Основы теории информации и кодирования

21 сентября 2022 г.

Требования к выполнению лабораторных работ

В данном разделе описаны базовые требования к лабораторным и курсовым работам, а также основные правила их оценивания. **Все особые случаи обсуждаются с преподавателем индивидуально,** решение в каждом случае принимается отдельно и только **после воплощения соответствующего случая в реальность.**

Язык, компилятор, IDE

Не регламентируются. Но если не можете выбрать — C++, коллекция компиляторов GCC, Qt Creator.

Отчёт и оформление

При выполнении лабораторной работы непосредственно перед защитой отчёт оформлять не обязательно; достаточно демонстрации результатов и устных комментариев.

В ином случае по результатам выполнения лабораторной работы оформляется отчёт в формате plain text, LaTeX, OpenDocument или PDF, а также программный код. Заголовок отчёта должен включать имя группы и ФИО авторов, а также тему работы. Отчёт должен содержать для каждого задания:

- номер и текст задания;
- номер и текст варианта (если есть);
- результат выполнения задания: результаты измерений с комментариями и ссылки на программы.

Если отчёт оформлен не как комментарии к коду, полный текст программ копировать в отчёт не нужно (текст программ предоставляется отдельно).

Отчёт в формате plain text может быть **совмещён с программным кодом (помещён в комментарии соответствующих модулей).**

Командная работа

Лабораторная работа выполняется совместно командой (двумя сидящими рядом студентами, при желании — одним или тремя). Команда из трёх человек выполняет дополнительные задания в некоторых лабораторных работах. Команды из четырёх и более студентов не допускаются.

Команда, независимо от количества участников, выполняет **один вариант задания, соответствующий номеру команды** и оформляет один отчёт.

Номер пары или команды в группе должен быть уникален. В спорных случаях номер пары или команды может быть назначен преподавателем.

Каждый из соавторов должен уметь объяснить все результаты лабораторной работы (программный код, результаты измерений) и модифицировать свою часть кода.

Оценивание

Максимально возможная оценка лабораторной работы — 10 баллов. Работа, выполненная не полностью или с ошибками, может быть зачтена с оценкой ниже максимальной.

Работа с явными признаками несамостоятельного выполнения (несоответствие варианту, дословное совпадение отчёта, неумение его пояснить и т. п.) **может быть зачтена только после выполнения дополнительного задания** (при попытке сдать одновременно серию таких работ может быть дано одно задание объединённой тематики) **и не более чем на 6 баллов** (если с учётом опоздания максимум составляет менее 6 баллов — не более максимума). **Бонусные задания такой работы не засчитываются.** Таким образом, фактически в этом случае оценивается только дополнительное задание.

Использование библиотек со свободной лицензией, если автор программы отделяет свой код от библиотечного и может внести изменения в свою часть — допустимо и не ведёт к снижению оценки.

Из оценки лабораторной работы, сданной с опозданием более чем на одно занятие без уважительной причины, вычитается величина опоздания (таблица ЛР.1).

Если лабораторные работы выполняются не по порядку, но при этом на каждом занятии (кроме, возможно, первого) выполняется и сдаётся какая-либо работа, то они оцениваются как сданные без опоздания.

Не зачтённые лабораторные работы помечаются в ОРИОКС литерой «н». Если курс в целом зачтён, перед закрытием ведомости «н» заменяется для корректной работы ОРИОКС значением «0,1».

Необязательные (бонусные) задания

Задания, отмеченные как **«Бонус»**, *необязательны;* за их выполнение начисляются дополнительные баллы. Баллы за необязательные задания добавляются к колонке «бонус (л/р)».

Бонусные задания могут быть сданы как одновременно с соответствующей лабораторной работой так и после неё.

В тексте задания указано *максимальное* количество дополнительных баллов. Количество баллов, начисленных за бонусные задания, зависит только от качества выполнения задания, но не от времени сдачи.

График снижения оценок за базовую часть лабораторных работ

Таблица ЛР.1

Неделя	max(**Л1**)	max(**Л2**)	max(**Л3**)	max(**Л4**)	max(**Л5**)	max(**Л6**)	max(**Л7**)
1, 2	10						
3, 4	10	10					
5, 6	9	10	10				
7, 8	8	9	10	10			
9, 10	7	8	9	10	10		
11, 12	6	7	8	9	10	10	
13, 14	5	6	7	8	9	10	10
15, 16	4	5	6	7	8	9	10
17, 18	3	4	5	6	7	8	9
сессия (досд.)	2	3	4	5	6	6	6
сессия (с нуля)	н	н	50 в совокупности				н

Бонусные задания работы с явными признаками несамостоятельного выполнения не засчитываются (см. выше).

КМ «Натяжка»

Баллы, не соответствующие никаким реальным достижениям студента, но необходимые для корректной работы ОРИОКС, выставляются в графу «Натяжка». Если студент с ненулевым КМ «Натяжка» что-либо досдаёт — КМ «Натяжка» обнуляется (при необходимости — пересчитывается после обнуления).

Курсовая работа

Вместо работ Л1-Л7 может быть выполнена одна курсовая работа (оценивается от 0 до 100 баллов); тематика должна быть согласована с преподавателем.

Зачёт

Дифзачёт (зачёт с оценкой) выставляется по сумме баллов: отлично (5) от 86, хорошо (4) от 70, удовлетворительно (3) от 50 баллов.

Замечания и дополнения

Замечания и дополнения к данному документу можно отправить в письменном виде по адресу `illinc@inbox.ru`.

Принятое замечание/дополнение приносит первому приславшему его студенту от 1 до 8 бонусных баллов.

Лабораторная работа 1

Двоичное представление информации. Исследование форматов файлов. Проектирование формата файлов

Цель работы: 1) изучить двоичное и шестнадцатеричное представление информации, научиться работать с шестнадцатеричным редактором; 2) изучить структуру простейших форматов файлов; 3) научиться создавать и обрабатывать двоичные файлы на ЯВУ.

Штраф за одно пропущенное обязательное задание −1 балла.

Л1.1. Задание на лабораторную работу

Задание Л1.№1. С помощью hexdump/xxd или шестнадцатеричного просмотровщика/редактора исследуйте файлы различных форматов (некоторые файлы представлены в папке «labs-files/files»). Выделите сигнатуры или иные признаки формата там, где это возможно.

Задание Л1.№2. В соответствии с номером варианта отредактируйте изображение colorchess16x16x2.bmp, используя шестнадцатеричный редактор. Откройте изменённый файл и убедитесь, что изменения корректны. Файл colorchess16x16x2.bmp — изображение 16×16 пикселей, сиренево-болотное (две сиреневые и две болотные клетки по 8×8 пикселей; некоторые просмотровщики неадекватно отображают неполноцветные bmp), глубина цвета — 1 бит на пиксель.

$(№ - 1)\%2 + 1$	**Вариант**
1	Поставить зелёную точку в правом нижнем углу изображения
2	Поставить сиреневую точку в правом верхнем углу изображения

Задание Л1.№3. Выберите сигнатуру для собственного формата файлов, которая будет использоваться во всех дальнейших работах (4-8 байт).

Задание Л1.№4. Разработайте формат файла-архива для дальнейших работ. Архив обязательно содержит заголовок и закодированные данные.

Заголовок обязательно включает:

- сигнатуру (в дальнейшем все архивы команды должны использовать ту же сигнатуру);
- версию формата;

– коды использованных алгоритмов сжатия и защиты от помех (с учётом того, что сжатие без учёта контекста часто применяется поверх сжатия с его учётом);
– исходную длину файла в символах кодирования (то есть байтах);

а также любые необходимые, по мнению автора, поля.

Формат должен предусматривать возможность добавления служебных данных для различных алгоритмов сжатия и защиты от помех (например, массив частот для методов сжатия без учёта контекста) без кардинальной его переработки.

Задание Л1.№5. Разработайте программу-кодек, состоящую из двух частей — *кодера* и *декодера*.

1. *Кодер* по заданному файлу X создаёт архив Y формата, разработанного в задании Л1.№4, состоящий из заголовка и несжатого текста X (то есть содержимого X без каких-либо преобразований, целиком).
   В качестве кодов алгоритмов, соответствующих отсутствию сжатия и защиты от помех, используйте значение 0.
2. *Декодер* по заданному архиву Y:
   – проверяет сигнатуру на соответствие выбранной в Л1.№3;
   – при корректной сигнатуре проверяет коды алгоритмов;
   – если коды соответствуют 0, восстанавливает файл $\widetilde{X}$ (который должен совпадать с исходным X) по данным архива Y.

Кодер и декодер могут быть реализованы как в виде двух раздельных модулей, так в одной программе (в последнем случае действие задаётся ключом командной строки или выбором меню — то есть должна быть возможность раздельного вызова кодера и декодера, а не только слитного преобразования $X \to Y \to \widetilde{X}$).

Л1.2. Дополнительные бонусные и штрафные баллы

−1 балл, если структура архива, создаваемого кодеком в задании Л1.№5, не совпадает с описанной в задании Л1.№4.

−1 балл, если декодер при некорректной сигнатуре архива Y всё равно создаёт файл $\widetilde{X}$.

−1 балл, если символом кодирования является не байт, а печатаемый символ текста (всё равно, utf8 или любой однобайтовой кодировки).

+3 балла, если в структуре заголовка и программе предусмотрена возможность собрать в один архив несколько файлов (но не папки) и восстановить их с прежними именами.

+3 балла, если в структуре заголовка и программе предусмотрена возможность собрать в один архив и восстановить ещё и иерархическую структуру папок.

Лабораторная работа 2

Источник без памяти. Исследование статистических характеристик исходных текстов (как бинарных файлов, так и файлов в формате простого текста). Работа с кодовыми таблицами русского языка

Штраф за одно пропущенное обязательное задание −2 балла.

Л2.1. Задание на лабораторную работу

Задание Л2.№1. Разработайте программу, которая по заданному файлу X рассчитывает:

- длину n файла X в символах первичного алфавита A_1;
- ν_i — общее количество вхождений каждого из символов $a_i \in A_1$ в X (целочисленную частоту a_i);

и оценивает, считая файл X порождённым источником без памяти:

- вероятность p_i каждого из символов $a_i \in A_1$;
- количество информации $I(a_i)$ в каждом символе $a_i \in A_1$;
- суммарное количество информации $I(X)$ в файле X (не среднее на символ, которое в n раз меньше, а именно суммарное!) в битах и байтах;

символом кодирования, как всегда, является *байт* (с учётом использования архитектуры x86 это 8 бит, октет), первичным алфавитом A_1 — множество возможных значений байта $(0 \ldots 255)$.

Полученные величины необходимо вывести на экран или сохранить в виде отчёта; причём таблица характеристик символов алфавита $a_i \in A_1$ должна выводиться дважды (либо в программе необходимо предусмотреть пересортировку) — отсортированной по алфавиту (по значению a_i) и отсортированной по убыванию частоты ν_i.

Проверьте разработанную программу на файлах различного формата (не только простом тексте; в том числе и на бинарных).

Задание Л2.№2. Разработайте программу, аналогичную Л2.№1, но считающую символом кодирования *печатный символ Unicode,* а первичным алфавитом A_1 — множество символов Unicode (строчных букв, заглавных букв, цифр, различных пробельных символов, знаков препинания и т. п) в файле X.

Рассчитайте две оценки количества информации в длинном текстовом файле в кодировке UTF-8 программами Л2.№1 и Л2.№2. Сравните результаты.

Задание Л2.№3. Рассчитайте частоты появления октетов в файлах, являющихся простым текстом в различных кодировках (см. «labs-files/files/plaintext»). Определите 4 наиболее частых октета среди всех ис-

пользуемых и 4 наиболее частых октета, не являющихся кодами печатных символов ASCII. Обратите внимание на распределение октетов многобайтовых кодировок.

Рассчитайте частоты появления октетов в файле, соответствующем варианту $N \bmod 9$ в папке «labs-files/variants/L2» (Z). Определите, является ли Z простым русскоязычным текстом в одной из стандартных кодировок (один из вариантов представляет собой нерусскоязычный текст); если да — определите кодировку.

Лабораторная работа 3

Источник без памяти. Алфавитное префиксное кодирование

Штраф за одно пропущенное обязательное задание −3 балла.

Л3.1. Задание на лабораторную работу

Задание Л3.№1. Разработайте программу-кодек (аналогично Л1.№5), реализующую сжатие/разжатие файла алгоритмом, соответствующим варианту (таблица Л3.1).

Варианты алгоритмов сжатия файла без учёта контекста (модель без памяти)

Таблица Л3.1

(№ − 1)%3 +1	Вариант
1	алгоритм Шеннона
2	алгоритм Шеннона—Фано
3	алгоритм Хаффмана

Сигнатура формата должна совпадать с выбранной в Л1.№3. Если в формат необходимо внести какие-либо изменения по сравнению с Л1.№4 — они должны быть описаны в отчёте, а номер версии — изменён.

Также в отчёте обязательно должны быть описаны:

- код алгоритма, не равный 0;
- состав информации для декодирования и формат её хранения в файле;
- принятые при построении дерева Шеннона—Фано или Хаффмана уточнения (порядок сортировки при совпадении частот, стратегия разбиения при невозможном равенстве веса частей и т. п.).

Сопоставьте длину сжатых данных с оценкой количества информации в исходном файле (Л2.№1).

Задание Л3.№2. Разработайте универсальный декодер разработанного формата, который:

- проверяет сигнатуру на соответствие выбранной в Л1.№3;
- при корректной сигнатуре проверяет номер версии и коды алгоритмов, после чего вызывает соответствующий им декодер из заданий Л1.№5

или Л3.№1 (в дальнейшем необходимо будет дополнять анализ при любом изменении формата или добавлении алгоритма).

Задание Л3.№3. Бонус +3 балла.

Разработайте «интеллектуальный» кодер, который анализирует суммарный объём n_{compr} сжатых данных и информации для декодирования Л3.№1 и, если $n_{compr} \geqslant n$, записывает в полученный архив несжатый текст исходного файла (то есть использует вместо кодера Л3.№1 кодер Л1.№5; код алгоритма при этом также должен быть 0).

Л3.2. Дополнительные бонусные и штрафные баллы

+1 балл, если при работе с несколькими файлами/папками каждый из файлов имеет собственный код алгоритма и информацию для декодирования (а в Л3.№3 — анализируется отдельно).

+4 балла, если архив позволяет как включить для каждого файла собственный код алгоритма и информацию для декодирования, так и рассмотреть совокупность файлов как единый исходный текст (но при декодировании восстановить исходную структуру файлов).

Лабораторная работа 4

Источник Маркова. Исследование статистических характеристик исходных текстов (как бинарных файлов, так и файлов в формате простого текста). Простейшие алгоритмы сжатия данных с учётом контекста

Штраф за одно пропущенное обязательное задание −2 балла.

Л4.1. Задание на лабораторную работу

Задание Л4.№1. Разработайте программу, которая по заданному файлу X рассчитывает:

- длину n файла X в символах первичного алфавита A_1;
- ν_{ij} — количество вхождений подстрок $a_j a_i$;
- ν_{*j} — общее количество вхождений любых двухсимвольных подстрок, начинающихся с a_j (для всех символов, кроме последнего символа файла, $\nu_{*j} = \nu_j$);

и оценивает, считая файл X порождённым источником Маркова первого порядка:

- безусловную вероятность $p(a_i)$ каждого из символов $a_i \in A_1$;
- условную вероятность $p(a_i|a_j)$ каждой пары символов $a_i, a_j \in A_1$;
- оценивает суммарное количество информации $I(X)$ в файле X в битах и байтах;

символом кодирования, как всегда, является *байт* (с учётом использования архитектуры x86 — октет), первичным алфавитом A_1 — множество возможных значений байта $(0 \ldots 255)$.

Полученные величины необходимо вывести на экран или сохранить в виде отчёта.

Проверьте разработанную программу на файлах различного формата (не только простом тексте; в том числе и на бинарных). Сопоставьте результат с Л2.№1.

Задание Л4.№2. Бонус +2 балла. Разработайте программу, аналогичную Л4.№1, но считающую X порождённым источником Маркова порядка N и, соответственно, рассматривающую $(N+1)$-символьные подстроки и условные вероятности $p(a_i|a_{j_1} a_{j_2} \ldots a_{j_N})$.

Для упрощения расчёта полагайте $n \gg N$, то есть вероятности первых N символов файла рассчитывайте без учёта контекста.

Проверьте разработанную программу на файлах различного формата и при различных $N \in [1, 4]$. Сопоставьте результат при $N = 1$ с Л4.№1.

Примечание: если реализован общий случай, но нет отдельной программы для $N = 1$ — обязательное задание Л4.№1 не считается пропущенным.

Задание Л4.№3. Разработайте программу-кодек, реализующую сжатие/разжатие файла алгоритмом RLE согласно варианту (таблица Л4.1).

Варианты алгоритмов семейства RLE

Таблица Л4.1

(№ − 1)%2 +1	**Вариант**
1	RLE с флаг-битами (цепочки несжатых символов и сжатые цепочки)
2	RLE с односимвольным префиксом (сжатые цепочки отделяются от несжатого теста символом-префиксом p). Префикс p выбирать для каждого исходного текста X отдельно, исходя из частот символов в X.

Упрощённое задание, −1 балл (может быть выбрано студентом вместо любого варианта): «наивный» алгоритм RLE (даже однократно повторённый ($L = 1$) символ c записывается парой (L, c)).

Сопоставьте длину сжатых данных с оценкой количества информации в исходном файле по марковской модели первого порядка (Л4.№1).

Задание Л4.№4. Доработайте универсальный декодер из задания Л3.№2 и кодер из задания Л3.№3 с учётом реализованных алгоритмов.

Л4.2. Дополнительные бонусные и штрафные баллы

+3 балла, если реализованные программы позволяют применить к файлу последовательно сжатие с учётом контекста, а затем сжатие без учёта контекста; но при этом сжатию без учёта контекста не подвергается, например, добавленная на этапе сжатия с контекстом сигнатура (то есть сжатие двумя алгоритмами не сводится к двойному запуску кодера).

Лабораторная работа 5

Источник Маркова. Алгоритмы сжатия данных с учётом контекста

Штраф за одно пропущенное обязательное задание -2 балла.

Л5.1. Задание на лабораторную работу

Задание Л5.№1. Разработайте программу-кодек, реализующую сжатие/разжатие файла алгоритмом семейства LZ77 согласно варианту (таблица Л5.1).

Запись $(S:10,\ L:6)$ обозначает: сначала записывается 10 бит S, затем 6 бит L. Для вариантов с флаг-битом выходной поток *битовый.* Для концепта Зива и Лемпеля символ c записывается как $(c:8)$ (то есть просто как байт c); для реализации с односимвольным префиксом p символ $c \neq p$ также записывается как $(c:8)$. Если ссылка отделяется от несжатого текста односимвольным префиксом p, значение p выбирать для каждого исходного текста X отдельно, исходя из частот символов в X.

Сопоставьте длину сжатых данных с оценками количества информации в исходном файле по марковским моделям порядка от 1 до 4 (Л4.№1, Л4.№2).

Задание Л5.№2. Доработайте универсальный декодер из задания Л3.№2 и кодер из задания Л3.№3 с учётом реализованных алгоритмов.

Варианты алгоритмов семейства LZ77 с односимвольным префиксом p

Таблица Л5.1

(№ − 1)%2 +1	**Вариант**
1	концепт Зива и Лемпеля; ссылка (S, L) как $(S:10,\ L:6)$
2	концепт Зива и Лемпеля; ссылка (S, L) как $(L:6,\ S:10)$
3	флаг-биты; ссылка (S, L) как $\Big(1:1,\ (S-1):9,\ (L-1):8\Big)$, символ c — как $(0:1,\ c:8)$
4	флаг-биты; ссылка (S, L) как $\Big(1:1,\ (S-1):10,\ (L-2):7\Big)$, символ c — как $(0:1,\ c:8)$
5	флаг-биты группируются во флаг-байты; ссылка (S, L) записывается как $\Big(S:10,\ (L-3):6\Big)$ — LZJB
6	флаг-биты группируются во флаг-байты; ссылка (S, L) записывается как $\Big((L-3):6,\ S:10\Big)$
7	флаг-биты группируются во флаг-байты; ссылка (S, L) записывается как $\Big((L-3):6,\ (S-1):10\Big)$
8	префикс p; ссылка (S, L) как $\Big(p:8,\ S:10,\ (L-4):6\Big)$, символ $c = p$ в тексте — как $\Big(p:8,\ 0:8,\ 0:8\Big)$
9	префикс p; ссылка (S, L) как $\Big(p:8,\ (L-3):6,\ (S-1):10\Big)$, символ $c = p$ в тексте — как $\Big(p:8,\ 0:8\Big)$

Л5.2. Дополнительные бонусные и штрафные баллы

+3 балла, если реализованные программы позволяют применить к файлу последовательно любой алгоритм сжатия с учётом контекста, а затем сжатие без учёта контекста (аналогично Л4).

Лабораторная работа 6

Помехи. Помехозащитное кодирование.

Штраф за одно пропущенное обязательное задание -3 балла.

Л6.1. Задание на лабораторную работу

Задание Л6.№1. Разработайте программу-кодек, реализующую защиту данных от помех методом Хэмминга, позволяющую исправить одиночную инверсию в блоке и обнаружить двойную инверсию в блоке.

Размер блока до кодирования соответствует варианту (таблица Л6.1). Код должен быть систематическим, количество контрольных символов — целым (если в контрольном символе при заданном размере блока остаются неиспользуемые биты — продублируйте бит общей чётности).

Варианты размера блока до кодирования

Таблица Л6.1

$(№ - 1)\%2 + 1$	**Вариант**
1	$N = 7$ байт
2	$N = 8$ байт

Упрощённый вариант, -1 балл: $N = 3$ байта.

Внесите в закодированные данные ошибки; убедитесь, что при декодировании действительно исправляется одиночная инверсия в блоке и обнаруживается двойная.

Задание Л6.№2. Разработайте программу-кодек, реализующую защиту данных от помех методом Рида—Соломона (используйте библиотеки, распространяемые по свободным лицензиям). Какова выбранная вами длина блока до кодирования (сколько информационных символов)? Сколько контрольных символов добавляется? Какое максимальное количество ошибок в блоке может быть исправлено?

Внесите в закодированные данные ошибки; убедитесь, что при декодировании действительно исправляется указанное выше количество ошибок.

Задание Л6.№3. К ключевым полям заголовка файла (сигнатура, версия, алгоритмы и т. п.) нельзя применить какой-либо помехозащитный код без нарушения читаемости заголовка, поэтому они должны обрабатываться

отдельно (например, троироваться, или дублироваться с применением к дубликату защиты от помех, или добавленные контрольные символы заголовка должны быть помещены вне заголовка).

Добавьте новую версию формата, где заголовок защищён от помех даже в том случае, когда к данным защита не применяется.

Доработайте универсальный декодер из задания Л3.№2 и кодер из задания Л3.№3 с учётом реализованных алгоритмов.

Л6.2. Дополнительные бонусные и штрафные баллы

−2 балла, если размер блока Хэмминга — весь файл.

−2 балла, если код несистематический.

−2 балла, если в блоке после кодирования есть неиспользуемый бит, а двойная ошибка в блоке не распознаётся.

+3 балла, если реализованные программы позволяют применить к файлу последовательно любое сжатие, а затем защиту от помех (аналогично Л4).

Лабораторная работа 7

Источник без памяти. Арифметическое кодирование

Л7.1. Задание на лабораторную работу

Задание Л7.№1. Разработайте программу-кодек, реализующую сжатие/разжатие файла целочисленным арифметическим (интервальным) алгоритмом.

Задание Л7.№2. Доработайте универсальный декодер из задания Л3.№2 и кодер из задания Л3.№3 с учётом реализованного алгоритма.