МАТЕРИАЛЫ ЗАДАНИЙ
Всероссийской междисциплинарной олимпиады школьников 8–11 класса «Национальная технологическая олимпиада»

по профилю «Летающая робототехника»

2024/25 учебный год

ntcontest.ru

УДК 373.5.016:[629.7:681.51]
ББК 74.263.2
Л52

Авторы:
Ф. А. Баталин, А. Е. Баталов, О. В. Зубков, К. Д. Кириченко, Н. Ю. Кузнецов, Д. А. Руфин, О. А. Рябова, В. Н. Трубицына

Л52 Всероссийская междисциплинарная олимпиада школьников 8–11 класса «Национальная технологическая олимпиада». Учебно-методическое пособие
Том 14 **Летающая робототехника**
— М.: Ассоциация участников технологических кружков, 2025. — 234 с.

ISBN 978-5-908021-13-5

Данное пособие разработано коллективом авторов на основе опыта проведения всероссийской междисциплинарной олимпиады школьников 8–11 класса «Национальная технологическая олимпиада» в 2024/25 учебном году, а также многолетнего опыта проведения инженерных соревнований для школьников. В пособии собраны основные материалы, необходимые как для подготовки к олимпиаде, так и для углубления знаний и приобретения навыков решения инженерных задач.

В издании приведены варианты заданий по профилю Национальной технологической олимпиады за 2024/25 учебный год с ответами, подробными решениями и комментариями. Пособие адресовано учащимся 8–11 классов, абитуриентам, школьным учителям, наставникам и преподавателям учреждений дополнительного образования, центров молодежного и инновационного творчества и детских технопарков.

Методические материалы также могут быть полезны студентам и преподавателям направлений, относящихся к группам:
09.00.00 Информатика и вычислительная техника
12.00.00 Фотоника, приборостроение, оптические и биотехнические системы и технологии
15.00.00 Машиностроение
23.00.00 Техника и технологии наземного транспорта
24.00.00 Авиационная и ракетно-космическая техника
25.00.00 Аэронавигация и эксплуатация авиационной и ракетно-космической техники
27.00.00 Управление в технических системах

ISBN 978-5-908021-13-5

УДК 373.5.016:[629.7:681.51]
ББК 74.263.2

Оглавление

1. Введение

1.1. Национальная технологическая олимпиада

Всероссийская междисциплинарная олимпиада школьников 8–11 класса «Национальная технологическая олимпиада» (далее — Олимпиада, НТО) проводится в соответствии с распоряжением Правительства Российской Федерации от 10.02.2022 № 211-р при координации Министерства науки и высшего образования Российской Федерации и при содействии Министерства просвещения Российской Федерации, Министерства цифрового развития, связи и массовых коммуникаций Российской Федерации, Министерства промышленности и торговли Российской Федерации, Ассоциации участников технологических кружков, Агентства стратегических инициатив по продвижению новых проектов, АНО «Россия — страна возможностей», АНО «Платформа Национальной технологической инициативы» и Российского движения детей и молодежи «Движение Первых».

Проектное управление Олимпиадой осуществляет структурное подразделение Национального исследовательского университета «Высшая школа экономики» — Центр Национальной технологической олимпиады. Организационный комитет по подготовке и проведению Национальной технологической олимпиады возглавляют первый заместитель Руководителя Администрации Президента Российской Федерации С. В. Кириенко и заместитель Председателя Правительства Российской Федерации Д. Н. Чернышенко.

Национальная технологическая олимпиада — это командная инженерная Олимпиада, позволяющая школьникам работать в самых передовых инженерных направлениях. Она базируется на опыте Олимпиады Кружкового движения НТИ и проводится с 2015 года, а с 2016 года входит в перечень Российского совета олимпиад школьников и дает победителям и призерам льготы при поступлении в университеты.

Всего заявки на участие в десятом юбилейном сезоне (2024–25 гг.) самых масштабных в России командных инженерных соревнованиях подали более 140 тысяч школьников. Общий охват олимпиады с 2015 года превысил 880 тысяч участников.

НТО способствует формированию профессиональной траектории школьников, увлеченных научно-техническим творчеством и помогает им:

- определить свой интерес в мире современных технологий;
- получить опыт решения комплексных инженерных задач;
- осознанно выбрать вуз для продолжения обучения и поступить в него на льготных условиях.

Кроме того, НТО позволяет каждому участнику познакомиться с перспективными направлениями технологического развития, ведущими экспертами и найти единомышленников.

Ценности НТО

Национальная технологическая олимпиада — командные инженерные соревнования для школьников и студентов. Олимпиада создает уникальное пространство, основанное на общих ценностях и смыслах, которыми делятся все участники процесса: школьники, студенты, организаторы, наставники и эксперты. В основе Олимпиады лежит представление о современном технологическом образовании как новом укладе жизни в быстро меняющемся мире. Эта модель предполагает:

- доступность качественного обучения для всех, кто стремится к знаниям;
- возможность непрерывного развития;
- совместное формирование среды, где гуманитарные знания и новые технологии взаимно усиливают друг друга.

Это — образ общества будущего, в котором участники Олимпиады оказываются уже сегодня.

Решать прикладные задачи, нацеленные на умножение общественного блага

В заданиях Олимпиады используются актуальные вызовы науки и технологий, адаптированные под уровень школьников. Они имеют прикладной характер и отражают реальные потребности общества, а системное и профессиональное решение подобных задач способствует развитию общего блага. Олимпиада предоставляет возможность попробовать себя в этом направлении уже сегодня и найти единомышленников.

Создавать, а не только потреблять

Стремление к созданию нового ценится выше потребления готового, а ориентация на общественную пользу — выше личной выгоды. Это не исключает заботу о собственных интересах, но подчеркивает: творчество приносит больше удовлетворения, чем пассивное потребление. Олимпиада — совместный труд организаторов, партнеров и участников, в котором важнее стремление решать общие задачи, чем критика чужих усилий.

Работать в команде

Командная работа рассматривается не только как эффективный способ достижения целей, но и как основа для формирования сообщества, объединенного общими ценностями. Команда помогает раскрыть индивидуальность каждого, при этом сохраняя уважение к другим. Такие горизонтальные связи необходимы для реализации амбициозных технологических проектов. Олимпиада способствует формированию подобного сообщества и приглашает к его созданию всех заинтересованных.

Осваивать и ответственно развивать новые технологии

Сообщество Национальной технологической олимпиады — часть Кружкового движения НТИ, объединенные интересом к современным технологиям, стремлением

к их пониманию и созданию нового. Возможности технологий постоянно расширяются, однако развитие должно сопровождаться ответственностью. Этика инженера и ученого предполагает осознание последствий своих решений. Главное правило — создавая новое, не навредить.

Играть честно и пробовать себя

Ценится честная победа, достигнутая в рамках установленных правил. Это предполагает отказ от списывания, давления и манипуляций. Честная игра означает уважение к себе, команде и соперникам. Олимпиада поддерживается как безопасное пространство, где каждый может пробовать новое, не опасаясь ошибок, и постепенно становиться сильнее и увереннее в себе.

Быть человеком

Соревнования — это сложный и эмоционально насыщенный процесс, в котором особенно важны порядочность, вежливость и чуткость. Эмпатия, уважение и забота делают участие полезным и комфортным. Высоко ценится бережное отношение к людям и их труду, отказ от токсичной критики и готовность нести ответственность за слова и поступки. Участие в общем деле помогает не только окружающим, но и самому человеку.

Организационная структура НТО

НТО — межпредметная олимпиада. Спектр соревновательных направлений (профилей НТО) сформирован на основе актуального технологического пакета и связан с решением современных проблем в различных технологических отраслях. С полным перечнем направлений (профилей) можно ознакомиться на сайте НТО: https://ntcontest.ru/tracks/nto-school/.

Соревнования в рамках НТО проводятся по четырем трекам:

1. НТО Junior для школьников (5–7 классы).
2. НТО школьников (8–11 классы).
3. НТО студентов.
4. Конкурс цифровых портфолио «Талант НТО».

В 2024/25 учебном году 21 профиль НТО включен в Перечень олимпиад школьников, ежегодно утверждаемый Приказом Министерства науки и высшего образования Российской Федерации, а также в Перечень олимпиад и иных интеллектуальных и (или) творческих конкурсов, утверждаемый приказом Министерства просвещения Российской Федерации. Это дает право победителям и призерам профилей НТО поступать в вузы страны без вступительных испытаний (БВИ), получить 100 баллов ЕГЭ или дополнительные 10 баллов за индивидуальные достижения. Преимущества при поступлении победителям и призерам НТО предлагают более 100 российских вузов.

НТО для школьников 8–11 классов проводится в три этапа:

- Первый отборочный этап — заочный индивидуальный. Участникам предлагаются предметный тур, состоящий из задач по двум предметам, связанным

с выбранным профилем, а также инженерный тур, задания которого погружают участников в тематику профиля; образовательный модуль формирует теоретические знания и представления.

- Второй отборочный этап — заочный командный. На этом этапе участники выполняют как индивидуальные задания на проверку компетенций, так и командные задачи, соответствующие выбранному профилю.
- Заключительный этап — очный командный. В течение 5–6 дней команды участников со всей страны, успешно прошедшие оба отборочных этапа, соревнуются в решении комплексных прикладных инженерных задач.

Профили НТО 2024/25 учебного года и соответствующий уровень РСОШ

Профили II уровня РСОШ:

- Автоматизация бизнес-процессов.
- Автономные транспортные системы.
- Беспилотные авиационные системы.
- Водные робототехнические системы.
- Инженерные биологические системы.
- Наносистемы и наноинженерия.
- Нейротехнологии и когнитивные науки.
- Технологии беспроводной связи.
- Цифровые технологии в архитектуре.
- Ядерные технологии.

Профили III уровня РСОШ:

- Анализ космических снимков и геопространственных данных.
- Аэрокосмические системы.
- Большие данные и машинное обучение.
- Геномное редактирование.
- Интеллектуальные робототехнические системы.
- Интеллектуальные энергетические системы.
- Информационная безопасность.
- Искусственный интеллект.
- Летающая робототехника.
- Спутниковые системы.
- Кластер «Виртуальные миры»:
    - ◇ Разработка компьютерных игр.
    - ◇ Технологии виртуальной реальности.
    - ◇ Технологии дополненной реальности.

Профили без уровня РСОШ:

- Инфохимия.
- Квантовый инжиниринг.
- Новые материалы.
- Программная инженерия в финансовых технологиях.

- Современная пищевая инженерия.
- Умный город.
- Урбанистика.
- Цифровые сенсорные системы.
- Разработка мобильных приложений.

Обратите внимание на то, что в олимпиаде 2025/26 учебного года список профилей, в т. ч. входящих в РСОШ, и уровни РСОШ могут поменяться.

Участие в НТО старшеклассников может принять любой школьник, обучающийся в 8–11 классе. Чаще всего Олимпиада привлекает:

- учащихся технологических кружков, интересующихся инженерными и робототехническими соревнованиями;
- школьников, увлеченных олимпиадами и предпочитающих межпредметный подход;
- энтузиастов передовых технологий;
- активных участников хакатонов, проектных конкурсов и профильных школ;
- будущих предпринимателей, ищущих команду для реализации стартап-идей;
- любознательных школьников, стремящихся выйти за рамки школьной программы.

Познакомить школьников с НТО и ее направлениями, а также мотивировать их на участие в Олимпиаде можно с помощью специальных мероприятий — Урока НТО и Дней НТО. Методические рекомендации для педагогов по проведению Урока НТО и организации Дня НТО в образовательной организации размещены на сайте: https://nti-lesson.ru. Здесь можно подобрать и скачать готовые сценарии занятий и подборки материалов по различным направлениям Олимпиады.

Участвуя в НТО, школьники получают возможность работать с практико-ориентированными задачами в области прорывных технологий, собирать команды единомышленников, погружаться в профессиональное сообщество, а также заработать льготы для поступления в вузы.

По всей стране работают площадки подготовки к НТО, которые помогают привлекать участников и проводят мероприятия по подготовке к этапам Олимпиады. Такие площадки могут быть открыты на базе:

- школ и учреждений дополнительного образования;
- частных кружков по программированию, робототехнике и другим технологическим направлениям;
- вузов;
- технопарков и других образовательных и научно-технических организаций.

Любое образовательное учреждение, ученики которого участвуют в НТО или НТО Junior, может стать площадкой подготовки к Олимпиаде и присоединиться к Кружковому движению НТИ. Подробные инструкции о том, как стать площадкой подготовки, размещены на сайте: https://ntcontest.ru. Условия регистрации и требования к ним актуализируются с развитием Олимпиады, а обновленная информация публикуется перед началом каждого нового цикла.

Наставники НТО

В Национальной технологической олимпиаде большое внимание уделяется работе с **наставниками** — людьми, сопровождающими участников на всех этапах подготовки и участия в Олимпиаде. Наставник оказывает поддержку как в решении организационных вопросов, так и в развитии технических и социальных навыков школьников, включая умение работать в команде.

Наставником НТО может стать любой взрослый, готовый помогать школьникам развиваться и готовиться к участию в инженерных соревнованиях. Это может быть:

- учитель школы или преподаватель вуза;
- педагог дополнительного образования;
- руководитель кружка;
- родитель школьника;
- специалист из технологической области или представитель бизнеса.

Даже если наставник сам не обладает достаточными знаниями в определенной области, он может привлекать к подготовке коллег и экспертов, а также оказывать поддержку и организовывать процесс обучения для самостоятельных учеников. Сегодня сообщество наставников НТО насчитывает более **7 000 человек** по всей стране.

Главная цель наставника — **организовать системную подготовку к Олимпиаде в течение всего учебного года**, поддерживать интерес и мотивацию участников, а также помочь им справляться с возникающими трудностями. Также наставник фиксирует цели команды и каждого участника, чтобы в дальнейшем можно было проанализировать развитие профессиональных и личных компетенций.

Основные направления работы наставника

Организационные задачи:

- Информирование и мотивация: наставник рассказывает учащимся об НТО, ее этапах и преимуществах, помогает с выбором подходящего профиля, ориентируясь на интересы и способности школьников.
- Составление программы подготовки: формируется расписание и план занятий, организуется работа по освоению необходимых знаний и навыков.
- Контроль сроков: наставник следит за календарем Олимпиады и напоминает участникам о сроках решения заданий отборочных этапов.

Содержательная подготовка:

- Оценка компетенций участников: наставник помогает определить сильные и слабые стороны учеников и подбирает задания и материалы для устранения пробелов.
- Подготовка к отборочным этапам: помощь в изучении рекомендованных материалов, заданий прошлых лет, онлайн-курсы по профилям.
- Подготовка к заключительному этапу: разбираются задачи заключительных этапов прошлых лет, отслеживаются подготовительные мероприятия (очные и дистанционные), в которых наставник рекомендует ученикам участвовать.

Развитие личных и командных навыков:

- Формирование команд: наставник помогает сформировать сбалансированные команды для второго отборочного и финального этапов, распределить роли, при необходимости ищет участников из других регионов и организует онлайн-коммуникацию.
- Анализ прогресса и опыта: после каждого этапа проводится совместная рефлексия, обсуждаются успехи и трудности, выявляются зоны роста и направления для дальнейшего развития.
- Поддержка и мотивация: наставник поддерживает интерес и энтузиазм участников (особенно в случае неудачных результатов), помогает справиться с разочарованием и сохранить настрой на дальнейшее участие.
- Построение индивидуальной образовательной траектории: наставник помогает школьникам осознанно планировать дальнейшее обучение: выбирать курсы, участвовать в конкурсах, определяться с вузами и направлениями подготовки.

Поддержка наставников НТО

Работе наставников посвящен отдельный раздел на сайте НТО: https://ntcontest.ru/mentors/.

Для систематизации знаний и подходов к работе наставников в рамках инженерных соревнований разработан курс «Дао начинающего наставника: как сопровождать инженерные команды»: https://stepik.org/course/124633/. Курс формирует общие представления об их работе в области подготовки участников к инженерным соревнованиям.

Для совершенствования профессиональных компетенций по направлениям профилей создан курс «Дао начинающего наставника: как развивать технологические компетенции»: https://stepik.org/course/186928/.

Для организации занятий с учениками педагогам предлагаются образовательные программы, разработанные на основе многолетнего опыта организации подготовки к НТО. В настоящий момент они представлены по передовым технологическим направлениям:

- компьютерное зрение;
- геномное редактирование;
- водная, летающая и интеллектуальная робототехника;
- машинное обучение и искусственный интеллект;
- нейротехнологии;
- беспроводная связь, дополненная реальность.

Программы доступны на сайте: https://ntcontest.ru/mentors/education-programs/.

Регистрируясь на платформе НТО, наставники получают доступ к личному кабинету, в котором отображается расписание отборочных соревнований и мероприятий по подготовке, требования к знаниям и компетенциям при решении задач отборочных этапов.

Сообщество наставников НТО существует и развивается. Ежегодно Кружковое движение НТИ проводит Всероссийский конкурс технологических кружков: https://konkurs.kruzhok.org/. Принять участие в конкурсе может каждый наставник.

В 2022 году было выпущено пособие «Технологическая подготовка инженерных команд. Методические рекомендации для наставников». Методические рекомендации предназначены для учителей технологий, а также наставников и педагогов кружков и центров дополнительного образования. Рекомендации направлены на помощь в процессе преподавания технологий в школе или в кружке. Пособие построено на примерах из реального опыта работы со школьниками, состоит из теоретических положений, посвященных популярным взглядам в педагогике на тему подготовки инженерных команд к соревнованиям. Электронное издание доступно по ссылке: https://journal.kruzhok.org/tpost/pggs3bp7y1-tehnologicheskaya-podgotovka-inzhenernih.

В нем рассмотрены особенности подготовки к пяти направлениям:

- Большие данные.
- Машинное обучение.
- Искусственный интеллект.
- Спутниковые системы.
- Летающая робототехника.

Для наставников НТО разработана и постоянно пополняется страница с материалами для профессионального развития: https://nto-forever.notion.site/c9b9cbd21542479b97a3fa562d15e32a.

1.2. Летающая робототехника

Профиль Летающая робототехника Национальной технологической олимпиады посвящен практической деятельности в области автоматизации управления квадрокоптерами при помощи компьютерного зрения, включая автоматический сбор, обработку и анализ данных.

Работа с летающей робототехнической платформой с открытым исходным кодом позволяет решать задачи в областях создания методов управления, позиционирования, сбора и обработки информации при помощи технического зрения, а также затрагивает поиск новых конструктивных решений для летающих робототехнических систем, их подсистем и отдельных модулей.

Основное назначение летающих робототехнических платформ — выполнение полезной работы для людей и оборудования. Поэтому все наработки впоследствии могут тиражироваться на сервисные беспилотники и решать повседневные задачи.

Организаторы профиля: ФГАОУ ВО «Санкт-петербургский государственный университет аэрокосмического приборостроения» (ГУАП) и ГБНОУ «Академия цифровых технологий» Санкт-Петербурга.

Для погружения в тематику профиля организаторами профиля разработаны:

- «Урок НТО», который знакомит учащихся с понятием компьютерного зрения, библиотекой алгоритмов OpenCV и предлагает написать программу на языке Python для распознавания изображения с камеры персонального компьютера;
- видеокурс по сборке, настройке и программированию квадрокоптера.

На первом отборочном дистанционном этапе участники решают задачи инженерного и предметного туров.

Предметный тур знакомит их с тематикой профиля и вовлекают в изучение образовательной программы, поэтому задачи выявляют общий уровень подготовки по школьным предметам: информатика и физика. По информатике проверяются базовые основы программирования, логике и комбинаторике, по физике сделан упор на разделы: механика, оптика, электричество.

К участию **во втором этапе** допускаются все участники, набравшие в первом отборочном этапе баллы выше установленного организаторами порогового значения. Они объединяются в команды по три–четыре человека, согласно имеющимся компетенциям:

- Инженер-программист (Python, Front-end) — работа с визуализацией и автоматизированной отправкой результатов мониторинга, написание кода для автономного полета квадрокоптера, разработка алгоритма безопасного полета квадрокоптера.
- Инженер-программист (C++, Python) — алгоритмы компьютерного зрения для реализации автономных миссий квадрокоптера, работа с датчиками. Работа в связке с ролью 1.
- Инженер-техник — моделирование и изготовление автономного комплекса для безопасного взлета/посадки квадрокоптера и его подзарядки, работа с датчиками, тестирование, техобслуживание и пилотирование квадрокоптера.

- Капитан/лидер команды — организация работы команды в GitHub (или аналоге), осуществление общего руководства работой команды, распределение обязанностей и контроль соблюдения дедлайнов. Рекомендуется совмещение данной роли с другими ролями.

Для решения и автоматической проверки используется платформа Яндекс.Контест. Задачи подразделяются на два блока.

Первый блок предназначается для проверки компетенций и определения роли в команде. Задания направлены на изучение строения и систем управления БПЛА, программирование на языке Python, работу с платформой ROS, сервером, компьютерным зрением (OpenCV), чтение чертежей и 3D-моделирование.

Второй блок состоит из комплексного задания, для решения которого необходимы навыки каждого члена команды. Здесь участники могут апробировать свои решения в симуляторе: инженеры — настроить квадрокоптер и подготовить виртуальный мир, а программисты — написать программный код для автономного полета квадрокоптера и визуализации данных. Чтобы успешно пройти второй блок, необходимо правильно распределить функции и наладить систему планирования и коммуникации внутри команды.

На втором этапе участники сталкиваются не только с отдельными элементами задачи предстоящего заключительного этапа, но заранее отрабатывают ее ключевые блоки — осознают свои сильные и слабые стороны и постепенно погружаются в формат командной работы.

Организаторами профиля предоставляется доступ к открытой документации платформы, видеолекциям и среде симуляции летающей робототехнической платформы «Клевер», где участники могут запускать и производить отладку своего программного кода, используя большинство функций, доступных на реальном квадрокоптере (https://clover.coex.tech/ru/programming.html).

Разработчиками профиля проводятся организационные вебинары и образовательные мастер-классы, направленные на подготовку ко второму и заключительному этапам. Получить помощь в решении текущих вопросов участники могут в онлайн-режиме в чате технической поддержки платформы «Клевер» и чате участников профиля в Telegram.

Заключительный этап посвящен разработке комплексной системы автоматизированного мониторинга объектов ТЭК при помощи квадрокоптера. Система состоит из трех частей:

- Разработка программного кода, позволяющего осуществлять мониторинг состояния и функционирования объектов ТЭК: нефтепроводов, систем теплоснабжения и т. д. Наблюдение за безопасностью нефтепровода (детектирование утечек и разливов нефти). Инспекция нефтепровода на предмет несанкционированных врезок. Определение тепловых потерь системы теплоснабжения.
- Написание программного кода для визуализации результатов мониторинга на карте и передачи на сервер в режиме реального времени.
- Создание автономного комплекса для безопасного взлета/посадки квадрокоптера и его подзарядки (дронопоинта и технической документации).

Эксплуатацию и техническое обслуживание оборудования (квадрокоптер и 3D-принтер) команды осуществляют самостоятельно. Для обеспечения равных условий командное задание направлено на решение одной задачи, разделенной на несколько

подзадач (полетных миссий) с постепенным повышением уровня сложности. Таким образом, ежедневно участники получают новое, усложненное задание, включающее в себя неизвестный ранее элемент.

Каждая подзадача имеет отдельные критерии оценки успешности полетной миссии, публикуется и оценивается в день ее выполнения. Команда-победитель определяется по наибольшему количеству баллов за сумму трех зачетных попыток (наиболее успешной автономной миссии) и по результатам инженерного задания.

Важной частью заключительного этапа является индивидуальный предметный тур, в ходе которого участники решают задачи по физике и информатике, проявляя свои знания и уровень подготовки по этим предметам.

Навыки и знания, приобретенные в процессе участия в профиле Летающая робототехника, закладывают основу для дальнейшей инженерной, исследовательской и конструкторской деятельности в различных областях. Многие участники продолжают разрабатывать свои проекты, участвовать в конкурсах и соревнованиях. Решения, полученные командами в практическом туре заключительного этапа, перспективны для масштабирования на большие производственные дроны, поскольку применяемые технологии идентичны.

Выпускники НТО продолжают свою деятельность в качестве соавторов профиля, а также кураторов, преподавателей и разработчиков как для российских, так и для международных мероприятий в рамках направления «Летающая робототехника». Это позволяет формировать преемственность возрастных категорий и расширять Российское сообщество профессионалов в сфере робототехники и эксплуатации БПЛА.

2. Первый отборочный этап

2.1. Работа наставника НТО на этапе

Педагог-наставник играет важную роль в подготовке участника к первому отборочному этапу Национальной технологической олимпиады. На этом этапе школьникам предстоит справиться как с предметными задачами, соответствующими профилю, так и с заданиями инженерного тура, погружающими в выбранную технологическую область.

Наставник может организовать подготовку участника, используя разнообразные форматы и ресурсы:

- Разбор заданий прошлых лет. Совместный анализ задач отборочного этапа предыдущих лет позволяет понять структуру, уровень сложности и типичные подходы к решению. Это формирует у школьника устойчивые стратегии работы с олимпиадными заданиями.
- Мини-соревнования. Проведение тренировочных турниров с заданиями предметных олимпиад муниципального уровня помогает развить соревновательный навык, тренирует скорость и уверенность при решении задач в ограниченное время.
- Углубленные занятия. Наставник может выстроить образовательную траекторию, опираясь на рекомендации разработчиков профиля, и провести занятия по ключевым темам. Это особенно важно для системного понимания предметной области.
- Использование онлайн-курсов. Для самостоятельной подготовки и проверки знаний участник может использовать предметные курсы НТО, размещенные на платформах Степик и Яндекс Контест. Наставник может также организовать занятия с использованием этих материалов в рамках групповой или индивидуальной подготовки.
- Привлечение внешних экспертов. Если у наставника нет достаточной экспертизы в какой-либо предметной области, он может пригласить других педагогов или специалистов для проведения тематических занятий.
- Поддержка в инженерном туре. Инженерный тур включает теоретические материалы и задания, помогающие глубже погрузиться в тематику профиля. Наставник может сопровождать изучение курса, помогать в разборе теоретических вопросов и тренировать участника на практических задачах.

Таким образом, наставник не только помогает систематизировать подготовку, но и мотивирует участника, создавая для него комфортную и продуктивную образовательную среду.

2.2. Предметный тур. Информатика

2.2.1. Первая волна. Задачи 8–11 класса

Задачи первой волны предметного тура по информатике открыты для решения. Соревнование доступно на платформе Яндекс.Контест: https://contest.yandex.ru/contest/63452/enter/.

Задача 2.2.1.1. Ускорение ускорения (10 баллов)

Имя входного файла: стандартный ввод или `input.txt`.

Имя выходного файла: стандартный вывод или `output.txt`.

Ограничение по времени выполнения программы: 1 с.

Ограничение по памяти: 64 Мбайт.

Условие

Рассмотрим модель движения тела. Будем фиксировать такие параметры, как координата, скорость, ускорение и ускорение ускорения (рывок). Если некоторый параметр равен a и имеет скорость изменения v, то в следующий момент времени этот параметр будет равен $a + v$.

Например, если тело имело координату, равную 10, скорость, равную 20, ускорение, равное 30 и ускорение ускорения, равное 40, то в следующий момент оно будет иметь координату 30, скорость 50 и ускорение 70. Ускорение ускорения будем считать в этой задаче постоянной величиной.

Задача довольно проста: тело в начальный момент времени 0 находится в точке с координатой 0, скоростью 0 и ускорением 0. На это тело действует постоянное ускорение ускорения, равное 6. Требуется определить, в точке с какой координатой окажется это тело в момент времени t.

Формат входных данных

В единственной строке находится одно число t, где $0 \leqslant t \leqslant 10^6$.

Формат выходных данных

Вывести одно число — координату, в которой окажется тело в момент времени t.

Примеры

Пример №1

Стандартный ввод
6
Стандартный вывод
120

Пример №2

Стандартный ввод
2
Стандартный вывод
0

Пример №3

Стандартный ввод
1000000
Стандартный вывод
999997000002000000

Решение

Ниже представлено решение на языке C++.

C++

```cpp
#include<bits/stdc++.h>
#define int long long
using namespace std;
signed main(){
    int t;
    cin >> t;
    cout << ((t * (t - 1)) * (t - 2)) << endl;
}
```

Задача 2.2.1.2. Двойное остекление (15 баллов)

Имя входного файла: стандартный ввод или `input.txt`.

Имя выходного файла: стандартный вывод или `output.txt`.

Ограничение по времени выполнения программы: 1 с.

Ограничение по памяти: 64 Мбайт.

Условие

У деда Василия есть два прямоугольных куска стекла. Один из них имеет размеры $a \times b$, другой — $c \times d$. Дед собирается из этих кусков сделать окно с двойным остеклением. Он хочет, чтобы окно было обязательно квадратным и как можно большим по размеру. Дед должен вырезать из имеющихся у него прямоугольников два одинаковых квадрата максимально возможного размера. Нужно написать программу, которая по заданным a, b, c, d найдет максимальные размеры квадратного окна. Имейте ввиду, что оба квадрата могут быть вырезаны и из одного прямоугольного куска стекла.

Формат входных данных

На вход подаются две строки. В первой строке находятся размеры первого прямоугольника a, b через пробел, во второй — размеры второго прямоугольника c, d через пробел, где $1 \leqslant a, b, c, d \leqslant 10^9$.

Формат выходных данных

Вывести одно число — максимальную сторону квадратного двойного окна, которое можно вырезать из заданных на входе прямоугольных кусков стекла. Ответ может быть нецелым, требуется вывести его с точностью 1 знак после десятичной точки.

Примеры

Пример №1

Стандартный ввод
5 10 9 6
Стандартный вывод
5

Пример №2

Стандартный ввод
4 10 9 6
Стандартный вывод
4.5

Комментарий

Второй пример показывает, что иногда лучше вырезать оба квадрата из одного и того же куска стекла.

Решение

Ниже представлено решение на языке C++.

C++

```cpp
#include<bits/stdc++.h>
#define int long long
using namespace std;
signed main(){
    double a, b, c, d;
    cin >> a >> b >> c >> d;
    double a0 = min({a, b, c, d});
    double a1 = min(max(a, b) / 2.0, min(a, b));
    double a2 = min(max(c, d) / 2.0, min(c, d));
    double ans =  max({a0, a1, a2});
    if( (int)ans == ans ){
        int ians = ans;
        cout << ians << endl;
        return 0;
    }
    cout.precision(1);
    cout << fixed<< ans << endl;
}
```

Задача 2.2.1.3. О золотой рыбке и... досках (20 баллов)

Имя входного файла: стандартный ввод или `input.txt`.

Имя выходного файла: стандартный вывод или `output.txt`.

Ограничение по времени выполнения программы: 1 с.

Ограничение по памяти: 64 Мбайт.

Условие

После событий известной сказки А. С. Пушкина старик решил принципиально не пользоваться услугами золотой рыбки. Поэтому для того чтобы изготовить новое корыто, он честно заготовил n одинаковых досок.

Но гостивший в это время у старика со старухой внук решил, что ему нужно научиться пилить. И, не сказав ничего своему деду, внук быстро распилил каждую из досок на две части. В итоге у старика оказались $2n$ кусков досок. Самое интересное, что все эти куски оказались разными по длине, но имели целочисленные размеры. К сожалению, старик забыл, какова была исходная длина целых досок.

Формат входных данных

В первой строке задается целое число n — исходное количество целых досок, где $1 \leqslant n \leqslant 10^5$.

Во второй строке заданы $2n$ целых чисел d_i — длины всех кусков, которые получились после «тренировки» внука, где $1 \leqslant d_i \leqslant 10^9$. Гарантируется, что эти числа попарно различны, и их можно разбить на пары одинаковых по сумме чисел.

Все эти части досок пронумерованы от 1 до $2n$ в том порядке, в котором они заданы на входе.

Формат выходных данных

В первую строку вывести одно число — исходную длину целых досок.

В следующих n строках вывести пары номеров кусков досок, которые составляют по длине целые доски. Номера выводить через один пробел, внутри пары сначала должен идти меньший номер, затем больший. Пары должны быть выведены в порядке возрастания первых номеров в парах.

Примеры

Пример №1

Стандартный ввод
3 4 8 2 3 6 7

Стандартный вывод
10 1 5 2 3 4 6

Комментарий

Отсортируем куски и далее будем брать один из начала и второй к нему из конца.

Решение

Ниже представлено решение на языке C++.

C++

```cpp
#include<bits/stdc++.h>
#define int long long
using namespace std;
signed main(){
    int n;
    cin >> n;
    vector<pair<int, int> > v(2 * n);
     for(int i = 0; i < 2 * n; i++){
        int d;
        cin >> d;
        v[i] = {d, i + 1};
    }
    sort(v.begin(), v.end());
    vector<pair<int, int> > ans(n);
    for(int i = 0; i < n; i++){
```

```
16          ans[i] = {v[i].second, v[2 * n - i - 1].second};
17          if(ans[i].first > ans[i].second){
18              swap(ans[i].first, ans[i].second);
19          }
20      }
21      sort(ans.begin(), ans.end());
22      cout << v[0].first + v.back().first<< endl;
23      for(int i = 0; i < n; i++){
24          cout << ans[i].first<<' '<< ans[i].second<< endl;
25      }
26  }
```

Задача 2.2.1.4. Бонусы и экономия (25 баллов)

Имя входного файла: стандартный ввод или `input.txt`.

Имя выходного файла: стандартный вывод или `output.txt`.

Ограничение по времени выполнения программы: 1 с.

Ограничение по памяти: 64 Мбайт.

Условие

Технология производства некоторой металлической детали предполагает вытачивание ее из металлической заготовки. При этом образуются стружки, которые не стоит выкидывать. Ведь из a комплектов стружек (оставшихся после обработки a заготовок) можно бесплатно выплавить еще одну заготовку, которую снова можно использовать для выточки детали и создания еще одного комплекта стружек.

Заготовки можно купить на оптовом складе, при этом в целях привлечения клиентов, проводится акция «купи b заготовок, тогда еще одну получишь бесплатно».

Требуется изготовить c деталей. Нужно определить минимальное число заготовок, которые нужно купить за деньги, чтобы с учетом бонусных заготовок и экономии на стружках можно было изготовить требуемое число деталей.

Формат входных данных

В одной строке через пробел заданы три целых числа a, b, и c такие, что $2 \leqslant$ $\leqslant a \leqslant 10^{18}$, $1 \leqslant b$, $c \leqslant 10^{18}$.

Формат выходных данных

Вывести одно целое число — минимальное количество заготовок, которые нужно купить, чтобы с учетом всех бонусов и экономии выточить c конечных деталей.

Примеры

Пример №1

Стандартный ввод
4 5 41
Стандартный вывод
26

Примечания

В примере из условия нужно закупить 26 заготовок. Тогда за каждые пять купленных заготовок будет предоставлена одна бесплатная, итого по акции добавится еще пять заготовок, то есть получится 31 заготовка. Далее из 31 заготовки выточится 31 деталь, останется 31 комплект стружек. Из каждых четырех комплектов выплавится дополнительная заготовка, получится семь заготовок и три комплекта стружек. Из семи заготовок выточится семь деталей и останется семь комплектов стружек, три комплекта стружек осталось с первого шага, итого 10 комплектов стружек. Из них выплавится еще две заготовки, дающие две детали и два комплекта стружек. Собрав эти два комплекта с двумя, оставшимися от 10, получим еще одну заготовку, из которой выточится еще одна деталь. Останется один комплект стружек, который уже никак не получится использовать. Итого будет произведена $31 + 7 + 2 + 1 = 41$ деталь.

Комментарий

Методом бинарного поиска можно подобрать минимальное необходимое количество исходных заготовок.

Решение

Ниже представлено решение на языке C++.

C++

```cpp
#include<bits/stdc++.h>
#define int long long
using namespace std;
int f1(int M, int a){
    int res = 0, z = 0;
    while(1){
        if(M == 0 && z < a){
            return res;
        }
        res += M;
        M = M + z;
        z = M % a;
        M = M / a;
    }
}
```

```
int f2(int M, int b){
    return M + M / b;
}
signed main(){
    int a, b, c;
    cin >> a >> b >> c;
    int L = 0, R = 1;
    while(f1(R, a) <= c){
        R *= 2;
    }
    while(R - L > 1){
        int M = (R + L) / 2;
        if(f1(M, a) < c){
            L = M;
        }
        else{
            R = M;
        }
    }
    int z = R;
    L = 0, R = 1;
    while(f2(R, b) <= z){
        R *= 2;
    }
    while(R - L > 1){
        int M = (R + L) / 2;
        if(f2(M, b) < z){
            L = M;
        }
        else{
            R = M;
        }
    }
    cout << R << endl;
}
```

Задача 2.2.1.5. Сон таксиста (30 баллов)

Имя входного файла: стандартный ввод или `input.txt`.

Имя выходного файла: стандартный вывод или `output.txt`.

Ограничение по времени выполнения программы: 1 с.

Ограничение по памяти: 64 Мбайт.

Условие

Одному таксисту приснился красочный сон. Во сне он живет и работает в некотором городе, где абсолютно все улицы с односторонним движением. Эти улицы устроены так, что невозможно проехать с какого-либо перекрестка так, чтобы вернуться обратно на этот же перекресток, то есть в дорожной сети города нет циклов.

Таким образом, если с перекрестка A можно попасть по направлению движения улиц на перекресток B, то люди вызывают такси, иначе их везет специальный муниципальный подземный транспорт бесплатно.

В связи с такими странными правилами, таксистам в этом городе разрешено законом везти пассажира по любому маршруту, не нарушающему направления движения. Все в этом городе привыкли к такой ситуации и абсолютно спокойно относятся к тому, что таксисты везут их самым длинным путем. Разумеется, заработок таксиста за одну поездку прямо пропорционален ее длине. Для упрощения будем считать, что стоимость 1 км поездки составляет ровно 1 руб.

Схема дорог города задана. Перекрестки города пронумерованы числами от 1 до n. Таксист в своем сне находится на перекрестке номер S. Напишите программу, которая подскажет ему, сколько он максимально сможет заработать, когда ему придет заказ от клиента. Так как он не знает, куда попросит его везти клиент, нужно для каждого перекрестка от 1 до n указать максимальную стоимость поездки до этого перекрестка из пункта S на такси. Если по правилам на такси добраться из пункта S до какого-то перекрестка нельзя, вывести -1.

Формат входных данных

Дорожная сеть задана следующим образом: в первой строке находятся два числа через пробел n и m — число перекрестков и число улиц в городе, где $2 \leqslant n, m \leqslant 2 \cdot \cdot 10^5$.

В следующих m строках задана очередная односторонняя улица в виде трех чисел A, B, d через пробел, где A — начало улицы, B — конец улицы и d — ее длина. $1 \leqslant A, B \leqslant n$, $1 \leqslant d \leqslant 10^9$. Гарантируется, что в этой дорожной сети нет циклов. Некоторые пары перекрестков могут быть соединены двумя и более односторонними улицами. Дорожная сеть может быть неплоской за счет мостов и тоннелей.

В последней строке ввода содержится номер стартового перекрестка S, $1 \leqslant S \leqslant \leqslant n$.

Формат выходных данных

Вывести n чисел в одну строку через пробел. i-е число обозначает длину самого длинного пути с перекрестка номер S до перекрестка номер i. Если до перекрестка номер i от S нельзя доехать, не нарушая правила движения, вывести -1.

Примеры

Пример №1

Стандартный ввод
10 20 9 10 15 9 8 3 8 10 7 7 8 4 7 10 10 5 8 2 5 9 10

Стандартный ввод

```
5 6 5
7 6 5
4 6 8
3 6 4
3 4 6
5 3 2
2 5 2
2 3 3
3 1 5
1 4 2
2 1 7
4 7 4
6 8 1
5
```

Стандартный вывод

```
7 -1 2 9 0 18 13 19 10 26
```

Комментарий

Задача решается методом динамического программирования на ориентированном ациклическом графе.

Решение

Ниже представлено решение на языке C++.

C++

```cpp
#include<bits/stdc++.h>
#define int long long
using namespace std;
int n, m;
vector<vector<pair<int, int> > > G;
vector<int> order, used;
void dfs(int a){
    used[a] = 1;
    for(auto to : G[a]){
        if(!used[to.first]){
            dfs(to.first);
        }
    }
    order.push_back(a);
}
signed main(){
    cin >> n >> m;
    G.resize(n + 1);
    used.resize(n + 1, 0);
    for(int i = 0; i < m; i++){
        int a, b, d;
        cin >> a >> b >> d;
        G[a].push_back({b, d});
    }
```

```
    int s;
    cin >> s;
    dfs(s);
    reverse(order.begin(), order.end());
    vector<int> dp(n + 1, -1);
    dp[s] = 0;
    for(auto el : order){
        for(auto to : G[el]){
            dp[to.first] = max(dp[to.first], dp[el] + to.second);
        }
    }
    for(int i = 1; i <= n; i++){
        cout << dp[i] <<' ';
    }
}
```

2.2.2. Вторая волна. Задачи 8–11 класса

Задачи второй волны предметного тура по информатике открыты для решения. Соревнование доступно на платформе Яндекс.Контест: https://contest.yandex.ru/contest/63454/enter/.

Задача 2.2.2.1. Игра на планшете (10 баллов)

Имя входного файла: стандартный ввод или `input.txt`.

Имя выходного файла: стандартный вывод или `output.txt`.

Ограничение по времени выполнения программы: 1 с.

Ограничение по памяти: 64 Мбайт.

Условие

Маленький Андрей изучает геометрические фигуры при помощи игры на планшете. У него есть прямоугольные треугольники четырех цветов и ориентаций: желтые, зеленые, красные и синие. Для каждой разновидности треугольников есть заданное количество экземпляров этих треугольников. Более точно: у Андрея есть a желтых, b зеленых, c красных и d синих треугольников. Помимо этого у него есть прямоугольная таблица $n \times m$.

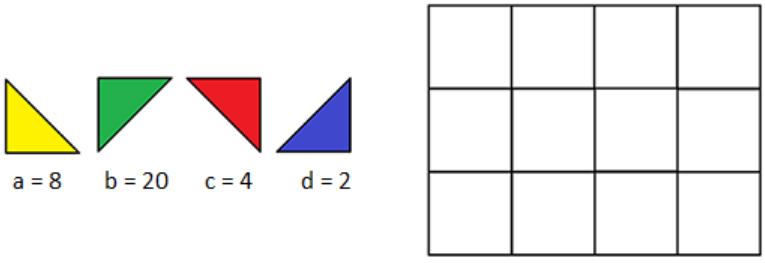


Рис. 2.2.1

Треугольники одного цвета имеют одну и ту же ориентацию, которую нельзя поменять. Андрей может только взять очередной треугольник и переместить его параллельным сдвигом в одну из ячеек этой прямоугольной таблицы. При этом в одну ячейку можно поместить либо вместе желтый и красный треугольники, либо вместе зеленый и синий, либо один любой треугольник из имеющихся.

Андрей хочет расположить в ячейках таблицы как можно больше треугольников из тех, что у него имеются. Нужно подсказать ему максимальное количество треугольников, которые получится разместить в таблице.

Формат входных данных

В первой строке содержатся четыре целых числа a, b, c и d через пробел — количество желтых, зеленых, красных и синих треугольников соответственно.

Во второй строке содержатся два целых числа n и m через пробел — размеры прямоугольной таблицы.

Все числа в пределах от 1 до 10^9.

Формат выходных данных

Вывести одно число — максимальное количество треугольников, которые можно при заданных условиях разместить в таблице.

Примеры

Пример №1

Стандартный ввод
`8 20 4 2` `3 4`

Стандартный вывод
`18`

Примечания

На рис. 2.2.2 представлен один из примеров размещения 18 треугольников из 34 заданных на входе.

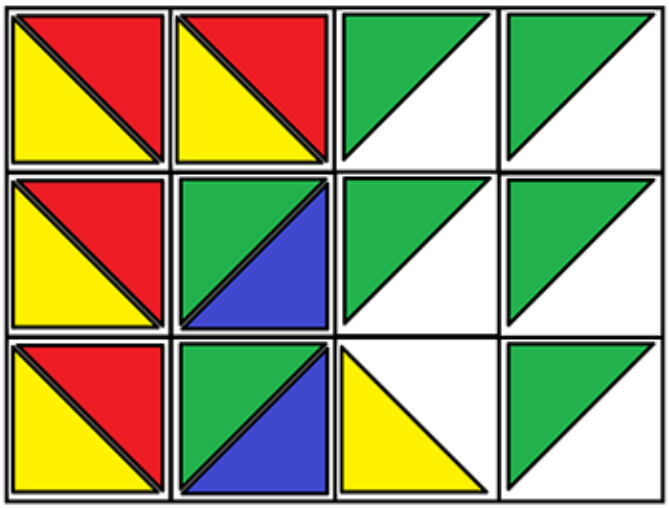


Рис. 2.2.2

Решение

Ниже представлено решение на языке C++.

C++

```
#include<bits/stdc++.h>
#define int long long
using namespace std;
signed main(){
    int a, b, c, d, n, m;
    cin >> a >> b >> c >> d >> n >> m;
    if(a > c){
        swap(a, c);
    }
    if(b > d){
        swap(b, d);
    }
    int f = a + b;
    int k = n * m;
    if(k <= f){
        cout << k * 2;
        return 0;
    }
    k -= f;
    c -= a;
    d -= b;
    cout << f * 2 + min(k, c + d) << endl;
}
```

Задача 2.2.2.2. Старая задача на новый лад (15 баллов)

Имя входного файла: стандартный ввод или `input.txt`.

Имя выходного файла: стандартный вывод или `output.txt`.

Ограничение по времени выполнения программы: 1 с.

Ограничение по памяти: 64 Мбайт.

Условие

Одна старая задача имеет следующий вид:

«Разбить число 45 на сумму четырех слагаемых так, что если к первому прибавить 2, из второго вычесть 2, третье умножить на 2, а четвертое разделить на 2, то получится одно и то же число».

Ответ к этой задаче — четыре числа 8, 12, 5 и 20. Можно убедиться, что в сумме они дают число 45, а если с каждым из них проделать соответствующую арифметическую операцию, то получится одно и то же число 10.

Необходимо решить чуть более общую задачу: даны числа n и k. Нужно представить число n в виде суммы четырех целых неотрицательных слагаемых $a + b + c + d$ таких, что $a + k = b - k = c \cdot k = d/k$. Гарантируется, что для заданных n и k такое разбиение существует.

Формат входных данных

В одной строке через пробел два числа n и k, где $1 \leqslant n \cdot k \leqslant 10^{18}$.

Формат выходных данных

Вывести через пробел в одну строку четыре целых неотрицательных числа a, b, c, d таких, что $a + b + c + d = n$ и $a + k = b - k = c \cdot k = d \,/\, k$.

Примеры

Пример №1

Стандартный ввод
`45 2`

Стандартный вывод
`8 12 5 20`

Пример №2

Стандартный ввод
`128 7`

Стандартный вывод
`7 21 2 98`

Решение

Ниже представлено решение на языке C++.

C++

```cpp
#include<bits/stdc++.h>
#define int long long
using namespace std;
signed main(){
    int n, k;
    cin >> n >> k;
    int x = (k * n) / (k * k + 2 * k + 1);
    cout << x - k <<' '<< x + k <<' '<< x / k <<' '<< x * k << endl;
}
```

Задача 2.2.2.3. Ладья и обязательная клетка (20 баллов)

Имя входного файла: стандартный ввод или `input.txt`.

Имя выходного файла: стандартный вывод или `output.txt`.

Ограничение по времени выполнения программы: 1 с.

Ограничение по памяти: 64 Мбайт.

Условие

Шахматная ладья находится в левом верхнем углу прямоугольного поля, разбитого на клетки размером $n \times m$. n обозначает число строк, m — число столбцов. Она хочет попасть в правую нижнюю клетку этого поля кратчайшим путем. Ладья может передвигаться либо вправо, либо вниз на любое количество клеток. Ладья обязана посетить заданную клетку с координатами (x, y), где x — номер строки этой клетки, а y — номер ее столбца.

Требуется найти количество способов построить путь ладьи из левого верхнего угла в правый нижний, которые проходят через обязательную клетку с заданными координатами.

Формат входных данных

В первой строке находятся два числа через пробел: n — число строк и m — число столбцов прямоугольного поля, $2 \leqslant n, m \leqslant 25$. Во второй строке через пробел находятся координаты (x, y) обязательной для посещения клетки, где $1 \leqslant x \leqslant n$, $1 \leqslant y \leqslant m$. Координаты x и y не совпадают с координатами левой верхней и правой нижней клеток.

Формат выходных данных

Вывести одно число — количество кратчайших путей ладьи из верхней левой в правую нижнюю клетку, проходящих через заданную клетку.

Примеры

Стандартный ввод
3 4 2 3

Стандартный вывод
6

Примечания

На рис. 2.2.3 представлены шесть путей, которыми ладья может пройти по полю размером 3×4, обязательно посещая по пути клетку $(2, 3)$.

Комментарий

Задачу можно решить как комбинаторными методами (произведение биномиальных коэффициентов), так и динамическим программированием.

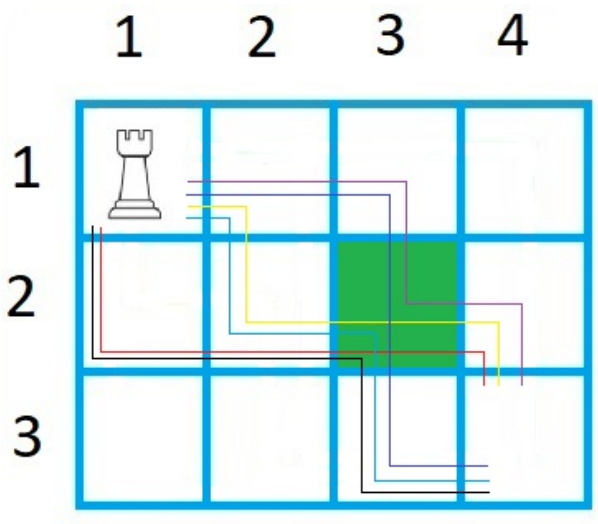


Рис. 2.2.3

Решение

Ниже представлено решение на языке C++.

C++

```
#include<bits/stdc++.h>
#define int long long
using namespace std;
signed main(){
    vector<vector<int> > bc(51, vector<int>(51, 0));
    bc[0][0] = 1;
    for(int i = 1; i <= 50; i++){
        for(int j = 0; j < 51; j++){
```

```
            bc[i][j] += bc[i - 1][j];
            if(j - 1 >= 0){
                bc[i][j] += bc[i - 1][j - 1];
            }
        }
    }
    int n, m, x, y;
    cin >> n >> m >> x >> y;
    int d1 = bc[x - 1 + y - 1][x - 1];
    int d2 = bc[n - x + m - y][n - x];
    int ans = d1 * d2;
    cout << ans << endl;
}
```

Задача 2.2.2.4. Танец с цифрами (25 баллов)

Имя входного файла: стандартный ввод или `input.txt`.

Имя выходного файла: стандартный вывод или `output.txt`.

Ограничение по времени выполнения программы: 1 с.

Ограничение по памяти: 64 Мбайт.

Условие

Десять танцоров репетируют на сцене новый танец. Каждый танцор одет в футболку, на которой написана одна из цифр от 1 до 9, цифры могут повторяться. Изначально они стоят в некотором порядке слева направо, и их цифры образуют некоторое десятизначное число A. Далее во время всего танца участники либо разбиваются на пять пар рядом стоящих танцоров и одновременно меняются местами внутри своих пар, либо самый левый танцор перемещается на самую правую позицию и становится самым правым танцором.

Сын постановщика танца от скуки на бумаге выписывает все получающиеся при каждом перемещении десятизначные числа. Так как танец длинный, то в итоге на бумаге окажутся все возможные числа, которые в принципе могут появится при этих условиях. Нужно найти разницу между самым большим и самым маленьким из этих чисел.

Формат входных данных

На вход подается одно десятизначное число A, обозначающее начальное расположение танцоров. В числе могут встречаться цифры от 1 до 9, некоторые из них могут повторяться.

Формат выходных данных

Вывести одно число, равное разности самого большого и самого маленького из чисел, которые могут быть получены во время танца.

Примеры

Пример №1

Стандартный ввод
1456531355
Стандартный вывод
5182160085

Примечания

Самое маленькое число, которое можно получить в примере, равно 1353155456, самое большое равно 6535315541.

Покажем, как получить эти числа из исходного числа 1456531355. Сначала получим самое большое следующим образом: две левых цифры, 1 и 4, переместим вправо, получим 5653135514, потом поменяем в парах цифры местами и получим самое большое — 6535315541. Далее опять поменяем порядок в парах и в числе 5653135514 переместим три левых цифры 5, 6 и 5 вправо, получим 3135514565 и здесь снова поменяем порядок в парах, получим самое маленькое — 1353155456. Таким образом, искомая разница равна 5182160085.

Решение

Ниже представлено решение на языке C++.

C++

```
#include<bits/stdc++.h>
#define int long long
using namespace std;
signed main(){
    string s;
    cin >> s;
    string mx = s, mn = s;

    for(int i = 0; i < 5; i++){
        for(int j = 0; j < 10; j++){
            mx = max(mx, s);
            mn = min(s, mn);
            if(j < 9){
                s = s.substr(1) + s[0];
            }
        }
        for(int j = 0; j < 5; j++){
            swap(s[2 * j], s[2 * j + 1]);
        }
    }
    stringstream ssmn;
    ssmn << mn;
    int imn;
    ssmn >> imn;
    stringstream ssmx;
```

```
26      ssmx << mx;
27      int imx;
28      ssmx >> imx;
29      cout << imx - imn << endl;
30  }
```

Задача 2.2.2.5. Трудная сортировка (30 баллов)

Имя входного файла: стандартный ввод или `input.txt`.

Имя выходного файла: стандартный вывод или `output.txt`.

Ограничение по времени выполнения программы: 3 с.

Ограничение по памяти: 64 Мбайт.

Условие

Иннокентий работает в отделе сортировки перестановок, подотделе сортировки вставками. Его задача заключается в сортировке перестановок, предоставленных заказчиками. Перестановкой длины n называется такая последовательность чисел, в которой встречаются все числа от 1 до n без повторений в некотором порядке.

Перестановка считается отсортированной, если в ней все числа расположены по возрастанию, то есть она имеет вид $1, \ldots, n$.

Иннокентий начинает рабочий день с пустой последовательности чисел. За день он сортирует вставками перестановку длины n. В начале каждой операции вставки он получает очередное число a_i из перестановки заказчика, после чего обрабатывает его, вставляя в отсортированную последовательность из ранее полученных чисел. После каждого такого добавления последовательность уже обработанных чисел должна быть отсортирована по возрастанию.

Перед тем как вставить число a_i в последовательность, он может выбрать, с какого края последовательности начать вставку. Далее он устанавливает число a_i с этого края и последовательно меняет вставляемое число с рядом стоящим числом b_j до тех пор, пока число a_i не встанет на свое место. На каждую перестановку вставляемого числа a_i с числом b_j Иннокентий тратит b_j единиц энергии.

Дана перестановка длины n из чисел a_i в том порядке, в котором Иннокентий их будет обрабатывать. Подскажите ему, какое минимальное количество энергии ему потребуется потратить, чтобы отсортировать всю перестановку.

Формат входных данных

В первой строке находится одно целое число n — длина перестановки, где $1 \leqslant \leqslant n \leqslant 2 \cdot 10^5$.

Во второй строке содержится n целых чисел a_i через пробел в том порядке, в котором они поступают на обработку Иннокентию. Гарантируется, что эти числа образуют перестановку длины n, то есть каждое число от 1 до n содержится в заданном наборе ровно один раз.

Формат выходных данных

Вывести одно число — минимальные суммарные энергозатраты Иннокентия для сортировки вставками заданной на входе перестановки.

Примеры

Пример №1

Стандартный ввод
9 2 9 1 5 6 4 3 8 7
Стандартный вывод
43

Примечания

Первым устанавливается число 2. Оно ни с чем не меняется местами, поэтому затрат нет.

Далее устанавливается число 9. Выбираем правый край и ставим его туда без потерь энергии.

Затем устанавливаем число 1. Выбираем левый край, ставим его туда и снова потерь нет.

Теперь нужно вставить число 5. Если его вставлять с правого края, придется менять местами с 9, а если с левого, то с 1 и 2, что суммарно явно лучше. Итого затраты на вставку 5 равны 3.

Число 6 снова лучше вставить слева, затраты на его вставку равны 8.

Число 4 вставим слева за 3.

Число 3 так же слева за 3.

А вот число 8 лучше вставить справа за 9.

И осталось число 7. Если вставлять слева, то затратим 21, а если справа, то всего 17.

Итого на сортировку заданной перестановки потратили: $0 + 0 + 0 + 3 + 8 + 3 + 3 + 9 + 17 = 43$.

Комментарий

Построим дерево отрезков на сумму, при обработке числа a будем находить, какая сумма на данный момент меньше: от 1 до $a - 1$ или от $a + 1$ до n. Прибавим ее к ответу и поместим в позицию a это число a.

Решение

Ниже представлено решение на языке C++.

C++

```cpp
#include<bits/stdc++.h>
#define int long long
using namespace std;
const int LG = 19;
int N = (1 << LG);
vector<int> tr(2 * N, 0);
void upd(int pos, int x){
    pos += N;
    tr[pos] = x;
    pos /= 2;
    while(pos){
        tr[pos] = {tr[2 * pos]+ tr[2 * pos + 1]};
        pos /= 2;
    }
}
int get(int l, int r){
    l += N;
    r += N;
    int res = 0;
    while(l <= r){
        if(l % 2 == 1){
            res += tr[l];
        }
        if(r % 2 == 0){
            res += tr[r];
        }
        l = (l + 1) / 2;
        r = (r - 1) / 2;
    }
    return res;
}
signed main(){
    int n, a;
    cin >> n;
    int ans = 0;
    for(int i = 0; i < n; i++){
        cin >> a;
        int sl = get(0, a - 1);
        int sr = get(a + 1, N - 1);
        ans += min(sl, sr);
        upd(a, a);
    }
    cout << ans << endl;
}
```

2.2.3. Третья волна. Задачи 8–11 класса

Задачи третьей волны предметного тура по информатике открыты для решения. Соревнование доступно на платформе Яндекс.Контест: https://contest.yandex.ru/contest/63456/enter/.

Задача 2.2.3.1. Туннель (10 баллов)

Имя входного файла: стандартный ввод или `input.txt`.

Имя выходного файла: стандартный вывод или `output.txt`.

Ограничение по времени выполнения программы: 1 с.

Ограничение по памяти: 64 Мбайт.

Условие

Рассмотрим классическую задачу прохождения группы с одним фонариком по туннелю. Есть четыре человека, и у них есть один фонарик. Нужно перевести всю группу на другой конец туннеля. По туннелю можно проходить только с фонариком и только либо вдвоем, либо в одиночку. По этой причине придется сделать пять рейсов по туннелю: три рейса туда и два рейса обратно. Туда идут двое, обратно — один, возвращая фонарик еще не прошедшей части группы. У каждого из четырех человек своя скорость передвижения по туннелю, но некоторые скорости могут совпадать. Двое идут со скоростью самого медленного в этой паре. Нужно найти минимальное время, за которое можно перевести группу по туннелю.

Здесь, в зависимости от скоростей персонажей, есть две стратегии. Проиллюстрируем их на примерах.

Пусть есть люди A, B, C, D. У A — время прохождения туннеля 1 мин, у B — 4 мин, у C — 5 мин, у D — 10 мин. Здесь работает наиболее очевидная стратегия: самый быстрый переводит текущего и возвращается с фонариком обратно за следующим. При этой стратегии нужно проходить так:

- A, B туда, затрачено 4 мин;
- A обратно, затрачена 1 мин;
- A, C туда, затрачено 5 мин;
- A обратно, затрачена 1 мин;
- A, D туда, затрачено 10 мин.

Общее время $4 + 1 + 5 + 1 + 10 = 21$ мин.

Но не всегда эта стратегия оптимальна. Уменьшим время прохождения туннеля персонажем B до 2 мин. По вышеопределенной стратегии будет 19 мин ($2 + 1 + 5 + + 1 + 10 = 19$), но имеется более быстрое решение:

- A, B туда, затрачено 2 мин;
- A обратно, затрачена 1 мин;
- C, D туда, затрачено 10 мин;
- B обратно, затрачено 2 мин;
- A, B туда, затрачено 2 мин.

Общее время $2 + 1 + 10 + 2 + 2 = 17$ мин.

Заметим, что для предыдущего примера такая стратегия не работает: $4+1+10+ + 4 + 4 = 23$ мин.

Если же персонаж B проходит туннель за 3 мин (а все остальные так же, как и в примерах), то независимо от стратегии будет затрачено 20 мин. В этом случае считаем, что работает первая стратегия.

Поразмыслив, станет понятно, от какого условия зависит выбор стратегии. Далее будем всегда считать, что A движется не медленнее B, B движется не медленнее C, C движется не медленнее D.

Дано время прохождения туннеля персонажами A, C, D. Нужно найти границу `border` для B такую, что если определить для B время прохождения строго меньшее, чем `border`, то выгодна вторая стратегия, иначе — первая.

Формат входных данных

В одной строке задано три целых чисел через пробел — время прохождения туннеля персонажами A, C, D. Времена даны по неубыванию. Все числа на входе в пределах от 1 до 100.

Формат выходных данных

Вывести одно число — границу `border` для B такую, что если определить время прохождения им туннеля строго меньше, чем `border`, нужно использовать вторую стратегию, иначе — первую. Ответ может быть нецелым, поэтому вывести его нужно с одним знаком после десятичной точки.

Примеры

Пример №1

Стандартный ввод
`1 5 10`
Стандартный вывод
`3`

Решение

Ниже представлено решение на языке C++.

C++

```cpp
#include<bits/stdc++.h>
#define int long long
using namespace std;
signed main(){
    int A, C, D;
    cin >> A >> C >> D;
    cout.precision(1);
    cout << fixed << (A + C) / 2.0 << endl;
}
```

Задача 2.2.3.2. Математический пазл (15 баллов)

Имя входного файла: стандартный ввод или `input.txt`.

Имя выходного файла: стандартный вывод или `output.txt`.

Ограничение по времени выполнения программы: 1 с.

Ограничение по памяти: 64 Мбайт.

Условие

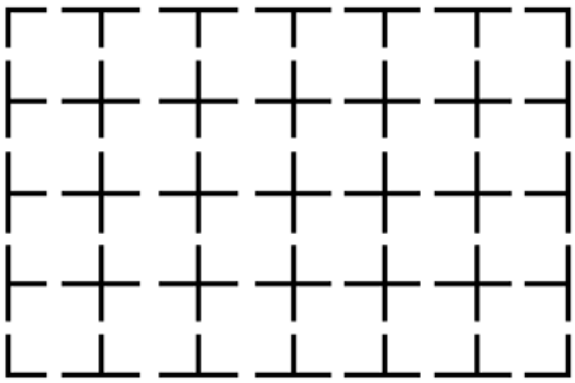


Рис. 2.2.4

Компания по производству пазлов решила освоить принципиально новый тип головоломок. Для этого берется прямоугольная решетка размера $n \times m$, каждый ее столбец и строка разрезаются посередине пополам. После этого образуются фигуры трех типов: четыре уголка, $2 \cdot (n+m-2)$ т-образных фигур и $(n-1) \cdot (m-1)$ крестиков.

Тому, кто решает головоломку, требуется сложить из этих фигур исходную прямоугольную решетку. При этом необходимо использовать абсолютно все имеющиеся в наличии фигуры.

Формат входных данных

В первой строке заданы через пробел два числа a — количество т-образных фигур и b — количество крестиков, которые находятся в одном из пазлов. При этом в наборе всегда есть еще четыре уголка. Известно, что этот комплект позволяет собрать прямоугольную решетку размера $n \times m$, где $1 \leqslant n,\ m \leqslant 10^9$.

Формат выходных данных

Требуется по числам a и b найти размеры исходной решетки n и m. Будем всегда считать, что $n \leqslant m$, то есть нужно вывести в одну строку через пробел два числа, первое из которых не превосходит второго, и вместе они задают размеры загаданной решетки.

Примеры

Пример №1

Стандартный ввод
16 15
Стандартный вывод
4 6

Пример №2

Стандартный ввод
0 0
Стандартный вывод
1 1

Комментарий

Задачу можно решить либо бинарным поиском, либо при помощи квадратного уравнения.

Решение

Ниже представлено решение на языке C++ при помощи бинпоиска.

C++

```cpp
#include<bits/stdc++.h>
#define int long long
using namespace std;
signed main(){
    int a, b;
    cin >> a >> b;
    int L = 0, R = a / 4 + 1;
    while(R - L > 1){
        int M = (R + L) / 2;
        int D = a / 2 - M;
        if(M * D <= b){
            L = M;
        }
        else{
            R = M;
        }
    }
    cout << L + 1 << ' ' << a / 2 - L + 1 << endl;
}
```

Задача 2.2.3.3. Восемь пирогов и одна свечка (20 баллов)

Имя входного файла: стандартный ввод или `input.txt`.

Имя выходного файла: стандартный вывод или `output.txt`.

Ограничение по времени выполнения программы: 1 с.

Ограничение по памяти: 64 Мбайт.

Условие

Мечта Карлсона наконец-то сбылась! Мама Малыша испекла восемь пирогов прямоугольной формы и в один из них воткнула свечку. После того как Карлсон съел семь пирогов, он решил-таки поделиться кусочком оставшегося восьмого пирога с Малышом. Но, будучи в хорошем настроении, он вынул из пирога свечу и предложил ему решить задачку.

«Так как я самый щедрый Карлсон в мире, то делить оставшийся пирог будешь ты. Но учти, ты должен разрезать пирог одним прямым разрезом так, чтобы линия прошла через один из углов и точку, где стояла свечка. После этого я выберу себе один из двух кусочков, а оставшийся, так и быть, достанется тебе».

Малыш не против этого замысла, однако считает, что разрезать пирог нужно как можно более справедливо, то есть так, чтобы разница между меньшим и большим кусками была как можно меньше. Подскажите Малышу, какой минимальной разницы между площадями кусков он сможет добиться.

Формат входных данных

В первой строке находятся два числа n и m через пробел — размеры прямоугольного пирога. Пирог размещен на координатной плоскости так, что его левый нижний угол находится в точке $(0, 0)$, а правый верхний — в точке (n, m), где $2 \leqslant n,\ m \leqslant 1\,000$.

Во второй строке находятся два числа x и y через пробел — координаты свечки, где $1 \leqslant x \leqslant n - 1$, $1 \leqslant y \leqslant m - 1$, то есть свечка находится строго внутри пирога.

Формат выходных данных

Вывести одно вещественное число с точностью не менее трех знаков после десятичной точки — минимальную разницу между площадями двух получающихся после разрезания кусков, которую сможет получить Малыш.

Примеры

Пример №1

Стандартный ввод
8 5 7 2
Стандартный вывод
12.571

Пример №2

Стандартный ввод
2 2 1 1
Стандартный вывод
0.000

Примечания

На рис. 2.2.5 представлены четыре варианта разделения пирога для первого примера из условия. Можно видеть, что самый близкий к справедливому способ разделения связан с разрезом из левого верхнего угла. Площадь треугольника в этом случае будет равна $96 / 7$, площадь четырехугольника равна $184 / 7$, и разница равна $88 / 7$, что при округлении до трех знаков равно 12,571.

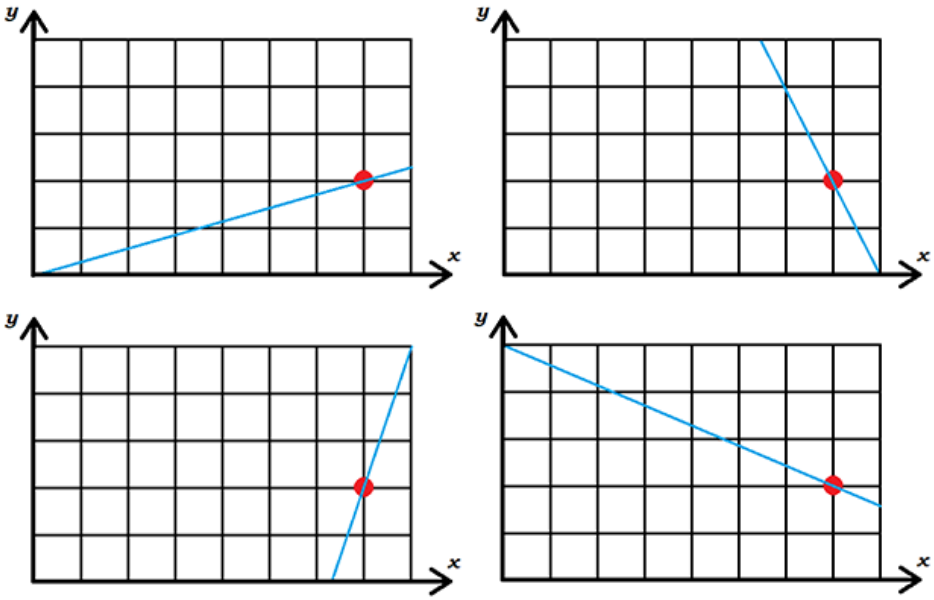


Рис. 2.2.5

Комментарий

Геометрия: для каждого из четырех случаев аккуратно находим катеты прямоугольного треугольника при помощи пропорции, затем находим площадь этого треугольника и, вычитая из всего прямоугольника эту площадь, находим площадь второго куска. Далее выбираем наиболее оптимальное отношение площадей.

Решение

Ниже представлено решение на языке C++.

C++

```
#include<bits/stdc++.h>
#define int long long
using namespace std;
const int INF = 1e18;
double katy(double x, double y, double n){
    return n * y / x;
}
double n, m, x, y;
double ans = INF;
double k1, k2;
void upd(){
    if(k1 < m){
        double st =k1 * n / 2;
        ans = min(ans, n * m - 2 * st);
    }
    else{
        double st =k2 * m / 2;
        ans = min(ans, n * m - 2 * st);
    }}
signed main(){
    cin >> n >> m >> x >> y;
    k1 = katy(x, y, n);
    k2 = katy(y, x, m);
    upd();
    k1 = katy(n - x, y, n);
    k2 = katy(y, n - x, m);
    upd();
    k1 = katy(x, m - y, n);
    k2 = katy(m - y, x, m);
    upd();
    k1 = katy(n - x, m - y, n);
    k2 = katy(m - y, n - x, m);
    upd();
    cout.precision(3);
    cout << fixed << ans<< endl;
}
```

Задача 2.2.3.4. Плетенка (25 баллов)

Имя входного файла: стандартный ввод или `input.txt`.

Имя выходного файла: стандартный вывод или `output.txt`.

Ограничение по времени выполнения программы: 1 с.

Ограничение по памяти: 64 Мбайт.

Условие

У Маши есть n полосок бумаги. i-я полоска имеет ширину 1 и длину a_i. Маша разделит эти полоски на две части и покрасит некоторые в желтый, а оставшиеся — в зеленый цвет. Она сама выберет, какие полоски как покрасить. Далее она хочет из этих полосок сплести максимально большую плетенку. Она расположит полоски одного цвета в некотором порядке горизонтально, а полоски другого цвета в некотором порядке вертикально. После этого она переплетет горизонтальные и вертикальные полоски так, что они будут чередоваться то сверху, то снизу, образуя в местах пересечения шахматную раскраску. Наконец, она обрежет выступающие края полосок так, что останется прямоугольная плетенка с ровными краями. Каждая клетка полученной плетенки должна иметь два слоя.

Маша хочет сплести максимально большую по площади прямоугольную плетенку. Подскажите ей, плетенку какой площади она сможет сделать. Заметим, что она может при создании плетенки использовать не все имеющиеся у нее полоски.

Формат входных данных

В первой строке на вход подается число n — количество полосок бумаги у Маши, где $2 \leqslant n \leqslant 2 \cdot 10^5$. Во второй строке через пробел заданы n целых чисел a_i через пробел — длины полосок, где $1 \leqslant a_i \leqslant 10^9$.

Формат выходных данных

Вывести одно число — площадь прямоугольника, форму которого может иметь самая большая плетенка Маши.

Примеры

Пример №1

Стандартный ввод
8 3 6 5 4 4 5 5 2

Стандартный вывод
12

Примечания

На рис. 2.2.6 представлен один из вариантов получения самой большой плетенки для полосок из примера. Синим обозначена граница полученной максимальной плетенки. Ее размер 3×4, и ее площадь 12. При ее создании Маша не должна использовать полоску номер 8, по этой причине неважно, как она раскрашена.

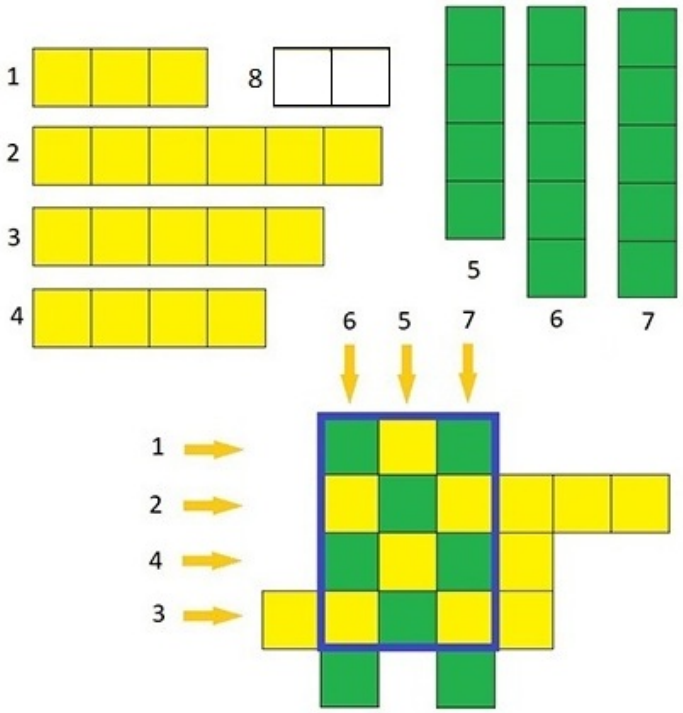


Рис. 2.2.6

Решение

Ниже представлено решение на языке C++.

C++

```cpp
#include<bits/stdc++.h>
#define int long long
using namespace std;
signed main(){
    int n;
    cin >> n;
    deque<int> v(n);
    for(int i = 0; i < n; i++){
        cin >> v[i];
    }
    sort(v.begin(), v.end());
    int ans = 0;
    int cnth = 0, minh;
    while(1){
        if(v.size() == 0){
            break;
        }
        cnth++;
        minh = v.back();
        v.pop_back();
        while(v.size() > 0 && v[0] < cnth){
            v.pop_front();
        }
        ans = max(ans, cnth * min(minh, (int)v.size()));
    }
    cout << ans << endl;
}
```

Задача 2.2.3.5. Английский в игровой форме (30 баллов)

Имя входного файла: стандартный ввод или `input.txt`.

Имя выходного файла: стандартный вывод или `output.txt`.

Ограничение по времени выполнения программы: 3 с.

Ограничение по памяти: 64 Мбайт.

Условие

Маша и Витя запоминают слова английского языка в оригинальной игровой форме. За день им нужно выучить n слов, где $20 \leqslant n \leqslant 100$, каждое из которых имеет длину от 5 до 8 символов. Маша выбирает из этого набора наугад несколько попарно различных слов (также от 5 до 8) и собирает их в одну строку без пробелов. Далее она переставляет буквы в этой строке так, что слова оказываются полностью перепутанными, и дает эту строку Вите. Теперь Витя должен восстановить все слова, которые выбрала Маша.

Но у Вити плохо получается, а Маша уже забыла, какие слова она выбрала. Нужно им помочь — написать программу, которая восстановит слова, выбранные Машей.

Формат входных данных

В первой строке находится строка, которую Маша предложила Вите. Во второй строке содержится число n — количество слов, которые нужно выучить детям, $20 \leqslant \leqslant n \leqslant 100$.

В следующих n строках содержатся эти слова по одному в строке. Все слова в этом наборе различны. Слова отсортированы в лексикографическом (алфавитном) порядке. Все слова состоят из маленьких букв от a до z. Обратите внимание, что в тестах к этой задаче все заданные слова реально существуют в английском языке и случайным образом выбраны из словаря.

Гарантируется, что длина каждого слова из предложенного набора (словаря) в пределах от 5 до 8, строка, которую получила Маша, может быть получена путем перестановки букв некоторых различных слов из предложенного словаря, причем, набор выбранных Машей слов определяется по ней однозначно. Количество слов, из которых составлена Машина строка, находится в пределах от 5 до 8.

Формат выходных данных

Вывести все слова, выбранные Машей, в алфавитном порядке по одному в строке.

Примеры

Пример №1

Стандартный ввод
stirbaexsudueoeidgomttcrnrwlunapntetacwri 24 bridge cranky document drawing farmer fighter figurine gravy havoc minimum reactant reply republic sonata soprano split subset tailor texture tomorrow trout vicinity wrist writer

Стандартный вывод
document drawing republic sonata texture wrist

Комментарий

В случае, выделенном в условии (слова являются случайными, взятыми из английского словаря), задача решается рекурсией с перебором вариантов.

Решение

Ниже представлено решение на языке C++.

C++

```cpp
#include<bits/stdc++.h>
#define int long long
using namespace std;
string frs;
int n;
vector<string> dict;
vector<int> msk(26, 0);
int cnt = 0;
vector<vector<int> > amsk;
vector<string> ans;
bool bigok = 0;
void p(int pos){
   if(!bigok){
        if(cnt == 0){
            sort(ans.begin(), ans.end());
            bigok = 1;
            return;
        }
        for(int i = pos; i < n; i++){
            string ts = dict[i];
            bool ok = 1;
            for(int j = 0; j < 26; j++){
                if(amsk[i][j] > msk[j]){
                    ok = 0;
                }
            }
            if(ok){
                ans.push_back(ts);
                for(int j = 0; j < 26; j++){
                    msk[j] -= amsk[i][j];
                    cnt -= amsk[i][j];
                }
                p(i + 1);
                if(!bigok){
                for(int j = 0; j < 26; j++){
                    msk[j] += amsk[i][j];
                    cnt += amsk[i][j];
                }
                ans.pop_back();
                }
            }
        }
   }
}
signed main(){
    cin >> frs;
    cin >> n;
    amsk.resize(n, vector<int>(26, 0));

    string ts;
    for(int i = 0; i < n; i++){
        cin >> ts;
        dict.push_back(ts);
    }
    for(int i = 0; i < n; i++){
        for(auto el : dict[i]){
            amsk[i][el - 'a']++;
        }
    }
```

```
60      for(auto el : frs){
61          msk[el - 'a']++;
62          cnt++;
63      }
64      p(0);
65      for(auto el : ans){
66          cout << el << endl;
67      }
68  }
```

2.2.4. Четвертая волна. Задачи 8–11 класса

Задачи четвертой волны предметного тура по информатике открыты для решения. Соревнование доступно на платформе Яндекс.Контест: https://contest.yandex.ru/contest/63457/enter/.

Задача 2.2.4.1. Квадратный флаг (10 баллов)

Имя входного файла: стандартный ввод или `input.txt`.

Имя выходного файла: стандартный вывод или `output.txt`.

Ограничение по времени выполнения программы: 1 с.

Ограничение по памяти: 64 Мбайт.

Условие

Одному портному заказали сделать одноцветный флаг. Особенность этого флага в том, что он должен быть квадратным. У портного есть два прямоугольных куска ткани заданного цвета. Один из них имеет размеры $a \times b$, другой — $c \times d$. Так как клиент будет платить пропорционально площади изготовленного флага, портной хочет сначала сшить имеющиеся у него прямоугольные куски, соединив их двумя какими-то сторонами, а затем из полученного полотна вырезать и сделать флаг с максимально большой стороной. Определить сторону получившегося у него флага.

Формат входных данных

На вход подаются две строки. В первой строке находятся размеры первого прямоугольника — целые числа a, b через пробел, во второй — размеры второго прямоугольника, также целые числа c, d через пробел, где $1 \leqslant a, b, c, d \leqslant 10^9$.

Формат выходных данных

Вывести одно число — сторону самого большого квадрата, который можно получить по условию задачи.

Примеры

Пример №1

Стандартный ввод
2 4 3 6

Стандартный вывод
4

Пример №2

Стандартный ввод
2 2 3 6

Стандартный вывод
3

Примечания

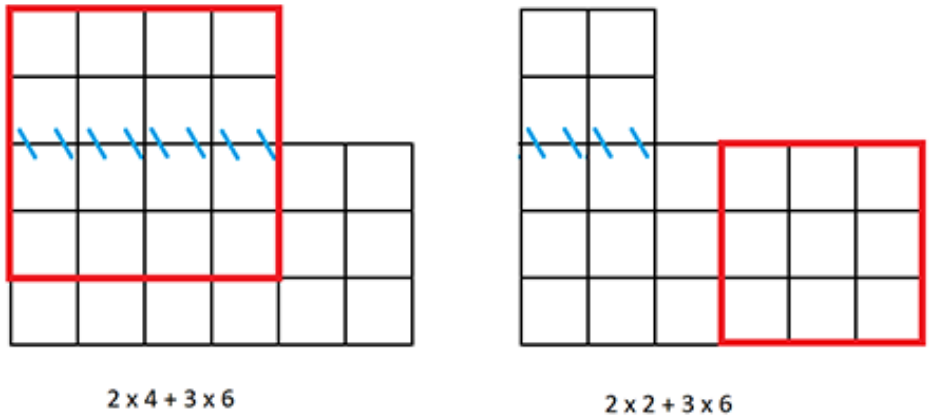


Рис. 2.2.7

На рис. 2.2.7 представлены иллюстрации для тестов из условия. Синими штрихами обозначено место сшивки двух кусков. Красный квадрат выделяет один из вариантов вырезания максимального квадрата.

Решение

Ниже представлено решение на языке C++.

C++

```cpp
#include<bits/stdc++.h>
#define int long long
using namespace std;
signed main(){
    int a, b, c, d;
    cin >> a >> b >> c >> d;
    int ans = max(min(a, b), min(c, d));
    int p1 = min(a + c, min(b, d));
    int p2 = min(a + d, min(b, c));
    int p3 = min(b + c, min(a, d));
    int p4 = min(b + d, min(a, c));
    ans = max({ans, p1, p2, p3, p4});
    cout << ans << endl;
}
```

Задача 2.2.4.2. Потерянная ДНК (15 баллов)

Имя входного файла: стандартный ввод или `input.txt`.

Имя выходного файла: стандартный вывод или `output.txt`.

Ограничение по времени выполнения программы: 1 с.

Ограничение по памяти: 64 Мбайт.

Условие

В данной задаче будем упрощенно считать, что ДНК представляется строкой длины от 10 до 100, состоящей из букв A, C, G, T.

Пусть даны две ДНК D_1 и D_2 одной и той же длины n. Выберем некоторое произвольное число i от 1 до $n-1$ и поменяем местами префиксы (начала) этих ДНК длины i. Будем говорить, что полученные новые две строки образованы путем скрещивания двух исходных по префиксу длины i.

Например, пусть $D_1 =$ **AACGGTAGGT**, а $D_2 =$ TCCCGGAACA. Выберем $i = 4$ и поменяем местами префиксы длины 4. Получим две новые ДНК, одна из которых будет иметь вид **AACG**GGAACA, а вторая — TCCC**GTAGGT**. Для наглядности были выделены части первой из них.

Полученные новые ДНК снова могут быть скрещены по любому префиксу длины от 1 до $n-1$.

Теперь можно рассмотреть популяцию из нескольких ДНК. Выберем из них две, произведем их скрещивание по префиксу какой-либо длины и поместим две новые ДНК в исходную популяцию. В данной задаче будем считать, что количество ДНК не увеличивается, то есть старые две ДНК заменяются на новые две ДНК.

Дана исходная популяция из m ДНК, каждая имеет одну и ту же длину n. После некоторого количества попарных скрещиваний была получена новая популяция. Но при итоговой обработке данных сведения об одной ДНК из новой популяции были потеряны. Задача состоит в отыскании этой потерянной ДНК по оставшимся $m-1$ ДНК из новой популяции.

Формат входных данных

В первой строке через пробел даны два числа n — длина ДНК и m — количество ДНК в исходной популяции, где $10 \leqslant n \leqslant 100$, $2 \leqslant m \leqslant 100$.

В следующих m строках содержится описание исходной популяции ДНК, каждая задается строкой длины n, состоящей из символов A, C, G и T.

Далее следует разделяющая строка, содержащая n символов «$-$».

Далее следует еще $m - 1$ строк, описывающих новую (заключительную) популяцию без одной ДНК.

Гарантируется, что данные верны, то есть $m - 1$ последняя ДНК является некоторой новой популяцией ровно без одной ДНК, полученной из исходной популяции, заданной в m первых строках.

Формат выходных данных

Вывести недостающую утерянную ДНК.

Примеры

Пример №1

Стандартный ввод
10 2 AACGGTAGGT TCCCGGAACA ---------- TCCCGTAGGT

Стандартный вывод
AACGGGAACA

Пример №2

Стандартный ввод
10 4 AACCGGTTAA ACGTACGTAC AAACCCGGGT CATTACTGGA ---------- AAGCGCTTAA CCACACGTGC AACTAGGGGT

Стандартный вывод
AATTCCTGAA

Комментарий

Для каждой позиции нужно найти недостающую букву из первого набора ДНК. Для этого удобнее всего использовать функцию `xor`.

Решение

Ниже представлено решение на языке C++.

C++

```cpp
#include<bits/stdc++.h>
#define int long long
using namespace std;
signed main(){
    int n, m;
    cin >> n >> m;
    vector<string> v1(m);
    for(int i = 0; i < m; i++){
        cin >> v1[i];
    }
    string d;
    cin >> d;
    vector<string> v2(m - 1);
    for(int i = 0; i < m - 1; i++){
        cin >> v2[i];
    }
    for(int j = 0; j < n; j++){
        int ss = 0;
        for(int i = 0; i < m; i++){
            ss ^= (int)(v1[i][j]);
        }
        for(int i = 0; i < m - 1; i++){
            ss ^= (int)(v2[i][j]);
        }
        cout << (char)(ss);
    }
    cout << endl;
}
```

Задача 2.2.4.3. Утомленные туристы (20 баллов)

Имя входного файла: стандартный ввод или `input.txt`.

Имя выходного файла: стандартный вывод или `output.txt`.

Ограничение по времени выполнения программы: 1 с.

Ограничение по памяти: 64 Мбайт.

Условие

Рассмотрим следующий вариант известной задачи на перемещение по туннелю группы из четырех человек. В общем виде она выглядит так: четыре туриста хотят пройти по темному туннелю. Имеется один фонарик. По туннелю можно перемещаться либо вдвоем, либо по одному, при этом у тех, кто движется в туннеле,

должен быть фонарик в руках. По этой причине движение должно быть следующим: двое переходят туда, один возвращается обратно и приносит фонарик тем, кто еще не перешел. После этого указанный маневр повторяется снова.

У каждого участника своя скорость движения в туннеле. Пусть участники проходят туннель за A, B, C и D мин. Если идут двое, то они движутся со скоростью того, кто идет медленнее. Требуется по заданным временам прохождения туннеля каждого из участников перевести их максимально быстро через туннель.

Немного усложним данную задачу. Введем фактор усталости. А именно, любой участник, пройдя по туннелю, устает и в следующий раз идет уже медленнее. После каждого прохождения туннеля время прохождения любого участника увеличивается на E мин. Например, если участник до начала движения проходит туннель за 1 мин, а показатель усталости E равен 3 мин, то первый раз участник пройдет туннель за 1 мин, второй раз — за 4 мин, третий раз — за 7 мин и т. д.

По заданным A, B, C, D и E узнать, за какое минимальное время можно провести всю группу через туннель согласно указанным правилам.

Формат входных данных

На вход подаются пять чисел. В первой строке через пробел четыре числа A, B, C и D — время прохождения туннеля каждым из четырех участников до того, как они начали движение. Во второй строке содержится число E — величина, на которую увеличивается время прохождения туннеля каждым участником после каждого перемещения. При этом $1 \leqslant A, B, C, D \leqslant 1\,000$, $0 \leqslant E \leqslant 1\,000$.

Формат выходных данных

Вывести одно число — минимальное время прохождения туннеля всей группой.

Примеры

Пример №1

Стандартный ввод
8 9 10 1 3
Стандартный вывод
44

Пример №2

Стандартный ввод
8 9 10 1 0
Стандартный вывод
29

Примечания

В первом примере при прохождении туннеля каждый турист устает и движется медленнее на 3 мин. Покажем, как перевести группу при этом за 44 мин.

Каждую ситуацию будем обозначать следующим образом: слева от двоеточия находятся туристы, которые стоят в начале туннеля, а справа — те, что стоят в конце туннеля. Туриста будем обозначать при помощи числа, соответствующего его текущему времени прохождения туннеля.

Тогда исходная ситуация имеет вид 1, 8, 9, 10 :.

Сначала идут туристы 1 и 8, каждый после перехода устает на 3 мин, получим ситуацию 9, 10 : 4, 11, затрачено 8 мин.

Обратно возвращается турист 4, он устает еще на 3 мин. Ситуация становится 7, 9, 10 : 11, затрачено $8 + 4 = 12$ мин.

Теперь идут туристы 7 и 9, получится ситуация 10 : 10, 11, 12, затрачено $8 + + 4 + 9 = 21$ мин.

Возвращается турист 10, получится 10, 13 : 11, 12, затрачено $8 + 4 + 9 + 10 = = 31$ мин.

Наконец, оставшиеся двое туристов 10 и 13 за 13 мин переходят туннель, итого затрачено $8 + 4 + 9 + 10 + 13 = 44$ мин.

Комментарий

Задача решается рекурсивным перебором всех вариантов прохождения.

Решение

Ниже представлено решение на языке C++.

C++

```cpp
#include<bits/stdc++.h>
#define int long long
using namespace std;
const int INF = 1e18;
vector<int> v(4);
int e, ans = INF;
void p(vector<int> &vl, vector<int> &vr, int tv){
    if(vl.size() == 2){
        ans = min(ans, tv + *max_element(vl.begin(), vl.end()));
        return;
    }
    for(int i = 0; i < vl.size() - 1; i++){
        for(int j = i + 1; j < vl.size(); j++){
            vector<int> vl1;
            for(int k = 0; k < vl.size(); k++){
                if(k != i && k != j){
                    vl1.push_back(vl[k]);
                }
            }
            vector<int> vr1 = vr;
```

```
21                vr1.push_back(vl[i] + e);
22                vr1.push_back(vl[j] + e);
23                int tmp = max(vl[i], vl[j]);
24                sort(vr1.rbegin(), vr1.rend());
25                vl1.push_back(vr1.back() + e);
26                vr1.pop_back();
27                p(vl1, vr1, tv + tmp + vl1.back() - e);
28            }
29        }
30    }
31    signed main(){
32        for(int i = 0; i < 4; i++){
33            cin >> v[i];
34        }
35        sort(v.begin(), v.end());
36        cin >> e;
37        vector<int> vl = v, vr;
38        p(vl, vr, 0);
39        cout << ans;
40    }
```

Задача 2.2.4.4. Проектируем мост (25 баллов)

Имя входного файла: стандартный ввод или `input.txt`.

Имя выходного файла: стандартный вывод или `output.txt`.

Ограничение по времени выполнения программы: 1 с.

Ограничение по памяти: 64 Мбайт.

Условие

При постройке моста используются два типа пролетов: П-образные (они прочные, но дорогие) и Т-образные (они дешевле, но менее надежные). Мост должен начинаться и заканчиваться П-образными пролетами. Любой Т-образный пролет должен иметь хотя бы один П-образный пролет в качестве соседнего.

Длина проектируемого моста — n пролетов. Муниципалитет выделил средства на постройку a П-образных и b Т-образных пролетов. При этом $a + b = n$. Требуется выяснить, сколькими способами при этих условиях можно скомпоновать мост. Два способа компоновки моста отличаются, если в одной на некоторой позиции стоит П-образный пролет, а в другой на этой же позиции стоит Т-образный пролет.

Формат входных данных

В одной строке через пробел заданы два числа: a — число П-образных пролетов и b — число Т-образных пролетов, на постройку которых выделены средства, где $2 \leqslant a \leqslant 10^6$, $0 \leqslant b \leqslant 10^6$.

Формат выходных данных

Вывести одно число — количество вариантов компоновки моста. Так как ответ может быть очень большим, требуется вывести остаток от его деления на 1 000 000 007 ($10^9 + 7$).

Примеры

Пример №1

Стандартный ввод
4 3

Стандартный вывод
7

Примечания

Для примера из условия имеется 7 вариантов компоновки моста (пробелы добавлены для лучшего восприятия вариантов):

П Т Т П Т П П

П Т Т П П Т П

П Т П Т Т П П

П Т П П Т Т П

П П Т П Т Т П

П П Т Т П Т П

П Т П Т П Т П

Комментарий

При заданных ограничениях задача решается только при помощи комбинаторики с вычислениями по модулю.

Решение

Ниже представлено решение на языке C++.

C++

```cpp
#include<bits/stdc++.h>
#define int long long
using namespace std;
const int INF = 1e18;
const int MOD = 1e9 + 7;
vector<int> f(2e6 + 1, 1);
```

```
int binpow (int a, int n) {
        int res = 1;
        while (n > 0) {
                if (n % 2 == 1)
                        (res *= a) %= MOD;
                (a *= a) %= MOD;
                n /= 2;
        }
        return res;
}

int bc(int n, int k){
    int res = f[n];
    int p1 = binpow(f[k], MOD - 2);
    int p2 = binpow(f[n - k], MOD - 2);
    (res *= p1) %= MOD;
    (res *= p2) %= MOD;
    return res;
}
signed main(){
    for(int i = 1; i <= 2e6; i++){
        f[i] = (f[i - 1] * i) % MOD;
    }
    int a, b;
    int ans = 0;
    cin >> a >> b;
    a--;
    for(int i = 0; i < a + 1; i++){
        if(2 * i <= b){
            int d = bc(a, i);
            if(b - 2 * i <= a - i){
                (d *= bc(a - i, b - 2 * i) ) %= MOD;
                (ans += d) %= MOD;
            }
        }
    }
    cout << ans << endl;
}
```

Задача 2.2.4.5. Джентльмены на прогулке (30 баллов)

Имя входного файла: стандартный ввод или `input.txt`.

Имя выходного файла: стандартный вывод или `output.txt`.

Ограничение по времени выполнения программы: 8 с.

Ограничение по памяти: 64 Мбайт.

Условие

По прямому участку улицы, которую будем считать отрезком AB длины d, прогуливаются n джентльменов. i-й джентльмен движется со скоростью v_i. Скорости всех джентльменов попарно различны. Дойдя до любого конца улицы, каждый джентльмен поворачивает и идет в обратную сторону.

При каждой встрече два джентльмена приветствуют друг друга, приподнимая головной убор. Приветствие происходит и в том случае, когда один джентльмен обгоняет другого. Если два джентльмена встречаются в момент их одновременного поворота, то происходит два приветствия: одно до поворота, другое — после поворота. Если происходит одновременная встреча трех и более джентльменов, то они приветствуют друг друга попарно, то есть каждый каждого. Допустим, если одновременно встретились четыре джентльмена где-то посреди улицы, произойдет шесть попарных приветствий. Если же эти четыре джентльмена встретились в момент их одновременного поворота, произойдет уже двенадцать приветствий.

В этой задаче считаем, что все действия происходят без остановок, то есть и повороты и приветствия происходят мгновенно. Джентльмены одновременно начинают свою прогулку из точки A в момент 0. В этот момент они уже производят свои первые попарные приветствия, то есть в момент 0 уже произведено $n \cdot (n-1)/2$ приветствий. Момент старта не считается моментом поворота, то есть на старте число приветствий не удваивается. Джентльмены гуляют достаточно долго, чтобы произошло любое заданное количество приветствий.

Требуется найти момент, в который было произведено k-е по порядку приветствие.

Формат входных данных

В первой строке ввода через пробел содержится два целых числа: d — длина отрезка AB и n — количество прогуливающихся джентльменов, где $1 \leqslant d \leqslant 200$, $2 \leqslant n \leqslant 2\,000$.

Во второй строке находятся n целых чисел v_i через пробел — скорости каждого джентльмена, где $1 \leqslant v_i \leqslant 2\,000$. Гарантируется, что все скорости попарно различны. Скорости даны в порядке возрастания, то есть $v_1 < v_2 < \ldots < v_n$.

В третьей строке содержится одно целое число k — номер требуемого приветствия, для которого нужно найти момент, когда оно произойдет, где $1 \leqslant k \leqslant 10^9$.

Формат выходных данных

Вывести одно вещественное число — время, когда произойдет k-е по порядку приветствие. Ответ вывести с точностью не менее двух знаков после десятичной точки.

Примеры

Пример №1

Стандартный ввод
5 4 2 5 8 10 6

Стандартный вывод
0.000

Пример №2

Стандартный ввод
5 4 2 5 8 10 7
Стандартный вывод
0.556

Пример №3

Стандартный ввод
5 4 2 5 8 10 11
Стандартный вывод
1.000

Пример №4

Стандартный ввод
5 4 2 5 8 10 15
Стандартный вывод
1.429

Пример №5

Стандартный ввод
5 4 2 5 8 10 17
Стандартный вывод
1.667

Пример №6

Стандартный ввод
5 4 2 5 8 10 19
Стандартный вывод
1.667

Пример №7

Стандартный ввод
5 4 2 5 8 10 21
Стандартный вывод
2.000

Примечания

На рис. 2.2.8 приведено положение джентльменов из примеров в моменты времени 0, 1 и 2. Джентльмены обозначены своими скоростями. Стрелками обозначены направления их движения в соответствующий момент. Перечислим и пронумеруем в порядке возрастания моменты попарных приветствий этих джентльменов до момента времени 2 включительно. Если два и более приветствия происходят одновременно, неважно какое из них конкретно имеет номер k, главное, что они происходят в один и тот же определенный момент времени.

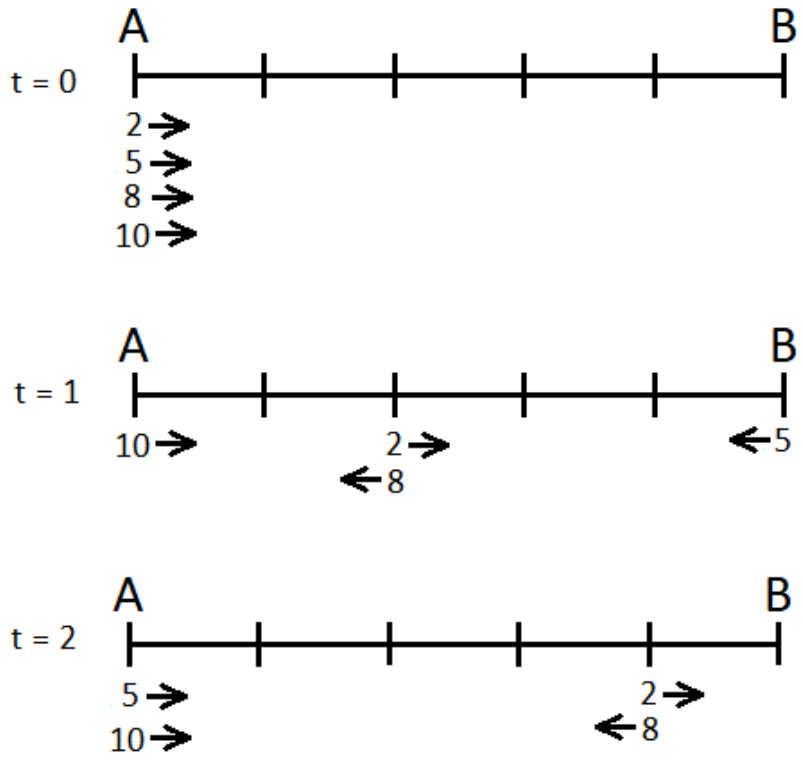


Рис. 2.2.8

1. 2 и 5 приветствуют друг друга в момент 0 (изображено на рис. 2.2.8).
2. 2 и 8 приветствуют друг друга в момент 0 (изображено на рис. 2.2.8).
3. 2 и 10 приветствуют друг друга в момент 0 (изображено на рис. 2.2.8).
4. 5 и 8 приветствуют друг друга в момент 0 (изображено на рис. 2.2.8).
5. 5 и 10 приветствуют друг друга в момент 0 (изображено на рис. 2.2.8).

6. 8 и 10 приветствуют друг друга в момент 0 (изображено на рис. 2.2.8).
7. 8 и 10 приветствуют друг друга в момент 0.556.
8. 5 и 10 приветствуют друг друга в момент 0.667.
9. 5 и 8 приветствуют друг друга в момент 0.769.
10. 2 и 10 приветствуют друг друга в момент 0.833.
11. 2 и 8 приветствуют друг друга в момент 1.000 (изображено на рис. 2.2.8).
12. 8 и 10 приветствуют друг друга в момент 1.111.
13. 2 и 10 приветствуют друг друга в момент 1.250.
14. 5 и 10 приветствуют друг друга в момент 1.333.
15. 2 и 5 приветствуют друг друга в момент 1.429.
16. 5 и 8 приветствуют друг друга в момент 1.538.
17. 2 и 8 приветствуют друг друга в момент 1.667.
18. 2 и 10 приветствуют друг друга в момент 1.667.
19. 8 и 10 приветствуют друг друга в момент 1.667 (в момент 1.667 встретятся одновременно три джентльмена 2, 8 и 10).
20. 2 и 8 приветствуют друг друга в момент 2.000 (изображено на рис. 2.2.8).
21. 5 и 10 приветствуют друг друга в момент 2.000 (до поворота).
22. 5 и 10 приветствуют друг друга в момент 2.000 (после поворота, изображено на рис. 2.2.8).

Комментарий

Задача решается при помощи бинпоиска с квадратичным нахождением ответа в каждой его итерации.

Решение

Ниже представлено решение на языке C++.

C++

```cpp
#include<bits/stdc++.h>
#define int long long
using namespace std;
const double EPS = 1e-7;
double x(double M, int V, int d){
    double dst = V * M;
    int cnt = floor((dst + EPS) / d);
    double pin = dst - cnt * d;
    if(cnt % 2 == 0){
        return pin;
    }
    else{
        return d - pin;
    }
}
int F(double M, vector<int> &v, int d){
    int res = 0;
    for(int i = 0; i < v.size(); i++){
        double dst = v[i] * M;
```

```
            int cnt = floor((dst + EPS) / d);
            res += cnt * i;
            double tx = x(M, v[i], d);
            for(int j = 0; j < i; j++){
                double txj = x(M, v[j], d);
                if(cnt % 2 == 0){
                    res += txj <= tx + EPS;
                }
                else{
                    res += txj >= tx - EPS;
                }
            }
        }
        return res;
}
signed main(){
    int d, n;
    cin >> d >> n;
    vector<int> v(n);
    for(int i = 0; i < n; i++){
        cin >> v[i];
    }
    int k;
    cin >> k;
    double L = 0, R = 1;
    while(F(R, v, d) <= k){
        R *= 2;
    }
    R *= 2;
    while(R - L > 1e-4){
        double M = (R + L) / 2.0;
        if(F(M, v, d) < k){
            L = M;
        }
        else{
            R = M;
        }
    }
    cout.precision(10);
    cout << fixed << L << endl;
}
```

2.3. Предметный тур. Физика

2.3.1. Первая волна. Задачи 8–9 класса

Задачи первой волны предметного тура по по физике за 8–9 класс открыты для решения. Соревнование доступно на платформе Яндекс.Контест: https://contest.yandex.ru/contest/63463/enter/.

Задача 2.3.1.1. Калориметр (10 баллов)

Условие

Внутренний стакан калориметра представляет собой цилиндр с радиусом $R = 8$ см и высотой $3R$. Внешний стакан также имеет форму цилиндра, стенки которого (как боковые, так и торцы) отстоят от стенок внутреннего на расстояние R. Какая масса теплоизоляционного материала с плотностью $\rho = 25$ кг/м^3 необходима, чтобы полностью заполнить пространство между стаканами?

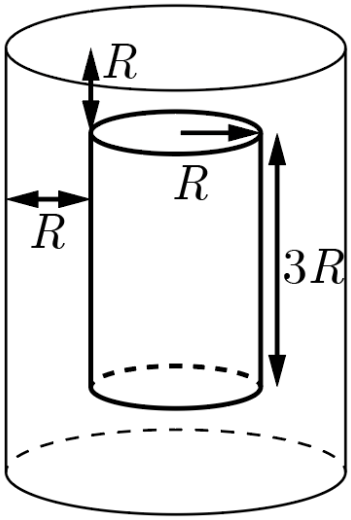


Рис. 2.3.1

Решение

Чтобы найти массу m теплоизоляционного материала, необходимо его плотность умножить на занимаемый им объем V:

$$m = \rho V. \tag{2.3.1}$$

Объем области, заполняемой теплоизоляцией, удобнее всего найти, вычтя из объема V_1 большого внешнего стакана объем V_2 маленького внутреннего. Для любого кругового цилиндра с высотой h и радиусом r объем может быть найден по

формуле $V = \pi r^2 h$. В случае большого цилиндра $r = 2R$ и $h = 5R$, следовательно,

$$V_1 = 5R \cdot \pi(2R)^2 = 20\pi R^3. \tag{2.3.2}$$

Аналогично, для маленького $r = R$, $h = 3R$ и, следовательно,

$$V_2 = 3\pi R^3. \tag{2.3.3}$$

Подставляя (2.3.2), (2.3.3) в (2.3.1), получим, что искомая масса составляет:

$$m = \rho(V_1 - V_2) = 17\pi R^3 \rho \approx 0{,}68\,\text{кг}. \tag{2.3.4}$$

Погрешность 0,01 кг.

Ответ: $m = 17\pi R^3 \rho = (0{,}68 \pm 0{,}01)$ кг.

Задача 2.3.1.2. Нить накала (15 баллов)

Условие

Нити накала ламп изготавливают из вольфрама, удельное сопротивление которого сильно зависит от температуры. По мере прогрева нити оно возрастает от $\rho_0 = 5{,}5 \cdot 10^{-8}$ Ом · м до $\rho_1 = 1{,}1 \cdot 10^{-6}$ Ом · м. Определите электрическую мощность, потребляемую лампой в первый момент после ее включения, если в рабочем режиме (полностью прогревшись) лампа потребляет от той же сети мощность $P_1 = 30$ Вт.

Решение

Электрическая мощность P, потребляемая нитью накала, может быть вычислена по закону Джоуля – Ленца, который для фиксированного напряжения U в сети удобно записать как

$$P = U^2/R. \tag{2.3.5}$$

При этом сопротивление R нити легко выразить через ее удельное сопротивление ρ, длину l и площадь поперечного сечения S

$$R = \frac{\rho l}{S}. \tag{2.3.6}$$

Подставляя (2.3.6) в (2.3.5), получим

$$P = \frac{U^2 S}{\rho l},$$

где только ρ зависит от температуры. В результате придем к выводу, что выделяющаяся в лампе мощность обратно пропорциональна ее удельному сопротивлению, откуда окончательно следует

$$P_0 = P_1 \frac{\rho_1}{\rho_0} \approx 600\,\text{Вт}.$$

Погрешность 1 Вт.

Ответ: $P_0 = (600 \pm 1)$ Вт.

Задача 2.3.1.3. Свая (20 баллов)

Условие

Бетонная свая высотой $h = 1{,}4$ м и массой $m = 160$ кг полностью погружена в грунт так, что ее верхний торец совпадает с уровнем почвы. К сожалению, сваю понадобилось извлечь. Определите, какую работу для этого необходимо совершить, если сила трения со стороны грунта, действующая на сваю, прямо пропорциональна площади соприкосновения ее боковой стороны с землей и в начальный момент ее извлечения равна $F = 4$ кН. Ускорение свободного падения $g \approx 9{,}8$ м/с^2.

Решение

Работа A, необходимая для извлечения сваи, складывается из увеличения потенциальной энергии сваи на величину mgh и работы по преодолению силы трения $A_{\text{тр}}$. Последняя должна быть найдена с учетом постепенного уменьшения силы трения $F_{\text{тр}}$ от максимального значения F до нуля. Поскольку это уменьшение происходит линейно, общая работа оказывается строго вдвое меньше, чем при постоянном значении $F_{\text{тр}} = F$ (аналогично тому, как вычисляется значение работы сил упругости пружины). В результате

$$A = mgh + \frac{Fh}{2} \approx 5\,\text{кДж}. \tag{2.3.7}$$

Погрешность $0{,}1$ кДж.

Ответ: $(5{,}0 \pm 0{,}1)$ кДж.

Задача 2.3.1.4. Двое из ларца (25 баллов)

Условие

Два дрона одновременно вылетают с общей пусковой станции и движутся по прямолинейным траекториям. Первый дрон на начальном этапе движения перемещается с постоянной скоростью $v_1 = 15$ м/с, а через время $\tau = 40$ с быстро переключается на движение с постоянной скоростью $v_2 = 20$ м/с. Второй дрон — наоборот, сначала движется со скоростью v_2, а через время τ переключается на скорость v_1. Наконец, дроны одновременно заканчивают полет. Определите, как долго длился этот полет, если по его итогам средняя путевая скорость первого дрона оказалась на $\Delta v = 1$ м/с выше, чем средняя путевая скорость второго.

Решение

По определению средняя путевая скорость — это отношение общего пройденного пути S к общему времени движения t:

$$v = \frac{S}{t}. \tag{2.3.8}$$

Для первого дрона это уравнение принимает вид

$$v_a = \frac{v_1\tau + v_2(t-\tau)}{t}, \tag{2.3.9}$$

а для второго, соответственно,

$$v_b = \frac{v_2\tau + v_1(t-\tau)}{t}. \tag{2.3.10}$$

Из условий задачи известно, что $v_a - v_b = \Delta v$. Подставляя в это уравнение формулы (2.3.9) и (2.3.10), а также домножая его на t, получим

$$v_1\tau + v_2(t-\tau) - v_2\tau - v_1(t-\tau) = \Delta v t. \tag{2.3.11}$$

Перегруппировав слагаемые, получим

$$v_2 t - 2v_2\tau - v_1 t + 2v_1\tau = \Delta v t, \tag{2.3.12}$$

откуда окончательно

$$t = \frac{2(v_2 - v_1)\tau}{v_2 - v_1 - \Delta v} = 100\,\text{с}. \tag{2.3.13}$$

Погрешность 1 с.

Ответ: (100 ± 1) с.

Задача 2.3.1.5. Призма (30 баллов)

Условие

Для тонкого контроля параметров призмы используется следующая установка: отмечается точка, в которую падает лазерный луч, направленный на экран строго под прямым углом (пунктирный на рисунке). Затем на пути луча устанавливается исследуемая призма так, что задняя (первая по ходу распространения луча) ее грань оказывается строго перпендикулярна лучу, и измеряется расстояние d, на которое в результате этого смещается пятно лазера.

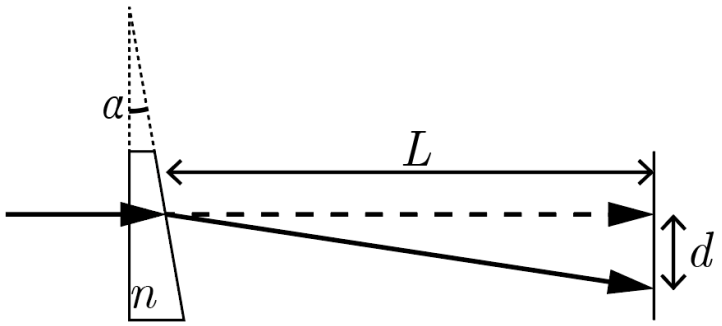


Рис. 2.3.2

Определите показатель преломления стекла, из которого изготовлена призма, если расстояние от передней грани призмы до экрана $L = 3$ м, смещение пятна при установке призмы $d = 12$ см, а угол между передней и задней поверхностями призмы $\alpha = 3°$. Используйте приближение малых углов.

Решение

На первой по ходу распространения луча грани призмы свет не преломляется, поскольку падает на нее под прямым углом. Следовательно, угол падения луча на переднюю грань призмы равен α. Тогда по закону Снеллиуса

$$n = \frac{\sin\beta}{\sin\alpha}, \tag{2.3.14}$$

где β — угол преломления луча, что с учетом приближения малых углов $\sin\alpha \approx \alpha \approx \approx \operatorname{tg}\alpha$ (в радианах) принимает форму

$$n \approx \frac{\beta}{\alpha}. \tag{2.3.15}$$

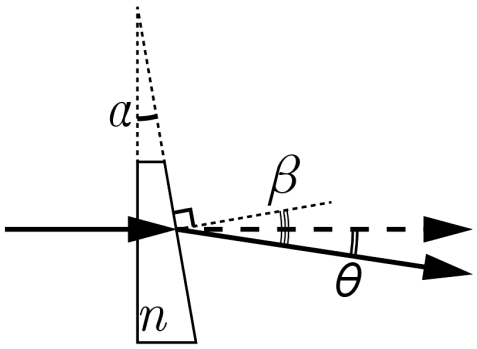


Рис. 2.3.3

Из геометрии рисунка легко видеть, что $\beta = \alpha + \theta$, где θ — угол, который преломленный луч составляет с направлением своего распространения до установки призмы. При этом $\operatorname{tg}\theta = \frac{d}{L}$, откуда

$$n\alpha \approx \beta = \alpha + \operatorname{arctg}\frac{d}{L} \Rightarrow n \approx 1 + \frac{d}{\alpha L} \approx 1{,}76. \tag{2.3.16}$$

Погрешность 0,02.

Ответ: $1{,}76 \pm 0{,}02$.

2.3.2. Первая волна. Задачи 10–11 класса

Задачи первой волны предметного тура по физике за 10–11 класс открыты для решения. Соревнование доступно на платформе Яндекс.Контест: https://contest.yandex.ru/contest/63480/enter/.

Задача 2.3.2.1. Беспилотник (10 баллов)

Условие

Беспилотник, двигаясь равномерно и прямолинейно и обладая при этом импульсом $p_0 = 10^4$ кг $\cdot$ м/с, преодолевает дистанцию $L = 20$ км ровно за 1,5 мин. За какое

время преодолеет ее этот же беспилотник, двигаясь также равномерно и прямолинейно, но обладая на $\Delta p = 10^3$ кг $\cdot$ м/с меньшим импульсом?

Решение

Импульс p тела — это произведение его массы на его скорость, поэтому скорость беспилотника легко может быть вычислена как

$$p_0 = \frac{mL}{t_0}, \tag{2.3.17}$$

где t_0 — время в пути с импульсом p_0. Искомое время t можно аналогично выразить через скорость v беспилотника во второй рассмотренной ситуации как

$$t = \frac{L}{v} = \frac{Lm}{p_0 - \Delta p}. \tag{2.3.18}$$

Выражая массу беспилотника из 2.3.17 и подставляя ее в 2.3.18, получим:

$$t = \frac{\cancel{L} p_0 t_0}{\cancel{L}(p_0 - \Delta p)} = 100\,\text{с}. \tag{2.3.19}$$

Погрешность 1 с.

Ответ: (100 ± 1) с.

Задача 2.3.2.2. Грузовик (15 баллов)

Условие

На плоское горизонтальное дно кузова транспортного грузовика погрузили большой грузовой контейнер и забыли его закрепить. Благодаря силе трения контейнер оставался в покое относительно грузовика до тех пор, пока ускорение последнего не превосходило $a_0 = 2{,}0\,\text{м/с}^2$, и начинал скользить при превышении этого значения. Совершая маневр, грузовик приобрел ускорение $a = 2{,}12\,\text{м/с}^2$, направленное по ходу движения. Какое время длился маневр, если в результате контейнер сдвинулся на $l = 1{,}5\,\text{м}$ относительно грузовика?

Решение

Исходя из того, что контейнер остается на месте при ускорении грузовика до a_0, можно, по второму закону Ньютона, заключить, что сила трения покоя, обеспечивающая это ускорение для контейнера, не превышает значения

$$F = ma_0, \tag{2.3.20}$$

где m — масса контейнера. При любом маневре, при котором контейнер начинает скользить, на него действует сила трения скольжения, равная F и, следовательно, его ускорение относительно дороги оказывается равно a_0.

Во время маневра ускорение контейнера относительно грузовика равно (по модулю) $a - a_0$, а пройденное контейнером относительно грузовика расстояние может быть выражено по законам кинематики как

$$l = \frac{(a - a_0)t^2}{2}, \tag{2.3.21}$$

где t — искомое в задаче время. Преобразуя эту формулу, получим

$$t = \sqrt{\frac{2l}{a - a_0}} = 5\,\text{с}. \tag{2.3.22}$$

Погрешность $0{,}1$ с.

Ответ: $(5{,}0 \pm 0{,}1)$ с.

Задача 2.3.2.3. Методичка (20 баллов)

Условие

На лабораторной работе по физике в распоряжении школьников оказались резисторы двух номиналов: с сопротивлениями x и y кОм, а также конденсаторы двух номиналов: с емкостями x и y мкФ, при этом $x > y$. В старой методичке, посвященной этой лабораторной работе, была изображена схема, приведенная на рисунке, чернила на которой сильно затерлись. В результате Витя решил, что изображенные элементы являются резисторами, и, соединив их согласно схеме, получил элемент с эквивалентным сопротивлением $R = 5$ кОм. Таня же решила, что это конденсаторы и, соединив их, получила элемент с эквивалентной емкостью $C = 2$ мкФ. Определите, чему равнялось число y.

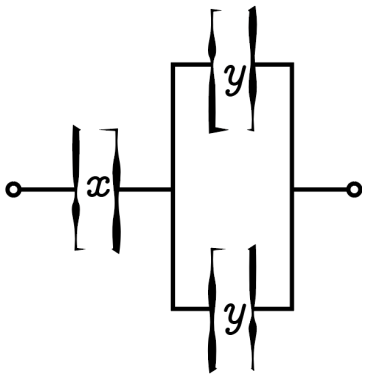


Рис. 2.3.4

Решение

При последовательном соединении резисторов их сопротивления складываются, а при параллельном — складываются обратные сопротивлениям величины. Поэтому сопротивление R схемы Вити через числа x и y в килоомах (кОм) может быть выражено по формуле

$$R = x + \frac{y}{2}. \tag{2.3.23}$$

Для конденсаторов правила поиска эквивалентной емкости при их последовательном и параллельном соединениях в точности обратные, поэтому емкость схемы Тани может быть выражена в микрофарадах (мкФ) по формуле

$$C = \frac{2xy}{x + 2y}. \tag{2.3.24}$$

Выразим x из уравнения (2.3.23) и подставим в (2.3.24):

$$C = \frac{2(R - y/2)y}{R + 3y/2}. \tag{2.3.25}$$

Домножив полученное уравнение на знаменатель дроби и раскрыв скобки, получим

$$RC + \frac{3Cy}{2} = 2Ry - y^2. \tag{2.3.26}$$

Поскольку величины x и y имеют разную размерность в разных частях задачи, имеет смысл сразу перейти к численным значениям

$$10 + 3y = 10y - y^2. \tag{2.3.27}$$

Это квадратное уравнение, которое легко привести к канонической форме

$$y^2 - 7y + 10 = 0 \tag{2.3.28}$$

и решить с помощью дискриминанта

$$y_{1,2} = \frac{7 \pm \sqrt{49 - 40}}{2} = \frac{7 \pm 3}{2}. \tag{2.3.29}$$

Снова воспользовавшись уравнением (2.3.23), легко определить, что при $y = 5$ (корень квадратного уравнения с плюсом) $x = 2{,}5$, что не удовлетворяет условию $x > y$. В то же время, при $y = 2$ (корень с минусом) $x = 4$, что удовлетворяет этому условию. Стало быть, верный корень — $y = 2$.

Погрешность $0{,}1$.

Ответ: $2{,}0 \pm 0{,}1$.

Задача 2.3.2.4. Морозилка (25 баллов)

Условие

Морозильная установка работает по циклу Карно, обходимому в обратном направлении. Какую работу должна потребить такая установка, чтобы заморозить $m = 0{,}4$ кг воды, взятой при ее температуре замерзания, передав полученную от нее теплоту в помещение, температура θ которого равна 30 °C? Удельная теплота плавления и кристаллизации воды $\lambda = 333$ кДж/кг, абсолютный ноль температур $T_0 = -273$ °C.

Решение

Идеальный тепловой двигатель (машина Карно) работает, как известно, по циклу, состоящему из двух изотерм и двух адиабат, и имеет КПД

$$\eta = \frac{T_2 - T_1}{T_2}, \tag{2.3.30}$$

где T_1 — минимальная, а T_2 — максимальная температуры в цикле. По определению КПД тепловой машины он равен отношению работы A, совершаемой газом за цикл, к теплоте Q_2, получаемой им от нагревателя

$$\frac{T_2 - T_1}{T_2} = \eta = \frac{A}{Q_2}. \tag{2.3.31}$$

Отсюда Q_2 может быть выражено как

$$Q_2 = \frac{AT_2}{T_2 - T_1}. \tag{2.3.32}$$

Поскольку за полный цикл внутренняя энергия не изменяется, количество теплоты Q_1, которую газ отдает холодильнику такой машины, равна разнице

$$Q_1 = Q_2 - A = \frac{AT_1}{T_2 - T_1}. \tag{2.3.33}$$

В рассматриваемой задаче тепловой двигатель заменен холодильной машиной, для чего его цикл необходимо обходить в обратном направлении. При этом тепловой резервуар с температурой T_2 начинает получать тепло, а с температурой T_1 — отдавать, но по модулю количества теплоты, которыми газ обменивается с этими тепловыми резервуарами, не изменяются. Остается заметить, что в описанной в условиях холодильной машине $T_1 = 0\,^\circ\mathrm{C} = 273\,\mathrm{K}$, $T_2 = \theta$, $Q_1 = \lambda m$, поскольку забираемая у теплового резервуара с меньшей температурой теплота идет на замораживание в нем воды. Подставив эти данные в (2.3.33), получим

$$\lambda m = \frac{AT_1}{\theta - T_1}, \tag{2.3.34}$$

откуда окончательно выразим ответ

$$A = \frac{\lambda m(\theta - T_1)}{T_1} \approx 14{,}6\,\text{Дж}. \tag{2.3.35}$$

Погрешность: $0{,}5$ Дж.

Ответ: $(14{,}6 \pm 0{,}5)$ Дж.

Задача 2.3.2.5. Электромагнит (30 баллов)

Условие

Для большого промышленного электромагнита критически важной стала проблема охлаждения. Было установлено, что при пропускании через электромагнит

тока $I_1 = 10$ А он нагревается до температуры $t_1 = 70\,^\circ$C, после чего перестает увеличивать свою температуру, а при пропускании через него тока $I_2 = 20$ А — до температуры $t_2 = 205\,^\circ$C.

Определите температуру θ в помещении цеха, в котором используется электромагнит, если известно, что основным механизмом, отвечающим за охлаждение магнита, выступает теплопроводность, мощность которой прямо пропорциональна разнице температур между телами, обменивающимися теплом.

Решение

Как указано в условиях, мощность теплопроводности прямо пропорциональна разнице температур между магнитом и окружающим его воздухом в помещении цеха. Обозначим коэффициент этой пропорциональности κ

$$\begin{cases} P_1 = \kappa(\theta - t_1), \\ P_2 = \kappa(\theta - t_2). \end{cases} \tag{2.3.36}$$

Повышение температуры останавливается, когда мощность производимого катушкой тепла и мощность тепла, уходящего от катушки, благодаря теплообмену, оказываются равны. Первую можно выразить из закона Джоуля – Ленца

$$\begin{cases} P_1 = I_1^2 R, \\ P_2 = I_2^2 R, \end{cases} \tag{2.3.37}$$

где R — сопротивление катушки.

Разделим друг на друга уравнения системы (2.3.36) и уравнения системы (2.3.37), а затем приравняем эти отношения:

$$\frac{P_1}{P_2} = \frac{\theta - t_1}{\theta - t_2} = \frac{I_1^2}{I_2^2}. \tag{2.3.38}$$

Тривиальными алгебраическими преобразованиями выразим θ

$$(\theta - t_1)I_2^2 = (\theta - t_2)I_1^2 \Rightarrow \theta = \frac{t_1 I_2^2 - t_2 I_1^2}{I_2^2 - I_1^2} = 25\,^\circ\text{C}. \tag{2.3.39}$$

Погрешность: $0{,}1\,^\circ$C.

Ответ: $(25{,}0 \pm 0{,}1)$.

2.3.3. Вторая волна. Задачи 8–9 класса

Задачи второй волны предметного тура по физике за 8–9 класс открыты для решения. Соревнование доступно на платформе Яндекс.Контест: https://contest.yandex.ru/contest/63464/enter/.

Задача 2.3.3.1. Аккумулятор тепла (10 баллов)

Условие

Для печи отопления требуется разработать аккумулятор тепла, представляющий собой емкость фиксированного объема, заполненную тем или иным минералом. Используя таблицу 2.3.1 плотностей ρ и удельных теплоемкостей $c_{\text{уд}}$ различных подходящих для этого пород, расположите их в порядке увеличения количества теплоты, которое может быть запасено в таком аккумуляторе при его нагреве до одной и той же температуры θ. Считайте, что θ заведомо меньше температур, при которых любой из этих минералов начнет плавиться или химически разрушаться, а тепловое расширение этих минералов при нагреве до θ пренебрежимо мало.

Таблица 2.3.1. Плотности и удельные теплоемкости

	Минерал	ρ, г/см^3	$c_{\text{уд}}$, кДж/(кг · °C)
A	Кварц	2,6	0,75
B	Базальт	2,8	0,85
C	Талькохлорит	2,75	0,98
D	Нефрит	3	1,1
E	Порфирит	1,45	0,83

Введите в поле ответа последовательность букв, соответствующих выбранным минералам, без пробелов, от наименьшего к наибольшему количеству запасаемой теплоты.

Решение

Количество тепла Q, которое может быть запасено в тепловом аккумуляторе фиксированного объема V, удобно выразить через массу m материала этого аккумулятора

$$Q = c_{\text{уд}} m(\theta - t_0), \tag{2.3.40}$$

где t_0 — начальная температура теплоаккумулятора. В свою очередь, масса m элементарно выражается через плотность вещества и объем V

$$m = \rho V, \tag{2.3.41}$$

откуда

$$Q = c_{\text{уд}} \rho V(\theta - t_0). \tag{2.3.42}$$

Поскольку величины V, θ, t_0 независимы от выбранного вещества, задача сводится к расположению в порядке возрастания произведений $c_{\text{уд}}\rho$. Найдем эти произведения для всей таблицы 2.3.1.

Таблица 2.3.2

	Минерал	ρ, г/см^3	$c_{\text{уд}}$, кДж/(кг · °C)	$c_{\text{уд}}\rho$, кДж/(м^3 · °C)
A	Кварц	2,6	0,75	1 950
B	Базальт	2,8	0,85	2 380
C	Талькохлорит	2,75	0,98	2 695
D	Нефрит	3	1,1	3 300
E	Порфирит	1,45	0,83	1 204

Ответ: EABCD.

Задача 2.3.3.2. Изображения (15 баллов)

Условие

Тонкая собирающая линза имеет фокусное расстояние $F = 20$ см. Вдоль ее оптической оси перед линзой расположено плоское зеркало, на расстоянии $h = 2$ см от которого и $d = 60$ см от линзы размещен светодиод S (рис. 2.3.5). Найдите расстояние между двумя действительными изображениями светодиода, формируемыми этой оптической системой.

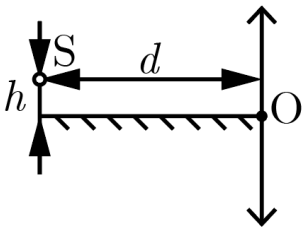


Рис. 2.3.5

Решение

Зеркало формирует мнимое изображение S_1 источника, расположенное в противоположном от него полупространстве на таком же расстоянии от зеркала, как и сам источник. В силу перпендикулярности зеркала и линзы, это мнимое изображение также окажется на расстоянии d от плоскости линзы. Далее линза формирует два действительных изображения: одно непосредственно от источника S (на рис. 2.3.6 оно обозначено S_2) и другое — от его мнимого изображения S_1 (S_3 на рисунке).

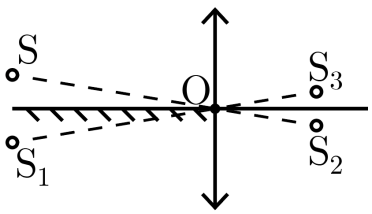


Рис. 2.3.6

Расстояние f, на котором находятся оба действительных изображения от плоскости линзы, легко найти по формуле тонкой линзы

$$\frac{1}{f}+\frac{1}{d}=\frac{1}{F}\Rightarrow f=\frac{Fd}{d-F}. \tag{2.3.43}$$

Искомое расстояние l между изображениями S_2 и S_3, благодаря подобию треугольников $\triangle \mathrm{OSS_1}$ и $\triangle \mathrm{OS_2S_3}$, относится к расстоянию $2h$ между источником и его мнимым изображением S_1 так же, как расстояния от соответствующих изображений и источников до плоскости линзы, являющиеся высотами указанных треугольников

$$\frac{l}{2h}=\frac{f}{d}=\frac{F}{d-F}. \tag{2.3.44}$$

Отсюда окончательно находим

$$l=2h\frac{F}{d-F}=2\,\text{см}. \tag{2.3.45}$$

Погрешность $0{,}1$ см.

Ответ: $l=(2{,}0\pm 0{,}1)$ см.

Задача 2.3.3.3. Пила (30 баллов)

Условие

Циркулярная пила представляет собой пильный диск диаметром $D=19$ см, вращающийся с частотой $4\,500$ об/мин. Определите среднюю силу сопротивления заготовки вращению полотна пилы, если за один пропил, длившийся $t=3$ с, выделилось $Q=5$ кДж тепла, а пила соприкасалась с заготовкой только узкой полоской своей внешней кромки.

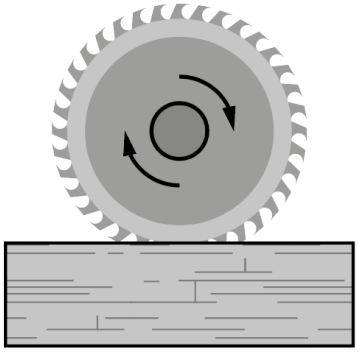


Рис. 2.3.7

Решение

При вращении пилы диссипативные силы (трения различных типов) переводят механическую энергию пильного диска в тепловую. При этом количество выделяющегося тепла равно по модулю работе A этих сил. Последнюю легко найти из ее определения

$$A = FS = Fvt, \tag{2.3.46}$$

где S — путь точек соприкосновения диска с заготовкой, v — скорость этих точек. При вращении диска скорость точек его кромки удобно выразить через период вращения T этого диска

$$v = \frac{\pi D}{T} = \pi D\nu, \tag{2.3.47}$$

где ν — частота вращения диска в оборотах в секунду. Подставляя (2.3.47) в (2.3.46) и учитывая $Q = A$, получим окончательно

$$Q = \pi FD\nu t \Rightarrow F = \frac{Q}{\pi D\nu t} \approx 37\,\text{Н}. \tag{2.3.48}$$

Погрешность 1 Н.

Ответ: $F = (37 \pm 1)\,\text{Н}$.

Задача 2.3.3.4. Питстоп (25 баллов)

Условие

Транспортный робот перемещается из города A в город B, двигаясь практически все время с некоторой постоянной скоростью v. Однако один раз за маршрут ему необходима остановка для заправки и краткого технического обслуживания. Инженеры установили, что при уменьшении длительности этой остановки вдвое скорость движения робота на остальной части маршрута можно будет снизить на $\delta = 10\%$, сохранив при этом его среднюю путевую скорость, что поможет повысить безопасность и экономичность движения. Определите, на сколько процентов (от исходного значения) удалось бы снизить скорость v без изменения средней путевой, если бы от технической остановки удалось полностью отказаться?

Решение

Средняя путевая скорость определяется как отношение всего пути ко всему времени, которое этот путь занимает. Поскольку расстояние между городами неизменно, сохранение средней путевой скорости означает и сохранение общего времени в пути (включая остановку). Следовательно, уменьшение длительности остановки на Δt эквивалентно увеличению времени непосредственного движения на ту же величину. Обозначим общее время робота в пути t, исходную длительность его остановки τ, а исходную скорость движения v_0. Тогда путь робота может быть выражен до и после снижения времени остановки как

$$S = v_0(t - \tau) = v_0(1 - \delta)\left(t - \frac{\tau}{2}\right). \tag{2.3.49}$$

Сократив v_0 и перегруппировав слагаемые, получим

$$\delta t = \tau\left(1 - \frac{1}{2} + \frac{\delta}{2}\right) = \tau\frac{\delta+1}{2}. \tag{2.3.50}$$

Полностью избавившись от остановки, таким образом, робот будет двигаться в течение времени

$$t = \tau\frac{\delta+1}{2\delta}. \tag{2.3.51}$$

Аналогично, время движения $t-\tau$ при наличии остановки удобно записать как

$$t - \tau = \tau\left(\frac{\delta+1}{2\delta} - 1\right) = \tau\frac{1-\delta}{2\delta}. \tag{2.3.52}$$

Обозначив v_1 скорость, которой можно добиться, исключив остановку, запишем через эти выражения путь и приравняем его в случаях с остановкой и без

$$v_1 t = v_0(t-\tau) \Rightarrow v_1\tau\frac{\delta+1}{2\delta} = v_0\tau\frac{1-\delta}{2\delta}. \tag{2.3.53}$$

Отсюда окончательно

$$\frac{v_1}{v_0} = \frac{1-\delta}{1+\delta} \approx 0{,}818. \tag{2.3.54}$$

Итого скорость движения без остановки может составлять приблизительно $81{,}8\%$ от исходной скорости, то есть ниже ее на $18{,}2\%$.

Погрешность $0{,}5\%$.

Ответ: $(18{,}2 \pm 0{,}5)\%$.

Задача 2.3.3.5. Сцепка (30 баллов)

Условие

Два абсолютно одинаковых ползунковых реостата, сопротивления которых могут изменяться в пределах от 0 до $R_0 = 2\,\text{кОм}$, размещены параллельно на печатной плате и соединены как изображено на рис. 2.3.8. Из-за ошибки в процессе пайки изоляция их ползунков слиплась таким образом, что ползунки всегда занимают одно и то же положение на обоих реостатах, но электрический контакт между ними отсутствует (это соединение обозначено на рисунке пунктиром). Найдите разницу между максимальным и минимальным сопротивлениями полученной батареи.

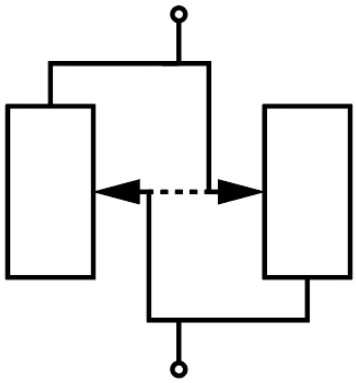


Рис. 2.3.8

Решение

Реостаты на схеме соединены параллельно, поэтому общее сопротивление схемы R может быть выражено через сопротивления $R_{1,2}$ каждого из реостатов по формуле

$$R = \frac{R_1 R_2}{R_1 + R_2}. \tag{2.3.55}$$

Несложно видеть из схемы, что когда ползунок находится в крайнем верхнем положении, левый реостат имеет нулевое сопротивление, а правый — сопротивление R_0 и наоборот. Величина сопротивления находится в линейной зависимости от длины провода. Из этого можно заключить, что при любом положении ползунка

$$R_2 = R_0 - R_1. \tag{2.3.56}$$

Подставляя этот результат в (2.3.55), получим

$$R = \frac{R_1(R_0 - R_1)}{R_0} = R_1 - \frac{R_1^2}{R_0}. \tag{2.3.57}$$

График полученной функции является параболой. Его минимумы и максимумы могут лежать либо на границах диапазона изменения R_1, либо в вершине соответствующей параболы. Поскольку коэффициент перед квадратным слагаемым отрицательный, парабола «повернута» ветвями вниз, то есть на границах диапазона (при $R_1 = 0$ или $R_1 = R_0$) сопротивления батареи минимальны и равны 0, а в ее вершине (которая, как легко видеть из симметрии или непосредственно по формуле $x_{max} = -b/(2a)$, лежит в центре диапазона, при $R_1 = R_2 = \dfrac{R_0}{2}$) сопротивление батареи равно $\dfrac{R_0}{4}$.

Таким образом,

$$R_{max} - R_{min} = \frac{R_0}{4} - 0 = \frac{R_0}{4} = 0{,}5\,\text{кОм}. \tag{2.3.58}$$

Погрешность 0,01 кОм.

Ответ: $l = (0{,}50 \pm 0{,}01)$ кОм.

2.3.4. Вторая волна. Задачи 10–11 класса

Задачи второй волны предметного тура по по физике за 10–11 класс открыты для решения. Соревнование доступно на платформе Яндекс.Контест: https://contest.yandex.ru/contest/63481/enter/.

Задача 2.3.4.1. Зарядка (10 баллов)

Условие

Для увеличения ресурса аккумулятора его зарядка происходит по специальной программе, учитывающей внешние условия, интенсивность использования прибора и другие факторы. Рассчитав оптимальный режим, зарядное устройство в течении $\tau = 10$ мин подавало на аккумулятор ток, линейно возраставший со временем от нуля до максимального значения $I_0 = 3$ А, затем в течение $2{,}5\tau$ поддерживало постоянное значение этого тока и, наконец, на протяжении $\tau/2$, также линейно опускало ток от максимального значения до нуля. Какой общий заряд поступил на положительную клемму аккумулятора за все это время?

Решение

Один из способов решения задачи состоит в обнаружении аналогии между током и движением. Подобно тому, как скорость описывает темп изменения координаты, сила тока описывает темп поступления заряда на аккумулятор. Из кинематики известно, что при равномерном увеличении этого темпа (равноускоренном движении) от нуля, либо при равномерном снижении этого темпа (равнозамедленном движении) до нуля тело проходит вдвое меньшее расстояние, чем при движении с постоянной скоростью, равной максимальной на рассматриваемом участке. Применяя этот результат к току, заметим, что за время $2{,}5\tau$ постоянного тока зарядки на аккумулятор поступил заряд

$$q_1 = 2{,}5\tau I_0, \tag{2.3.59}$$

а за общее время $1{,}5\tau$ увеличения и уменьшения силы тока — заряд

$$q_2 = \frac{1{,}5\tau I_0}{2} = 0{,}75\tau I_0. \tag{2.3.60}$$

Складывая эти заряды, получим окончательно

$$q = 3{,}25\tau I_0 = 5850\,\text{Кл}. \tag{2.3.61}$$

Задача также может быть решена графически, изображением зависимости $I(t)$ и вычислением площади под ней. Фактически такое решение также является применением аналогии, но геометрической, а не кинематической.

Погрешность 50 Кл.

Ответ: (5850 ± 50) Кл.

Задача 2.3.4.2. Принтер (15 баллов)

Условие

Печатающая головка 3D-принтера может перемещаться вдоль направляющей (координата x), которая также может смещаться в перпендикулярном направлении (координата y) под действием двух сервоприводов. Для изготовления детали на принтер была передана программа, согласно которой сервоприводы должны перемещать головку по законам $x(t) = 0{,}2\sin(t/10)$; $y(t) = 0{,}1 + 0{,}2\cos(t/10)$, где t — время, а все величины даны в основных единицах СИ. Найдите величину ускорения печатающей головки при выполнении этой программы.

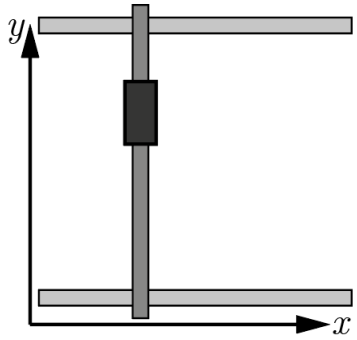


Рис. 2.3.9

Решение

Прежде всего заметим, что уравнения приведенного типа параметрически задают окружность. Это следует из самого определения синуса и косинуса. Радиус R этой окружности равен множителю перед синусом и косинусом (т. к. в математическом определении тригонометрических функций используется единичная окружность), то есть $R = 0{,}2$ м. Здесь было учтено, что основными единицами СИ для измерения длины являются метры.

Поскольку зависимость угла на окружности (аргумента синуса и косинуса) от времени линейна, модуль скорости v печатающей головки постоянен. Чтобы найти его, определим период обращения. Печатающая головка описывает полную окружность, когда аргумент синуса и косинуса меняется на 2π, что происходит при достижении t значения $T = 20\pi$ с. Тогда

$$v = \frac{2\pi R}{T}. \tag{2.3.62}$$

Окончательно заметим, что при неизменной по модулю скорости головки единственное ее ускорение — центростремительное, которое может быть найдено как

$$a = \frac{v^2}{R} = \frac{4\pi^2 R}{T^2} = \frac{4\cancel{\pi^2}\cdot 0{,}2}{400\cancel{\pi^2}} \approx 2\ \text{мм/с}^2. \tag{2.3.63}$$

Погрешность $0{,}1\ \text{мм/с}^2$.

Ответ: $(2{,}0 \pm 0{,}1)\ \text{мм/с}^2$.

Задача 2.3.4.3. Плита (20 баллов)

Условие

Квадратная плита ABCD со стороной $a = 40\,\text{см}$ шарнирно закреплена в одной точке и вращается вокруг оси, перпендикулярной ее плоскости с постоянной угловой скоростью. При этом в некоторый момент времени скорость вершины C этой плиты направлена строго на вершину D, а ускорение вершины A — строго на вершину B (см. рис. 2.3.10). На каком расстоянии от центра пластины находится шарнир?

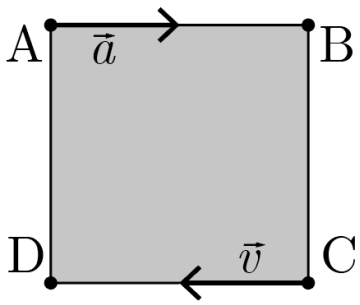


Рис. 2.3.10

Решение

При плоском вращении скорость каждой точки тела направлена перпендикулярно направлению из этой точки на ось вращения, а ускорение (центростремительное) — непосредственно на эту ось. Поэтому, проведя одну прямую через точку C перпендикулярно ее скорости, а другую — через точку A вдоль ее ускорения, можно найти ось вращения как точку пересечения этих прямых. Такой точкой будет, разумеется, вершина B, а расстояние от нее до центра пластины равно

$$l = \frac{\sqrt{2}}{2}a \approx 28{,}3\,\text{см}. \tag{2.3.64}$$

Погрешность $0{,}5\,\text{см}$.

Ответ: $(28{,}3 \pm 0{,}5)\,\text{см}$.

Задача 2.3.4.4. Прямоугольники (25 баллов)

Условие

К проекту модельного теплового двигателя, рабочим телом которого является идеальный одноатомный газ, прилагается pV-диаграмма его рабочего цикла, представляющая собой прямоугольник 1234. К сожалению, автор не указал ни давления, ни объемы характерных точек, а ограничился «площадями» (в энергетических единицах) некоторых прямоугольников на данной диаграмме, которые указаны на рис. 2.3.10. Да к тому же самая важная площадь — площадь внутри цикла 1234, стерлась. Тем не менее определите КПД данного двигателя.

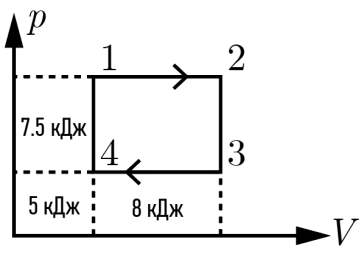


Рис. 2.3.11

Решение

КПД η теплового двигателя определяется как отношение работы A газа за один цикл этого двигателя к количеству теплоты Q, получаемой газом за этот цикл от нагревателя:

$$\eta = \frac{A}{Q}. \tag{2.3.65}$$

Первая — есть площадь прямоугольника 1234. Найти ее просто, если заметить, что поскольку высота (вдоль оси p) у верхних двух прямоугольников одинакова, их площади относятся так же, как и их ширины (вдоль оси V), и то же верно для нижней пары прямоугольников

$$A = 7{,}5\frac{8}{5} = 12\,\text{кДж}. \tag{2.3.66}$$

Чтобы найти теплоту Q, воспользуемся первым началом термодинамики $Q = A + \Delta U$ и заметим, что газ получает теплоту на процессах 41 и 12. Работа за эти два процесса равна полной площади под отрезком 12

$$A_{412} = 12 + 8 = 20\,\text{кДж}, \tag{2.3.67}$$

а внутренняя энергия может быть удобно вычислена по формуле

$$U = \frac{3}{2}PV, \tag{2.3.68}$$

из которой следует, что внутренняя энергия U_4 газа в состоянии 4 равна

$$U_4 = \frac{3}{2}5 = 7{,}5\,\text{кДж}, \tag{2.3.69}$$

поскольку произведение PV для этого состояния есть площадь одного соответствующего прямоугольника. Аналогично, внутренняя энергия U_2 газа в состоянии 2 равна

$$U_2 = \frac{3}{2}(7{,}5 + 12 + 5 + 8) = 48{,}75\,\text{кДж}, \tag{2.3.70}$$

поскольку в этом состоянии соответствующее произведение PV равно общей площади всех прямоугольников на диаграмме.

Подставляя все найденные величины в исходное уравнение (2.3.65), получим окончательно

$$\eta = \frac{A}{A_{412} + U_2 - U_4} = \frac{12}{20 + 48{,}75 - 7{,}5} \approx 19{,}6\%. \tag{2.3.71}$$

Погрешность 1%.

Ответ: $(19{,}6 \pm 1{,}0)\%$

Задача 2.3.4.5. Трюм (30 баллов)

Условие

Трюм грузового судна представляет собой призму с основанием в виде равностороннего треугольника вершиной вниз. Длина внешней стороны треугольника $a = 15\,\text{м}$, толщина его стенок $d = 1{,}5\,\text{м}$, длина киля судна (высота призмы) $l = 40\,\text{м}$. Инженерами было рассчитано, что для сохранения устойчивости судна центр тяжести сыпучего груза, перевозимого в трюме, должен быть хотя бы на $h = 2\,\text{м}$ ниже центра тяжести вытесняемой трюмом воды при его полном погружении. Какой максимальный объем груза можно разместить в трюме, если при насыпании его центр тяжести занимает самое низкое доступное положение?

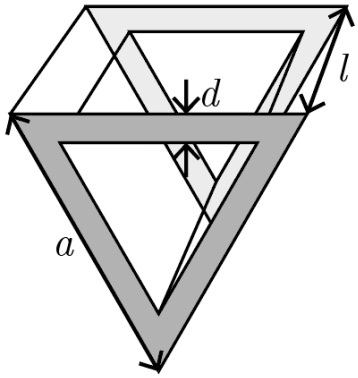


Рис. 2.3.12

Решение

Сыпучий груз занимает объем, форма которого также является призмой с основанием в виде равностороннего треугольника, во всяком случае, при необходимости понизить центр тяжести. В силу симметрии, центр тяжести равностороннего треугольника находится в его геометрическом центре. Поскольку центр тяжести груза должен быть на h ниже, чем центр тяжести вытесненной воды, центр треугольника, формируемого сечением насыпанного груза (темный на рис. 2.3.13), на h ниже центра трюма.

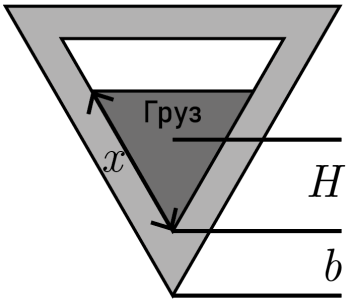


Рис. 2.3.13

Отсчитывая от киля (нижней вершины треугольника) высоту $y_в$, на которой находится центр тяжести вытесненной воды, легко выразить как

$$y_в = \frac{\sqrt{3}}{3}a, \tag{2.3.72}$$

поскольку радиус описанной окружности для равностороннего треугольника в $\sqrt{3}/3$ раз меньше его стороны.

Высота $y_г$ центра тяжести груза может быть найдена как

$$y_г = b + H, \tag{2.3.73}$$

где b — толщина стенки вдоль соответствующего направления (см. рис. 2.3.13), а H — высота центра тяжести груза от нижней внутренней точки трюма. Обе эти величины вычисляются из тригонометрии

$$b = 2d; \quad H = \frac{\sqrt{3}}{3}x, \tag{2.3.74}$$

где x — сторона треугольника, являющегося поперечным сечением груза.

Учитывая $y_г + h = y_в$, получим

$$\frac{\sqrt{3}}{3}x + 2d + h = \frac{\sqrt{3}}{3}a, \tag{2.3.75}$$

откуда

$$x = a - \sqrt{3}(2d + h) \approx 6{,}34\,\text{м}. \tag{2.3.76}$$

Объем, занимаемый грузом при такой длине его стороны, находится как произведение площади равностороннего треугольника на длину киля

$$V = \frac{\sqrt{3}}{4}x^2 l \approx 696\,\text{м}^3. \tag{2.3.77}$$

Погрешность 10 м^3.

Ответ: (696 ± 10) м^3.

2.3.5. Третья волна. Задачи 8–9 класса

Задачи третьей волны предметного тура по физике за 8–9 класс открыты для решения. Соревнование доступно на платформе Яндекс.Контест: https://contest.yandex.ru/contest/63465/enter/.

Задача 2.3.5.1. Башня (10 баллов)

Условие

В гидравлической системе используется башня, заполненная минеральным маслом с плотностью $\rho = 900\,\text{кг/м}^3$. Верхний уровень масла расположен на $h = 60\,\text{м}$ выше, чем смотровое окно в трубе с маслом, закрепленное на ней при помощи $n = 16$ одинаковых болтов. Определите, какую нагрузку должен выдерживать каждый из этих болтов на разрыв, чтобы обеспечить трехкратный запас прочности? Площадь смотрового окна $S = 0{,}1\,\text{м}^2$. Ускорение свободного падения $g = 9{,}8\,\text{м/с}^2$.

Решение

Давление p масла на уровне окна элементарно вычисляется по формуле гидростатического давления

$$p = \rho g h. \tag{2.3.78}$$

Исходя из определения всякого давления p как отношения силы F к площади S, на которую действует эта сила, найдем силу со стороны жидкости, «пытающуюся выдавить» смотровое окно

$$F = pS = \rho g h S. \tag{2.3.79}$$

Искомая рассчетная нагрузка f каждого из болтов может быть получена домножением этой силы на 3 (требуемый запас прочности) и делением на число болтов n, по которым распределяется нагрузка

$$f = \frac{3F}{16} = \frac{3\rho g h S}{16} \approx 9{,}9\,\text{кН}. \tag{2.3.80}$$

Погрешность $0{,}1$ кН.

Ответ: $(9{,}9 \pm 0{,}1)$ кН.

Задача 2.3.5.2. Реостат (15 баллов)

Условие

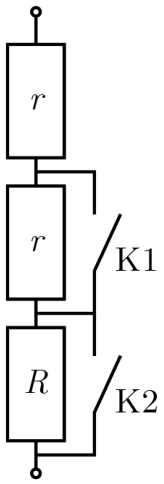


Рис. 2.3.14

Имея в своем распоряжении резисторы только двух различных номиналов, начинающий радиолюбитель изготовил ступенчатый реостат оригинальной конструкции, позволяющий, переключая ключи, получить четыре различных значения сопротивления. Увы, на приложенной к прибору схеме он забыл указать сопротивления отдельных резисторов, указав только, какие из них совпадают. В техническом паспорте устройства остались данные о том, что при замыкании ключа K_1 и размыкании ключа K_2 оно имеет сопротивление $R_1 = 2$ кОм, а напротив, при замыкании ключа K_2 и размыкании ключа K_1 — сопротивление $R_2 = 3$ кОм. Найдите максимальное сопротивление, которое можно получить, используя этот реостат.

Решение

Когда параллельно резистору коротко замыкается цепь, этот резистор фактически перестает работать, поскольку сопротивление провода пренебрежимо мало. Следовательно, замыкание ключа K1 фактически эквивалентно замене среднего резистора на отрезок провода, а ключа K2 — такой же замене нижнего. Учитывая это и применяя формулу эквивалентного сопротивления последовательно соединенных резисторов, легко получим

$$\begin{cases} R_1 = r + R, \\ R_2 = 2r. \end{cases} \tag{2.3.81}$$

Решая эту систему, находим $r = \frac{R_2}{2}$ и $R = R_1 - \frac{R_2}{2}$. Теперь точно так же составим выражения для оставшихся двух конфигураций реостата: R_3 с обоими замкнутыми ключами и R_4 с обоими разомкнутыми

$$\begin{cases} R_3 = r = \dfrac{R_2}{2} = 1{,}5\,\text{кОм}, \\ R_4 = 2r + R = R_1 + \dfrac{R_2}{2} = 3{,}5\,\text{кОм}. \end{cases} \tag{2.3.82}$$

Погрешность $0{,}01$ кОм.

Ответ: $(3{,}50 \pm 0{,}1)$ кОм.

Задача 2.3.5.3. Теплоноситель (20 баллов)

Условие

В некоторых типах ядерных реакторов в качестве теплоносителя используются жидкие металлы. Определите, сколько теплоты за 1 с забирает у реактора жидкий свинец с удельной теплоемкостью $c = 155$ Дж/(кг·°C) и средней плотностью $\rho = 10^4$ кг/м^3, если, двигаясь в трубе диаметром $d = 10$ см со скоростью $v = 20$ м/с, он нагревается от $t_1 = 400$ °C до $t_2 = 900$ °C.

Решение

Обозначим рассматриваемый промежуток времени (1 с) τ. Двигаясь со скоростью v, свинец успевает за это время пройти в трубе дистанцию $l = v\tau$. Учитывая площадь сечения трубы $S = \pi d^2/4$, можно заключить из этого, что за время τ в реактор поступает и из реактора уходит объем

$$V = Sl = \frac{\pi}{4} d^2 v\tau \tag{2.3.83}$$

расплавленного свинца. Количество теплоты Q, которое этот свинец забирает у реактора, задается выражением

$$Q = cm(t_2 - t_1) = c\rho V(t_2 - t_1) = \frac{\pi}{4} c\rho d^2 v\tau (t_2 - t_1) \approx 122\,\text{МДж}, \tag{2.3.84}$$

где m — масса поступившей и ушедшей порции свинца.

Погрешность 2 МДж.

Ответ: (122 ± 2) МДж.

Задача 2.3.5.4. Шкала (25 баллов)

Условие

Шкала вольтметра, используемого в эксперименте, имеет вид, представленный на рис. 2.3.15, и общую длину $l = 15$ см (от минимальной до максимальной отметки). Экспериментатор, глядя на прибор под углом $45°$ к плоскости шкалы, считал показания вольтметра как $U_1 = 1{,}2$ В ровно, однако на самом деле стрелка прибора находилась напротив отметки $U_2 = 0{,}8$ В. Определите, на какое расстояние отстоит стрелка от шкалы, если известно, что глаза экспериментатора находились со шкалой строго на одном уровне высоты, а деления расположены на шкале равномерно.

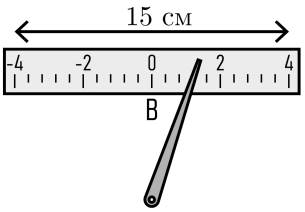


Рис. 2.3.15

Решение

Согласно условиям, глаза экспериментатора находятся на одной высоте со шкалой, поэтому удобно изобразить систему в горизонтальной плоскости (вид сверху), см. рис. 2.3.16. Поскольку угол α, под которым наблюдатель смотрит на стрелку, равен $45°$, искомое расстояние x в точности равно расстоянию y между точкой A действительных показаний прибора и точкой B считанных экспериментатором показаний (треугольник ABC, где C — стрелка — прямоугольный и равнобедренный).

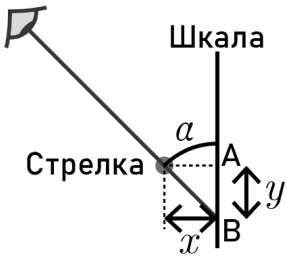


Рис. 2.3.16

Тогда остается вычислить расстояние y между отметками шкалы, соответствующими значениям U_1 и U_2. Поскольку шкала равномерная, это расстояние относится к полной длине шкалы так же, как величина абсолютной ошибки к ее разнице между ее верхним U_{max} и нижним U_{min} пределами измерений

$$\frac{y}{l} = \frac{U_2 - U_1}{U_{max} - U_{min}}. \tag{2.3.85}$$

Отсюда окончательно

$$x = y = l\frac{U_2 - U_1}{U_{max} - U_{min}} = 7{,}5 \text{ мм}. \tag{2.3.86}$$

Погрешность 0,1 мм.

Ответ: $(7{,}5 \pm 0{,}1)$ мм.

Задача 2.3.5.5. Площадка (30 баллов)

Условие

Три робота одновременно стартуют в углу A прямоугольной площадки ABCD. Все они движутся с постоянными по модулю скоростями и все заканчивают движение в точке D одновременно. Но первый робот движется по прямой вдоль стороны AD, второй — по трехзвенной ломаной ABCD, а третий — по двузвенной: сначала вдоль диагонали AC, а затем — по стороне CD. Во сколько раз средняя путевая скорость третьего робота выше, чем первого, если средняя путевая скорость второго робота выше, чем первого, в 2,5 раза? Временем на разгон, остановку и развороты роботов можно пренебречь.

Решение

Обозначив длину стороны AB (и, соответственно, CD) прямоугольника a, а длину стороны BC (и, соответственно, DA) — b, можно легко выразить через эти стороны пути $S_{1,2,3}$ всех трех роботов

$$\begin{cases} S_1 = b, \\ S_2 = 2a + b, \\ S_3 = \sqrt{a^2 + b^2} + a. \end{cases} \tag{2.3.87}$$

Поскольку время движения всех роботов совпадало, отношения их путей точно такие же, как и средних путевых скоростей:

$$\frac{v_2}{v_1} = \frac{S_2}{S_1} = \frac{2a}{b} + 1 = 2{,}5, \tag{2.3.88}$$

откуда легко найти

$$\frac{2a}{b} = 1{,}5 \Rightarrow b = \frac{4}{3}a. \tag{2.3.89}$$

Теперь, пользуясь той же логикой, найдем ответ на вопрос задачи как отношение путей третьего и первого роботов

$$\frac{v_3}{v_1} = \frac{S_3}{S_1} = \frac{\sqrt{a^2 + b^2} + a}{b} = \frac{\sqrt{\dfrac{25a^2}{9}} + a}{\dfrac{4a}{3}} = \frac{\dfrac{8a}{3}}{\dfrac{4a}{3}} = 2. \tag{2.3.90}$$

Погрешность 0,01.

Ответ: $2{,}00 \pm 0{,}01$.

2.3.6. Третья волна. Задачи 10–11 класса

Задачи третьей волны предметного тура по по физике за 10–11 класс открыты для решения. Соревнование доступно на платформе Яндекс.Контест: https://contest.yandex.ru/contest/63482/enter/.

Задача 2.3.6.1. На коленке (10 баллов)

Условие

На конференции, посвященной освоению труднодоступных северных регионов, был представлен проект теплового двигателя, для работы которого используются два тепловых резервуара. Их можно собрать «на коленке»: в качестве холодильника выступает емкость с мокрым снегом, а в качестве нагревателя — котелок с кипящей водой. При этом, по заверениям авторов проекта, КПД двигателя только в $\alpha = 1{,}6$ раз уступает КПД идеальной тепловой машины с такими же холодильником и нагревателем. Найдите этот КПД. Абсолютный ноль температур $T_0 = -273\,^\circ\text{C}$. Двигатель используется при нормальном атмосферном давлении на уровне моря.

Решение

Мокрый снег представляет собой смесь льда и воды, поэтому может существовать только при температуре плавления льда, $t_\text{х} = 0\,^\circ\text{C}$. Аналогично, при атмосферном давлении температура кипящей воды может быть равна только $t_\text{н} = 100\,^\circ\text{C}$. КПД идеальной тепловой машины (машины Карно) с известными абсолютными термодинамическими температурами $T_\text{н}$ и $T_\text{х}$ холодильника задается выражением

$$\eta_0 = \frac{T_\text{н} - T_\text{х}}{T_\text{н}}. \tag{2.3.91}$$

Чтобы дать ответ на вопрос задачи, таким образом, остается разделить этот КПД на α и перевести температуры в шкалу Кельвина

$$\eta = \frac{\eta_0}{\alpha} = \frac{t_\text{н} - t_\text{х}}{\alpha(t_\text{н} - T_0)} \approx 16{,}8\%. \tag{2.3.92}$$

Погрешность: $0{,}2\%$.

Ответ: $(16{,}8 \pm 0{,}2)\%$.

Задача 2.3.6.2. Патруль (15 баллов)

Условие

Корабль береговой охраны движется с постоянной скоростью $v = 12\,\text{м/с}$ относительно поверхности воды. Наблюдательный дрон запрограммирован летать на постоянной высоте по траектории, в системе отсчета корабля представляющей собой окружность с радиусом $R = 2\,\text{км}$ и центром на этом корабле, двигаясь в этой

системе отсчета равномерно и совершая полный оборот за время $T = 10$ мин. Во сколько раз максимальная скорость дрона относительно поверхности воды выше его минимальной скорости относительно нее же?

Решение

Согласно правилу сложения скоростей, скорость $\vec{v}_{\text{дв}}$ дрона относительно воды равна (векторной) сумме его скорости $\vec{v}_{\text{дк}}$ относительно корабля и скорости $\vec{v}_{\text{кв}}$ корабля относительно воды. Поскольку направление вектора $\vec{v}_{\text{кв}}$ неизменно, а вектор $\vec{v}_{\text{дк}}$ в ходе движения дрона принимает все возможные в горизонтальной плоскости направления, обязательно найдутся такие моменты времени, когда эти два вектора сонаправлены и такие, когда они противонаправлены. Эти два случая и будут соответствовать максимальному и минимальному значениям модуля суммы этих векторов, равным $v_{max} = |\vec{v}_{\text{дк}}| + |\vec{v}_{\text{кв}}|$ и $v_{min} = ||\vec{v}_{\text{дк}}| - |\vec{v}_{\text{кв}}||$ соответственно.

Модуль вектора $\vec{v}_{\text{кв}}$ дан напрямую: он равен v. Модуль вектора $\vec{v}_{\text{дк}}$ легко получить, разделив путь дрона в СО корабля на период его обращения в этой СО

$$v_{\text{дк}} = \frac{2\pi R}{T}. \tag{2.3.93}$$

Отсюда получим окончательно

$$\frac{v_{max}}{v_{min}} = \frac{vT + 2\pi R}{|vT - 2\pi R|} \approx 3{,}68. \tag{2.3.94}$$

Погрешность 0,02.

Ответ: $3{,}68 \pm 0{,}02$.

Задача 2.3.6.3. Конденсатор (20 баллов)

Условие

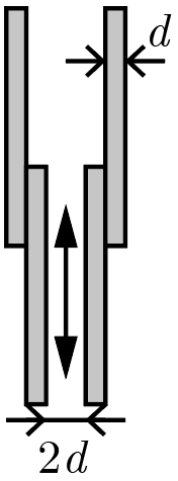


Рис. 2.3.17

Переменный конденсатор состоит из двух пар металлических пластинок толщиной d и площадью $S \gg d^2$, разделенных воздушными зазорами. При этом одна из пар (внутренняя) может частично или полностью входить в зазор другой (внешней), плотно прилегая к ней так, что электрический контакт между соответствующими пластинами никогда не нарушается, но при полном выдвижении площадь этого контакта пренебрежимо мала в сравнении с S.

Определите, во сколько раз максимальная емкость такого конденсатора превосходит минимальную, если зазор между пластинами внутренней пары имеет ширину $2d$.

Решение

Рассматриваемый конденсатор может быть представлен как батарея из двух параллельно соединенных конденсаторов, один из которых (внутренняя пара пластин) всегда имеет зазор $2d$ и площадь обкладок S, а другой (внешняя пара) имеет зазор $4d$ и площадь обкладок, которая может изменяться в пределах от 0 до S. Поскольку эти конденсаторы соединены параллельно, эквивалентная емкость батареи равна сумме их емкостей, а значит, максимальна, когда пары пластин максимально раздвинуты и минимальна, когда они полностью вдвинуты одна в другую. Используя формулу емкости плоского конденсатора, найдем, что емкость внутренней пары пластин (она же минимальная емкость всей батареи) равна

$$C_1 = \frac{S\varepsilon_0}{2d}, \tag{2.3.95}$$

где ε_0 — диэлектрическая постоянная.

Аналогично, емкость полностью выдвинутой внешней пары равна

$$C_2 = \frac{S\varepsilon_0}{4d}, \tag{2.3.96}$$

а максимальная емкость батареи составляет, соответственно, $C_1 + C_2$.

Тогда для искомого отношения максимальной и минимальной емкостей получим:

$$\frac{C_{max}}{C_{min}} = \frac{C_1 + C_2}{C_1} = \frac{\cancel{S\varepsilon_0}/(4\cancel{d}) + \cancel{S\varepsilon_0}/(2\cancel{d})}{\cancel{S\varepsilon_0}/(2\cancel{d})} = \frac{1/4 + 1/2}{1/2} = 1{,}5. \tag{2.3.97}$$

Погрешность 0,01.

Ответ: $1{,}50 \pm 0{,}01$.

Задача 2.3.6.4. Аэростат (25 баллов)

Условие

Горелка теплового аэростата способна поддерживать среднюю температуру воздуха в его оболочке не более, чем на $\Delta t = 70\,^\circ\text{C}$ выше, чем температура окружающего шар воздуха. Аэростат имеет объем $V = 645$ м^3, а общая масса его оболочки, корзины и полезной нагрузки $M = 150$ кг. Определите, при какой максимальной температуре окружающей среды аэростат сможет взлететь?

Абсолютный ноль температур $T_0 = -273\,^\circ\text{C}$, атмосферное давление $p_0 = 100\,\text{кПа}$, молярная масса воздуха $\mu = 29\,\text{г/моль}$, универсальная газовая постоянная $R = 8{,}31\,\text{Дж/(моль}\cdot\text{К)}$.

Решение

Оболочки монгольфьеров (тепловых аэростатов) представляют собой открытые сосуды, поэтому давление внутри и снаружи оболочки должно совпадать (и равняться p_0). Тогда из уравнения Менделеева – Клапейрона

$$p_0 V = \frac{m}{\mu} RT, \tag{2.3.98}$$

где T — абсолютная термодинамическая температура газа, m — его масса.

Легко выразить массу m_1 воздуха внутри оболочки и массу m_2 вытесненного атмосферного воздуха

$$\begin{aligned} m_1 &= \frac{\mu p_0 V}{R(T_0 + \Delta t)}, \\ m_2 &= \frac{\mu p_0 V}{R T_0}, \end{aligned} \tag{2.3.99}$$

где T_0 — искомая температура окружающего воздуха.

Для того чтобы аэростат мог подняться в воздух, необходимо, чтобы вес вытесняемого им воздуха превысил его общий вес (включая вес газа в оболочке)

$$m_2 \cancel{g} = M\cancel{g} + m_1\cancel{g} \Rightarrow \frac{\mu p_0 V}{R T_0} = M + \frac{\mu p_0 V}{R(T_0 + \Delta t)}, \tag{2.3.100}$$

где g — ускорение свободного падения (сразу сокращающееся во всех слагаемых).

Домножим это выражение на $RT_0(T_0 + \Delta T)$ и получим квадратное уравнение относительно T_0

$$\mu p_0 V(\cancel{T_0} + \Delta t) = MRT_0(T_0 + \Delta t) + \cancel{\mu p_0 V T_0}. \tag{2.3.101}$$

В канонической форме:

$$T_0^2 + T_0 \Delta t - \frac{\mu p_0 V \Delta t}{MR} = 0. \tag{2.3.102}$$

Остается найти (методом дискриминанта) единственный положительный корень этого уравнения

$$T_0 = -\frac{\Delta t}{2} + \frac{1}{2}\sqrt{\Delta t^2 + \frac{4\mu p_0 V}{MR}\Delta t} \approx 291\,\text{К} \approx 18\,^\circ\text{C}. \tag{2.3.103}$$

Погрешность $0{,}1\,^\circ\text{C}$.

Ответ: $(18{,}0 \pm 0{,}1)\,^\circ\text{C}$.

Задача 2.3.6.5. Спутник (30 баллов)

Условие

Спутник, движущийся вокруг Земли по высокой круговой орбите, перевели на другую круговую орбиту, в результате чего его кинетическая энергия увеличилась на 5%. На сколько процентов увеличился модуль потенциальной энергии взаимодействия спутника с планетой, если она считается равной нулю на бесконечном удалении от планеты?

Решение

Обозначим радиус орбиты спутника R, его орбитальную скорость v, его массу m, а массу планеты M. На спутник действует сила всемирного тяготения

$$F = G\frac{mM}{R^2}, \tag{2.3.104}$$

где G — гравитационная постоянная, связанная с центростремительным ускорением v^2/R спутника вторым законом Ньютона

$$\cancel{m}\frac{v^2}{\cancel{R}} = G\frac{\cancel{m}M}{R^{\cancel{2}}}. \tag{2.3.105}$$

Из этого выражения легко видеть, что квадрат орбитальной скорости спутника v^2 обратно пропорционален радиусу орбиты. Разумеется, кинетическая энергия спутника $mv^2/2$ прямо пропорциональна этому квадрату скорости и, следовательно, тоже обратно пропорциональна R.

Чтобы понять, как потенциальная энергия спутника зависит от радиуса его орбиты, проще всего обратить внимание на аналогию между гравитацией и электростатическими силами. Сила Кулона взаимодействия двух точечных зарядов зависит от расстояния между ними и обратно пропорциональна квадрату разделяющего их расстояния, точно так же, как сила всемирного тяготения. Одновременно потенциальная энергия взаимодействия этих зарядов обратно пропорциональна первой степени расстояния между ними, следовательно, то же справедливо и для гравитации. В результате видно, что модуль потенциальной энергии обратно пропорционален R, как и кинетическая энергия. Следовательно, при изменении орбиты спутника он изменится ровно во столько же раз.

Погрешность $0{,}01\%$.

Ответ: $(5{,}00 \pm 0{,}01)\%$.

2.3.7. Четвертая волна. Задачи 8–9 класса

Задачи четвертой волны предметного тура по физике за 8–9 класс открыты для решения. Соревнование доступно на платформе Яндекс.Контест: https://contest.yandex.ru/contest/63466/enter/.

Задача 2.3.7.1. Расплав (10 баллов)

Условие

Расплавленная соль предлагается как эффективный аккумулятор тепла для некоторых типов теплоцентралей. Удельная теплота плавления и кристаллизации соли $\lambda = 28{,}7$ кДж/кг, ее теплоемкость в твердой форме $c_1 = 0{,}92$ кДж/(кг°C), а в жидкой — $c_2 = 1{,}5$ кДж/(кг°C), температура ее плавления $\theta = 800$ °C. Определите, какую массу соли необходимо взять, чтобы при ее нагреве от $t_0 = 20$ °C до $t_1 = 1200$ °C запасти $Q = 1$ МДж тепла.

Решение

Количество теплоты, требуемое на нагрев твердой соли до температуры плавления, задается выражением

$$Q_1 = c_1 m(\theta - t_0), \tag{2.3.106}$$

где m — масса нагреваемой соли.

Количество теплоты, уходящее непосредственно на плавление

$$Q_2 = \lambda m. \tag{2.3.107}$$

Наконец, количество теплоты, уходящее на нагрев расплава соли,

$$Q_3 = c_2 m(t_1 - \theta). \tag{2.3.108}$$

Складывая все эти порции тепла, получим, что общая запасаемая теплота

$$Q = Q_1 + Q_2 + Q_3 = m(c_1(\theta - t_0) + \lambda + c_2(t_1 - \theta)), \tag{2.3.109}$$

откуда окончательно

$$m = \frac{Q}{(c_1(\theta - t_0) + \lambda + c_2(t_1 - \theta))} \approx 743\,\text{г}. \tag{2.3.110}$$

Погрешность 1 г.

Ответ: (743 ± 1) г.

Задача 2.3.7.2. Катафот (15 баллов)

Условие

Катафот представляет собой два одинаковых квадратных зеркала, соединенных общей гранью под прямым углом друг к другу. При падении на него видимого света каждое зеркало поглощает $\eta = 20\%$ достигающей его световой энергии, а остальную — отражает. Параллельно биссектрисе образованного зеркалами угла в плоскости, перпендикулярной к их общему ребру, на середину одного из зеркал падает узкий лазерный луч, переносящий мощность $P = 5$ мВт. Какое количество световой энергии поглотит второе зеркало за $\tau = 10$ с?

Решение

Прежде всего отметим, что любой луч, падающий на катафот параллельно его биссектрисе, отразится последовательно от обоих зеркал катафота, как изображено на рис 2.3.18. При этом после первого отражения мощность луча снизится в $(1 - \eta)$ раз, и доля η от этой оставшейся мощности будет поглощена вторым зеркалом. В результате связь между изначальной P и поглощаемой P_1 мощностями имеет следующий вид:

$$P_1 = (1 - \eta)\eta P. \tag{2.3.111}$$

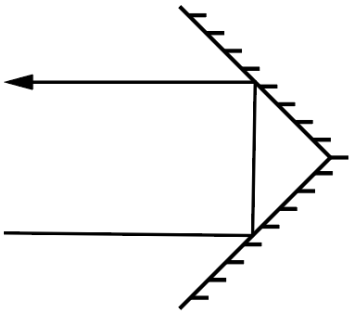


Рис. 2.3.18

Теперь остается лишь вспомнить определение мощности, как отношения энергии (в данном случае переносимой лазерным лучом или поглощаемой зеркалом) ко времени, чтобы получить окончательный ответ

$$Q = P_1\tau = (1 - \eta)\eta P\tau \approx 8\,\text{мДж}. \tag{2.3.112}$$

Погрешность $0{,}1$ мДж.

Ответ: $(8{,}0 \pm 0{,}1)$ мДж.

Задача 2.3.7.3. Соты (20 баллов)

Условие

Композитный материал изготавливают, вырезая из алюминия (плотность $\rho_1 = 2{,}7\,\text{г/см}^3$) строго периодическую вдоль двух взаимно перпендикулярных осей квадратную сетку с толщиной стенки d и длиной внутренней стороны ячейки $4d$, фрагмент которой изображен на рис. 2.3.19. Затем полости заполняют смолой, после затвердевания имеющей плотность $\rho_2 = 1{,}2\,\text{г/см}^3$. Найдите среднюю плотность большого листа из такого материала.

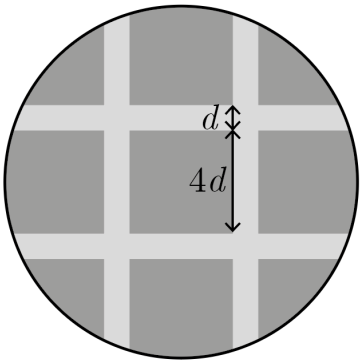


Рис. 2.3.19

Решение

По мере увеличения размеров листа материала роль его краев в общей плотности постепенно снижается, поэтому среднюю плотность большого листа следует вычислять как среднюю плотность одного элемента периодичности, границы которого изображены пунктиром на рис. 2.3.20. Объем V этого элемента равен $25dh$, где h — толщина листа материала.

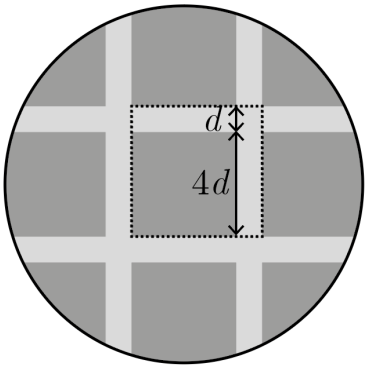


Рис. 2.3.20

Массу элемента найдем, сложив массы алюминия (индексы 1) и смолы (индексы 2)

$$m = m_1 + m_2 = \rho_1 V_1 + \rho_2 V_2 = \rho_1 9dh + \rho_2 16dh. \tag{2.3.113}$$

Тогда искомая плотность окончательно равна

$$\rho = \frac{m}{V} = \frac{\rho_1 9\cancel{dh} + \rho_2 16\cancel{dh}}{25\cancel{dh}} \approx 1{,}74\,\text{г/см}^3. \tag{2.3.114}$$

Погрешность $0{,}01\,\text{г/см}^3$.

Ответ: $(1{,}74 \pm 0{,}01)\,\text{г/см}^3$.

Задача 2.3.7.4. Домкрат (25 баллов)

Условие

На рис. 2.3.21 приведена схема устройства гидравлического домкрата. Его поршни представляют собой цилиндры с радиусами $R = 21\,\text{см}$ и $r = 3\,\text{см}$. Чтобы поднимать при помощи этого домкрата груз $m = 1{,}4\,\text{т}$, установленный на платформе большого цилиндра, к точке A рычага необходимо приложить силу не менее $F = 87{,}5\text{Н}$. Определите отношение $AB : BC$. Ускорение свободного падения $g = 9{,}8\,\text{м/с}^2$. На рисунке точный масштаб не сохранен.

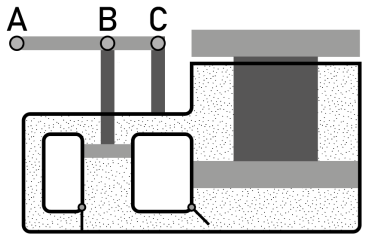


Рис. 2.3.21

Решение

Изображенный домкрат дает выигрыш в силе благодаря двум механизмам: рычагу и гидравлическому прессу. Выигрыш в силе, обеспечиваемый прессом, равен отношению площадей его цилиндров

$$\frac{mg}{F_1} = \frac{\pi R^2}{\pi r^2} \Rightarrow F_1 = mg\frac{r^2}{R^2}, \tag{2.3.115}$$

где F_1 — сила давления малого поршня.

В свою очередь, выигрыш в силе, обеспечиваемый рычагом, равен отношению его плеч, но так как рычаг закреплен в точке C, нужно сравнивать плечи AC и BC

$$\frac{F_1}{F} = \frac{AC}{BC} = \frac{AB + BC}{BC} = 1 + \frac{AB}{BC}. \tag{2.3.116}$$

Совместив эти уравнения, получим окончательно

$$\frac{AB}{BC} = \frac{F_1}{F} - 1 = \frac{mgr^2}{FR^2} - 1 = 2{,}2. \tag{2.3.117}$$

Погрешность $0{,}1$.

Ответ: $2{,}2 \pm 0{,}1$.

Задача 2.3.7.5. Номинальная мощность (30 баллов)

Условие

Изучая электронагреватель прямого действия, ученик заметил, что увеличение подаваемой на него силы тока на $\Delta I = 0{,}1\,\text{А}$ над номинальным значением приводит к увеличению тепловой мощности, выделяемой прибором, на $\Delta P_1 = 44\,\text{Вт}$,

а уменьшение силы тока на ту же величину от номинальной, приводит к уменьшению мощности на $\Delta P_2 = 36\,\text{Вт}$. Найдите номинальную мощность прибора, считая его сопротивление независимым от температуры.

Решение

Согласно закону Джоуля – Ленца, тепловая мощность электронагревателя равна

$$P = I^2 R, \tag{2.3.118}$$

где I — сила пропускаемого через него тока, а R — сопротивление прибора. Последнее по условиям задачи можно считать неизменным, поэтому, введя обозначения I_0 для номинальной силы тока и P_0 для номинальной мощности прибора, можно составить пропорции

$$\begin{cases} \dfrac{P_0 + \Delta P_1}{P_0} = \left(\dfrac{I_0 + \Delta I}{I_0}\right)^2, \\ \dfrac{P_0 - \Delta P_2}{P_0} = \left(\dfrac{I_0 - \Delta I}{I_0}\right)^2. \end{cases} \tag{2.3.119}$$

Обозначив $\frac{\Delta I}{I_0}$ буквой x и частично сократив дроби, приведем их к виду

$$\begin{cases} 1 + \dfrac{\Delta P_1}{P_0} = (1+x)^2, \\ 1 - \dfrac{\Delta P_2}{P_0} = (1-x)^2. \end{cases} \tag{2.3.120}$$

Эту систему можно решить, вычитая второе уравнение из первого

$$\frac{\Delta P_1 + \Delta P_2}{P_0} = 4x \Rightarrow x = \frac{\Delta P_1 + \Delta P_2}{4P_0} \tag{2.3.121}$$

и подставляя результат в любое уравнение системы (2.3.120)

$$\cancel{1} + \frac{\Delta P_1}{P_0} = \cancel{1} + \frac{\Delta P_1 + \Delta P_2}{2P_0} + \frac{(\Delta P_1 + \Delta P_2)^2}{16P_0^2}. \tag{2.3.122}$$

Домножим на $16P_0^2$

$$16\Delta P_1 P_0 = 8P_0(\Delta P_1 + \Delta P_2) + (\Delta P_1 + \Delta P_2)^2. \tag{2.3.123}$$

и выразим окончательно

$$P_0 = \frac{(\Delta P_1 + \Delta P_2)^2}{8(\Delta P_1 - \Delta P_2)} = 100\,\text{Вт}. \tag{2.3.124}$$

Погрешность 1 Вт.

Ответ: 100 ± 1 Вт.

2.3.8. Четвертая волна. Задачи 10–11 класса

Задачи четвертой волны предметного тура по по физике за 10–11 класс открыты для решения. Соревнование доступно на платформе Яндекс.Контест: https://contest.yandex.ru/contest/63483/enter/.

Задача 2.3.8.1. Шестеренки (10 баллов)

Условие

В сложной трансмиссии две шестеренки А и Б вращаются в различных частях механизма так, что угловая скорость вращения шестеренки А в 3 раза выше, чем шестеренки Б, но линейная скорость зубцов шестеренки Б в 2 раза выше, чем шестеренки А. Найдите отношение центростремительного ускорения зубцов шестеренки А к центростремительному ускорению зубцов шестеренки Б.

Решение

Два хорошо известных выражения для центростремительного ускорения a

$$a = \frac{v^2}{R} = \omega^2 R, \tag{2.3.125}$$

где R — радиус траектории, v — линейная скорость, ω — угловая скорость. Перемножив эти выражения, получим

$$a^2 = \frac{v^2}{\cancel{R}}\omega^2\cancel{R} \Rightarrow a = v\omega. \tag{2.3.126}$$

Таким образом, центростремительное ускорение равно произведению линейной скорости на угловую, из чего следует

$$\frac{a_\text{А}}{a_\text{Б}} = \frac{v_\text{А}}{v_\text{Б}} \cdot \frac{\omega_\text{А}}{\omega_\text{Б}} = \frac{1}{2} \cdot \frac{3}{1} = 1{,}5. \tag{2.3.127}$$

Погрешность 0,01.

Ответ: $1{,}50 \pm 0{,}01$.

Задача 2.3.8.2. Маятник (15 баллов)

Условие

Заряженный металлический шарик закреплен на конце тонкой шелковой нити и несет заряд $q = 20$ мкКл. Другой конец нити закреплен к потолку. Определите массу шарика, если при помещении такого маятника в однородное электрическое поле, вектор напряженности которого направлен строго горизонтально и равен по модулю $E = 2{,}5$ кВ/м, сила натяжения нити после установления равновесия оказывается равна $T = 130$ мН. Ускорение свободного падения $g = 9{,}8$ м/с^2.

Решение

На шарик действуют три силы: горизонтально направленная сила электростатического взаимодействия $q\vec{E}$, вертикально вниз направленная сила тяжести $m\vec{g}$ и направленная вдоль нити сила ее натяжения $\vec{T}$. Разумеется, маятник может быть в равновесии, только если векторная сумма этих сил равна нулю, то есть эти три вектора образуют замкнутый треугольник

$$q\vec{E} + m\vec{g} + \vec{T} = \vec{0}. \tag{2.3.128}$$

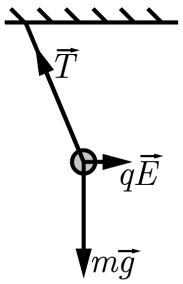


Рис. 2.3.22

Поскольку направления $\vec{E}$ и $\vec{g}$ известны, этот треугольник прямоугольный, а $\vec{T}$ — его гипотенуза. Тогда из теоремы Пифагора

$$q^2E^2 + m^2g^2 = T^2 \Rightarrow m = \frac{\sqrt{T^2 - q^2E^2}}{g} \approx 12{,}2\,\text{г}. \tag{2.3.129}$$

Погрешность $0{,}2$ г.

Ответ: $(12{,}2 \pm 0{,}2)$ г.

Задача 2.3.8.3. Автопилот (20 баллов)

Условие

Автомобиль с автопилотом запрограммирован таким образом, что при движении по прямой на трассе он всегда старается поддерживать дистанцию между собой и движущимся непосредственно перед ним автомобилем ровно такой же, как между собой и движущимся непосредственно за ним автомобилем. В некоторый момент движения скорости всех трех этих автомобилей были равны. Определите ускорение автомобиля, движущегося непосредственно за автопилотируемым, если модули ускорения самого автопилотируемого автомобиля и движущегося непосредственно перед ним в этот момент оба оказались равны $a = 0{,}2\,\text{м/с}^2$, но дистанция между ними при этом начала сокращаться. Колеса автомобилей движутся без проскальзывания, и автопилоту удается соблюдать требования своей программы.

Решение

Программа автомобиля означает, что (до тех пор, пока это позволяет мощность двигателя и сцепление колес) координата x вдоль оси, совпадающей с дорогой, ав-

томобиля с автопилотом равна среднему арифметическому координат x_1 идущего впереди и x_2 идущего позади

$$x = \frac{x_1 + x_2}{2}. \tag{2.3.130}$$

Поскольку такая кинематическая связь справедлива в любые два момента времени t_1 и t_2, для средней скорости v автомобиля на автопилоте в проекции на $0x$ на любом промежутке времени можно записать

$$v_x = \frac{x(t_2) - x(t_1)}{t_2 - t_1} = \frac{x_1(t_2) + x_2(t_2) - x_1(t_1) - x_2(t_1)}{2(t_2 - t_1)} = \frac{v_{x1} + v_{x2}}{2}, \tag{2.3.131}$$

где использована та же система индексов. Таким образом, средняя скорость на любом промежутке времени и, следовательно, мгновенная скорость в любой момент времени автомобиля на автопилоте в проекции на $0x$ равна среднему арифметическому мгновенной скорости в этой же проекции впереди и позади идущих автомобилей. Повторение этих рассуждений приводит к аналогичному результату для ускорений

$$a_x = \frac{a_{x1} + a_{x2}}{2}. \tag{2.3.132}$$

Выразим из этого уравнения проекцию на $0x$ искомого ускорения замыкающего автомобиля a_{x2}

$$a_{x2} = 2a_x - a_{x1}. \tag{2.3.133}$$

По условиям задачи в рассматриваемый момент скорости всех трех автомобилей равны, а ускорения a_1 и a совпадают по модулю, но дистанция начинает сокращаться. Это возможно только если передний автомобиль тормозит — имеет отрицательную проекцию ускорения на направление движения, а автопилотируемый автомобиль, напротив, ускоряется (имеет положительную проекцию). Тогда напрямую из (2.3.133) получим

$$a_{x2} = 2a_x - a_{x1} = 2a - (-a) = 3a = 0{,}6\,\text{м/с}^2. \tag{2.3.134}$$

Погрешность $0{,}01\,\text{м/с}^2$.

Ответ: $(0{,}60 \pm 0{,}01)\,\text{м/с}^2$.

Задача 2.3.8.4. Лифт (25 баллов)

Условие

В лифте, движущемся вверх с некоторым ускорением a, сонаправленным его скорости, уронили без начальной скорости относительно лифта мячик с высоты $h_1 = 0{,}6\,\text{м}$ над уровнем пола лифта. Мячик абсолютно упруго ударился о пол, но своим ударом спровоцировал срабатывание системы аварийной остановки, в результате чего в момент удара ускорение лифта резко поменяло направление на противоположное, а его модуль возрос втрое. После отскока мячик поднялся до высоты $h_2 = 1{,}8\,\text{м}$. Определите a. Ускорение свободного падения $g = 9{,}8\,\text{м/с}^2$.

Решение

В системе отсчета, связанной с лифтом, начальное ускорение мяча равно $g + a$. Проходя с этим ускорением расстояние h_1, мяч приобретает скорость (относительно лифта) v, которую легко вычислить из соотношения

$$\frac{v^2}{2} = (g + a)h_1. \tag{2.3.135}$$

В момент удара резко меняется ускорение, но не скорость лифта, поэтому значение v при абсолютно упругом ударе по модулю остается неизменным.

В процессе подъема мяч уже имеет относительно лифта ускорение $g - 3a$, что позволяет записать аналогично

$$\frac{v^2}{2} = (g - 3a)h_2. \tag{2.3.136}$$

Совмещая эти равенства, получим окончательно

$$(g + a)h_1 = (g - 3a)h_2 \Rightarrow a(h_1 + 3h_2) = g(h_2 - h_1) \Rightarrow a = g\frac{h_2 - h_1}{h_1 + 3h_2} = 1{,}96\,\text{м/с}^2. \tag{2.3.137}$$

Погрешность $0{,}02$ м/с^2.

Ответ: $(1{,}96 \pm 0{,}02)$ м/с^2.

Задача 2.3.8.5. Эффект Лейденфроста (30 баллов)

Условие

Капля воды при температуре немного ниже температуры кипения упала на раскаленную поверхность, в результате чего $\alpha = 10^{-5}$ ее массы практически мгновенно испарилось. Известно, что $\eta = 3\%$ полученной каплей энергии пошло на работу расширяющегося пара над оставшейся частью капли.

На какую высоту «подпрыгнет» капля вертикально вверх в результате такого испарения, если сопротивлением воздуха ее движению, а также потерями тепла в окружающую среду и работой пара против воздуха можно пренебречь?

Удельная теплота парообразования воды равна $L = 2{,}26$ МДж/кг, ускорение свободного падения $g = 9{,}8\,\text{м/с}^2$.

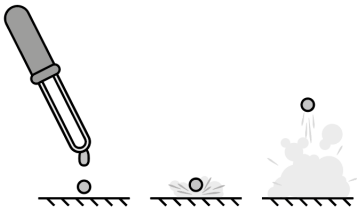


Рис. 2.3.23

Решение

Согласно первому началу термодинамики та теплота, полученная каплей, которая не пошла на работу расширяющегося пара над жидкой частью капли, ушла на изменение ее внутренней энергии; в данном случае — на испарение. Таким образом, можно записать для для этой части энергии

$$\Delta U = (1-\eta)Q = Lm\alpha, \tag{2.3.138}$$

где m — масса капли до испарения, а Q — общая полученная каплей теплота. Отсюда легко найти Q

$$Q = \frac{Lm\alpha}{1-\eta}. \tag{2.3.139}$$

В то же время работа пара над каплей полностью идет на увеличение ее механической энергии, которая в верхней точке траектории чисто потенциальна

$$A = \eta Q = (1-\alpha)mgh. \tag{2.3.140}$$

Выражая из этого уравнения h и подставляя в него Q, получаем

$$h = \frac{\eta Q}{(1-\alpha)mg} = \frac{L\alpha\eta}{(1-\alpha)(1-\eta)g} \approx 7{,}1\,\text{см}. \tag{2.3.141}$$

Разумеется, если величину $1-\alpha$ в этом выражении считать просто единицей, ответ не изменится в пределах любой разумной погрешности.

Погрешность $0{,}5$ см.

Ответ: $(7{,}1 \pm 0{,}5)$ см.

2.4. Инженерный тур

Задачи первого этапа инженерного тура открыты для решения. Соревнование доступно на платформе Яндекс.Контест: https://contest.yandex.ru/contest/66695/enter/.

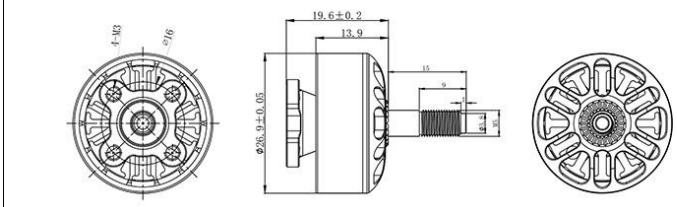
Задача 2.4.1. Подбор аккумулятора (14 баллов)

Темы: квадрокоптер, аккумулятор.

Условие

iFlight XING-E X2207 2450KV

Technical Datas	
KV	2450
Configu-ration	12N14P
Stator Diamter	22mm
Stator Length	7mm
Shaft Diameter	4mm
Motor Dimension(Dia.*Len)	Φ28.5*19.7mm
Weight(g)	34.6
Idle current(10)@10V(A)	<2.0
No.of Cells(Lipo)	2~4S
Max Continuous Power(W)180S	682.1
Internal Resistance	46.8mΩ
Max Current(180S)	42.63A



Prop (inch)	Voltages (V)	Throttle (%)	Load Currency (A)	Pull(g)	Power(W)	Efficiency (g/W)	Temperature(in full throttle load 1 min)
5050	16	50%	7.93	495	126.9	3.901	60C°
		60%	12.33	655	197.3	3.320	
		70%	17.08	821	273.3	3.004	
		80%	24.22	1036	387.5	2.673	
		90%	31.48	1223	503.7	2.428	
		100%	39.56	1498	633.0	2.367	
5.1x3.1x3	16	50%	7.33	483	117.3	4.118	57C°
		60%	10.93	622	174.9	3.557	
		70%	15.35	799	245.6	3.253	
		80%	21.35	1051	341.6	3.077	
		90%	27.75	1189	444.0	2.678	
		100%	35.68	1391	570.9	2.437	
5045	16	50%	6.8	476	108.8	4.375	55C°
		60%	10.27	636	164.3	3.870	
		70%	14.47	792	231.5	3.421	
		80%	19.55	1008	312.8	3.223	
		90%	25.22	1207	403.5	2.991	
		100%	32.02	1465	512.3	2.860	
5043	16	50%	7.67	457	122.7	3.724	58C°
		60%	10.93	619	174.9	3.540	
		70%	15.28	765	244.5	3.129	
		80%	21.35	982	341.6	2.875	
		90%	27.68	1195	442.9	2.698	
		100%	35.48	1398	567.7	2.463	
6045	16	50%	9.33	648	149.3	4.341	63C°
		60%	13.87	798	221.9	3.596	
		70%	19.42	1042	310.7	3.354	
		80%	26.08	1315	417.3	3.151	
		90%	34.15	1496	546.4	2.738	
		100%	42.63	1679	682.1	2.462	
■Airplane			☐Helicopter			■Vtol	

Рис. 2.4.1. Мотор

Необходимо подобрать аккумулятор для квадрокоптера, на котором установлены моторы `IFlight Xing` (рис. 2.4.1) с пятью дюймовыми пропеллерами, шаг винта которых 4,3″ (дюйма).

Время полета данного коптера должно быть не менее 57 с при полном газе.

Выберите правильный вариант ответа, для расчета используете ток нагрузки:

1. 4s 850 mAh 80c,
2. 4s 2200 mAh 75c,
3. 4s 2600 mAh 45c,
4. 4s 2300 mAh 80c,
5. 6s 2400 mAh 120c,
6. 6s 850 mAh 120c.

Решение

Согласно таблице на рис. 2.4.1 максимальный ток нагрузки на один мотор с пропеллером 5043 (размер 5″, шаг винта 4,3″) составляет: 35,48 А.

Для квадрокоптера (четыре мотора): $35{,}48 \cdot 4 = 141{,}92$ А.

Максимальный разрядный ток:

1. для аккумулятора № 1: $0{,}85 \cdot 80 = 68$ А,
2. для аккумулятора № 2: 165А,
3. для аккумулятора № 3: 117А,
4. для аккумулятора № 4: 184А,
5. для аккумулятора № 5: 288А,
6. для аккумулятора № 6: 102А.

Вариант № 5 не является верным, так как данные моторы не предназначены для использования с `6S` аккумулятором.

Время полета с аккумулятором № 2 равно $(2{,}2/141{,}92) \cdot 60 \cdot 60 = 55{,}81$ с.

Время полета с аккумулятором № 4 равно $(2{,}3/141{,}92) \cdot 60 \cdot 60 = 58{,}34$ с.

Ответ: 4.

Задача 2.4.2. Сборка (11 баллов)

Темы: регулятор оборотов, полетный контроллер.

Условие

При сборке коптера стажер подключил сигнальные провода регуляторов оборотов двигателей в пины полетного контроллера COEX PIX по следующей схеме:

1. M1 — пины A,
2. M2 — пины B,
3. M4 — пины C,

4. M3 — пины D.

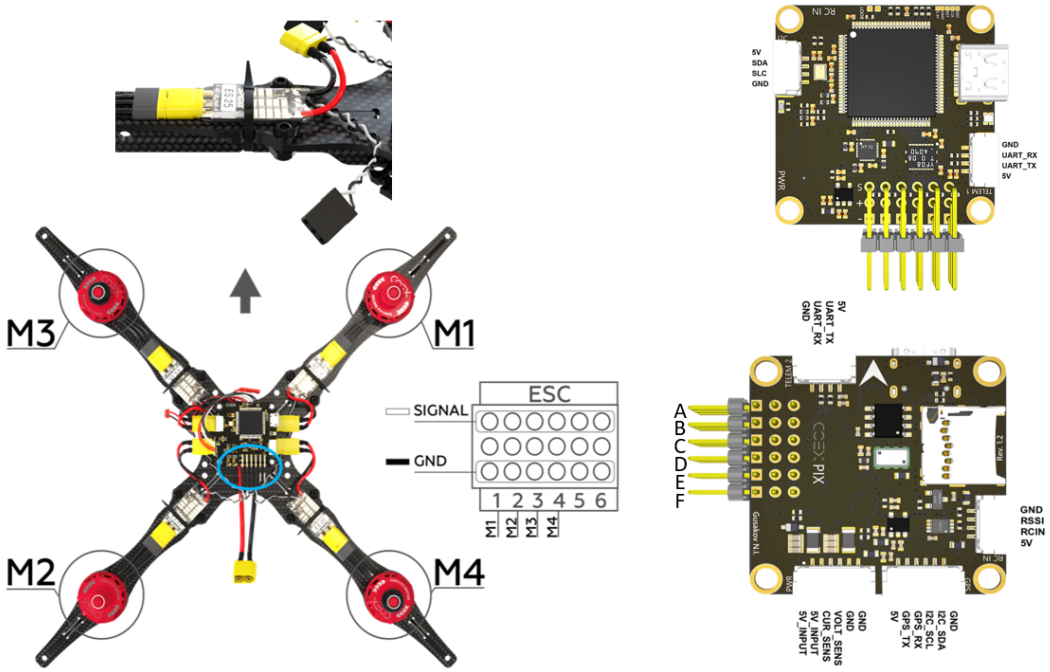


Рис. 2.4.2. Подключение сигнальных проводов

Посмотрите внимательно на рис. 2.4.2 и отметьте, какие сигнальные провода регуляторов оборотов двигателей **НЕВЕРНО** подключены в пины полетного контроллера `COEX PIX`.

Отметьте неверные позиции:

1. M1,
2. M2,
3. M3,
4. M4.

Решение

Подключение сигнальных проводов регуляторов оборотов двигателей происходит по следующей схеме:

1. M1 — пины F,
2. M2 — пины E,
3. M3 — пины D,
4. M4 — пины C.

Ответ: M1, M2.

Задача 2.4.3. Полетный контроллер (10 баллов)

Темы: Pixracer, полетный контроллер.

Условие

На рис. 2.4.3 изображена распиновка полетного контроллера `Pixracer`.

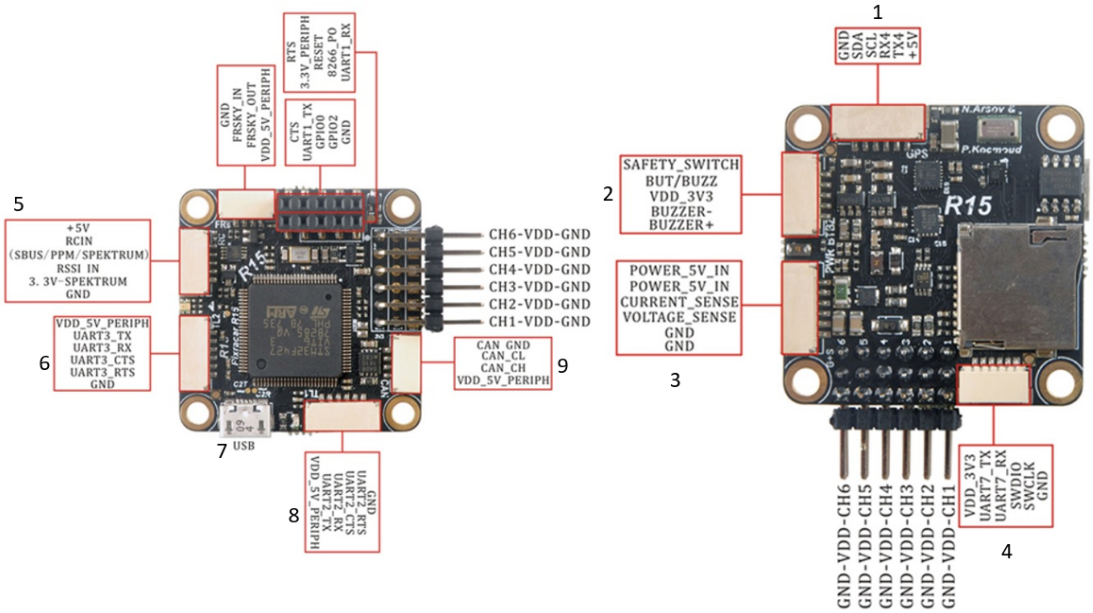


Рис. 2.4.3. Полетный контроллер

Укажите вариант ответа, в котором обозначен разъем для подключения радиоприемника к полетному контроллеру.

Решение

Подключение радиоприемника к полетному контроллеру осуществляется в разъем `RC IN`.

Ответ: 5.

Задача 2.4.4. К полету готовы? (10 баллов)

Темы: настройка, пульт радиоуправления.

Условие

Во время дистанционной работы напарник-стажер сообщил о том, что справился со сборкой и настройкой квадрокоптера и готов к первому полету.

Не допустил ли он ошибку при сборке?

Проверим готовность по фотографии (рис. 2.4.4).

Рис. 2.4.4. Пульт радиоуправления

К полету готов? Отметьте верный вариант.

1. Потеряна связь с радиоприемником.
2. Тримы газа и крена сбиты в крайнее правое и верхнее положение соответственно.
3. Тримы тангажа и рыскания сбиты в крайнее верхнее и правое положение соответственно.
4. Тримы тангажа и рыскания сбиты в крайнее верхнее и левое положение соответственно.
5. Разрядился источник питания пульта радиоуправления.
6. Низкий уровень заряда аккумулятора квадрокоптера.

Решение

Для решения задачи необходимо внимательно рассмотреть рис. 2.4.4. На рисунке можно заметить — что тримы радиоуправления сбиты. Трим рыскания в крайнем правом положении, а трим тангажа — в крайнем верхнем.

Ответ: 3.

Задача 2.4.5. Подбор оптимальной ВМГ (14 баллов)

Темы: сборка дрона, эксплуатация.

Условие

Необходимо собрать беспилотник мультироторного типа из деталей, имеющихся на складе (таблица 2.4.1).

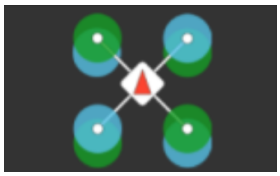
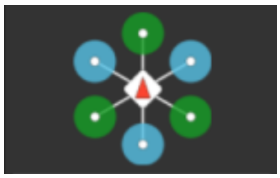
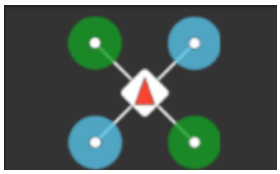
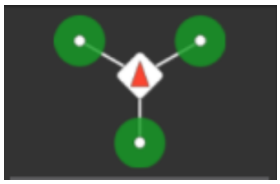
Техническое задание

Существует список компонентов, имеющихся на складе. Используя эти компоненты, необходимо создать особый квадрокоптер для мониторинга линий электропередач, имеющий минимально возможную массу.

Также необходимо иметь возможность установки камеры весом 154 г. Время полета на одном аккумуляторе должно быть максимально возможным.

Временем полета считается зависание беспилотника на месте с полезной нагрузкой. Вес беспилотника принять за вес компонентов в его составе. Учитывать необходимо только компоненты, находящиеся в таблице. Вес рамы не учитывать.

Таблица 2.4.1. Имеющееся на складе оборудование

	Двигатель	ESC	LiPo	Frame
A	XING2 1806 FPV Motor 2500kv Количество: 6 шт.	T-motor CINE55A 8S 8IN1 32BIT Количество: 2 шт.	Betafpv BT2.0 300mAh 1S 30C Battery Количество: 50 шт.	
B	Betafpv 7x16mm Brushed Motors Количество: 7 шт.	T-motor MINI F45A 6S 4IN1 Количество: 2 шт.	Samsung 30Q 18650 3000mah LiIon Количество: 3 шт.	
C	T-motor U15II KV80 Количество: 14 шт.	T-motor AT 115A 14S Количество: 24 шт.	GNB 5000mah 11.1 50C Количество: 5 шт.	
D	T-motor P1604 2850 KV Количество: 3 шт.	BLITZ F411 1S Whoop AIO Количество: 2 шт.	GNB 3500 mAh 6S 70C Lipo Количество: 9 шт.	

В качестве ответа следует выписать буквы, под которыми расположены необходимые компоненты.

Пример ответа: ECAB.

Ответ: ABDC.

Задача 2.4.6. Передача данных с датчиков (27 баллов)

Темы: Python, сервер.

Условие

Одной из ключевых задач при создании робототехнической системы является обеспечение точного и своевременного получения данных от различных сенсоров, которые помогают роботу ориентироваться в окружающей среде, избегать препятствий и принимать решения на основе собранной информации.

Импровизированный робот оснащен несколькими датчиками, результаты работы которых можно считать из файлов в директории `./sensors/`. Каждый файл в этой директории имеет имя `<sensor_name>.value`.

Необходимо разработать модуль, который будет считывать данные с датчиков, выполнять минимальную их проверку и передавать основному контроллеру для дальнейшей обработки.

Проверять необходимо формат данных — они должны быть числом (целым или вещественным) и не выходить за пределы $-1\,000 \leqslant value \leqslant 1\,000$. Основной контроллер принимает данные по протоколу `HTTP`, в формате `json`.

Формат входных данных

Набор файлов в директории `./sensors/`. Каждый файл в этой директории имеет имя `<sensor_name>.value`, где `sensor_name` — имя датчика.

Каждый файл содержит единственную строку с показанием датчика.

Пример 1

Структура файлов:

```
.
├── sensors
│   ├── light_sensor.value
│   └── smoke_sensor.value
└── solution.py
```

Файл `light_sensor.value`:

106.346

Файл `smoke_sensor.value`:

−74.937

Пример 2

Структура файлов:

```
.
├── sensors
│   ├── carbon_monoxide_sensor.value
│   ├── flow_sensor.value
│   └── tilt_sensor.value
└── solution.py
```

Файл `carbon_monoxide_sensor.value`:

−183.213

Файл `flow_sensor.value`:

−1538.902

Файл `tilt_sensor.value`:

94.346

Формат выходных данных

Для каждого датчика необходимо сделать `HTTP POST`-запрос на сервер.

Расположение сервера:

```
Host: 127.0.0.1
Port: 8000
```

Формат POST-запроса:

`URL: /sensor/<sensor_name>`, где `sensor_name` — имя датчика;

`Headers`: обязательно `'Content-Type': 'application/json'`

`Body`:

- в случае успешно пройденной проверки данных `{"status": "ok", "value": <sensor_value>}`, где `sensor_value` — значение датчика;
- в случае НЕ успешно пройденной проверки данных `{"status": "error", "message": "invalid fomat"}`.

Порядок запросов может быть любым.

Пример 1

```
POST http://127.0.0.1:8000/sensor/smoke_sensor
Accept-Encoding: gzip, deflate
User-Agent: python-requests/2.11.1
Accept: /
Connection: keep-alive
Content-Length: 34
Content-Type: application/json
{"status": "ok", "value": -74.937}
POST http://127.0.0.1:8000/sensor/light_sensor
Accept-Encoding: gzip, deflate
User-Agent: python-requests/2.11.1
Accept: /
Connection: keep-alive
Content-Length: 34
Content-Type: application/json
{"status": "ok", "value": 106.346}
```

Пример 2

```
POST http://127.0.0.1:8000/sensor/tilt_sensor
Accept-Encoding: gzip, deflate
Connection: keep-alive
User-Agent: python-requests/2.11.1
Accept: /
Content-Length: 33
Content-Type: application/json
{"status": "ok", "value": 94.346}
```

```
POST http://127.0.0.1:8000/sensor/carbon_monoxide_sensor
Accept-Encoding: gzip, deflate
Connection: keep-alive
User-Agent: python-requests/2.11.1
Accept: /
Content-Length: 35
Content-Type: application/json
{"status": "ok", "value": -183.213}
POST http://127.0.0.1:8000/sensor/flow_sensor
Accept-Encoding: gzip, deflate
Accept: /
Connection: keep-alive
User-Agent: python-requests/2.11.1
Content-Length: 48
Content-Type: application/json
{"status": "error", "message": "invalid format"}
```

Решение

Ниже представлено решение на языке Python.

Python

```python
from pathlib import Path
import requests

p = Path('sensors')

for file in p.glob('*.value'):
    with file.open() as f:
        value = f.readline()
    try:
        value = float(value)
    except ValueError:
        requests.post(f"http://localhost:5000/sensor/{file.stem}",
        ↪  json={"status": "error", "message": "invalid format"})
        continue

    requests.post(f"http://localhost:5000/sensor/{file.stem}",
    ↪  json={"status": "ok", "value": value})
```

Задача 2.4.7. You Only Look Once (14 баллов)

Темы: Python, YOLO.

Условие

В современном мире компьютерного зрения алгоритмы детектирования объектов играют важную роль в различных приложениях: от безопасности до автономного вождения. Одной из наиболее популярных и мощных моделей для детектирования объектов является `YOLO` (You Only Look Once). Эта модель может быстро и точно определять объекты на изображениях и видеопотоках, классифицируя их по заранее изученным классам.

Благодаря библиотеке `Ultralytics`, работа с моделями `YOLO` стала значительно проще и интуитивнее. Эта библиотека предоставляет удобный и мощный интерфейс для загрузки, использования и анализа моделей `YOLO`, что позволяет быстро и эффективно интегрировать их в различные приложения.

Необходимо извлечь из предоставленного `.pt` файла `YOLO` модели, обученной с использованием вышеуказанной библиотеки, список всех классов, которые модель может детектировать. Этот список классов будет использоваться для настройки интерфейса пользователя, отображения результатов детектирования и других задач, связанных с классификацией объектов.

Модель: https://disk.yandex.ru/d/wKOcOdHPgGrccA.

Формат входных данных

Список классов, который модель может детектировать в алфавитном порядке через пробел.

Пример: human cat dog.

Решение

Ниже представлено решение на языке Python.

Python

```python
from ultralytics import YOLO

model = YOLO("model.pt")
print(",".join(sorted(model.names.values())))
```

3. Второй отборочный этап

3.1. Работа наставника НТО на этапе

На втором отборочном этапе НТО участникам предстоит решать как индивидуальные, так и командные задачи в рамках выбранного профиля. Подготовка к этому этапу требует от них не только глубокого понимания предметной области, но и умения работать в команде, эффективно распределять роли и применять полученные знания на практике. Наставник играет здесь важную роль — он помогает участникам выстроить осмысленную и целенаправленную траекторию подготовки.

Вот основные направления, в которых наставник может поддержать участника:

- **Подготовка по образовательным программам НТО.** Наставник может готовить участников, используя готовые образовательные программы по технологическим направлениям, рекомендованные организаторами, а также адаптировать их под уровень подготовки школьников.
- **Разбор заданий прошлых лет.** Изучение задач второго отборочного этапа прошлых лет помогает участникам понять формат заданий, определить типовые ошибки и выработать стратегии решения.
- **Онлайн-курсы.** Участники могут пройти курсы по разбору задач прошлых лет или курсы, рекомендованные разработчиками отдельных профилей. Наставник может включить эти курсы в план подготовки, а также сопровождать процесс изучения и помогать с возникшими вопросами.
- **Анализ материалов профиля.** Совместный разбор методических материалов, размещенных на страницах профилей, помогает уточнить требования к участникам и направить подготовку на ключевые темы.
- **Практикумы.** Это важный элемент подготовки, позволяющий применять знания на практике. Наставник может:
  - ◇ организовать практикумы по методическим материалам с сайта профиля;
  - ◇ декомпозировать задачи заключительного этапа прошлых лет на отдельные элементы и проработать их с участниками;
  - ◇ провести анализ требуемых профессиональных компетенций и спланировать занятия для развития наиболее значимых из них;
  - ◇ направить участников на практикумы и мероприятия от организаторов, которые анонсируются в официальных сообществах НТО, например, в телеграм-канале для наставников: https://t.me/kruzhok_association.
- **Командная работа.** Одной из ключевых задач наставника на втором этапе является помощь в формировании команды или в поиске подходящей. Наставник может помочь участникам определить их сильные стороны, выбрать роль в команде и сориентироваться в процессе командообразования, включая участие в бирже команд в рамках конкретного профиля.

Если участники не прошли отборочный этап

Случается, что несмотря на усилия и серьезную подготовку, участники не проходят во второй или заключительный этап Олимпиады. В такой ситуации особенно важна поддержка наставника.

- **Поддержка и признание усилий.** Наставнику важно подчеркнуть ценность пройденного пути: полученные знания, навыки, преодоленные трудности и личностный рост. Это помогает участникам сохранить мотивацию и не воспринимать результат как окончательное поражение.
- **Рефлексия.** Полезно организовать встречу для обсуждения впечатления от участия, трудности, с которыми столкнулись школьники и то, что они узнали о себе и команде. Наставник может направить разговор в конструктивное русло: какие выводы можно сделать? Что сработало хорошо? Что можно улучшить?
- **Анализ ошибок и пробелов.** Наставник вместе с участниками анализирует, какие темы вызвали наибольшие затруднения, чего не хватило в подготовке — теоретических знаний, практических навыков, командного взаимодействия. Это позволяет выстроить более эффективную стратегию на будущее.
- **Планирование дальнейшего пути.** Участникам можно предложить:
  - ◇ продолжить углубленное изучение профиля или смежных направлений;
  - ◇ заняться проектной деятельностью, которая укрепит знания и навыки;
  - ◇ сформировать план по подготовке к следующему циклу НТО, начиная с работы над типовыми заданиями и курсами.
- **Создание устойчивой мотивации.** Важно показать школьникам, что участие в НТО — это не просто соревнование, а часть большого образовательного маршрута. Даже неудачный результат может стать толчком к профессиональному росту, если воспринимать его как точку развития, а не как конец пути.

Таким образом, наставник помогает участникам не только готовиться к этапам НТО, но и справляться с неудачами, выстраивать долгосрочную стратегию и сохранять интерес к инженерному и технологическому творчеству.

3.2. Инженерный тур

Участникам заключительного этапа предстоит разработать систему автоматизированного мониторинга объектов топливно-энергетического комплекса при помощи квадрокоптера.

Чтобы успешно решить данную задачу необходимо:

- знать и понимать принципы устройства и работы квадрокоптера, принципы базовой и программной настройки квадрокоптера;
- обладать навыками 3D-моделирования, программирования, работы с компьютерным зрением и машинным обучением.

Задания второго отборочного этапа готовят команды к решению задачи заключительного этапа и представляют собой его декомпозированные компетентностные подзадачи. Например: программирование блока управления на Python, программная настройка квадрокоптера, автономный полет квадрокоптера и обработка данных, машинное обучение.

Пакет заданий состоит из индивидуальных задач, проверяющих навыки участников команды согласно их компетенциям, а также для того, чтобы каждый участник мог попробовать себя в другой роли.

Командное задание позволяет участникам апробировать свои решения в симуляторе: инженерам — настроить квадрокоптер и подготовить виртуальный мир, а программистам — написать программный код для автономного полета квадрокоптера. Для успешного выполнения заданий второго отборочного этапа участникам необходимо правильно распределить задачи, а также наладить систему планирования и коммуникации внутри команды.

Компетенции, необходимые участникам для решения задач второго этапа:

- базовые навыки работы с летающими робототехническими системами;
- программирование (Python);
- навыки работы с компьютерным зрением (OpenCV);
- 3D-моделирование;
- Front-end разработка.

3.2.1. Индивидуальные задачи

Индивидуальные задачи второго этапа инженерного тура открыты для решения. Соревнование доступно на платформе Яндекс.Контест: https://contest.yandex.ru/contest/69899/enter/.

Задача 3.2.1.1. Сборка квадрокоптера (8 баллов)

Тема: базовые навыки работы с летающими робототехническими системами.

Условие

Ознакомьтесь с видео и отметьте все электронные компоненты и части коптера, которые были подключены или установлены неправильно (ссылка на видео: https://disk.yandex.ru/i/0AeXVnc6tzUWHQ).

1. Подключение шлейфа радиоприемника к полетному контроллеру.
2. Подключение шлейфа радиоприемника к радиоприемнику.
3. Подключение питания к полетному контроллеру.
4. Подключение сигнального провода регуляторов оборота от мотора № 1 к полетному контроллеру.
5. Подключение сигнального провода регуляторов оборота от мотора № 2 к полетному контроллеру.
6. Подключение сигнального провода регуляторов оборота от мотора № 3 к полетному контроллеру.
7. Подключение сигнального провода регуляторов оборота от мотора № 4 к полетному контроллеру.
8. Подключение силовых проводов от регулятора оборотов мотора № 1 к плате распределения питания.
9. Подключение силовых проводов от регулятора оборотов мотора № 2 к плате распределения питания.
10. Подключение силовых проводов от регулятора оборотов мотора № 3 к плате распределения питания.
11. Подключение силовых проводов от регулятора оборотов мотора № 4 к плате распределения питания.
12. Подключение питания светодиодной ленты.
13. Подключение сигнального провода светодиодной ленты к Raspberry Pi.
14. Подключение питания (5V) к Raspberry Pi.
15. Подключение лазерного дальномера к Raspberry Pi.
16. Подключение шлейфа от камеры к Raspberry Pi.
17. Подключение проводной связи от полетного контроллера к Raspberry Pi.
18. Установка пропеллера на мотор № 1.
19. Установка пропеллера на мотор № 2.
20. Установка пропеллера на мотор № 3.
21. Установка пропеллера на мотор № 4.

Решение

Для решения задачи необходимо внимательно посмотреть видео по сборке квадрокоптера из образовательного курса: https://stepik.org/lesson/1431188/step/7?unit=1449603 или ознакомиться с документацией по сборке квадрокоптера: https://clover.coex.tech/ru/assemble_4_2_ws.

Ответ: 1, 3, 4, 5, 14, 15, 16, 17, 18, 20, 21.

Задача 3.2.1.2. Настройка квадрокоптера (5 баллов)

Тема: базовые навыки работы с летающими робототехническими системами.

Условие

Ознакомьтесь с видео и отметьте все ошибки, допущенные при настройке квадрокоптера (ссылка на видео: https://disk.yandex.ru/i/CqE4B18KRF-JMw).

1. При загрузке прошивки в полетный контроллер.
2. В выборе конфигурации рамы квадрокоптера.
3. При калибровке компаса.
4. При калибровке гироскопа.
5. При калибровке акселерометра.
6. При калибровке уровня горизонта.
7. При установке ориентации полетного контроллера.
8. При калибровке радиоаппаратуры управления.
9. При настройке режимов полетного_ контроллера.
10. При назначении аварийного отключения моторов.
11. При настройке параметров питания.

Решение

Для решения задачи необходимо внимательно посмотреть видео по сборке квадрокоптера из образовательного курса: https://stepik.org/lesson/1431188/step/7?unit=1449603 или ознакомиться с документацией по сборке квадрокоптера: https://clover.coex.tech/ru/assemble_4_2_ws.

Ответ: 7, 8, 10, 11.

Задача 3.2.1.3. Автономный полет. Настройка образа (6 баллов)

Темы: настройка квадрокоптера, базовые навыки работы с летающими робототехническими системами.

Условие

Ознакомьтесь с видео и отметьте все ошибки, допущенные при настройке образа квадрокоптера (ссылка на видео: https://disk.yandex.ru/i/Qm5kl5WyMX6jhg).

1. Настройка параметра `aruco`.
2. Настройка параметра `aruco_detect`.
3. Настройка параметра `aruco_map`.
4. Настройка параметра `aruco_vpe`.
5. Настройка параметра `map`.
6. Настройка размера `aruco`-маркера по умолчанию.

7. Настройка параметра `direction_z`.
8. Настройка параметра `direction_y`.
9. Настройка параметров карты: длина маркера.
10. Настройка параметров карты: количество маркеров по оси X.
11. Настройка параметров карты: количество маркеров по оси Y.
12. Настройка параметров карты: расстояние между центрами меток по оси X.
13. Настройка параметров карты: расстояние между центрами меток по оси Y.
14. Настройка параметров карты: номер первого маркера.
15. Настройка параметров карты: «ключ» начала нумерации при генерации карты.
16. Перезагрузка пакетов клевера на образе.

Примечания

Сборка квадрокоптера является стандартной.

Решение

Для решения задачи необходимо внимательно посмотреть видео по сборке квадрокоптера из образовательного курса: https://stepik.org/lesson/1431188/step/7?unit=1449603 или ознакомиться с документацией по сборке квадрокоптера: https://clover.coex.tech/ru/assemble_4_2_ws.

Ответ: 5, 7, 8, 9, 15.

Задача 3.2.1.4. PID-регулятор. Настройка коптера (5 баллов)

Темы: настройка квадрокоптера, базовые навыки работы с летающими робототехническими системами.

Условие

Ознакомьтесь с видео и по характеру полета квадрокоптера соотнесите их с подобранными коэффициентами PID-регулятора.

Ссылки на видео:

1. https://disk.yandex.ru/d/uj6JXzlr7uSDFQ/01пид.mp4.
2. https://disk.yandex.ru/d/uj6JXzlr7uSDFQ/02пид.mp4.
3. https://disk.yandex.ru/d/uj6JXzlr7uSDFQ/03пид.mp4.
4. https://disk.yandex.ru/d/uj6JXzlr7uSDFQ/04пид.mp4.

Примечания

Сборка квадрокоптера Clever 4 является стандартной. Для демонстрации менялся один из коэффициентов относительно рекомендованного значения в прошивке PX4 полетного контроллера.

A.
```
MC_PITCHRATE_P = 0.087
MC_PITCHRATE_I = 0.037
MC_PITCHRATE_D = 0.0044
MC_PITCH_P = 8.5
MC_ROLLRATE_P = 0.015
MC_ROLLRATE_I = 0.037
MC_ROLLRATE_D = 0.0044
```

B.
```
MC_PITCHRATE_P = 0.087
MC_PITCHRATE_I = 0.037
MC_PITCHRATE_D = 0.0044
MC_PITCH_P = 8.5
MC_ROLLRATE_P = 0.087
MC_ROLLRATE_I = 0.037
MC_ROLLRATE_D = 0.010
```

C.
```
MC_PITCHRATE_P = 0.300
MC_PITCHRATE_I = 0.037
MC_PITCHRATE_D = 0.0044
MC_PITCH_P = 8.5
MC_ROLLRATE_P = 0.087
MC_ROLLRATE_I = 0.037
MC_ROLLRATE_D = 0.0044
```

D.
```
MC_PITCHRATE_P = 0.020
MC_PITCHRATE_I = 0.037
MC_PITCHRATE_D = 0.0044
MC_PITCH_P = 8.5
MC_ROLLRATE_P = 0.087
MC_ROLLRATE_I = 0.037
MC_ROLLRATE_D = 0.0044
```

Решение

Для решения задачи необходимо внимательно посмотреть видео по сборке квадрокоптера из образовательного курса: https://stepik.org/lesson/1431188/step/7?unit=1449603 или ознакомиться с документацией по настройке квадрокоптера Clover: https://clover.coex.tech/ru/setup.html#усредненные-коэффициенты-pid-для-клевера-4.

Ответ: 1 — D, 2 — C, 3 — A, 4 — B.

Задача 3.2.1.5. Пакуем сумки (10 баллов)

Темы: ROS, Linux, системы координат.

Условие

Вы удаленно работаете инженером-программистом БПЛА в компании, которая занимается мониторингом объектов топливно-энергетического комплекса (ТЭК).

Для выполнения мониторинга был создан квадрокоптер, работающий на платформе ROS, с использованием образа Clover: https://clover.coex.tech/ru/image.html.

Коллеги, находящиеся за километры от вас на объекте ТЭК, столкнулись с проблемой. Во время последней миссии квадрокоптер патрулировал участок трубопровода, чтобы обнаружить потенциальные утечки и повреждения. В процессе выполнения задачи дрон несколько раз отклонялся от намеченной траектории.

Для того чтобы провести отладку и исправить баг, существуют два варианта: паковать сумки и ехать к коллегам, или просить их паковать `rosbag` файл, который позволяет сохранять данные, публикуемые в топики ROS. Конечно, вы, как уважающий себя удаленщик, выбрали второй метод.

На этом этапе НТО выполните более простую задачу — используя `rosbag` файл, восстановите миссию, по которой должен был пролететь дрон.

`rosbag` файл: https://disk.yandex.ru/d/sVxcVxwBsyXf-w.

Файл начал записываться после взлета квадрокоптера и закончил после его посадки и дизарма. Восстановите `navigate` и `land` команды, выполненные в полете (не учитывая взлет).

Каждая команда `navigate` была выполнена с `frame_id="aruco_map"`. Поэтому координаты в восстановленных командах должны быть в системе координат `aruco_map`.

Формат входных данных

Миссия задается псевдокодом с командами

```
navigate(x: float, y: float, z: float)
```

Для простоты опустим все параметры из оригинальной команды, кроме `x, y, z`, но примем `frame_id="aruco_map"` и `land()`.

Формат выходных данных

В ответе укажите набор команд (каждая на своей строке), которым следовал дрон.

В команде `navigate` указывайте имена параметров и передавайте числа, округленные до десятых (включая `.0` для целых чисел).

Пример указания команды `navigate`: `navigate(x=1.0, y=1.0, z=2.0)`.

Решение

Чтобы восстановить миссию дрона, используя файл `rosbag`, можно следовать следующему плану. Файл `rosbag` содержит данные, публикуемые в топики ROS, такие как координаты дрона, ориентация, данные IMU и другие телеметрические данные.

Для начала ознакомимся с имеющимися в `rosbag` файле топиками. Для этого воспользуемся библиотекой `bagpy`: https://github.com/jmscslgroup/bagpy.

Python

```python
from bagpy import bagreader

bag = bagreader("drone_flight.bag")
for t in bag.topics:
    print(t)
```

Получим такой список:

```
/mavros/altitude
/mavros/battery
/mavros/estimator_status
/mavros/extended_state
/mavros/imu/data
/mavros/imu/data_raw
/mavros/imu/mag
/mavros/imu/static_pressure
/mavros/imu/temperature_imu
/mavros/local_position/odom
/mavros/local_position/pose
/mavros/local_position/velocity_body
/mavros/local_position/velocity_local
/mavros/px4flow/ground_distance
/mavros/px4flow/raw/optical_flow_rad
/mavros/px4flow/raw/send
/mavros/px4flow/temperature
/mavros/rc/out
/mavros/setpoint_position/local
/mavros/setpoint_raw/target_attitude
/mavros/setpoint_raw/target_local
/mavros/state
/mavros/time_reference
/mavros/timesync_status
/mavros/vision_pose/pose
/tf
/tf2_web_republisher/status
/tf_static
```

Понимаем, что были записаны не все топики, однако достаточно имеющихся топиков, связанных с `setpoint` (низкоуровневое задание точки полета квадрокоптера) и `tf` (системы координат и их преобразования).

Далее задачу можно решить двумя способами: с использованием воспроизведения `rosbag`-файла и с использованием геометрии, один из которых мы рассмотрим.

Этот способ более прост для понимания участниками Олимпиады, однако требует установленного ROS или виртуальной машины Clover.

Для начала запустим в терминале `roscore`.

После этого в другом терминале запустим `rosbag play drone_flight.bag`, тем самым начав передачу данных в топики, которые были записаны в `BAG`-файле.

Для визуализации данных в топике запустим `rviz` и через кнопку **Add** добавим отображения `TF` (рис. 3.2.1).

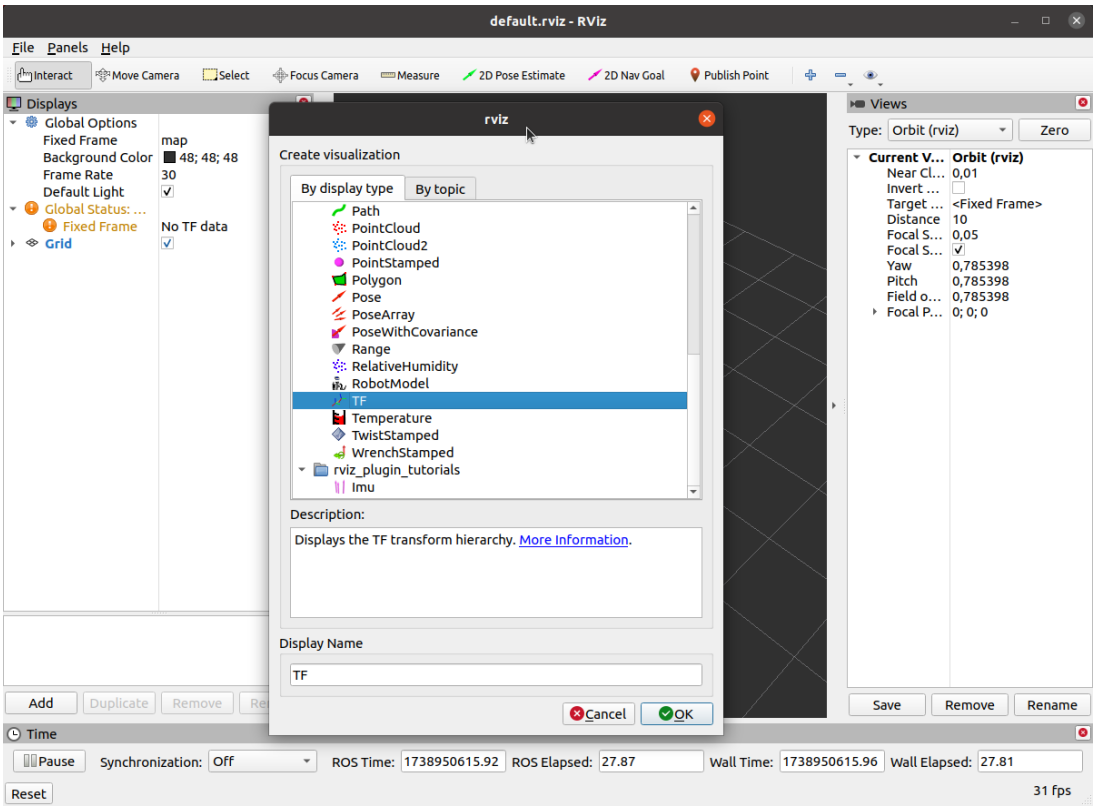


Рис. 3.2.1. Добавление отображения

После добавления увидим все системы координат, которые существовали на момент записи BAG-файла (рис. 3.2.2).

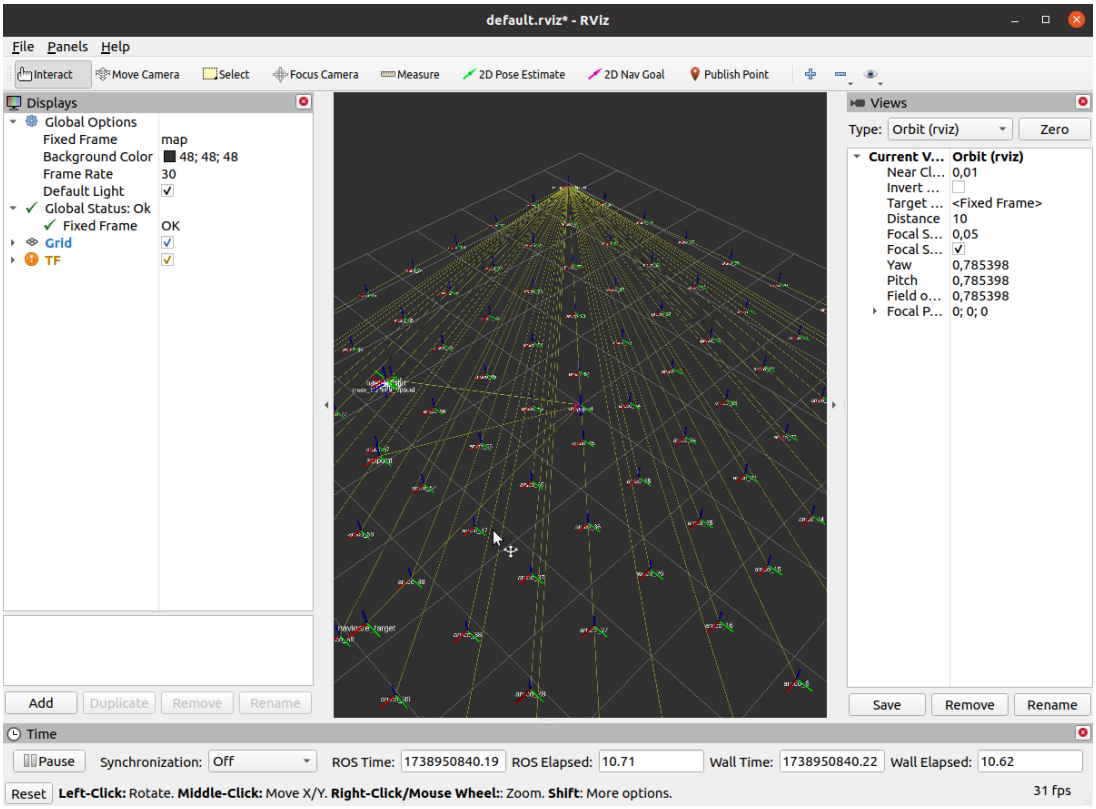


Рис. 3.2.2. Системы координат в BAG-файле

Здесь можно увидеть, что начало системы координат `navigate_target` указывает на координаты точки, в которую сейчас летит дрон с использованием `navigate`.

После этого напишем простой код, который регистрирует изменения преобразований системы координат `navigate_target` относительно `aruco_map`. Смещение СК `navigate_target` относительно `aruco_map` будет показывать координаты точки в СК `aruco_map`, в которую летит дрон.

Python

```python
import rospy
import tf2_ros

rospy.init_node("rosbag_solution")

tfBuffer = tf2_ros.Buffer()
tf2_ros.TransformListener(tfBuffer)

rate = rospy.Rate(1)
last_coords = None
while not rospy.is_shutdown():
    transform = tfBuffer.lookup_transform(
        target_frame="aruco_map",
        source_frame="navigate_target",
        time=rospy.Time(),
        timeout=rospy.Duration(10),
    )
    coords = (
        round(transform.transform.translation.x, 2),
        round(transform.transform.translation.y, 2),
        round(transform.transform.translation.z, 2),
    )
    if coords != last_coords:
        last_coords = coords
        print(last_coords)
    rate.sleep()
```

Запустим код, следом за ним запустим воспроизведение BAG-файла. Получаем следующий результат (рис. 3.2.3).

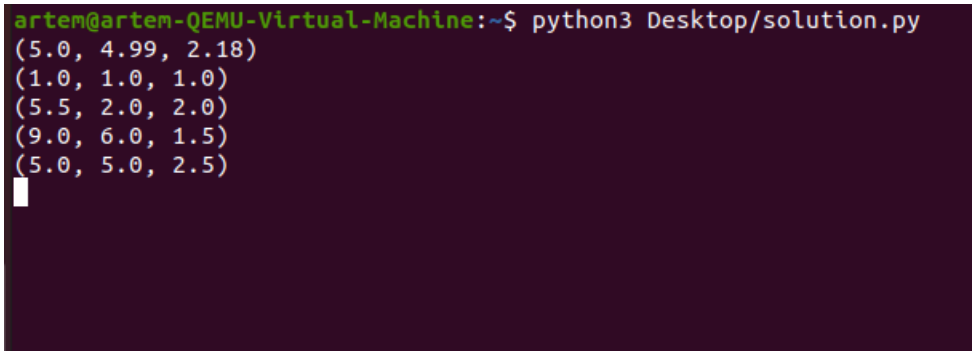


Рис. 3.2.3. Запуск кода

Первую точку откинем, так как это координата предыдущей точки полета дрона (команда `navigate` была выполнена до начала записи BAG-файла, что противоречит условию).

Остается подставить полученные координаты под формат ответа, не забыв про `land()`.

Ответ:

```
navigate(x=1.0, y=1.0, z=1.0)
navigate(x=5.5, y=2.0, z=2.0)
navigate(x=9.0, y=6.0, z=1.5)
navigate(x=5.0, y=5.0, z=2.5)
land()
```

Задача 3.2.1.6. Летаем оптимально (7 баллов)

Темы: математика, кратчайший путь, представление цвета.

Условие

После исправления ошибки, возникшей на дроне, необходимо провести мониторинг участка магистрального нефтепровода.

Магистральный нефтепровод — это сложная инженерная система, включающая в себя линейную часть, головные и промежуточные перекачивающие станции, резервуарные парки и другие сооружения. Каждому типу сооружений присвоен определенный цвет.

Напишите ROS-ноду, которая выполняет следующие действия:

- взлет квадрокоптера;
- включение светодиодной ленты заданного цвета, коды цветов: https://gist.github.com/ntomaterials/60d0d05a6e1b47e1c6edb87fa328e011;
- облет по точкам, в которых находятся сооружения заданного цвета;
- посадка в конечной точке маршрута.

В качестве ответа создайте программу, генерирующую псевдокод ROS-ноды (см. формат выходных данных), который будет выполнять задание наиболее оптимально (кратчайшим путем).

В случае, если оптимальных решений несколько, выведите любое из них.

Формат входных данных

Первая строка содержит координаты старта (x_{start}, y_{start}) через пробел; вторая строка содержит координаты финиша (x_{finish}, y_{finish}) через пробел.

Далее поочередно идут $1 \leqslant N \leqslant 1000$ строк, содержащие параметры каждой метки: $color_i$ — наименование цвета i-й метки; x_i , y_i — координаты расположения i-й метки.

$0 \leqslant x_{start}, y_{start}, x_{finish}, y_{finish}, x_i, y_i < 100$.

В конце ввода идет строка, содержащая $color_{req}$ — наименование цвета маркеров, по которым необходимо выполнить полет.

Формат выходных данных

Псевдокод для квадрокоптера, содержащий только функции:

- `navigate(x: int, y: int, auto_arm: bool = False)`
- `set_effect(r: int, g: int, b: int)`
- `land()`

Примеры

Стандартный ввод
71 36 8 18 Red 28 46 Red 13 39 Red 12 45 SlateBlue 12 13 SlateBlue 75 41 SlateBlue 0 1 Red 78 89 SlateBlue

Стандартный вывод
navigate(x=71, y=36, auto_arm=true) set_effect(r=255, g=0, b=0) navigate(x=75, y=41) navigate(x=12, y=13) navigate(x=0, y=1) navigate(x=8, y=18) land()

Решение

1. Зажигаем светодиодную ленту соответствующим цветом.
2. Зная координаты необходимых меток, составляем все вариации полета со старта и до финиша через функцию `permutations` из библиотеки `itertools`, при этом пролетая все метки.
3. Рассчитываем дистанцию полета от одной метки к другой:

$$distance = \sqrt{(x_b - x_a)^2 + (y_b - y_a)^2}.$$

4. Также не забываем о том, что до первого маркера необходимо долететь со старта, а с последнего — на финиш. Получаем расстояния каждой возможной вариации и берем наименьшую из всех.

Ниже представлено решение на языке Python.

Python

```python
from collections import defaultdict
from itertools import permutations

COLORS = {...}  # коды цветов из условия


def dist(p1, p2):
    return ((p1[0] - p2[0]) ** 2 + (p1[1] - p2[1]) ** 2) ** 0.5


def hex_to_rgb(hex: str) -> tuple[int, int, int]:
    return tuple(int(hex.replace("#", "")[i : i + 2], 16) for i in (0, 2, 4))


start = tuple(map(int, input().split()))
finish = tuple(map(int, input().split()))

colors = defaultdict(list)

while len(m := input().split()) == 3:
    color, *coords = m
    colors[color].append(list(map(int, coords)))

points = colors[m[0]]
led_command = "set_effect(r={}, g={}, b={})".format(*hex_to_rgb(COLORS[m[0]]))

min_perm = []
if points:
    min_perm = min(
        permutations(points),
        key=lambda perm: sum(dist(perm[i], perm[i + 1]) for i in range(len(perm) - 1))
        + dist(start, perm[0])
        + dist(perm[-1], finish),
    )

ans = (
    [led_command, "navigate(x={}, y={}, auto_arm=true)".format(*start)]
    + ["navigate(x={}, y={})".format(*e) for e in min_perm]
    + ["navigate(x={}, y={})".format(*finish), "land()"]
)

print("\n".join(ans))
```

Задача 3.2.1.7. Сколько перекрестков? (8 баллов)

Темы: OpenCV, контуры.

Условие

Для анализа инфраструктуры на одном из участков, оборудованном сложной сетью трубопроводов, был использован дрон, который проводил воздушную съемку и фиксировал состояние труб. Обработайте полученное изображение, чтобы опреде-

лить количество пересечений трубопроводов на участке. Данная информация будет полезна для планирования обслуживания и оценки износа труб в местах перекрестков.

Изображение представляет собой черно-белое фото, где дороги выделены белыми линиями на черном фоне. Гарантируется, что в изображении присутствуют только белые и черные пиксели; помимо этого гарантируется, что пересекаться может не больше двух дорог.

Формат входных данных

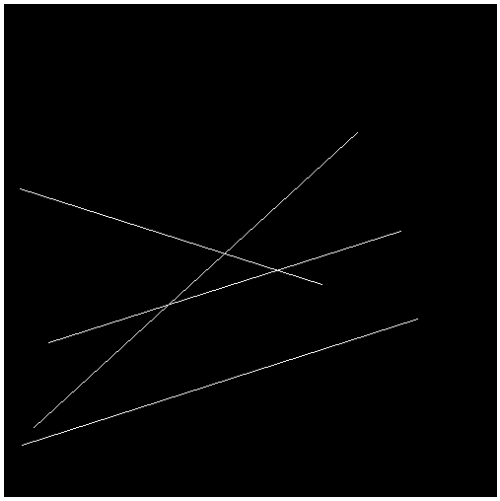
По пути `./input.jpg` будет находиться файл с входным изображением, которое необходимо обработать.

Формат выходных данных

Одно число — количество перекрестков белых линий.

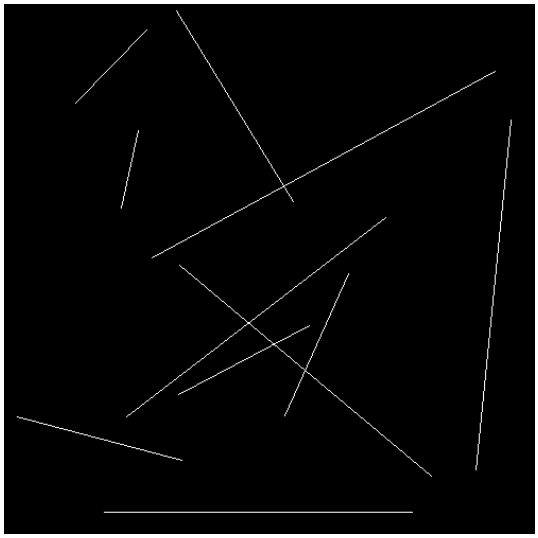
Примеры

Пример №1

Стандартный ввод


Стандартный вывод
3

Пример №2

Стандартный ввод


Стандартный вывод
4

Решение

1. **Загрузка и подготовка изображения:**
   1.1. Программа открывает файл с изображением (`input.png`) и переводит его в черно-белый формат. Это значит, что каждый пиксель становится либо белым (значение 1), либо черным (значение 0).
   1.2. Изображение превращается в массив чисел (с помощью библиотеки `numpy`), чтобы его можно было легко обрабатывать программно.
2. **Выделение областей с пересечениями (с помощью свертки):**
   - Программа использует специальную маленькую матрицу, которую называют «ядром» (`kernel`). Это матрица размером 3×3:

$$\begin{matrix} 1 & 1 & 1 \\ 1 & 5 & 1 \\ 1 & 1 & 1 \end{matrix}$$

   - Идея в том, чтобы для каждого пикселя изображения посчитать сумму значений его соседей с такими коэффициентами. Если в этой области пересекаются линии (то есть вокруг пикселя много белых точек), то сумма будет высокой.
   - Функция `convolve` (из библиотеки `scipy`) «пробегается» по всему изображению и для каждого пикселя считает такую сумму. Получается новое изображение, где значение пикселя говорит о том, сколько вокруг него белых пикселей с учетом коэффициентов ядра.
3. **Нахождение точек, где могут быть пересечения:**

- После свертки программа ищет все пиксели, у которых полученное значение больше или равно 8. Почему 8? Подобрано экспериментально.
- Таким образом находятся все потенциальные точки пересечения.

4. **Группировка близко расположенных точек (кластеризация):**
   - Из-за того как работает свертка, одно реальное пересечение может быть представлено несколькими соседними пикселями с высоким значением. Но в данном случае нужно посчитать каждое пересечение только один раз.
   - Для этого используется алгоритм `DBSCAN` (из библиотеки `scikit-learn`). Он группирует (кластеризует) близко расположенные точки вместе.
   - Каждая группа точек (кластер) считается одним пересечением. То есть количество таких кластеров и будет искомым числом перекрестков.
5. **Вывод результата:**
   - После группировки программа считает количество получившихся кластеров.
   - Это число выводится как результат работы программы — количество пересечений труб (перекрестков белых линий).

Ниже представлено решение на языке Python.

Python

```python
from PIL import Image
import numpy as np
from scipy.ndimage import convolve
from sklearn.cluster import DBSCAN


def count_intersections(image_path):
    image = Image.open(image_path).convert("1")
    image_array = np.array(image)

    kernel = np.array([
                       [1, 1, 1],
                       [1, 5, 1],
                       [1, 1, 1],
                       ])

    convolved = convolve(image_array.astype(np.int32), kernel,
    ↪  mode='constant', cval=0)

    im = convolved * (255 / convolved.max())
    im = im.astype(np.uint8)

    intersection_points = np.where(convolved >= 8)

    intersection_points = list(zip(*intersection_points))
    if not intersection_points:
        return 0

    clustering = DBSCAN(eps=10, min_samples=2).fit(intersection_points)
    n_clusters = len(set(clustering.labels_)) - (1 if -1 in
    ↪  clustering.labels_ else 0)

    return n_clusters
```

```
32
33  if __name__ == "__main__":
34      print(count_intersections("input.png"))
```

Задача 3.2.1.8. Проектирование фланца (5 баллов)

Темы: квадрокоптер, расчет.

Условие

После неудачной посадки квадрокоптера повредилась деталь крепления луча. Для крепления луча к корпусу квадрокоптера используется специализированный фланец. Для изготовления нового фланца необходима 3D-модель, но на производстве есть только чертеж данной детали. Спроектируйте фланец в CAD-программе, используя чертеж. Для контроля соответствия детали определите массу получившейся детали в CAD-программе.

Материал — алюминий (плотность: 2 800 кг/м^3).

Чертеж фланца см. на рис. 3.2.4.

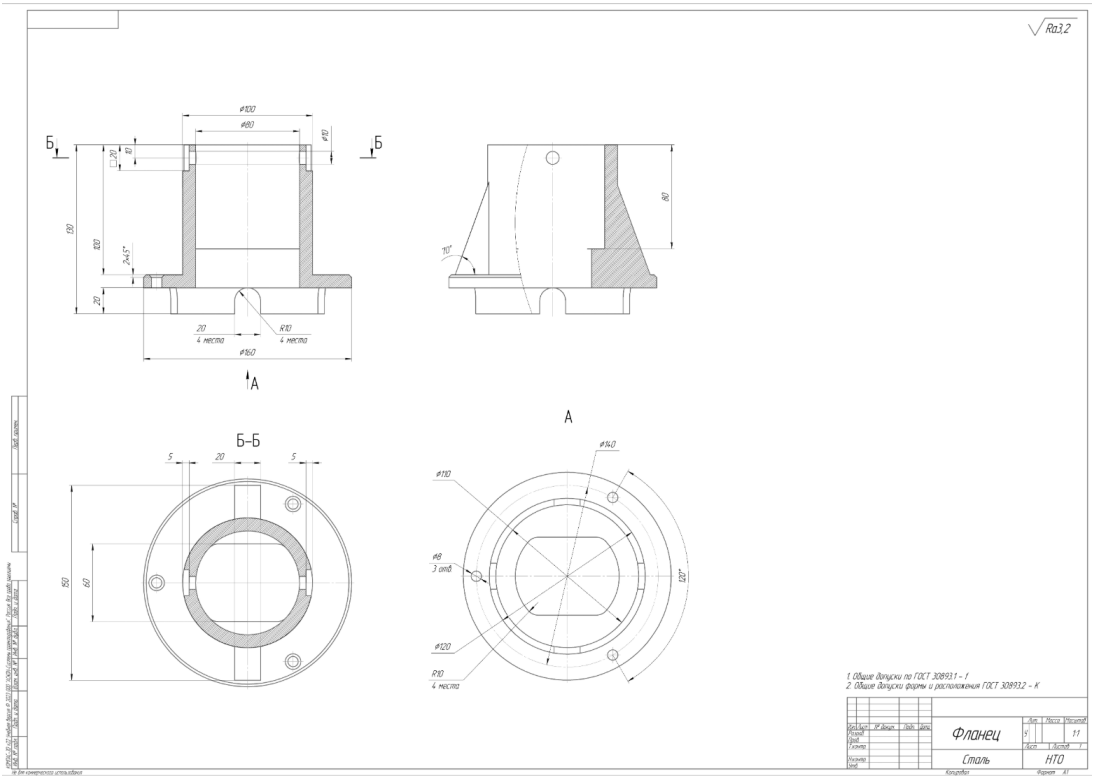


Рис. 3.2.4. Чертеж фланца

Решение

Необходимо спроектировать фланец по чертежу в CAD-программе, задать материал и взвесить модель при помощи анализа модели в CAD-программе.

Ответ: 1435.

Задача 3.2.1.9. Конструкторская документация (5 баллов)

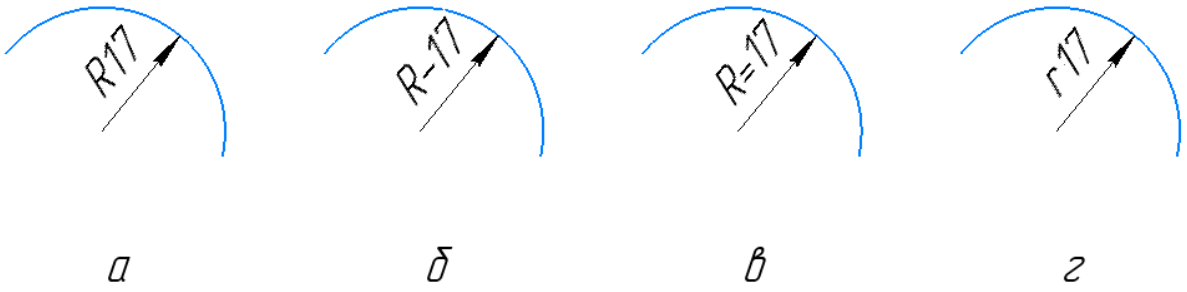
Тема: конструкторская документация.

Условие

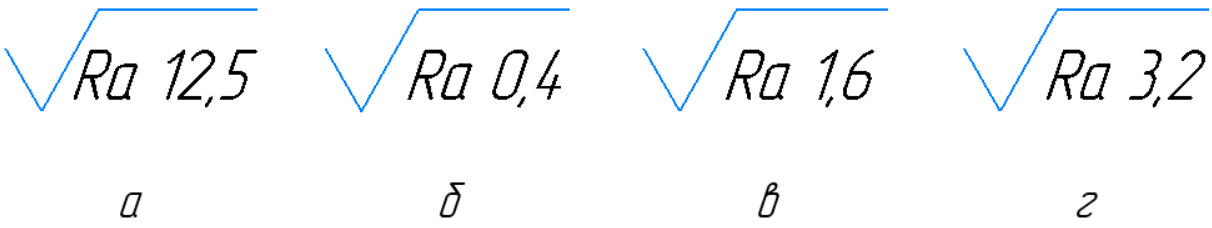
Тест на знание конструкторской документации, 1 балл за каждый верный ответ.

Ответьте на вопросы:

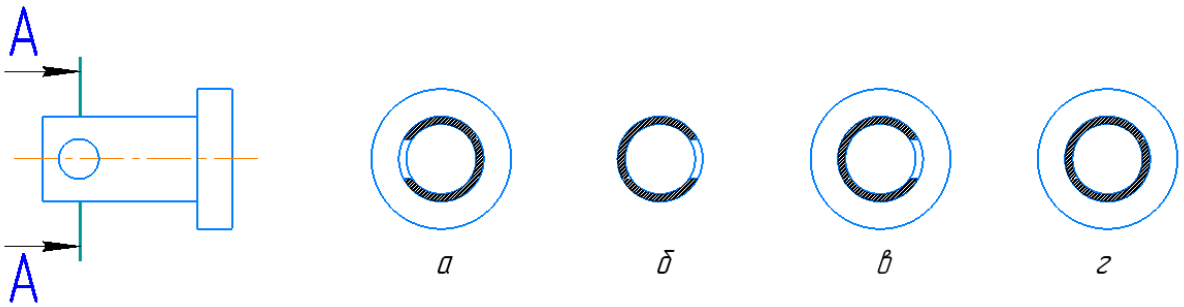
1. Какие документы КД относится к текстовым?
   а. Чертеж детали.
   б. Перечень элементов.
   в. Схема.
   г. Спецификация.
2. Какое обозначение радиуса правильное?



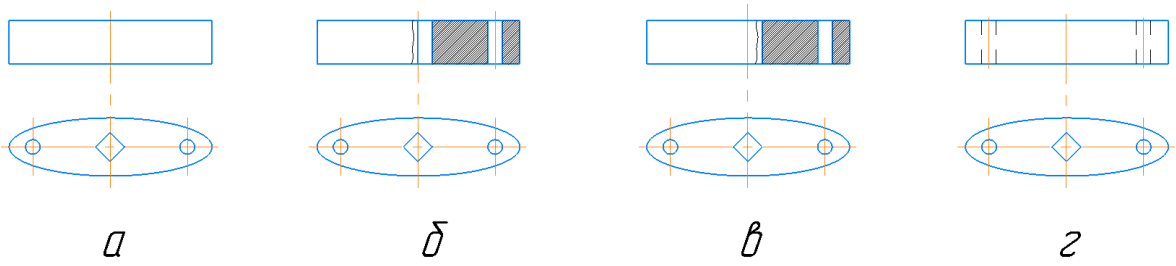
3. Какая поверхность имеет наибольшую шероховатость?



4. Сечение А–А соответствует изображению...



5. Укажите, где правильно соединен вид и разрез.



Решение

1. Согласно ГОСТ 2.102-2013 из четырех вариантов ответа к текстовой документации относится только перечень элементов (б) и спецификация (г).
   Правильный вариант ответа: б, г.
2. Согласно ГОСТ 2.102-2013 обозначение радиуса допускается только большой буквой R и численным обозначением, следующим сразу за буквой R.
   Правильный вариант ответа: а.
3. Согласно ГОСТ 2.102-2013 шероховатость тем меньше чем меньше число ее обозначающее.
   Правильный вариант ответа: б.
4. Согласно ГОСТ 2.102-2013 разрез — это изображение, полученное при мысленном рассечении предмета плоскостью.
   Правильный вариант ответа: в.
5. Согласно ГОСТ 2.102-2013 правильно соединен вид и разрез только на второй картинке.
   Правильный вариант ответа: б.

Ответ: 1 — б, г, 2 — а, 3 — б, 4 — в, 5 — б.

3.2.2. Командные задачи

Командные задачи второго этапа инженерного тура открыты для решения. Соревнование доступно на платформе Яндекс.Контест: https://contest.yandex.ru/contest/69921/enter/.

Задача 3.2.2.1. Расчет луча квадрокоптера (15 баллов)

Темы: квадрокоптер, сопротивление материалов.

Условие

Необходимо рассчитать на прочность луч квадрокоптера.

На рис. 3.2.5 показано сечение луча и нагрузки, действующие на него.

Рассчитайте:

1. момент инерции сечения луча, мм^4 (округлить до сотых);
2. момент сопротивления сечения луча, мм^3 (округлить до сотых);
3. максимальные нормальные напряжения луча, МПа (округлить до тысячных);
4. максимальные касательные напряжения луча, МПа (округлить до тысячных);
5. максимальные главные напряжения луча, МПа (округлить до тысячных).

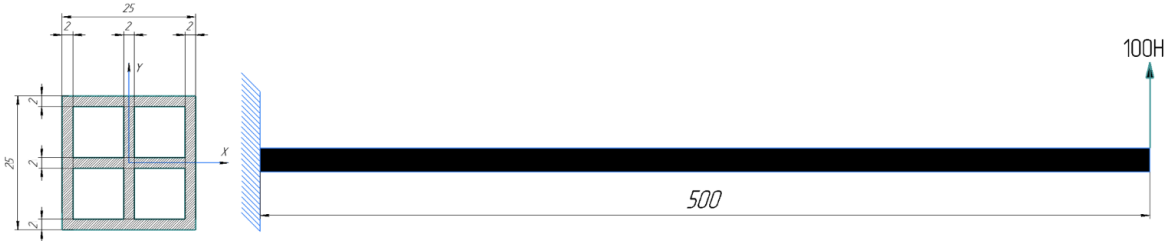


Рис. 3.2.5. Луч квадрокоптера

Решение

Так как сечение является квадратным, значения по оси X и Y равны.

1. **Расчет момента инерции сечения**.

   Сечение луча состоит из квадратов, формула для расчета момента инерции квадрата

   $$I_z = \frac{h^4}{12},$$

   где h — сторона квадрата.

   1.1. Расчет момента инерции внешнего квадрата

   $$I_1 = \frac{h^4}{12} = \frac{25^4}{12} = 32\,552{,}08 \text{ мм}^4.$$

   1.2. Расчет внутренних вырезов

   $$I_z = I_c + md^2,$$

   $$I_2 = 4 \cdot \left(\frac{h^4}{12} + h^2 \cdot 5{,}75\right) = 4 \cdot \left(\frac{9{,}5^4}{12} + 9{,}5^2 \cdot 5{,}75\right) = 14\,650{,}58 \text{ мм}^4,$$

   где $h = 9{,}5$ мм — сторона вырезанного квадрата, $m = 5{,}75$ мм — расстояние между центром большого квадрата и маленьких квадратов.

   1.3. Расчет всего сечения.

   Вычитаем из момента инерции большого квадрата моменты инерции вырезов

   $$I_{x,y} = I_1 - I_2 = 32\,552{,}08 - 14\,650{,}58 = 17\,901{,}49 \text{ мм}^4.$$

   Момент инерции сечения:

   $$I_{x,y} = 17\,901{,}49 \text{ мм}^4.$$

2. **Расчет момента сопротивления сечения**.

Максимальный момент сопротивления сечения считается находится в точке максимального удаления от оси, то есть

$$\frac{y}{2} = \frac{h}{2} = 12{,}5 \text{ мм}.$$

Формула для расчета сопротивления сечения квадрата:

$$W_x y = \frac{I_{x,y}}{\frac{y}{2}} = \frac{17\,901{,}49}{12{,}5} = 1\,432{,}12 \text{ мм}^3.$$

Момент сопротивления сечения: $W_{xy} = 1\,432{,}12$ мм^3.

Расчет максимальных напряжений.

Формула для расчета нормальных напряжений в точке (формула Навье – Стокса):

$$\sigma = \frac{M_z \cdot y}{I}.$$

Формула для расчета максимальных нормальных напряжений:

$$\sigma_{max} = \frac{M_z}{W}.$$

Формула для расчета касательных напряжений (формула Журавского):

$$\tau = \frac{Q \cdot S_z}{I_x \cdot B_y}.$$

Построим эпюру нагружения, рис. 3.2.6.

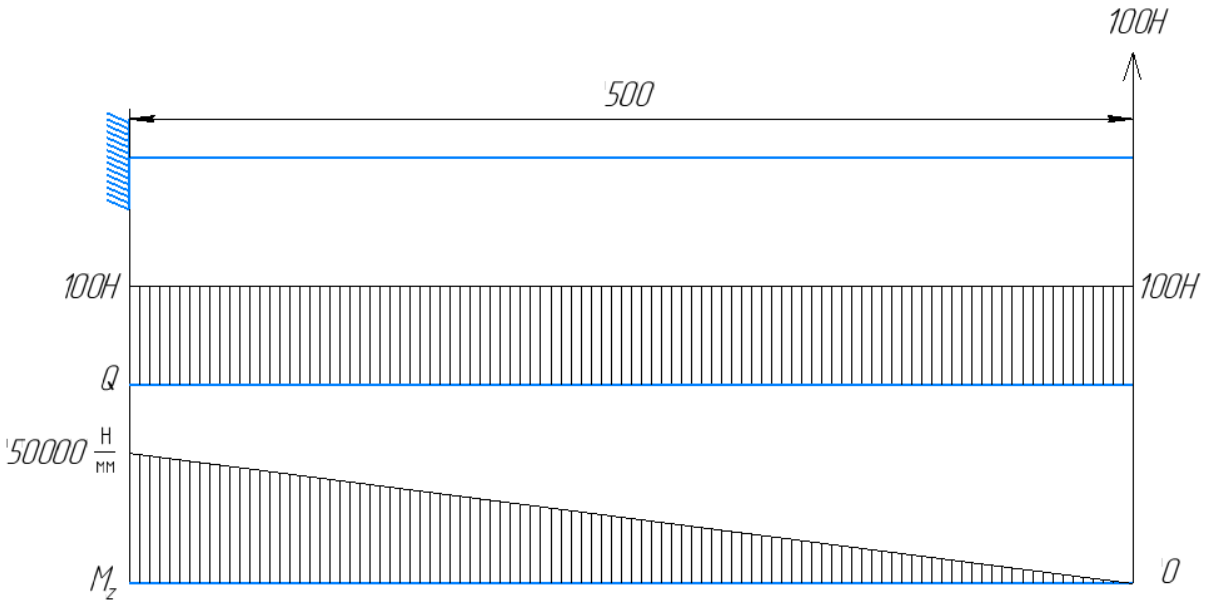


Рис. 3.2.6. Эпюра нагружения

3. **Расчет максимальных нормальных напряжений**.

   Подставляем значения в формулу:

$$\sigma_{\max} = \frac{M_z}{W} = \frac{5\,000}{1\,432{,}12} = 34{,}913 \text{ МПа}.$$

   Максимальные нормальные напряжения:

$$\sigma_{\max} = 34{,}913 \text{ МПа}.$$

4. **Расчет максимальных касательных напряжений**.

   Для нахождения максимальных касательных напряжений необходимо посчитать касательные напряжения в четырех точках и построить эпюру.

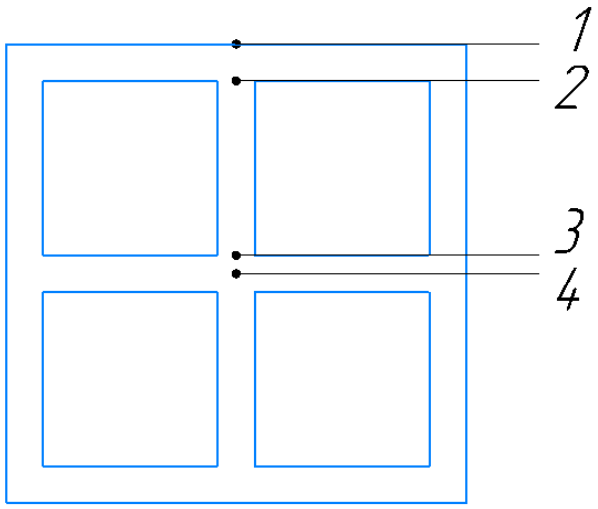


Рис. 3.2.7. Касательные напряжения

4.1. **Расчет касательных напряжений в точке 1**.

   Формула для расчета касательных напряжений:

$$\tau = \frac{Q \cdot S_z}{I_x \cdot B_y}.$$

   Рассчитываем площадь отсеченной части сечения, которая находится выше точки 1. В данном случае это значение равно 0:

$$A_{\text{отс1}} = 0.$$

   Рассчитываем центр тяжести отсеченной части, в данном случае центр тяжести находится в точке 1:

$$y_{c1} = \frac{h}{2} = \frac{25}{2} = 12{,}5 \text{ мм}.$$

   Статический момент отсеченной части:

$$S_x = y_{c1} \cdot A_{\text{отс1}} = 12{,}5 \cdot 0 = 0 \text{ мм}^3.$$

Считаем касательные напряжения:

$$\tau_{\text{т.1}} = \frac{Q \cdot S_z}{I_x \cdot B_y} = \frac{100 \cdot 0}{17\,901{,}49 \cdot 25} = 0 \text{ МПа},$$

где $B_y = 25$ мм — ширина сечения.

4.2. **Расчет касательных напряжений в точке 2**.

Формула для расчета касательных напряжений:

$$\tau = \frac{Q \cdot S_z}{I_x \cdot B_y}.$$

Рассчитываем площадь отсеченной части сечения, которая находится выше точки 2:

$$A_{\text{отс2}} = h \cdot t = 25 \cdot 2 = 50 \text{ мм}^2,$$

где $t = 2$ мм — высота сечения.

Рассчитываем центр тяжести отсеченной части:

$$y_{c2} = \frac{h}{2} - \frac{t}{2} = \frac{25}{2} - \frac{2}{2} = 11{,}5 \text{ мм}.$$

Статический момент отсеченной части:

$$S_x = y_{c2} \cdot A_{\text{отс2}} = 11{,}5 \cdot 50 = 575 \text{ мм}^3.$$

Так как в точке 2 происходит изменение ширины сечения, посчитаем значение касательных напряжений до и после изменения:

$$B_y^1 = 25 \text{ мм},$$

$$B_y^2 = 6 \text{ мм}.$$

Считаем касательные напряжения в точке 1:

$$\tau_{\text{т.2}}^1 = \frac{Q \cdot S_z}{I_x \cdot B_y^1} = \frac{100 \cdot 575}{17\,901{,}49 \cdot 25} = 0{,}128 \text{ МПа}.$$

Считаем касательные напряжения в точке 2:

$$\tau_{\text{т.2}}^2 = \frac{Q \cdot S_z}{I_x \cdot B_y^2} = \frac{100 \cdot 575}{17\,901{,}49 \cdot 6} = 0{,}535 \text{ МПа}.$$

4.3. **Расчет касательных напряжений в точке 3**.

Формула для расчета касательных напряжений:

$$\tau = \frac{Q \cdot S_z}{I_x \cdot B_y}.$$

Рассчитываем площадь отсеченной части сечения, которая находится выше точки 3:

$$A_{\text{отс3}} = 3(b \cdot t) + (h \cdot t) = 3 \cdot (9{,}5 \cdot 2) + (25 \cdot 2) = 107 \text{ мм}^2,$$

где $t = 2$ мм — высота сечения, $b = 9{,}5$ мм — высота выреза.

Рассчитываем центр тяжести отсеченной части. Так как отсеченная фигура является сложной, воспользуемся формулой расчета центра тяжести сложных фигур:

$$y_{c3} = \frac{\sum A_i \cdot x_i}{\sum A_i},$$

где $\sum A_i$ — площадь простой фигуры, x_i — расстояния от центра масс простой фигуры до начала системы координат.

$$y_{c3} = \frac{\sum A_i \cdot x_i}{\sum A_i} = \frac{(25 \cdot 2) \cdot 11{,}5 + (9{,}5 \cdot 2 \cdot 3) \cdot 5{,}75}{(25 \cdot 2) + (9{,}5 \cdot 2 \cdot 3)} = 8{,}436\,92 \text{ мм}.$$

Статический момент отсеченной части:

$$S_x = y_{c3} \cdot A_{\text{отс3}} = 8{,}436\,92 \cdot 107 = 902{,}75 \text{ мм}^3.$$

Так как в точке 3 происходит изменение ширины сечения, посчитаем значение касательных напряжений до и после изменения:

$$B_y^1 = 6 \text{ мм},$$

$$B_y^2 = 25 \text{ мм}.$$

Считаем касательные напряжения в точке 1:

$$\tau_{\text{т.3}}^1 = \frac{Q \cdot S_z}{I_x \cdot B_y^1} = \frac{100 \cdot 902{,}75}{17\,901{,}49 \cdot 6} = 0{,}8404 \text{ МПа}.$$

Считаем касательные напряжения в точке 2:

$$\tau_{\text{т.3}}^2 = \frac{Q \cdot S_z}{I_x \cdot B_y^2} = \frac{100 \cdot 902{,}75}{17\,901{,}49 \cdot 25} = 0{,}2017 \text{ МПа}.$$

4.4. **Расчет касательных напряжений в точке 4**.

Формула для расчета касательных напряжений:

$$\tau = \frac{Q \cdot S_z}{I_x \cdot B_y}.$$

Рассчитываем площадь отсеченной части сечения, которая находится выше точки 4:

$$A_{\text{отс4}} = 3(b \cdot t) + (h \cdot t) + \left(h \cdot \frac{t}{2}\right) = 3 \cdot (9{,}5 \cdot 2) + (25 \cdot 2) + \left(25 \cdot \frac{2}{2}\right) = 132 \text{ мм}^2,$$

где $t = 2$ мм — высота сечения, $b = 9{,}5$ мм — высота выреза.

Рассчитываем центр тяжести отсеченной части. Так как отсеченная фигура является сложной, воспользуемся формулой расчета центра тяжести сложных фигур:

$$y_{c4} = \frac{\sum A_i \cdot x_i}{\sum A_i},$$

где $\sum A_i$ — площадь простой фигуры, x_i — расстояния от центра масс простой фигуры до начала системы координат.

$$y_{c4} = \frac{\sum A_i \cdot x_i}{\sum A_i} = \frac{(25 \cdot 2) \cdot 11{,}5 + (9{,}5 \cdot 2 \cdot 3) \cdot 5{,}75 + \left(25 \cdot \frac{2}{2}\right) \cdot 0{,}5}{(25 \cdot 2) + (9{,}5 \cdot 2 \cdot 3) + \left(25 \cdot \frac{2}{2}\right)} = \\ = 6{,}9337 \text{ мм}.$$

Статический момент отсеченной части:

$$S_x = y_{c4} \cdot A_{\text{отс4}} = 6{,}9337 \cdot 132 = 915{,}25 \text{ мм}^3.$$

Считаем касательные напряжения:

$$\tau_{\text{т.4}} = \frac{Q \cdot S_z}{I_x \cdot B_y^1} = \frac{100 \cdot 915{,}25}{17\,901{,}49 \cdot 25} = 0{,}2045 \text{ МПа}.$$

4.5. Эпюра касательных напряжений

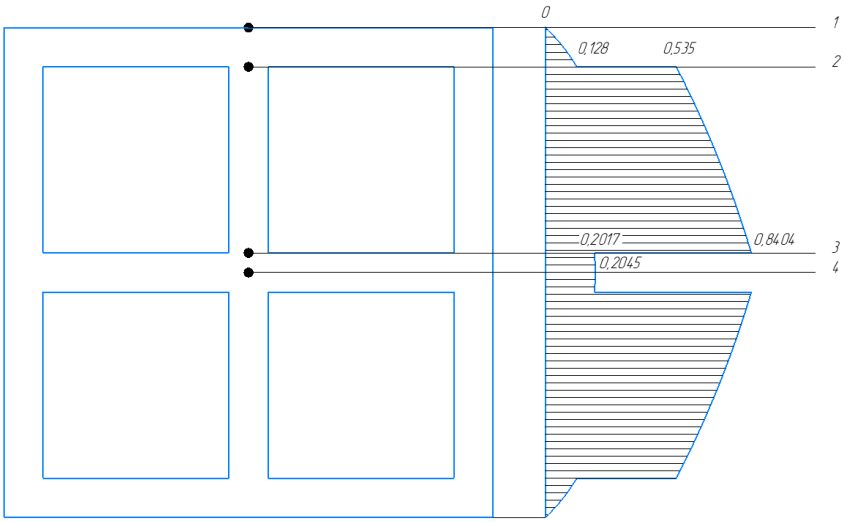


Рис. 3.2.8. Эпюра касательных напряжений

Из этого следует, максимальные касательные напряжения:

$$\tau_{\max} = 0{,}8404 \text{ МПа}.$$

5. Расчет максимальных главных напряжений

Общая формула расчета главных напряжений:

$$\sigma_{\max} = \frac{\sigma}{2} + \frac{1}{2}\sqrt{\sigma^2 + 4\tau^4}.$$

Так как касательные напряжения много меньше нормальных напряжений и в точке 4 действуют максимальные нормальные напряжения, а касательные равны 0, то максимальные главные напряжения равны максимальным нормальным.

Максимальные главные напряжения:

$$\sigma_{\max} = 34{,}913 \text{ МПа}.$$

Ответ: 17 901,49; 1432,12; 34,913; 0,84; 34,913.

Задача 3.2.2.2. Почти как на финале (30 баллов)

Темы: ROS, Gazebo, визуализация, Front-end.

Условие

Легенда

Для мониторинга угольного завода используются квадрокоптеры на базе ROS. Крыша у каждого строения, расположенного на территории завода, имеет свой цвет и определяется его назначением:

- административные здания имеют красные крыши,
- лаборатории — зеленые,
- вход в шахту — желтый,
- здания для обогащения угля — синие.

Разработайте систему, которая позволит операторам эффективно управлять работой дронов на заводе. Представьте работу системы перед тем, как ее использование будет одобрено на реальном заводе.

Задание

1. Установить образ симулятора Gazebo или ПО на свою систему.
2. Подготовить `bash/python` скрипт для настройки образа симулятора (включение навигации по `aruco`-картам и т. д.).
   2.1. Настройка `clover.launch`.
   2.2. Настройка `aruco.launch\verb`.
   2.3. Дополнительные настройки, необходимые для выполнения задания.
3. Подготовить `bash/python` скрипт для случайной генерации мира.
   3.1. Использовать эту модель https://github.com/bart02/dronepoint в качестве зданий.
   3.2. Установить здания в количестве 5 шт.
   3.3. Минимальное расстояние между центрами зданий 1 м.
   3.4. Каждое здание должно находиться в пределах $x \in [0{,}9]$, $y \in [0{,}9]$.
   3.5. Координаты и цвет каждого здания генерируются случайно.
4. Подготовить `python` скрипт (ROS-ноду) для выполнения полета и выполнения сканирования.
   4.1. При сканировании определить координаты здания в системе `aruco_map` и цвет его крыши.
   4.2. На миссию ограничено количество взлетов и посадок — 1.
   4.3. Возврат на старт после выполнения сканирования.
   4.4. Все результаты сканирования записывать в топик (название `/buildings`, тип данных на усмотрение команды).
5. Разработать веб-приложение с графическим интерфейсом пользователя, в котором будут находиться кнопки управления миссией (старт, пауза, стоп), найденные здания с привязкой к карте.
   5.1. Кнопки управления миссией:
      5.1.1. старт — запуск кода с предыдущего шага (если код запущен — кнопка недоступна);

5.1.2. стоп — прерывание кода, выполнение посадки;

5.1.3. `kill switch` — прерывание кода, дизарм дрона.

5.2. Автоматически обновляемая 2D-карта с возможностью отображения начала координат и каждого найденного здания (квадрат с учетом цвета их крыши).

5.3. Автоматически обновляемый список найденных зданий с отображением цвета, координат, типа (административные здания имеют красные крыши, лаборатории — зеленые, вход в шахту — желтый, здания для обогащения угля — синие).

Критерии оценивания

№	Критерий	Балл за 1 случай	Количество случаев	Сумма
1	Установка образа симулятора Gazebo или ПО на систему	2	1	2
2	Подготовка bash/python скрипта для настройки симулятора			
2.1	Настройка `clover.launch`	1	1	1
2.2	Настройка `aruco.launch`	1	1	1
3	Подготовка bash/python скрипта для случайной генерации мира			
3.1	Использование модели зданий из репозитория https://github.com/bart02/dronepoint	1	1	1
3.2	Установка зданий в количестве 5 шт.	0,4	5	2
3.3	Минимальное расстояние между центрами зданий 1 м	1	1	1
3.4	Каждое здание находится в пределах $x \in [0,9]$, $y \in [0,9]$	0,2	5	1
3.5	Случайная генерация координат и цвета каждого здания	1	1	1
4	Подготовка ROS-ноды для выполнения полета и выполнения сканирования			
4.1	Сканирование зданий с определением координат и цвета крыши	0,5	10	5
4.2	Ограничение на количество взлетов и посадок — 1	2	1	2
4.3	Возврат на старт после выполнения сканирования	2	1	2
4.4	Запись результатов сканирования в топик (`/buildings`)	2	1	2
5	Разработка веб-приложения с графическим интерфейсом для управления миссией			
5.1	Реализация кнопок управления миссией (старт, пауза, стоп, `kill switch`)	1	3	3
5.2	Автоматически обновляемая 2D-карта с координатами и цветом зданий	3	1	3

№	Критерий	Балл за 1 случай	Количество случаев	Сумма
5.3	Список найденных зданий с отображением цвета, координат и типа здания	3	1	3
			ИТОГО	**30**

Решение

Генерация карты. Код программы на языке Python:

Python

```python
import os
import shutil
import sys

# Карта ArUco в виде строки
aruco_map = '''# id length  x   y   z   rot_z   rot_y   rot_x
0   0.22    0.0 0.0 0   0   0   0
1   0.22    1.0 0.0 0   0   0   0
2   0.22    2.0 0.0 0   0   0   0
3   0.22    3.0 0.0 0   0   0   0
4   0.22    4.0 0.0 0   0   0   0
5   0.22    5.0 0.0 0   0   0   0
6   0.22    6.0 0.0 0   0   0   0
7   0.22    7.0 0.0 0   0   0   0
8   0.22    8.0 0.0 0   0   0   0
9   0.22    9.0 0.0 0   0   0   0
10  0.22    0.0 1.0 0   0   0   0
...
99  0.22    9.0 9.0 0   0   0   0
'''

def replace_string(file_path, old_string, new_string):
    """
    Заменяет строку в файле.

    Аргументы:
        file_path (str): Путь к файлу, который нужно изменить.
        old_string (str): Строка, которую нужно найти в файле.
        new_string (str): Строка, на которую нужно заменить найденную
        ↪  строку.
    """
    with open(file_path, 'r') as file:
        lines = file.readlines()

    for index, line in enumerate(lines):
        if old_string in line:
            lines[index] = new_string + '\n'  # Заменяем строку
            break

    with open(file_path, 'w') as file:
        file.writelines(lines)  # Записываем изменения в файл

# Определение имени пользователя и пути
if len(sys.argv) > 1:
    if len(sys.argv) > 2:
        directory_name = ' '.join(sys.argv[1:])
```

```
46      else:
47          directory_name = sys.argv[1]
48      user_name = directory_name.split('/')[0]
49  else:
50      directory_name = 'clover/Desktop'
51      user_name = 'clover'
52
53  # Сохраняем карту ArUco в файл
54  with open('Nto.txt', 'w') as file:
55      file.write(aruco_map)
56
57  # Проверка и копирование карты ArUco
58  if 'Nto.txt' not in os.listdir(f"/home/{user_name}/catkin_ws/src/clover/aruco_pose/map"):
59      shutil.copy('Nto.txt', f"/home/{user_name}/catkin_ws/src/clover/aruco_pose/map")
60
61  # Проверка наличия моделей в директории .gazebo
62  if 'models' in os.listdir(f"/home/{user_name}/.gazebo/"):
63      for model in os.listdir(f"/home/{user_name}/.gazebo/models"):
64          # Удаляем ненужные модели
65          if model in ['aruco_Nto_txt', 'dronepoint_blue', 'dronepoint_green', 'dronepoint_red', 'dronepoint_yellow']:
66              shutil.rmtree(f"/home/{user_name}/.gazebo/models/{model}")
67              continue
68
69      # Копирование моделей в директорию .gazebo
70      for model_folder in os.listdir(f"/home/{directory_name}/II_stage_KZ/models"):
71          shutil.copytree(f'/home/{directory_name}/II_stage_KZ/models/{model_folder}', f"/home/{user_name}/.gazebo/models/{model_folder}", dirs_exist_ok=True)
72  else:
73      shutil.copytree(f'/home/{directory_name}/II_stage_KZ/models', f"/home/{user_name}/.gazebo/models")
74
75  # Изменение строк в launch файле для запуска ArUco
76  replace_string(f'/home/{user_name}/catkin_ws/src/clover/clover/launch/aruco.launch', '<arg name="aruco_detect"', '    <arg name="aruco_detect" default="true"/>')
77  replace_string(f'/home/{user_name}/catkin_ws/src/clover/clover/launch/aruco.launch', '<arg name="aruco_map"', '    <arg name="aruco_map" default="true"/>')
78  replace_string(f'/home/{user_name}/catkin_ws/src/clover/clover/launch/aruco.launch', '<arg name="aruco_vpe"', '    <arg name="aruco_vpe" default="true"/>')
79  replace_string(f'/home/{user_name}/catkin_ws/src/clover/clover/launch/aruco.launch', '<arg name="placement"', '    <arg name="placement" default="floor"/>')
80  replace_string(f'/home/{user_name}/catkin_ws/src/clover/clover/launch/aruco.launch', '<arg name="length"', '    <arg name="length" default="0.22"/>')
81  replace_string(f'/home/{user_name}/catkin_ws/src/clover/clover/launch/aruco.launch', '<arg name="map"', '    <arg name="map" default="Nto.txt"/>')
82
83  # Изменение строк в launch файле для запуска Clover
```

```
84  replace_string(f'/home/{user_name}/catkin_ws/src/clover/clover/launch
    ↪  /clover.launch', '<arg name="fcu_conn"', '    <arg name="fcu_conn"
    ↪  default="usb"/>')
85  replace_string(f'/home/{user_name}/catkin_ws/src/clover/clover/launch
    ↪  /clover.launch', '<arg name="fcu_ip"', '    <arg name="fcu_ip"
    ↪  default="127.0.0.1"/>')
86  replace_string(f'/home/{user_name}/catkin_ws/src/clover/clover/launch
    ↪  /clover.launch', '<arg name="fcu_sys_id"', '    <arg
    ↪  name="fcu_sys_id" default="1"/>')
87  replace_string(f'/home/{user_name}/catkin_ws/src/clover/clover/launch
    ↪  /clover.launch', '<arg name="gcs_bridge"', '    <arg
    ↪  name="gcs_bridge" default="tcp"/>')
88  replace_string(f'/home/{user_name}/catkin_ws/src/clover/clover/launch
    ↪  /clover.launch', '<arg name="web_video_server"', '    <arg
    ↪  name="web_video_server" default="true"/>')
89  replace_string(f'/home/{user_name}/catkin_ws/src/clover/clover/launch
    ↪  /clover.launch', '<arg name="rosbridge"', '    <arg name="rosbridge"
    ↪  default="true"/>')
90  replace_string(f'/home/{user_name}/catkin_ws/src/clover/clover/launch
    ↪  /clover.launch', '<arg name="main_camera"', '    <arg
    ↪  name="main_camera" default="true"/>')
91  replace_string(f'/home/{user_name}/catkin_ws/src/clover/clover/launch
    ↪  /clover.launch', '<arg name="optical_flow"', '    <arg
    ↪  name="optical_flow" default="true"/>')
92  replace_string(f'/home/{user_name}/catkin_ws/src/clover/clover/launch
    ↪  /clover.launch', '<arg name="aruco"', '    <arg name="aruco"
    ↪  default="true"/>')
93  replace_string(f'/home/{user_name}/catkin_ws/src/clover/clover/launch
    ↪  /clover.launch', '<arg name="rangefinder_vl53l1x"', '    <arg
    ↪  name="rangefinder_vl53l1x" default="true"/>')
94  replace_string(f'/home/{user_name}/catkin_ws/src/clover/clover/launch
    ↪  /clover.launch', '<arg name="led"', '    <arg name="led"
    ↪  default="true"/>')
95  replace_string(f'/home/{user_name}/catkin_ws/src/clover/clover/launch
    ↪  /clover.launch', '<arg name="blocks"', '    <arg name="blocks"
    ↪  default="true"/>')
96  replace_string(f'/home/{user_name}/catkin_ws/src/clover/clover/launch
    ↪  /clover.launch', '<arg name="rc"', '    <arg name="rc"
    ↪  default="false"/>')
97  replace_string(f'/home/{user_name}/catkin_ws/src/clover/clover/launch
    ↪  /clover.launch', '<arg name="force_init"', '    <arg
    ↪  name="force_init" default="true"/>')
98  replace_string(f'/home/{user_name}/catkin_ws/src/clover/clover/launch
    ↪  /clover.launch', '<arg name="simulator"', '    <arg name="simulator"
    ↪  default="false"/>')
99
100 # Изменение строк в launch файле для симулятора
101 replace_string(f'/home/{user_name}/catkin_ws/src/clover/
    ↪  clover_simulation/launch/simulator.launch', '<arg name="type"', '
    ↪  <arg name="type" default="gazebo"/>')
102 replace_string(f'/home/{user_name}/catkin_ws/src/clover/
    ↪  clover_simulation/launch/simulator.launch', '<arg name="mav_id"', '
    ↪  <arg name="mav_id" default="0"/>')
103 replace_string(f'/home/{user_name}/catkin_ws/src/clover/
    ↪  clover_simulation/launch/simulator.launch', '<arg name="est"', '
    ↪  <arg name="est" default="ekf2"/>')
104 replace_string(f'/home/{user_name}/catkin_ws/src/clover/
    ↪  clover_simulation/launch/simulator.launch', '<arg name="vehicle"', '
    ↪  <arg name="vehicle" default="clover"/>')
```

```
105 replace_string(f'/home/{user_name}/catkin_ws/src/clover/
    ↪ clover_simulation/launch/simulator.launch', '<arg
    ↪ name="main_camera"', '    <arg name="main_camera"
    ↪ default="true"/>')
106 replace_string(f'/home/{user_name}/catkin_ws/src/clover/
    ↪ clover_simulation/launch/simulator.launch', '<arg
    ↪ name="maintain_camera_rate"', '    <arg name="maintain_camera_rate"
    ↪ default="false"/>')
107 replace_string(f'/home/{user_name}/catkin_ws/src/clover/
    ↪ clover_simulation/launch/simulator.launch', '<arg
    ↪ name="rangefinder"', '    <arg name="rangefinder"
    ↪ default="true"/>')
108 replace_string(f'/home/{user_name}/catkin_ws/src/clover/
    ↪ clover_simulation/launch/simulator.launch', '<arg name="led"', '
    ↪ <arg name="led" default="true"/>')
109 replace_string(f'/home/{user_name}/catkin_ws/src/clover/
    ↪ clover_simulation/launch/simulator.launch', '<arg name="gps"', '
    ↪ <arg name="gps" default="false"/>')
110 replace_string(f'/home/{user_name}/catkin_ws/src/clover/
    ↪ clover_simulation/launch/simulator.launch', '<arg
    ↪ name="use_clover_physics"', '    <arg name="use_clover_physics"
    ↪ default="false"/>')
111 replace_string(f'/home/{user_name}/catkin_ws/src/clover/
    ↪ clover_simulation/launch/simulator.launch', '<arg name="gui"', '
    ↪ <arg name="gui" default="true"/>')
112 replace_string(f'/home/{user_name}/catkin_ws/src/clover/
    ↪ clover_simulation/launch/simulator.launch', '<arg name="world_name"
    ↪ value="$(find clover_simulation)', '        <arg name="world_name"
    ↪ value="$(find clover_simulation)/resources/worlds/Nto.world"/>')
113 replace_string(f'/home/{user_name}/catkin_ws/src/clover/
    ↪ clover_simulation/launch/simulator.launch', '<arg name="verbose"', '
    ↪ <arg name="verbose" value="true"/>')
```

Код полета. Программа на языке Python:

Python

```python
import rospy
import cv2
import math
import os
import sys
import pickle
import numpy as np
from sensor_msgs.msg import Image, CameraInfo
from std_msgs.msg import String
from geometry_msgs.msg import PointStamped, Point
from cv_bridge import CvBridge
from clover import srv
from std_srvs.srv import Trigger
import tf2_ros
import tf2_geometry_msgs
import image_geometry
from pymavlink import mavutil
from mavros_msgs.srv import CommandLong
from mavros_msgs.msg import State

# Инициализация ROS-ноды
rospy.init_node('cv', disable_signals=True)

```

```
24  # Прокси-сервисы
25  get_telemetry = rospy.ServiceProxy('get_telemetry', srv.GetTelemetry)
26  navigate = rospy.ServiceProxy('navigate', srv.Navigate)
27  set_attitude = rospy.ServiceProxy('set_attitude', srv.SetAttitude)
28  land = rospy.ServiceProxy('land', Trigger)
29  send_command = rospy.ServiceProxy('mavros/cmd/command', CommandLong)
30
31  # Публикаторы
32  buildings_pub = rospy.Publisher('/buildings', String, queue_size=1)
33
34  # Настройка TF2
35  tf_buffer = tf2_ros.Buffer()
36  tf_listener = tf2_ros.TransformListener(tf_buffer)
37  camera_model = image_geometry.PinholeCameraModel()
38  camera_model.fromCameraInfo(rospy.wait_for_message('main_camera/
    ↪  camera_info', CameraInfo))
39
40  # Мост для конвертации изображений OpenCV
41  bridge = CvBridge()
42
43  # Цветовые коды и названия моделей зданий
44  color_text = {
45      'red': '\033[31m\033[1m',
46      'green': '\033[32m\033[1m',
47      'yellow': '\033[33m\033[1m',
48      'cyan': '\033[36m\033[1m',
49      'white': '\033[37m\033[1m',
50      'blue': '\033[34m\033[1m'
51  }
52
53  colors = {
54      'red': (0, 0, 255),
55      'green': (0, 255, 0),
56      'yellow': (0, 255, 255),
57      'blue': (255, 0, 0)
58  }
59
60  model_name = {
61      'red': 'Найдено административное здание',
62      'green': 'Найдена лаборатория',
63      'yellow': 'Найден вход в шахту',
64      'blue': 'Найдено здание для обогащения угля'
65  }
66
67  # Инициализация переменных
68  buildings = ''
69  count_models = 1
70  length = 86.7
71  opr_models = {1: [], 2: [], 3: [], 4: [], 5: []}
72  img_map = None
73
74  # Попытка загрузить изображение карты
75  try:
76      img_map = cv2.imread('static/image.png')
77  except FileNotFoundError:
78      pass
79
80
81  def real_round(number: float, poz: int = 0) -> float:
82      """
```

```
    Округляет число с плавающей запятой до заданной точности.

    Аргументы:
        number (float): Число для округления.
        poz (int): Количество знаков после запятой.

    Возвращает:
        float: Округленное число.
    """
    a = number * 10 * 10 ** poz
    return math.floor(a / 10) / 10 ** poz if a % 10 < 5 else
    ↪ math.ceil(a / 10) / 10 ** poz


def check_position(x: float, y: float, color: str) -> bool:
    """
    Проверяет, является ли позиция уникальной (не слишком близкой к
    ↪ уже найденным моделям).

    Аргументы:
        x (float): Координата X.
        y (float): Координата Y.
        color (str): Цвет объекта.

    Возвращает:
        bool: True, если позиция уникальна, иначе False.
    """
    models = filter(lambda model: len(model) == 3 and model[2] ==
    ↪ color, opr_models.values())
    for model in models:
        x0, y0, _ = model
        if (x0 - x) ** 2 + (y0 - y) ** 2 <= 1:
            return False
    return True


def land_wait():
    """
    Ожидает, пока дрон не приземлится и не будет разармен.
    """
    land()
    while get_telemetry().armed:
        rospy.sleep(0.2)
    rospy.
    ↪ loginfo(f'{color_text["yellow"]}[DISARM]{color_text["white"]}')


def navigate_wait(x: float = 0, y: float = 0, z: float = 0, yaw: float
↪ = float('nan'), speed: float = 0.6,
                  frame_id: str = 'aruco_map', auto_arm: bool = False,
                   ↪ tolerance: float = 0.2):
    """
    Навигация дрона в указанную точку и ожидание достижения цели.

    Аргументы:
        x (float): Координата X цели.
        y (float): Координата Y цели.
        z (float): Координата Z (высота).
        yaw (float): Угол поворота дрона.
        speed (float): Скорость дрона.
```

```
137         frame_id (str): Система координат.
138         auto_arm (bool): Нужно ли армировать дрон.
139         tolerance (float): Допустимое отклонение от цели.
140     """
141     navigate(x=x, y=y, z=z, yaw=yaw, speed=speed, frame_id=frame_id,
        ↪ auto_arm=auto_arm)
142     if auto_arm:
143         rospy.
            ↪ loginfo(f'{color_text["yellow"]}[ARM]{color_text["white"]}')
144         rospy.loginfo(f'Takeoff by
            ↪ {color_text["green"]}Body{color_text["white"]} Z = {z}')
145     else:
146         rospy.loginfo(f'Flight to {color_text["cyan"]}{frame_id.
            ↪ capitalize()}{color_text["white"]}')
147 
148     while not rospy.is_shutdown():
149         try:
150             telem = get_telemetry(frame_id='navigate_target')
151         except rospy.ServiceException:
152             continue
153 
154         if math.sqrt(telem.x ** 2 + telem.y ** 2 + telem.z ** 2) <
            ↪ tolerance:
155             break
156         rospy.sleep(0.2)
157 
158 
159 def img_xy_to_point(xy: tuple, dist: float) -> Point:
160     """
161     Преобразует координаты пикселей на изображении в 3D мировые
        ↪ координаты.
162 
163     Аргументы:
164         xy (tuple): Координаты пикселя (X, Y).
165         dist (float): Расстояние от камеры до объекта.
166 
167     Возвращает:
168         Point: 3D-координаты объекта.
169     """
170     xy_rect = camera_model.rectifyPoint(xy)
171     ray = camera_model.projectPixelTo3dRay(xy_rect)
172     return Point(x=ray[0] * dist, y=ray[1] * dist, z=dist)
173 
174 
175 def get_center_of_mass(mask: np.ndarray) -> tuple:
176     """
177     Вычисляет центр масс бинарной маски.
178 
179     Аргументы:
180         mask (np.ndarray): Бинарная маска, где пиксели, принадлежащие
            ↪ объекту, равны 1.
181 
182     Возвращает:
183         tuple: Координаты центра масс (X, Y).
184     """
185     M = cv2.moments(mask)
186     if M['m00'] == 0:
187         return None
188     return M['m10'] // M['m00'], M['m01'] // M['m00']
189 
```

```
190
191 def color_coord(mask: np.ndarray, color_name: str, msg) -> None:
192     """
193     Обрабатывает найденные цветные объекты, вычисляет их координаты и
        ↪ обновляет карту и лог.
194
195     Аргументы:
196         mask (np.ndarray): Бинарная маска цвета.
197         color_name (str): Название цвета объекта.
198         msg: Сообщение с заголовком и данными.
199     """
200     global count_models, buildings, img_map
201     xy = get_center_of_mass(mask)
202     if xy is None:
203         return
204     try:
205         altitude = get_telemetry('terrain').z
206     except rospy.ServiceException:
207         return
208     xy3d = img_xy_to_point(xy, altitude)
209     target = PointStamped(header=msg.header, point=xy3d)
210     try:
211         setpoint = tf_buffer.transform(target, 'aruco_map',
            ↪ timeout=rospy.Duration(0.2))
212     except rospy.ServiceException as e:
213         rospy.logerr(f'Transform error: {e}')
214         return
215
216     if check_position(setpoint.point.x, setpoint.point.y, color_name):
217         buildings += f'{model_name[color_name]}: X =
            ↪ {real_round(setpoint.point.x, 2)} Y =
            ↪ {real_round(setpoint.point.y, 2)} color = {color_name}\n'
218         with open('static/otchet.txt', 'a') as file:
219             file.write(f'{model_name[color_name]}:\n X =
                ↪ {real_round(setpoint.point.x, 2)} Y =
                ↪ {real_round(setpoint.point.y, 2)} color =
                ↪ {color_name}\n\n')
220
221         rospy.loginfo(f'{color_text["red"]}[DEBUG]:
            ↪ {model_name[color_name]}:
            ↪ {color_text["yellow"]}{color_name} {color_text["white"]}X
            ↪ = {real_round(setpoint.point.x, 2)} Y =
            ↪ {real_round(setpoint.point.y, 2)}')
222
223         opr_models[count_models] = [setpoint.point.x,
            ↪ setpoint.point.y, color_name]
224         with open('static/coord_model.pkl', 'wb') as file_pkl:
225             pickle.dump(opr_models, file_pkl)
226
227         x, y, color = opr_models[count_models]
228         bgr_color = colors[color]
229         cv2.rectangle(img_map,
230                       (108 + int(real_round((x - 0.3) * length)), 892
                          ↪ - int(real_round((y - 0.3) * length))),
231                       (112 + int(real_round((x + 0.3) * length)), 888
                          ↪ - int(real_round((y + 0.3) * length))),
232                       (0, 0, 0), -1)
233         cv2.rectangle(img_map,
234                       (110 + int(real_round((x - 0.3) * length)), 890
                          ↪ - int(real_round((y - 0.3) * length))),
```

```
235                         (110 + int(real_round((x + 0.3) * length)), 890
                            ↪  - int(real_round((y + 0.3) * length))),
236                         bgr_color, -1)
237         cv2.putText(img_map, f'color = {color_name}',
238                     (90 + int(real_round((x - 0.3) * length)), 905 -
                        ↪  int(real_round((y - 0.3) * length))),
239                     cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0, 0, 0), 5)
240         cv2.putText(img_map, f'color = {color_name}',
241                     (90 + int(real_round((x - 0.3) * length)), 905 -
                        ↪  int(real_round((y - 0.3) * length))),
242                     cv2.FONT_HERSHEY_SIMPLEX, 0.5, bgr_color, 2)
243         cv2.putText(img_map, f'X = {real_round(setpoint.point.x, 2)}',
244                     (100 + int(real_round((x - 0.3) * length)), 920 -
                        ↪  int(real_round((y - 0.3) * length))),
245                     cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0, 0, 0), 5)
246         cv2.putText(img_map, f'X = {real_round(setpoint.point.x, 2)}',
247                     (100 + int(real_round((x - 0.3) * length)), 920 -
                        ↪  int(real_round((y - 0.3) * length))),
248                     cv2.FONT_HERSHEY_SIMPLEX, 0.5, bgr_color, 2)
249         cv2.putText(img_map, f'Y = {real_round(setpoint.point.y, 2)}',
250                     (100 + int(real_round((x - 0.3) * length)), 935 -
                        ↪  int(real_round((y - 0.3) * length))),
251                     cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0, 0, 0), 5)
252         cv2.putText(img_map, f'Y = {real_round(setpoint.point.y, 2)}',
253                     (100 + int(real_round((x - 0.3) * length)), 935 -
                        ↪  int(real_round((y - 0.3) * length))),
254                     cv2.FONT_HERSHEY_SIMPLEX, 0.5, bgr_color, 2)
255
256         cv2.imwrite('static/image.png', img_map)
257         count_models += 1
258
259
260 shutdown_initiated = True
261
262
263 @long_callback
264 def image_callback(msg):
265     """
266     Обрабатывает входящее изображение от камеры, определяет объекты по
        ↪  цвету
267     и обновляет карту с координатами.
268
269     Аргументы:
270         msg: Сообщение с изображением от камеры.
271     """
272     global buildings, shutdown_initiated
273     if shutdown_initiated:
274         return
275
276     img = bridge.imgmsg_to_cv2(msg, 'bgr8')
277     hsv = cv2.cvtColor(img, cv2.COLOR_BGR2HSV)
278
279     blue = cv2.inRange(hsv, (115, 150, 150), (125, 255, 255))
280     green = cv2.inRange(hsv, (55, 150, 150), (65, 255, 255))
281     red = cv2.inRange(hsv, (0, 150, 150), (5, 255, 255))
282     yellow = cv2.inRange(hsv, (25, 150, 150), (35, 255, 255))
283
284     if cv2.countNonZero(blue) > 10:
285         color_coord(mask=blue, color_name='blue', msg=msg)
286     elif cv2.countNonZero(green) > 10:
```

```
287             color_coord(mask=green, color_name='green', msg=msg)
288         elif cv2.countNonZero(red) > 10:
289             color_coord(mask=red, color_name='red', msg=msg)
290         elif cv2.countNonZero(yellow) > 10:
291             color_coord(mask=yellow, color_name='yellow', msg=msg)
292 
293         if buildings:
294             buildings_pub.publish(data=buildings)
295 
296         rospy.sleep(0.3)
297 
298 
299 def main():
300     """
301     Основная функция для навигации дрона и выполнения задач.
302     """
303     global count_models, shutdown_initiated, img_map, opr_models
304 
305     # Проверяем, если есть сохраненные данные о стадии или координатах
        ↪ моделей
306     if os.path.exists('static/stage.pkl'):
307         with open('static/stage.pkl', 'rb') as file:
308             stage_nav = pickle.load(file)
309         if os.path.exists('static/coord_model.pkl'):
310             with open('static/coord_model.pkl', 'rb') as file_pkl:
311                 opr_models = pickle.load(file_pkl)
312     else:
313         stage_nav = 0
314         img_map = cv2.resize(bridge.imgmsg_to_cv2(rospy.
            ↪ wait_for_message('aruco_map/image', Image), 'bgr8'), (1000,
            ↪ 1000))
315         cv2.imwrite('static/image.png', img_map)
316         try:
317             os.remove('static/otchet.txt')
318         except FileNotFoundError:
319             pass
320         with open('static/otchet.txt', 'a') as file:
321             file.write('\n')
322 
323     # Навигация к стартовой точке
324     navigate_wait(z=1.5, frame_id='body', auto_arm=True, speed=0.5)
325     shutdown_initiated = False
326     for stage, aruco_id in enumerate([9, 19, 10, 20, 29, 39, 30, 40,
        ↪ 49, 59, 50, 60, 69, 79, 70, 80, 89, 99, 90][stage_nav:],
        ↪ start=stage_nav):
327         navigate_wait(z=1.5, frame_id=f'aruco_{aruco_id}')
328         with open('static/stage.pkl', 'wb') as file:
329             pickle.dump(stage + 1, file)
330 
331     # Возврат к начальной точке
332     navigate_wait(z=1.5, frame_id='aruco_0', speed=0.7, tolerance=0.3,
        ↪ yaw=0)
333     navigate_wait(z=0.2, frame_id='aruco_0', speed=0.3, tolerance=0.15,
        ↪ yaw=0)
334     land_wait()
335 
336     # Очистка временных файлов
337     os.remove('static/stage.pkl')
338     os.remove('static/coord_model.pkl')
339     shutdown_initiated = True
```

```
    rospy.sleep(0.3)
    exit()


if __name__ == "__main__":
    # Запуск программы
    if '--land' not in sys.argv and '--kill' not in sys.argv:
        main()
        rospy.spin()
    elif '--land' in sys.argv and '--kill' not in sys.argv:
        land_wait()
        rospy.loginfo(f'{color_text["red"]}[LAND]{color_text["white"]}')
        exit()
    elif '--kill' in sys.argv and '--land' not in sys.argv:
        if get_telemetry().armed:
            set_attitude(thrust=0)
        else:
            rospy.loginfo(f'{color_text["red"]}Дрон не заармлен{color_text["white"]}')
        rospy.loginfo(f'{color_text["red"]}[KILL]{color_text["white"]}')
        exit()
```

Веб-приложение. Код решения:

Html

```
<!DOCTYPE html>
<html>
  <head>
    <title>Control Panel</title>
    <style>
      body {
        background-color: #1c1c1c;
        color: #f0f0f0;
        font-family: Arial, sans-serif;
        margin: 0;
        display: flex;
        justify-content: center;
        align-items: center;
        height: 100vh;
      }

      /* Основной контейнер */
      .container {
        display: flex;
        flex-direction: row;
        align-items: center;
        gap: 20px;
        width: 95%;
        justify-content: space-between;
      }

      /* Кнопки*/
      .button-container {
        display: flex;
        flex-direction: column;
        gap: 70px;
        align-items: flex-start;
```

```
33          margin-left: 100px;
34        }
35
36        /* Вид кнопок */
37        .rounded-button {
38          padding: 35px 70px;
39          font-size: 32px;
40          color: #fff;
41          background-color: #333;
42          border: none;
43          border-radius: 20px;
44          cursor: pointer;
45          width: 350px;
46          text-align: center;
47          transition: all 0.3s ease;
48        }
49
50        /* Анимация наведения на кнопку*/
51        .rounded-button:not(:disabled):hover {
52          background-color: #555;
53          transform: scale(1.1);
54        }
55
56        /* Вид неактивных кнопок */
57        .rounded-button:disabled {
58          background-color: #666;
59          cursor: not-allowed;
60          opacity: 0.5;
61        }
62
63        /* Изображения */
64        .image-container img {
65          max-width: 800px;
66          border-radius: 20px;
67          border: 5px solid #333;
68          box-shadow: 0 8px 20px rgba(0, 0, 0, 0.5);
69          margin-right: 100px;
70        }
71
72        /* Текстовое окно для buildings */
73        .text-container {
74          display: flex;
75          flex-direction: column;
76          gap: 20px;
77          align-items: flex-start;
78          margin-left: 10px;
79          margin-right: 10px;
80        }
81
82        /* Вид этого же окна*/
83        .text-box {
84          padding: 10px;
85          font-size: 20px;
86          font-weight: bold;
87          color: #fff;
88          background-color: #333;
89          border: none;
90          border-radius: 20px;
91          width: 350px;
92          height: 440px;
```

```
93          overflow-y: auto;
94          box-shadow: 0 4px 10px rgba(0, 0, 0, 0.5);
95          white-space: pre-wrap;
96          text-align: center;
97        }
98        /* Подпись */
99        .signature {
100         position: absolute;
101         bottom: 10px;
102         left: 10px;
103         font-size: 15px;
104         color: #888;
105       }
106     </style>
107   </head>
108   <body>
109     <div class="container">
110       <div class="button-container">
111         <button class="rounded-button" id="start">START</button>
112         <button class="rounded-button" id="stop"
            ↪  disabled>STOP</button>
113         <button class="rounded-button" id="kill">KILL SWITCH</button>
114       </div>
115       <div class="text-container">
116         <div class="text-box" id="buildings-text"></div>
117       </div>
118       <div class="image-container">
119         <img src="image.png" alt="Control Image" />
120       </div>
121       <div class="signature">by Obradovich Vlad</div>
122     </div>
123     <script>
124       document.addEventListener("DOMContentLoaded", () => {
125         const startButton = document.getElementById("start");
126         const stopButton = document.getElementById("stop");
127         const killButton = document.getElementById("kill");
128         const imageElement = document.querySelector(".image-container
            ↪  img");
129         const buildingsTextElement =
            ↪  document.getElementById("buildings-text");
130
131         const baseUrl = "/api";
132
133         const updateButtons = (response) => {
134           startButton.disabled = !response.start;
135           stopButton.disabled = !response.stop;
136           killButton.textContent = `KILL SWITCH`;
137         };
138
139         const sendRequest = async (button) => {
140           try {
141             const response = await fetch(`${baseUrl}/${button}`, {
142               method: "POST",
143               headers: {
144                 "Content-Type": "application/json",
145               },
146             });
147             if (!response.ok) {
148               throw new Error(`HTTP error! status:
                  ↪  ${response.status}`);
```

```
149             }
150             const data = await response.json();
151             updateButtons(data);
152           } catch (error) {
153             console.error("Error:", error);
154           }
155         };
156 
157         startButton.addEventListener("click", () => {
158           if (!startButton.disabled) sendRequest("start");
159         });
160 
161         stopButton.addEventListener("click", () => {
162           if (!stopButton.disabled) sendRequest("stop");
163         });
164 
165         killButton.addEventListener("click", () =>
            ↪  sendRequest("kill"));
166 
167         // Получение начального состояния
168         fetch(`${baseUrl}/state`)
169           .then((response) => response.json())
170           .then(updateButtons)
171           .catch((error) => console.error("Error:", error));
172 
173         // Функция для обновления изображения
174         const updateImage = () => {
175           const timestamp = new Date().getTime();
176           imageElement.src = `image.png?t=${timestamp}`;
177         };
178 
179         //  Обновление изображения каждые 0.2 сек
180         setInterval(updateImage, 200);
181 
182         // Функция для обновления текста
183         const updateVariableText = async () => {
184           try {
185             const response = await fetch(`${baseUrl}/buildings_text`,
                ↪  {
186               method: "GET",
187               headers: {
188                 "Content-Type": "application/json",
189               },
190             });
191             if (!response.ok) {
192               throw new Error(`HTTP error! status:
                  ↪  ${response.status}`);
193             }
194             const data = await response.json();
195             buildingsTextElement.innerHTML = data.text.replace(/\n/g,
                ↪  "<br>");
196           } catch (error) {
197             console.error("Error:", error);
198           }
199         };
200 
201         // Обновление текста каждые 0.2 сек
202         setInterval(updateVariableText, 200);
203       });
204     </script>
```

```
205     </body>
206   </html>
```

4. Заключительный этап

4.1. Работа наставника НТО при подготовке к этапу

На этапе подготовки к заключительному этапу НТО наставник решает две важные задачи: помощь участникам в подготовке к предстоящим соревнованиям и формирование устойчивой и слаженной команды. Заключительный этап требует высокой слаженности, уверенности и глубоких знаний, и наставник становится тем, кто объединяет усилия участников и направляет их в нужное русло.

Наставник помогает участникам:

- разобрать задания прошлых лет, используя официальные сборники, чтобы понять структуру финальных испытаний, типы задач и ожидаемый уровень сложности;
- изучить организационные особенности заключительного этапа, включая формат проведения, регламент, продолжительность и технические нюансы;
- спланировать подготовку — на основе даты начала финала составляется четкий график занятий, в котором распределены темы, практикумы и командные тренировки;
- обратиться (при необходимости) за консультацией к разработчикам заданий по профилю, уточнить, на какие аспекты подготовки следует обратить особое внимание, и получить дополнительные материалы.

Также рекомендуется участие в мероприятиях от организаторов, таких как:

- установочные вебинары и открытые разборы задач;
- хакатоны, практикумы и мастер-классы для финалистов;
- встречи в онлайн-формате, информация о которых публикуется в группе НТО во «ВКонтакте» и в телеграм-чатах профилей.

Наставнику необходимо уделить внимание работе на формированием устойчивой, продуктивной и мотивированной команды:

- **Сплочение команды.** Это особенно актуально, если участники живут в разных городах. Регулярные онлайн-встречи, совместная работа над задачами и неформальное общение помогают наладить доверие и улучшить командную динамику.
- **Анализ ролей.** Наставник вместе с командой определяет, кто за что отвечает, какие задачи входят в зону ответственности каждого участника. Также обсуждаются возможности взаимозаменяемости на случай непредвиденных ситуаций.
- **Оценка компетенций.** Важно определить, какими знаниями и навыками уже обладают участники, а какие необходимо развить. На основе этого формируется индивидуальный и командный план подготовки.

- **Участие в подготовительных мероприятиях от разработчиков профилей.** Перед заключительным этапом проводятся установочные вебинары, разборы задач прошлых лет, практикумы, мастер-классы для финалистов. Информация о таких мероприятиях публикуется в группе НТО в VK и в чатах профилей в Telegram.
- **Практика в формате хакатонов.** Наставник может организовать дистанционные хакатоны или практикумы с использованием заданий прошлых лет и методических рекомендаций из официальных сборников.

Таким образом, наставник становится координатором и моральной опорой команды, помогая пройти заключительный этап НТО с максимальной уверенностью и результатом.

4.2. Предметный тур

Задачи третьего этапа предметного тура профиля по информатике открыты для решения. Участие в соревновании доступно на платформе Яндекс.Контест: https://contest.yandex.ru/contest/72668/enter/.

4.2.1. Информатика. 8–11 классы

Задача 4.2.1.1. Совместный полет (8 баллов)

Имя входного файла: стандартный ввод или `input.txt`.

Имя выходного файла: стандартный вывод или `output.txt`.

Ограничение по времени выполнения программы: 1 с.

Ограничение по памяти: 64 Мбайт.

Условие

Алиса пишет программу для шоу дронов. Одним из номеров будет совместный полет $n \times m$ дронов, построенных в n рядов, по m дронов в каждом. Для создания большего эффекта Алиса хочет, чтобы расстояния между соседними дронами в одном ряду и между соседними рядами совпадали и были минимально допустимыми. Но она опасается допустить столкновение некоторых дронов, поэтому намерена распределить все дроны по 4 коридорам по высоте так, чтобы соседние дроны занимали разные коридоры.

Схематично можно представить расположение дронов в виде таблицы из n строк и m столбцов, где в каждой ячейке будет находиться число от 1 до 4, обозначающее номер коридора для конкретного дрона. Алиса хочет распределить коридоры так, чтобы соседние дроны по **вертикали**, **горизонтали** или **диагонали** занимали различные коридоры.

Напишите программу, которая найдет любое размещение дронов с заданным условием.

Формат входных данных

На вход поступает два целых числа n, m — размеры таблицы, $(1 \leqslant n, m \leqslant 10)$.

Формат выходных данных

Программа должна вывести таблицу требуемого размера, состоящую из чисел от 1 до 4.

Примеры

Стандартный ввод
3 4

Стандартный вывод
1 4 3 2 3 2 1 4 1 4 3 2

Решение

Эта задача допускает много возможных решений. Например, можно чередовать в строках с четными номерами числа 1 и 2, а в строках с нечетными номерами числа 3 и 4.

Пример программы-решения

Ниже представлено решение на языке Python.

Python

```python
n, m = map(int, input().split())
for i in range(n):
    if i % 2 == 0:
        print(*([1,2]*5)[:m])
    else:
        print(*([3,4]*5)[:m])
```

Критерии оценивания

Тест № 1 совпадает с тестом из условия задачи. Баллы за него не начисляются.

Успешное прохождение каждого теста №№ 2–9 оценивается в один балл.

Задача 4.2.1.2. Дроны и горностаевая моль (16 баллов)

Имя входного файла: стандартный ввод или `input.txt`.

Имя выходного файла: стандартный вывод или `output.txt`.

Ограничение по времени выполнения программы: 2 с.

Ограничение по памяти: 128 Мбайт.

Условие

В городе Энске есть большой парк прямоугольной формы. Некоторые деревья в нем нужно обработать от вредителей — горностаевой моли. Известны координаты на плоскости всех деревьев, нуждающихся в обработке. Специальный дрон будет

двигаться от одного дерева к другому и обрабатывать их инсектицидными аэрозолями. Но для дрона необходимо составить маршрут движения, по возможности близкий к оптимальному.

Было решено использовать следующий алгоритм. Зададим координаты так, что левый нижний угол парка будет находиться в точке $(0,\ 0)$, а правый верхний — в точке $(w,\ h)$. Разделим получившийся прямоугольник на вертикальные полосы шириной s. Последняя полоса может иметь меньшую ширину. Пронумеруем все полосы числами от нуля. Будем считать, что точка принадлежит полосе с номером k, если для ее координаты x выполняется неравенство $ks \leqslant x < (k+1)s$. Сначала дрон обработает все деревья, попавшие в нулевую полосу, потом — в первую и так далее.

Двигаясь по полосам с четными номерами, дрон будет обрабатывать деревья в порядке возрастания их координаты y. Если же номер полосы будет нечетным, то дрон будет двигаться в порядке убывания координаты y. Если координаты y у двух деревьев в пределах одной полосы совпадут, то дрон будет обрабатывать их в порядке возрастания координаты x вне зависимости от четности номера полосы.

Обратите внимание, что алгоритм будет исполняться формально, то есть на его исполнение, например, не окажет влияние отсутствие деревьев в некоторых полосах или то, что последняя полоса будет иметь ширину 1. Для лучшего понимания алгоритма движения посмотрите на схему ниже. Здесь пять полос, в нулевую полосу попали все деревья с координатой x в полуоткрытом интервале $[0; 5)$. Они обходятся снизу вверх. В первую и третью полосу попали деревья с координатой x из полуоткрытых интервалов $[5; 10)$ и $[15; 20)$. Они обходятся сверху вниз. Последняя полоса содержит только деревья с координатой $x = 20$.

Напишите программу, которая по заданным координатам деревьев найдет длину пути дрона от первого обработанного дерева до последнего.

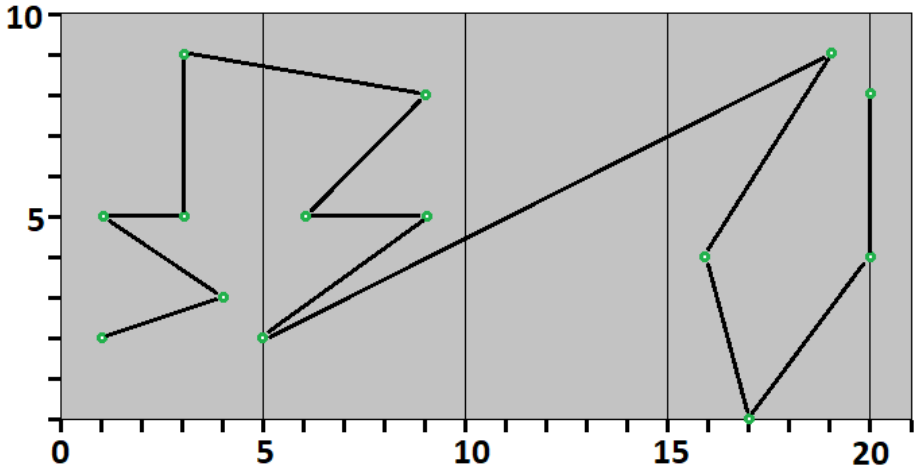


Рис. 4.2.1

Формат входных данных

В первой строке на вход программы поступают три целых числа w, h, s — размеры парка по оси Ox, по оси Oy и ширина полосы $(1 \leqslant w, h, s \leqslant 10\,000)$.

Во второй строке на вход программы поступает одно натуральное число n — количество деревьев в парке, $2 \leqslant n \leqslant 200\,000$.

Далее в n строках записаны по два числа x_i и y_i — координаты на плоскости каждого из n деревьев, $0 \leqslant x_i < w$, $0 \leqslant y_i < h$.

Формат выходных данных

Выведите одно вещественное число — расстояние, пройденное дроном от первого дерева в маршруте до последнего. Ответ засчитывается верным, если он будет отличаться от точного значения не более, чем на 10^{-3}.

Примечания

Маршрут для первого теста изображен на рис. 4.2.1.

Примеры

Пример №1

Стандартный ввод
21 10 5 14 5 2 20 4 9 5 6 5 20 8 17 0 9 8 1 5 19 9 3 5 3 9 1 2 4 3 16 4

Стандартный вывод
65.69976551601135

Решение

Для решения этой задачи требуется упорядочить координаты деревьев в заданной последовательности. В Python для этого можно использовать встроенный алгоритм сортировки с параметром `key`, задающим порядок. Сохраним координаты деревьев в список кортежей, где нулевой элемент кортежа будет задавать координату x, а первый — y.

Номер полосы можно вычислить по формуле $\frac{x}{s}$. В момент сортировки координату y в полосах с нечетным номером можно умножить на (-1), чтобы обеспечить порядок по убыванию. Таким образом сформируем ключ сортировки в виде кортежа с тремя параметрами: номер полосы, координата y, домноженная на (-1) в нечетных полосах, координата x.

Два кортежа сравниваются по нулевому элементы, в случае равенства — по первому, в случае равенства первых элементов — по второму. Это обеспечивает требуемый порядок сортировки.

В конце программы требуется просто посчитать сумму расстояний между соседними деревьями.

Пример программы-решения

Ниже представлено решение на языке Python.

Python

```python
w, h, s = map(int,input().split())
n = int(input())
p = sorted([tuple(map(int,input().split())) for _ in range(n)],\
    key = lambda p:(p[0]//s,p[1]*(-1 if (p[0]//s)%2==1 else 1),p[0]))
ans = 0.
for i in range(1,len(p)):
    ans+=((p[i][0]-p[i-1][0])**2+(p[i][1]-p[i-1][1])**2)**0.5
print(ans)
```

Критерии оценивания

Тест № 1 совпадает с тестом из условия задачи. Баллы за него не начисляются.

Успешное прохождение каждого теста №№ 2–17 оценивается в один балл.

В тестах №№ 2–3 выполняется $s = w$, то есть в этих тестах ровно одна полоса.

В тестах №№ 4–5 выполняется $w/2 \leqslant s < w$, то есть в этих тестах ровно две полосы.

В тестах №№ 2–9 все координаты y в пределах одной полосы различны.

В тестах №№ 2–13 количество деревьев не превосходит 10 000.

Задача 4.2.1.3. Совместный полет 2 (20 баллов)

Имя входного файла: стандартный ввод или `input.txt`.

Имя выходного файла: стандартный вывод или `output.txt`.

Ограничение по времени выполнения программы: 1 с.

Ограничение по памяти: 128 Мбайт.

Условие

Рассмотрим координатную плоскость. На оси Ox в точках с координатами $(1,\ 0)$, $(2,\ 0)$, $\ldots$, $(n,\ 0)$ в воздухе на некоторой высоте неподвижно висит n дронов. Даны n целых чисел $t_1, t_2, ..., t_n$, задающих некоторые моменты времени. Дрон, изначально находящийся в точке $(i, 0)$, стартует в момент времени t_i и двигается параллельно оси Oy в сторону увеличения координаты y.

Аналогично на оси Oy в воздухе на другой высоте расположены еще m дронов в точках с координатами $(0, 1)$, $(0, 2)$, $\ldots$, $(0, m)$. Для них также заданы моменты времени $q_1, q_2, \ldots, q_m$. Дрон, изначально находящийся в точке с координатами $(0, j)$, стартует в момент времени q_j и двигается параллельно оси Ox в сторону увеличения координаты x.

Все дроны летят с одинаковыми скоростями. Если единицу расстояния считать метром, а единицу времени — секундой, то все скорости будут равны 1 м/с.

Поскольку дроны летят на различных высотах, столкновений не произойдет, однако будем считать событием ситуацию, когда один дрон пролетит строго над другим в один и тот же момент времени.

Рассмотрим пример на рис. 4.2.2. С оси Ox стартуют четыре дрона в моменты времени $0, 3, 1, 2$. С оси Oy стартует три дрона в моменты времени 2, 0, 4.

Два дрона пролетят над точкой $(2, 1)$ в момент времени 4; над точкой $(3, 2)$ в момент времени 3; над точкой $(4, 2)$ в момент времени 4; над точкой $(2, 3)$ в момент времени 6. Ни в какой другой точке два дрона одновременно не окажутся.

Напишите программу, которая посчитает количество указанных событий.

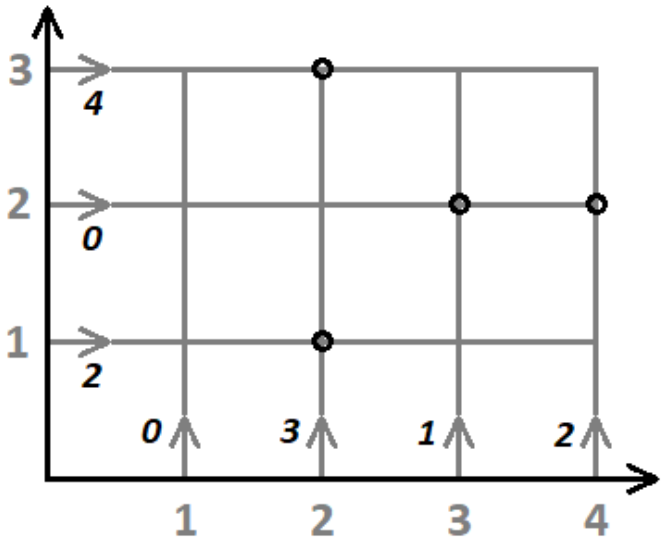


Рис. 4.2.2

Формат входных данных

В первой строке на вход поступают два натуральных числа n и m — количество дронов на оси Ox и на оси Oy соответственно, $1 \leqslant n, m \leqslant 100\,000$.

Во второй и в третьей строках на вход поступают целые числа $t_1, \ldots, t_n$ и $q_1, \ldots, q_m$ — моменты начала движения для дронов, ожидающих на оси Ox и на оси Oy соответственно, $0 \leqslant t_i, q_i \leqslant 10^9$.

Формат выходных данных

Выведите одно целое число — количество событий указанного вида.

Примеры

Пример №1

Стандартный ввод
4 3 0 3 1 2 2 0 4

Стандартный вывод
4

Примечания

Тест из условия задачи разобран в примере.

Решение

Из условия задачи следует, что два дрона, стартующих из точек $(i, 0)$ и $(0, j)$, могут оказаться одновременно в точке (i, j). Первый дрон окажется там в момент времени $t_i + j$, а второй — $t_j + i$. Таким образом, требуется посчитать количество пар (i, j), для которых выполняется равенство $t_i + j = t_j + i$. Решение, перебирающее все пары двумя вложенными циклами, сможет набрать не более 12 баллов. Чтобы получить полное решение, перепишем равенство в виде $t_i - i = t_j - j$. Найдем все значения $k_i = t_i - i$ и составим словарь из пар $k_i : c(k_i)$, где $c(k_i)$ — количество найденных чисел k_i.

Теперь переберем отдельным циклом все значения q_j и просуммируем все значения $c(q_j - j)$ из словаря.

Пример программы-решения

Ниже представлено решение на языке Python.

Python

```python
n, m = map(int,input().split())
cnt = dict()
for i,t in enumerate(map(int,input().split())):
    if not t-i in cnt.keys():
        cnt[t-i]=0
```

```
    cnt[t-i]+=1
ans = 0
for j,q in enumerate(map(int,input().split())):
    if q-j in cnt.keys():
        ans+=cnt[q-j]
print(ans)
```

Критерии оценивания

Тест № 1 совпадает с тестом из условия задачи. Баллы за него не начисляются.

Успешное прохождение каждого теста №№ 2–41 оценивается в 0,5 балла.

В тестах №№ 2–13 n, m и все значения времени не превосходят 100.

В тестах №№ 14–25 n, m и все значения времени не превосходят 2 000.

Задача 4.2.1.4. Ошибка в программе (28 баллов)

Имя входного файла: стандартный ввод или `input.txt`.

Имя выходного файла: стандартный вывод или `output.txt`.

Ограничение по времени выполнения программы: 2 с.

Ограничение по памяти: 256 Мбайт.

Условие

Фермер Василий попал в очень неприятную ситуацию. Он доверил написать программу для обработки своего поля двумя сельскохозяйственными дронами одному болтливому попугаю, и тот все напутал.

Поле Василия имеет прямоугольную форму и условно разделено на квадратные клетки. Будем считать, что каждая клетка имеет размер 1 м, а поле имеет размер $n \times m$. Можно сказать, что поле состоит из n горизонтальных и m вертикальных рядов. Все ряды пронумерованы, начиная с единицы, горизонтальные — снизу вверх, вертикальные — слева направо.

Программа, написанная болтливым попугаем, заключалась в следующем. Первый дрон двигался по горизонтальным рядам и обрабатывал клетки удобрениями, но из-за ошибки в программе он посетил не все клетки в каждом ряду. Более точно: в i-м ряду дрон обработал ровно a_i идущих подряд клеток, начиная с западного края поля. А второй дрон двигался по вертикальным рядам и в i-м ряду обработал ровно b_i идущих подряд клеток, начиная с южного края поля. В результате некоторые клетки были обработаны дважды, а некоторые — ни разу. Теперь Василий должен как-то исправить ситуацию, но для начала ему надо понять, сколько клеток всего осталось необработанными.

Для лучшего понимания проблемы посмотрите пример на рис. 4.2.3. Числа на схеме задают номера рядов. $\tilde{a} = (1,\ 6,\ 0,\ 5,\ 7)$, $\tilde{b} = (2,\ 4,\ 0,\ 5,\ 1,\ 3,\ 0,\ 4)$. Клетки, обработанные первым дроном, обозначены горизонтальными прямоугольниками, а вторым — вертикальными. Как видим, восемь клеток были обработаны дважды, а десять — ни разу.

Напишите программу, которая найдет количество клеток поля, которые не были обработаны ни одним из двух дронов.

Обратите внимание, что в этой задаче есть несколько групп тестов, для которых можно реализовать более простые алгоритмы решения, чем в общем случае.

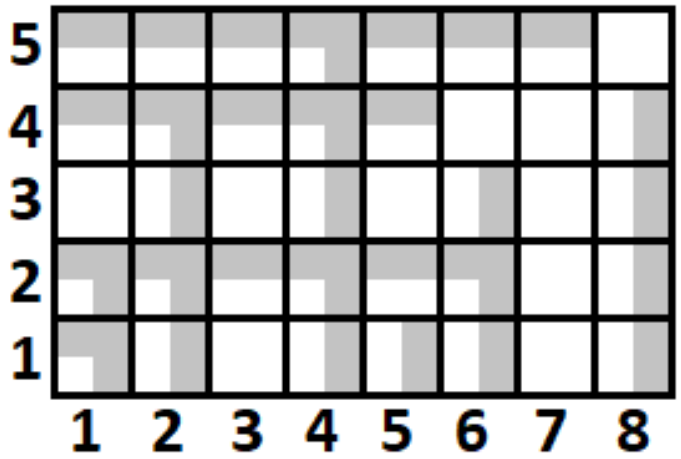


Рис. 4.2.3

Формат входных данных

В первой строке на вход поступают два натуральных числа n и m — размеры поля, $1 \leqslant n, m \leqslant 200\,000$.

Во второй строке записаны числа $a_1, \ldots, a_n$ — количество обработанных клеток в каждом из горизонтальных рядов, $0 \leqslant a_i \leqslant m$.

В третьей строке записаны числа $b_1, \ldots, b_m$ — количество обработанных клеток в каждом из вертикальных рядов, $0 \leqslant b_i \leqslant n$.

Формат выходных данных

Программа должна вывести одно целое число — количество необработанных клеток поля.

Примеры

Пример №1

Стандартный ввод
5 8 1 6 0 5 7 2 4 0 5 1 3 0 4

Стандартный вывод
10

Примечания

Тест из условия задачи разобран в примере.

Решение

Попробуем записать решение, связанное с перебором столбцов поля и подсчетом необработанных клеток в каждом столбце. Рассмотрим столбец с номером j. Двигаясь по этому столбцу, дрон обработал клетки с номерами от 1 до b_j. Таким образом, надо узнать, сколько клеток с номерами от b_j+1 до n не были обработаны, когда дрон двигался по строкам. Формально это означает, что требуется определить, сколько из чисел $a_{b_j+1}, \ldots, a_n$ удовлетворяют неравенству $a_i < j$. Если размеры поля невелики, то это можно сделать перебором. Если числа a_i упорядочены, то можно использовать бинарный поиск. Для решения задачи в общем случае потребуются более сложные структуры данных, такие как деревья.

Будем перебирать пары (j,b_j) в порядке убывания b_j. Для этого их можно сначала отсортировать. В начале каждого шага цикла будем добавлять в дерево элементы списка a так, чтобы в нем находились все числа $a_{b_j+1}, \ldots, a_n$. Из упорядоченности b_j следует, что потребуется не более n операций добавления. Теперь, поскольку рассматривается j-й столбец, надо найти и прибавить к ответу количество элементов дерева, которые строго меньше j.

Для решения задачи можно использовать бинарное сбалансированное дерево с подсчетом числа узлов, дерево Фенвика или дерево отрезков. Дополнительную информацию об этих структурах данных можно найти по ссылке http://e-maxx.ru/algo/.

Пример программы-решения

Ниже представлено решение на языке Python.

Python

```
n, m =map(int,input().split())
a = [int(t) for t in input().split()]
blst = sorted(list(enumerate(map(int,input().split()))),
       key = lambda x:-x[1])
tree = [0]*(2*(m+1))
i = n
ans = 0
for b in blst:
    while i>b[1]:
        i-=1
        pos = a[i]+m+1
        while pos>0:
            tree[pos]+=1
            pos//=2
    posr = m+1+b[0]+1
    posl = m+1
    while posr>posl:
        if posr%2==1:
            posr-=1
            ans += tree[posr]
```

```
        if posl%2==1:
            ans += tree[posl]
            posl+=1
        posr//=2
        posl//=2
print(ans)
```

Критерии оценивания

Тест № 1 совпадает с тестом из условия задачи. Баллы за него не начисляются.

Успешное прохождение каждого теста №№ 2–29 оценивается в 1 балл.

В тестах №№ 2–7 размеры поля не превышают 100 по каждому измерению.

В тестах №№ 8–14 для размеров поля выполняются неравенства $1 \leqslant n \leqslant 200\,000$, $1 \leqslant m \leqslant 10\,000$.

В тестах №№ 15–21 значения $a_1, \ldots, a_n$ упорядочены по неубыванию.

Задача 4.2.1.5. Логические функции (28 баллов)

Имя входного файла: стандартный ввод или `input.txt`.

Имя выходного файла: стандартный вывод или `output.txt`.

Ограничение по времени выполнения программы: 1 с.

Ограничение по памяти: 128 Мбайт.

Условие

Логические или булевы функции — это функции многих переменных, определенные на множестве $\{0, 1\}$. Они широко используются при проектировании различных микросхем. Напишите программу для работы с такими функциями.

Наиболее распространенный способ задания булевых функций — через таблицу истинности, см. таблицу 4.2.1.

Таблица 4.2.1. Пример таблицы истинности

x_1	x_2	x_3	$f(x_1, x_2, x_3)$
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	0

Если функция зависит от n переменных, то в левой части таблицы расположены все двоичные наборы длины n, обычно упорядоченные в натуральном порядке. Поэтому булеву функцию можно задать вектором из правой части таблицы. Длина такого вектора будет равна 2^n. Для данного примера вектор будет равен $f(x_1, x_2, x_3) = (00111010)$.

Функции также можно задавать при помощи формул. В этой задаче будем строить формулы с использованием заданного набора m функций n переменных $f_1(x_1, \ldots, x_n), \ldots, f_m(x_1, \ldots, x_n)$ и бинарных операций $\vee$ («или») и $\wedge$ (И). Обратите внимание, что в формулах нет отрицаний, операции всегда применяются к двум функциям n переменных, а все заданные функции $f_i(x_1, \ldots, x_n)$ всегда зависят от n переменных, записанных только в указанном порядке $x_1, \ldots, x_n$.

Напомним, что результат операции $\vee$ равен единице, если хотя бы один из аргументов равен единице. Результат операции $\wedge$ равен единице, если оба аргумента равны единице.

Например, найдем функцию, заданную формулой $g = (f_1 \vee f_2) \wedge (f_3 \vee f_2)$, где f_1, f_2, f_3 — функции трех переменных, $f_1 = (00010111)$, $f_2 = (10000001)$, $f_3 = (10110100)$.

Таблица 4.2.2

f_1	f_2	f_3	$f_1 \vee f_2$	$f_3 \vee f_2$	g
0	1	1	1	1	1
0	0	0	0	0	0
0	0	1	0	1	0
1	0	1	1	1	1
0	0	0	0	0	0
1	0	1	1	1	1
1	0	0	1	0	0
1	1	0	1	1	1

Теперь можно сформулировать саму задачу. Пусть заданы два набора функций от n переменных $f_1, \ldots, f_m$ и $g_1, \ldots, g_k$. Напишите программу, которая для каждой из функций g_i определит, можно ли ее представить в виде формулы указанного вида, использующей только функции f_j. Каждую функцию можно использовать несколько раз или не использовать вовсе. Формула может состоять из одного символа функции без операций, но не может быть пустой.

Формат входных данных

В первой строке на вход подается одно натуральное число n — количество переменных в функциях, $1 \leqslant n \leqslant 10$.

Во второй строке записано одно натуральное число m — количество функций $f_1, \ldots, f_m$, используемых при построении формул, $1 \leqslant m \leqslant 2\,000$.

Далее в m строках записаны векторы, задающие функции $f_1, \ldots, f_m$. Каждый вектор записан в виде строки длины 2^n из нулей и единиц.

Далее в $(m+3)$-й строке записано одно натуральное число k — количество функций $g_1, \ldots, g_k$, для которых требуется найти формулу, $1 \leqslant k \leqslant 2\,000$.

Далее в k строках записаны векторы, задающие функции $g_1, \ldots, g_k$ в том же формате, как и функции f_i.

Формат выходных данных

Программа должна вывести через пробел последовательность из k нулей или единиц. Если функцию g_i можно представить формулой указанного вида, то i-й символ в последовательности должен быть единицей, иначе — нулем.

Примеры

Пример №1

Стандартный ввод
3 3 00010111 10000001 10110100 6 00000000 10000001 11111111 10010101 00010110 01000000
Стандартный вывод
1 1 0 1 0 0

Примечания

Функцию с нулевым вектором можно получить по формуле $f_1 \wedge f_2 \wedge f_3$.

Функция g_2 совпадает с f_2.

Построение функции g_4 разобрано в примере.

Можно показать, что остальные функции нельзя получить в виде формул указанного вида.

Решение

Для булевых функций имеет место тождество $(x \vee y) \wedge z = (x \wedge z) \vee (y \wedge z)$. Используя это тождество, можно преобразовать любую формулу к виду $K_1 \vee K_2 \vee \cdots \vee \vee K_s$, где $K_j = f_{i_1} \wedge f_{i_2} \wedge \cdots \wedge f_{i_r}$. Поэтому будем искать представление произвольной функции g в виде формул такого вида. Обозначим за $f(x)$ значение функции f на

двоичном наборе x. Заметим, что если f — это векторное представление функции в виде строки, то $f(x)$ — это символ на позиции x. Определим T_x как результат применения операции $\wedge$ ко всем функциям f_i, таким что $f_i(x) = 1$.

$$T_x = \bigwedge_{f_i(x)=1} f_i.$$

Если не существует f_i таких, что $f_i(x) = 1$, то $T_x = (0 \dots 0)$.

Заметим, что если $K_j(x) = 1$, то $K_j \vee T_x = K_j$, так как в этом случае K_j состоит только из тех функций, которые входят в T_x.

Рассмотрим представление $g = K_1 \vee K_2 \vee \dots \vee K_s$. Если для некоторого x $g(x) = 1$, то найдется K_j такое, что $K_j(x) = 1$. Из этого, в частности, следует, что если некоторая функция g может быть представлена в виде формулы от $f_1, \dots, f_n$, и $g(x) = 1$, то $g \vee T_x = g$.

Определим функцию g' по следующей формуле:

$$g' = \bigvee_{g(x)=1} T_x.$$

Из построения $g \vee g' = g'$. С другой стороны, если g может быть представлена в виде формулы от $f_1, \dots, f_n$, то $g \vee g' = g$. Тогда, $g = g'$. Отсюда следует алгоритм для решения задачи. Сначала найдем функции $T(x)$ для всех x от нуля до $2^k - 1$.

Далее для каждой функции g_j найдем g'_j. Если функции совпадут, то ответ существует, иначе — нет. Следует также отдельно рассмотреть случай $g_j = (0 \cdots 0)$.

При реализации следует обязательно использовать один из способов компактного хранения векторов. В языке Python каждый из двоичных векторов следует перевести в целое число, что упростит реализацию и существенно уменьшит время работы программы.

Пример программы-решения

Ниже представлено решение на языке Python.

Python

```python
from functools import reduce
k = 2**int(input())
n = int(input())
fbits = [input() for _ in range(n)]
fint = [int(t,2) for t in fbits]
fw = [0]*k
fwbits = ['']*k
for i in range(k):
    b = 0
    for j in range(n):
        if fbits[j][i]=='1':
            if b==0:
                b = fint[j]
            else:
                b &= fint[j]
```

```
16      fw[i]=b
17      t = bin(b)[2:]
18      fwbits[i] = '0'*(k-len(t))+t
19  wed = reduce(lambda x,y:x&y,fint)
20  m = int(input())
21  for i in range(m):
22      g = input()
23      gint = int(g,2)
24      if gint==0:
25          print(int(wed==0),end=' ')
26      else:
27          b = 0
28          for j in range(k):
29              if g[j]=='1':
30                  b |= fw[j]
31          print(int(b==gint),end=' ')
```

Критерии оценивания

Тест № 1 совпадает с тестом из условия задачи. Баллы за него не начисляются.

Успешное прохождение каждого теста №№ 2–29 оценивается в 1 балл.

В тестах №№ 2–7 количество переменных не превосходит 3. В качестве функций g_i используются все возможные функции от заданного числа переменных.

В тестах №№ 8–13 количество переменных не превосходит 5. При этом $m \leqslant 50$, $k \leqslant 100$.

В тестах №№ 14–21 количество переменных не превосходит 8. При этом $m \leqslant$ $\leqslant 500$, $k \leqslant 500$.

4.2.2. Физика. 8–9 классы

Задача 4.2.2.1. Взмах (16 баллов)

Условие

Грузовой дрон представляет собой небольшой летающий беспилотник, несущий груз также небольших геометрических размеров, прикрепленный к нему снизу на легком нерастяжимом тросе длиной $l = 6$ м. Беспилотник движется по прямолинейному горизонтальному участку своего маршрута поступательно с постоянной скоростью. В некоторый момент ему поступает сигнал от пункта управления, и беспилотник практически мгновенно снижает свою скорость в $n = 2$ раза, сохраняя ее направление, в результате чего закрепленный на конце троса груз, качнувшись на тросе подобно маятнику, поднимается на максимальную высоту $h = l/4$ от своего прежнего равновесного положения. Определите начальную скорость беспилотника. Ускорение свободного падения $g = 9{,}8$ м/с^2. Влиянием сопротивления воздуха пренебречь.

Решение

Без учета влияния воздуха на груз, подвешенный на тросе, действуют только две силы: тяжести и натяжения троса. Пока беспилотник движется по прямой с постоянной скоростью v, так же движется и груз, значит, его ускорение оказывается равно нулю, а трос натянут вертикально, чтобы скомпенсировать действие силы тяжести.

После резкого замедления груза задачу удобнее решать в системе отсчета, связанной с беспилотником. Начальная скорость груза в этой системе равна $v_0 = v(1 - 1/n)$, а начальную высоту удобно считать нулевой. Поскольку за исключением однократного быстрого изменения скорости эта система инерциальна, а сопротивление воздуха мало, движение груза после замедления подчиняется закону сохранения механической энергии:

$$\frac{mv_0^2}{2} + 0 = 0 + mgh \Rightarrow v_0 = \sqrt{2gh}.$$

Чтобы получить ответ на вопрос задачи, эту скорость остается разделить на $(1-1/n)$:

$$v = \frac{v_0}{1 - 1/n} = \frac{n}{n-1}\sqrt{2gh} = \frac{n}{n-1}\sqrt{\frac{gl}{2}} \approx 10{,}8 \text{ м/с}.$$

Ответ:

$$v = \frac{n}{n-1}\sqrt{\frac{gl}{2}} \approx 10{,}8 \text{ м/с}.$$

Критерии оценивания

Верно записан закон сохранения механической энергии	5 баллов

Возможность применения ЗСМЭ на том интервале времени и в той системе отсчета, в которой он записан, обоснована	5 балла
Получен правильный ответ	6 баллов
Всего	**16 баллов**

Задача 4.2.2.2. Нелинейный элемент (18 баллов)

Условие

В управляющей цепи дрона используется нелинейный электрический элемент, вольт-амперная характеристика которого изображена на рис. 4.2.4: при напряжениях U меньше некоторого значения U_0 ток через этот элемент не протекает вовсе, а при напряжениях $U > U_0$ сила тока оказывается прямо пропорциональна разности $U - U_0$. Известно, что при приложении к нему напряжения $3U_0$ через элемент протекает ток $I_1 = 4\,\text{мА}$, а при приложении очень больших (много больше U_0) напряжений нелинейный элемент ведет себя как резистор с сопротивлением $R_1 = 3\,\text{кОм}$. Найдите U_0.

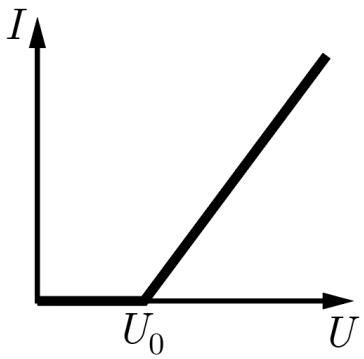


Рис. 4.2.4

Решение

Согласно условиям, при $U > U_0$ сила тока I через нелинейный элемент прямо пропорциональна разности $U - U_0$. Обозначим коэффициент этой пропорциональности k:

$$I(U) = k(U - U_0).$$

Электрическое сопротивление R элемента — это отношение напряжения на его концах к силе протекающего через него тока, поэтому зависимость $R(U)$ принимает вид

$$R(U) = \frac{U}{I(U)} = \frac{U}{k(U - U_0)}.$$

Легко видеть, что при $U \gg U_0$ знаменатель этого выражения становится близок к U, а значит, сопротивление нелинейного элемента приближается к величине $1/k$, следовательно

$$k = \frac{1}{R_1}.$$

Подставив найденное значение коэффициента в выражение для силы тока при напряжении $3U_0$, получим:

$$I_1 = I(3U_0) = \frac{3U_0 - U_0}{R_1} = \frac{2U_0}{R_1} \Rightarrow U_0 = \frac{I_1 R_1}{2} = 6\,\text{В}.$$

Ответ:

$$U_0 = \frac{I_1 R_1}{2} = 6\,\text{В}.$$

Критерии оценивания

Верно записан закон Ома для участка цепи	4 балла
Составлено уравнение, отражающее прямую пропорциональность тока разнице $U - U_0$	4 балла
Найден коэффициент этой пропорциональности.	5 баллов
Получен правильный ответ	5 баллов
Всего	**18 баллов**

Задача 4.2.2.3. Атмосфера (18 баллов)

Условие

Благодаря уникальному климату и составу далекой планеты, ее атмосфера имеет два слоя с достаточно четкими границами, в пределах каждого из которых ее плотность практически постоянна, и столь же резкую верхнюю границу с пустотой космоса в верхней части второго слоя. При этом нижний слой атмосферы в два раза тоньше ее верхнего слоя. Два абсолютно одинаковых аэростата с жесткими оболочками висят неподвижно в серединах соответствующих слоев. Один из них удерживается в атмосфере без груза, второй — с грузом, масса которого равна массе самого аэростата. При этом барометр первого показывает давление $p_1 = 12\,\text{кПа}$. Какое давление показывает барометр второго?

Решение

Очевидным следствием закона Архимеда является то, что слой с меньшей плотностью (обозначим ее ρ_1) находится выше слоя с большей (обозначим ее ρ_2).

Для каждого аэростата сила Архимеда $\rho g V$, где ρ — плотность атмосферного слоя, в котором он находится, g — ускорение свободного падения планеты, а V — объем аэростата, должна в точности компенсировать силу тяжести, действующую на аэростат с грузом. Объем V не зависит от внешнего давления, поскольку по условиям оболочки аэростатов жесткие. Обозначив массу пустого аэростата m, условия их равновесия запишем в виде

$$\begin{cases} \rho_1 g V = mg, \\ \rho_2 g V = 2mg. \end{cases}$$

Из этой системы элементарно следует

$$\rho_2 = 2\rho_1.$$

Обозначив h толщину верхнего слоя, запишем теперь выражения для давлений, действующих на барометры каждого из аэростатов. На каждый из них давление оказывает половина воздушного слоя, в котором он находится, и вышележащий слой, если такой есть:

$$\begin{cases} p_1 = \dfrac{\rho_1 h}{2} g, \\ p_2 = \left(\rho_1 h + \dfrac{\rho_2 2h}{2}\right) g = (\rho_1 + \rho_2) gh. \end{cases}$$

Подставляя в последнее уравнение полученное выражение для ρ_2, приведем его к виду

$$p_2 = 3\rho_1 gh = 3p_1 = 36\,\text{кПа}.$$

Ответ:

$$p_2 = 3p_1 = 36\,\text{кПа}.$$

Критерии оценивания

Верно записаны условия равновесия аэростатов	по 3 балла за аэростат
Верно записаны выражения для гидростатических давлений, действующих на аэростаты	по 3 балла за аэростат
Получен правильный ответ	6 баллов
Всего	**18 баллов**

Задача 4.2.2.4. Прожектор (20 баллов)

Условие

Поисковый дрон снабжен прожектором, состоящим из мощного точечного источника света и плоской линзы с оптической силой $D = 2\,\text{дптр}$, жестко закрепленной так, что когда дрон зависает над зоной спасательной операции, плоскость линзы оказывается параллельна земле. Источник света может перемещаться относительно линзы. В начальный момент источник находился в главном фокусе линзы, в результате чего дрон, висящий на высоте $H = 15\,\text{м}$ над равнинной местностью, создавал на земле круглое световое пятно. Определите, на какое расстояние должен сдвинуться вниз вдоль главной оптической оси линзы источник света, чтобы площадь светового пятна увеличилась в четыре раза.

Решение

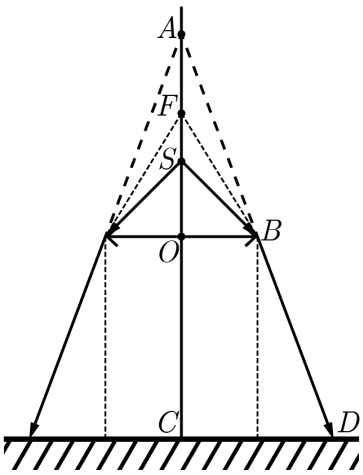


Рис. 4.2.5

Когда источник света находится строго в фокусе линзы, прожектор создает пучок лучей, параллельных главной оптической оси и перпендикулярных экрану (поверхности Земли), а значит, независимо от высоты дрона, радиус светового пятна равен радиусу линзы. Чтобы увеличить площадь светового пятна в четыре раза, его радиус необходимо увеличить в два раза. Поскольку по условиям источник сдвигается вниз (ближе к линзе), линза сформирует его мнимое изображение. Изобразим ход лучей в этом случае на чертеже (рис. 4.2.5). На нем O — оптический центр линзы, F — ее фокус, S — источник после смещения, A — мнимое изображение источника в линзе, B — край линзы, C — центр светового пятна, D — его край. Сплошные стрелки изображают реальный ход световых лучшей, редкий пунктир — их продолжения, частый тонкий пунктир — ход лучей до сдвига источника.

Из чертежа легко видеть, что треугольники $\triangle ABO$ и $\triangle ADC$ подобны (по двум углам), а значит,

$$\frac{AC}{AO} = \frac{CD}{OB} = 2.$$

При этом длина отрезка AO — есть расстояние f от мнимого изображения до плоскости линзы, а длина отрезка OC — высота зависания дрона. С учетом этого перепишем отношение в виде

$$\frac{f+H}{f} = 2 \Rightarrow f = H$$

и применим данный результат в формуле тонкой линзы:

$$D = \frac{1}{d} - \frac{1}{H} \Rightarrow d = \left(D + \frac{1}{H}\right)^{-1} = \frac{H}{DH+1},$$

где d — расстояние от линзы до источника (длина отрезка OS на рисунке). Чтобы окончательно дать ответ на вопрос задачи, вычтем новое значение d из прежнего, равного фокусному расстоянию $1/D$:

$$b = \frac{1}{D} - d = \frac{1}{D} - \frac{H}{DH+1} \approx 1{,}6\,\text{см}.$$

Ответ:

$$b = \frac{1}{D} - \frac{H}{DH+1} \approx 1{,}6\,\text{см}.$$

Критерии оценивания

Качественно правильно изображен ход лучей	4 балла
Определено, что изначальный радиус светового пятна равен радиусу линзы	2 балла
Определено, что конечный радиус светового пятна в два раза больше начального	3 балла
Верно найдено положение мнимого изображения	3 баллов
Верно записана формула тонкой линзы	4 баллов
Получен правильный ответ	4 балла
Всего	**20 баллов**

Задача 4.2.2.5. Форсаж (24 балла)

Условие

Роботизированный самолет снабжен электродвигателем, контроллер которого поддерживает на нем постоянное напряжение U. Сила сопротивления воздуха, которую приходится преодолевать электродвигателю для горизонтального движения с постоянной скоростью v, прямо пропорциональна квадрату этой скорости, а КПД η двигателя зависит от этой скорости по закону $\eta(v) = \dfrac{3v(2v_0 - v)}{4v_0^2}$, где $v_0 = 150$ м/с. В штатных условиях самолет движется с крейсерской скоростью, при которой КПД его двигателя максимален, а в режиме форсажа — в $k = 1{,}5$ раз быстрее. Определите, во сколько раз необходимо повысить силу протекающего через двигатель тока, чтобы перейти с крейсерской скорости в этот режим.

Решение

Прежде всего проанализируем характер зависимости $\eta(v)$. Раскрыв скобки, легко видеть, что функция

$$\eta(v) = -\frac{3}{4v_0^2}v^2 + \frac{3}{2v_0}v$$

квадратичная, причем коэффициент перед ее старшим слагаемым отрицательный. Это значит, что ее график представляет собой параболу ветвями вниз.

Из исходного вида этой функции очень легко видеть, что она имеет два корня: $\eta(v)$ равна нулю либо когда $v = 0$, либо когда $v = 2v_0$, откуда следует, что вершина параболы лежит ровно в середине между этими двумя корнями: в точке $v = v_0$. Подставив это значение, получим

$$\eta_{max} = \eta(v_0) = \frac{3v_0(2v_0 - v_0)}{4v_0^2} = \frac{3}{4}.$$

Это и есть КПД двигателя на крейсерской скорости v_0. Найдем теперь КПД η_f двигателя в режиме форсажа:

$$\eta_f = \eta\left(\frac{3}{2}v_0\right) = \frac{3\frac{3}{2}v_0(2v_0 - \frac{3}{2}v_0)}{4v_0^2} = \frac{9}{16}.$$

Полезная мощность P, которую необходимо развивать двигателю для преодоления силы сопротивления F, определяется по формуле $P = vF$, а потребляемая им мощность P_0 равна, соответственно, $P_0 = vF/\eta$. В то же время из раздела Электричество известно, что эта последняя равна UI. Тогда сила тока в любом режиме может быть выражена как

$$I = \frac{vF}{\eta U}.$$

Чтобы ответить на вопрос задачи, составим пропорцию между силами тока I_0 в крейсерском и I_f в форсажном режиме:

$$\frac{I_f}{I_0} = \frac{v_f F_f}{\eta_f \cancel{U}} \cdot \frac{\eta_{max}\cancel{U}}{v_0 F_0} = \frac{v_f}{v_0} \cdot \frac{F_f}{F_0} \cdot \frac{3}{4} \cdot \frac{16}{9},$$

где индексы 0 и f также используются для обозначения величин, относящихся к крейсерскому и форсажному режимам соответственно.

Согласно условиям задачи $v_f/v_0 = k$, а соответствующее увеличение силы сопротивления воздуха $F_f/F_0 = (v_f/v_0)^2 = k^2$. Тогда окончательно

$$\frac{I_f}{I_0} = k^3\frac{4}{3} = \frac{9}{2}.$$

Ответ:

$$\frac{I_f}{I_0} = \frac{9}{2} = 4{,}5.$$

Критерии оценивания

Верно записана связь мощности со скоростью и силой сопротивления	3 балла
Верно записана связь мощности с напряжением и силой тока	3 балла
Продемонстрировано понимание того, что зависимость КПД от скорости квадратичная и имеет один максимум	5 балла
Показано, что крейсерская скорость равна v_0	4 балла
Верно найден максимальный КПД двигателя	3 балла
Верно найден КПД двигателя при форсаже	4 балла
Получен правильный ответ	6 баллов
Всего	**28 баллов**

4.2.3. Физика. 10–11 классы

Задача 4.2.3.1. Поворотная призма (15 баллов)

Условие

В оптической системе наблюдательного самолета используется поворотная призма с основанием в виде равнобедренного тупоугольного треугольника, тупой угол которого обозначен α. Пучок световых лучей попадает на призму через одну из ее меньших прямоугольных граней, падая на эту грань вдоль ее нормали (см. рис. 4.2.6).

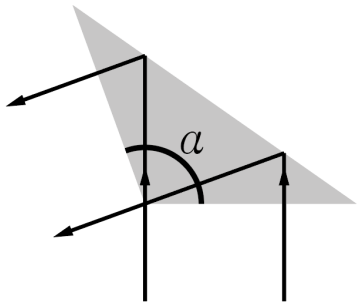


Рис. 4.2.6

Из-за требований к обтекаемости прибора угол α требуется сделать как можно больше. Определите, при каком максимальном значении α свет испытает полное внутреннее отражение, если оптическое стекло, из которого изготовлена призма, имеет абсолютный показатель преломления $n = 1{,}7$, а снаружи от нее находится воздух, достаточно разреженный, чтобы считать его показатель преломления единицей.

Решение

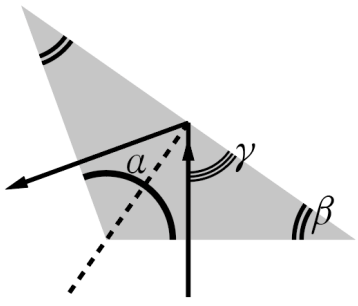


Рис. 4.2.7

Изобразим ход одного луча в призме (рис. 4.2.7). В силу симметрии, лучи выходят из призмы по нормали к ее границе раздела, поэтому полное внутреннее отражение возможно только на самой широкой стороне призмы. Найдем угол падения света на эту границу. Угол β при основании призмы легко найти, опираясь на сумму углов

треугольника и равенство углов при основании равнобедренного треугольника:

$$\alpha + 2\beta = 180^\circ \Rightarrow \beta = 90^\circ - \frac{\alpha}{2}.$$

Угол γ, который падающий луч составляет с широкой гранью призмы, можно найти, опираясь на сумму углов в треугольнике, образованном этим лучем и двумя гранями призмы. Поскольку первое падение по условиям задачи происходит по нормали (под углом 90° к границе раздела), а угол между гранями равен β, справедливо равенство

$$\gamma + \beta + 90^\circ = 180^\circ \Rightarrow \gamma = 90^\circ - \beta = \frac{\alpha}{2}.$$

Наконец, угол падения луча ϕ по определению является углом между падающим лучом и нормалью к границе раздела; следовательно, он равен

$$\phi = 90^\circ - \gamma = 90^\circ - \frac{\alpha}{2}.$$

Согласно закону Снеллиуса, критический угол $\phi_{\text{кр}}$ полного внутреннего отражения достигается тогда, когда синус угла падения оказывается равен относительному показателю преломления сред:

$$\frac{1}{n} = \sin\phi_{\text{кр}} = \cos\frac{\alpha_{\text{кр}}}{2}.$$

При меньших углах падения (и, следовательно, больших углах α) явление полного внутреннего отражения наблюдаться не будет. Отсюда

$$\alpha_{\text{кр}} = 2\arccos\left(\frac{1}{n}\right) \approx 108^\circ.$$

Ответ:

$$\alpha_{\text{кр}} = 2\arccos\left(\frac{1}{n}\right) \approx 108^\circ.$$

Критерии оценивания

Верно записан закон Снеллиуса или условие ПВО	4 балла
Верно записана связь между углом α и углом падения луча на широкую границу раздела	3 балла
Указанная взаимосвязь геометрически обоснована	3 балла
Получен правильный ответ	5 баллов
Всего	**15 баллов**

Задача 4.2.3.2. Наперегонки (17 баллов)

Условие

В сравнительных испытаниях принимают участие два дрона. К характеристикам дрона — рабочей массе и запасу хода — предъявляются суровые требования,

поэтому каждый из них способен развивать свое максимальное ускорение a (одинаковое для обоих дронов) довольно недолго. В ходе испытаний дронам предстоит преодолеть по прямой одинаковое расстояние S в спокойном воздухе. Первый дрон двигался с ускорением a на протяжении некоторого времени t_1 а затем, из-за разрядки аккумуляторов, был вынужден снизить ускорение до $a/2$ и, двигаясь с ним на протяжении такого же времени, достиг финиша. Второй дрон, напротив, взял запас и на протяжении некоторого времени t_2 двигался с ускорением $a/2$, а уже по прошествии этого времени, обнаружив, что запас хода достаточен, поднял ускорение до a и, двигаясь на протяжении такого же времени с ним, дошел до конца маршрута. Какой дрон пришел к финишу первым? Во сколько раз меньшее время занял у него маршрут? Начальные скорости обоих дронов равнялись нулю.

Решение

Рассмотрим движение первого дрона. На первом участке пути S_1 он движется без начальной скорости с постоянным ускорением a, поэтому для этой половины закон движения имеет вид

$$S_1 = \frac{at_1^2}{2}.$$

На втором участке пути S_1' дрон движется с меньшим ускорением, но обладает начальной скоростью $v_1 = at_1$, поэтому его закон движения принимает вид

$$S_1' = v_1 t_1 + \frac{at_1^2}{4} = \frac{5}{4} at_1^2.$$

Таким образом, общий путь S_1 дрона связан с его временем в движении выражением

$$S = S_1 + S_1' = \frac{7}{4} at_1^2.$$

Аналогичные рассуждения для второго дрона (все величины с индексом 2 вместо 1) имеют вид

$$S_2 = \frac{at_2^2}{4}; \qquad v_2 = \frac{at_2}{2},$$

$$S_2' = v_2 t_2 + \frac{at_2^2}{2} = at_2^2,$$

$$S = S_2 + S_2' = \frac{5}{4} at_2^2.$$

Приравнивая эти совокупные пути, получим

$$\frac{7\cancel{a}t_1^2}{\cancel{4}} = \frac{5\cancel{a}t_2^2}{\cancel{4}} \Rightarrow \frac{t_2}{t_1} = \sqrt{\frac{5}{7}} \approx 1{,}18.$$

Ответ: первый дрон прибудет к финишу быстрее в $\sqrt{\frac{7}{5}} \approx 1{,}18$ раза.

Критерии оценивания

Верно записан хотя бы один закон равноускоренного движения без начальной скорости	3 балла
Верно записан хотя бы один закон равноускоренного движения с учетом правильной начальной скорости на втором участке	5 баллов
Верно выражены пути обоих дронов через a и $t_{1,2}$	3 балла
Получен правильный ответ	6 баллов
Всего	**17 баллов**

Задача 4.2.3.3. Неизведанная атмосфера (20 баллов)

Условие

Спускаемый аппарат разрабатывается для работы в условиях атмосферы неизведанной планеты, состоящей из сложной смеси одноатомных, двухатомных и многоатомных газов, которая, однако, ведет себя как достаточно близкий к идеальному газ. Аппарат будет приводиться в движение тепловым двигателем, рабочий цикл которого состоит из двух изохор и двух изобар. При этом максимальное давление газа в цикле в $m = 2$ раза выше минимального, а максимальный объем — в $n = 3$ раз выше минимального. В качестве рабочего тела в двигателе выступает забортная атмосфера. Измерения показали, что работающий так двигатель обладает коэффициентом полезного действия $\eta = 1/8$. Найдите молярную теплоемкость атмосферы неизведанной планеты в изохорном процессе. Универсальная газовая постоянная $R = 8{,}31$ Дж/моль·К.

Решение

Изобразим цикл на pV-координатах: он имеет вид прямоугольника (рис. 4.2.8). Легко заметить, что площадь внутри этого прямоугольника (работа за цикл)

$$A = (m-1)(n-1)p_1V_1.$$

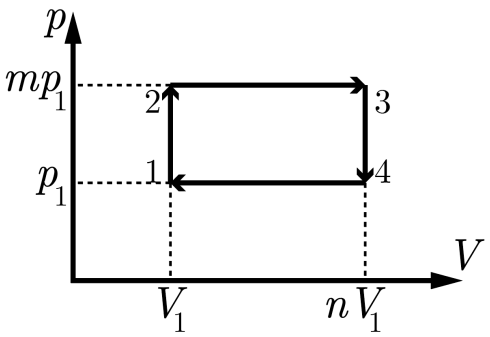


Рис. 4.2.8

Обозначим искомую молярную теплоемкость C_V. Процесс 1–2 изохорный, поэтому газ получает на нем теплоту в количестве, полностью определяемом этой величиной:

$$Q_{12} = C_V\nu(T_2 - T_1),$$

где ν — количество вещества рабочего газа, а $T_{1,2}$ — абсолютные температуры газа в точках 1 и 2 соответственно. Газ также получает теплоту на изобарном процессе 2–3, где количество этой теплоты можно выразить при помощи первого начала термодинамики:

$$Q_{23} = C_V\nu(T_3 - T_2) + mp_1(n-1)V_1,$$

где T_3 — абсолютная температура газа в точке T_3.

Таким образом, КПД цикла определяется соотношением

$$\eta = \frac{A}{Q_{12} + Q_{23}} = \frac{(m-1)(n-1)p_1V_1}{C_V\nu(T_3 - T_1) + m(n-1)p_1V_1}.$$

Поскольку рабочее тело ведет себя как идеальный газ, к нему применимо уравнение Менделеева – Клапейрона, из которого следует

$$\nu T_1 = \frac{p_1V_1}{R}; \qquad \nu T_3 = \frac{mnp_1V_1}{R}.$$

С учетом этих соотношений выражение для КПД принимает вид

$$\eta = \frac{(m-1)(n-1)\cancel{p_1V_1}}{C_V/R(mn-1)\cancel{p_1V_1} + m(n-1)\cancel{p_1V_1}},$$

откуда окончательно выразим C_V:

$$C_V = \frac{(m-1)(n-1)}{\eta(mn-1)}R - \frac{m(n-1)}{mn-1}R = \frac{(n-1)(m-m\eta-1)}{\eta(mn-1)}R = \\ = \frac{12}{5}R \approx 20\,\text{Дж/(моль·К)}.$$

Ответ:

$$C_V = \frac{(n-1)(m-m\eta-1)}{\eta(mn-1)} = \frac{12}{5}R \approx 20\,\text{Дж/(моль}\cdot\text{К)}.$$

Примечание: хотя полное решение было представлено в общем виде, оно будет значительно короче, если переменные m и n сразу заменить их численными значениями. Снижать оценку за подобную подстановку не следует.

Критерии оценивания

Верно записано уравнение Менделеева – Клапейрона	3 балла
Верно указано, на каких участках цикла газ получает и теряет тепло	3 балла
Верно найдена работа газа за цикл	3 балла
Верно записано первое начало термодинамики	3 балла
Продемонстрировано понимание термина «молярная теплоемкость в изохорном процессе»	3 балла
Получен правильный ответ	5 баллов
Всего	**20 баллов**

Задача 4.2.3.4. Испытания (23 балла)

Условие

Двенадцать двигателей дрона должны были питаться от единого источника с пренебрежимо малым внутренним сопротивлением и ЭДС $\mathcal{E}_0 = 240\,\text{В}$. Но очутившись с презентацией на технической выставке, команда разработчиков обнаружила, что вместо заявленного, организаторы предоставили им источник тока с ЭДС $\mathcal{E} = 600\,\text{В}$, а вдобавок — имеющий не столь малое внутреннее сопротивление $r = 2\,\text{Ом}$. Команда разработчиков ненадолго пришла в замешательство: прежняя схема подключения вывела бы двигатели из строя; но вскоре один из инженеров обнаружил, что, разбив двигатели на несколько одинаковых групп, внутри каждой из которых они будут соединены последовательно, а затем уже эти группы параллельно друг другу соединив с источником, можно добиться того, что через каждый двигатель будет протекать строго номинальная сила тока I_0. Найдите I_0.

Решение

Напряжение на двигателях, подключенных параллельно к идеальному источнику, будет равно ЭДС $\mathcal{E}_0$ этого источника. Напряжение на любой группе двигателей, подключенных к неидеальному источнику, не может превышать ЭДС источника, а значит, на каждом из двигателей оно не будет превышать $\mathcal{E}/n$, где n — число двигателей в группе. Для протекания через отдельный двигатель номинальной силы тока напряжение на двигателе также должно быть номинальным. Но учитывая известные значения $\mathcal{E}$ и $\mathcal{E}_0$, легко видеть, что $\mathcal{E}_0 < \mathcal{E}/n$ только при $n = 1$ и $n = 2$. Первый вариант невозможен, поскольку в этом случае двигатели остаются фактически подключены по прежней схеме, а по условиям такое подключение выведет их из строя. Значит, в каждой группе последовательно соединенных элементов по два двигателя.

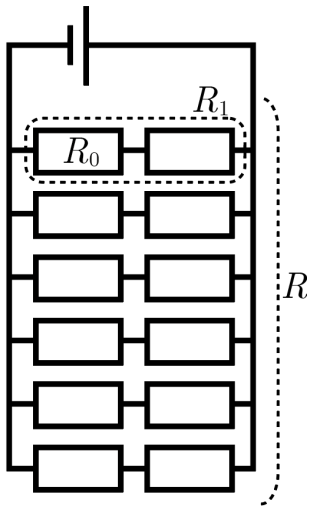


Рис. 4.2.9

Обозначим сопротивление одного двигателя R_0. Тогда эквивалентное сопротивление одной группы двигателей $R_1 = 2R_0$, а шести таких групп, соединенных парал-

лельно:

$$R = \frac{R_1}{6} = \frac{R_0}{3}.$$

По закону Ома для полной цепи через эту батарею будет протекать ток

$$I = \frac{\mathcal{E}}{R + r} = \frac{\mathcal{E}}{R_0/3 + r},$$

а значит, через каждый двигатель будет протекать ток, равный

$$I_0 = \frac{I}{6} = \frac{\mathcal{E}}{2R_0 + 6r}.$$

В то же время, при подключении двигателей по изначальному плану, эта сила тока составила бы

$$I_0 = \frac{\mathcal{E}_0}{R_0}.$$

Приравнивая эти два выражения, получим

$$\frac{\mathcal{E}_0}{R_0} = \frac{\mathcal{E}}{2R_0 + 6r} \Rightarrow R_0 = \frac{6r\mathcal{E}_0}{\mathcal{E} - 2\mathcal{E}_0}.$$

Используя последние два соотношения, находим окончательно ответ на вопрос задачи:

$$I_0 = \frac{\mathcal{E}_0}{R_0} = \frac{\mathcal{E} - 2\mathcal{E}_0}{6r} = 10\,\text{А}.$$

Ответ:

$$I_0 = \frac{\mathcal{E} - 2\mathcal{E}_0}{6r} = 10\,\text{А}.$$

Критерии оценивания

Указано, что при соединении по изначальной схеме напряжение на каждом двигателе равно ЭДС источника	3 балла
Обосновано, что двигателей в группе может быть только два	3 балла
Найдено эквивалентное сопротивление смешанного соединения двигателей	3 балла
Верно записан закон Ома для полной цепи	3 балла
Получено выражение для силы тока или напряжения на одном двигателе при смешанном соединении	5 баллов
Получен правильный ответ	6 баллов
Всего	**23 балла**

Задача 4.2.3.5. Мультикоптер (25 баллов)

Условие

Мультикоптер удерживается в спокойном воздухе неподвижно благодаря одновременной работе шести одинаковых электродвигателей, расположенных в вершинах

правильного шестиугольника $ABCDEF$, в центре которого находится центр масс коптера. Лопасти каждого из двигателей вращаются с одинаковой угловой скоростью ω_0, при этом лопасти двигателей, расположенных в вершинах A, C и E, имеют такую форму, чтобы создавать подъемную силу при вращении в одном направлении, а расположенных в вершинах B, D и F — в противоположном. В остальном они идентичны. Подъемная сила, создаваемая каждым из двигателей, прямо пропорциональна скорости его вращения, и также прямо пропорционален ей момент силы, вращающей лопасти относительно оси двигателя. В некоторый момент двигатели в вершинах A и E одновременно отказали и остановились. Бортовой компьютер быстро вычислил, каким образом необходимо поменять скорости вращения каждого из оставшихся исправными двигателей, чтобы коптер остался неподвижен. Найдите эти скорости.

Решение

Прежде всего введем систему обозначений. Обозначим искомые скорости вращения уцелевших двигателей $\omega_{B,C,D,F}$. Аналогично эти же индексы будем использовать для других величин, связанных с двигателем в соответствующей вершине. Величины, касающиеся работы двигателей до поломки, будем помечать индексом 0. Введем систему координат так, как это изображено на рис. 4.2.10: ось $0x$ проведем через вершины F и C, а $0y$ — перпендикулярно ей. Начало системы координат поместим в центр масс мультикоптера. Сторону шестиугольника обозначим a.

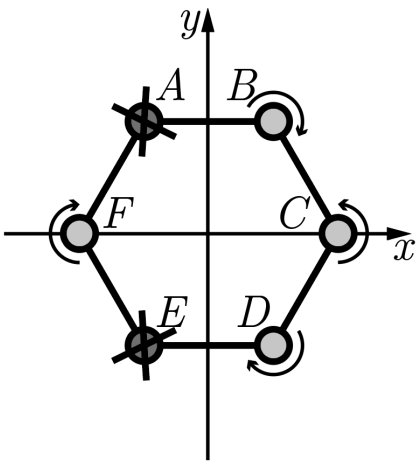


Рис. 4.2.10

Для того чтобы коптер остался в равновесии, должны выполниться четыре условия. Первое из них достаточно очевидно: сумма подъемных сил F, действующих на коптер, должна остаться неизменной, так как не изменился вес коптера:

$$F_B + F_C + F_D + F_F = 6F_0.$$

Второе условие: суммарный момент подъемных сил относительно $0x$ должен остаться равен нулю. Относительно этой оси плечи подъемных сил в вершинах F и C, а также плечо силы тяжести равны нулю. Плечи подъемных сил в вершинах B и D одинаковы и равны $\sqrt{3}a/2$, при этом эти вершины находятся по разные стороны от оси. Следовательно, уравнение моментов принимает вид

$$F_B \frac{\sqrt{3}a}{2} = F_D \frac{\sqrt{3}a}{2}.$$

Третье условие: суммарный момент подъемных сил относительно $0y$ должен остаться равен нулю. Относительно этой оси плечи подъемных сил в вершинах F и C одинаковы и равны a. Плечи подъемных сил в вершинах B и D одинаковы и равны $a/2$. Плечо силы тяжести вновь равно нулю. С учетом направлений действия соответствующих сил, уравнение моментов принимает вид

$$F_F \cancel{a} = F_C \cancel{a} + (F_B + F_D)\frac{\cancel{a}}{2}.$$

Последнее условие предотвращает вращение мультикоптера в плоскости рисунка. Каждый двигатель, вращая свои лопасти и действуя на них с некоторым моментом M, создает момент $-M$, действующий на коптер в противоположную сторону (по третьему закону Ньютона). Это значит, что суммарный момент сил относительно осей двигателей для всех двигателей, вращающихся в одну сторону (в вершинах F, B и D), должен равняться суммарному моменту двигателей, вращающихся в другую сторону (только в вершине C из уцелевших):

$$M_C = M_B + M_D + M_F.$$

Во всех четырех составленных уравнениях учтем пропорциональность сил и моментов соответствующим угловым скоростям, и перепишем систему через эти скорости (коэффициенты пропорциональности во всех уравнениях ниже сразу сократим):

$$\begin{cases} \omega_B + \omega_C + \omega_D + \omega_F = 6\omega_0, \\ \omega_B = \omega_D, \\ \omega_F = \omega_C + \dfrac{\omega_B + \omega_D}{2}, \\ \omega_C = \omega_B + \omega_D + \omega_F. \end{cases}$$

Эта система имеет единственное решение:

$$\begin{cases} \omega_B = \omega_D = 0, \\ \omega_F = \omega_C = 3\omega_0. \end{cases}$$

Такое равновесие, разумеется, является неустойчивым и не может удерживаться без контроля со стороны электроники коптера.

Ответ:

$$\begin{cases} \omega_B = \omega_D = 0, \\ \omega_F = \omega_C = 3\omega_0. \end{cases}$$

Критерии оценивания

Верно составлено первое условие равновесия (сумма сил)	4 балла
Верно составлено второе условие равновесия (сумма моментов относительно $0x$)	5 баллов
Верно составлено третье условие равновесия (сумма моментов относительно $0y$)	5 баллов

Верно составлено четвертое условие равновесия (сумма моментов в плоскости коптера)	6 баллов
Получен правильный ответ	5 баллов
Всего	**25 баллов**

4.3. Инженерный тур

4.3.1. Общая информация

Цель заключительного этапа — создание комплексной системы автоматизированного мониторинга объектов ТЭК с использованием квадрокоптера. Это подразумевает разработку программного обеспечения, обеспечивающего контроль состояния и функционирования объектов, позволяющего выявлять утечки и разливы нефти, проводить инспекцию нефтепроводов на предмет несанкционированных врезок и определять тепловые потери в системах теплоснабжения.

Кроме того, участникам необходимо создать программный код для визуализации результатов мониторинга на карте в режиме реального времени.

Дополнительно требуется разработка автономного комплекса, обеспечивающего безопасный взлет и посадку квадрокоптера, а также его подзарядку с использованием дронпоинта, что сопровождается подготовкой соответствующей технической документации.

4.3.2. Легенда задачи

Рис. 4.3.1

Эффективное наблюдение за объектами ТЭК с использованием квадрокоптера является ключом к оптимизации эксплуатационных процессов, повышению безопасности и снижению затрат. Применение современных решений в области летающей робототехники позволяет оперативно получать данные о состоянии таких объектов, как нефтепроводы и системы теплоснабжения, а также проводить детальный визуальный анализ, что способствует принятию своевременных и обоснованных решений.

Интегрированный подход значительно повышает эффективность мониторинга объектов ТЭК и обеспечивает оперативное реагирование на возможные инциденты.

4.3.3. Требования к команде и компетенциям участников

Количество участников в команде: 3–4 человека.

Компетенции, которыми должны обладать члены команды:

- **Инженер-программист (Python)** — написание кода для автономного полета квадрокоптера, работа с сервером, разработка алгоритма безопасного полета квадрокоптера.
- **Инженер-программист (C++, Python)** — алгоритмы компьютерного зрения для реализации автономных миссий квадрокоптера, машинное обучение. Работа в связке с инженером-программистом.
- **Инженер-техник** — моделирование и изготовление устройства, тестирование, техобслуживание и пилотирование квадрокоптера, работа с технической документацией.
- **Капитан/лидер команды** — работа с системами построения карты в RViz, осуществление общего руководства работой команды, распределение обязанностей и контроль соблюдения дедлайнов. Рекомендуется совмещение данной роли с другими ролями.

4.3.4. Оборудование и программное обеспечение

Таблица 4.3.1

Наименование	**Описание**
Полетный полигон ($5 \times 8 \times 3$ м)	Полигон для запуска автономных полетных миссий
Персональный компьютер (Intel Core i5 7260u 2.2 GHz, 8 Gb RAM)	Разработка программного кода для запуска полетной миссии
Ноутбук (Ryzen 5 5500u 2.1 GHz, 20 Gb RAM)	Разработка дронопорта
Графическая станция (Ryzen 7 1700 3.7 GHz, 16Gb RAM, GeForce GTX 1060 3Gb)	Разработка программного кода для запуска полетной миссии
Конструктор программируемого квадрокоптера «Клевер 4 Code»	Запуск полетных миссий
3D-принтер	Печать деталей для разработанного дронопорта и крепления тепловизора
Тепловизор	Автономный поиск утечек тепла
Лазерный станок	Резка фанеры для изготовления разработанного дронопорта
Полетный полигон ($4 \times 2 \times 2$ м)	Полигон для запуска тестовых автономных полетных миссий

Таблица 4.3.1

Наименование	Описание
• T-flex CAD 17, • Компас 3D v22, • Cura, • Repetier host, • VMWare Workstation (Gazebo), • Putty, • Winscp, • Notepad++ , • QGroundControl, • Etcher, • ColorMania, • Arduino IDE, • VLC player, • VScode, • Python3, • Termius.	Программное обеспечение для разработки программного кода для выполнения заданной полетной миссии

4.3.5. Описание задачи

Конфигурация площадки

Общая конфигурация площадки приведена на рис. 4.3.2.

- Зона «H» является точкой взлета и точкой посадки.
- Конфигурация застройки может изменяться.
- Положение датчиков на квадрокоптере может быть изменено при необходимости.
- Миссия мониторинга должна проходить полностью в автономном режиме.
- Доступ к сети осуществляется через роутер, выданный каждой команде; также имеется подключенный к сети роутер в полетной зоне.
- Команда несет ответственность за сохранность данных при работе в сети (нестандартный пароль на Wi-Fi, нестандартный пароль для SSH на квадрокоптере).
- Везде в задании началом координат принят левый нижний угол (по рис. 4.3.2 и 4.3.3), центр ArUco-маркера. Если в задании упоминаются координаты, то подразумевается СК aruco_map, если не сказано иное.
- На площадке для выполнения задания доступны пять тепловизоров TR256i (https://mileseeytools.com/products/thermal-imaging-camera-for-phone-mileseey-tr256i): один в полетной зоне, три доступны для работы на местах, один запасной.
- Правила использования тепловизоров: тепловизоры для работы на местах доступны на 30 мин в порядке живой очереди или в порядке расположения

столов, по усмотрению организаторов (правила могут быть изменены организаторами).

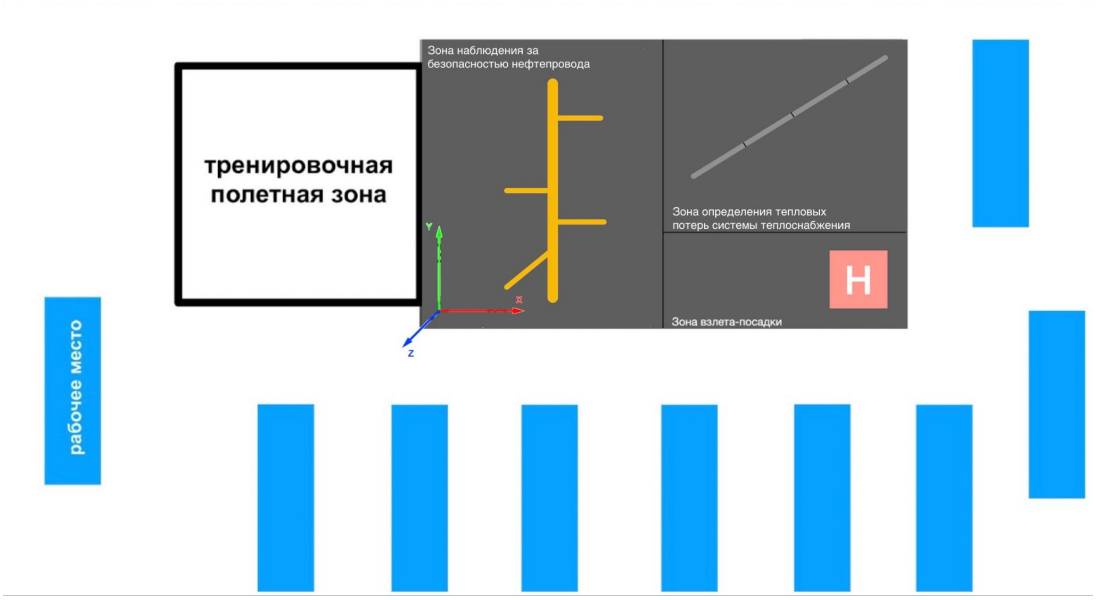


Рис. 4.3.2. Общая конфигурация площадки

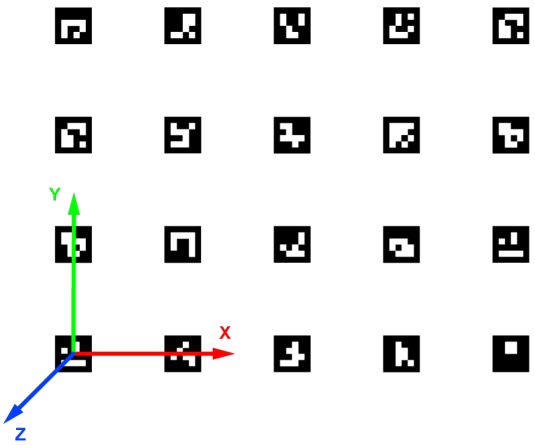


Рис. 4.3.3. Расположение начала координат

Программная часть

1.1. Подключение и настройка тепловизора

Поскольку на площадке используется тепловизор, в первую очередь проверим его работу и научимся с ним работать. Задание выполняется в следующем порядке:

1. Произвести подключение тепловизора перед началом сбора данных.
2. Получить видеопоток с тепловизора в полном объеме.
3. На основании сохраненного RAW-потока тепловизора, сформировать:

- термоизображения в цветовых схемах IRONBOW или JET;
- матрицу температур в градусах Цельсия, где каждая ячейка матрицы соответствует пикселю исходного термоизображения;
- матрицу температур в градусах Цельсия, где каждая ячейка матрицы соответствует пикселю исходного термоизображения.

4. Опубликовать термоизображения и матрицу температур в топик (название на усмотрение участника).

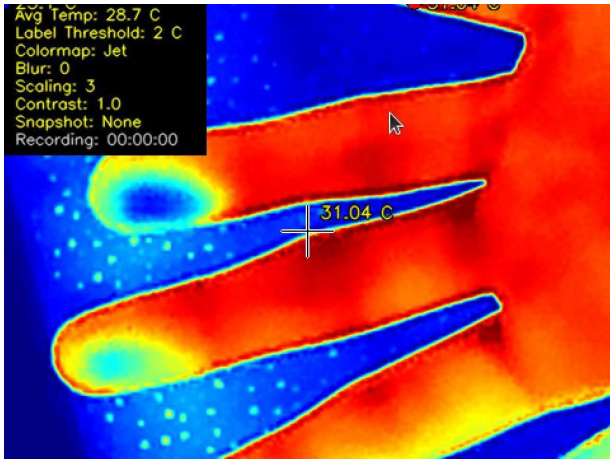


Рис. 4.3.4. Пример изображения в цветовой схеме JET

Обратите внимание, что эта задача оценивается не во время зачетных полетов. Для оценки, в любое время (за исключением перерывов), подойдите к любому эксперту на площадке и попросите оценить задание 1.1.

1.2. Определение тепловых потерь системы теплоснабжения

От координационного центра теплосетей города команде поставлена задача — провести мониторинг состояния участка теплосети. На заключительном этапе используется макет участка теплосети (**прямой**, см. рис. 4.3.2) с функцией нагрева отдельных участков.

Для успешного выполнения задания необходимо:

1. Выполнить взлет: совершить автономный взлет с зоны «H».
2. Осуществить полет к началу участка теплосети (координаты: (3.95, 0.4)).
3. Произвести сканирование участка:
   - выполнить пролет вдоль участка на любой удобной высоте;
   - выполнить и сохранить видео пролета с тепловизора (в цветовой схеме IRONBOW/JET) и видео с обычной камеры;
   - определить координаты участка(ов) с выявленными утечками тепла (определяется тепловизором, выше 40 °C).
4. Осуществить полет к концу участка теплосети (координаты: (6.45, 3.4)).
5. Отобразить в системе визуализации (см. 1.4):
   - каждый найденный участок, привязанный к системе координат;
   - список участков, содержащий координаты начала участка, координаты конца участка, максимальную температуру на данном участке.

1.3. Наблюдение за безопасностью нефтепровода

После осуществления мониторинга теплосетей города поставлена задача — выявление незаконных врезок в нефтепровод.

Для успешного выполнения задания необходимо:

1. Получить информацию об узлах нефтепровода (в т. ч. начало и конец нефтепровода), а также список легальных ответвлений с сервера. Пример полученных данных:

```
{"nodes": [{"x": 1.0, "y": 4.0}, {"x": 5.0, "y": 6.0}, ...],
↪  "tappings": [{"x": 1.0, "y": 4.0, "legal": 1}, ...]}.
```

   Допустимая погрешность в координатах возможных ответвлений — 0,1 м.

   JSON-файлы на сервере: http://157.180.22.113:8080/coords_1_legal.json. Кроме того, имеется возможность получить список возможных ответвлений (и легальных, и нелегальных) с сервера JSON-файлы на сервере: http://157.180.22.113:8080/coords_1.json.

   Авторизация на сервере: добавить Header:

```
Authorization: Bearer xxx-xxx-xxx,
```

   xxx-xxx-xxx — токен, который находится на гугл диске.

2. Выполнить полет к началу нефтепровода.
3. Осуществить полет к возможным ответвлениям, исходя из полученных с сервера данных.
4. Произвести фотофиксацию нелегального ответвления, если там на самом деле совершена врезка; выделить врезку любым образом.
5. Выполнить полет к концу нефтепровода.
6. Автоматически сформировать отчет, включающий в себя список точек, в которых осуществлялся мониторинг, а также результат мониторинга. Отчет должен быть сформирован в формате `.txt`. Пример отчета см. ниже.

```
Tapping 1. x=1.0, y=4.0, legal=1, illegal=1
Tapping 2. x=1.0, y=4.0, legal=0, illegal=1
...
Tapping n. x=1.0, y=4.0, legal=1, illegal=0
```

7. Отобразить в системе визуализации (см. 1.4):
   - нефтепровод (принять за прямую) и ответвления с привязкой к системе координат: нефтепровод должен быть желтого цвета, масштаб относительно карты ArUco-маркеров должен соответствовать реальному;
   - список координат незаконных врезок с возможностью посмотреть фото врезки.

1.4. Разработка веб-приложения

Разработать веб-приложение с графическим интерфейсом пользователя, в котором будут реализованы:

- отображение состояния систем: статус готовности дрона к полету, уровень заряда батареи в процентах, уровень заряда батареи в вольтах;
- кнопки управления миссией:
  - ◇ старт — запуск автономной миссии **с начала,**
  - ◇ стоп[1] — прерывание кода, остановка в воздухе (без посадки),
  - ◇ `killswitch`[2] — прерывание кода, дизарм дрона (кнопка активна в любой момент работы web-приложения),
  - ◇ `home` — возврат квадрокоптера в зону взлета/посадки и выполнение посадки,
  - ◇ `land` — посадка в текущей точке, в процессе осуществления автономной миссии посадка доступна после нажатия кнопки стоп/прерывания кода;
- автоматически обновляемая 2D-карта:
  - ◇ фон-карта ArUco-маркеров из топика `aruco_map/image` с отображением начала координат,
  - ◇ вывод объектов распознавания (требуемых в заданиях 1.2 и 1.3) с привязкой к системе координат `aruco-map`,
  - ◇ список участков, содержащий координаты начала участка, координаты конца участка, максимальную температуру на данном участке (для задания 1.2),
  - ◇ список координат незаконных врезок (для задания 1.3).

Команда вправе сама выбирать как используемые библиотеки и фреймворки, так и то, где хостить веб-приложение (ПК или квадрокоптер) и делать выбор между собственным бэкендом или готовым адаптером к ROS.

Напоминаем, что участники должны загрузить код зачетной попытки с комментариями на GitHub **за 10 мин** до начала зачетных полетов.

Оценка результатов

В задании присутствуют требования визуализации некоторых частей полигона. Порядок оценки визуализации:

1. Для оценивания визуализации на зачетной попытке команде необходимо представить экспертам компьютер с открытой системой визуализации.
2. В визуализации должны присутствовать отображения начала системы координат `aruco_map`.
3. По требованию экспертов участники быть готовыми (иметь настроенное для этого ПО, например, OBS) запустить запись экрана в начале зачетных полетов команды.

[1] Проверка работоспособности осуществляется во время тестовых полетов.
[2] Проверка работоспособности осуществляется во время тестовых полетов.

4. По окончании полета все визуализированные элементы должны остаться на своих местах.

Конструкторская часть

Конструкторское задание включает разработку двух устройств, расширяющих функционал квадрокоптера, и инфраструктуры, позволяющей ему выполнять задачи по наблюдению за объектами ТЭК.

2.1 Разработка крепления инфракрасной камеры

1. **Наименование выполняемых работ**

   Разработка крепления инфракрасной камеры TR256i на квадрокоптер.

2. **Цель выполнения работ**

   Установить датчик, позволяющий квадрокоптеру определять тепловые потери системы теплоснабжения.

3. **Срок выполнения работы**

   Документация и создание устройства: три рабочих дня, с 01.04.2025 г. по 03.04.2025 г. 19.00.

4. **Требования к устройству:**

   - Устанавливаться на квадрокоптер COEX Clever 4 без доработки конструкции углепластиковой рамы. (Изменение конструкции защиты пропеллеров возможна в случае сохранения ее защитных свойств). **Доработкой является нарушение целостности материала конструкции.**
   - Быстрая установка тепловизионной камеры (10 с установка и подключение (физическое), 10 с отключение (физическое) и съем).
   - Кронштейн должен надежно крепить тепловизионную камеру исключая возможность отключения и отсоединения во время полета, транспортировки и при падениях квадрокоптера.

5. **Требования к документации:**

   - 3D-модель устройства в сборке (`.step` и исходный формат программы).
   - В случае если кронштейн представляет собой одну деталь — чертеж детали.
   - В случае если кронштейн является сборной конструкцией — сборочный чертеж.
     Список крепежа, необходимого для установки кронштейна на квадрокоптер, нужно указать на чертеже/сборочном чертеже.

2.2 Разработка дронпорта

Техническое задание

1. **Наименование выполняемых работ**

Разработка дронпорта — устройства для безопасного взлета, посадки и подзарядки квадрокоптера.

2. **Цель выполнения работ**

   Создание устройства для осуществления безопасного взлета, посадки и подзарядки квадрокоптера с интеграцией всех необходимых функций и соблюдением требований безопасности.

3. **Срок выполнения работ**

   - Документация и создание устройства: три рабочих дня, с 01.04.2025 г. по 03.04.2025 г. 19.00.
   - Демонстрация работоспособности прототипа устройства: с 13:30 04.04. 2025 г., в 12:20 разработанное устройство сдается на карантин.
   - Финальное испытание устройства: выполнение зачетного полета 04.04. 2025 г.

4. **Требования к устройству:**

   - Взлет и посадка квадрокоптера: обеспечение безопасности взлета и посадки квадрокоптера.
   - Безопасность: обеспечение безопасности операций с квадрокоптером, включая защиту от несанкционированного доступа и вмешательства. Квадрокоптер должен быть защищен от внешнего воздействия со всех сторон.
   - Сигнализирование о нахождении квадрокоптера внутри устройства.
   - Сигнализирование об открытии устройства для совершения взлета и посадки квадрокоптера.
   - Сигнализирование о закрытии устройства после совершения взлета и посадки.
   - Сигнализирование об осуществлении подзарядки квадрокоптера. Считается, что дрон заряжается в случае, если он находится внутри закрытого дронпорта более 5 с.
   - Устройство должно быть единым (все датчики и механическая часть должны быть интегрированы в данное устройство).
   - Максимальные габариты устройства в закрытом состоянии: $800 \times 800 \times \times 800$ мм.
   - Все вышеперечисленные функции должны выполняться в автономном режиме.
   - Сигнализация на устройстве должна быть реализована при помощи при адресной светодиодной матрицы/светодиодной ленты.

5. **Требования к документации:**

   - 3D-модель устройства в сборке (`.step` и исходный формат программы).
   - Сборочный чертеж устройства, оформленный согласно ГОСТ 2.109-73: https://dgng.pstu.ru/sprav/10.4.htm:
     - ◇ отображение всех деталей и стандартных изделий;
     - ◇ количество изображений должно быть наименьшим, но достаточным для представления расположения и взаимной связи состав-

ных частей и обеспечивающим возможность осуществления сборки и контроля сборки.

- Спецификация, оформленная согласно ГОСТ 2.109-73: https://dgng.pstu.ru/sprav/10.4.htm:
  - ◇ наличие всех деталей;
  - ◇ наличие стандартных изделий.
- Инструкция по сборке устройства (все пункты относятся как к сборке механической части устройства, так и электрической):
  - ◇ список всех компонентов и материалов, необходимых для сборки устройства;
  - ◇ последовательное описание шагов сборки, начиная с подготовки рабочего места и инструментов;
  - ◇ иллюстрации или схемы, помогающие понять процесс сборки;
  - ◇ подробное описание каждого шага с указанием порядка действий и возможных трудностей;
  - ◇ инструкции по проверке правильности сборки и испытания устройства;
  - ◇ рекомендации по безопасности при работе с материалами и инструментами.
- Инструкция по эксплуатации устройства:
  - ◇ описание устройства: общие характеристики, назначение, особенности конструкции;
  - ◇ инструкции по началу эксплуатации: правила подключения, настройки и первоначального запуска;
  - ◇ рекомендации по использованию и безопасности: ограничения по применению, рекомендации по уходу и обслуживанию, предупреждения о потенциальных опасностях, правила безопасного использования изделия;
  - ◇ программный код, необходимый для эксплуатации устройства.

2.3 Требования к оформлению конструкторской части

Таблица 4.3.2. Документация

№ п.п	Наименование	Формат
Кронштейн тепловизионной камеры		
1	3D-модель функционального устройства в сборке	`.step` и проприетарный формат
2	Чертеж/сборочный чертеж кронштейна	`.pdf`
Дронпорт		
1	3D-модель функционального устройства в сборке	`.step` и проприетарный формат
2	Сборочный чертеж устройства	`.pdf`
3	Спецификация	`.pdf`

№ п.п	Наименование	Формат
4	Инструкция по сборке устройства	`.pdf`
5	Инструкция по эксплуатации устройства	`.pdf`

Во время демонстрации экспертами проверяется соответствие устройства требованиям из п. 4.

Финальное испытание устройства (дронпорта)

Финальное испытание устройства проходит во время выполнения финальной зачетной миссии. Оценка результатов происходит только в тех случаях, если все составляющие итогового результата:

- предоставлены организаторам не позднее указанного срока;
- все разработанные материалы и само устройство соответствуют техническому заданию.

Оценка результатов

Оценка результатов происходит только в тех случаях, если все составляющие итогового результата:

- предоставлены организаторам не позднее указанного срока;
- все разработанные материалы и само устройство соответствуют техническому заданию.

4.3.6. Система оценивания

Во время инженерного тура участники получают баллы за выполнение зачетных миссий согласно критериям оценки успешности прохождения миссии.

Максимальный балл за инженерный тур составляет 100 баллов.

Оценка инженерного задания разделена на две части:

- оценка работоспособности разработанного устройства (проверяется непосредственно во время проведения зачетной миссии);
- оценка технической документации разработанного устройства и его соответствие техническому заданию.

Определение команды-победителя в заключительном этапе проводится по результатам суммы баллов за зачетные попытки и инженерную часть. Оценка результатов зачетных испытаний происходит после окончания последней зачетной попытки. При несоблюдении правил приема результатов за каждое отдельное нарушение снимаются от 30% до 70% баллов. С команды снимаются 20 баллов (за каждое нарушение), на усмотрение организатора, в случаях:

- препятствования работе других команд, разработчиков и организаторов;
- употребления ненормативной лексики;
- агрессивного поведения (этически ненормированного);

- осуществления запуска квадрокоптера без предупреждения оператора и организаторов;
- нарушений техники безопасности при эксплуатации оборудования (квадрокоптер, 3D-принтер, лазерный станок и т. д.), в случае грубого нарушения участник может быть дисквалифицирован.

Критерии оценивания успешности прохождения миссии

Таблица 4.3.3. Документация

№	Критерий	Баллы за один случай	Количество случаев*	Баллы за все случаи
1. Подготовка к полету и взлет				
1.1	Соблюдение техники безопасности. По умолчанию полный балл — 0,1. За случай: • разряженный аккумулятор менее 14 В, • подключение АКБ вне полетной зоны с подключенными двигателями и установленными пропеллерами, • полет вне полетной зоны, • полеты в полетной зоне при нахождении в ней людей, попадание объектов в область вращения пропеллеров.	0,7	—	0,7
1.2	Выполнилась функция сигнализирования** об открытии устройства перед совершением взлета квадрокоптера. Функция может осуществляться только в автономном режиме.	0,6	1	0,6
1.3	Крышка дронпорта открывается перед осуществлением взлета в режиме: • в автономном (связь с квадрокоптером, компьютером) — 100%; • полуавтономном режиме (управление через кнопки, web-интерфейс, подобные устройства) — 50%; • в ручном режиме (внешнее воздействие: любые механические способы воздействия на устройство) — 20%. Нахождение участников в полетной зоне во время попыток строго запрещено.	1	1	1
1.4	Совершен автономный взлет из дронпорта (с дронпорта 20 %/ с зоны «Н» 10%).	1	1	1

№	Критерий	Баллы за один случай	Количество случаев*	Баллы за все случаи
1.5	Крышка дронпорта закрывается после осуществлением взлета: • в автономном (связь с квадрокоптером, компьютером) — 100%; • в полуавтономном режиме (управление через кнопки, web-интерфейс, подобные устройства) — 50%; • в ручном режиме (внешнее воздействие: любые механические способы воздействия на устройство) — 20%. Нахождение участников в полетной зоне во время попыток строго запрещено.	1	1	1
1.6	Выполнилась функция сигнализирования** о закрытии устройства после совершения взлета квадрокоптера. Функция может осуществляться только в автономном режиме.	0,6	1	0,6
2. Определение тепловых потерь системы теплоснабжения				
2.1	Выполнен полет к началу участка теплосети.	0,1	1	0,1
2.2	Выполнена запись видео пролета над участком теплосети (с тепловизора).	1,5	1	1,5
2.3	Выполнена запись видео пролета над участком теплосети (с камеры).	0,5	1	0,5
2.4	Верно определены координаты начала участка теплопотерь; возможная погрешность 0,25 м по каждой оси.	3	—	3
2.5	Верно определены координаты конца участка теплопотерь; возможная погрешность 0,25 м по каждой оси.	3	—	3
2.6	Определена максимальная температура каждого участка теплопотерь.	2	—	2
2.7	Выполнен полет к концу трубы.	0,1	1	0,1
3. Наблюдение за безопасностью нефтепровода				
3.1	Данные с сервера получены.	0,5	1	0,5
3.2	Данные с сервера получены (только список легальных ответвлений). Балл выставляется только в случае хотя бы вывода одной корректной нелегальной врезки (критерий 3.3) и отсутствия запросов к `coords_1.json` (только `coords_1_legal.json`).	2	1	2
3.3	Выполнен полет к началу нефтепровода.	0,4	1	0,4
3.4	Выведен список координат незаконных врезок, врезки определены корректно; возможная погрешность 0,25 м по каждой оси. В случае если выведенное количество врезок в списке превышает реальное количество, за каждую лишнюю врезку — 0,3 балла.	3	—	3
3.5	Получена фотография незаконной врезки в нефтепровод.	0,5	—	0,5

№	Критерий	Баллы за один случай	Количество случаев*	Баллы за все случаи
3.6	Получена фотография незаконной врезки в нефтепровод с ее выделением.	2	—	2
4. Разработка веб-приложения				
4.1	Реализовано отображение уровня батареи в процентах.	0,15	1	0,15
4.2	Реализовано отображение уровня батареи в вольтах.	0,1	1	0,1
4.3	Реализована и функционирует кнопка запуска миссии «старт».	0,3	1	0,3
4.4	Реализована и функционирует кнопка остановки миссии «стоп». Проверка работоспособности осуществляется во время тестовых полетов.	0,3	1	0,3
4.5	Реализована и функционирует кнопка экстренного отключения `killswitch`. Проверка работоспособности осуществляется во время тестовых полетов.	0,3	1	0,3
4.6	Реализована и функционирует кнопка посадки `land`.	0,3	1	0,3
4.6	Реализована и функционирует кнопка прерывания миссии и возврата в точку взлета `home`.	0,3	1	0,3
	Уровень батареи и кнопки не проверяются и устанавливается максимальный балл, если в предыдущие дни за эту кнопку получен максимальный балл.			
4.7	Автоматически обновляемая 2D-карта: фон-карта ArUco-маркеров из топика `aruco_map/image` с отображением начала координат.	0,2	1	0,2
4.8	Автоматически обновляемая 2D-карта: вывод распознаваемых участков из задания 1.2 с привязкой к системе координат `aruco-map`.	0,5	—	0,5
4.10	Отображение списка участков, содержащего координаты начала участка, координаты конца участка, максимальную температуру на данном участке из задания 1.2.	0,5	—	0,5
4.11	Автоматически обновляемая 2D-карта: вывод нефтепровода и врезок (цвет отображен корректно) из задания 1.3 с привязкой к системе координат aruco- map.	0,5	—	0,5
4.12	Отображение списка врезок, содержащего координаты из задания 1.3.	0,5	—	0,5
5. Посадка: оценивается во время полета				
5.1	Осуществлена посадка квадрокоптера: • внутрь устройства — 100% баллов; • на устройство — 50% баллов; • в точку «H» — 20% баллов.	2	—	2
5.2	Выполнилась функция сигнализирования** об открытии устройства перед совершением посадки квадрокоптера. Функция может осуществляться только в автономном режиме.	0,6	1	0,6

№	Критерий	Баллы за один случай	Количество случаев*	Баллы за все случаи
5.3	Крышка дронпорта открывается перед осуществлением посадки: • в автономном (связь с квадрокоптером, компьютером) — 100%; • полуавтономном режиме (управление через кнопки, web-интерфейс, подобные устройства) — 50%; • в ручном режиме (внешнее воздействие: любые механические способы воздействия на устройство) — 20%. Нахождение участников в полетной зоне во время попыток строго запрещено.	1	1	1
5.4	Выполнилась функция сигнализирования** о закрытии устройства после совершения посадки квадрокоптера. Функция может осуществляться только в автономном режиме.	0,6	1	0,6
5.5	Крышка дронпорта закрывается после осуществлением посадки: • в автономном (связь с квадрокоптером, компьютером) — 100%; • в полуавтономном режиме (управление через кнопки, web-интерфейс, подобные устройства) — 50%; • в ручном режиме (внешнее воздействие: любые механические способы воздействия на устройство) — 20%. Нахождение участников в полетной зоне во время попыток строго запрещено.	1	1	1
5.6	Выполнилась функция сигнализирования** о нахождении квадрокоптера внутри устройства. Функция может осуществляться только в автономном режиме.	0,6	1	0,6
5.7	Выполнилась сигнализация осуществления подзарядки квадрокоптера в дронпорте. Функция может осуществляется только в автоматическом режиме.	0,6	1	0,6
6. Оценка результата				
6.1	Код зачетной попытки с комментариями выложен на Github.	0,5	1	0,5
6.2	В программе присутствует логика считывания данных с датчика (любого) или алгоритм компьютерного зрения.	0,5	1	0,5
6.3	В присутствует логика считывания температуры с тепловизора.	1	1	1

№	Критерий	Баллы за один случай	Количество случаев*	Баллы за все случаи
6.4	В форму для участников не позднее завершения финальных зачетных попыток отправлен отчет, содержащий: • техническое описание — доступное и понятное описание статьи на GitHub (с подробным описанием решенных задач и использованных методов), где прикреплен программный код всех зачетных попыток с аннотациями типов у методов и функций и понятными и подробными комментариями; • описание за что отвечает каждый файл программного кода; • краткое описание всех алгоритмов, которые используются в коде; • документацию разработанного устройства (3D-модель, изображение 3D-модели и самого устройства, видеодемонстрация работающего устройства, программный код для устройства и вся требуемая — в инженерной части документация); • фото- и видеоматериалы работы команды (фотография командная).	Как при отсутствии итогового отчета от команды, так и при непредоставлении итогового отчета в положенный срок — взимается 70% баллов от суммы полученных баллов за все 3 зачетных попытки.		
Итого за решение задачи				34,85

**Примечания*

***Сигнализирование дронпорта:*

- функция реализована при помощи светодиодов — 50% баллов;
- функция реализована при помощи светодиодной ленты — 80% баллов;
- функция реализована при помощи светодиодной LED-матрица — 100% баллов.

Количество случаев может варьироваться. В случае изменения количества баллы за все случаи делятся на их количество. При «—» общее количество случаев неизвестно, баллы за все случаи делятся на их количество.

Документация. Отчет выполнения инженерной части

Таблица 4.3.4

№	Критерии	Баллы
Кронштейн тепловизионной камеры		
1. 3D-модель функционального устройства в сборке		
1.1	Оригинальное инженерное решение.	1,25
1.2	Наличие 3D-модели.	0,25
1.3	Заданы материалы деталей.	0,25

№	Критерии	Баллы
2. Сборочный чертеж устройства (если кронштейн является сборной конструкцией) 2.*Чертеж детали (если кронштейн является деталью) Проверяется только в случае соответствия ГОСТ 2.109-73.		
2.1	Количество изображений достаточно для представления расположения и взаимной связи составных частей и обеспечивает возможность осуществления сборки и контроля сборки. Пронумерованы детали. *Присутствуют все необходимые элементы оформления чертежа.	0,75
2.2	Указаны все габаритные размеры и все размеры присоединений. *Указаны все необходимые размеры.	0,5
Дронпорт		
1. 3D-модель функционального устройства в сборке		
1.1	Оригинальное инженерное решение.	0,5
1.2	Наличие 3D-модели.	0,5
1.3	Наличие всех элементов устройства в соответствии с документацией (спецификация, сборочный чертеж, инструкция по сборке).	1
1.4	Заданы материалы деталей.	0,25
1.5	Деталям задана цветовая палитра позволяющая лучше читать сборку (сопрягаемые детали имеют различную цветовую палитру).	0,25
2. Сборочный чертеж устройства Проверяется только в случае соответствия ГОСТ 2.109-73.		
2.1	Отображены все детали и их нумерация.	0,75
2.2	Отображены все стандартные изделия и их нумерация.	0,75
2.3	Количество изображений достаточно для представления расположения и взаимной связи составных частей и обеспечивает возможность осуществления сборки и контроля сборки.	0,75
2.4	Указаны все габаритные размеры и все размеры присоединений.	0,5
3. Спецификация Проверяется только в случае соответствия ГОСТ 2.109-73.		
3.1	Наличие всех деталей (в соответствии со сборочным чертежом).	1
3.2	Наличие стандартных изделий (в соответствии со сборочным чертежом).	1
4. Инструкция по сборке устройства		
4.1	Наличие списка всех компонентов и материалов, необходимых для сборки устройства.	0,5
4.2	Наличие последовательного описания шагов сборки, начиная с подготовки рабочего места и инструментов.	0,5
4.3	Наличие иллюстраций и/или схемы, помогающих понять процесс сборки.	0,5
4.4	Наличие подробного описания каждого шага с указанием порядка действий и возможных трудностей.	0,5
4.5	Наличие инструкций по проверке правильности сборки и испытания устройства.	0,25
4.6	Наличие рекомендаций по безопасности при работе с материалами и инструментами.	0,25
5. Инструкция по эксплуатации устройства		
5.1	Наличие описания устройства: общих характеристик, назначения, особенности конструкции.	0,5

№	Критерии	Баллы
5.2	Наличие инструкции по началу эксплуатации: правила подключения, настройки и первоначального запуска.	0,5
5.3	Наличие рекомендаций по использованию и безопасности: ограничения по применению, рекомендации по уходу и обслуживанию, предупреждения о потенциальных опасностях, правила безопасного использования изделия.	0,75
5.4	Наличие программного кода, необходимого для эксплуатации устройства.	0,5
Итого за решение задачи		15

Демонстрация работоспособности прототипа устройства

Таблица 4.3.5

№	Критерии	Баллы
Кронштейн тепловизионной камеры		
1. Функциональные возможности		
1.1	Установка кронштейна на квадрокоптер не требует существенной доработки конструкции.	0,2
1.2	Наличие фиксатора предохраняющего от отсоединения камеры при падениях.	0,2
1.3	Время установки и подключения менее 10 с. Если время более 20 с, то баллы за п. 1 не выставляются.	0,4
1.4	Время отключения и съема менее 10 с. Если время более 20 с, то баллы за п. 1 не выставляются.	0,4
Дронпорт		
2. Функциональные возможности		
2.1	Квадрокоптер может осуществлять функцию взлета с устройства.	0,25
2.2	Квадрокоптер может осуществлять функцию посадки в/на устройство.	0,25
2.3	Безопасность: • квадрокоптер защищен от внешнего воздействия со всех сторон (вертикальные стенки, посадочная площадка и крышка); • при отсутствии крышки баллы уменьшаются на 60% баллов; • при отсутствии одной из вертикальных стенок баллы уменьшаются на 10% баллов.	1,75
2.4	Устройство является цельным (все датчики и механическая часть интегрированы в устройство).	1
2.5	Надежность конструкции: • все элементы конструкции закреплены надежно, отсутствуют люфты; • изолента, скотч (все виды) и клей не являются крепежным материалами.	0,75
2.6	Правильно выполнен кабель-менеджмент: • кабели закреплены на/в корпусе; • отсутствует провисание проводов; • все контакты защищены изоляцией.	0,75

№	Критерии	Баллы
2.7	• Все функции п. 4 технического задания выполняются в автономном режиме — 100% баллов. • Более 50% функций п.4 технического задания выполняются в автономном режиме — 70% баллов. • Менее 50% функций п. 4 технического задания выполняются в автономном режиме — 50% баллов. • Все функции выполняются не в автоматическом режиме — 10% баллов.	3
2.8	Устройство позволяет осуществлять функцию открытия и закрытия одной из поверхностей устройства для осуществления взлета и посадки квадрокоптера внутрь устройства. Использование механической силы участников запрещено.	2
2.9	Наличие центрирования квадрокоптера внутри устройства. Использование механической силы участников запрещено.	1,5
3. Сигнализирование статусов дронпорта Способ отображения сигнализирования: • анимированные пиксельные иконки — коэффициент 1; • статичные пиксельные иконки — коэффициент 0,7; • текстовые/буквенное сообщения — коэффициент 0,4; • цветовое обозначения — коэффициент 0,2.		
3.1	Выполняется функция сигнализирования о нахождении квадрокоптера внутри устройства.	0,75
3.2	Выполняется функция сигнализирования об открытии устройства перед совершением взлета квадрокоптера.	0,75
3.3	Выполняется функция сигнализирования о закрытии устройства после совершения взлета квадрокоптера.	0,75
3.4	Выполняется функция сигнализирования об открытии устройства перед совершением посадки квадрокоптера.	0,75
3.5	Выполняется функция сигнализирования о закрытии устройства после совершения посадки квадрокоптера.	0,75
3.6	Выполняется функция сигнализирования об осуществлении подзарядки квадрокоптера. В случае отображения данной сигнализации сигнализация о нахождении квадрокоптера внутри устройства не требуется.	0,75
3.7	Оригинальность	1
Итого за решение задачи		17,95

4.3.7. Решение задачи

Программирование квадрокоптера

Работа с квадрокоптером

Первым шагом при получении квадрокоптера является его тщательная проверка и подготовка к работе, которую можно разделить на следующие этапы:

1. Выполнение калибровки датчиков квадрокоптера (например, компаса и гироскопа) согласно рекомендациям производителя.

2. Настройка карты ArUco-маркеров в соответствии с реальным полем на площадке.
3. Подтверждение того, что все системы квадрокоптера (включая полетный контроллер, Raspberry Pi, камеру, систему навигации по ArUco-маркерам) работают исправно. Для этого можно воспользоваться инструментом `self-check`, согласно инструкции на сайте https://clover.coex.tech/ru/selfcheck.html; а также `web video server`, который позволяет выполнить визуальную проверку.

1.1. Подключение и настройка тепловизора

Для успешного выполнения задания по подключению и настройке тепловизора сначала необходимо изучить несколько важных моментов. Рекомендуется начать с поиска информации по следующим вопросам:

- Работа с видеопотоком тепловизора — ознакомиться с примерами получения и обработки видеопотока (обычно это RAW-данные, которые необходимо конвертировать для визуализации).
- Преобразование RAW-данных в температуру — изучить документацию на тепловизор: какие формулы необходимы для пересчета «сырых» значений пикселей в физические температуры в градусах Цельсия.
- Преобразование в цветовые схемы — найти методы визуализации термальных изображений в цветовых палитрах JET или IRONBOW (могут быть готовые функции в OpenCV, matplotlib и других библиотеках).

Для поиска этого достаточно сделать запрос в поисковую систему:

- tr256i linux;
- tr256i github;
- tr256i code.

При поиске информации по этим темам рекомендуется обратить внимание на полезные ресурсы:

- Gist на GitHub (https://gist.github.com/marcelrv/e81253c14053bcd78753554df1230dd3), содержащий ссылки на готовые решения;
- тред о похожем тепловизоре на форуме (https://www.eevblog.com/forum/thermal-imaging/infiray-and-their-p2-pro-discussion/200/), содержащий опыт reverse-инжиниринга тепловизора такого типа.

После изучения найденных источников, а также тестирования нескольких готовых решений, приходим к выводу, что код, представленный в репозитории https://github.com/leswright1977/PyThermalCamera, хорошо работает с используемым тепловизором.

При изучении кода и/или треда на форуме, извлекаем полезную информацию о формате данных, получаемых с тепловизора.

Тепловизор TR256i выдает видеоформат `yuyv422`, разрешение 256×384, в котором:

- **Верхняя половина кадра (256 × 192)** содержит нормализованное 8-битное градационное изображение, предобработанное камерой, обеспечивая черно-

белый вид. Для перевода верхней половины кадра (8-битного черно-белого изображения, полученного с тепловизора) в цветовую схему JET достаточно воспользоваться следующей строкой на языке Python с использованием библиотеки OpenCV:

Python

```python
cv2.applyColorMap(gray_image, cv2.COLORMAP_JET)
```

Здесь `gray_image` — это `numpy`-массив, содержащий 8-битное изображение, а результатом будет изображение с наложенной цветовой картой JET, что позволяет визуально выделить диапазоны температур в привычной палитре от синего к красному. Однако для получения матрицы температур, верхней матрицы недостаточно из-за того, что она содержит уже предобработанные цветовые значения.

- **Нижняя половина кадра (256 × 192)** содержит данные температуры напрямую в Кельвинах. Они представлены 14 битами, смещенными в старшие биты, и умноженными на коэффициент 16. Для приближенного перевода значений из нижней половины кадра в градусы Цельсия можно применить формулу:

$$T(x) \approx \frac{x}{64} - 273{,}15.$$

Это необходимо сделать с каждым пикселем. Удобнее всего применить формулу ко всему массиву сразу с помощью `numpy` (см. ниже).

Python

```python
# thermal_raw — numpy-массив размера (192, 256) с 14-битными температурными значениями
temperature_celsius = (thermal_raw / 64) - 273.15
```

Для тесной интеграции с ROS рекомендуется вынести этот процесс в отдельную ROS-ноду, чтобы можно было извне получать доступ к топикам обработанных данных — см. ниже.

Python

```python
import rospy
from cv_bridge import CvBridge
from sensor_msgs.msg import Image
import cv2
import numpy as np
# 1. Инициализация ноды и паблишеров
rospy.init_node('thermal')
pub_raw     = rospy.Publisher('/thermal/img_raw',    Image, queue_size=1)
pub_jet     = rospy.Publisher('/thermal/img_jet',    Image, queue_size=1)
pub_celsius = rospy.Publisher('/thermal/celsius',    Image, queue_size=1)
# 2. Открываем видеопоток с камеры (ID может отличаться)
cap = cv2.VideoCapture(1)
# Отключаем встроенную конвертацию в RGB — нам нужны сырые данные
cap.set(cv2.CAP_PROP_CONVERT_RGB, 0)
bridge = CvBridge()
rate = rospy.Rate(30)
```

```
while not rospy.is_shutdown():
    ok, img_raw = cap.read()
    if not ok:
        rospy.logwarn("Не удалось получить кадр с тепловизора")
        rate.sleep()
        continue
    # 3. Преобразуем в одномерный массив и затем в 16-битную матрицу (384×256)
    img_raw = img_raw.flatten()
    img16   = img_raw.view(dtype=np.uint16).reshape((384, 256))
    # 4. Делим на верхнюю (визуализация) и нижнюю (сырые температуры) половины
    img_vis, img_th = np.array_split(img16, 2)
    # Верхняя половина: уже нормированная 8-бит ч/б картинка
    img_vis = img_vis.astype(np.uint8)
    # Нижняя половина: 14-бит данные, смещённые в старшие биты и умноженные на 16
    kelvins = temp_raw / 64.0
    celsius = kelvins - 273.15
    # Чтобы отправить в виде 8-битного изображения, нормируем и обрезаем диапазон 0-255
    celsius_img = np.clip(celsius, 0, 255).astype(np.uint8)
    # 5. Строим JET-карту по верхнему изображению
    img_jet = cv2.applyColorMap(img_vis, cv2.COLORMAP_JET)
    # 6. Публикуем результаты
    pub_raw.publish(bridge.cv2_to_imgmsg(img_vis, encoding="mono8"))
    pub_jet.publish(bridge.cv2_to_imgmsg(img_jet, encoding="bgr8"))
    pub_celsius.publish(bridge.cv2_to_imgmsg(celsius_img, encoding="mono8"))
    # 7. Логгируем средние и максимальные температуры для отладки
    rospy.loginfo(f"Средняя t: {celsius.mean():.2f} °C, макс t: {celsius.max():.2f} °C")
    rate.sleep()
cap.release()
```

В задании используется топик типа `sensor_msgs/Image` для публикации матрицы температур, потому что `Image` — это стандартный формат ROS для передачи двумерных массивов данных, и его удобно обрабатывать с помощью существующих инструментов ROS (например, визуализировать через `rqt_image_view`, записывать в `rosbag` или быстро преобразовывать в `numpy`-массив через `cv_bridge`). Это позволяет легко интегрировать матрицу температур в ROS-инфраструктуру и применить стандартные средства для дальнейшего анализа или визуализации.

1.2. Определение тепловых потерь системы теплоснабжения

Взлет и полет к точке выполняются стандартными функциями квадрокоптера Clover. После прибытия в начальную точку обследуемого участка запускается одновременная запись видео с тепловизора (в цветовой палитре JET) и с обычной камеры, чтобы зафиксировать весь маршрут пролета для последующего анализа.

Для выполнения записи рекомендуется использовать встроенный функционал OpenCV — VideoWriter.

Большинство современных контейнеров для хранения видеофайлов требуют корректного завершения записи (вызова `release` у VideoWriter/OpenCV, `close` у `ffmpeg` и т. п.), чтобы записать служебную информацию в шапку файла, закрыть потоки, вычислить индексы и т. д.

Для надежной работы важно корректно завершать работу с VideoWriter с помощью метода `.release()`, причем делать это даже в случае возникновения исключений. Для этого чаще всего используют конструкции `try...finally`.

Аналог структуры `try...finally` можно реализовать с помощью менеджеров контекста (см. ниже).

Python

```python
import contextlib
@contextlib.contextmanager
def safe_video_writer(*args, **kwargs):
    out = cv2.VideoWriter(*args, **kwargs)
    try:
        yield out
    finally:
        out.release()
# Использование:
with safe_video_writer('result.mp4', fourcc, 30.0, (640, 480)) as out:
    out.write(frame)
    # и т.д.
```

Рекомендуется использовать контейнер `AVI` с кодеком `MJPEG`, который считается более устойчивым при записи данных в полевых условиях или нестабильных системах, что особенно актуально для задач, где важна сохранность даже частично записанной информации. В такой связке OpenCV сразу пишет кадры как набор `RIFF`-чанков с индексами внутри данных, а не откладывает всю таблицу оглавления до закрытия файла. Если код аварийно завершится, плееры все равно смогут воспроизвести записанную часть, а индекс легко восстановить при открытии.

Пример записи в `AVI` с использованием `MJPG` см. ниже.

Python

```python
fourcc = cv2.VideoWriter_fourcc(*'MJPG')  # Кодек для AVI (можно попробовать 'MJPG')
out = cv2.VideoWriter('output.avi', fourcc, 20.0, (frame_width, frame_height))
```

Далее дрон равномерно движется вдоль макета трубы с заданной невысокой скоростью, чтобы обеспечить детальное сканирование всей поверхности. Во время пролета происходит покадровый анализ теплового изображения: из каждой тепловой карты выделяются области с температурой выше 40 °C.

Для определения координат границ участков теплопотерь может использоваться как простая система с детекцией по маске, выравнивания к нему по двухосевому (x и y) ПИД-регулятору и получение текущих координат дрона или более сложная с использованием геометрии изображения с камеры (см. https://waksoft.susu.ru/2020/02/23/geometriya-formirovaniya-izobrazhenij/), однако этот метод имеет сложности при работе с тепловизором, поскольку для него отсутствует матрица калибровки.

Примерный код выполнения полета вдоль трубы и определения точек участка теплопотерь см. ниже.

Python

```python
DIFF = 1.25 # отношение размеров термо к камере
camera_model = image_geometry.PinholeCameraModel()
camera_model.fromCameraInfo(rospy.wait_for_message('main_camera/camera_info', CameraInfo))
def img_xy_to_point(xy, dist):
    xy_rect = camera_model.rectifyPoint(xy)
    ray = camera_model.projectPixelTo3dRay(xy_rect)
    return Point(x=ray[0] * dist, y=ray[1] * dist, z=dist)
# Трансформирование точки с помощью tf2 из координат камеры в координаты на aruco_map
def transform_point(x, y, z, data):
    xy3d = img_xy_to_point((x, y), z)
    target = PointStamped(header=data.header, point=xy3d)
    return tf_buffer.transform(target, "aruco_map", timeout=rospy.Duration(0.2))
# Проекция точки на перпендикуляр к отрезку
def perpendicular_project(start, end, point):
    x1, y1 = start
    x2, y2 = end
    px, py = point
    # Вектор AB
    ab_x = x2 - x1
    ab_y = y2 - y1
    # Вектор AP
    ap_x = px - x1
    ap_y = py - y1
    # Скалярное произведение AP и AB
    dot_product = ap_x * ab_x + ap_y * ab_y
    ab_length_sq = ab_x ** 2 + ab_y ** 2
    # Если отрезок вырожден (точки совпадают)
    if ab_length_sq == 0:
        return (x1, y1)
    # Параметр t для проекции
    t = dot_product / ab_length_sq
    t = max(0, min(1, t))  # Ограничение t между 0 и 1
    # Находим проекцию точки на отрезок
    qx = x1 + t * ab_x
    qy = y1 + t * ab_y
    return (qx, qy)
res = navigate(x=PIPE_END[0], y=PIPE_END[1], z=1, yaw=math.pi / 2, speed=0.25, frame_id='aruco_map', auto_arm=False)
while not rospy.is_shutdown():
    img, img_data = get_image()
    out.write(img)
    jet, _ = get_jet()
    out_jet.write(jet)
    cels, _ = get_celsius()
    mx = int(cels.max())
    max_tmp.publish(String(str(mx)))
    telem = get_telemetry(frame_id='navigate_target')
    if math.sqrt(telem.x ** 2 + telem.y ** 2 + telem.z ** 2) < 0.2:
        break
    telem_jet = get_telemetry(frame_id='aruco_map')
    cels_mask = cv2.inRange(cels, 40, 255)
    pts_resp = []
    P = list(np.argwhere(jet_mask != 0))[::300]
    for p in P:
        PX, PY = int(p[0] * DIFF), int(p[1] * DIFF)
        pt = transform_point(PX, PY, telem_jet.z, img_data)
```

```
56          pts_resp.append(list(perpendicular_project(PIPE_START,
            ↪  PIPE_END, (pt.point.x, pt.point.y))))
57      metal_pts.publish(String(json.dumps(pts_resp)))
58      print(f"still going..., pushed array of {len(pts_resp)} points and
        ↪  {mx} max temp")
59      rospy.sleep(0.05)
```

1.3. Наблюдение за безопасностью нефтепровода

Первым этапом решения этой подзадачи является получение данных с сервера.

```
pipe_data = requests.get("http://157.180.22.113:8080/coords_1.json",
↪  headers={"Authorization": "Bearer xxx-xxx-xxx"}).json()
```

В случае получения данных обо всех возможных врезках остается только выполнить полет по точкам возможных врезок и проверить каждую точку, не помеченную как законную, на наличие врезки.

Поскольку труба является прямой, пролет вдоль нее является тривиальной задачей.

Рассмотрим подробнее алгоритм автоматического обнаружение врезки по изображению с камеры, который может быть выполнен с помощью анализа цвета и геометрии выявленных областей:

1. Изображение преобразуется в цветовое пространство HSV для более устойчивого выделения нужных цветов.
2. Среди найденных на изображении контуров ищется контур характерного желтого цвета (цвет трубы).
3. Вокруг предполагаемого центра врезки строится вырезанный фрагмент (предполагается, что труба всегда находится в центре кадра), к которому применяется цветовая маска, выделяющая интересующий оттенок.
4. Чтобы убрать небольшие шумы, маска изображения дополнительно обрабатывается эрозией.

На полученной маске снова ищутся контуры; если таковые есть, выбирается наиболее крупный. Для этого контура вычисляется минимальный ограничивающий прямоугольник и его площадь, а также площадь найденного контура. Критерием выделения врезки служит то, что реальная площадь контура значительно меньше площади описывающего прямоугольника (отношение меньше 0,6).

В случае получения данных с сервера только о законных врезках, данный алгоритм следует применять на каждой точке во время пролета вдоль трубы.

Python

```
yellow = ((20, 70, 80), (40, 255, 255))
def find_contours(img, color):
    c = []
    img_mask = cv2.inRange(img, color[0], color[1])
    contours, _ = cv2.findContours(img_mask, cv2.RETR_EXTERNAL,
    ↪  cv2.CHAIN_APPROX_SIMPLE)
    for i in contours:
        area = cv2.contourArea(i)
```

```
        if area < 100:
            continue
        else:
            c.append(i)
    return c
def get_counter(cnts):
    centers = []
    contours = cnts
    for cnt in contours:
        area = cv2.contourArea(cnt)
        if area > 100:
            moments = cv2.moments(cnt)
            try:
                x = int(moments['m10'] / moments['m00'])
                y = int(moments['m01'] / moments['m00'])
                centers.append((x, y))
            except ZeroDivisionError:
                return centers
    return centers
def vrez(img) -> bool:
    hsv = cv2.cvtColor(img, cv2.COLOR_BGR2HSV)
    cnts = find_contours(hsv, yellow)
    c = get_counter(cnts)
    cropped = hsv[(c[0][1] - 30) :(c[0][1] + 30) ,(c[0][0] - 30)
    ↪  :(c[0][0] + 30)]
    colors_mask = cv2.inRange(cropped, yellow[0], yellow[1])
    colors_mask = cv2.erode(colors_mask, np.array(
        [[0, 1, 1, 1, 0],
        [1, 1, 1, 1, 1],
        [1, 1, 1, 1, 1],
        [1, 1, 1, 1, 1],
        [0, 1, 1, 1, 0]],
        dtype=np.uint8)
    )
    cnts, _ = cv2.findContours(colors_mask, cv2.RETR_EXTERNAL,
    ↪  cv2.CHAIN_APPROX_SIMPLE)
    if not cnts:
        return False, None
    cnt = max(cnts, key=cv2.contourArea)
    (_, _), (w, h), _ = cv2.minAreaRect(cnt)
    rect_area = w * h
    cnt_area = cv2.contourArea(cnt)
    r = (1 - cnt_area / rect_area)
    if (r > 0.4):
        return True, c[0]
    return False, None
    data = get_pipe_data()
    start = data["nodes"][0]
    end = data["nodes"][1]
    print(f'going to start node {start}')
    navigate_wait(x=start["x"], y=start["y"], z=1.0,
    ↪  frame_id='aruco_map')
    f = open("pipe-audit.txt", "w")
    for i, tap in enumerate(data["tappings"]):
        print(f"going to tapping {tap}")
        navigate_wait(x=tap["x"], y=tap["y"], z=1.0,
        ↪  frame_id='aruco_map', tolerance=0.1)
        img, _ = get_image()
        illegal, p = vrez.vrez(img)
        if tap["legal"] == 0 and illegal:
```

```
64              print(f"Tapping {i+1}. x={tap['x']}, y={tap['y']}, legal=0
                ↪  illegal=1 (!)", file=f)
65              print("^ illegal!")
66              cv2.circle(img, (p[0], p[1]), 5, (0, 0, 255), -1)
67              cv2.imwrite(f"tapping_{i}", img)
68              tapping_pts.publish(String(f"[{tap['x']}, {tap['y']}, 1]"))
69          elif tap["legal"] == 1:
70              print(f"Tapping {i+1}. x={tap['x']}, y={tap['y']}, legal=1
                ↪  illegal=0", file=f)
71              tapping_pts.publish(String(f"[{tap['x']}, {tap['y']}, 0]"))
72      f.close()
```

1.4. Разработка веб-приложения

В рамках этой задачи существует два основных подхода к разработке веб-приложения:

1. Написание собственного бэкэнда

   В этом варианте между frontend (клиентской частью веб-приложения) и ROS-средой разрабатывается отдельный backend-сервер, например, на Python (FastAPI, Flask) или Node.js. Бэкэнд обеспечивает:

   - подписку на нужные ROS-топики, обработку, агрегацию и преобразование данных,
   - предоставление REST- или WebSocket API для клиентской части,
   - управление миссией и трансляцию команд управления от пользователя (например, запуск/остановка скриптов миссии, команды арминга/дизарминга дрона и т. д.).

   Такой подход позволяет реализовать дополнительные уровни защиты, логику доступов, хранение данных, кастомную обработку сообщений, обеспечивает лучшую независимость frontend'а от ROS-инфраструктуры и большую гибкость масштабирования.

2. Использование `roslib.js`.

   Альтернативный вариант — прямое взаимодействие с ROS из браузера с помощью сторонней библиотеки `roslib.js`. В этом случае ROS-ноды запускаются с `rosbridge`-сервером (`rosbridge_suite`), а браузерное приложение напрямую подписывается на ROS-топики, публикует команды и обрабатывает сообщения без отдельного backend-а. Это ускоряет разработку приложения, но делает разработанный сервис менее гибким.

   Остановимся подробнее на этом варианте как наиболее быстром в разработке, что является важным в условиях ограниченного времени на финале НТО.

Для отображения карты и найденных объектов с привязкой к карте наиболее простым вариантом является использование HTML-canvas. Для преобразования мировых координат в пиксельные на холсте используются предварительно заданные смещения и масштабы (sx, sy, dx, dy), благодаря которым любая точка в глобальной системе (x, y) проецируется на правильную позицию на картинке.

Постоянно, с интервалом 2 с, отправляется запрос на ROS-сервис `get_telemetry`, и полученные данные о напряжении аккумулятора конвертируются в проценты и вольты, после чего обновляют текстовые поля в интерфейсе. Похожим образом

подписчики на топики `/metal_pts`, `/tapping_pts` и `/max_tmp` отслеживают в реальном времени координаты дефектов теплосети, координаты незаконных врезок и выводятся и список и на карту.

Кнопки `Run`, `Killswitch`, `Stop`, `Home` и `Land` привязаны к соответствующим ROS-сервисам `set_running`, `set_killswitch`, `set_stopping`, `set_home` и `set_landing`. При нажатии каждая кнопка вызывает метод callService и по получении ответа выводит простое `JavaScript`-окно `alert`, уведомляющее об успешной отправке команды (`ok let's go...`, `disarming...` и т. д.). В это время запускается обработчик сервиса на дроне, который выполняет запрограммированные действия.

Пример регистрации сервиса и его обработчика в коде миссии см. ниже.

Python

```python
landing = False
def set_landing_cb(_):
    global landing
    landing = True
    return TriggerResponse(success=True)
set_landing = rospy.Service("set_landing", Trigger, set_landing_cb)
```

Управление крышкой дронпорта

На встроенном контроллере дронпорта (ESP-32) требуется реализовать HTTP-сервер, который принимает команды для управления исполнительными механизмами дронпорта — в первую очередь, для открытия и закрытия крышки. Такой сервер обеспечивает взаимодействие между системой управления дроном (миссионный компьютер, внешнее веб-приложение и т. д.) и аппаратной частью дронпорта через стандартные HTTP-запросы.

После выполнения команды ESP-32 возвращает HTTP-ответ с результатом — строку «OK» и код состояния 200, если операция успешно завершена.

Функция управления крышкой дронпорта отправляет HTTP-запросы к ESP-32. В зависимости от переданного параметра mode она формирует URL вида `http://.../open` или `http://.../close` и делает запрос. После этого программа анализирует полученный ответ, чтобы удостовериться, что команда выполнена успешно. Если ответа нет или код ответа не 200, или тело ответа отличается от ожидаемого («OK»), то команда считается невыполненной и повторяет попытку.

Python

```python
mode = "open"  # or " close"
while True:
    try:
        resp = requests.get(f"http://<esp_32_ip>/{mode}", timeout=3)
        print(resp.text)
        if resp.ok:
            print("OK!")
            break
        else:
            print(f"not ok? {resp.status_code}")
    except Exception as e:
        print(e)
    print("retrying in 1 sec...")
    rospy.sleep(1)
```

Конструкторская часть

Перед началом разработки устройства необходимо ознакомиться с техническим заданием и обратить внимание на имеющиеся ограничения:

- Временные рамки на проектирование, изготовление и доработку устройства.
- Электронную компонентную базу (список доступных компонентов и материалов представляется в первый день инженерного тура).
- Работу с 3D-принтером:
    - ◇ для минимизации использования поддержек необходимо выбрать правильное направление и ориентацию деталей, кроме того, можно использовать геометрические формы, требующие минимальное число поддержек;
    - ◇ во избежание деформации и провисания пластика во время печати необходимо обеспечить достаточную толщину стенок деталей и их правильное расположение;
    - ◇ для гарантированного соединения деталей между собой необходимо расположить детали на печатном столе так, чтобы поверхности и отверстия, с помощью которых детали соединяются, имели максимальную точность;
    - ◇ для минимизации времени печати и использования материала следует подобрать оптимальное расположение деталей на печатной платформе (закладывать минимальное расстояние между деталями; сторону детали, имеющую максимальный линейный размер, располагать горизонтально на поверхности стола и т. п.);
    - ◇ для обеспечения правильного прижима первого слоя путем последовательной проверки зазора между соплом и столом по нескольким противолежащим точкам в углах стола перед началом работы с принтером нужно произвести калибровку печатной платформы;
    - ◇ для настройки температуры перед запуском основной печати следует напечатать тестовую модель из необходимого пластика;
    - ◇ использовать подходящие настройки скорости печати для конкретных деталей; чтобы определить оптимальную скорость перемещений на холостом ходу для имеющегося принтера, необходимо распечатать тестовую модель при различных скоростях движения, начиная со 100 мм/с и изменяя ее с приростом 5 мм/с; скорость печати можно увеличивать, если качество поверхности приемлемое, и уменьшать, если качество 3D-печати ухудшается (необходимо обращать внимание на такие дефекты, как несовпадение слоев).
- На работу с лазерным станком с ЧПУ для резки фанеры:
    - ◇ при проектировании частей устройства и необходимо учитывать количество материала;
    - ◇ для минимизации использования материала необходимо подобрать оптимальное расположение деталей на каждом листе: детали размещаются на листе с зазором не менее 1 мм.

При сборке, тестировании и эксплуатации устройства необходимо соблюдать технику безопасности.

Результатом решения инженерной задачи является разработка устройства дронпорта, кронштейна крепления тепловизионной камеры и изготовление документации для данных изделий.

Крепление тепловизионной камеры

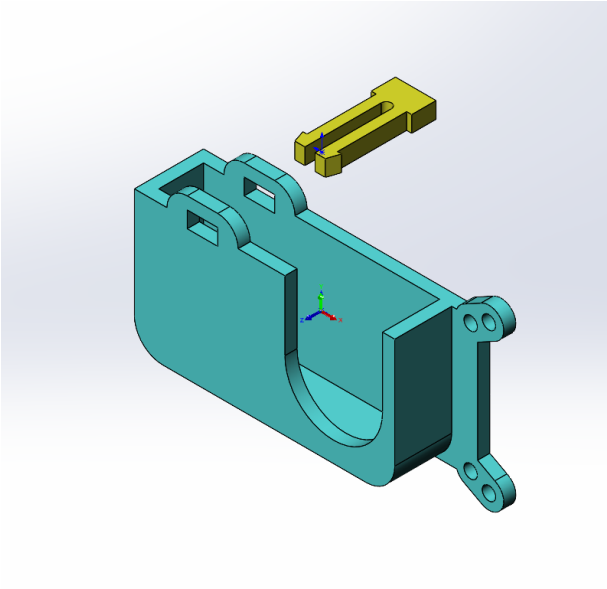


Рис. 4.3.5. 3D-модель кронштейна

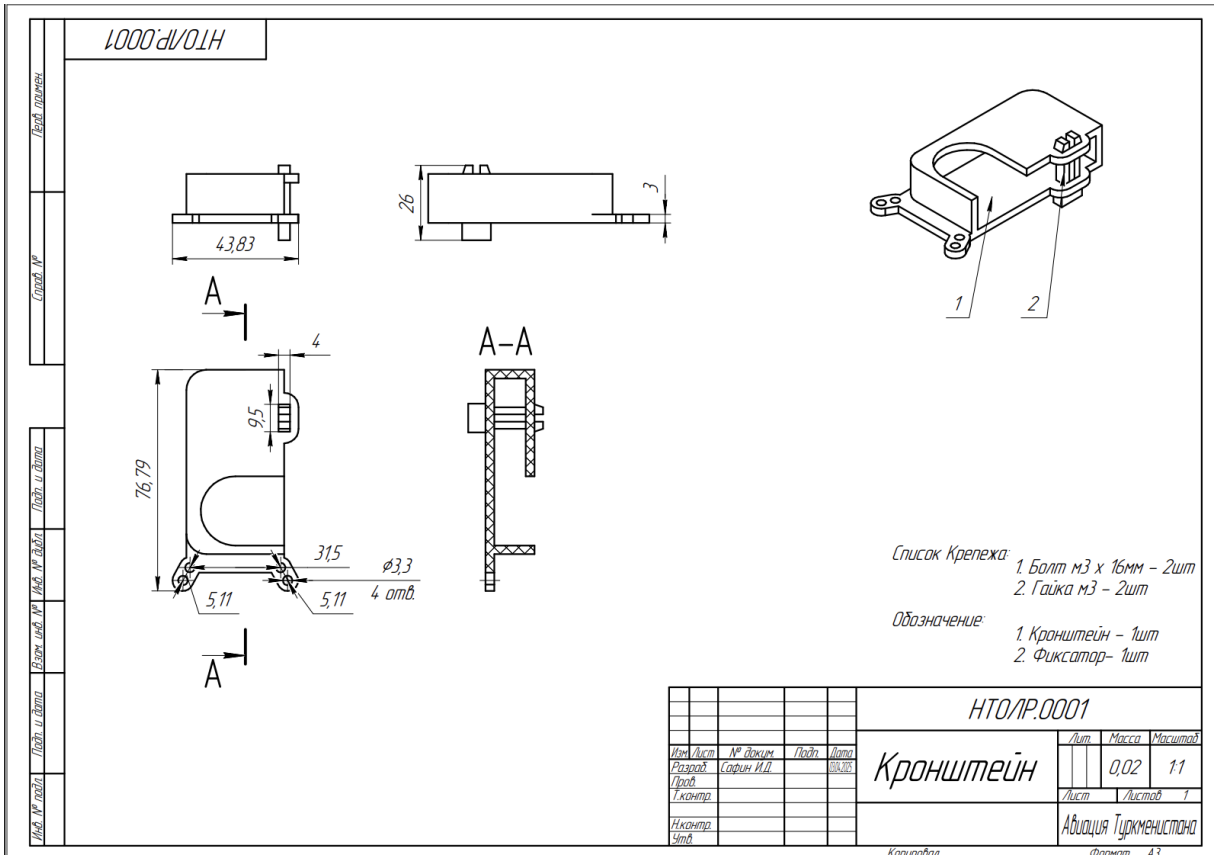


Рис. 4.3.6. Сборочный чертеж кронштейна

Дронопорт

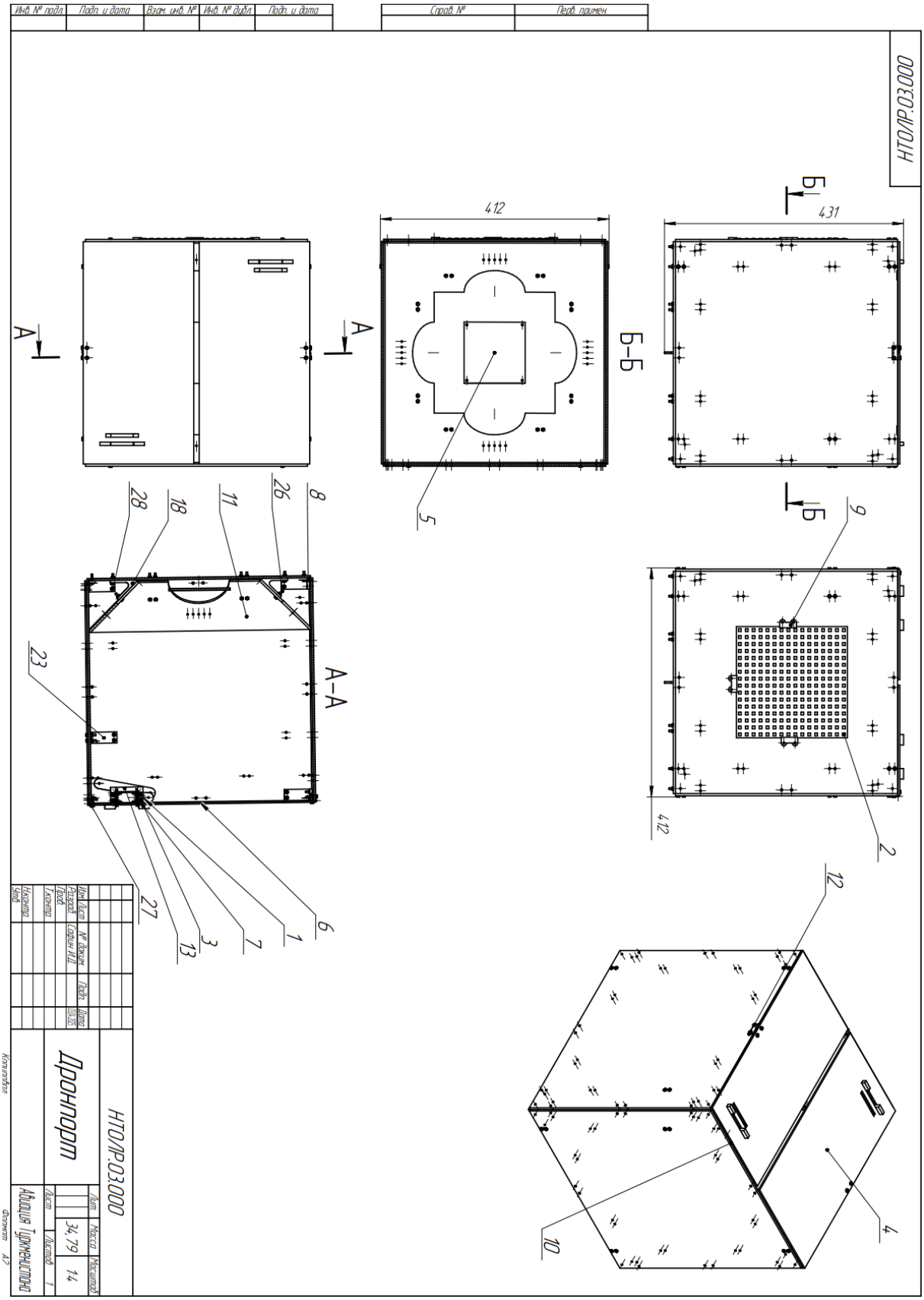
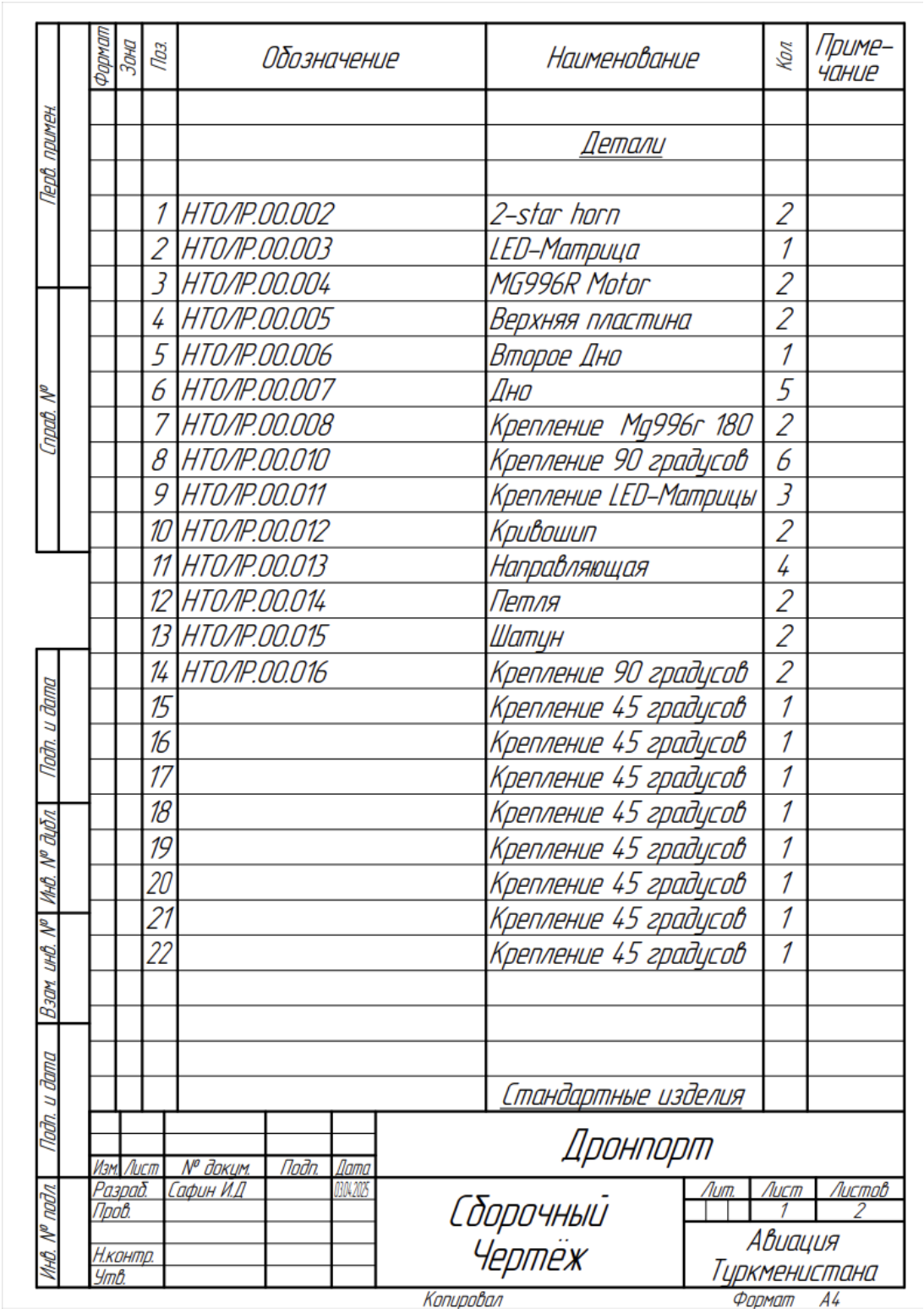


Рис. 4.3.7. Сборочный чертеж дронпорта



Формат	Зона	Поз.	Обозначение	Наименование	Кол.	Приме-чание
				Детали		
		1	НТО/ЛР.00.002	2-star horn	2	
		2	НТО/ЛР.00.003	LED-Матрица	1	
		3	НТО/ЛР.00.004	MG996R Motor	2	
		4	НТО/ЛР.00.005	Верхняя пластина	2	
		5	НТО/ЛР.00.006	Второе Дно	1	
		6	НТО/ЛР.00.007	Дно	5	
		7	НТО/ЛР.00.008	Крепление Mg996r 180	2	
		8	НТО/ЛР.00.010	Крепление 90 градусов	6	
		9	НТО/ЛР.00.011	Крепление LED-Матрицы	3	
		10	НТО/ЛР.00.012	Кривошип	2	
		11	НТО/ЛР.00.013	Направляющая	4	
		12	НТО/ЛР.00.014	Петля	2	
		13	НТО/ЛР.00.015	Шатун	2	
		14	НТО/ЛР.00.016	Крепление 90 градусов	2	
		15		Крепление 45 градусов	1	
		16		Крепление 45 градусов	1	
		17		Крепление 45 градусов	1	
		18		Крепление 45 градусов	1	
		19		Крепление 45 градусов	1	
		20		Крепление 45 градусов	1	
		21		Крепление 45 градусов	1	
		22		Крепление 45 градусов	1	
				Стандартные изделия		

Перв. примен. | Справ. № | Подп. и дата | Инв. № дубл. | Взам. инв. № | Подп. и дата | Инв. № подл.

Дронпорт

Изм.	Лист	№ докум.	Подп.	Дата
Разраб.		Сафин И.Д.		13.04.2015
Пров.				
Н.контр.				
Утв.				

Сборочный Чертёж

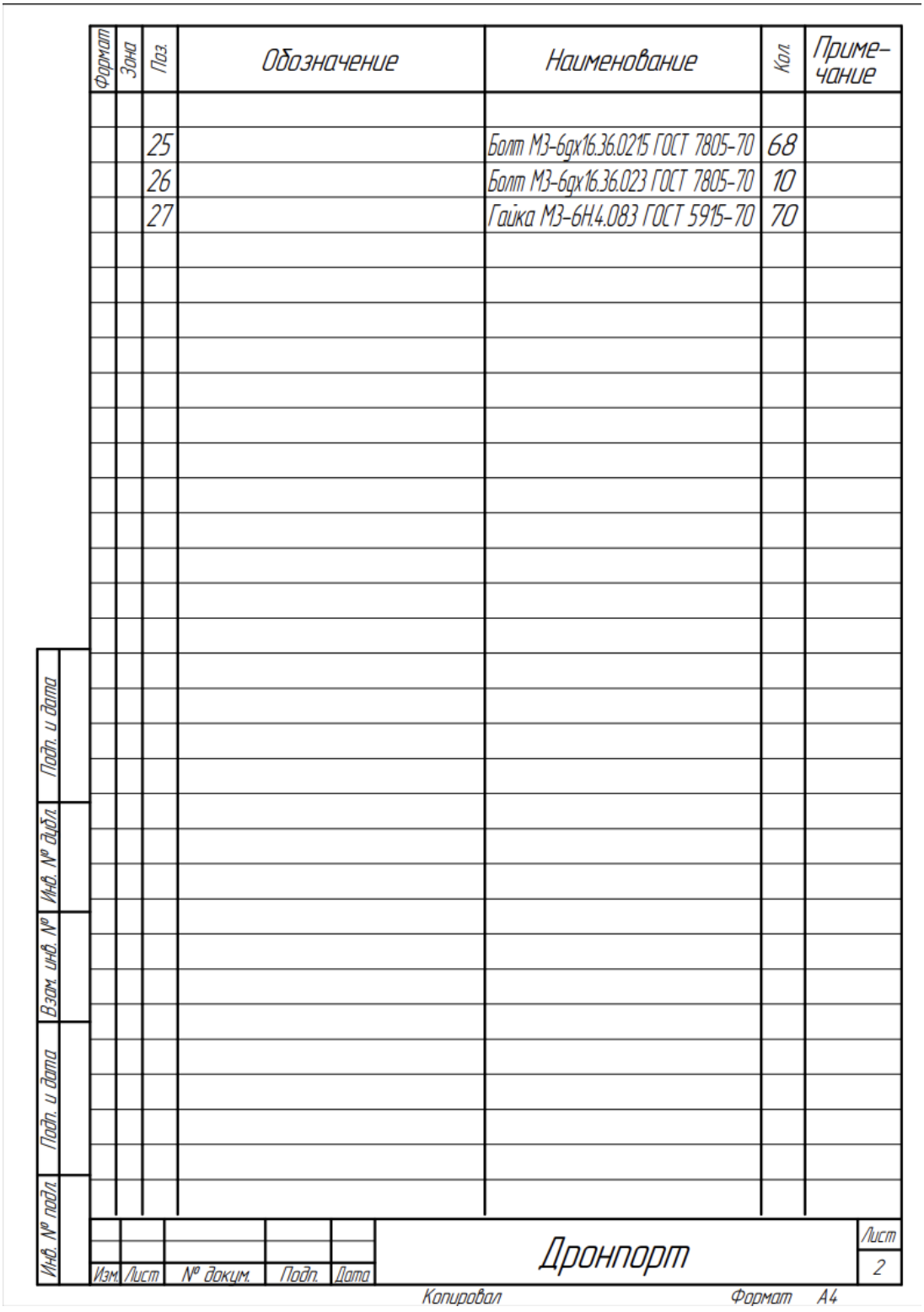
Лит.	Лист	Листов
	1	2

Авиация Туркменистана

Копировал

Формат А4

Рис. 4.3.8. Спецификация, лист 1



Формат	Зона	Поз.	Обозначение	Наименование	Кол.	Приме-чание
		25		Болт М3-6gх16.36.0215 ГОСТ 7805-70	68	
		26		Болт М3-6gх16.36.023 ГОСТ 7805-70	10	
		27		Гайка М3-6Н.4.083 ГОСТ 5915-70	70	

Подп. и дата | Инв. № дубл. | Взам. инв. № | Подп. и дата | Инв. № подл.

Изм. | Лист | № докум. | Подп. | Дата

Дронпорт

Лист 2

Копировал

Формат А4

Рис. 4.3.9. Спецификация, лист 2

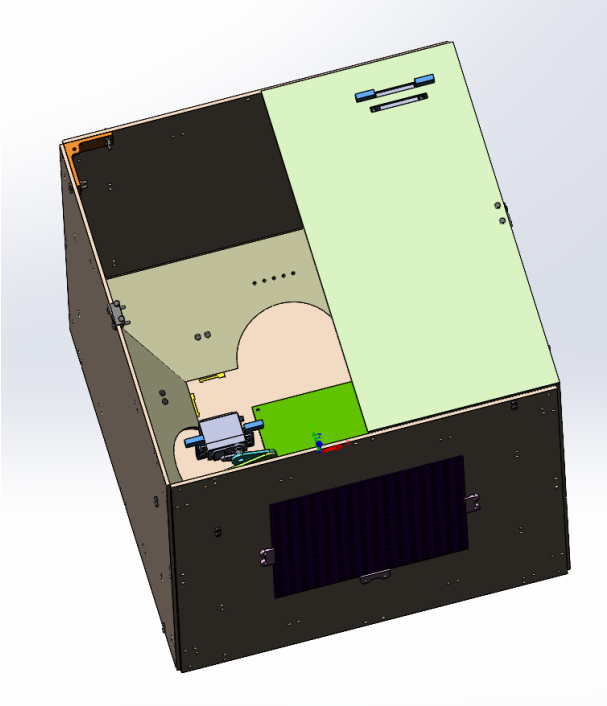


Рис. 4.3.10. 3D-модель дронпорта

Инструкция по сборке дронпорта: https://disk.yandex.ru/i/BSmEGFrCXF1j5g.

Инструкция по эксплуатации дронпорта: https://disk.yandex.ru/i/K1KBqU5uWWE7wQ.

Код микроконтроллера

C++

```cpp
#include <WiFi.h>
#include <WebServer.h>
#include <ESP32Servo.h>
#include <FastLED.h>
// Конфигурация
const char *ssid = "turk";
const char *password = "torvalds";
#define SERVO_1_PIN 13
#define SERVO_2_PIN 12
#define BUTTON_PIN 27
#define LED_PIN 14
#define MATRIX_SIZE 16
#define NUM_LEDS 256
#define CLOSED_TIMEOUT 5000 // 5 секунд
#define one_frame 1000
int state = 0;
int anim = 0;
int frame = 0;
Servo servo1;
Servo servo2;
CRGB leds[NUM_LEDS];
WebServer server(80);
volatile bool isOpen = false;
bool buttonPressed = false;
unsigned long lidClosedTime = 0;
```

```
26  bool yellowAlert = false;
27  void anim0()
28  {
29      FastLED.clear();
30      leds[112] = CRGB::Yellow;
31      <...>
32      leds[224] = CRGB::Yellow;
33      leds[173] = CRGB::Blue;
34      <...>
35      leds[228] = CRGB::Blue;
36      FastLED.show();
37  }
38  void anim1()
39  {
40      FastLED.clear();
41      leds[84] = CRGB::Yellow;
42      // <...>
43      leds[224] = CRGB::Yellow;
44      leds[173] = CRGB::Blue;
45      // <...>
46      leds[228] = CRGB::Blue;
47      FastLED.show();
48      delay(one_frame);
49      FastLED.clear();
50      leds[11] = CRGB::Yellow;
51      // <...>
52      leds[224] = CRGB::Yellow;
53      leds[173] = CRGB::Blue;
54      // <...>
55      leds[228] = CRGB::Blue;
56      FastLED.show();
57      delay(one_frame);
58      FastLED.clear();
59      leds[241] = CRGB::Yellow;
60      // <...>
61      leds[255] = CRGB::Yellow;
62      leds[173] = CRGB::Blue;
63      // <...>
64      leds[228] = CRGB::Blue;
65      FastLED.show();
66  }
67  // Другие анимации <...>
68  void setup()
69  {
70      Serial.begin(115200);
71      // Инициализация сервоприводов
72      servo1.attach(SERVO_1_PIN);
73      servo2.attach(SERVO_2_PIN);
74      servo1.write(0);
75      servo2.write(0);
76      pinMode(BUTTON_PIN, INPUT_PULLUP);
77      FastLED.addLeds<WS2812B, LED_PIN, GRB>(leds,
78                                             NUM_LEDS);
79      FastLED.setBrightness(30);
80      updateLED();
81      WiFi.begin(ssid, password);
82      while (WiFi.status() != WL_CONNECTED)
83          delay(500);
84      server.on("/", []()
```

```
85              { server.send(200, "text/plain", "Status: " +
                ↪  String(isOpen ? "Open" : "Closed")); });
86      server.on("/open", handleOpen);
87      server.on("/close", handleClose);
88      server.begin();
89      anim0();
90  }
91  void handleOpen()
92  {
93      if (!isOpen)
94      {
95          servo1.write(180);
96          servo2.write(180);
97          isOpen = true;
98          yellowAlert = false;
99          lidClosedTime = 0;
100         Serial.println("Opened via web");
101     }
102     updateLED();
103     server.send(200, "text/plain", "OK");
104 }
105 void handleClose()
106 {
107     if (isOpen)
108     {
109         servo1.write(0);
110         servo2.write(0);
111         isOpen = false;
112         lidClosedTime = millis();
113         Serial.println("Closed via web");
114     }
115     updateLED();
116     server.send(200, "text/plain", "OK");
117 }
118 void updateLED()
119 {
120     // Исправлено: заменено & на &&
121     bool yellowCondition = !isOpen &&
122                            (millis() - lidClosedTime) >=
123                                CLOSED_TIMEOUT &&
124                            buttonPressed;
125     if (yellowCondition)
126     {
127         animc();
128     }
129     else if (!yellowCondition)
130     {
131         int new_state = state; // Объявлено перед
132         использованием
133         // Исправлено: заменено buttonColor на
134         buttonPressed и &на &&if (buttonPressed && !isOpen)
135         {
136             new_state = 0;
137         }
138         else if (buttonPressed && isOpen)
139         {
140             new_state = 1;
141         }
142         else if (!buttonPressed && isOpen)
143         {
```

```
144             new_state = 2;
145         }
146         else if (!buttonPressed && !isOpen)
147         {
148             new_state = 3;
149         }
150         // Исправлено: заменено & на &&
151         if (state != new_state)
152         {
153             if ((state == 0 | state == -1) && new_state ==
154                                                     1)
155             {
156                 anim1();
157             }
158             else if (state == 1 && new_state == 2)
159             {
160                 anim2();
161             }
162             else if (state == 2 && new_state == 3)
163             {
164                 anim3();
165             }
166             else if (state == 3 && new_state == 2)
167             {
168                 anim4();
169             }
170             else if (state == 2 && new_state == 1)
171             {
172                 anim5();
173             }
174             else if (state == 1 && new_state == 0)
175             {
176                 anim6();
177             }
178             state = new_state;
179         }
180     }
181     FastLED.show();
182     // Исправлен вывод в Serial
183     Serial.print("Anim: ");
184     Serial.print(anim);
185     Serial.print(" State: ");
186     Serial.println(state);
187 }
188 void loop()
189 {
190     server.handleClient();
191     // Обработка кнопки
192     static bool lastButtonState = HIGH;
193     static uint32_t lastDebounceTime = 0;
194     bool reading = digitalRead(BUTTON_PIN);
195     if (reading != lastButtonState)
196     {
197         lastDebounceTime = millis();
198     }
199     if ((millis() - lastDebounceTime) > 50)
200     {
201         if (buttonPressed != (reading == LOW))
202         {
203             buttonPressed = (reading == LOW);
```

```
204             lidClosedTime = millis();
205             updateLED();
206             Serial.print("Button state: ");
207             Serial.println(buttonPressed ? "PRESSED" : "RELEASED");
208         }
209     }
210     lastButtonState = reading;
211     static unsigned long lastUpdate = 0;
212     if (millis() - lastUpdate > 100)
213     {
214         updateLED();
215         lastUpdate = millis();
216     }
217 }
```

Рис. 4.3.11. Фотография собранного и рабочего дронпорта

Рис. 4.3.12. Фотография собранного и рабочего дронпорта

4.3.8. Материалы для подготовки

1. Статьи, видеолекции по тематике профиля на сайте документации платформы «Клевер»: https://clover.coex.tech/.
2. Видеокурс по сборке, настройке и программированию квадрокоптера: https://cat.2035.university/rall/course/27662/.
3. Среда симуляции платформы «Клевер»: https://clover.coex.tech/ru/simulation_vm.html.
4. Работа с OpenCV при помощи Python: https://github.com/sfalexrog/coex_kb/blob/master/kb014_opencv_python.md.
5. Программирование на Python: https://stepik.org/course/67/promo.
6. Введение в ROS: https://stepik.org/course/3222/promo.
7. Введение в Git: https://ru.hexlet.io/courses/intro_to_git.

5. Критерии определения победителей и призеров

Первый отборочный этап

В первом отборочном этапе участники решали задачи предметного тура по двум предметам: физике и информатике и инженерного тура. В каждом предмете максимально можно было набрать 100 баллов, в инженерном туре 100 баллов. Для того чтобы пройти во второй этап, участники должны были набрать в сумме по обоим предметам и инженерному туру не менее 60,0 баллов, независимо от уровня.

Второй отборочный этап

Количество баллов, набранных при решении всех задач второго отборочного этапа, суммируется. Победители второго отборочного этапа должны были набрать не менее 61,5 балла, независимо от уровня.

Заключительный этап

Индивидуальный предметный тур

- физика — максимально возможный балл за все задачи — 100 баллов;
- информатика — максимально возможный балл за все задачи — 100 баллов.

Командный инженерный тур

Команды заключительного этапа получали за командный инженерный тур от 0 до 100,00 баллов: команда, набравшая наибольшее число баллов среди других команд, становилась командой-победителем.

Все результаты команд нормировались по формуле:

$$\frac{100 \times x}{MAX},$$

где x — число баллов, набранных командой,

MAX — число баллов, максимально возможное за инженерный тур.

В заключительном этапе олимпиады индивидуальные баллы участника складываются из двух частей, каждая из которых имеет собственный вес: баллы за индивидуальное решение задач по предмету 1 (физика) с весом $K_1 = 0{,}2$, по предмету 2

(информатика) с весом $K_2 = 0{,}2$, баллы за командное решение задач инженерного тура с весом $K_3 = 0{,}6$.

Итоговый балл определяется по формуле:

$$S = K_1 \cdot S_1 + K_2 \cdot S_2 + K_3 \cdot S_3,$$

где S_1 — балл первой части заключительного этапа по физике (предметный тур) ($S_{1\ \text{макс}} = 100$);

S_2 — балл первой части заключительного этапа по информатике (предметный тур) ($S_{2\ \text{макс}} = 100$);

S_3 — итоговый балл инженерного командного тура ($S_{3\ \text{макс}} = 100$).

Итого максимально возможный индивидуальный балл участника заключительного этапа — 100 баллов.

Критерий определения победителей и призеров

Чтобы определить победителей и призеров (независимо от класса) на основе индивидуальных результатов участников, был сформирован общий рейтинг всех участников заключительного этапа. С начала рейтинга были выбраны 2 победителя и 5 призеров (первые 25% участников рейтинга становятся победителями или призерами, из них первые 8% становятся победителями, оставшиеся — призерами).

Критерий определения победителей и призеров (независимо от уровня)

Категория	**Количество баллов**
Победители	46,90 и выше
Призеры	От 31,94 до 44,10

6. Работа наставника после НТО

Участие школьника в Олимпиаде может завершиться после любого из этапов: первого или второго отборочных, либо после заключительного этапа. В каждом случае после завершения участия наставнику необходимо провести с учениками рефлексию — обсудить полученный опыт и проанализировать, что позволило достичь успеха, а что привело к неудаче. Подробные материалы о проведении рефлексии представлены в курсе «Наставник НТО»: https://academy.sk.ru/events/310.

Наставнику важно проинформировать руководство образовательного учреждения, если его учащиеся стали финалистами, призерами и победителями. Публичное признание высоких результатов дополнительно повышает мотивацию.

В процессе рефлексии с учениками, не ставшими призерами или победителями, рекомендуется уделить особое внимание особенностям командной работы: распределению ролей, планированию работы, возникающим проблемам. Для этого могут использоваться опросники для самооценки собственной работы и взаимной оценки участниками других членов команды (P2P). Они могут выявить внутренние проблемы команды, для решения которых в план подготовки можно добавить мероприятия, направленные на ее сплочение.

Стоит рассказать, что в истории НТО было много примеров, когда не победив в первый раз, на следующий год участники показывали впечатляющие результаты, одержав победу сразу в нескольких профилях. Конечно, важно отметить, что так происходит только при учете прошлых ошибок и подготовке к Олимпиаде в течение года.

Важным фактором успешного участия в следующих сезонах НТО может стать поддержка родителей учеников. Знакомство с ними помогает наставнику продемонстрировать важность компетенций, развиваемых в процессе участия в НТО, для будущего образования и карьеры школьников. Поддержка родителей помогает мотивировать участников и позволяет выделить необходимое время на занятия в кружке.

С участниками-выпускниками наставнику рекомендуется обсудить их дальнейшее профессиональное развитие и его связь с выбранными профилями НТО. Отдельно можно обратить внимание на льготы для победителей и призеров, предлагаемые в вузах с интересующими ученика направлениями. Кроме того, ряд вузов предлагает льготы для всех финалистов НТО, а также учитывает результаты Конкурса цифровых портфолио «Талант НТО».