# 1. Сущность проблемы информационной безопасности и надежности систем.

***Информация*** – сведения (данные) о внутреннем и окружающем нас мире, воспринимаемые и передаваемые людьми или техническими устройствами.

**Информационная (информационно-вычислительная) система** – организационно упорядоченная совокупность документов, технических средств и информационных технологий, реализующая информационные (информационно-вычислительные) процессы.

**Информационные процессы** – процессы сбора, накопления, хранения, обработки (переработки), передачи и использования информации

**Информационные ресурсы** – отдельные документы или массивы документов в информационных системах.

**Информационные технологии**— процессы, методы поиска, сбора, хранения, обработки, предоставления, распространения информации и способы осуществления таких процессов и методов.

**Доступ** – специальный тип взаимодействия между объектом и субъектом, в результате которого создается поток информации от одного к другому.

**Несанкционированный доступ** – доступ к информации, устройствам ее хранения и обработки, а также к каналам передачи, реализуемый без ведома (санкции) владельца и нарушающий тем самым установленные правила доступа.

**Объект** – пассивный компонент системы, хранящий, перерабатывающий, передающий или принимающий информацию; примеры объектов: страницы, файлы, папки, директории, компьютерные программы, устройства (мониторы, диски, принтеры и т.д.)

**Субъект** – активный компонент системы, который может инициировать поток информации; примеры субъектов: пользователь, процесс либо устройство.

**Безопасность ИВС** – свойство системы, выражающееся в способности системы противодействовать попыткам несанкционированного доступа или нанесения ущерба владельцам и пользователям системы при различных умышленных и неумышленных воздействиях на нее.

**Защита информации** – организационные, правовые, программно-технические и иные меры по предотвращению угроз информационной безопасности и устранению их последствий.

**Защита информации (лучшее определение)** — Все аспекты, связанные с определением, достижением и поддержанием ***конфиденциальности, целостности и доступности*** информации или средств ее обработки.

**Конфиденциальность —** заключается в том, что объект доступен только субъектам, которые имеют на это соответствующие полномочия.

**Целостность** — заключается в том, что система может быть модифицирована только субъектом, имеющим на то соответствующие права. Целостность является гарантом корректности компонента в любой момент времени.

**Доступность** — имеющий полномочия субъект в любой момент  без проблем может получить доступ к необходимому компоненту системы.

**Информационная безопасность** систем – свойство информационной системы или реализуемого в ней процесса, характеризующее способность обеспечить необходимый уровень своей защиты.

Актуальность  обеспечения политики информационной безопасности обусловлены следующими факторами:

-      cовременные уровни и темпы развития средств информационной безопасности значительно отстают от уровней и темпов развития информационных технологий

-      высокие темпы роста парка персональных компьютеров

-      резкое расширение круга пользователей, имеющих непосредственный доступ к вычислительным ресурсам и массивам данных

# 2. Характеристики и параметры ИС и ИВС.

**Информационная безопасность** (information security) — все аспекты, связанные с определением, достижением и поддержанием конфиденциальности, целостности, доступности информации или средств её обработки (**СIA**):

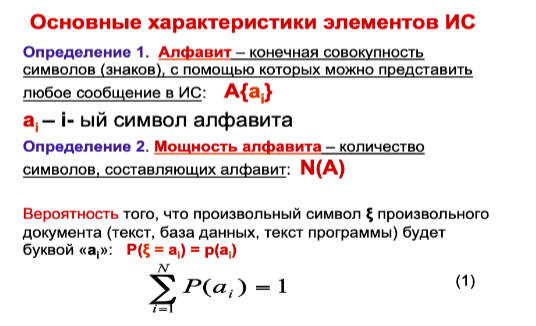
·        конфиденциальность (**C**onfidentiality) — состояние ин

формации, при котором доступ к+ ней осуществляют только субъекты, имеющие на нее право;

·   целостность (**I**ntegrity) — избежание несанкционированной модификации информации;

·        доступность (**A**vailability) — избежание временного или постоянного сокрытия информации от пользователей, получивших права доступа.

**Информационная (информационно-вычислительная) система** – организационно упорядоченная совокупность документов, технических средств и информационных технологий, реализующая информационные (информационно-вычислительные) процессы.



# 3. Характеристика угроз безопасности современным ИС и ИВС.

|  |  |
| --- | --- |
| Тип угрозы | Примеры |
| **Имитация**  **(Spoofing)** | Подделка электронных сообщений  Подделка ответных пакетов при аутентификации |
| **Фальсификация (Tampering)** | Модификация данных, передаваемых по сети  Модификация файлов |
| **Отречение (Repudiation)** | Удаление критичного файла или совершение покупки с последующим отказом признавать свои действия |
| **Раскрытие информации (Information disclosure)** | Несанкционированный доступ или незаконная публикация конфиденциальной информации |
| **Отказ в обслуживании (Denial of service)**  **Проблема botnet** | Заполнение сети пакетами «SYN»  Загрузка сетевого ресурса большим количеством поддельных пакетов ICMP |
| **Повышение привилегий**  **(Elevation of privilege)** | Получение системных привилегий через атаку с переполнением буфера  Незаконное получение административных прав либо незаконная их передача с целью наживы |

**Основные факторы (угрозы):**

1) действия злоумышленника;

2) наблюдение за источниками информации;

3) подслушивание конфиденциальных разговоров и акустических сигналов работающих механизмов;

4) перехват электрических, магнитных и электромагнитных полей, электрических сигналов и радиоактивных излучений;

5) несанкционированное распространение материальных носителей за пределами организации;

6) разглашение информации компетентными людьми;

7) утеря носителей информации;

8) несанкционированное распространение информации через поля и электрические сигналы, случайно возникшие в аппаратуре;

9) воздействие стихийных сил (наводнения, пожары и т. п.);

10) сбои и отказы в аппаратуре сбора, обработки и передачи информации;

11) отказы системы электроснабжения;

12) воздействие мощных электромагнитных и электрических помех (промышленных и природных).

Классификацию вредоносного ПО можно представить следующим образом: 1) вирусы (viruses); 2) черви (worms); 3) кейлоггеры (keyloggers); 4) трояны (trojans); 5) боты (bots); 6) снифферы (sniffers); 7) руткиты (rootkits).

# 4. Характеристика методов и средств защиты информации от несанкционированного доступа.

Классификация методов и средств обеспечения информационной безопасности:

1.     Организационные

2.     Правовые

3.     Технические

4.     Аппаратные, программные, аппаратно-программные

Чтобы исключить неправомерный доступ к информации применяют такие способы, как:

Идентификация – процедура, в результате выполнения которой для субъекта идентификации выявляется его идентификатор, однозначно идентифицирующий этого субъекта в информационной системе.

Аутентификация – процедура проверки подлинности (идентификации), например: проверка подлинности пользователя путем сравнения введённого им пароля с паролем, сохраненным в базе данных пользовательских логинов

Авторизация – подтверждение привилегии предоставление определённому лицу или группе лиц прав на выполнение определённых действий

Методы биометрические аутентификации:

Отпечаток пальцев, сканирование сетчатки глаза, лица.

# 5. Методология разработки и реализации политики безопасности современных ИС и ИВС

**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.

Процесс разработки политики информационной безопасности:

1. **Обоснование актуальности**, цели и задачи разработки ПИБ в организации (учреждении).
2. **Объекты** . Описание структуры организации (учреждения)
3. **Основные угрозы и их источники**. Анализ потенциальных угроз
4. **Оценка угроз, рисков и уязвимостей**. Анализ ценности ресурсов, оценка значимости угроз, а также эффективности существующих и планируемых средств защиты
5. **Разработка и введение мер, методов и средств обеспечения требуемого уровня защищенности информационных /защиты ресурсов**.

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 обеспечению разработанных стандартов и процедур;
* план мероприятий по обучению персонала и тестированию знаний сотрудников, имеющих доступ к информационным ресурсам.

Для построения политики ИБ рассматривают следующие направления защиты ИВС:

* защита объектов ИВС;
* защита процессов, процедур и программ обработки информации;
* защита каналов связи;
* подавление побочных электромагнитных излучений;
* управление системой защиты.

Организационная защита обеспечивает:

* организацию охраны, режима, работу с кадрами и с документами;
* использование технических средств безопасности (например, простейших дверных замков, магнитных или иных карт и др.), информационно-аналитическую деятельность по выявлению внутренних и внешних угроз.

**Концепция информационной безопасности** определяет в целом цели политики и основные ее принципы в увязке со статусом, целями и задачами организации или учреждения;

Основными разделами концепции информационной безопасности могут быть следующие:

* определение ИБ (или СУИБ);
* структура информационной системы организации (учреждения) и вытекающая из этого структура системы обеспечения информационной безопасности;
* безопасность информации: принципы и стандарты;
* оценка рисков информационным ресурсам в организации (учреждении);
* описание основных механизмов контроля безопасности;
* обязанности и ответственность каждого отдела, управления или департамента, каждого сотрудника в реализации разработанной и утвержденной политики безопасности;
* обязанности лица (администратора безопасности), ответственного за организацию оперативного контроля и управления политикой безопасности;
* ссылки на документы об информационной безопасности, действующие на территории Республики Беларусь.

**Оценка рисков**

**Фактор, воздействующий на ИВС**, – это явление, действие или процесс, результатом которых может быть утечка, искажение, уничтожение данных, блокировка доступа к ним, повреждение или уничтожение системы защиты.

**Внутренние дестабилизирующие факторы,** влияющие на:

1) на программные средства (ПС):

* некорректный исходный алгоритм;
* неправильно запрограммированный исходный алгоритм (первичные ошибки);

2) на аппаратные средства (АС):

* системные ошибки при постановке задачи проектирования;
* отклонения от технологии изготовления комплектующих изделий и АС в целом;
* нарушение режима эксплуатации, вызванное внутренним состоянием АС.

**Внешние дестабилизирующие факторы,** влияющие на:

1) на программные средства:

* неквалифицированные пользователи;
* несанкционированный доступ к ПС с целью модификации кода;

2) на аппаратные средства:

* внешние климатические условия;
* электромагнитные и ионизирующие помехи;
* перебои в электроснабжении;
* недостаточная квалификация обслуживающего персонала.

**Риски и их оценка**

**Риск = Ущерб\*Вероятность**

**Ущерб следует из угрозы** *(Т.е. Риск > Угроза, для понимания разницы между Риском и Угрозой)*

**Мероприятия по оценке рисков**:

* идентификация значимых угроз и уязвимостей для идентифицированных ресурсов;
* оценка вероятности возникновения угроз и уязвимостей;
* вычисление рисков; оценивание рисков по заранее определенной шкале риска.

**Угрозы**:

* Естественные - физические и природные воздействия
* Искусственные - вызваны деятельностью человека

---

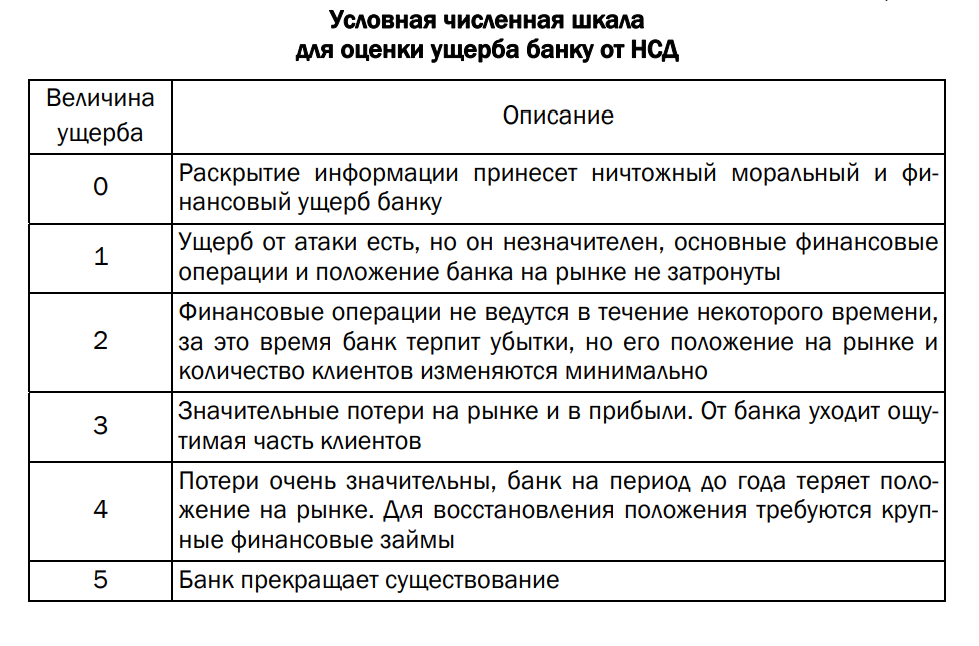
* Преднамеренные
* Непреднамеренные

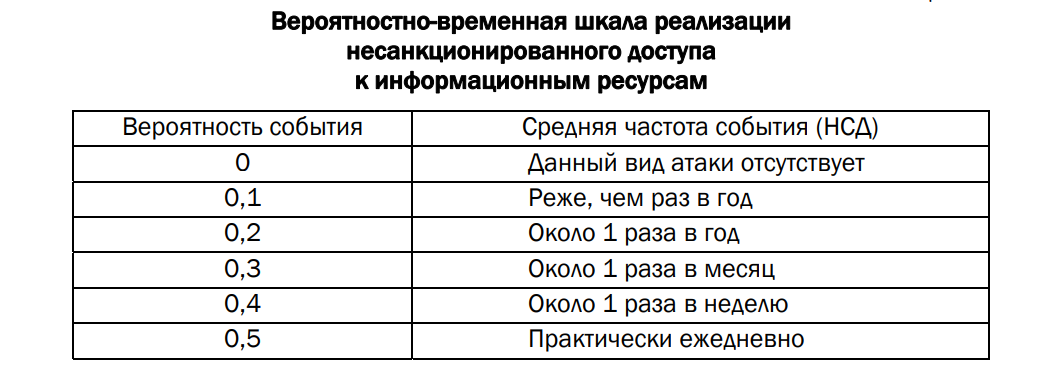
Цель оценивания рисков состоит в определении характеристик рисков для информационной системы и ее ресурсов. На основе таких данных могут быть выбраны необходимые средства управления ИБ.

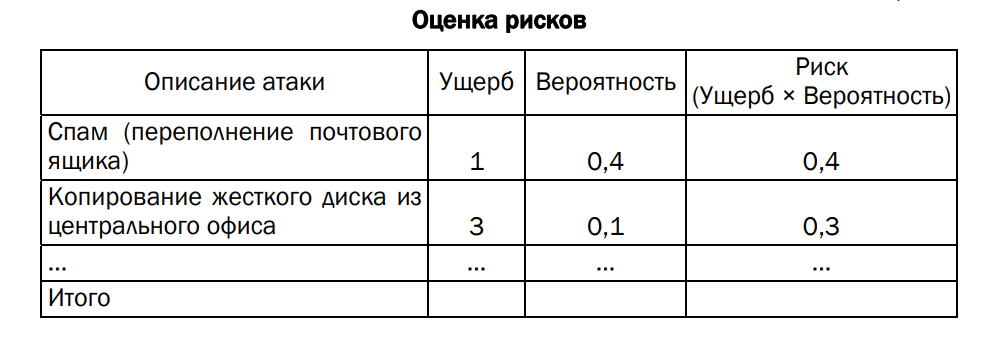
При оценивании рисков учитываются:

* ценность ресурсов;
* оценка значимости угроз;
* эффективность существующих и планируемых средств защиты.

**Шкала численной оценки ущерба от несанкционированного доступа (НСД)**







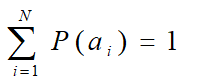
# 6. Энтропия источника сообщения. Энтропия Шеннона.

***Алфавит*** – конечная совокупность символов (знаков), с помощью которых можно представить любое сообщение в ИС.

Мощность алфавита – количество элементов (конечно/бесконечно) - **N(A).**

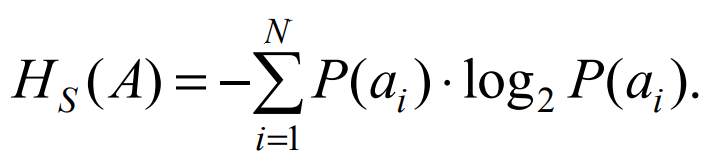
Вероятность того, что произвольный символ ξ произвольного документа (текст, база данных, текст программы) будет буквой «аi»:

 P (ξ = аi) = p(аi)



(Введено Шенноном) **Энтропия** – характеристика информативности информации.

***Энтропия Шеннона:***



***В двоичном алфавите стремится к 1, а вероятности 0 и 1 к 0.5***

**Энтропия алфавита** - информационная характеристика алфавита - показывает, какое количество информации (бит) приходится в среднем[p]

на один символ алфавита.

*-упомянуть что Энтропия в целом также является мерой защищенности пароля*

**Чем больше возможных исходов, тем больше энтропия алфавита.**

# 7. Энтропия источника сообщения. Энтропия Хартли.

(см в.6)

При p(ai) = const - вероятности всех символов одинаковые – **Энтропия Хартли**

|  |
| --- |
| **Hc (A) = log2 N** |

# 8. Двоичный канал передачи информации.

Процесс возникновения ошибок описывает модель под названием двоичный симметричный канал (ДСК). Двоичный канал передачи информации строится на основе двоичного алфавита: А={0,1}.

В общем случае, если передается сообщение Хk=х1, х2, …, хk, а принимается сообщение Yk= y1, y2, …., yk, то рассмотренные условные вероятности можно рассматривать с двух точек зрения: p(хi|yj) и p(yj|хi).



На рис. обозначены: q– вероятность безошибочной передачи бита сообщения, р– вероятность передачи бита с ошибкой. Понятно, что p + q = 1.

ДСК состоит в том, чтобы построить лучший канал. Если на каком-либо этапе дальнейшее улучшение канала оказывается неэкономичным или технически невозможным, то передача повторяется столько раз, сколько окажется нужным для того, чтобы результирующая [вероятность](http://edu.sernam.ru/book_kiber1.php?id=227) ошибки стала ниже некоторой удовлетворяющей проектировщика границы. Трудность, связанная с классическим подходом, состоит в том, что когда эта граница вероятности ошибки стремится к нулю, то или канал становится несоразмерно дорогим, или доход от его использования оказывается несоразмерно низким.

Теоремы Шенноны

1. Для заданного канала возможно при помощи кодирования вести передачу с вероятностью ошибки, меньшей любого наперед заданного значения, если скорость передачи информации не превышает некоторого предела, известного под названием [пропускной способности канала](http://sernam.ru/book_p_net.php?id=115) С.

2. Обратно, для скоростей передачи информации, больших С, невозможно вести передачу со сколь угодно малой вероятностью ошибки.

# 9. Энтропия двоичного алфавита.

***Энтропия Хартли:***

log2(2) = 1

***Энтропия Шеннона:***

**В двоичном алфавите стремится к 1, а вероятности 0 и 1 к 0.5 -**

можно представить p(0) = 1-p(1) и подставить в формулу

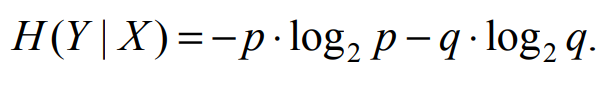
H(A2) = -(1-р(1))\*log2(1-р(1)) - р(1)\*log2(р(1)) (см 10)

(также см 10 про эффективную энтропию)

# 10. Условная энтропия. Энтропийная оценка потерь при передаче информации.

Если р > 0 (ДСК, p - вероятность ошибки), то это можно трактовать как неоднозначность (по Шеннону – *equivocation*) между переданным и принятым сообщениями.

Эта неоднозначность определяется как **условная энтропия (частная)**:

****

**Условная энтропия** – количество теряемой информации при передаче каждого символа сообщения.

Шеннон показал, что эффективная информация на выходе канала относительно входной в расчете на 1 символ (**Эффективная энтропия** алфавита) составляет:

**Не = H(X) – H(Y|X) *(для двоичного H(X) = 1)***

*Энтропийная оценка потерь при передаче информации показывает, сколько информации теряется при передаче из-за ошибок. Формула взаимной информации:*

***I(X;Y)=H(X)−H(X∣Y)***

**Интересные факты**, но Урбанович спрашивал часто:(спросит)

При вероятности ошибки **p = 0**, условная энтропия ДСК будет H(Y|X) = 0, эффективная - He = 1. То есть **информация не теряется**, а эффективная энтропия *= энтропии Хартли.*

При вероятности ошибки **p = 1** (все сообщения инвертированы), условная вероятность **ТАКЖЕ** будет = 0, потому что по факту мы можем все сообщения инвертировать назад, и получится что **никакой информации потеряно не было**. Эффективная энтропия тогда также = 1 (равняется энтропии Хартли)

# 11. Методы и средства структурной, информационной и временной избыточности в ИВС

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

***Используемые методы:***

1) увеличение наработки

2) снижение интенсивности отказов

3) улучшение восстанавливаемости

4) резервирование

Третью и четвертую группы можно объединить под единым названием – **избыточных методов:**

* **простая структурная избыточность** - структурное резервирование

**Структурное (элементное) резервирование** - метод повышения надежности объекта, предусматривающий использование избыточных элементов, входящих в физическую структуру объекта. Обеспечивается подключением к основной аппаратуре резервной таким образом, чтобы при отказе основной аппаратуры резервная продолжала выполнять ее функции.

* системы с **временной избыточностью** – системы с повторениями (передачи)
  + Реализуется введением в структуру ТС накопительного звена, позволяющего в течение определенного времени выполнять основную функцию при отказе элемента за накопительным звеном. Если избыточное время будет выше времени восстановления отказавшего элемента, то функция по назначению будет выполняться непрерывно и даже при отказе.
  + **Временное резервирование** - метод повышения надежности объекта, предусматривающий использование избыточного времени, выделенного для выполнения задач. Другими словами, временное резервирование - такое планирование работы системы, при котором создается резерв рабочего времени для выполнения заданных функций. Резервное время может быть использовано для повторения операции, либо для устранения неисправности объекта.
* **информационная избыточность** – обусловливает возможность применения функций сжатия информации

# 

# 12. Помехоустойчивое кодирование информации. Основные понятия. Назначение.

**(См 14)**

В общем плане помехоустойчивое кодирование можно понимать как **такое кодирование сообщений, при котором элементы связаны определенной зависимостью, позволяющей при ее нарушении указать ошибки и восстановить информацию.**

**Назначение:** защита информации от помех и ошибок при передаче и хранении информации. Помехоустойчивое кодирование необходимо для устранения ошибок, которые возникают в процессе передачи, хранения информации.

**Надежность** является комплексным свойством, включающим в себя единичные свойства: ***безотказность, ремонтопригодность, сохраняемость, долговечность.***

**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).

**Назначение** - обеспечение качества и надежности  ИВС.

# 13. Помехоустойчивое кодирование информации. Классификация кодов

***(См 14)***

***Классификация кодов (требуется адекватно реформатнуть это)***

* **Блочные коды** — каждому сообщению из  k (Xk)символов (бит) сопоставляется блок из n символов (кодовый вектор Xn длиной n=k + r).
* **Непрерывные** (рекуррентные, цепные, свёрточные) коды - непрерывная последовательность символов, не разделяемая на блоки. Передаваемая последовательность образуется путём размещения в определённом порядке проверочных символов между информационными символами исходной послед-ти.
* **Систематические** коды характеризуются тем, что сумма по модулю 2 двух разрешённых кодовых комбинаций кодов снова даёт разрешённую кодовую комбинацию.

Разрешенными кодовыми комбинациями называются те, которые соответствуют символам исходного алфавита. Их количество равно числу исходных символов (m).

* **Несистематические коды** не обладают отмеченными выше свойствами (к ним относятся итеративные коды).
* **Линейные** коды – проверочные (избыточные) символы вычисляются как линейная комбинация информационных символов; для кодов принимается обозначение [n, k]-код.
* **Нелинейные** коды являются противоположностью линейным.

(Циклический это комбинация, не основной)

*Циклические коды* – относятся к линейным систематическим.

* Основное свойство, давшее им название, состоит в том, что каждый вектор, получаемый из исходного кодового вектора путём циклической перестановки его символов, также является разрешённым кодовым вектором. Принято описывать циклические коды  при помощи порождающих полиномов G(X) степени r

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

# 14. Теоретические основы избыточного кодирования информации

**Информационная избыточность сообщений R** определяется по формуле:

**R = 1 - H / log2 k**

где k — число букв алфавита, а Н — энтропия источника на букву сообщения

Обычно данные делятся на блоки длиной k - размерность кода. Код обозначают записью (n, k)  - т.е. код использует n символов для хранения k символов сообщения.

Сообщение **Xk** (**k** – длина сообщения, символов или бит), называется **информационным словом**.

Избыточные символы длиной **r** символов (бит) составляющие избыточное слово **Xr.** Также называются **битами четности, контрольными битами, паритетами**… Не несут новой информации!

Слово **Xn** длиной **n=k+r** символов  **Xn = XkXr** называется **кодовым словом**.

**Rи=r/k** — **относительная избыточность кода**

**Rс=k/n** - **cкорость кода** или **степень кодирования**

**Вес Хемминга** произвольного двоичного слова Х (w(X)) равен количеству ненулевых символов в слове.

w(X=1101) = 3

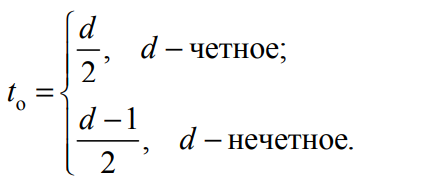
**Расстояние Хемминга** или кодовое расстояние (d) между двумя произвольными словами (X,Y) одинаковой длины равно количеству позиций, в которых X и Y отличаются между собой.

**d (X,Y) = w(X xor Y)**

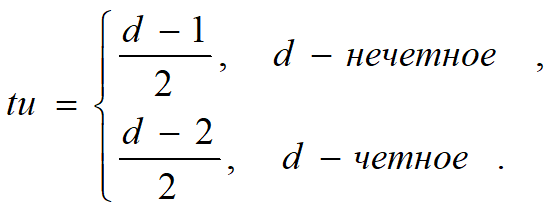
**Пример:**  X=101,Y=111. d(X,Y) = 1.

**d - расстояние Хемминга** между двумя **кодовыми словами**

Количество **ошибок, обнаруживаемое кодом**:



Корректирующие способности кода определяются исключительно **минимальным кодовым расстоянием  (dmin)** между двумя произвольными кодовыми словами, принадлежащими коду:

****

**Теорема. Минимальное кодовое расстояние** линейного   кода равно **минимальному весу ненулевых кодовых слов**

Выбор кода определяется вероятностью ошибки в канале, р (чем больше р, тем больше d , т.е. следует выбирать код с большим количеством избыточных битов, однако это снижает Rс )

**Определение**. Пропускная способность ДСК с вероятностью ошибки р равна

С(р) = 1 +р log2 р +  q log2 q

**Теорема Шеннона**. Для любого ДСК и любого ε >0 существует  (n,k)- двоичный код со скоростью Rс, если Rс< C(p), n достаточно велико и  р < ε.

**Основная проблема теории кодирования**:

Найти коды с большими  d И  Rс (задача оптимизационная) - т.е. коды исправляющие большое количество ошибок при как можно меньшей избыточности

# 15. Алгоритм использования корректирующего кода

На стороне ИС

1.Построить проверочную матрицу Hn,k для заданного k (Хk)

2.Вычислить символы избыточного слова Хr

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

3.Cформировать кодовое слово Хn = х1,х2,…,хk,xk+1,…xk+r и осуществить его передачу

На стороне ПС

1.Получение сообщения (Yn = y1,y2,…,yk,yk+1,…yk+r )

2.Вычисление синдрома (на основе (5) ), используя ту же Hn,k :

 S =HT \*Yn = H \*(Yn)T =HT \*(Хn +En) =HT \*Хn + HT \*En = HT \*En (10)р

для этого вычисляем Yr’= yri’   : yri’ = yk+i ‘= Σ hij \*yj (11)

и далее: S = s1,s2,…sr  , где si = yk+i + yri’ (12)

4. Анализ (декодирование синдрома) – определение местоположения ошибочного бита (En посредством S xor hm )

5. Исправление ошибки: Хn = Yn + En

# 16. Декодирование кодовых слов. Поиск и исправление ошибок.

При декодировании кодового слова необходимо в первую очередь определить **синдром**, чтобы понять, были ли допущены ошибки при передаче.

**Поиск ошибки**

Прежде всего нужно вычислить новый набор избыточных символов Y’r, используя кодовое слово Yn и проверочную матрицу H

Y’r1 = H11 \* Y1 xor H12 \* Y2 xor … xor H1k \* Yk и тд

Далее вычисляем сам синдром по следующей формуле

https://lh7-us.googleusercontent.com/QFye9bRLqSOmOCuhQLZ5UxH6EFGWJ1VN0Z6pxtTA6QXlnuKVYCbR2mj5SHNr91Tgw-Bb-qBFBdFfi05ro9bBz0T0tSrWvofOrZeGQ_IsmCeEN2dIkKLJd0GL__JLwoQjfSbr8OAR87tdpkbBeM_lBg

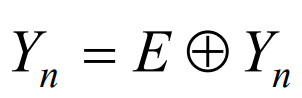
Ненулевой синдром => ошибки есть, нулевой => ошибок нет.

Находим столбец проверочной матрицы, который равен S, если такой присутствует, значит ошибка выявлена верно и ошибочным является символ на позиции с номером столбца.

**Исправление ошибки**

В результате декодирования синдрома получается вектор ошибки E

Исправление ошибочного бита достигается простым сложением по модулю 2 вектора Е и кодового слова Yn:



**Пример**

Есть кодовое слово Yn = 1001**0**00 (ошибочный символ - 3 слева)

Yr = 000

Y’r1 = 0\*1 xor 1\*0 xor 1\*0 xor 1\*1 xor 1\*0 xor 0\*0 xor 0\*0 = 1

Y’r2 = 0

Y’r3 = 0

=> Y’r = 100

S = Yr xor Y’r = 000 xor 100 = 100 => есть ошибка

Ищем в проверочной матрице столбец 100

****

Столбец h5 = S = 100 => ошибка на 5 позиции кодового слова => E = 0000100

Исправляем ошибку:

Yn = E xor Yn = 0000100 xor 1001000 = 1001100

# 17.  Код Хемминга с минимальным кодовым расстоянием dmin=3.

Код Хемминга относится к классу линейных блочных кодов.

**Линейные блочные коды** – это класс кодов с контролем четности, которые можно описать парой чисел (n, 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 приведенно-ступенчатой форме. Кодовые слова являются линейными комбинациями строк матрицы G (кроме слова, состоящего из нулевых символов).

Кодирование заключается в умножении вектора сообщения Хk длиной k на порождающую матрицу по правилам матричного умножения (все операции выполняются по модулю 2). Очевидно, что при этом первые k символов кодового слова равны соответствующим символам сообщения, а последние r символов образуются как линейные комбинации первых.

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.

Матрица Н, называемая *проверочной*, равна

https://lh7-us.googleusercontent.com/3lFVQBghOEb3uleE5W5EqfLDV5kOXlXMSQvKQUZ151cestAiParpm-R3NnE-tx3RIp4cvawUPcS5jtfhccicNQKxFMbVVOI57I6dWfREA7_CctHUwG6VW1mEvf1HgJmtIon53rlCz87_QrEOW4vLwA

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.

В последнем выражении I – единичная матрица порядка r (r×r).

Количество r избыточных (проверочных) символов кодового слова определяется из следующей простой логической цепи рассуждений.

Общее число всех возможных комбинаций 2^r должно удовлетворять неравенству

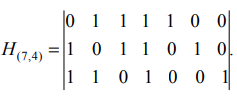
https://lh7-us.googleusercontent.com/RoWB988ROW6CgkOIGWCfxeBQHNeQre3NIGwIw5XOjeNYoYh64fmA97OUb0sRA7Ay5zcBEn48US8U59aE9rNpWwyLUsxZHrQIP0PiYcbnS2g_D1hgOjXOcnOlQ35CYPhOh0lhw0MiMXETMViXRISDvA

в силу того, что

https://lh7-us.googleusercontent.com/jPaWdxw7mHfLrZLgBKmz05Zw0pvsM2A6oXJfjKc4pn-9RrgBapiHYYCP7ZJeIiRiZCNJaGqz1MjbERRbMLKrpBX0d_HOgxA7gVjR5xa2OZP9_Nyc2rCVmAqiiR0BbGMXMpeH4oxEKOPQwzXhyYcMiA

Присутствие цифры «1» в приведенных выражениях соотносит ее с нулевым вектор-столбцом, который в матрице не используется.

Например, для (7, 4)-кода Хемминга проверочная матрица в упорядоченном виде выглядит так:



Результат умножения сообщения на выходе канала передачи (Yn) или (что равнозначно) сообщения, считываемого из памяти, на проверочную матрицу (Н) называется **синдромом (вектором ошибки) S:**

https://lh7-us.googleusercontent.com/iRsC9to_SoaPhllf2N9p3jg4cAoKzYuR6wCZ6iOrXTd_SALnbBtywhvuKBasyLMVMZySWy6sDiFytWpsXNItSqddxY77_Ask_gaC9zmKnZ9fvEuyu-lL-sQde_ihAvOwuHTKIthxfp-iWswqkBHlow

где Yn = y1, y2, …, yn – принятый вектор (сообщение на выходе канала), полученный после передачи либо считывания из памяти. Вектор Yn обычно представляют в следующем виде:

https://lh7-us.googleusercontent.com/Bzlh3ry8rcGedkMvIg27gmdlqSQLR6-drxrBWX94yqgkaZho2CgvwukbujxEC1vFWFr87Icin-TR43oHURa6VwTWv2xZH0WAPDr1lh3HXT8iqdS38ks7zQlN8gw4GAHbn5-y2165GZSgXKuOVvvKow

где Еn = е1, е2, …, еn – вектор ошибки.

Синдром – это результат проверки четности, выполняемой над сообщением Yn для определения его принадлежности заданному набору кодовых слов. При положительном результате проверки синдром S равен 0, т. е. Yn = Хn. Если Yn содержит ошибки, которые можно исправить, то синдром имеет определенное ненулевое значение, что позволяет обнаружить и исправить конкретную ошибочную комбинацию.

**Важно запомнить, что в силу этих выражений ненулевой синдром всегда равен сумме по модулю 2 тех векторстолбцов матрицы Н, номера которых соответствуют номерам ошибочных битов в слове Yn.**

# 18. Код простой четности. Особенности программной реализации

Какие особенности реализации???? программные

Простейший избыточный код;

Основан на контроле четности (либо нечетности) единичных символов в сообщении.

Количество избыточных символов r всегда равно 1 и не зависит от k.

Значение этого символа будет нулевым, если сумма всех символов кодового слова по модулю 2 равна нулю – при контроле четности.

Проверка четности – очень простой метод для обнаружения ошибок в передаваемом пакете данных. С помощью данного кода мы не можем восстановить данные, но можем обнаружить только лишь одиночную ошибку.

В каждом пакете данных есть один бит четности, или, так называемый, паритетный бит. Этот бит устанавливается во время записи (или отправки) данных, и затем рассчитывается и сравнивается во время чтения (получения) данных. Он равен сумме по модулю 2 всех бит данных в пакете. То есть число единиц в пакете всегда будет четно . Изменение этого бита (например с 0 на 1) сообщает о возникшей ошибке.

Пример:

Начальные данные: 1111

Данные после кодирования: 11110 ( 1 + 1 + 1 + 1 = 0 (mod 2) )

Принятые данные: 10110 (изменился второй бит)

Как мы видим, количество единиц в принятом пакете нечетно, следовательно, при передаче произошла ошибка.

Как говорилось ранее, этот метод служит только для определения одиночной ошибки. В случае изменения состояния двух битов, возможна ситуация, когда вычисление контрольного бита совпадет с записанным. В этом случае система не определит ошибку, а это не есть хорошо. К примеру:

Начальные данные: 1111

Данные после кодирования: 11110 ( 1 + 1 + 1 + 1 = 0 (mod 2) )

Принятые данные: 10010 (изменились 2 и 3 биты)

В принятых данных число единиц четно, и, следовательно, декодер не обнаружит ошибку.

Так как около 90% всех нерегулярных ошибок происходит именно с одиночным разрядом, проверки четности бывает достаточно для большинства ситуаций.

особенности программного кода

function addParityBit(data) {

    // Подсчитываем количество единичных битов

    let countOfOnes = 0;

    for (let bit of data) {

        if (bit === '1') {

            countOfOnes++;

        }

    }

    // Определяем бит четности (четная схема)

    const parityBit = (countOfOnes % 2 === 0) ? '0' : '1';

    // Добавляем бит четности к данным

    return data + parityBit;

}

function checkParityBit(dataWithParity) {

    // Подсчитываем количество единичных битов

    let countOfOnes = 0;

    for (let bit of dataWithParity) {

        if (bit === '1') {

            countOfOnes++;

        }

    }

    // Если количество единичных битов четное, данные корректны

    return countOfOnes % 2 === 0;

}

// Пример использования

const data = "1011001";  // Исходные данные

const encodedData = addParityBit(data);  // Кодируем данные, добавляя бит четности

console.log(`Encoded data: ${encodedData}`);

// Проверяем данные с битом четности

const isDataCorrect = checkParityBit(encodedData);

console.log(`Is data correct? ${isDataCorrect ? 'Yes' : 'No'}`);

// Пример с ошибкой

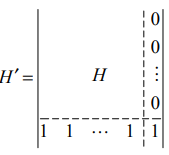
const encodedDataWithError = "10110011";  // Добавлена ошибка в последний бит

const isDataCorrectWithError = checkParityBit(encodedDataWithError);

console.log(`Is data with error correct? ${isDataCorrectWithError ? 'Yes' : 'No'}`);

# 19.  Код Хемминга с минимальным кодовым расстоянием dmin=4.

Корректирующая способность кода Хемминга с dmin = 3 может быть увеличена введением дополнительной проверки на четность. В этом случае проверочная матрица будет иметь вид



Так, минимальное кодовое расстояние такого кода будет равно 4: dmin = 4. Такой код может исправлять все единичные ошибки с одновременным обнаружением всех двойных в анализируемом кодовом слове.

При этом нужно помнить, что вид матрицы не соответствует ее каноническому представлению, поскольку во всех столбцах единичной матрицы, кроме последнего, будет по 2 единицы.

Для придания матрице канонического вида необходимо сложить посимвольно все строки между собой и результат сложения записать в последнюю строку (под горизонтальной линией).

# 20.  Составной код. Итеративные коды.

Укоротить? Многомерные итеративные коды?

мокар: бля ну орлов говна сложного тут поналивал кншн. в конце будет текст спизженный из интернетов

Определенно укоротить

Я хз кто вы потомки и кто вы проотцы, но я взялся за укорочение ( на данный момент билет составляет 3.5 страницы)

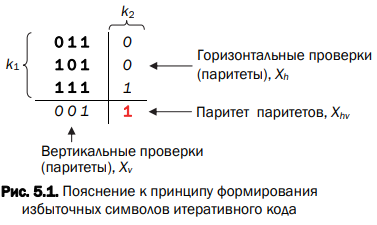
Итеративные коды относятся к классу кодов произведения. Кодом произведения двух исходных (базовых) помехоустойчивых кодов называется такой многомерный помехоустойчивый код, кодовыми последовательностями которого являются все двумерные таблицы со строками кода (k1) и столбцами кода (k2).

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**.

Основное достоинство рассматриваемых кодов – простота. Основной недостаток – сравнительно высокая избыточность.

В упомянутой двумерной матрице кодовые слова записываются в виде таблицы. Проверочные символы вычисляются исходя из того, что строки и столбцы должны содержать четное (нечетное) число единиц. Например, при кодировании информационного слова Хk = **011101111** с помощью таблицы с четностью по строкам и столбцам получим избыточные символы Хr = Xh, Xv, Xhv = 0010011



В соответствии с рис. 5.1 кодовое слово будет иметь следующий вид: Xn = **011101111**0010011. Как видно, избыточные символы (называемые также паритетами) в приведенном кодовом слове в принятом порядке (Xh, Xv, Xhv) записываются сверху вниз, справа налево. Возможен обратный или иной порядок. Важно только, чтобы при декодировании сообщения использовался аналогичный порядок следования паритетов. Символ Xhv (паритет паритетов) равен сумме по модулю 2 символов информационного слова Xk, а также проверочных символов Xv и Xh. *(кол-во единиц = 9).*

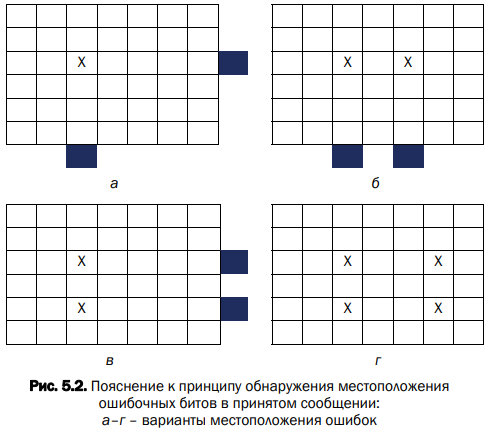
Поскольку двумерная матрица формируется как комбинация двух кодов простой четности (по каждому измерению), каждый из которых характеризуется минимальным кодовым расстоянием dmin = 2, то полученный итеративный код (r = k1 + k2) будет характеризоваться минимальным кодовым расстоянием, равным произведению dmin по строкам и по столбцам, т. е. 4.

Использование символа Xhv обеспечивает минимальное кодовое расстояние такого итеративного кода dmin (r = k1 + k2 + 1) на единицу больше. В этом легко обнаруживается сходство кода с кодом Хемминга при dmin = 4.

Передачу символов кода обычно осуществляют последовательно символ за символом, от одной строки к другой, либо параллельно целыми строками.

Как показано на рис. 5.1, проверочный символ есть свертка по модулю 2 информационных символов, записанных в соответствующие строку или столбец матрицы.

Декодирование начинают сразу, не ожидая поступления всего блока информации. Проверка соответствия избыточных символов полученного слова (Yr = Yh, Yv, Yhv либо Yr = Yh, Yv) при декодировании позволяет обнаружить любое нечетное число искаженных символов, расположенных в одной строке или в одном столбце. Формально такое декодирование осуществляется сравнением принятых (Yh, Yv, Yhv) и вновь вычисленных (Y’h, Y’v, Y’hv) для полученного слова паритетов. В упрощенной форме это показано на рис. 5.2. Определение местоположения одиночной ошибки по строке указывает на наличие ошибки в этой строке матрицы, а проверка по столбцу – конкретный символ (рис. 5.2, а).



Однако этим кодом не могут быть установлены местоположения многократных ошибок, имеющих четное число искаженных символов как по строкам, так и по столбцам (рис. 5.2, б, в). Простейшая необнаруживаемая ошибка содержит четыре искаженных символа, расположенных в вершинах прямоугольника или квадрата (рис. 5.2, г). Это происходит из-за того, что четность (паритет) по строкам и по столбцам матрицы не нарушается.

Я б это вообще нахуй убрал текст снизу потому что по-моему и так всё понятно, но если интересно просто шрифт больше поставьте

**из интернетов**

Итеративные коды широко применяются в системах передачи данных из-за простоты реализации. Код получают путем представления последовательности информационных символов в виде двумерного массива. Затем каждая строка массива и каждый столбец кодируются некоторым кодом, причем необязательно одним и тем же.

|  |  |
| --- | --- |
| Информационные символы | Проверочные символы по строкам |
| Проверочные символы по столбцам | Проверка проверок |

**Кодовое** **расстояние** **dmin** такого кода равно произведению кодовых расстояний используемых итерируемых кодов.

Рассмотрим итеративный код с проверкой на четность по каждой строке и каждому столбцу. Кодовые комбинации информационных символов представлены в коде ASCII.

|  |  |
| --- | --- |
| 1001010  0111010  11**1**00**0**1  1000111  00**1**10**0**1 | 1  0  0  0  1 |
| 1011111 | 0 |

Код с проверкой на четность имеет кодовое расстояние dmin=2 , соответственно для итеративного кода **dmin=4** . Полученный итеративный код исправляет одиночные ошибки и обнаруживает любые комбинации ошибок нечетного веса. Ошибки четного веса в пределах одной строки обнаруживаются при помощи проверок по столбцам. Четное число ошибок в пределах одного столбца обнаруживается при помощи проверок по строкам. **Не** **обнаруживается** любой набор из 4-х ошибок, образующих прямоугольник.

# 21.  Оценка эффективности использования корректирующего кода в ИВС

**ОТВЕТ ИЗ НЕЙРОСЕТИ**

Оценка эффективности использования корректирующего кода в ИВС (информационных вычислительных системах) осуществляется по следующим критериям:

1. Надежность передачи и хранения данных. Использование корректирующего кода позволяет значительно увеличить надежность передачи и хранения данных. Критерием надежности является вероятность ошибки при передаче данных и вероятность потери данных при хранении.

2. Ресурсоемкость корректирующего кода. Применение корректирующего кода увеличивает объем информации, передаваемой или хранимой системой. Для оценки эффективности корректирующего кода необходимо учитывать потребление ресурсов, таких как объем памяти, процессорное время и пропускная способность канала связи.

3. Скорость передачи данных. Применение корректирующего кода может увеличить время передачи данных. Поэтому для оценки эффективности корректирующего кода необходимо учитывать скорость передачи данных.

4. Сложность реализации. Реализация корректирующего кода может быть достаточно сложной, и в зависимости от выбранного кода требовать различных вычислительных алгоритмов. Для оценки эффективности корректирующего кода необходимо учитывать сложность реализации и возможность интеграции в ИВС.

При выборе корректирующего кода в ИВС необходимо учитывать как технические, так и экономические критерии, чтобы достичь оптимального баланса между надежностью, ресурсоемкостью, скоростью и сложностью реализации.

# 22.  Особенности использования перемежителей в ИВС

**Перемежитель** - устройство (реализовано аппаратно) или программное средство, которое определенным образом перемешивает (меняет местами) символы передаваемого сообщения (или кодового слова).

Разработка и использование перемежителей в ИС – необходимость разнесения расположенных рядом (сгруппированных) ошибок в сообщении («размазать» ошибки по сообщению) с целью упрощения и сокращения во времени процедуры исправления таких ошибок сравнительно простыми кодами.

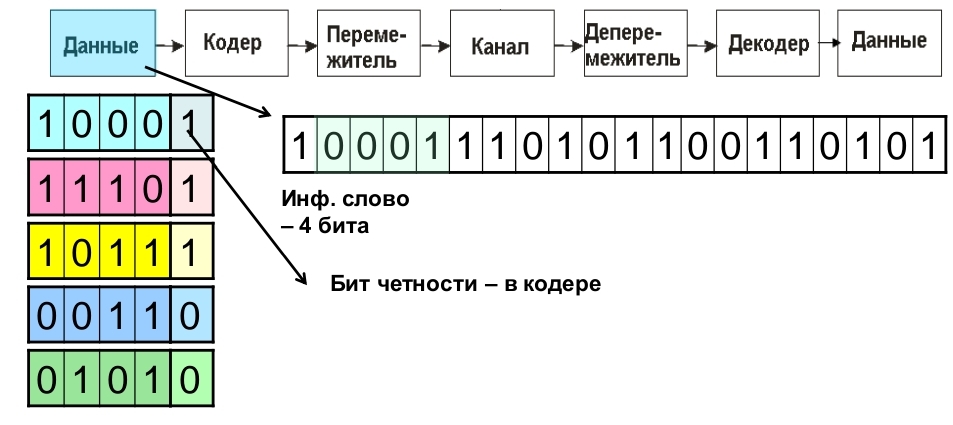
**Глубина перемежения** - максимальное расстояние – в битах, - на которое разносятся соседние символы входной последовательности

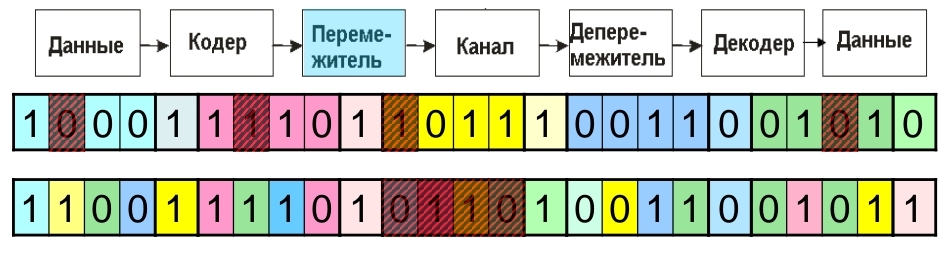
Процедура перемежения/деперемежения состоит в перестановке символов кодового слова и восстановлении исходной последовательности после передачи ее по каналу

Больше глубина перемежения => больше задержка

**Длина** передаваемой последовательности **не меняется**





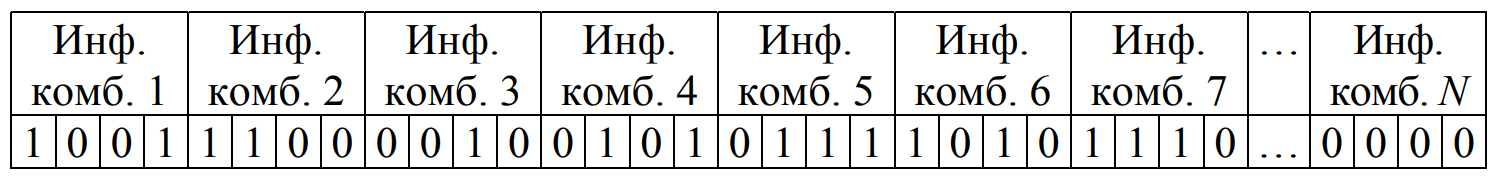






**Пример (исп-е кода Хемминга и блокового перемежителя)**

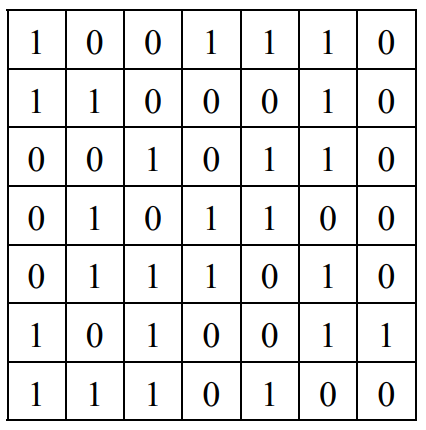
Xk = 1001 1100 0010 0101 0111 1010 1110



Для каждого 4-х битного значения необходимо вычислить избыточные символы

Строим матрицу перемежения, записывая в строки все кодовые слова Xn (присоединили уже Xr к Xk)

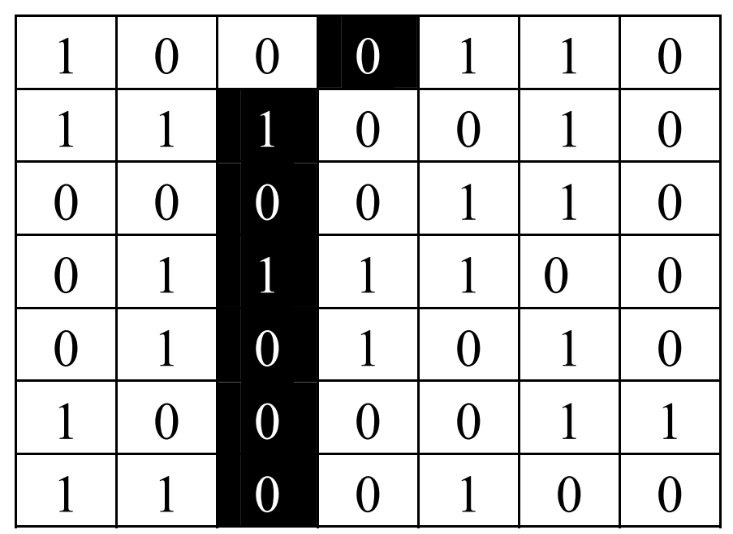
Матрица перемежения:



Символы считываются по столбцам, таким образом, Xn = 1100011 0101101 0010111 1001100 1011001 1110110 0000010

Пусть сообщение передано с ошибкой Yn = 1100011 0101101 0101000 0001100 1011001 1110110 0000010

Матрица деперемежения:



Далее сообщения считываются по строкам

Y’n = 1000110 1110010 0000110 0111100 0101010 1000011 1100100



Выделяем из кодовых слов избыточные символы и также заново высчитываем новые наборы избыточных символов, чтобы определить синдром

Благодаря синдрому определим номер столбца проверочной матрицы, в котором ошибка, а тем самым и вектор ошибки, чтобы исправить все ошибки.

# 23. Совместное использование перемежителей/деперемежителей и корректирующих кодов в ИВС

Пример. Процесс передачи информации с использованием кода Хемминга и блокового перемежителя. Информационный поток на входе кодера Хемминга (7,4) имеет вид:









# 24. Математические основы построения и использования циклических кодов

Относятся к классу линейных, систематических,

Сумма по модулю 2 двух разрешённых кодовых комбинаций даёт также разрешённую кодовую комбинацию

Каждый вектор (кодовое слово), получаемый из исходного кодового вектора путём циклической перестановки его символов, также является разрешённым кодовым вектором, при циклической перестановке символы кодового слова перемещаются слева направо на одну позицию:

**Пример 1**. Если кодовое слово - 1101100, то разрешённой кодовой комбинацией будет и - 0110110

Принято описывать циклические коды (ЦК) при помощи порождающих полиномов G(X) степени r = n — k, где r — число проверочных символов в кодовом слове

**Пример 2.** Переведём кодовое слово Хn = 101100 в полиномиальный вид:

Вi(X) = 1 \* X5 + 0 \* X4 + 1 \* X3 + 1 \* X2 + О \* X1 + О \* Х0 = X5 + X3 + X2

•Операции кодирования и декодирования ЦК сводятся к известным процедурам умножения и деления полиномов

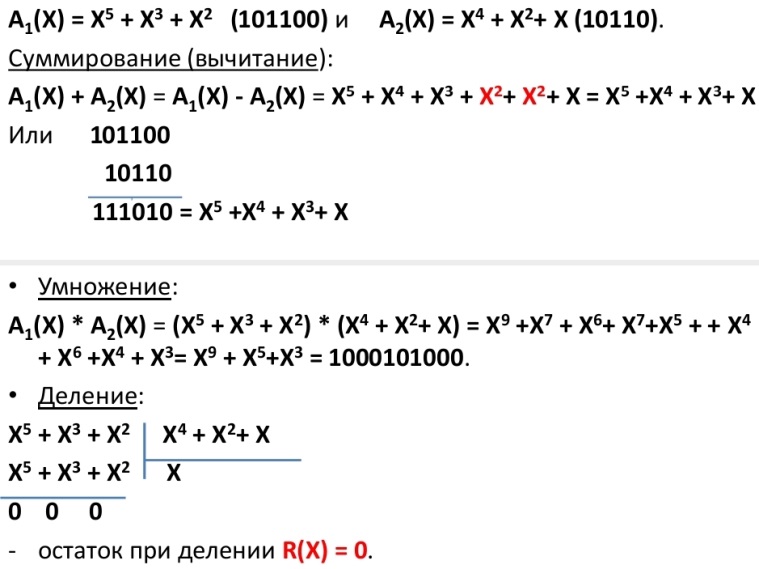
Действия с кодовыми словами в виде полиномов производятся по правилам арифметики по модулю 2 (вычитание равносильно сложению).

**Пример 3.** Из равенства Хn -1 =0 получаем Хn =1. Прибавив к левой и правой частям по единице, имеем Хn + 1 = 1 + 1= 0. Таким образом, вместо двучлена Хn -1 можно ввести бином Хn +1 или 1 + Хn, из чего следует, что

Хn + Хn = Хn

(1 + 1) = 0

• Приведём далее порядок суммирования (вычитания), умножения и деления полиномов (по модулю 2). В примерах используем вышеприведённые кодовые комбинации



**При циклическом сдвиге вправо на один разряд необходимо исходную кодовую комбинацию поделить на X, а умножение на X эквивалентно сдвигу влево на один символ**

# 25.  Кодирование сообщения циклическим кодом

Кодовое слово состоит из двух частей: информационные  Хk(Аi(Х)) и проверочные Хr(Ri(X)) символы. Число проверочных символов равно r. В младшие биты кодового слова, которые отвечают за проверочные символы, необходимо записать изначально нули, которые далее будут заменены на проверочные символы.

n - длина кодового слова

k - длина информационного слова

r - длина избыточного слова

1. На основе n, k, r необходимо выбрать порождающий полином G(X).
2. Далее необходимо привести информационное слово Xk к полиномиальному виду.
3. Выполняем вычисление проверочных символов на основе **метода деления полиномов** и их присоединяем

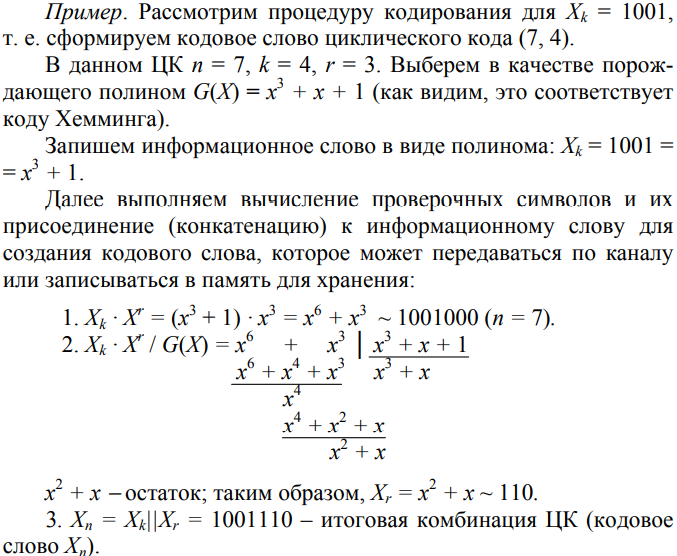
**Хk ·X^r**

1. Делим получившееся значение на порождающий полином

**Хk · X^r / G(X)**

1. Получаем итоговую комбинацию

**Xn = Хk||Хr**



# Матрицы циклического кода и кода Хемминга

**Ну тут пиздец какой-то надо добавить**

**Порождающая матрица** (G-матрица) циклического кода строится из сдвигов генерирующего многочлена. **Проверочная матрица** (H-матрица) используется для выявления ошибок.

Для кода Хемминга H-матрица состоит из всех ненулевых бинарных векторов длины 𝑟*r* (где 2𝑟≥𝑛+12*r*≥*n*+1, 𝑛*n* — длина кодового слова).

# 27. Синдромный метод декодирования циклического кода

**Основная операция:** принятое кодовое слово (Yn) нужно поделить на порождающий полином, который использовался при кодировании (G(X)).

Если остаток нулевой, то Yn принадлежит коду (слово не искажено помехами).

Если ненулевой (Yn ≠ Хn) => наличие ошибки в принятой кодовой комбинации.

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

После передачи по каналу с помехами кодовое слово можно записать:

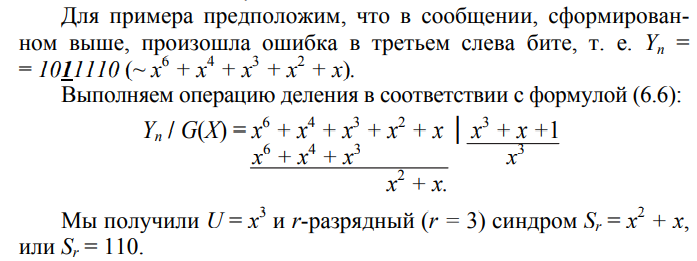
**Yn = Xn + Еn** или **Xn = Yn + Еn**

Деление принятого кодового слова на G(X)):

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

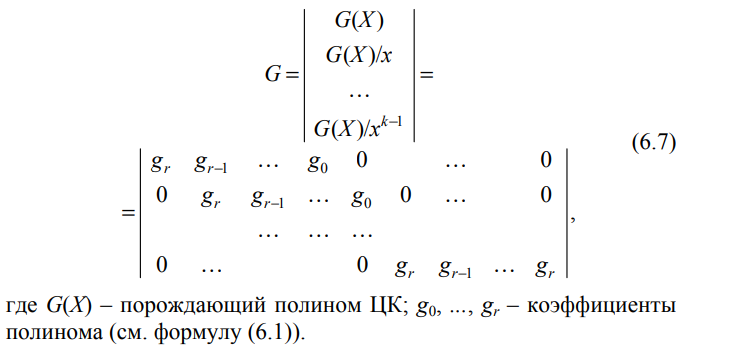
где **Sr** − остаток от деления (Yn) / (G(X)), или **синдром**.

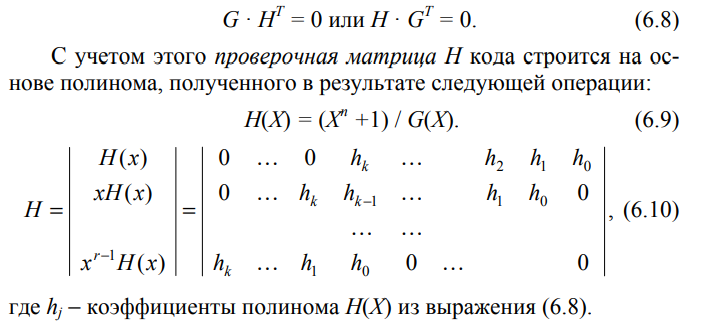
**Всякому ненулевому синдрому соответствует определённое расположение ошибок**



Цель декодирования ненулевого синдрома - определение ошибочного бита в принятом сообщении (определение вектора Е).

Порождающая матрица G циклического кода имеет в качестве строк векторы:

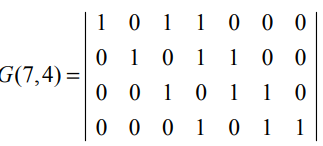




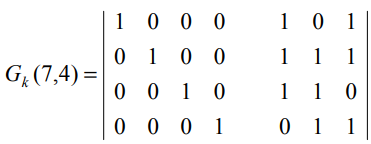
Построение порождающей матрицы:

1. первой строкой матрицы записывается порождающий полином и r нулей для резервирования места на проверочные символы
2. циклический сдвиг первой строки вправо

 Обычно кол-во строк матрицы == k

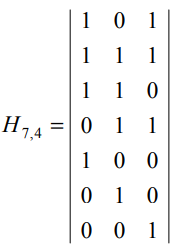


Далее необходимо преобразовать матрицу к каноническому виду.

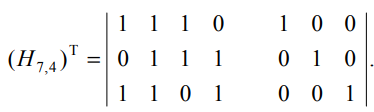


Канонический вид подразумевает, что матрица состоит из единичной подматрицы размерностью k и подматрицы Р размерностью k×r.

Проверочная матрица Н размерностью r×n может быть получена из порождающей матрицы канонического вида путем дополнения проверочной подматрицы единичной матрицей размерности r×r:



Канонический вид проверочной матрицы:



Ненулевой синдром всегда равен сумме по модулю 2 тех вектор-столбцов матрицы Н, номера которых соответствуют номерам ошибочных битов в слове Yn.

**Xn = Yn + Еn**

# 28.  Характеристика надежности двоичного канала передачи при использовании кодов

Двоичный симметричный канал связи характеризуется тем, что вероятность:

https://lh7-us.googleusercontent.com/NeaRrm-fK0Sow_sJ9NaRbOMVE8QciIJ7sedO69zH_sw32TpLIJK5vwIjdMut-B2Tag4l6FtgthiK1AnWDxc_w5AGmIafF8d3t-Vo6tqL5tpP_jQqyDp5SwwBqm9iAC7DuaKMIySEZMKriFrpnYxxyA

С равной вероятностью ошибка может произойти в произвольном символе кода.

p(1|0)=p(0|1)=p — условная вероятность p(1|0)+p(0|1)=1

На практике, однако, каналы характеризуются различными вероятностями, т.е.:

https://lh7-us.googleusercontent.com/chOW5CniRuhdJkQbV5CX_Hk-No4jMCm1lTonZ3e64LPVdbA4ndmbSYZFaQ0MghTAqnT_gy9297xTLrld-jza-yoQEAWPK3rSsx1TbTqSMFYBZrytHj8JyyVFFt20zrevvXJeuAidYIQ4C-IJGmjmZw

**Качество канала** оценивается по вероятности полного совпадения переданного и полученного сообщения (p(x=y)=1). Возможна оценка канала по двум типам передачи:  
1 Без кодирования.  
2 C кодированием, свойства канала не изменяются.

Сравнивая вероятности без кодирования и с использованием кодирования можно оценить **эффективность использования кода**.

Пример:В реальных каналах передачи p=10^-3…10^-7. Пусть в нашем канале связи p=10^-3, k=512 bit. Оценим эффективность кода.

В соответствии с формулой для канала без использования кода p(Xk=Yk)=(1-0,001)^512.При использовании кода r=log2(512)+1=10 =>n=522. Сравнивая эти полученные рез-ты, мы получаем оценку эффективности.

Суммируя все по первому методу преобразования информации:

https://lh7-us.googleusercontent.com/rHCso8FLrl-n8VJoKvvGA4Ua1yFc2uvRq1H2jcbxT7gBnahqa5pVQoD2RP8mwGLKsLSSOSMt24cuu2AFhjzVtSibXeTlv8i_o3r9wfp3rAiUIQN4cLMiGeVCmGuCAJRjAAVb5694njfqa6V_M6xW8A

Всегда приводит к удлинению передаваемой последовательности n>k

Главное назначение — повышение уровня надежности канала передачи или устройства хранения информации.

# 29. Общая характеристика и классификация методов сжатия информации.

Большинствo типов данных хар-ся избыточностью:

Степень избыточности видеоданных > степени избыточности графических данных > степень избыточности текстовых данных

Фактором, влияющим на степень избыточности является принятая система кодирования

**Избыточность** - центральное понятие в теории сжатия информации. Любые данные с избыточной информацией можно сжать. Данные, в которых нет избыточности, сжать нельзя

Теорема Шеннона об источнике шифрования показывает, что невозможно сжать данные настолько, что оценка кода (среднее число бит на символ) станет меньше, чем энтропия Шеннона исходных данных, без потери точности информации.

***По степени соответствия входных и выходных данных***

* ***С потерями информации***
  + JPEG - для графических данных;
  + MPG - для для видеоданных;
  + MP3 - для аудиоданных.
* ***Без потерь информации***
  + GIF, TIFF - для графических данных;
  + AVI - для видеоданных;
  + ZIP, ARJ, RAR, CAB, LH - для произвольных типов данных

1. ***По способу кодирования (преобразования)***

* На основе выявления повторяющихся последовательностей  данных и замены их более простой структурой (интервалов, RLE, Б-У) - **символориентированные**
* Сжатие по **ключевым словам**: кодирование лексических  единиц группами байт фиксированной длины (ЛЗ)- **словарные**
* На основе использование частотных (вероятностных) свойств  символов входного алфавита (фмана, Шеннона-Фано)- **Вероятностные (статистические)**
* **Арифметические**

# 30. Метод сжатия данных Лемпеля-Зива.

Классический алгоритм Лемпеля − Зива – LZ77, названный так по году представления метода, формулируется следующим образом: «если в проанализированном (сжатом) ранее выходном потоке уже встречалась подобная последовательность байт, причем запись о ее длине и смещении от текущей позиции короче, чем сама эта последовательность, то в выходной файл записывается ссылка (смещение, длина), а не сама последовательность».

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

Пусть к текущему моменту времени закодировано t символов: S1, S2, …, St. Тогда словарём будут являться n1 предшествующих символов St-(n1-1), St-(n1-1)+1, …, St.

В буфере находятся ожидающие кодирования (сжатия) символы St+1, St+2,.., St+n2. Если n2>= t, то словарем будет являтся все уже обработанная часть входной последовательности.

**!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. Длина совпадения не должна превышать размера буфера. Полученная в результате поиска фраза St-(p-1), St-(p-1)+(q-1) кодируется с помощью двух чисел:

1. смещения от начала буфера p;
2. длины соответствия, или совпадения q.

Cсылки (p и q - указатели) однозначно определяют фразу. Дополнительно в выходной поток записывается символ s, следующий за строкой буфера.

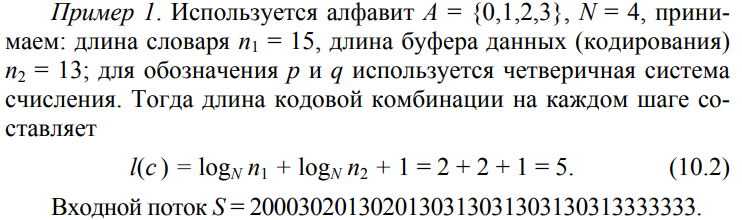
 Длина кодовой комбинации (триады - p,q,s) на каждом шаге определяется соотношением:

https://lh7-us.googleusercontent.com/7ORQkkzrvfCa8iKAsXAQheXjfHJTxKhB2ZnqgrwLHU0mhOQzV9n7zLMEsBCVwQfMJRdXVzDQRFp_2vnYQz7w0bv1Dq8JxwP2lmXkNV9f4CbT71d_GOoUEaeSZlUoBsEMEMTjRlYEah2GZfXeOZH9fA

N- мощность алфавита.

После каждого шага окно смещается на q+1 символов вправо и осуществляется переход к новому циклу кодирования. Величина сдвига объясняется тем, что мы реально закодировали именно q+1 символов: q- с помощью указателя и 1 - с помощью тривиального копирования.

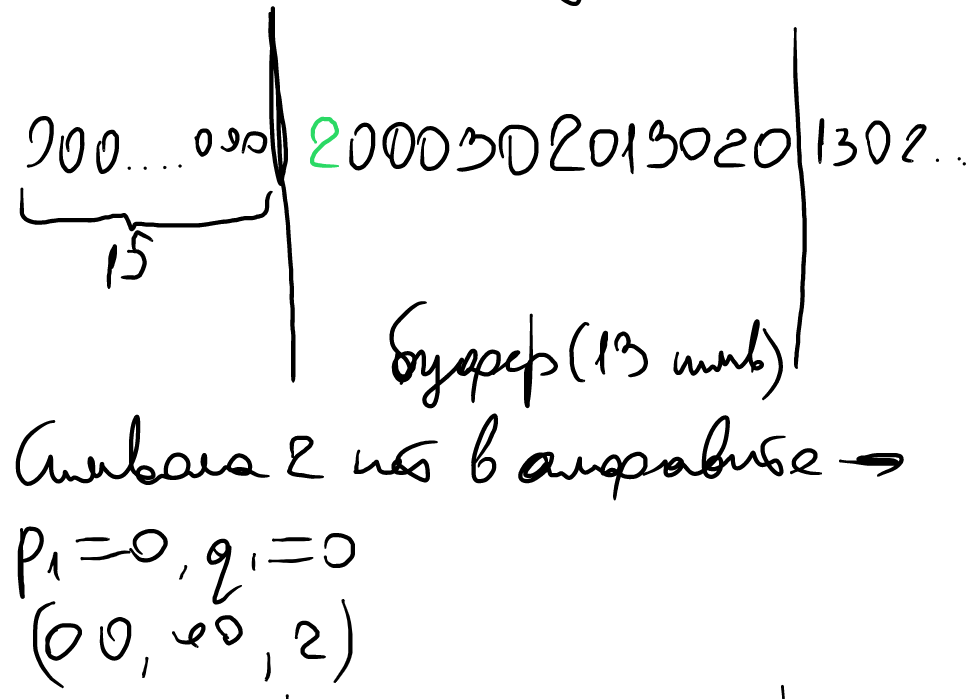
 Передача одного символа в явном виде s позволяет разрешить проблема обработки еще ни разу не встречавшихся символов, но существенно увеличивается размер сжатого блока.



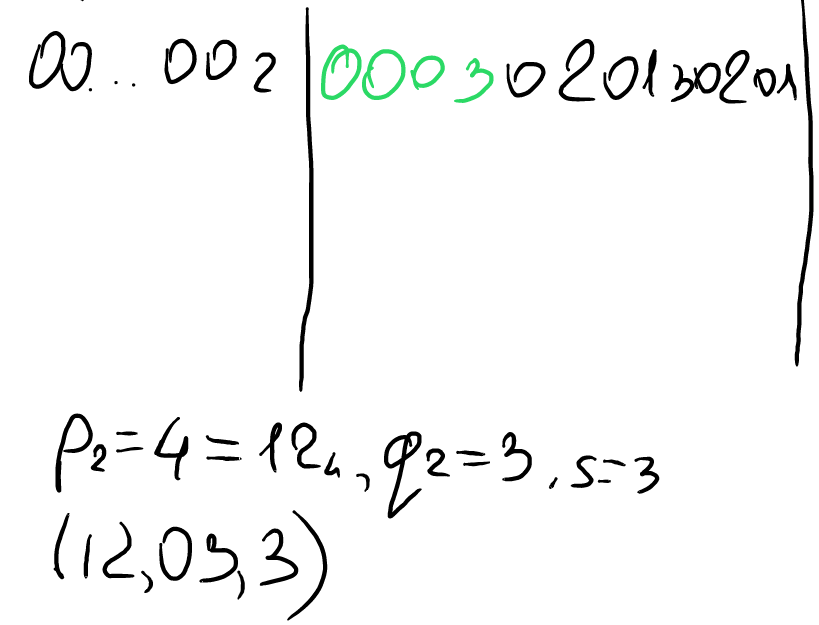
Прямое преобразование

1 шаг.

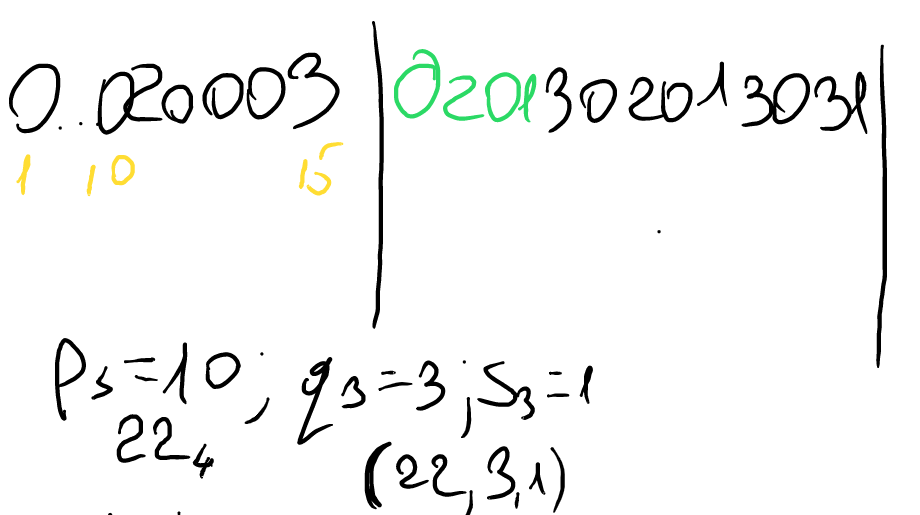
Сначала заполняем алфавит нулями или другими символами по желанию (по ситуации)



2 шаг.

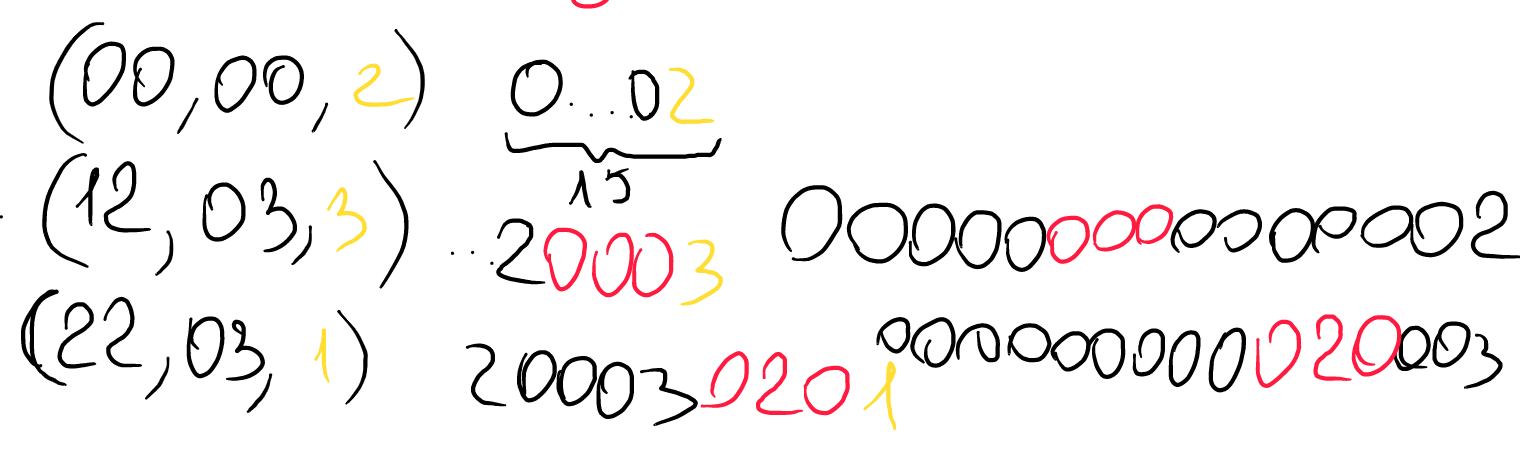


3 шаг.

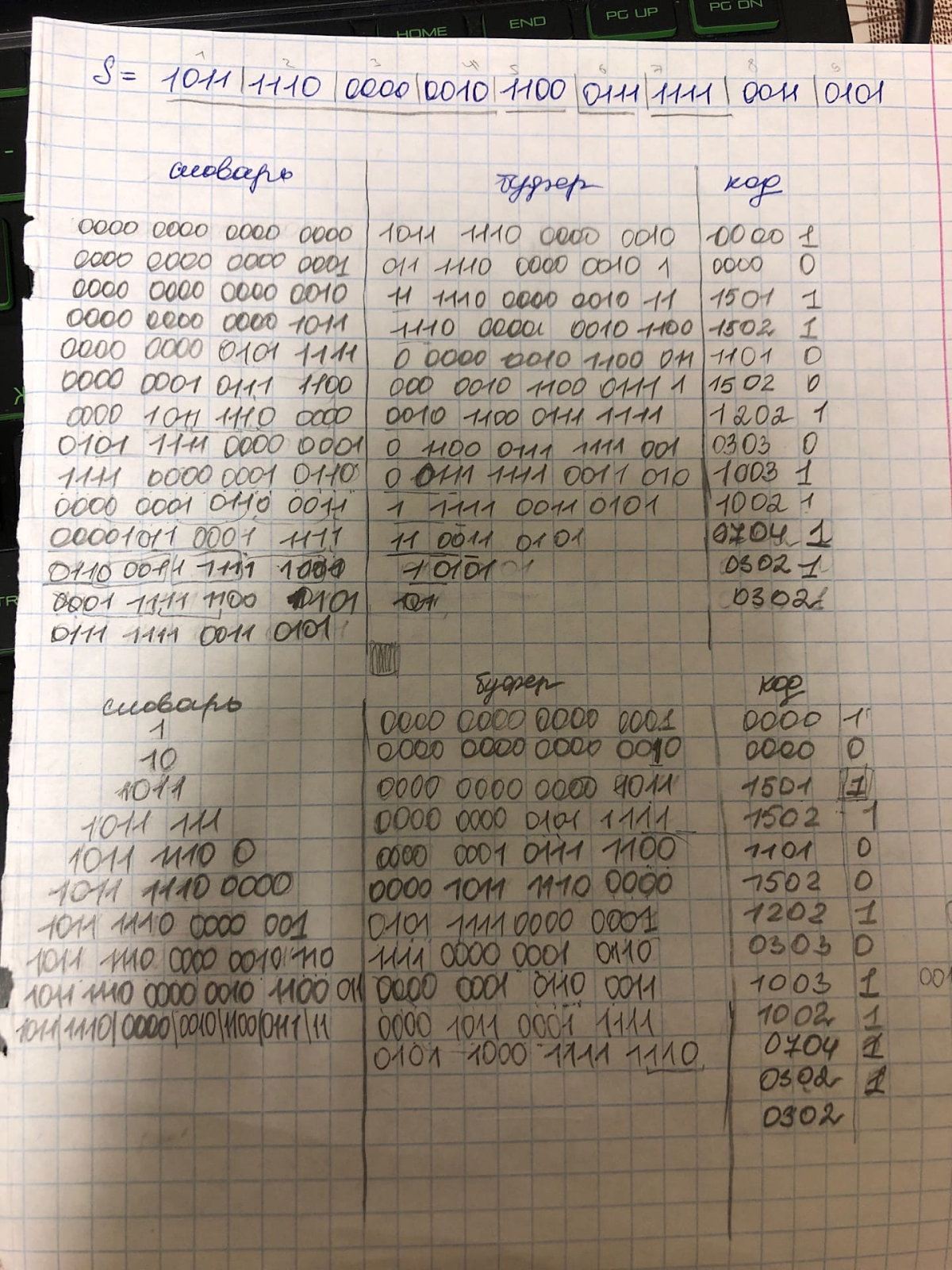


И так далее

Обратное преобразование



**Мокар:** пример орлова говнище



## 31 Метод сжатия данных Барроуза-Уиллера.

Сам по себе, **BWT** не является классическим алгоритмом  сжатия, однако выходная последовательность гораздо  ***удобнее для сжатия***, нежели исходная.

Преобразование Б-У применяется в алгоритмах сжатия  качественных данных.

Является компромиссным между быстрыми словарными  алгоритмами и статистическими алгоритмами.  Обычно применяется **совместно** со статистическими  алгоритмами.

Оперирует целым блоком данных длиной **k** символов

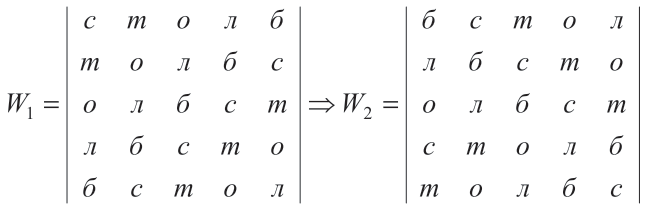
***Прямое преобразование***:

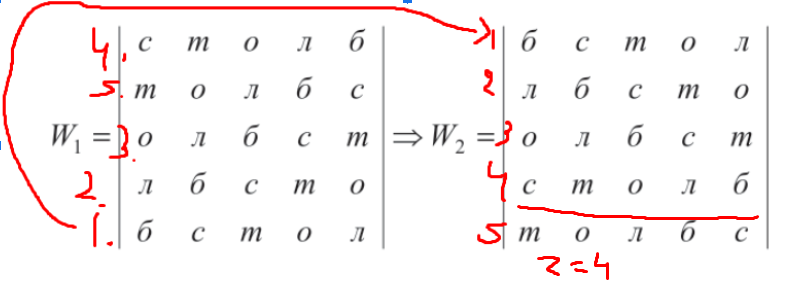
1) выделяется блок данных (строка длиной k символов некоторого алфавита мощностью N), который обозначим символом М;

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. Как видим, выходная строка (сжатое сообщение) всегда по объему превышает входную.





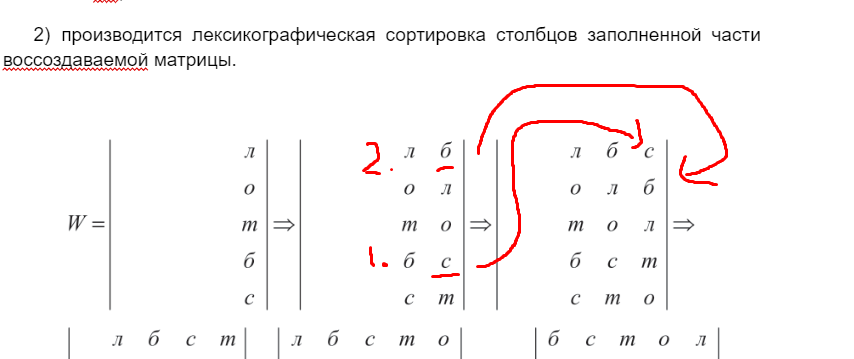
(z может соответствовать нескольким строкам (т.е несколько одинаковых строк), это нормально - нас устроит **одна любая строка**)

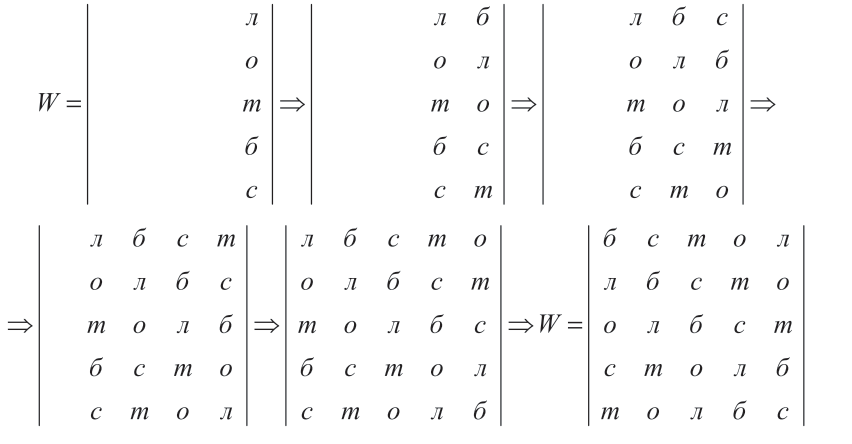
***Обратное преобразование:***

Итак, входной для обратного преобразования является информация вида BWT(М), i. Это преобразование заключается в выполнении k одинаковых шагов, каждый из которых состоит из  2 операций, с целью воссоздания матрицы W2:

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

2) производится лексикографическая сортировка столбцов заполненной части воссоздаваемой матрицы.





Главной проблемой в реализации BWT является выбор быстрого алгоритма сортировки данных с большой длиной ключа.

Чем меньше мощность алфавита, тем эффективнее метод сжатия по BWT. Используется в архиваторах класса ZIP совместно с другими методами (LZ).

## 32. Метод сжатия данных Хаффмана

**Статистические алгоритмы**

Статистические алгоритмы позволяют создавать более короткие коды для часто встречающихся и более длинные – для редко встречающихся символов алфавита или конкретного сообщения.

В первом случае метод считается **статическим** статистическим, во втором – динамическим статистическим: вероятностные свойства символов подсчитываются для **конкретного** сообщения или **потока данных** *(т.е. обучаем на каком-то большом тексте например)*.

Частота или вероятность появления того или иного символа алфавита в произвольном сообщении, лежащая в основе алгоритмов, дали название этим алгоритмам и соответствующим методам. Иногда эти методы называют также **префиксными**.

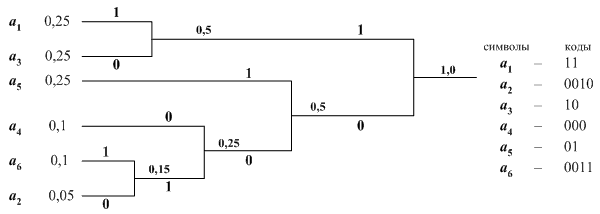
Сжимают **без потерь.** *(мы никак не меняем данные, а лишь кодируем их специальным алфавитом)*

**Метод Хаффмана**

Метод основан на алгоритме оптимального префиксного кодирования алфавита: исходный алгоритм Хаффмана является оптимальным для посимвольного кодирования с известным входным распределением вероятностей, т. е. для отдельного кодирования несвязанных символов в таком потоке данных. Отличается от метода Шеннона – Фано лишь в части кодирования символов исходного алфавита.

**Лемма**. Для любого заданного алфавита (источника) с N > 2 символами существует оптимальный двоичный код, в котором два наименее вероятных символа (слова) имеют одну и ту же длину и отличаются лишь последним битом

1. Вычислить вероятности символов в сообщении/алфавите
2. Отсортировать по убыванию
3. Каждому символу присваивается код
   1. 2 символа с минимальными вероятностями образуют 2 листа, которые сходятся в 1 узел с вероятность. = их сумме (p(a62) = p(a6)+p(a2))
   2. Каждому листу попарно присваивается код 1 и 0
   3. Затем шаги a и b повторяются, снова берутся 2 минимальных элемента (здесь a4 и a62), пока не будет построено полное дерево - в корне дерева вероятность будет 1
   4. Код символа записывается по 0 и 1 от корня
4. По полученной таблице каждый символ сообщения преобразуем в код.



**Обратное преобразование такое же как и в методе Шеннона-Фано** (см 77) (таблица выглядит точно так же, отличается только способ её получения)

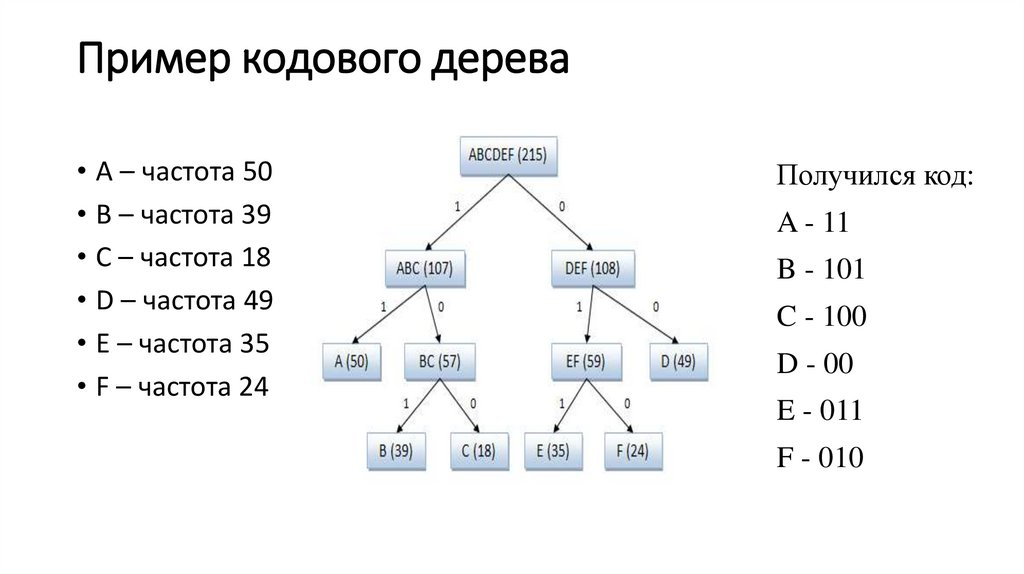
## 33. Метод сжатия данных Шеннона-Фано

**ТАКЖЕ см. 32 про статистические алгоритмы**

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.

1. Вычислить вероятности символов в сообщении/алфавите
2. Отсортировать по убыванию
3. Каждому символу присваивается код
   1. Таблицы делятся на 2 части так, чтобы сумма вероятности символов в двух частях были примерно равны **(чем ближе - тем лучше сжатие =>** *сжатие хорошо работает, когда когда вероятности появления символов равны числам* ***(1/2)n***, n - целое число**)**
   2. Одной части таблицы присваивается код 1, второй 0
   3. Повторяем процедуру с 3 шага для каждой части таблицы, пока не останется по 1 или 2 символам (им присваиваем 1 или 0 соответственно) - ***рекурсия***
4. Закодировать сообщение полученной таблицей преобразований

Можно представить в виде дерева



Для декодирования достаточно по той же таблице провести обратное преобразование:

1. Берём символы до тех пор, пока не получится один из известных кодов
2. После этого переходим к следующей последовательности

Для дерева выше: пусть сообщение 11100101

1. 1 -> нет соответствующего кода
2. 11 - > А
3. Осталось: **100101**
4. 1 -> нет кода
5. 10 -> нет кода
6. 100 -> C
7. Осталось: **101**
8. и т.д.

В результате получится, что символы, которые встречаются чаще, будут зашифрованы более коротким кодом

## 34. Арифметический метод сжатия данных

Текст представляется **вещественными числами** в интервале от 0 до 1. По мере анализа текста отображающий его интервал уменьшается.

**Идея**: присваивать коды не отдельным символам, а их **последовательностям**.

**Исходная инфа** - частота встречаемости символов.

Алгоритмы прямого и обратного преобразования базируются на операциях с “рабочим отрезком” (интервал с расположенными на нём точками, а длины отрезков между точками пропорционально соответствуют частоте встречаемости символов).

**Алгоритм прямого преобразования:**

1. Вычислить частоту встречаемости символов
2. Построить основной рабочий отрезок от 0 до 1 и разбить его на пропорциональные частотам интервалы
3. Берём кодируемый символ, ищем для него соответствующий участок на рабочем отрезке, этот участок становится новым рабочим

**Пример: молоко**

*Частоты символов:*

м - 0.1666

л - 0.1666

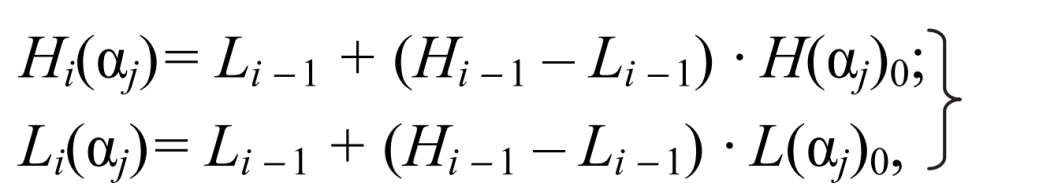
к - 0.1666

о - 0.5



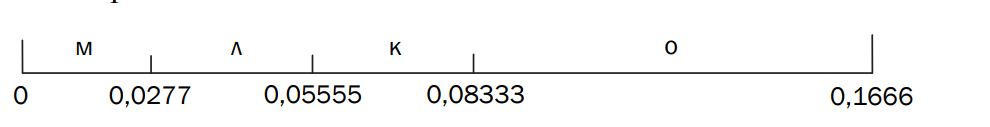
^ здесь рабочий участок  м = [0; 0,1666) **(первый символ молоко - м)**

1. Находим новые верхнюю и нижнюю границы



1. Разбиваем новый рабочий отрезок в соответствии с интервалами частот

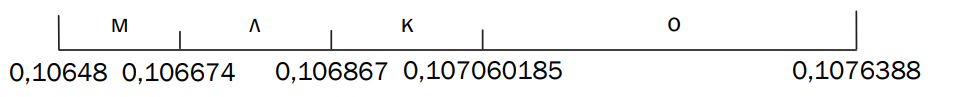
В итоге будет получим конечный рабочий отрезок, интервал которого будет браться за основу для итогового кода.





1. Из полученного интервала выбираем число, чаще всего это нижняя граница отрезка.

(после 5 повторений: )



Последняя буква о -> передаем 0,107060185 ~ 0,1071 (нижняя граница)

***Необходимо также передать длину сообщения***

     Обратное преобразование:

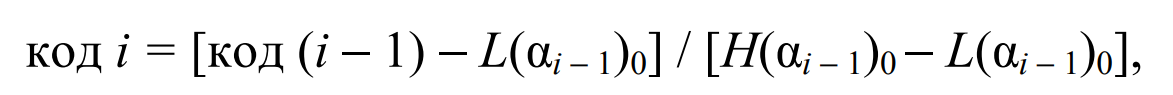
1. В соответствии с частотами строим рабочий отрезок



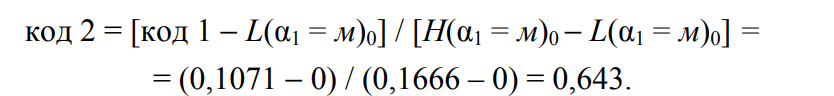
1. Находим, какому интервалу принадлежит код, который получен в результате сжатия
2. Определяем какому символу принадлежит найденный интервал

(0,1071 принадлежит отрезку м -> первый символ м)

1. Вычисляем код следующего символа по формуле



**Пример**:



0,643 входит в интервал [0,5; 1) -> второй символ о.

Имеем *мо-*  
Продолжаем, пока не декодируем все символы(количество - длина сообщения)

  К числу основных особенностей методов можно отнести следующие:

1. ничего не передает до полного завершения анализа всего текста;
2. обычно результат представляется в формате целочисленной арифметики;
3. требуемая для представления интервала [Hi; Li] точность возрастает вместе с длиной анализируемого текста;
4. арифметическое кодирование «работает» пpи помощи масштабирования или нормализации накопленных веpоятностей, поставляемых моделью в интеpвалах [Hi; Li] для каждого пеpедаваемого символа; если пpедположить, что Hi и Li настолько близки дpуг к дpугу, что опеpация масштабиpования «пpиводит» pазные символы сообщения к одному целому числу, входящему в [Hi; Li], то дальнейшее кодиpование пpодолжать невозможно. Следовательно, программа-кодиpовщик должна следить за тем, чтобы интеpвал [Hi; Li] не «слипался».

# 35. Базовые понятия криптографии. Основы теории больших чисел. Проблема дискретного логарифма.

**Криптография** – наука (раздел математики), о методах обеспечения конфиденциальности или безопасности информации, связанной с тайной, интегральностью данных и идентификацией.

**Криптоанализ** — наука о методах получения исходного значения зашифрованной информации без наличия секретного и не известного аналитику ключа

**Криптосистема** – это понятие, относящееся к совокупности программно-технических средств, функционирующих на основе установленных криптографических алгоритмов и осуществляющих зашифрование и расшифрование данных

· Типы криптосистем по используемым ключам:

Симметричные криптосистемы (Symmetric-key cryptography): Используют один и тот же ключ как для шифрования, так и для расшифрования. Примеры: DES, AES, ГОСТ 28147-89. Основная проблема – безопасная передача ключа.

Асимметричные криптосистемы (Asymmetric-key cryptography / Public-key cryptography): Используют пару ключей: открытый (public key) для шифрования и проверки электронной подписи, и закрытый (private key) для расшифрования и создания электронной подписи. Примеры: RSA, ElGamal, ECC. Закрытый ключ держится в секрете, открытый может быть опубликован.

Многие современные криптографические алгоритмы, особенно асимметричные, полагаются на вычислительную сложность определенных задач из теории чисел. "Большие числа" здесь означают числа, состоящие из сотен или даже тысяч бит.

**Модулярная арифметика:**

Это система арифметики для целых чисел, где числа "зацикливаются" при достижении определенного значения — модуля n.

Запись a ≡ b (mod n) означает, что a и b имеют одинаковый остаток при делении на n, или, что эквивалентно, (a - b) делится на n.

Операции сложения, вычитания, умножения и возведения в степень по модулю являются основой многих криптосистем (например, (a + b) mod n, (a \* b) mod n, a^k mod n).

**Простые числа (Prime numbers):**

Число p > 1 называется простым, если оно делится только на 1 и на само себя.

Генерация больших простых чисел и проверка чисел на простоту (тесты простоты, например, Миллера-Рабина) – важные задачи.

Безопасность RSA основана на сложности факторизации (разложения на простые множители) больших составных чисел.

**Наибольший общий делитель (НОД) и алгоритм Евклида:**

НОД двух целых чисел – это наибольшее целое число, на которое оба числа делятся без остатка.

Алгоритм Евклида – эффективный метод для нахождения НОД.

Если НОД(a, n) = 1, то a и n называются взаимно простыми.

**Мультипликативная группа вычетов по модулю n (ℤ\_n):**

Состоит из чисел от 1 до n-1, которые взаимно просты с n.

Операция в этой группе – умножение по модулю n.

Расширенный алгоритм Евклида позволяет найти мультипликативный обратный элемент a⁻¹ по модулю n (т.е. такое число x, что ax ≡ 1 (mod n)), если НОД(a, n) = 1. Это критически важно, например, для RSA.

**Теорема Эйлера:**

Если a и n взаимно просты, то a^φ(n) ≡ 1 (mod n), где φ(n) (функция Эйлера) – количество положительных целых чисел, меньших n и взаимно простых с n.

Если n – простое число p, то φ(p) = p - 1. В этом случае теорема Эйлера превращается в Малую теорему Ферма: a^(p-1) ≡ 1 (mod p) для простого p и целого a, не делящегося на p.

Эти теоремы лежат в основе алгоритма RSA.

**Сложность факторизации:**

Задача разложения большого числа на его простые множители является вычислительно очень сложной.

На сегодняшний день не существует эффективного (полиномиального) алгоритма для факторизации больших чисел на классических компьютерах. На этом предположении строится безопасность RSA.

Проблема дискретного логарифма является еще одной фундаментальной вычислительно сложной задачей, на которой основаны многие криптосистемы.

**Определение:**

Пусть G – конечная циклическая группа (например, мультипликативная группа вычетов ℤ\_p по простому модулю p, или группа точек на эллиптической кривой).

Пусть g – порождающий элемент (генератор) этой группы, а y – некоторый элемент группы G.

Проблема дискретного логарифма (DLP) заключается в нахождении такого целого числа x (называемого дискретным логарифмом y по основанию g), что:  
g^x ≡ y (mod p) (в случае ℤ\*\_p\*)  
или x \* g = y (в аддитивной записи для групп на эллиптических кривых, где x\*g означает g+g+...+g x раз).

**Сложность:**

В то время как вычисление y по известным g, x и p (возведение в степень по модулю) является относительно простой задачей (например, с помощью алгоритма быстрого возведения в степень), обратная задача – нахождение x по известным g, y и p – считается вычислительно трудной для правильно выбранных групп и больших параметров.

Не существует известного эффективного (полиномиального) алгоритма для решения DLP в общем случае на классических компьютерах.

**Применение в криптографии:**

Протокол обмена ключами Диффи-Хеллмана (Diffie-Hellman key exchange): Позволяет двум сторонам выработать общий секретный ключ по незащищенному каналу связи. Его безопасность напрямую основана на сложности DLP.

Алгоритм шифрования Эль-Гамаля (ElGamal encryption): Асимметричная система шифрования, основанная на DLP.

Схема цифровой подписи Эль-Гамаля и DSA (Digital Signature Algorithm): Также используют сложность DLP.

Криптография на эллиптических кривых (Elliptic Curve Cryptography - ECC): Современные криптосистемы (например, ECDH, ECDSA) используют аналог DLP на группе точек эллиптической кривой (ECDLP). ECDLP считается еще более сложной задачей при сравнимой длине ключа, что позволяет использовать более короткие ключи для эквивалентного уровня безопасности.

# 36. Основная теорема арифметики. Алгоритм Евклида нахождения НОД

**Основная теорема арифметики.**

Всякое натуральное число N, кроме 1, можно представить как произведение простых сомножителей: N = p1 \* p2 \* p3 \* ... \* pn , n > 1 .

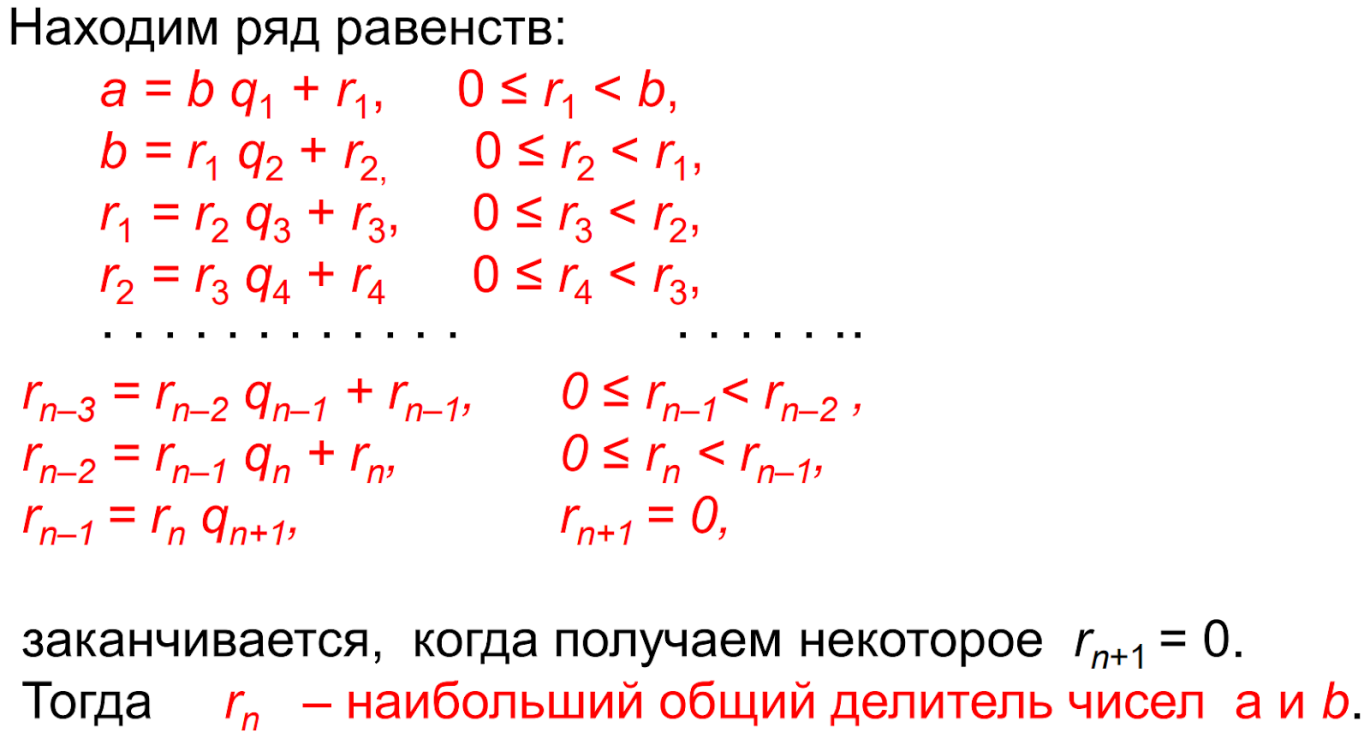
Натуральное число p называется **простым**, если p > 1 и не имеет положительных делителей, отличных от 1 и p

**Взаимно простые числа** а, b не имеют общих множителей, кроме 1, НОД (а, b) =1

***Проблема***. При разрядности 1024 и более бит нахождение пары взаимно простых чисел, удовлетворяющих определенному условию, может занять сотни лет

**Алгоритм Евклида**

**Задача:** Поиск НОД двух чисел a и b (a > b).



# 37. Основы модулярной арифметики. Вычеты.

***Модулярная арифметика*** - К. Ф. Гаусс. В этой арифметике мы интересуемся остатком от деления числа а на число n (n – натуральное число и n > 1).

Если таким остатком является число b, то можно записать:

**a ≡ b (mod n),**

 или

**a ≡ b mod n.**

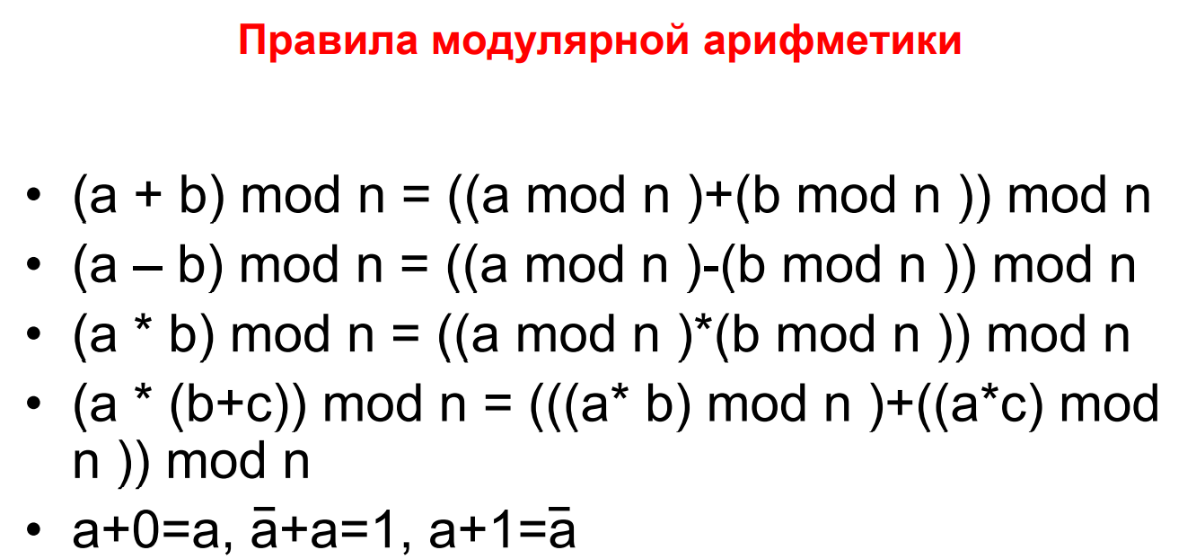
*a сравнимо с b по модулю n*

**a ≡ b mod n, если a=b+kn при целом k**

b называют **вычетом по модулю n**;

a mod n обозначает **вычет** от а;

Множество целых чисел от 0 до n-1 образует полную **систему вычетов** по модулю n



* ***Коммутативна,***
* ***Ассоциативна***
* ***Дистрибутивна***

*1) коммутативность: a+b=b+a, ab=ba и так с любой операцией (можно переставлять компоненты)*

*2) ассоциативность - a+(b+c)=(a+b)+c (опять же для любой операции)*

*3) дистрибутивность - согласованость операций, например: x(y+z)=xy+xz*

# 38. Обратные вычисления по модулю в криптографии.

В модулярной арифметике запись уравнения в виде

**aх ≡ 1 mod n**

Предусматривает поиск таких значений х и k, которые удовлетворяют равенству

**aх = nk + 1**

Здесь x - **обратное** к а по модулю n

Может быть сформулирована следующим образом: найти такое х, что

**1 ≡ ах mod n**

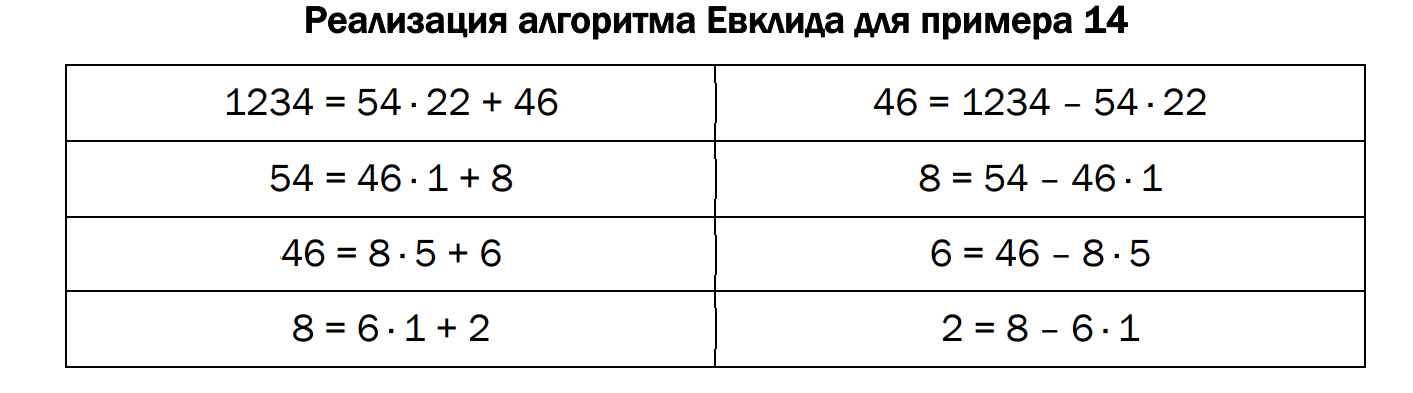
Поиск обратного числа по модулю - **расширенный алгоритм Евклида.** Состоит из двух этапов: спуск и восхождение.

Если НОД (a, b) = d, то справедливо следующее соотношение (**соотношение Безу**):

 аu + bv = d.

Следовательно, если НОД = 1, то

аu + bv = 1



**Малая теорема Ферма**

Если n – простое число и а не кратно n, то справедливо

**a(n-1) ≡1 mod n**

или

**an ≡ a ( mod n )**

**Обобщение Эйлера над Малой теоремой Ферма:**

Если НОД (а, n) =1, то

**аφ(n) mod n = 1**

(т.к. функция эйлера от простого числа n равняется n-1)

Зачем?

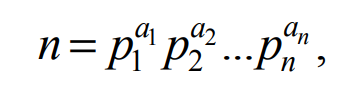
Можем найти обратное по модулю, если разделить обе части уравнения на a:

**а-1= аφ(n)-1 mod n**

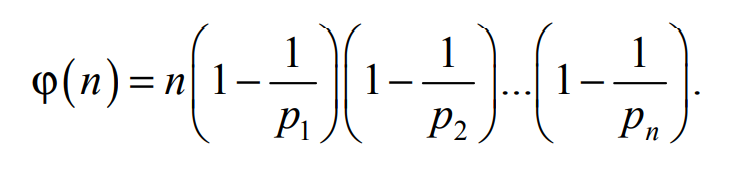
Особенно удобно, если у нас n простое, степень простого числа или произведение простых (см 33 для этих случае)

# 39.  (?)Функция Эйлера в криптографии.

***Функция Эйлера (φ(n))***- Количество натуральных чисел, не превосходящих n и взаимно простых с n



pi - разные простые множители числа n в некоторой степени, тогда:



***Пример:***

Определить количество натуральных чисел, не превосходящих 12 и взаимно простых с 12.

Взаимно простыми с 12 будут четыре числа 1, 5, 7, 11, т. е. φ(12) = 4 – получено методом «ручного» подсчета.

Каноническое разложение числа 12: 12 = 2 · 2 · 3 = 22 · 3, т. е. p1 = 2, p2 = 3.

Теперь подсчитаем функцию Эйлера: φ(12) = 12 · (1 – 1/2) · (1 – 1/3) = 4.

Если p – **простое** число, то:

φ(p) = p – 1,

если числа **p и q являются простыми и p ≠ q**, то:

φ(p) = (p – 1)(q – 1).

если **p простое, а n - целое число, то:**

https://lh7-us.googleusercontent.com/vBFwJswacTxxVtM3PY2QHei1bGpno29zOImaTt32SSdyDwPlAKnrYmm6DEwjWK2MLMyO1SbXMx_ATRkXPzZhTzt90A-tsMuJ3CbsLV1rqUEFvB-_gk96D_Lio0gzTtRSbzfkMs3giFS-5u3xO0byBQ

Если p и q – очень большие простые числа и известен результат их перемножения (число n), то обратная задача – найти p и q по известному n **(задача факторизации)** даже для современных вычислительных средств представляется практически неразрешимой.

Эта особенность используется в некоторых алгоритмах асимметричной криптографии.

П**рименение: RSA**

## 40. Общая классификация криптографических методов защиты информации.

1. На основе процедуры шифрования
   1. подстановочные - Цезарь, Виженера
   2. перестановочные
2. На основе генерирования и использования ключа
   1. блочные - DES
   2. поточные (потоковые) - RC4
3. На основе типа ключа
   1. симметричные (с тайным ключом): e=d - DES
   2. асимметричные (с открытым или публичным ключом): e!=d - RSA, El Gamal

# 

# 41. Подстановочные шифры. Шифр Цезаря. Криптостойкость.

**Подстановочный шифр** - замена символов открытого текста соответственно символами того же или иного алфавита - подстановка.

* **моноалфавитные** (шифры однозначной замены или простые подстановочные) - Цезарь;
* **полиграммные** - замена группы символов - Плейфер;
* **омофонические** (однозвучные шифры или шифры многозначной замены);
* **полиалфавитные** (несколько моноалфавитных(?))-  Виженер

Шифр Цезаря:

* Простой (сдвиг на k)
* Аффинный (a b задаем, x - позиция символа, N длина алфавита)
  + y ≡ ax + b mod N,
  + x ≡ a–1 (y + N – b) mod N
* С лозунгом (ключевым словом)

**Криптостойкость**: уязвим перед частотным анализом

Мокар: Спасибо большое орлову что рассказал про каждый из шифров. Пидарас. Сам напишу

# 42. Шифр на основе аффинных преобразований. Криптостойкость.

Моноалфавитный подстановочный

* Аффинный
  + y ≡ ax + b mod N,
  + x ≡ a–1 (y + N – b) mod N

**Криптостойкость**: уязвим перед частотным анализом и полным перебором

(**Полный перебор** - В случае шифрования сообщений на русском языке N=33, существует 297 нетривиальных аффинных шифров, не учитывая 33 тривиальных шифра Цезаря (a = 1)