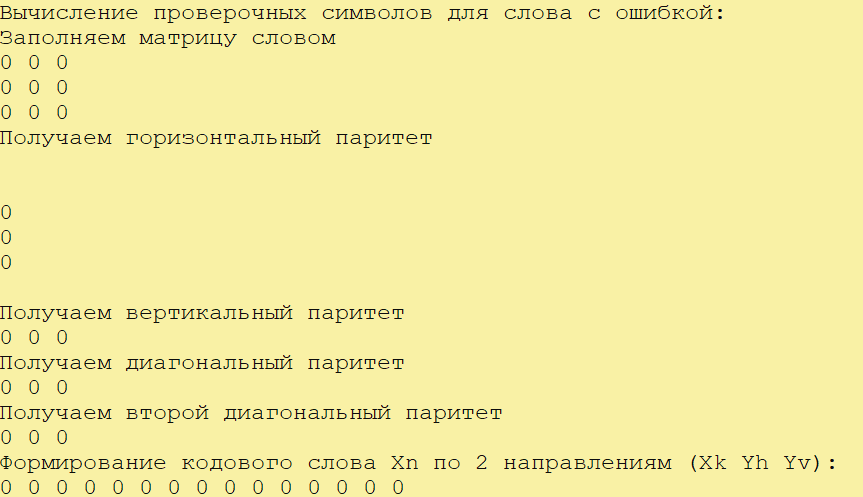


Рисунок 5 – Полученное кодовое слово с ошибкой

Далее определяем местоположение ошибочных символов итеративным кодом в слове *Yn* в соответствии с используемыми группами паритетов (рис. 6).



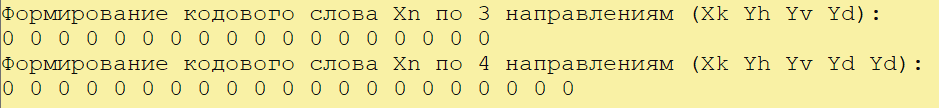


Рисунок 6 – Вычисление паритетов для сообщения с ошибкой

Выполняем анализ корректирующей способности используемого кода путем сравнения соответствующих слов *Xn* и *Yn’* (рис. 7).

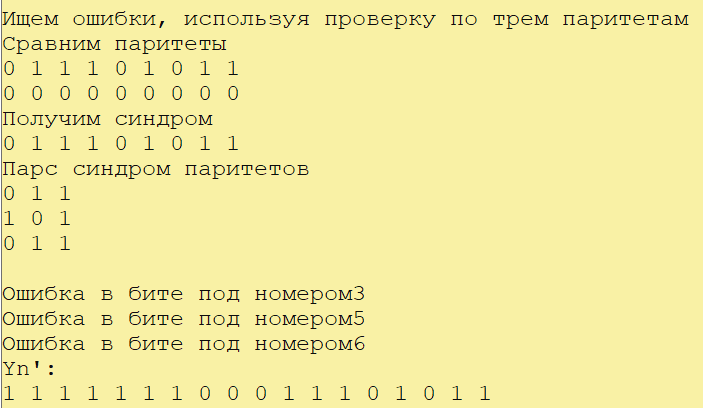


Рисунок 7 – Сравнение паритетов и определение ошибок

Итеративный код позволяет обнаружить нечетное число ошибок, однако гарантировано может исправить только одиночную ошибку.

Также ниже приведены результаты работы программы для трехмерной матрицы на рисунках 8, 9, 10, 11 и 12.

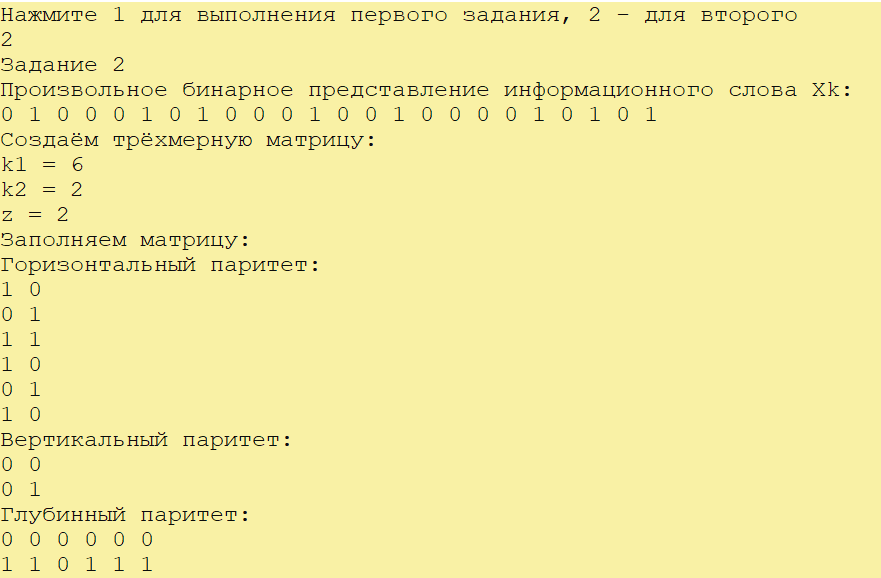


Рисунок 8 – Пример работы программы для трехмерной матрицы

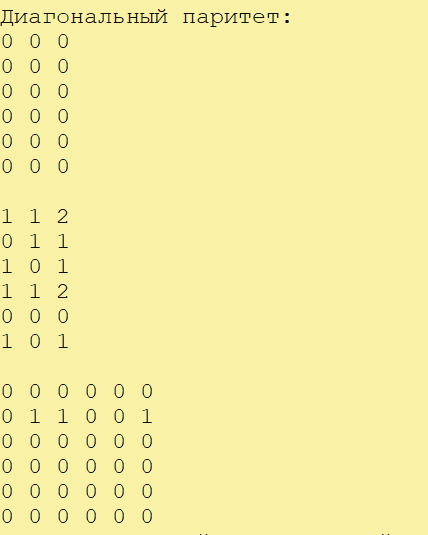


Рисунок 9 – Диагональный паритет трехмерной матрицы

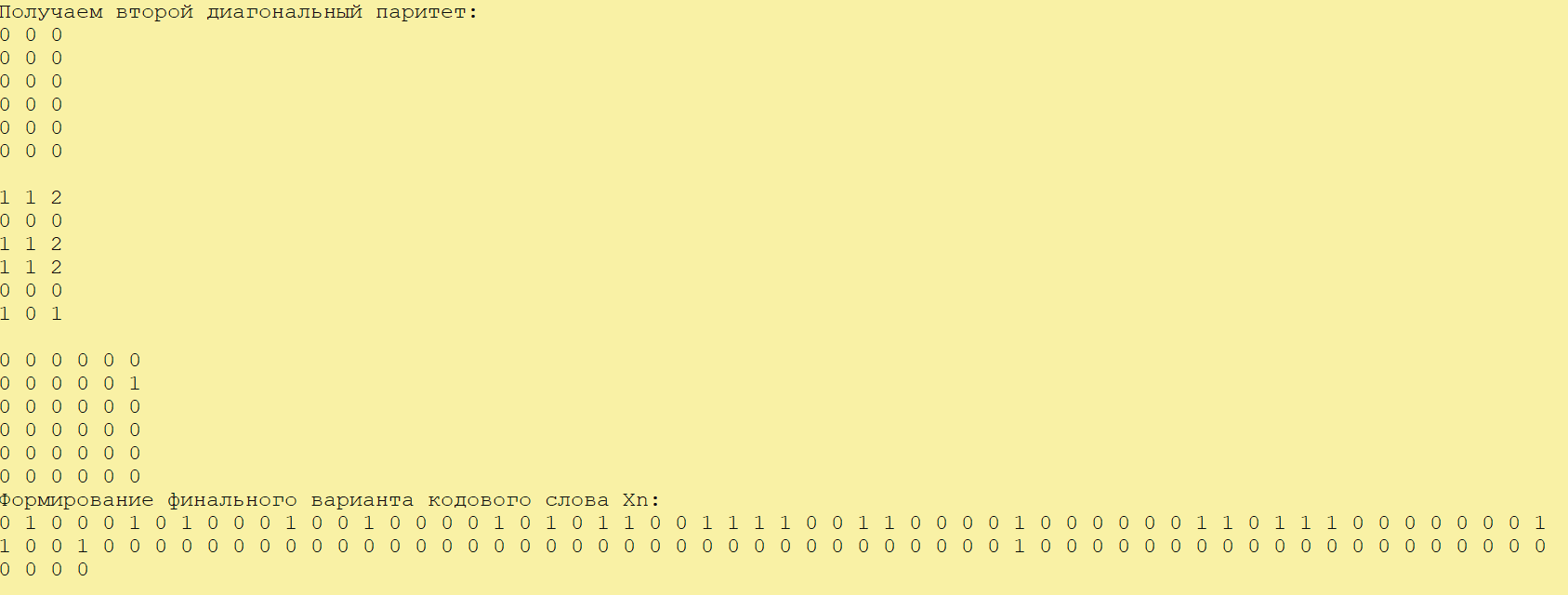


Рисунок 10 – Формирование окончательного варианта кодового слова для трехмерной матрицы

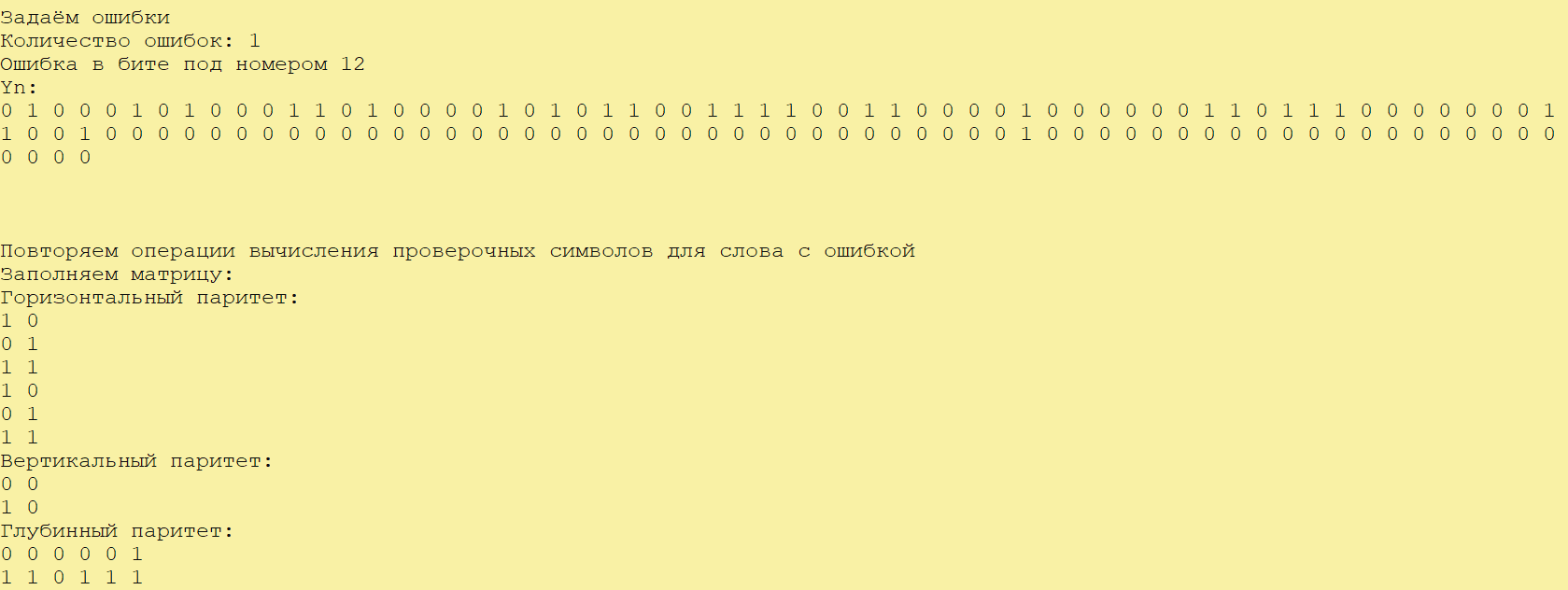


Рисунок 11 – Задаем ошибки в битах кодового слова

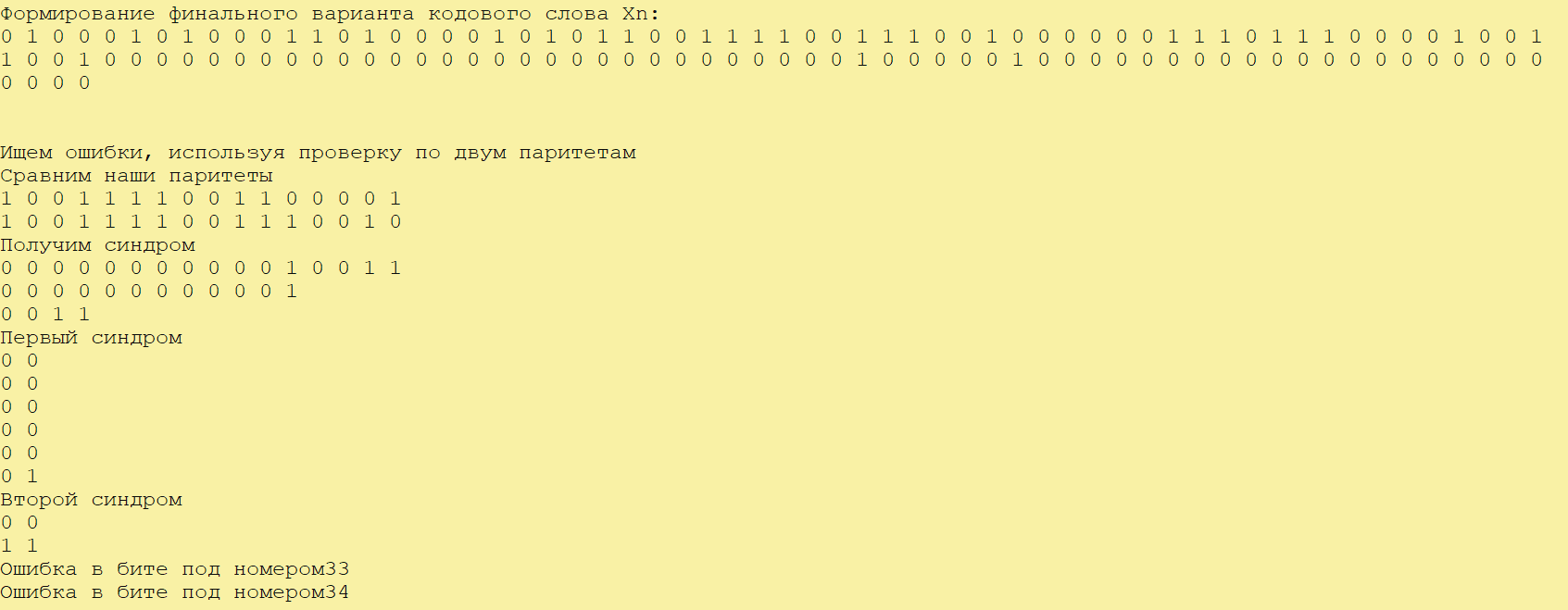


Рисунок 12 – Поиск ошибок в кодовом слове трехмерной матрицы

**Вывод:** результатом лабораторной работы является приложение, позволяющее осуществлять кодирование двоичных данных при использовании итеративных кодов, а также определять и исправлять сгенерированные ошибки.