



НТО

МАТЕРИАЛЫ ЗАДАНИЙ
Всероссийской междисциплинарной олимпиады
школьников 8–11 класса
«Национальная технологическая олимпиада»
по профилю
«Беспилотные авиационные системы»

2024/25 учебный год

ntcontest.ru

УДК 373.5.016:629.73.02

ББК 74.263.0

Б23

Авторы:

Д. С. Васильев, О. В. Зубков, К. Д. Кириченко, Н. Ю. Кузнецов, К. С. Лельков,
Д. А. Сурков, Д. В. Ульянов, А. Н. Ушаков, Т. С. Хорев

Б23 Всероссийская междисциплинарная олимпиада школьников 8–11 класса
«Национальная технологическая олимпиада». Учебно-методическое пособие
Том 5 **Беспилотные авиационные системы**
— М.: Ассоциация участников технологических кружков, 2025. — 204 с.

ISBN 978-5-908021-04-3

Данное пособие разработано коллективом авторов на основе опыта проведения все-российской 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

11.00.00 Электроника, радиотехника и системы связи

12.00.00 Фотоника, приборостроение, оптические и биотехнические системы и технологии

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 978-5-908021-04-3

УДК 373.5.016:629.73.02

ББК 74.263.0



Оглавление

1 Введение	5
1.1 Национальная технологическая олимпиада	5
1.2 Беспилотные авиационные системы	13
2 Первый отборочный этап	15
2.1 Работа наставника НТО на этапе	15
2.2 Предметный тур. Информатика	16
2.2.1 Первая волна. Задачи 8–11 класса	16
2.2.2 Вторая волна. Задачи 8–11 класса	26
2.2.3 Третья волна. Задачи 8–11 класса	36
2.2.4 Четвертая волна. Задачи 8–11 класса	49
2.3 Предметный тур. Физика	64
2.3.1 Первая волна. Задачи 8–9 класса	64
2.3.2 Первая волна. Задачи 10–11 класса	68
2.3.3 Вторая волна. Задачи 8–9 класса	73
2.3.4 Вторая волна. Задачи 10–11 класса	80
2.3.5 Третья волна. Задачи 8–9 класса	85
2.3.6 Третья волна. Задачи 10–11 класса	90
2.3.7 Четвертая волна. Задачи 8–9 класса	94
2.3.8 Четвертая волна. Задачи 10–11 класса	100
2.4 Инженерный тур	105
3 Второй отборочный этап	117
3.1 Работа наставника НТО на этапе	117
3.2 Инженерный тур	119
3.2.1 Командные задачи	119

4	Заключительный этап	135
4.1	Работа наставника НТО при подготовке к этапу	135
4.2	Предметный тур	137
4.2.1	Информатика. 8–11 классы	137
4.2.2	Физика. 8–9 классы	152
4.2.3	Физика. 10–11 классы	159
4.3	Инженерный тур	169
4.3.1	Общая информация	169
4.3.2	Легенда задачи	169
4.3.3	Требования к команде и компетенциям участников	169
4.3.4	Оборудование и программное обеспечение	170
4.3.5	Задачи	170
4.3.6	Материалы для подготовки	200
5	Критерии определения победителей и призеров	202
6	Работа наставника после НТО	204

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 Кружко-

вое 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. Беспилотные авиационные системы

Профиль Беспилотные авиационные системы нацелен на вовлечение школьников 8–11 классов в техническую и инновационную деятельность в области проектирования систем автоматического управления беспилотными летательными аппаратами (БЛА). Особое внимание уделяется организации процесса разработки автопилота, при котором возможна практическая проверка разработанных программ на симуляторе полета БВС самолетного типа с вертикальным взлетом и посадкой (СВВП) UAVIANT.

Профиль направлен на решение следующих технологических барьеров НТИ:

- сенсоры и преобразующая аппаратура оптического, теплового, гиперспектрального, радиолокационного зондирования поверхности, радиолокационные станции бортового обзора, в том числе с функцией распознавания образов людей, животных, транспортных средств и потоков, мобильных и стационарных объектов для обеспечения мониторинга, подсчета наблюдаемых объектов и выявления их характерных признаков, а также для выявления признаков чрезвычайных ситуаций;
- беспроводная система мониторинга систем и узлов БВС СВВП;
- электронно-оптическая система «улучшенного видения» для повышения качества изображения в условиях тумана, плохой видимости, при наличии препятствий;
- БВС для сбора, хранения и обработки информации о характеристиках окружающего пространства.

Основной задачей заключительного этапа является разработка системы автоматического управления БВС СВВП для поиска объектов в автоматическом режиме с целью мониторинга зоны возникновения чрезвычайной ситуации (наводнения) и доставки медицинского груза адресатам в зоне с затрудненным доступом наземных транспортных средств.

Акцент при разработке командной комплексной инженерной задачи направлен на четкое разделение основной задачи на подзадачи, при успешном решении которых участники достигают поставленной цели. Каждая подзадача на заключительном этапе имеет свой уровень сложности и оценивается соответствующим баллом. Подзадачи распределены по соответствующим этапам.

Первый отборочный этап включает в себя решение олимпиадных задач по двум школьным предметам: информатика и физика. Хорошие знания по этим дисциплинам крайне необходимы участникам профиля, так как требуются для решения задач второго отборочного и заключительного этапов.

Кроме того, на первом этапе проводится инженерный тур. Его основная задача — погружение участников в тематику профиля. Здесь предлагаются задачи по математике, физике, а также по ориентации и навигации БВС, а кроме того, тесты, посвященные основным понятиям в области беспилотных авиационных систем.

На **втором отборочном этапе** участники решают задачи по определению траектории полета, определению параметров полета БВС, а также по обработке изображений. Они носят междисциплинарный характер и в упрощенной форме воссоздают

инженерную задачу заключительного этапа.

На **заключительном этапе** все полученные в предыдущих этапах знания участники используют при работе с БВС самолетного типа с вертикальным взлетом и посадкой (настройка и отладка системы автоматического управления БВС, распознавание зоны доставки медицинского груза при помощи системы технического зрения, моделирование полета, сборка макета медицинского груза и проверка своих решений при помощи специально разработанного симулятора и в рамках летных испытаний на аэродроме). Такой подход дает возможность глубоко понять объект исследования и принцип его работы.

Для популяризации НТО и распространении информации о профиле Беспилотные авиационные системы его представители предпринимают поездки в общеобразовательные учебные заведения, вовлекая школьников в олимпиадное движение.

Участники профиля Беспилотные авиационные системы поступают в ведущие вузы России (в том числе в МАИ) на специальности, связанные с авиастроением, информационными технологиями, управлением в технических системах и другими. Причастность школьников к Олимпиаде по профилю Беспилотные авиационные системы не только повышает популярность авиационной отрасли среди них, но и способствует пополнению профильных инженерных вузов талантливыми и перспективными студентами.

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 $0 \leq t \leq 10^6$.

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++

```
1 #include<bits/stdc++.h>
2 #define int long long
3 using namespace std;
4 signed main(){
5     int t;
6     cin >> t;
7     cout << ((t * (t - 1)) * (t - 2)) << endl;
8 }
```

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 $1 \leq a, b, c, d \leq 10^9$.

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++

```

1  #include<bits/stdc++.h>
2  #define int long long
3  using namespace std;
4  signed main(){
5      double a, b, c, d;
6      cin >> a >> b >> c >> d;
7      double a0 = min({a, b, c, d});
8      double a1 = min(max(a, b) / 2.0, min(a, b));
9      double a2 = min(max(c, d) / 2.0, min(c, d));
10     double ans = max({a0, a1, a2});
11     if( (int)ans == ans ){
12         int ians = ans;
13         cout << ians << endl;
14         return 0;
15     }
16     cout.precision(1);
17     cout << fixed<< ans << endl;
18 }
```

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 $1 \leq n \leq 10^5$.

Во второй строке заданы $2n$ целых чисел d_i — длины всех кусков, которые получились после «тренировки» внука, где $1 \leq d_i \leq 10^9$. 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++

```

1  #include<bits/stdc++.h>
2  #define int long long
3  using namespace std;
4  signed main(){
5      int n;
6      cin >> n;
7      vector<pair<int, int> > v(2 * n);
8      for(int i = 0; i < 2 * n; i++){
9          int d;
10         cin >> d;
11         v[i] = {d, i + 1};
12     }
13     sort(v.begin(), v.end());
14     vector<pair<int, int> > ans(n);
15     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 $2 \leq a \leq 10^{18}$, $1 \leq b, c \leq 10^{18}$.

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++
1  #include<bits/stdc++.h>
2  #define int long long
3  using namespace std;
4  int f1(int M, int a){
5      int res = 0, z = 0;
6      while(1){
7          if(M == 0 && z < a){
8              return res;
9          }
10         res += M;
11         M = M + z;
12         z = M % a;
13         M = M / a;
14     }
15 }

```

```

16 int f2(int M, int b){
17     return M + M / b;
18 }
19 signed main(){
20     int a, b, c;
21     cin >> a >> b >> c;
22     int L = 0, R = 1;
23     while(f1(R, a) <= c){
24         R *= 2;
25     }
26     while(R - L > 1){
27         int M = (R + L) / 2;
28         if(f1(M, a) < c){
29             L = M;
30         }
31         else{
32             R = M;
33         }
34     }
35     int z = R;
36     L = 0, R = 1;
37     while(f2(R, b) <= z){
38         R *= 2;
39     }
40     while(R - L > 1){
41         int M = (R + L) / 2;
42         if(f2(M, b) < z){
43             L = M;
44         }
45         else{
46             R = M;
47         }
48     }
49     cout << R << endl;
50 }

```

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 $2 \leq n, m \leq 2 \cdot 10^5$.

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. $1 \leq A, B \leq n$, $1 \leq d \leq 10^9$. 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 , $1 \leq S \leq n$.

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++

```

1  #include<bits/stdc++.h>
2  #define int long long
3  using namespace std;
4  int n, m;
5  vector<vector<pair<int, int> > > G;
6  vector<int> order, used;
7  void dfs(int a){
8      used[a] = 1;
9      for(auto to : G[a]){
10         if(!used[to.first]){
11             dfs(to.first);
12         }
13     }
14     order.push_back(a);
15 }
16 signed main(){
17     cin >> n >> m;
18     G.resize(n + 1);
19     used.resize(n + 1, 0);
20     for(int i = 0; i < m; i++){
21         int a, b, d;
22         cin >> a >> b >> d;
23         G[a].push_back({b, d});
24     }

```

```

25     int s;
26     cin >> s;
27     dfs(s);
28     reverse(order.begin(), order.end());
29     vector<int> dp(n + 1, -1);
30     dp[s] = 0;
31     for(auto el : order){
32         for(auto to : G[el]){
33             dp[to.first] = max(dp[to.first], dp[el] + to.second);
34         }
35     }
36     for(int i = 1; i <= n; i++){
37         cout << dp[i] << ' ';
38     }
39 }

```

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 $n \times m$.

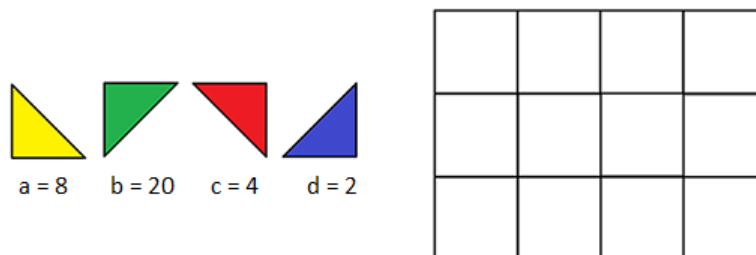


Рис. 2.2.1

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.

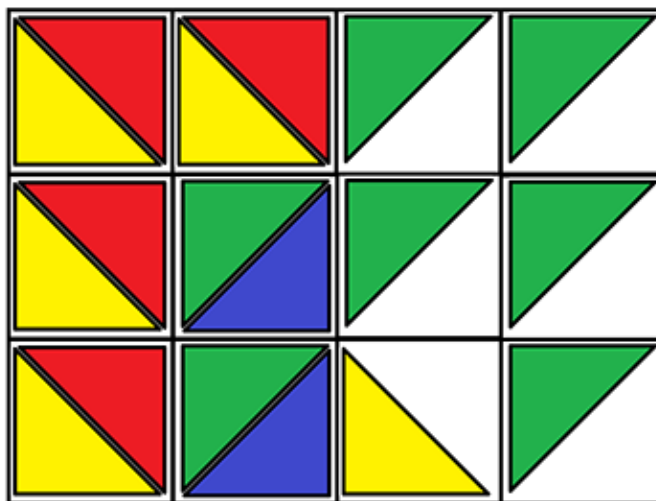


Рис. 2.2.2

Решение

Ниже представлено решение на языке C++.

C++

```

1  #include<bits/stdc++.h>
2  #define int long long
3  using namespace std;
4  signed main(){
5      int a, b, c, d, n, m;
6      cin >> a >> b >> c >> d >> n >> m;
7      if(a > c){
8          swap(a, c);
9      }
10     if(b > d){
11         swap(b, d);
12     }
13     int f = a + b;
14     int k = n * m;
15     if(k <= f){
16         cout << k * 2;
17         return 0;
18     }
19     k -= f;
20     c -= a;
21     d -= b;
22     cout << f * 2 + min(k, c + d) << endl;
23 }
```

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 $1 \leq n \cdot k \leq 10^{18}$.

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++
1  #include<bits/stdc++.h>
2  #define int long long
3  using namespace std;
4  signed main(){
5      int n, k;
6      cin >> n >> k;
7      int x = (k * n) / (k * k + 2 * k + 1);
8      cout << x - k << ' ' << x + k << ' ' << x / k << ' ' << x * k << endl;
9  }

```

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, $2 \leq n, m \leq 25$. Во второй строке через пробел находятся координаты (x, y) обязательной для посещения клетки, где $1 \leq x \leq n, 1 \leq y \leq m$. 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.

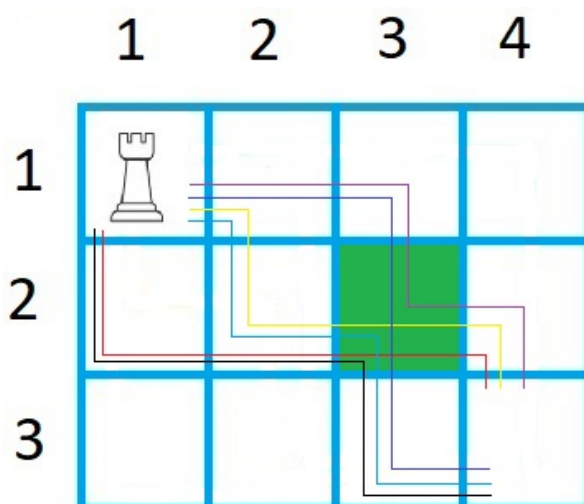


Рис. 2.2.3

Решение

Ниже представлено решение на языке C++.

C++

```

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

```

```

9         bc[i][j] += bc[i - 1][j];
10        if(j - 1 >= 0){
11            bc[i][j] += bc[i - 1][j - 1];
12        }
13    }
14 }
15 int n, m, x, y;
16 cin >> n >> m >> x >> y;
17 int d1 = bc[x - 1 + y - 1][x - 1];
18 int d2 = bc[n - x + m - y][n - x];
19 int ans = d1 * d2;
20 cout << ans << endl;
21 }

```

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 появиться 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++

```

1  #include<bits/stdc++.h>
2  #define int long long
3  using namespace std;
4  signed main(){
5      string s;
6      cin >> s;
7      string mx = s, mn = s;
8
9      for(int i = 0; i < 5; i++){
10         for(int j = 0; j < 10; j++){
11             mx = max(mx, s);
12             mn = min(s, mn);
13             if(j < 9){
14                 s = s.substr(1) + s[0];
15             }
16         }
17         for(int j = 0; j < 5; j++){
18             swap(s[2 * j], s[2 * j + 1]);
19         }
20     }
21     stringstream ssmn;
22     ssmn << mn;
23     int imn;
24     ssmn >> imn;
25     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 $1, \dots, n$.

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 $1 \leq n \leq 2 \cdot 10^5$.

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++

```

1  #include<bits/stdc++.h>
2  #define int long long
3  using namespace std;
4  const int LG = 19;
5  int N = (1 << LG);
6  vector<int> tr(2 * N, 0);
7  void upd(int pos, int x){
8      pos += N;
9      tr[pos] = x;
10     pos /= 2;
11     while(pos){
12         tr[pos] = {tr[2 * pos]+ tr[2 * pos + 1]};
13         pos /= 2;
14     }
15 }
16 int get(int l, int r){
17     l += N;
18     r += N;
19     int res = 0;
20     while(l <= r){
21         if(l % 2 == 1){
22             res += tr[l];
23         }
24         if(r % 2 == 0){
25             res += tr[r];
26         }
27         l = (l + 1) / 2;
28         r = (r - 1) / 2;
29     }
30     return res;
31 }
32 signed main(){
33     int n, a;
34     cin >> n;
35     int ans = 0;
36     for(int i = 0; i < n; i++){
37         cin >> a;
38         int sl = get(0, a - 1);
39         int sr = get(a + 1, N - 1);
40         ans += min(sl, sr);
41         upd(a, a);
42     }
43     cout << ans << endl;
44 }
```

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++

```

1 #include<bits/stdc++.h>
2 #define int long long
3 using namespace std;
4 signed main(){
5     int A, C, D;
6     cin >> A >> C >> D;
7     cout.precision(1);
8     cout << fixed << (A + C) / 2.0 << endl;
9 }
```

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

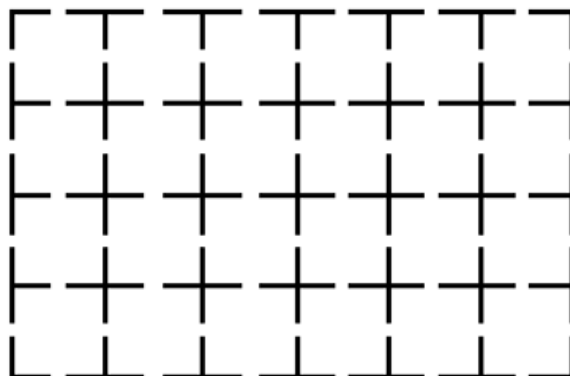


Рис. 2.2.4

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 $1 \leq n, m \leq 10^9$.

Формат выходных данных

Требуется по числам a и b найти размеры исходной решетки n и m . Будем всегда считать, что $n \leq m$, 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 бинарного поиска.

C++

```

1  #include<bits/stdc++.h>
2  #define int long long
3  using namespace std;
4  signed main(){
5      int a, b;
6      cin >> a >> b;
7      int L = 0, R = a / 4 + 1;
8      while(R - L > 1){
9          int M = (R + L) / 2;
10         int D = a / 2 - M;
11         if(M * D <= b){
12             L = M;
13         }
14         else{
15             R = M;
16         }
17     }
18     cout << L + 1 << ' ' << a / 2 - L + 1 << endl;
19 }
```

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 задачу.

«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 $2 \leq n, m \leq 1000$.

Во второй строке находятся два числа x и y через пробел — координаты свечки, где $1 \leq x \leq n - 1, 1 \leq y \leq m - 1$, 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.

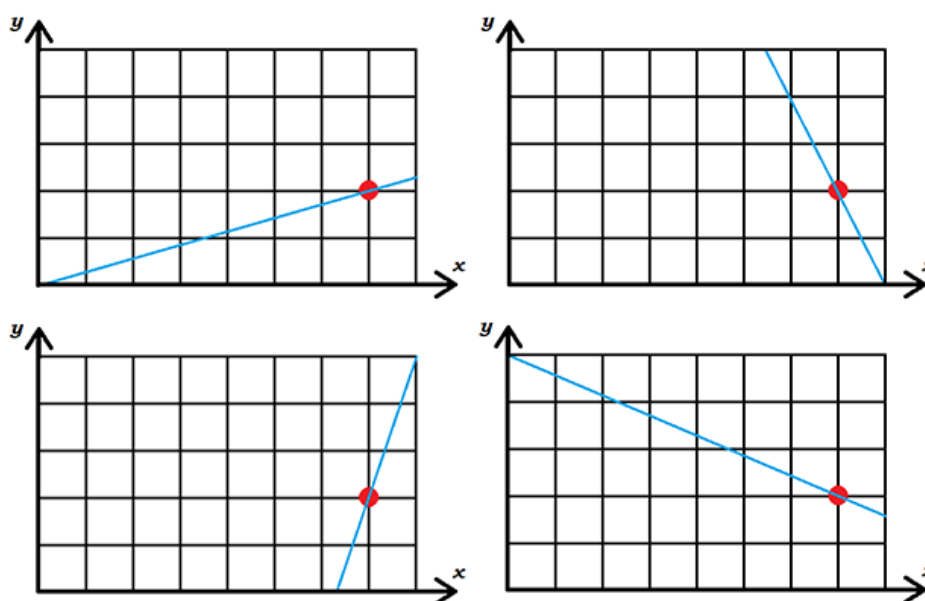


Рис. 2.2.5

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++
1  #include<bits/stdc++.h>
2  #define int long long
3  using namespace std;
4  const int INF = 1e18;
5  double katy(double x, double y, double n){
6      return n * y / x;
7  }
8  double n, m, x, y;
9  double ans = INF;
10 double k1, k2;
11 void upd(){
12     if(k1 < m){
13         double st = k1 * n / 2;
14         ans = min(ans, n * m - 2 * st);
15     }
16     else{
17         double st = k2 * m / 2;
18         ans = min(ans, n * m - 2 * st);
19     }
20 }
21 signed main(){
22     cin >> n >> m >> x >> y;
23     k1 = katy(x, y, n);
24     k2 = katy(y, x, m);
25     upd();
26     k1 = katy(n - x, y, n);
27     k2 = katy(y, n - x, m);
28     upd();
29     k1 = katy(x, m - y, n);
30     k2 = katy(m - y, x, m);
31     upd();
32     k1 = katy(n - x, m - y, n);
33     k2 = katy(m - y, n - x, m);
34     upd();
35     cout.precision(3);
36     cout << fixed << ans << endl;
37 }

```

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 $2 \leq n \leq 2 \cdot 10^5$. Во второй строке через пробел заданы n целых чисел a_i через пробел — длины полосок, где $1 \leq a_i \leq 10^9$.

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.

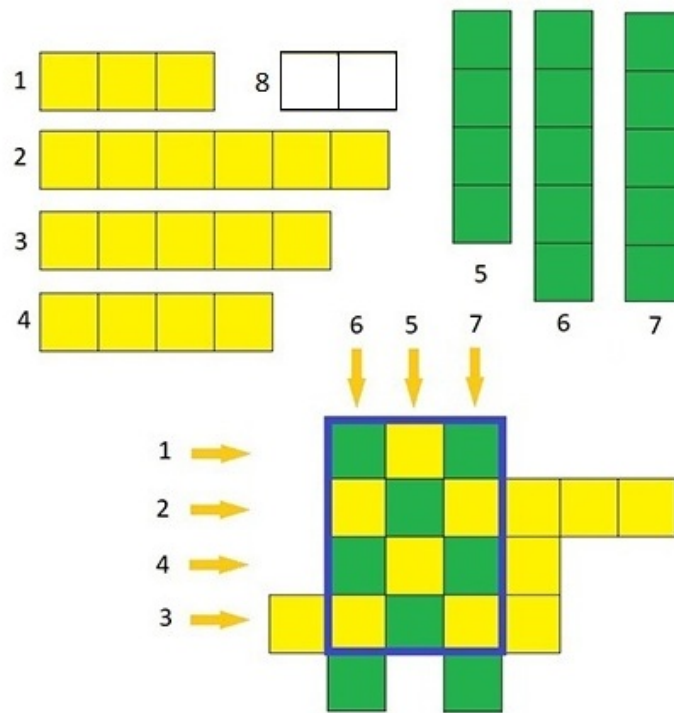


Рис. 2.2.6

Решение

Ниже представлено решение на языке C++.

C++

```

1  #include<bits/stdc++.h>
2  #define int long long
3  using namespace std;
4  signed main(){
5      int n;
6      cin >> n;
7      deque<int> v(n);
8      for(int i = 0; i < n; i++){
9          cin >> v[i];
10     }
11     sort(v.begin(), v.end());
12     int ans = 0;
13     int cnth = 0, minh;
14     while(1){
15         if(v.size() == 0){
16             break;
17         }
18         cnth++;
19         minh = v.back();
20         v.pop_back();
21         while(v.size() > 0 && v[0] < cnth){
22             v.pop_front();
23         }
24         ans = max(ans, cnth * min(minh, (int)v.size()));
25     }
26     cout << ans << endl;
27 }
```

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 $20 \leq n \leq 100$, 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, $20 \leq n \leq 100$.

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++

```

1  #include<bits/stdc++.h>
2  #define int long long
3  using namespace std;
4  string frs;
5  int n;
6  vector<string> dict;
7  vector<int> msk(26, 0);
8  int cnt = 0;
9  vector<vector<int>> amsk;
10 vector<string> ans;
11 bool bigok = 0;
12 void p(int pos){
13     if(!bigok){
14         if(cnt == 0){
15             sort(ans.begin(), ans.end());
16             bigok = 1;
17             return;
18         }
19         for(int i = pos; i < n; i++){
20             string ts = dict[i];
21             bool ok = 1;
22             for(int j = 0; j < 26; j++){
23                 if(amsk[i][j] > msk[j]){
24                     ok = 0;
25                 }
26             }
27             if(ok){
28                 ans.push_back(ts);
29                 for(int j = 0; j < 26; j++){
30                     msk[j] -= amsk[i][j];
31                     cnt -= amsk[i][j];
32                 }
33                 p(i + 1);
34                 if(!bigok){
35                     for(int j = 0; j < 26; j++){
36                         msk[j] += amsk[i][j];
37                         cnt += amsk[i][j];
38                     }
39                 }
40                 ans.pop_back();
41             }
42         }
43     }
44 }
45 signed main(){
46     cin >> frs;
47     cin >> n;
48     amsk.resize(n, vector<int>(26, 0));
49
50     string ts;
51     for(int i = 0; i < n; i++){
52         cin >> ts;
53         dict.push_back(ts);
54     }
55     for(int i = 0; i < n; i++){
56         for(auto el : dict[i]){
57             amsk[i][el - 'a']++;
58         }
59     }

```

```
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 $1 \leq a, b, c, d \leq 10^9$.

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

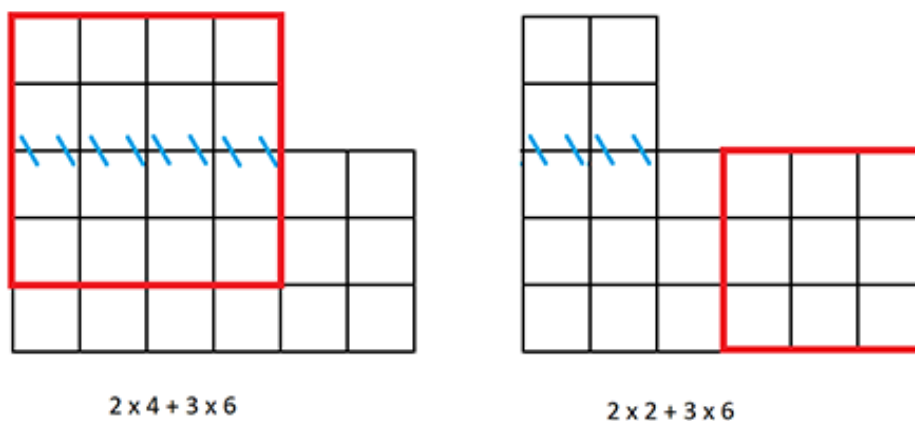
Примечания

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++

```

1 #include<bits/stdc++.h>
2 #define int long long
3 using namespace std;
4 signed main(){
5     int a, b, c, d;
6     cin >> a >> b >> c >> d;
7     int ans = max(min(a, b), min(c, d));
8     int p1 = min(a + c, min(b, d));
9     int p2 = min(a + d, min(b, c));
10    int p3 = min(b + c, min(a, d));
11    int p4 = min(b + d, min(a, c));
12    ans = max({ans, p1, p2, p3, p4});
13    cout << ans << endl;
14 }

```

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 А, С, G, Т.

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 $D_1 = \mathbf{AACGGTAGGT}$, а $D_2 = \mathbf{TCCCGGAACA}$. Выберем $i = 4$ и поменяем местами префиксы длины 4. Получим две новые ДНК, одна из которых будет иметь вид $\mathbf{AACGGGAACA}$, а вторая — $\mathbf{TCCCGTAGGT}$. 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 $10 \leq n \leq 100$, $2 \leq m \leq 100$.

В следующих m строках содержится описание исходной популяции ДНК, каждая задается строкой длины n , состоящей из символов А, С, G и Т.

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
AACCGGTТАА
ACGTACGTAC
AAACCCGGGT
САТТАСТGGA

AAGCGCTТАА
ССАСАСGTGC
ААСТАGGGGT

Стандартный вывод
ААТТССТGAA

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++

```

1  #include<bits/stdc++.h>
2  #define int long long
3  using namespace std;
4  signed main(){
5      int n, m;
6      cin >> n >> m;
7      vector<string> v1(m);
8      for(int i = 0; i < m; i++){
9          cin >> v1[i];
10     }
11     string d;
12     cin >> d;
13     vector<string> v2(m - 1);
14     for(int i = 0; i < m - 1; i++){
15         cin >> v2[i];
16     }
17     for(int j = 0; j < n; j++){
18         int ss = 0;
19         for(int i = 0; i < m; i++){
20             ss ^= (int)(v1[i][j]);
21         }
22         for(int i = 0; i < m - 1; i++){
23             ss ^= (int)(v2[i][j]);
24         }
25         cout << (char)(ss);
26     }
27     cout << endl;
28 }
```

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 $1 \leq A, B, C, D \leq 1000$, $0 \leq E \leq 1000$.

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++

```

1  #include<bits/stdc++.h>
2  #define int long long
3  using namespace std;
4  const int INF = 1e18;
5  vector<int> v(4);
6  int e, ans = INF;
7  void p(vector<int> &v1, vector<int> &vr, int tv){
8      if(v1.size() == 2){
9          ans = min(ans, tv + *max_element(v1.begin(), v1.end()));
10         return;
11     }
12     for(int i = 0; i < v1.size() - 1; i++){
13         for(int j = i + 1; j < v1.size(); j++){
14             vector<int> v11;
15             for(int k = 0; k < v1.size(); k++){
16                 if(k != i && k != j){
17                     v11.push_back(v1[k]);
18                 }
19             }
20             vector<int> vr1 = vr;
```

```

21         vrl.push_back(vl[i] + e);
22         vrl.push_back(vl[j] + e);
23         int tmp = max(vl[i], vl[j]);
24         sort(vrl.rbegin(), vrl.rend());
25         vll.push_back(vrl.back() + e);
26         vrl.pop_back();
27         p(vll, vrl, tv + tmp + vll.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 $2 \leq a \leq 10^6$, $0 \leq b \leq 10^6$.

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++

```

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

```

```

7  int binpow (int a, int n) {
8      int res = 1;
9      while (n > 0) {
10         if (n % 2 == 1)
11             (res *= a) %= MOD;
12         (a *= a) %= MOD;
13         n /= 2;
14     }
15     return res;
16 }
17
18 int bc(int n, int k){
19     int res = f[n];
20     int p1 = binpow(f[k], MOD - 2);
21     int p2 = binpow(f[n - k], MOD - 2);
22     (res *= p1) %= MOD;
23     (res *= p2) %= MOD;
24     return res;
25 }
26 signed main(){
27     for(int i = 1; i <= 2e6; i++){
28         f[i] = (f[i - 1] * i) % MOD;
29     }
30     int a, b;
31     int ans = 0;
32     cin >> a >> b;
33     a--;
34     for(int i = 0; i < a + 1; i++){
35         if(2 * i <= b){
36             int d = bc(a, i);
37             if(b - 2 * i <= a - i){
38                 (d *= bc(a - i, b - 2 * i) ) %= MOD;
39                 (ans += d) %= MOD;
40             }
41         }
42     }
43     cout << ans << endl;
44 }

```

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 прогуливающих джентльменов, где $1 \leq d \leq 200$, $2 \leq n \leq 2000$.

Во второй строке находятся n целых чисел v_i через пробел — скорости каждого джентльмена, где $1 \leq v_i \leq 2000$. Гарантируется, что все скорости попарно различны. Скорости даны в порядке возрастания, то есть $v_1 < v_2 < \dots < v_n$.

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

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.

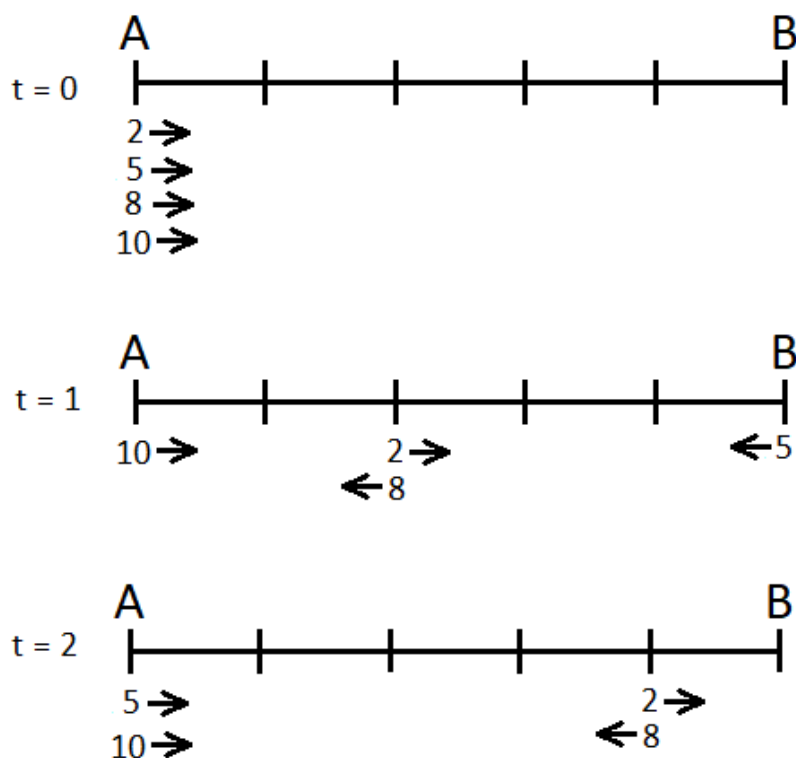


Рис. 2.2.8

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 бинарного поиска с квадратичным нахождением ответа в каждой его итерации.

Решение

Ниже представлено решение на языке C++.

C++

```

1  #include<bits/stdc++.h>
2  #define int long long
3  using namespace std;
4  const double EPS = 1e-7;
5  double x(double M, int V, int d){
6      double dst = V * M;
7      int cnt = floor((dst + EPS) / d);
8      double pin = dst - cnt * d;
9      if(cnt % 2 == 0){
10         return pin;
11     }
12     else{
13         return d - pin;
14     }
15 }
16 int F(double M, vector<int> &v, int d){
17     int res = 0;
18     for(int i = 0; i < v.size(); i++){
19         double dst = v[i] * M;

```

```
20     int cnt = floor((dst + EPS) / d);
21     res += cnt * i;
22     double tx = x(M, v[i], d);
23     for(int j = 0; j < i; j++){
24         double txj = x(M, v[j], d);
25         if(cnt % 2 == 0){
26             res += txj <= tx + EPS;
27         }
28         else{
29             res += txj >= tx - EPS;
30         }
31     }
32 }
33 return res;
34 }
35 signed main(){
36     int d, n;
37     cin >> d >> n;
38     vector<int> v(n);
39     for(int i = 0; i < n; i++){
40         cin >> v[i];
41     }
42     int k;
43     cin >> k;
44     double L = 0, R = 1;
45     while(F(R, v, d) <= k){
46         R *= 2;
47     }
48     R /= 2;
49     while(R - L > 1e-4){
50         double M = (R + L) / 2.0;
51         if(F(M, v, d) < k){
52             L = M;
53         }
54         else{
55             R = M;
56         }
57     }
58     cout.precision(10);
59     cout << fixed << L << endl;
60 }
```

2.3. Предметный тур. Физика

2.3.1. Первая волна. Задачи 8–9 класса

Задачи первой волны предметного тура по физике за 8–9 класс открыты для решения. Соревнование доступно на платформе Яндекс.Контест: <https://contests.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/м³ необходима, чтобы полностью заполнить пространство между стаканами?

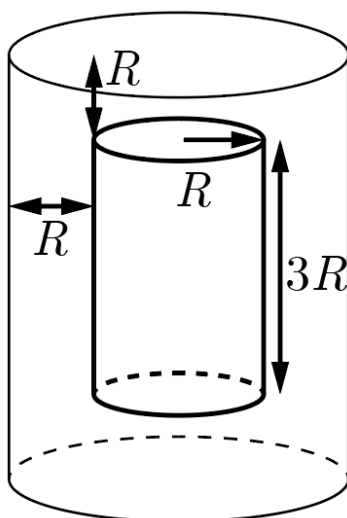


Рис. 2.3.1

Решение

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

$$m = \rho V. \quad (2.3.1)$$

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,

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

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

$$V_2 = 3\pi R^3. \quad (2.3.3)$$

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

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

Погрешность 0,01 кг.

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

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 $\rho_0 = 5,5 \cdot 10^{-8} \text{ Ом} \cdot \text{м}$ до $\rho_1 = 1,1 \cdot 10^{-6} \text{ Ом} \cdot \text{м}$. Определите электрическую мощность, потребляемую лампой в первый момент после ее включения, если в рабочем режиме (полностью прогревшись) лампа потребляет от той же сети мощность $P_1 = 30 \text{ Вт}$.

Решение

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

$$P = U^2/R. \quad (2.3.5)$$

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

$$R = \frac{\rho l}{S}. \quad (2.3.6)$$

Подставляя (2.3.6) в (2.3.5), получим

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

где только ρ зависит от температуры. В результате приходим 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: $P_0 = (600 \pm 1) \text{ Вт}$.

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/с².

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

$$A = mgh + \frac{Fh}{2} \approx 5 \text{ кДж.} \quad (2.3.7)$$

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 :

$$v = \frac{S}{t}. \quad (2.3.8)$$

Для первого дрона это уравнение принимает вид

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

а для второго, соответственно,

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

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

$$v_1\tau + v_2(t - \tau) - v_2\tau - v_1(t - \tau) = \Delta vt. \quad (2.3.11)$$

Перегруппировав слагаемые, получим

$$v_2t - 2v_2\tau - v_1t + 2v_1\tau = \Delta vt, \quad (2.3.12)$$

откуда окончательно

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

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.

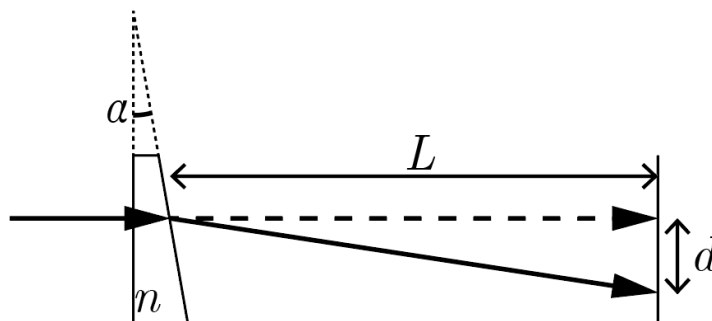


Рис. 2.3.2

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 $\alpha = 3^\circ$. 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

$$n = \frac{\sin \beta}{\sin \alpha}, \quad (2.3.14)$$

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

$$n \approx \frac{\beta}{\alpha}. \quad (2.3.15)$$

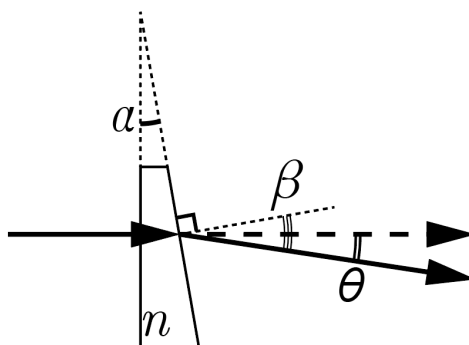


Рис. 2.3.3

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 $\text{tg } \theta = \frac{d}{L}$, откуда

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

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://contests.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 · м/с, преодолевает дистанцию $L = 20$ км ровно за 1,5 мин. За какое

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

Решение

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

$$p_0 = \frac{mL}{t_0}, \quad (2.3.17)$$

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

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

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

$$t = \frac{Lp_0t_0}{L(p_0 - \Delta p)} = 100 \text{ с}. \quad (2.3.19)$$

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 $a_0 = 2,0$ м/с², и начинал скользить при превышении этого значения. Совершая маневр, грузовик приобрел ускорение $a = 2,12$ м/с², направленное по ходу движения. Какое время длился маневр, если в результате контейнер сдвинулся на $l = 1,5$ 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

$$F = ma_0, \quad (2.3.20)$$

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

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

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

$$t = \sqrt{\frac{2l}{a - a_0}} = 5 \text{ с.} \quad (2.3.22)$$

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 .

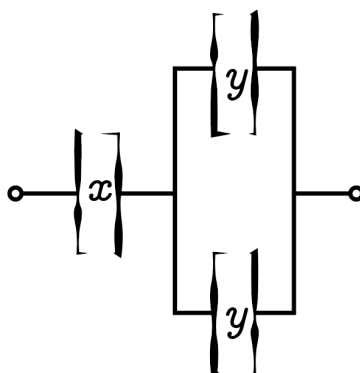


Рис. 2.3.4

Решение

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

$$R = x + \frac{y}{2}. \quad (2.3.23)$$

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

$$C = \frac{2xy}{x + 2y}. \quad (2.3.24)$$

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

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

Домножив полученное уравнение на знаменатель дроби и раскрыв скобки, получим

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

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

$$10 + 3y = 10y - y^2. \quad (2.3.27)$$

Это квадратное уравнение, которое легко привести к канонической форме

$$y^2 - 7y + 10 = 0 \quad (2.3.28)$$

и решить с помощью дискриминанта

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

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, необходимому 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 $T_0 = -273^\circ\text{C}$.

Решение

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

$$\eta = \frac{T_2 - T_1}{T_2}, \quad (2.3.30)$$

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

$$\frac{T_2 - T_1}{T_2} = \eta = \frac{A}{Q_2}. \quad (2.3.31)$$

Отсюда Q_2 может быть выражено как

$$Q_2 = \frac{AT_2}{T_2 - T_1}. \quad (2.3.32)$$

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

$$Q_1 = Q_2 - A = \frac{AT_1}{T_2 - T_1}. \quad (2.3.33)$$

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 $T_1 = 0^\circ\text{C} = 273\text{K}$, $T_2 = \theta$, $Q_1 = \lambda m$, поскольку забираемая у теплового резервуара с меньшей температурой теплота идет на замораживание в нем воды. Подставив эти данные в (2.3.33), получим

$$\lambda m = \frac{AT_1}{\theta - T_1}, \quad (2.3.34)$$

откуда окончательно выразим ответ

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

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 $t_1 = 70$ °С, после чего перестает увеличивать свою температуру, а при пропускании через него тока $I_2 = 20$ А — до температуры $t_2 = 205$ °С.

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

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

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

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

где R — сопротивление катушки.

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

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

Тривиальными алгебраическими преобразованиями выразим θ

$$(\theta - t_1)I_2^2 = (\theta - t_2)I_1^2 \Rightarrow \theta = \frac{t_1 I_2^2 - t_2 I_1^2}{I_2^2 - I_1^2} = 25 \text{ °С}. \quad (2.3.39)$$

Погрешность: 0,1 °С.

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://contests.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 $c_{уд}$ 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	$c_{уд}$, кДж/(кг · °С)
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

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

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

$$m = \rho V, \quad (2.3.41)$$

откуда

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

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

Таблица 2.3.2

	Минерал	ρ , г/см ³	$c_{уд}$, кДж/(кг · °С)	$c_{уд}\rho$, кДж/(м ³ · °С)
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.

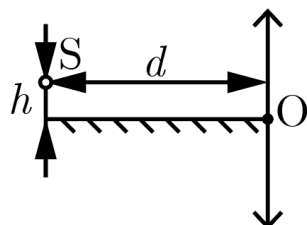


Рис. 2.3.5

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).

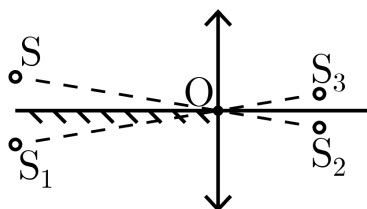


Рис. 2.3.6

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

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

Искомое расстояние l между изображениями S_2 и S_3 , благодаря подобию треугольников $\triangle OSS_1$ и $\triangle OS_2S_3$, 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

$$\frac{l}{2h} = \frac{f}{d} = \frac{F}{d - F}. \quad (2.3.44)$$

Отсюда окончательно находим

$$l = 2h \frac{F}{d - F} = 2 \text{ см}. \quad (2.3.45)$$

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 пропилов, длившийся $t = 3$ с, выделилось $Q = 5$ кДж тепла, а пила соприкасалась с заготовкой только узкой полоской своей внешней кромки.

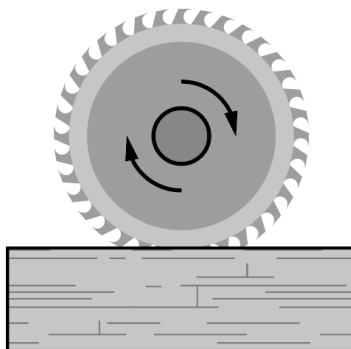


Рис. 2.3.7

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

$$A = FS = Fvt, \quad (2.3.46)$$

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

$$v = \frac{\pi D}{T} = \pi D\nu, \quad (2.3.47)$$

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

$$Q = \pi F D \nu t \Rightarrow F = \frac{Q}{\pi D \nu t} \approx 37 \text{ Н}. \quad (2.3.48)$$

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

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

Сократив v_0 и перегруппировав слагаемые, получим

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

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

$$t = \tau \frac{\delta + 1}{2\delta}. \quad (2.3.51)$$

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

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

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

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

Отсюда окончательно

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

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. Сценка (30 баллов)

Условие

Два абсолютно одинаковых ползунковых реостата, сопротивления которых могут изменяться в пределах от 0 до $R_0 = 2$ 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.

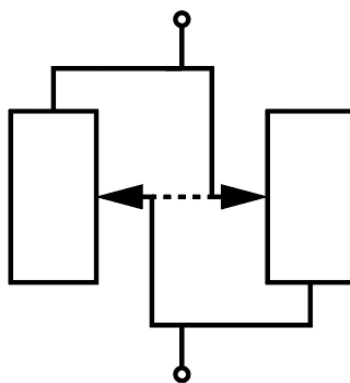


Рис. 2.3.8

Решение

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

$$R = \frac{R_1 R_2}{R_1 + R_2}. \quad (2.3.55)$$

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

$$R_2 = R_0 - R_1. \quad (2.3.56)$$

Подставляя этот результат в (2.3.55), получим

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

График полученной функции является параболой. Её 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 её вершине (которая, как легко видеть из симметрии или непосредственно по формуле $x_{max} = -b/(2a)$, лежит в центре диапазона, при $R_1 = R_2 = \frac{R_0}{2}$) сопротивление батареи равно $\frac{R_0}{4}$.

Таким образом,

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

Погрешность 0,01 кОм.

Ответ: $l = (0,50 \pm 0,01) \text{ кОм}$.

2.3.4. Вторая волна. Задачи 10–11 класса

Задачи второй волны предметного тура 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 течение $\tau = 10$ мин подавало на аккумулятор ток, линейно возрастающий 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

$$q_1 = 2,5\tau I_0, \quad (2.3.59)$$

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

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

Складывая эти заряды, получим окончательно

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

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.

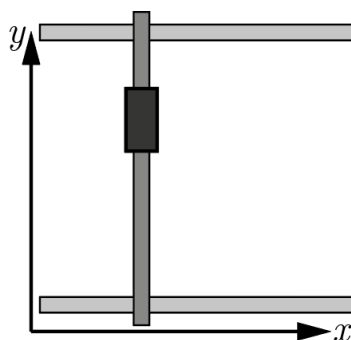


Рис. 2.3.9

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

$$v = \frac{2\pi R}{T}. \quad (2.3.62)$$

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

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

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 $a = 40$ 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?

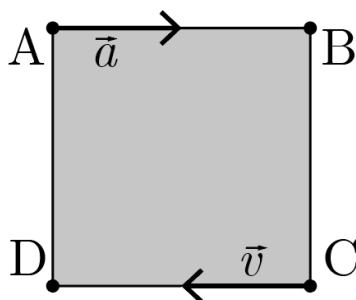


Рис. 2.3.10

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

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

Погрешность 0,5 см.

Ответ: $(28,3 \pm 0,5)$ 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.

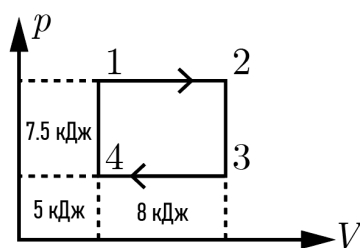


Рис. 2.3.11

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:

$$\eta = \frac{A}{Q}. \quad (2.3.65)$$

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

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

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

$$A_{412} = 12 + 8 = 20 \text{ кДж}, \quad (2.3.67)$$

а внутренняя энергия может быть удобно вычислена по формуле

$$U = \frac{3}{2}PV, \quad (2.3.68)$$

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

$$U_4 = \frac{3}{2}5 = 7,5 \text{ кДж}, \quad (2.3.69)$$

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

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

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

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

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 $a = 15$ м, толщина его стенок $d = 1,5$ м, длина киля судна (высота призмы) $l = 40$ м. Инженерами было рассчитано, что для сохранения устойчивости судна центр тяжести сыпучего груза, перевозимого в трюме, должен быть хотя бы на $h = 2$ 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?

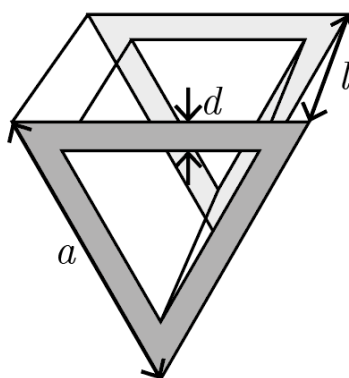


Рис. 2.3.12

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.

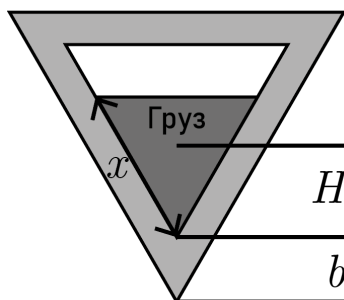


Рис. 2.3.13

Отсчитывая от киля (нижней вершины треугольника) высоту y_B , на которой

находится центр тяжести вытесненной воды, легко выразить как

$$y_{\text{в}} = \frac{\sqrt{3}}{3}a, \quad (2.3.72)$$

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

Высота $y_{\text{г}}$ центра тяжести груза может быть найдена как

$$y_{\text{г}} = b + H, \quad (2.3.73)$$

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

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

где x — сторона треугольника, являющегося поперечным сечением груза.

Учитывая $y_{\text{г}} + h = y_{\text{в}}$, получим

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

откуда

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

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

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

Погрешность 10 м^3 .

Ответ: $(696 \pm 10) \text{ м}^3$.

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://contests.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

$$p = \rho gh. \quad (2.3.78)$$

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

$$F = pS = \rho ghS. \quad (2.3.79)$$

Искомая расчетная 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

$$f = \frac{3F}{16} = \frac{3\rho ghS}{16} \approx 9,9 \text{ кН}. \quad (2.3.80)$$

Погрешность 0,1 кН.

Ответ: $(9,9 \pm 0,1)$ кН.

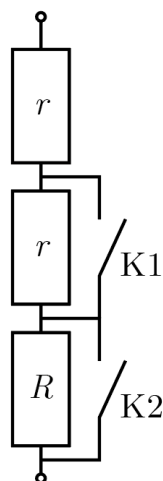
Задача 2.3.5.2. Реостат (15 баллов)**Условие**

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 К1 фактически эквивалентно замене среднего резистора на отрезок провода, а ключа К2 — такой же замене нижнего. Учитывая это и применяя формулу эквивалентного сопротивления последовательно соединенных резисторов, легко получим

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

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

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

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·°С) и средней плотностью $\rho = 10^4$ кг/м³, если, двигаясь в трубе диаметром $d = 10$ см со скоростью $v = 20$ м/с, он нагревается от $t_1 = 400$ °С до $t_2 = 900$ °С.

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

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

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

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

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.

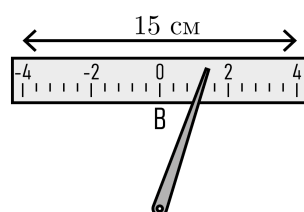


Рис. 2.3.15

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).

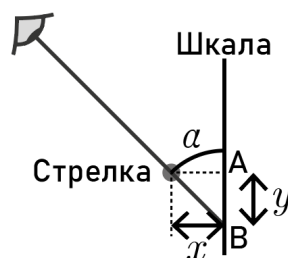


Рис. 2.3.16

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

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

Отсюда окончательно

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

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 А 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

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

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

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

откуда легко найти

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

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

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

Погрешность 0,01.

Ответ: $2,00 \pm 0,01$.

2.3.6. Третья волна. Задачи 10–11 класса

Задачи третьей волны предметного тура по 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, $t_x = 0^\circ\text{C}$. Аналогично, при атмосферном давлении температура кипящей воды может быть равна только $t_n = 100^\circ\text{C}$. КПД идеальной тепловой машины (машины Карно) с известными абсолютными термодинамическими температурами T_n и T_x холодильника задается выражением

$$\eta_0 = \frac{T_n - T_x}{T_n}. \quad (2.3.91)$$

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

$$\eta = \frac{\eta_0}{\alpha} = \frac{t_n - t_x}{\alpha(t_n - T_0)} \approx 16,8\%. \quad (2.3.92)$$

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 $\vec{v}_{дв}$ дрона относительно воды равна (векторной) сумме его скорости $\vec{v}_{дк}$ относительно корабля и скорости $\vec{v}_{кв}$ корабля относительно воды. Поскольку направление вектора $\vec{v}_{кв}$ неизменно, а вектор $\vec{v}_{дк}$ 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 противоположны. Эти два случая и будут соответствовать максимальному и минимальному значениям модуля суммы этих векторов, равным $v_{max} = |\vec{v}_{дк}| + |\vec{v}_{кв}|$ и $v_{min} = ||\vec{v}_{дк}| - |\vec{v}_{кв}||$ соответственно.

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

$$v_{дк} = \frac{2\pi R}{T}. \quad (2.3.93)$$

Отсюда получим окончательно

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

Погрешность 0,02.

Ответ: $3,68 \pm 0,02$.

Задача 2.3.6.3. Конденсатор (20 баллов)

Условие

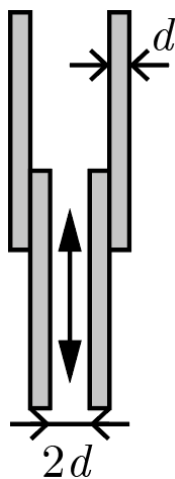


Рис. 2.3.17

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 выдвигении 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

$$C_1 = \frac{S\varepsilon_0}{2d}, \quad (2.3.95)$$

где ε_0 — диэлектрическая постоянная.

Аналогично, емкость полностью выдвинутой внешней пары равна

$$C_2 = \frac{S\varepsilon_0}{4d}, \quad (2.3.96)$$

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

Тогда для искомого отношения максимальной и минимальной емкостей получим:

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

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 $V = 645 \text{ м}^3$, а общая масса его оболочки, корзины и полезной нагрузки $M = 150 \text{ кг}$. 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

$$p_0V = \frac{m}{\mu}RT, \quad (2.3.98)$$

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

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

$$\begin{aligned} m_1 &= \frac{\mu p_0 V}{R(T_0 + \Delta t)}, \\ m_2 &= \frac{\mu p_0 V}{RT_0}, \end{aligned} \quad (2.3.99)$$

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)

$$m_2g = Mg + m_1g \Rightarrow \frac{\mu p_0 V}{RT_0} = M + \frac{\mu p_0 V}{R(T_0 + \Delta t)}, \quad (2.3.100)$$

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

$$\mu p_0 V(T_0 + \Delta t) = MRT_0(T_0 + \Delta t) + \mu p_0 V T_0. \quad (2.3.101)$$

В канонической форме:

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

Остается найти (методом дискриминанта) единственный положительный корень этого уравнения

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

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

$$F = G \frac{mM}{R^2}, \quad (2.3.104)$$

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

$$m \frac{v^2}{R} = G \frac{mM}{R^2}. \quad (2.3.105)$$

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 $^{\circ}$ С), а в жидкой — $c_2 = 1,5$ кДж/(кг $^{\circ}$ С), температура ее плавления $\theta = 800$ $^{\circ}$ С. Определите, какую массу соли необходимо взять, чтобы при ее нагреве от $t_0 = 20$ $^{\circ}$ С до $t_1 = 1200$ $^{\circ}$ С запастись $Q = 1$ МДж тепла.

Решение

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

$$Q_1 = c_1 m (\theta - t_0), \quad (2.3.106)$$

где m — масса нагреваемой соли.

Количество теплоты, уходящее непосредственно на плавление

$$Q_2 = \lambda m. \quad (2.3.107)$$

Наконец, количество теплоты, уходящее на нагрев расплава соли,

$$Q_3 = c_2 m (t_1 - \theta). \quad (2.3.108)$$

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

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

откуда окончательно

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

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, переносающий 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:

$$P_1 = (1 - \eta)\eta P. \quad (2.3.111)$$

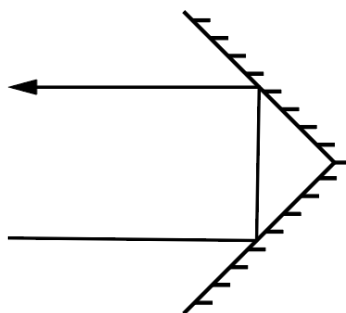


Рис. 2.3.18

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

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

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.

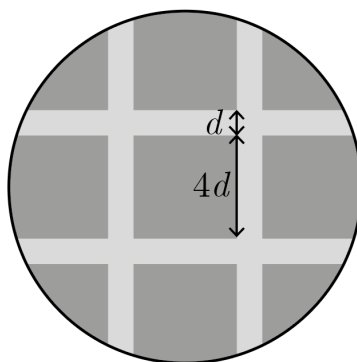


Рис. 2.3.19

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.

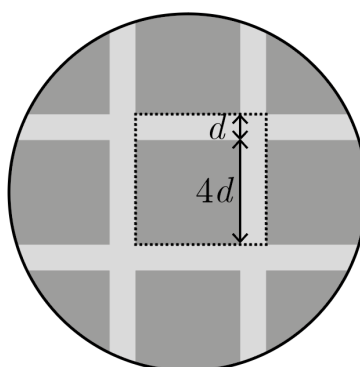


Рис. 2.3.20

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

$$m = m_1 + m_2 = \rho_1 V_1 + \rho_2 V_2 = \rho_1 9dh + \rho_2 16dh. \quad (2.3.113)$$

Тогда искомая плотность окончательно равна

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

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 $R = 21$ см и $r = 3$ см. Чтобы поднять при помощи этого домкрата груз $m = 1,4$ т, установленный на платформе большого цилиндра, к точке A рычага необходимо приложить силу не менее $F = 87,5$ Н. Определите отношение $AB : BC$. Ускорение свободного падения $g = 9,8$ м/с². На рисунке точный масштаб не сохранен.

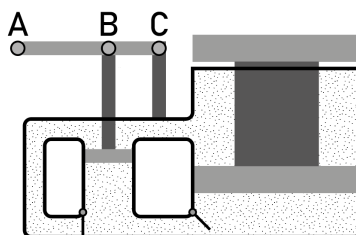


Рис. 2.3.21

Решение

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

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

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

$$\frac{F_1}{F} = \frac{AC}{BC} = \frac{AB + BC}{BC} = 1 + \frac{AB}{BC}. \quad (2.3.116)$$

Совместив эти уравнения, получим окончательно

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

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 $\Delta I = 0,1$ А над номинальным значением приводит к увеличению тепловой мощности, выделяемой прибором, на $\Delta P_1 = 44$ Вт,

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

Решение

Согласно закону Джоуля – Ленца, тепловая мощность электронагревателя равна

$$P = I^2 R, \quad (2.3.118)$$

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

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

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

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

Эту систему можно решить, вычитая второе уравнение из первого

$$\frac{\Delta P_1 + \Delta P_2}{P_0} = 4x \Rightarrow x = \frac{\Delta P_1 + \Delta P_2}{4P_0} \quad (2.3.121)$$

и подставляя результат в любое уравнение системы (2.3.120)

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

Домножим на $16P_0^2$

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

и выразим окончательно

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

Погрешность 1 Вт.

Ответ: 100 ± 1 Вт.

2.3.8. Четвертая волна. Задачи 10–11 класса

Задачи четвертой волны предметного тура по 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

$$a = \frac{v^2}{R} = \omega^2 R, \quad (2.3.125)$$

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

$$a^2 = \frac{v^2}{R} \omega^2 R \Rightarrow a = v\omega. \quad (2.3.126)$$

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

$$\frac{a_A}{a_B} = \frac{v_A}{v_B} \cdot \frac{\omega_A}{\omega_B} = \frac{1}{2} \cdot \frac{3}{1} = 1,5. \quad (2.3.127)$$

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/с².

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

$$q\vec{E} + m\vec{g} + \vec{T} = \vec{0}. \quad (2.3.128)$$

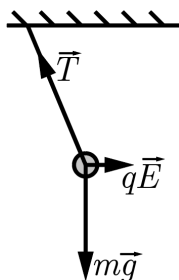


Рис. 2.3.22

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

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

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

$$x = \frac{x_1 + x_2}{2}. \quad (2.3.130)$$

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 Ox на любом промежутке времени можно записать

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

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 Ox равна среднему арифметическому мгновенной скорости в этой же проекции впереди и позади идущих автомобилей. Повторение этих рассуждений приводит к аналогичному результату для ускорений

$$a_x = \frac{a_{x1} + a_{x2}}{2}. \quad (2.3.132)$$

Выразим из этого уравнения проекцию на Ox искомого ускорения замыкающего автомобиля a_{x2}

$$a_{x2} = 2a_x - a_{x1}. \quad (2.3.133)$$

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

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

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

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

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

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

Совмещая эти равенства, получим окончательно

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

Погрешность $0,02 \text{ м/с}^2$.

Ответ: $(1,96 \pm 0,02) \text{ м/с}^2$.

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

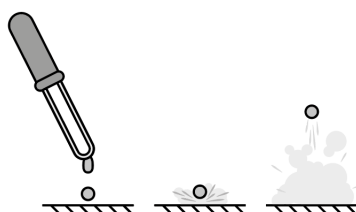


Рис. 2.3.23

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 этой части энергии

$$\Delta U = (1 - \eta)Q = Lm\alpha, \quad (2.3.138)$$

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

$$Q = \frac{Lm\alpha}{1 - \eta}. \quad (2.3.139)$$

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

$$A = \eta Q = (1 - \alpha)mgh. \quad (2.3.140)$$

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

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

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/66689/enter/>.

Задача 2.4.1. Аэродинамическое качество (14 баллов)

Тема: аэродинамическое качество.

Условие

Летательный аппарат (ЛА) совершает горизонтальный полет на высоте $H = 13\,500$ м с постоянной скоростью $V = 550$ км/ч. Масса ЛА $m = 9\,980$ кг, площадь крыла $S = 36,7$ м², коэффициент лобового сопротивления $Cx_a = 0,025$. Плотность воздуха $\rho = 0,2465$ кг/м³, ускорение свободного падения $g = 9,81$ м/с².

Горизонтальный полет — прямолинейный полет с постоянной скоростью на постоянной высоте. Схема сил, действующих на самолет, показана на рис. 2.4.1.

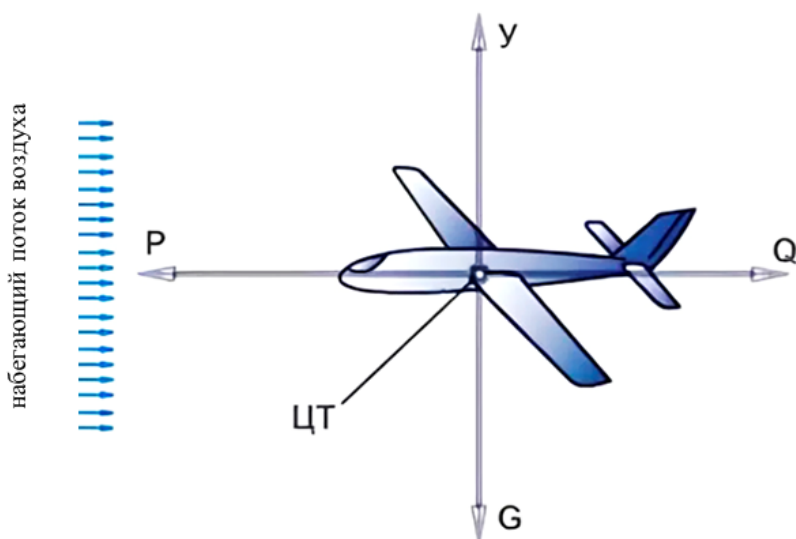


Рис. 2.4.1. Полет ЛА на постоянной высоте: Y — подъемная сила, P — тяга самолета, Q — сила лобового сопротивления, $ЦТ$ — центр тяжести, G — сила тяжести

$$Q = Cx_a \frac{\rho V^2}{2} S \text{ — лобовое сопротивление, Н;}$$

$$Y = Cy_a \frac{\rho V^2}{2} S \text{ — подъемная сила, Н.}$$

Аэродинамическое качество K в горизонтальном полете равно отношению подъемной силы к силе лобового сопротивления:

$$K = \frac{Y}{Q}.$$

Определите аэродинамическое качество K в горизонтальном полете. Ответ округлите до целого числа.

Решение

Введем систему координат, связанную с летательным аппаратом. Начало координат совпадает с его центром тяжести; ось X направлена вдоль продольной оси в сторону носа ЛА; ось Y перпендикулярна оси X , лежит в вертикальной плоскости симметрии ЛА и направлена вверх; ось Z перпендикулярна осям X и Y и направлена вдоль крыла вправо.

Воспользуемся вторым законом Ньютона и запишем уравнение для проекций на ось OY сил, действующих на летательный аппарат:

$$Y - mg = 0.$$

Выразим коэффициент подъемной силы через силу тяжести:

$$C_{y_a} = \frac{mg}{\frac{\rho V^2}{2} S}.$$

Найдем выражение для аэродинамического качества, используя формулы из условия:

$$K = \frac{Y}{Q} = \frac{C_{y_a} \frac{\rho V^2}{2}}{C_{x_a} \frac{\rho V^2}{2}} = \frac{C_{y_a}}{C_{x_a}} = \frac{\frac{mg}{\frac{\rho V^2}{2} S}}{C_{x_a}} = \frac{2mg}{C_{x_a} \rho V^2 S}.$$

Подставим значения из условия и учтем перевод скорости из километров в час в метры в секунду:

$$K = \frac{2 \cdot 9980 \cdot 9,81}{0,025 \cdot 0,2465 \cdot \left(\frac{550}{3,6}\right)^2 \cdot 36,7} \approx 37,0925.$$

По условию требуется округлить ответ до целого числа. Получаем 37.

Ответ: 37.

Задача 2.4.2. Настройка ПИД-регулятора (25 баллов)

Тема: настройка ПИД-регулятора.

Условие

Одним из самых простых и эффективных подходов к управлению движением БВС является использование ПИД-регуляторов. Такие регуляторы включают в себя три звена: пропорциональное, интегральное и дифференциальное. Структура ПИД-регулятора представлена на рис. 2.4.2.

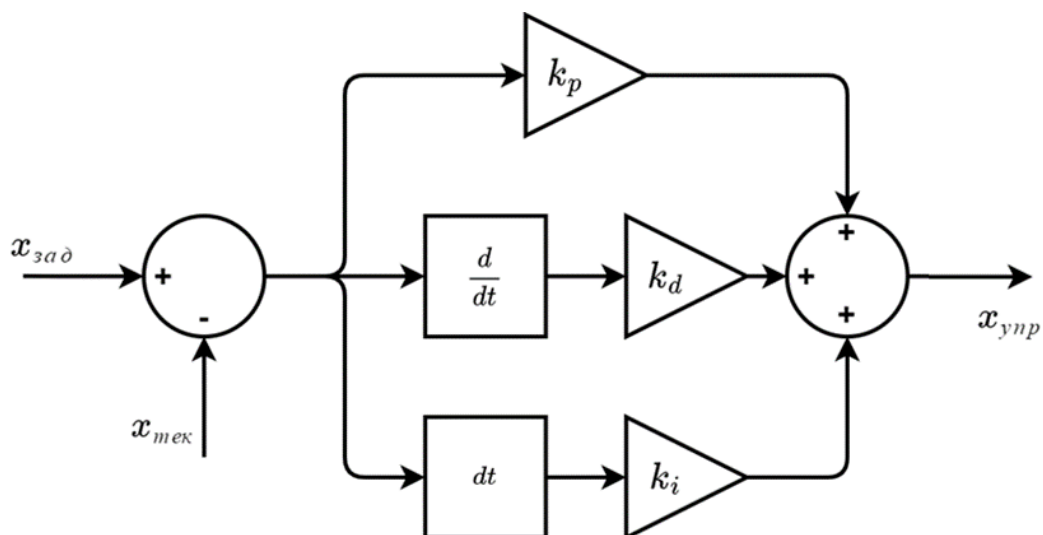


Рис. 2.4.2

Как показано на рис. 2.4.2, на вход регулятора подается значение ошибки регулируемой величины (разница целевого значения $x_{зад}$ и текущего значения $x_{тек}$). Эта ошибка поступает на вход трех звеньев: пропорционального, интегрального и дифференциального. Пропорциональное звено умножает (усиливает) значение ошибки на некоторый постоянный коэффициент k_p , интегральное звено умножает интеграл ошибки на коэффициент k_i , а дифференциальное звено умножает производную ошибки на коэффициент k_d . На выходе ПИД-регулятора формируется управляющий сигнал $x_{упр}$, размерность управляющего сигнала соответствует производной управляемой величины.

Математическое представление этой структуры имеет вид:

$$\delta = x_{зад} - x_{тек};$$

$$x_{упр} = k_p \delta + k_i \int \delta dt + k_d \frac{d}{dt} \delta.$$

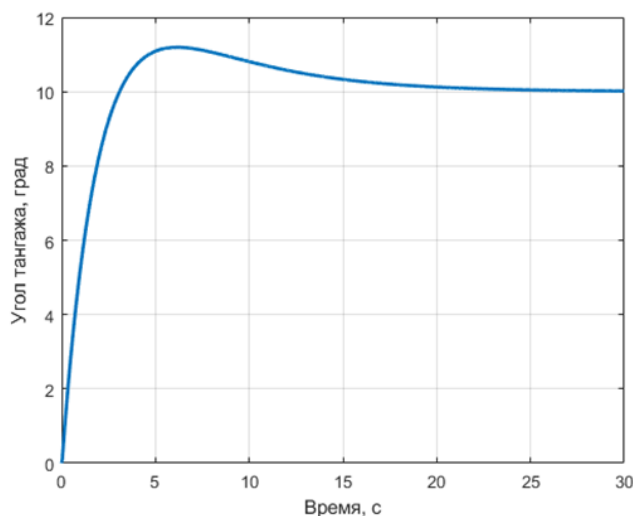


Рис. 2.4.3

Таким образом, для формирования управляющего сигнала необходимо знать только целевое значение регулируемой величины, ее текущее значение и значения трех постоянных коэффициентов.

Естественным подходом к нахождению значений коэффициентов k_p , k_i , k_d является ручной перебор их значений с целью получения удовлетворительного переходного процесса. Пример такого процесса представлен на рис. 2.4.3.

При моделировании точные значения коэффициентов можно вычислить, если построить математическую модель всей системы в целом (БВС). Однако для реального БВС точной модели движения не существует, поэтому можно найти лишь приближенные значения коэффициентов. Для этого используют сложные математические методы оптимизации вроде метода градиентного спуска и других. Применение этих методов на практике довольно трудозатратно и не всегда возможно.

В то же время большинство современных полетных контроллеров обладают функцией автоматического определения параметров регулятора. Они тоже используют математический метод оптимизации, только упрощенный — метод Циглера – Никольса.

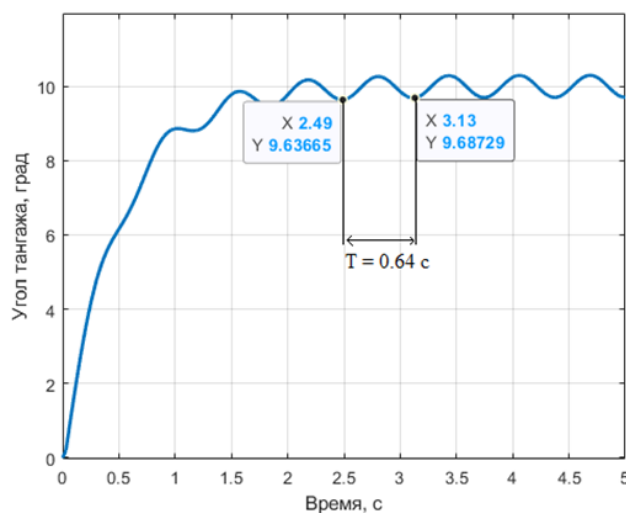


Рис. 2.4.4

Этот метод достаточно прост и заключается в последовательном выполнении следующих операций:

1. обнуляем все коэффициенты регулятора;
2. задаем некоторое целевое значение регулируемой величины $x_{зад}$;
3. постепенно начинаем увеличивать пропорциональный коэффициент k_p и отслеживаем реакцию системы (БВС);
4. при определенном значении коэффициента возникнут незатухающие колебания регулируемой величины (см. рис. 2.4.4);
5. записываем значение пропорционального коэффициента, при котором возникают незатухающие колебания, как K .

Из полученного значения K можно рассчитать значения всех коэффициентов

ПИД-регулятора. Для этого воспользуемся соотношениями:

$$k_p = 0,6 \times K;$$

$$k_i = \frac{2k_p}{T};$$

$$k_d = \frac{k_p T}{8},$$

где T — период колебаний системы (см. рис. 2.4.4).

Рассчитайте значения всех трех коэффициентов k_p , k_i , k_d ПИД-регулятора управления углом тангажа БВС, если при формировании целевого значения угла тангажа $x_{зад} = 10^\circ$, и значении пропорционального коэффициента $k_p = 1,4$ 1/с был получен следующий отклик системы (переходный процесс) — рис. 2.4.5.

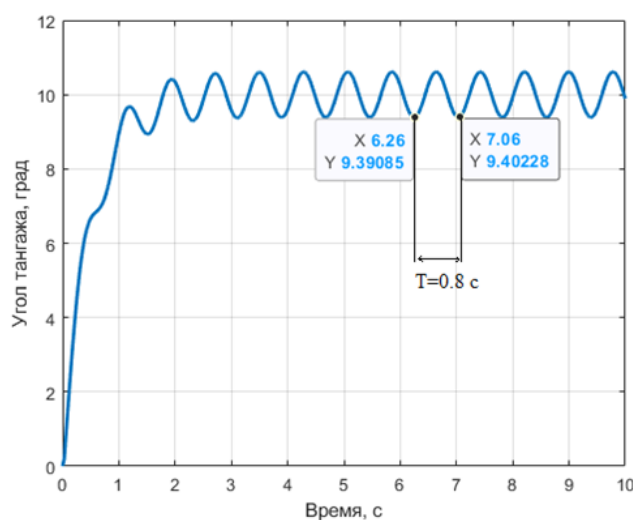


Рис. 2.4.5

Варианты ответа:

1. $k_p = 1,12$, $k_i = 2,8$, $k_d = 0,112$;
2. $k_p = 1,4$, $k_i = 3,5$, $k_d = 0,14$;
3. $k_p = 1,75$, $k_i = 4,375$, $k_d = 0,175$;
4. $k_p = 0,84$, $k_i = 2,1$, $k_d = 0,084$;
5. $k_p = 1,4$, $k_i = 2,5$, $k_d = 0,08$;
6. $k_p = 0,84$, $k_i = 1,05$, $k_d = 0,084$.

Решение

Согласно методу Циглера – Никольса нужно значение K . Из условия задачи получаем, что $K = 1,4$.

Для расчета коэффициентов ПИД-регулятора также требуется значение T . Из представленного в условии графика получаем $T = 0,8$ с.

Подставим полученные значения в формулы коэффициентов из условия:

$$k_p = 0,6 \times K = 0,84;$$

$$k_i = \frac{2k_p}{T} = 2,1;$$

$$k_d = \frac{k_p T}{8} = 0,084.$$

Ответ: 4.

Задача 2.4.3. Кватернион ориентации (25 баллов)

Темы: математика, навигация.

Условие

Привычным способом представления ориентации объекта являются углы его поворота вокруг трех взаимоперпендикулярных осей — углы Эйлера. В бытовых задачах это действительно самый удобный и интуитивно понятный способ представления ориентации объекта, однако для представления ориентации подвижных объектов с шестью степенями свободы (например, БВС) углы Эйлера плохо подходят по двум основным причинам. Причина первая — при вращении БВС вокруг трех осей итоговая ориентация будет зависеть от порядка поворотов. Вторая причина — явление «складывания рамок» (gimbal lock), при котором одна из осей поворота фактически вырождается. Подробнее про этот эффект можно почитать по ссылке: https://ru.wikipedia.org/wiki/Складывание_рамок.

В этой связи в задачах навигации подвижных объектов для представления их ориентации используют кватернионы. С научной точки зрения кватернион — вектор из четырех гиперкомплексных чисел. С практической точки зрения — это вектор из четырех параметров, который позволяет однозначно представить ориентацию объекта в пространстве и не обладает недостатками углов Эйлера. Математически кватернион $q(w, x, y, z)$ можно записать так:

$$q = w + xi + yj + zk,$$

где w, x, y, z — вещественные числа; i, j, k — мнимые единицы.

Параметры x, y, z характеризуют направление оси вращения (проекции орта оси вращения на оси i, j, k). Параметр w характеризует величину угла поворота.

Таким образом, кватернион описывает ориентацию объекта как кратчайший поворот вокруг произвольно направленной оси (в то время как при использовании углов Эйлера вращение всегда происходит вокруг одних и тех же осей).

Кватернион обладает следующими свойствами:

- модуль кватерниона всегда равен единице;
- кватернион может быть использован как для представления ориентации объекта, так и для представления изменения ориентации (вращения);
- кватернион вращения всегда описывает кратчайший поворот.

Допустим, у нас есть кватернион ориентации объекта S и кватернион вращения R . Тогда итоговую ориентацию объекта T после поворота из ориентации S на кватернион R можно получить путем векторного перемножения:

$$\begin{aligned} T &= R \times S; \\ T_w &= R_w \cdot S_w - R_x \cdot S_x - R_y \cdot S_y - R_z \cdot S_z; \\ T_x &= R_x \cdot S_w + R_w \cdot S_x + R_z \cdot S_y - R_y \cdot S_z; \\ T_y &= R_y \cdot S_w - R_z \cdot S_x + R_w \cdot S_y + R_x \cdot S_z; \\ T_z &= R_z \cdot S_w + R_y \cdot S_x - R_x \cdot S_y + R_w \cdot S_z. \end{aligned}$$

Найдите итоговый кватернион ориентации БВС T , если до осуществления поворота ориентация БВС была представлена кватернионом $S = (0,707, 0, 0,707, 0)$, а кватернион поворота равен $R = (0,95, 0,083, -0,026, 0,3)$.

Примечание: в силу округления до третьего знака после запятой и ошибок вычисления, модуль итогового кватерниона может незначительно отличаться от единицы.

Варианты ответа:

1. $T = (0,478, -0,002, -0,833, 0,278)$,
2. $T = (0,69, 0,293, 0,562, 0,349)$,
3. $T = (0,269, -0,39, 0,748, -0,465)$,
4. $T = (0,269, 0,691, 0,152, 0,654)$,
5. $T = (0,679, -0,297, 0,570, -0,354)$,
6. $T = (0,69, 0,2708, 0,6533, 0,1534)$.

Решение

Воспользуемся формулами для итогового кватерниона ориентации БВС T , представленными в условии, и подставим из условия значения кватернионов R и S :

$$\begin{aligned} T_w &= R_w \cdot S_w - R_x \cdot S_x - R_y \cdot S_y - R_z \cdot S_z = 0,69; \\ T_x &= R_x \cdot S_w + R_w \cdot S_x + R_z \cdot S_y - R_y \cdot S_z = 0,2708; \\ T_y &= R_y \cdot S_w - R_z \cdot S_x + R_w \cdot S_y + R_x \cdot S_z = 0,6533; \\ T_z &= R_z \cdot S_w + R_y \cdot S_x - R_x \cdot S_y + R_w \cdot S_z = 0,1534. \end{aligned}$$

Ответ: 6.

Задача 2.4.4. Обработка изображения с БВС (25 баллов)

Темы: геометрия, локализация объекта.

Условие

При поиске различных объектов при помощи БВС чаще всего используются алгоритмы распознавания объектов на изображении, однако после обнаружения объекта на изображении необходимо определить его координаты в реальном мире. При

решении такой задачи необходимо учитывать множество факторов, в частности, рельеф обследуемой местности.

Определите координату X и высоту h точки на местности, которая лежит на оптической оси камеры БВС, летящего горизонтально вдоль оси X так, что его камера находится на высоте 30 м. При этом угол θ отклонения оптической оси камеры от плоскости горизонта составляет 45° , по другим осям отклонений камеры от осей БВС нет. Координата X камеры БВС 3 м. Для определения высоты имеется карта высот с шагом 1 м. В промежутках между отметками высот высота определяется линейной аппроксимацией.

Карта высот необходимого участка приведена в таблице 2.4.1.

Таблица 2.4.1

Координата X , м.	0	1	2	3	4	5	6	7
Высота h , м.	0	0	0	2	4	7	7	4
Координата X , м.	8	9	10	11	12	13	14	15
Высота h , м.	3	2	-1	-4	-4	-6	0	5
Координата X , м.	16	17	18	19	20	21	22	23
Высота h , м.	18	16	6	7	7	5	4	3
Координата X , м.	24	25	26	27	28	29	30	31
Высота h , м.	2	1	0	0	0	0	0	5

Точность ответа — до второго знака после запятой. Иллюстрация полета БВС и данных параметров приведены на рис. 2.4.6.

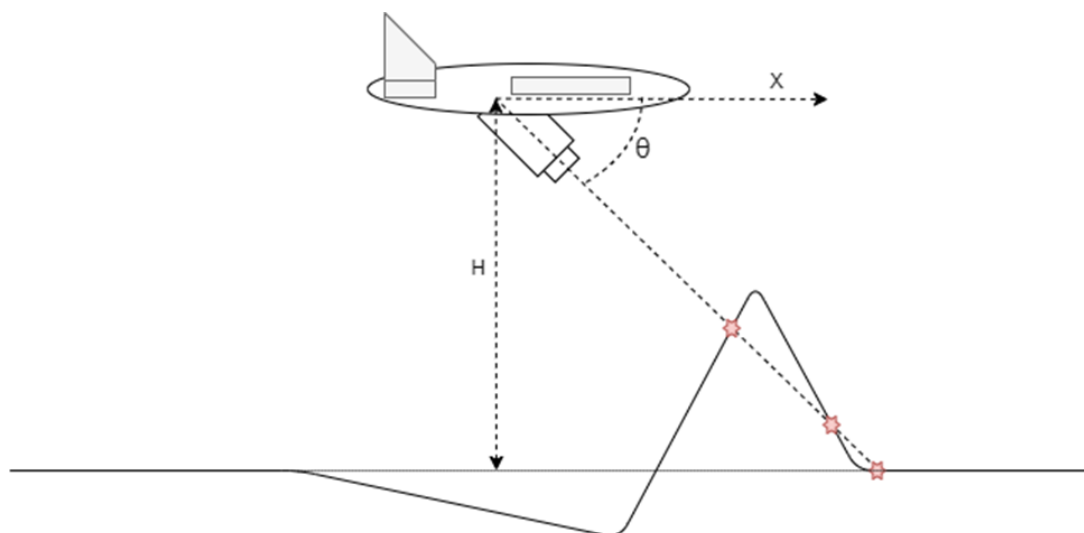


Рис. 2.4.6

Формат вывода

Координата X и координата h , разделенные пробелом. Значения указываются с точностью ровно два знака после запятой. В качестве разделителя используйте

точку.

Решение

Для определения участка пересечения оптической оси с рельефом в первом приближении воспользуемся графическим методом. Построим график карты высот и линии оптической оси. Так как угол отклонения камеры равен 45° , то оптическую ось можно построить как гипотенузу равнобедренного прямоугольного треугольника, то есть в виде отрезка с двумя точками, положением камеры (3, 30) и точки с нулевой высотой на расстоянии 30 м от положения камеры (33, 0). Полученный график приведен на рис. 2.4.7.

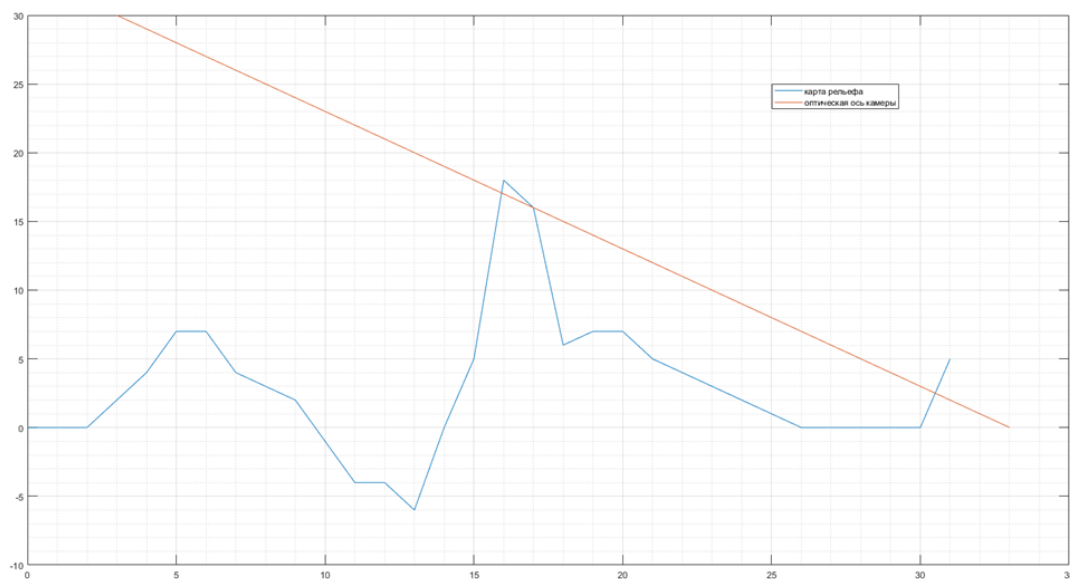


Рис. 2.4.7

Пересечение происходит между отметкам 15 м и 16 м, таким образом, правильным ответом является точка пересечения отрезка [(3, 30); (33, 0)] и отрезка [(15, 5); (16, 18)].

Для поиска точки пересечения воспользуемся уравнением прямой на плоскости:

$$\frac{x - x_a}{x_b - x_a} = \frac{y - y_a}{y_b - y_a}.$$

Прямые будут иметь следующие уравнения:

$$y = -x + 33,$$

$$y = 13x - 190.$$

Точка пересечения этих прямых определяется решением этих уравнений как системы.

Получаем ответ

$$X = 15 \frac{13}{14} \approx 15.92857;$$

$$h = 17 \frac{1}{14} \approx 17.07143.$$

Ответ: 15.93 17.07.

Задача 2.4.5. Тест по основным понятиям БАС (10 баллов)

Тема: основные понятия из области БАС.

1. Что такое угол крена?

- A. Угол между продольной осью летательного аппарата и направлением его движения относительно земной поверхности.
- B. Угол между продольной осью летательного аппарата и направлением вектора скорости набегающего на летательный аппарат воздушного потока.
- C. Угол, который возникает при повороте корпуса летательного аппарата вокруг его продольной оси.
- D. Угол между продольной осью летательного аппарата и горизонтальной плоскостью.

Ответ: С.

2. Что такое угол сноса?

- A. Угол между продольной осью летательного аппарата и направлением его движения относительно земной поверхности.
- B. Угол между продольной осью летательного аппарата и направлением вектора скорости набегающего на летательный аппарат воздушного потока.
- C. Угол, который возникает при повороте корпуса летательного аппарата вокруг его продольной оси.
- D. Угол между продольной осью летательного аппарата и горизонтальной плоскостью.

Ответ: А.

3. Что измеряет гироскоп?

- A. Угловую скорость вращения тела, на котором он установлен, относительно одной, двух или трех осей в зависимости от его типа.
- B. Интенсивность одной или нескольких составляющих магнитного поля, позволяет определять углы между собственными осями сенсора X , Y , Z и силовыми линиями магнитного поля Земли.
- C. Расстояние, время и определяет местоположение во всемирной системе координат WGS-84.
- D. Ускорение (перегрузка), возникающее на летательных аппаратах, ракетах, самолетах и других движущихся объектах.

Ответ: А.

4. Какое давление регистрируется с помощью трубки Пито?

- A. Статическое давление.
- B. Динамическое давление набегающего потока воздуха.
- C. Полное давление набегающего потока воздуха.
- D. Атмосферное давление.

Ответ: С.

5. Где на летательном аппарате самолетного типа нормальной схемы находится руль направления?
- A. На крыле.
 - B. На киле.
 - C. На стабилизаторе высоты.
 - D. На фюзеляже.

Ответ: B.

6. Для чего предназначены винглеты на крыльях летательного аппарата самолетного типа?
- A. Для красоты.
 - B. Для подвески дополнительных топливных баков.
 - C. Для увеличения эффективного размаха крыла, снижения индуктивного сопротивления и увеличения подъемной силы на концах крыла.
 - D. Для увеличения скорости.

Ответ: C.

7. Какое устройство из приведенного ниже списка не относится к элементам механизации крыла летательного аппарата самолетного типа (не обязательно присутствие всех возможных элементов механизации крыла на конкретном самолете)?
- A. Спойлеры.
 - B. Предкрылки.
 - C. Надкрылья.
 - D. Закрылки.

Ответ: C.

8. Какая сила направлена противоположно вектору скорости полета летательного аппарата?
- A. Сила тяги.
 - B. Подъемная сила.
 - C. Вес.
 - D. Сила лобового сопротивления.

Ответ: D.

9. Какая высота полета летательного аппарата измеряется от его поверхности до точки на земной поверхности, над которой в данный момент он пролетает?
- A. Абсолютная.
 - B. Относительная.
 - C. Истинная.
 - D. Приведенная.

Ответ: C.

10. Центр давления крыла летательного аппарата самолетного типа...
- A. всегда совпадает с центром масс летательного аппарата;
 - B. находится в условной точке приложения равнодействующей аэродинамических сил, действующих в полете на летательный аппарат;

- C. всегда находится в геометрическом центре летательного аппарата;
- D. всегда совпадает с аэродинамическим фокусом крыла летательного аппарата.

Ответ: B.

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

На втором отборочном этапе участники решают задачи по определению траектории полета, определению положения обнаруженного объекта, а также по обработке изображений. Задачи носят междисциплинарный характер и в упрощенной форме воссоздают элементы решения комплексной инженерной задачи заключительного этапа.

3.2.1. Командные задачи

Командные задачи второго этапа инженерного тура открыты для решения. Соревнование доступно на платформе Яндекс.Контест: <https://contest.yandex.ru/contest/69913/enter/>.

Задача 3.2.1.1. Построение оптимальной траектории по заданным промежуточным пунктам маршрута (25 баллов)

Темы: программирование, математика, геометрия, навигация, планирование путей полета.

Условие

Современные системы автоматического управления беспилотных летательных аппаратов (БЛА) самолетного типа преимущественно имеют следующие контуры управления:

- **управление низкого уровня** — выход на необходимые значения угловой ориентации (тангаж, крен, курс) и горизонтальной скорости полета БЛА;
- **управление высокого уровня** — формирование опорной траектории по полетному заданию, выданному оператором БЛА.

В рамках данной задачи необходимо разработать программу на языке программирования Python, которая по входному полетному заданию (ПЗ) — набору промежуточных пунктов маршрута (ППМ) в стартовой системе координат — рассчитывает оптимальную опорную траекторию по критерию кратчайшей траектории полета БЛА с учетом условий прохождения ППМ.

Считается, что БЛА передвигается на постоянной высоте $H = const$ с постоянной крейсерской скоростью $V_g = 120$ км/ч, ограничение на максимальный угол крена $-60^\circ \leq \gamma \leq 60^\circ$, ускорение свободного падения $g = 9,8$ м/с², значение минимально допустимого радиуса разворота можно определить по формуле:

$$R = \frac{V_g^2}{g \cdot \operatorname{tg}(\gamma)}.$$

Начальное положение БЛА — $A(x_0, z_0) = (0, 0)$ м.

Входные данные на задачу ограничены тремя точками, СТ — точка старта, ППМ — промежуточный пункт маршрута с дополнительным критерием прохождения и КПМ — конечный пункт маршрута.

Критерием прохождения ППМ (точка 2) является попадание полного диаметра полезной нагрузки БЛА $d_{obs} = 2r_{obs}$ в радиус ППМ r_n . Рекомендуется использовать методы пролета точки с радиусом упреждения (Fillet Path).

Критерием прохождения КПМ (точка 3) является момент пересечения координат КПМ с опорной траекторией БЛА.

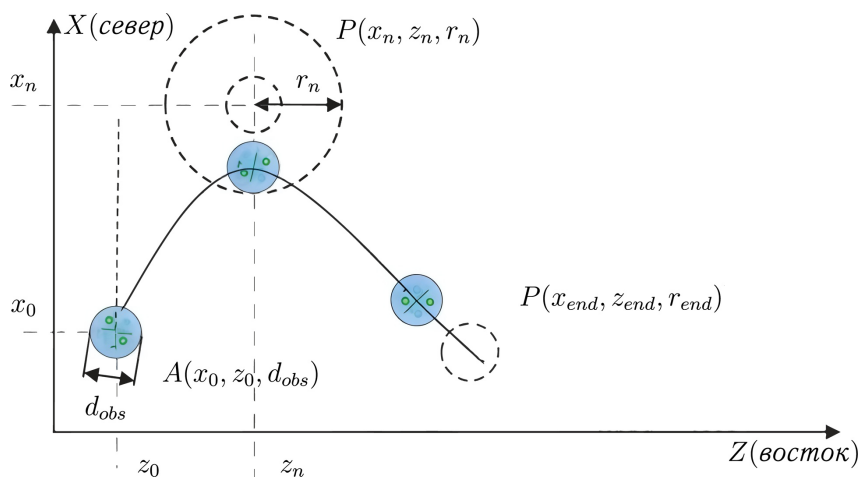


Рис. 3.2.1. Внешний вид траектории БЛА

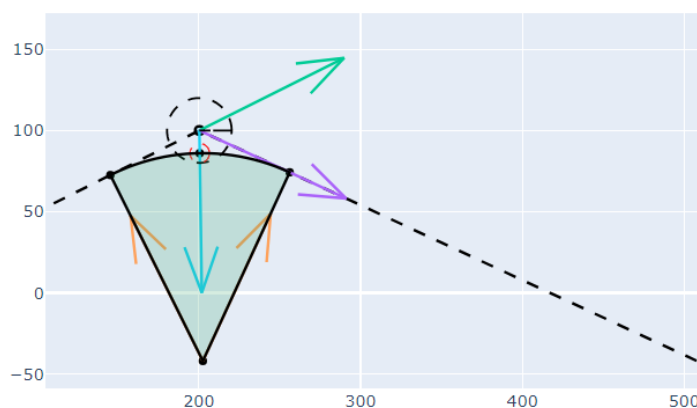


Рис. 3.2.2. Визуализация прохождения ППМ по радиусу упреждения

Формат входных данных

На вход программы поступает файл в формате CSV:

```

1 Z, X, r
2 0.0, 0.0, 9
3 1323.310446804855, 741.2170326700434, 35
4 3751.0326937190257, -425.39693527203053, 0

```

Данный файл можно легко прочитать, используя библиотеку `Pandas`, при помощи функции `read_csv`. На выходе данной функции вас будет ожидать `DataFrame` следующего вида, см. рис. 3.2.3.

	Z	X	r
0	0.000000	0.000000	9
1	1323.310447	741.217033	35
2	3751.032694	-425.396935	0

Рис. 3.2.3. Визуализация `DataFrame`

Нулевым индексом является исходное положение БЛА (СТ), в столбцах Z и X приведены соответствующие координаты (всегда равны нулю), в столбце с наименованием r приведен радиус действия полезной нагрузки $d_{obs} = 2r_{obs}$, значение r_{obs} отличается в каждом новом тесте.

Первым индексом является ППМ с условием прохождения, в столбцах Z и X приведены соответствующие координаты, значения которых отличаются в каждом новом тесте, в столбце с наименованием r приведен радиус зоны интереса ППМ r_{ppm} , значение которого отличается в каждом новом тесте.

Вторым индексом является КПМ, в столбцах Z и X приведены соответствующие координаты, значения которых отличаются в каждом новом тесте, значение столбца с наименованием r на данном индексе всегда равно нулю ввиду условия задачи.

Формат выходных данных

На стандартный вывод программы требуется вывести одно число: суммарная длина полученной опорной траектории в метрах.

Пример:

```
1 4193.68
```

Значение длины полученной опорной траектории всегда выводится с точностью до второго знака после запятой (даже если это нули).

Решение

Краткий алгоритм решения задачи:

1. прочитать входные данные из файла;
2. определить минимальный радиус разворота;
3. определить угол между линиями пути треугольника точек;
4. найти биссектрису угла между линиями пути;
5. найти точку, при прохождении которой выполняются условия перемещения радиуса интереса ППМ с полным диаметром действия полезной нагрузки;
6. найти радиус дуги и центр окружности, формирующей эту дугу, для построения траектории при пересечении этой точки;

7. найти точки перехода с прямолинейной траектории на дугу окружности;
8. найти длины всех участков траектории и сложить их, чтобы получить длину всей траектории.

Представленный алгоритм можно реализовать с помощью следующего программного кода на языке Python:

Python

```

1  import numpy as np
2  import pandas as pd
3
4  from math import cos, sin, hypot, tan, hypot, pi, pow
5  import math
6
7  # Матрица пересчета на ось Z
8  def rotz(theta=None):
9      # Rotation around z
10     R = np.array([ [cos(theta),    -sin(theta),    0],
11                   [sin(theta),    cos(theta),    0],
12                   [0,             0,             1] ])
13     return R
14
15 # Угол между векторами через библиотеку numpy
16 def angle_between_vectors_np(u, v):
17     u = np.array(u)
18     v = np.array(v)
19     cos_theta = np.dot(u, v) / (np.linalg.norm(u) * np.linalg.norm(v))
20     angle_rad = np.arccos(np.clip(cos_theta, -1.0, 1.0))
21     angle_deg = np.degrees(angle_rad)
22     return angle_rad, angle_deg
23
24 def solve(filename):
25     # минимальный радиус разворота
26     Vg = 120/3.6
27     gamma_max = math.radians(60)
28     g = 9.8
29     R_min = math.pow(Vg, 2) / (g * math.tan(gamma_max))
30     # Загрузка ПЗ
31     df_input = pd.read_csv(filename)
32     # Объявим точки как массив
33     w_0 = np.array([df_input["Z"][0], df_input["X"][0], 0]).T
34     w_1 = np.array([df_input["Z"][1], df_input["X"][1], 0]).T
35     w_2 = np.array([df_input["Z"][2], df_input["X"][2], 0]).T
36     # Загрузим радиус действия полезной нагрузки и радиус интереса
37     ↪ ППМ
38
39     # Радиус обзора полезной нагрузки
40     R_obs = df_input["r"][0]
41     # Радиус интереса ППМ
42     R_wp = df_input["r"][1]
43
44     # Найдем единичные вектора между точками
45     q_s = (w_1 - w_0) / np.linalg.norm(w_1 - w_0, ord=2)
46     q_n = (w_2 - w_1) / np.linalg.norm(w_2 - w_1, ord=2)
47
48     # Найдем угол между линиями пути треугольника точек
49     theta, theta_degrees = angle_between_vectors_np(-q_s.T, q_n)
50

```

```

51     # Найдем вектор отражающий биссектрису найденного угла
52     q_R = q_n @ rotz(theta/2)
53
54     # Найдем точку при прохождении которой, выполняются условия
55     ↪ прохождения радиуса интереса ППМ с полным диаметром действия
56     ↪ полезной нагрузки
57
58     # point_org_global = w_1 - np.dot(-R_wp+R_obs, q_R)
59
60     # Найдем необходимый радиус дуги окружности для построения
61     ↪ траектории при пересечении этой точки
62
63     R_calc = (abs(-R_wp+R_obs))/(1/sin(theta/2) - 1)
64
65     R = R_calc if (R_calc > R_min) else R_min
66
67     # Найдем центр окружности формирующей дугу
68
69     center_circle = w_1 - (R/sin(theta/2))*(q_s - q_n) /
70     ↪ np.linalg.norm(q_s - q_n, ord=2)
71
72     # Найдем точки перехода с прямолинейной траектории на дугу
73     ↪ окружности
74
75     r1 = w_1 - np.dot(R/tan(theta/2), q_s)
76     r2 = w_1 + np.dot(R/tan(theta/2), q_n)
77
78     # Найдем угол сектора, чтобы рассчитать длину дуги
79
80     q_r1 = (r1 - center_circle) / np.linalg.norm(r1
81     ↪ -center_circle, ord=2)
82     q_r2 = (r2 - center_circle) / np.linalg.norm(r2 -
83     ↪ center_circle, ord=2)
84
85     theta_R, theta_R_degrees = angle_between_vectors_np(q_r1, q_r2)
86
87     # Найдем длину первого прямолинейного участка до точки r1
88     L_0 = hypot(w_0[0]-r1[0], w_0[1]-r1[1])
89     # Найдем длину дуги окружности
90     L_R = theta_R * R
91     # Найдем длину второго участка траектории после точки r2
92     L_2 = hypot(r2[0]-w_2[0], r2[1]-w_2[1])
93
94     # Найдем общую длину траектории
95     L = L_0 + L_R + L_2
96     # Выведем длину участка, с точностью до 2 знака после запятой
97
98     print("%.2f" % L)
99
100 solve("input.txt")

```

Критерии оценивания

К задаче разработано пять автоматических тестов. За прохождение каждого из этих тестов решение участников набирает 5 баллов (максимум 25 баллов за задачу).

Ссылка на тестовые данные и материалы задачи: <https://disk.yandex.ru/d/tF438BK2KnLDCg/1>.

Задача 3.2.1.2. Распознавание объектов системой технического зрения (30 баллов)

Темы: программирование, компьютерное зрение, обработка изображений, геометрия.

Условие

Задача поиска объектов является одной из основных задач, решаемых бортовой системой технического зрения.

При решении этой задачи требуется не только обрабатывать полученные изображения и распознавать на них искомые объекты, но и определять их координаты на основе информации о текущем местоположении БЛА, дальности и угломерной информации о визируемом объекте. Также может решаться обратная задача. При заранее известных координатах визируемых объектов определяются параметры движения самого БЛА. Для этого могут использоваться либо заведомо известные параметры искомого объекта, либо дополнительные датчики (например, лазерный дальномер).

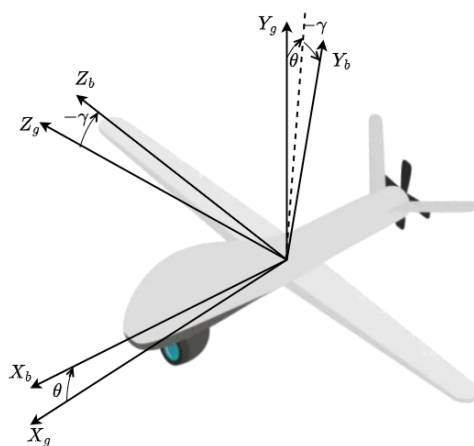


Рис. 3.2.4

Предположим, что БЛА совершает автоматический полет и его бортовая программа должна, на основе измерения системы технического зрения, определять количество искомого ориентиров в кадре. Кроме того, БЛА оснащен лазерным дальномером, оси которого жестко связаны со строительными осями БЛА. Измерение дальности происходит по оси Y_b в обратном направлении. Кватернион ориентации БЛА относительно нормальной земной системы координат известен.

Математически кватернион $q(w, x, y, z)$ можно записать так:

$$q = w + xi + yj + zk,$$

где w, x, y, z — вещественные числа; i, j, k — мнимые единицы.

Параметры x, y, z характеризуют направление оси вращения (проекции орта оси вращения на оси i, j, k). Параметр w характеризует величину угла поворота.

Таким образом, кватернион описывает ориентацию объекта как кратчайший поворот вокруг произвольно направленной оси (в то время как при использовании

углов Эйлера вращение всегда происходит вокруг одних и тех же осей). Кватернион обладает следующими свойствами:

- модуль кватерниона всегда должен быть равен единице;
- кватернион может быть использован как для представления ориентации объекта, так и для представления изменения ориентации (вращения);
- кватернион вращения всегда описывает кратчайший поворот.

Напишите программу, обрабатывающую входное изображение, определяющую количество контрастных ориентиров на этом изображении и реализующую расчет высоты полета БЛА по измерениям лазерного дальномера с учетом кватерниона ориентации БЛА.

Формат входных данных

На вход программы поступает файл в формате PGM (<https://netpbm.sourceforge.net/doc/pgm.html>):

```
1 P2
2 # 130.5 0.966 0.0 0.0 0.259
3 320 240 220 63 63 63 63 63 63 63 63 63 63 63 63 63 63 63 63 63 63
  ↪ 63 63 63 63 129 63 63 115 63 63 63 63 63 63 63
```

Первое значение P2 — идентификатор формата `pgm`.

Вторая строка, отмеченная символом `#`, — это комментарий. Он не содержит данных, связанных с изображением, но содержит измерения лазерного дальномера и кватернион ориентации БЛА в момент получения изображения, записанные через пробел.

В рассматриваемом примере:

- значение 130.5 — расстояние в метрах, измеренное лазерным дальномером;
- 0.966 — значение W кватерниона ориентации;
- 0.0 0.0 0.259 — значения проекций вектора поворота на орты осей X_b , Y_b , Z_b соответственно.

Первые два числа третьей строки — разрешение изображения (количество пикселей по горизонтали и вертикали).

В примере изображение содержит 320 px (пикселей) по горизонтали и 240 px по вертикали.

Третье число третьей строки — максимальная яркость пикселя изображения в файле.

В примере максимальная яркость составляет 220 единиц.

Каждое следующее число — яркость 1 px (пикселя) изображения в интервале от 0 до указанного максимального значения (которое не может превышать 255). Яркость пикселей записывается в порядке слева направо и сверху вниз. В представленном примере первые 320 пикселей после значения максимальной яркости описывают первую (верхнюю) строку изображения.

Значение яркости 0 соответствует черному цвету на изображении. Значение 255 — белому цвету.

Формат выходных данных

На стандартный вывод программы требуется вывести два числа, записанные через пробел: высоту полета БЛА и число контрастных ориентиров в кадре.

Пример ответа:

```
1 90.57 8
```

Значение высоты всегда выводится с точностью до второго знака после запятой (даже если это нули). Количество ориентиров в кадре всегда целое.

Примечания

Ориентирами считаются «светлые» области кадра, значение яркости которых превышает некоторое пороговое значение.

Гарантируется, что на вход программы поступает изображение в формате `rgb`.

Гарантируется, что вторая строка файла — комментарий, содержащий измерение лидача (лазерного дальномера) и кватернион ориентации.

Решение

Краткий алгоритм решения задачи:

1. прочитать входные данные из файла;
2. найти углы Эйлера, используя заданный кватернион;
3. найти высоту полета БЛА, используя найденные углы Эйлера и заданное значение показаний дальномера;
4. выполнить размытие изображения;
5. подобрать значение пороговой яркости;
6. выполнить поиск и подсчет контрастных ориентиров.

Представленный алгоритм можно реализовать с помощью следующего программного кода:

Python

```
1 from math import sin, cos, sqrt, atan2, asin, pi
2 import sys
3 import copy
4
5 def normalize_quaternion(q):
6     w, x, y, z = q
7     norm = sqrt(w**2 + x**2 + y**2 + z**2)
8     if norm == 0:
9         raise ValueError("Cannot normalize a zero quaternion")
10    return (w / norm, x / norm, y / norm, z / norm)
11
12 def quaternion_to_euler(q):
13     # Normalize the quaternion
14     q = normalize_quaternion(q)
15     w, x, y, z = q
```

```

16     # Calculate roll (phi)
17     roll = atan2(2 * (y * z + w * x), 1 - 2 * (y**2 + x**2))
18     # Calculate pitch (theta)
19     pitch = asin(2 * (w * y - x * z))
20     # Calculate yaw (psi)
21     yaw = atan2(2 * (x * y + w * z), 1 - 2 * (z**2 + y**2))
22     return (roll, pitch, yaw)
23
24 def apply_simple_blur(image):
25     """Apply a simple blur to a 2D list representing an image."""
26     height = len(image)
27     width = len(image[0])
28     blurred_image = [[0] * width for _ in range(height)]
29
30     # Define the blur radius
31     radius = 1 # Since we want to blur within 3 pixels area (1 on
32     ↪ each side)
33
34     for i in range(height):
35         for j in range(width):
36             total = 0
37             count = 0
38
39             # Iterate over the neighborhood
40             for dx in range(-radius, radius + 1):
41                 for dy in range(-radius, radius + 1):
42                     nx, ny = i + dx, j + dy
43                     # Check bounds
44                     if 0 <= nx < height and 0 <= ny < width:
45                         total += image[nx][ny]
46                         count += 1
47
48             # Calculate the average and assign it to the blurred image
49             blurred_image[i][j] = total // count # Use integer
50             ↪ division
51
52     return blurred_image
53
54 def numIslands(grid, weight) -> int:
55     # weight = 0.8
56
57     def dfs(row, col):
58         if grid[row][col] < weight:
59             return 0
60         grid[row][col] = 0.0
61         if (col + 1) < len(grid[0]):
62             dfs(row, col+1)
63         if col > 0:
64             dfs(row, col-1)
65         if (row + 1) < len(grid):
66             dfs(row+1, col)
67         if row > 0:
68             dfs(row - 1, col)
69         return 1
70
71     count = 0
72     for i in range(len(grid)):
73         for j in range(len(grid[0])):
74             if grid[i][j] >= weight:
75                 count += 1
76                 dfs(i, j)

```

```

74         return count
75
76     def solve(filename):
77         f = open(filename, "r")
78         inp = f.read()
79         arr = inp.split(" ")
80         laser = float(arr[2])
81         quat = [float(x) for x in arr[3:7]]
82         quat_sum = sqrt(quat[0]**2 + quat[1]**2 + quat[2]**2 + quat[3]**2)
83         if not quat_sum == 1:
84             quat = [quat[0] / quat_sum, quat[1] / quat_sum, quat[2] /
85                    ↪ quat_sum, quat[3] / quat_sum]
86
87         width = int(arr[7])
88         height = int(arr[8])
89         max_brightness = int(arr[9])
90         pgm_image = [int(x) for x in arr[10:-1]]
91         rot = quaternion_to_euler([quat[0], quat[1], quat[3], quat[2]])
92         angles = rot
93         H = cos(angles[0]) * cos(angles[1]) * laser
94
95         norm_image = [float(x) for x in pgm_image]
96         norm_image_arr = []
97         for i in range(int(len(norm_image) / width)):
98             norm_image_arr.append(norm_image[i*(width):(i+1)*width])
99         norm_image_arr = apply_simple_blur(norm_image_arr)
100        sys.setrecursionlimit(85000)
101        sacrefise = copy.deepcopy(norm_image_arr)
102        sol = numIslands(sacrefise, 0.8 * max_brightness)
103        bright_pixels = [x for x in norm_image if x > 0.8*max_brightness]
104
105        print(f'H:.2f {sol}')
106
107    solve('input.txt')

```

Критерии оценивания

К задаче разработано 15 автоматических тестов. За прохождение каждого из этих тестов решение участников набирает 2 балла (максимум 30 баллов за задачу).

Ссылка на тестовые данные и материалы задачи: <https://disk.yandex.ru/d/tF438BK2KnLDCg/2>.

Задача 3.2.1.3. Определение положения обнаруженного объекта (45 баллов)

Темы: техническое зрение, локализация объекта.

Условие

При поиске различных объектов при помощи БЛА чаще всего используются алгоритмы распознавания объектов на изображении. Однако после обнаружения объекта на изображении необходимо определить его координаты в реальном мире. При решении такой задачи необходимо учитывать множество факторов, в частности, рельеф обследуемой местности.

Напишите программу для определения долготы и широты обнаруженного объекта, который лежит на оптической оси камеры БЛА, летящего горизонтально вдоль поверхности земли.

Для решения известны координаты БЛА, кватернион ориентации и карта рельефа местности. Координаты БЛА задаются в географической системе координат: широта, долгота, истинная высота. Считается, что координаты БЛА совпадают с координатами камеры. Кватернион описывает ориентацию осей камеры относительно осей нормальной земной системы координат. Направление осей нормальной земной системы координат вводится следующим образом:

- ось XX направлена на восток,
- ось YY направлена на север,
- ось ZZ направлена вертикально вверх.

Направление осей камеры отображено на рис. 3.2.5.

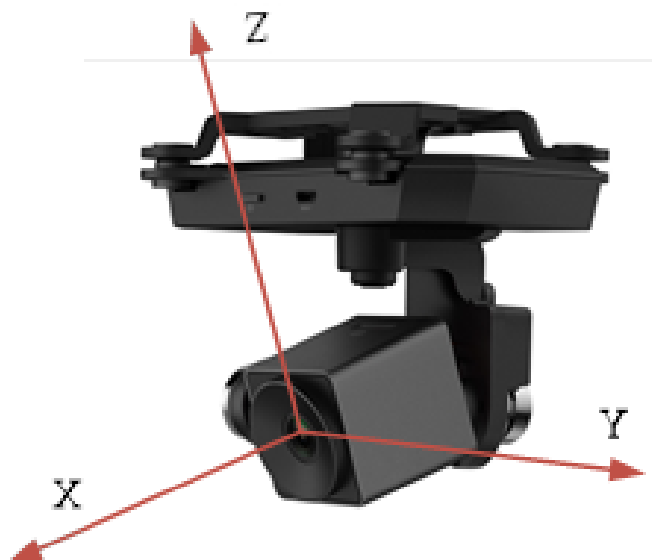


Рис. 3.2.5. Изображение направления осей камеры

Сам рельеф задан в виде высот над общеземным эллипсоидом, записанных в виде матрицы, где каждое значение определяет фиксированную высоту в квадрате 1×1 м. Для позиционирования матрицы задаются координаты центра квадрата ее верхнего левого угла. Каждый ряд матрицы — это набор высот в положительном направлении оси x , а каждый столбец — это набор высот в отрицательном направлении оси y . Базовым направлением оптической оси камеры считается направление строго на восток.

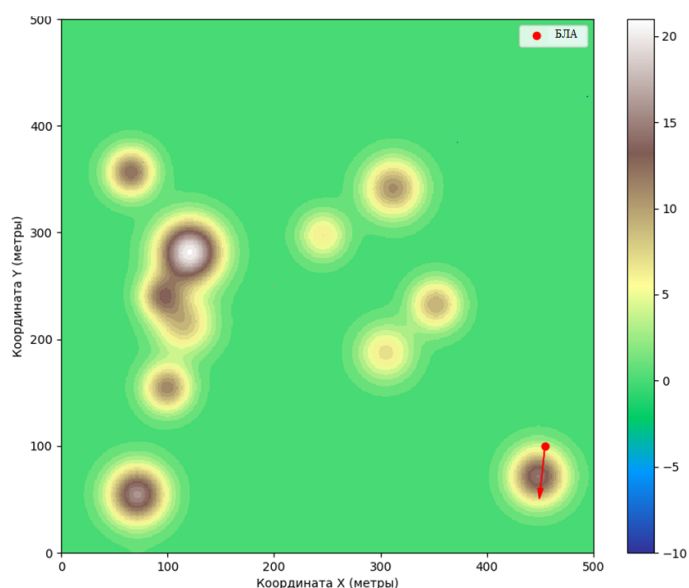


Рис. 3.2.6. Рельеф местности с положением БЛА и направлением камеры

В масштабах, рассматриваемых в задаче, местность считается плоской, то есть кривизну Земли учитывать не надо, однако изменения градуса долготы на разных широтах учитывать необходимо. Атмосферные искажения и другие внешние факторы, влияющие на распространение оптического сигнала, также учитывать не нужно. Считать, что одна угловая минута по широте равна одной морской миле. Угловое расстояние по долготе пересчитать с использованием сферической геометрии.

Теоретическую информацию по понятию «кватернион» можно получить из задачи 2.4.3 инженерного тура первого этапа «Кватернион ориентации». На рис. 3.2.6 представлена визуализация примера исходных данных — изображен рельеф местности (используются данные матрицы абсолютных высот 500×500) с положением БЛА (координаты БЛА) и направлением камеры (кватернион ориентации камеры).

Формат входных данных

В реальных данных задачи рельеф задается матрицей 500×500 .

Пример

Координаты БЛА (широта, долгота) и истинная высота:

```
1 55.811539 37.499727 8
```

Кватернион ориентации камеры в формате $w x y z$:

```
1 1 0 0 0
```

Координаты верхнего левого угла матрицы абсолютных высот:

```
1 55.811539 37.499727
```

Фрагмент матрицы абсолютных высот (500×500), задающей рельеф местности:

```

1 22 17 16 20 30
2 23 17 15 18 27
3 22 17 14 19 25
4 20 17 10 19 23
5 18 17 10 12 20

```

Формат выходных данных

На стандартный вывод программы требуется вывести два числа, записанные через пробел: долгота и широта обнаруженного объекта, лежащего на оптической оси камеры. В случае невозможности по любой причине определения координат обнаруженного объекта выведите координаты БЛА.

Пример ответа:

```

1 55.801453 37.471452

```

Значение всегда выводится с точностью до шестого знака после запятой (даже если это нули).

Решение

Краткий алгоритм решения задачи:

1. прочитать входные данные из файла;
2. преобразовать градусы в метры;
3. преобразовать координаты БЛА в его координаты в матрице рельефа;
4. вычислить направление камеры;
5. нормализовать вектор направления камеры;
6. найти координаты точки пересечения вектора с матрицей рельефа;
7. преобразовать координаты точки пересечения в географические координаты.

Представленный алгоритм можно реализовать с помощью следующего программного кода на языке Python:

Python

```

1 import numpy as np
2
3
4 def quaternion_rotate_vector(q, v):
5     # Поворот вектора v с помощью кватерниона q
6     qvec = np.array([q[0], q[1], q[2]])
7     uv = np.cross(qvec, v)
8     uuv = np.cross(qvec, uv)
9     uv *= (2.0 * q[3])
10    uuv *= 2.0
11    return v + uv + uuv
12
13 def get_terrain_height(terrain_matrix, x, y):
14     # Получение высоты рельефа в точке (x, y) без интерполяции
15     x0 = int(np.floor(x))
16     y0 = int(np.floor(y))

```

```

17     if x0 < 0 or x0 >= terrain_matrix.shape[1] or y0 < 0 or y0 >=
18         ↪ terrain_matrix.shape[0]:
19         return None
20     return terrain_matrix[y0, x0]
21
22 def find_intersection(uav_pos, direction, terrain_matrix):
23     t = 0.0
24     delta_t = 0.5 # Шаг в метрах
25     max_distance = 10000 # Максимальное расстояние поиска (в метрах)
26
27     uav_x, uav_y, uav_z = uav_pos
28     dx, dy, dz = direction
29     terrain_size = terrain_matrix.shape[0]
30
31     while t < max_distance:
32         x = uav_x + t * dx
33         y = uav_y + t * dy
34         z = uav_z + t * dz
35
36         if x < 0 or x >= terrain_size - 1 or y < 0 or y >=
37             ↪ terrain_size - 1:
38             # Луч вышел за пределы рельефа без пересечения
39             return None
40
41         terrain_height = get_terrain_height(terrain_matrix, x, y)
42         if terrain_height is None:
43             return None
44         if z <= terrain_height:
45             # Найдена точка пересечения
46             # Уточнение точки пересечения
47             # Возврат назад и использование меньших шагов
48             t_high = t
49             t_low = t - delta_t
50             for _ in range(100):
51                 t_mid = (t_low + t_high) / 2.0
52                 x_mid = uav_x + t_mid * dx
53                 y_mid = uav_y + t_mid * dy
54                 z_mid = uav_z + t_mid * dz
55                 terrain_height_mid =
56                 ↪ get_terrain_height(terrain_matrix, x_mid, y_mid)
57                 if terrain_height_mid is None:
58                     return None
59                 if z_mid <= terrain_height_mid:
60                     t_high = t_mid
61                 else:
62                     t_low = t_mid
63             t = t_high
64             x = uav_x + t * dx
65             y = uav_y + t * dy
66             z = uav_z + t * dz
67             return x, y, z # Точка пересечения в координатах рельефа
68
69     t += delta_t
70
71     # Пересечение не найдено в пределах максимального расстояния
72     return None
73
74 def read_and_compute(filename):
75     with open(filename, 'r') as f:
76         lines = f.readlines()

```

```

74
75 # Чтение координат БЛА и высоты
76 uav_line = lines[0].strip()
77 uav_latitude, uav_longitude, uav_altitude = map(float,
78 ↪ uav_line.split())
79 if uav_altitude < 0:
80     print(f"{uav_latitude:.6f} {uav_longitude:.6f}")
81     return
82
83 # Чтение кватерниона
84 quat_line = lines[1].strip()
85 qw, qx, qy, qz = map(float, quat_line.split())
86
87 # Чтение начальных координат рельефа
88 terrain_line = lines[2].strip()
89 terrain_latitude, terrain_longitude = map(float,
90 ↪ terrain_line.split())
91
92 # Чтение матрицы рельефа
93 terrain_data = lines[3:]
94 terrain_matrix = np.array([list(map(int, line.strip().split()))
95 ↪ for line in terrain_data])
96 terrain_matrix = np.flipud(terrain_matrix)
97
98 # Конвертация градусов в метры
99 latitude_rad = np.radians(terrain_latitude)
100 meters_per_degree_latitude = 111120 # Среднее количество метров
101 ↪ на градус широты
102 meters_per_degree_longitude = 111120 * np.cos(latitude_rad) #
103 ↪ Метров на градус долготы зависит от широты
104
105 # Преобразование координат БЛА в координаты матрицы рельефа
106 delta_lat_uav = uav_latitude - (terrain_latitude -
107 ↪ terrain_matrix.shape[0]/meters_per_degree_latitude)
108 delta_lon_uav = uav_longitude - terrain_longitude
109
110 delta_x_meters = delta_lon_uav * meters_per_degree_longitude
111 delta_y_meters = delta_lat_uav * meters_per_degree_latitude
112
113 # Координаты БЛА в матрице рельефа
114 uav_x = delta_x_meters
115 uav_y = delta_y_meters
116
117 terrain_height_under_uav = get_terrain_height(terrain_matrix,
118 ↪ uav_x, uav_y)
119
120 if terrain_height_under_uav is None:
121     print(f"{uav_latitude:.6f} {uav_longitude:.6f}")
122     return
123
124 uav_z = uav_altitude + terrain_height_under_uav
125
126 # Вычисление направления камеры
127 q = np.array([qx, qy, qz, qw])
128 camera_forward = np.array([1, 0, 0]) # Ось X в системе координат
129 ↪ камеры (направление оптической оси)
130 direction_vector = quaternion_rotate_vector(q, camera_forward)
131
132 # Нормализация вектора направления

```

```
126 norm_dir = np.linalg.norm(direction_vector)
127 if norm_dir == 0:
128     print(f"{uav_latitude:.6f} {uav_longitude:.6f}")
129     return
130 direction = direction_vector / norm_dir
131
132 # Поиск точки пересечения
133 intersection = find_intersection((uav_x, uav_y, uav_z), direction,
134     ↪ terrain_matrix)
135
136 if intersection is None:
137     print(f"{uav_latitude:.6f} {uav_longitude:.6f}")
138     return
139
140 intersect_x, intersect_y, intersect_z = intersection
141
142 # Преобразование координат точки пересечения обратно в
143     ↪ географические координаты
144 delta_lat_intersect = intersect_y / meters_per_degree_latitude
145 delta_lon_intersect = intersect_x / meters_per_degree_longitude
146
147 object_latitude = (terrain_latitude -
148     ↪ terrain_matrix.shape[0]/meters_per_degree_latitude) +
149     ↪ delta_lat_intersect
150 object_longitude = terrain_longitude + delta_lon_intersect
151
152 print(f"{object_latitude:.6f} {object_longitude:.6f}")
153
154 read_and_compute('input.txt')
```

Критерии оценивания

К задаче разработано девять автоматических тестов. За прохождение каждого из этих тестов решение участников набирает 5 баллов (максимум 45 баллов за задачу).

Ссылка на тестовые данные и материалы задачи: <https://disk.yandex.ru/d/tF438BK2KnLDCg/3>.

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/72662/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, ($1 \leq n, m \leq 10$).

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
1 n, m = map(int, input().split())
2 for i in range(n):
3     if i % 2 == 0:
4         print(*([1,2]*5)[:m])
5     else:
6         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 $ks \leq x < (k+1)s$. 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.

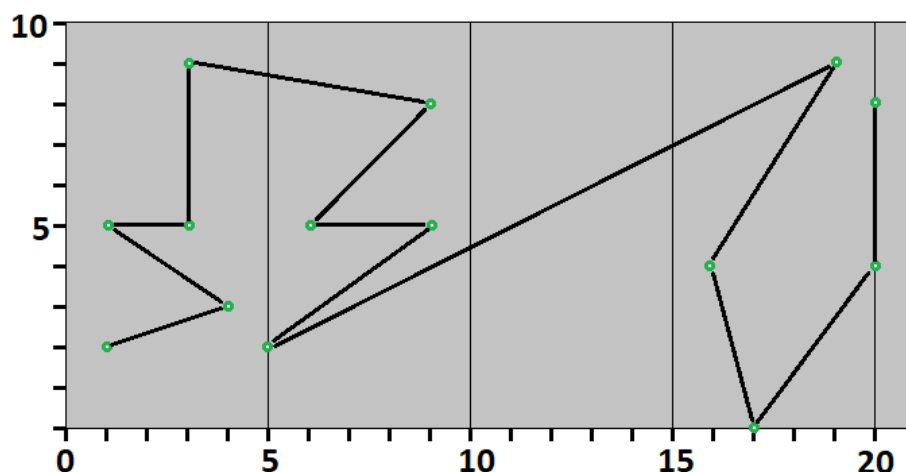


Рис. 4.2.1

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 ($1 \leq w, h, s \leq 10\,000$).

Во второй строке на вход программы поступает одно натуральное число n — количество деревьев в парке, $2 \leq n \leq 200\,000$.

Далее в n строках записаны по два числа x_i и y_i — координаты на плоскости каждого из n деревьев, $0 \leq x_i < w$, $0 \leq y_i < h$.

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 элементу, 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

```

1 w, h, s = map(int, input().split())
2 n = int(input())
3 p = sorted([tuple(map(int, input().split())) for _ in range(n)], \
4             key = lambda p: (p[0]//s, p[1]*(-1 if (p[0]//s)%2==1 else 1), p[0]))
5 ans = 0.
6 for i in range(1, len(p)):
7     ans += ((p[i][0]-p[i-1][0])**2 + (p[i][1]-p[i-1][1])**2)**0.5
8 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 $w/2 \leq s < w$, 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 $(1, 0)$, $(2, 0)$, \dots , $(n, 0)$ в воздухе на некоторой высоте неподвижно висит n дронов. Даны n целых чисел t_1, t_2, \dots, t_n , задающих некоторые моменты времени. Дрон, изначально находящийся в точке $(i, 0)$, стартует в момент времени t_i и движется 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 $(0, 1)$, $(0, 2)$, \dots , $(0, m)$. Для них также заданы моменты времени q_1, q_2, \dots, q_m . Дрон, изначально находящийся в точке с координатами $(0, j)$, стартует в момент времени q_j и движется 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.

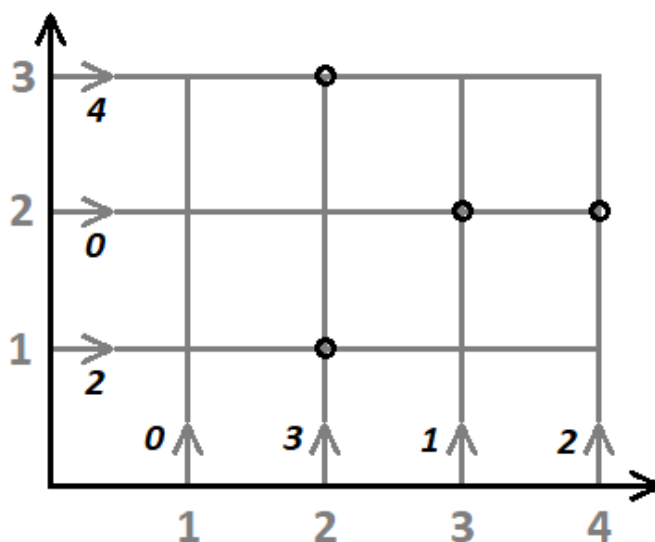


Рис. 4.2.2

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, $1 \leq n, m \leq 100\,000$.

Во второй и в третьей строках на вход поступают целые числа t_1, \dots, t_n и q_1, \dots, q_m — моменты начала движения для дронов, ожидающих на оси Ox и на оси Oy соответственно, $0 \leq t_i, q_i \leq 10^9$.

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

```

1 n, m = map(int, input().split())
2 cnt = dict()
3 for i, t in enumerate(map(int, input().split())):
4     if not t-i in cnt.keys():
5         cnt[t-i]=0

```

```

6     cnt[t-i]+=1
7     ans = 0
8     for j,q in enumerate(map(int,input().split())):
9         if q-j in cnt.keys():
10            ans+=cnt[q-j]
11     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 2000.

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.

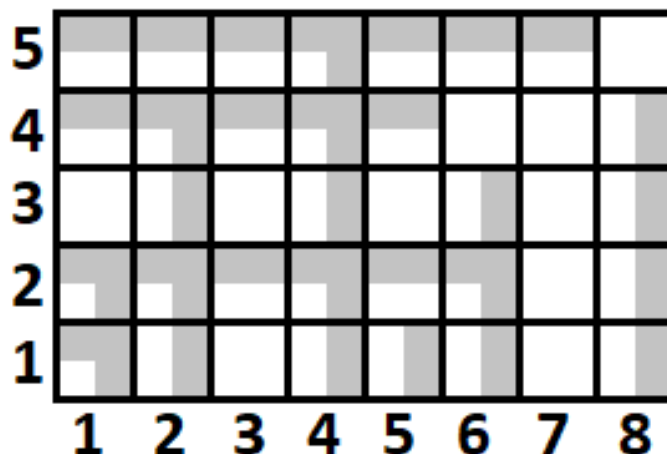


Рис. 4.2.3

Формат входных данных

В первой строке на вход поступают два натуральных числа n и m — размеры поля, $1 \leq n, m \leq 200\,000$.

Во второй строке записаны числа a_1, \dots, a_n — количество обработанных клеток в каждом из горизонтальных рядов, $0 \leq a_i \leq m$.

В третьей строке записаны числа b_1, \dots, b_m — количество обработанных клеток в каждом из вертикальных рядов, $0 \leq b_i \leq n$.

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}, \dots, 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}, \dots, 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

```

1  n, m = map(int, input().split())
2  a = [int(t) for t in input().split()]
3  blst = sorted(list(enumerate(map(int, input().split()))),
4                key = lambda x: -x[1])
5  tree = [0] * (2 * (m + 1))
6  i = n
7  ans = 0
8  for b in blst:
9      while i > b[1]:
10         i -= 1
11         pos = a[i] + m + 1
12         while pos > 0:
13             tree[pos] += 1
14             pos //= 2
15         posr = m + 1 + b[0] + 1
16         posl = m + 1
17         while posr > posl:
18             if posr % 2 == 1:
19                 posr -= 1
20             ans += tree[posr]
```

```

21     if posl%2==1:
22         ans += tree[posl]
23         posl+=1
24     posr//=2
25     posl//=2
26     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 $1 \leq n \leq 200\,000$, $1 \leq m \leq 10\,000$.

В тестах №№ 15–21 значения a_1, \dots, 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 $f_1(x_1, \dots, x_n), \dots, f_m(x_1, \dots, x_n)$ 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 $f_i(x_1, \dots, x_n)$ всегда зависят от n переменных, записанных только в указанном порядке x_1, \dots, 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, \dots, f_m и g_1, \dots, 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, $1 \leq n \leq 10$.

Во второй строке записано одно натуральное число m — количество функций f_1, \dots, f_m , используемых при построении формул, $1 \leq m \leq 2000$.

Далее в m строках записаны векторы, задающие функции f_1, \dots, 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, \dots, g_k , для которых требуется найти формулу, $1 \leq k \leq 2000$.

Далее в k строках записаны векторы, задающие функции g_1, \dots, 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 $K_1 \vee K_2 \vee \dots \vee K_s$, где $K_j = f_{i_1} \wedge f_{i_2} \wedge \dots \wedge f_{i_r}$. 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 $g_j = (0 \dots 0)$.

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

```

1 from functools import reduce
2 k = 2**int(input())
3 n = int(input())
4 fbits = [input() for _ in range(n)]
5 fint = [int(t,2) for t in fbits]
6 fw = [0]*k
7 fwbits = ['']*k
8 for i in range(k):
9     b = 0
10    for j in range(n):
11        if fbits[j][i]=='1':
12            if b==0:
13                b = fint[j]
14            else:
15                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 $m \leq 50$, $k \leq 100$.

В тестах №№ 14–21 количество переменных не превосходит 8. При этом $m \leq 500$, $k \leq 500$.

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 $g = 9,8 \text{ м/с}^2$. 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 баллов
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 $I_1 = 4$ мА, а при приложении очень больших (много больше U_0) напряжений нелинейный элемент ведет себя как резистор с сопротивлением $R_1 = 3$ кОм. Найдите U_0 .

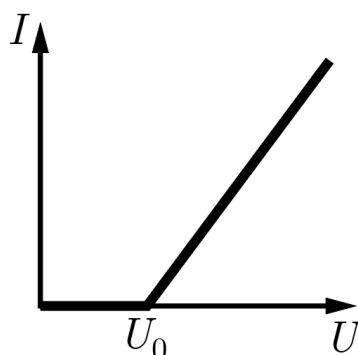


Рис. 4.2.4

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 $p_1 = 12$ 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 ρgV , 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

$$\begin{cases} \rho_1 gV = mg, \\ \rho_2 gV = 2mg. \end{cases}$$

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:

$$\begin{cases} p_1 = \frac{\rho_1 h}{2} g, \\ p_2 = \left(\rho_1 h + \frac{\rho_2 2h}{2} \right) g = (\rho_1 + \rho_2) g h. \end{cases}$$

Подставляя в последнее уравнение полученное выражение для ρ_2 , приведем его к виду

$$p_2 = 3\rho_1 g h = 3p_1 = 36 \text{ кПа}.$$

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. Проектор (20 баллов)

Условие

Поисковый дрон снабжен прожектором, состоящим из мощного точечного источника света и плоской линзы с оптической силой $D = 2$ 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 $H = 15$ 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

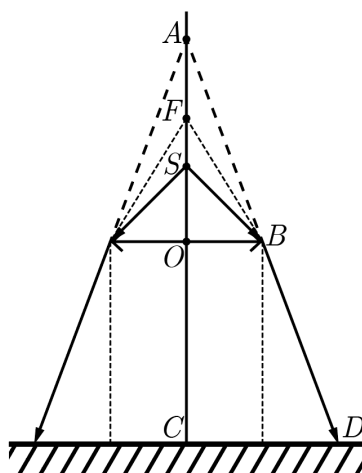


Рис. 4.2.5

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 лучей, 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 балла
Верно записана формула тонкой линзы	4 балла
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 $\eta(v) = \frac{3v(2v_0 - v)}{4v_0^2}$, 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:

$$\frac{I_f}{I_0} = \frac{v_f F_f}{\eta_f U} \cdot \frac{\eta_{max} U}{v_0 F_0} = \frac{v_f}{v_0} \cdot \frac{F_f}{F_0} \cdot \frac{3}{4} \cdot \frac{16}{9},$$

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 баллов
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).

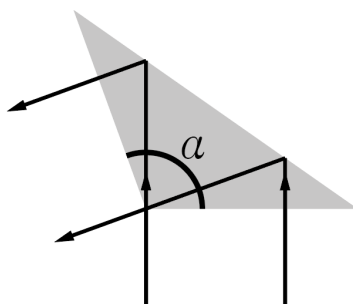


Рис. 4.2.6

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

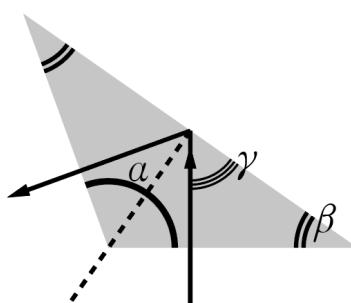


Рис. 4.2.7

Изобразим ход одного луча в призме (рис. 4.2.7). В силу симметрии, лучи выйдут 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 меньше 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

$$S_2 = \frac{at_2^2}{4}; \quad v_2 = \frac{at_2}{2},$$

$$S'_2 = v_2 t_2 + \frac{at_2^2}{2} = at_2^2,$$

$$S = S_2 + S'_2 = \frac{5}{4}at_2^2.$$

Приравнивая эти совокупные пути, получим

$$\frac{7at_1^2}{4} = \frac{5at_2^2}{4} \Rightarrow \frac{t_2}{t_1} = \sqrt{\frac{5}{7}} \approx 1,18.$$

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)

$$A = (m - 1)(n - 1)p_1V_1.$$

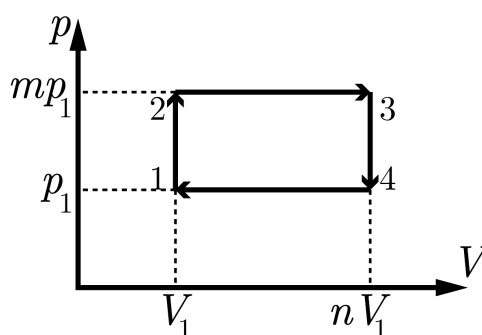


Рис. 4.2.8

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:

$$Q_{23} = C_V \nu (T_3 - T_2) + m p_1 (n - 1) V_1,$$

где T_3 — абсолютная температура газа в точке T_3 .

Таким образом, КПД цикла определяется соотношением

$$\eta = \frac{A}{Q_{12} + Q_{23}} = \frac{(m - 1)(n - 1)p_1 V_1}{C_V \nu (T_3 - T_1) + m(n - 1)p_1 V_1}.$$

Поскольку рабочее тело ведет себя как идеальный газ, к нему применимо уравнение Менделеева – Клапейрона, из которого следует

$$\nu T_1 = \frac{p_1 V_1}{R}; \quad \nu T_3 = \frac{m n p_1 V_1}{R}.$$

С учетом этих соотношений выражение для КПД принимает вид

$$\eta = \frac{(m - 1)(n - 1)p_1 V_1}{C_V / R (m n - 1)p_1 V_1 + m(n - 1)p_1 V_1},$$

откуда окончательно выразим C_V :

$$\begin{aligned} C_V &= \frac{(m - 1)(n - 1)}{\eta(mn - 1)} R - \frac{m(n - 1)}{mn - 1} R = \frac{(n - 1)(m - m\eta - 1)}{\eta(mn - 1)} R = \\ &= \frac{12}{5} R \approx 20 \text{ Дж}/(\text{моль} \cdot \text{К}). \end{aligned}$$

Ответ:

$$C_V = \frac{(n - 1)(m - m\eta - 1)}{\eta(mn - 1)} R = \frac{12}{5} R \approx 20 \text{ Дж}/(\text{моль} \cdot \text{К}).$$

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 $\mathcal{E}_0 = 240$ В. Но очутившись с презентацией на технической выставке, команда разработчиков обнаружила, что вместо заявленного, организаторы предоставили им источник тока с ЭДС $\mathcal{E} = 600$ В, а вдобавок — имеющий не столь малое внутреннее сопротивление $r = 2$ 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.

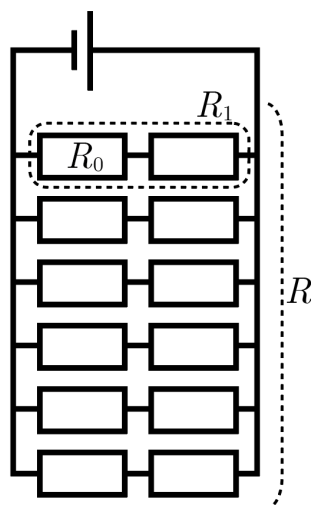


Рис. 4.2.9

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:

$$I_0 = \frac{\mathcal{E}_0}{R_0} = \frac{\mathcal{E} - 2\mathcal{E}_0}{6r} = 10 \text{ A}.$$

Ответ:

$$I_0 = \frac{\mathcal{E} - 2\mathcal{E}_0}{6r} = 10 \text{ A}.$$

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 работ двигателей до поломки, будем помечать индексом 0. Введем систему координат так, как это изображено на рис. 4.2.10: ось Ox проведем через вершины F и C , а Oy — перпендикулярно ей. Начало системы координат поместим в центр масс мультикоптера. Сторону шестиугольника обозначим a .

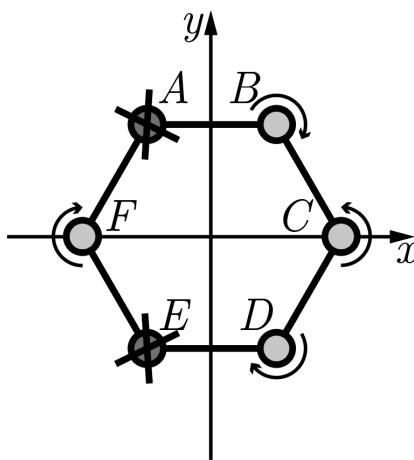


Рис. 4.2.10

Для того чтобы коптер остался в равновесии, должны выполняться 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 Ox должен остаться равен нулю. Относительно этой оси плечи подъемных сил в вершинах F и C , а также плечи 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 Oy 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

$$F_F a = F_C a + (F_B + F_D) \frac{a}{2}.$$

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):

$$\begin{cases} \omega_B + \omega_C + \omega_D + \omega_F = 6\omega_0, \\ \omega_B = \omega_D, \\ \omega_F = \omega_C + \frac{\omega_B + \omega_D}{2}, \\ \omega_C = \omega_B + \omega_D + \omega_F. \end{cases}$$

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 Ox)	5 баллов
Верно составлено третье условие равновесия (сумма моментов относительно Oy)	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

Ссылка на папку с дополнительными материалами: <https://disk.yandex.ru/d/kT-A8SCPpVZs-w>.

4.3.1. Общая информация

Участникам инженерного тура предлагается осуществить мониторинг зоны чрезвычайной ситуации (ЧС) — наводнения — в автоматическом режиме с помощью беспилотного воздушного судна (БВС) самолетного типа, а также доставить груз в места с затрудненным доступом наземных транспортных средств.

4.3.2. Легенда задачи

Моделируется возникновение чрезвычайной ситуации (наводнения). На затопленных территориях могут находиться люди, которым может потребоваться помощь, в том числе и медицинская. В частности, необходимость экстренной доставки медицинских препаратов в зону с затрудненным доступом наземных транспортных средств.

Задача — осуществить поиск и мониторинг зоны чрезвычайной ситуации и доставить медицинский груз в несколько мест, к которым отрезан доступ транспортных средств по земле из-за наводнения.

На заключительном этапе участники работают в симуляторе, собирают макет груза, формируют полетное задание для БВС в рамках летных испытаний на аэродроме.

4.3.3. Требования к команде и компетенциям участников

Количество участников в команде: 3–4 человека.

Компетенции, которыми должны обладать члены команды:

- **Капитан**, лидер команды: общее руководство работой команды, распределение задач, контроль выполнения проекта.
- **Программист C++ и Python**: реализация алгоритмов в виде ПО для БВС.
- **Электронщик**: сборка электрической схемы, пайка компонентов на печатной плате.
- **Физик, математик**: расчет траекторий полета и параметров БВС, составление алгоритмов программ.

4.3.4. Оборудование и программное обеспечение

Таблица 4.3.1

Наименование	Описание
Комплект оборудования наземной станции управления	Используется в рамках летных испытаний на аэродроме
Комплект меток красного цвета	Для имитации точки доставки груза в рамках летных испытаний на аэродроме
Комплект оборудования для пайки	Для пайки компонент, формирующих электрическую схему макета груза
Комплект деталей для макета груза	Для сборки макета груза
БВС с вертикальным взлетом и посадкой	Для доставки макета груза в точку доставки в рамках летных испытаний на аэродроме
Языки программирования C++ и Python	Для написания программного кода во всех задачах финала
Симулятор полета БВС UAViant	Для отработки решений участников в виртуальной среде.

4.3.5. Задачи

Задача 4.3.5.1. (9 баллов)

Одна из перспективных задач БВС — оперативная доставка грузов в труднодоступные места. Чтобы доставка прошла успешно, необходимо сначала обнаружить точку доставки, которую часто обозначают ориентиром, например, цветовой маркировкой.

Условие

Разработайте программу, реализующую обнаружение на изображении точки доставки груза и определение ее координат (при наличии точки).

Входные данные

Изображение, на котором может быть зона доставки груза; файл телеметрии, в котором содержатся координаты и ориентация камеры в момент съемки, а также высота, на которой находится точка доставки. Для открытого примера дан правильный ответ.

Выходные данные

Программа на языке Python, выводящая координаты точки доставки, а в случае отсутствия точки доставки — None.

Критерии оценивания

Программа оценивается на одном открытом примере и на двух закрытых.

За правильное решение открытого примера начисляется 1 балл, за каждый правильно решенный закрытый — по 3 балла.

За каждый вопрос по теории начисляется 1 балл (всего 2 вопроса).

Решение

Предлагаемый алгоритм решения:

1. Разработать алгоритм поиска красного объекта на изображении.
2. Найти луч направления на объект.
3. Перевести направление луча в глобальную систему координат с помощью матрицы поворота.
4. Используя информацию о положении камеры и высоте искомого объекта, найти его координаты.

Пример программы-решения

Ниже представлено решение на языке Python.

Python

```

1  import cv2
2  import numpy as np
3  import pandas as pd
4  class SpatialEstimator:
5      """
6      Класс для обработки телеметрии и изображений с целью определения
7      ↪ мировых координат красного объекта.
8      """
9      def __init__(self, ph=3.9e-06, pw=4.6e-06, f=0.1, resolution=(640,
10     ↪ 380)):
11         """
12         Инициализация параметров камеры и объекта.
13         :param ph: Высота пикселя (м).
14         :param pw: Ширина пикселя (м).
15         :param f: Фокусное расстояние (м).
16         :param resolution: Кортеж (ширина, высота) изображения.
17         :param object_Z: Известная Z-координата объекта в мировых
18         ↪ координатах ....
19         """
20         self.ph = ph
21         self.pw = pw
22         self.f = f
23         self.resolution = resolution

```

```

21 def read_telemetry(self, file_path):
22     """
23     Считывает телеметрию из CSV и возвращает первую строку данных.
24     Ожидается наличие столбцов: X, Y, Z, Yaw, Pitch, Roll.
25     :param file_path: Путь к файлу CSV.
26     :return: Первая строка DataFrame в виде Series.
27     """
28     df = pd.read_csv(file_path)
29     return df.iloc[0]
30 def detect_red_object(self, image):
31     """
32     Находит красный объект на изображении.
33     Преобразует изображение в HSV, проводит пороговую фильтрацию
34     ↪ по красному цвету
35     и вычисляет центр объекта по моментам.
36     :param image: Изображение в формате BGR.
37     :return: Кортеж (u, v) с координатами центра объекта или None,
38     ↪ если объект не найден.
39     """
40     hsv = cv2.cvtColor(image, cv2.COLOR_BGR2HSV)
41     # Диапазоны для красного цвета (учитываются два диапазона по
42     ↪ Hue)
43     lower_red1 = np.array([0, 100, 100])
44     upper_red1 = np.array([10, 255, 255])
45     lower_red2 = np.array([160, 100, 100])
46     upper_red2 = np.array([180, 255, 255])
47     mask1 = cv2.inRange(hsv, lower_red1, upper_red1)
48     mask2 = cv2.inRange(hsv, lower_red2, upper_red2)
49     mask = cv2.bitwise_or(mask1, mask2)
50     # Убираем шумы (открытие и закрытие)
51     kernel = np.ones((3, 3), np.uint8)
52     mask = cv2.morphologyEx(mask, cv2.MORPH_OPEN, kernel)
53     mask = cv2.morphologyEx(mask, cv2.MORPH_CLOSE, kernel)
54     contours, _ = cv2.findContours(mask, cv2.RETR_EXTERNAL,
55     ↪ cv2.CHAIN_APPROX_SIMPLE)
56     if not contours:
57         return None
58     # Выбираем самый большой контур
59     largest_contour = max(contours, key=cv2.contourArea)
60     M = cv2.moments(largest_contour)
61     if M["m00"] == 0:
62         return None
63     cx = int(M["m10"] / M["m00"])
64     cy = int(M["m01"] / M["m00"])
65     return [cx, cy]
66 def rotation_matrix(self, yaw, pitch, roll):
67     """
68     Формирует матрицу поворота по трём углам (Yaw, Pitch, Roll).
69     Углы задаются в градусах. Порядок поворотов: сначала курс
70     ↪ (Yaw, вокруг оси Z),
71     затем тангаж (Pitch, вокруг оси Y), затем крен (Roll, вокруг
72     ↪ оси X).
73     :param yaw: Угол курса.
74     :param pitch: Угол тангажа.
75     :param roll: Угол крена.
76     :return: Матрица поворота 3x3.
77     """
78     # Переводим углы в радианы
79     yaw = np.deg2rad(-yaw)
80     pitch = np.deg2rad(pitch)

```

```

75     roll = np.deg2rad(roll)
76     # Матрица поворота вокруг оси Z (курс)
77     Rz = np.array([
78         [np.cos(yaw), -np.sin(yaw), 0],
79         [np.sin(yaw),  np.cos(yaw), 0],
80         [0, 0, 1]
81     ])
82     # Матрица поворота вокруг оси Y (тангаж)
83     Ry = np.array([
84         [ np.cos(pitch), 0, np.sin(pitch)],
85         [ 0, 1, 0],
86         [-np.sin(pitch), 0, np.cos(pitch)]
87     ])
88     # Матрица поворота вокруг оси X (крен)
89     Rx = np.array([
90         [1, 0, 0],
91         [0, np.cos(roll), -np.sin(roll)],
92         [0, np.sin(roll),  np.cos(roll)]
93     ])
94     # Итоговая матрица поворота
95     R = Rz @ Ry @ Rx
96     return R
97 def compute_world_coordinates(self, telemetry, pixel_coords):
98     """
99     Вычисляет мировые координаты объекта.
100     :param telemetry: Объект с данными телеметрии (поля: X, Y, Z,
101     ↪ Yaw, Pitch, Roll, ObjectHeight).
102     :param pixel_coords: Кортеж (u, v) с координатами объекта на
103     ↪ изображении.
104     :return: Массив с мировыми координатами объекта.
105     """
106     # Убедитесь, что ObjectHeight доступен PI telemetry
107     try:
108         object_Z = telemetry['ObjectHeight']
109     except KeyError:
110         print("Ошибка: 'ObjectHeight' отсутствует в данных
111         ↪ телеметрии.")
112         return None # Или обработать иначе
113     u, v = pixel_coords
114     width, height = self.resolution
115     cx = width / 2
116     cy = height / 2
117     # 1. Луч в координатах камеры (X вправо, Y вниз, Z вперед)
118     x_sensor = (u - cx) * self.pw
119     y_sensor = (v - cy) * self.ph # Убедитесь, что v растет вниз
120     ray_cam = np.array([x_sensor, y_sensor, self.f])
121     # 2. Матрица поворота Тело -> Мир
122     R_body2world = self.rotation_matrix(telemetry['Yaw'],
123     ↪ telemetry['Pitch'], telemetry['Roll'])
124     # 3. Матрица поворота Камера -> Тело (Фиксированная)
125     # Предполагаем ту же ориентацию, что в первом скрипте:
126     # Ось X камеры совпадает с осью X тела
127     # Ось Y камеры противоположна оси Y тела
128     # Ось Z камеры (оптическая) противоположна оси Z тела
129     R_cam2body = np.array([
130         [1, 0, 0],
131         [0, -1, 0],
132         [0, 0, -1]
133     ])

```

```

130     # ВНИМАНИЕ: Если установка камеры другая, эта матрица должна
131     ↪ быть изменена!
132     # 4. Полная матрица поворота Камера -> Мир
133     R_cam2world = R_body2world @ R_cam2body
134     # 5. Преобразуем луч в мировую систему координат
135     ray_world = R_cam2world @ ray_cam
136     # print(f"Ray Cam: {ray_cam}") # PkC,P»P°P°P°P°P°
137     # print(f"Ray World: {ray_world}") # PkC,P»P°P°P°P°P°
138     # 6. Положение камеры в мировой системе координат
139     cam_pos = np.array([telemetry['X'], telemetry['Y'],
140     ↪ telemetry['Z']])
141     # 7. Решаем уравнение прямой для пересечения с Z = object_Z
142     # Проверка деления на ноль (луч параллелен плоскости Z)
143     if abs(ray_world[2]) < 1e-9:
144         print("Предупреждение: Луч почти параллелен плоскости
145         ↪ Z=const.")
146         return None # Или вернуть очень большие координаты / NaN
147     t = (object_Z - cam_pos[2]) / ray_world[2]
148     # Можно добавить проверку t > 0 (пересечение перед камерой)
149     if t < 0:
150         print("Предупреждение: Точка пересечения находится позади
151         ↪ камеры.")
152         # Можно вернуть None или оставить как есть, в зависимости
153         ↪ от требований
154     # 8. Вычисляем мировые координаты точки пересечения
155     obj_world = cam_pos + t * ray_world
156     return obj_world
157 def process(self, telemetry_file, image_file):
158     """
159     Полный процесс: считывание телеметрии, загрузка изображения,
160     обнаружение красного объекта и вычисление его мировых
161     ↪ координат.
162     :param telemetry_file: Путь к файлу телеметрии (CSV).
163     :param image_file: Путь к файлу изображения.
164     :return: Мировые координаты объекта или None, если объект не
165     ↪ найден или ошибка.
166     """
167     try:
168         telemetry = self.read_telemetry(telemetry_file)
169     except FileNotFoundError:
170         print(f" Ошибка: Файл телеметрии не найден:
171         ↪ {telemetry_file}")
172         return None
173     except Exception as e:
174         print(f" Ошибка при чтении телеметрии: {e}")
175         return None
176     image = cv2.imread(image_file)
177     if image is None:
178         print(f" Ошибка: Не удалось загрузить изображение:
179         ↪ {image_file}")
180         return None
181     red_object_pixel = self.detect_red_object(image)
182     if red_object_pixel is None:
183         print("Красный объект не найден на изображении.")
184         return None
185     obj_world = self.compute_world_coordinates(telemetry,
186     ↪ red_object_pixel)
187     return obj_world
188 # --- Остальной код остается без изменений ---
189 if __name__ == "__main__":

```

```

180     estimator = SpatialEstimator(ph=3.9e-06, pw=4.6e-06, f=0.1) #
      ↪ Используем заданные параметры
181     telemetry_file = 'telemetry3.csv'
182     image_file = 'img3.png'
183     obj_world = estimator.process(telemetry_file, image_file)
184     if obj_world is not None:
185         print("\n Вычисленные мировые координаты объекта
186 :")
187         print(f"X={obj_world[0]:.4f}, Y={obj_world[1]:.4f},
      ↪ Z={obj_world[2]:.4f}")

```

Задача 4.3.5.2. (8 баллов)

Одной из перспективных областей применения БВС является оперативная доставка грузов первой необходимости в зоны чрезвычайной ситуации. Это направление развития беспилотных систем особенно актуально для регионов России с низкой транспортной доступностью.

Условие

Разработайте программу, реализующую автоматическое управление углом крена и тангажа БВС для осуществления автоматической доставки груза в зону ЧС.

Входные данные

Симулятор полета БВС; система автоматического управления БВС (кроме регуляторов в каналах крена и тангажа); навигационные параметры БВС в каждый момент времени (координаты и углы ориентации), координаты точки доставки.

Выходные данные

Программа на Arduino, реализующая автоматическую доставку груза по заданным координатам.

Критерии оценивания

Оценивается точность доставки груза. Оценка осуществляется путем расчета расстояния от фактической точки доставки до центра зоны доставки:

- Если расстояние от точки доставки до центра зоны доставки меньше 0,5 м — за решение задачи начисляется 6 баллов.
- За каждые дополнительные 3 м ошибки доставки груза снимается 1 балл.
- Правильный ответ на вопрос по теории прибавляет 1 балл (всего 2 вопроса).

Решение

Предлагаемый алгоритм решения:

1. Выбрать структуру законов управления в каналах крена и тангажа.
2. Определить необходимые значения угла курса и высоты для достижения точки доставки груза.
3. Рассчитать момент времени для сброса груза.
4. Отработать разработанный алгоритм на симуляторе.
5. Проанализировать полученный результат и доработать алгоритмы управления.

Пример программы-решения

C++

```

1  #include "Tasks/Task.h"
2  // Данная функция выполняется только один раз при запуске программы
3  // (при подаче питания на плату, перепрошивке, или нажатии кнопки
   ↪ reset)
4  void Task_solution::setup(HardwareSerial* a_Serial)
5  {
6      debug_serial = a_Serial;
7      // Примеры вывода в отладочную консоль
8      printtoDebugSerial("Hello team!");
9      printtoDebugSerial(String(3.1415926));
10     // Путьевые точки в произвольном порядке
11     _PointsArray[0] = FlyPoints(341, 541, 45);
12 }
13 /*
14 ### УЧАСТНИКАМ ОЛИМПИАДЫ - Основная функция отправки пакета значений
   ↪ крена и тангажа (град) ####
15 Данная функция осуществляет циклический обмен данных с симулятором с
   ↪ частотой 100 Гц.
16 */
17 SignalBody Task_solution::loop(Skywalker2015PacketTelemetry
   ↪ a_telemetry)
18 {
19     /* a_telemetry - структура данных, полученная с симулятора. Она
   ↪ содержит текущие параметры БВС
20     a_telemetry.L - координата X (север) [м]
21     a_telemetry.Z - координата Z (восток) [м]
22     a_telemetry.H - координата Y (высота) [м]
23     a_telemetry.Psi - угол курса [град]
24     a_telemetry.Gam - Угол крена [град]
25     a_telemetry.Tan - Угол тангажа [град]
26     a_telemetry.V - Скорость полёта БВС [м/с]
27     a_telemetry.Vx1 - Продольная скорость [м/с]
28     a_telemetry.Vz1 - Поперечная скорость [м/с]
29     a_telemetry.Vy1 - Вертикальная скорость [м/с]
30     a_telemetry.wx - Угловая скорость вокруг продольной оси [1/с]
31     a_telemetry.wy - Угловая скорость вокруг вертикальной оси [1/с]
32     a_telemetry.wz - Угловая скорость вокруг поперечной оси [1/с]
33     */
34     // РЕКОМЕНДУЕМЫЙ АЛГОРИТМ РЕШЕНИЯ ЗАДАЧИ
35     //
36     SignalBody _ans; // Структура которая отсылается на симулятор
37     // Проверить близость БВС к текущей путевой точке. изменить путевую
   ↪ точку при необходимости
38     // _Point_Index = GetNowPointIndex(a_telemetry.L, a_telemetry.Z,
   ↪ a_telemetry.H);
39     _Point_Index = 0;

```

```

40 // Через координаты путевой точки и текущего местоположения БВС
41 ↪ найти направление на путевую точку (пеленг)
42 _ans.Gamma_direct = PointsFlyGam(_Point_Index, a_telemetry.L,
43 ↪ a_telemetry.Z, a_telemetry.Psi);
44 // Рассчитать необходимое изменение угла курса
45 // Рассчитать требуемый угол крена (для регулирования курса)
46 // На основе текущей высоты БВС и заданной высоты путевой точки
47 ↪ рассчитать необходимое изменение высоты
48 _ans.Tang_direct = PointsFlyTan(a_telemetry.H, a_telemetry.Vy1,
49 ↪ _Point_Index);
50 // Рассчитать требуемый угол тангажа (для регулирования высоты
51 )
52
53 float delX = _PointsArray[0].North - a_telemetry.L;
54 float delY = _PointsArray[0].East - a_telemetry.Z;
55 float delZ = _PointsArray[0].Height - a_telemetry.H;
56 float dist = sqrtf(pow(delX, 2) + pow(delY, 2));
57
58 if (dist < 106.5)
59 {
60     cargo_drop = true;
61     _PointsArray[0] = FlyPoints(341, 541, 60);
62 }
63 // Отправляем команды на симулятор
64 // БВС будет пытаться выдерживать заданные углы крена и тангажа
65 return _ans;
66 }
67
68 float roll_err_last = 0;
69 float Task_solution::GammaReg(float Psi, float Psi_dir)
70 {
71     float Kpsi = 1.6;
72     float roll_err = AngDefines(Psi - Psi_dir);
73     float gamma_cmd = 3.8 * roll_err + 45 * (roll_err - roll_err_last);
74     roll_err_last = roll_err;
75     gamma_cmd = constrain(gamma_cmd, -30, 30);
76     return gamma_cmd;
77 }
78
79 float Task_solution::HeightReg(float Yg, float Vy, float Hz)
80 {
81     float K_H = 0.72;
82     float K_vy_1 = 7.7591;
83     float K_vy_2 = 9.8941;
84     float kh1 = constrain(K_H * (Hz - Yg), -7.5, 7.5);
85     float _Pitch_direct = constrain((kh1 * K_vy_1 - Vy * K_vy_2), -20,
86 ↪ 20);
87     return _Pitch_direct;
88 }
89
90 float Task_solution::ToPointXY(float _Xt, float _Yt, float a_Xg, float
91 ↪ a_Zg, float Psi)
92 {
93     float Psi_cmd = -atan2(a_Zg - _Yt, a_Xg - _Xt);
94     Psi_cmd *= 57.3;
95     return GammaReg(Psi, Psi_cmd);
96 }
97
98 float Task_solution::PointsFlyGam(const size_t& a_Point_index, float
99 ↪ _Xt, float _Yt, float Psi)
100 {
101     float Xg_cmd = _PointsArray[_Point_Index].North;
102     float Zg_cmd = _PointsArray[_Point_Index].East;

```

```

93     return ToPointXY(_Xt, _Yt, Xg_cmd, Zg_cmd, Psi);
94 }
95 float Task_solution::PointsFlyTan(float H, float Vy, const size_t&
↪ a_Point_index)
96 {
97     float H_z = _PointsArray[_Point_Index].Height;
98     return HeightReg(H, Vy, H_z);
99 }
100 size_t Task_solution::GetNowPointIndex(float X, float Z, float H)
101 {
102     size_t max_index = sizeof(_PointsArray) / sizeof(_PointsArray[0]);
103     float Xg_cmd = _PointsArray[_Point_Index].North;
104     float Zg_cmd = _PointsArray[_Point_Index].East;
105     float Yg_cmd = _PointsArray[_Point_Index].Height;
106     float delX = Xg_cmd - X;
107     float delY = Zg_cmd - Z;
108     float delZ = Yg_cmd - H;
109     float Size = sqrtf(pow(delX, 2) + pow(dely, 2) + pow(delZ, 2));
110     // Calculate crosstrack error
111     float x_og = 0.0;
112     float z_og = 0.0;
113     if (_Point_Index > 0)
114     {
115         x_og = _PointsArray[_Point_Index - 1].North;
116         z_og = _PointsArray[_Point_Index - 1].East;
117     }
118     else
119     {
120         x_og = _PointsArray[max_index - 1].North;
121         z_og = _PointsArray[max_index - 1].East;
122     }
123     float relx = X - x_og;
124     float rely = Z - z_og;
125     float delx = Xg_cmd - x_og;
126     float dely = Zg_cmd - z_og;
127     float U = (relx * delx + rely * dely) / (delx * delx + dely * dely);
128     float cte = (rely * delx - relx * dely) / (delx * delx + dely *
↪ dely);
129     if (Size < 7.0)
130     {
131         _Point_Index++;
132     }
133     if (_Point_Index >= max_index)
134     {
135         _Point_Index = 0;
136     }
137     return _Point_Index;
138 }

```

Задача 4.3.5.3. (16 баллов)

Одной из базовых функций, реализуемых БВС, является автономный полет по заданной траектории. Для этого разрабатываются специальные системы автоматического управления, способные регулировать все параметры навигации БВС в зависимости от его положения относительно заданной траектории.

Условие

Разработайте программу, реализующую автоматическое управление полетом БВС с целью осмотра зоны ЧС. Траектория движения формируется участниками команды произвольным образом с целью обеспечения максимальной площади сканирования за заданное время.

Входные данные

Границы зоны ЧС, система автоматического управления БВС (с предыдущей задачи); навигационные параметры БВС в каждый момент времени (координаты и углы ориентации).

Выходные данные

Программа на Arduino, реализующая автономный осмотр зоны ЧС.

Критерии оценивания

Время полета БВС ограничено 3 мин. Оценивается процент просканированной площади зоны ЧС.

- Если доля просканированной площади больше 99,5% площади ЧС, начисляется 14 баллов.
- За каждые «пропущенные» 5% площади зоны ЧС снимается 1 балл.
- Правильный ответ на вопрос по теории прибавляет 1 балл (всего 2 вопроса).

Решение

Предлагаемый алгоритм решения:

1. Отладить структуру законов управления в каналах крена и тангажа.
2. Выбрать траекторию движения БВС (последовательность путевых точек) с учетом заданного ограничения по времени.
3. Отработать разработанный алгоритм на симуляторе.
4. Проанализировать полученный результат и доработать алгоритмы управления.

Пример программы-решения

C++

```

1  #include "Tasks/Task.h"
2  // Данная функция выполняется только один раз при запуске программы
3  // (при подаче питания на плату, перепрошивке, или нажатии кнопки
   ↪ reset)
4  void Task_solution::setup(HardwareSerial* a_Serial)
5  {
6      debug_serial = a_Serial;

```

```

7 // Примеры вывода в отладочную консоль
8 printtoDebugSerial("Hello team!");
9 printtoDebugSerial(String(3.1415926));
10 // Путьевые точки в произвольном порядке
11 _PointsArray[0] = FlyPoints(-550, 60, 65);
12 // _PointsArray[1] = FlyPoints(310, -110, 60);
13 _PointsArray[2] = FlyPoints(-500, -135, 65);
14 // _PointsArray[2] = FlyPoints(-329, -71, 65);
15 _PointsArray[3] = FlyPoints(65, 323, 65);
16 _PointsArray[4] = FlyPoints(-82, -106, 65);
17 _PointsArray[5] = FlyPoints(59, -247, 65);
18 _PointsArray[6] = FlyPoints(347, 41, 65);
19 _PointsArray[7] = FlyPoints(488, -100, 65);
20 _PointsArray[8] = FlyPoints(200, -388, 65);
21 }
22 /*
23 ### УЧАСТНИКАМ ОЛИМПИАДЫ - Основная функция отправки пакета значений
  ↳ крена и тангажа (град) ####
24 Данная функция осуществляет циклический обмен данных с симулятором с
  ↳ частотой 100 Гц.
25 */
26 SignalBody Task_solution::loop(Skywalker2015PacketTelemetry
  ↳ a_telemetry)
27 {
28 /* a_telemetry - структура данных, полученная с симулятора. Она
  ↳ содержит текущие параметры БВС
29 a_telemetry.L - координата X (север) [м]
30 a_telemetry.Z - координата Z (восток) [м]
31 a_telemetry.H - координата Y (высота) [м]
32 a_telemetry.Psi - угол курса [град]
33 a_telemetry.Gam - Угол крена [град]
34 a_telemetry.Tan - Угол тангажа [град]
35 a_telemetry.V - Скорость полёта БВС [м/с]
36 a_telemetry.Vx1 - Продольная скорость [м/с]
37 a_telemetry.Vz1 - Поперечная скорость [м/с]
38 a_telemetry.Vy1 - Вертикальная скорость [м/с]
39 a_telemetry.wx - Угловая скорость вокруг продольной оси [1/с]
40 a_telemetry.wy - Угловая скорость вокруг вертикальной оси [1/с]
41 a_telemetry.wz - Угловая скорость вокруг поперечной оси [1/с]
42 */
43 // РЕКОМЕНДУЕМЫЙ АЛГОРИТМ РЕШЕНИЯ ЗАДАЧИ
44 //
45 SignalBody _ans; // Структура которая отсылается на симулятор
46 // Проверить близость БВС к текущей путевой точке. Изменить путевую
  ↳ точку при необходимости
47 _Point_Index = GetNowPointIndex(a_telemetry.L, a_telemetry.Z,
  ↳ a_telemetry.H);
48 // _Point_Index = 0;
49 // Через координаты путевой точки и текущего местоположения БВС
  ↳ найти направление на путевую точку (пеленг)
50 _ans.Gamma_direct = PointsFlyGam(_Point_Index, a_telemetry.L,
  ↳ a_telemetry.Z, a_telemetry.Psi);
51 // Рассчитать необходимое изменение угла курса
52 // Рассчитать требуемый угол крена (для регулирования курса)
53 // На основе текущей высоты БВС и заданной высоты путевой точки
  ↳ рассчитать необходимое изменение высоты
54 _ans.Tang_direct = PointsFlyTan(a_telemetry.H, a_telemetry.Vy1,
  ↳ _Point_Index);
55 // Рассчитать требуемый угол тангажа (для регулирования высоты)
56

```

```

57
58 // float delX = _PointsArray[0].North - a_telemetry.L;
59 // float delY = _PointsArray[0].East - a_telemetry.Z;
60 // float delZ = _PointsArray[0].Height - a_telemetry.H;
61 // float dist = sqrtf(pow(delX, 2) + pow(dely, 2));
62
63 // if (dist < 106.5)
64 // {
65 //   cargo_drop = true;
66 //   _PointsArray[0] = FlyPoints(-300, 300, 60);
67 // }
68 // Отправляем команды на симулятор
69 // БВС будет пытаться выдерживать заданные углы крена и тангажа
70 return _ans;
71 }
72 float roll_err_last = 0;
73 float Task_solution::GammaReg(float Psi, float Psi_dir)
74 {
75   float Kpsi = 1.6;
76   float roll_err = AngDefines(Psi - Psi_dir);
77   float gamma_cmd = 3.8 * roll_err + 45 * (roll_err - roll_err_last);
78   roll_err_last = roll_err;
79   gamma_cmd = constrain(gamma_cmd, -20, 20);
80   return gamma_cmd;
81 }
82 float Task_solution::HeightReg(float Yg, float Vy, float Hz)
83 {
84   float K_H = 0.72;
85   float K_vy_1 = 7.7591;
86   float K_vy_2 = 9.8941;
87   float kh1 = constrain(K_H * (Hz - Yg), -7.5, 7.5);
88   float _Pitch_direct = constrain((kh1 * K_vy_1 - Vy * K_vy_2), -20,
89   ↪ 20);
89   return _Pitch_direct;
90 }
91 float Task_solution::ToPointXY(float _Xt, float _Yt, float a_Xg, float
92 ↪ a_Zg, float Psi)
93 {
94   float Psi_cmd = -atan2(a_Zg - _Yt, a_Xg - _Xt);
95   Psi_cmd *= 57.3;
96   return GammaReg(Psi, Psi_cmd);
97 }
98 float Task_solution::PointsFlyGam(const size_t& a_Point_index, float
99 ↪ _Xt, float _Yt, float Psi)
100 {
101   float Xg_cmd = _PointsArray[_Point_Index].North;
102   float Zg_cmd = _PointsArray[_Point_Index].East;
103   return ToPointXY(_Xt, _Yt, Xg_cmd, Zg_cmd, Psi);
104 }
105 float Task_solution::PointsFlyTan(float H, float Vy, const size_t&
106 ↪ a_Point_index)
107 {
108   float H_z = _PointsArray[_Point_Index].Height;
109   return HeightReg(H, Vy, H_z);
110 }
111 size_t Task_solution::GetNowPointIndex(float X, float Z, float H)
112 {
113   size_t max_index = sizeof(_PointsArray) / sizeof(_PointsArray[0]);
114   float Xg_cmd = _PointsArray[_Point_Index].North;
115   float Zg_cmd = _PointsArray[_Point_Index].East;

```

```

113 float Yg_cmd = _PointsArray[_Point_Index].Height;
114 float delX = Xg_cmd - X;
115 float delY = Zg_cmd - Z;
116 float delZ = Yg_cmd - H;
117 float Size = sqrtf(pow(delX, 2) + pow(dely, 2) + pow(delZ, 2));
118 // Calculate crosstrack error
119 float x_og = 0.0;
120 float z_og = 0.0;
121 if (_Point_Index > 0)
122 {
123     x_og = _PointsArray[_Point_Index - 1].North;
124     z_og = _PointsArray[_Point_Index - 1].East;
125 }
126 else
127 {
128     x_og = _PointsArray[max_index - 1].North;
129     z_og = _PointsArray[max_index - 1].East;
130 }
131 float relx = X - x_og;
132 float rely = Z - z_og;
133 float delx = Xg_cmd - x_og;
134 float dely = Zg_cmd - z_og;
135 float U = (relx * delx + rely * dely) / (delx * delx + dely * dely);
136 float cte = (rely * delx - relx * dely) / (delx * delx + dely *
↪ dely);
137 if (Size < 7.0)
138 {
139     _Point_Index++;
140 }
141 if (_Point_Index >= max_index)
142 {
143     _Point_Index = 0;
144 }
145 return _Point_Index;
146 }

```

Задача 4.3.5.4. (27 баллов)

В рамках данной задачи участникам выдан макет медицинского груза, а также необходимые для решения задачи комплектующие. Участникам предстоит собрать блок электроники для управления парашютной системой груза и рассчитать значение максимальной перегрузки при падении груза.

Условие

1. Составьте принципиальную схему электронного блока медицинского груза, позволяющую реализовать его функциональные требования.
2. Припаяйте и смонтируйте при помощи выданного набора электронный блок медицинского груза.
3. Напишите программу на ESP32, открывающую парашют по внешней команде, идентифицирующую падение груза с высоты 1 м и более.

Входные данные

Макетная плата; контактные колодки; принадлежности для пайки; датчик МР U6050; плата ESP32 Super mini; АКБ 2S, преобразователь напряжения; фурнитура и провода; пищалка; концевой выключатель; шаблон функционального ПО.

Выходные данные

Принципиальная схема электронного блока, функциональное программное обеспечение.

Критерии оценивания

- Наличие принципиальной схемы подключения: 2 балла.
- Собранный макет медицинского груза со светодиодной индикацией работы всех систем: 10 баллов.
- Работающая система раскрытия парашюта: 7 баллов.
- Работающая система идентификации падения со звуковой сигнализацией: 3 балла.
- Работающая система беспроводной передачи данных: 3 балла.
- Правильный ответ на вопрос по теории прибавляет 2 балла (всего 1 вопрос).

Решение

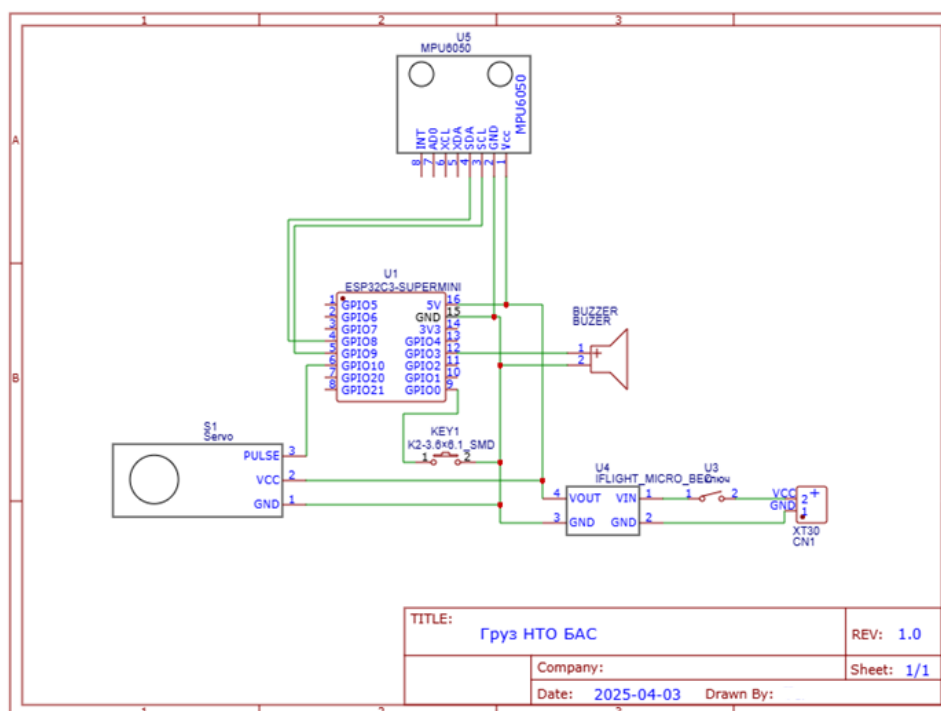


Рис. 4.3.1. Пример принципиальной электрической схемы электронного блока медицинского груза



Рис. 4.3.2. Модель медицинского груза. Общий вид груза

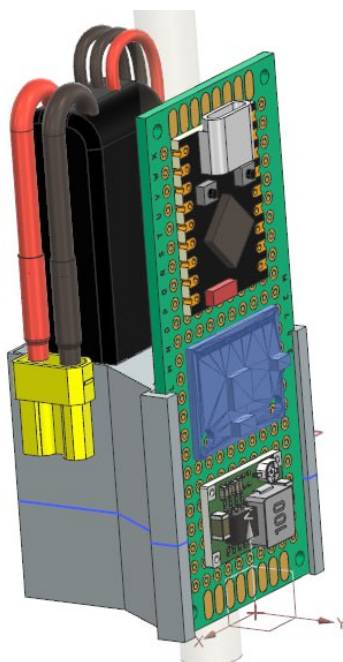


Рис. 4.3.3. Вид модели электронного блока медицинского груза без соединительных проводов

Пример программы-решения

C++

```

1 //Инициализация библиотек
2 #include <Adafruit_MPU6050.h>
3 #include <Adafruit_Sensor.h>
4 #include <Wire.h>
5 #include <ESP32Servo.h> // Библиотека для работы с сервоприводом на
   ↳ ESP32
6 #include <WiFi.h>
7 const char* loraAP_ssid = "LORA_AP"; // Сеть для отправки
   ↳ данных
  
```

```

8  const char* loraAP_pass = "12345678";
9  WiFiClient loraClient;
10 #if defined(ARDUINO_SUPER_MINI_ESP32C3)
11     static uint8_t ledPin = BUILTIN_LED;    // LED connected to digital
        ↳ pin
12 #else
13     static uint8_t ledPin = 8;              // LED connected to digital
        ↳ pin
14 #endif
15 static uint8_t ledOn = LOW;
16 #define DELTA 10
17 #define DELAY 50
18 unsigned long LEDdelaytime = 0;
19 int delta = 5;
20 int fade = 0;
21 void pulse(void) {
22     if (millis() - LEDdelaytime > DELAY) {
23         fade += delta;
24         if (fade <= 0) {
25             fade = 0;
26             delta = DELTA;
27         } else if (fade >= 255) {
28             fade = 255;
29             delta = - DELTA;
30         }
31         analogWrite(ledPin, fade);
32         LEDdelaytime = millis();
33     }
34 }
35 Servo myServo; // Сервопривод
36 // Переменные для хранения смещений
37 float accelOffsetX = 0;
38 float accelOffsetY = 0;
39 float accelOffsetZ = 0;
40 const int buttonPin = 4;    // Пин для оптического датчика
41 const int servoPin = 3;    // Пин для сервопривода
42 const int buzzerPin = 2;   // Пин для пищалки
43 int16_t ax, ay, az;
44 int16_t gx, gy, gz;
45 bool objectDropped = false; //Состояние сброса груза
46 unsigned long dropTime = 0; //Время сброса
47 float maxAcceleration = 0; //Максимальное значение
        ↳ акселерометра
48 const int numReadings = 7;
49
50 float readings1[numReadings]; // данные, считанные с входного
        ↳ аналогового контакта
51 float readings2[numReadings]; // данные, считанные с входного
        ↳ аналогового контакта
52 int sss1 = 0; // индекс для значения, которое
        ↳ считывается в данный момент
53 float total1 = 0; // суммарное значение
54 float average1 = 0; // среднее значение
55
56 int sss2 = 0; // индекс для значения, которое
        ↳ считывается в данный момент
57 float total2 = 0; // суммарное значение
58 float average2 = 0; // среднее значение
59 int K=0;
60 float S2=0;

```

```

61 float S3=0;
62 float S4=0;
63 float XX=0;
64 float YY=0;
65 float ZZ=0;
66 float totalAbs = 0;
67 Adafruit MPU6050 mpul;
68 void calibrateMPU6050() {
69   Serial.println("Начало калибровки MPU6050...");
70   Serial.println("Убедитесь, что датчик неподвижен!");
71   int calibrationSamples = 1000; // Количество выборок для калибровки
72   float totalAccelX = 0, totalAccelY = 0, totalAccelZ = 0;
73   for (int i = 0; i < calibrationSamples; i++) {
74     sensors_event_t a, g, temp;
75     mpul.getEvent(&a, &g, &temp);
76     // Суммируем значения акселерометра
77     totalAccelX += a.acceleration.x;
78     totalAccelY += a.acceleration.y;
79     totalAccelZ += a.acceleration.z;
80     totalAbs += sqrt(a.acceleration.x*a.acceleration.x +
81     ↪ a.acceleration.y*a.acceleration.y +
82     ↪ a.acceleration.z*a.acceleration.z);
83     delay(5); // Задержка для стабильности
84   }
85   // Вычисляем средние значения смещений
86   accelOffsetX = totalAccelX / calibrationSamples;
87   accelOffsetY = totalAccelY / calibrationSamples;
88   accelOffsetZ = totalAccelZ / calibrationSamples;
89   totalAbs = totalAbs / calibrationSamples;
90   // Выводим результаты калибровки
91   Serial.println("Калибровка завершена!");
92   Serial.print("Смещение по X: "); Serial.println(accelOffsetX);
93   Serial.print("Смещение по Y: "); Serial.println(accelOffsetY);
94   Serial.print("Смещение по Z: "); Serial.println(accelOffsetZ);
95   analogWrite(ledPin, 255);
96 }
97 void setup() {
98   //Serial.setTxTimeoutMs(0);
99   Serial.begin(115200); //38400
100  Wire.begin();
101  mpul.begin(104);
102  mpul.setAccelerometerRange(MPU6050_RANGE_16_G);
103
104  // Калибровка MPU6050
105  calibrateMPU6050();
106
107  ESP32PWM::allocateTimer(0); // Выделение таймера для сервопривода
108  myServo.setPeriodHertz(50); // Стандартная частота для сервоприводов
109  ↪ (50 Гц)
110  myServo.attach(servoPin); // Подключение сервопривода
111  myServo.write(0); // Установка начального положения (0
112  ↪ градусов)
113
114  // Инициализация пищалки
115  pinMode(buzzerPin, OUTPUT);
116  // Инициализация оптического датчика
117  pinMode(buttonPin, INPUT);
118 }

```

```

117 float gammaa1 = 0.0;
118 float acc1[] = {-0.36, -0.09, -1.8};
119 float acc_scale1 = 1;
120
121 float gammaa2 = 0.0;
122 float acc2[] = {-0.36, -0.09, -1.8};
123 float acc_scale2 = 1;
124
125 void loop() {
126     // Считывание показаний оптического датчика
127     int buttonValue = digitalRead(buttonPin);
128
129     // Если груз ещё не сброшен и датчик переключился с 0 на 1
130     if (!objectDropped && buttonValue == 1) {
131         objectDropped = true;
132         dropTime = millis();
133     }
134     if (objectDropped)
135     {
136         // Получение данных с акселерометра
137         //код для обработки данных с MPU6050
138         sensors_event_t a1, g1, temp1;
139         mpul1.getEvent(&a1, &g1, &temp1);
140
141         float acc_x1 = (a1.acceleration.x) * acc_scale1;
142         float acc_y1 = (a1.acceleration.y) * acc_scale1;
143         float acc_z1 = (a1.acceleration.z) * acc_scale1;
144
145
146
147         //Y=0.995581*(acc_x2 - S2 + 0.9987*Y);
148         // XX=0.99*(acc_x1 - S2 + 0.99*XX);
149         // S2 = acc_x1;
150         // YY=0.99*(acc_y1 - S3 + 0.99*YY);
151         // S3 = acc_y1;
152         // ZZ=0.99*(acc_z1 - S4 + 0.99*ZZ);
153         // S4 = acc_z1;
154         float S = (sqrt((acc_x1*acc_x1) + (acc_y1*acc_y1) +
155             ↪ (acc_z1*acc_z1)));
156         //Serial.println("S");
157         //Serial.println(S);
158         if (S<3)
159         {
160             S = 0;
161         }
162         //Serial.println("S");
163         //Serial.println(S);
164         //Serial.println(" ");
165         // Фиксация максимального ускорения
166         S = S - totalAbs;
167         if ((S) > maxAcceleration && millis() > 1000) {
168             //maxAcceleration = 0;
169             maxAcceleration = S;
170         }
171         //Serial.println(maxAcceleration);
172         /*
173         if (S > maxAcceleration && millis() - dropTime > 100) {
174             maxAcceleration = 0;
175             maxAcceleration = S;
176         }

```

```

176     */
177     // Через 2 секунды после сброса дать команду на сервопривод
178     if (millis() - dropTime > 2000) {
179         myServo.write(180); // Поворот сервопривода на 180 градусов
180     }
181     // Через 30 секунд после сброса
182     if (millis() - dropTime > 10000) {
183         // Вывод максимального ускорения в UART
184         Serial.print("Max Acceleration: ");
185         Serial.println(maxAcceleration);
186         // Подключаемся к другой сети
187         WiFi.begin(loraAP_ssid, loraAP_pass);
188         while (WiFi.status() != WL_CONNECTED) {
189             WiFi.begin(loraAP_ssid, loraAP_pass);
190             // delay(500);
191             // digitalWrite(buzzerPin, LOW); // Выключить пищалку
192             pulse();
193         }
194         if (loraClient.connect("192.168.4.1", 80))
195         {
196             loraClient.println("acc:" + String(maxAcceleration));
197         }
198
199
200         // Serial.print(acc_x1);
201         // Serial.print(",");
202         // Serial.print(acc_y1);
203         // Serial.print(",");
204         // Serial.print(acc_z1);
205         // Serial.println(",");
206         // Импульсный сигнал на пищалку
207         // static unsigned long lastBuzzerToggle = 0;
208         // static bool buzzerState = false;
209         // if (millis() - lastBuzzerToggle > 200) {
210         //     buzzerState = !buzzerState;
211         //     if (buzzerState)
212         //     {
213         //         digitalWrite(buzzerPin, HIGH); // Включить пищалку
214         //     }
215         //     else
216         //     {
217         //         digitalWrite(buzzerPin, LOW); // Выключить пищалку
218         //     }
219         //     lastBuzzerToggle = millis(); // Обновление времени
220         //     ↪ последнего переключения
221
222         // }
223     }
224     //код для обработки данных с MPU6050
225
226     /* Get new sensor events with the readings */
227 }

```

Задача 4.3.5.5. (12 баллов)

Участникам предстоит по бортовой телеметрии с видеокamеры (навигация и ориентация БВС — крен, тангаж, курс, высота, долгота, широта) идентифицировать

объект и определить его координаты, после чего добавить их в полетное задание, а также установить метку сброса груза.

Осуществить сброс груза можно при помощи добавления ППМ Gripper release.

Условие

1. Обработайте видео и определите координаты объекта.
2. Сформируйте полетное задание с точкой сброса груза.
3. Отработайте полетное задание на летных испытаниях.

Входные данные

Видео с бортовой камеры; телеметрия полета; угол обзора камеры 75° (по диагонали), бланк полетного задания.

Выходные данные

Координаты объекта в составе полетного задания и результаты летных испытаний.

Критерии оценивания

- Определение координат объекта с заданной точностью: 4 балла.
- Факт сброса груза при выполнении полетного задания: 4 балла.
- Работающая система раскрытия парашюта: 4 балла.
- Максимальное число баллов за задачу: 12 баллов.

Решение

Ниже представлен пример полетного задания.

```
json
1  {
2    "fileType": "Plan",
3    "geoFence": {
4      "circles": [
5        ],
6      "polygons": [
7        ],
8      "version": 2
9    },
10   "groundStation": "QGroundControl",
11   "mission": {
12     "cruiseSpeed": 15,
13     "firmwareType": 3,
14     "globalPlanAltitudeMode": 1,
15     "hoverSpeed": 5,
16     "items": [
```

```
17     {
18         "AMSLAltAboveTerrain": 30,
19         "Altitude": 30,
20         "AltitudeMode": 1,
21         "autoContinue": true,
22         "command": 84,
23         "doJumpId": 1,
24         "frame": 3,
25         "params": [
26             0,
27             0,
28             0,
29             0,
30             56.097170899999995,
31             35.8763223,
32             30
33         ],
34         "type": "SimpleItem"
35     },
36     {
37         "AMSLAltAboveTerrain": 50,
38         "Altitude": 50,
39         "AltitudeMode": 1,
40         "autoContinue": true,
41         "command": 16,
42         "doJumpId": 2,
43         "frame": 3,
44         "params": [
45             0,
46             0,
47             0,
48             null,
49             56.097959452143826,
50             35.874978916685166,
51             50
52         ],
53         "type": "SimpleItem"
54     },
55     {
56         "autoContinue": true,
57         "command": 3000,
58         "doJumpId": 3,
59         "frame": 0,
60         "params": [
61             4,
62             0,
63             0,
64             0,
65             0,
66             0,
67             0
68         ],
69         "type": "SimpleItem"
70     },
71     {
72         "AMSLAltAboveTerrain": null,
73         "Altitude": 50,
74         "AltitudeMode": 1,
75         "autoContinue": true,
76         "command": 16,
```

```
77         "doJumpId": 4,  
78         "frame": 3,  
79         "params": [  
80             0,  
81             0,  
82             0,  
83             null,  
84             56.09804814731677,  
85             35.87444633766345,  
86             50  
87         ],  
88         "type": "SimpleItem"  
89     },  
90     {  
91         "autoContinue": true,  
92         "command": 211,  
93         "doJumpId": 5,  
94         "frame": 2,  
95         "params": [  
96             1,  
97             0,  
98             0,  
99             0,  
100            0,  
101            0,  
102            0  
103         ],  
104         "type": "SimpleItem"  
105     },  
106     {  
107         "AMSLAltAboveTerrain": 70,  
108         "Altitude": 70,  
109         "AltitudeMode": 1,  
110         "autoContinue": true,  
111         "command": 16,  
112         "doJumpId": 6,  
113         "frame": 3,  
114         "params": [  
115             0,  
116             0,  
117             0,  
118             null,  
119             56.09831421908459,  
120             35.87284858587688,  
121             70  
122         ],  
123         "type": "SimpleItem"  
124     },  
125     {  
126         "AMSLAltAboveTerrain": 70,  
127         "Altitude": 70,  
128         "AltitudeMode": 1,  
129         "autoContinue": true,  
130         "command": 16,  
131         "doJumpId": 7,  
132         "frame": 3,  
133         "params": [  
134             0,  
135             0,  
136             0,
```

```
137         null,
138         56.09869265231478,
139         35.874126428124725,
140         70
141     ],
142     "type": "SimpleItem"
143 },
144 {
145     "AMSLAltAboveTerrain": 70,
146     "Altitude": 70,
147     "AltitudeMode": 1,
148     "autoContinue": true,
149     "command": 16,
150     "doJumpId": 8,
151     "frame": 3,
152     "params": [
153         0,
154         0,
155         0,
156         null,
157         56.09839044750299,
158         35.876002922561895,
159         70
160     ],
161     "type": "SimpleItem"
162 },
163 {
164     "AMSLAltAboveTerrain": 0,
165     "Altitude": 0,
166     "AltitudeMode": 1,
167     "autoContinue": true,
168     "command": 21,
169     "doJumpId": 9,
170     "frame": 3,
171     "params": [
172         0,
173         0,
174         0,
175         null,
176         56.097805553443784,
177         35.876140121248255,
178         0
179     ],
180     "type": "SimpleItem"
181 }
182 ],
183 "plannedHomePosition": [
184     56.09779451800145,
185     35.87598385677313,
186     162
187 ],
188 "vehicleType": 1,
189 "version": 2
190 },
191 "rallyPoints": {
192     "points": [
193     ],
194     "version": 2
195 },
196 "version": 1
```

Задача 4.3.5.6. (28 баллов)

Своевременная доставка грузов в зону ЧС требует решения целого ряда задач. Одна из них — логистическая:

- Как и куда нужно лететь?
- Где сбросить груз?
- В каком порядке осуществлять доставку?

Условие

Разработайте программу на языке Python, реализующую автоматическое траекторное управление полетом БВС и автономную доставку двух грузов в зону ЧС с использованием системы технического зрения.

Входные данные

Симулятор полета БВС; границы зоны ЧС; навигационные параметры БВС в текущий момент времени; видеопоток данных с бортовой камеры БВС; недоработанное решение подзадачи 3.

Выходные данные

Программа на Python, реализующая автоматический расчет координат местоположений точек доставки груза с использованием алгоритмов технического зрения, а также собственно автоматическую доставку грузов.

Критерии оценивания

- Оценивается точность доставки груза в каждую из двух точек.
- При попадании груза в пределах 5 м от координат точки доставки начисляется *13 баллов*. За каждый дополнительный 1 м ошибки снимается 1 балл.
- Доставка в каждую точку оценивается отдельно.
- Время полета БВС ограничено 5 мин.
- Правильный ответ на вопрос по теории прибавляет 1 балл (всего 2 вопроса).
- Таким образом, за задачу можно получить *до 28 баллов*.

Решение

Предлагаемый алгоритм решения:

1. Сформировать траекторию полета БВС; написать код алгоритма траекторного управления для реализации этой траектории.

2. Доработать систему управления БВС.
3. Отладить алгоритм определения координат точки доставки груза.
4. Продумать логику сброса груза после нахождения координат точек доставки
5. Отработать разработанные алгоритмы на симуляторе.

Пример программы-решения

Python

```

1  # -*- coding: utf-8 -*-
2  import socket
3  from threading import Thread
4  import time
5  from math import atan2, cos, sin, sqrt, pi, atan
6  import cv2
7  import numpy as np
8  import uav_state
9  import serial
10 import av
11 import subprocess
12 # Последние полученные данные телеметрии
13 last_telemetry = uav_state.uav_state()
14 running = True
15 # capture = cv2.VideoCapture()
16 # container = av.open("rtp-forwarder.sdp",
17 #                    format="sdp",
18 #                    options={"protocol_whitelist": "file,udp,rtp"})
19 ser = serial.Serial(
20     port='COM8',
21     baudrate = 115200,
22     parity = serial.PARITY_NONE,
23     stopbits = serial.STOPBITS_ONE,
24     bytesize = serial.EIGHTBITS,
25     timeout = 0.1 )
26 # UDP сервер для параллельного получения данных телеметрии
27 def udp_server_thread():
28     host = '127.0.0.1'
29     port = 63504
30     addr = (host, port)
31     server = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
32     server.bind(addr)
33     while running:
34         d = server.recvfrom(1024)
35         received = d[0]
36         addr = d[1]
37
38         received = received.decode("utf-8")
39         received = received.replace("\t\r\n", "")
40         #print("----")
41         #print(received)
42         arr = [float(i) for i in received.split('\t')]
43         if len(arr) == 18:
44             last_telemetry.update(arr)
45     server.close()
46 def sendNewWaypoint(newx, newz, newh):
47     ser.write("wpnt {:.2f} {:.2f} {:.2f} \n".format(newx, newz,
48     ↪ newh).encode())
49     print("wpnt {:.2f} {:.2f} {:.2f}".format(newx, newz, newh))

```

```

49 def sendDropCargo():
50     ser.write("drop \n".encode())
51     print("cargo drop attempt!")
52 def find_surroundings(image, center, radius):
53     dots = []
54     h, w = image.shape
55     for row in range(center[1] - radius, center[1] + radius):
56         if row < h and row >= 0:
57             dots.append(image[row][center[0]])
58     for col in range(center[0] - radius, center[0] + radius):
59         if col < w and col >= 0:
60             dots.append(image[center[1]][col])
61     avg_val = sum(dots) / len(dots)
62     return avg_val
63 # Поток обработки данных с камеры
64 def camera_processing_thread():
65     # Параметры камеры
66     F = 6.1 * 1e-3 # Фокусное расстояние
67     dv = 7.8 * 1e-6 # высота пикселя
68     dh = 9.2 * 1e-6 # ширина пикселя
69     xadj = 2
70     zadj = -8
71     # cap = cv2.VideoCapture("rtp-forwarder.sdp")
72     # cap = cv2.VideoCapture("circle.png")
73     # cap.set(cv2.CAP_PROP_BUFFERSIZE, 0)
74     yellow_lower = np.array([20, 100, 100])
75     yellow_upper = np.array([30, 255, 255])
76     # lower mask (0-10)
77     lower_red1 = np.array([0, 150, 150])
78     upper_red1 = np.array([10, 255, 255])
79
80     # upper mask (170-180)
81     lower_red2 = np.array([175, 150, 150])
82     upper_red2 = np.array([180, 255, 255])
83     # Open the SDP file with PyAV.
84     # The 'options' dictionary lets you pass extra parameters (like
85     ↪ protocol whitelisting).
86     container = av.open("rtp-forwarder.sdp",
87                        format="sdp",
88                        options={"protocol_whitelist":
89                                ↪ "file,udp,rtp"})
90     # Find the video stream (assumes there's at least one video stream
91     ↪ in the SDP)
92     video_stream = next((s for s in container.streams if s.type ==
93                          ↪ "video"), None)
94     if video_stream is None:
95         print("No video stream found in the SDP file.")
96         exit(1)
97     print("ready to search")
98     coords = (200, 900)
99     stage = 1 # wp number
100    evasion = False
101    drop = False
102    point_one_found = False
103    point_two_found = False
104    point_one_coords = [-999, -999]
105    point_two_coords = [-999, -999]
106    # sendNewWaypoint(-245, 40, 80)
107    target = (-245, 40)
108    drop_dist = 0

```

```

105     # Основной цикл программы
106     # Decode video frames
107     for packet in container.demux(video_stream):
108         if point_two_found:
109             break
110     for frame in packet.decode():
111         # Convert the frame to a numpy array in a format suitable
112         ↪ for OpenCV (BGR)
113         # Считанное изображение с бортовой камеры
114         img = frame.to_ndarray(format="bgr24")
115
116         # Добавьте ваше решение сюда! -----
117         height, width, channels = img.shape
118         tele = last_telemetry
119         X = tele.x
120         Z = tele.z
121         H = tele.h
122         Heading = tele.yaw * pi / 180
123         img = cv2.GaussianBlur(img, (5, 5), 0)
124         hsv = cv2.cvtColor(img, cv2.COLOR_BGR2HSV)
125         yellow_mask = cv2.inRange(hsv, yellow_lower, yellow_upper)
126         # mask0 = cv2.inRange(hsv, lower_red1, upper_red1)
127         # mask1 = cv2.inRange(hsv, lower_red2, upper_red2)
128         # yellow_mask = mask0+mask1
129         contours, hierarchy = cv2.findContours(yellow_mask,
130         ↪ cv2.RETR_LIST, cv2.CHAIN_APPROX_SIMPLE)
131         maxlen = 0
132         if len(contours) > 0:
133             truecont = contours[0]
134             for icontour in contours:
135                 if len(icontour) > maxlen:
136                     maxlen = len(icontour)
137                     truecont = icontour
138         if maxlen > 600: # 560
139             print(f"max yellow contour: {maxlen}")
140         if maxlen > 1600: # 560
141
142             cv2.drawContours(img, truecont, -1, (255,0,0), 2)
143             sum_x = 0
144             sum_y = 0
145             for i in range(0, len(truecont)):
146                 sum_x += truecont[i][0][0]
147                 sum_y += truecont[i][0][1]
148             avg_x = int(round(sum_x / len(truecont)))
149             avg_y = int(round(sum_y / len(truecont)))
150             dxp = avg_x - width / 2
151             dyp = height/2 - avg_y
152             dxx = H / F * dh
153             dxy = H / F * dv
154             # print(f"Pixel size in meters: {dxx}")
155             shift_x = dxp * dxx
156             shift_y = dyp * dxy
157             geo_x = shift_y * cos(Heading) - shift_x *
158             ↪ sin(Heading) + X
159             geo_y = shift_y * sin(Heading) + shift_x *
160             ↪ cos(Heading) + Z
161             print("Delivery coords X:{:2f} Z:{:2f}".format(geo_x,
162             ↪ geo_y))
163             # These are used to avoid multiple detections of the
164             ↪ same point

```

```

159         dist1 = sqrt((geo_x - point_one_coords[0]) ** 2 +
160 ↪ (geo_y - point_one_coords[1]) ** 2)
161         dist2 = sqrt((geo_x - point_two_coords[0]) ** 2 +
162 ↪ (geo_y - point_two_coords[1]) ** 2)
163         if not point_one_found:
164             point_one_found = True
165             point_one_coords = [geo_x, geo_y]
166             print("Point 1 found")
167         elif point_one_found and not point_two_found and dist1
168 ↪ > 250:
169             point_two_found = True
170             point_two_coords = [geo_x, geo_y]
171             print("Point 2 found")
172             break
173             # print(f'{X} {Z}')
174             # Отобразить изображение
175             # cv2.imshow("Live RTP Stream", img)
176             if cv2.waitKey(5) & 0xFF == ord("q"):
177                 break
178         else:
179             # Continue if inner loop didn't break
180             continue
181             # Break outer loop if inner loop was broken.
182             break
183     cv2.destroyAllWindows()
184     print("Both points have been found")
185     print(f" point 1: {point_one_coords}")
186     print(f" point 2: {point_two_coords}")
187     print(f'Approaching first delivery point: {point_one_coords[0]}
188 ↪ {point_one_coords[1]} {45}')
189     sendNewWaypoint(point_one_coords[0], point_one_coords[1], 48)
190     dist = 999
191     while dist > 100:
192         tele = last_telemetry
193         X = tele.x
194         Z = tele.z
195         dist = sqrt((X - point_one_coords[0]) ** 2 + (Z -
196 ↪ point_one_coords[1]) ** 2)
197     sendDropCargo()
198     sendNewWaypoint(point_one_coords[0], point_one_coords[1], 57)
199     time.sleep(4)
200     print(f'Approaching second delivery point: {point_two_coords[0]}
201 ↪ {point_two_coords[1]} {45}')
202     sendNewWaypoint(point_two_coords[0], point_two_coords[1], 48)
203     dist = 999
204     while dist > 100:
205         tele = last_telemetry
206         X = tele.x
207         Z = tele.z
208         dist = sqrt((X - point_two_coords[0]) ** 2 + (Z -
209 ↪ point_two_coords[1]) ** 2)
210     sendDropCargo()
211     sendNewWaypoint(point_two_coords[0], point_two_coords[1], 57)
212     time.sleep(4)
213     # cv2.destroyAllWindows()
214 # УЧАСТНИКАМ НТО - ЗАДАЧА -----
215 # Основная функция для управления потоками
216 if __name__ == '__main__':
217     # Open RTP livestream through golang script
218     # sendNewWaypoint(-245, 40, 80)

```

```

212     subprocess.Popen(["go", "run", "./main.go"])
213     time.sleep(5)
214     capture_thr = Thread(target=udp_server_thread, args=[])
215     capture_thr.daemon = True
216     capture_thr.start()
217     process_thr = Thread(target=camera_processing_thread, args=[])
218     process_thr.daemon = True
219     process_thr.start()
220     try:
221         while running:
222             time.sleep(1)
223     except KeyboardInterrupt:
224         print('\ninterrupted!')

```

Ниже представлена программа для ESP32 (выдается шаблон, который может быть использован и изменен участниками на их усмотрение).

Arduino

```

1  #include "Tasks/Task.h"
2  // Данная функция выполняется только один раз при запуске программы
3  // (при подаче питания на плату, перепрошивке, или нажатии кнопки
   ↪ reset)
4  void Task_solution::setup(HardwareSerial* a_Serial)
5  {
6      debug_serial = a_Serial;
7      // Примеры вывода в отладочную консоль
8      printtoDebugSerial("Hello team!");
9      printtoDebugSerial(String(3.1415926));
10     // Путьевые точки в произвольном порядке
11     _PointsArray[0] = FlyPoints(-700, 0, 65);
12     _PointsArray[1] = FlyPoints(-500, -135, 65);
13     _PointsArray[2] = FlyPoints(65, 323, 65);
14     _PointsArray[3] = FlyPoints(-82, -106, 65);
15     _PointsArray[4] = FlyPoints(59, -247, 65);
16     _PointsArray[5] = FlyPoints(347, 41, 65);
17     _PointsArray[6] = FlyPoints(488, -100, 65);
18     _PointsArray[7] = FlyPoints(200, -388, 65);
19 }
20 /*
21 ### Основная функция отправки пакета значений крена и тангажа (град)
   ↪ ###
22 Данная функция осуществляет циклический обмен данных с симулятором с
   ↪ частотой 100 Гц.
23 */
24 SignalBody Task_solution::loop(Skywalker2015PacketTelemetry
   ↪ a_telemetry)
25 {
26     /* a_telemetry - структура данных, полученная с симулятора. Она
   ↪ содержит текущие параметры БВС
27     a_telemetry.L - координата X (север) [м]
28     a_telemetry.Z - координата Z (восток) [м]
29     a_telemetry.H - координата Y (высота) [м]
30     a_telemetry.Psi - угол курса [град]
31     a_telemetry.Gam - Угол крена [град]
32     a_telemetry.Tan - Угол тангажа [град]
33     a_telemetry.V - Скорость полёта БВС [м/с]
34     a_telemetry.Vx1 - Продольная скорость [м/с]
35     a_telemetry.Vz1 - Поперечная скорость [м/с]
36     a_telemetry.Vy1 - Вертикальная скорость [м/с]

```

```

37  a_telemetry.wx - Угловая скорость вокруг продольной оси [1/c]
38  a_telemetry.wy - Угловая скорость вокруг вертикальной оси [1/c]
39  a_telemetry.wz - Угловая скорость вокруг поперечной оси [1/c]
40  */
41  // РЕКОМЕНДУЕМЫЙ АЛГОРИТМ РЕШЕНИЯ ЗАДАЧИ 4
42  //
43  SignalBody _ans; // Структура которая отсылается на симулятор
44  // Проверить близость БВС к текущей путевой точке. Изменить путевую
  ↪ точку при необходимости
45  _Point_Index = GetNowPointIndex(a_telemetry.L, a_telemetry.Z,
  ↪ a_telemetry.H);
46  // _Point_Index = 0;
47  // Через координаты путевой точки и текущего местоположения БВС
  ↪ найти направление на путевую точку (пеленг)
48  _ans.Gamma_direct = PointsFlyGam(_Point_Index, a_telemetry.L,
  ↪ a_telemetry.Z, a_telemetry.Psi);
49  // Рассчитать необходимое изменение угла курса
50  // Рассчитать требуемый угол крена (для регулирования курса)
51  // На основе текущей высоты БВС и заданной высоты путевой точки
  ↪ рассчитать необходимое изменение высоты
52  _ans.Tang_direct = PointsFlyTan(a_telemetry.H, a_telemetry.Vy1,
  ↪ _Point_Index);
53  // Рассчитать требуемый угол тангажа (для регулирования высоты)
54
55  // Отправляем команды на симулятор
56  // БВС будет пытаться выдерживать заданные углы крена и тангажа
57  return _ans;
58  }
59  float roll_err_last = 0;
60  float Task_solution::GammaReg(float Psi, float Psi_dir)
61  {
62      float Kpsi = 1.6;
63      float roll_err = AngDefines(Psi - Psi_dir);
64      float gamma_cmd = 3.8 * roll_err + 45 * (roll_err - roll_err_last);
65      roll_err_last = roll_err;
66      gamma_cmd = constrain(gamma_cmd, -20, 20);
67      return gamma_cmd;
68  }
69  float Task_solution::HeightReg(float Yg, float Vy, float Hz)
70  {
71      float K_H = 0.72;
72      float K_vy_1 = 7.7591;
73      float K_vy_2 = 9.8941;
74      float kh1 = constrain(K_H * (Hz - Yg), -7.5, 7.5);
75      float _Pitch_direct = constrain((kh1 * K_vy_1 - Vy * K_vy_2), -20,
  ↪ 20);
76      return _Pitch_direct;
77  }
78  float Task_solution::ToPointXY(float _Xt, float _Yt, float a_Xg, float
  ↪ a_Zg, float Psi)
79  {
80      float Psi_cmd = -atan2(a_Zg - _Yt, a_Xg - _Xt);
81      Psi_cmd *= 57.3;
82      return GammaReg(Psi, Psi_cmd);
83  }
84  float Task_solution::PointsFlyGam(const size_t& a_Point_index, float
  ↪ _Xt, float _Yt, float Psi)
85  {
86      float Xg_cmd = _PointsArray[_Point_Index].North;
87      float Zg_cmd = _PointsArray[_Point_Index].East;

```

```

88     if (userControlEnabled)
89     {
90         Xg_cmd = now_VPP.North;
91         Zg_cmd = now_VPP.East;
92     }
93     return ToPointXY(_Xt, _Yt, Xg_cmd, Zg_cmd, Psi);
94 }
95 float Task_solution::PointsFlyTan(float H, float Vy, const size_t&
    ↪ a_Point_index)
96 {
97     float H_z = _PointsArray[_Point_Index].Height;
98     if (userControlEnabled)
99     {
100         H_z = now_VPP.Height;
101     }
102     return HeightReg(H, Vy, H_z);
103 }
104 size_t Task_solution::GetNowPointIndex(float X, float Z, float H)
105 {
106     size_t max_index = sizeof(_PointsArray) / sizeof(_PointsArray[0]);
107     float Xg_cmd = _PointsArray[_Point_Index].North;
108     float Zg_cmd = _PointsArray[_Point_Index].East;
109     float Yg_cmd = _PointsArray[_Point_Index].Height;
110     if (userControlEnabled)
111     {
112         Xg_cmd = now_VPP.North;
113         Zg_cmd = now_VPP.East;
114         Yg_cmd = now_VPP.Height;
115     }
116     float delX = Xg_cmd - X;
117     float delY = Zg_cmd - Z;
118     float delZ = Yg_cmd - H;
119     float Size = sqrtf(pow(delX, 2) + pow(delY, 2) + pow(delZ, 2));
120     if (Size < 7.0)
121     {
122         if (userControlEnabled)
123             userControlEnabled = false;
124         else
125             _Point_Index++;
126     }
127     if (_Point_Index >= max_index)
128     {
129         _Point_Index = 0;
130     }
131     return _Point_Index;
132 }

```

4.3.6. Материалы для подготовки

1. Вводный курс по Arduino: <http://edurobots.ru/kurs-arduino-dly-a-nachinayushhix/>.
2. Вводный курс МАИ по БПЛА самолетного типа: <https://stepik.org/course/58930/syllabus>.
3. Сквозные технологии. Электроника: <https://stepik.org/course/57121/syllabus>.
4. Шпаргалка по OpenCV-Python: <https://tproger.ru/translations/op>

[encv-python-guide/](#).

5. Курс «Программирование на Python» для начинающих: <https://stepik.org/course/67/promo>.
6. Шакирьянов Эдуард Данисович. Компьютерное зрение на Python: первые шаги: учебное пособие: [6+] / Шакирьянов Эдуард Данисович. — эл. изд. — Москва: Лаборатория знаний, 2021. — 163 с.: ил. — (Школа юного инженера). — Режим доступа: по подписке. — URL: <https://biblioclub.ru/index.php?page=book&id=601815> (дата обращения: 14.04.2025). — ISBN 978-5-00101-944-2. — Текст: электронный.

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 90,0 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 75,0 баллов, 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 $K_1 = 0,15$, по предмету 2

(информатика) с весом $K_2 = 0,15$, баллы за командное решение задач инженерного тура с весом $K_3 = 0,7$.

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 3 победителя и 7 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	44,25 и выше
Призеры	От 39,55 до 43,55

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 (Р2Р). 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».