## 1. Сущность проблемы информационной безопасности и надежности систем.

**Информационная (информационно-вычислительная) система (ИС (ИВС)) —** это организационно упорядоченная совокупность документов, технических средств и информационных технологий, реализующая информационные (информационно-вычислительные) процессы.1 Информационные процессы включают сбор, накопление, хранение, обработку (переработку), передачу и использование информации.1

**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** — это свойство системы противодействовать попыткам несанкционированного доступа или нанесения ущерба владельцам и пользователям системы при различных умышленных и неумышленных воздействиях на нее.1 **Защита информации** включает организационные, правовые, программно-технические и иные меры по предотвращению угроз информационной безопасности и устранению их последствий.1

**Информационная безопасность** охватывает все аспекты, связанные с определением, достижением и поддержанием конфиденциальности, целостности и доступности информации или средств ее обработки 1:

* **Конфиденциальность** (confidentiality) — состояние информации, при котором доступ к ней осуществляют только субъекты, имеющие на нее право.1
* **Целостность** (integrity) — избежание несанкционированной модификации информации.1
* **Доступность** (availability) — избежание временного или постоянного сокрытия информации от пользователей, получивших права доступа.1

**Надежность системы** характеризует способность программного, аппаратного или аппаратно-программного средства выполнять требуемые функции в определенных условиях в течение заданного периода времени.1 **Достоверность работы системы (устройства)** — это свойство, характеризующее истинность конечного результата работы, определяемое способностью средств контроля фиксировать правильность или ошибочность работы.1

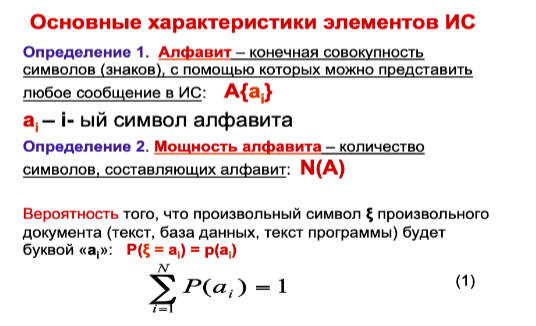
Изначально акцент в надежности систем делался на их способности безотказно функционировать в течение определенного времени, что в основном касалось технических аспектов.1 Однако с развитием информационных технологий и увеличением числа угроз, понятие надежности расширилось до информационной безопасности, которая включает в себя не только техническую исправность, но и защиту от несанкционированного доступа, модификации и скрытия информации.1

## 2. Характеристики и параметры ИС и ИВС.

Информационные системы (ИС) и информационно-вычислительные системы (ИВС) характеризуются рядом фундаментальных понятий и параметров 1:

* **Информация** — это сведения (данные) о внутреннем и окружающем мире, событиях, процессах, явлениях, воспринимаемые и передаваемые людьми или техническими устройствами.1
* **Доступ** — специальный тип взаимодействия между объектом и субъектом, результатом которого является создание потока информации.1
* **Несанкционированный доступ (НСД)** — это доступ к информации, устройствам ее хранения и обработки, а также к каналам передачи, реализуемый без ведома владельца и нарушающий установленные правила доступа.1
* **Объект** — пассивный компонент системы, хранящий, перерабатывающий, передающий или принимающий информацию (например, файлы, программы, устройства).1
* **Субъект** — активный компонент, который может инициировать поток информации (например, пользователь, процесс, устройство).1
* **Идентификация** — процесс распознавания компонентов с помощью уникальных идентификаторов.1
* **Аутентификация** — проверка идентификации для принятия решения о разрешении доступа к ресурсам системы.1

Простейшая информационная система состоит из источника сообщения (ИсС), канала передачи и получателя сообщения (ПС).1 В системах хранения информации канал передачи представлен устройством хранения (памятью).1 Информация передается в форме **сообщений**, которые отображаются в виде **сигналов** — физических процессов, характеризующихся изменяющимися информационными параметрами (амплитуда, частота, фаза).1 Сигналы могут быть непрерывными (аналоговыми) или дискретными (с конечным числом значений, часто 0 или 1).1 **Кодирование** в широком смысле — это преобразование сообщения в сигнал, а в узком — представление исходных знаков в другом алфавите для повышения надежности и удобства передачи.1 **Двоичный канал** использует два значения сигнала (0 или 1), а **двоичный симметричный канал (ДСК)** характеризуется равной вероятностью искажения 0 в 1 и 1 в 0.1



## 3. Характеристика угроз безопасности современным ИС и ИВС.

Фактор, воздействующий на ИВС, определяется как явление, действие или процесс, результатом которых может быть утечка, искажение, уничтожение данных, блокировка доступа к ним, повреждение или уничтожение системы защиты.1 Все многообразие дестабилизирующих факторов делится на два класса: внутренние и внешние.1

**Внутренние дестабилизирующие факторы** влияют на программные средства (ПС) и аппаратные средства (АС) 1:

* **На ПС:** некорректный исходный алгоритм, неправильно запрограммированный исходный алгоритм (первичные ошибки).1
* **На АС:** системные ошибки при постановке задачи проектирования, отклонения от технологии изготовления комплектующих изделий и АС в целом; нарушение режима эксплуатации, вызванное внутренним состоянием АС.1

**Внешние дестабилизирующие факторы** также влияют на ПС и АС 1:

* **На ПС:** неквалифицированные пользователи, несанкционированный доступ к ПС с целью модификации кода.1
* **На АС:** внешние климатические условия, электромагнитные и ионизирующие помехи, перебои в электроснабжении, недостаточная квалификация обслуживающего персонала, несанкционированный (в том числе удаленный) доступ с целью нарушения работоспособности АС.1

Одним из важнейших факторов, дестабилизирующих работу ИВС, являются **электромагнитные и ионизирующие излучения**.1 Источниками могут быть естественными (солнечная радиация) или искусственными (электроприборы, целенаправленные излучатели, изотопы урана и тория).1 Основное последствие их влияния — ошибки в информации, снижающие функциональную надежность АС.1

Несанкционированный доступ часто осуществляется с помощью **деструктивных программных средств** (вредоносного ПО) через компьютерные сети.1 Классификация вредоносного ПО включает:

* **Вирусы:** саморазмножающиеся программы, дописывающие код к исполняемым файлам.1
* **Черви:** самостоятельно размножаются по сети и, в отличие от вирусов, не дописывают себя (как правило) к исполняемым файлам; все черви «съедают» ресурсы компьютера, «нагоняют» интернет-трафик и могут привести к утечке данных с вашего компьютера.1
* **Кейлоггеры (анализаторы клавиатуры):** программы, которые регистрируют нажатия клавиш, делают снимки рабочего стола, отслеживают действия пользователя во время работы за компьютером и сохраняют эти данные в скрытый файл на диске, затем этот файл попадает к злоумышленнику.1
* **Трояны (троянские кони):** собирают конфиденциальную информацию с компьютера пользователя (пароли, базы данных и пр.) и тайно по сети высылают их злоумышленнику (своему хозяину).1
* **Боты:** распространенный в наше время вид зловредного ПО, который устанавливается на компьютерах пользователей (сети botnet) и используется для атак на другие компьютеры.1
* **Снифферы (анализаторы сетевого трафика):** могут использоваться в составе зловредного ПО, скрытно устанавливаться на компьютере пользователя и отслеживать данные, которые отправляет или получает пользователь по сети.1
* **Руткиты:** сами по себе не являются зловредным ПО; назначение – скрывать работу других зловредных программ (кейлоггеров, троянов, червей и т. д.) как от пользователя, так и от программ и средств обеспечения безопасности (антивирусов, файерволов (firewalls), систем обнаружения атак и пр.).1

**Человеческий фактор** также является критическим источником угроз.1 К лицам, способным получить несанкционированный доступ, относятся:

* **Сотрудники организации:** программисты, системные администраторы и даже администраторы информационной безопасности; технический персонал.1
* **Лица, не являющиеся сотрудниками:** посетители офиса; ранее уволенные сотрудники (особенно «обиженные» увольнением); хакеры.1

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

Защита информации определяется как комплекс организационных, правовых, программно-технических и иных мер по предотвращению угроз информационной безопасности и устранению их последствий.1 Методы и средства защиты информации от несанкционированного доступа (НСД) классифицируются следующим образом 1:

1. **Законодательная, нормативная и научная база:**
   * **Национальное законодательство:** Включает международные договоры, Конституцию, законы (например, Закон РБ «Об информации, информатизации и защите информации»), указы Президента, постановления Правительства, нормативные правовые акты министерств и ведомств (например, ОАЦ при Президенте РБ, Минсвязи) и акты местных органов самоуправления.1
   * **Международные стандарты:** Включают BS 7799-1:2005 (практические правила менеджмента ИБ), BS 7799-2:2005 (спецификация СУИБ для сертификации), ISO/IEC 17799:2005 (международный стандарт на основе BS 7799-1), ISO/IEC 27001:2005 (международный стандарт на основе BS 7799-2), ISO/IEC 27002 (текущая версия ISO/IEC 17799:2005), ISO/IEC 27005 (руководство по менеджменту рисков ИБ).1
2. **Организационно-технические и режимные меры и методы (политика информационной безопасности):**
   * Построение политики ИБ основывается на системном подходе, включающем анализ угроз, оценку последствий и адекватность контрмер.1
   * **Направления защиты ИС:** защита объектов ИС, защита процессов/процедур/программ обработки информации, защита каналов связи, подавление побочных электромагнитных излучений, управление системой защиты.1
   * **Организационная защита обеспечивает:** организацию охраны, режима, работу с кадрами и документами; использование технических средств безопасности (замки, карты); информационно-аналитическую деятельность по выявлению угроз.1
3. **Аппаратные, программно-аппаратные и программные способы и средства обеспечения ИБ:**
   * **Средства защиты от НСД:** средства авторизации, аудит.1
   * **Системы мониторинга сетей:** системы мониторинга, анализаторы протоколов.1
   * **Антивирусные средства:** антивирусные программы, антиспамовые средства, межсетевые экраны (firewalls).1
   * **Криптографические средства:** шифрование данных, электронная цифровая подпись.1
   * **Системы бесперебойного питания**.1
   * **Системы аутентификации:** пароль, ключ доступа (физический/электронный), биометрия (анализаторы отпечатков пальцев, сетчатки глаза, голоса, геометрии ладони).1

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 — это 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.1

Для создания эффективной системы информационной безопасности целесообразно разработать 1:

* **Концепцию информационной безопасности:** 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. Все вышеуказанные элементы должны быть взаимосвязанными и непротиворечивыми.1 Также рекомендуется разработать **аварийные планы** для восстановления информационных систем при возникновении форс-мажорных обстоятельств: аварий, катастроф и т. д.1

**Концепция политики ИБ** разрабатывается в соответствии с законодательством по информационной безопасности Республики Беларусь, соответствующими нормативными документами министерства или ведомства, к которому относится организация или учреждение, а также решениями Оперативно-аналитического центра при Президенте Республики Беларусь.1 Обеспечение ИБ на предприятиях и в учреждениях, как правило, является неотъемлемой частью общей системы управления, необходимой для достижения уставных целей и задач.1 Мероприятия по разработке и внедрению политики информационной безопасности в соответствии со стандартом BS ISO/IEC 27001:2005, на основе которого разработан национальный стандарт России ГОСТ Р ИСО/МЭК 27001-2006 , должны начинаться с **определения области действия системы управления информационной безопасностью (СУИБ)**.1 Область действия СУИБ может охватывать всю организацию в целом либо конкретный бизнес-процесс или информационную систему.1

**Основные разделы концепции информационной безопасности** могут быть следующие 1:

* 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. Помимо упомянутых выше законодательных и нормативных актов, в общем плане структура системы обеспечения ИБ должна базироваться на организационно-технических и режимных мерах и методах.1 Для построения политики ИБ рассматривают следующие направления защиты ИВС 1:
* защита объектов ИВС;
* защита процессов, процедур и программ обработки информации;
* защита каналов связи;
* подавление побочных электромагнитных излучений;
* управление системой защиты.

**Организационная защита обеспечивает** 1:

* 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.

**Оценка рисков для информационных ресурсов** 1:

* **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.
* Все многообразие дестабилизирующих факторов можно разделить на два класса: внутренние и внешние.
* **Оценка рисков** включает в себя следующие действия и мероприятия: идентификация значимых угроз и уязвимостей для идентифицированных ресурсов; оценка вероятности возникновения угроз и уязвимостей; вычисление рисков; оценивание рисков по заранее определенной шкале риска.
* Все многообразие потенциальных угроз безопасности информации по природе их возникновения разделяются на два класса: естественные (объективные) и искусственные (субъективные).
* **Источники угроз** по отношению к самой информационной системе могут быть как внешними, так и внутренними.
* **Человеческий фактор:** К ним следует отнести: сотрудников организации (учреждения) (программисты, системные администраторы и даже администраторы информационной безопасности; технический персонал); лиц, не являющихся сотрудниками (посетители офиса; ранее уволенные сотрудники (особенно «обиженные» увольнением); хакеры).
* **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.

Мероприятия по внедрению политики безопасности 1:

После того как документация по информационной безопасности готова, необходима плановая деятельность по ее внедрению в повседневную работу. Основу таких мероприятий составляют инструкции, содержащие подробное описание (алгоритмы) действий по организации информационной защиты и обеспечению разработанных стандартов и процедур, и план мероприятий по обучению персонала и тестированию знаний сотрудников, имеющих доступ к информационным ресурсам.

Можно выделить следующие общие направления мероприятий:

* управление персоналом;
* физическая защита инфраструктуры ИВС;
* поддержание работоспособности ИВС;
* реагирование на нарушения режима безопасности ИВС;
* планирование восстановительных работ. Основой программно-технического уровня являются следующие механизмы безопасности: идентификация и аутентификация пользователей; управление доступом; протоколирование и аудит; криптография; экранирование; обеспечение высокой доступности и т. д.

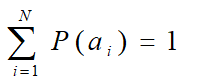
## 6. Энтропия источника сообщения. Энтропия Шеннона.

***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»:

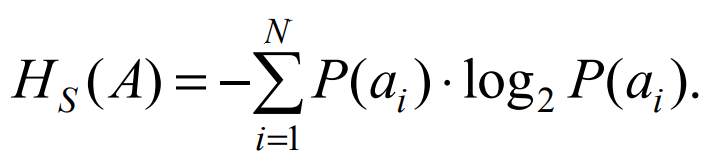
 P (ξ = аi) = p(аi)



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

**Энтропия** является информационной характеристикой алфавита или источника сообщений, основанного на этом алфавите.1 Этот термин был введен К. Шенноном и Р. Хартли.1 С физической точки зрения, энтропия алфавита показывает, какое количество информации приходится в среднем на один символ алфавита.1

Формула Шеннона для расчета энтропии алфавита A = {ai} с N символами выглядит следующим образом:



Вероятности P(ai) могут быть получены путем анализа частотных свойств символов алфавита из достаточно большого электронного текстового документа (не менее нескольких десятков килобайт).1

Количество информации I(Xk) в сообщении Xk, состоящем из k символов, определяется как произведение энтропии алфавита H(A) на количество символов: I(Xk​)=H(A)⋅k.1

**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.**

Для **двоичного алфавита** A = {0, 1} энтропия рассчитывается по формуле: H(A)=−[P(0)⋅log2​(P(0))]−[P(1)⋅log2​(P(1))].1 Если сообщение состоит только из одного типа символов (например, только из единиц, что является моноалфавитом), то вероятность этого символа равна 1, а другого — 0. В таком случае энтропия моноалфавита составляет 0 бит.1 Это означает, что такое сообщение не несет новой информации для получателя, поскольку его содержание полностью предсказуемо.1 Максимальное значение энтропии для двоичного алфавита составляет 1 бит и достигается при равновероятном появлении символов (P(0) = P(1) = 0.5).1

## 7. Энтропия источника сообщения. Энтропия Хартли.

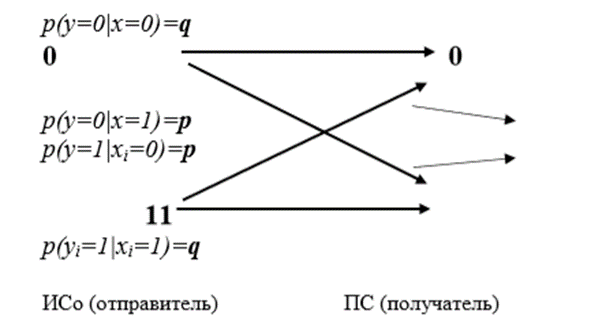
**Энтропия Хартли** рассматривается как частный случай энтропии Шеннона.1 Ее отличительное условие заключается в том, что **все вероятности появления символов в алфавите одинаковы и постоянны для всех символов**.1

Формула Хартли для расчета энтропии выглядит следующим образом: Hс​(A)=log2​(N), где N — мощность (количество символов) алфавита.1 Например, энтропия Хартли для латинского (английского) алфавита, состоящего из 26 символов, составляет приблизительно 4.7 бит.1

При сравнении энтропии Шеннона и энтропии Хартли для одного и того же алфавита (при N > 2) обнаруживается, что их значения не совпадают.1 Это расхождение указывает на **информационную избыточность** любого алфавита.1 Избыточность алфавита определяется как уменьшение информационной нагрузки на один символ вследствие разной вероятности и взаимозависимости появления его символов в сообщениях.1 Эта избыточность является фундаментальным принципом, на котором строятся и функционируют все современные системы сжатия (компрессии) информации.1

## 8. Двоичный канал передачи информации.

**Двоичный канал передачи информации** строится на основе двоичного алфавита A = {0, 1}.1 Это дискретный канал, где сообщения или их передача основываются на двух значениях сигнала — 0 или 1.1



Особым видом двоичного канала является **двоичный симметричный канал (ДСК)**.1 В ДСК вероятности искажения переданного 0 в 1 (p(1|0)) и переданного 1 в 0 (p(0|1)) равны.1 Также предполагается, что вероятности передачи 0 (p(0)) и 1 (p(1)) равны.1 В схематическом представлении ДСК 'q' обозначает вероятность правильной (безошибочной) передачи бита сообщения, а 'p' — вероятность передачи бита с ошибкой, при этом p+q=1.1

## 9. Энтропия двоичного алфавита.

Энтропия двоичного алфавита A = {0, 1} рассчитывается по формуле Шеннона: H(A)=−[P(0)⋅log2​(P(0))]−[P(1)⋅log2​(P(1))].1 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)). Энтропия зависит от вероятностей появления символов.1

Если сообщение состоит только из одного типа символов (например, только из единиц, Xk = 11...1), то вероятность этого символа равна 1, а другого — 0. В этом случае энтропия двоичного алфавита составляет 0 бит.1 Это означает, что сообщение не несет новой информации, поскольку его содержание полностью предсказуемо для получателя.1

Максимальное значение энтропии для двоичного алфавита составляет 1 бит и достигается при равновероятном появлении символов (P(0) = P(1) = 0.5).1 В этом случае сообщение является максимально непредсказуемым, что соответствует максимальной информационной плотности.1

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

**Условная энтропия H(Y | X)** представляет собой среднее количество информации для входного символа относительно принятого сообщения Y, или потери информации на один символ переданного сообщения.1 Она количественно выражает неопределенность, которая остается относительно переданного символа после получения принятого символа.1

Для двоичного симметричного канала (ДСК) условная энтропия рассчитывается по формуле: H(Y∣X)=−p⋅log2​(p)−q⋅log2​(q).1 Здесь 'p' — вероятность ошибки, а 'q' — вероятность правильной передачи бита, при этом p+q=1.1

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

**Эффективная энтропия** He(A) алфавита, или пропускная способность канала, определяется как He=H(X)–H(X∣Y), где H(X) — энтропия источника.1 Если вероятность ошибки p = 0, то H(Y | X) = 0, что означает отсутствие потерь информации при передаче.1 Однако если p > 0, информация теряется. Например, при p = 0.01, H(Y | X) = 0.081 бит, что означает потерю 0.081 бит информации на каждый переданный двоичный символ.1

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.

Избыточность в информационно-вычислительных системах (ИВС) сознательно вводится для повышения их надежности, то есть функциональной надежности.1 Это является ключевым шагом в обеспечении качества и надежности на всех этапах жизненного цикла устройств и каналов передачи информации.1

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

Различают несколько видов избыточности:

* **Информационная избыточность:** Является характеристикой любого сообщения.1 Она может быть удалена с помощью методов сжатия (компрессии), но также искусственно вводится в систему для повышения надежности.1 Помехоустойчивое кодирование является основным методом введения информационной избыточности.1 Это преобразование исходного информационного сообщения Xk в кодовое слово Xn путем добавления r избыточных символов Xr, которые не несут новой информации, но служат для обнаружения и исправления ошибок.1
* **Структурная избыточность:** Подразумевается использованием аппаратных средств контроля, таких как избыточные аппаратные средства.1 Это может быть дублирование компонентов или введение дополнительных структурных элементов для обеспечения отказоустойчивости.
* **Временная избыточность:** Реализуется через методы перемежения/деперемежения данных.1 Эти методы переупорядочивают данные во времени, чтобы распределить сгруппированные ошибки, делая их более управляемыми для коррекции.1

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

**Помехоустойчивое кодирование** — это один из ключевых методов повышения надежности (функциональной надежности) информационных систем.1 Его основное назначение — противодействовать помехам, которые воздействуют на систему и приводят к ошибкам в данных.1

Суть метода заключается в преобразовании исходного информационного сообщения Xk (длиной k символов), называемого информационным словом, в избыточное сообщение (кодовое слово) Xn длиной n символов (n > k).1 К информационному слову Xk дополнительно присоединяются r избыточных символов, составляющих избыточное слово Xr.1 Эти избыточные символы не несут новой информации, но их назначение — обнаружение и исправление ошибок.1

Основные понятия, используемые в помехоустойчивом кодировании 1:

* **Информационное слово (Xk):** Исходное сообщение, состоящее из k бит.
* **Избыточные (проверочные) биты (Xr):** Дополнительные r бит, добавляемые к информационному слову (r=n−k).
* **Кодовое слово (Xn):** Блок из n бит, сформированный из информационного и избыточного слов.
* **Степень кодирования (code rate):** Отношение числа бит данных к общему числу бит данных (k/n), показывающее долю полезной информации в коде.
* **Расстояние Хемминга (d):** Число позиций, в которых два кодовых слова различаются.
* **Минимальное кодовое расстояние (dmin):** Минимальное расстояние Хемминга между любыми двумя разрешенными кодовыми словами. Оно определяет способность кода обнаруживать и исправлять ошибки.
  + Количество обнаруживаемых ошибок (t0​): t0​=d/2 (если d четно) или t0​=(d−1)/2 (если d нечетно).
  + Количество исправляемых ошибок (tu​): tu​=(d−1)/2 (если d нечетно) или tu​=(d−2)/2 (если d четно).
* **Синдром (S):** Результат умножения принятого сообщения (Yn) на транспонированную проверочную матрицу (H). Если синдром равен 0, ошибок не обнаружено; иначе ошибки присутствуют.
* **Кратность ошибки:** Количество ошибочных двоичных символов.

## 13. Помехоустойчивое кодирование информации. Классификация кодов.

Помехоустойчивое кодирование информации является частью прикладной теории кодирования, направленной на обнаружение и исправление ошибок между переданным и принятым сообщениями.1 Коды классифицируются по нескольким признакам 1:

1. **По способу формирования кодового слова:**
   * **Блочные коды:** Каждому сообщению из k символов сопоставляется блок нового сообщения (кодового слова) из n символов (n=k+r).1
   * **Непрерывные (рекуррентные, цепные, сверточные) коды:** Непрерывная последовательность символов, не разделяемая на блоки, где проверочные символы вставляются между информационными.1
2. **По наличию или отсутствию систематической структуры:**
   * **Систематические коды:** Характеризуются тем, что сумма по модулю 2 двух разрешенных кодовых комбинаций снова дает разрешенную кодовую комбинацию.1
   * **Несистематические коды:** Не обладают отмеченными выше свойствами (к ним относятся итеративные коды).1
3. **По способу вычисления проверочных символов:**
   * **Линейные коды:** Проверочные (избыточные) символы вычисляются как линейная комбинация информационных символов. Обозначаются как [n,k]-код.1
   * **Нелинейные коды:** Противоположность линейным.1

**Специфические типы кодов** 1:

* **Код простой четности:** Простейший избыточный код, r=1. Обнаруживает все нечетные ошибки, но не исправляет их. Минимальное кодовое расстояние dmin​=2.
* **Код Хемминга:** Характеризуется dmin​=3. Позволяет обнаруживать и исправлять одиночную ошибку. Использует расширенный контроль четности групп символов информационного слова (r>1).
* **Модифицированный код Хемминга:** Характеризуется dmin​=4. Позволяет обнаруживать две ошибки и исправлять одну. Отличается от dmin​=3 видом проверочной матрицы.
* **Итеративные коды (составные коды):** Иногда называются прямоугольными или композиционными (product code). Являются одними из самых простых в аппаратной и программной реализации, но имеют сравнительно высокую избыточность. Могут быть двумерными, трехмерными и выше. Для двумерного кода, построенного на двух кодах простой четности, dmin​=4.
* **Циклические коды (ЦК):** Относятся к классу линейных, систематических кодов. Любой циклический сдвиг кодового слова также является разрешенным кодовым словом. Описываются порождающими полиномами G(X).

## 14. Теоретические основы избыточного кодирования информации.

Теоретические основы избыточного кодирования информации направлены на повышение надежности информационных систем путем искусственного введения избыточности.1

**Теорема Шеннона** для дискретной передачи информации по каналу связи с помехами утверждает, что вероятность ошибочного декодирования принимаемых сигналов может быть обеспечена сколь угодно малой путем выбора соответствующего способа кодирования.1 Эта теорема указывает на принципиальную возможность достижения сколь угодно высокой надежности передачи, хотя и не описывает конкретные методы построения кодов.1

**Линейные блочные коды (ЛБК)** представляют собой класс кодов с контролем четности, которые можно описать парой чисел (n,k), где n — длина кодового слова, k — длина информационного слова.1 Для формирования r проверочных символов используется **порождающая матрица (G)** размерностью k×n с единичной подматрицей (I) в первых k строках и столбцах: G=[I∣P].1 Кодирование заключается в умножении вектора сообщения Xk длиной k на матрицу G по модулю 2, в результате чего первые k символов кодового слова Xn равны соответствующим символам сообщения, а последние r символов (Xr) образуются как линейные комбинации первых.1

Для каждой порождающей матрицы G существует **проверочная матрица (H)** размерностью r×n, которая задает базис нулевого пространства кода и удовлетворяет равенству G⋅HT=0.1 Матрица H может быть представлена как H=.1 Кодовое слово Xn также удовлетворяет тождеству H⋅XnT=0.1

**Синдром (S)** — это результат умножения принятого сообщения (Yn) на транспонированную проверочную матрицу (H).1 Принятый вектор Yn может быть представлен как Yn=Xn⊕En, где En — вектор ошибки.1 Если все r символов синдрома нулевые (S=0), то ошибок в принятом сообщении нет; в противном случае ошибки присутствуют.1 Ненулевой синдром всегда равен сумме по модулю 2 тех вектор-столбцов матрицы H, номера которых соответствуют номерам ошибочных битов в слове Yn.1

**Минимальное кодовое расстояние (dmin​)** между двумя кодовыми словами определяет количество обнаруживаемых (t0​) и исправляемых (tu​) ошибок 1:

* t0​=d/2 (если d четно) или t0​=(d−1)/2 (если d нечетно).1
* tu​=(d−1)/2 (если d нечетно) или tu​=(d−2)/2 (если d четно).1 Количество избыточных символов r определяется из неравенства 2r≥n+1 или r≥log2​(k+r+1).1

**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

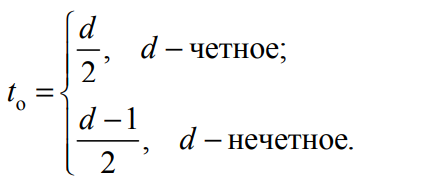
**Расстояние Хемминга** или кодовое расстояние (d) между двумя произвольными словами (X,Y) одинаковой длины равно количеству позиций, в которых X и Y отличаются между собой.

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

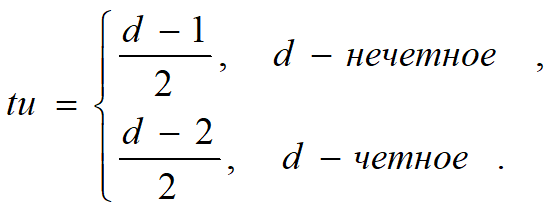
**Пример:**  X=101,Y=111. d(X,Y) = 1.

**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

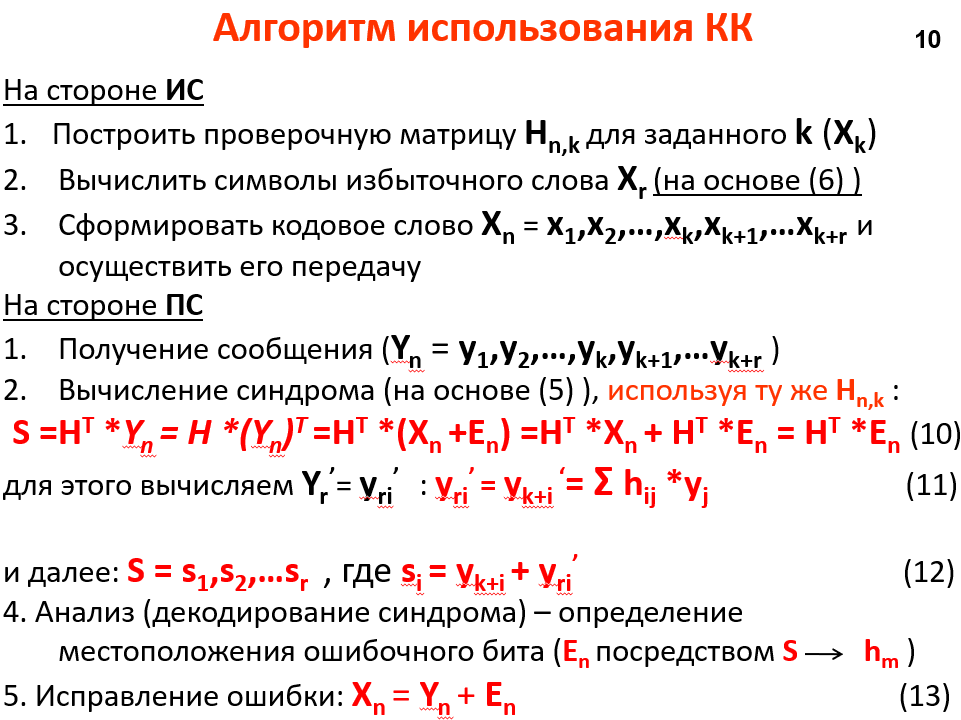
**Теорема Шеннона**. Для любого ДСК и любого ε >0 существует  (n,k)- двоичный код со скоростью Rс, если Rс< C(p), n достаточно велико и  р < ε.

**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.

Алгоритм использования корректирующего кода включает процессы кодирования на передающей стороне и декодирования на принимающей стороне для обнаружения и исправления ошибок.1



**Общий алгоритм** 1:

1. **Кодирование:** Исходное информационное сообщение Xk преобразуется в кодовое слово Xn путем добавления r избыточных бит Xr.
2. **Передача:** Кодовое слово Xn передается по каналу связи.
3. **Декодирование:** Принятое слово Yn обрабатывается для обнаружения и исправления ошибок.

**Кодирование (для линейных блочных кодов, например, кода Хемминга)** 1:

* Избыточные символы Xr вычисляются с использованием порождающей матрицы G. Кодовое слово Xn формируется путем умножения вектора сообщения Xk на G по модулю 2. При этом первые k символов Xn равны Xk, а последние r — это Xr.

**Декодирование (синдромный метод)** 1:

* На принимающей стороне вычисляется **синдром (S)** путем умножения принятого сообщения Yn на транспонированную проверочную матрицу HT.
* Если S=0, то ошибок не обнаружено.
* Если S=0, то ошибки присутствуют. Синдром S равен сумме по модулю 2 столбцов матрицы H, соответствующих позициям ошибок.
* **Исправление ошибок:** Для одиночных ошибок (например, в коде Хемминга с dmin​=3), синдром S прямо указывает на позицию ошибки. Формируется **вектор ошибки En** (с единицей на позиции ошибки и нулями в остальных). Исправленное слово Yn′ получается путем сложения Yn и En по модулю 2: Yn′=Yn⊕En.

**Кодирование (для циклических кодов, метод деления полиномов)** 1:

1. К комбинации слова Xk справа дописывается r нулей, что эквивалентно умножению Xk на оператор сдвига Xr.
2. Произведение Xk⋅Xr делится на соответствующий порождающий полином G(X), и определяется остаток R(X), степень которого не превышает (r−1). Этот остаток R(X) и представляет собой группу проверочных символов (Xr).
3. Вычисленный остаток R(X) присоединяется справа к Xk для формирования Xn.

**Декодирование (для циклических кодов, синдромный метод)** 1:

1. Принятое кодовое слово Yn делится на порождающий полином G(X). Остаток от этого деления является синдромом Sr.
2. Если Sr=0, то ошибок нет.
3. Если Sr=0, то ошибки присутствуют. Sr соответствует определенной конфигурации ошибок.
4. Определяется вектор ошибки En из Sr. Исправленное слово Yn′ получается сложением Yn и En по модулю 2.

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

Декодирование кодовых слов является критически важным этапом для обеспечения надежности передачи информации, поскольку оно позволяет обнаруживать и исправлять ошибки, возникающие в канале связи. Основным инструментом для этого является **синдром**.1

**Вычисление синдрома** 1:

* Для линейных кодов, таких как коды Хемминга, синдром S вычисляется путем умножения принятого сообщения Yn на транспонированную проверочную матрицу HT: S=Yn⋅HT.
* Для циклических кодов синдром Sr является остатком от деления принятого кодового слова Yn на порождающий полином G(X).

**Поиск ошибок (обнаружение)** 1:

* Если все символы синдрома равны нулю (S=0), то считается, что ошибок в принятом сообщении нет.
* Если синдром не равен нулю (S=0), это указывает на наличие ошибок.

**Исправление ошибок (локализация и коррекция)** 1:

* **Локализация ошибки:** Ненулевой синдром соответствует определенной конфигурации ошибок.1 Для кодов Хемминга с dmin​=3, синдром S прямо указывает на позицию одиночной ошибки, поскольку он будет равен столбцу проверочной матрицы H, соответствующему этой позиции.1 Для циклических кодов синдром также имеет аналогичные свойства и позволяет определить местоположение ошибки.1
* **Формирование вектора ошибки (En):** На основе локализованной позиции ошибки формируется вектор ошибки En. Этот вектор является унарным, то есть содержит единицу на позиции, где произошла ошибка, и нули в остальных позициях.1
* **Коррекция ошибки:** Исправление ошибочного бита достигается путем сложения (по модулю 2, то есть операции XOR) принятого слова Yn и вектора ошибки En: Yn′=Yn⊕En.1 Эта операция инвертирует ошибочный бит, восстанавливая исходное значение.

**Ограничения** 1:

* Код Хемминга с dmin​=3 не может однозначно идентифицировать или исправлять четное число ошибок (например, две ошибки), так как синдром в этом случае будет равен сумме двух столбцов H, что может соответствовать другой одиночной ошибке или быть неопределенным.1
* Итеративные коды также не могут определять местоположение множественных ошибок, если они имеют четное число искаженных символов как по строкам, так и по столбцам (например, четыре ошибки, расположенные в вершинах прямоугольника).1

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

**Код Хемминга с минимальным кодовым расстоянием dmin​=3** является одним из наиболее известных линейных блочных кодов.1

**Характеристики и возможности** 1:

* **dmin​=3**.
* Позволяет **обнаруживать и исправлять одиночную ошибку** в кодовом слове.
* Использует **расширенный контроль четности** групп символов информационного слова, что означает, что количество избыточных символов r>1.
* Количество избыточных символов r может быть вычислено по соотношению: r≥log2​(k+1), где k — длина информационного слова.

**Кодирование** 1:

* Кодирование сообщения должно удовлетворять соотношению H⋅XnT=0, где H — проверочная матрица, а XnT — транспонированное кодовое слово.
* Проверочная матрица H имеет классический вид и состоит из двух подматриц: A (размером r×k) и I (единичная матрица размером r×r): H=[A∣I].
* Вес столбцов подматрицы A должен быть больше либо равен 2.
* Первые k символов кодового слова равны соответствующим символам сообщения, а последние r символов (Xr) образуются как линейные комбинации первых.

**Декодирование (синдромный метод)** 1:

* Результат умножения принятого сообщения (Yn) на транспонированную проверочную матрицу (HT) называется **синдромом (S)**: S=Yn⋅HT.
* Если все r символов синдрома нулевые (S=0), то ошибок нет.
* Если синдром не равен нулю (S=0), это означает наличие ошибки. Местоположение ошибки выявляется анализом (декодированием) синдрома: синдром S будет равен столбцу матрицы H, номер которого соответствует номеру ошибочного символа в принятом кодовом слове.
* Исправление ошибки достигается простым сложением по модулю 2 вектора ошибки (En) с принятым кодовым словом (Yn): Yn′=Yn⊕En. Вектор En является унарным (имеет единичный вес) и содержит 1 на позиции ошибки.

**Ограничения** 1:

* При возникновении количества ошибок, кратного двум (например, двух ошибок на позициях l и m), код Хемминга с dmin​=3 не позволяет однозначно идентифицировать ошибки, поскольку синдром будет равен сумме столбцов H, соответствующих этим позициям (S=hl​⊕hm​).

## 18. Код простой четности. Особенности программной реализации.

**Код простой четности** является простейшим избыточным кодом.1

**Характеристики** 1:

* Количество избыточных символов r всегда равно 1 и не зависит от длины информационного слова k.
* Значение этого проверочного символа будет нулевым, если сумма всех символов кодового слова по модулю 2 равна нулю.
* Код простой четности позволяет **обнаруживать все нечетные ошибки** (нечетное число ошибок), но **не позволяет их исправить**.
* Минимальное кодовое расстояние для этого кода равно 2 (dmin​=2).

**Обнаружение ошибок** 1:

* Для определения синдрома (S) достаточно вычислить дополнительный символ (Y′r), который является сверткой по модулю 2 информационного слова (Yk), а затем найти синдром: S=Yr⊕Y′r.
* Неравенство синдрома нулю означает, что получено сообщение с ошибкой (или с ошибками).1

Особенности программной реализации (подразумеваемые):

Хотя конкретный программный код для реализации кода простой четности в предоставленных материалах не приводится, его принципы крайне просты и легко реализуемы.1

1. **Вычисление проверочного бита:** Для кодирования достаточно просуммировать (или выполнить операцию XOR) все биты информационного слова. Результат этой операции (0 или 1) и будет проверочным битом.
2. **Формирование кодового слова:** Проверочный бит просто добавляется (конкатенируется) к информационному слову.
3. **Проверка на принимающей стороне:** На принимающей стороне выполняется та же операция XOR над всеми битами принятого кодового слова (включая проверочный бит). Если результат равен 0, то нечетных ошибок нет. Если результат равен 1, то обнаружена нечетная ошибка.

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

**Модифицированный код Хемминга** характеризуется минимальным кодовым расстоянием dmin​=4.1

**Возможности** 1:

* Позволяет **обнаруживать две ошибки и исправлять одну**.

**Алгоритм использования** 1:

* Алгоритм использования этого кода такой же, как и для кода Хемминга с dmin​=3.
* Принципиальное отличие состоит в **виде проверочной матрицы**.
* Проверочная матрица при неизменной длине информационного слова имеет **одну дополнительную строку и один дополнительный столбец** (в подматрице I) по сравнению с матрицей для dmin​=3.
* Запись матрицы в таком виде не считается канонической, так как подматрица I не является единичной диагональной матрицей. Для преобразования к каноническому виду используется свойство линейного кода: в дополнительную строку необходимо записать сумму по модулю 2 соответствующих символов матрицы кода с dmin​=3.
* В общем случае, любая проверочная матрица кода Хемминга с dmin​=4 имеет **нечетный вес столбцов** (вес любого из столбцов подматрицы A′ может быть равен 3, 5, 7 и т.д.).

**Интерпретация синдрома** 1:

* Равенство нулю синдрома означает отсутствие ошибок.
* Если синдром не равен нулю и имеет **нечетный вес**, то это говорит о том, что произошла **одиночная ошибка**.
* Если синдром не равен нулю и его **вес четный**, то произошла **двойная ошибка**, так как вес суммы любых двух столбцов всегда четный.

## 20. Составной код. Итеративные коды.

**Итеративные коды**, также известные как прямоугольные коды (rectangular code) или композиционные коды (product code), являются одним из самых простых классов избыточных кодов с точки зрения аппаратной реализации, позволяющих исправлять ошибки в информационных словах.1

**Основные характеристики** 1:

* **Простота реализации:** Главное достоинство — простота как аппаратной, так и программной реализации.
* **Высокая избыточность:** Основной недостаток — сравнительно высокая избыточность.
* **Структура:** Кодовые слова записываются в виде двумерной таблицы (матрицы). Могут быть построены на основе двумерных, трехмерных матриц и более высоких измерений.
* **Кодирование:** Проверочные символы вычисляются таким образом, что строки и столбцы должны содержать четное (или нечетное) число единиц. Например, для двумерной матрицы проверочные символы могут быть горизонтальными (Xh​), вертикальными (Xv​) и "паритетом паритетов" (Xhv​). Символ Xhv​ равен сумме по модулю 2 всех символов информационного слова, а также проверочных символов Xv​ и Xh​.
* **Минимальное кодовое расстояние (dmin​):** Поскольку двумерная матрица формируется как комбинация двух кодов простой четности (по каждому измерению), каждый из которых имеет dmin​=2, то полученный итеративный код (без Xhv​) будет иметь dmin​=4. Использование символа Xhv​ обеспечивает увеличение dmin​ на единицу.

**Декодирование** 1:

* Декодирование начинается немедленно, не дожидаясь поступления всего блока информации.
* Проверка соответствия избыточных символов принятого слова (Yr) с вновь вычисленными (Y'r) позволяет обнаруживать любое нечетное число искаженных символов, расположенных в одной строке или в одном столбце.
* Определение местоположения одиночной ошибки по строке указывает на наличие ошибки в этой строке, а проверка по столбцу — на конкретный символ.
* **Ограничения:** Итеративный код не может определять местоположение множественных ошибок, которые имеют четное число искаженных символов как по строкам, так и по столбцам (например, четыре искаженных символа, расположенных в вершинах прямоугольника). Это происходит потому, что четность по строкам и столбцам не нарушается.

**Многомерные итеративные коды:**

* Код считается многомерным, если число измерений, по которым рассчитываются и анализируются паритеты, составляет не менее 3.1
* Простейший многомерный линейный итеративный код — трехмерный.1 К горизонтальным и вертикальным проверкам могут быть добавлены диагональные (Xd​).1

## 21. Оценка эффективности использования корректирующего кода в ИВС.

Оценка эффективности использования корректирующего кода в информационно-вычислительных системах (ИВС) является многоаспектной задачей, требующей учета различных параметров и компромиссов.1

**Ключевые метрики эффективности** 1:

* **Степень кодирования (k/n):** Прямо измеряет эффективность кода, показывая долю полезной информации в общем объеме кодового слова. Чем выше k/n, тем меньше избыточности и выше эффективность использования канала.
* **Корректирующая/обнаруживающая способность (dmin​, tu​, t0​):** Определяет, сколько ошибок код может обнаружить (t0​) и исправить (tu​). Чем выше эти показатели, тем надежнее код.
* **Избыточность (r):** Количество добавленных проверочных бит (r=n−k). Меньшая избыточность означает меньшие накладные расходы.
* **Сложность реализации:** Относится к простоте аппаратной и программной реализации кода.
* **Задержка:** Время, необходимое для кодирования, передачи (особенно с перемежением) и декодирования.
* **Вычислительная стоимость:** Ресурсы процессора и памяти, необходимые для выполнения операций кодирования/декодирования.

**Компромиссы** 1:

* Повышение надежности (например, увеличение dmin​) часто приводит к увеличению избыточности (снижению степени кодирования).
* Более высокая способность (например, dmin​=4 по сравнению с dmin​=3) может означать небольшое увеличение сложности и избыточности.
* Коррекция пакетов ошибок с помощью перемежения добавляет задержку и увеличивает сложность системы.

Метод оценки эффективности (на основе лабораторной работы) 1:

Для количественной оценки корректирующей способности кода может использоваться следующий подход:

1. Для определенного кодового слова Xn и заданной кратности ошибок (например, 3 ошибки) генерируется N1​ вариантов этого кодового слова (Yn) с ошибками.
2. Подсчитывается количество случаев (N2​), когда кратность ошибки была правильно идентифицирована.
3. Подсчитывается количество случаев (N3​), когда все ошибки были правильно скорректированы.
4. Вычисляются соотношения N2​/N1​ (эффективность обнаружения) и N3​/N1​ (эффективность коррекции).

## 22. Особенности использования перемежителей в ИВС.

**Перемежители** (интерливеры) — это устройства (аппаратные) или программные средства, которые переупорядочивают (перемешивают) символы передаваемого сообщения или кодового слова.1

**Назначение и принцип действия** 1:

* Основная цель использования перемежителей — борьба с **группами (пакетами) ошибок**, которые часто встречаются в каналах передачи и устройствах памяти (особенно магнитной и полупроводниковой).
* Перемежитель трансформирует сгруппированные ошибки в случайные (обычно одиночные) ошибки, что значительно упрощает и ускоряет процесс их исправления с помощью относительно простых корректирующих кодов.
* Суть заключается в том, что биты каждого кодового слова передаются не в обычной последовательности, а через интервалы, превышающие ожидаемую длину пакета ошибок. В промежутки между битами одного слова вставляются биты других кодовых слов.
* На принимающей стороне используется **деперемежитель**, который восстанавливает исходную последовательность символов, "размазывая" сгруппированную ошибку по всему набору кодовых слов, составляющих сообщение.

**Преимущества** 1:

* Позволяет почти полностью устранить влияние помех, преобразуя пакеты ошибок в случайные, которые затем могут быть исправлены классическими методами (линейные итеративные коды, сверточные коды, турбо-коды).

**Ключевые характеристики перемежителей** 1:

* **Глубина перемежения:** Минимальное расстояние (в битах), на которое разносятся соседние символы входной последовательности. Большая глубина позволяет исправлять более длинные пакеты ошибок, но увеличивает задержку и сложность аппаратной/программной реализации.
* **Время перемежения:** Время, необходимое для выполнения операции перемежения.
* **Рандомизация бит:** Местоположение любого бита в выходной последовательности должно отличаться от его местоположения в исходной последовательности.

**Типы перемежителей** 1:

* **Блочное перемежение:** Простейший метод, основанный на формировании прямоугольной матрицы, из которой символы считываются по столбцам бит за битом через определенные интервалы. Применяется в GSM.
* **S-случайное перемежение:** Выбирает следующую позицию символа случайным образом, обеспечивая минимальное расстояние S от предыдущих S символов в выходной последовательности.
* **Циклически-сдвиговое перемежение:** Описывается выражением π(i)=(p⋅i+s)(modK), где s — размер сдвига, p — размер шага.
* Другие методы: псевдослучайный, сверточный, случайный, диагональный, многошаговый.

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

Совместное использование перемежителей/деперемежителей и корректирующих кодов в информационно-вычислительных системах (ИВС) является распространенной практикой, направленной на повышение надежности передачи данных, особенно в условиях воздействия помех, вызывающих **пакетные ошибки**.1 Хотя существуют специальные коды для коррекции пакетных ошибок, на практике чаще применяют перемежение/деперемежение в сочетании с традиционными кодами.1

**Принцип совместного использования** 1:

1. **Кодирование:** Исходная информация (Xk) сначала кодируется с помощью помехоустойчивого кода (например, кода Хемминга), формируя кодовое слово (Xn).
2. **Перемежение:** Кодовое слово Xn затем поступает на перемежитель. Перемежитель переупорядочивает символы Xn таким образом, чтобы биты каждого кодового слова передавались не последовательно, а через интервалы, превышающие ожидаемую длину пакета ошибок. Это "размазывает" потенциальные пакетные ошибки по всему сообщению.
3. **Передача:** Перемеженная последовательность передается по каналу связи, где на нее могут воздействовать помехи.
4. **Деперемежение:** На принимающей стороне сообщение сначала проходит через деперемежитель, который восстанавливает исходный порядок символов. В результате этого процесса пакетная ошибка, которая затронула последовательные биты в перемеженном потоке, теперь распределяется как одиночные (или с низкой кратностью) ошибки по различным кодовым словам.
5. **Декодирование:** Деперемеженный сигнал, содержащий теперь случайные ошибки, поступает на декодер корректирующего кода. Поскольку большинство традиционных кодов (например, код Хемминга) высокоэффективны для исправления одиночных или случайных ошибок, они могут успешно восстановить исходную информацию.

**Преимущества и синергия** 1:

* Эта двухэтапная процедура декодирования (деперемежение с последующей коррекцией ошибок) позволяет **почти полностью устранить влияние помех**.
* Пакетные ошибки трансформируются в одиночные (формально — независимые) ошибки, что делает их легко исправляемыми простыми кодами, такими как код Хемминга.
* Это позволяет использовать менее сложные корректирующие коды, чем те, которые потребовались бы для прямой коррекции пакетных ошибок, что упрощает аппаратную и программную реализацию системы.

Примером совместного использования является стандарт GSM, где применяется блочное перемежение в сочетании с корректирующими кодами.1

## 24. Математические основы построения и использования циклических кодов.

**Циклические коды (ЦК)** представляют собой семейство помехоустойчивых кодов, которые относятся к классу линейных и систематических кодов.1

**Основные свойства** 1:

* **Линейность и систематичность:** Сумма по модулю 2 двух разрешенных кодовых комбинаций также дает разрешенную кодовую комбинацию.
* **Циклическая перестановка:** Каждый вектор (кодовое слово), получаемый из исходного кодового вектора путем циклической перестановки его символов, также является разрешенным кодовым вектором. Например, если 1101100 — кодовое слово, то 0110110 также является разрешенной комбинацией.
* **Минимальное кодовое расстояние (dmin​):** Поскольку нулевая комбинация (000...00) является разрешенной, dmin​ для ЦК определяется минимальным весом разрешенной кодовой комбинации.

**Полиномиальное представление** 1:

* Описание и использование циклических кодов основывается на **полиномах**. Двоичные последовательности могут быть переведены в полиномиальную форму (например, 101100 соответствует X5+X3+X2).
* Все операции с кодовыми словами в полиномиальной форме выполняются по правилам **арифметики по модулю 2** (сложение и вычитание эквивалентны XOR). Например, Xn+1=0.

**Порождающие полиномы (G(X))** 1:

* ЦК описываются порождающими полиномами G(X) степени r=n−k, где r — число проверочных символов.
* Все разрешенные кодовые комбинации Bj​(X) делятся на G(X) без остатка: Bj​(X)/G(X)=Aj​(X), где Aj​(X) — информационное слово.
* G(X) должен быть делителем бинома Xn+1.

**Порождающая матрица (G)** 1:

* Строки матрицы G (k×n) формируются на основе порождающего полинома G(X), например, G(x),xG(x),…,xk−1G(x).
* Матрица G может быть преобразована к **каноническому виду** (Gk​=[I∣P]), где I — единичная подматрица, что упрощает кодирование.

**Проверочная матрица (H)** 1:

* Для каждой G существует проверочная матрица H (r×n), удовлетворяющая G⋅HT=0.
* H может быть получена из G(X) по формуле H(X)=(Xn+1)/G(X).
* Канонический вид H=.

## 25. Кодирование сообщения циклическим кодом.

Кодирование сообщения циклическим кодом (ЦК) чаще всего основывается на **методе деления полиномов**.1 Этот метод позволяет представить разрешенные к передаче кодовые комбинации в виде разделенных информационных (Xk) и проверочных (Xr) символов, то есть получить блочный код.1

**Алгоритм формирования разрешенных кодовых комбинаций в кодере состоит из трех этапов** 1**:**

1. **Дописывание нулей:** К комбинации слова Xk справа приписывается r нулей. Это эквивалентно умножению Xk на оператор сдвига Xr.
2. **Деление на порождающий полином:** Произведение Xk⋅Xr делится на соответствующий порождающий полином G(X). Определяется остаток R(X), степень которого не превышает (r−1). Этот остаток R(X) и представляет собой группу проверочных символов (Xr).
3. **Формирование кодового слова:** Вычисленный остаток R(X) присоединяется справа к Xk. Таким образом, итоговая кодовая комбинация (кодовое слово) Xn формируется как Xn=Xk⋅Xr∣∣R(X).

Пример кодирования 1:

Пусть информационная последовательность Xk=1001, что соответствует полиному X3+1.

Для циклического кода (7, 4) (где n=7,k=4,r=3) выберем порождающий полином G(X)=X3+X+1 (этот полином соответствует коду Хемминга).

1. Умножение Xk на Xr:  
   Xk⋅Xr=(X3+1)⋅X3=X6+X3.  
   В двоичном представлении это 1001000 (длина n=7).
2. Деление Xk⋅Xr на G(X):  
   Выполняется деление полинома X6+X3 на X3+X+1.  
   Результат деления: X3+X.  
   Остаток R(X)=X2+X, что в двоичном виде соответствует 110. Это и есть проверочные символы Xr.
3. Формирование итогового кодового слова Xn:  
   Xn=Xk∣∣R(X)=1001∣∣110=1001110.

## 26. Матрицы циклического кода и кода Хемминга.

Матричные представления являются фундаментальными для построения и анализа как циклических кодов, так и кодов Хемминга.

**Матрицы циклического кода:**

* **Порождающая матрица (G):** Строки матрицы G (k×n) являются векторами, полученными из порождающего полинома G(X). Например, для ЦК (7,4) с G(X)=X3+X+1, строки G будут G(x),xG(x),x2G(x),x3G(x).1
  + Пример G(7,4):  
    1011000  
    0101100  
    0010110  
    0001011  
    1
  + Для формирования разделимого блочного кода матрица G может быть преобразована к **каноническому виду** (Gk​=[I∣P]), где I — единичная подматрица порядка k.1
    - Пример канонической Gk​(7,4):  
      1000 101  
      0100 111  
      0010 110  
      0001 011  
      1
* **Проверочная матрица (H):** Для циклического кода H (r×n) строится на основе полинома H(X)=(Xn+1)/G(X).1 Она удовлетворяет равенству G⋅HT=0.1
  + Пример H7,4​ в каноническом виде:  
    1110 100  
    0111 010  
    1101 001  
    1

**Матрицы кода Хемминга:**

* **Порождающая матрица (G):** Для линейных блочных кодов, к которым относится код Хемминга, G имеет вид G=[I∣P].1
* **Проверочная матрица (H):** Для кода Хемминга с dmin​=3, проверочная матрица H имеет классический вид и состоит из двух подматриц: A′ (размером k×r) и I (единичная матрица размером r×r).1
  + Пример H7,4​ для кода Хемминга:  
    1101001  
    1011010  
    0111100  
    1
* **Модифицированный код Хемминга (dmin​=4):** Проверочная матрица для этого кода отличается от dmin​=3 тем, что имеет одну дополнительную строку и один дополнительный столбец (в подматрице I) по сравнению с матрицей для dmin​=3.1

## 27. Синдромный метод декодирования циклического кода.

Синдромный метод декодирования циклического кода включает следующие шаги:

1. **Деление принятого кодового слова:** Основная операция — деление принятого кодового слова (Yn) на порождающий полином G(X), который использовался при кодировании.1
   * Если Yn принадлежит коду (то есть не искажено помехами), остаток от этого деления (синдром) будет нулевым.1
   * Ненулевой остаток свидетельствует о наличии ошибок в принятой кодовой комбинации (Yn=Xn).1
2. **Определение вектора ошибки:** Если синдром не равен нулю, необходимо определить вектор (полином) ошибки (En).1
   * Синдром для циклических кодов обладает теми же свойствами, что и для кодов Хемминга, и используется для декодирования синдрома.1
   * Каждый ненулевой синдром соответствует определенному расположению (конфигурации) ошибок.1
3. **Исправление ошибки:** После определения вектора ошибки (En), исходное кодовое слово (Xn) может быть восстановлено путем сложения (по модулю 2) принятого кодового слова (Yn) и вектора ошибки (En): Xn=Yn⊕En.1

Пример синдромного декодирования 1:

Пусть принято кодовое слово Yn=1011110, а порождающий полином G(X)=X3+X+1.

1. **Деление Yn на G(X):** Yn в полиномиальном виде: X6+X4+X3+X2+X. Деление (X6+X4+X3+X2+X) на (X3+X+1) дает остаток Sr=X2+X, что соответствует двоичному 110.
2. **Декодирование синдрома:** Синдром Sr=110 равен третьему столбцу проверочной матрицы H7,4​ (для кода Хемминга). Это указывает на то, что ошибка произошла в третьем бите. Вектор ошибки En=0010000.
3. **Коррекция ошибки:** Yn⊕En=1011110⊕0010000=1001110. Полученное исправленное слово 1001110 соответствует исходному переданному кодовому слову.

## 28. Характеристика надежности двоичного канала передачи при использовании кодов.

Надежность двоичного канала передачи при использовании кодов характеризуется способностью противодействовать ошибкам, возникающим в процессе передачи данных.

**Основные принципы повышения надежности:**

* **Избыточность:** В информационные системы сознательно вводится избыточность для повышения их функциональной надежности.1 Это достигается путем добавления дополнительных символов (битов) к исходному сообщению, которые не несут новой информации, но служат для обнаружения и исправления ошибок.1
* **Помехоустойчивое кодирование:** Это основной метод введения избыточности. Он преобразует исходное информационное сообщение (Xk) в избыточное кодовое слово (Xn) путем добавления r проверочных символов (Xr).1
* **Декодирование:** На принимающей стороне декодер использует избыточные символы для обнаружения и исправления ошибок. Если принятое сообщение (Yn) отличается от переданного (Xn), декодер должен выявить и, по возможности, скорректировать ошибки.1

**Характеристики кодов, влияющие на надежность:**

* **Степень кодирования (k/n):** Отношение числа информационных битов (k) к общей длине кодового слова (n). Показывает долю полезной информации в коде.1
* **Расстояние Хемминга (d):** Число позиций, в которых два кодовых слова различаются.1
* **Минимальное кодовое расстояние (dmin​):** Минимальное расстояние Хемминга между любыми двумя разрешенными кодовыми словами. Определяет способность кода обнаруживать (t0​) и исправлять (tu​) ошибки.1
  + t0​=d/2 (если d четно) или t0​=(d−1)/2 (если d нечетно).1
  + tu​=(d−1)/2 (если d нечетно) или tu​=(d−2)/2 (если d четно).1

**Типы кодов и их надежность:**

* **Код простой четности:** Простейший код с r=1. Обнаруживает все нечетные ошибки, но не исправляет их. dmin​=2.1
* **Код Хемминга (dmin​=3):** Позволяет обнаруживать и исправлять одиночную ошибку.1
* **Модифицированный код Хемминга (dmin​=4):** Позволяет обнаруживать две ошибки и исправлять одну.1
* **Циклические коды (ЦК):** Линейные систематические коды, обладающие свойством циклической перестановки. Описываются порождающими полиномами G(X).1 Синдромный метод декодирования ЦК позволяет обнаруживать и исправлять ошибки.1

**Борьба с пакетными ошибками (Перемежение/Деперемежение):**

* В каналах передачи часто встречаются **группы (пакеты) ошибок**, с которыми традиционные коды справляются неэффективно.1
* **Перемежители** (интерливеры) переупорядочивают символы сообщения, превращая сгруппированные ошибки в случайные (обычно одиночные), которые затем легко исправляются простыми корректирующими кодами.1
* На принимающей стороне **деперемежитель** восстанавливает исходную последовательность, "размазывая" пакетную ошибку.1
* Совместное использование перемежителей и корректирующих кодов позволяет почти полностью устранить влияние помех.1

Таким образом, надежность двоичного канала передачи при использовании кодов достигается за счет целенаправленного введения избыточности, выбора кодов с достаточным минимальным кодовым расстоянием и применения методов, таких как перемежение, для эффективной борьбы с различными типами ошибок.

## 29. Общая характеристика и классификация методов сжатия информации.

Общие характеристики методов сжатия:

Основная цель сжатия данных — обеспечить более компактное представление данных, вырабатываемых источником, то есть уменьшить физический объем сообщений и сократить время их передачи по каналам связи.1 Согласно фундаментальной теореме К. Шеннона о кодировании информации, стоимость кодирования всегда как минимум равна энтропии источника, хотя может быть сколь угодно близка к ней.1 Таким образом, для любого алгоритма сжатия всегда существует предел степени (или эффективности) сжатия, определяемый энтропией входного потока.1

Все алгоритмы сжатия преобразуют входной поток данных, где минимальной единицей является бит, а максимальной — байт или несколько байтов.1 Основные технические характеристики процессов сжатия и их результатов включают:

* **Коэффициент сжатия (R):** Отношение объема сжатых данных (Vсс​) к объему исходных данных (Vис​): R=Vсс​/Vис​.1 Также может выражаться как R′=(Vис​−Vсс​)/Vис​=1−R, показывая степень уменьшения объема.1
* **Скорость сжатия:** Время, необходимое для сжатия определенного объема информации.
* **Качество сжатия:** Насколько выходной поток сжимается при повторном применении того же или другого алгоритма. Это важно, так как большинство современных архиваторов используют несколько методов сжатия.

Классификация методов сжатия:

По характеру или формату данных, а также по степени соответствия между сжатыми и распакованными данными, все методы сжатия делятся на два класса:

1. **Обратимое сжатие (сжатие без потерь информации):** Гарантирует, что данные не теряются в процессе упаковки текстовых документов, кодов компьютерных программ или файлов баз данных.1
2. **Необратимое сжатие (сжатие с частичной потерей информации):** Включает частичную потерю информации. Применяется для "потребительских задач", например, для сжатия фотографий (алгоритм JPEG) или видео и музыки (алгоритмы MPEG).1 Для дальнейшей обработки файлов (например, редактирования) потеря информации недопустима.1

**Классы методов по используемому алгоритму:**

1. **Символориентированные и словарные методы:**
   * **Метод кодирования длин серий (RLE):** Заменяет последовательность одинаковых символов одним символом и числом его повторений.1
   * **Метод Бэрроуза-Уилера (BWT):** Преобразует блок данных в формат, исключительно подходящий для последующего сжатия, хотя сам по себе не сжимает данные (объем выходной последовательности всегда превышает исходный).1
   * **Метод Лемпеля-Зива (LZ77, LZ78):** Создает "словарь" часто встречающихся последовательностей данных и заменяет их ссылками (смещением и длиной).1
2. **Вероятностные (статистические) методы:** Присваивают разное количество битов символам в зависимости от их вероятности появления в тексте. Часто называются префиксными кодами.1
   * **Метод Шеннона-Фано:** Создает отсортированную таблицу символов алфавита и рекурсивно делит ее на две группы с приблизительно одинаковой суммарной частотой, присваивая "0" или "1" каждой группе.1
   * **Метод Хаффмана:** Строит бинарное дерево, где ветви помечены бинарными символами. Код символа — это последовательность меток от корня до листа, соответствующего символу. Цель — оптимальное сжатие путем присвоения более коротких кодов более частым символам.1
3. **Арифметические методы:** Присваивают коды не отдельным символам, а их последовательностям. Текст представляется вещественными числами в интервале от 0 до 1, который сужается по мере анализа текста.1

Современные архиваторы, как правило, строятся на базе сочетания различных методов.1

## 30. Метод сжатия данных Лемпеля-Зива.

Метод Лемпеля-Зива (ЛЗ), разработанный в 1977 году Авраамом Лемпелем и Яковом Зивом, представляет собой новый подход к сжатию данных. Основная идея метода заключается в создании "словаря" общих последовательностей данных, которые затем заменяются соответствующими кодами из этого словаря.1

Классический алгоритм LZ77, названный по году публикации, достаточно прост. Он основан на принципе, что если аналогичная последовательность байтов уже была встречена в ранее проанализированном (сжатом) выходном потоке, и если запись её длины и смещения от текущей позиции короче самой последовательности, то вместо самой последовательности в выходной файл записывается ссылка (смещение, длина).1 Метод кодирования длин серий (RLE), который заменяет последовательность одинаковых символов одним символом и его количеством, считается подклассом алгоритма LZ77.1

Суть метода LZ77 (и его последующих модификаций) заключается в том, что компрессор постоянно хранит определённое количество последних обработанных символов в буфере. По мере обработки входного потока новые символы поступают в конец буфера, сдвигая предыдущие символы и вытесняя самые старые. Размер этого буфера, также известного как скользящий словарь, варьируется в различных реализациях систем сжатия. Скользящее окно имеет длину n и состоит из двух частей:1

* Последовательность уже закодированных символов длиной n1​=n−n2​ (словарь).
* Буфер просмотра (или буфер предпросмотра) длиной n2​ (буфер кодирования).

Формально, если до текущего момента закодировано t символов S1​,S2​,...,St​, то словарем будут являться n1​ предшествующих символов: St−(n1​−1)​,St−(n1​−1)+1​,...,St​. В буфере находятся символы, ожидающие кодирования (сжатия) символы St+1​,St+2​,...,St+n2​​. Если n2​≥t, то вся уже обработанная часть входной последовательности служит словарём.1

Алгоритм ищет самое длинное совпадение между строкой буфера кодирования, начинающейся с St+1​, и всеми фразами словаря. Эти фразы могут начинаться с любого символа из St−(n1​−1)​,St−(n1​−1)+1​,...,St​, выходить за пределы словаря в область буфера, но должны оставаться в пределах окна. Буфер не может сравниваться сам с собой. Длина совпадения не должна превышать размера буфера. Полученная фраза St−(p−1)​,St−(p−1)+1​,St−(p−1)+(q−1)​ кодируется с использованием двух чисел:1

1. **Смещение (p):** от начала буфера.
2. **Длина совпадения (q):** длина соответствия или совпадения.

Эти ссылки (p- и q-указатели) однозначно определяют фразу. Дополнительно в выходной поток записывается символ s, следующий за совпавшей строкой буфера. Длина кодовой комбинации (триады — p, q, s) на каждом шаге определяется соотношением: l(Ci​)=logN​n1​+logN​n2​+1, где N — размер алфавита. После каждого шага окно сдвигается на (q+1) символов вправо, и начинается новый цикл кодирования. Величина сдвига составляет (q+1), потому что q символов были закодированы с помощью указателя, и 1 символ был скопирован тривиально. Передача одного символа в явном виде (s) решает проблему обработки символов, которые ещё не встречались, но значительно увеличивает размер сжатого блока.1

Пример прямой трансформации LZ77:

Пусть алфавит A = {0,1,2,3}, N=4. Длина словаря n1​=15, длина буфера данных (кодирования) n2​=13. Длина кодовой комбинации на каждом шаге: l(Ci​)=log4​15+log4​13+1=2+2+1=5.

Пусть входное сообщение S=2000302013020130313031303130313333333.

* **Шаг 1:** Первый символ в буфере кодирования — '2'. В словаре совпадений нет (все нули). Триада: (00,00,2)4​. Окно сдвигается на 1 позицию.1
* **Шаг 2:** Найдено повторение '000'. Длина q2​=3. Начальная позиция p2​=6. Триада: (12,03,3)4​. Окно сдвигается на 4 позиции.1
* **Шаг 3:** Найдено повторение '020'. Длина q3​=3. Начальная позиция p3​=10. Триада: (22,03,1)4​. Окно сдвигается на 4 позиции.1
* **Шаг 4:** Найдено повторение '3020130'. Длина q4​=7. Начальная позиция p4​=11. Триада: (23,13,3)4​. Окно сдвигается на 8 позиций.1 Процесс продолжается до сжатия всего сообщения.1

Пример обратной трансформации LZ77 (декомпрессии):

Пусть запакованное сообщение в 4-й системе счисления: 00002 12033 22031 23133 30301 02013 32103.... Изначально окно заполнено 15 нулями.

* **Шаг 1:** Анализируется триада (00,00,2)4​. p1​=0,q1​=0. Символ '2' записывается в окно.1
* **Шаг 2:** Анализируется триада (12,03,3)4​. p2​=6,q2​=3. Идентифицируется повторение '000'. Символ '3' добавляется. Распакованное сообщение: 20003.1
* **Шаг 3:** Анализируется триада (22,03,1)4​. p3​=10,q3​=3. Идентифицируется повторение '020'. Символ '1' добавляется. Окно: 000000200030201.1 Процесс продолжается до полного восстановления сообщения.1

**Недостатки LZ77:**

1. Невозможность кодирования подстрок, расположенных дальше длины словаря.
2. Ограничение длины кодируемой подстроки размером буфера. Увеличение размера словаря приводит к увеличению времени трансформации и длины кодовых последовательностей.1

Алгоритм LZ78:

В 1978 году создатели LZ77 разработали улучшенный алгоритм LZ78, который устраняет недостатки LZ77. LZ78 не использует "скользящее" окно. Вместо этого он хранит словарь уже просмотренных подстрок. В начале алгоритма этот словарь содержит только одну пустую строку. Алгоритм считывает символы сообщения до тех пор, пока накапливаемая подстрока полностью не будет содержаться в одной из фраз словаря. Как только эта подстрока перестаёт совпадать с какой-либо фразой словаря, алгоритм генерирует код, состоящий из индекса строки словаря, которая содержала входную строку до последнего введённого символа, и символа, вызвавшего несовпадение. Новая введённая подстрока затем добавляется в словарь. Если словарь заполнен, удаляется наименее часто используемая фраза. Ключевым фактором для размера получаемых кодов является размер словаря в терминах фраз, поскольку каждый код в этом методе имеет постоянную длину.1

Алгоритмы LZ-формата используются, когда требуется универсальное сжатие. Примеры включают стандарт модема V.42bis, протокол передачи данных ZModem, форматы GIF, TIFF и ARC, а также другие приложения. Некоторые алгоритмы этого класса используются в утилитах сжатия дисков, таких как DoubleSpace и Stacker, графических форматах, таких как PNG, и универсальных утилитах архивирования и сжатия, включая ZIP, GZIP и LHA.1

## 31. Метод сжатия данных Барроуза-Уилера.

Преобразование Бэрроуза-Уилера (BWT) — это техника сжатия информации, особенно эффективная для текстов, разработанная в 1983 году.1 BWT не сжимает данные в классическом понимании, но преобразует блок данных в формат, исключительно подходящий для последующего сжатия.1

BWT оперирует сразу целым блоком данных, который выделяется из входного потока (сообщения). Прямое преобразование (формально — сжатие) выполняется в четыре этапа:1

1. **Выделение блока данных:** Выделяется блок данных (строка длиной k символов некоторого алфавита мощностью N), который обозначается символом M.
2. **Создание таблицы циклических сдвигов:** Составляется таблица W1​ размером k×k всех циклических сдвигов входной строки M.
3. **Лексикографическая сортировка:** Производится лексикографическая (в алфавитном порядке) сортировка строк таблицы W1​, в результате чего получается таблица W2​ того же размера.
4. **Выбор выходной строки:** В качестве выходной строки (обозначим её BWT(M),z) выбирается последний столбец (Mk​) таблицы W2​ преобразования и номер строки z, совпадающей с исходной строкой M.

Важно отметить, что выходная строка (сжатое сообщение) всегда по объёму превышает входную.1

Пример прямой трансформации (сжатия):

Пусть M=«столб», k=5.

Таблица W1​ (циклические сдвиги):

столб  
толбс  
олбст  
лбсто  
бстол

После лексикографической сортировки W2​ становится:

бстол  
лбсто  
олбст  
столб <-- Исходная строка (z=4)  
толбс

Результат преобразования: Mk​=«лотбс» (последний столбец W2​) и z=4 (поскольку исходная строка «столб» является 4-й строкой в W2​).1

Обратная трансформация (декомпрессия):

Обратимость преобразования, то есть возможность извлечь M из BWT(M),z, основана на рекурсивной природе преобразования. Обратная трансформация принимает BWT(M),z в качестве входных данных и включает k идентичных шагов, каждый из которых состоит из двух операций, для реконструкции матрицы W2​:1

1. **Добавление столбца:** Последовательность символов Mk​ записывается в крайний правый пустой столбец матрицы.
2. **Сортировка столбцов:** Заполненная часть реконструируемой матрицы сортируется лексикографически по её столбцам.

После k шагов матрица W2​ будет реконструирована. Зная значение z, можно получить исходный входной блок M сообщения.1

Пример обратной трансформации:

Дано Mk​=«лотбс» и z=4.

* **Шаг 1:** Добавляем «лотбс» в крайний правый столбец. Сортируем строки на основе этого столбца.
* **Шаг 2:** Добавляем «лотбс» в следующий пустой столбец (второй справа). Сортируем строки на основе двух крайних правых столбцов. Этот процесс продолжается до полной реконструкции матрицы W2​. После получения W2​, исходное сообщение M=«столб» находится в строке, указанной z.1

BWT также может создавать бинарные последовательности с блоками повторяющихся символов, которые затем легко преобразуются в гораздо более короткие последовательности с помощью методов, таких как кодирование длин серий.1

## 32. Метод сжатия данных Хаффмана.

Метод сжатия Хаффмана — это статистическая техника сжатия данных, которая присваивает символам в сообщении двоичные коды переменной длины. Он особенно эффективен, когда частоты символов пропорциональны 1/2i, где i — натуральное положительное число. В отличие от некоторых других методов, коды Хаффмана всегда состоят из целого числа битов.1

Основная идея:

Фундаментальный принцип кодирования Хаффмана, как и других статистических методов (например, Шеннона-Фано), заключается в присвоении более коротких кодов часто встречающимся символам и более длинных кодов — менее частым. Этот подход направлен на уменьшение общего размера данных.1

Построение дерева (кодирование):

Двоичные коды в методе Хаффмана генерируются на основе бинарного дерева, где ветви помечены двоичными символами (0 и 1). Процесс построения этого дерева начинается с:1

1. **Сортировка символов:** Расположите символы исходного алфавита в порядке убывания их вероятностей (частот) появления в сообщении.
2. **Объединение наименее вероятных символов:** Выберите два символа с наименьшими вероятностями и объедините их в новый "узел" или "виртуальный символ".
3. **Присвоение меток ветвям:** Присвойте ветвям, исходящим из этого нового узла, метки '1' и '0'. Обычно ветви, соответствующие символам с большей вероятностью, получают '1', а с меньшей — '0'.
4. **Итеративный процесс:** Этот новый виртуальный символ рассматривается как обычный символ с вероятностью, равной сумме вероятностей его составляющих. Два исходных символа исключаются из дальнейшего рассмотрения. Этот процесс повторяется рекурсивно: новый набор символов (включая виртуальные) сортируется, и два наименее вероятных объединяются в новый узел, пока не останется только один корневой узел.
5. **Формирование корня:** Корень дерева формируется двумя символами с наибольшими вероятностями.

Генерация кода:

Двоичный код для каждого исходного символа определяется путем прослеживания пути от корня дерева до листового узла, соответствующего символу, и сбора двоичных меток вдоль ветвей.1

Оптимальность:

Цель состоит в создании таблицы кодов, которая приводит к минимальному значению интегрального коэффициента 'C', представляющего собой среднее количество информации на символ в битах:

C=∑p(ai​)⋅li​, где p(ai​) — вероятность символа ai​, а li​ — длина его двоичного кода.1

Дэвид Хаффман создал оптимальное дерево для английского алфавита, которое считается наилучшим, поскольку дает наименьшее значение 'C'.1

## 33. Метод сжатия данных Шеннона-Фано.

Метод Шеннона-Фано представляет собой статистический метод сжатия, который присваивает символам двоичные коды переменной длины в зависимости от их вероятности появления. Этот метод относится к сжатию без потерь, что означает отсутствие потери информации в процессе сжатия и распаковки.1

Эффективность метода Шеннона-Фано, как и статистического сжатия в целом, обусловлена присущей естественным языкам и потокам данных избыточностью. Использование неравномерного распределения частот символов позволяет достигать значительной степени сжатия. Например, в английском языке буква «e» встречается гораздо чаще, чем «z». Эта неравномерность означает, что некоторые символы несут меньше «неопределенности» или «информации», поскольку их появление более предсказуемо. Присваивая более короткие коды этим предсказуемым (высокочастотным) символам, общее количество битов, необходимое для представления сообщения, сокращается, что и приводит к сжатию. Это подчеркивает, что сжатие не «удаляет» информацию, а представляет ее более эффективно за счет устранения статистической избыточности.

Процедура сжатия включает несколько этапов:

* **Вычисление вероятностей:** Определяются вероятности (или частоты) появления каждого символа (p(aᵢ)) в алфавите или сообщении. Этот процесс может быть статическим (с использованием заранее определенных вероятностей) или динамическим (специфичным для конкретного сообщения).1
* **Сортировка символов:** Символы сортируются в порядке убывания их вероятностей.1
* **Присвоение двоичных кодов (рекурсивное деление):** Отсортированный набор символов делится на две группы таким образом, чтобы суммы вероятностей в каждой группе были максимально близки к 0,5. Одной группе присваивается «0» в качестве первого бита, а другой – «1». Эта процедура рекурсивно повторяется для подгрупп, содержащих более одного символа, до тех пор, пока каждая подгруппа не будет содержать только один символ.1

В таблице 1 представлен пример создания кодовой таблицы Шеннона-Фано.

**Таблица 1. Пример создания кодовой таблицы Шеннона-Фано**

|  |  |  |  |
| --- | --- | --- | --- |
| **Символ** | **Вероятность (p(aᵢ))** | **Отсортированный порядок** | **Двоичный код** |
| a4 | 0.20 | 1 | 11 |
| a5 | 0.15 | 2 | 101 |
| a9 | 0.15 | 3 | 100 |
| a6 | 0.12 | 4 | 011 |
| a2 | 0.10 | 5 | 010 |
| a8 | 0.10 | 6 | 0011 |
| a1 | 0.05 | 7 | 0010 |
| a7 | 0.05 | 8 | 0001 |
| a10 | 0.05 | 9 | 00000 |
| a3 | 0.03 | 10 | 00001 |

Таблица наглядно организует сложное отображение символов в двоичные коды переменной длины. Она четко демонстрирует результат основного алгоритма (рекурсивного деления и присвоения кодов) и позволяет легко проверить присвоение кодов на основе правил сортировки и деления вероятностей. Эта таблица является прямым результатом, используемым для фактического сжатия сообщения, что делает ее центральной для понимания практического применения метода.

Алгоритм обратного преобразования (распаковки)

Алгоритм распаковки принимает двоичную последовательность в качестве входных данных и восстанавливает исходное сообщение:

* Входными данными является двоичная последовательность. Анализируются первые l\_min (минимальная длина кода) символов последовательности.
* Осуществляется поиск совпадения в кодовой таблице.
* Если совпадение найдено: выводится соответствующий символ, и процесс повторяется для следующих l\_min символов.
* Если совпадение не найдено: длина анализируемой последовательности увеличивается на 1 бит (l\_min + 1), и поиск повторяется. Процесс продолжается до нахождения совпадения или достижения l\_max (максимальной длины кода).1
* Обработка ошибок: если совпадение не найдено даже при l\_max, это указывает на ошибку, и архиваторы могут использовать коды с коррекцией ошибок.1

Способность однозначно распаковывать объединенный двоичный поток гарантируется свойством «префиксного кода». Это означает, что ни один код символа не является префиксом другого кода. Например, если «10» является кодом, то «101» не может быть другим кодом. Это позволяет декодеру жадно сопоставлять самый длинный возможный префикс из входящего битового потока с допустимым кодом в таблице, тем самым правильно сегментируя поток на отдельные коды символов.

Ключевые характеристики, оптимальность и ограничения

Метод Шеннона-Фано характеризуется использованием кодов переменной длины, префиксных кодов (ни один код не является префиксом другого) и уникальностью кодов.1 Например, сообщение из 10 символов, которое в кодировке ASCII занимало бы 80 битов, после сжатия может быть сокращено до 34 битов, что дает коэффициент сжатия R' = 46/80.1

Однако метод имеет ограничения: он не всегда оптимален, что означает, что он может не достигать абсолютного минимума избыточности. Для одного и того же распределения вероятностей может существовать несколько кодов Шеннона-Фано, дающих различные результаты.1

Субоптимальность Шеннона-Фано по сравнению с кодированием Хаффмана проистекает из его жадного подхода к делению сверху вниз. Хотя метод стремится к равным суммам вероятностей, он не гарантирует абсолютно кратчайшую среднюю длину кода. Метод Хаффмана, напротив, достигает этого благодаря своей конструкции дерева снизу вверх. Деление Шеннона-Фано сверху вниз может приводить к субоптимальным разбиениям на ранних этапах, которые не могут быть исправлены позднее. Подход Хаффмана, постоянно объединяющий наименее вероятные символы, гарантирует, что наиболее часто встречающиеся символы оказываются ближе к корню (более короткие коды), а наименее часто встречающиеся – глубже (более длинные коды), что минимизирует общую среднюю длину кода. Это различие в стратегии построения является основной причиной гарантированной оптимальности Хаффмана по сравнению с потенциальной субоптимальностью Шеннона-Фано.

## 34. Арифметический метод сжатия данных

1. Основная идея: присвоение кодов последовательностям символов

Арифметический метод сжатия присваивает коды не отдельным символам, а последовательностям символов. Он представляет текст в виде вещественных чисел в интервале от 0 до 1. По мере анализа текста интервал, представляющий его, сужается, и количество битов, необходимое для его представления, увеличивается.1

Этот метод преодолевает ограничение дискретных длин битов, присущее кодированию Хаффмана/Шеннона-Фано, эффективно присваивая символам дробные длины битов. Это приводит к более высокой эффективности сжатия, особенно когда вероятности символов не являются степенями 2. Представляя все сообщение как одно вещественное число в интервале , длина этого интервала (и, следовательно, точность, необходимая для представления числа в нем) может быть сколь угодно близка к теоретическому пределу энтропии, что фактически позволяет использовать «дробные» биты на символ в среднем. Это является ключевым преимуществом для достижения более высоких коэффициентов сжатия.

2. Алгоритм прямого преобразования (сжатия)

* **Инициализация:** На нулевом шаге устанавливается начальный рабочий интервал [a0 = 0, b0 = 1].1
* **Итерация:** Для каждого символа (αj) в последовательности:
  + Находится соответствующий ему сегмент в текущем рабочем интервале [Li-1, Hi-1].
  + Этот сегмент становится новым рабочим интервалом [Li, Hi].
  + Новые границы вычисляются по формулам: Hi(αj) = Li-1 + (Hi-1 – Li-1) · H(αj)0; Li(αj) = Li-1 + (Hi-1 – Li-1) · L(αj)0.1
* **Конечный результат:** Любое число из конечного, суженного рабочего интервала, обычно его нижняя граница, представляет сжатое сообщение.1

В таблице 2 представлен пример арифметического кодирования сообщения «молоко» с использованием заданных вероятностей символов.

**Таблица 2. Пример арифметического кодирования сообщения «молоко»**

| Шаг | Символ | Текущий интервал [L, H] | Интервал символа [L₀, H₀] | Новые границы [L\_new, H\_new] |

| :-- | :----- | :---------------------- | :------------------------- | :---------------------------- |

| 0 | - | | - | |

| 1 | м | | [0, 0.1666] | [0, 0.1666] |

| 2 | о | [0, 0.1666] | [0.5, 1] | [0.083333, 0.1666] |

| 3 | л | [0.083333, 0.1666] | [0.1666, 0.3333] | [0.0972, 0.1111] |

| 4 | о | [0.0972, 0.1111] | [0.5, 1] | [0.104166, 0.111111] |

| 5 | к | [0.104166, 0.111111] | [0.3333, 0.5] | [0.10648148, 0.10763888] |

| 6 | о | [0.10648148, 0.10763888] | [0.5, 1] | [0.107060185, 0.10763888] |

Конечный результат сжатия: 0.107060185 (округлено до 0.1071).

Таблица эффективно визуализирует динамическое сужение интервала, что является основным механизмом арифметического кодирования. Она четко демонстрирует применение рекурсивных формул для обновления L и H на каждом шаге и позволяет проследить процесс сжатия для данного сообщения, показывая, как каждый символ влияет на конечное сжатое число. Это также подчеркивает отличие от кодирования символ за символом, демонстрируя единый, непрерывно уточняемый интервал для всего сообщения.

3. Алгоритм обратного преобразования (распаковки)

Для восстановления исходного сообщения необходима информация о значении сжатого числа, количестве символов в сжатом сообщении и таблице вероятностей символов исходного сообщения.1

* **Метод 1 (прямой расчет):** На каждом шаге определяется интервал, в который попадает текущий код, выводится соответствующий символ, и вычисляется новый код по формуле: code\_i = [code\_(i-1) − L(α\_(i-1))0] / [H(α\_(i-1))0 − L(α\_(i-1))0].1
* **Метод 2 (свойство рекуррентности):** Итеративно вычисляются новые границы интервала на основе восстановленного символа и проверяется, в какой подинтервал попадает входное число для восстановления исходного сообщения.1

4. Ключевые особенности, соображения по точности и сравнение с префиксными методами

Ключевые особенности арифметического метода включают то, что он не передает ничего до полного анализа всего текста, а результат обычно представляется в формате целочисленной арифметики. Точность, необходимая для представления интервала, возрастает с длиной текста, что требует тщательной обработки потенциального переполнения (схлопывания интервала).1

Необходимость увеличения точности с ростом длины сообщения выявляет фундаментальную вычислительную проблему в арифметическом кодировании. Это прямое следствие представления всего сообщения в сужающемся интервале, что требует большего количества битов для различения конечного, очень малого интервала. Если интервал [Hi; Li] сужается с каждым обработанным символом, то для уникальной идентификации числа в очень малом интервале требуется большее количество битов (большая точность). Это напрямую приводит к необходимости использования арифметики произвольной точности или тщательных методов масштабирования в программных реализациях для предотвращения численного переполнения или потери точности. Это объясняет, почему, несмотря на свою теоретическую эффективность, арифметическое кодирование может быть более вычислительно интенсивным и сложным в правильной реализации по сравнению с более простыми методами, такими как кодирование Хаффмана, которые на каждом шаге оперируют только с целочисленными кодами фиксированного размера.

По сравнению с методами Хаффмана/Шеннона-Фано, арифметический метод, как правило, более эффективен, особенно когда частоты символов не являются степенями 2.1

## 35. Базовые понятия криптографии. Основы теории больших чисел. Решето Эратосфена.

1. Фундаментальные определения: целые числа, натуральные числа, делимость, простые и составные числа

* **Целые числа (Z):** Множество всех целых чисел, включая отрицательные, ноль и положительные (..., -2, -1, 0, 1, 2,...).1
* **Натуральные числа (N):** Подмножество целых чисел, состоящее из положительных чисел (1, 2, 3,...).1
* **Делимость:** Для целого числа a и натурального числа b говорят, что a делится на b, если существует целое число q такое, что bq = a. При этом любое натуральное число является делителем нуля, единица является делителем любого целого числа, и любое натуральное число является делителем самого себя.1
* **Собственный делитель:** Делитель a числа b называется собственным, если 1 < |a| < |b|. Наименьший положительный собственный делитель составного числа n не превышает sqrt(n).1
* **Простое число:** Натуральное число n > 1, не имеющее других положительных делителей, кроме 1 и самого себя. Примеры: 2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 73, 2521. Количество простых чисел бесконечно.1
* **Составное число:** Натуральное число n > 1, имеющее хотя бы один положительный делитель, отличный от 1 и n. Примеры: 14 (делится на 2 и 7), 15 (делится на 3 и 5).1
* **Единица:** Не считается ни простым, ни составным числом.1

Определения простых и составных чисел являются основополагающими, устанавливая бинарную классификацию для всех натуральных чисел, больших единицы. Это разделение не просто академическое; оно лежит в основе безопасности современной асимметричной криптографии, которая опирается на вычислительную сложность обращения операций, связанных с этими двумя типами чисел. Безопасность таких алгоритмов, как RSA 1, зависит от «проблемы факторизации» 1, то есть от сложности разложения больших составных чисел на их простые множители. Если бы было легко факторизовать большие числа, то открытый ключ (составное число) мог бы быть легко использован для вывода закрытого ключа (связанного с его простыми множителями), что привело бы к взлому шифрования. Таким образом, присущая математическая особенность простых и составных чисел, а именно сложность факторизации, является краеугольным камнем цифровой безопасности.

2. Свойства простых чисел

* **Уникальная факторизация:** Любое составное число может быть однозначно представлено в виде произведения простых чисел (Основная теорема арифметики). Например, 39 616 304 = 2^4 · 7^2 · 13^3 · 23.1
* **Бесконечность:** Простых чисел бесконечно много; примерно n/ln(n) простых чисел меньше n.1
* **Наименьший простой делитель:** Наименьший простой делитель составного числа n не превышает sqrt(n). Это свойство используется при проверке числа на простоту.1
* **Гипотеза Гольдбаха (подразумевается):** Любое четное число, большее 2, представимо в виде суммы двух простых чисел; любое нечетное число, большее 5, представимо в виде суммы трех простых чисел.1
* **Постулат Бертрана (подразумевается):** Для любого натурального n > 1 существует хотя бы одно простое число в интервале (n, 2n).1
* **Простые-близнецы:** Два простых числа, отличающиеся на 2 (например, 5 и 7).1
* **Правила проверки на простоту:** Правила, основанные на делимости на 2, 3, 5, 7, 11, 13 (например, по последней цифре, сумме цифр, знакопеременной сумме цифр, сумме/разности троек цифр).1

«Проблема факторизации» — вычислительная сложность разложения больших чисел на простые множители — определяет криптостойкость асимметричных алгоритмов, таких как RSA.1 Основная теорема арифметики 1 утверждает, что каждое составное число однозначно представляется в виде произведения простых чисел. Хотя умножить два больших простых числа легко, обратный процесс (нахождение простых чисел по их произведению) вычислительно очень сложен для достаточно больших чисел. Асимметричное шифрование 1, в частности RSA 1, строит свой открытый ключ из произведения двух больших простых чисел, а безопасность его закрытого ключа зависит от знания этих простых чисел. Присущая сложность факторизации гарантирует, что злоумышленник, даже имея открытый ключ, не сможет эффективно вывести закрытый ключ, обеспечивая тем самым безопасность связи. Это является прямой причинно-следственной связью между математическим свойством и практической безопасностью.

3. Алгоритм «Решето Эратосфена» для генерации простых чисел

* **Назначение:** Находит все простые числа до заданного целого числа n.1
* **Шаги:**
  1. Записать подряд все целые числа от 2 до n. Присвоить переменной s значение 2.
  2. Удалить из списка числа, кратные s (2s, 3s,...).
  3. Найти первое оставшееся число в списке, большее s, и присвоить его s.
  4. Повторять шаги 2 и 3 до тех пор, пока s^2 не превысит n.1
* **Пример:** Для n=15 найденные простые числа: 2, 3, 5, 7, 13.1
* **Эффективность:** Описывается как «наименее эффективный» среди современных алгоритмов.1

В таблице 3 представлен пример работы алгоритма «Решето Эратосфена».

**Таблица 3. Пример работы алгоритма «Решето Эратосфена» для n=15**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Исходный список чисел** | **Шаг 1 (s=2): Удаление кратных 2** | **Шаг 2 (s=3): Удаление кратных 3** | **Шаг 3 (s=5): Удаление кратных 5** | **Конечный список простых чисел** |
| 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15 | 2, 3, -, 5, -, 7, -, 9, -, 11, -, 13, -, 15 | 2, 3, -, 5, -, 7, -, -, -, 11, -, 13, -, - | 2, 3, -, 5, -, 7, -, -, -, 11, -, 13, -, - | 2, 3, 5, 7, 11, 13 |

Таблица наглядно демонстрирует пошаговый процесс работы «Решета Эратосфена», делая логику исключения чисел понятной. Она помогает понять, как составные числа систематически идентифицируются и удаляются для выделения простых чисел. Хотя алгоритм прост, таблица неявно показывает его итеративную природу, которая для очень больших чисел становится вычислительно интенсивной, подтверждая, почему он считается «наименее эффективным» для крупномасштабной генерации простых чисел в криптографии.

## 36. Основная теорема арифметики. Алгоритм Евклида нахождения НОД.

1. Основная теорема арифметики и уникальная факторизация на простые множители

Основная теорема арифметики гласит, что каждое натуральное число n > 1 может быть однозначно представлено в виде произведения простых множителей.1 Например, число 39 616 304 может быть записано как 2 · 2 · 2 · 2 · 7 · 7 · 13 · 13 · 13 · 23.1

Эта теорема является краеугольным камнем, на котором строятся многие криптографические алгоритмы, особенно асимметричные, такие как RSA. Она гарантирует, что простые числа являются фундаментальными «атомами» целых чисел, а сложность разложения больших составных чисел на эти атомы (факторизация) используется для обеспечения безопасности. Неповторимость факторизации означает, что каждое составное число имеет уникальный «отпечаток», состоящий из простых чисел. Криптографические алгоритмы, такие как RSA 1, основываются на генерации открытого ключа (большого составного числа n = p\*q), где p и q — большие простые числа. Безопасность закрытого ключа зависит от знания этих p и q. Уникальность факторизации означает, что если бы злоумышленник мог эффективно найти p и q из n, он мог бы взломать закрытый ключ. Вычислительная сложность этой факторизации для очень больших n делает RSA безопасным. Теорема обеспечивает теоретическую уверенность в том, что такие p и q существуют и являются уникальными, что делает схему математически обоснованной.

2. Наибольший общий делитель (НОД) и его значение

Наибольший общий делитель (НОД) двух целых чисел a и b — это наибольшее целое число, которое делит оба числа без остатка. Обозначается НОД (a, b) или GCD(a, b).1 Например, НОД (24, 32) = 8.1

Понятие НОД напрямую связано с «взаимно простыми» числами, которые необходимы для существования обратных элементов по модулю. Обратные элементы по модулю критически важны для расшифрования во многих криптосистемах с открытым ключом, включая RSA и аффинные шифры. Два числа являются взаимно простыми, если их НОД равен 1.1 Для существования обратного элемента a^-1 mod n, числа a и n должны быть взаимно простыми.1 В RSA открытый показатель e должен быть взаимно простым с функцией Эйлера φ(n).1 В аффинных шифрах коэффициент a должен быть взаимно простым с размером алфавита N.1 Без возможности эффективного вычисления НОД (и, следовательно, определения взаимной простоты) было бы сложно генерировать допустимые ключи, обеспечивающие обратимое шифрование/расшифрование в этих системах. Алгоритм Евклида обеспечивает это эффективное вычисление.

3. Алгоритм Евклида для вычисления НОД

Алгоритм Евклида — это простой и эффективный метод вычисления НОД двух целых чисел a и b. Он основан на правиле деления: ai = bi \* qi + ri, где ri — остаток. Последний ненулевой остаток в этой последовательности делений является НОД.1 Например, НОД (1234, 54) = 2. Последовательность шагов: 1234 = 54\*22 + 46, 54 = 46\*1 + 8, 46 = 8\*5 + 6, 8 = 6\*1 + 2, 6 = 2\*3 + 0.1

Расширенный алгоритм Евклида используется для нахождения целых чисел u и v таких, что au + bv = НОД(a, b) (тождество Безу). Это имеет решающее значение для нахождения обратных элементов по модулю.1

В таблице 4 представлен пример вычисления НОД с использованием алгоритма Евклида.

**Таблица 4. Пример вычисления НОД с использованием алгоритма Евклида**

| Шаг | Делимое (aᵢ) | Делитель (bᵢ) | Частное (qᵢ) | Остаток (rᵢ) | Выражение остатка |

| :-- | :----------- | :------------ | :----------- | :----------- | :---------------- |

| 1 | 1234 | 54 | 22 | 46 | 46 = 1234 - 54 \* 22 |

| 2 | 54 | 46 | 1 | 8 | 8 = 54 - 46 \* 1 |

| 3 | 46 | 8 | 5 | 6 | 6 = 46 - 8 \* 5 |

| 4 | 8 | 6 | 1 | 2 | 2 = 8 - 6 \* 1 |

| 5 | 6 | 2 | 3 | 0 | |

Последний ненулевой остаток равен 2, следовательно, НОД (1234, 54) = 2.

Таблица обеспечивает четкое, пошаговое визуальное представление алгоритма Евклида, явно показывая, как НОД выводится как последний ненулевой остаток. Эта таблица является прямым входом для процесса «обратной подстановки» расширенного алгоритма Евклида, который жизненно важен для вычислений обратных элементов по модулю.

## 37. Основы модулярной арифметики. Вычеты.

1. Определение и обозначение модулярной арифметики

Понятие «модулярная арифметика» было введено К.Ф. Гауссом. В этой арифметике основное внимание уделяется остатку от деления числа a на число n (n – натуральное число и n > 1). Если таким остатком является число b, то это записывается как a ≡ b (mod n) или a ≡ b mod n. Такая формальная запись читается как «a сравнимо с b по модулю n». Это означает, что a = b + kn для некоторого целого числа k.1

Например, при a = 13 и n = 4 имеем b = 1, т.е. 13 = 1 + 3 · 4. Отсюда следует, что число 13 по модулю 4 равно 1, или числа 13 и 1 равны по модулю 4. Другие примеры: -5 ≡ 7 mod 4 ≡ 11 mod 4 ≡ 23 mod 4 ≡ 3 mod 4.1

Модулярная арифметика по своей сути вводит циклическое поведение, когда числа «заворачиваются» после достижения определенного модуля. Это свойство бесценно в криптографии, поскольку оно ограничивает результаты вычислений в управляемом диапазоне, предотвращая бесконечный рост чисел и делая операции вычислительно осуществимыми. В RSA, ElGamal и аффинных шифрах операции, такие как возведение в степень или умножение, могут производить чрезвычайно большие числа при выполнении в стандартной целочисленной арифметике. Выполнение этих операций по модулю большого числа n позволяет промежуточным и конечным результатам оставаться в фиксированном, управляемом диапазоне. Это предотвращает переполнение, уменьшает требования к хранению данных и делает вычисления практически применимыми даже с очень большими числами (например, 2048-битные числа в RSA). Это свойство является ключевым фактором, обеспечивающим эффективность и осуществимость криптографических операций с большими числами, что необходимо для обеспечения безопасности.

2. Свойства модулярной арифметики (коммутативность, ассоциативность, дистрибутивность)

Модулярная арифметика коммутативна, ассоциативна и дистрибутивна, подобно обычной арифметике. Это позволяет выполнять операции, такие как сложение, вычитание, умножение и возведение в степень, поэлементно. Например, (a^m mod n) ≡ (a mod n)^m.1

Тот факт, что модулярная арифметика сохраняет фундаментальные алгебраические свойства, означает, что сложные криптографические операции могут быть разработаны и проанализированы с использованием привычных алгебраических правил, даже несмотря на то, что базовые числа «заворачиваются». Это упрощает проектирование и проверку криптографических алгоритмов. Если бы эти операции вели себя непредсказуемо по модулю, проектирование и доказательство правильности алгоритмов были бы чрезвычайно сложными. Сохранение этих свойств позволяет криптографам использовать хорошо изученные алгебраические структуры (такие как группы и кольца) для построения алгоритмов. Эта предсказуемость гарантирует, что операции шифрования и расшифрования, даже если они включают сложные последовательности модульной арифметики, будут последовательно давать правильные результаты при условии, что ключи действительны.

3. Понятие вычетов и сравнений

Число b иногда называют вычетом по модулю n, а числа a и b — сравнениями по модулю n.1

Понятие вычетов по существу группирует целые числа в классы эквивалентности на основе их остатка при делении на модуль. Это формирует основу для конечных полей и колец, которые являются алгебраическими структурами, в которых работают многие криптографические алгоритмы. Числа «сравниваются» или «равны», если они имеют одинаковый остаток по модулю n.1 Это естественным образом образует «классы эквивалентности» — все целые числа, дающие одинаковый остаток, принадлежат одному и тому же классу. Криптографические алгоритмы часто работают в конечных алгебраических структурах (например, кольцо целых чисел по модулю n или конечные поля GF(p)). Эти структуры построены непосредственно на этих классах эквивалентности. Например, в криптографии на эллиптических кривых (ECC) 1 операции выполняются над конечными полями. Элементы этих полей — это именно вычеты по модулю простого числа. Это абстрактное математическое понятие обеспечивает точные конечные области, необходимые для криптографических операций, гарантируя, что вычисления не только ограничены, но и обладают желаемыми математическими свойствами (например, обратимостью для расшифрования) в рамках этих конкретных структур.

## 38. Обратные вычисления по модулю в криптографии.

1. Определение обратного элемента по модулю

В модулярной арифметике обратным элементом для a по модулю n является значение x такое, что ax ≡ 1 mod n. Это аналогично a \* (1/a) = 1 в вещественных числах.1 Например, для a = 5, n = 14, x = 3 является обратным элементом, поскольку 5 \* 3 = 15 ≡ 1 mod 14.1

2. Условия существования обратного элемента по модулю

Единственное решение для ax ≡ 1 mod n существует тогда и только тогда, когда a и n взаимно просты (т.е. НОД (a, n) = 1). В противном случае решений нет.1

Строгое условие существования обратного элемента по модулю (взаимная простота) является фундаментальным требованием для разработки обратимых криптографических операций, в частности расшифрования. Без этого зашифрованный текст не мог бы быть однозначно преобразован обратно в открытый текст. Обратный элемент a^-1 mod n существует тогда и только тогда, когда НОД(a, n) = 1.1 Операция расшифрования во многих шифрах (например, аффинных, RSA) является математически обратной операцией шифрования. Например, в аффинном шифре x ≡ a⁻¹ (y + N – b) mod N.1 Если a^-1 не существует (поскольку НОД(a, N) ≠ 1), то формула расшифрования будет неопределенной, и исходный открытый текст не может быть восстановлен из зашифрованного текста. Алгоритмы генерации криптографических ключей должны явно гарантировать, что выбранные ключи (например, a в аффинном шифре, d в RSA) удовлетворяют условию взаимной простоты с модулем, чтобы обеспечить существование их обратных элементов по модулю, тем самым обеспечивая правильное расшифрование.

3. Нахождение обратных элементов по модулю с помощью расширенного алгоритма Евклида

Обратные элементы по модулю легко находятся с помощью расширенного алгоритма Евклида.1 Расширенный алгоритм Евклида находит целые числа u и v такие, что au + nv = НОД(a, n). Если НОД(a, n) = 1, то au + nv = 1, что означает au ≡ 1 mod n. Таким образом, u является обратным элементом для a по модулю n.1

Например, чтобы решить 7y ≡ 1 mod 40:

* НОД(7, 40) = 1.
* Расширенный алгоритм Евклида: 1 = 5 - 2\*2 = 5 - 2\*(7 - 5\*1) = 5\*3 + 7\*(-2) = (40 - 7\*5)\*3 + 7\*(-2) = 3\*40 + 7\*(-17).
* Таким образом, 7\*(-17) ≡ 1 mod 40. Поскольку -17 mod 40 = 23, обратный элемент y = 23.1

В таблице 5 представлен пример вычисления обратного элемента по модулю с использованием расширенного алгоритма Евклида.

**Таблица 5. Пример вычисления обратного элемента по модулю (7y ≡ 1 mod 40)**

| Шаг | Делимое (aᵢ) | Делитель (bᵢ) | Частное (qᵢ) | Остаток (rᵢ) | Выражение остатка (rᵢ = aᵢ - bᵢqᵢ) |

| :-- | :----------- | :------------ | :----------- | :----------- | :--------------------------------- |

| 1 | 40 | 7 | 5 | 5 | 5 = 40 - 7 \* 5 |

| 2 | 7 | 5 | 1 | 2 | 2 = 7 - 5 \* 1 |

| 3 | 5 | 2 | 2 | 1 | 1 = 5 - 2 \* 2 |

Обратная подстановка:

1 = 5 - 2 \* 2

1 = 5 - 2 \* (7 - 5 \* 1) = 5 - 2 \* 7 + 2 \* 5 = 3 \* 5 - 2 \* 7

1 = 3 \* (40 - 7 \* 5) - 2 \* 7 = 3 \* 40 - 15 \* 7 - 2 \* 7 = 3 \* 40 - 17 \* 7

Таким образом, 7 \* (-17) ≡ 1 mod 40. Поскольку -17 mod 40 = 23, то y = 23.

Таблица четко иллюстрирует двухфазный процесс расширенного алгоритма Евклида (прямое деление и обратная подстановка). Она показывает, как строится тождество Безу, что непосредственно приводит к обратному элементу по модулю. Это является стандартным и эффективным методом вычисления обратных элементов по модулю, что является основной операцией при генерации ключей и расшифровании для многих асимметричных шифров.

## 39. Функция Эйлера в криптографии.

1. Определение и вычисление функции Эйлера (φ(n))

Функция Эйлера, обозначаемая как φ(n), подсчитывает количество натуральных чисел, меньших или равных n, которые взаимно просты с n.1

Формула: Если n = p1^a1 \* p2^a2 \*... \* pk^ak (разложение на простые множители), то φ(n) = n \* (1 - 1/p1) \* (1 - 1/p2) \*... \* (1 - 1/pk).1

Особые случаи:

* Если p — простое число: φ(p) = p - 1.
* Если p и q — различные простые числа: φ(pq) = (p - 1)(q - 1).1

Например, φ(12) = 12 \* (1 - 1/2) \* (1 - 1/3) = 4. Взаимно простыми числами являются 1, 5, 7, 11.1

В таблице 6 представлен пример вычисления функции Эйлера.

**Таблица 6. Пример вычисления функции Эйлера (φ(12))**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Число (n)** | **Разложение на простые множители** | **Формула φ(n)** | **Результат** | **Взаимно простые числа ≤ n** |
| 12 | 2² \* 3 | 12 \* (1 - 1/2) \* (1 - 1/3) = 12 \* (1/2) \* (2/3) | 4 | 1, 5, 7, 11 |

Таблица четко демонстрирует, как применяется формула для φ(n) с использованием простых множителей, а также усиливает связь между разложением на простые множители (Основная теорема арифметики) и свойствами взаимно простых чисел. Это вычисление является прямым шагом в процессе генерации ключей для RSA, что делает его иллюстрацию ценной.

2. Обобщение малой теоремы Ферма Эйлером

Если НОД(a, n) = 1, то a^φ(n) ≡ 1 mod n.1

Эта теорема предоставляет способ вычисления обратных элементов по модулю: a^(-1) mod n ≡ a^(φ(n) - 1) mod n.1

Например, для нахождения обратного элемента 5 mod 7: φ(7) = 6. Используя формулу, 5^(6-1) mod 7 = 5^5 mod 7 = 3. Таким образом, 5\*3 ≡ 1 mod 7.1

Теорема Эйлера о тотиенте предоставляет мощный и часто более прямой метод вычисления обратных элементов по модулю, особенно когда φ(n) известно. Это критически важно для эффективности расшифрования в криптографии с открытым ключом. Хотя расширенный алгоритм Евклида 1 является общим, эта теорема предлагает альтернативу для вычисления обратного элемента по модулю. Для больших чисел модульное возведение в степень (a^(φ(n) - 1) mod n) может быть эффективно вычислено с использованием алгоритмов, таких как возведение в степень по квадрату. Это означает, что если φ(n) известно (как для законного получателя в RSA), обратный элемент по модулю может быть быстро вычислен. В RSA закрытый ключ d является обратным элементом открытого ключа e по модулю φ(n). Способность эффективно вычислять d с использованием этой теоремы является фундаментальной для процесса расшифрования RSA.

3. Значение в криптографических алгоритмах (например, RSA)

Функция Эйлера является фундаментальной для генерации открытых и закрытых ключей в RSA. n = pq, φ(n) = (p-1)(q-1). Открытый ключ e выбирается взаимно простым с φ(n). Закрытый ключ d удовлетворяет ed ≡ 1 (mod φ(n)).1

Безопасность RSA основана на вычислительной сложности факторизации n для нахождения p и q, которые необходимы для вычисления φ(n). Это означает, что злоумышленник не может легко вычислить φ(n) и, следовательно, не может легко найти закрытый ключ d с использованием теоремы Эйлера. Генерация ключей RSA напрямую использует φ(n).1 Безопасность RSA основана на сложности факторизации больших чисел.1 Для вычисления φ(n) = (p-1)(q-1) необходимо знать p и q. Нахождение p и q из n=p\*q является сложной проблемой факторизации. Злоумышленник, зная только открытый ключ (e, n), не может эффективно вычислить φ(n), поскольку он не может факторизовать n. Без φ(n) он не может вычислить закрытый ключ d с использованием свойства обратного элемента по модулю, выведенного из теоремы Эйлера. Функция Эйлера и связанная с ней теорема не просто математические курьезы; они являются неотъемлемой частью проектирования и обеспечения безопасности RSA, создавая «одностороннюю» функцию, где операции легко выполняются в одном направлении (шифрование с открытым ключом), но трудно обратимы без секретной информации (закрытый ключ, выведенный из φ(n)).

## 40. Общая классификация криптографических методов защиты информации.

1. Симметричная криптография

Симметричные криптосистемы используют один общий секретный ключ как для шифрования, так и для расшифрования. Основной проблемой является безопасное распределение этого общего секретного ключа между взаимодействующими сторонами.1

Типы симметричных шифров включают: подстановочные (моноалфавитные, полиграфические, омофонические, полиалфавитные), перестановочные, блочные и потоковые шифры.1 Примеры: шифр Цезаря, Атбаш, Виженера, Энигма, DES, AES, RC4.1

Симметричная криптография, как правило, быстрее и эффективнее для массового шифрования данных, но ее основная уязвимость заключается в безопасном обмене общим секретным ключом, что часто требует внеполосных методов или опирается на асимметричные методы для установления ключа. Симметричные системы используют один ключ для шифрования и расшифрования.1 Такая конструкция часто позволяет создавать более простые и быстрые алгоритмы по сравнению с асимметричными. Однако возникает «проблема управления ключами» 1, поскольку обеим сторонам нужен один и тот же секретный ключ, а безопасный обмен им по незащищенному каналу сложен. Это приводит к компромиссу: высокая скорость шифрования, но сложная задача распределения ключей. Эта проблема часто решается с помощью асимметричной криптографии (например, Диффи-Хеллмана, RSA) для безопасного обмена симметричным ключом.

2. Асимметричная криптография

Асимметричные шифры, или криптография с открытым ключом, используют пару математически связанных ключей: открытый ключ для шифрования и закрытый (секретный) ключ для расшифрования. Они основываются на односторонних функциях, которые легко вычислить в одном направлении, но вычислительно сложно обратить (например, проблема факторизации, проблема дискретного логарифмирования, операции на эллиптических кривых).1

Применение: шифрование/расшифрование данных, цифровая подпись, распределение секретных ключей.1 Примеры: RSA, Эль-Гамаль, Диффи-Хеллман, ECC.1

Асимметричная криптография решает проблему распределения ключей, присущую симметричным системам, позволяя осуществлять безопасную связь без предварительно общих секретов. Это обеспечивает масштабируемую безопасную связь в больших сетях, таких как Интернет. Асимметричные системы используют пары открытых/закрытых ключей.1 Открытый ключ может свободно распространяться, что решает проблему обмена ключами. Это позволяет любой стороне зашифровать сообщение для получателя, используя его открытый ключ, и только получатель (со своим закрытым ключом) может его расшифровать. Это значительно упрощает управление ключами в крупномасштабных системах связи, обеспечивая безопасную связь между сторонами, которые никогда не встречались или не обменивались секретами заранее.

3. Хеш-функции

Хеш-функции – это математические функции, которые преобразуют входную строку переменной длины в выходную строку фиксированной длины (хеш-код или дайджест сообщения).1

Свойства:

* **Детерминированность:** Один и тот же вход всегда дает один и тот же хеш.1
* **Скорость вычисления:** Эффективное вычисление хеша.1
* **Односторонность/стойкость к нахождению прообраза:** Вычислительно нецелесообразно обратить хеш для нахождения исходного входа.1
* **Чувствительность к изменениям входа:** Небольшие изменения во входе приводят к значительным изменениям в хеше.1
* **Коллизионная стойкость:**
  + **Коллизия 1-го рода (стойкость к нахождению первого прообраза):** Трудно найти другой вход, который дает тот же хеш, что и данный вход.1
  + **Коллизия 2-го рода (стойкость к нахождению второго прообраза):** Трудно найти любые два разных входа, которые дают один и тот же хеш.1

Применение: хранение паролей, проверка целостности данных, защита файлов, обнаружение вредоносного ПО.1 Метод атаки на коллизии: «парадокс дней рождения».1

Хеширование обеспечивает криптографический примитив для обеспечения целостности и подлинности данных без необходимости шифрования. Это крайне важно для проверки того, что данные не были подделаны, даже если их содержимое не является секретным. Хеш-функции производят выход фиксированной длины из входа переменной длины и являются односторонними.1 Свойство коллизионной стойкости 1 означает, что вычислительно нецелесообразно найти два разных входа, которые дают один и тот же выход, или вход для данного выхода. Если отправитель вычисляет хеш сообщения и отправляет как сообщение, так и хеш, получатель может пересчитать хеш. Если хеши совпадают, то весьма вероятно, что сообщение не было изменено. Это обеспечивает надежную гарантию целостности и подлинности данных, даже если само сообщение не зашифровано. Это отдельная служба безопасности, отличная от конфиденциальности.

4. Стеганография

Стеганография скрывает само *существование* секретного сообщения внутри другой, казалось бы, безобидной части информации (контейнера). Она отличается от криптографии тем, что в стеганографии секретом является *факт* связи, тогда как в криптографии секретом является *содержание* связи.1

Стеганография решает иную модель угроз, чем криптография: избегание обнаружения самой связи. Это критически важно в сценариях, где сам факт отправки зашифрованного сообщения может вызвать подозрение. В определенных средах даже зашифрованные сообщения могут быть помечены и перехвачены, если их существование известно. Встраивая секретные данные в, казалось бы, невинные файлы (изображения, аудио, текст) 1, стеганография стремится обойти наблюдение, которое ищет необычный или зашифрованный трафик. Это обеспечивает уровень скрытности, который криптография сама по себе не может предложить, что делает ее ценной в контекстах, где отрицание факта связи является первостепенным.

5. Нейрокриптография

Нейрокриптография использует искусственные нейронные сети (ИНС) для криптографических целей, в частности для согласования ключей. Она основана на модели машины четности дерева (TPM), где две ИНС взаимно обучаются и синхронизируют свои векторы весов для установления общего секретного ключа.1

Нейрокриптография представляет собой развивающийся, нетрадиционный подход к криптографии, исследующий, как принципы искусственного интеллекта могут быть использованы для безопасной связи, потенциально предлагая новые решения сложных проблем или устойчивость к новым атакам. Это может предложить альтернативные примитивы безопасности, которые не связаны напрямую с классическими проблемами теории чисел, потенциально обеспечивая устойчивость к новым типам атак (например, атакам квантовых компьютеров, которые угрожают текущим асимметричным схемам). Это подчеркивает продолжающуюся эволюцию и диверсификацию криптографических исследований, направленных на поиск новых вычислительных задач для построения безопасности.

## 41. Подстановочные шифры. Шифр Цезаря. Криптостойкость.

1. Сущность и типы подстановочных шифров

Подстановочные шифры включают замену символов в открытом тексте символами в зашифрованном тексте. Могут использоваться один и тот же или разные алфавиты. Секретом является сам алгоритм подстановки.1

Типы: моноалфавитные (простая подстановка), полиграфические, омофонические, полиалфавитные.1

Классификация подстановочных шифров отражает историческое развитие от простых, легко взламываемых методов (моноалфавитных) к более сложным, многоуровневым подходам (полиалфавитным, омофоническим), разработанным для противодействия криптоаналитическим атакам. Шифры классифицируются по способу подстановки (моно-, полиалфавитные и т.д.).1 Простые моноалфавитные шифры (например, Цезаря) были ранними формами. Эти простые шифры были уязвимы для частотного анализа.1 В ответ на это были разработаны более сложные типы, такие как полиалфавитные и омофонические шифры, специально для маскировки частот букв. Эта эволюция демонстрирует непрерывную «гонку вооружений» между криптографами (разрабатывающими шифры) и криптоаналитиками (взламывающими их), где сложность шифра напрямую отвечает на изощренность атак.

2. Шифр Цезаря: принципы, математическое описание и примеры

Шифр Цезаря — это классический пример моноалфавитного подстановочного шифра. Каждый символ заменяется символом, смещенным на фиксированное число позиций (ключ k) в алфавите.1

* **Шифрование:** y ≡ x + k mod N, где x, y — индексы символов, N — размер алфавита.1
* **Расшифрование:** x ≡ y – k mod N.1

Примеры: «cba» -> «fed» (k=3, N=26). «VENI VIDI VICI» -> «YHQL YLGL YLFL».1 Модификации: Атбаш (обращает алфавит), лозунговый шифр (использует ключевое слово для переупорядочивания алфавита).1

В таблице 7 представлен пример подстановки в шифре Цезаря.

**Таблица 7. Пример подстановки в шифре Цезаря (k=3, N=26)**

| Открытый алфавит | 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 |

| :--------------- | :- | :- | :- | :- | :- | :- | :- | :- | :- | :- | :- | :- | :- | :- | :- | :- | :- | :- | :- | :- | :- | :- | :- | :- | :- | :- |

| Шифр-алфавит | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z | A | B | C |

Пример шифрования: "HELLO" -> "KHOOR".

Пример расшифрования: "KHOOR" -> "HELLO".

Таблица наглядно иллюстрирует простую, однозначную подстановку, характерную для моноалфавитных шифров. Она четко показывает прямое применение правила x + k mod N в читаемом формате и служит наиболее базовым примером подстановки, обеспечивая четкую отправную точку для понимания более сложных шифров.

3. Криптостойкость и уязвимости (с акцентом на частотный анализ)

Подстановочные шифры, включая шифр Цезаря, обладают низкой криптостойкостью из-за уязвимости к частотному анализу.1 Если буква часто встречается в открытом тексте, соответствующий ей символ также будет часто встречаться в зашифрованном тексте.1

Исторический контекст: Шихаб ал-Калкашанди (1412 г.) использовал частоты букв в арабском языке. Существуют таблицы частот для различных языков (например, частоты русских букв из «Национального корпуса русского языка»).1 Другие атаки включают анализ биграмм (пар букв), триграмм и исключение невозможных буквенных комбинаций (например, «чя» в русском языке).1

В таблице 8 представлены частоты появления букв русского языка.

**Таблица 8. Частота появления букв русского языка в текстах**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **№ п/п** | **Буква** | **Частота, %** | **№ п/п** | **Буква** | **Частота, %** |
| 1 | О | 10,97 | 18 | Ь | 1,74 |
| 2 | Е | 8,45 | 19 | Г | 1,70 |
| 3 | А | 8,01 | 20 | З | 1,65 |
| 4 | И | 7,35 | 21 | Б | 1,59 |
| 5 | Н | 6,70 | 22 | Ч | 1,44 |
| 6 | Т | 6,26 | 23 | Й | 1,21 |
| 7 | С | 5,47 | 24 | Х | 0,97 |
| 8 | Р | 4,73 | 25 | Ж | 0,94 |
| 9 | В | 4,54 | 26 | Ш | 0,73 |
| 10 | Л | 4,40 | 27 | Ю | 0,64 |
| 11 | К | 3,49 | 28 | Ц | 0,48 |
| 12 | М | 3,21 | 29 | Щ | 0,36 |
| 13 | Д | 2,98 | 30 | Э | 0,32 |
| 14 | П | 2,81 | 31 | Ф | 0,26 |
| 15 | У | 2,62 | 32 | Ъ | 0,04 |
| 16 | Я | 2,01 | 33 | Ё | 0,04 |
| 17 | Ы | 1,90 |  |  |  |

Таблица наглядно демонстрирует статистический паттерн, который криптоаналитики используют для взлома моноалфавитных шифров. Она предоставляет конкретные данные (проценты), лежащие в основе атаки частотным анализом. Это объясняет, почему шифр Цезаря (и его варианты) слабы: они сохраняют базовое частотное распределение открытого текста, делая зашифрованный текст уязвимым для статистических атак.

## 42. Шифр на основе аффинных преобразований. Криптостойкость.

1. Математическое описание и процедура шифрования

Аффинный шифр является моноалфавитным подстановочным шифром. Процедура шифрования определяется формулой: y ≡ ax + b mod N.1

Переменные: x, y — индексы символов (начиная с 0), a, b — целочисленные ключи, N — размер алфавита.1

Аффинный шифр вводит мультипликативный компонент (a) в дополнение к аддитивному сдвигу (b) шифра Цезаря. Хотя это увеличивает пространство ключей (больше возможных пар a и b) и делает полный перебор немного сложнее, чем для простого шифра Цезаря, он фундаментально остается моноалфавитной подстановкой. Аффинный шифр использует ax + b mod N 1, тогда как шифр Цезаря использует x + k mod N.1 Аффинный шифр имеет два ключа (a и b) по сравнению с одним ключом Цезаря (k). Количество допустимых значений a равно φ(N) (функция Эйлера1), а для b — N значений. Это, как правило, значительно больше, чем N для Цезаря. Несмотря на увеличенное пространство ключей, каждая буква открытого текста x всегда отображается на *одну и ту же* букву зашифрованного текста y для данной пары ключей (a, b). Это означает, что он по-прежнему является моноалфавитным шифром, и его безопасность основывается на том же фундаментальном принципе, что и у Цезаря: отображение фиксировано.

2. Условия однозначного соответствия (взаимная простота)

Для однозначного соответствия (обеспечивающего уникальное расшифрование) должны выполняться следующие условия: 0 ≤ a, b < N, и НОД(a, N) = 1 (т.е. a и N должны быть взаимно простыми).1

Условие взаимной простоты не является произвольным; это прямое математическое требование для существования модульного мультипликативного обратного элемента a⁻¹, который необходим для расшифрования. Без этого шифрование было бы необратимым. Условие НОД(a, N) = 1 является необходимым для однозначного отображения.1 Модульный обратный элемент a^-1 mod N существует тогда и только тогда, когда НОД(a, N) = 1.1 Операция расшифрования включает умножение на a^-1.1 Если a и N не взаимно просты, a^-1 не существует, и операция расшифрования не может быть выполнена, что делает шифрование необратимым. Это подчеркивает, как фундаментальные понятия теории чисел (НОД, обратный элемент по модулю) напрямую определяют осуществимость и правильность криптографических алгоритмов.

3. Процедура расшифрования с использованием обратного элемента по модулю

Расшифрование использует формулу: x ≡ a⁻¹ (y + N – b) mod N. a⁻¹ — это обратный элемент для a по модулю N, удовлетворяющий a \* a⁻¹ ≡ 1 mod N.1

Пример: N=26, a=3, b=5. Для 'Q' (y=16), x ≡ 3⁻¹ (16 + 26 – 5) mod 26. Сначала находим 3⁻¹ mod 26. 3\*9 = 27 ≡ 1 mod 26, поэтому 3⁻¹ = 9. Затем x ≡ 9 \* (16 + 21) mod 26 = 9 \* 37 mod 26 = 9 \* 11 mod 26 = 99 mod 26 = 21. x=21 соответствует 'V'.1

В таблице 9 представлен пример подстановки в аффинном шифре.

**Таблица 9. Пример подстановки в аффинном шифре (N=26, a=3, b=5)**

| Открытый текст | V | E | N | I | V | I | D | I | V | I | C | I |

| :------------- | :- | :- | :- | :- | :- | :- | :- | :- | :- | :- | :- | :- |

| Индекс (x) | 21 | 4 | 13 | 8 | 21 | 8 | 3 | 8 | 21 | 8 | 2 | 8 |

| 3x+5 mod 26 | 16 | 17 | 18 | 3 | 16 | 3 | 14 | 3 | 16 | 3 | 11 | 3 |

| Шифр-текст | Q | R | S | C | Q | C | O | C | Q | C | L | C |

Таблица четко демонстрирует применение аффинного преобразования для каждого символа. Она визуализирует, как числовые индексы отображаются между открытым и зашифрованным текстом, что крайне важно для понимания базовой математической операции. Показывая последовательное отображение, она неявно подготавливает почву для понимания того, как обратная операция восстановит оригинал.

4. Криптостойкость и уязвимости (унаследованные от моноалфавитных шифров)

Аффинные шифры, как моноалфавитные шифры, обладают низкой криптостойкостью. Они уязвимы для частотного анализа.1 Фиксированное однозначное отображение между символами открытого и зашифрованного текста сохраняет частотное распределение символов, делая их уязвимыми для статистических атак.1

Несмотря на возросшую математическую сложность по сравнению с базовым шифром Цезаря, аффинный шифр наследует фундаментальную слабость всех моноалфавитных шифров: он не скрывает статистические свойства (например, частоты букв) открытого текста. Это делает его легко взламываемым с помощью частотного анализа, независимо от размера ключа. Аффинные шифры классифицируются как моноалфавитные.1 Каждая буква открытого текста всегда отображается на *одну и ту же* букву зашифрованного текста для данного ключа.1 Это фиксированное отображение означает, что если «E» является наиболее частой буквой в английском открытом тексте, то соответствующая ей буква зашифрованного текста (например, «X») будет наиболее частой буквой в зашифрованном тексте. Злоумышленник может проанализировать частоту букв в зашифрованном тексте и сопоставить их с известными частотами языка открытого текста.1 Таким образом, аффинный шифр, несмотря на более сложную математическую формулу и большее пространство ключей по сравнению с Цезарем, не предлагает реального улучшения криптостойкости против целеустремленного криптоаналитика, использующего частотный анализ. Он остается «игрушечным шифром» по современным стандартам безопасности.