Учреждение образования

«Белорусский государственный технологический университет»

**Защита информации и надежность информационных систем**

**Лабораторная работа №1**

Разработка и внедрение политики безопасности организации или учреждения

Тип организации юридическая компания

Выполнил:

Студент 3 курса 2 группы ФИТ

Максимова Вера Владимировна

Проверил:

Ржеутская Надежда Викентьевна

**2022 г.**

1. **Обоснование актуальности, цели и задачи разработки ПИБ в организации (учреждении).**

*Политика информационной безопасности* (ПИБ) организации или учреждения – совокупность правил, процедур, практических методов, руководящих принципов, документированных управленческих решений, направленных на защиту информации и связанных с ней ресурсов и используемых всеми сотрудниками организации или учреждения в своей деятельности.

Актуальность ПИБ состоит в том, чтобы не допустить законодательно или на основе правил утечки конфиденциальной информации в свободный доступ. Для каждой организации или учреждения необходимо иметь собственный набор правил, составленных на основе законодательства РБ чтобы уменьшить риски и наложить ответственность в ответ на противоправные действия со стороны сотрудников или сторонних субъектов.

**Цели:**

* Разработка и внедрение политики безопасности для Юридической компании

**Задачи:**

* Установка объектов защиты
* Установка основных угроз и их источников
* Оценка угроз, рисков и уязвимостей
* Установка мер, методов и средств обеспечения требуемого уровня защищенности информационных ресурсов

1. **Объекты защиты.**

Юридическая компания.



Структура юридической компании:

* Директор
* Директор кадрового отдела
* Работник кадрового отдела
* Директор бухгалтерии
* Бухгалтер
* Юрист
* Специалист по работе с клиентами

1. **Основные угрозы и их источники.**

Анализ потенциальных угроз: естественных и искусственных, а также преднамеренных и непреднамеренных, внешних и внутренних.

Внешние угрозы, влияющие:

* Атаки со стороны информационной безопасности
* Атаки со стороны конкурентов

Внутренние угрозы:

* Атаки со стороны сотрудников
* Разглашение информации о клиентах компетентными людьми

Непреднамеренные угрозы:

* Разговоры на тему работы
* Утеря носителей информации

Преднамеренные угрозы:

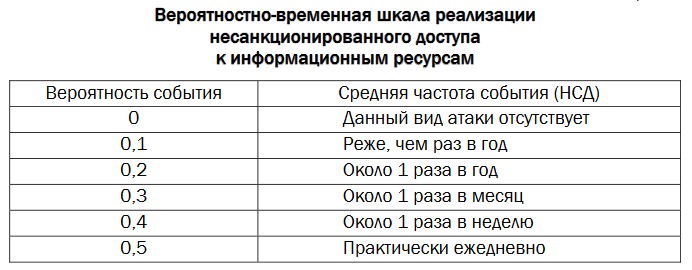
* Дезинформация
* Подделка документов
* Кража
* Угрозы сотрудникам, начальству, уборщикам помещений, арендодателям, заказчикам
* Восстановление удаленной информации с жестких дисков
* Несанкционированное распространение информации через поля и электрические сигналы

Естественные угрозы:

* Отсутствуют

1. **Основные угрозы и их источники.**

|  |  |
| --- | --- |
| Величина ущерба | Описание |
| 0 | Раскрытие информации, не являющейся конфиденциальной и не раскрывающая данных о заказчиках |
| 1 | Ущерб атаки есть, но он незначителен и не связан с положением компании на рынке |
| 2 | Существует вероятность потери некоторого числа клиентов, компания теряет часть прибыли |
| 3 | Потеря весомой доли клиентов и потеря доли рынка, компания несет убытки |
| 4 | Полная потеря рынка и конкурентоспособности, долги, а так же полная потеря клиентов, банкротство |
| 5 | Полное банкротство компании и прекращение существования |



Условная численная шкала для оценки ущерба 0-5.

Вероятностно-временная шкала реализации несанкционированного доступа к информационным ресурсам 0-0,5.

|  |  |  |  |
| --- | --- | --- | --- |
| Описание атаки | Ущерб | Вероятность | Риск |
| Подделка документов | 4 | 0.1 | 0.4 |
| Разговоры на тему работы | 2 | 0.3 | 0.6 |
| Атаки со стороны конкурентов | 3 | 0.1 | 0.3 |
| Атаки со стороны информационной безопасности | 3 | 0.2 | 0.6 |
| Разглашение информации о клиентах компетентными людьми | 3 | 0.1 | 0.3 |
| Атаки со стороны сотрудников | 3 | 0.1 | 0.3 |
| Итого | 6 | 0,15 | Сумма 2,5 |

Оптимальное значение коэффициента риска -- 0,3.

Критические значением считается 0,7, в случае его превышения финансовое состояние предприятия близко к банкротству.

Среднее значение риска вышло 0.42 – оптимальное.

1. **Меры, методы и средства обеспечения требуемого уровня защищенности информационных ресурсов.**

* Установление норм и правил в договорах сотрудников о коммерческой тайне.
* Использование шифрования и защищенных сетей
* Использование специализированных ПО по утилизации цифровой информации или физическая утилизация жестких дисков при их замене в качестве издержек
* Исключение дезинформации и ее получение у соответствующих лиц
* Найм компетентных сотрудников

**Вывод:** юридическая компания “Бизнес-инфо” обладает относительно низким риском потери информации в любом из его проявлений, однако политика безопасности представленная выше при соблюдении оной всеми сотрудниками компании поможет снизить риск потери информации, либо же денежных средств до минимума.

**Ответы на контрольные вопросы:**

1. Охарактеризовать актуальность и основные причины проблемы информационной безопасности организации, страны.

Политика необходима для того, чтобы донести до работников компании цели и задачи информационной безопасности. Что в конечном счёте уменьшает расходы компании

1. Сформулировать цели и задачи политики информационной безопасности.

Основные цели политики информационной безопасности:

* Основными целями ПИБ являются:
* выявление рисков, угрожающих всему периметру безопасности;
* внедрение средств защиты;
* расследование инцидентов информационной безопасности, зафиксированных случаев утечек;
* выявление виновников и привлечение их к ответственности с целью сохранения финансовых средств предприятия.
* Для создания эффективной системы информационной безопасности организации или учреждения целесообразно разработать:
* концепцию информационной безопасности, которая определяет в целом цели политики и основные ее принципы в увязке со статусом, целями и задачами организации или учреждения;
* стандарты (менеджмента качества) – правила и принципы защиты информации по каждому конкретному направлению деятельности;
* процедуры – описание конкретных действий по защите информации при работе с ней: персональных данных, порядка доступа к информационным носителям, системам и ресурсам;
* инструкции, содержащие подробное описание (алгоритмы) действий по организации информационной защиты и обеспечения разработанных стандартов и процедур;
* план мероприятий по обучению персонала и тестированию знаний сотрудников, имеющих доступ к информационным ресурсам.

Учреждение образования

«Белорусский государственный технологический университет»

**Защита информации и надежность информационных систем**

**Лабораторная работа №2**

ЭЛЕМЕНТЫ ТЕОРИИ ИНФОРМАЦИИ. ПАРАМЕТРЫ И ХАРАКТЕРИСТИКИ ДИСКРЕТНЫХ ИНФОРМАЦИОННЫХ СИСТЕМ

Выполнил:

Студент 3 курса 2 группы ФИТ

Максимова Вера Владимировна

Проверил:

Ржеутская Надежда Викентьевна

**2022 г.**

Цель: приобретение практических навыков расчета и анализа параметров и информативных характеристик дискретных ИС.

Задачи:

1. Закрепить теоретические знания по основам теории информации.

2. Разработать приложение для расчета и анализа параметров и информативных характеристик дискретных ИС.

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

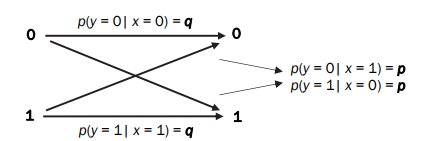
**Теоретические сведения**

Передача информации (данных) осуществляется между двумя абонентами, называемыми источником сообщения (ИcС) и получателем сообщения (ПС). Третьим элементом информационной системы является канал (среда) передачи, связывающий ИсС и ПС. Отметим также, что и в системах с хранением информации всегда можно выделить ИcС и ПС. В данном случае каналом передачи здесь выступает устройство хранения информации (память). Например, при записи данных в ОЗУ (оперативное запоминающее устройство) компьютера в качестве ИcС и ПС может выступать процессор (соответственно при записи и чтении данных). Таким образом, простейшая информационная система состоит из трех элементов: источника сообщения, канала передачи сообщения и получателя сообщения. Отображение сообщения обеспечивается изменением какойлибо физической величины, характеризующей процесс (например, амплитуда, частота, фаза). Эта величина является информационным параметром сигнала (в общем случае – информационной системы). 22 Лабораторная работа № 2 Сигналы, как и сообщения, могут быть непрерывными и дискретными. Информационный параметр непрерывного сигнала с течением времени может принимать любые мгновенные значения в определенных пределах. Непрерывный сигнал часто называют аналоговым, а каналы и устройства, функционирующие на основе такого типа сигналов, – аналоговыми. Дискретный сигнал (устройство или канал передачи) характеризуется конечным числом значений информационного параметра. Дискретные сообщения состоят из последовательности дискретных знаков. Часто этот параметр принимает всего два значения (0 или 1). Сообщение или канал его передачи на основе этих двух значений сигнала называют двоичным или бинарным. Построение сигнала по определенным правилам, обеспечивающим соответствие между сообщением и сигналом, называют кодированием. Кодирование в широком смысле – преобразование сообщения в сигнал. Кодирование в узком смысле – представление исходных знаков, называемых символами, в другом алфавите с меньшим числом знаков. Оно осуществляется с целью повышения надежности и преобразования сигналов к виду, удобному для передачи по каналам связи. Последний тип кодирования относится к так называемой прикладной теории кодирования информации, занимающейся поиском и реализацией методов и средств обнаружения несоответствий (ошибок) между переданным Xk и принятым Yk сообщениями. Рассмотрим основные характеристики и параметры двоичных систем. Важнейшая характеристика источника, получателя или канала – алфавит. Алфавит, А – это общее число знаков или символов (N), используемых для генерации или передачи сообщений. Символы алфавита будем обозначать через {аi}, где 1 ≤ i ≤ N; N – мощность алфавита. Минимальное число элементов алфавита Nmin = 2, А = {0, 1} – двоичный код. Один дискретный знак представляет собой элементарное сообщение, последовательность знаков – сообщение. Набор элементов алфавита, создаваемых дискретным источником сообщений, заранее, априори (до опыта) известен получателю. ИсС в каждый дискретный момент времени выдает один элемент алфавита. Этот элемент сообщения является одним из символов алфавита. Понятно, что ПС заранее не известно, какой это элемент. Если обозначить вероятность выбора каждого элемента алфавита p(аi), то



Вероятности p(аi) могут быть получены в результате анализа частотных свойств символов алфавита, если на входе такого анализатора принять документ на основе соответствующего алфавита. Причем объем документа должен быть таким, чтобы от частости (частоты) появления каждого символа в анализируемом документе можно было перейти к вероятности соответствующего события. Можно предположить, что указанному требованию будет соответствовать объем электронного документа не менее нескольких десятков килобайт.

Двоичный канал передачи информации строится на основе двоичного алфавита: А = {0, 1}. При этом канал, в котором вероятности искажения переданного 0 (принята соответственно 1; этому событию соответствует условная вероятность р(1|0)) и переданной 1 (принят соответственно 0; этому событию соответствует условная вероятность р(0|1)) равны, как и равны вероятности передачи 0 (р(0)) и 1 (р(1)), называют двоичным симметричным каналом (ДСК). В общем случае, если передается сообщение Хk = х1, х2, …, хk, а принимается сообщение Yk = y1, y2, …, yk, то данные условные вероятности можно рассматривать с двух точек зрения: p(хi|yj) и p(yj|хi). На рисунке далее представлен ДСК. На рисунке обозначены: q – вероятность правильной (безошибочной) передачи бита сообщения, р – вероятность передачи бита с ошибкой. Понятно, что p + q = 1. Информационной характеристикой алфавита (источника сообщений на основе этого алфавита) является энтропия. Этот термин применительно к техническим системам был введен К. Шенноном и Р. Хартли.



Энтропию алфавита А = {ai} по К. Шеннону рассчитывают по следующей формуле:



С физической точки зрения энтропия алфавита показывает, какое количество информации приходится в среднем на один символ алфавита. Частным случаем энтропии Шеннона считается энтропия Хартли. Дополнительным условием при этом является то, что все вероятности одинаковы и постоянны для всех символов алфавита. С учетом этого формулу (2.1) можно преобразовать к виду:



Сообщение Хk, которое состоит из k символов, должно характеризоваться определенным количеством информации I(Хk):



Здесь Н(А) – энтропия алфавита с соответствующим распределением вероятностей р(аi). Если принять, что р(аi = 1) = р(1) и р(аi = 0) = р(0), используя выражение (2.1), вычислим энтропию бинарного алфавита:



К примеру, если сообщение Хk состоит только из единиц (Хk = 11…1) и имеет длину k, то вероятность того, что произвольный символ равен единице, составляет единицу (р(аi = 1) = 1), и другая вероятность р(аi = 0) = 0 для i =1,N . Фактически здесь имеет место использование моноалфавита: алфавита, состоящего из одного символа.

Учитывая, что сумма р(1) + р(0) = 1, и выражая одну вероятность через другую (например, р(1) = 1 – р(0)), можно теоретически доказать информативность бинарного алфавита, решив дифференциальное уравнение [dH(A)/dp(1)] = 0 (вспомним из курса математики, как найти экстремум функции; можно для этого воспользоваться пособием [5]). Если вероятность ошибки в ДСК отлична от 0 (р > 0), переданное сообщение может содержать ошибки: Хk ≠ Yk. Количество информации в таком сообщении при его передаче по ДСК будет определяться не энтропией двоичного алфавита (в соответствии с выражением (2.3)), а эффективной энтропией Hе(A) алфавита или пропускной способностью канала:



где H(Y | X) – условная энтропия:



**Код практического задания:**

char[] EngAlph = { 'a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j', 'k', 'l', 'm', 'n', 'o', 'p', 'q', 'r', 's', 't', 'u', 'v', 'w', 'x', 'y', 'z' };

char[] BelRusAlph = { 'а', 'б', 'в', 'г', 'д', 'е', 'ё', 'ж', 'з', 'i', 'й', 'к', 'л', 'м', 'н', 'о', 'п', 'р', 'с', 'т', 'у', 'Y', 'ф', 'ч', 'ц', 'ч', 'ш', 'ы', 'ь', 'э', 'ю', 'я' };

char[] Eng = { 'h', 'e', 'l', 'l', 'o' };

char[] Belrus = { 'п', 'р', 'ы', 'в', 'i', 'т',};

double lengEng = EngAlph.Length;

double lengBelRus = BelRusAlph.Length;

double Hbinary = 0;//Бинарный язык

double Heng = 0;//Энтропия для каждого языка

double HBelrus = 0;

Console.WriteLine("Кол-во букв в английском алфавите :" + lengEng);

Console.WriteLine("Кол-во букв в белорусском алфавите :" + lengBelRus);

foreach (char i in EngAlph)

{

double count = 0;

foreach (char z in Eng)

{

if (z == i)

count++;

}

double Pai = count / lengEng;

// Console.WriteLine(Pai);

if (Pai != 0)

{

Heng += Pai \* Math.Log2(Pai);

}

}

Heng = Heng \* (-1);

Console.WriteLine("Энтропия английского алфавита:");

Console.WriteLine(Heng);

foreach (char i in BelRusAlph)

{

double count = 0;

foreach (char z in Belrus)

{

if (z == i)

count++;

}

double Pai = count / lengBelRus;

// Console.WriteLine(Pai);

if (Pai != 0)

{

HBelrus += Pai \* Math.Log2(Pai);

}

}

HBelrus = HBelrus \* (-1);

Console.WriteLine("Энтропия белорусского алфавита:");

Console.WriteLine(HBelrus);

int[] countLetter = new int[2];

double[] probabilityLetters = new double[2];

string letters = "01";

using (StreamReader streamReader = new StreamReader(@"D:\3 курс 1 сем\ЗАЩИТА ИНФОРМАЦИИ И НАДЕЖНОСТЬ ИНФОРМАЦИОННЫХ СИСТЕМ\2 Лабораторная работа\binary.txt"))

{

string file = streamReader.ReadToEnd();

double binaryleng = file.Count();

for (int j = 0; j < 2; j++)

{

countLetter[j] = file.Count(x => x == letters[j]);

probabilityLetters[j] = (double)countLetter[j] / binaryleng;

Console.WriteLine();

Hbinary += (probabilityLetters[j] \* (Math.Log(probabilityLetters[j])) / Math.Log(2));

}

}

Hbinary = Hbinary \* (-1);

Console.WriteLine("Энтропия бинарного алфавита:");

Console.WriteLine(Hbinary);

Console.WriteLine("Введите Ваше ФИО: ");

string FIO = Console.ReadLine();

double countBelInformation = HBelrus \* FIO.Replace(" ", "").ToLower().Count();

double countBinaryInformation = Hbinary \* FIO.Replace(" ", "").ToLower().Count();

Console.WriteLine("Количество информации с использованием энтропии беларусского алфавита:");

Console.WriteLine(countBelInformation);

Console.WriteLine("Количество информации с использованием энтропии бинарного алфавита:");

Console.WriteLine(countBinaryInformation);

Console.WriteLine();

double ascii = FIO.ToLower().Count() \* 8;

Console.WriteLine();

Console.WriteLine("Количество информации с использованием ASСII для белорусского алфавита:");

double ASIIBelrInformation = ascii \* HBelrus;

Console.WriteLine(ASIIBelrInformation);

Console.WriteLine();

Console.WriteLine("Количество информации с использованием ASСII для бинарного алфавита:");

double ASIIbinaryInformation = ascii \* Hbinary;

Console.WriteLine(ASIIbinaryInformation);

Console.WriteLine("Вероятность ошибочной передачи единичного бита сообщения состовляет 0,1");

float p = 0.1F;

float q = 1 - p;

Console.WriteLine("Условная энтропия ");

double H01 = -1 \* p \* (Math.Log2(p) / Math.Log(2)) - q \* (Math.Log2(q) / Math.Log(2));

Console.WriteLine(H01);

Console.WriteLine("Эффективная энтропия");

double effectH01 = 1 - H01;

Console.WriteLine(effectH01);

Console.WriteLine("Вероятность ошибочной передачи единичного бита сообщения состовляет 0,5");

p = 0.5F;

q = 1 - p;

Console.WriteLine("Условная энтропия ");

double H05 = -1 \* p \* (Math.Log2(p) / Math.Log(2)) - q \* (Math.Log2(q) / Math.Log(2));

Console.WriteLine(H05);

Console.WriteLine("Эффективная энтропия");

double effectH05 = 1 - H01;

Console.WriteLine(effectH05);

Console.WriteLine("Вероятность ошибочной передачи единичного бита сообщения состовляет 1,0");

p = 1;

q = 1 - p;

Console.WriteLine("Условная энтропия ");

double H1 = -1 \* p \* Math.Log2(p) - q \* Math.Log2(q);

Console.WriteLine(H1);

Console.WriteLine("Эффективная энтропия");

double effectH1 = 1 - H1;

Console.WriteLine(effectH1);

}

}

**Ответы на контрольные вопросы:**

1. Что такое алфавит источника сообщения?

*Алфавит*– это общее число знаков или символов (*N*), используемых генерации или передачи сообщений.

2. Что такое мощность алфавита источника сообщения?

Количество символов в алфавите.

3. Какова мощность алфавита белорусского языка?

34

4. Какова мощность алфавита русского языка?

33

5. Какова мощность алфавита «компьютерного» языка?

2

6. Что такое энтропия алфавита?

Энтропия – информационная характеристика алфавита (источника сообщений на основе этого алфавита), которая показывает, какое количество информации приходится в среднем на один символ алфавита (сообщения).

С физической точки зрения энтропия алфавита показывает, какое количество информации приходится в среднем на один символ алфавита

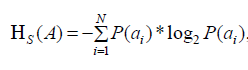
7. Что такое энтропия сообщения?

Энтропия – информационная характеристика алфавита (источника сообщений на основе этого алфавита), которая показывает, какое количество информации приходится в среднем на один символ алфавита (сообщения).

8. От чего зависит энтропия алфавита?

От частоты встречаемости символов и мощности алфавита.

9. Записать формулу для вычисления энтропии.

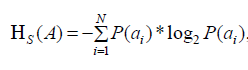


10. Что нужно знать для вычисления энтропии алфавита?

Частоту встречаемости каждого символа алфавита и мощность.

11. Как рассчитываются энтропия Шеннона и энтропия Хартли? В чем принципиальное различие между этими характеристиками? Дайте толкование физического смысла энтропии.

Энтропию алфавита А={*ai*} по К.Шеннону рассчитывают по следующей формуле, представленной на рисунке



Частным случаем энтропии Шеннона является энтропия Хартли. Дополнительным условием при этом является то, что все вероятности одинаковы и постоянны для всех символов алфавита. С учетом этого формулу (2.1) можно преобразовать к виду, представленному на рисунке

https://lh6.googleusercontent.com/iVtaU6Me-5i_RKzvvF6QnO0j7MxiztNrSXZBNC9RRZ0wTtZ5d-vnXp1zOR3kojDdVF_vMibCAvpJgsK8HWeTepN2fzjOcZJJiHvyR3q9iTTTlsZtxnVGJYJrnwnO9AYDMZRk6DcC-Edbb2_YQL0vlxqPbEaW0BMUXNam5WQtmmbq9vyczvVzpOQPVJPCMC5bpjsE

12. Пояснить назначение знака «минус» в формулах (2.1) и (2.4).

Поскольку мы находим логарифмы от чисел меньших 1, то логарифм всегда будет отрицательным, поэтому мы добавляем минус перед логарифмом

13. Что такое избыточность алфавита и избыточность сообщений,

сформированных в компьютерных системах? Принцип действия каких систем основан на существовании данной избыточности?

Избыточностью алфавита называется уменьшение информационной нагрузки на один символ вследствие неравновероятности и взаимозависимости появления его символов.

Информационная избыточность характеризует относительную нагруженность алфавита.

14. Расположить в порядке возрастания энтропии известные Вам алфавиты.

3,2 – белорусского

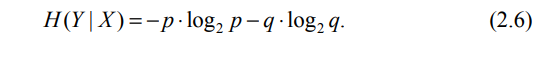
3,895 – французского

4,7 – английского

15. Вычислить энтропию алфавита белорусского (русского) языка.

3.2 бит

16. Вычислить энтропию Шеннона бинарного алфавита, если вероятность появления в произвольном документе на основе этого алфавита одного из символов составляет 0,25, другого – 0,75; либо 0 и 1,0; либо 0,5 и 0,5.



H(А) = 1-(0,25\*log20,25 + 0,75\*log20.75)

при p =0,25 и q = 0,75

H(А) = 1-(0,5\*log20,5 + 0,5\*log20.5) = 0

при p =0,5 и q = 0,5

H(А) = 1-(1\*log21+ 0\*log20) = 1

при p =1 и q = 0

17. Чему равна энтропия алфавита по Хартли, если мощность этого алфавита равна: а) 1 символ; б) 2 символа; в) 8 символов?

a) -1\*log21 = 0

б) -(0.5\*log20.5 + 0.5\*log20.5 ) = 1

в) -1\*log2(1/8) = 3

Вывод: в ходе лабораторной работы были приобретены практические навыки расчета и анализа параметров и информативных характеристик дискретных ИС, высчитаны энтропии текстов на латинице кириллице и текста в бинарном виде. Если посмотреть на гистограмму, то станет заметно, что чаще всего встречаются гласные буквы.

Учреждение образования

«Белорусский государственный технологический университет»

**Защита информации и надежность информационных систем**

**Лабораторная работа №3**

ЭЛЕМЕНТЫ ТЕОРИИ ИНФОРМАЦИИ.

ИНФОРМАТИВНОСТЬ ДАННЫХ

В РАЗЛИЧНЫХ КОДИРОВКАХ

Выполнил:

Студент 3 курса 2 группы ФИТ

Максимова Вера Владимировна

Проверил:

Ржеутская Надежда Викентьевна

**2022 г.**

Цель: приобретение практических навыков трансформации  
данных и сопоставление энтропийных свойств используемых при  
этом алфавитов

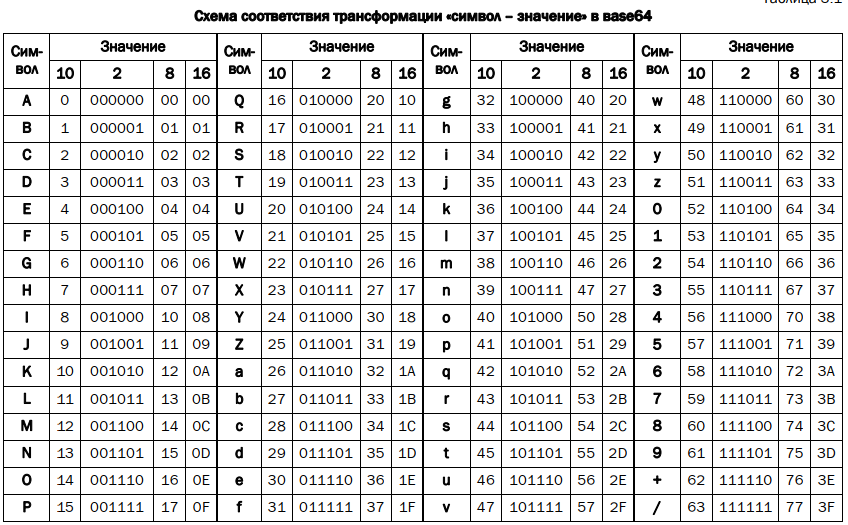
**Задачи:**

1. Конвертировать произвольный документ на латинице в документ формата Base64
2. С помощью приложения разработанного в лабораторной №1 рассчитать распределение частотных свойств Латинского и Base64 алфавитов и вычислить энтропию по Хартли и по Шеннону. Используя рассчитанную энтропию вычислить избыточность алфавитов.
3. Написать функцию расчёта операции XOR над двумя строками и получить для двух строк a и b результат применения: a XOR b и a XOR b XOR b.

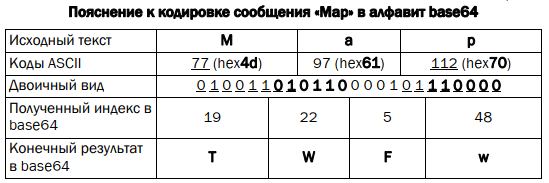
**Теоретические сведения**

**Избыточностью алфавита** называют уменьшение информационной нагрузки на один символ вследствие разной вероятности и взаимозависимости появления его символов в сообщениях.

****

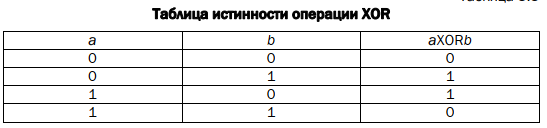
****

Рассмотрим пример. Пусть необходимо закодировать сообщение «Мар».

****

В приведенной таблице подчеркивание используется для разделения двоичных кодов символов в ASCII.

При изучении раздела курса, касающегося криптографического преобразования данных, мы вернемся к вопросу о расширения области применения base64-формата. Сейчас же ограничимся рассмотрением особенностей дальнейшего преобразования данных этого формата с использованием операции XOR (вспомним, что эта операция называется также cложением по модулю 2, логическим сложением, исключающим «ИЛИ», строгой дизъюнкций, поразрядным дополнением).

****

**Код разработанного приложения**

**using System;**

**using System.Collections.Generic;**

**using System.IO;**

**using System.Linq;**

**using System.Text;**

**using System.Threading.Tasks;**

**namespace Lab3**

**{**

**class Program**

**{**

**static void Main(string[] args)**

**{**

**//LAB 2**

**//task 1**

**ToBase64 toBase64 = new ToBase64();**

**toBase64.MethodToBase64();**

**//task 2**

**Superfluity izbitochoct = new Superfluity();**

**izbitochoct.CalculationSuperfluity();**

**//task 3**

**XORNameFamily xORNameFamily = new XORNameFamily();**

**xORNameFamily.CalculateXOR("Vera", "Maksimova");**

**Console.WriteLine();**

**}**

**class ToBase64**

**{**

**public void MethodToBase64()**

**{**

**string base64 = "";**

**using (StreamReader streamReader = new StreamReader(@"D:\3 курс 1 сем\ЗАЩИТА ИНФОРМАЦИИ И НАДЕЖНОСТЬ ИНФОРМАЦИОННЫХ СИСТЕМ\3 Лабораторная работа\Lab3\english.txt"))**

**{**

**string file = streamReader.ReadToEnd();**

**Console.WriteLine("Исходный текст:");**

**Console.WriteLine(file);**

**Console.WriteLine();**

**for (int i = 0; i < file.Length; i++)**

**{**

**byte[] ascii = Encoding.ASCII.GetBytes(file);**

**base64 = Convert.ToBase64String(ascii);**

**}**

**Console.WriteLine("Перевод в base64:");**

**Console.WriteLine(base64);**

**}**

**using (StreamWriter sw = new StreamWriter(@"D:\3 курс 1 сем\ЗАЩИТА ИНФОРМАЦИИ И НАДЕЖНОСТЬ ИНФОРМАЦИОННЫХ СИСТЕМ\3 Лабораторная работа\Lab3\base64.txt", false))**

**{**

**sw.WriteLine(base64);**

**}**

**}**

**}**

**class Superfluity**

**{**

**public void CalculationSuperfluity()**

**{**

**//english**

**EntropyEnglish entropyenglish = new EntropyEnglish();**

**entropyenglish.EntropyEnglishAlphabet();**

**double entropyEnglishAlphabetHartly = Math.Log(33, 2);**

**Console.WriteLine($"Энтропия по Хартли (EN): {entropyEnglishAlphabetHartly}");**

**double superfluityEnglish = 1 - (ShannonEntropyOfTheGAlphabet / entropyEnglishAlphabetHartly);**

**Console.WriteLine($"Избыточность информации (EN): {superfluityEnglish}");**

**//base**

**EntropyBase entropyBase = new EntropyBase();**

**entropyBase.EntropyBaseAlphabet();**

**double entropyBinaryAlphabetHartly = Math.Log(64, 2);**

**Console.WriteLine($"Энтропия по Хартли Base текста: {entropyBinaryAlphabetHartly}");**

**double superfluityBinary = 1 - (ShannonEntropyOfTheBAlphabet / entropyBinaryAlphabetHartly);**

**Console.WriteLine($"Избыточность информации Base текста: {superfluityBinary}");**

**}**

**}**

**class XORNameFamily**

**{**

**public void CalculateXOR(string a, string b)**

**{**

**byte[] asciiName = Encoding.ASCII.GetBytes(a);**

**byte[] asciFamily = Encoding.ASCII.GetBytes(b);**

**string base64Name = "";**

**string base64Family = "";**

**byte[] newasciiName = new byte[] { };**

**byte[] newasciiFamily = new byte[] { };**

**if (asciiName.Length < asciFamily.Length)**

**{**

**Console.WriteLine();**

**newasciiFamily = asciFamily;**

**newasciiName = new byte[asciFamily.Length];**

**for (int i = 0**

**; i < asciiName.Length; i++)**

**{**

**newasciiName[i] = asciiName[i];**

**Console.Write(newasciiName[i] + " ");**

**}**

**for (int j = asciiName.Length; j < asciFamily.Length; j++)**

**{**

**newasciiName[asciiName.Length + 1] = 0;**

**Console.Write(newasciiName[j] + " ");**

**}**

**Console.WriteLine();**

**for (int i = 0; i < asciFamily.Length; i++)**

**{**

**Console.Write(newasciiFamily[i] + " ");**

**}**

**base64Name = Convert.ToBase64String(newasciiName);**

**base64Family = Convert.ToBase64String(asciFamily);**

**Console.WriteLine();**

**Console.WriteLine(base64Name);**

**Console.WriteLine(base64Family);**

**}**

**else**

**{**

**Console.WriteLine();**

**newasciiName = asciiName;**

**newasciiFamily = new byte[asciiName.Length];**

**for (int i = 0; i < asciFamily.Length; i++)**

**{**

**newasciiFamily[i] = asciFamily[i];**

**Console.WriteLine(newasciiFamily[i]);**

**}**

**for (int j = asciFamily.Length; j < asciiName.Length; j++)**

**{**

**newasciiFamily[asciFamily.Length + 1] = 0;**

**Console.WriteLine(newasciiFamily[j]);**

**}**

**base64Name = Convert.ToBase64String(newasciiName);**

**base64Family = Convert.ToBase64String(asciFamily);**

**Console.WriteLine(base64Name);**

**Console.WriteLine(base64Family);**

**}**

**Console.WriteLine("\n\nРезультат логического сложения в ASCII:");**

**int[] result = new int[newasciiName.Length];**

**int[] aaa = new int[base64Name.Length];**

**for (int i = 0; i < newasciiName.Length; i++)**

**{**

**result[i] = newasciiName[i] ^ newasciiFamily[i] ^ newasciiFamily[i];**

**//Console.Write(result[i] + " ");**

**char ch = (char)result[i];**

**Console.Write(ch);**

**}**

**string res = "";**

**Console.WriteLine();**

**for (int i = 0; i < base64Name.Length; i++)**

**{**

**aaa[i] = base64Name[i] ^ base64Family[i] ^ base64Family[i];**

**//Console.Write(result[i] + " ");**

**char ch = (char)aaa[i];**

**res += ch;**

**Console.Write(ch);**

**}**

**Console.WriteLine();**

**Console.WriteLine(Encoding.ASCII.GetString(Convert.FromBase64String(res)));**

**}**

**}**

**static double ShannonEntropyOfTheGAlphabet = 0;**

**static double ShannonEntropyOfTheBAlphabet = 0;**

**static double ShannonEntropyOfTheBinaryAlphabet = 0;**

**class EntropyEnglish**

**{**

**public void EntropyEnglishAlphabet()**

**{**

**int[] countLetter = new int[26];**

**int countLettersInFile = 0;**

**double[] probabilityLetters = new double[26];**

**// double ShannonEntropyOfTheAlphabet = 0;**

**string letters = "abcdefghijklmnopqrstuvwxyz";**

**using (StreamReader streamReader = new StreamReader(@"D:\3 курс 1 сем\ЗАЩИТА ИНФОРМАЦИИ И НАДЕЖНОСТЬ ИНФОРМАЦИОННЫХ СИСТЕМ\3 Лабораторная работа\Lab3\english.txt"))**

**{**

**string file = streamReader.ReadToEnd().ToLower();**

**countLettersInFile = file.Count();**

**Console.WriteLine($"Количество символов в файле: {countLettersInFile}");**

**Console.WriteLine();**

**Console.WriteLine("Количество вхождений каждой буквы:\n");**

**for (int j = 0; j < 26; j++)**

**{**

**countLetter[j] = file.Count(x => x == letters[j]);**

**if (countLetter[j] != 0)**

**{**

**Console.WriteLine($"{letters[j]}: {countLetter[j]}");**

**probabilityLetters[j] = (double)countLetter[j] / countLettersInFile;**

**Console.WriteLine($"P({letters[j]}): {probabilityLetters[j]}");**

**Console.WriteLine();**

**ShannonEntropyOfTheGAlphabet += probabilityLetters[j] \* (Math.Log(probabilityLetters[j]) / Math.Log(2)) \* (-1);**

**}**

**}**

**Console.WriteLine("Энтропия Английского алфавита по Шеннону:");**

**Console.WriteLine(ShannonEntropyOfTheGAlphabet);**

**}**

**}**

**}**

**class EntropyBase**

**{**

**public void EntropyBaseAlphabet()**

**{**

**int[] countLetter = new int[64];**

**int countLettersInFile = 0;**

**double[] probabilityLetters = new double[64];**

**// double ShannonEntropyOfTheAlphabet = 0;**

**string letters = "ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+/";**

**using (StreamReader streamReader = new StreamReader(@"D:\3 курс 1 сем\ЗАЩИТА ИНФОРМАЦИИ И НАДЕЖНОСТЬ ИНФОРМАЦИОННЫХ СИСТЕМ\3 Лабораторная работа\Lab3\base64.txt"))**

**{**

**string file = streamReader.ReadToEnd().ToLower();**

**countLettersInFile = file.Count();**

**Console.WriteLine($"Количество символов в файле: {countLettersInFile}");**

**Console.WriteLine();**

**Console.WriteLine("Количество вхождений каждой буквы:\n");**

**for (int j = 0; j < 64; j++)**

**{**

**countLetter[j] = file.Count(x => x == letters[j]);**

**if (countLetter[j] != 0)**

**{**

**Console.WriteLine($"{letters[j]}: {countLetter[j]}");**

**probabilityLetters[j] = (double)countLetter[j] / countLettersInFile;**

**Console.WriteLine($"P({letters[j]}): {probabilityLetters[j]}");**

**Console.WriteLine();**

**ShannonEntropyOfTheBAlphabet += probabilityLetters[j] \* (Math.Log(probabilityLetters[j]) / Math.Log(2)) \* (-1);**

**}**

**}**

**Console.WriteLine("Энтропия алфавита Base по Шеннону:");**

**Console.WriteLine(ShannonEntropyOfTheBAlphabet);**

**}**

**}**

**}**

**class EntropyBinaryDigit**

**{**

**public void EntropyBinaryAlphabet()**

**{**

**int[] countLetter = new int[2];**

**int countLettersInFile = 0;**

**double[] probabilityLetters = new double[2];**

**//double ShannonEntropyOfTheBinaryAlphabet = 0;**

**string letters = "01";**

**using (StreamReader streamReader = new StreamReader(@"D:\3 курс 1 сем\ЗАЩИТА ИНФОРМАЦИИ И НАДЕЖНОСТЬ ИНФОРМАЦИОННЫХ СИСТЕМ\3 Лабораторная работа\Lab3\base64.txt"))**

**{**

**string file = streamReader.ReadToEnd();**

**countLettersInFile = file.Count();**

**Console.WriteLine();**

**Console.WriteLine();**

**Console.WriteLine("BINARY DIGIT");**

**Console.WriteLine($"Количество символов в файле: {countLettersInFile}");**

**Console.WriteLine();**

**Console.WriteLine("Количество вхождений каждой цифры:");**

**for (int i = 0; i < file.Length; i++)**

**{**

**for (int j = 0; j < 2; j++)**

**{**

**countLetter[j] = file.Count(x => x == letters[j]);**

**Console.WriteLine($"{letters[j]}: {countLetter[j]}");**

**probabilityLetters[j] = (double)countLetter[j] / countLettersInFile;**

**Console.WriteLine($"P({letters[j]}): {probabilityLetters[j]}");**

**Console.WriteLine();**

**ShannonEntropyOfTheBinaryAlphabet += (probabilityLetters[j] \* (Math.Log(probabilityLetters[j])) / Math.Log(2)) \* (-1);**

**}**

**Console.WriteLine($"Энтропия бинарного алфавита по Шеннону: {ShannonEntropyOfTheBinaryAlphabet}");**

**break;**

**}**

**}**

**}**

**}**

**}**

**}**

**Ответы на контрольные вопрсоы:**

1. Что такое base64?

Base64 кодовая таблица которая состоит из 64 символов. И кодируется шестью битами. Используется для кодирования данных

2. Как проверить, была ли определенная строка символов закодирована в base64?

Если строка символов закодирована в Base64, то её длинна будет кратна шести, а также строка будет содержать только символы из кодировки Base64 или специальный символ «=».

3. Как с помощью base64 проверить подлинность вводимых данных в форму пароля и логина?

Если пароль хранится в кодировке base64, то проверить корректность введённых данных можно преобразовав их в формат base64 и сравнив длину полученного сообщения и пароля, можно определить некорректность введённых данных, если же длинна строк совпадает, то идёт посимвольное сравнение строк.

4. Охарактеризовать энтропийные свойства алфавитов в проанализированных форматах данных.

Энтропия алфавита Base64 меньше чем у английского алфавита и кодов ASCII, что означает меньшую избыточность Base64 алфавита.

5. Объяснить результат операции аXORbXORb. Где может найти применение такая операция?

b xor b даст в результате нулевой битовый вектор.

a xor 0 = a.

В результате операции аXORbXORb получится сообщение а;

Эта операция может найти применение в строке, где находится много операторов. Из-за того, что XOR коммутативна, мы можем сокращать повторяющиеся операторы.

6. Как будут выглядеть строки:

efd8b295a633908a3c0828b2, faea8766, 4d72cde3aaa0, после их конвертации в base64?

efd8b295a633908a3c0828b2=> ZWZkOGIyOTVhNjMzOTA4YTNjMDgyOGIy

faea8766 => ZmFlYTg3NjY

4d72cde3aaa0 = > NGQ3MmNkZTNhYWEwIA

7. Результатом операции аXORb (а – каждый байт строки, b – некоторая неизменная величина) будет строка:

1f180d1e1f04051c404c0f19

1f180308050d024c030a4c18

04094c1f18030009024c1c00

Найти значение b.

Пусть a1 xor b = 1f180d1e1f04051c404c0f19 = с1, a2 xor b = 1f180308050d024c030a4c18 = с2, a3 xor b = 04094c1f18030009024c1c00 = с3.

Тогда c1 xor c2 xor c3 = a1 xor b xor a2 xor b xor a3 xor b = a1 xor a2 xor a3 xor b.

a1 xor a3 = c1 xor c3 = 1b1141010707051542001319

a2 xor a3 = c2 xor c3 = 1b114f171d0e024501465018

a1 xor a2 = c1 xor c2 = e161a09075043464301

c1 xor c2 xor c3 = 4094209020a0759410a5f01

Вывод: в ходе лабораторной были исследованы сообщения в кодах ASCII и base64. Было выяснено, что кодировка base64 эффективнее использовать, чем кодировку ASCII так как она обладает меньшей избыточностью.

Учреждение образования

«Белорусский государственный технологический университет»

**Защита информации и надежность информационных систем**

**Лабораторная работа №4**

ИЗБЫТОЧНОЕ КОДИРОВАНИЕ ДАННЫХ

В ИНФОРМАЦИОННЫХ СИСТЕМАХ.

КОД ХЕММИНГА

Выполнил:

Студент 3 курса 2 группы ФИТ

Максимова Вера Владимировна

Проверил:

Ржеутская Надежда Викентьевна

**2022 г.**

Цель: приобретение практических навыков кодирования/декодирования двоичных данных при использовании кода Хемминга.

Задачи:

1. Закрепить теоретические знания по использованию методов помехоустойчивого кодирования для повышения надежности передачи и хранения в памяти компьютера двоичных данных.

2. Разработать приложение для кодирования/декодирования двоичной информации кодом Хемминга с минимальным кодовым расстоянием 3 или 4.

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

**Теоретические сведения**

**Надежность системы** – характеристика способности программного, аппаратного, аппаратно-программного средства выполнить при определенных условиях требуемые функции в течение конкретного периода времени.

**Достоверность** работы системы (устройства) – свойство, характеризующее истинность конечного (выходного) результата работы (выполнения программы), определяемое способностью средств контроля фиксировать правильность или ошибочность работы.

**Ошибка** устройства – неправильное значение сигнала (бита – в цифровом устройстве) на внешних выходах устройства или отдельного его узла, вызванное технической неисправностью, или воздействующими на него помехами (преднамеренными либо непреднамеренными), или иным способом.

**Ошибка** программы – проявляется в не соответствующем реальному (требуемому) промежуточном или конечном значении (результате) вследствие неправильно запрограммированного алгоритма или неправильно составленной программы.

**Безотказность** – это свойство технического объекта непрерывно сохранять работоспособное состояние в течение некоторого времени (или наработки). Наработка, как правило, измеряется в единицах времени.

**Ремонтопригодность** – это свойство технического объекта, заключающееся в приспособленности к поддержанию и восстановлению работоспособного состояния путем технического обслуживания, ремонта (или с помощью дополнительных, избыточных технических средств, функционирующих параллельно с объектом). Большинство современных цифровых систем и устройств (в том числе компьютеры и компьютерные системы, отдельные блоки и модули компьютеров – полупроводниковая, магнитная или оптическая память) содержат специальные средства, призванные автоматически восстанавливать работоспособность этих объектов при нарушении нормального функционирования.

Такие специальные средства контроля называются *избыточными*.

Изначальной причиной нарушения нормальной работы цифрового устройства являются технические дефекты (неисправности), возникающие внутри узлов или блоков устройства либо в каналах связи между ними.

**Линейные блочные коды** – это класс кодов с контролем четности, которые можно описать парой чисел (п, k).

Для формирования r проверочных символов (кодирования), т. е. вычисления проверочного слова Xr, используется порождающая матрица G: совокупность базисных векторов будем далее записывать в виде матрицы G размерностью k×n с единичной подматрицей (I) в первых k строках и столбцах:

G = [P | I] .

Более точно матрица G называется порождающей матрицей линейного корректирующего кода в приведено-ступенчатой форме. Кодовые слова являются линейными комбинациями строк матрицы G (кроме слова, состоящего из нулевых символов).

Для всякой порождающей матрицы G существует матрица Н размерности r×n, задающая базис нулевого пространства кода и удовлетворяющая равенству

G\*HT = 0

Справедливо также

Xn\*HT=h\*(Xn)T=0

В последнем выражении символ «T» означает транспонирование, а Xn = x1, x2, … xn.

В коде Хемминга с минимальным кодовым расстоянием dmin = 3 проверочная матрица Н имеет классический вид и состоит из двух подматриц: P’ размером k×r и I размером r×r соответственно.

**Код разработанного приложения:**

using System;

namespace laba5

{

class Program

{

static void Main(string[] args)

{

string str;

do

{

Console.Clear();

Console.Write("Введите строку (не менее трех символов) = ");

str = Console.ReadLine();

}

while (str.Length < 3);

int k = str.Length;

int r = LenghtHemminga(k);

int n = k + r;

int[] mas = new int[str.Length + r];

int[,] checkMatrix = new int[n, r];

int error;

//Преобразование строки в массив

mas = StrInMas(str, k);

Console.Write("\nВходная строка = ");

OutMass(mas, k);

//Получение проверочной матрицы

checkMatrix = CheckMatrix(k);

Console.WriteLine("\nПроверочная матрица");

OutMass(checkMatrix, k);

//Добавление проверочных битов

Sindrom(checkMatrix, mas, k);

Console.WriteLine("\nПолная строка");

OutMass(mas, k);

try

{

Console.WriteLine("Введите место первой ошибки");

error = Convert.ToInt32(Console.ReadLine()) - 1;

if (mas[error] == 1) mas[error] = 0;

else mas[error] = 1;

}

catch { }

try

{

Console.WriteLine("Введите место второй ошибки");

error = Convert.ToInt32(Console.ReadLine()) - 1;

if (mas[error] == 1) mas[error] = 0;

else mas[error] = 1;

}

catch { }

Console.WriteLine("\nСтрока с ошибкой");

OutMass(mas, k);

mas = SearchError(mas, checkMatrix, k);

Console.WriteLine("\nСтрока без ошибки");

OutMass(mas, k);

}

//Считаем r (кол-во пров. симв.)

public static int LenghtHemminga(int k)

{

int r = (int)(Math.Log(k, 2) + 2.99f);

return r;

}

//Создание пров. матрицы

public static int[,] CheckMatrix(int k)

{

int r = LenghtHemminga(k);

int n = r + k;

int[,] mas = new int[n, r];

double rDouble = r - 1;

int rPow = (int)(Math.Pow(2, rDouble));

int[,] combinations = new int[rPow, r];

//все заполняем нулями

for (int i = 0; i < rPow; i++)

for (int j = 0; j < r; j++)

combinations[i, j] = 0;

//генератор бит.мн.

for (int segmentLenght = 0; segmentLenght < r - 2; segmentLenght++)

{

for (int i = 0; i < segmentLenght + 2; i++)

{

combinations[segmentLenght \* (r - 1), i] = 1;

}

for (int segmentPositin = 1; segmentPositin < r - 1; segmentPositin++)

{

for (int i = 0; i < r - 2; i++)

{

combinations[segmentLenght \* (r - 1) + segmentPositin, i + 1] = combinations[segmentLenght \* (r - 1) + segmentPositin - 1, i];

}

combinations[segmentLenght \* (r - 1) + segmentPositin, 0] = combinations[segmentLenght \* (r - 1) + segmentPositin - 1, r - 2];

}

//Заполнение посл. сроки 1-ми

if (segmentLenght == r - 2)

{

for (int i = 0; i < r; i++)

{

combinations[rPow - 1, i] = 1;

}

}

}

for (int i = 0; i < k; i++)

{

int amount = 0;

for (int j = 0; j < r - 1; j++)

{

if (combinations[i, j] == 1) amount++;

}

if (amount % 2 == 0)

combinations[i, r - 1] = 1;

}

for (int i = 0; i < k; i++)

for (int j = 0; j < r; j++)

mas[i, j] = combinations[i, j];

for (int i = 0; i < r; i++)

mas[i + k, i] = 1;

return mas;

}

//Поиск синдрома

public static int[] Sindrom(int[,] CheckMatrix, int[] mas, int k)

{

int r = LenghtHemminga(k);

int n = r + k;

int[] sindrom = new int[r];

for (int i = 0, l = 0; i < r; i++, l = 0)

{

for (int j = 0; j < k; j++)

{

if (CheckMatrix[j, i] == 1 && mas[j] == 1) l++;

else sindrom[i] = 0;

}

if (l % 2 == 1) sindrom[i] = 1;

else sindrom[i] = 0;

}

for (int i = 0; i < r; i++)

{

mas[i + k] = sindrom[i];

}

return mas;

}

//Нахождение ошибок

public static int[] SearchError(int[] mas, int[,] checkMatrix, int k)

{

int r = LenghtHemminga(k);

int n = r + k;

int[] beforeSindrom = new int[r];

//запоминаем проверочные биты

for (int i = k; i < n; i++)

{

beforeSindrom[i - k] = mas[i];

}

mas = Sindrom(checkMatrix, mas, k);

//Складываем синдром по модулю два

for (int i = k, j = 0; i < n; i++)

{

if (beforeSindrom[i - k].Equals(mas[i]))

{

mas[i] = 0;

j++;

//если сумма по модулю два все пров. бит равны нулю

if (j == r)

{

for (int l = k; l < n; l++)

{

mas[l] = beforeSindrom[l - k];

}

Console.WriteLine("Ошибок нет");

return mas;

}

}

else

{

mas[i] = 1;

}

}

for (int i = 0; i < n; i++)

{

int l = 0;

for (int j = 0; j < r; j++)

{

if (checkMatrix[i, j].Equals(mas[j + k])) l++;

}

if (l == r)

{

mas[i] = (mas[i] + 1) % 2;

}

if (l != r && i == n - 1)

{

Console.WriteLine("Ошибок кратно 2 и они исправлены неправильно");

}

}

mas = Sindrom(checkMatrix, mas, k);

return mas;

}

//Преобразование строки в массив

public static int[] StrInMas(string str, int k)

{

int r = LenghtHemminga(k);

int[] mas = new int[str.Length + r];

for (int i = 0; i < str.Length; i++)

{

if (str[i] == 49)

mas[i] = 1;

else mas[i] = 0;

}

return mas;

}

//вывод матрицы

public static void OutMass(int[,] mas, int k)

{

int r = LenghtHemminga(k);

int n = r + k;

for (int i = 0; i < r; i++)

{

for (int j = 0; j < n; j++)

{

Console.Write(mas[j, i]);

if (j + 1 == k) Console.Write("|");

}

Console.WriteLine();

}

}

//вывод одномерного массива

public static void OutMass(int[] mas, int k)

{

int n = LenghtHemminga(k) + k;

for (int i = 0; i < n; i++)

{

if (i == k) Console.Write("|");

Console.Write(mas[i]);

}

Console.WriteLine("\n");

}

}

}

**Ответы на контрольные вопросы**

1. В чем заключается цель и функциональная сущность преобразования информации на основе избыточного кодирования?

Цель – нахождение ошибок и при возможности их исправление.

Функциональная сущность – добавление к произвольному сообщению проверочных(избыточных) символов

2. Пояснить зависимость r от длины информационного слова k. Охарактеризовать относительную избыточность сообщения и время его передачи по сети.

r = log2(k) + 1

Относительная избыточность = r/(r+k).

Время передачи сообщения по сети: v\*k/n, где v – производительность источника сообщения (кол-во символов в секунду обычно принимают как 1), а n = r+k.

3. Записать проверочную матрицу кода Хемминга с dmin = 3 и dmin = 4 для k = 4; 6; 8; 9; 10; 15; 16.

dmin = 3:

Для k = 4; r = 3;

1 1 1 0| 1 0 0

1 1 0 1| 0 1 0

1 0 1 1| 0 0 1

Для k = 6; r = 4;

1 1 0 1 1 1 | 1 0 0 0

1 0 0 1 1 0 | 0 1 0 0

0 0 1 1 0 1 | 0 0 1 0

0 1 1 0 1 1 | 0 0 0 1

Для k = 8; r = 4;

1 1 0 1 1 1 0 1 | 1 0 0 0

1 0 0 1 1 0 1 1 | 0 1 0 0

0 0 1 1 0 1 1 1 | 0 0 1 0

0 1 1 0 1 1 1 1 | 0 0 0 1

Для k = 9; r = 5;

1 1 1 1 1 1 1 1 0 | 1 0 0 0 0

1 0 0 0 1 1 1 0 1 | 0 1 0 0 0

0 1 0 0 1 1 0 1 1 | 0 0 1 0 0

0 0 1 0 1 0 1 1 1 | 0 0 0 1 0

0 0 0 1 0 1 1 1 1 | 0 0 0 0 1

Для k = 10; r = 5;

1 1 1 1 1 1 1 1 1 1 | 1 0 0 0 0

1 0 0 0 1 1 1 0 0 0 | 0 1 0 0 0

0 1 0 0 1 0 0 1 1 0 | 0 0 1 0 0

0 0 1 0 0 1 0 1 0 1 | 0 0 0 1 0

0 0 0 1 0 0 1 0 1 1 | 0 0 0 0 1

Для k = 15; r = 5;

1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 | 1 0 0 0 0

1 0 0 0 1 1 1 0 0 0 1 1 1 0 0 | 0 1 0 0 0

0 1 0 0 1 0 0 1 1 0 1 0 0 1 1 | 0 0 1 0 0

0 0 1 0 0 1 0 1 0 1 0 1 0 1 0 | 0 0 0 1 0

0 0 0 1 0 0 1 0 1 1 0 0 1 0 1 | 0 0 0 0 1

Для k = 16; r = 5;

1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 1 | 1 0 0 0 0

1 0 0 0 1 1 1 0 0 0 1 1 1 0 0 1 | 0 1 0 0 0

0 1 0 0 1 0 0 1 1 0 1 0 0 1 1 1 | 0 0 1 0 0

0 0 1 0 0 1 0 1 0 1 0 1 0 1 0 1 | 0 0 0 1 0

0 0 0 1 0 0 1 0 1 1 0 0 1 0 1 1 | 0 0 0 0 1

dmin = 4:

Для k = 4; r = 3;

1 1 1 0 | 1 0 0 0

1 1 0 1 | 0 1 0 0

1 0 1 1 | 0 0 1 0

1 1 1 1 | 1 1 1 1

Для k = 6; r = 4;

1 1 0 1 1 1 | 1 0 0 0 0

1 0 0 1 1 0 | 0 1 0 0 0

0 0 1 1 0 1 | 0 0 1 0 0

0 1 1 0 1 1 | 0 0 0 1 0

1 1 1 1 1 1 | 1 1 1 1 1

Для k = 8; r = 4;

1 1 0 1 1 1 0 1 | 1 0 0 0 0

1 0 0 1 1 0 1 1 | 0 1 0 0 0

0 0 1 1 0 1 1 1 | 0 0 1 0 0

0 1 1 0 1 1 1 1 | 0 0 0 1 0

1 1 1 1 1 1 1 1 | 1 1 1 1 1

Для k = 9; r = 5;

1 1 1 1 1 1 1 1 0 | 1 0 0 0 0 0

1 0 0 0 1 1 1 0 1 | 0 1 0 0 0 0

0 1 0 0 1 1 0 1 1 | 0 0 1 0 0 0

0 0 1 0 1 0 1 1 1 | 0 0 0 1 0 0

0 0 0 1 0 1 1 1 1 | 0 0 0 0 1 0

1 1 1 1 1 1 1 1 1 | 1 1 1 1 1 1

Для k = 10; r = 5;

1 1 1 1 1 1 1 1 1 1 | 1 0 0 0 0 0

1 0 0 0 1 1 1 0 0 0 | 0 1 0 0 0 0

0 1 0 0 1 0 0 1 1 0 | 0 0 1 0 0 0

0 0 1 0 0 1 0 1 0 1 | 0 0 0 1 0 0

0 0 0 1 0 0 1 0 1 1 | 0 0 0 0 1 0

1 1 1 1 1 1 1 1 1 1 | 1 1 1 1 1 1

Для k = 15; r = 5;

1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 | 1 0 0 0 0 0

1 0 0 0 1 1 1 0 0 0 1 1 1 0 0 | 0 1 0 0 0 0

0 1 0 0 1 0 0 1 1 0 1 0 0 1 1 | 0 0 1 0 0 0

0 0 1 0 0 1 0 1 0 1 0 1 0 1 0 | 0 0 0 1 0 0

0 0 0 1 0 0 1 0 1 1 0 0 1 0 1 | 0 0 0 0 1 0

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 | 1 1 1 1 1 1

Для k = 16; r = 5;

1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 1 | 1 0 0 0 0 0

1 0 0 0 1 1 1 0 0 0 1 1 1 0 0 1 | 0 1 0 0 0 0

0 1 0 0 1 0 0 1 1 0 1 0 0 1 1 1 | 0 0 1 0 0 0

0 0 1 0 0 1 0 1 0 1 0 1 0 1 0 1 | 0 0 0 1 0 0

0 0 0 1 0 0 1 0 1 1 0 0 1 0 1 1 | 0 0 0 0 1 0

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 | 1 1 1 1 1 1

4. Записать проверочную матрицу кода простой четности для k из вопроса 3. Пояснить на примере определение минимального кодового расстояния Хемминга для данного кода.

Для k = 4; r = 3;

1 1 1 0 | 1 0 0 0

1 1 0 1 | 0 1 0 0

1 0 1 1 | 0 0 1 0

1 1 1 1 | 1 1 1 1

Минимальное расстояние Хемминга равно минимальному количеству столбцов, которые нужно сложить, чтобы получился нулевой столбец. В данном случае нужно сложить минимум 4 столбца => минимальное кодовое расстояние Хемминга равно 4.

Можно сложить столбцы под номерами 3, 4, 5, 6.

5. Предположим, есть выбор (при построении матрицы кода) между вектор-столбцами большего и меньшего веса. Какой вариант Вы предпочтете и почему?

При выборе между вектор-столбцами большего и меньшего веса лучше выбирать вектор-столбцы меньшего веса, так как при меньшем количестве нулей в матрице будет легче считать избыточные биты.

Вывод: в ходе лабораторной работы были приобретены практические навыки кодирования/декодирования двоичных данных при использовании кода Хемминга, было определено, что, если при передаче сообщения возникают две ошибки, то эти 2 ошибки нельзя исправить стандартным методом.

Учреждение образования

«Белорусский государственный технологический университет»

**Кафедра информационных систем и технологий**

**Защита информации и надёжность информационных систем**

**Лабораторная работа №5**

Избыточное кодирование данных в информационных системах. Итеративные коды.

Вариант № 6

Выполнил:

Студент 3 курса 2 группы ФИТ

Максимова Вера Владимировна

Цель: приобретение практических навыков кодирование/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. Разобрать 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-, трехмерных матриц (таблиц) и более высоких размерностей. Каждая из отдельных последовательностей информационных символов кодируется определенным линейным кодом (групповым или циклическим). Получаемый таким образом итеративный код также является линейным.

Простейшим из итеративных кодов является двумерный код с проверкой на четность по строкам и столбцам. Итеративные коды, иногда называемые прямоугольными кодами (англ. rectangular code) либо композиционными (англ. product code), являются одними из самых простых (с точки зрения аппаратной реализации) избыточных кодов, позволяющих исправлять ошибки в информационных словах.

Основное достоинство рассматриваемых кодов – простота как аппаратной, так и программной реализации. Основной недостаток – сравнительно высокая избыточность.

**Код практического задания:**

**using System;**

**namespace Lab**

**{**

**class Program**

**{**

**static void Main(string[] args)**

**{**

**Console.WriteLine("6 Вариант");**

**Console.WriteLine("Генирируемое сообщение 20 бит");**

**int[] array\_message = new int[20];**

**Random rand = new Random();**

**for (int i = 0; i < array\_message.Length; i++) {**

**array\_message[i] = rand.Next(0,2);**

**Console.Write(array\_message[i] + " ");**

**}**

**Console.WriteLine('\n' + "Матрица k1 = 2, k2 = 10");**

**for (int i = 0; i < array\_message.Length; i++)**

**{**

**if(i == 10)**

**{**

**Console.WriteLine("");**

**}**

**Console.Write(array\_message[i] + " ");**

**}**

**Console.WriteLine("");**

**#region Расчет паритета строк**

**Console.WriteLine("Паритет строк");**

**int[] array\_paritet\_rows = new int[2];**

**int flag = 0;**

**for (int i = 0; i < array\_message.Length; i++)**

**{**

**if (array\_message[i] == 1)**

**{**

**flag += 1;**

**}**

**if (i == 9)**

**{**

**if (flag % 2 == 0) { array\_paritet\_rows[0] = 0; }**

**else { array\_paritet\_rows[0] = 1; }**

**flag = 0;**

**}**

**if (i == 19)**

**{**

**if (flag % 2 == 0) { array\_paritet\_rows[1] = 0; }**

**else { array\_paritet\_rows[1] = 1; }**

**flag = 0;**

**}**

**}**

**#endregion**

**for (int i = 0; i < array\_paritet\_rows.Length; i++)**

**{**

**Console.Write(array\_paritet\_rows[i] + " ");**

**}**

**Console.WriteLine();**

**#region Расчет паритета столбцов**

**Console.WriteLine("Паритет столбцов");**

**int[] array\_paritet\_colums = new int[10];**

**if ((array\_message[0] + array\_message[10]) % 2 == 0) { array\_paritet\_colums[0] = 0; }**

**else { array\_paritet\_colums[0] = 1; }**

**if ((array\_message[1] + array\_message[11]) % 2 == 0) { array\_paritet\_colums[1] = 0; }**

**else { array\_paritet\_colums[1] = 1; }**

**if ((array\_message[2] + array\_message[12]) % 2 == 0) { array\_paritet\_colums[2] = 0; }**

**else { array\_paritet\_colums[2] = 1; }**

**if ((array\_message[3] + array\_message[13]) % 2 == 0) { array\_paritet\_colums[3] = 0; }**

**else { array\_paritet\_colums[3] = 1; }**

**if ((array\_message[4] + array\_message[14]) % 2 == 0) { array\_paritet\_colums[4] = 0; }**

**else { array\_paritet\_colums[4] = 1; }**

**if ((array\_message[5] + array\_message[15]) % 2 == 0) { array\_paritet\_colums[5] = 0; }**

**else { array\_paritet\_colums[5] = 1; }**

**if ((array\_message[6] + array\_message[16]) % 2 == 0) { array\_paritet\_colums[6] = 0; }**

**else { array\_paritet\_colums[6] = 1; }**

**if ((array\_message[7] + array\_message[17]) % 2 == 0) { array\_paritet\_colums[7] = 0; }**

**else { array\_paritet\_colums[7] = 1; }**

**if ((array\_message[8] + array\_message[18]) % 2 == 0) { array\_paritet\_colums[8] = 0; }**

**else { array\_paritet\_colums[8] = 1; }**

**if ((array\_message[9] + array\_message[19]) % 2 == 0) { array\_paritet\_colums[9] = 0; }**

**else { array\_paritet\_colums[9] = 1; }**

**for (int i = 0; i < array\_paritet\_colums.Length; i++)**

**{**

**Console.Write(array\_paritet\_colums[i] + " ");**

**}**

**#endregion**

**Console.WriteLine('\n'+"Сообщение с избыточными битами");**

**for (int i = 0; i < array\_message.Length; i++)**

**{**

**Console.Write(array\_message[i] + " ");**

**}**

**Console.Write(".");**

**for (int i = 0; i < array\_paritet\_rows.Length; i++)**

**{**

**Console.Write(array\_paritet\_rows[i] + " ");**

**}**

**for (int i = 0; i < array\_paritet\_colums.Length; i++)**

**{**

**Console.Write(array\_paritet\_colums[i] + " ");**

**}**

**Console.WriteLine("");**

**Console.WriteLine("");**

**#region Ситуация когда ошибок 0**

**Console.WriteLine("Ситуация когда ошибок 0");**

**Console.WriteLine('\n' + "Париет строк");**

**for (int i = 0; i < array\_paritet\_rows.Length; i++)**

**{**

**Console.Write(array\_paritet\_rows[i] + " ");**

**}**

**Console.WriteLine('\n' + "XOR");**

**for (int i = 0; i < array\_paritet\_rows.Length; i++)**

**{**

**Console.Write(array\_paritet\_rows[i] + " ");**

**}**

**Console.WriteLine('\n' + "---" + '\n' + "0 0");**

**Console.WriteLine('\n' + "Париет столбцов");**

**for (int i = 0; i < array\_paritet\_colums.Length; i++)**

**{**

**Console.Write(array\_paritet\_colums[i] + " ");**

**}**

**Console.WriteLine('\n' + "XOR");**

**for (int i = 0; i < array\_paritet\_colums.Length; i++)**

**{**

**Console.Write(array\_paritet\_colums[i] + " ");**

**}**

**Console.WriteLine('\n' + "-------------------" + '\n' + "0 0 0 0 0 0 0 0 0 0");**

**Console.WriteLine('\n' + "Ошибок нет");**

**#endregion**

**Console.WriteLine("");**

**#region Ситуация когда ошибка 1**

**Console.WriteLine("Ситуация когда 1 ошибка");**

**int error\_in\_rows = 0;**

**int error\_in\_colums = 0;**

**int[] array\_message\_with\_one\_error = new int[20];**

**array\_message\_with\_one\_error = array\_message;**

**int rand\_index = rand.Next(0, 20);**

**if (array\_message\_with\_one\_error[rand\_index] == 1) { array\_message\_with\_one\_error[rand\_index] = 0; }**

**else**

**{**

**array\_message\_with\_one\_error[rand\_index] = 1;**

**}**

**Console.WriteLine("");**

**for (int i = 0; i < array\_message\_with\_one\_error.Length; i++)**

**{**

**if (i == 10)**

**{**

**Console.WriteLine("");**

**}**

**Console.Write(array\_message\_with\_one\_error[i] + " ");**

**}**

**Console.WriteLine("");**

**Console.WriteLine("Паритет строк");**

**for (int i = 0; i < array\_paritet\_rows.Length; i++)**

**{**

**Console.Write(array\_paritet\_rows[i] + " ");**

**}**

**Console.WriteLine("");**

**Console.WriteLine("Паритет столбцов");**

**for (int i = 0; i < array\_paritet\_colums.Length; i++)**

**{**

**Console.Write(array\_paritet\_colums[i] + " ");**

**}**

**Console.WriteLine("");**

**#region Новый паритет строк**

**Console.WriteLine("Новый паритет строк");**

**int[] array\_paritet\_rows\_with\_one\_error = new int[2];**

**for (int i = 0; i < array\_message\_with\_one\_error.Length; i++)**

**{**

**if (array\_message\_with\_one\_error[i] == 1)**

**{**

**flag += 1;**

**}**

**if (i == 9)**

**{**

**if (flag % 2 == 0) { array\_paritet\_rows\_with\_one\_error[0] = 0; }**

**else { array\_paritet\_rows\_with\_one\_error[0] = 1; }**

**flag = 0;**

**}**

**if (i == 19)**

**{**

**if (flag % 2 == 0) { array\_paritet\_rows\_with\_one\_error[1] = 0; }**

**else { array\_paritet\_rows\_with\_one\_error[1] = 1; }**

**flag = 0;**

**}**

**}**

**for (int i = 0; i < array\_paritet\_rows\_with\_one\_error.Length; i++)**

**{**

**Console.Write(array\_paritet\_rows\_with\_one\_error[i] + " ");**

**}**

**Console.WriteLine("");**

**#endregion**

**#region Новый паритет столбцов**

**Console.WriteLine("Новый паритет столбцов");**

**int[] array\_paritet\_colums\_with\_one\_error = new int[10];**

**if ((array\_message\_with\_one\_error[0] + array\_message\_with\_one\_error[10]) % 2 == 0) { array\_paritet\_colums\_with\_one\_error[0] = 0; }**

**else { array\_paritet\_colums\_with\_one\_error[0] = 1; }**

**if ((array\_message\_with\_one\_error[1] + array\_message\_with\_one\_error[11]) % 2 == 0) { array\_paritet\_colums\_with\_one\_error[1] = 0; }**

**else { array\_paritet\_colums\_with\_one\_error[1] = 1; }**

**if ((array\_message\_with\_one\_error[2] + array\_message\_with\_one\_error[12]) % 2 == 0) { array\_paritet\_colums\_with\_one\_error[2] = 0; }**

**else { array\_paritet\_colums\_with\_one\_error[2] = 1; }**

**if ((array\_message\_with\_one\_error[3] + array\_message\_with\_one\_error[13]) % 2 == 0) { array\_paritet\_colums\_with\_one\_error[3] = 0; }**

**else { array\_paritet\_colums\_with\_one\_error[3] = 1; }**

**if ((array\_message\_with\_one\_error[4] + array\_message\_with\_one\_error[14]) % 2 == 0) { array\_paritet\_colums\_with\_one\_error[4] = 0; }**

**else { array\_paritet\_colums\_with\_one\_error[4] = 1; }**

**if ((array\_message\_with\_one\_error[5] + array\_message\_with\_one\_error[15]) % 2 == 0) { array\_paritet\_colums\_with\_one\_error[5] = 0; }**

**else { array\_paritet\_colums\_with\_one\_error[5] = 1; }**

**if ((array\_message\_with\_one\_error[6] + array\_message\_with\_one\_error[16]) % 2 == 0) { array\_paritet\_colums\_with\_one\_error[6] = 0; }**

**else { array\_paritet\_colums\_with\_one\_error[6] = 1; }**

**if ((array\_message\_with\_one\_error[7] + array\_message\_with\_one\_error[17]) % 2 == 0) { array\_paritet\_colums\_with\_one\_error[7] = 0; }**

**else { array\_paritet\_colums\_with\_one\_error[7] = 1; }**

**if ((array\_message\_with\_one\_error[8] + array\_message\_with\_one\_error[18]) % 2 == 0) { array\_paritet\_colums\_with\_one\_error[8] = 0; }**

**else { array\_paritet\_colums\_with\_one\_error[8] = 1; }**

**if ((array\_message\_with\_one\_error[9] + array\_message\_with\_one\_error[19]) % 2 == 0) { array\_paritet\_colums\_with\_one\_error[9] = 0; }**

**else { array\_paritet\_colums\_with\_one\_error[9] = 1; }**

**for (int i = 0; i < array\_paritet\_colums\_with\_one\_error.Length; i++)**

**{**

**Console.Write(array\_paritet\_colums\_with\_one\_error[i] + " ");**

**}**

**#endregion**

**#region Ошибка в строке**

**Console.WriteLine('\n'+"Поиск ошибки в строке");**

**for (int i = 0; i < array\_paritet\_rows.Length; i++)**

**{**

**Console.Write(array\_paritet\_rows[i] + " ");**

**}**

**Console.WriteLine('\n' + "XOR");**

**for (int i = 0; i < array\_paritet\_rows\_with\_one\_error.Length; i++)**

**{**

**Console.Write(array\_paritet\_rows\_with\_one\_error[i] + " ");**

**}**

**Console.WriteLine('\n' + "---");**

**int[] rows\_paritet\_with\_errors = new int[2];**

**if ((array\_paritet\_rows[0] == 0 && array\_paritet\_rows\_with\_one\_error[0] == 0) || (array\_paritet\_rows[0] == 1 && array\_paritet\_rows\_with\_one\_error[0] == 1))**

**{**

**rows\_paritet\_with\_errors[0] = 0;**

**}**

**else**

**{**

**rows\_paritet\_with\_errors[0] = 1;**

**}**

**if ((array\_paritet\_rows[1] == 0 && array\_paritet\_rows\_with\_one\_error[1] == 0) || (array\_paritet\_rows[1] == 1 && array\_paritet\_rows\_with\_one\_error[1] == 1))**

**{**

**rows\_paritet\_with\_errors[1] = 0;**

**}**

**else**

**{**

**rows\_paritet\_with\_errors[1] = 1;**

**}**

**for (int i = 0; i < rows\_paritet\_with\_errors.Length; i++)**

**{**

**Console.Write(rows\_paritet\_with\_errors[i] + " ");**

**if(rows\_paritet\_with\_errors[i] == 1)**

**{**

**error\_in\_rows = i;**

**}**

**}**

**#endregion**

**#region Ошибка в колонке**

**Console.WriteLine('\n' + "Поиск ошибки в паритете столбцов");**

**for (int i = 0; i < array\_paritet\_colums.Length; i++)**

**{**

**Console.Write(array\_paritet\_colums[i] + " ");**

**}**

**Console.WriteLine('\n' + "XOR");**

**for (int i = 0; i < array\_paritet\_colums\_with\_one\_error.Length; i++)**

**{**

**Console.Write(array\_paritet\_colums\_with\_one\_error[i] + " ");**

**}**

**Console.WriteLine('\n' + "--------------");**

**int[] colums\_paritet\_with\_errors = new int[10];**

**if ((array\_paritet\_colums[0] == 0 && array\_paritet\_colums\_with\_one\_error[0] == 0) || (array\_paritet\_colums[0] == 1 && array\_paritet\_colums\_with\_one\_error[0] == 1))**

**{**

**colums\_paritet\_with\_errors[0] = 0;**

**}**

**else { colums\_paritet\_with\_errors[0] = 1; }**

**if ((array\_paritet\_colums[1] == 0 && array\_paritet\_colums\_with\_one\_error[1] == 0) || (array\_paritet\_colums[1] == 1 && array\_paritet\_colums\_with\_one\_error[1] == 1))**

**{**

**colums\_paritet\_with\_errors[1] = 0;**

**}**

**else { colums\_paritet\_with\_errors[1] = 1; }**

**if ((array\_paritet\_colums[2] == 0 && array\_paritet\_colums\_with\_one\_error[2] == 0) || (array\_paritet\_colums[2] == 1 && array\_paritet\_colums\_with\_one\_error[2] == 1))**

**{**

**colums\_paritet\_with\_errors[2] = 0;**

**}**

**else { colums\_paritet\_with\_errors[2] = 1; }**

**if ((array\_paritet\_colums[3] == 0 && array\_paritet\_colums\_with\_one\_error[3] == 0) || (array\_paritet\_colums[3] == 1 && array\_paritet\_colums\_with\_one\_error[3] == 1))**

**{**

**colums\_paritet\_with\_errors[3] = 0;**

**}**

**else { colums\_paritet\_with\_errors[3] = 1; }**

**if ((array\_paritet\_colums[4] == 0 && array\_paritet\_colums\_with\_one\_error[4] == 0) || (array\_paritet\_colums[4] == 1 && array\_paritet\_colums\_with\_one\_error[4] == 1))**

**{**

**colums\_paritet\_with\_errors[4] = 0;**

**}**

**else { colums\_paritet\_with\_errors[4] = 1; }**

**if ((array\_paritet\_colums[5] == 0 && array\_paritet\_colums\_with\_one\_error[5] == 0) || (array\_paritet\_colums[5] == 1 && array\_paritet\_colums\_with\_one\_error[5] == 1))**

**{**

**colums\_paritet\_with\_errors[5] = 0;**

**}**

**else { colums\_paritet\_with\_errors[5] = 1; }**

**if ((array\_paritet\_colums[6] == 0 && array\_paritet\_colums\_with\_one\_error[6] == 0) || (array\_paritet\_colums[6] == 1 && array\_paritet\_colums\_with\_one\_error[6] == 1))**

**{**

**colums\_paritet\_with\_errors[6] = 0;**

**}**

**else { colums\_paritet\_with\_errors[6] = 1; }**

**if ((array\_paritet\_colums[7] == 0 && array\_paritet\_colums\_with\_one\_error[7] == 0) || (array\_paritet\_colums[7] == 1 && array\_paritet\_colums\_with\_one\_error[7] == 1))**

**{**

**colums\_paritet\_with\_errors[7] = 0;**

**}**

**else { colums\_paritet\_with\_errors[7] = 1; }**

**if ((array\_paritet\_colums[8] == 0 && array\_paritet\_colums\_with\_one\_error[8] == 0) || (array\_paritet\_colums[8] == 1 && array\_paritet\_colums\_with\_one\_error[8] == 1))**

**{**

**colums\_paritet\_with\_errors[8] = 0;**

**}**

**else { colums\_paritet\_with\_errors[8] = 1; }**

**if ((array\_paritet\_colums[9] == 0 && array\_paritet\_colums\_with\_one\_error[9] == 0) || (array\_paritet\_colums[9] == 1 && array\_paritet\_colums\_with\_one\_error[9] == 1))**

**{**

**colums\_paritet\_with\_errors[9] = 0;**

**}**

**else { colums\_paritet\_with\_errors[9] = 1; }**

**for (int i = 0; i < colums\_paritet\_with\_errors.Length; i++)**

**{**

**Console.Write(colums\_paritet\_with\_errors[i] + " ");**

**if(colums\_paritet\_with\_errors[i] == 1)**

**{**

**error\_in\_colums = i;**

**}**

**}**

**#endregion**

**Console.WriteLine('\n' + "Следовательно ошибка в ");**

**Console.WriteLine("Строке " + error\_in\_rows);**

**Console.WriteLine("Колонке " + error\_in\_colums);**

**Console.WriteLine("Исправленная матрица");**

**for (int i = 0; i < array\_message\_with\_one\_error.Length; i++)**

**{**

**if (i == 10)**

**{**

**Console.WriteLine("");**

**}**

**if(i == (error\_in\_colums + error\_in\_rows))**

**{**

**if (error\_in\_rows == 1)**

**{**

**if (array\_message\_with\_one\_error[i + 9] == 1) { array\_message\_with\_one\_error[i + 9] = 0; }**

**else { array\_message\_with\_one\_error[i + 9] = 1; }**

**}**

**else**

**{**

**if (array\_message\_with\_one\_error[i] == 1) { array\_message\_with\_one\_error[i] = 0; }**

**else { array\_message\_with\_one\_error[i] = 1; }**

**}**

**}**

**Console.Write(array\_message\_with\_one\_error[i] + " ");**

**}**

**#endregion**

**}**

**}**

**}**

**Ответы на контрольные вопросы**

1. Охарактеризовать основные параметры итеративного кода.

В итеративных кодах стоится таблица с чётностью по строкам и находятся избыточные биты, используя паритеты.

2. Сравнить основные параметры кодов Хемминга и итеративных кодов.

В итеративных кодах строится проверочная матрица, которая включается в себя биты исходного сообщения, в то время, в кодах Хемминга проверочная матрица связана с информационным сообщение только размерами.

3. Какое максимальное число ошибок может быть обнаружено итеративным кодом? При каком условии?

В итеративных кодах может быть обнаружено 2 ошибки и любое нечётное количество ошибок, если ошибки не находятся в одних строках и столбцах.

4. Определить, какая геометрическая фигура, являющаяся формой для записи символов информационного слова, обеспечивает наименьшую относительную избыточность кодового слова при фиксированном (каком?) k.

При пяти линейно независимых паритетах избыточность сообщения будет меньше, чем при семи и девяти линейно независимых паритетах с К =64.



**Вывод:** многомерныеитеративные коды могут обнаружить и исправить большое количество ошибок, но они сложнее в использовании, нежели одномерные. В целом итеративные коды находят и исправляют больше ошибок, чем код Хемминга.

Учреждение образования

«Белорусский государственный технологический университет»

**Защита информации и надежность информационных систем**

**Лабораторная работа №6**

Избыточное кодирование данных в информационных системах. Циклические коды.

Выполнил:

Студент 3 курса 2 группы ФИТ

Максимова Вера Владимировна

Проверил:

Ржеутская Надежда Викентьевна

**2022 г.**

Цель: приобретение практических навыков кодирования/декодирования двоичных данных при использовании циклических кодов (ЦК).

Задачи:

1. Закрепить теоретические знания по алгебраическому описанию и использованию ЦК для повышения надежности передачи и хранения в памяти компьютера двоичных данных, для контроля интегральности файлов информации.

2. Разработать приложение для кодирования/декодирования двоичной информации циклическим кодом.

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

**Теоретические сведения**

Циклические коды − это семейство помехоустойчивых кодов, одной из разновидностей которых являются коды Хемминга.

Формирование разрешенных кодовых комбинаций ЦК Bj(X) основано на предварительном выборе порождающего (генераторного или образующего) полинома G(X), который обладает важным отличительным признаком: все комбинации Bj(X) делятся на порождающий полином G(X) без остатка:

Bj(X) / G(X) = Aj(X), (6.2)

здесь Bj(X) = Xn – кодовое слово; Aj(X) = Xk – информационное слово.

Степень порождающего полинома определяет число проверочных символов: r = n – k. Из этого свойства следует простой способ формирования разрешeнных кодовых слов ЦК − умножение информационного слова A(X) на порождающий полином G(X):

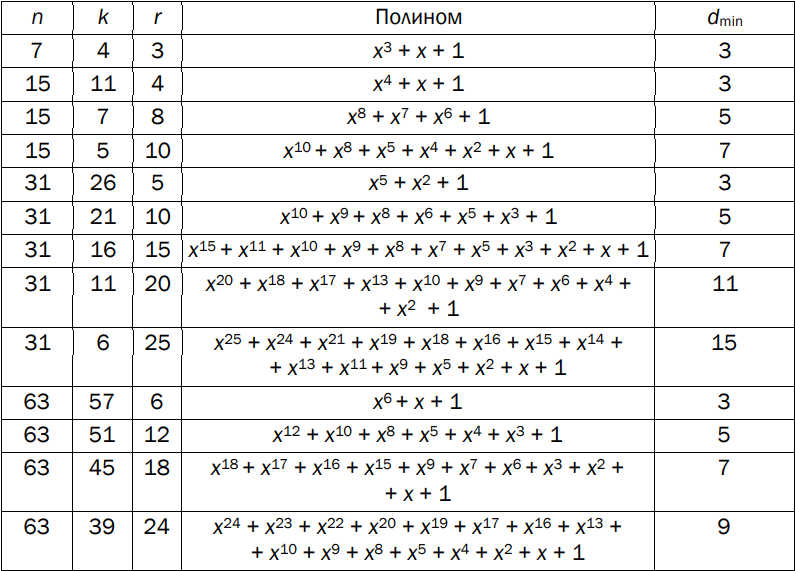
Bj(X) = Aj(X) · G(X). (6.3)

Порождающими могут быть только такие полиномы, которые являются делителями двучлена (бинома) Хz + 1:

(Хz + 1) / G(X) = H(X) (6.4)

при нулевом остатке: R(X) = 0.

Параметры некоторых циклических кодов

****

Декодирование принятого сообщения по синдрому. Основная операция: принятое кодовое слово (Yn) нужно поделить на порождающий полином, который использовался при кодировании.

Для исправления ошибки нужно определить вектор (полином) ошибки Еn.

Принимаем, что после передачи по каналу с помехами кодовое слово можно записать в виде:

Yn = Xn + Еn или Xn = Yn + Еn. (6.5)

Деление принятого кодового слова на G(X) формально запишем в следующем виде:

Yn / G(X) = U, Sr, (6.6)

где Sr − остаток от деления (Yn) / (G(X)), или синдром.

**Код практического задания:**

using System;

namespace lab6

{

class Program

{

static void Main(string[] args)

{

string Xk = "11001";

string Xr = "110111";

int k = Xk.Length;

int n = 31;

int r = n - k;

int error;

int[] masXk = new int[k];

StrInMas(masXk, Xk);

int[] masXr = new int[Xr.Length];

StrInMas(masXr, Xr);

Console.WriteLine("Входная строка: " + Xk);

Console.WriteLine("Порождающий полином: " + Xr);

Console.WriteLine("k = {0}, r = {1}, n = {2}", k, r, n);

int[,] generationMatrix = new int[k, n];

CreateGenerationMatrix(generationMatrix, masXr, k, n);

Console.WriteLine("\nПорождающая матрица");

OutMatrix(generationMatrix, k, n);

CreateCanonicalMatrix(generationMatrix, k, n);

Console.WriteLine("\nКаноническая матрица");

OutMatrix(generationMatrix, k, n);

int[,] checkMatrix = new int[n, r];

CreateCheckMatrix(checkMatrix, generationMatrix, k, n);

Console.WriteLine("\nПроверочная матрица");

OutMatrix(checkMatrix, n, r);

//6.2

int[] masXn = new int[n];

Shift(masXn, masXk, r);

//2.

Console.WriteLine("\nДеление");

SearchResidue(masXn, masXr);

Console.WriteLine("Остаток:");

OutMass(masXn);

Console.WriteLine("Итоговая строка:");

Shift(masXn, masXk, r);

OutMass(masXn);

try

{

Console.WriteLine("Введите место первой ошибки");

error = Convert.ToInt32(Console.ReadLine()) - 1;

if (masXn[error] == 1) masXn[error] = 0;

else masXn[error] = 1;

}

catch { }

Console.WriteLine("Ошибочная строка:");

OutMass(masXn);

SearchError(masXn, masXr, checkMatrix, r);

}

public static int[] SearchError(int[] masXn, int[] masXr, int[,] checkMatrix, int r)

{

int n = masXn.Length;

int k = n - r;

int[] masXnSecond = new int[n];

for (int i = 0; i < n; i++)

{

masXnSecond[i] = masXn[i];

}

Console.WriteLine("\nДеление");

SearchResidue(masXnSecond, masXr);

Console.WriteLine("\nОстаток:");

OutMass(masXnSecond);

for (int i = 0; i < n; i++)

{

int coincidence = 0;

for (int j = 0; j < r; j++)

{

if (checkMatrix[i, j] == masXnSecond[k + j])

{

coincidence++;

}

}

if (coincidence == r)

{

masXn[i] = (masXn[i] + 1) % 2;

break;

}

}

Console.WriteLine("\nИсправленная строка:");

OutMass(masXn);

return masXn;

}

public static int[] SearchResidue(int[] masXn, int[] masXr)

{

int end = masXn.Length - masXr.Length + 1;

for (int i = 0; i < end; i++)

{

if (masXn[i] == 1)

{

AddingMasMod2(masXn, masXr, i);

OutMass(masXn);

}

}

return masXn;

}

//Сложение массивов по модулю 2 с опр. позиции

public static int[] AddingMasMod2(int[] mas1, int[] mas2, int pos)

{

int end = pos + mas2.Length;

for (int i = pos; i < end; i++)

{

mas1[i] = (mas1[i] + mas2[i - pos]) % 2;

}

return mas1;

}

//Смещение на массива r

public static int[] Shift(int[] shiftMas, int[] mas, int r)

{

for (int i = 0; i < mas.Length; i++)

{

shiftMas[i] = mas[i];

}

return shiftMas;

}

//Преобразование сторки в массив

public static int[] StrInMas(int[] mas, string str)

{

for (int i = 0; i < str.Length; i++)

{

if (str[i] == 49)

mas[i] = 1;

else mas[i] = 0;

}

return mas;

}

//Создание Порождающей матрицы

static int[,] CreateGenerationMatrix(int[,] generationMatrix, int[] mas, int k, int n)

{

//Заполняем первую строку в проверочной матрице

for (int i = 0; i < n; i++)

{

if (i < mas.Length)

{

generationMatrix[0, i] = mas[i];

}

else

{

generationMatrix[0, i] = 0;

}

}

//Сдвигаем каждую строки вправо от предыдущей

for (int i = 1; i < k; i++)

{

for (int j = 0; j < n - 1; j++)

{

generationMatrix[i, j + 1] = generationMatrix[i - 1, j];

}

generationMatrix[i, 0] = generationMatrix[i - 1, n - 1];

}

return generationMatrix;

}

//Приведение порождающей матрицы к каноническому виду

static int[,] CreateCanonicalMatrix(int[,] generationMatrix, int k, int n)

{

//Перебираем строки для преведению к каноническому виду

for (int i = 0; i < k; i++)

{

int i2 = i + 1;

//Перебираем элементы строки, но только до k-элемента

for (int j = i + 1; j < k; j++)

{

//если мы нашли единицу в строке, то...

if (generationMatrix[i, j] == 1)

{

//перебираем этот столбец, пока не найдем единицу

for (; i2 < k; i2++)

{

bool repeat = false;

//Если нашли, то складываем обе строки

if (generationMatrix[i2, j] == 1)

{

for (int j2 = j - 1; j2 > 0; j2--)

{

//Проверяем, есть ли до этой 1 еще 1, если есть то эту строку пропускаем

if (generationMatrix[i2, j2] == 1)

{

repeat = true;

}

}

if (repeat)

continue;

Console.WriteLine(i + " " + i2);

AddingLinesMatrixMod2(generationMatrix, i, i2, n);

i2++;

break;

}

}

}

}

}

return generationMatrix;

}

//Преобразование канонической матрицы в проверочную

static int[,] CreateCheckMatrix(int[,] checkMatrix, int[,] generationMatrix, int k, int n)

{

int r = n - k;

for (int i = 0; i < k; i++)

{

for (int j = 0; j < r; j++)

{

checkMatrix[i, j] = generationMatrix[i, k + j];

}

}

for (int i = k; i < n; i++)

{

for (int j = 0; j < r; j++)

{

if (j == i - k)

{

checkMatrix[i, j] = 1;

}

else

{

checkMatrix[i, j] = 0;

}

}

}

return checkMatrix;

}

//Сложение строк матрицы

public static int[,] AddingLinesMatrixMod2(int[,] matrix, int str1, int str2, int lengthString)

{

//Console.WriteLine(str1 + " и " + str2);

for (int i = 0; i < lengthString; i++)

{

matrix[str1, i] = (matrix[str1, i] + matrix[str2, i]) % 2;

}

return matrix;

}

//вывод матрицы

public static void OutMatrix(int[,] matrix, int k, int n)

{

for (int i = 0; i < k; i++)

{

for (int j = 0; j < n; j++)

{

Console.Write(matrix[i, j]);

//if (j + 1 == k) Console.Write("|");

}

Console.WriteLine();

}

}

//вывод одномерного массива

public static void OutMass(int[] mas)

{

for (int i = 0; i < mas.Length; i++)

{

//if (i == k) Console.Write("|");

Console.Write(mas[i]);

}

Console.WriteLine("");

}

}

}

**Ответы на контрольные вопросы**

1. Охарактеризовать основные параметры циклических кодов.

Для циклических кодов строится порождающая матрица, которая приводится к каноническому виду. Сообщение приводится к полиномиальному виду для определения избыточных битов. Используется деление и умножение в полиномиальном виде.

2. Сравнить основные параметры кодов Хемминга, итеративных и циклических кодов.

В итеративных кодах, вычисляются паритеты и не строится отдельная проверочная матрица. В циклических кодах порождающая матрица стоится на основе выбранного порождающий полином и используются деление и умножения в полиномиальном виде.

3. Пары десятичных чисел: 14, 11; 19, 15; 29, 13; 35, 45 преобразовать в двоичные числа и представить их в виде полиномов. Выполнить арифметические операции над двоичными числами (по модулю 2) и полиномами.

14 = 1110 = x3 + x2 +x

11 = 1011 = x3 + x + 1

1110+1011=0101

х3 + x2 + x + x3 + x + 1 = x2 + 1

19 = 10011 = x4 + x +1

15 = 01111 = x3 + x2 + x + 1

10011 +01111 =11100

x4 + x + 1 + x3 + x2 + x + 1 = x4 + x3 + x2

29 = 11100 = x4 + x3 +x2

13 = 01101 = x4 + x2 + 1

11100 +01101 =0101

x4 + x2 + x + x4 + x + 1 = x2 + 1

35 = 1110 = x4 + x2 +x

45 = 1011 = x4 + x + 1

1110+1011=0101

x4 + x2 + x + x4 + x + 1 = x2 + 1

4. Если кодовое слово ЦК имеет следующий вид: 11011010, то будут ли разрешенными кодовыми комбинациями следующие: 01101101, 10110101, 11011011?

01101101 – да.

10110101 – да.

11011011 – нет.

5. Запишите в полиномиальном виде кодовое слово ЦК: 11011011, 1010101010, 0000000, 0000000001.

11011011 = x6 + x6 + x4 + x3 + x + 1

1010101010 = x9 + x7 + x5 + x3 + x

0000000 = nothing

0000000001 = 1

6. Чему равен результат операции по модулю два над полиномами x6 + x3 + x2 и x4 + х2 + x, если такой операцией будет: сложение, вычитание, деление, умножение? Записать результат операции в двоичной форме.

x6 + x3 + x2 + x4 + х2 + x = x6 + x4 + x3 + x

(x6 + x3 + x2) \* (x4 + х2 + x) = x10 + x8 + x7 + x7 + x5 + x4 + x6 + x4 + x3 = x10 + x8 + x6 + x5 + x3

x6 + x3 + x2 | x4 + х2 + x

x6 + х4 + x3 | x2 + 1

х4 + x2

x4 + х2 + x

x – остаток

x6 + x3 + x2 = 1001100

x4 + х2 + x = 0010110

1001100+0010110 = 1011010

1001100\*0010110 = 1011000+10110000+10110000000 = 10101101000

+1001100|10110

10110 |101

=0010100

+0010110

=0000010- Остаток

7. Определить число и вычислить значения проверочных символов ЦК, если код задается полиномом x3 + х2 + 1, а информационное слово имеет вид: 1010; 1100.

1011100

1110010

0111001

1010.001

1100.101

8. Составить проверочную матрицу канонического вида, если порождающий полином имеет вид x3 + х2 + 1.

1101

1101000 1=1+2+3 = 1000110 1011100

0110100 2=2+3+4 = 0100011 1110010

0011010 3=3+4 = 0010111 0111001

0001101 4=4 = 0001101

Вывод: в ходе лабораторной работы были приобретены практические навыки кодирования/декодирования двоичных данных при использовании циклического кода, было определено, что, если при передаче сообщения возникают две ошибки, то эти 2 ошибки нельзя исправить стандартным методом.

Учреждение образования

«Белорусский государственный технологический университет»

**Защита информации и надежность информационных систем**

**Лабораторная работа №7**

ПЕРЕМЕЖЕНИЕ/ДЕПЕРЕМЕЖЕНИЕ ДАННЫХ В ИНФОРМАЦИОННО-ВЫЧИСЛИТЕЛЬНЫХ СИСТЕМАХ

Выполнил:

Студент 3 курса 2 группы ФИТ

Максимова Вера Владимировна

Проверил:

Ржеутская Надежда Викентьевна

**2022 г.**

Цель: приобретение практических навыков использования методов перемежения/деперемежения двоичных данных в информационных системах.

Задачи:

1. Закрепить теоретические знания по алгебраическому описанию и использованию методов перемежения/деперемежения двоичных данных в информационных системах.

2. Разработать приложение для реализации метода перемежения/деперемежения символов в сообщениях на основе двоичного алфавита.

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

**Теоретические сведения**

Идея перемежения/деперемежения состоит в следующем. Если биты каждого кодового слова Хn передаются не в обычной последовательности, а через интервалы, превышающие ожидаемую длину пакета ошибок (в промежутки между битами одного слова вставляются биты других кодовых слов), то при возникновении такого типа ошибки обратная перемежению операция – деперемежение – разнесет («размажет») группу ошибок по всей совокупности кодовых слов, составляющих данное сообщение.

Рассмотрим процесс передачи информации с использованием кода Хемминга и блокового перемежителя. Информационный поток (в виде отдельных информационных слов длиной 4 бита) на входе кодера Хемминга (используется код (7, 4)) имеет вид, как показано на рис. 7.1.

****

Рис. 7.1 Структура информационного потока на входе кодера

На выходе кодера сообщение будет иметь вид, представленный на рис. 7.2.

****

Рис. 7.2. Последовательность символов сообщения на выходе кодера

Предположим, что в процессе передачи информации по каналу возник пакет ошибок (выделено черным) длиной 7 битов (рис. 7.3).

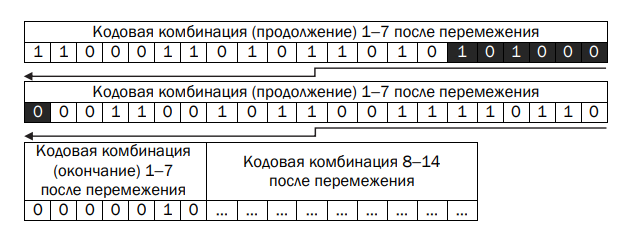


Рис. 7.3. Передаваемое сообщение, содержащее группу ошибок



Рис. 7.4. Ошибки разнесены по всему сообщению

После деперемежения пакет ошибок преобразован в одиночные (формально – независимые) ошибки кратности 1 для каждой из кодовых комбинаций кода Хемминга. После исправления одиночных ошибок с большой вероятностью получится исходных поток.



Рис. 7.5. Информационный поток на выходе декодера кода Хемминга

**Код практического задания**

namespace Lab7

{

class Program

{

static void Main(string[] args)

{

Console.WriteLine("<---Передача информации с использованием кода Хемминга и блокового перемежителя--->");

Console.WriteLine($"\nПоток поделен на информационные слова: \nдлина потока - 16\nдлина информационного слова - 4\n");

int flowLenght = 16;

int k = (int)(Math.Sqrt(flowLenght));

int r = HemmingCode.GetNumOfVerifChar(k);

int n = k + r;

int codeCombinationLenght = flowLenght + (r \* k);

int[] informationFlow = new int[16] { 1, 0, 0, 1, 1, 1, 0, 0, 0, 0, 1, 0, 0, 1, 0, 1};

int[] infoFlowAfterChanges = new int[flowLenght];

int[] codeCombination = new int[flowLenght + (r \* k)];

int[,] HemmingsVerificMatrix = new int[n, r];

Console.WriteLine("Структура информационного потока на входе кодера: ");

MatrixOperations.PrintArray(informationFlow);

HemmingsVerificMatrix = HemmingCode.GetVerifMatrix(k);

//MatrixOperations.PrintVerifMatrix(HemmingsVerificMatrix, n, r);

Interleaver.AddCheckBits(informationFlow, codeCombination, HemmingsVerificMatrix);

Console.WriteLine("\n\nПоследовательность символов сообщения на выходе кодера (после добавления проверочных битов): ");

MatrixOperations.PrintArray(codeCombination);

Interleaver.ExecInterleaving(codeCombination, k);

Console.WriteLine("\nЗакодированное сообщение после перемежения: ");

MatrixOperations.PrintArray(codeCombination);

try

{

Console.WriteLine("\n\n<---Предположим, что в процессе передачи информации по каналу возник пакет ошибок--->");

Console.WriteLine("Укажите позицию начала пакета ошибки: ");

int error = Convert.ToInt32(Console.ReadLine());

Console.WriteLine("Укажите длину пакета ошибок (3, 5, 7): ");

int errorLenght = Convert.ToInt32(Console.ReadLine());

if (errorLenght == 3 || errorLenght == 5 || errorLenght == 7)

{

for (int i = error; i < (error + errorLenght); i++)

{

codeCombination[i] = (codeCombination[i] + 1) % 2;

}

Console.WriteLine("\nПередаваемое сообщение, содержащее группу ошибок: ");

MatrixOperations.PrintArray(codeCombination);

Interleaver.ExecDeInterleaving(codeCombination, k);

Console.WriteLine("\nСобщение с разнесёнными ошибками: ");

MatrixOperations.PrintArray(codeCombination);

Interleaver.LineByLineSearchError(codeCombination, HemmingsVerificMatrix, k);

Console.WriteLine("\n\nИнформационный поток на выходе декодера кода Хемминга (после исправления ошибок): ");

MatrixOperations.PrintArray(codeCombination);

Interleaver.RemoveCheckBits(infoFlowAfterChanges, codeCombination, HemmingsVerificMatrix);

Console.WriteLine("\n\nИнформационный поток после удаления проверочных бит: ");

MatrixOperations.PrintArray(infoFlowAfterChanges);

Console.WriteLine("\n\nИсходный информационный поток: ");

MatrixOperations.PrintArray(informationFlow);

Console.WriteLine();

}

else

{

Console.WriteLine("!Для выполнения задания по варианту необходимо ввести длину пакета ошибок равную 3, 5 или 7!");

}

}

catch

{

Console.WriteLine("!Вероятно допущена ошибка при указании позиции начала пакета ошибок, либо длины пакета!");

}

}

}

}

**Ответы на контрольные вопросы**

1. Пояснить назначение и особенности использования технологии перемежения/деперемежения данных в ИС.

Большое сообщение разбивается на блоки с небольшим количеством символов и кодируются, после символы перемешиваются и сообщение передаётся. Так как при передаче сообщения чаще всего ошибки идут блоками (друг за другом), то при деперемежении ошибки с большой вероятностью будут в разных блоках, которые можно будет исправить.

2. Что такое группирующиеся ошибки и с чем, по Вашему мнению, связано их появление в каналах передачи данных, в полупроводниковой памяти, на магнитных носителях?

При передаче сообщений ошибки чаще всего будут возникать друг за другом так как ошибки возникают при наличии каких-то непостоянных помех.

На дисках это могут быть царапины, при передаче по каналу это может быть другой канал, по которому передается сообщнице.

3. Что такое глубина перемежения и как она влияет на эффективность перемежения/деперемежения данных?

Глубина перемежения – расстояние между позициями одного и того же символа до и после перемежения. Чем больше глубина перемежения, тем больший блок ошибок мы можем исправить. Но чем больше глубина перемежения, тем сложнее аппаратно-программная реализация оборудования и больше задержка сигнала.

4. Пояснить принцип работы блокового метода перемежения.

Сообщение разбивается на несколько частей одинаковой длины, эти части шифруются и перемешиваются. После отправки сообщения, порядок сообщения возвращается в исходное положение и начинается поиск ошибок в каждой части.

**Вывод**: технология перемежения хорошо работает за счет того, что чаще всего, при возникновении ошибок, они будут находится рядом, стоя друг за другом. Но при возникновении нескольких блоков ошибок в разных местах определить ошибки становится сложнее.

Учреждение образования

«Белорусский государственный технологический университет»

**Защита информации и надежность информационных систем**

**Лабораторная работа №8**

СЖАТИЕ/РАСПАКОВКА ДАННЫХ

МЕТОДОМ БАРРОУЗА – УИЛЕРА

Выполнил:

Студент 3 курса 2 группы ФИТ

Максимова Вера Владимировна

Проверил:

Ржеутская Надежда Викентьевна

**2022 г.**

Цель: приобретение практических навыков использования метода Барроуза − Уилера для сжатия/распаковки данных.

Задачи:

1. Закрепить теоретические знания по алгебраическому описанию и использованию методов сжатия/распаковки (архивации/ разархивации) данных на основе метода Барроуза − Уилера (BurrowsWheeler transform, BWT).

2. Разработать приложение для реализации метода Барроуза − Уилера.

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

**Теоретические сведения**

Сжатие информации является одним из способов ее кодирования.

Все алгоритмы сжатия преобразуют входной поток данных, минимальной единицей которых является бит, а максимальной – байт или несколько байт. Основными техническими характеристиками процессов сжатия и результатов их работы являются:

• степень сжатия (англ. compress rating), или отношение R (англ. ratio) объемов исходного (до сжатия, Vдс) и результирующего (после сжатия, Vпс) потоков данных (сообщений);

• скорость сжатия − время, затрачиваемое на сжатие некоторого объема информации входного потока до получения из него эквивалентного выходного потока;

• качество сжатия − величина, показывающая, насколько сильно сжат выходной поток при помощи применения к нему повторного сжатия по этому же или иному алгоритму.

Степень сжатия R обычно оценивается следующим образом:

R1 = (Vпс / Vдс) · 100%,

R2 = (Vдс − Vпс) / Vдс= (1 − R1) · 100%.

**Код практического задания:**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace Lab8

{

public partial class Form1 : Form

{

private string[] arrayW;

private string[] arrayB;

private string m;

private string mm;

private int z;

private int now = 0;

private int size = 0;

public Form1()

{

InitializeComponent();

buttonLast.Visible = false;

}

private void Form1\_Load(object sender, EventArgs e)

{

}

private void buttonGen\_Click(object sender, EventArgs e)

{

if(textBoxInput.Text.Length > 1)

{

size = textBoxInput.Text.Length;

arrayW = new string[size];

arrayB = new string[size];

mm = textBoxInput.Text;

string cur = textBoxInput.Text;

for(int i = 0; i < size; i++)

{

arrayW[i] = cur;

char tempLetter = cur[0];

cur = cur.Substring(1, size - 1) + tempLetter;

}

richTextBoxW1.Text = String.Join("\n", arrayW);

Array.Sort(arrayW);

richTextBoxW2.Text = String.Join("\n", arrayW);

z = Array.IndexOf(arrayW, textBoxInput.Text);

m = "";

for(int i = 0; i < size; i++)

{

m = m + arrayW[i][size - 1];

}

textBoxM.Text = m;

textBoxK.Text = (z + 1).ToString();

now = 0;

}

}

private void buttonBack\_Click(object sender, EventArgs e)

{

if (now >= size)

{

richTextBoxB1.Text = String.Join("\n", arrayB);

reColorRTB();

return;

}

for (int i = 0; i < size; i++)

{

arrayB[i] = m[i] + arrayB[i];

}

richTextBoxB1.Text = String.Join("\n", arrayB);

Array.Sort(arrayB);

now++;

textBoxNow.Text = now.ToString();

}

private void buttonLast\_Click(object sender, EventArgs e)

{

for (; now < size; now++)

{

for (int i = 0; i < size; i++)

{

arrayB[i] = m[i] + arrayB[i];

}

Array.Sort(arrayB);

}

richTextBoxB1.Text = String.Join("\n", arrayB);

textBoxNow.Text = now.ToString();

reColorRTB();

}

private void reColorRTB()

{

int pos = richTextBoxB1.Text.IndexOf(mm);

richTextBoxB1.Select(pos, mm.Length);

richTextBoxB1.SelectionFont = new Font(richTextBoxB1.Font, FontStyle.Bold);

}

private void textBoxK\_TextChanged(object sender, EventArgs e)

{

}

}

}

**Ответы на контрольные вопросы**

1. Сформулировать цели применения методов сжатия и архивирования данных.

Основная цель сжатия – обеспечить более компактное представление данных, вырабатываемых источником, т. е. уменьшить физический объем сообщений, генерируемых источником, и сократить время его по каналам связи.

2. Охарактеризовать основные технические характеристики процессов сжатия/распаковки и результатов.

степень сжатия (англ. compress rating), или отношение R (англ. ratio) объемов исходного (до сжатия, Vдс) и результирующего (после сжатия, Vпс) потоков данных (сообщений);

• скорость сжатия − время, затрачиваемое на сжатие некоторого объема информации входного потока до получения из него эквивалентного выходного потока;

• качество сжатия − величина, показывающая, насколько сильно сжат выходной поток при помощи применения к нему повторного сжатия по этому же или иному алгоритму.

3. Как можно рассчитать степень сжатия файла?

Степень сжатия R обычно оценивается следующим образом:

R1 = (Vпс / Vдс) · 100%,

R2 = (Vдс − Vпс) / Vдс= (1 − R1) · 100%.

Vпс - объемов результирующего сообщения

Vдс - объемов исходного сообщения

4. В каких случаях и почему применяется сжатие без потерь, а в каких – с потерей информации?

Методы и алгоритмы сжатия с потерей информации применяют обычно для решения так называемых потребительских задач. Это значит, например, что если фотография передается для просмотра, а музыка для воспроизведения, то подобные алгоритмы применять можно. Если же они передаются для дальнейшей обработки, например, для редактирования, то никакая потеря информации в исходном файле недопустима

5. В чем сущность символ-ориентированных методов сжатия? Какие известные Вам методы относятся к этому классу?

Сущность символ-ориентированных (словарных) методов состоит в последова­тельном анализе сжимаемой информации с целью поиска повторяющихся или проана­лизированных ранее в данном документе последовательностей и замене таких последовательностей на более короткие.

BWT, LZW, LZO.

6. Составить алгоритмы сжатия/распаковки данных методом Барроуза − Уилера.

Сжатия

1) выделяется блок данных (строка длиной k символов некоторого алфавита мощностью N), который обозначим символом М;

2) составляется таблица W1 размером k×k всех циклических сдвигов входной строки M: W1 = (M);

3) производится лексикографическая (в алфавитном порядке) сортировка строк таблицы W1, в результате чего получается таблица W2 того же размера;

4) в качестве выходной строки (обозначим ее BWT(М), z) выбирается последний столбец (Мk) таблицы W2 преобразования и номер строки z, совпадающей с исходной строкой М

Распаковки

1) в крайний справа пустой столбец матрицы записывается последовательность символов Мk;

2) производится лексикографическая сортировка столбцов заполненной части воссоздаваемой матрицы.

7. Что поменяется, если процедуру формирования матрицы W1 строить на основе циклических сдвигов вправо?

При декодировании надо будет вставлять строки не справа налево, а слева направо.

8. В каких известных архиваторах используется метод Барроуза − Уилера?

bzip2

**Вывод**: методом Барроуза – Уилера используется во многих архиваторах совместно с другими методами сжатия, отдельно же этот метод почти не используется.

Учреждение образования

«Белорусский государственный технологический университет»

**Защита информации и надежность информационных систем**

**Лабораторная работа №9**

СЖАТИЕ/РАСПАКОВКА ДАННЫХ НА ОСНОВЕ СТАТИСТИЧЕСКИХ МЕТОДОВ

Выполнил:

Студент 3 курса 2 группы ФИТ

Максимова Вера Владимировна

Проверила:

Ржеутская Надежда Викентьевна

**2022 г.**

**Цель**: приобретение практических навыков использования статистических методов Шеннона − Фано и Хаффмана (ShannonFano and Huffman coding) для сжатия/распаковки данных.

**Задачи**:

1. Закрепить теоретические знания по алгебраическому описанию и использованию методов сжатия/распаковки (архивации/разархивации) данных на основе методов Шеннона − Фано и Хаффмана.

2. Разработать приложение для реализации методов Шеннона − Фано и Хаффмана.

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

**Теоретические сведения**

Код Шеннона – Фано не является оптимальным (обеспечивает минимальную избыточность) в общем смысле, хотя и дает оптимальные результаты при некоторых распределениях вероятностей. Для одного и того же распределения вероятностей можно построить, вообще говоря, несколько кодов Шеннона – Фано, и все они могут дать различные результаты. Итак, необходимо выполнить следующие действия:

1) подсчитать вероятностные параметры символов алфавита А = {ai} (реализуется статическая версия алгоритма);

2) отсортировать – обычно в порядке убывания (невозрастания, т. е. могут иметь место повторяющиеся значения) вероятностей р(аi); р(аi) – вероятность появления в сжимаемом сообщении на произвольной позиции символа аi алфавита, т. е. создать таблицу символов алфавита, на основе которого генерируется сжимаемое сообщение;

3) каждому символу отсортированного множества поставить в соответствие бинарный код, для чего это множество (таблица) символов делится на две группы таким образом, чтобы каждая из групп имела приблизительно одинаковую суммарную частоту (вероятность).

Далее рассмотрим процесс создания таблицы кодов. Отсортируем таблицу символов в порядке убывания вероятностей. Разделим ее на две части (два подмножества. Символам верхней части общей таблицы определим старший символ кода (1), нижней части – 0.

Далее в качестве исходного рассматриваем каждое из двух подмножеств (на текущем шаге). Выполняем рекурсивно одну и ту же процедуру.

Далее переходим к кодированию символов после первого деления исходной таблицы. Выполнив стандартные операции, получим таблицу бинарных кодов.

Замечаем, что выполнено основное требование: все кодовые комбинации разные. И соблюдено «требование префикса»: ни одна из кодовых комбинаций меньшей длины не является началом кодовой комбинации большей длины.

Алгоритм прямого преобразования: необходимо выполнить одну операцию: заменить символы входного сообщения соответствующими бинарными кодами.

Алгоритм обратного преобразования: на входе – сообщение в виде бинарной последовательности.

Шаг 1.

Анализируются lmin начальных бинарных символов: осуществляется поиск в таблице соответствующего совпадения. Если такое будет найдено, то на выходе будет символ исходного алфавита с совпадающим кодом. После этого процедура повторяется, т. е. анализируются очередные lmin символов. Если не найдено в таблице совпадения, переходим к шагу 2.

Шаг 2. Длина анализируемой последовательности увеличивается на 1 бит: lmin + 1. Осуществляется поиск совпадающей бинарной комбинации такой же длины в таблице. Если такая комбинация существует, на выходе распаковщика формируется соответствующий символ исходного алфавита, если нет – длина анализируемой последовательности увеличивается еще на один бит, и т. д.

**Код практического задания:**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.IO;

namespace lab9

{

public class ShannonFanoSymbol

{

public char symbol;

public int count;

public double viorite;

public string code;

public ShannonFanoSymbol(char sim, int count, double vior, string code)

{

this.viorite = vior;

this.symbol = sim;

this.count = count;

this.code = code;

}

public static List<ShannonFanoSymbol> AddSymbols(List<ShannonFanoSymbol> symbols, string line)

{

foreach (var character in line)

{

if (symbols.Find(x => x.symbol == character) == null)

{

symbols.Add(new ShannonFanoSymbol(character, 1, 0.0, ""));

}

else

{

symbols.Where(x => x.symbol == character).ToList().ForEach(x => x.count++);

}

}

return symbols;

}

public static void Show(List<ShannonFanoSymbol> symbols)

{

foreach (var symbol in symbols)

{

Console.Write("Symbol: {0} Amount:: {1} ", symbol.symbol, symbol.count);

if (symbol.viorite != 0) {

Console.Write("P: {0}", symbol.viorite);

}

if (symbol.code != "") {

Console.Write(" Code: {0}", symbol.code);

}

Console.WriteLine();

}

Console.WriteLine();

}

}

class Program

{

static void Main(string[] args)

{

List<ShannonFanoSymbol> symbols = new List<ShannonFanoSymbol>();

using (StreamReader stream = new StreamReader(@"..\..\text\latin.txt", Encoding.Default))

{

string messagef;

while ((messagef = stream.ReadLine()) != null)

{

symbols = ShannonFanoSymbol.AddSymbols(symbols, messagef);

}

}

Console.WriteLine();

Console.WriteLine("Analysis of statistic data:");

Console.WriteLine("Table of symbols: ");

Console.WriteLine();

ShannonFanoSymbol.Show(symbols);

double symbolssum = symbols.Sum(x => x.count);

Console.WriteLine("Sum of latin symbol in text: " + symbolssum);

for (int i = 0; i < symbols.Count; i++)

{

symbols[i].viorite = symbols[i].count / symbolssum;

}

Console.WriteLine("Sum of table's symbols: " + (symbols.Sum(x => x.viorite)));

Console.WriteLine();

symbols = symbols.OrderByDescending(x => x.viorite).ToList();

ShannonFanoSymbol.Show(symbols);

Console.WriteLine();

Console.WriteLine("Table of code for each symbol: " );

Console.WriteLine();

symbols = AddCodes(symbols);

foreach (var symbol in symbols)

{

symbol.code = symbol.code.Remove(symbol.code.Length - 1, 1);

}

ShannonFanoSymbol.Show(symbols);

string blockofFIO = "Vera Maksimava";

string decodingOfFIO = "";

foreach (var charFIO in blockofFIO)

{

decodingOfFIO += (symbols.Where(x => x.symbol == charFIO).FirstOrDefault()).code;

}

Console.WriteLine("Inputed text: ");

Console.WriteLine(blockofFIO);

Console.WriteLine("Encoded: ");

Console.WriteLine(decodingOfFIO);

Console.WriteLine("Amount of symbols in ASCII: " + blockofFIO.Count() \* 8);

Console.WriteLine("Amount of symbols in encoded table: " + decodingOfFIO.Count());

Console.WriteLine("\nDecoded: ");

string Encoded = "";

string FIOdecoded = "";

for (int i = 0; i < decodingOfFIO.Count(); i++)

{

Encoded += decodingOfFIO[i];

if (symbols.Find(x => x.code == Encoded) != null)

{

FIOdecoded += symbols.Find(x => x.code == Encoded).symbol;

Encoded = "";

}

}

Console.WriteLine(FIOdecoded);

Console.WriteLine();

Console.WriteLine("Dynamic, based on analysis of inputedd symbols: ");

Console.WriteLine();

symbols.Clear();

string message = "Maksimava";

symbols = ShannonFanoSymbol.AddSymbols(symbols, message);

ShannonFanoSymbol.Show(symbols);

symbolssum = symbols.Sum(x => x.count);

Console.WriteLine("Sum of symbols in latin: " + symbolssum);

for (int i = 0; i < symbols.Count; i++)

{

symbols[i].viorite = symbols[i].count / symbolssum;

}

Console.WriteLine("Sum of P for each symbols: " + (symbols.Sum(x => x.viorite)));

Console.WriteLine();

symbols = symbols.OrderByDescending(x => x.viorite).ToList();

ShannonFanoSymbol.Show(symbols);

Console.WriteLine();

Console.WriteLine("Code for each symbol: ");

Console.WriteLine();

symbols = AddCodes(symbols);

foreach (var symbol in symbols)

{

symbol.code = symbol.code.Remove(symbol.code.Length - 1, 1);

}

ShannonFanoSymbol.Show(symbols);

blockofFIO = "Maksimava";

decodingOfFIO = "";

foreach (var charFIO in blockofFIO)

{

decodingOfFIO += (symbols.Where(x => x.symbol == charFIO).First()).code;

}

Console.WriteLine("Inputed text: ");

Console.WriteLine(blockofFIO);

Console.WriteLine("Encoded: ");

Console.WriteLine(decodingOfFIO);

Console.WriteLine("Amount of symbols in ASCII: " + blockofFIO.Count() \* 8);

Console.WriteLine("Amount of symbols in encoded table: " + decodingOfFIO.Count());

Console.WriteLine("\nDecoded:");

Encoded = "";

FIOdecoded = "";

for (int i = 0; i < decodingOfFIO.Count(); i++)

{

Encoded += decodingOfFIO[i];

if (symbols.Find(x => x.code == Encoded) != null)

{

FIOdecoded += symbols.Find(x => x.code == Encoded).symbol;

Encoded = "";

}

}

Console.WriteLine(FIOdecoded);

}

public static List<ShannonFanoSymbol> AddCodes(List<ShannonFanoSymbol> symbols)

{

int counter = 0;

double probability = 0.0;

List<ShannonFanoSymbol> first = new List<ShannonFanoSymbol>();

List<ShannonFanoSymbol> second = new List<ShannonFanoSymbol>();

while (probability < (symbols.Sum(x => x.viorite) / 2))

{

probability += symbols[counter].viorite;

counter++;

}

for (int i = 0; i < counter; i++)

{

symbols[i].code += "0";

first.Add(symbols[i]);

}

for (int i = counter; i < symbols.Count; i++)

{

symbols[i].code += "1";

second.Add(symbols[i]);

}

if (symbols.Count > 1)

{

first = AddCodes(first);

second = AddCodes(second);

first.AddRange(second);

symbols = first;

}

return symbols;

}

}

}

**Ответы на контрольные вопросы**

1. Что такое бинарное дерево, чем характеризуется его структура?

Бинарное дерево — это иерархическая структура данных, в которой каждый узел имеет значение и ссылки на левого и правого потомка. Узел, находящийся на самом верхнем уровне называется корнем. Узлы, не имеющие потомков, называются листьями.

2. Какие коды называются префиксными?

Префиксный код — код, в котором никакое кодовое слово не является префиксом какого-то другого кодового слова. То есть, никакое кодовое слово не начинается с другого кодовое слова.

3. Как, на Ваш взгляд, при увеличении мощности алфавита меняется его избыточность (или не меняется)?

При увеличении мощности алфавита избыточность уменьшается, но после перехода границы log2X, где X – мощность алфавита, избыточность сильно увеличивается. Поэтому можно сказать, что при увеличении мощности алфавита, его избыточно периодически уменьшается и увеличивается.

4. В чем состоит суть кодирования по методам Шеннона – Фано и Хаффмана?

По этим методам нужно построить таблицу символ-код, таким образом, чтобы символам, которые встречаются чаще всего соответствовало меньшее количество битов, чем символам, которые встречаются реже.

**Вывод:** статистические методы сжатия строятся на основе частоты появления символов, они часто используются в архиваторах совместно с другими методами сжатия.

Учреждение образования

«Белорусский государственный технологический университет»

**Защита информации и надежность информационных систем**

**Лабораторная работа №10**

СЖАТИЕ/РАСПАКОВКА ДАННЫХ МЕТОДОМ ЛЕМПЕЛЯ − ЗИВА

Выполнил:

Студент 3 курса 2 группы ФИТ

Максимова Вера Владимировна

Проверила:

Ржеутская Надежда Викентьевна

**2022 г.**

**Цель**: приобретение практических навыков использования метод Лемпеля − Зива (Lempel-Ziv) для сжатия/распаковки данных.

**Задачи**:

1. Закрепить теоретические знания по алгебраическому описанию и использованию методов сжатия/распаковки (архивации/ разархивации) данных на основе метода Лемпеля − Зива.

2. Разработать приложение для реализации метода Лемпеля − Зива.

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

**Теоретические сведения**

Суть метода LZ77 (как и последующих его модификаций) состоит в следующем: упаковщик постоянно хранит некоторое количество последних обработанных символов в буфере. По мере обработки входного потока вновь поступившие символы попадают в конец буфера, сдвигая предшествующие символы и вытесняя самые старые. Размеры этого буфера, называемого также скользящим словарем (англ. sliding dictionary), варьируются в разных реализациях систем сжатия. Скользящее окно имеет длину n, т. е. в него помещается n символов, и состоит из двух частей

: • последовательности длины n1 = n − n2 уже закодированных символов (словарь);

• упреждающего буфера (буфера предварительного просмотра, lookahead) длиной n2 – буфера кодирования.

Нужно найти самое длинное совпадение между строкой буфера кодирования, начинающейся с символа St + 1, и всеми фразами словаря.

Эти фразы могут начинаться с любого символа St − (n1 − 1), St − (n1 − 1) + 1, …, St, выходить за пределы словаря, вторгаясь в область буфера, но должны лежать в окне. Буфер не может сравниваться сам с собой. Длина совпадения не должна превышать размера буфера. Полученная в результате поиска фраза St − (р − 1), St − (р − 1) + 1, St − (р − 1) + (q − 1) кодируется с помощью двух чисел:

1) смещения (англ. offset) от начала буфера p;

2) длины соответствия, или совпадения (англ. match length) q.

Длина кодовой комбинации (триады – p, q, s) на каждом шаге определяется соотношением

l(с ) = logN n1 +logN n2 + 1, (10.1)

где N – мощность алфавита.

**Код практического задания:**

using System;

using System.Collections.Generic;

using System.IO;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace LAB\_10

{

public class LZMessageList

{

private const string Bin = "01";

private const string Quad = "0123";

private const string Oct = "01234567";

private const string Hex = "0123456789ABCDEFG";

const string Step = "Step ";

const string Separator = "----------------------------------------------------";

private char[] Dictionary;

private char[] Buffer;

public int MessageLength;

public Mode MessageMode { get; private set; }

private List<LZMessage> LZMessages;

public int Index { get; private set; }

public int Count

{

get

{

return LZMessages.Count();

}

}

private string FilePath;

public LZMessageList(Mode mode, int dictionaryLength, int bufferLength, string path)

{

MessageMode = mode;

Index = 0;

Dictionary = new char[dictionaryLength];

Buffer = new char[bufferLength];

LZMessages = new List<LZMessage>();

FilePath = path;

int logn1 = (int)Math.Ceiling(Math.Log(dictionaryLength, (double)mode));

int logn2 = (int)Math.Ceiling(Math.Log(bufferLength, (double)mode));

MessageLength = logn1 + logn2 + 1;

}

public void AddText(string text)

{

var sf = File.Create(FilePath);

sf.Close();

LZMessages.Clear();

int textIndex = 0;

Queue<char> queue = new Queue<char>(text);

for (int i = 0; i < Dictionary.Length; i++)

Dictionary[i] = '0';

for (int i = 0; i < Buffer.Length; i++, textIndex++)

{

if (queue.Count != 0)

Buffer[i] = queue.Dequeue();

else

Buffer[i] = '-';

}

Console.WriteLine("----------------------------------------------------");

Console.WriteLine("Encryption");

Console.WriteLine("------------------------------------------------------------");

Console.WriteLine(new string(Dictionary) + " " + new string(Buffer) + " " + new string(queue.ToArray()));

Additions.EnterLineToFile(FilePath, Separator);

Additions.EnterLineToFile(FilePath, "Encryption");

Additions.EnterLineToFile(FilePath, Separator);

Additions.EnterLineToFile(FilePath, new string(Dictionary) + " " + new string(Buffer) + " " + new string(queue.ToArray()));

int numlen = (MessageLength - 1) / 2;

int windowlen = Dictionary.Length + Buffer.Length;

int index = 0;

while (Buffer[0] != '-')

{

string window = new string(Dictionary) + new string(Buffer);

int bufstart = Dictionary.Length;

int len = 0;

int place = 0;

for (int i = 1; i < Buffer.Length; i++)

{

string symbols = window.Substring(bufstart, i);

if (symbols.Last() == '-') break;

int find = window.IndexOf(symbols);

if (find != -1 && find < bufstart)

{

place = find;

len = i;

}

else

break;

}

char s = window[bufstart + len];

if (s == '-')

{

len = 0;

place = 0;

s = window[bufstart + len];

}

string p = Convert(place);

string q = Convert(len);

while (p.Length < numlen)

p = p.Insert(0, "0");

while (q.Length < numlen)

q = q.Insert(0, "0");

var message = new LZMessage(MessageMode, MessageLength, p + q + s);

Add(message);

Console.WriteLine("------------------------------------------------------------");

Console.WriteLine(new string(Dictionary) + " " + new string(Buffer) + " " + new string(queue.ToArray()));

Console.WriteLine(LZMessages.Last().ToString());

Additions.EnterLineToFile(FilePath, Separator);

Additions.EnterLineToFile(FilePath, Step + (index + 1).ToString());

Additions.EnterLineToFile(FilePath, new string(Dictionary) + " " + new string(Buffer) + " " + new string(queue.ToArray()));

Additions.EnterLineToFile(FilePath, LZMessages.Last().ToString());

len++;

index++;

for (int i = 0; i < Dictionary.Length; i++)

{

if (i + len < Dictionary.Length)

Dictionary[i] = Dictionary[i + len];

else

Dictionary[i] = Buffer[i + len - bufstart];

}

for (int i = 0; i < Buffer.Length; i++)

{

if (i + len < Buffer.Length)

Buffer[i] = Buffer[i + len];

else

{

if (queue.Count != 0)

Buffer[i] = queue.Dequeue();

else

Buffer[i] = '-';

}

}

}

Console.WriteLine("------------------------------------------------------------");

Console.WriteLine(new string(Dictionary) + " " + new string(Buffer) + " " + new string(queue.ToArray()));

Console.WriteLine("------------------------------------------------------------");

Additions.EnterLineToFile(FilePath, Separator);

Additions.EnterLineToFile(FilePath, new string(Dictionary) + " " + new string(Buffer) + " " + new string(queue.ToArray()));

Additions.EnterLineToFile(FilePath, Separator);

}

public string GetText()

{

string result = "";

for (int i = 0; i < Dictionary.Length; i++)

Dictionary[i] = '0';

Console.WriteLine("----------------------------------------------------");

Console.WriteLine("Decryption");

Console.WriteLine("----------------------------------------------------");

Additions.EnterLineToFile(FilePath, Separator);

Additions.EnterLineToFile(FilePath, "Decryption");

Additions.EnterLineToFile(FilePath, Separator);

Console.WriteLine(new string(Dictionary) + " " + GetMessagesString(0, true));

int numlen = (MessageLength - 1) / 2;

for (int i = 0; i < LZMessages.Count; i++)

{

Console.WriteLine("----------------------------------------------------");

Console.WriteLine(new string(Dictionary) + " " + GetMessagesString(i, true));

Console.WriteLine(result.GetSplitedString((int)Math.Log(256, (int)MessageMode)));

Additions.EnterLineToFile(FilePath, Separator);

Additions.EnterLineToFile(FilePath, Step + (i + 1).ToString());

Additions.EnterLineToFile(FilePath, new string(Dictionary) + " " + GetMessagesString(i, true));

Additions.EnterLineToFile(FilePath, result.GetSplitedString((int)Math.Log(256, (int)MessageMode)));

var mes = LZMessages[i];

string dict = new string(Dictionary);

string buf = mes.Message;

int p = buf.Substring(0, numlen).ArbitraryToDecimalSystem((int)MessageMode);

int q = buf.Substring(numlen, numlen).ArbitraryToDecimalSystem((int)MessageMode);

char s = buf.Last();

string part = "";

if (p + q > dict.Length)

{

int razn = Math.Abs(dict.Length - p - q);

int dictcount = q - razn;

string dictbuf;

dictbuf = dict.Substring(p, dictcount);

part += dictbuf;

for (int j = 0; j < razn; j++)

{

part += dictbuf[j % dictbuf.Length];

}

part += s;

result += part;

}

else

{

part = dict.Substring(p, q) + s;

result += part;

}

int offset = part.Length;

int partIndex = 0;

for (int j = 0; j < Dictionary.Length; j++)

{

if (j + offset < Dictionary.Length)

{

Dictionary[j] = Dictionary[j + offset];

}

else

{

Dictionary[j] = part[partIndex];

partIndex++;

}

}

}

Console.WriteLine("----------------------------------------------------");

Console.WriteLine(result.GetSplitedString((int)Math.Log(256, (int)MessageMode)));

Additions.EnterLineToFile(FilePath, Separator);

Additions.EnterLineToFile(FilePath, Step + "Last");

Additions.EnterLineToFile(FilePath, result.GetSplitedString((int)Math.Log(256, (int)MessageMode)));

return result;

}

private void Add(LZMessage message)

{

if (message.MessageMode == MessageMode)

LZMessages.Add(message);

else

throw new Exception("Modes of messages and list don't intercept");

}

private string Convert(int num)

{

string result = "";

result = num.DecimalToArbitrarySystem((int)MessageMode);

return result;

}

private LZMessage GetNextMessage()

{

if (Index < Count)

{

Index++;

return LZMessages[Index];

}

else

throw new ArgumentOutOfRangeException();

}

public string GetMessagesString(int from, bool spaces)

{

string result = "";

for (int i = from; i < LZMessages.Count; i++)

{

if (spaces)

result += LZMessages[i].ToString();

else

result += LZMessages[i].Message;

}

return result;

}

public override string ToString()

{

string result = "";

foreach (var mes in LZMessages)

result += mes.ToString();

return result;

}

}

}

**Ответы на контрольные вопросы**

1. К какому классу методов сжатия относится метод LZ и почему?

К словарному методу сжатия.

2. В чем заключается, на Ваш взгляд, оптимизация алгоритма LZ?

Нужно найти самое длинное совпадение между строкой буфера кодирования, начинающейся с символа St + 1, и всеми фразами словаря.

3. Какое значение имеет последний символ в сжатом текстовом сообщении методом LZ?

Последний символ в исходном сообщении.

4. Какие модификации метода LZ77 Вам известны? В чем заключается особенность модификаций?

LZ78, суть в том, что кроме соответствий, смотрятся соответствия в просмотренной части фразы.

5. Пояснить физический смысл соотношения (10.1).

l(с) = logN n1 +logN n2 + 1

Длина кодовой комбинации состоит из начала соответствия + количество символов в соответствии и символа, идущего за соответствием.

6. Как будет выглядеть сжатое сообщение, состоящее из символа моноалфавита при определенных значениях n1, n2?

В LZ78 (0, 0, A), (n1, n2-2, A) если сообщение полностью входит в буфер данных.

**Вывод**: метод сжатия LZ78 основан на поиске последовательностей, которые уже встречались в сообщении, при увеличении размеров словаря и буфера данных, увеличивается процент сжатия и время сжатия сообщения.

Учреждение образования

«Белорусский государственный технологический университет»

**Кафедра информационных систем и технологий**

**Защита информации и надёжность информационных систем**

**Лабораторная работа №11**

СЖАТИЕ/РАСПАКОВКА ДАННЫХ АРИФМЕТИЧЕСКИМ МЕТОДОМ.

Вариант № 6

Выполнил:

Студент 3 курса 2 группы ФИТ

Максимова Вера Владимировна

Цель: приобретение практических навыков использования арифметических методов сжатия/распаковки данных.

Задачи:

1. Закрепить теоретические знания по алгебраическому описанию и использованию арифметических методов сжатия/распаковки (архивации/разархивации) данных.

2. Разработать приложение для реализации арифметических методов.

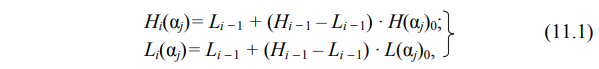
3. Результаты выполнения лабораторной работы оформить в виде описания разработанного приложения, методики выполнения экспериментов с использованием приложения и результатов эксперимента.

**Теоретические сведения**

При арифметическом сжатии (кодировании) текст представляется вещественными числами в интервале от 0 до 1. По мере анализа текста отображающий его интервал уменьшается, а количество битов для его представления возрастает. Очередные символы сокращают величину интеpвала, исходя из значений, соответствующих вероятностей.

Прямое преобразование (сжатие). Один шаг сжатия (кодирования) заключается в простой операции: берется кодируемый символ, для него ищется соответствующий участок на рабочем отрезке. Найденный участок становится новым рабочим отрезком. Его тоже необходимо разбить с помощью точек.

Это и последующие разбиения отрезка (на шаге i) подразумевают определение новых значений верхней (Hi) и нижней (Li) границ для всего участка и осуществляются по следующим правилам:



где αj – j-й символ сжимаемой последовательности, Li − 1 и Hi − 1 – соответственно нижняя и верхняя границы рабочего отрезка на (i − 1)-м шаге, L(αj)0 и H(αj)0 – соответственно исходные нижняя и верхняя границы символа αj.

Шаг 1: в нашем примере на первом шаге берется первый символ последовательности и ищется соответствующий участок на рабочем отрезке. Он становится новым рабочим отрезком и опять разбивается согласно статистике и соотношениям. Повторяем первый шаг до тех пор, пока не останется 1 символ. В качестве результата кодирования берём минимальная граница отрезка последнего символа.

Обратное преобразование (декомпрессия). Для восстановления исходного сообщения необходима информация:

• о значении числа, являющегося итогом сжатия сообщения (в нашем случае 0,1071);

• количестве символов в сжатом сообщении;

• вероятностных параметрах всех символов исходного сообщения (таблица вероятностей).

Как и при сжатии, вначале необходимо начальный рабочий отрезок [0; 1) разбить на интервалы, длины которых равны вероятностям появления соответствующих символов, т. е. создать рабочий отрезок.

Анализ будем проводить с использованием конкретных данных, взятых из последнего примера.

Итак, в качестве исходного участка для обратного преобразования принимается исходный участок для прямого преобразования с одинаковыми точками его разбиения.

На каждом шаге обратного преобразования выбираем отрезок, в который попадает текущее число (код). Символ, который соответствует данному отрезку, является очередным символом восстановленного (распакованного) сообщения.

В общем случае код символа, восстанавливаемого на шаге i, вычисляется соотношением:

код i = [код (i − 1) − L(αi − 1)0] / [H(αi − 1)0 − L(αi − 1)0],

где код (i − 1) – число, анализ которого производился на предыдущем шаге – (i − 1)-м; H(αi − 1)0 и L(αi − 1)0 – соответственно верхняя и нижняя исходные границы символа сообщения, восстановленного на предыдущем шаге.

**Код практического задания:**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace \_11

{

public class SymbolWithCode

{

public SymbolWithCode(char s, int cnt, double p, string c)

{

this.symbol = s;

this.count = cnt;

this.probalility = p;

this.code = c;

this.section\_a = 0;

this.section\_b = 0;

}

public char symbol;

public int count;

public double probalility;

public string code;

public double section\_a;

public double section\_b;

public static void ShowSymbolsWithCodes(List<SymbolWithCode> symbolsWithCodes)

{

foreach (var symbolWithCode in symbolsWithCodes)

{

Console.WriteLine("{0} {1} {2} {3} [ {4} - {5} ]",

symbolWithCode.symbol,

symbolWithCode.count,

symbolWithCode.probalility,

symbolWithCode.code,

symbolWithCode.section\_a,

symbolWithCode.section\_b

);

}

Console.WriteLine();

}

public static List<SymbolWithCode> AddSymbolsOfLine(List<SymbolWithCode> symbolsWithCodes, string line)

{

foreach (var character in line)

{

if (symbolsWithCodes.Find(x => x.symbol == character) == null)

{

symbolsWithCodes.Add(new SymbolWithCode(character, 1, 0, ""));

}

else

{

symbolsWithCodes.Where(x => x.symbol == character).ToList().ForEach(x => x.count++);

}

}

return symbolsWithCodes;

}

}

class Program

{

public static List<SymbolWithCode> AddCodes(List<SymbolWithCode> symbolsWithCodes)

{

int counter = 0;

double probability = 0;

List<SymbolWithCode> firstPartOfSymbolsWithCodes = new List<SymbolWithCode>();

List<SymbolWithCode> secondPartOfSymbolsWithCodes = new List<SymbolWithCode>();

while (probability < (symbolsWithCodes.Sum(x => x.probalility) / 2))

{

probability += symbolsWithCodes[counter].probalility;

counter++;

}

if (counter > 1)

counter--;

for (int i = 0; i < counter; i++)

{

symbolsWithCodes[i].code += "0";

firstPartOfSymbolsWithCodes.Add(symbolsWithCodes[i]);

}

for (int i = counter; i < symbolsWithCodes.Count; i++)

{

symbolsWithCodes[i].code += "1";

secondPartOfSymbolsWithCodes.Add(symbolsWithCodes[i]);

}

if (symbolsWithCodes.Count > 1)

{

firstPartOfSymbolsWithCodes = AddCodes(firstPartOfSymbolsWithCodes);

secondPartOfSymbolsWithCodes = AddCodes(secondPartOfSymbolsWithCodes);

firstPartOfSymbolsWithCodes.AddRange(secondPartOfSymbolsWithCodes);

symbolsWithCodes = firstPartOfSymbolsWithCodes;

}

return symbolsWithCodes;

}

static void Main(string[] args)

{

string[] names = { "электрифицированный ", "электрифицированныйводоворотоподобный " };

foreach (var name in names)

{

List<SymbolWithCode> symbolsWithCodes = new List<SymbolWithCode>();

symbolsWithCodes = SymbolWithCode.AddSymbolsOfLine(symbolsWithCodes, name);

SymbolWithCode.ShowSymbolsWithCodes(symbolsWithCodes);

double sumChars = symbolsWithCodes.Sum(x => x.count);

for (int i = 0; i < symbolsWithCodes.Count; i++)

{

symbolsWithCodes[i].probalility = symbolsWithCodes[i].count / sumChars;

}

symbolsWithCodes = symbolsWithCodes.OrderBy(x => x.probalility).ToList();

SymbolWithCode.ShowSymbolsWithCodes(symbolsWithCodes);

double b = 0;

for (int i = 0; i < symbolsWithCodes.Count; i++)

{

symbolsWithCodes[i].section\_a = b;

b += symbolsWithCodes[i].probalility;

symbolsWithCodes[i].section\_b = b;

}

SymbolWithCode.ShowSymbolsWithCodes(symbolsWithCodes);

double L = 0, H = 0, L0 = 0, H0 = 0, L1 = 0, H1 = 0;

double sec\_a, sec\_b;

L0 = symbolsWithCodes.Find(x => x.symbol == name[0]).section\_a;

H0 = symbolsWithCodes.Find(x => x.symbol == name[0]).section\_b;

for (int i = 1; i < name.Length; i++)

{

sec\_a = symbolsWithCodes.Find(x => x.symbol == name[i]).section\_a;

sec\_b = symbolsWithCodes.Find(x => x.symbol == name[i]).section\_b;

L1 = L0 + (H0 - L0) \* sec\_a;

H1 = L0 + (H0 - L0) \* sec\_b;

L0 = L1;

H0 = H1;

//Console.WriteLine(H1 - L1);

Console.WriteLine("[{0} - {1}]", L1, H1);

}

Console.WriteLine("Нижняя граница, являющаяся итогом кодирования = {0}", L1);

string newName = "";

SymbolWithCode newSymbol;

for (int i = 0; i < name.Length; i++)

{

newSymbol = symbolsWithCodes.Find(x => (L1 <= x.section\_b) && (L1 >= x.section\_a));

//newSymbol = symbolsWithCodes.Find(x => (L1 <= x.section\_b));

sec\_a = newSymbol.section\_a;

sec\_b = newSymbol.section\_b;

newName += newSymbol.symbol;

L1 = (L1 - sec\_a) / (sec\_b - sec\_a);

}

Console.WriteLine("Декодированное сообщение - {0}", newName);

}

}

}

}

**Ответы на контрольные вопросы:**

1. В чем заключается суть арифметического метода сжатия?

Суть арифметического метода сжатия заключается в разбиении вех возможных комбинаций символов на вероятности от 0 до 1.

2. Пояснить принцип определения границ отрезков на каждом шаге (на каждом рабочем интервале) прямого и обратного преобразования.

В начале берется интервал от 0 до 1, а разбиение идет по вероятностям появления символов. На следующих шагах в место интервала от 0 до 1 берутся нижняя и верхняя границы символа и этот отрезок делится на интервалы, в зависимости от вероятностей появления символов.

3. В чем заключаются основные отличия метода арифметического сжатия от метода Хаффмана?

В методе Хаффмана строится дерево, по которому находятся коды отдельных символов, а в арифметическом методе вычисляется конечная последовательность в виде числа, не считая коды отдельных символов.

4. Какой недостаток, свойственный префиксным методам, отсутствует в арифметическом кодировании?

Наличие промежуточных результатов.

5. Насколько эффективен арифметический метод для кодирования данных с равномерно распределенными вероятностями появления символов?

Эффективен за счёт возможности передачи последовательности каждого уникального символа и двух чисел – вероятности и конечной последовательности.

6. Какие требования предъявляет арифметический метод к его программной реализации?

Переменная с большой точностью для хранения выходящего значения.

7. Какие сложности возникают при реализации алгоритма в программном коде? Как можно их решить?

Проблемы с округлением чисел с большой точностью. Это обходится, если использовать небольшие последовательности.

**Вывод**: арифметический метод сжатия довольно эффективен, но при использовании метода может происходить округление некоторых чисел. Что приведет к потере данных.

Учреждение образования

«Белорусский государственный технологический университет»

**Кафедра информационных систем и технологий**

**Защита информации и надёжность информационных систем**

**Лабораторная работа №11**

Основы теории чисел и их использование в криптографии.

Вариант № 6

Выполнил:

Студент 3 курса 2 группы ФИТ

Максимова Вера Владимировна

**Цель**: приобретение практических навыков выполнения операций с числами для решения задач в области криптографии и разработка приложений для автоматизации этих операций.

**Задачи**:

1. Закрепить теоретические знания по высшей арифметике.

2. Научиться практически решать задачи с использованием простых и взаимно простых чисел, вычислений по правилам модулярной арифметики и нахождению обратных чисел по модулю.

3. Ознакомиться с особенностями реализации готового программного средства L\_PROST и особенностями выполнения с его помощью операций над простыми числами.

4. Разработать приложение для реализации указанных преподавателем операций с числами.

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

**Теоретические сведения**

Определение 1. Множество всех целых чисел (обозначим буквой Z) есть набор всех действительных чисел без дробной части: {..., –3, –2, –1, 0, 1, 2, 3, ...}.

Определение 2. Натуральные числа являются подмножеством целых чисел и образуют множество N: {1, 2, 3, ...}.

Определение 3. Делимость – одно из основных понятий теории чисел. Если для некоторого целого числа a и натурального числа b существует целое число q, при котором bq = a, то говорят, что число a делится на b. В этом случае b называется делителем числа a, а a называется кратным числу b. При этом используются следующие обозначения:

a ⋮ b – a делится на b, или b | a – b делит a.

Из последнего определения следует, что:

• любое натуральное число является делителем нуля;

• единица является делителем любого целого числа;

• любое натуральное число является делителем самого себя.

Определение 4. Делитель a называется собственным делителем числа b, если 1 < |a| < |b|, и несобственным – в противном случае.

Определение 5. Всякое целое число а можно представить с помощью положительного целого числа b равенством вида а = bq + r, 0 ≤ r ≤ b. Число q называется неполным частным, а число r – остатком отделения а на b.

Определение 6. Натуральное число n называется простым, если n > 1 и не имеет положительных делителей, отличных от 1 и n.

Свойство 1. Любое составное число представляется уникальным образом в виде произведения простых чисел; иначе еще говорят, что разложение числа на простые множители однозначно.

Свойство 2. Простых чисел бесконечно много, причем существует примерно n / ln(n) простых чисел, меньших числа n.

Свойство 3. Наименьший простой делитель составного числа n не превышает √n, поэтому для проверки простоты числа достаточно проверить его делимость на 2 и на все нечетные (а еще лучше простые) числа, не превосходящие √n; как видим, данное свойство коррелирует со свойством 1 собственного делителя.

Свойство 4. Любое четное число, большее 2, представимо в виде суммы двух простых чисел, а любое нечетное, большее 5, представимо в виде суммы трех простых чисел.

Свойство 5. Для любого натурального n, большего 1, существует хотя бы одно простое число на интервале от n до 2n.

Определение 7. Натуральное число n называется составным, если n > 1 и имеет по крайней мере один положительный делитель, отличный от 1 и n.

Определение 8. Если два простых числа отличаются на 2, то их называют числами-близнецами.

Определение 9. Наибольшее целое число, которое делит без остатка числа a и b, называется наибольшим общим делителем этих чисел – НОД (a, b).

Определение 10. Взаимно простыми являются целые числа, наибольший общий делитель которых равен 1.

**Код практического задания**

using System;

using System.Collections.Generic;

namespace Nod

{

class Program

{

static void Main(string[] args)

{

int c = 5;

Console.WriteLine("Приложение для подсчёта НОД 2-ух числе, а также для поиска простых чисел");

Console.WriteLine();

while (c == 5)

{

Console.WriteLine();

Console.WriteLine("Нажмите");

Console.WriteLine("1 - Поиск НОД");

Console.WriteLine("2 - Поиск простых чисел от 1 до n ");

Console.WriteLine("3 - Поиск простых чисел от m до n ");

Console.WriteLine("0 - Выход из консоли ");

string selection = Console.ReadLine();

switch (selection)

{

case "1":

Console.WriteLine("Поиск НОД!");

Console.Write("Введите a(>0): ");

int a = Convert.ToInt32(Console.ReadLine());

Console.Write("Введите b(>0): ");

int b = Convert.ToInt32(Console.ReadLine());

int result = Nod(a, b);

Console.WriteLine("NOD");

Console.WriteLine(result);

break;

case "2":

Console.WriteLine("Введите n: ");

int z = Convert.ToInt32(Console.ReadLine());

Simple(z);

break;

case "3":

Console.WriteLine("Введите n: ");

int q = Convert.ToInt32(Console.ReadLine());

Console.WriteLine("Введите m: ");

int x = Convert.ToInt32(Console.ReadLine());

Simple2(q, x);

break;

case "0":

Environment.Exit(0);

break;

default:

Console.WriteLine("Вы нажали неизвестную букву");

break;

}

}

}

static int Nod(int a, int b)

{

if (a != 0 && b != 0)

{

if (a > b)

{

a = a % b;

}

else

{

b = b % a;

}

return Nod(a,b);

}

else

{

return a + b;

}

}

static void Simple(int z)

{

List<int> num = new List<int> { };

for (int i = 2; i <= z; i++)

{

num.Add(i);

}

for (int i = 0; i < num.Count; i++)

{

for (int j = 2; j < z; j++)

num.Remove(num[i] \* j);

}

if (num.Count != 0)

{

Console.WriteLine("Простые числа от 1 до " + z);

foreach (int w in num)

{

Console.Write(w);

Console.Write("; ");

}

Console.WriteLine("Количесвто простых чисел: " + num.Count);

}

else

{

Console.WriteLine("Простых чисел в данном диапозоне нет!!!");

}

}

static void Simple2(int q, int z)

{

List<int> num = new List<int> { };

for (int i = 2; i <= z; i++)

{

num.Add(i);

}

for (int i = 0; i < num.Count; i++)

{

for (int j = 2; j < z; j++)

num.Remove(num[i] \* j);

}

if (num.Count != 0)

{

Console.WriteLine("Простые числа от "+ q + " до " + z);

foreach (int w in num)

{

if (w > q)

{

Console.Write(w);

Console.Write("; ");

}

}

}

else

{

Console.WriteLine("Простых чисел в данном диапозоне нет!!!");

}

}

}

}

**Ответы на контрольные вопросы:**

1. Дать определение понятий: целое число, натуральное число, делимость чисел, собственный делитель, НОД.

Целое число – действительные числа без дробной части.

Натуральное число – целые числа, которые используются при счёте.

Делимость чисел – свойство числа а и b, при котором a= b\*q, где все числа целые.

Собственный делитель – Делитель a называется собственным делителем числа b, если 1 < |a| < |b|.

НОД – Наибольший общий делитель одного или нескольких чисел.

2. Сформулировать основную теорему арифметики. Представить примеры ее применения.

Всякое натуральное число n, кроме 1, можно представить, как произведение простых множителей. 12 = 3\*2\*2.

3. Пояснить сущность проблемы факторизации и ее связь с прикладной криптографией.

Проблема факторизации - сложность решения задачи разложения больших чисел на простые сомножители. Если числа имеют большой размер, то компьютеры могут искать ответ столетиями, что делает системы криптоустойчивами.

4. Найти НОД: пар чисел: 333 и 100; 56 и 200; 99 и 200; 61 и 987; 123 и 456; трех чисел: 21, 43, 342; 57, 31, 200; 42, 11, 98.

НОД(333,100) = 1;

НОД(56,200) = 8;

НОД(99,200) = 1;

НОД(61,987) = 1;

НОД(123,456) = 3;

НОД(21,43,342) = 1;

НОД(57,31,200) = 1;

НОД(42,11,98) = 1;

5. Записать каноническое разложение чисел: 2770; 3780; 6224.

2770 = 2\*5\*277

3780 = 2\*5\*2\*7\*3\*3\*3.

6224 = 2\*2\*2\*2\*389

6. Записать соотношение Безу. Показать пример его практического использования.

Соотношение между парой целых чисел a, b и их наибольшим общим делителем: НОД (a, b) = ax + by.

7. Подсчитать число взаимно простых чисел с числами 2770, 3780, 6224.

2770 = 2770\*(1-1/2)\* (1-1/5)\* (1-1/277) = 1104

3780 = 3780\*(1-1/2)\* (1-1/2)\* (1-1/5) \*(1-1/3)\* (1-1/3)\* (1-1/3) \* (1-1/7) = 192

6224 =6224\*(1-1/2)\* (1-1/2)\* (1-1/2) \*(1-1/2) \*(1-1/389) = 388

8. Сформулировать малую теорему Ферма. Показать примеры ее практического применения.

Если p — простое число и a — целое число, не делящееся на p, то ар-1-1 делится на p.

К примеру, если a=2; p=7, то 26=64, и 64-1=63=7\*9.

**Вывод**: модулярная арифметика применяется в криптографии для того, чтобы было нельзя было эффективно использовать методы подбора.