Министерство образования и науки Российской Федерации
федеральное государственное автономное образовательное учреждение
высшего образования «Национальный исследовательский университет
«Московский институт электронной техники»

Основы теории информации и кодирования

Студенты, изучающие по индивидуальному плану ОТИК раньше,
чем ОргЭВМ (что противоречит рабочей программе ОТИК)
не могут быть допущены к выполнению лабораторных работ по ОТИК
до демонстрации Л0 (коды, используемые в ЭВМ по умолчанию)

Кононова А. И.

Актуальную версию можно найти на `https://gitlab.com/illinc/otik`

2023

Приложение А. Регламент курса

В данном разделе описан регламент курса в целом. **Все особые случаи обсуждаются с преподавателем индивидуально,** решение в каждом случае принимается отдельно и только **после воплощения соответствующего случая в реальность.**

Допуск для индивидуальщиков и должников

Для допуска к курсу ОТИК (специфические коды для сжатия и защиты от помех) необходимо **подтвердить знакомство с кодами, используемыми в ЭВМ по умолчанию.**

Студенты, обучающиеся по плану группы, допущены к изучению ОТИК, явное подтверждение не нужно.

Студентам, обучающимся по индивидуальному плану, а также должникам, перед защитой первой лабораторной работы Л1 необходимо:

- если в ОРИОКС есть положительная оценка по курсу ОргЭВМ [и Асм] — предъявить её преподавателю;
- если её нет — выполнить и защитить вводную лабораторную Л0.

Должникам, выполнившим Л0 в срок, долг по ОргЭВМ [и Асм] закрывается автоматически (если у преподавателя лабораторных работ нет доступа — сообщите лектору, что долг снят).

Итоговая оценка

Оценка формируется ОРИОКС автоматически по общей сумме баллов в момент закрытия ведомости: отлично (5) от 86 баллов, хорошо (4) от 70, удовлетворительно (3) от 50 баллов. При сумме, меньшей 50, в ведомости формируется оценка «неудовлетворительно» (2).

Примечание: в ОРИОКС проверить оценку можно только по полю «Общая сумма баллов»; поля «Текущая сумма баллов» и «Текущая оценка» *некорректны* вплоть до даты экзамена («Текущая оценка» при неизменных баллах может отличаться от оценки, формируемой ОРИОКС в ведомости, как в бóльшую, так и в меньшую сторону).

Если оценка вас не устраивает, баллы можно повысить до конца сессии; в том числе — решив на экзамене (или консультации перед экзаменом) задачу из билета и, возможно, дополнительную задачу по выбору преподавателя. Также можно изменить оценку (в любую сторону), сдав экзамен.

Неудовлетворительная оценка в конце сессии = долг.

Лабораторные работы

Основной путь прохождения курса — выполнение лабораторных работ (регламент лабораторных работ подробно описан в приложении Б). Если к сессии будут выполнены не все лабораторные работы, но общая сумма баллов не менее 50, курс считается успешно изученным.

Курсовая работа — временно не реализуется

Так как мне надоело получать в конце сессии нечто, что не тянет даже на лабораторную работу, но предлагается как «курсовая» — этот путь для курса ОТИК закрыт. Желающие меня переубедить — подходите лично, с планом конкретной работы.

Прочие бонусные и штрафные баллы

Баллы, соответствующие решаемым на лекциях задачам, выставляются в графу «ЗдЛк». Посещаемость поточных лекций принципиально не оценивается.

Бонусные баллы за достижения, не описанные в данном пособии, выставляются в графы:

– «бонус (л/р)», если баллы назначает преподаватель лабораторных работ;
– «бонус (прочее)», если лектор;

штрафные баллы, не описанные в пособии, аналогично уменьшают эти же графы либо базовую оценку лабораторной работы.

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).

Экзамен

К экзамену допускаются все (возможно повышение оценки с 0 до 3).

Если автоматически формируемая по баллам оценка устраивает — на экзамен можно придти только для простановки оценки в зачётку (если оценка в зачётку была выставлена на консультации/чужом экзамене заранее — можно не приходить).

Ответ оценивается не на баллы, а на оценку от 2 до 5. Если полученная на экзамене оценка выше оценки по баллам — колонка «Итог» маркируется значением 1, а необходимое количество баллов добавляется в колонку «бонус (прочее)». Если оценка на экзамене равна оценке по баллам — в ОРИОКС ничего не меняется. Если оценка на экзамене меньше — в зависимости от ситуации возможен как поиск ошибки в ОРИОКС, так и сохранение текущих баллов.

Зачтение статей из Интернета и даже текста лекций ответом на вопрос не считается; оценивается как «неудовлетворительно». Говорите своими словами. Поясняйте сказанное на примерах.

«Чего можно на троечку сдать» на зачётной неделе или в сессию

Выберите любой один из следующих вариантов:

- написать КР1 + КР2 (итоговая оценка «неуд.»/«3 с натяжкой»);
- получить и выполнить индивидуальное задание (оценивается от 0 до 100 баллов);
- сдать экзамен.

Крайне нежелательно сдавать лабораторные как в семестре:
а) сумма без бонусов в любом случае не превысит 60 баллов (таблица ЛРЕ-ГЛ.1);
б) если **хотя бы в одной работе** есть признаки несамостоятельного выполнения — бонусные задания не засчитываются **для всех** $\Longrightarrow$ итоговая оценка «неуд.»/«удовл.»

Долги и пересдачи

Неудовлетворительную оценку можно пересдать (закрыть долг) в течение семестра после официальной даты экзамена.

1. До конца соответствующей сессии — новая оценка попадает в дополнительную ведомость, формально долг не образуется.
2. **До конца следующего семестра** — оформляется **направление** на пересдачу. Закрывать долг желательно до официальной даты пересдачи, указанной в ОРИОКС: найти преподавателя и получить у него задание.

Начиная со второго после экзамена семестра (следующая осень и позже), согласно новым указаниям проректора по учебной работе, направление не может быть оформлено без письменного разрешения ответственного кафедры.

Для закрытия долга необходимо:

– либо (на 3) написать КР1 + КР2;
– либо (на 3/4/5) получить и выполнить индивидуальное задание.

Должник не имеет права сдавать тот же набор лабораторных работ, что и его однокурсники ранее, если только преподаватель не даст ему такое задание.

Примечание: демонстрация должником полного комплекта лабораторных работ чужого авторства будет в лучшем случае оцениваться на 0 баллов, а в худшем — как неуважение к преподавателю и намеренная трата его времени.

Замечания и дополнения

Замечания и дополнения к данному документу можно отправить в письменном виде по адресу `https://gitlab.com/illinc/otik/issues`.

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

Обновление пособия

В течение семестра могут уточняться формулировки заданий и регламента лабораторных работ (если исходные вызывают недопонимание) и появляться новые бонусные задания.

Качественное обновление заданий и регламента лабораторных работ происходит только между семестрами; поэтому выполнять задания по версии пособия сентября 2023 года можно до конца 2023 года. Но лучше всё-таки периодически скачивать актуальный файл `https://gitlab.com/illinc/otik/-/raw/master/otik-labs.pdf?inline=false`.

Приложение Б. Лабораторный практикум по основам теории информации и кодирования

Требования к выполнению лабораторных работ

В данном разделе описаны базовые требования к лабораторным 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 — используйте язык C++, коллекцию компиляторов GCC, среду 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);
- описание структур данных, в том числе поясняющие схемы и рисунки.

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. **Примечание:** если команда может придти защищать лабораторную работу в полном составе — это необходимо сделать. Но если у кого-то нет возможности придти — лучше сдать работу в неполном составе, но вовремя. При этом все участники команды получают одну оценку; защищаться отдельно отсутствующему соавтору не нужно (если его отсутствие не становится систематическим). Неполная тройка всё равно выполняет задания для троек.

Оценивание

Работа, выполненная не полностью или с ошибками, может быть зачтена с оценкой ниже максимальной $\beta_{\max}$ (значения $\beta_{\max}$ для различных групп см. ниже; штраф за одно пропущенное обязательное задание см. в шапке лабораторной работы).

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** $\beta_{\text{п}} \approx \frac{\beta_{\max}}{2}$ баллов (если с учётом опоздания максимум составляет менее $\beta_{\text{п}}$ — 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 студент:

- указал источник и лицензию библиотеки;
- отделяет свой код от библиотечного и может внести изменения в свою часть;

допустимо и не ведёт к снижению оценки.

Примечание: свободная (открытая) лицензия — это, в частности, GNU GPL различных версий, GNU LGPL, BSD и т. д. (список крайне велик). Код, который выложен на форуме или кодообменнике без явного указания лицензии, свободной библиотекой не является (и не факт, что правильно работает).

Если же студент вообще забудет сообщить в начале защиты лабораторной работы, что использован сторонний код — это оценивается как списывание.

Из оценки лабораторной работы, сданной с опозданием более чем на одно занятие без уважительной причины, вычитается величина опоздания

(таблица ЛРЕГЛ.1). Обратите внимание, что на каждую из Л3–Л5 отводится два занятия.

График снижения оценок за базовую часть лабораторных работ (ПИН-3*, ПИН-45)

Таблица ЛРЕГЛ.1

Группы ПИН-3*, ПИН-45					
Неделя	max(Л1)	max(Л2)	max(Л3)/ max($Л3_{упр}$)	max(Л4)/ max($Л4_{упр}$)	max(Л5)/ max($Л5_{упр}$)
1, 2	7				
3, 4	7	7			
5, 6	6	7	12 / 7		
7, 8	5	6	12 / 7		
9, 10	4	5	12 / 7	12 / 7	
11, 12	3	4	11 / 7	12 / 7	
13, 14	2	3	10 / 7	12 / 7	12 / 7
15, 16	2	2	9 / 7	11 / 7	12 / 7
17, 18	2	2	8 / 7	10 / 7	12 / 7
сессия (досд.)	2	2	8 / 7	8 / 7	8 / 7
сессия (с нуля)	н	н	50 в совокупности		

Индивидуальщики и должники						
Неделя	max(Л0)	max(Л1)	max(Л2)	max(Л3)/ max($Л3_{упр}$)	max(Л4)/ max($Л4_{упр}$)	max(Л5)/ max($Л5_{упр}$)
1, 2	7	7				
3, 4	7	7	7			
5, 6	6	6	7	12 / 7		
7, 8	5	5	6	12 / 7		
9, 10	4	4	5	12 / 7	12 / 7	
11, 12	3	3	4	11 / 7	12 / 7	
13, 14	2	2	3	10 / 7	12 / 7	12 / 7
15, 16	2	2	2	9 / 7	11 / 7	12 / 7
17, 18	2	2	2	8 / 7	10 / 7	12 / 7
сессия (досд.)	2	2	2	8 / 7	8 / 7	8 / 7
сессия (с нуля)	н	н	н	50 в совокупности		

В некоторых лабораторных работах доступны упрощённые версии с максимальной оценкой $\beta_{упр} < \beta_{max}$. Бонусные задания упрощённой лабораторной работы могут быть засчитаны так же, как и для полной — если они выполнены самостоятельно.

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» (см. приложение А).

Лабораторная работа 0

Представление данных в ЭВМ (только для тех, кто не изучал ОргЭВМ!)

Версия 2023 г. — задания могут быть засчитаны только в 2023 г.

Студентам, сдавшим ОргЭВМ [и Асм], выполнять Л0 не нужно; баллы за неё начисляться не будут.

Должникам, выполнившим Л0 в срок, долг по ОргЭВМ [и Асм] закрывается автоматически (если у преподавателя лабораторных работ нет доступа — сообщите лектору, что долг снят).

Штраф за одно пропущенное обязательное задание -1 балл.

Л0.1. Задание на лабораторную работу

Задание Л0.№1. Изучите, как интерпретируется одна и та же область памяти, если она рассматривается как знаковое или беззнаковое целое число, а также — как одно и то же число записывается в различных системах счисления.

Для этого на языке C/C++ разработайте функцию void $print16(\text{void} *p)$, которая печатает для 16-битной области памяти по заданному адресу p:

а) целочисленную беззнаковую интерпретацию в шестнадцатеричном представлении;
б) целочисленную беззнаковую интерпретацию в двоичном представлении;
в) целочисленную беззнаковую интерпретацию в десятичном представлении;
г) целочисленную знаковую интерпретацию в шестнадцатеричном представлении;
д) целочисленную знаковую интерпретацию в двоичном представлении;
е) целочисленную знаковую интерпретацию в десятичном представлении.

Штраф -2 балла, если количество выводимых цифр двоичного представления отлично от количества бит в числе (16 для $print16()$) либо если количество выводимых цифр шестнадцатеричного представления отлично от количества тетрад в числе ($\frac{16}{4} = 4$ для $print16()$).

Бонус $+1$ балл, если вывод $print16()$ занимает одну строку (так на экран поместится больше чисел). **Бонус $+2$ балла,** если при этом младшая цифра находится под младшей цифрой предыдущей строки (чего можно добиться заданием ширины поля вывода).

Примечание: для получения различных интерпретаций одного и того же участка памяти в C++ можно использовать объединения (`union`) или преобразование *указателя p в указатель на другой тип* оператором `reinterpret_cast`

или приведением в стиле C. Обратите внимание, что преобразование *значения в значение* оператором `static_cast` или приведением в стиле C не обеспечивает необходимого эффекта (хотя для целых типов одного размера *signed* ↔ *unsigned* преобразование значения в значение чаще всего приводит к тому же результату, что разыменованный преобразованный указатель).

Имена беззнаковых целочисленных типов C++ содержат ключевое слово *unsigned* (так, беззнаковый тип того же размера, что и *short*, называется *unsigned short*). Имена знаковых целочисленных типов могут содержать ключевое слово *signed* либо никакого (так, *signed short* и *short* — синонимы).

Соответственно, целочисленную беззнаковую интерпретацию памяти по адресу p для (а) и (в) можно получить, как `*(reinterpret_cast<unsigned short *>(p))`, знаковую для (г) и (е) — как `*(reinterpret_cast<short *>(p))`.

Шестнадцатеричное и десятичное представление целых чисел можно получить, используя различные форматы вывода функции *printf*() библиотеки libc либо манипуляторы *hex* и *dec* потокового вывода.

Шестнадцатеричный формат вывода для целочисленных переменных соответствует компактной записи двоичного кода, поэтому совпадает для беззнаковой и знаковой интерпретаций (и равен беззнаковой интерпретации в шестнадцатеричной системе счисления).

На вывод чисел с плавающей запятой манипуляторы *hex*, *oct*, *dec* никакого влияния не оказывают.

Двоичное (битовое) представление чисел можно получить, используя шаблон `std::bitset<N>`, где N — количество бит в представлении — необходимо задать вручную.

Проверьте работу функции *print*16() на 16-битных целочисленных переменных, принимающих следующие значения:

- минимальное целое 16-битное значение без знака;
- максимальное целое 16-битное значение без знака;
- минимальное целое 16-битное значение со знаком;
- максимальное целое 16-битное значение со знаком;
- значение x, соответствующее варианту (таблица Л0.1);
- значение y, соответствующее варианту (таблица Л0.1);

(запишите каждое из значений в 16-битную целочисленную переменную и передайте её адрес функции).

Убедитесь, что (а) и (г) — одно и то же шестнадцатеричное представление; аналогично, (б) и (д) — одно и то же двоичное представление. Измените функцию *print*16() так, чтобы убрать дублирование, и в дальнейшем пользуйтесь вариантом без дублей.

Варианты значений

Таблица Л0.1

(№ − 1)%2 +1	**Вариант**
1	$x = 9$, $y = -9$, $z =$ `0x88776155`
2	$x = 5$, $y = -5$, $z =$ `0xFF007101`

Задание Л0.№2. Разработайте на языке C/C++ функцию *print*32() аналогичную *print*16() для размера 32 (каждое из дублирующихся представлений — шестнадцатеричное (а) и (г), двоичное (б) и (д) — выводить один раз).

Кроме целочисленных интерпретаций, *print*32() должна рассматривать память по адресу p как 32-битное число с плавающей запятой («вещественное») одинарной точности (*float*) и печатать:

ж) интерпретацию с плавающей запятой в представлении с фиксированным количеством цифр после запятой;

з) интерпретацию с плавающей запятой в экспоненциальном представлении.

Примечание: для вывода в поток обратите внимание на манипулятор *setprecision*() и метод *setf*().

Проверьте работу *print*32() на 32-битных целочисленных переменных, принимающих следующие значения:

- минимальное целое 32-битное значение без знака;
- максимальное целое 32-битное значение без знака;
- минимальное целое 32-битное значение со знаком;
- максимальное целое 32-битное значение со знаком;
- целочисленное значение x, соответствующее варианту (таблица Л0.1);
- целочисленное значение y, соответствующее варианту (таблица Л0.1);
- целочисленное значение z, соответствующее варианту (таблица Л0.1);

(аналогично *print*16()), а также переменных с плавающей запятой (*float*):

- *float*-значение x, соответствующее варианту;
- *float*-значение y, соответствующее варианту;
- *float*-значение z, соответствующее варианту;

значения x, y, z смотрите в таблице Л0.1.

Сравните структуру целой переменной и *float*-переменной, имеющих равные значения (в частности, x). Сравните *float*-значения x и $y = -x$.

Задание Л0.№3. Бонус +2 балла. Разработайте на языке C/C++ функцию *print*64(), аналогичную *print*32() для размера 64 бита и, соответственно, числа с плавающей запятой двойной точности, *double*.

Аналогично Л0.№2, проверьте *print*64() на граничных целочисленных 64-битных значениях, целочисленных значениях x, y, z и *double*-значениях x, y, z.

Штраф −2 балла, если для *print*32() либо *print*64() количество выводимых цифр двоичного представления отлично от количества бит в числе либо количество выводимых цифр шестнадцатеричного — от количества тетрад.

Бонус +1 балл, если вывод *print*32() (и *print*64(), если она реализована) занимает одну строку. **Бонус +2 балла,** если при этом младшая цифра находится под младшей цифрой предыдущей строки.

Задание Л0.№4. Разработайте на языке C/C++ функцию, которая упаковывает два беззнаковых числа — k-битное v и $(m-k)$-битное w — в одно m-битное при помощи побитовых операций и сдвигов. Значения v и w функция получает как параметры; k смотрите в таблице Л0.2.

Рассчитайте минимальные и максимальные возможные значения v и w.

Варианты значений

Таблица Л0.2

(№ − 1)%3 +1	**Вариант**
1	$k = 5$, $m = 16$
2	$k = 6$, $m = 16$
3	$k = 12$, $m = 32$

Разработайте на языке C/C++ функцию, которая извлекает v и w из 16-битного значения. Проверьте корректность её работы (все значения должны выводиться при помощи *print*16() или *print*32(), в зависимости от разрядности).

Лабораторная работа 1

Двоичное представление информации. Исследование форматов файлов. Проектирование формата файлов

Версия 2023 г. — задания могут быть засчитаны только в 2023 г.

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 балл.

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, сиреневоболотное (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. О терминах

Символ кодирования = байт x86 = октет, а не печатный символ ASCII, KOI-8, Unicode etc.

Несжатый текст X = любой бинарный файл; сжатый текст, который следует в Y после кода алгоритма и служебной информации — тоже бинарный файл.

Не обманывайтесь принятой в кодировании терминологией! (Кто обманется и воспользуется функциями $fgetc()$ и т. п. — минус 2 балла).

Л1.3. Дополнительные бонусные и штрафные баллы

−2 балла за текстовое представление архива или если кодер обрабатывает только файлы в текстовом представлении — см. раздел Л1.2.

−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 $\widetilde{X}$.

+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

Версия 2023 г. — задания могут быть засчитаны только в 2023 г.

Штраф за одно пропущенное обязательное задание −2 балла.

Л2.1. Задание на лабораторную работу

Задание Л2.№1. Разработайте программу или используйте набор программ (в частности, это может быть набор скриптов-однострочников, использующий стандартные утилиты GNU/Linux), который 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.

Обратите внимание, что это — единственное *задание лабораторных работ, где символ кодирования не обязательно является байтом, а под исходным текстом не всегда понимается произвольный бинарный файл.*

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 (Л2.№1) в файлах, являющихся простым текстом в различных кодировках (см. «labs-files/files/plaintext»). Определите 4 наиболее частых октета среди всех используемых 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» (далее — файл 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

Версия 2023 г. — задания могут быть засчитаны только в 2023 г.

Штраф за одно пропущенное обязательное задание −3 балла.

Л3.1. Упрощённое задание (не более 7 баллов за работу)

Используйте демонстрационные программы любой свободной библиотеки для сжатия методом Хаффмана, Шеннона—Фано, Шеннона либо целочисленным арифметическим (интервальным) алгоритмом. Узнайте из документации, какой метод используется и что является символом кодирования.

Продемонстрируйте работу кодера и декодера. Сопоставьте длину сжатых данных с оценкой количества информации в исходном файле (Л2.№1).

Л3.2. Задание на лабораторную работу

Задание Л3.№1. Разработайте программу-кодек (аналогично Л1.№5), реализующую сжатие/разжатие файла алгоритмом 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 (если выполняются бонусные задания ниже — разным алгоритмам должны соответствовать разные коды);
- состав информации для декодирования и формат её хранения в файле;
- принятые при построении дерева уточнения (порядок сортировки при совпадении частот, для Шеннона—Фано ниже — 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).

Задание Л3.№4. Бонус +12 баллов. Разработайте программу-кодек (аналогично Л3.№1–Л3.№3), реализующую сжатие/разжатие файла целочисленным арифметическим (интервальным) алгоритмом.

Задание Л3.№5. Бонус +2 балла. Разработайте программу-кодек (аналогично Л3.№1–Л3.№3), реализующую сжатие/разжатие файла алгоритмом, соответствующим варианту (таблица Л3.1).

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

Таблица Л3.1

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

Задание Л3.№6. Бонус +2 балла.

Разработайте программу, рассчитывающую для файла длиной до $2^{32}-1$ байт коды Хаффмана в трёх случаях:

32) по ненормализованным частотам байтов в диапазоне $0...2^{32}-1$ (для хранения одного значения необходимо $|\nu_{32}| = 4$ байта);
8) по частотам, приведённым к диапазону 0...255 ($|\nu_8| = 1$ байт);
4) по частотам, приведённым к диапазону 0...15 ($|\nu_4| = \frac{1}{2}$ байта).

Частоты нормализовывать таким образом, чтобы ноль переходил в 0, единица в 1 (то есть чтобы ненулевые частоты не превращались в нулевые).

Для каждого $|\nu_i|$ программа должна рассчитывать:

– длину C_i сжатых данных файла в байтах;
– общую длину $D_i = C_i + 256 \cdot |\nu_i|$ данных, необходимых для распаковки (сжатых данных C_i и массива частот) в байтах.

Собственно архивы со сжатыми данными можно не создавать.

Для файла программа должна рассчитывать $i^*: \ D_{i^*} = \min\limits_i(D_i)$.

Для каждого из нескольких различных файлов сравните C_{32}, C_8 и C_4, а также D_{32}, D_8 и D_4; запишите в отчёт.

Дополнительно +1 балл, если программа перебирает для файла все возможные разрядности i от 3 до 32 бит; для каждой разрядности:

– приводит частоты к диапазону $0...(2^i - 1)$;
– по приведённым частотам рассчитывает C_i и $D_i = C_i + 256 \cdot \frac{i}{8} = C_i + 32 \cdot i$ в байтах.

Какая разрядность i, по вашему мнению, лучше подходит для использования (удобнее в чтении/записи и при этом даёт достаточно малое D_i)? Предложите способ сохранения массива из 256 i-битных частот в $32 \cdot i$ байтов.

Дополнительно +2 балла за графическое изображение зависимостей $\frac{C_i}{C_{32}}$ и $\frac{D_i}{D_{32}}$ от i для набора из 10 или более файлов разного типа. В итоге должно получиться одно изображение для C (либо со множеством графиков на нём, либо с «ящиками с усами») и одно для D, независимо от количества файлов — если на каждый файл будет свой график, баллы не начисляются.

Л3.3. Дополнительные бонусные и штрафные баллы

−4 балла за текстовое представление архива или если кодер обрабатывает только файлы в текстовом представлении — см. раздел Л1.2.

−2 балла, если массив частот может для какого-то одного файла иметь длину более 256 байтов.

−2 балла, если формат архива не соответствует Л1.№4 (штраф не начисляется, если в отчёте описана новая версия формата, которая логичнее исходной).

−4 балла, если даже сигнатура не совпадает с Л1.№3.

−2 балла, если ранее разработанный декодер Л1.№5 пытается декодировать архив Л3.№1, а не выдаёт сообщение о несоответствии алгоритма.

+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. Алгоритмы сжатия данных с учётом контекста

Версия 2023 г. — задания могут быть засчитаны только в 2023 г.

Штраф за одно пропущенное обязательное задание -2 балла.

Л4.1. Упрощённое задание (не более 7 баллов за работу)

Разработайте кодер и декодер «наивного» RLE. Продемонстрируйте работу кодера и декодера.

Штраф −3 балла за текстовое представление архива или если кодер обрабатывает только файлы в текстовом представлении — см. раздел Л1.2.

Л4.2. Задание на лабораторную работу

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

Вычислите минимальные и максимальные возможные значения L для своего варианта (для всех возможных случаев).

Задание Л4.№2. Бонус +12 баллов. Разработайте программу-кодек, реализующую сжатие/разжатие файла алгоритмом семейства LZ77 согласно варианту (таблица Л4.2).

Запись в таблице (S:10, L:6) обозначает: сначала записывается 10 бит S, затем 6 бит L. 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.

Вычислите минимальные и максимальные возможные значения S и L для своего варианта.

Задание Л4.№3. Разработайте программу-кодек, реализующую сжатие/разжатие файла алгоритмом семейства LZ78 согласно варианту (таблица Л4.2).

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

Варианты алгоритмов семейства RLE

Таблица Л4.1

(№ − 1)%3 +1	**Вариант**
1	флаг-бит сжатая/несжатая цепочка; цепочка из $L \geqslant 3$ одинаковых символов $c...c$ как $\Big(1:1,\ (L-3):7,\ c:8\Big)$, из $L \geqslant 1$ разных $c_1...c_L$ — как $\Big(0:1,\ (L-1):7,\ c_1:8,\ ...\ c_L:8\Big)$
2	префикс p; цепочка из $L \geqslant 4$ одинаковых символов $c...c,\ c \neq p$ как $\Big(p:8,\ (L-3):8,\ c:8\Big)$, цепочка из $L \geqslant 2$ символов $p...p$ как $\Big(p:8,\ (L-1):8,\ p:8\Big)$, одиночный p как $(p:8,\ 0:8)$
3	префикс p; цепочка из $L \geqslant 4$ одинаковых символов $c...c,\ c \neq p$ как $\Big(p:8,\ (L-4):8,\ c:8\Big)$, цепочка из $L \geqslant 1$ символов $p...p$ как $\Big(p:8,\ (L-1):8,\ p:8\Big)$, в том числе одиночный p как $(p:8,\ 0:8,\ p:8)$

Варианты алгоритмов семейства LZ77

Таблица Л4.2

(№ − 1)%7 +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-бит ссылка/цепочка; ссылка $\{S, L\}$ как $\Big(1:1,\ (L-3):7,\ (S-1):8\Big)$, цепочка $c_1...c_L$ как $\Big(0:1,\ (L-1):7,\ c_1:8,\ ...\ c_L:8\Big)$,
4	флаг-биты ссылка/символ группируются во флаг-байты; ссылка $\{S, L\}$ записывается как $\Big(S:10,\ (L-3):6\Big)$ — LZJB
5	флаг-биты ссылка/символ группируются во флаг-байты; ссылка $\{S, L\}$ записывается как $\Big((L-3):6,\ S:10\Big)$
6	префикс p; ссылка $\{S, L\}$ как $\Big(p:8,\ S:10,\ (L-4):6\Big)$, символ $c = p$ в тексте — как $\Big(p:8,\ 0:8,\ 0:8\Big)$
7	префикс p; ссылка $\{S, L\}$ как $\Big(p:8,\ (L-3):6,\ (S-1):10\Big)$, символ $c = p$ в тексте — как $\Big(p:8,\ 0:8\Big)$

Варианты алгоритмов семейства LZ78

Таблица Л4.3

(№ − 1)%2 +1	**Вариант**
1	концепт Зива и Лемпеля 1978 года
2	LZW

Л4.3. Дополнительные бонусные и штрафные баллы

+8 баллов, если реализованные программы позволяют применить к файлу последовательно любой алгоритм сжатия с учётом контекста, а затем сжатие без учёта контекста.

Лабораторная работа 5

Помехи. Помехозащитное кодирование

Версия 2023 г. — задания могут быть засчитаны только в 2023 г.

Штраф за одно пропущенное обязательное задание −2 балла.

Л5.1. Упрощённое задание (не более 7 баллов за работу)

Используйте демонстрационные программы любой свободной библиотеки для защиты от помех методом Хэмминга либо Рида—Соломона. Узнайте из документации, какой метод используется, какое количество ошибок E он исправляет в блоке, размер блока, тип исправляемой ошибки.

Продемонстрируйте, что E ошибок при декодировании исправляется, а $E + 1$ — уже нет.

Л5.2. Задание на лабораторную работу

Задание Л5.№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 Л5.1). 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 Л5.1

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

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

Задание Л5.№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 Л5.№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.

Задание Л5.№4. Доработайте универсальный декодер из задания Л3.№2 и кодер из задания Л3.№3 с учётом реализованных алгоритмов.

Л5.3. 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.

+8 баллов, если реализованные программы позволяют применить к файлу последовательно любое сжатие, а затем защиту от помех (аналогично Л4).