МАТЕРИАЛЫ ЗАДАНИЙ

Всероссийской междисциплинарной олимпиады школьников
«Национальная технологическая олимпиада»

по профилю
«Летающая робототехника»

2023/24 учебный год

http://ntcontest.ru

УДК 373.5.016:[629.7:681.51]
ББК 74.263.2
Л52

Авторы:
Е. Анисевич, Ф. А. Баталин, А. Е. Баталов, Э. С. Гафаров, К.Д. Кириченко, Н.Ю. Кузнецов, А.А. Никонов, Д. А. Руфин, В. Н. Трубицына

Л52 Всероссийская междисциплинарная олимпиада школьников 8-11 класса «Национальная технологическая олимпиада». Учебно-методическое пособие
Том 14 **Летающая робототехника**
—М.: ООО «ВАШ ФОРМАТ», 2024. — 186 с.

ISBN 978-5-00147-604-7

Данное пособие разработано коллективом авторов на основе опыта проведения всероссийской междисциплинарной олимпиады школьников 8-11 класса «Национальная технологическая олимпиада» в 2023/24 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 2023/24 учебный год с ответами, подробными решениями и комментариями. Пособие адресовано учащимся 8–11 классов, абитуриентам, школьным учителям, наставникам и преподавателям учреждений дополнительного образования, центров молодежного и инновационного творчества и детских технопарков.
Методические материалы также могут быть полезны студентам и преподавателям направлений, относящихся к группам:
09.00.00 Информатика и вычислительная техника
12.00.00 Фотоника, приборостроение, оптические и биотехнические системы и технологии
15.00.00 Машиностроение
23.00.00 Техника и технологии наземного транспорта
24.00.00 Авиационная и ракетно-космическая техника
25.00.00 Аэронавигация и эксплуатация авиационной и ракетно-космической техники
27.00.00 Управление в технических системах

ISBN 978-5-00147-604-7

УДК 373.5.016:[629.7:681.51]
ББК 74.263.2

Оглавление

Введение

Национальная технологическая олимпиада

Всероссийская междисциплинарная олимпиада школьников «Национальная технологическая олимпиада» (далее — 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».

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.

Всероссийская междисциплинарная олимпиада школьников 8–11 класса «Национальная технологическая олимпиада» — это командная инженерная Олимпиада, позволяющая школьникам работать в 41-м инженерном направлении. 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 девятом сезоне (2023–24 гг.) самых масштабных в России командных инженерных соревнованиях подали более 141 тысячи школьников и студентов из всех регионов страны и семи зарубежных государств: Азербайджана, Белоруссии, Казахстана, Киргизии, Молдовы, Узбекистана и Черногории. Общий охват олимпиады с 2015 года превысил 660 000 участников. https://journal.kruzhok.org/tpost/pggs3bp7y1-tehnologicheskaya-podgotovka-inzhenernih

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

- 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 и ведущими экспертами, а также найти единомышленников.

Ценности НТО

Национальная технологическая олимпиада — командные инженерные соревнования для школьников и студентов. Особое пространство Олимпиады создают общие ценности и смыслы, которые предлагается разделять всем: участникам, организаторам, наставникам, экспертам.

Основа всей олимпиады — это современное технологическое образование как новый уклад жизни в современном мире. Этот уклад подразумевает доступность качественного образования для каждого заинтересованного человека, возможность постепенно и непрерывно учиться и развиваться, совместно создавать среду, в которой гуманитарное знание и новые технологии взаимно дополняют друг друга. Это идеал будущего общества. Участники Олимпиады уже сейчас попадают в такое будущее.

Как организаторы мы надеемся, что принципы, заложенные в основу НТО, станут общими принципами для всех, кто имеет отношение к Олимпиаде.

Решать прикладные задачи, нацеленные на умножение общественного блага

В соревнованиях и подготовке к ним мы адаптируем реальные задачи современной науки и производства к знаниям и навыкам, которые могут освоить школьники и студенты. Задачи имеют прикладное значение для людей и не оторваны от реальности. Мы стремимся к тому, чтобы участники понимали, для чего нужно решать такие задачи, кому в мире станет лучше, если они будут решаться системно и профессионально. Ценность Олимпиады заключается в том, что здесь можно попробовать себя в этом, и найти единомышленников для решения подобных задач в будущем.

Создавать, а не только потреблять

Создание новых решений мы ставим выше стремления потреблять уже созданное. Создание ценности для других ставим выше поиска личной выгоды. Это не значит, что нужно забыть о себе и самоотверженно посвятить всю свою жизнь делу технологического прогресса. Но творчество всегда приносит большую радость, чем потребление. Это относится и ко всей олимпиаде.

Олимпиада — это общее дело организаторов, партнеров и участников. Способность принимать проблемы олимпиады как свои и пытаться решить их ценнее для творческого человека, чем желание найти недостатки в работе других.

Работать в команде

Способность работать в команде — это не только эффективная стратегия действия в современном мире. Работа в команде не отрицает наличия свободной воли каждого конкретного участника, его значимости и права на собственное мнение. Но в сообща мы стремимся достигнуть общей цели, опираясь на взаимное уважение всех участников, учитывая интересы и слабые и сильные стороны каждого.

Команды формируют целые сообщества, которые имеют сходные цели и ценности и могут очень многое, поскольку сильные горизонтальные связи помогают реализовывать самые дерзкие и амбициозные задачи. Это то, что нужно для технологического развития. Мы заняты построением такого сообщества и надеемся, что вы захотите стать его частью.

Осваивать и ответственно развивать новые технологии

Сообщество Национальной технологической олимпиады — часть Кружкового движения НТИ. Это прежде всего сообщество людей, увлеченных современными технологиями. Нас всех объединяет стремление разобраться в них, создать что-то новое и найти таких же увлеченных единомышленников.

Мы — часть сообщества технологических энтузиастов, и для нас границы возможностей технологий всегда подвижны. Именно поэтому просим не забывать об этике инженера и ученого, ответственности за свои изобретения перед людьми, которых это касается. Творя новое, не навреди!

Играть честно и пробовать себя

Мы признаем, что победа в соревнованиях важна и нужна. Но утверждаем, что для победы не все средства хороши и цель не является оправданием для грязной игры. Победа должна быть заслужена в рамках правил, единых для всех. Человек, который играет честно, не будет списывать, интриговать, подставлять других и заниматься прочей нездоровой конкуренцией.

Человек, который играет честно, — уважает себя, свою команду и соперников. Он принимает правила игры и в заданных рамках доказывает право на победу.

Мы бережем пространство Олимпиады как безопасное для всех участников. Это помогает искать себя, и при этом не бояться пробовать новые задачи, определять свой дальнейший путь, учиться на ошибках и каждый год становиться более сильным и подготовленным.

Быть человеком

Соревнования — это очень сложный и эмоционально насыщенный процесс. Чтобы он приносил радость и пользу всем, мы призываем всех участников вести себя порядочно и думать не только о себе.

Вежливость, эмпатия и забота — вот что сделает процесс комфортным и полезным для всех. Мы ценим уважение труда каждого человека и его позиции, бережное отношение к работе и жизни каждого. И просим отказаться от токсичной оценочной критики — она не решит ваши проблемы, а сделает хуже вам, другому и всей

Олимпиаде в целом.

Человек, который остается человеком, умеет признавать ошибки и отвечать за слова и дела перед другими. Здесь это ценят. Встав перед альтернативой между сиюминутной выгодой, капризом и общей целью соревнования — человек выберет последнее и поможет другим, организаторам и участникам, поддержать эту цель.

Важное замечание. Этот текст — живое выражение смыслов и ценностей Национальной технологической олимпиады. Он будет меняться вместе с развитием нашего сообщества. Авторы с благодарностью примут помощь от всех, кто чувствует сопричастность ценностям и готов включиться в их доработку.

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 направлениям:

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

В 2023/24 учебном году 28 профилей НТО включены в Перечень олимпиад школьников, 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, что 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 старшеклассников проводится в три этапа:

- Первый отборочный этап — заочный индивидуальный. На данном этапе участникам предлагаются задачи по двум предметам, соответствующим тому или

иному профилю, а также задания, формирующие теоретические знания и представления по направлениям выбранных профилей.

- Второй отборочный этап — заочный командный. На данном этапе участникам предлагаются индивидуальные компетентностные и командные задачи, связанные с направлением выбранного профиля.
- Заключительный этап — очный командный. Этап представляет собой очные соревнования длительностью 5–6 дней, куда приезжают команды со всей страны, успешно справившиеся с двумя отборочными этапами, и решают комплексные прикладные инженерные задачи.

Профили НТО 2023/24 учебного года и соответствующий уровень РСОШ

Профили II уровня РСОШ

- Автоматизация бизнес-процессов
- Беспилотные авиационные системы
- Водные робототехнические системы
- Инженерные биологические системы
- Интеллектуальные робототехнические системы
- Нейротехнологии и когнитивные науки
- Технологии беспроводной связи

Профили III уровня РСОШ

- Автономные транспортные системы
- Анализ космических снимков и геопространственных данных
- Аэрокосмические системы
- Большие данные и машинное обучение
- Геномное редактирование
- Интеллектуальные энергетические системы
- Информационная безопасность
- Искусственный интеллект
- Летающая робототехника
- Наносистемы и наноинженерия
- Новые материалы
- Передовые производственные технологии
- Разработка компьютерных игр
- Спутниковые системы
- Технологии виртуальной реальности
- Технологии дополненной реальности
- Технологическое предпринимательство
- Умный город
- Фотоника
- Цифровые технологии в архитектуре
- Ядерные технологии

Профили без уровня РСОШ

- Научная медиакоммуникация
- Программная инженерия в финансовых технологиях
- Современная пищевая инженерия
- Технологическое мейкерство
- Урбанистика
- Цифровое производство в машиностроении
- Цифровой инжиниринг в строительстве
- Цифровые сенсорные системы

Новые профили без уровня РСОШ

- Инфохимия
- Квантовый инжиниринг
- Технологии компьютерного зрения и цифровые сервисы
- Цифровая гидрометеорология
- Цифровое месторождение

Обратите внимание, что в олимпиаде 2024/25 года список профилей, в т.ч. входящих в РСОШ, и уровни РСОШ — могут поменяться.

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

- учащихся технологических кружков, любители инженерных и робототехнических соревнований;
- олимпиадников, которым интересны межпредметные олимпиады;
- фанатов и адептов передовых технологий;
- школьников, участвующих в хакатонах, проектных конкурсах и школах;
- будущих предпринимателей, намеревающихся найти на Олимпиаде единомышленников для будущего стартапа;
- увлекающихся школьников, которые хотят видеть предмет шире учебника.

Познакомить школьников с НТО и ее направлениями, замотивировать принять участие в НТО можно с помощью специальных мероприятий: Урок НТО и Дни НТО. Как педагогу провести Урок НТО, или как в образовательном учреждении организовать День НТО можно познакомиться в методических рекомендациях на сайте НТО. Там же можно выбрать и скачать необходимые уроки и подборки материалов по направлениям https://nti-lesson.ru/.

Участвуя в НТО, школьники получают возможность работать с практикоориентированными задачами в области прорывных технологий, собирать команды единомышленников, включаться в профессиональное экспертное сообщество, а также заработать льготы для поступления в вузы.

У НТО есть площадки подготовки по всей стране, которые занимаются привлечением участников и проводят мероприятия по подготовке к соревнованиям. Они могут быть открыты:

- в организациях общего и дополнительного образования;
- на базе частных кружков в области программирования, робототехники и иных технологий;
- в вузах;
- технопарках

и других организациях.

Каждое образовательное учреждение, ученики которого участвуют в НТО или НТО Junior, может стать площадкой подготовки к олимпиаде, что дает возможность включиться в Кружковое движение НТИ.

На сайте НТО размещены инструкции о том, как организация может стать площадкой подготовки: https://ntcontest.ru/mentors/stat-ploshadkoi/. Условия регистрации и требования к работе площадок подготовки обновляются вместе с развитием олимпиады. Обновленная версия размещается на сайте перед началом нового цикла олимпиады.

Наставники НТО

В НТО большое внимание уделяется работе с наставниками. Наставник НТО оказывает всестороннюю поддержку участникам Олимпиады, помогая решать организационные вопросы и развивать как технические знания и компетенции, так и социальные навыки, связанные с работой в команде.

Наставником может стать любой человек, которому интересно сопровождать участников и помогать им формировать необходимые для решения технологических задач компетенции и готовиться к соревнованиям. Это может быть преподаватель школы или вуза, педагог дополнительного образования, руководитель кружка, эксперт в технологической области, представитель бизнеса и т. п. Если наставнику не хватает собственных знаний, он может привлекать коллег и внешних экспертов и

поддерживать усилия и мотивацию учеников, которые разбирают задачи самостоятельно. На данный момент сообщество наставников НТО включает в себя более 7 тысяч человек.

Главная задача наставника — выстроить комплексную структуру подготовки к Олимпиаде в течение всего учебного года. В области ответственности наставника находится поддержка мотивации участников и помощь в решении возникающих проблем. Не менее важно зафиксировать цели и ожидания от предстоящих соревнований, что поможет оценить прирост профессиональных компетенций, личных и командных навыков за время подготовки.

Примеры организационных задач, которые стоят перед наставником НТО:

- Информирование и работа с мотивацией. На этапе регистрации на Олимпиаду наставник привлекает участников, рассказывая, что такое НТО и какие преимущества она предлагает. Наставнику необходимо разобраться в устройстве НТО, этапах и расписании этапов, а также изучить профили, чтобы помочь каждому ученику выбрать наиболее перспективные и интересные для него направления.
- Формирование программы подготовки. Наставник составляет график подготовки к НТО и следит за его реализацией, руководя процессом подготовки учеников.
- Отслеживание сроков. Наставник следит за сроками проведения этапов НТО и напоминает участникам о необходимости своевременной загрузки решений на платформу.

Примеры задач наставника, связанных с непосредственной подготовкой к соревнованиям:

- Анализ компетенций участников. Наставник вместе с учениками оценивает компетенции, которые необходимы для успешного участия в НТО, выявляет нехватку знаний и навыков и отбирает материалы и задачи, которые ученикам нужно изучить и решить.
- Содержательная подготовка к первому и второму отборочному этапу. Наставник вместе с учениками изучает материалы для подготовки, рекомендованные разработчиками выбранных профилей, а также разбирает и решает задачи НТО прошлых сезонов. Рекомендуется использовать записи вебинаров, материалы и онлайн-курсы профилей.
- Содержательная подготовка к заключительному этапу. Наставник может использовать разборы задач заключительного этапа прошлых лет, а также следить за расписанием подготовительных очных и дистанционных мероприятий и рекомендовать ученикам их посещать.

Примеры задач наставника в области развития социальных навыков, связанных с развитием личной эффективности и взаимодействия с другими участниками:

- Формирование команд. Второй отборочный этап НТО проходит в командном формате. Наставник помогает ученикам сформировать эффективную команду с оптимальным распределением ролей. В ряде случаев он может содействовать в поиске недостающих участников команды, в том числе в других городах и стать наставником такой команды, коммуникация в которой осуществляется через web-сервисы.
- Отслеживание прогресса и анализ полученного опыта. Наставник проводит ре-

флексию прогресса отдельных участников и команды по результатам каждого этапа НТО и после завершения участия в соревнованиях. Это помогает участникам оценить свое движение по траектории соревнований, сильные и слабые стороны, сформулировать, каких компетенций не хватило для более высокого результата и как их можно улучшить в будущем.

- Поддержка и мотивирование участников. Наставник поддерживает интерес учеников к соревнованиям, а также помогает им сохранять высокую мотивацию, что особенно важно, если команда показала результаты хуже, чем ожидалось.
- Выстраивание индивидуальной образовательной траектории. Наставник может помочь ученикам осознанно создать собственную траекторию развития, в том числе вне НТО: подбор обучающих курсов и соревнований, выбор вуза и направления дальнейшего обучения.

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/promo. Курс формирует общие представления о работе наставников в области подготовки участников к инженерным соревнованиям.

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

Наставникам для ведения занятий с учениками предлагаются образовательные программы, разработанные на основе восьмилетнего опыта организации подготовки к НТО. В настоящий момент такие программы представлены по 10-ти передовым технологическим направлениям:

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

и др.

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.

Формируется сообщество наставников НТО. Ежегодно Кружковое движение НТИ проводит Всероссийский конкурс технологических кружков: https://konkurs.kruzhok.org, принять участие в котором может каждый наставник. По итогам конкурса кружки-участники размещаются на Всероссийской карте кружков: https://map.kruzhok.org.

В 2022 году был разработан Навигатор для наставников команд или отдельных участников НТО: https://www.notion.so/bdlv/5a1866975c2744728c2bd8ba80d21ec2.

Навигатор ориентирован на начинающих наставников и помогает погрузиться в работу с НТО. Опытным наставникам Навигатор может быть полезен как сборник важных рекомендаций и статей:

- Смогут ли мои ученики принять участие в НТО.
- Как наставнику зарегистрироваться в НТО.
- Как помочь участникам выбирать профили.
- Что можно успеть сделать, если я и мои ученики начнем участвовать с нового учебного года.
- Как убедить руководство включиться в НТО.
- Что важно знать, начиная подготовку школьников.
- Как организовать подготовку.
- Как проводить рефлексию.
- Как мотивировать участников.
- Как работать с командой участников НТО.

Организаторы Олимпиады также оказывают экспертно-методическую поддержку сообществу наставников. Были разработаны методические рекомендации для наставников: «Технологическая подготовка инженерных команд»: https://journal.kruzhok.org/tpost/pggs3bp7y1-tehnologicheskaya-podgotovka-inzhenernih. Рассмотрены особенности подготовки к 5-ти направлениям:

- Большие данные.
- Машинное обучение.

- Искусственный интеллект.
- Спутниковые системы.
- Летающая робототехника.

Для наставников НТО разработан и постоянно пополняется страница с материалами для профессионального развития: http://clc.to/for-mentor.

Летающая робототехника

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

Работа с летающей робототехнической платформой с открытым исходным кодом позволяет решать задачи в областях разработки методов управления, позиционирования, сбора и обработки информации при помощи технического зрения, а также затрагивает поиск новых конструктивных решений для летающих робототехнических систем, их подсистем и отдельных модулей. Разработка решений автоматизированного сбора и анализа информации, впоследствии может тиражироваться на сервисные беспилотники и решать повседневные задачи, так как основное назначение летающих робототехнических платформ — выполнение полезной работы для людей и оборудования.

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

Для погружения в тематику профиля организаторами профиля разработаны:

- урок НТО, который знакомит учащихся с понятием компьютерного зрения, библиотекой алгоритмов OpenCV и предлагает написать программу на языке Python для распознавания изображения с камеры персонального компьютера;
- видеокурс по сборке, настройке, программированию квадрокоптера и 3D моделированию.

В первом отборочном дистанционном этапе профиля участники решают задачи инженерного и предметного туров. Задачи инженерного тура знакомят участников с тематикой профиля и вовлекают в изучение технологий профиля. Задачи предметного тура определяют общий уровень подготовки участников по школьным предметам: информатика и физика.

К участию во втором этапе допускаются все участники, набравшие в первом отборочном этапе баллы выше установленного организаторами порогового значения. Для участия в данном этапе необходимо объединиться в команды в количестве 3–4 человека согласно имеющимся компетенциям:

Роль 1. Инженер-программист (Python) — написание кода для автономного полета квадрокоптера, работа с сервером, разработка алгоритма безопасного полета квадрокоптера.

Роль 2. Инженер-программист (C++, Python) — алгоритмы компьютерного зрения для реализации автономных миссий квадрокоптера, машинное обучение. Работа в связке с ролью 1.

Роль 3. Инженер-техник — моделирование и изготовление устройства, тестирование, техобслуживание и пилотирование квадрокоптера, работа с технической документацией.

Роль 4. Капитан/лидер команды — работа с системами построения карты в

`Rviz`, осуществление общего руководства работой команды, распределение обязанностей и контроль соблюдения дедлайнов. Рекомендовано совмещение данной роли с другими ролями.

Для решения и автоматической проверки задач используется онлайн-платформа. Задачи второго этапа подразделяются на два блока. Первый блок предназначается для проверки имеющихся компетенций и определения своей роли в команде. Задачи направлены на изучение строения и систем управления БПЛА, программирование на языке Python, работу с платформой ROS, сервером, компьютерным зрением (`OpenCV`), чтение чертежей и 3D моделирование, работу с алгоритмами для распознавания образов и изображений с помощью ИИ.

Второй блок состоит из комплексного командного задания, для решения которого необходимы навыки каждого участника команды. Командное задание позволяет участникам апробировать свои решения в симуляторе: инженерам — настроить квадрокоптер и подготовить виртуальный мир, а программистам — написать программный код для автономного полета квадрокоптера. Для успешного выполнения заданий второго отборочного этапа участникам необходимо правильно распределить задачи, а также наладить систему планирования и коммуникации внутри команды.

В процессе решения задачи второго этапа участники сталкиваются не только с отдельными элементами задачи предстоящего заключительного этапа, но таким образом заранее отрабатывают ее ключевые блоки. В процессе работы над решением участники осознают свои сильные и слабые стороны, а также постепенно погружаются в формат командной работы.

Разработчиками профиля проводятся организационные вебинары и образовательные мастер-классы, направленные на подготовку к решению задач второго и заключительного этапов олимпиады. Получить помощь в решении текущих вопросов участники могут в онлайн-режиме в чате технической поддержки платформы Клевер и чате участников профиля в Telegram.

Задача заключительного этапа посвящалась разработке комплексной системы автономного мониторинга строительной площадки с использованием квадрокоптера, включая:

- разработку программного кода для осуществления мониторинга и анализа застройки нового микрорайона:
- распознавание зданий и расчет их этажности;
- выявление качественных и количественных характеристик прокладки дорожного полотна;
- разработку программного кода для визуализации результатов мониторинга на карте в режиме реального времени;
- работу с алгоритмами распознавания образов и изображений с помощью ИИ для контроля строительной спецтехники на территории застройки;
- передачу отчетов о результатах мониторинга в режиме реального времени с квадрокоптера на сервер;
- разработку дронопорта — устройства для безопасного взлета, посадки и подзарядки квадрокоптера (разработка самого устройства, конструкторской документации и программного кода для его автономной работы);
- объединение всех составных частей в единую систему для запуска автономной миссии квадрокоптера.

Эксплуатацию и техническое обслуживание оборудования (квадрокоптер и 3D-принтер) команды осуществляют самостоятельно. Работа с лазерным станком осуществлялась совместно со специалистом площадки.

В целях обеспечения равных условий заключительного этапа командное задание было направлено на решение одной задачи, которая была разбита на несколько подзадач (полетных миссий) с постепенным повышением уровня сложности. Таким образом, ежедневно участники получили новое, усложненное задание, включающее в себя неизвестный ранее элемент. Каждая подзадача имела отдельные критерии оценки успешности полетной миссии, публиковалась и оценивалась в день ее выполнения. Команда-победитель определялась по наибольшему количеству баллов за сумму трех зачетных попыток (наиболее успешной автономной миссии) и результатам инженерного задания.

Важной частью заключительного этапа является индивидуальный предметный тур, в ходе которого участники решали задачи по физике и информатике, на котором участники могут продемонстрировать свои предметные знания и уровень подготовки.

Решения, полученные командами в практическом туре заключительного этапа, имеют возможность масштабирования на большие производственные дроны, так как использующиеся технологии идентичны.

Навыки и знания приобретенные в процессе участия в профиле «Летающая робототехника» закладывают основу для дальнейшей инженерной, исследовательской и конструкторской деятельности в различных областях. Многие участники профиля продолжают разрабатывать свои проекты, участвовать в конкурсах и соревнованиях.

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

Работа наставника НТО на первом отборочном этапе

На первом отборочном этапе НТО участникам предлагаются задачи по предметам, соответствующим выбранным профилям. Для подготовки к первому отборочному этапу Олимпиады наставник может использовать следующие рекомендуемые форматы и мероприятия:

- Разбор задач первого отборочного этапа НТО прошлых лет.
- Мини-соревнования по решению задач предметных олимпиад муниципального уровня.
- Углубленные занятия по разделам предметов в соответствии с рекомендациями разработчиков профилей.

Для проверки, самостоятельного решения или проведения мини-соревнований могут использоваться предметные курсы НТО на платформе Stepik. Также возможно привлечение других преподавателей-предметников для проведения занятий в случае, если у наставника недостаточно компетенций в области предметных олимпиад.

Инженерный тур состоит из курса или теоретических материалов, погружающих участников в тематику профиля, и теоретических и практических заданий, как правило связанных с теорией.

Первый отборочный этап

Предметный тур. Информатика и программирование

Первая волна. Задачи 8–11 класса

Задача П.1.1.1. Поздравление в конверте (10 баллов)

Темы: задачи для начинающих.

Условие

Алиса хочет поздравить Боба с днем рождения. Она взяла прямоугольный лист бумаги размера $a \times b$ и написала на нем поздравление в стихах. У Алисы есть красивый конверт тоже прямоугольной формы размером u на v. Алиса хочет положить свое поздравление в этот конверт. Однако лист может не войти в конверт. В этом случае Алиса готова сложить лист пополам вдоль одной из сторон, чтобы поместить его в конверт. Обратите внимание, что Алиса может сделать не более одного сгиба. Лист можно поворачивать, но одна из сторон листа должна быть параллельной одной из сторон конверта.

Напишите программу, которая определит, сможет ли Алиса уложить лист в конверт по указанным правилам.

Мы будем считать, что лист входит в конверт, если сторона листа будет строго меньше соответствующей стороны конверта.

Формат входных данных

На вход в первой строке подается два натуральных числа a и b — длины сторон листа. Во второй строке на вход подаются натуральные числа u и v — размеры конверта. Все числа не превосходят 1000.

В языке Python прочитать два целых числа, записанных в одной строке можно, используя следующий код.

```
a, b = map(int, input().split())
```

Формат выходных данных

Если поздравление можно вложить в конверт без сгиба, то следует вывести число 0. Иначе, если поздравление можно вложить в конверт сделав один сгиб, то следует вывести число 1. В остальных случаях следует вывести число -1.

Методика проверки

Программа проверяется на 20-ти тестах. Прохождение каждого теста оценивается в 0,5 балла. Тесты из условия задачи при проверке не используются.

Примеры

Пример №1

Стандартный ввод
120 200 130 250
Стандартный вывод
0

Пример №2

Стандартный ввод
120 200 110 130
Стандартный вывод
1

Пример №3

Стандартный ввод
400 100 200 150
Стандартный вывод
-1

Решение

В этой задаче требуется аккуратно написать требуемые по условию логические выражения. Их запись существенно упростится, если упорядочить длины так, чтобы всегда имел место инвариант $a \leqslant b$ и $u \leqslant v$.

Тогда для проверки возможности вложения листа в конверт меньшую сторону листа следует всегда сравнивать с меньшей стороной конверта.

Для проверки возможности вложения листа в конверт после сгиба надо поочередно поделить меньшую и большую сторону на два и использовать такую же проверку. Следует не забыть, что после деления длины большей стороны на два, она может стать меньше, чем меньшая сторона.

Пример программы-решения

Ниже представлено решение на языке Python 3.

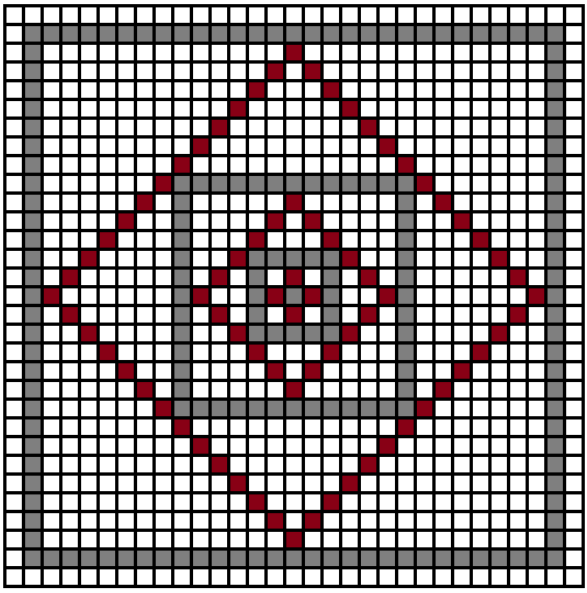
```python
a, b = sorted(list(map(int, input().split())))
u, v = sorted(list(map(int, input().split())))
if a<u and b<v:
    print(0)
elif a/2<u and b<v or a<u and b/2<v or b/2<u and a<v:
    print(1)
else:
    print(-1)
```

Задача II.1.1.2. Квадраты (15 баллов)

Темы: задачи для начинающих, комбинаторика.

Условие

У Алисы и Боба есть прямоугольный лист бумаги в клеточку. Они по очереди рисуют квадраты, закрашивая некоторые из клеточек. Алиса рисует квадраты, ориентируя их вдоль сторон листа, а Боб — под углом в 45 °. При этом Алиса рисует первый квадрат из одной клеточки, а каждый новый квадрат описывается вокруг предыдущего. Для лучшего понимания смотрите рисунок. Серым цветом на нем нарисовано четыре квадрата Алисы, а коричневым нарисовано три квадрата Боба. Всего семь квадратов.



Алиса и Боб вместе нарисовали n квадратов. Напишите программу, которая определит, сколько клеточек на листе бумаги будет закрашено.

Формат входных данных

На вход подается единственное натуральное число n — количество квадратов, $1 \leqslant n \leqslant 100$.

Формат выходных данных

Выведите одно натуральное число — суммарное количество закрашенных клеточек.

Методика проверки

Программа проверяется на 15-ти тестах. Прохождение каждого теста оценивается в 1 балл. Тесты из условия задачи при проверке не используются.

Примеры

Пример №1

Стандартный ввод
7
Стандартный вывод
253

Пример №2

Стандартный ввод
2
Стандартный вывод
5

Решение

Заметим, что по условию задачи количество квадратов не превышает 100, поэтому посчитаем, из скольких клеточек состоит каждый квадрат, и просуммируем полученные значения в цикле.

Обратим внимание на количество клеточек на стороне квадрата. Легко заметить и доказать, что если предыдущий квадрат был серый, и его сторона содержала k клеточек, то сторона следующего за ним коричневого квадрата будет содержать $k+1$ клеточку. Если же предыдущий квадрат был коричневым, и его сторона содержала k клеточек, то сторона следующего серого квадрата будет содержать $2k+1$ клеточку.

В приведенной ниже программе в переменной `ln` хранится количество клеточек, из которых состоит одна сторона текущего квадрата. В переменной `ans` накапливается сумма.

Пример программы-решения

Ниже представлено решение на языке Python 3.

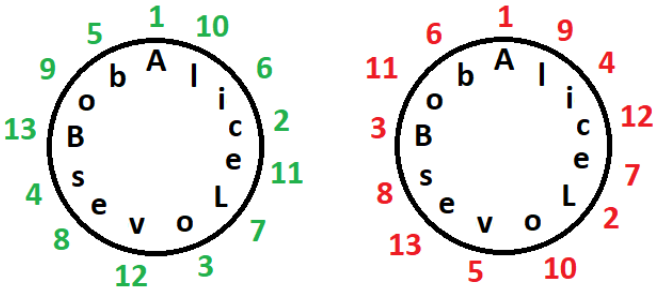
```
n = int(input())
ans = 1
ln = 1
for i in range(n):
    ans += (ln - 1) * 4
    if i%2==0:
        ln += 1
    else:
        ln = 2 * ln + 1
print(ans)
```

Задача П.1.1.3. Две строки (15 баллов)

Темы: строки, структуры данных.

Условие

У Алисы и Боба есть секретная информация, которая записана в виде строки s. Чтобы сохранить секрет Алиса сделала перестановку символов в строке по следующему правилу. Она записала все символы строки по кругу, потом записала в ответ первый символ и далее стала выписывать символы из кольца через два.



Рассмотрим пример. Пусть секретная строка — *AliceLovesBob*. Алиса запишет эту строку, как показано на рисунке. Далее она выпишет первую букву строки A, пропустит два следующих символа, напишет букву c, пропустит еще два символа, напишет букву o и так далее по кругу. В результате у нее будет записана строка *AcosbiLeolevB*. Номера на рисунке слева соответствуют последовательности перечисления букв Алисой.

Боб шифрует эту же строку таким же алгоритмом, однако, в отличие от Алисы, он пропускает по четыре буквы, а не по две. Номера на рисунке справа соответствуют последовательности перечисления букв Бобом. Таким образом Боб получит строку *ALBivbeslooce*.

Боб был небрежен и потерял зашифрованную строку, однако у него есть строка, зашифрованная Алисой. Напишете программу, которая по зашифрованной строке Алисы найдет зашифрованную строку Боба.

Формат входных данных

На вход подается одна непустая строка — шифр Алисы. Строка состоит только из строчных и заглавных символов латиницы. Длина строки не превосходит 1000 и не кратна трем и пяти.

Формат выходных данных

Выведите одну строку — шифр Боба.

Методика проверки

Программа проверяется на 15-ти тестах. Прохождение каждого теста оценивается в 1 балл. Тест из условия задачи при проверке не используется.

Примеры

Пример №1

Стандартный ввод
`AcosbiLeolevB`
Стандартный вывод
`ALBivbeslooce`

Решение

Решение задачи состоит из двух частей. В первой части требуется восстановить исходную строку по коду Алисы. Для этого можно сделать список из символов нужной длины и каждый i-тый символ из кодовой строки записывать в позицию $3i \mod n$, где n — длина строки. Операция взятия остатка от деления здесь используется для движения по кольцу.

Во второй части при помощи аналогичного приема полученная строка кодируется по обратной формуле с множителем 5.

Пример программы-решения

Ниже представлено решение на языке Python 3.

```
s = input()
n = len(s)
tmp = [''] * n
ans = ''
for i in range(n):
    tmp[(i * 3) % n] = s[i]
for i in range(n):
    ans += tmp[(i * 5) % n]
print(ans)
```

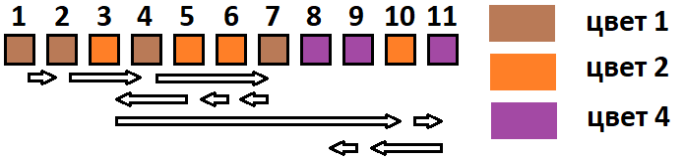
Задача П.1.1.4. Покраска кубиков (30 баллов)

Темы: реализация, сортировки, структуры данных, динамическое программирование, комбинаторика.

Условие

На ленте в один ряд расставлено n кубиков. Каждый кубик необходимо покрасить в определенный цвет. Все цвета пронумерованы числами от 1 до k. Покраска выполняется роботом, который может перемещаться от одного кубика к другому и красить один выбранный кубик в определенный цвет. Конструктивно робот устроен так, что он может сначала красить кубики в цвет номер 1, затем в цвет номер 2 и так далее в порядке возрастания. Вернуться к цвету с меньшим номером после того, как был выбран цвет с большим номером, нельзя.

Среди всего прочего робот тратит время на перемещение между кубиками. Будем считать, что перемещение между двумя соседними кубиками занимает ровно одну с. Требуется составить последовательность действий для робота, в которой время, затраченное на перемещение между кубиками, будет минимально возможным.



Рассмотрим пример на схеме. Имеется 11 кубиков, которые надо покрасить в три цвета с номерами $1, 2, 4$. Робот начнет движение от кубика номер 1 направо к кубику номер 2 (1 с), далее к кубику с номером 4 (2 с), и наконец к кубику номер 7 (3 с). После этого робот поменяет цвет. Будет выгоднее, если робот начнет сначала двигаться влево. Он пройдет к кубику номер 6 (1 с), далее к кубику номер 5 (1 с), далее к кубику номер 3 (2 с). После этого он развернется и пойдет к кубику номер 10 (7 с). Далее робот сменит цвет на 4, так как нет кубиков, которые требуется красить в цвет 3, и пойдет направо к кубику номер 11 (1 с). После этого он развернется и пойдет к кубику номер 9 (2 с) и наконец к кубику номер 8 (1 с). В этот момент робот остановит работу, затратив на перемещения суммарно 21 с.

Обратите внимание, что по условию задачи робот может выбрать цвет 4 только после цвета 2. Существуют другие возможные маршруты движения робота, но они займут больше времени.

Напишите программу, которая найдет минимально возможное суммарное время перемещения робота между кубиками до момента пока все они не будут покрашены. Изначально робот находится у кубика номер 1.

Формат входных данных

На вход в первой строке подается два натуральных числа n и k — количество кубиков и количество цветов, $1 \leqslant n, k \leqslant 100000$. Во второй строке на вход подается n натуральных чисел $c_1, c_2, \ldots, c_n$, где c_i — требуемый цвет i-того кубика, $1 \leqslant c_i \leqslant k$.

Формат выходных данных

Выведите одно число — минимально возможное время перемещения между кубиками.

Методика проверки

Программа проверяется на 60-ти тестах. Прохождение каждого теста оценивается в 0,5 балла. Тест из условия задачи при проверке не используется. Ниже в таблице приведены возможные тестовые случаи.

Тестовый случай	**Номера тестов**
$n = k$; $n \leqslant 1000$; все c_i различны.	1–8
$n = k$; $n \leqslant 100000$; все c_i различны.	9–20
$k = 3$; $n \leqslant 1000$	21–24
$k = 4$; $n \leqslant 1000$	25–30
$k = 5$; $n \leqslant 1000$	31–40
$n \leqslant 1000$	41–50
$n \leqslant 100000$	51–60

Примеры

Пример №1

Стандартный ввод
`11 4` `1 1 2 1 2 2 1 4 4 2 4`
Стандартный вывод
`21`

Решение

Данная задача может быть решена методом динамического программирования. Пусть, начиная покраску кубиков в цвет i, робот находится в некоторой точке x_i, причем самый левый из кубиков этого цвета находится в точке l_i, а самый правый — в r_i. Тогда оптимальным будет один из двух вариантов: из точки x_i пойти в r_i, а потом в l_i или сначала пойти в l_i, а потом в r_i. Таким образом, робот закончит покраску кубиков определенного цвета либо в точке l_i, либо в r_i, причем в первом случае время перемещения робота увеличится на $|x_i - r_i| + r_i - l_i$ с, а во втором — на $|x_i - l_i| + r_i - l_i$ с.

Обозначим за $f_l(i)$ и $f_r(i)$ — оптимальное время покраски кубиков в первые i цветов при условии, что робот остановится в точке l_i и r_i соответственно. Тогда можно определить следующие формулы:

$$
\begin{aligned}
&f_l(0) = 0;\\
&f_r(0) = 0;\\
&f_l(i) = r_i - l_i + \min(f_l(i-1) + |l_{i-1} - r_i|, f_r(i-1) + |r_{i-1} - r_i|);\\
&f_r(i) = r_i - l_i + \min(f_l(i-1) + |l_{i-1} - l_i|, f_r(i-1) + |r_{i-1} - l_i|).
\end{aligned}
$$

В данных формулах находится время для двух вариантов перемещения: от самого левого и от самого правого кубика предыдущего цвета, после чего из полученных значений выбирается минимальное.

Программа будет содержать два цикла. В первом цикле для каждого цвета определяется местоположение самого левого и самого правого кубика, а во втором — выполняются вычисления по формулам.

При реализации программы надо учесть, что некоторые цвета могут отсутствовать. Поэтому в переменные `l` и `r` записываются координаты самого левого и правого кубика для предыдущего цвета.

Пример программы-решения

Ниже представлено решение на языке Python 3.

```
n, k = map(int, input().split())
left = [n + 1] * (k + 1)
right = [0] * (k + 1)
i = 1
for col in map(int, input().split()):
    left[col] = min(left[col], i)
    right[col] = max(right[col], i)
    i += 1
l, r, fl, fr = 1, 1, 0, 0
for i in range(1, k + 1):
    if right[i] > 0:
        dist = right[i] - left[i]
        tl = dist + min(fl + abs(l - right[i]),
                        fr + abs(r - right[i]))
        tr = dist + min(fl + abs(l - left[i]),
                        fr + abs(r - left[i]))
        l, r, fl, fr = left[i], right[i], tl, tr
print(min(fl, fr))
```

Задача II.1.1.5. Обработка запросов (30 баллов)

Темы: реализация, структуры данных, два указателя, двоичный поиск.

Условие

Алиса проектирует вычислительную систему, предназначенную для обработки большого числа однотипных запросов. Проектируемая система будет содержать некоторое количество одинаковых процессоров. В каждый момент времени каждый процессор может быть либо свободен, либо занят обработкой ровно одного запроса. Продолжительность обработки является одинаковой для всех запросов и составляет s мс. Система должна работать в режиме реального времени, то есть каждый поступивший запрос должен незамедлительно передаваться на обработку любому свободному процессору.

Алиса хочет понять, сколько процессоров должна содержать вычислительная система. Для этого она собрала статистические данные о работе подобных систем в прошлом. Каждый набор данных содержит n чисел $t_1, t_2, \ldots, t_n$, где t_i — момент

поступления запроса с номером i. Числа в наборе могут повторяться, однако они упорядочены по неубыванию. Если некоторый процессор приступил к обработке некоторого запроса в момент t_i, то в момент $t_i + s$ он сможет начать обрабатывать новый запрос.

Набор может содержать достаточно большой объем данных, поэтому от вас требуется написать программу, которая определит, какое минимальное число процессоров должно быть в вычислительной системе, чтобы все запросы были обработаны в момент их поступления.

Формат входных данных

На вход в первой строке подается два натуральных числа n и s — количество запросов и время обработки одного запроса, $1 \leqslant n \leqslant 200000$, $1 \leqslant s \leqslant 10^9$. Во второй строке записаны целые неотрицательные числа t_1, t_2, …, t_n, задающие моменты времени поступления запросов, $0 \leqslant t_1 \leqslant t_2 \leqslant \cdots \leqslant t_n \leqslant 10^9$.

Формат выходных данных

Вывести одно число — минимальное количество процессоров, которое позволит обработать все запросы в момент их поступления.

Методика проверки

Программа проверяется на 30-ти тестах. Прохождение каждого теста оценивается в 1 балл. Тесты из условия задачи при проверке не используются. Ниже в таблице приведены возможные тестовые случаи.

Тестовый случай	**Номера тестов**
$n \leqslant 1000$; для всех t_i выполняется одно из двух условий: либо $t_i = t_{i+1}$, либо $t_i + s \leqslant t_{i+1}$.	1–5
$n \leqslant 1000$	6–15
$n \leqslant 200000$	16–30

Примеры

Пример №1

Стандартный ввод
`9 30` `90 90 90 120 120 120 120 200 200`

Стандартный вывод
`4`

Пример №2

Стандартный ввод
10 30 0 25 110 125 125 130 140 140 140 155
Стандартный вывод
6

Пояснения к примерам

Первый пример соответствует первому тестовому случаю. В момент времени 120 приходит сразу четыре запроса, для обработки которых потребуется четыре процессора.

Расписание выполнения запросов во втором примере можно представить в следующей таблице.

Время поступления запроса	**Время завершения обработки запроса**	**Номер процессора**
0	30	1
25	55	2
110	140	1
125	155	2
125	155	3
130	160	4
140	170	1
140	170	5
140	170	6
155	175	2

Пять процессоров для своевременной обработки всех запросов будет уже недостаточно.

В задаче требуется найти такое число k, что для всех $i \leqslant n - k$ выполняется неравенство $t_{i+k} - t_i \leqslant s$. Действительно, пусть это утверждение имеет место. Тогда процессор, выполнявший задание с номером i, всегда сможет выполнить задание с номером $i+k$, и все задания можно выполнить своевременно, выдавая их процессорам по кругу. С другой стороны, пусть это утверждение не выполняется, то есть найдется такой номер j, что $t_{j+k} - t_j < s$. Тогда в момент времени t_{j+k} все k процессоров будут заняты исполнением запросов с номерами от j до $j + k - 1$, и все запросы не смогут быть выполнены своевременно.

Найти число k, для которого выполняется указанное утверждение можно при помощи двоичного поиска или метода двух указателей. Вариант решения с использованием двух вложенных циклов наберет лишь часть баллов, так как будет превышать ограничение по времени работы.

Метод двух указателей основан на использовании двух переменных `left` и `right`, которые используются в качестве индексов в массиве. Цикл строится таким образом, чтобы для каждого значения `left` находить минимальное значение `right` при котором `t[right]-t[left]>=s`.

Пример программы-решения

Ниже представлено решение на языке Python 3.

```
n, s = map(int, input().split())
t = list(map(int, input().split()))
ans = 1
left = 0
right = 0
while right < n:
    if t[right] - t[left] >= s:
        left += 1
    else:
        right += 1
    ans = max(ans, right - left)
print(ans)
```

Вторая волна. Задачи 8–11 класса

Задача II.1.2.1. Три мешка конфет (10 баллов)

Темы: задачи для начинающих, реализация.

Условие

Алиса и Боб получили в подарок три мешка конфет и они хотят поделить их поровну. Для этого Алиса возьмет некоторое количество конфет из каждого мешка, а остальные конфеты отдаст Бобу. Возможно, что из некоторого мешка Алиса возьмет все конфеты или не возьмет ни одной. Известно, что суммарное количество конфет является четным числом.

Напишите программу, которая определит, сколько конфет Алиса должна взять из каждого мешка, чтобы у нее оказалось ровно половина всех конфет. Программа может вывести любой правильный ответ.

Формат входных данных

На вход в первой строке подается три натуральных числа a, b и c — количество конфет в каждой кучке, $1 \leqslant a, b, c \leqslant 1000$.

В языке Python прочитать три целых числа, записанных в одной строке можно, используя следующий код.

```
a, b, c = map(int, input().split())
```

Формат выходных данных

Выведите в одной строке через пробел три целых неотрицательных числа — количество конфет, которое возьмет Алиса из каждого мешка.

Методика проверки

Программа проверяется на 20 тестах. Прохождение каждого теста оценивается в 0,5 балла. Тест из условия задачи при проверке не используется.

Примеры

Пример №1

Стандартный ввод
10 5 5
Стандартный вывод
7 3 3

Пояснения к примерам

Ответ 7 3 0 удовлетворяет всем требованиям. Но существует и множество других вариантов, например, 7 2 1 или 0 5 5.

Решение

Существует много способов составить требуемый набор чисел. Например, можно заметить, что если сумма трех чисел четная, то хотя бы одно из слагаемых тоже обязательно четное. Тогда это слагаемое можно поделить на два, еще одно поделить на два с округлением вниз, а последнее — с округлением вверх.

Пример программы-решения

Ниже представлено решение на языке Python 3.

```
a, b, c = map(int, input().split())
if a % 2 == 0:
    print(a // 2, b // 2, (c + 1) // 2)
else:
    print((a + 1) // 2, b // 2, c // 2)
```

Задача II.1.2.2. Трехцветная сортировка (15 баллов)

Темы: задачи для начинающих, структуры данных.

Условие

У Алисы есть упорядоченный набор карточек, каждая из которых раскрашена в один из трех цветов: красный, зеленый, синий. Кроме того, на каждой карточке записано некоторое натуральное число. Алиса хочет выполнить сортировку чисел, чтобы сначала шли все числа на красных карточках, далее — на зеленых и наконец — на синих. При этом взаимное расположение карточек одного цвета не должно измениться. Например, если в исходном наборе было две красных карточки с числами 20

и 10, причем карточка с числом 20 располагалась раньше, чем карточка с числом 10, то после упорядочивания 20 по-прежнему должна находиться раньше, чем 10.

Напишите программу, которая отсортирует карточки в требуемом порядке.

Формат входных данных

На вход в первой строке подается последовательность символов $c_1, c_2, \cdots, c_n$, где c_i задает цвет i-той карточки и может принимать одно из трех значений r, g или b. Каждый из символов обозначает определенный цвет: r — красный, g — зеленый, b — синий. Символы записаны без пробелов и других разделителей, $1 \leqslant n \leqslant 1000$.

Во второй строке записана последовательность натуральных чисел $a_1, a_2, \cdots, a_n$, где a_i задает число, записанное на i-той карточке. Все числа различны и не превосходят n.

Формат выходных данных

В одной строке через пробел вывести требуемую последовательность чисел после сортировки.

В языке Python для вывода чисел в цикле на одной строке через пробел можно использовать следующую команду.

```
print(x, end=' ')
```

Методика проверки

Программа проверяется на 15-ти тестах. Прохождение каждого теста оценивается в 1 балл. Тесты из условия задачи при проверке не используются.

Примеры

Пример №1

Стандартный ввод
bbb 3 1 2
Стандартный вывод
3 1 2

Пример №2

Стандартный ввод
rgrg 4 1 2 3
Стандартный вывод
4 2 1 3

Пример №3

Стандартный ввод
brrg 1 2 3 4
Стандартный вывод
2 3 4 1

Пояснения к примерам

В первом примере все карточки одного цвета, поэтому упорядочивать нечего.

Во втором примере карточки 4 и 2 красного цвета, поэтому они окажутся в начале, сохранив взаимное расположение. Карточки 1 и 3 зеленого цвета, поэтому они сдвинутся в конец, также сохранив взаимное расположение.

В третьем примере в начале последовательности будут красные карточки 2 и 3, далее зеленая 4, далее синяя 1.

Решение

Для решения этой задачи достаточно сохранить цвета и номера карточек в списке. Далее в трех циклах вывести сначала номера красных карточек, потом — зеленых, и наконец, — синих.

Пример программы-решения

Ниже представлено решение на языке Python 3.

```
s = input()
p = list(map(int, input().split()))
for a in 'rgb':
    for i in range(len(s)):
        if s[i] == a:
            print(p[i], end = ' ')
```

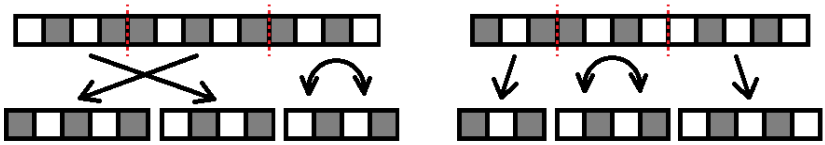
Задача II.1.2.3. Черно-белая полоска (15 баллов)

Темы: задачи для начинающих, реализация, строки.

Условие

У Алисы есть полоска бумаги, расчерченная на клеточки. Полоска имеет ширину в одну клеточку и длину в n клеточек. Алиса хотела раскрасить каждую клеточку в белый или черный цвет так, чтобы клеточки разных цветов чередовались. Но после того, как вся полоска была раскрашена, выяснилось, что Алиса ошиблась, и существует ровно две непересекающихся пары соседних клеточек, раскрашенных в один цвет. Отметим, что *три и более клеток подряд не могут иметь один цвет*. Чтобы исправить свои ошибки, Алиса решила разрезать полоску в двух местах, переставить

и, возможно, развернуть полученные три части, а затем склеить их. На рисунке ниже показаны два примера разрезания и склейки полоски.



В примере на картинке слева исходная полоска разрезается на три части и склеивается в следующем порядке. Кусочек из середины от с номерами клеток из диапазона $[5; 9]$ становится самым левым. Далее к нему пристыковывается кусочек с номерами клеток $[1; 4]$. И, наконец, справа пристыковывается кусочек с номерами клеток $[10; 13]$, который при этом разворачивается на 180°. В результате будет получена полоска из клеток с чередующимися цветами.

В примере на картинке справа кусочки полоски остаются на своих местах, но средняя полоска с номерами клеток $[4; 7]$ разворачивается на 180°.

Напишите программу, которая определит, сможет ли Алиса указанным способом сделать полоску из клеточек чередующихся цветов и, если это возможно, то составит схему разрезания существующей полоски на три кусочка и склейки этих кусочков. Полученная полоска может начинаться как с клетки белого, так и черного цвета. Если требуемую полоску можно получить различными способами, то в качестве ответа можно взять любой из них.

Формат входных данных

На вход подается одна строка, описывающая вид исходной полоски. Строка состоит из символов w и b, обозначающих клетку белого и черного цвета соответственно. Длина строки не превосходит 1000. Гарантируется, что строка имеет вид, описанный в условии задачи.

Формат выходных данных

Если составить полоску из клеток чередующихся цветов невозможно, то программа должна вывести единственное слово *no*. В противном случае вывод должен содержать ровно три строки, каждая из которых описывает кусочек исходной ленты в виде трех чисел. Первое и второе число задают номера начальной и конечной клетки кусочка соответственно. Третье число может иметь одно из двух значений — 0 или 180 в зависимости от того, поворачивается кусочек на 180° или нет. Строки должны следовать в порядке склейки кусочков слева направо.

Методика проверки

Программа проверяется на 15-ти тестах. Прохождение каждого теста оценивается в 1 балл. Тесты из условия задачи при проверке не используются.

Примеры

Пример №1

Стандартный ввод
wbwbbwbwbbwbw
Стандартный вывод
5 9 0 1 4 0 10 13 180

Пример №2

Стандартный ввод
bwbbwbwwbwbw
Стандартный вывод
1 3 0 8 12 180 4 7 0

Пример №3

Стандартный ввод
bbwbbw
Стандартный вывод
no

Пример программы-решения

Ниже представлено решение на языке Python 3.

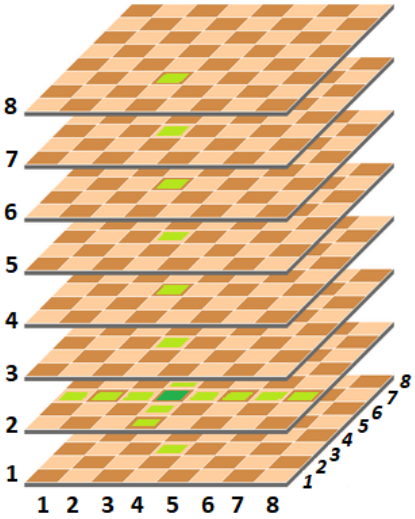
```
s = input()
n = len(s)
x = []
for i in range(1, n):
    if s[i] == s[i-1]:
        x.append(i)
if s[x[0]] != s[x[1]]:
    print(1, x[0], 0)
    print(x[0] + 1, x[1], 180)
    print(x[1] + 1, n, 0)
elif s[x[0]] == s[0] or s[x[0]] == s[-1]:
    print('no')
else:
    print(x[0] + 1, x[1], 0)
    print(1, x[0], 0)
    print(x[1] + 1, n, 180)
```

Задача II.1.2.4. Расстановка ладей на трехмерной шахматной доске (30 баллов)

Темы: реализация, сортировки, структуры данных, динамическое программирование, комбинаторика.

Условие

Алиса и Боб учатся пространственному воображению и решают для этого математические головоломки на трехмерной шахматной доске размера n. Такая доска состоит из n двумерных квадратных досок, расположенных друг над другом. На рисунке изображена трехмерная шахматная доска размера 8.



Каждую клетку трехмерной доски можно задать тремя целыми числами из диапазона $[1; n]$: порядковым номером двумерной доски, номером вертикали на двумерной доске и номером горизонтали. Например, клетка, выделенная на рисунке темно-зеленым цветом, задается тройкой чисел $(2, 4, 3)$.

Трехмерная шахматная ладья ходит по двумерной доске по стандартным правилам, то есть за один ход может переместиться на любую клетку в той же вертикали или горизонтали, где она находится. Вместе с тем трехмерная ладья может за один перейти на любую другую доску в клетку с такой же двумерной координатой. На рисунке светло-зеленым цветом показаны клетки в которые может перейти ладья из клетки с координатами $(2, 4, 3)$. В этом случае говорят, что ладья бьет эти клетки.

Алиса и Боб уверены, что на трехмерной шахматной доске размера n можно расставить n^2 ладей так, что они не будут бить друг друга, но никак не могут понять принцип расстановки.

Напишите программу, которая найдет любую допустимую расстановку ладей на трехмерной шахматной доске размера n так, чтобы они не били друг друга.

Формат входных данных

На вход подается единственное натуральное число n — размер доски, $1 \leqslant n \leqslant 30$.

Формат выходных данных

Выведите координаты n^2 клеток, на которых будут расположены ладьи. Три координаты каждой клетки выводятся через пробел в отдельной строке. Порядок перечисления клеток может быть произвольным.

Методика проверки

Программа проверяется на 30-ти тестах. Номер теста совпадает с числом n.

Примеры

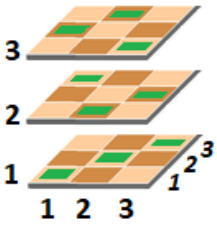
Пример №1

Стандартный ввод
3

Стандартный вывод
1 2 2 1 3 3 2 1 3 2 2 1 2 3 2 3 1 2 3 2 3 3 3 1

Пояснения к примеру

Расстановка ладей из примера показана на рисунке ниже.



Пример программы-решения

Ниже представлено решение на языке Python 3.

```
n = int(input())
for i in range(n):
    for j in range(n):
        print(i + 1, j + 1, (i + j) % n + 1)
```

Задача II.1.2.5. Удаление скобок (30 баллов)

Темы: математика, строки, рекурсивные алгоритмы.

Условие

Боб любит формализм во всем, включая запись математических выражений, поэтому при их записи он ставит скобки так, чтобы каждая операция выделялась своей парой скобок. Например, выражение $(a + b + c) * d$ он запишет как $(((a + b) + c) * d)$ или как $((a + (b + c)) * d)$, а выражение $a * b + c * d$ как $((a * b) + (c * d))$. Таким образом, количество пар скобок в выражениях Боба всегда равно количеству операций, а для операции, которая выполняется последней, пара скобок всегда ограничивает все выражение.

Алиса считает такой перфекционизм избыточным и старается ставить скобки только там, где они нужны для правильных вычислений. Например, в выражении $(a + b + c) * d$ скобки убрать уже нельзя, поскольку выражение $a + b + c * d$ по математическим правилам задает другой порядок применения операций, и результаты вычисления этих двух выражений могут различаться. А вот в выражении $a + (b + c)$ скобки убрать уже можно, поскольку для сложения имеет место сочетательное свойство или, как говорят математики, аксиома ассоциативности. Также можно убрать скобки и в выражении $(a * b) + (c * d)$, поскольку по договоренностям умножение выполняется раньше, чем сложение.

Напишите программу, которая перепишет выражение, записанное Бобом, в тот вид, который нравится Алисе. Обратите внимание, что программа должна просто убрать все избыточные скобки. Другие преобразования делать нельзя. Полученное выражение должно иметь результат вычислений, совпадающий с результатом исходного выражения, при любых значениях параметров.

Формат входных данных

На вход в единственной строке поступает правильно записанное арифметическое выражение, состоящие из имен параметров, скобок и операций «+» и «*». Каждый параметр записывается в виде одной строчной буквы латиницы. Имена параметров не повторяются и встречаются в алфавитном порядке, таким образом, количество операций не превосходит 25. Выражение содержит как минимум одну операцию. Каждая операция в выражении выделяется своей парой скобок, как записано в условии задачи.

Формат выходных данных

Программа должна вывести исходное выражение без избыточных скобок. Порядок следования параметров в ответе должен совпадать с порядком в исходном выражении. В частности, это означает что для теста $(a + b)$ ответ $b + a$ будет считаться ошибочным.

Методика проверки

Программа проверяется на 30-ти тестах. Прохождение каждого теста оценивается в 1 балл. Выражения в первых девяти тестах содержат не более двух операций. Тесты из условия задачи при проверке не используются.

Примеры

Пример №1

Стандартный ввод
`(a+b)`
Стандартный вывод
`a+b`

Пример №2

Стандартный ввод
`((a+(b+c))*d)`
Стандартный вывод
`(a+b+c)*d`

Пример №3

Стандартный ввод
`((a*b)+(c*d))`
Стандартный вывод
`a*b+c*d`

Пример программы-решения

Ниже представлено решение на языке Python 3.

```
1  def parse(s,i):
2      if s[i] != '(':
3          return (s[i], i + 1, s[i])
4      else:
5          left, i, lop = parse(s, i + 1)
6          op = s[i]
7          right, i, rop = parse(s, i + 1)
8          if op == '*' and lop == '+':
9              left = '(' + left + ')'
10         if op == '*' and rop == '+':
11             right = '(' + right + ')'
12         return (left + op + right, i + 1, op)
13 print(parse(input(), 0)[0])
```

Третья волна. Задачи 8–11 класса

Задача II.1.3.1. Кодировка подмножеств (10 баллов)

Темы: задачи для начинающих, математика, реализация.

Условие

Недавно Алиса узнала об одном способе закодировать одним целым числом любое подмножество некоторого заданного конечного множества. Для этого необходимо сопоставить каждому элементу множества число, равное некоторой степени двойки. Теперь в качестве кода произвольного подмножества можно взять сумму чисел, соответствующих элементам этого подмножества.

Алиса составила множество из шести своих друзей и поставила им в соответствие последовательные степени двойки:

- 1 — *Anna*;
- 2 — *Boris*;
- 4 — *Cary*;
- 8 — *David*;
- 16 — *Eva*;
- 32 — *Fiona*.

Например, подмножество {*Anna, Cary, Eva, Fiona*} будет закодировано числом 53. $(1 + 4 + 16 + 32 = 53)$.

Алиса тренируется быстро декодировать подмножество по его коду. Напишите программу, которая позволит проверить ее навыки. Программа должна получать на вход некоторый код и выводить имена друзей, входящих в подмножество с этим кодом.

Формат входных данных

На вход подается единственное натуральное число n — код подмножества, $1 \leqslant n \leqslant 63$.

Формат выходных данных

Выведите имена друзей Алисы, которые входят в закодированное подмножество. Каждое имя следует выводить в отдельной строке. Порядок имен может быть произвольным.

Ниже приведен фрагмент программы на языке Python в котором создается список с правильным написанием слов.

```
Names = ['Anna', 'Boris', 'Cary', 'David', 'Eva', 'Fiona']
```

Методика проверки

Программа проверяется на 20-ти тестах. Прохождение каждого теста оценивается в 0,5 балла. Тест из условия задачи при проверке не используется. Первые шесть тестов — это последовательные степени двойки от 1 до 32.

Примеры

Пример №1

Стандартный ввод
53

Стандартный вывод
Anna Cary Eva Fiona

Пример программы-решения

Ниже представлено решение на языке Python 3.

```
Names = ['Anna', 'Boris', 'Cary', 'David', 'Eva', 'Fiona']
n = int(input())
for i in range(6):
    if n % 2 == 1:
        print(Names[i])
    n //= 2
```

Задача II.1.3.2. Алфавитные подстроки (15 баллов)

Темы: задачи для начинающих, строки, реализация.

Условие

Алиса разрабатывает обучающую игру для младших школьников. В ней игроку дается строка из строчных символов латиницы, а он должен разбить ее на подстроки из последовательных символов алфавита. Такие подстроки далее будем называть правильными. В правильной подстроке после буквы a должна идти буква b, после b — c и так далее. При этом правильная подстрока может начинаться с любого символа. Например, строка *bcdefaabcef* должна быть разбита на $bcdef + a + abc + ef$. Обратите внимание, что подстрока может состоять и из одного символа.

Конечно, игрок может ошибиться и разбить строку неправильным или неоптимальным способом. Например, игрок может разбить строку *bcdefaabcef* на $bcd + ef + aabc + ef$. Чтобы учесть такую возможность Алиса считает очки за найденное разбиение. Если подстрока является правильной, то игроку добавляется количество очков, равное квадрату длины подстроки. Неправильные подстроки не учитываются. Например, за разбиение $bcdef + a + abc + ef$ игрок получит $5^2 + 1^2 +$

$+3^2 + 2^2 = 39$ очков, а за разбиение $bcd{+}ef{+}aabc{+}ef$ лишь $3^2 + 2^2 + 2^2 = 17$ очков.

Напишите программу, которая посчитает количество очков, полученных игроком за сделанное разбиение произвольной строки.

Формат входных данных

На вход в первой строке подается одно натуральное число n — количество фрагментов в разбиении, $1 \leqslant n \leqslant 100$. Далее записаны сами фрагменты разбиения. Каждый фрагмент записан в отдельной строке и состоит только из строчных символов латиницы. Длина каждого фрагмента не превосходит 26.

Формат выходных данных

Выведите одно целое число — количество очков, которое получит игрок за сделанное разбиение.

Методика проверки

Программа проверяется на 15-ти тестах. Прохождение каждого теста оценивается в 1 балл. Тест из условия задачи при проверке не используется. В первых четырех тестах разбиение состоит из одного фрагмента. В следующих четырех тестах каждый фрагмент содержит не более двух символов.

Примеры

Пример №1

Стандартный ввод
4 bcd ef aabc ef
Стандартный вывод
17

Пример программы-решения

Ниже представлено решение на языке Python 3.

```
n = int(input())
ans = 0
for i in range(n):
    s = input()
    for j in range(1, len(s)):
        if ord(s[j]) - ord(s[j - 1]) != 1:
            break
```

```
    else:
        ans += len(s) ** 2
print(ans)
```

Задача II.1.3.3. Пат и Паташон (15 баллов)

Темы: жадные алгоритмы, реализация.

Условие

Боб придумал логическую игру «Пат и Паташон». В этой игре на виртуальной сцене находится n персонажей различного роста, которые выстроены в один ряд. Игрок может удалить часть персонажей со сцены, при этом оставшиеся персонажи смыкаются, не изменяя своего взаимного расположения. После этого персонажи на сцене разбиваются на пары: первый со вторым, третий с четвертым, пятый с шестым и так далее.

В момент удаления игрок должен позаботится о том, чтобы количество оставшихся персонажей стало четным. Первого персонажа в паре (с нечетным номером) будем называть Патом, а второго (с четным номером) — Паташоном. Эффектностью пары будем назвать разность роста Пата и Паташона. Эффектность может быть отрицательной, если окажется, что Пат ниже, чем Паташон. За один раунд игрок получает количество очков, равное сумме эффектностей всех пар. Игрок может удалить со сцены всех персонажей. В этом случае он получит ноль очков.

Рассмотрим пример. Пусть на сцене изначально находилось девять персонажей, рост которых задается массивом чисел $(120, 160, 180, 160, 120, 110, 150, 170, 100)$. Игрок удалил со сцены первого второго и седьмого персонажа. На сцене осталось шесть персонажей с ростом $(180, 160, 120, 110, 170, 100)$. Они разбились на три пары $(180, 160), (120, 110), (170, 100)$, при этом эффектность первой пары — 20, второй — 10, третей — 70. Таким образом, за такое разбиение на пары игрок получит 100 очков. Однако, если игрок оставит на сцене четырех персонажей с ростом $(180, 110, 170, 100)$, то он получит 140 очков.

Напишите программу, которая посчитает максимальное количество очков, которые может получить игрок, для заданной последовательности персонажей.

Формат входных данных

На вход в первой строке подается одно натуральное число n — количество персонажей на сцене в начале игры, $1 \leqslant n \leqslant 1000$. Далее в одной строке через пробел записана последовательность из n натуральных чисел, которые задают рост персонажей. Числа не превосходят 1000.

Формат выходных данных

Выведите одно целое число — максимальное количество очков, которое может получить игрок для заданной последовательности персонажей.

Методика проверки

Программа проверяется на 15-ти тестах. Прохождение каждого теста оценивается в 1 балл. Тест из условия задачи при проверке не используется. В первых пяти тестах на сцене изначально находится ровно четыре персонажа.

Примеры

Пример №1

Стандартный ввод
9 120 160 180 160 120 110 150 170 100
Стандартный вывод
140

Пример программы-решения

Ниже представлено решение на языке Python 3.

```
n = int(input())
x = list(map(int,input().split()))
print(sum([max(x[i]-x[i+1],0) for i in range(n-1)]))
```

Задача II.1.3.4. Выезд на экскурсию (30 баллов)

Темы: математика, структуры данных, реализация.

Условие

Администрация школы организовала автобусную экскурсию для своих учеников. Всего было заказано n автобусов разной вместимости. Обозначим за c_i количество детей, которые могут находиться в i-том автобусе. Все c_i являются *четными* числами. Учителя неформально делят всех учеников на активных и спокойных. Всего на экскурсию поедет m активных и k спокойных детей. Учителя хотели бы распределить детей по автобусам так, чтобы количество спокойных и активных детей в каждом автобусе отличалось как можно меньше. Формально это означает следующее. Обозначим за x_i и y_i количество активных и спокойных детей соответственно в i-том автобусе. Вычислим модули разности количества активных и спокойных детей а каждом автобусе и просуммируем полученные числа. Полученная величина $\sum_{i=1}^{n} |x_i - y_i|$ должна оказаться минимально возможной.

Но когда автобусы подъехали, все пошло не по плану. Часть детей выбежали из школы и расселись по автобусам произвольно. После подсчетов выяснилось, что в i-том автобусе уже находится a_i активных детей и b_i спокойных. Чтобы не увеличивать неразбериху, было решено оставить их на своих местах и постараться рассадить оставшихся детей в соответствии с изначально выбранным принципом.

Напишите программу, которая найдет значения x_i и y_i с учетом всех требований, а именно:

- $x_i \geqslant a_i$;
- $y_i \geqslant b_i$;
- $x_i + y_i \leqslant c_i$;
- $\sum_{i=1}^{n} x_i = m$;
- $\sum_{i=1}^{n} y_i = k$;
- $\sum_{i=1}^{n} |x_i - y_i| \to \min$.

Если допустимых ответов несколько, то можно вывести любой.

Формат входных данных

На вход в первой строке через пробел подается три целых числа n, m и k — количество автобусов, количество активных и количество спокойных детей соответственно; $1 \leqslant n \leqslant 100$; $0 \leqslant m, k \leqslant 10000$. Во второй строке через пробел записаны числа $a_1, a_2, \cdots, a_n$, задающие количество активных детей, изначально находящихся в каждом из автобусов. В третьей строке аналогично записаны числа $b_1, b_2, \cdots, b_n$, задающие количество спокойных детей, изначально находящихся в каждом из автобусов. Наконец, в четвертой строке записаны натуральные четные числа $c_1, c_2, \cdots, c_n$, задающие вместимость каждого из автобусов; $2 \leqslant c_i \leqslant 100$. Все входные значения заданы корректно в соответствии с условием задачи. В том числе гарантируется, что общее количество школьников не превосходит суммарной вместимости всех автобусов.

Формат выходных данных

Вывод должен состоять из двух строк. В первой строке через пробел следует вывести значения x_i — количество активных детей в каждом из автобусов. Во второй строке аналогично вывести значения y_i — количество спокойных детей в каждом из автобусов.

Методика проверки

Программа проверяется на 30-ти тестах. Прохождение каждого теста оценивается в 1 балл. Тест из условия задачи при проверке не используется. В первых пяти тестах количество автобусов равно двум. В следующих пяти тестах суммарное количество школьников равно суммарной вместимости автобусов.

Примеры

Пример №1

Стандартный ввод

`4 50 80`
`10 20 0 0`
`25 5 20 0`
`40 30 40 30`

Стандартный вывод
`10 20 10 10` `25 10 25 20`

Пояснения к примеру

Ответ удовлетворяет всем ограничениям. Сумма всех x_i равна 50. Сумма всех y_i равна 80. В первом и третьем автобусе едет по 35 детей, а во втором и четвертом — по 30. Эти значения не превосходят вместимости соответствующих автобусов. Также для всех i выполняются неравенства $x_i \geqslant a_i$ и $y_i \geqslant b_i$. Значение выражения $\sum_{i=1}^{n} |x_i - y_i|$ равно 50. Можно доказать, что другие допустимые варианты распределения не дадут меньшей величины.

Возможны и другие правильные ответы, например, следующий.

```
15 20 15 0
25 10 20 25
```

Для этого ответа также выполнены все ограничения, а сумма $\sum_{i=1}^{n} |x_i - y_i|$ равна 50.

Пример программы-решения

Ниже представлено решение на языке Python 3.

```python
n, m, k = map(int, input().split())
a = list(map(int, input().split()))
b = list(map(int, input().split()))
c = list(map(int, input().split()))
m -= sum(a)
k -= sum(b)
for i in range(n):
    if a[i] > b[i]:
        v = min(k, a[i] - b[i], c[i] - a[i] - b[i])
        b[i] += v
        k -= v
    else:
        v = min(m, b[i] - a[i], c[i] - a[i] - b[i])
        a[i] += v
        m -= v
for i in range(n):
    v = min(m, k, (c[i] - a[i] - b[i]) // 2)
    a[i] += v
    b[i] += v
    m -= v
    k -= v
for i in range(n):
    v = min(m, c[i] - a[i] - b[i])
    a[i] += v
    m -= v
    v = min(k, c[i] - a[i] - b[i])
    b[i] += v
    k -= v
print(*a)
print(*b)
```

Задача II.1.3.5. Нескучные каникулы (30 баллов)

Темы: сортировки, структуры данных, реализация.

Условие

У Алисы закончился очередной учебный год, и она составляет расписание на каникулы. Алиса планирует, что в ее каникулы состоится некоторое число событий, таких как посещение концертов, празднование дней рождений и так далее. Алиса называет i-тый день каникул нескучным, если для него выполняется хотя бы одно из двух условий:

- в i-тый день состоится хотя бы одно событие;
- хотя бы одно событие состоится в день с номером $i-1$ и в день с номером $i+1$.

Рассмотрим пример. Пусть в каникулах 10 дней и некоторые события произойдут в дни с номерами $2, 3, 5, 9, 10$. Тогда нескучными будут все эти дни, а также день с номером 4, поскольку некоторые события произойдут в два соседних с ним дня.

При составлении расписания Алиса учитывает, что для некоторых событий заранее известна дата, а для других она сама может подобрать подходящий день. Алиса хочет расставить события с открытой датой так, чтобы каникулы получились наиболее нескучными, то есть чтобы количество нескучных дней в каникулах было максимальным.

Напишите программу, которая подберет дни для событий с открытой датой так, чтобы каникулы получились наиболее нескучными.

Формат входных данных

На вход в первой строке через пробел подается три целых числа n, m и k — продолжительность каникул в днях, количество событий с открытой датой и количество событий с заданной датой соответственно; $1 \leqslant n \leqslant 100000$; $1 \leqslant m \leqslant 100000$; $0 \leqslant k \leqslant 100000$.

Во второй строке через пробел записаны k натуральных чисел $d_1, d_2, \cdots, d_k$ — номера дней, в которые произойдут события с известной датой; $1 \leqslant d_i \leqslant n$. Числа могут повторяться и следовать в произвольном порядке. Если k будет равно нулю, то вторая строка будет пустой.

Формат выходных данных

В первой строке выведите одно натуральное число s — количество нескучных дней в каникулах. Во второй строке через пробел выведите m натуральных чисел $t_1, \cdots, t_m$ — номера дней, в которые Алиса должна запланировать события с открытой датой. Если допустимых ответов будет несколько, то можно вывести любой. Числа могут повторяться и следовать в произвольном порядке.

Методика проверки

Программа проверяется на 30-ти тестах. Прохождение каждого теста оценивается в 1 балл. Тесты из условия задачи при проверке не используются. В трех первых тестах $k = 0$. В следующих трех тестах $k = 1$. В первых 15-ти тестах n, m и k не превосходят 100.

Примеры

Пример №1

Стандартный ввод
11 5 6 1 3 5 7 9 11
Стандартный вывод
11 1 1 1 1 1

Пример №2

Стандартный ввод
11 2 0
Стандартный вывод
3 2 4

Пример №3

Стандартный ввод
15 2 5 1 2 8 12 14
Стандартный вывод
11 4 6

Пояснения к примеру

В первом примере все дни каникул являются нескучными из-за событий с известной датой, поэтому пять событий с открытой датой можно расставить произвольно.

В ответе ко второму примеру нескучными будут дни с номерами 2, 3, 4. Улучшить ответ нельзя.

В ответе к третьему примеру нескучными будут 11 дней с номерами 1, 2, 3, 4, 5, 6, 7, 8, 12, 13, 14. Улучшить этот ответ нельзя, хотя набор дней может быть другим, например, 4, 10 или 6, 10.

Пример программы-решения

Ниже представлено решение на языке Python 3.

```
n, m, k = map(int, input().split())
d = [False] * (n + 2)
for x in map(int, input().split()):
    d[x] = True
if k == 0:
    ans = list(range(1, min(n, 2 * m), 2))
```

```
    ans.extend([n] * max(m - len(ans), 0))
else:
    p = 0
    segs = []
    for i in range(1,n + 1):
        if d[i]:
            if p == 0:
                plen = i - 1
            elif  i - p > 2:
                segs.append((p + 2, i))
            p = i
    segs.sort(key = lambda x: x[1] - x[0] + ((x[1] - x[0]) % 2) * 100000)
    ans = []
    for (a, b) in segs:
        ans.extend(range(a, b, 2))
    if n - p > 1:
        ans.extend(range(p + 2, n + 1, 2))
    if plen > 1:
        ans.extend(range(plen - 1, 0, -2))
    if (n - p) % 2 == 1:
        ans.append(n)
    ans = ans[: min(m, len(ans))]
    ans.extend([1] * (m - len(ans)))
for i in ans:
    d[i] = True
s = 0
for i in range(1, n + 1):
    if d[i] or d[i - 1] and d[i + 1]:
        s += 1
print(s)
print(*ans)
```

Предметный тур. Физика

Первая волна. Задачи 8–9 класса

Задача II.2.1.1. Конвейер (13 баллов)

Темы: кинематика.

Условие

Робот-доставщик по ошибке заехал на один из концов конвейерной ленты длиной L, движущуюся с постоянной скоростью v в противоположном ее движению направлении. Продолжая двигаться с постоянной относительно ленты скоростью u, робот сумел покинуть конвейер через время t. Определите u. Ответ дайте в м/с, округлив до сотых.

Решение

Скорость робота относительно земли $v_1 = L/t$ складывается (с правильным учетом знаков) из его скорости относительно поверхности ленты и скорости ленты относительно земли:

$$\frac{L}{t} = v_1 = u - v.$$

Откуда простыми алгебраическими преобразованиями получим:

$$u = v + \frac{L}{t}.$$

Погрешность $0,02$ м/с.

Диапазоны

Величина	min	max	Шаг
L, м	19	23	1
v, м/с	1,6	1,9	0,05
t, с	150	250	10

Ответ: $u = v + \frac{L}{t}$.

Задача II.2.1.2. Выжигатель (17 баллов)

Темы: плотность, тепловые явления.

Условие

Температура кипения сплава на t выше его текущей температуры. Его удельная теплоемкость c, удельная теплота возгонки (испарения из твердого состояния) L, плотность ρ. Какой должна быть минимальная энергия E лазерного импульса, чтобы он был способен испарить кубик сплава со стороной a при условии полного поглощения энергии импульса веществом? Считайте, что импульс настолько кратковременный, что сплав не успевает пройти жидкую фазу и испаряется непосредственно из твердой. Ответ дайте в Дж, округлив до десятых.

Решение

Теплота, необходимая для нагрева и последующего испарения сплава массы m, находится по формуле:

$$Q = (ct + L)m.$$

По условиям вся энергия импульса поглощается веществом, то есть переходит в тепло. Следовательно, $E = Q$. Масса сплава может быть выражена через его плотность и объем испаренной порции: $m = \rho a^3$. Подставляя, получим:

$$E = (ct + L)\rho a^3.$$

Погрешность $0,2$ Дж.

Диапазоны

Величина	min	max	Шаг
t, °C	2200	2700	10
c, Дж/(кг·°C)	400	500	10
L, кДж/кг	5500	7000	100
ρ, кг/м^3	6000	8000	100
a, мм	0,6	1	0,1

Ответ: $E = (ct + L)\rho a^3$. С учетом порядков: $E = (ct + L[\cdot 10^3])\rho a^3[\cdot 10^{-9}]$.

Задача II.2.1.3. Катушка (20 баллов)

Темы: закон Ома.

Условие

При изготовлении реостата на диэлектрическую бобину диаметром D был намотан в N одинаковых, плотно прилегающих к бобине витков, константановый провод в лаковой изоляции. Площадь поперечного сечения провода S, удельное сопротивление константана $\rho = 0,4$ Ом·мм^2/м. Номинальное сопротивление резистора было вычислено и указано в паспорте устройства, исходя из этих параметров, однако от износа n последних витков проволоки перетерлось и отвалилось. Найдите максимальное значение n, при котором сопротивление реостата отличается от номинального не более, чем на ΔR.

Решение

Один виток провода имеет длину $l = \pi D$ и электрическое сопротивление:

$$r = \frac{\rho l}{S} = \frac{\pi \rho D}{S}.$$

Каждый отвалившийся виток уменьшает общее сопротивление резистора на величину r. Таким образом,

$$n = \left\lfloor \frac{\Delta R}{r} \right\rfloor = \left\lfloor \frac{\Delta R S}{\pi \rho D} \right\rfloor.$$

Погрешность 1.

Диапазоны

Величина	min	max	Шаг
D, см	3	5	$0,1$
S, мм^2	$0,1$	$0,2$	$0,02$
N	100	250	10
ΔR, Ом	3	5	$0,5$

Ответ: $n = \left\lfloor \frac{\Delta R S}{\pi \rho D} \right\rfloor$. С учетом порядков: $n = \left\lfloor \frac{\Delta R S}{\pi \rho D} [\cdot 10^2] \right\rfloor$.

Задача II.2.1.4. Болт (20 баллов)

Темы: золотое правило механики, работа.

Условие

При болтовом соединении деталей болт немного удлиняется, работая как растянутая пружина, прижимающая детали друг к другу с некоторой описанной в технической документации силой F. Определите, какую работу A нужно совершить, чтобы растянуть болт на величину Δl и создать таким образом силу F с помощью ключа длиной L, если диаметр резьбы болта равен d, диаметр шляпки D, шаг резьбы — b? Трение и деформацию самих соединяемых деталей считайте пренебрежимо малым. Ответ дайте в Дж, округлив до целого.

Решение

Резьба болта и гаечный ключ представляют собой простые механизмы, дающие выигрыш в силе, но не изменяющие, согласно золотому правилу механики, работу. Поэтому общая работа, которая нужна, чтобы растянуть болт, вне зависимости от способа, считается как работа переменной силы или как разность потенциальных энергий деформированного тела (болта). При этом малость деформации позволяет применить здесь закон Гука:

$$A = \frac{F \Delta l_к}{2} - \frac{F \Delta l_н}{2}.$$

Учитывая $\Delta l_{\text{н}} = 0$, получим окончательно:

$$A = \frac{F\Delta l}{2}.$$

Погрешность 1 Дж.

Диапазоны

Величина	min	max	Шаг
F, кН	200	250	10
Δl, мм	$0,25$	$0,35$	$0,01$
L, см	40	70	5
d, мм	20	25	1
D, мм	30	40	1
b, мм	2	3	$0,1$

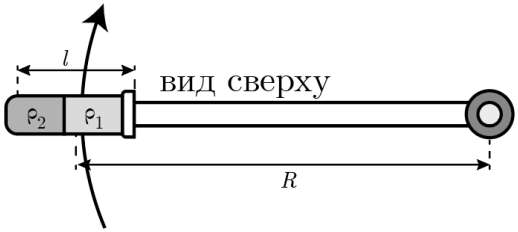
Ответ: $A = \dfrac{F\Delta l}{2}$.

Задача II.2.1.5. Центрифуга (25 баллов)

Темы: центростремительное ускорение, гидростатическое давление.

Условие

Для разделения жидких смесей в лаборатории используется центрифуга, представляющая собой горизонтальное колесо радиуса R, на ободе которого закрепляются пробирки с жидкостью, ориентированные дном строго от центра колеса. Найдите давление, оказываемое на дно пробирки длиной l при ее вращении в центрифуге, если пробирка совершает один оборот за время T и ровно на половину своего объема пробирка заполнена составом с плотностью ρ_1 и ровно на половину — другим составом с плотностью ρ_2. Считайте $R \gg l$, а влияние силы тяжести пренебрежимо малым. Ответ дайте в кПа, округлив до целого. Длина окружности в 2π раз больше ее радиуса.



Решение

Линейная скорость пробирки может быть найдена по формуле $v = 2\pi R/T$, поскольку путь, проходимый пробиркой за время T, равен длине окружности с радиусом R. Соответствующее ей центростремительное ускорение a равно:

$$a = \frac{v^2}{R} = \frac{4\pi^2 R}{T^2}.$$

Это ускорение играет роль ускорения свободного падения в формуле гидростатического давления $\rho g h$, что может быть доказано из условий равновесия элемента жидкости. Поскольку жидкости в пробирке две и каждая из них создает столб «высотой» $l/2$, для общего давления получим:

$$p = \frac{2\pi^2 R l}{T^2}(\rho_1 + \rho_2).$$

Погрешность 3 кПа.

Диапазоны

Величина	min	max	Шаг
ν, об/с	5	7	0,5
R, см	90	120	5
l, см	6	8	1
ρ_1, кг/м^2	800	980	20
ρ_2, кг/м^2	1020	1200	20

Ответ: $p = \frac{2\pi^2 R l}{T^2}(\rho_1 + \rho_2)$. С учетом порядков $p = \frac{2\pi^2 R l}{T^2}(\rho_1 + \rho_2)[\cdot 10^{-7}]$.

Задача II.2.1.6. (5 баллов)

Темы: физики России.

Условие

Этого выдающегося ученого, обучавшегося еще в технологическом институте Николая I, но удостоенного и ленинской, и сталинской премий, нередко называют родоначальником советской физики. Такие известные физики как Капица и Курчатов достигли своих выдающихся результатов под его руководством, а сам он был воспитан под руководством первооткрывателя «икс-лучей», используемых теперь в каждой поликлинике.

1. Петр Николаевич Лебедев.
2. Абрам Федорович Иоффе.
3. Николай Алексеевич Умов.
4. Александр Александрович Фридман.
5. Сергей Александрович Ахманов.

6. Рем Викторович Хохлов.
7. Владимир Александрович Фок.
8. Михаил Васильевич Остроградский.

Ответ: 2.

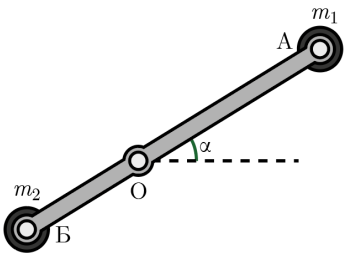
Первая волна. Задачи 10–11 класса

Задача II.2.2.1. Противовес (15 баллов)

Темы: кинематика, импульс.

Условие

Легкий рычаг АОБ, способный вращаться вокруг неподвижной точки О, где БО $= l$, ОА $= 2l$ используется на производственной линии для перемещения грузов. В его точке А закреплен груз m_1, в точке Б — противовес m_2. В некоторый момент модуль импульса груза в точке А равен p_1. Найдите модуль импульса противовеса в этот же момент времени. Дайте ответ в кг $\cdot$ м/с, округлив до целого.



Решение

Поскольку БО : ОА $= 1 : 2$, скорости двух масс всегда связаны соотношением $v_1 = 2v_2$. Тогда их импульсы связаны соотношением:

$$\frac{p_2}{p_1} = \frac{m_2 v_2}{m_1 v_1} = \frac{m_2}{2m_1}.$$

Отсюда окончательно получим:

$$p_2 = p_1 \frac{m_2}{2m_1}.$$

Погрешность 1 кг $\cdot$ м/с.

Диапазоны

Величина	min	max	Шаг
l, м	1	1, 5	0, 1
m_1, кг	50	80	5
m_2, кг	50	80	5
p_1, кг · м/с	50	160	10

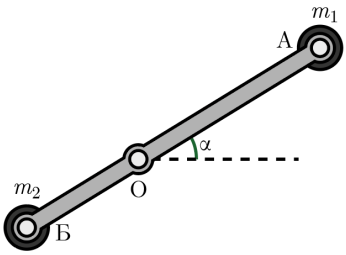
Ответ: $p_2 = p_1 \dfrac{m_2}{2m_1}$.

Задача II.2.2.2. Качели (20 баллов)

Темы: законы сохранения.

Условие

Легкий рычаг АОБ, способный вращаться вокруг неподвижной точки О, где БО $= l$, ОА $= 2l$ используется, на производственной линии для перемещения грузов. В его точке А закреплен груз m_1, в точке Б — противовес m_2. В некоторый момент электропривод в точке О удерживал рычаг под углом α к горизонту. В этот момент произошла авария, в результате которой привод в точке О перестал действовать, и рычаг пришел в свободное вращение вокруг этой оси. Определите скорость v_1 точки А рычага в момент, когда она оказалась строго под точкой Б. Дайте ответ в м/с, округлив до десятых. Ускорение свободного падения $g = 9,8\,\text{м/с}^2$.



Решение

Поскольку БО : ОА $= 1 : 2$, скорости двух масс всегда связаны соотношением $v_1 = 2v_2$. Высоты (относительно точки О), на которых находились массы в момент аварии, равны $h_{01} = 2l \sin\alpha, h_{02} = -l \sin\alpha$. Аналогично высоты, на которых находились массы в момент, когда точка Б оказывается строго под точкой А, равны $h_1 = -2l, h_2 = l$.

Тогда закон сохранения энергии для масс на концах рычага имеет вид:

$$(2m_1 - m_2)gl \sin\alpha = (m_2 - 2m_1)gl + \left(m_1 + \frac{m_2}{4}\right)\frac{v_1^2}{2}.$$

Из этого уравнения можно непосредственно выразить скорость v_1:

$$v_1 = \sqrt{\frac{8gl(2m_1 - m_2)(1 + \sin\alpha)}{4m_1 + m_2}}.$$

Погрешность $0,2$ км/ч.

Диапазоны

Величина	min	max	Шаг
l, м	1	$1,5$	$0,1$
m_1, кг	50	80	5
m_2, кг	50	80	5
α, $^\circ$	30	70	5

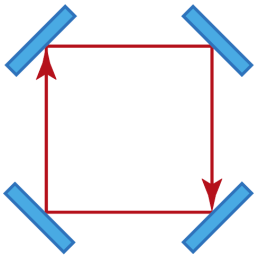
Ответ: $v_1 = \sqrt{\dfrac{8gl(2m_1 - m_2)(1 + \sin\alpha)}{4m_1 + m_2}}$.

Задача II.2.2.3. Резонатор (20 баллов)

Темы: оптика.

Условие

Кольцевой оптический резонатор представляет собой квадрат со стороной a, в углах которого расположены наклоненные под 45° к ходу луча зеркала. При каждом падении на зеркало $k\%$ энергии света отражается, а остальные $(100 - k)\%$ поглощаются материалом зеркала. Какая доля изначальной энергии останется у светового луча, когда он сделает N полных витков? Ответ дайте в процентах.



Решение

За один виток свет совершает 4 отражения. В ходе каждого из них его энергия изменяется в $k/(100\%)$ раз. Таким образом, общая η доля сохранившейся энергии равна:

$$\eta = \left(\frac{k}{100\%}\right)^{4N} \cdot 100\%.$$

Погрешность 1%.

Диапазоны

Величина	min	max	Шаг
a, см	10	50	5
k	98,5	99,5	0,5
N	20	30	1

Ответ: $\eta = \left(\frac{k}{100\%}\right)^{4N} \cdot 100\%.$

Задача II.2.2.4. Спидометр (20 баллов)

Темы: кинематика.

Условие

Спидометр спутниковой системы навигации, показывая текущую скорость, на самом деле определяет ее как среднюю скорость за последние t движения. Определите мгновенную скорость аппарата, если его спидометр сейчас показывает скорость v_1, t назад показывал скорость v_2 и известно, что аппарат движется равноускоренно. Ответ дайте в км/ч, округлив до десятых.

Решение

При равноускоренном движении на протяжении некоторого времени t с начальной скоростью v_0 и ускорением a конечная скорость оказывается равна $v = v_0 + at$, а средняя

$$v_{\text{ср}} = \frac{v_0 + v}{2} = v_0 + \frac{at}{2}.$$

При этом, поскольку по прошествии дополнительного времени t и v, и v_0 вырастут на at, средняя скорость также вырастет на at. Следовательно, ускорение аппарата может быть найдено как

$$a = \frac{v_1 - v_2}{t},$$

а его мгновенная скорость:

$$v = v_1 + \frac{at}{2} = v_1 + \frac{v_1 - v_2}{2} = \frac{3v_1 - v_2}{2}.$$

Погрешность 0,1 км/ч.

Диапазоны

Величина	min	max	Шаг
t, с	$1,5$	3	$0,25$
v_1, км/ч	52	56	$0,5$
v_2, км/ч	47	51	$0,5$

Ответ: $v = \dfrac{3v_1 - v_2}{2}$.

Задача II.2.2.5. Разрядка (20 баллов)

Темы: электростатика.

Условие

Испытываемая в лаборатории антенна состоит из двух одинаковых металлических шаров, разнесенных на некоторое расстояние и соединенных перемычкой из материала с очень высоким, но конечным сопротивлением. В результате шары обмениваются зарядом таким образом, что за каждый интервал времени t разница между их зарядами сокращается вдвое. В некоторый момент первый шар был заряжен, а второй — нет. Спустя t было измерено, что сила электростатического взаимодействия между шарами антенны равна F_0. Какой станет эта сила, спустя еще t времени? Явлением электростатической индукции пренебречь. Ответ дайте в нН (наноньютоны), округлив до целого.

Решение

Пусть в начальный момент времени заряженный шар имеет заряд q. Тогда, спустя t, заряды на шарах должны отличаться на $q/2$, что, по закону сохранения заряда, возможно только если их заряды будут равны $3q/4$ и $q/4$. Соответствующая сила взаимодействия находится по закону Кулона:

$$F_0 = k\frac{(q/4)(3q/4)}{r^2} = \frac{3}{16}\cdot\frac{kq^2}{r^2}.$$

Спустя еще t разница между зарядами должна оказаться равна $q/4$, а заряды, соответственно, $5q/8$ и $3q/8$. Соответствующая сила:

$$F = k\frac{(3q/8)(5q/8)}{r^2} = \frac{15}{64}\cdot\frac{kq^2}{r^2} = \frac{5}{4}F_0.$$

Погрешность 1 нН.

Диапазоны

Величина	min	max	Шаг
t, с	5	40	1
F, нН	10	100	2

Ответ: $F = \dfrac{5}{4}F_0$.

Задача II.2.2.6. (5 баллов)

Темы: физики России.

Условие

Этого выдающегося ученого, обучавшегося еще в технологическом институте Николая I, но удостоенного и ленинской, и сталинской премий, нередко называют родоначальником советской физики. Такие известные физики как Капица и Курчатов достигли своих выдающихся результатов под его руководством, а сам он был воспитан под руководством первооткрывателя «икс-лучей», используемых теперь в каждой поликлинике.

1. Рем Викторович Хохлов.
2. Абрам Федорович Иоффе.
3. Владимир Александрович Фок.
4. Александр Александрович Фридман.
5. Сергей Александрович Ахманов.
6. Петр Николаевич Лебедев.
7. Николай Алексеевич Умов.
8. Михаил Васильевич Остроградский.

Ответ: 2.

Вторая волна. Задачи 8–9 класса

Задача II.2.3.1. Катапульта (10 баллов)

Темы: кинематика.

Условие

Для запуска беспилотного летательного аппарата используется катапульта — длинная балка с установленным на ней линейным двигателем, способным создавать постоянное ускорение a. Для успешного взлета аппарат должен успеть набрать скорость v относительно воздуха до отрыва от катапульты. Вычислите минимальную необходимую длину катапульты, если взлет должен успешно осуществляться при попутном ветре со скоростью не выше u. Ответ дайте в м, округлив до десятых.

Решение

При попутном ветре скорость аппарата относительно воздуха равна $v = v_0 + u$, где v_0 — его скорость относительно катапульты. Как известно из кинематики (или законов сохранения), на расстоянии l, двигаясь с постоянным ускорением a, беспилотник может увеличить квадрат скорости на величину скорость $v^2 = 2al_0$. Отсюда найдем изначальную длину балки:

$$l = \frac{v^2}{2a} = \frac{(v+u)^2}{2a}.$$

Погрешность $0,1$ м.

Диапазоны

Величина	min	max	Шаг
a, м/с^2	25	40	1
v, м/с	9	12	$0,5$
u, м/с	2	4	$0,5$

Ответ: $l = \dfrac{(v+u)^2}{2a}$.

Задача II.2.3.2. Отвердевание (15 баллов)

Темы: плотность.

Условие

Некоторый полимер был получен в жидком состоянии, в котором он целиком занимал лабораторную чашку объемом V_0 и имел плотность ρ_0. Затем он был нагрет в специальной печи, в результате чего из состава испарились все летучие фракции общей массой Δm, а оставшееся вещество запеклось, образовав твердый сгусток с плотностью ρ. Определите объем, занимаемый твердым веществом в чашке, если известно, что пустоты в процессе запекания не образуются и никакие внешние соединения не включаются в полимер. Ответ дайте в см^3 , округлив до целого.

Решение

Изначальная масса полимера равна $m_0 = \rho_0 V_0$. За счет испарения она уменьшилась на Δm. Процесс отвердевания не изменяет массы вещества, поэтому объем образовавшегося твердого вещества равен $V = (m_0 - \Delta m)/\rho$. Окончательно:

$$V = \frac{\rho_0 V_0 - \Delta m}{\rho}.$$

Погрешность 5 см^3.

Диапазоны

Величина	min	max	Шаг
V_0, см^3	1200	1500	100
ρ_0, г/см^3	$0,9$	$1,1$	$0,02$
ρ, г/см^3	$1,4$	$1,7$	$0,02$
Δm, г	100	150	10

Ответ: $V = \dfrac{\rho_0 V_0 - \Delta m}{\rho}$.

Задача II.2.3.3. Три образца (20 баллов)

Темы: электростатика.

Условие

В лабораторию поступило три металлических образца одинаковых размеров А, Б и В, электрически заряженных в ходе трех разных малоисследованных процессов. Лаборанты записали результаты измерений их зарядов $q_{1,2,3}$, но позже выяснилось, что процедура проведения эксперимента была нарушена и было допущено соприкосновение образцов друг с другом. В ходе анализа записей лабораторной камеры удалось установить, что вначале соприкоснулись образцы А и Б, после чего на одном из них был измерен заряд q_1. Затем с этим образцом дополнительно соприкоснулся образец В, после чего на нем был измерен заряд q_2. После этого все три образца соприкоснулись одновременно и на одном из них был измерен заряд q_3. Восстановите по этим данным исходное значение заряда образца В. Считайте, что процедура измерения величину заряда на образце не меняет. Ответ дайте в нКл (нанокулонах), округлив до целого.

Решение

Когда два металлических предмета одинаковых размеров соприкасаются, на них устанавливаются одинаковые электрические заряды. По закону сохранения заряда они должны быть равны среднему арифметическому исходных зарядов.

Тогда после первого соприкосновения

$$q_1 = \frac{q_\text{А} + q_\text{Б}}{2},$$

после второго

$$q_2 = \frac{q_1 + q_\text{В}}{2},$$

а после третьего

$$q_3 = \frac{q_\text{А} + q_\text{Б} + q_\text{В}}{3} = \frac{2q_1 + q_\text{В}}{3}.$$

Решая эту систему уравнений, легко получим:

$$q_\text{В} = 4q_2 - 3q_3.$$

Погрешность 1 нКл.

Диапазоны

Величина	min	max	Шаг
q_1, нКл	-20	20	4
q_2, нКл	-20	20	4
q_3, нКл	-20	20	4

Ответ: $q_\text{В} = 4q_2 - 3q_3$.

Задача II.2.3.4. Топливо (25 баллов)

Темы: тепловые явления.

Условие

На некоторой планете были обнаружены залежи жидкого состава, состоящего из двух трудно разделимых компонент. Первая из них химически инертна (не участвует ни в каких превращениях) и имеет теплоемкость c_1. Вторая имеет теплоемкость c_2 и горит в атмосфере планеты с удельной теплотой сгорания q. Смесь воспламеняется при температуре θ, а окружающая среда планеты имеет температуру t. Определите, какую минимальную долю от общей массы смеси должна составлять масса горючей компоненты, чтобы горение смеси производило не меньше энергии, чем уходит на ее нагрев до температуры воспламенения. Ответ дайте в процентах, округлив до десятых.

Решение

Рассмотрим некоторую массу m смеси. Она содержит $m_2 = \alpha m$ горючей и $m_1 = (1-\alpha)m$ негорючей жидкостей. Для воспламенения необходимо нагреть смесь на $\Delta t = \theta - t$, на что уйдет

$$Q_1 = (c_1 m_1 + c_2 m_2)\Delta t = m(c_1(1-\alpha) + c_2\alpha)(\theta - t)$$

теплоты. При этом от сгорания второй компоненты выделится $Q_2 = qm_2$ теплоты. Приравнивая Q_1 и Q_2, получим:

$$\cancel{m}\alpha q = \cancel{m}(c_1(1-\alpha) + c_2\alpha)(\theta - t),$$

откуда окончательно выразим ответ:

$$\alpha = \frac{c_1(\theta - t)}{(\theta - t)(c_1 - c_2) + q} \cdot 100\%.$$

Погрешность $0,1\%$.

Диапазоны

Величина	min	max	Шаг
c_1, Дж/(кг·°C)	300	500	20
c_2, Дж/(кг·°C)	310	510	20
q, МДж/(кг)	13	17	$0,5$
θ, °C	800	950	10
t, °C	-80	-50	5

Ответ: $\alpha = \dfrac{c_1(\theta - t)}{(\theta - t)(c_1 - c_2) + q} \cdot 100\%.$

С учетом порядков $\alpha = \dfrac{c_1(\theta - t)}{(\theta - t)(c_1 - c_2) + q[\cdot 10^6]} \cdot 100\%.$

Задача II.2.3.5. Геккон (25 баллов)

Темы: давление, сила трения.

Условие

Робот-геккон может перемещаться по гладким вертикальным поверхностям, используя присоски. Под каждой присоской площади S при помощи системы компрессоров, откачивающих из-под присосок воздух, создается давление p, в то время как атмосферное давление $p_0 = 100\,\text{кПа}$.

Какую максимальную массу может иметь такой робот, чтобы не соскальзывать с вертикальной поверхности, если он одновременно использует n одинаковых присосок (коэффициент трения резины присосок равен μ)?

Ускорение свободного падения принять равным $g = 9,8\,\text{м/с}^2$. Ответ дайте в кг, округлив до десятых.

Решение

На каждую присоску действует прижимная сила, равная произведению ее площади на разницу внешнего и внутреннего давлений:

$$N = S(p_0 - p).$$

Возникающие в этих присосках силы трения $F_{\text{тр}} = \mu N$ должны суммарно уравновешивать вес робота:

$$mg = nF_{\text{тр}} = \mu nS(p_0 - p).$$

Таким образом, окончательно

$$m = \frac{\mu nS(p_0 - p)}{g}.$$

Погрешность $0,2\,\text{кг}$.

Диапазоны

Величина	min	max	Шаг
S, см^2	10	35	2,5
μ	0,12	0,25	0,01
p, кПа	10	40	5
n	6	12	2

Ответ: $m = \dfrac{\mu nS(p_0 - p)}{g}$. С учетом порядков $m = \dfrac{\mu nS(p_0 - p)}{g}[\cdot 10^{-1}]$.

Задача II.2.3.6. (5 баллов)

Темы: физики России.

Условие

Один из основателей нелинейной оптики, новой ветви науки, продемонстрировавшей, что достаточно интенсивные лучи света могут взаимодействовать друг с другом и сами с собой, фокусироваться без линзы и неожиданно менять цвет. В его честь на территории Московского университета названа улица, лаборатория, спортивный клуб и несколько учебных аудиторий. Помимо выдающихся научных достижений, он также проявил себя как талантливый организатор, способствовал развитию кооперации исследователей самых разных направлений, включая биологию и экологию, а также был профессиональным альпинистом с двадцатилетним стажем. Трагедия в одном из горных походов, к несчастью, оборвала его выдающуюся жизнь.

1. Сергей Александрович Ахманов.
2. Рем Викторович Хохлов.
3. Анатолий Алексеевич Логунов.
4. Петр Николаевич Лебедев.
5. Абрам Федорович Иоффе.
6. Александр Александрович Фридман.
7. Роберт Эмильевич Ленц.
8. Николай Алексеевич Умов.

Ответ: 2.

Вторая волна. Задачи 10–11 класса

Задача II.2.4.1. Кабель (12 баллов)

Темы: закон Ома, сопротивление.

Условие

На мобильной исследовательской станции используются стандартные резервные кабели для большинства электроприборов, имеющие площадь поперечного сечения s и длину l. В документации кабеля указано, что при подключении к стандартному лабораторному источнику постоянного тока I падение напряжения на кабеле составляет U. Определите удельное сопротивление материала кабеля. Ответ дайте в Ом $\cdot$ мм^2/м, округлив до тысячных.

Решение

Сопротивления r кабеля питания вычисляются по формуле:

$$r = \frac{\rho l}{s}.$$

Согласно закону Ома для участка цепи $I = U/r$, оно также может быть выражено в виде:

$$\frac{\rho l}{s} = r = \frac{U}{I}.$$

Отсюда окончательно получим:

$$\rho = \frac{Us}{Il}.$$

Погрешность $0,002$ Ом·мм^2/м.

Диапазоны

Величина	min	max	Шаг
l, м	5	8	$0,5$
s, мм^2	$1,6$	2	$0,1$
I, А	1	2	$0,1$
U, В	$0,1$	$0,2$	$0,01$

Ответ: $\rho = \dfrac{Us}{Il}$.

Задача II.2.4.2. Мороз (15 баллов)

Темы: закон Джоуля – Ленца.

Условие

На мобильной полярной станции вышла из строя основная система обогрева, и пришлось в срочном порядке подключать давно не использовавшуюся резервную. Резервный нагреватель представляет собой теплопроводящий корпус, защищающий катушку, на которую намотан провод длиной L и площадью поперечного сечения S, обеспечивающий эффективную конвекцию. Штатный кабель питания от нагревателя, к сожалению, был утерян, поэтому нагреватель пришлось подключить к сети при помощи стандартного резервного кабеля, имеющего площадь s и длину l. Определите отношение $P_\text{н}/P_\text{к}$ тепловой мощности, выделяющейся в нагревателе к тепловой мощности, выделяющейся в кабеле питания, если токонесущие жилы кабеля питания и провода нагревателя изготовлены из одного вещества. Дайте ответ с точностью до сотых.

Решение

Сопротивления R нагревателя и r кабеля питания вычисляются по формулам:

$$R = \rho\frac{L}{S}; \quad r = \rho\frac{l}{s},$$

где ρ — (одинаковое в обоих случаях) удельное сопротивление. Их отношение равно

$$\frac{R}{r} = \frac{Ls}{lS}.$$

Поскольку кабель и нагреватель включены в цепь последовательно, проходящие через них силы тока равны и тепловую мощность удобно искать по закону Джоуля – Ленца:

$$P_\text{н} = I^2R; \quad P_\text{к} = I^2r.$$

Таким образом, окончательно

$$\frac{P_{\text{н}}}{P_{\text{к}}} = \frac{\cancel{I^2}R}{\cancel{I^2}r} = \frac{Ls}{lS}.$$

Погрешность $0,05$.

Диапазоны

Величина	min	max	Шаг
L, м	15	20	1
l, м	5	8	$0,5$
S, мм^2	$2,5$	$3,5$	$0,2$
s, мм^2	$1,6$	2	$0,1$

Ответ: $\dfrac{P_{\text{н}}}{P_{\text{к}}} = \dfrac{Ls}{lS}$.

Задача II.2.4.3. Стопка (20 баллов)

Темы: конденсаторы.

Условие

На тонкий полимерный лист с обеих сторон напыляется металлическое покрытие. Измерения показывают, что емкость такого конденсатора равна C_0. Затем лист разрезают на n равных частей и складывают их в стопку. Определите емкость C такой стопки при подключении источника напряжения между самым верхним и самым нижним ее слоями. Ответ дайте в пФ (пикофарадах), округлив до целого.

Решение

Емкость плоского конденсатора задается формулой:

$$C = \frac{\varepsilon\varepsilon_0 S}{d},$$

поэтому при разрезании листа на n частей площадь и, соответственно, емкость C_1 каждой из этих частей, оказывается в n раз меньше исходной: $C_1 = C_0/n$.

В то же время стопка конденсаторов представляет собой цепь из n одинаковых конденсаторов, соединенных последовательно. Поскольку эффективная емкость C такой цепи может быть найдена по формуле:

$$\frac{1}{C} = \frac{1}{C_1} + \frac{1}{C_1} + \cdots + \frac{1}{C_1} = \frac{n}{C_1} = \frac{n^2}{C_0},$$

получим окончательно

$$C = \frac{C_0}{n^2}.$$

Погрешность 2 пФ.

Диапазоны

Величина	min	max	Шаг
C, нФ	2	6	0,5
n	4	10	1

Ответ: $C = \dfrac{C_0}{n^2}$. С учетом порядков $C = \dfrac{C_0}{n^2}[\cdot 10^3]$.

Задача II.2.4.4. Два колеса (23 баллов)

Темы: кинематика.

Условие

Два колеса двухколесного балансирующего робота, исследующего отдаленный астрономический объект с очень гладкой поверхностью, расположены на разных концах одной оси длиной l. Проведя сеанс связи с Землей, робот получил указание двигаться прямо и стал вращать колеса с одинаковой угловой скоростью. Однако в ходе посадки робота одно из колес было слегка повреждено, в связи с чем его радиус оказался на Δr меньше, чем радиус r другого. Найдите Δr, если известно, что вместо прямой робот начал двигаться по окружности радиуса R (измеренного по центру робота). Колеса робота не проскальзывают по поверхности, кривизна астрономического объекта пренебрежимо мала. Ответ дайте в мм, округлив до десятых.

Решение

При равных угловых скоростях ω линейные скорости колес имеют величины ωr и $\omega(r - \Delta r)$. При этом, поскольку робот движется по окружности, то за время, за которое внешнее колесо проходит дугу длиной $\alpha(R+l/2)$, внутреннее колесо проходит дугу $\alpha(R - l/2)$. Следовательно

$$\frac{\cancel{\omega}(r - \Delta r)}{\cancel{\omega} r} = \frac{R - l/2}{R + l/2}.$$

Преобразуя это выражение, получим окончательно:

$$\Delta r = \frac{2rl}{2R + l}.$$

Погрешность 0,2 мм.

Диапазоны

Величина	min	max	Шаг
r, см	40	60	4
l, см	80	120	10
R, м	80	120	5

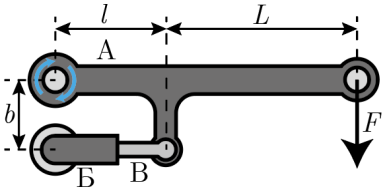
Ответ: $\Delta r = \dfrac{2rl}{2R+l}$. С учетом порядков $\Delta r = \dfrac{2rl[\cdot 10^1]}{2R[\cdot 10^2]+l}$.

Задача II.2.4.5. Рычаг (25 баллов)

Темы: статика, давление газа.

Условие

Подъемный рычаг А, геометрические параметры b, l, L которого изображены на рисунке, шарнирно соединен с приводящим его в движение поршнем В, входящим в цилиндр Б, внутри которого содержится идеальный газ. Пренебрегая весом самого рычага, определите избыточное (над атмосферным) давление p в цилиндре, которое необходимо для удержания нагрузки F, приложенной к концу рычага, если площадь поршня равна S. Ответ дайте в мПа, округлив до десятых.



Решение

Относительно шарнира рычага сила F имеет плечо $L+l$. Сила давления поршня равна pS и имеет плечо b. Тогда условие равновесия имеет вид:

$$bpS = (l+L)F.$$

Отсюда

$$p = \frac{(l+L)F}{bS}.$$

Погрешность $0,2$ МПа.

Диапазоны

Величина	min	max	Шаг
l, см	15	25	2
L, см	30	50	5
b, см	8	16	1
S, см^2	16	24	2
F, кН	2	3	0,5

Ответ: $p = \dfrac{(l+L)F}{bS}$. С учетом порядков $p = \dfrac{(l+L)F}{bS}[\cdot 10^1]$.

Задача II.2.4.6. (5 баллов)

Темы: физики России.

Условие

Один из основателей нелинейной оптики, новой ветви науки, продемонстрировавшей, что достаточно интенсивные лучи света могут взаимодействовать друг с другом и сами с собой, фокусироваться без линзы и неожиданно менять цвет. В его честь на территории Московского университета названа улица, лаборатория, спортивный клуб и несколько учебных аудиторий. Помимо выдающихся научных достижений, он также проявил себя как талантливый организатор, способствовал развитию кооперации исследователей самых разных направлений, включая биологию и экологию, а также был профессиональным альпинистом с двадцатилетним стажем. Трагедия в одном из горных походов, к несчастью, оборвала его выдающуюся жизнь.

1. Сергей Александрович Ахманов.
2. Рем Викторович Хохлов.
3. Анатолий Алексеевич Логунов.
4. Петр Николаевич Лебедев.
5. Александр Александрович Фридман.
6. Николай Алексеевич Умов.
7. Абрам Федорович Иоффе.
8. Роберт Эмильевич Ленц

Ответ: 2.

Третья волна. Задачи 8–9 класса

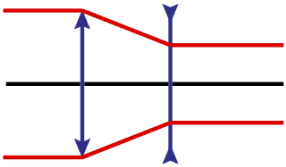
Задача II.2.5.1. Большой глаз (15 баллов)

Темы: геометрическая оптика.

Условие

Первый элемент системы ночного наблюдения имеет своей целью сузить пучок параллельных лучей, собираемых с большой площади, оставив его параллельным и неперевернутым. Он состоит из двух линз: собирающей и рассеивающей с оптическими силами D_1 и D_2 соответственно, установленных друг за другом и имеющих общую оптическую ось. Найдите расстояние между линзами. Ответ дайте в см, округлив до десятых.

Напоминание: оптической силой называется величина, обратная фокусному расстоянию линзы.



Решение

Изобразим описанную в условиях задачи оптическую схему. Из рисунка несложно видеть, что для описанного хода лучей фокусы двух линз должны совпадать. Но по определению оптической силы, фокусное расстояние линзы равно

$$F = \frac{1}{D}.$$

Подставляя в эту формулу D_1, D_2 и находя разницу между соответствующими расстояниями (верно учитывая знаки), получим:

$$l = \frac{1}{D_1} + \frac{1}{D_2}.$$

Погрешность $0,2$ см.

Диапазоны

Величина	min	max	Шаг
D_1, дптр	3	5	$0,2$
D_2, дптр	-10	-8	$0,2$

Ответ: $l = \frac{1}{D_1} + \frac{1}{D_2}$. С учетом порядков $l = \left(\frac{1}{D_1} + \frac{1}{D_2}\right)$ $[\cdot 10^2]$.

Задача II.2.5.2. Атмосфера (15 баллов)

Темы: давление газа, гидростатика.

Условие

Оцените массу атмосферы, окружающей планету земного типа радиусом R, если ускорение свободного падения на ее поверхности равно g, а атмосферное давление — p_0. Площадь сферы вычисляется по формуле $S = 4\pi R^2$. Ответ дайте в квинтлн (квинтиллионах) (10^{18}) кг, округлив до десятых.

Решение

Давление — это отношение силы к площади, перпендикулярно которой эта сила действует. В данном случае сила — общий вес атмосферы:

$$p_0 = \frac{mg}{S} = \frac{mg}{4\pi R^2}.$$

Для любой планеты земного типа толщина атмосферы пренебрежимо мала в сравнении с радиусом планеты, поэтому изменением ускорения свободного падения с высотой можно пренебречь. Получим окончательно

$$m = \frac{4\pi R^2 p_0}{g}.$$

Погрешность $0,1 \cdot 10^{15}$ т.

Диапазоны

Величина	min	max	Шаг
p_0, кПа	12	24	1
g, м/с^2	2	3,5	0,1
R, км	3600	4600	100

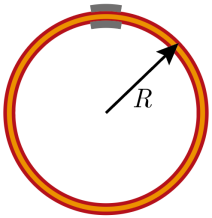
Ответ: $m = \frac{4\pi R^2 p_0}{g}$. С учетом порядков $m = \frac{4\pi R^2 p_0}{g}[\cdot 10^{-9}]$.

Задача II.2.5.3. Магниты (20 баллов)

Темы: центростремительное ускорение, динамика.

Условие

К тонкому ободу медного колеса радиусом R, расположенного в горизонтальной плоскости, снаружи и изнутри прикреплены два одинаковых маленьких магнитных датчика массой m каждый. Датчики держатся только за счет притяжения друг к другу. Колесо начинают постепенно раскручивать вокруг неподвижной оси, и в момент, когда период его вращения достигает величины T, внешний датчик отлетает от колеса. Определите силу давления внутреннего датчика на обод колеса непосредственно после этого. Ответ дайте в Н, округлив до целого. Длина окружности в 2π раз больше ее радиуса.



Решение

Внешний датчик отделяется в момент, когда сила F взаимодействия между магнитами оказывается недостаточна для того, чтобы создавать центростремительное ускорение $a = v^2/R$, где скорость v может быть найдена как отношение длины окружности к периоду обращения ($v = 2\pi R/T$):

$$F = ma = \frac{mv^2}{R} = \frac{4\pi^2}{T^2}mR.$$

Поскольку размеры датчиков и толщина колеса считаются пренебрежимо малыми, внутренний датчик продолжает двигаться с тем же ускорением, а значит, давить на колесо с той же силой F:

$$F = \frac{4\pi^2}{T^2}mR.$$

Погрешность 1 Н.

Диапазоны

Величина	min	max	Шаг
R, см	36	45	3
m, г	200	300	20
T, с	0,3	0,5	0,05

Ответ: $F = \frac{4\pi^2}{T^2}mR$. С учетом порядков $F = \frac{4\pi^2}{T^2}mR[\cdot 10^{-5}]$.

Задача II.2.5.4. Пот (20 баллов)

Темы: тепловые явления, кинематика жидкости.

Условие

Для охлаждения антропоморфного робота разрабатывается система, воспроизводящая потоотделение человека. По тонким капиллярам с площадью поперечного сечения S на поверхность робота поступает вода, которая затем растекается по «коже» робота и постепенно испаряется. Определите, какой должна быть постоянная скорость v течения воды в капилляре, если один такой капилляр должен обеспечивать отведение тепловой мощности N. Считайте, что вся вода успевает испариться. Удельная теплота парообразования воды $L = 2300\,\text{кДж/кг}$, ее плотность $\rho = 1000\,\text{кг/м}^3$. Ответ дайте в мм/с, округлив до целого.

Решение

Рассмотрим произвольный промежуток времени t. Согласно условиям, теплота Q, которую необходимо отвести от робота за это время, равна $Q = Nt$. Она может

быть определена через удельную теплоту парообразования и плотность воды как

$$Nt = Q = mL = \rho V L,$$

где объем испаренной жидкости V равен произведению расстояния d, которое прошла вода в капилляре на площадь его поперечного сечения:

$$V = dS = vtS.$$

Сопоставляя эти уравнения, получим

$$N\cancel{t} = \rho v\cancel{t} SL,$$

откуда окончательно

$$v = \frac{N}{\rho SL}.$$

Погрешность 1 мм/с.

Диапазоны

Величина	min	max	Шаг
N, Вт	10	20	1
S, мм^2	0,1	0,2	0,01

Ответ: $v = \dfrac{N}{\rho SL}$. С учетом порядков $v = \dfrac{N}{\rho SL}[\cdot 10^6]$.

Задача II.2.5.5. Вагончик (25 баллов)

Темы: закон Ома, кинематика.

Условие

Автоматизированный вагончик движется по двум длинным рельсам, по одному из которых на него подается, а с другого — снимается электрический ток. Каждый метр одного рельса имеет сопротивление r, а двигатели вагончика — полное сопротивление R. Источник питания подает на рельсы напряжение U_0, а для поддержания нормальной работы двигателей напряжение на них должно быть не ниже U. За какое время, стартовав от источника и двигаясь с постоянной скоростью v, вагончик сможет уехать достаточно далеко, чтобы двигатели перестали работать? Ответ дайте в с, округлив до целых.

Решение

Общее сопротивление цепи, в которую включены двигатели вагончика, равно

$$R_0 = 2r\frac{L}{l} + R,$$

где L — расстояние до источника, $l = 1$ м. Сила тока в цепи $I = U_0/R_0$, а напряжение на двигателе $U_д = IR$ (согласно закону Ома). Подставляя $U_д = U$, $L = vt$, получим:

$$U = U_0\frac{R}{R_0} = U_0\frac{R}{2rvt/l + R}.$$

Решая это уравнение относительно t, получим окончательный ответ:

$$t = \frac{R(U_0 - U)}{2rvU} \cdot 1\,\text{м}.$$

Погрешность 1 мин.

Диапазоны

Величина	min	max	Шаг
R, Ом	400	640	40
r, мОм	400	640	40
U, В	200	220	5
U_0, В	320	360	5
v, м/с	6	8	0,5

Ответ: $t = \dfrac{R(U_0 - U)}{2rvU} \cdot 1\,\text{м}$. С учетом порядков $t = \dfrac{R(U_0 - U)}{2rvU}[\cdot 10^3]$.

Задача II.2.5.6. (5 баллов)

Темы: физики России.

Условие

Вектор, описывающий плотность потока энергии в волнах, в англоязычной традиции носит фамилию английского физика, который в 1884 году вывел выражения для данного вектора в случае электромагнитных волн. Десятью годами ранее русский физик и философ вывел аналогичные уравнения для упругих волн, а потому в русской традиции вектор носит и его имя. Помимо выдающихся успехов в теории упругости, он сделал множество значимых открытий в оптике, а также выдвинул гипотезы о качественно правильном характере связи между массой и энергией, позже развитые в знаменитую формулу Эйнштейна $E = mc^2$. Выберите из приведенного списка выдающихся физиков имя этого русского ученого.

1. Владимир Александрович Фок.
2. Николай Алексеевич Умов.
3. Михаил Васильевич Остроградский.
4. Роберт Эмильевич Ленц.
5. Петр Николаевич Лебедев.
6. Александр Александрович Фридман.
7. Анатолий Алексеевич Логунов.
8. Рем Викторович Хохлов.

Ответ: 2.

Третья волна. Задачи 10–11 класса

Задача II.2.6.1. Поглотитель (15 баллов)

Темы: тепловые явления, радиоактивность.

Условие

Быстро движущиеся нейтроны, образующиеся при делении атомных ядер, взаимодействуя с молекулами воды, теряют энергию до значений, соответствующих энергии теплового движения, которая, как правило, много меньше энергии ядерного распада. Оцените, какое число нейтронов, движущихся со скоростью v, должно потерять свою энергию в кювете с водой в виде прямоугольного параллелепипеда со сторонами a, b, d, чтобы вода в этой кювете нагрелась на $\Delta t = 1$ °C. Удельная теплоемкость воды $c = 4,2$ кДж/(кг·°C), ее плотность $\rho = 1000$ кг/м^2. Масса нейтрона $m = 1,67 \cdot 10^{-27}$ кг. Ответ дайте в квдрлн (квадриллионах) (10^{15}) штук, округлив до целого.

Решение

Один нейтрон обладает кинетической энергией $K = mv^2/2$. Поскольку эта энергия на несколько порядков величины выше характерной энергии теплового движения, которую можно оценить как $kT \approx 4 \cdot 10^{-21}$ Дж для комнатной температуры, можно считать, что вся его энергия передается воде. Для нагрева воды на Δt требуется энергия, равная

$$Q = cm\Delta t = c\rho abd\Delta t.$$

Приравнивая ее nK, получим ответ:

$$n = 2\frac{c\rho abd\Delta t}{mv^2}.$$

Погрешность 3 квдрлн.

Диапазоны

Величина	min	max	Шаг
v, км/с	10 000	20 000	1000
a, см	40	60	5
b, см	10	16	2
d, см	10	16	2

Ответ: $n = 2\frac{c\rho abd\Delta t}{mv^2}$. С учетом порядков $n = 2\frac{c\rho abd\Delta t}{mv^2}[\cdot 10^{-24}]$.

Задача II.2.6.2. Дрейф (18 баллов)

Темы: электрический ток, МКТ.

Условие

В некотором полупроводнике концентрация подвижных электронов равна n. Какой должна быть дрейфовая (средняя) скорость электронов в элементе с площадью поперечного сечения S, изготовленном из этого полупроводника, чтобы сила электрического тока в данном сечении была равна I? Модуль заряда электрона $e = 1,6 \cdot 10^{-19}$ Кл. Ответ дайте в см/с, округлив до целого.

Решение

Рассмотрим произвольный отрезок времени t. При силе тока I за это время через сечение проводника должен пройти заряд $q = It$. Этот заряд равен произведению числа N пересекающих сечение электронов на заряд одного из них e. В то же время при скорости дрейфа v в среднем за время t электроны проходят расстояние $l = vt$. Это значит, что через сечение проходят электроны, содержащиеся в объеме $V = lS = vtS$. Из определения концентрации следует, что их общее число $N = nV$. Совмещая все эти выражения, получим

$$It = q = enV = envtS,$$

откуда легко выражается ответ:

$$v = \frac{I}{Sne}.$$

Погрешность 2 см/с.

Диапазоны

Величина	min	max	Шаг
$n, 10^{18}$ м$^{-3}$	6	9	1
S, мм^2	2,4	3	0,1
I, мкА	1,5	2,5	0,1

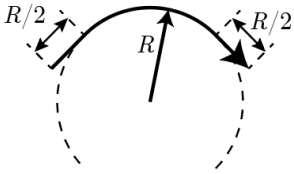
Ответ: $v = \frac{I}{Sne}$. С учетом порядков $v = \frac{I}{Sne}[\cdot 10^{-16}]$.

Задача II.2.6.3. Вираж (20 баллов)

Темы: кинематика.

Условие

Квадракоптер может двигаться по любой траектории с условием, что его скорость ни в одной точке этой траектории не превышает v, а его ускорение не превышает a (при этом направления скорости и ускорения не имеют значения). За какое минимальное время он сможет пройти по траектории, изображенной на рисунке, состоящей из двух прямолинейных участков и четверти окружности радиуса R, если в начальной и в конечной точках коптер должен иметь строго нулевую скорость? Ответ дайте в с, округлив до десятых.



Решение

Общая длина траектории $(4 + \pi/2)R < 6R$. Однако, даже двигаясь все время с набором скорости на расстоянии $6R$ с ускорением a, строго сонаправленным со скоростью, коптер успел бы разогнаться только до

$$v_1 = \sqrt{3aR}.$$

Поскольку эта скорость меньше v, ограничения на максимальную скорость в задаче на самом деле несущественны. В то же время ограничения на скорость накладывает центростремительное ускорение, с которым коптер проходит изогнутый участок: поскольку оно не должно превышать a, скорость на повороте не может быть выше $v_1 = \sqrt{aR}$. Но из формулы

$$\frac{v_1^2 - v_0^2}{2} = al = \frac{aR}{2}$$

можно видеть, что v_1 в точности совпадает со скоростью, которую успевает набрать квадрокоптер на первом прямолинейном и сбросить на втором прямолинейном участках. Таким образом, он должен двигаться равноускоренно на двух участках длиной $R/2$, каждый из которых отнимет время

$$t_1 = \frac{v_1}{a} = \sqrt{\frac{R}{a}}$$

и равномерно на повороте, на что уйдет время

$$t_2 = \frac{\pi R}{2v_1} = \frac{\pi R}{2\sqrt{aR}} = \frac{\pi}{2}\sqrt{\frac{R}{a}}$$

В общей сложности, подобный полет займет время $t_2 + 2t_1$:

$$t = \sqrt{\frac{R}{a}}\left(2 + \frac{\pi}{2}\right).$$

Погрешность $0,2$ с.

Диапазоны

Величина	min	max	Шаг
v, м/с	20	40	2
a, м/с^2	3	4	$0,2$
R, м	15	20	1

Ответ: $t = \sqrt{\frac{R}{a}}\left(2 + \frac{\pi}{2}\right).$

Задача II.2.6.4. Звездная величина (20 баллов)

Темы: фотометрия, геометрическая оптика.

Условие

Для получения качественного изображения тусклых звезд важно собрать в телескоп максимально возможную долю испущенного такой звездой излучения, что требует больших размеров основного зеркала или линзы телескопа. Яркость звезды в астрономии принято измерять в единицах видимой звездной величины (ВЗВ), увеличение ВЗВ на 1 означает уменьшение световой энергии, испускаемой звездой, в $k = \sqrt[5]{100}$ раз. При помощи телескопа-рефрактора с площадью главного зеркала s получено качественное изображение некоторой звезды. Найдите площадь S главного зеркала второго телескопа, позволяющего получить в таком же качестве изображение звезды, ВЗВ которой больше на n единиц. Ответ дайте в см^2, округлив до целого.

Решение

Полная мощность светового потока, попадающего в объектив телескопа, определяется как произведение площади S этого объектива на плотность этого потока I. Увеличение видимой звездной величины на n единиц означает уменьшение плотности светового потока в k^n раз. Следовательно, для компенсации этого изменения площадь объектива должна быть в k^n раз увеличена:

$$\frac{S}{s} = k^n.$$

Отсюда

$$S = sk^n = 100^{n/5}s.$$

Погрешность $5\,\text{см}^2$.

Диапазоны

Величина	min	max	Шаг
s, см^2	20	50	10
n	2	5	1

Ответ: $S = sk^n = 100^{n/5}s$.

Задача II.2.6.5. Большая линза (22 баллов)

Темы: фотометрия, масса и плотность.

Условие

Яркость звезды принято измерять в единицах видимой звездной величины (ВЗВ), одна единица которой означает изменение световой энергии, испускаемой звездой,

в $k = \sqrt[5]{100}$ раз. Определите, какой массы линзу объектива телескопа-рефрактора пришлось бы использовать при сохранении пропорций и материала линзы, чтобы получить изображение некоторой тусклой звезды в таком же качестве, как было получено изображение на n единиц ВЗВ более яркой звезды при помощи телескопа с объективом, масса которого составляла m. Ответ дайте в т, округлив до десятых.

Решение

Как и в предыдущей задаче, площадь объектива должна быть в k^n раз увеличена. Это требует изменения радиуса объектива в $\sqrt{k^n} = k^{n/2}$ раз. Но при сохранении пропорций объем любого тела и, следовательно, масса такого объектива должны измениться пропорционально кубу радиуса:

$$M = mk^{3n/2}.$$

Погрешность 0,1 т.

Диапазоны

Величина	min	max	Шаг
m, г	100	200	20
n	7	9	1

Ответ: $vM = k^{3n/2}m = 100^{0,3n}m$. С учетом порядков $M = 100^{0,3n}m[\cdot 10^{-6}]$.

Задача II.2.6.6. (5 баллов)

Темы: физики России.

Условие

Вектор, описывающий плотность потока энергии в волнах, в англоязычной традиции носит фамилию английского физика, который в 1884 году вывел выражения для данного вектора в случае электромагнитных волн. Десятью годами ранее русский физик и философ вывел аналогичные уравнения для упругих волн, а потому в русской традиции вектор носит и его имя. Помимо выдающихся успехов в теории упругости, он сделал множество значимых открытий в оптике, а также выдвинул гипотезы о качественно правильном характере связи между массой и энергией, позже развитые в знаменитую формулу Эйнштейна $E = mc^2$. Выберите из приведенного списка выдающихся физиков имя этого русского ученого.

1. Владимир Александрович Фок.
2. Александр Александрович Фридман.
3. Рем Викторович Хохлов.
4. Николай Алексеевич Умов.
5. Михаил Васильевич Остроградский.
6. Роберт Эмильевич Ленц.

7. Петр Николаевич Лебедев.
8. Анатолий Алексеевич Логунов.

Ответ: 4.

Инженерный тур

Задачи инженерного тура готовят участников ко второму и заключительному этапам. Они представляют собой облегченные декомпозированные компетентностные подзадачи заключительного этапа: подбор ВМГ, сборка квадрокоптера, настройка и программирование.

Задача II.3.1. Сборка (11 баллов)

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

Условие

Посмотрите внимательно на рисунок II.3.1 и подключите сигнальные провода регуляторов оборотов двигателей в требуемые пины полетного контроллера `COEX PIX`.

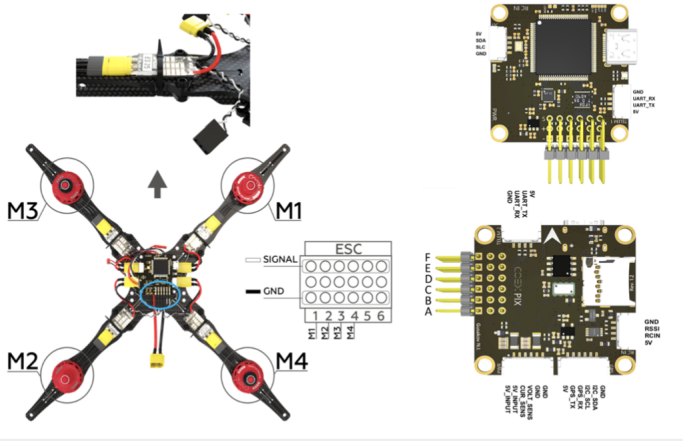


Рис. II.3.1. Подключение сигнальных проводов

	A	*B*	*C*	*D*	*E*	*F*
M_1						
M_2						
M_3						
M_4						

Решение

Подключение сигнальных проводов регуляторов оборотов двигателей происходит по следующей схеме:

1. M_1 — пин A;
2. M_2 — пин B;
3. M_3 — пин C;
4. M_4 — пин B.

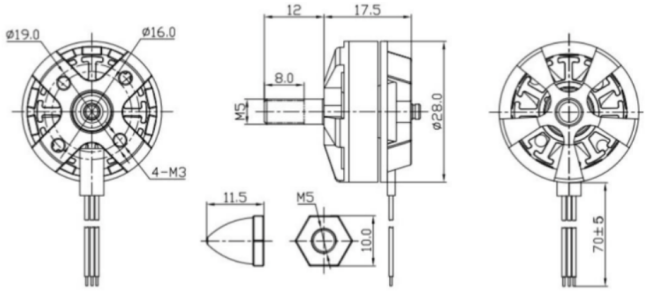
Ответ.

	A	B	C	D	E	F
M_1	+					
M_2		+				
M_3			+			
M_4				+		

Задача II.3.2. Подбор аккумулятора (10 баллов)

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

Условие



MOTOR PERFORMANCE DATA

MODEL	KV (rpm/V)	Voltage (V)	Prop	Load Current (A)	Pull (g)	Power (W)	Efficiency (g/W)	Lipo Cell	Weight (g)Approx
D2204	2300	7.4	6x3	5.8	220	43	5.1	2-3S	25
		11.1	5043	9.5	365	105	3.5		
			6x3	10.5	408	117	3.5		

Рис. II.3.2. Моторы

Подберите аккумулятор для гексакоптера, на котором установлены моторы D2204 с пропеллерами 5043.

Время полета данного коптера должно быть не менее 2 мин 30 с.

Выберите правильный вариант ответа, для расчета используете ток нагрузки:

1. 4s 2300 mAh 40c;
2. 3s 2600 mAh 15c;
3. 3s 2600 mAh 20c;
4. 3s 2500 mAh 25c.

Решение

Согласно таблице на рисунке II.3.2, ток нагрузки на один мотор D2204 с пропеллером 5043 составляет: $9,5$ А.

Для гексакоптера (шесть моторов): $9,6A \cdot 6 = 57$ А.

Максимальный разрядный ток:

1. для аккумулятора №1: $2,3 \cdot 40 = 92$ А;
2. для аккумулятора №2: 39 А;
3. для аккумулятора №3: 52 А;
4. для аккумулятора №4: $62,5$ А.

Вариант №1 не является верным, так как данные моторы не предназначены для использования с $4S$ аккумулятором.

Время полета с аккумулятором №4 равно: $\frac{2,5}{57} \cdot 60 \cdot 60 = 157,9$ с (2 мин 38 с).

Ответ: 4.

Задача II.3.3. Полетный контроллер (10 баллов)

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

Условие

Перед вами распиновка полетного контроллера `Pixracer`.

Рис. II.3.3. Подключение полетного контроллера

Отметьте верный вариант. Укажите вариант ответа, в котором обозначен номер разъем для подключения питания полетного контроллера:

1. №1;
2. №2;
3. №3;
4. №4;
5. №5;
6. №6;
7. №7;
8. №8;
9. №9.

Ответ: 7.

Задача II.3.4. К полету готовы? (10 баллов)

Темы: базовые навыки работы с летающими робототехническими системами, предполетная подготовка.

Условие

Вы сегодня работаете из дома, а стажер сообщил о том, что справился со сборкой квадрокоптера и готов к первому полету.

Не допустил ли он ошибку при сборке или настройке?

Проверим готовность по фотографии, представленной на рисунке II.3.4.

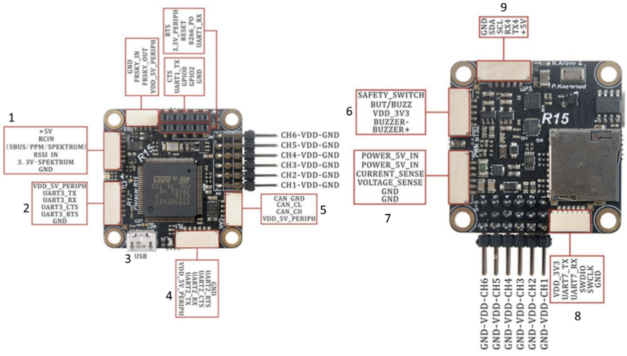


Рис. II.3.4. Пульт радиоуправления

К полету готов? Отметьте верный вариант:

1. выставлен неверный протокол передачи;
2. потеряна связь с радиоприемником;
3. тримы сбиты в крайние положения;

4. разрядился источник питания пульта радиоуправления.

Решение

Для решения задачи необходимо внимательно рассмотреть рисунок в условиях задачи. На рисунке можно заметить, что источник питания пульта радиоуправления полностью разряжен.

Ответ: 4.

Задача II.3.5. Подбор оптимальной ВМГ (18 баллов)

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

Условие

Соберите беспилотник мультироторного типа из деталей имеющихся на складе (таблица №II.3.1).

Техническое задание

Дрон должен иметь возможность поднимать полезную нагрузку в виде камеру весом 350 г, время полета на одном аккумуляторе должно быть максимально возможным. Временем полета считается зависание беспилотника на месте с полезной нагрузкой. Вес беспилотника принять за вес компонентов в его составе. Учитывать необходимо только компоненты находящиеся в таблице. Вес рамы не учитывать.

В качестве ответа выпишите буквы, под которыми расположены необходимые компоненты. Например, *ECAB*.

Таблица II.3.1: Имеющееся на складе оборудование

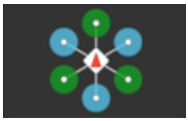
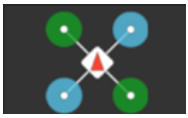
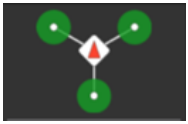
	Двигатель	*ESC*	*LiPo*	*Frame*
A	`+XING2 1404 FPV Motor 3000kv` Количество: 6 шт.	`T-motor CINE55A 8S 8IN1 32BIT` Количество: 2 шт.	`Betafpv BT2.0 300mAh 1S 30C Battery` Количество: 50 шт.	
B	`Betafpv 7x16mm Brushed Motors` Количество: 7 шт.	`+T-motor MINI F45A 6S 4IN1` Количество: 2 шт.	`+Samsung 30Q 18650 3000mah LiIon 4s2p` Количество: 3 шт.	
C	`T-motor U15II KV80` Количество: 14 шт.	`T-motor AT 115A 14S` Количество: 24 шт.	`TATTU 16000MAH 22.2V 30C` Количество: 5 шт.	

Таблица II.3.1: Имеющееся на складе оборудование

	Двигатель	*ESC*	*LiPo*	*Frame*
E	`T-motor P1604 2850 KV` Количество: 3 шт.	`BLITZ F411 1S Whoop AIO` Количество: 2 шт.	`ONBO 850mAh 3S 45C Lipo` Количество: 9 шт.	

Ответ: ABBC.

Задача II.3.6. Меряем напряжение (27 баллов)

Темы: программирование на Python, OpenCV.

Условие

Одним из применений дронов в области нефтегазового дела является проверка значений аналоговых датчиков, которые не имеют связи с внешним миром и отображают значения на стрелочном циферблате.

Рис. II.3.5. Манометр

Сейчас перед вами стоит подобная задача, но считывать значения мы будем с советского лабораторного стрелочного вольтметра.

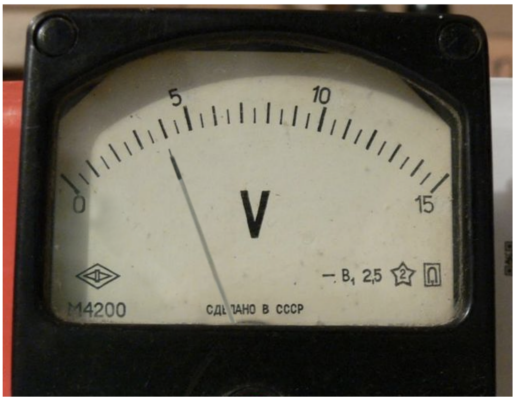


Рис. II.3.6. Вольтметр

С использованием алгоритма технического зрения распознайте, какое напряжение показывает вольтметр. Гарантируется, что стрелка находится в отдалении от деления на шкале не более чем на 1,5 °.

Формат входных данных

Ссылка на изображение вольтметра, напряжение с которого необходимо распознать. Гарантируется, что ссылка находится в домене `stepik.org` и доступна для загрузки во время проверки кода. Функция, позволяющая скачать изображение и импортировать в `OpenCV` приложена к заданию.

Формат выходных данных

Напряжение на вольтметре U; $0 \leqslant U \leqslant 150 \leqslant U \leqslant 15$; U кратно 0,5.

Десятичная часть обязательна, при вводе отделять точкой.

Примеры

Пример №1

Стандартный ввод

`https://disk.yandex.ru/i/wbQwJG3_mXz_Jg`.

Стандартный вывод

```
4.0
```

Пример №2

Стандартный ввод

`https://disk.yandex.ru/i/bOG29H1RPVeQPA`.

Стандартный вывод

```
10.5
```

Пример программы-решения

Ниже представлено решение на языке Python 3.

```python
from urllib.request import urlopen
import cv2  # v. 3.4.4
import numpy as np


def get_image(url):
    resp = urlopen(url)
    image = np.asarray(bytearray(resp.read()), dtype="uint8")
    image = cv2.imdecode(image, cv2.IMREAD_COLOR)
    return image

im = get_image(input())

def calc_angle(image: np.ndarray) -> float:
    M = cv2.moments(image)

    cX = int(M["m10"] / M["m00"]) - image.shape[1] // 2
    cY = (image.shape[0] - 50) - int(M["m01"] / M["m00"])
    vector = np.array([cX, cY])
    unit_vector = np.array([0, 1])

    angle = np.arctan2(np.cross(vector, unit_vector), np.dot(vector, unit_vector))
    return angle

arrow = cv2.inRange(im, (110, 130, 125), (120, 140, 140))

mask = np.zeros_like(arrow)
cv2.circle(mask, (im.shape[1] // 2, im.shape[0] - 50), 140, 255, 10)

andd = cv2.bitwise_and(arrow, mask)
andd = cv2.erode(andd, np.ones((2, 2), np.uint8))

print(round(calc_angle(andd) / 0.06 + 15 + 0.4) / 2)
```

Задача II.3.7. Порисуем (14 баллов)

Темы: программирование на Python, OpenCV.

Условие

Одной из ключевых задач для автономных дронов является способность корректно воспринимать окружающий мир и делать на основе этого адекватные решения. Подготовка и адаптация алгоритмов технического зрения требует тщательного тестирования. Одним из способов проверки и отладки таких алгоритмов является визуализация результатов на изображениях. Рисование различных геометрических фигур на изображении (обведение в круг/прямоугольник детектируемого объекта) позволит участникам оценить корректность и точность их алгоритмов в различных

условиях, что является важной ступенью на пути к созданию эффективного и надежного решения для автономных летающих роботов.

В этой задаче нарисуйте на пустом холсте базовые геометрические фигуры с использованием `OpenCV` окружность, прямоугольник и точку.

Под пустым холстом подразумевается 8-битное (диапазон цвета 0–255) черно-белое изображение размерности 100×100 пкс.

Формат входных данных

Первая строка содержит целое число k ($1 \leqslant k \leqslant 7$) — количество фигур, которые необходимо нарисовать. Затем идут строки одного из трех видов:

- `circle <cx> <cy> <radius> <color> <thickness>`;
- `rectangle <ulx> <uly> <drx> <dry> <color> <thickness>`;
- `point <px> <py> <color>`.

Где:

- $color$ ($0 \leqslant color \leqslant 255$) — 8-битный цвет фигуры (0 — черный; 255 — белый);
- $thickness$ ($1 \leqslant thickness \leqslant 10$) — толщина фигуры в пкс;
- cx ($2 \leqslant cx \leqslant 98$) — координата центра окружности по горизонтальной оси (начиная с левого верхнего угла);
- cy ($2 \leqslant cy \leqslant 98$) — координата центра окружности по вертикальной оси (начиная с левого верхнего угла);
- $radius$ ($1 \leqslant adius \leqslant 50$) — радиус окружности в пикселях;
- ulx ($0 \leqslant ulx \leqslant 90$) — координата левого верхнего угла прямоугольника по горизонтальной оси (начиная с левого верхнего угла);
- uly ($0 \leqslant uly \leqslant 90$) — координата левого верхнего угла прямоугольника по вертикальной оси (начиная с левого верхнего угла);
- drx ($1 \leqslant drx \leqslant 99$) — координата правого-нижнего угла прямоугольника по горизонтальной оси (начиная с левого верхнего угла);
- dry ($1 \leqslant dry \leqslant 99$) — координата правого-нижнего угла прямоугольника по вертикальной оси (начиная с левого верхнего угла);
- px ($0 \leqslant px \leqslant 99$) — координата точки по горизонтальной оси (начиная с левого верхнего угла);
- py ($0 \leqslant py \leqslant 99$) — координата точки по вертикальной оси (начиная с левого верхнего угла).

Формат выходных данных

100 строк, каждая строка содержит 100 чисел (0...255), каждое число кодирует свой пкс.

Карта расположения пикселей (x, y):

$$\begin{matrix} (0,0) & \cdots & (99,0) \\ \vdots & \ddots & \vdots \\ (0,99) & \cdots & (99,99) \end{matrix}$$

Примеры

Пример №1

Стандартный ввод

```
3
rectangle 59 76 64 91 71 10
point 48 35 189
rectangle 1 62 59 90 211 2
```

Стандартный вывод

https://disk.yandex.ru/d/csptIWzqXqzdGw.

Пример программы-решения

Ниже представлено решение на языке Python 3.

```
import cv2
import numpy as np

def draw(img, figure, *args):
    if figure == 'circle':
        center_x, center_y, radius_px, color, thickness_px = args
        img = cv2.circle(img, (center_x, center_y), radius_px, color,
        ↪  thickness_px)
    if figure == 'rectangle':
        upleft_x, upleft_y, downright_x, downright_y, color, thickness_px = args
        img = cv2.rectangle(img, (upleft_x, upleft_y), (downright_x, downright_y),
        ↪  color, thickness_px)
    if figure == 'point':
        x, y, color = args
        img[y, x] = color
    return img

im = np.zeros((100,100), 'uint8')
k = int(input())

for _ in range(k):
    fig, *args = input().split()
    args = list(map(int, args))

    im = draw(im, fig, *args)

for i in range(100):
    for j in range(100):
        print(im[i][j], end=' ' if j != 99 else '')
    print()
```

Работа наставника НТО на втором отборочном этапе

На втором отборочном этапе участникам предлагаются индивидуальные и командные задачи в рамках выбранных профилей. Для подготовки к нему наставник может использовать следующие рекомендуемые форматы и мероприятия:

- Подготовка по образовательным программам НТО по ряду технологических направлений.
- Разбор задач второго отборочного этапа НТО прошлых лет.
- Прохождение онлайн-курсов по разбору задач НТО прошлых лет.
- Прохождение онлайн-курсов, рекомендованных разработчиками профилей.
- Разбор материалов для подготовки к профилям.
- Практикумы. Для организации практикумов возможно использовать разные подходы или их комбинации:
  - Проведение практикумов по описаниям на страницах профилей и материалов для подготовки.
  - Декомпозиция задач заключительных этапов прошлых лет для выделения наиболее актуальных элементов и их изучения.
  - Анализ технических знаний и навыков (hard skills), требуемых для конкретного профиля, и самостоятельная разработка или поиск занятия для развития наиболее актуальных из них.
  - Посещение практикумов на площадках подготовки и онлайн-мероприятий от разработчиков профилей. Объявления о таких мероприятиях публикуются в группах НТО в VK и в телеграм-канале для наставников НТО (https://t.me/kruzhok_association).

Второй отборочный этап

Участники заключительного этапа разрабатывают систему автоматизированного мониторинга объектов городского строительства и инфраструктуры при помощи квадрокоптера.

Для успешного решения поставленной задачи необходимо:

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

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

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

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

Компетенции необходимые участникам для решения задач второго этапа:

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

Индивидуальные задачи

Задача IV.1.1. Сборка квадрокоптера (8 баллов)

Темы: базовые навыки работы с летающими робототехническими системами.

Условие

Ознакомьтесь с видео и отметьте все электронные компоненты и части коптера, которые были подключены или установлены неправильно (ссылка на видео: https://www.youtube.com/embed/MNbqA2HherA):

1. подключение шлейфа радиоприемника к полетному контроллеру;

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

Решение

Для решения задачи необходимо внимательно посмотреть видео по сборке квадрокоптера из образовательного курса: https://stepik.org/lesson/769532/step/7?unit=771988 или ознакомиться с документацией по сборке квадрокоптера: https://clover.coex.tech/ru/assemble_4_2_ws.

Ответ: 1, 5, 6, 8, 15, 17, 18, 19, 20, 21.

Задача IV.1.2. Настройка квадрокоптера (5 баллов)

Темы: базовые навыки работы с летающими робототехническими системами.

Условие

Ознакомьтесь с видео и отметьте все ошибки, допущенные при настройке квадрокоптера (ссылка на видео: https://www.youtube.com/embed/M9rCzhQYY14):

1. ошибка загрузки прошивки в полетный контроллер;
2. ошибка в выборе конфигурации рамы квадрокоптера;
3. ошибка при калибровке компаса;
4. ошибка при калибровке гироскопа;
5. ошибка при калибровке акселерометра;
6. ошибка при калибровке уровня горизонта;
7. ошибка при установке ориентации полетного контроллера;
8. ошибка при калибровке радиоаппаратуры управления;
9. ошибка при настройке режимов полетного контроллера;
10. ошибка при назначении аварийного отключения моторов;
11. ошибка при настройке параметров питания;
12. ошибки при калибровке регуляторов (ESC).

Решение

Для решения задачи необходимо внимательно посмотреть видео по сборке квадрокоптера из образовательного курса https://stepik.org/lesson/769532/step/7?unit=771988 или ознакомиться с документацией по сборке квадрокоптера: https://clover.coex.tech/ru/assemble_4_2_ws.

Ответ: 7, 10, 11.

Задача IV.1.3. Автономный полет. настройка образа (6 баллов)

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

Условие

Ознакомьтесь с видео и отметьте все ошибки, допущенные при настройке образа квадрокоптера(ссылка на видео: https://www.youtube.com/embed/oxBCuTof674):

1. настройка параметра `aruco`;
2. настройка параметра `aruco_detect`;
3. настройка параметра `aruco_map`;
4. настройка параметра `aruco_vpe`;
5. настройка параметра `map`;
6. настройка размера `aruco` маркера по умолчанию;
7. настройка параметра `direction_z`;
8. настройка параметра `direction_y`;
9. настройка параметров карты: длина маркера;

10. настройка параметров карты: количество маркеров по оси X;
11. настройка параметров карты: количество маркеров по оси Y;
12. настройка параметров карты: расстояние между центрами меток по оси X;
13. настройка параметров карты: расстояние между центрами меток по оси Y;
14. настройка параметров карты: номер первого маркера;
15. перезагрузка пакетов клевера на образе.

Примечание: сборка квадрокоптера является стандартной.

Решение

Для решения задачи необходимо внимательно посмотреть видео по сборке квадрокоптера из образовательного курса https://stepik.org/lesson/769532/step/7?unit=771988 или ознакомиться с документацией по сборке квадрокоптера https://clover.coex.tech/ru/assemble_4_2_ws.

Ответ: 1, 5, 8, 10, 11, 14.

Задача IV.1.4. PID-регулятор. Настройка коптера (5 баллов)

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

Условие

Ознакомьтесь с видео и отметьте, какие составляющие коэффициентов PID-регулятора были подобраны неправильно для квадрокоптера (ссылки на видео: №1: https://www.youtube.com/embed/nrptLn66VvA, №2: https://www.youtube.com/embed/Xn8bi4P0Wtc, №3: https://www.youtube.com/embed/yeOmeApPpao, №4: https://www.youtube.com/embed/llW9TQMOQyQ):

1. Высокие значения коэффициентов `ROLLHRATE_P` или `ROLLHRATE_D`.
2. Малые значения коэффициентов `PITCHRATE_P` или `PITCHRATE_D`.
3. Малые значения коэффициентов `ROLLHRATE_P` или `ROLLHRATE_D`.
4. Высокие значения коэффициентов `PITCHRATE_P` или `PITCHRATE_D`.

Примечание: сборка квадрокоптера является стандартной; для демонстрации менялся один из коэффициентов в минимальное или максимальное значение, рекомендованное в прошивке PX4 полетного контроллера.

Решение

Для решения задачи необходимо внимательно прочитать документацию по сборке квадрокоптера: https://clover.coex.tech/ru/assemble_4_2_ws.

Ответ: видео №1 — вариант №2, видео №2 — вариант № 1, видео №3 — вариант №4, видео №4 — вариант №3.

Задача IV.1.5. Подбор оптимальной ВМГ (8 баллов)

Темы: работа с летающими робототехническими системами, конструирование, сборка квадрокоптера.

Условие

Вашей компании пришел срочный заказ на изготовление дронов, которые должны выполнять задачи по доставке малогабаритных грузов. В связи со срочностью заказа вы можете использовать только компоненты, имеющиеся на складе. Нужно подобрать оптимальные компоненты, чтобы получившиеся аппарат(ы) удовлетворяли следующим требованиям.

Требования:

- Возможность перевозки груза 20 кг за один раз.
- Груз может перевозить один или несколько одинаковых беспилотников.
- Минимальное время полета с грузом должно составлять 50 мин.
- Компоненты должны подходить друг к другу.
- Цена итогового решения должна быть минимальной.

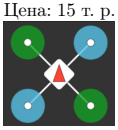
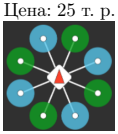
Дополнительные условия:

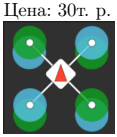
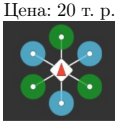
- Процент газа зависания беспилотника с грузом должен быть в промежутке от 50–60%.
- Ток, потребляемый аппаратом во время полета, на 20% больше тока зависания.
- Для решения задачи рекомендуется использовать технические параметры производителей.

В ответе запишите буквы, соответствующие компонентам в порядке, соответствующем таблице и цену готового решения.

Пример записи ответа:

1. CDBABD.
2. 123.

№	Рама	Мотор	PROP	ESC	LIPO	Полетный контроллер
A	Количество: 1 шт. Цена: 15 т. р. 	T-motor U13II KV65 Количество: 4 шт. Цена: 40 т. р.	T-motor G30x10.5 Количество: 8 шт. Цена: 16 т. р.	FVT LittleBee 20A ESC BLHeli-S OPTO 2-4S Количество: 8 шт. Цена: 1 т. р.	Li-Ion 21700 Vapcell INR21700 F52 12s7c Масса: 6 кг Количество: 3 шт. Цена: 52 т. р.	iFlight BLITZ Mini F7 V1.1 Количество: 3 шт. Цена: 5 т. р.
B	Количество: 4 шт. Цена: 25 т. р. 	T-motor U8Lite KV100 Количество: 9 шт. Цена: 30 т. р.	HQ X-Class Prop 13X9X3V2R Количество: 36 шт. Цена: 1 т. р.	Happymodel SuperX HD ELRS AIO 12A 1-2s Количество: 5 шт. Цена: 5 т. р.	TATTU 22000MAH 22.2V 30C 6S1P Масса: 3 кг Количество: 5 шт. Цена: 65 т. р.	HEX Pixhawk 2.1 Cube Orange+GPS HEX HERE GNSS V3 Количество: 2 шт. Цена: 46 т. р.

№	Рама	Мотор	PROP	ESC	LIPO	Полетный контроллер
C	Количество: 3 шт. Цена: 30т. р. 	T-motor P80III KV100 Количество: 7 шт. Цена: 20 т. р.	T-motor G40x13.1 Количество: 8 шт. Цена: 45 т. р.	T-motor FLAME 60A 12S Количество: 6 шт. Цена: 10 т. р.	CNHL 2000mah 6S 100C Масса: 1 кг Количество: 11 шт. Цена: 4т. р.	Happymodel CrazyF405 ELRS HD Количество: 6 шт. Цена: 5 т. р.
D	Количество: 2 шт. Цена: 20 т. р. 	TOA BI-TURBINE 2507-1450KV Количество: 24 шт. Цена: 3 т. р.	T-motor G27x8.8 Количество: 12 шт. Цена: 18 т. р.	T-motor ALPHA 40A 6S Количество: 20 шт. Цена: 6 т. р.	Samsung INR21700-50E Li-ion 12s18c Масса: 15 кг Количество: 1 шт. Цена: 100 т. р.	Holybro Pixhawk 6C + PM02 V3 Power Module Количество: 8 шт. Цена: 58 т. р.

Ответ:

1. DCACDB;
2. 442.

Задача IV.1.6. Проектирование многогранника Каплера (6 баллов)

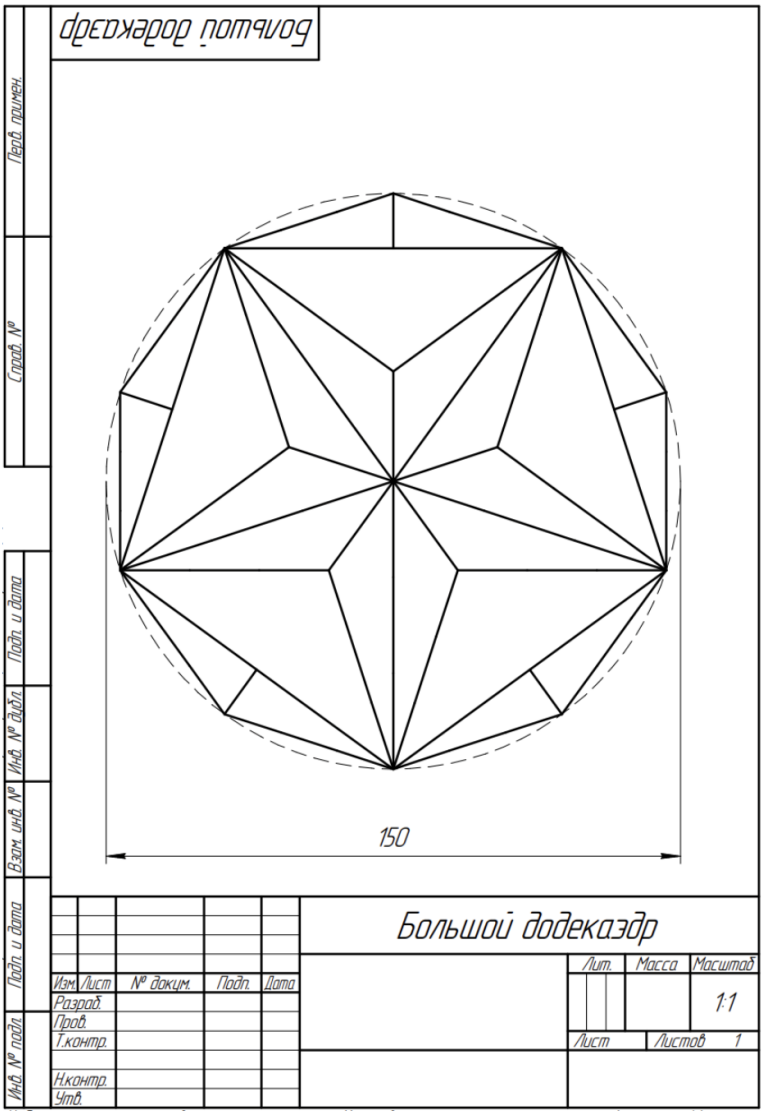
Темы: 3D-моделирование.

Условие

Спроектируйте геометрическую фигуру большой додекаэдр и рассчитайте его объем.

В качестве ответа напишите объем фигуры в см^3, значение округлите до целых.

Ответ: 1059 ± 1.



Задача IV.1.7. Проектирование 3D-модели винта квадрокоптера (10 баллов)

Темы: 3D-моделирование.

Условие

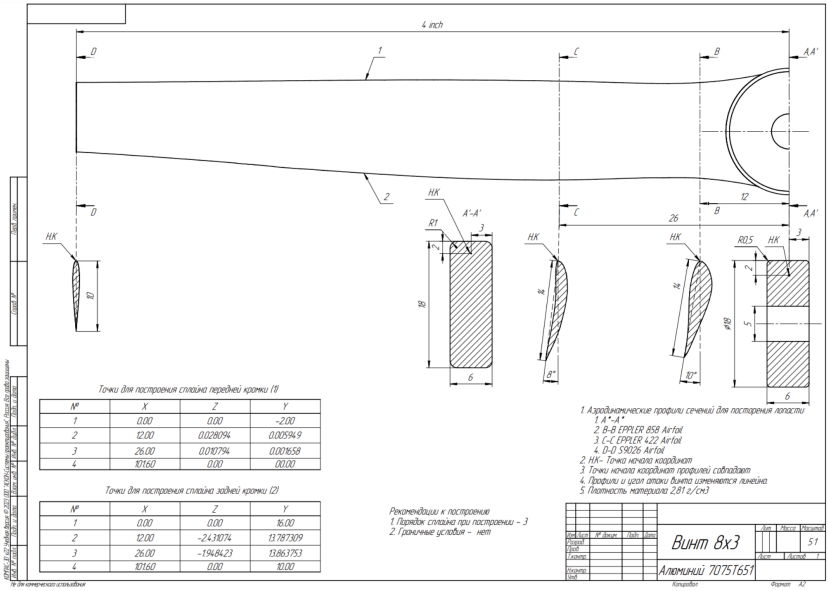
Спроектируйте винт по чертежу, подсчитайте массу и объем детали.

Ответ запишите в г и мм^3, значение округлите до целых.

В ответ необходимо внести массу детали и объем детали.

Задание необходимо выполнять в T-FLEX. Возможные альтернативные варианты:

- SolidWorks.
- Компас-3D v21.



Ответ:

- $11 \pm 0,4$ г;
- 3915 ± 20 мм^3.

Задача IV.1.8. Мониторинг территории (9 баллов)

Темы: программирование на Python, OpenCV.

Условие

Для проведения эффективного мониторинга территорий с дрона специалистам требуется разработать систему, способную распознавать различные формы и цвета объектов на земле. Это позволит быстро и автоматически идентифицировать интере-

сующие объекты, такие как автомобили, здания, растения или другие стационарные и мобильные объекты.

Перед вами стоит задача разработать программу для распознавания простых объектов и классификации их форм и цветов.

В задаче используются следующие простые объекты (в скобках указано название, которое требуется использовать для выходных данных):

- треугольник (`triangle`);
- прямоугольник (`rectangle`);
- пятиугольник (`pentagon`);
- шестиугольник (`hexagon`);
- звезда (`star`);
- круг (`circle`).

В задаче используются следующие цвета (указано название цвета и его BGR значения).

```
COLORS = {
#   name        B    G    R
    "red":    (0,   0,   255),
    "green":  (0,   255, 0),
    "blue":   (255, 0,   0),
    "yellow": (0,   255, 255),
    "orange": (0,   165, 255),
    "black":  (0,   0,   0),
    "gray":   (128, 128, 128)
}
```

Формат входных данных

1 строка с ссылкой на изображение, содержащее N фигур. $10 \leqslant N \leqslant 80$.

Формат выходных данных

N строк формата: `<fig_name> <color_name> <Cx> <Cy>`, где:

- `<fig_name>` — название фигуры (см. список фигур выше);
- `<color_name>` — текстовое название цвета (см. список цветов выше);
- `<Cx> <Cy>` — координаты центра масс фигуры (x — горизонтальная ось, y — вертикальная).

Допускается ошибка в координатах в 1 пкс по каждой оси.

Примеры

Пример №1

Стандартный ввод

https://disk.yandex.ru/d/8FRimVsKm4gRlw/form_color_200826e1.png

Стандартный вывод

```
triangle green 27 57
triangle gray 445 286
pentagon yellow 442 448
triangle blue 120 339
rectangle blue 179 492
circle red 285 138
triangle orange 574 54
star orange 74 181
hexagon orange 320 534
star black 243 292
```

Пример №2

Стандартный ввод

https://disk.yandex.ru/d/8FRimVsKm4gRlw/form_color_0eb2ac7e.png

Стандартный вывод

```
star blue 290 332
pentagon red 162 543
pentagon red 123 75
hexagon yellow 36 545
rectangle green 84 321
rectangle gray 492 424
star green 367 163
triangle green 497 276
star gray 256 103
circle gray 528 62
```

Пример программы-решения

Ниже представлено решение на языке Python 3.

```python
from urllib.request import urlopen
import cv2  # v. 3.4.4
import numpy as np

def get_image(url):
    resp = urlopen(url)
    image = np.asarray(bytearray(resp.read()), dtype="uint8")
    image = cv2.imdecode(image, cv2.IMREAD_COLOR)
    return image

im = get_image(input())

COLORS = {
    "red": (0, 0, 255),
    "green": (0, 255, 0),
    "blue": (255, 0, 0),
    "yellow": (0, 255, 255),
    "orange": (0, 165, 255),
    "black": (0, 0, 0),
```

```
    "gray": (128, 128, 128)
}

rCOLORS = {v: k for k, v in COLORS.items()}


figures = 255 - cv2.inRange(im, (250, 250, 250), (255, 255, 255))

_, cnt, _ = cv2.findContours(figures, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)
for c in cnt:
    m = cv2.moments(c)
    if m["m00"] > 100:
        peri = cv2.arcLength(c, True)
        circularity = m["m00"] / (np.pi * ((peri / (2 * np.pi)) ** 2))

        approx = cv2.approxPolyDP(c, 0.03 * peri, True)
        angles = len(approx)

        cX = int(m["m10"] / m["m00"])
        cY = int(m["m01"] / m["m00"])

        figure = None
        if angles == 3:
            figure = "triangle"
        elif angles == 4:
            figure = "rectangle"
        elif angles == 5:
            figure = "pentagon"
        elif angles == 6:
            figure = "hexagon"
        elif angles == 10:
            figure = "star"
        elif circularity > 0.8:
            figure = "circle"

        color = rCOLORS[tuple(im[cY, cX])]

        print(figure, color, cX, cY)
```

Задача IV.1.9. Сломалась ли стена (15 баллов)

Темы: машинное обучение, OpenCV.

Условие

Каменная и кирпичная кладка широко распространена как в исторической, так и в современной архитектуре. Кроме того, в настоящее время наблюдается всплеск интереса к использованию каменной кладки для создания устойчивой инфраструктуры в будущем. Одной из причин длительного срока службы каменной кладки является ее способность к постепенному ремонту; жертвенный раствор и модульный характер отдельных блоков каменной кладки означают, что ее обслуживание обходится дешевле, чем монолитных материалов, таких как бетонные плиты. Однако кладка подвержена растрескиванию под воздействием тепловых нагрузок, возникающих в результате циклов замораживания и оттаивания или несовместимости материалов, гидроскопических нагрузок от осадков или поднимающейся влаги, а также механических нагрузок от оседаний или землетрясений.

Дроны могут эффективно служить для мониторинга кирпичной или каменной кладки. Предлагаем вам разработать программу, использующую машинное обучение и компьютерное зрение, выполняющую классификацию испорченных участков кладки на основе изображений, полученных с дронов. Для обучения модели вам необходимо использовать тренировочный датасет: https://disk.yandex.ru/d/8FRimVsKm4gRlw/train.zip.

По ссылке находится архив, содержащий в себе две папки: 0 и 1. В папке 0 содержатся изображения кладки без повреждений, а в папке 1 — с повреждениями.

Датасет содержит изображения, извлеченные из аэрофотоснимков зданий Speicherstadt и Kesselhaus в Гамбурге, предоставленных администрацией города. Исходные 834 изображения высокого разрешения (5472×3648 пкс) были разделены на более мелкие изображения (227×227 пкс).

На рисунке показан пример изображений в датасете.

Вам необходимо построить и обучить (или использовать предобученную) модель, способную классифицировать участок стены с повреждениями. Для проверки модели будет использоваться специальный тестовый набор изображений, имеющий ту же природу.

Вам необходимо скачать тестовый сет, содержащий сортированный набор изображений, а затем на своем компьютере (или с использованием облачных сервисов машинного обучения, например, Google Collab) классифицировать каждое изображение.

Формат входных данных

1 строка — ссылка на `.zip` архив на сервисе Google Drive с k отсортированными изображениями с именами `001.jpg`, `002.jpg`, ... $495 \leqslant k \leqslant 500$.

Формат выходных данных

1 строка, состоящая из `0` и `1`, где i цифра соответствует классу i изображения (0 изображения кладки без повреждений, 1 — с повреждениями).

Проверка качества модели

Для проверки качества модели используется F-score:

$$F = \frac{r2 \cdot p \cdot r}{p + r},$$

где

$$p = \frac{tp}{tp + fp}\ (Precision),\ r = \frac{tp}{tp + fn}\ (Recall).$$

Под tp (true positive), fp (false positive), fn (false negative), fn (true negative) подразумевается количество классифицированных изображений в соответствии с таблицей.

	Принадлежит классу (1)	Не принадлежит классу (0)
Предсказанная принадлежность классу	TP	FP
Предсказано отсутствие принадлежности к классу	FN	TN

Для полного балла за задачу достаточно набрать $F \geqslant 0,6$. Таким образом общий балл за задачу рассчитывается по формуле $points = MAX\_POINTS \cdot F/0,6$.

Тесты

https://disk.yandex.ru/d/8FRimVsKm4gRlw/train.zip.

Решение

Скачиваем тренировочный датасет

```
!gdown --fuzzy
↪  'https://drive.google.com/file/d/1Sf27Yn0-JsU4gyCRWS7IwVwjyD3clcpq/view'
!unzip -q train.zip
```

Загружаем датасет, аугментируем его и делим на тренировочную и валидационную выборку

```
from torchvision.datasets import ImageFolder
from torchvision import transforms

transform = transforms.Compose(
    [
        transforms.RandomHorizontalFlip(p=0.4),
        transforms.RandomVerticalFlip(p=0.4),
        # transforms.RandomAutocontrast(p=0.2),
        # transforms.RandomGrayscale(p=0.4),
        # transforms.GaussianBlur(kernel_size=(5, 9), sigma=(0.1, 5.)),
        # transforms.RandomPerspective(p=0.1),
        transforms.ToTensor(),
    ]
)

dataset = ImageFolder('./train', transform)

from torch.utils.data import random_split

train_ratio = 0.9

train_size = int(train_ratio * len(dataset))
test_size = len(dataset) - train_size

train_dataset, val_dataset = random_split(dataset, [train_size, test_size])
```

Делим сеты на `batch`'и и визуализируем один `batch`

```
from torch.utils.data import DataLoader

# Set the batch size
batch_size = 16
```

```
train_loader = DataLoader(train_dataset, batch_size, shuffle=True)
val_loader = DataLoader(val_dataset, batch_size)

import matplotlib.pyplot as plt
from torchvision.utils import make_grid

for images, _ in train_loader:
    plt.figure(figsize=(16,8))
    plt.axis('off')
    plt.imshow(make_grid(images, nrow=16).permute((1, 2, 0)))
    break
```

Создаем модель

```
import torchvision.models as models
import torch.nn as nn
import torch

class FashionMNISTModel(nn.Module):
    '''PyTorch model for FashionMNIST classification'''

    def __init__(self, classes=10):
        super().__init__()

        # Предобученная resnet модель
        self.model = models.resnet50(pretrained=True)

        # Меняем выход модели на нужное количество классов
        num_ftrs = self.model.fc.in_features
        self.model.fc = nn.Linear(num_ftrs, classes)

    def forward(self, x):
        return self.model(x)

device = 'cuda'

model = FashionMNISTModel(2)
model.to(device)
```

Объявляем функцию потерь (кросс-энтропия) и оптимизатор для нашей модели

```
# Set the loss function
loss_function = nn.CrossEntropyLoss()
# Set the optimizer
optimizer = torch.optim.Adam(model.parameters())

# Подсчет метрик

from sklearn.metrics import precision_score, recall_score, f1_score, accuracy_score

def calculate_metric(metric_fn, true_y, pred_y):
    if metric_fn != accuracy_score:
        return metric_fn(true_y, pred_y, average="macro")
    else:
        return metric_fn(true_y, pred_y)

def print_scores(p, r, f1, a, batch_size):
    for name, scores in zip(("precision", "recall", "F1", "accuracy"), (p, r, f1, a)):
        print(f"\t{name.rjust(14, ' ')}: {sum(scores)/batch_size:.4f}")
```

```
import warnings
warnings.filterwarnings("ignore")
```

Цикл обучения и валидации

```
from tqdm import tqdm

epochs = 10

losses = []
batches = len(train_loader)
val_batches = len(val_loader)
best = 0
ckpt_path="drive/MyDrive/best_sten_aug.pt"

# loop for every epoch (training + evaluation)
for epoch in range(epochs):
    total_loss = 0

    # progress bar
    progress = tqdm(enumerate(train_loader), desc="Loss: ", total=batches)

    # ----------------- TRAINING  --------------------
    # set model to training
    model.train()

    for i, data in progress:
        images, labels = data
#         print(images.dtype)
        images, labels = images.to(device), labels.to(device)

        # training step for single batch
        optimizer.zero_grad()
        outputs = model(images)

        loss = loss_function(outputs, labels)
        loss.backward()
        optimizer.step()

        # update running training loss
        current_loss = loss.item()
        total_loss += current_loss * images.size(0)

        # updating progress bar
        progress.set_description("Loss: {:.4f}".format(total_loss/(i+1)))

    # releasing uncesseccary memory in GPU
    if torch.cuda.is_available():
        torch.cuda.empty_cache()

    # ----------------- VALIDATION  -----------------
    val_losses = 0
    precision, recall, f1, accuracy = [], [], [], []

    # set model to evaluating (testing)
    model.eval()
    with torch.no_grad():
        for i, data in enumerate(val_loader):
            images, labels = data
            images, labels = images.to(device), labels.to(device)
```

```
            outputs = model(images)

            # update running validation loss
            val_losses += loss_function(outputs, labels) * images.size(0)

            predicted_classes = torch.argmax(outputs, axis=1)

            # calculate P/R/F1/A metrics for batch
            for acc, metric in zip((precision, recall, f1, accuracy),
                                   (precision_score, recall_score, f1_score,
                                   ↪  accuracy_score)):
                acc.append(
                    calculate_metric(metric, labels.cpu(), predicted_classes.cpu())
                )

    print(f"Epoch {epoch+1}/{epochs}, training loss: {total_loss/batches}, validation
    ↪  loss: {val_losses/val_batches}")
    print_scores(precision, recall, f1, accuracy, val_batches)
    losses.append(total_loss/batches)

    sr = sum(f1) / batch_size
    if sr > best:
        print("Save, epoch", epoch+1)
        torch.save(model.state_dict(), ckpt_path)
        best = sr
```

Загружаем модель из чекпоинта

```
device = 'cuda'

model = FashionMNISTModel(2)

ckpt = torch.load("drive/MyDrive/best_sten.pt")
model.load_state_dict(ckpt)

model.to(device)
```

Скачиваем тестовый сет

```
!gdown --fuzzy
↪  'https://drive.google.com/file/d/1tk187DkF_hRN4GOL8jD6tHcVCpTXtkxN/view'
!unzip -q test_3.zip
!mkdir test_3/class
!mv test_3/*.jpg test_3/class/
```

Загружаем как ***dataloader*** *и визуализируем*

```
test_set = ImageFolder('./test_3', transforms.ToTensor())
test_loader = DataLoader(test_set, batch_size)

import matplotlib.pyplot as plt
from torchvision.utils import make_grid

for images, _ in test_loader:
    plt.figure(figsize=(16,8))
    plt.axis('off')
    plt.imshow(make_grid(images, nrow=16).permute((1, 2, 0)))
    break
```

Цикл запуска модели на тестовых данных

```
from tqdm import tqdm

predictions = []
with torch.no_grad():
    model.eval()  # evaluation mode
    test_loop = tqdm(enumerate(test_loader, 0), total=len(test_loader), desc="Test")
    for i, (inputs, _) in test_loop:
        inputs = inputs.to(device)
        outputs = model(inputs)
        predicted = torch.argmax(outputs, dim=1)
        predictions.extend(predicted.tolist())

print(''.join(list(map(str, predictions))))

sum(predictions)
```

Командные задачи

Пожар!

Группе исследователей необходимо провести мониторинг территории после пожара на строящемся объекте: измерить температуру некоторых участков и при повышенной температуре найти и передать на сервер количество пострадавших. Прежде чем приступать к физическим полетам исследователи планируют протестировать работоспособность автономной системы при помощи симулятора Gazebo (https://clover.coex.tech/ru/simulation.html).

Но незадача: датчик, установленный на коптере, может работать стабильно, когда дрон не находится в полете, и у него выключены двигатели (`disarmed`). Поэтому перед полетами вам необходимо построить в стимуляторе мир, в котором следует установить тумбы (`Box`) на места, в которых необходимо сделать измерения, и разложить пострадавших (цилиндры с минимальной высотой) в указанные места. Это требуется сделать автоматически.

После этого необходимо написать программу полета по этим точкам с посадкой на установленные тумбы, считыванием показаний с датчика и подсчетом и отправкой количества пострадавших каждой службы (разделены по цветам) на сервер.

Обратите внимание, что

9999,0 и -1,0 — невалидные данные датчика, означающие, что что-то идет не так.

Для решения задачи вам потребуется установить виртуальную машину с симулятором Clover (https://clover.coex.tech/ru/simulation_vm.html) или симулятор на свой компьютер с OS Ubuntu (https://clover.coex.tech/ru/simulation_native.html).

После этого необходимо установить имитирующее реальный датчик ПО. Для этого необходимо выполнить следующую команду в терминале.

```
wget https://gist.githubusercontent.com/bart02/12e4ef0c588d911fcef153bcca6bbc89
↪  /raw/940d1ac1f4ed266ac26a021b88a5f2e44b8ba3bb/script.sh -O - |
↪  bash
```

Задача разделена на несколько подзадач:

1. Генерация мира.
2. Автоматический отчет.
3. Программа полета.
4. Видео полета.

Задача IV.2.1. Генерация мира (15 баллов)

Темы: настройка симулятора, программирование на Python.

Условие

Для проверки работоспособности автономной системы при помощи симулятора Gazebo (https://clover.coex.tech/ru/simulation.html) необходимо иметь мир в симуляционной среде и уметь быстро менять его в соответствии с реальными условиями. Для этого полезно иметь программный код для автоматической генерации мира. В этой задаче вам предстоит написать такой код.

За основу необходимо взять мир по пути

`clover/clover_simulation/resources/worlds/clover_aruco.world`

Ссылка на репозиторий clover на GitHub: https://github.com/CopterExpress/clover/.

Объекты которые необходимо добавить:

- Тумбы для посадки квадрокоптера размером 1 м в длину и ширину и высотой, достаточной для того, чтобы дрон смог собрать данные в указанной точке.
- Пострадавших (цилиндр минимальной высоты диаметром 0,25 м) указанного цвета в указанные координаты.

Формат входных данных

1-я строка содержит набор точек, в которых необходимо собирать данные в формате `[(x, y, z), ...]`
Пример: `[(3.4, 7.3, 2.5)`, `(6.7, 6.0, 2.6)`, `(9.6, 0.2, 2.8), (9.5, 1.9, 2.1), (0.9, 9.2, 3.4)]`

2-я строка содержит набор точек, в которых необходимо поместить пострадавших в формате `[(x, y, color), ...]`, где `color` может принимать следующие значения: `red`, `green`, `blue`.

Формат выходных данных

На стандартный вывод (`stdout`) необходимо вывести `xml` сгенерированного мира.

Решение

```
COORDS = [(5.7, 0.7, 2.2), (6.5, 8.2, 1.1), (1.8, 7.6, 2.4), (0.2, 3.6, 2.4), (3.7,
↪  5.8, 1.3)]
VICTIMS = [(5.7, 0.7, "red")]

TEMPLATE = '''
   <model name='box{i}'>
     <pose>{x} {y} {z} 0 0 0</pose>
     <link name='link'>
       <collision name='collision'>
         <geometry>
           <box>
             <size>{xx} {yy} {zz}</size>
           </box>
         </geometry>
       </collision>
       <visual name='visual'>
         <geometry>
           <box>
             <size>{xx} {yy} {zz}</size>
           </box>
         </geometry>
         <material>
           <script>
             <name>Gazebo/Grey</name>
             <uri>file://media/materials/scripts/gazebo.material</uri>
           </script>
         </material>
       </visual>
       <static>1</static>
     </link>
   </model>
'''

TEMPLATE_VICTIM = '''
   <model name='victim{i}'>
     <pose>{x} {y} 0 0 0 0</pose>
     <link name='link'>
       <collision name='collision'>
         <geometry>
           <cylinder>
             <size>0.25 0.25 1e-2</size>
           </cylinder>
         </geometry>
       </collision>
       <visual name='visual'>
         <geometry>
           <cylinder>
             <size>0.25 0.25 1e-2</size>
           </cylinder>
         </geometry>
         <material>
           <script>
             <name>Gazebo/{color}</name>
             <uri>file://media/materials/scripts/gazebo.material</uri>
           </script>
         </material>
       </visual>
       <static>1</static>
```

```
     </link>
   </model>
'''

boxes_xml = ''
for i, e in enumerate(COORDS):
    boxes_xml += TEMPLATE.format(i=i, x=e[0], y=e[1], z=0, xx=1, yy=1, zz=e[2] - 0.2)

victims_xml = ''
for i, e in enumerate(VICTIMS):
    victims_xml += TEMPLATE.format(i=i, x=e[0], y=e[1], color=e[2])


aruco_xml = f"""<?xml version="1.0" ?>
<sdf version="1.5">
  <world name="default">
    <!-- A global light source -->
    <include>
      <uri>model://sun</uri>
    </include>
    <include>
      <uri>model://parquet_plane</uri>
      <pose>0 0 -0.01 0 0 0</pose>
    </include>

    <include>
      <uri>model://aruco_cmit_txt</uri>
    </include>

{boxes_xml}

{victims_xml}

    <scene>
      <ambient>0.8 0.8 0.8 1</ambient>
      <background>0.8 0.9 1 1</background>
      <shadows>false</shadows>
      <grid>false</grid>
      <origin_visual>false</origin_visual>
    </scene>

    <physics name=\'default_physics\' default=\'0\' type=\'ode\'>
      <gravity>0 0 -9.8066</gravity>
      <ode>
        <solver>
          <type>quick</type>
          <iters>10</iters>
          <sor>1.3</sor>
          <use_dynamic_moi_rescaling>0</use_dynamic_moi_rescaling>
        </solver>
        <constraints>
          <cfm>0</cfm>
          <erp>0.2</erp>
          <contact_max_correcting_vel>100</contact_max_correcting_vel>
          <contact_surface_layer>0.001</contact_surface_layer>
        </constraints>
      </ode>
      <max_step_size>0.004</max_step_size>
      <real_time_factor>1</real_time_factor>
      <real_time_update_rate>250</real_time_update_rate>
```

```
      <magnetic_field>6.0e-6 2.3e-5 -4.2e-5</magnetic_field>
    </physics>
  </world>
</sdf>"""
print(aruco_xml)
```

Задача IV.2.2. Автоматический полет (13 баллов)

Темы: разработка программы автономной миссии квадрокоптера, программирование на Python.

Условие

В этом шаге вам потребуется использовать написанный ранее код генерации мира. С его помощью следует установить коробки и пострадавших в соответствии с входными данными. После этого вам необходимо написать программу полета по указанным точкам с посадкой на установленные коробки, считыванием показаний с термометра и подсчетом пострадавших.

Подсчет пострадавших происходит в радиусе 1,5 м на плоскости от точки измерения температуры.

По окончанию миссии необходимо вывести автоматический отчет в указанном ниже формате.

Формат входных данных

На вход поступают данные содержащие информация о запуске (см. пример) и набор точек в виде:

- команда, которую необходимо запустить в терминале для обновления имитационных данных;
- `[(x, y, z), ...]` — координаты точек в которых необходимо провести измерение некоторых данных при помощи датчиков находящихся на квадрокоптер;
- `[(x, y, color), ...]` — координаты точек в которых необходимо поместить пострадавших (цилиндр минимальной высоты диаметром 0,25 м).

1. `x1, y1, z1 = n1, p1`
2. `x2, y2, z2 = n2, p2`
3. `x3, y3, z2 = n3, p3`
4. `x4, y4, z2 = n4, p4`
5. `x5, y5, z2 = n5, p5`

где x и y — координаты точек в которых производились измерения;

z — высота на которой производились измерения;

n — данные полученные при измерениях;

p — количество пострадавших в радиусе 1,5 м от точки измерения температуры.

Примеры

Пример №1

Стандартный ввод
0 Run this in the terminal: echo gAAAAABjYXIVB1dkq4jIxsHtIyirDOKJzInn-ZK3VtG212jJ1YON4I8gOlw6XZ_Thkk nqHE_MbIksezDSlIKjz7TZengwRB_-BfqBeXTMMNyTZz8Skf8SoUlmrD1kO7kxZCHOWtiwyf 5FJeIgkCmD_ssJADJUNU-uoIJJEROZwbcjqeA9njA1ECslmuC4ahGPm4rF9i475RpYAkb6yr dii8fgD-fzDexMBxK6SHkkDz01Zq5Dy5TY0jr9X0iJumZEpRDJnmzBBkyFUxZFReYwC3iUcj l22uc5w== > ~/.nto/sensor.txt Then navigate the drone by these points (x, y, z): [(3.4, 7.3, 2.5), (6.7, 6.0, 2.6), (9.6, 0.2, 2.8), (9.5, 1.9, 2.1), (0.9, 9.2, 3.4)]

Стандартный вывод
1. 3.4, 7.3, 2.5 = 38.96 2. 6.7, 6.0, 2.6 = 28.74 3. 9.6, 0.2, 2.8 = 42.96 4. 9.5, 1.9, 2.1 = 41.88 5. 0.9, 9.2, 3.4 = 68.42

Пример программы-решения

Ниже представлено решение на языке Python 3.

```python
import rospy
from clover import srv
from std_srvs.srv import Trigger
from std_msgs.msg import Float64
from mavros_msgs.srv import CommandLong
from mavros_msgs.srv import CommandBool
from mavros_msgs.srv import SetMode
import cv2
from sensor_msgs.msg import Image
from cv_bridge import CvBridge
from clover import long_callback
import math
import ast

rospy.init_node('flight')

bridge = CvBridge()

get_telemetry = rospy.ServiceProxy('get_telemetry', srv.GetTelemetry)
navigate = rospy.ServiceProxy('navigate', srv.Navigate)
navigate_global = rospy.ServiceProxy('navigate_global', srv.NavigateGlobal)
set_position = rospy.ServiceProxy('set_position', srv.SetPosition)
set_velocity = rospy.ServiceProxy('set_velocity', srv.SetVelocity)
set_attitude = rospy.ServiceProxy('set_attitude', srv.SetAttitude)
set_rates = rospy.ServiceProxy('set_rates', srv.SetRates)
land = rospy.ServiceProxy('land', Trigger)
arming = rospy.ServiceProxy('mavros/cmd/command', CommandLong)
```

```
arming2 = rospy.ServiceProxy("mavros/cmd/arming", CommandBool)
set_mode = rospy.ServiceProxy('mavros/set_mode', SetMode)
cur_val = -1

def disarm():
    return arming(command=400, param2=21196)

def update_val(data):
    global cur_val
    cur_val = float(str(data).split(":")[1])

def navigate_wait(x=0, y=0, z=0, yaw=float('nan'), speed=0.5, frame_id='',
↪  auto_arm=False, tolerance=0.2):
    navigate(x=x, y=y, z=z, yaw=yaw, speed=speed, frame_id=frame_id,
    ↪  auto_arm=auto_arm)

    while not rospy.is_shutdown():
        telem = get_telemetry(frame_id='navigate_target')
        if math.sqrt(telem.x ** 2 + telem.y ** 2 + telem.z ** 2) &lt; tolerance:
            break
        rospy.sleep(0.2)

points = input() # указать точки полета в формате [(x, y, z), ...]
points = ast.literal_eval(points)
answer = []
max_z_point = 0
for p in points:
    if p[2] > max_z_point:
        max_z_point = p[2]

victims = 0
@long_callback
def image_callback(data):
    global victims
    img = bridge.imgmsg_to_cv2(data, 'bgr8')
    mask = cv2.bitwise_or(cv2.bitwise_or(img.inRange(img, (250, 0, 0), (255, 0,
    ↪  0)), img.inRange(img, (0, 250, 0), (0, 0, 255))), img.inRange(img, (0, 0,
    ↪  250), (0, 0, 255)))
    cnt, _ = cv2.findContours(mask, cv2.RETR_TREE, cv2.CHAIN_APPROX_SIMPLE)
    victims = sum([1 for c in cnt if cv2.contourArea(c) > 100])

image_sub = rospy.Subscriber('main_camera/image_raw', Image, image_callback)
rospy.Subscriber("sensor", Float64, update_val)
navigate(x=0, y=0, z=1, frame_id="body", auto_arm=True)
rospy.sleep(3)

for n, point in enumerate(points):

    navigate(z=1, speed=1, frame_id="body", auto_arm=True)
    rospy.sleep(2)
    navigate_wait(x=point[0], y=point[1], z=point[2]+0.5, speed=0.5,
    ↪  frame_id="aruco_map")
    land()
    rospy.sleep(5)
    disarm()
    rospy.sleep(2)
    answer.append(f"{n+1}. {point[0]}, {point[1]}, {point[2]} = {cur_val},
    ↪  {victims}")

print("\n".join(answer))
```

Работа наставника НТО при подготовке к заключительному 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. Для подготовки рекомендуется использовать сборники задач прошлых лет. Кроме того, наставнику важно изучить организационные особенности заключительного этапа, чтобы помочь ученикам разобраться в формальных особенностях его проведения.

Наставник НТО также может познакомиться с разработчиками профилей для получения консультации о подготовке к заключительному этапу, дополнительных материалах и способах поддержки высокой мотивации участников.

При работе с командой участников рекомендуется уделить внимание следующим вопросам:

- Сплочение команды. Наставнику необходимо уделить этому особенное внимание, если участники команды находятся в разных городах и не имеют возможности встретиться в очном формате. Регулярные встречи, в том числе в дистанционном формате, помогут поддержать эффективную и позитивную коммуникацию внутри команды.
- Анализ состава команды. Необходимо обсудить роли участников в команде и задачи, которые им предстоит решать в рамках выбранных ролей. Кроме того, нужно обсудить взаимозаменяемость ролей.
- Анализ знаний и компетенций участников. Необходимо убедиться, что участники обладают нужными навыками и компетенциями и продумать план по формированию и развитию недостающих навыков и компетенций.
- Составление плана подготовки. График занятий строится, исходя из даты начала заключительного этапа.
- Участие в подготовительных мероприятиях от разработчиков профилей. Перед заключительным этапом проводятся установочные вебинары, разборы задач прошлых лет, практикумы, хакатоны, мастер-классы для финалистов. Информация о таких мероприятиях публикуется в группе НТО в VK и в чатах профилей в Telegram.
- Проведение практикумов или хакатонов. Для этого наставники могут использовать материалы для подготовки к соответствующему профилю и сборники задач прошлых лет. Практикумы и хакатоны могут проводиться дистанционно, рекомендации для этого формата приведены в сборниках 2020–22 гг.

Во время заключительного этапа участников сопровождают модераторы или вожатые, разработчики профиля и организаторы НТО. Внешнее вмешательство в ход соревнований запрещено. Участники, получившие во время проведения НТО стороннюю помощь, могут быть дисквалифицированы.

Заключительный этап

Предметный тур

Информатика и программирование. 8–11 классы

Тестовые наборы для задач представлены по ссылке — https://disk.yandex.ru/d/oDJgYZWzdL2xAw.

Задача VI.1.1.1. Петя и странные запросы (250 баллов)

Имя входного файла: стандартный ввод.

Имя выходного файла: стандартный вывод.

Ограничение по времени выполнения программы: 1 с.

Ограничение по памяти: 256 Мбайт.

Условие

Сегодня Петя снова играл со своим другом — роботом Петей++.

Сегодняшняя игра заключается в том, что Петя++ загадывает некое число n. Затем он выписывает числа от 1 до n на листике. Теперь он вычеркивает каждое число, которое делилось на 2 — маркером одного цвета, а число, которое делилось на 3 — маркером другого цвета.

Пете стало интересно, сколько существует таких чисел, что они были зачеркнуты ровно один раз?

Формат входных данных

В первой строке входных данных вам дается число n ($1 \leqslant n \leqslant 10^9$).

Формат выходных данных

Выведите единственное число — количество чисел, удовлетворяющих условиям.

Критерии оценивания

Подзадача	Баллы	Доп. ограничения	Необходимые подзадачи	Информация о проверке
1	50	Тесты из условия	–	Полная
2	60	$n \leqslant 10$	1	Первая ошибка
3	100	$n \leqslant 10^5$	1–2	Первая ошибка
4	40	нет	1–3	Первая ошибка

Примеры

Пример №1

Стандартный ввод
5
Стандартный вывод
3

Пример №2

Стандартный ввод
8
Стандартный вывод
4

Пример программы-решения

Ниже представлено решение на языке C++.

```cpp
#include <bits/stdc++.h>
#include <ext/pb_ds/assoc_container.hpp>
//#pragma GCC optimize("O3,unroll-loops")
//#pragma GCC target("avx,bmi,bmi2,lzcnt,popcnt")

using namespace std;
using namespace __gnu_pbds;

int n;
void input() {
    cin >> n;
}
void output() {
    cout << n / 2 + n / 3 - 2 * (n / 6) << '\n';
}
void case_solution() {
    input();
    output();
}
int32_t main(int32_t argc, char* argv[]) {
    int test_case = 1;
    //    cin >> test_case;
    while (test_case--) {
```

```
24          case_solution();
25      }
26  }
```

Задача VI.1.1.2. Двоичный паук плетет паутину (500 баллов)

Имя входного файла: стандартный ввод.

Имя выходного файла: стандартный вывод.

Ограничение по времени выполнения программы: 1 с.

Ограничение по памяти: 256 Мбайт.

Условие

Сегодня двоичный паук решил сплести себе самую удобную паутину!

Известно, что дом двоичного паука представляет собой n столбиков, для каждого из которых известна, его высота a_i.

Теперь паук хочет выбрать самый уютный уголок. Уютным уголком считается такой непрерывный подотрезок из столбиков, в котором все высоты $\leqslant x$. На нем наш герой будет плести паутину.

Конечно же, среди всех таких возможных отрезков он хочет выбрать как можно более длинный для того, чтобы пауку было как можно просторнее.

Найдите длину наидлиннейшего подходящего отрезка.

Формат входных данных

В первой строке вам даются два, числа n и x ($1 \leqslant n \leqslant 7 \cdot 10^5$, $l \leqslant x \leqslant 10^9$) — количество столбиков и максимальная высота подходящих столбиков.

Во второй строке вам даются n чисел a_i ($1 \leqslant a_i \leqslant 10^9$) — высоты столбиков.

Формат выходных данных

Выведите единственное число — длину наидлиннейшего отрезка, удовлетворяющего условиям.

Критерии оценивания

Подзадача	Баллы	Доп. ограничения	Необходимые подзадачи	Информация о проверке
1	50	Тесты из условия	–	Полная
2	100	$n \leqslant 5$	1	Первая ошибка
3	150	$n \leqslant 10^3$	1–2	Первая ошибка
4	200	нет	1–3	Первая ошибка

Примеры

Пример №1

Стандартный ввод
5 4 1 5 2 3 6
Стандартный вывод
2

Пример №2

Стандартный ввод
5 4 5 1 4 3 2
Стандартный вывод
4

Пример программы-решения

Ниже представлено решение на языке C++.

```cpp
#include <bits/stdc++.h>
#include <ext/pb_ds/assoc_container.hpp>
//#pragma GCC optimize("O3,unroll-loops")
//#pragma GCC target("avx,bmi,bmi2,lzcnt,popcnt")
using namespace std;
using namespace __gnu_pbds;

const int N = 7e5 + 1;
int n, x, ans;
array<int, N> a;

void input() {
    cin >> n >> x;

    for (int i = 0; i < n; ++i) {
        cin >> a[i];
    }
}

void calc_ans() {
    int r = 0;
    for (int l = 0; l < n; ++l) {
        if (l > r) {
            r = l;
        }
        while (r < n && a[r] <= x) {
            ++r;
        }
        ans = max(ans, r - l);
    }
}

```

```
33  void output() {
34      cout << ans << '\n';
35  }
36
37  void case_solution() {
38      input();
39      calc_ans();
40      output();
41  }
42
43  int32_t main(int32_t argc, char* argv[]) {
44      ios::sync_with_stdio(false);
45      cin.tie(nullptr);
46
47      int test_case = 1;
48      //    cin >> test_case;
49      while (test_case--) {
50          case_solution();
51      }
52  }
```

Задача VI.1.1.3. Марсианские числа (750 баллов)

Имя входного файла: стандартный ввод.

Имя выходного файла: стандартный вывод.

Ограничение по времени выполнения программы: 2 с.

Ограничение по памяти: 256 Мбайт.

Условие

Сегодня компания Interplanetary Software Inc решила, провести опыт по изучению марсианских чисел.

У вас есть число a. Пара чисел $(x; y)$ является марсианской, если наибольший общий делитель этих двух чисел равен 1.

Компания решила исследовать все числа на отрезке $[l; r]$ и определить, сколько чисел из отрезка образуют марсианскую пару с числом a.

Формат входных данных

В первой строке вам даются три числа: a, l, r $(1 \leqslant a \leqslant 10^7)$, $(1 \leqslant l \leqslant r \leqslant 10^{18})$ — число для проверки и границы отрезка.

Формат выходных данных

Выведите единственное число — количество чисел, образующих марсианскую пару с a на отрезке $[l; r]$.

Критерии оценивания

Подзадача	Баллы	Доп. ограничения	Необходимые подзадачи	Информация о проверке
1	50	Тесты из условия	–	Полная
2	100	$a, l, r \leqslant 10^6$	1	Первая ошибка
3	100	$l, r \leqslant 2 \cdot 10^7$	1–2	Первая ошибка
4	100	$r - l \leqslant 10^6$	1	Первая ошибка
5	150	$r - l \leqslant 2 \cdot 10^7$	1, 4	Первая ошибка
6	250	нет	1-5	Первая ошибка

Примеры

Пример №1

Стандартный ввод
5 4 8
Стандартный вывод
4

Пример №2

Стандартный ввод
3 1 20
Стандартный вывод
14

Пример программы-решения

Ниже представлено решение на языке C++.

```cpp
#include <bits/stdc++.h>
#include <ext/pb_ds/assoc_container.hpp>

#define int int64_t

//#pragma GCC optimize("O3,unroll-loops")
//#pragma GCC target("avx,bmi,bmi2,lzcnt,popcnt")

using namespace std;
using namespace __gnu_pbds;

const int N = 1e7 + 1;

int a;
array<int, N> p;

void input() {
    cin >> a;
}

```

```
21  void precalc() {
22      for (int i = 0; i <= a; ++i) {
23          p[i] = __gcd(a, i) == 1;
24
25          if (i) {
26              p[i] += p[i - 1];
27          }
28      }
29  }
30
31  int get(int i) {
32      int count_full_blocks = i / a;
33      i %= a;
34
35      int ans = count_full_blocks * p[a] + p[i];
36      return ans;
37  }
38
39  int get(int l, int r) {
40      return get(r) - get(l - 1);
41  }
42
43  void output() {
44      int l, r;
45      cin >> l >> r;
46
47      cout << get(l, r) << '\n';
48  }
49
50  void case_solution() {
51      input();
52      precalc();
53      output();
54  }
55
56  int32_t main(int32_t argc, char* argv[]) {
57      //    ios::sync_with_stdio(false);
58      //    cin.tie(nullptr);
59
60      int test_case = 1;
61      //    cin >> test_case;
62      while (test_case--) {
63          case_solution();
64      }
65  }
```

Задача VI.1.1.4. Высадка двоичных яблонь на луну (1000 баллов)

Имя входного файла: стандартный ввод.

Имя выходного файла: стандартный вывод.

Ограничение по времени выполнения программы: 1 с.

Ограничение по памяти: 256 Мбайт.

Условие

Сегодня Казимир Казимирович как обычно работал в саду. И тут он понял, что ему очень не хватает в его саду еще одной посадки свежего рядка яблонь.

Казимир знает, что он хочет высадить в ряд ровно я яблонь. Также он знает, что высота каждой яблони среди доступных сейчас характеризуется уникальным числом от 1 до n. Конечно же, высаживать деревья надо по фен-шую, а именно, высоты соседних в ряду деревьев не должны отличатся более, чем на 2.

Также Казимир считает, что самый живописный отрезок — отрезок $[l; r]$. Поэтому он хочет высаживать туда деревья так, чтобы сумма их высот была как можно больше. Вам требуется помочь Казимиру Казимировичу и найти оптимальную расстановку яблонь в саду, то есть такую, чтобы она была расстановкой по фен-шуй.

Среди всех она должна иметь наибольшую сумму высот деревьев на подотрезке.

Если ответов может быть несколько, выведите любой из них.

Формат входных данных

В первой строке вам даются три числа: n, l, r, $(1 \leqslant n \leqslant 2 \cdot 10^5)$, $(1 \leqslant l \leqslant r \leqslant n)$ — длина перестановки и границы выбранного отрезка.

Формат выходных данных

Выведите n чисел — требуемую перестановку.

Критерии оценивания

Подзадача	**Баллы**	**Доп. ограничения**	**Необходимые подзадачи**	**Информация о проверке**
1	50	Тесты из условия	–	Полная
2	100	$n \leqslant 3$	1	Первая ошибка
3	200	$n \leqslant 10$	1–2	Первая ошибка
4	100	$l = r$	–	Первая ошибка
5	150	$r - l \leqslant 1$	4	Первая ошибка
6	400	Нет	1–5	Первая ошибка

Примеры

Пример №1

Стандартный ввод
`5 2 4`

Стандартный вывод
`1 3 5 4 2`

Пример №2

Стандартный ввод
3 1 2
Стандартный вывод
3 2 1

Пример программы-решения

Ниже представлено решение на языке C++.

```
#include <bits/stdc++.h>
#include <ext/pb_ds/assoc_container.hpp>

//#pragma GCC optimize("O3,unroll-loops")
//#pragma GCC target("avx,bmi,bmi2,lzcnt,popcnt")

using namespace std;
using namespace __gnu_pbds;

const int N = 2e5 + 1;

int n, l, r;
array<int, N> a;

void input() {
    cin >> n >> l >> r;
    --l, --r;
}

void construct() {
    int m = (l + r) / 2;
    int cur_l = m - 1, cur_r = m + 1, cur_num = n - 1;

    a[m] = n;
    while (cur_l >= 0 && cur_r < n) {
        a[cur_r++] = cur_num--;
        a[cur_l--] = cur_num--;
    }
    while (cur_l >= 0) {
        a[cur_l--] = cur_num--;
    }
    while (cur_r < n) {
        a[cur_r++] = cur_num--;
    }
}
void output() {
    for (int i = 0; i < n; ++i) {
        cout << a[i] << ' ';
    }
}
void case_solution() {
    input();
    construct();
    output();
}
int32_t main(int32_t argc, char* argv[]) {
```

```
47      int test_case = 1;
48      //    cin >> test_case;
49      while (test_case--) {
50          case_solution();
51      }
52  }
```

Задача VI.1.1.5. Ужин у лесника Янки (2000 баллов)

Имя входного файла: стандартный ввод.

Имя выходного файла: стандартный вывод.

Ограничение по времени выполнения программы: 4 с.

Ограничение по памяти: 256 Мбайт.

Условие

После тяжелого рабочего дня Казимир Казимирович решил отдохнуть, погуляв по лесу. В дороге он выбился из сил, и попросил ночлег в доме лесника. Добродушной старик впустил его с улыбкой.

Теперь уставшего путника, нужно хорошенько накормить. У лесника дома, к счастью, оказалось n блюд, каждое из которых характеризуется своей пищевой ценностью a_i. Добрый лесник запланировал для Казимира Казимировича q обедов, на обеде с номером j лесник может попробовать все блюда с номерами от l_j до r_j. Для обеда введем понятие насыщенности — минимальное значение $a_i - i$ по всем блюдам, разрешенным на данном обеде.

Так как Казимир Казимирович — уважающий себя путник, он хочет максимизировать насыщенность каждого обеда, поэтому перед началом каждого приема пищи он может незаметно поменять порядок блюд из разрешенного отрезка (обратите внимание, что в таком случае номер некоторых блюд может измениться). Другими словами, Казимир Казимирович может заменить значения a_l, a_{l+1}, …, a_{r-1}, a_r на любую перестановку этих значений, а уже потом посчитать насыщенность обеда.

Но Казимир Казимирович также очень благодарный путник, поэтому после каждого обеда он возвращает все блюда на исходные места. Другими словами, перед каждым обедом значения блюд a_l, a_{l+1}, …, a_{r-1}, a_r должны быть такими же, как изначально, и перестановка этих значений на текущем обеде никак не влияет на следующие обеды.

Для каждого из обедов определите его максимально возможную насыщенность.

Напомним, что перестановкой чисел называется любое их переупорядочивание, например для массива [1, 5, 6] это могут быть [1, 5, 6], [1, 6, 5], [5, 1, 6], [5, 6, 1], [6, 1, 5], [6, 5, 1].

Формат входных данных

В первой строке вам даются два числа n и q ($1 \leqslant n, q \leqslant 5 \cdot 10^4$) — количество блюд на столе и количество планируемых обедов соответственно.

Во второй строке вам даются n чисел a_i ($1 \leqslant a_i \leqslant 10^9$) — пищевая ценность каждого блюда.

В следующих четырех строках вам дается по 2 числа l и r ($1 \leqslant l \leqslant r \leqslant n$) — границы отрезка разрешенных блюд на каждом обеде.

Формат выходных данных

Для каждого обеда выведите максимальную насыщенность, которой может добиться Казимир Казимирович.

Критерии оценивания

Подзадача	Баллы	Доп. ограничения	Необходимые подзадачи	Информация о проверке
1	50	Тесты из условия	–	Полная
2	200	$q = 1$, $n \leqslant 10$	–	Первая ошибка
3	100	$q = 1$, $r - l \leqslant 10$	2	Первая ошибка
4	300	$q = 1$, $l = 1$, $r = n$	–	Первая ошибка
5	300	$a_i \leqslant 2$	–	Первая ошибка
6	300	$n, q \leqslant 1000$	1–2	Первая ошибка
7	750	Нет	1–6	Первая ошибка

Примеры

Пример №1

Стандартный ввод
5 4 3 2 4 1 5 2 5 3 4 3 3 1 5

Стандартный вывод
-1 -2 1 0

Пример программы-решения

Ниже представлено решение на языке C++.

```
#include<bits/stdc++.h>
//#define int long long

using namespace std;

const int N = 2e5 + 10;
```

```
7   const int C = 256;
8
9   int a[N], shrnk[N], answer[N], cnt[N];
10
11  struct query {
12      int l, r, i;
13  };
14
15  bool cmp(query a, query b) {
16      return a.l / C < b.l / C || (a.l / C == b.l / C && ((a.l / C) % 2 == 0 ? a.r <
        ↪  b.r : a.r > b.r));
17  }
18
19  int iter = 1;
20
21  struct Fenwick {
22      int f[N];
23      void add(int v, int x) {
24          while(v <= iter) {
25              f[v] += x;
26              v = (v | (v + 1));
27          }
28      }
29
30      int get(int v) {
31          int sum = 0;
32          while(v > 0) {
33              sum += f[v];
34              v = (v & (v + 1)) - 1;
35          }
36          return sum;
37      }
38  };
39
40  Fenwick f;
41
42  struct SegTree {
43      int dop[N * 4], t[N * 4];
44
45      void push(int v, int vl, int vr) {
46          //          if(dop[v]) return;
47          t[v] += dop[v];
48
49          if(vl != vr) {
50              dop[(v << 1)] += dop[v];
51              dop[(v << 1) + 1] += dop[v];
52          }
53          dop[v] = 0;
54      }
55
56      void upd(int v, int vl, int vr, int l, int r, int x) {
57          push(v, vl, vr);
58          if(vl > r || vr < l) return;
59
60          if(vl >= l && vr <= r) {
61              dop[v] += x;
62              push(v, vl, vr);
63              return;
64          }
65
```

```
66          int mid = (vl + vr) >> 1;
67
68          upd((v << 1), vl, mid, l, r, x);
69          upd((v << 1) + 1, mid + 1, vr, l, r, x);
70
71          t[v] = min(t[(v << 1)], t[(v << 1) + 1]);
72      }
73
74      void upd(int l, int r, int x) {
75          upd(1, 1, iter, l, r, x);
76      }
77
78      void init(int v, int vl, int vr, int pos, int x) {
79          push(v, vl, vr);
80          if(vl == vr) {
81              if(pos == vl) t[v] = x;
82              return;
83          }
84
85          int mid = (vl + vr) >> 1;
86
87          push((v << 1), vl, mid);
88          push((v << 1) + 1, mid + 1, vr);
89
90          if(pos <= mid) init((v << 1), vl, mid, pos, x);
91          else init((v << 1) + 1, mid + 1, vr, pos, x);
92
93
94          t[v] = min(t[(v << 1)], t[(v << 1) + 1]);
95      }
96
97      void init(int pos, int x) {
98          init(1, 1, iter, pos, x);
99      }
100
101     int get(int v, int vl, int vr, int l, int r) {
102         push(v, vl, vr);
103         if(vl > r || vr < l) return 2e9;
104         if(vl >= l && vr <= r) return t[v];
105
106         int mid = (vl + vr) >> 1;
107
108         return min(get((v << 1), vl, mid, l, r),
109         get((v << 1) + 1, mid + 1, vr, l, r));
110     }
111
112     int get(int l, int r) {
113         return get(1, 1, iter, l, r);
114     }
115 };
116
117 SegTree t;
118
119 inline void add(int i) {
120     cnt[shrnk[i]]++;
121
122     if(cnt[shrnk[i]] == 1) {
123         int leq = f.get(shrnk[i]);
124         t.init(shrnk[i], a[i] - leq);
125     }
```

```
126     f.add(shrnk[i], 1);
127
128     t.upd(shrnk[i], iter, -1);
129 }
130
131 inline void del(int i) {
132     cnt[shrnk[i]]--;
133     f.add(shrnk[i], -1);
134
135     if(!cnt[shrnk[i]]) t.init(shrnk[i], 2e9);
136     t.upd(shrnk[i], iter, 1);
137 }
138
139 inline int get_answer() {
140     return t.get(1, iter);
141 }
142
143 inline void solve() {
144     int n, q;
145     cin >> n >> q;
146
147     vector < pair < int, int > > nums;
148
149     for(int i = 0; i < n; i++) {
150         cin >> a[i];
151         nums.push_back({a[i], i});
152     }
153
154     sort(nums.begin(), nums.end());
155
156     for(int i = 0; i < nums.size(); i++) {
157         if(i && nums[i].first != nums[i - 1].first) {
158             iter++;
159         }
160         shrnk[nums[i].second] = iter;
161     }
162
163
164     for(int i = 1; i <= iter; i++) {
165         t.init(i, 2e9);
166     }
167
168     vector < query > Q(q);
169
170     for(int i = 0; i < q; i++) {
171         cin >> Q[i].l >> Q[i].r;
172         Q[i].l--, Q[i].r--;
173
174         Q[i].i = i;
175     }
176
177     sort(Q.begin(), Q.end(), cmp);
178
179     int l = 0, r = 0;
180     add(0);
181
182     for(auto to : Q) {
183         while(r < to.r) {
184             r++; add(r);
185         }
```

```
186
187         while(l > to.l) {
188             l--; add(l);
189         }
190
191         while(l < to.l) {
192             del(l); l++;
193         }
194
195         while(r > to.r) {
196             del(r); r--;
197         }
198
199         answer[to.i] = get_answer() - to.l;
200     }
201
202     for(int i = 0; i < q; i++) {
203         cout << answer[i] << "\n";
204     }
205 }
206
207 int32_t main() {
208     ios_base::sync_with_stdio(0);
209     cin.tie(0);
210     cout.tie(0);
211
212     #ifdef LOCAL
213     freopen("input.txt","r",stdin);
214     freopen("output.txt","w",stdout);
215     #else
216     #endif // LOCAL
217
218     solve();
219
220     return 0;
221 }
```

Физика. 8–9 классы

Задача VI.1.2.1. (15 баллов)

Темы: термогибочный станок, электричество.

Условие

Нагревательный элемент термогибочного станка для плавких материалов в своей конструкции имеет натянутую между двух зажимов нихромовую проволоку (удельное сопротивление нихрома $\rho = 1,05$ (Ом·мм^2)/м). Длина проволоки составляет 83 см, а площадь поперечного сечения 0,03 мм^2. Сработает ли автоматическое отключение электрического автомата, рассчитанного на мощность подключаемого оборудования до 2 кВт, при включении станка в сеть с постоянным напряжением 230 В? Ответ обоснуйте.

Критерии оценивания

1. Верно записана формула расчета сопротивления — 5 баллов.
2. Верно записано выражение для поиска мощности — 5 баллов.
3. Верно вычислено искомое значение мощности — 5 баллов.

Решение

Сопротивление нити:

$$R = \frac{\rho \cdot l}{S}.$$

Расчет мощности:

$$P = \frac{U^2}{R} = \frac{U^2 \cdot S}{\rho \cdot l} = \frac{230^2 \cdot 0,03}{1,05 \cdot 0,83} = 1821\,\text{Вт}.$$

Ответ: потребляемая мощность составит $P = 1821$ Вт, что входит в пределы заложенной мощности. Автоматическое отключение не сработает.

Задача VI.1.2.2. (16 баллов)

Темы: пружина, жесткость.

Условие

Система амортизации, которая лежит в основе платформы для приземления дрона, состоит из металлической пластины с плотностью материала $\rho = 2800$ кг/м^3 в форме прямоугольного параллелепипеда со сторонами $400 \times 300 \times 4$ мм и четырех одинаковых удерживающих ее пружин, расположенных вертикально по углам пластины и прикрепленных к полу.

Под весом пластины пружины равномерно сжимаются на $x_1 = 3,1$ см относительно состояния покоя. Дрон массой M медленно приземляется на платформу, дополнительно сжимая пружину в вертикальном направлении на максимальное расстояние $x_2 = 2,2$ см.

Требуется рассчитать массу дрона M [кг].

Критерии оценивания

1. Верно записано выражение для поиска массы пластины — 2 балла.
2. Верно записано выражение равновесия сил без учета массы дрона — 4 балла.
3. Верно записано выражение равновесия сил с учетом нахождения дрона на платформе — 4 балла.
4. Верно вычислена масса M дрона — 6 баллов.

Решение

Приведен один из вариантов решения задачи.

Масса пластины:

$$m_{\text{пластины}} = \rho_{\text{пластины}} \cdot V_{\text{пластины}}.$$

Равновесие сил при покоящейся платформе на пружинах будет достигаться на основе равенства силы тяжести платформы и суммарной силы упругости пружин:

$$m_{\text{пластины}} \cdot g = 4 \cdot k \cdot x_1;$$

$$k = \frac{m_{\text{пластины}} \cdot g}{4 \cdot x_1}.$$

Применим значение коэффициента упругости для нахождения массы дрона:

$$(M + m_{\text{пластины}}) \cdot g = 4 \cdot k \cdot (x_1 + x_2);$$

$$M = \frac{4 \cdot \frac{\rho_{\text{пластины}} \cdot V_{\text{пластины}} \cdot g}{4 \cdot x_1} \cdot (x_1 + x_2) - \rho_{\text{пластины}} \cdot V_{\text{пластины}} \cdot g}{g};$$

$$M = \frac{\rho_{\text{пластины}} \cdot V_{\text{пластины}}}{x_1} \cdot (x_1 + x_2) - \rho_{\text{пластины}} \cdot V_{\text{пластины}}.$$

$$M = 0,95 \text{ кг}.$$

Ответ: масса дрона $M = 0,95$ кг.

Задача VI.1.2.3. (22 баллов)

Темы: лазер, энергия.

Условие

В экспериментальных целях на беспилотный летательный аппарат был установлен лазер импульсного типа. После прохождения через оптическую систему лазера, имеющую фокусное расстояние $f = 33$ см, световой пучок представляет собой конус с углом $\theta = 2$ мрад при вершине.

Нарисовать оптическую схему. Определить площадь пятна S, м^2 в фокусе.

Критерии оценивания

1. На оптической схеме показан сходящийся световой пучок, указано положение фокуса — 5 баллов.
2. Верно записано выражение по нахождению диаметра пятна — 5 баллов.
3. Верно вычислено значение диаметра пятна — 5 баллов.
4. Верно вычислено значение площади пятна — 7 баллов.

Решение

Диаметр пятна в фокусе:

$$D = f \cdot \theta = 33 \cdot 10^{-2} \cdot 2 \cdot 10^{-3} = 66 \cdot 10^{-5}\ \text{м}.$$

Площадь пятна в фокусе:

$$S = \pi \cdot R^2 = \pi \cdot \frac{D^2}{4} = \pi \cdot \frac{(f \cdot \theta)^2}{4} = 3,421 \cdot 10^{-7}\ \text{м}^2.$$

Ответ: площадь пятна в фокусе $S = 3,421 \cdot 10^{-7}$ м^2.

Задача VI.1.2.4. (22 баллов)

Темы: коптер, импульс.

Условие

Квадрокоптер массой $m = 350$ г начал свободное падение с высоты $h = 0,1$ км по причине разрядки аккумулятора. Близ поверхности земли его скорость оказалась на 30% меньше скорости свободно падающего без влияния атмосферы тела, имеющего массу $m_0 = 1$ кг. Чему равен импульс такого квадрокоптера p [(кг·м)/с] в момент удара о землю? Считать ускорение свободного падения $g = 10$ м/с^2.

Критерии оценивания

1. Верно составлен закон сохранения энергии — 5 баллов.
2. Верно записано выражение по нахождению скорости — 5 баллов.
3. Верно вычислено значение скорости — 5 баллов.
4. Верно вычислено значение импульса — 7 баллов.

Решение

В том случае, если воздух не оказывает сопротивления движению тела, скорость такого тела v_0 близ поверхности будет рассчитываться через закон сохранения энергии:

$$\frac{m_0 \cdot v_0^2}{2} = m_0 \cdot g \cdot h;$$

$$v_0 = \sqrt{2gh} = 44,72\,\text{м/с};$$

скорость квадрокоптера у поверхности земли составит:

$$v = 0,7 \cdot v_0 = 31,30\,\text{м/с};$$

импульс квадрокоптера составит:

$$p = m \cdot v = 0,35 \cdot 31,30 = 10,96\ (\text{кг} \cdot \text{м})/\text{с}.$$

Ответ: импульс квадрокоптера $p = 10,96$ (кг·м)/с.

Задача VI.1.2.5. (25 баллов)

Темы: робот, плотность.

Условие

Алюминиевый робот в форме идеального шара, в котором имеется замкнутая полость, покоится на поверхности воды. Наружный объем робота составляет $V = 22$ см^3. Каков объем внутренней полости робота $V_\text{п}$ [см^3], если робот плавает на поверхности, погрузившись в воду на 35% от наружного объема? Принять плотность алюминия $\rho_\text{а} = 2700$ кг/м^3, плотность воды $\rho_\text{в} = 1020$ кг/м^3. Массой воздуха в полости пренебречь, ответ округлить до целого.

Критерии оценивания

1. Верно записано уравнения равновесия сил — 5 баллов.
2. Верно записано выражение для расчета объема алюминия — 10 баллов.
3. Верно вычислено значение объема внутренней полости — 10 баллов.

Решение

Уравнение равновесия сил [1]:

$$m \cdot g = \rho_\text{в} \cdot g \cdot V_\text{погр.ч.};$$

объем алюминиевой части робота:

$$V_\text{а} = V - V_\text{п};$$

масса робота сосредоточена в алюминиевой составляющей:

$$m = \rho_\text{а} \cdot V_\text{а};$$

уравнение равновесия сил [2]:

$$\rho_{\text{а}} \cdot g \cdot V_{\text{а}} = 0,35 \cdot \rho_{\text{в}} \cdot g \cdot V;$$

объем алюминиевой составляющей:

$$V_{\text{а}} = \frac{0,35 \cdot \rho_{\text{в}} \cdot g \cdot V}{\rho_{\text{а}} \cdot g};$$

объем внутренней полости робота:

$$V_{\text{п}} = V - \frac{0,35 \cdot \rho_{\text{в}} \cdot g \cdot V}{\rho_{\text{а}} \cdot g} = V\left(1 - \frac{0,35 \cdot \rho_{\text{в}}}{\rho_{\text{а}}}\right) = 19\,\text{см}^3.$$

Ответ: объем внутренней полости робота $V_{\text{п}} = 19$ см^3.

Физика. 10–11 классы

Задача VI.1.3.1. (12 баллов)

Темы: пружина, жесткость.

Условие

Система амортизации, которая лежит в основе платформы для приземления дрона, состоит из металлической пластины с плотностью материала $\rho = 2800$ кг/м^3 в форме прямоугольного параллелепипеда со сторонами $400 \times 300 \times 4$ мм и четырех одинаковых удерживающих ее пружин, расположенных вертикально по углам пластины и прикрепленных к полу.

Под весом пластины пружины равномерно сжимаются на $x_1 = 3,1$ см относительно состояния покоя. Дрон массой M медленно приземляется на платформу, дополнительно сжимая пружину в вертикальном направлении на максимальное расстояние $x_2 = 2,2$ см.

Требуется рассчитать массу дрона M [кг].

Критерии оценивания

1. Верно записано выражение для поиска массы пластины — 2 балла.
2. Верно записано выражение равновесия сил без учета массы дрона — 2 баллов.
3. Верно записано выражение равновесия сил с учетом нахождения дрона на платформе — 4 баллов.
4. Верно вычислена масса M дрона — 4 баллов.

Решение

Приведен один из вариантов решения задачи.

Масса пластины:

$$m_{\text{пластины}} = \rho_{\text{пластины}} \cdot V_{\text{пластины}}.$$

Равновесие сил при покоящейся платформе на пружинах будет достигаться на основе равенства силы тяжести платформы и суммарной силы упругости пружин:

$$m_{\text{пластины}} \cdot g = 4 \cdot k \cdot x_1;$$

$$k = \frac{m_{\text{пластины}} \cdot g}{4 \cdot x_1}.$$

Применим значение коэффициента упругости для нахождения массы дрона:

$$(M + m_{\text{пластины}}) \cdot g = 4 \cdot k \cdot (x_1 + x_2);$$

$$M = \frac{4 \cdot \frac{\rho_{\text{пластины}} \cdot V_{\text{пластины}} \cdot g}{4 \cdot x_1} \cdot (x_1 + x_2) - \rho_{\text{пластины}} \cdot V_{\text{пластины}} \cdot g}{g};$$

$$M = \frac{\rho_{\text{пластины}} \cdot V_{\text{пластины}}}{x_1} \cdot (x_1 + x_2) - \rho_{\text{пластины}} \cdot V_{\text{пластины}}.$$

$$M = 0,95 \text{ кг}.$$

Ответ: масса дрона $M = 0,95$ кг.

Задача VI.1.3.2. (12 баллов)

Темы: коптер, импульс.

Условие

Квадрокоптер массой $m = 350$ г начал свободное падение с высоты $h = 0,1$ км по причине разрядки аккумулятора. Близ поверхности земли его скорость оказалась на 30% меньше скорости свободно падающего без влияния атмосферы тела, имеющего массу $m_0 = 1$ кг. Чему равен импульс такого квадрокоптера p [(кг·м)/с] в момент удара о землю? Считать ускорение свободного падения $g = 10$ м/с^2.

Критерии оценивания

1. Верно составлен закон сохранения энергии — 3 балла.
2. Верно записано выражение по нахождению скорости — 3 баллов.
3. Верно вычислено значение скорости — 3 баллов.
4. Верно вычислено значение импульса — 3 баллов.

Решение

В том случае если воздух не оказывает сопротивления движению тела, скорость такого тела v_0 близ поверхности будет рассчитываться через закон сохранения энергии:

$$\frac{m_0 \cdot v_0^2}{2} = m_0 \cdot g \cdot h;$$

$$v_0 = \sqrt{2gh} = 44,72\,\text{м/с};$$

скорость квадрокоптера у поверхности земли составит:

$$v = 0,7 \cdot v_0 = 31,30\,\text{м/с};$$

импульс квадрокоптера составит:

$$p = m \cdot v = 0,35 \cdot 31,30 = 10,96\,(\text{кг} \cdot \text{м})/\text{с}.$$

Ответ: импульс квадрокоптера $p = 10,96$ (кг·м)/с.

Задача VI.1.3.3. (22 баллов)

Темы: конвейер, мощность.

Условие

Промышленное производство автомобиля с автопилотом организовано на основе выполнения простых операций автоматически в отведенных зонах. Перемещение деталей от зоны обезжиривания и склеивания до следующей зоны выполняется с помощью конвейерной ленты длиной $L = 10$ м, угол наклона к горизонту равен $\alpha = 20°$, скорость движения ленты $v = 1,5$ м/с. Каждая деталь имеет массу $m = 3$ кг, на ленту конвейера по ширине помещается ряд по 3 таких детали, а расстояние между центрами соседних рядов деталей по длине (в проекции на горизонт) составляет $l = 20$ см. Конвейер приводится в движение с помощью полезной работы двигателя, КПД механической части ленты $\eta = 0,75$. Рассчитать минимальную необходимую мощность N, кВт двигателя, если требуется перемещать максимальное количество деталей вверх.

Критерии оценивания

1. Верно учтен наклон поверхности конвейера при расчете — 5 баллов.
2. Верно записано выражение по нахождению количества помещающихся деталей — 5 баллов.
3. Верно применен КПД при решении — 5 баллов.
4. Верно вычислено значение мощности — 7 баллов.

Решение

Общее количество деталей, помещающихся на ленту, с учетом наклона:

$$n = 3 \cdot \frac{L \cdot \cos\alpha}{l}.$$

Сила тяжести всех деталей составит:

$$F_\text{т} = n \cdot m \cdot g.$$

Вертикальная составляющая скорости перемещения:

$$v_y = v \cdot \sin\alpha.$$

Мощность, вырабатываемая двигателем, лишь частично идет на полезную работу по перемещению. С учетом КПД исходная мощность двигателя:

$$N = \frac{N_0}{\eta} = \frac{F_т \cdot v_y}{\eta} = \frac{n \cdot m \cdot g \cdot v \cdot \sin\alpha}{\eta} = \frac{3 \cdot L \cdot \cos\alpha \cdot m \cdot g \cdot v \cdot \sin\alpha}{\eta \cdot l} =$$

$$= \frac{3 \cdot 10 \cdot 0,94 \cdot 3 \cdot 10 \cdot 1,5 \cdot 0,34}{0,75 \cdot 0,2} = 2,9\,\text{кВт}.$$

Ответ: мощность двигателя $N = 2,9$ кВт.

Задача VI.1.3.4. (25 баллов)

Темы: лазер, энергия.

Условие

В экспериментальных целях на беспилотный летательный аппарат был установлен лазер импульсного типа. После прохождения через оптическую систему лазера, имеющую фокусное расстояние $f = 33$ см, световой пучок представляет собой конус с углом $\theta = 2$ мрад при вершине. Чтобы выполнить задачу, световой импульс должен достигать плотности мощности $q_0 = 2 \cdot 10^8$ Вт/см^2 в фокусе.

Нарисовать оптическую схему. Определить мощность импульса P, Вт и энергию импульса (округлить до целого) E, нДж при длительности импульса $\tau = 10^{-6}$ с.

Критерии оценивания

1. На оптической схеме показан сходящийся световой пучок, указано положение фокуса — 2 балла.
2. Верно записано выражение по нахождению диаметра пятна — 5 баллов.
3. Верно вычислено значение площади пятна — 5 баллов.
4. Верно вычислено значение мощности — 5 баллов.
5. Верно вычислено значение энергии импульса — 5 баллов.

Решение

Диаметр пятна в фокусе:

$$D = f \cdot \theta = 33 \cdot 10^{-2} \cdot 2 \cdot 10^{-3} = 66 \cdot 10^{-5}\,\text{м}.$$

Площадь пятна в фокусе:

$$S = \pi \cdot R^2 = \pi \cdot \frac{D^2}{4} = \pi \cdot \frac{(f \cdot \theta)^2}{4} = 3,421 \cdot 10^{-7}\,\text{м}^2.$$

Мощность импульса в фокусе:

$$P = q_0 \cdot S = q_0 \cdot \pi \cdot \frac{(f \cdot \theta)^2}{4} = 2 \cdot 10^4 \cdot 3,421 \cdot 10^{-7} = 6,8 \cdot 10^{-3}\,\text{Вт}.$$

Энергия импульса:

$$E = P \cdot \tau = 0,000068424\,\text{Дж} = 6,8\,\text{нДж}.$$

Ответ: диаметр пятна в фокусе $D = 66 \cdot 10^{-5}$ м; площадь пятна в фокусе $S = 3,421 \cdot 10^{-7}$ м^2; мощность импульса в фокусе $P = 6,8 \cdot 10^{-3}$ Вт; энергия импульса $E = 68$ нДж.

Задача VI.1.3.5. (32 баллов)

Темы: робот, плотность.

Условие

Алюминиевый робот состоит из оптического модуля в форме половины шара и правильной шестиугольной призмы в основе. Центр шара принадлежит центральной точке верхней плоскости призмы. Оптический модуль сплошной (без полости) и имеет среднюю плотность, равную плотности алюминия. В призме присутствует замкнутая полость в форме правильной шестиугольной призмы, закрытая сверху и снизу тонкими стенками, не имеющими объема.

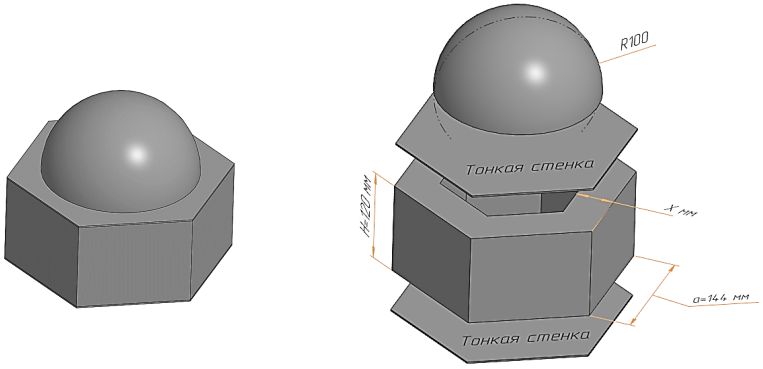


Рис. VI.1.1. Строение робота

Робот плавает в воде таким образом, что над уровнем воды расположено 40% от объема оптического модуля робота. Какова толщина стенки, если плотность алюминия $\rho_\text{а} = 2700$ кг/м^3, плотность воды $\rho_\text{воды} = 1000$ кг/м^3?

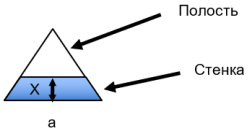
Критерии оценивания

1. Верно записано выражение силы тяжести — 5 баллов.
2. Верно записано выражение силы Архимеда, действующей на корпус — 5 баллов.

3. Верно вычислено значение (применена формула) объема оптического модуля — 2 балла.
4. Верно вычислено значение (применена формула) объема призмы — 2 балла.
5. Верно вычислено значение объема полости — 6 баллов.
6. Верно записано выражение по нахождению толщины стенки, исходя из геометрии корпуса — 6 баллов.
7. Дан верный ответ — 6 баллов.

Решение

Рассмотрим один сегмент основания робота. В сечении он представляет собой 2 правильных подобных треугольника с общим углом при вершине 60°.



Высота большего треугольника состоит из высоты меньшего треугольника h и толщины стенки X. Площади подобных треугольников относятся так же, как относятся квадраты их высот:

$$\frac{S}{S_{\text{полости}}} = \left(\frac{X+h}{h}\right)^2.$$

Высота большего треугольника может быть вычислена через длину стороны:

$$X + h = \frac{a \cdot \operatorname{ctg} 30^\circ}{2} = \frac{a \cdot \sqrt{3}}{2};$$

$$(X+h)^2 = 0,75 \cdot a^2;$$

$$h = \sqrt{\frac{S_{\text{полости}} \cdot 0,75 \cdot a^2}{S}}.$$

Толщина стенки может быть найдена как разность большей и меньшей высот треугольников:

$$X = a \cdot \left(\frac{\sqrt{3}}{2} - \sqrt{\frac{S_{\text{полости}} \cdot 0,75}{S}}\right).$$

Площадь сечения призмы равна отношению объема к высоте. Учитывая нулевой объем верхней и нижней стенок призмы, выразим толщину стенки через объемы:

$$X = a \cdot \left(\frac{\sqrt{3}}{2} - \sqrt{\frac{V_{\text{полости}} \cdot 0,75}{V_{\text{призмы}}}}\right).$$

Объем робота равен сумме объема половины шара и объема призмы:

$$V = V_{\text{п.ш.}} + V_{\text{призмы}}.$$

Объем алюминиевого корпуса равен разности общего объема робота и объема полости:

$$V_{\text{а}} = V - V_{\text{п}} = V_{\text{п.ш.}} + V_{\text{призмы}} - V_{\text{полости}}.$$

Сила тяжести, действующая на робота:

$$F_{\text{т}} = g \cdot \rho_{\text{а}} \cdot V_{\text{а}} = g \cdot \rho_{\text{а}} \cdot (V_{\text{п.ш.}} + V_{\text{призмы}} - V_{\text{полости}}).$$

Погруженный в воду объем:

$$V_{\text{погр}} = 0,6 \cdot V_{\text{п.ш.}} + V_{\text{призмы}}.$$

Сила Архимеда, действующая на робота:

$$F_{\text{А}} = g \cdot \rho_{\text{воды}} \cdot V_{\text{погр}} = g \cdot \rho_{\text{воды}} \cdot (0,6 \cdot V_{\text{п.ш.}} + V_{\text{призмы}}).$$

Уравнение баланса сил с учетом преобразований:

$$\rho_{\text{а}} \cdot V_{\text{полости}} = V_{\text{п.ш.}} \cdot (\rho_{\text{а}} - 0,6 \cdot \rho_{\text{воды}}) + V_{\text{призмы}} \cdot (\rho_{\text{а}} - \rho_{\text{воды}}).$$

Объем полости с подстановкой значений плотностей материалов (без потери точности):

$$V_{\text{полости}} = \frac{7}{9} V_{\text{п.ш.}} + \frac{17}{27} V_{\text{призмы}}.$$

Объем призмы:

$$V_{\text{призмы}} = \frac{3\sqrt{3}a^2 H}{2} = 0,006465\,\text{м}^3.$$

Объем полости:

$$V_{\text{полости}} = \frac{7}{9} \cdot \frac{\pi \cdot D^3}{12} + \frac{17}{27} \cdot 0,006465 = 0,005700\,\text{м}^3.$$

Отношение объемов полости и призмы:

$$\frac{V_{\text{полости}}}{V_{\text{призмы}}} = 0,881671.$$

С учетом того факта, что полученное отношение будет справедливо как для полных значений объемов, так и для объемов одного сегмента призмы, подставим значение в формулу расчета толщины стенки:

$$X = 0,144 \cdot \left(\frac{\sqrt{3}}{2} - \sqrt{0,881671 \cdot 0,75} \right) = 0,007610\,\text{м}.$$

Ответ: $X = 7,61$ мм.

Инженерный тур

Общая информация

Целью задачи заключительного этапа НТО по профилю Летающая робототехника является разработка комплексной системы мониторинга строительной площадки с использованием квадрокоптера. Система должна обеспечивать автоматизированный сбор, обработку и анализ данных для оптимизации строительных процессов и повышения их эффективности.

Легенда задачи

Эффективный мониторинг и управление процессами на строительных площадках играют ключевую роль в сокращении времени и затрат на строительство, а также в повышении уровня безопасности труда. С использованием квадрокоптеров строительные команды получают возможность оперативно собирать данные о состоянии объектов, получать точные изображения, тем самым выполнять контроль за выполнением строительных работ в соответствии с проектной документацией, что способствует принятию своевременных и обоснованных решений.

Требования к команде и компетенциям участников

Количество участников в команде: 3–4 человека.

Роль 1. Инженер-программист (Python) — написание кода для автономного полета квадрокоптера, работа с сервером, разработка алгоритма безопасного полета квадрокоптера.

Роль 2. Инженер-программист (C++, Python) — алгоритмы компьютерного зрения для реализации автономных миссий квадрокоптера, машинное обучение. Работа в связке с ролью 1.

Роль 3. Инженер-техник — моделирование и изготовление устройства, тестирование, техобслуживание и пилотирование квадрокоптера, работа с технической документацией.

Роль 4. Капитан/лидер команды — работа с системами построения карты в `RViz`, осуществление общего руководства работой команды, распределение обязанностей и контроль соблюдения дедлайнов. Рекомендуется совмещение данной роли с другими ролями.

Оборудование и программное обеспечение

Описание оборудования, ПО, полигона, стендов и других ресурсов, которые используются для решения задачи.

Наименование	**Описание**
Полетный полигон ($5 \times 7 \times 3$ м)	Полигон для запуска автономных полетных миссий
Персональный компьютер (Intel Core i5 7260u 2.2 GHz, 8 Gb RAM)	Разработка программного кода для запуска полетной миссии
Ноутбук (Ryzen 5 5500u 2.1 GHz, 20 Gb RAM)	Разработка дронопорта
Графическая станция (Ryzen 7 1700 3.7 GHz, 16Gb RAM, GeForce GTX 1060 3Gb)	Разработка программного кода для запуска полетной миссии
Конструктор программируемого квадрокоптера «COEX Клевер 4 Code» или «Гаскар Клевер 4 Code»	Запуск полетных миссий
3D-принтер	Печать деталей для разработанного дронопорта
Лазерный станок	Резка фанеры для изготовления разработанного дронопорта
Полетный полигон ($5 \times 7 \times 3$ м)	Полигон для запуска автономных полетных миссий
• T-flex CAD 17 • Компас 3D v21 • Cura • Repetier host	Программное обеспечение для разработки и печати деталей для дронопорта

Наименование	Описание
• VMWare Workstation (Gazebo) • Putty • Winscp • Notepad++ • QGroundControl • Etcher • ColorMania • Arduino IDE • VLC player • VScode • Python3 • Termius	Программное обеспечение для разработки программного кода для выполнения заданной полетной миссии

Описание задачи

Конфигурация площадки

Программирование автономной миссии квадрокоптера для осуществления мониторинга и анализа застройки нового микрорайона и прилегающей к нему дороги.

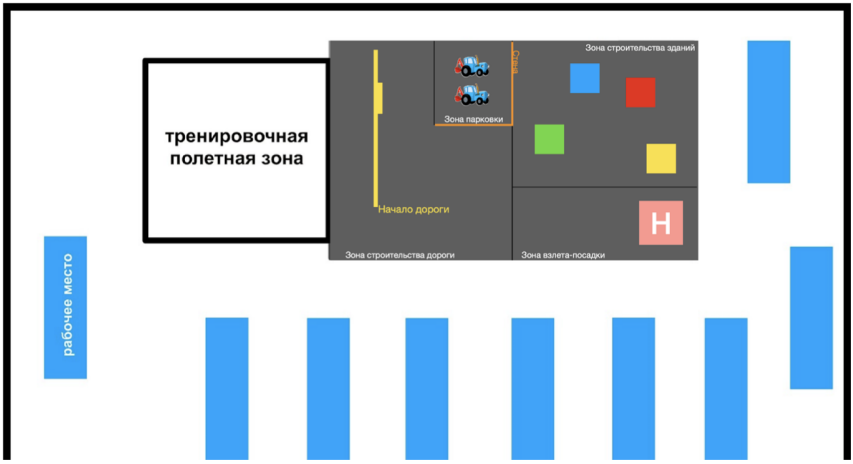


Рис. VI.2.1. Общая конфигурация площадки

1. Зона «H» является точкой взлета и точкой посадки.
2. Конфигурация застройки может изменяться.
3. Положение датчиков на квадрокоптере может быть изменено при необходимости.

4. Миссия мониторинга должна проходить полностью в автономном режиме.
5. Доступ к сети осуществляется через роутер, выданный каждой команде; также имеется подключенный к сети роутер в полетной зоне.
6. Для общения с сервером используется протокол REST API.
7. Сервер находится в интернете, доступ возможен с любого устройства:
   - для доступа в интернет квадрокоптер необходимо перенастроить в режим клиента: https://clover.coex.tech/ru/network.html#переключение-адаптера-в-режим-клиента;
   - адрес сервера: http://65.108.156.108;
   - на странице http://65.108.156.108/docs доступна Swagger документация каждого запроса;
   - в каждый запрос необходимо подложить токен команды доступный на вашем сетевом диске с использованием `http` заголовка `Authorization: Bearer <token>`.
8. Команда несет ответственность за сохранность данных при работе в сети (нестандартный пароль на Wi-Fi, нестандартный пароль для SSH на квадрокоптере).
9. В каждом задании началом координат принят левый нижний угол (рис. VI.2.1), центр aruco-маркера.

Программная часть

Программирование автономной миссии квадрокоптера для осуществления мониторинга и анализа застройки нового микрорайона и прилегающей к нему дороги.

1.1. Мониторинг зданий

Рис. VI.2.2. Вид зданий сбоку

От координационного центра строительства нового микрорайона вам поставлена задача провести мониторинг стадии строительства зданий. Для успешного выполнения задания необходимо:

1. Совершить автономный взлет с зоны «H».
2. Выполнить полет к застраиваемому микрорайону (координаты четырех углов (4,0; 1,0); (4,0; 4,0); (7,0; 4,0); (7,0; 1,0)).
3. Выполнить общее фото зоны строительства.
4. Выполнить сканирование зоны застройки:
   - найти все здания в зоне строительства и определить координаты их центров (в системе `aruco_map`, возможная погрешность 0,2 м по каждой оси);
   - по цветному маркеру определить целевую (`target_height`) этажность здания:
     4.1. красный — 4 этажа;
     4.2. синий — 3 этажа;
     4.3. зеленый — 2 этажа;
     4.4. желтый — 1 этаж.
   - определить количество возведенных этажей каждого здания на момент мониторинга (1 этаж = 0,25 м).
5. Отобразить каждое обнаруженное здание, привязанное к системе координат в системе визуализации (RViz или аналоге, используя marker array), соответствующего цвета и реальной высоты.
6. Сделать запрос на сервер в соответствии со спецификацией http://65.108.156.108/docs#/default/send_building_buildings_post.
7. В ответ от сервера получить информацию о здании, которое необходимо проанализировать более детально в формате $[x; y]$.
8. Подлететь к зданию, координаты которого был получены в ответ и выполнить фото с реальной высоты здания + 0,5 м.

1.2. Мониторинг строительства дороги

Выполнение мониторинга дорожного полотна, примыкающего к зоне строительства, и анализ:

- эффективности работы строительной бригады;
- качественных и количественных характеристик прокладки дорожного полотна.

Для успешного выполнения задания необходимо:

- Прилететь к началу дорожного полотна (координаты (0,5; 0,5)).
- Выполнить полет вдоль дорожного полотна, выполняя мониторинг:
  - количества проложенных полос дорожного полотна (одна полоса — N см) для каждого отрезка (см. рис. VI.2.3 и рис. VI.2.4);
  - координаты границы каждого отрезка;
  - общую длину построенной дороги и длину каждого отрезка (м);
  - количество рабочих (каски или цветные метки красного цвета), выполняющих свою работу (сотрудник считается выполняющим работу, если он находится не дальше 0,3 м от дорожного полотна по перпендикулярной прямой).
- Отобразить дорогу и рабочих с привязкой к системе координат в системе визуализации (RViz или аналоге, используя marker array):
  - дорога должна быть желтого цвета, размер должен соответствовать фи-

зическому;

- рабочие должны быть обозначены соответствующим цветом, форма — круг, размер — 0,1 м;
- иные сотрудники должны быть обозначены соответствующим цветом, форма — квадрат, размер — 0,1 м.

- Отправить на сервер полученные данные в соответствии со спецификацией http://65.108.156.108/docs#/default/send_road_roads_post.

Рис. VI.2.3. Пример 1

Рис. VI.2.4. Пример 2

1.3. Контроль автомобильной спецтехники

Автомобильная спецтехника используется для разработки участков, перемещения строительных материалов по территории застройки, выполнения монтажных и демонтажных работ. Одной из задач мониторинга территории застройки является контроль доступности строительной спецтехники.

Необходимо осуществить контроль доступности строительной спецтехники:

- Прилететь в зону парковки спецтранспорта (координаты четырех углов (1,5; 3,0); (1,5; 4,5); (3,0; 4,5); (3,0; 3,0)).
- Определить доступные типы строительной спецтехники из указанных ниже в пределах зоны с использованием любого алгоритма.
- Отправить на сервер полученные данные в соответствии со спецификацией http://65.108.156.108/docs#/default/send_cars_cars_post.

бетоносмеситель `cement_mixer` | автокран `truck_crane` | грузовик `truck` | экскаватор `excavator`

Рис. VI.2.5. Типы строительной спецтехники

Инженерное задание

Техническое задание на разработку дронопорта

1. Наименование выполняемых работ: разработка дронопорта – устройства для безопасного взлета, посадки и подзарядки квадрокоптера.
2. Цель выполнения работ: создание устройства для осуществления безопасного взлета, посадки и подзарядки квадрокоптера с интеграцией всех необходимых функций и соблюдением требований безопасности.
3. Срок выполнения работ:
   - Документация и создание устройства: 3 рабочих дня, с 19.03.2024 по 21.03.2024 г.
   - Демонстрация работоспособности прототипа устройства: с 11:00 22.03.2024 г, в 11:00 разработанное устройство сдается на карантин.
   - Финальное испытание устройства: во выполнение зачетного полета 22.03.2024 г.
4. Требования к устройству:
   - Взлет и посадка квадрокоптера: обеспечение безопасности взлета и посадки квадрокоптера.
   - Безопасность: обеспечение безопасности операций с квадрокоптером, включая защиту от несанкционированного доступа и вмешательства. Квадрокоптер должен быть защищен от внешнего воздействия со всех сторон.
   - Сигнализирование о нахождении квадрокоптера внутри устройства.
   - Сигнализирование об открытии устройства для совершения взлета и посадки квадрокоптера.
   - Сигнализирование о закрытии устройства после совершения взлета и посадки.
   - Осуществление замыкания контактной площадки подзарядки:
     – контактные площадки команда самостоятельно проектирует и создает;
     – контактная площадка дронопорта должны состоять из одной площадки размером не более 3×6 см (имитация контактных площадок $+$ и $-$); ответный контакт на квадрокоптере, касающийся контактной площадки, должен быть размером не более 3×6 см;
     – под контактной площадкой дронопорта должен находиться концевой выключатель;

  - контактная площадка должна быть подпружинена для недопущения срабатывания концевика при отсутствии внешнего воздействия;
  - функция подзарядки осуществляется при длительном (не менее 5 с) срабатывании концевика.
- Сигнализирование об осуществлении подзарядки квадрокоптера.
- Устройство должно быть единым (все датчики и механическая часть должны быть интегрированы в данное устройство).
- Максимальные габариты устройства в закрытом состоянии — $800 \times 800 \times \times 800$.
- Все вышеперечисленные функции должны выполняться в автономном режиме.

5. Требования к документации:
   - 3D-модель устройства в сборке (`.step` и исходный формат программы).
   - Сборочный чертеж устройства (https://dgng.pstu.ru/sprav/10.4.htm) (оформление согласно ГОСТ 2.109-73 https://drive.google.com/file/d/1BKR930fFXkLQBRlPpwxInyyJr9_Fn1Dn/view?usp=sharing):
     - отображение всех деталей и стандартных изделий;
     - количество изображений должно быть наименьшим, но достаточным для представления расположения и взаимной связи составных частей и обеспечивающим возможность осуществления сборки и контроля сборки.
   - Спецификация (оформление согласно ГОСТ 2.109-73 https://drive.google.com/file/d/1BKR930fFXkLQBRlPpwxInyyJr9_Fn1Dn/view?usp=sharing):
     - наличие всех деталей;
     - наличие стандартных изделий.
   - Инструкция по сборке устройства (все пункты относятся как к сборке механической части устройства, так и электрической):
     - список всех компонентов и материалов, необходимых для сборки устройства;
     - последовательное описание шагов сборки, начиная с подготовки рабочего места и инструментов;
     - иллюстрации или схемы, помогающие понять процесс сборки;
     - подробное описание каждого шага с указанием порядка действий и возможных трудностей;
     - инструкции по проверке правильности сборки и испытания устройства;
     - рекомендации по безопасности при работе с материалами и инструментами.
   - Инструкция по эксплуатации устройства:
     - описание устройства: общие характеристики, назначение, особенности конструкции;
     - инструкции по началу эксплуатации: правила подключения, настройки и первоначального запуска;
     - рекомендации по использованию и безопасности: ограничения по применению, рекомендации по уходу и обслуживанию, предупреждения о потенциальных опасностях, правила безопасного исполь-

зования изделия;

– программный код, необходимый для эксплуатации устройства.

6. Результат работ: устройство, соответствующее требованиям, указанным в п. 4 Технического задания.
7. Сдача работы: документация; отчет выполнения инженерной части.

Отчет выполнения инженерной части

№ п. п.	Наименование	Формат
1	3D-модель функционального устройства в сборке	`.step` и проприетарный формат
2	Сборочный чертеж устройства	`.pdf`
3	Спецификация	`.pdf`
4	Инструкция по сборке устройства	`.pdf`
5	Инструкция по эксплуатации устройства	`.pdf`

Демонстрация работоспособности прототипа устройства

Проводится на рабочем месте команды. Во время демонстрации экспертами проверяется соответствие устройства требованиям из п. 4.

Финальное испытание устройства

Проходит во время выполнения финальной зачетной миссии. Оценка результатов происходит только в тех случаях, если все составляющие итогового результата:

- предоставлены организаторам не позднее указанного срока;
- все разработанные материалы и само устройство соответствуют техническому заданию.

Система оценивания

Задачу заключительного этапа команды получают в виде отдельных заданий на каждый день. В таблице представлены критерии оценки заключительной зачетной попытки Задание. День 4.

В задании присутствуют требования визуализации некоторых частей полигона. Порядок оценки визуализации:

- Для оценивания визуализации на зачетной попытке команда должна представить экспертам компьютер с открытым визуализатором (`RViz` или аналог).
- В визуализации должны присутствовать отображения начал систем координат (компонент `TF` в `RViz`) `aruco_map`, `map`.
- По требованию экспертов быть готовыми (иметь настроенное для этого ПО, пример OBS) запустить запись экрана в начале зачетных полетов команды, предоставить эксперту доступ к вращению мира при успешном окончании полета.

- По окончанию полета все визуализированные элементы должны остаться на своих местах.

В таблице представлены критерии оценки заключительной зачетной миссии.

№	Критерий (условия выполнения миссии)	Баллы за один случай	Кол-во случаев[1]	Баллы за все
1.1	Выполнилась функция сигнализирования[2] об открытии устройства перед совершением взлета квадрокоптера. Функция может осуществляться только в автономном режиме.	0,6	1	0,6
1.2	Крышка дронопорта открывается перед осуществлением взлета в автоматическом режиме: • в автономном или полуавтономном режиме (связь с квадрокоптером, компьютером, планшетом, телефоном, сервером) — 100%; • в ручном режиме (внешнее воздействие: кнопки или подобных устройств) — 50%. Использование механической силы участников запрещено.	1	1	1
1.3	Совершен автономный взлет с/из дронопорта	0,8	1	0,8
1.4	Крышка дронопорта закрывается после осуществлением взлета: • в автономном или полуавтономном режиме (связь с квадрокоптером, компьютером, планшетом, телефоном, сервером) — 100%; • в ручном режиме (внешнее воздействие: кнопки или подобные устройства) — 50%.	2	1	2
1.5	Выполнилась функция сигнализирования[2] о закрытии устройства после совершения взлета квадрокоптера. Функция может осуществляться только в автономном режиме.	0,7	1	0,7
1.6	Получено общее фото зоны строительства с высоты 2 м, из точки центра.	0,5	1	0,5
1.7	Для каждого здания в зоне строительства верно определены координаты центров (в системе `aruco_map`, возможная погрешность 0,2 м по каждой оси). Проверяется по данным отправленным на сервер.	0,7	4	2,8
1.8	Верно определено количество возведенных этажей здания на момент съемки. Проверяется по данным отправленным на сервер.	0,8	4	3,2
1.9	Верно определено целевое количество этажей. Проверяется по данным отправленным на сервер.	0,6	4	2,4

№	Критерий (условия выполнения миссии)	Баллы за один случай	Кол-во случаев[1]	Баллы за все
1.10	Получено фото здания с реальной высоты здания + 0,5 м (координаты здания были получены в ответ от сервера).	0,5	1	0,5
2. Мониторинг строительства дороги				
2.1	Верно определена общая протяженность проложенной дороги (м).	1	1	1
2.2	Верно определены координаты границ каждого отрезка дороги (в системе `aruco_map`).	–	–	2,2
2.3	Верно определено количество дорожных полос для каждого отрезка.	–	–	1,2
2.4	Верно определена длина каждого отрезка дороги (м).	–	–	1
2.5	Верно определено количество рабочих (каски или цветные метки красного цвета), выполняющих свою работу.	–	–	1,5
3. `RViz` или аналог **Привязка к системе координат в системе визуализации (`Rviz` или аналоге, используя `marker array`)**				
3.1	Отображено найденное здание, привязанное к системе координат в системе визуализации.	0,5	4	2
3.2	Отображена дорога желтого цвета. Дорога соответствует физической: размер, количество полос, количество отрезков.	1	1	1
3.3	Отображено местоположение всех рабочих: соответствующего цвета, формы круга, размера 0,1 м.	1,2	1	1,2
4. Контроль автомобильной спецтехники				
4.1	Верно определены доступная спецтехника.	2,5	2	5
5. Посадка **Оценивается во время полета**				
5.1	Выполнилась функция сигнализирования[2] об открытии устройства перед совершением посадки квадрокоптера. Функция может осуществляться только в автономном режиме.	0,7	1	0,7
5.2	Крышка дронопорта открывается перед осуществлением посадки: • в автономном или полуавтономном режиме (связь с квадрокоптером, компьютером, планшетом, телефоном, сервером) — 100%; • в ручном режиме (внешнее воздействие: кнопки или подобных устройств) — 50%.	2	1	2
5.3	Осуществлена посадка квадрокоптера • внутрь устройства — 100% баллов; • на устройство — 50% баллов.	2	1	2

№	Критерий (условия выполнения миссии)	Баллы за один случай	Кол-во случаев[1]	Баллы за все
5.4	Выполнилась функция сигнализирования[2] о закрытии устройства после совершения посадки квадрокоптера. Функция может осуществляться только в автономном режиме.	0,6	1	0,6
5.5	Крышка дронопорта закрывается после осуществлением посадки: • в автономном или полуавтономном режиме (связь с квадрокоптером, компьютером, планшетом, телефоном, сервером) — 100%; • в ручном режиме (внешнее воздействие: кнопки или подобных устройств) — 50%.	1	1	1
5.6	Выполнилась функция сигнализирования[2] о нахождении квадрокоптера внутри устройства. Функция может осуществляться только в автономном режиме.	0,6	1	0,6
5.7	После посадки квадрокоптер начал осуществлять подзарядку в дронопорте. Функция может осуществляется только в автоматическом режиме. Оценивается при выполнении функции сигнализирования[2] об осуществлении подзарядки квадрокоптера.	1,8	1	1,8
6. Оценка результата				
6.1	Код зачетной попытки (день 4) с комментариями выложен на Github.	0,3	1	0,3
6.2	В программе присутствует логика считывания данных с датчика (любого) или алгоритм компьютерного зрения, выполняющий поиск зданий, измеряющий количество этажей, определяющий необходимые данные о построенном участке дорожного полотна.	–	–	0,8
ИТОГО ЗА РЕШЕНИЕ ЗАДАЧИ				38,8

Решение задачи

Первым шагом при получении квадрокоптера является его тщательная проверка и подготовка к работе, которую можно разделить на следующие этапы:

1. Выполнить калибровку датчиков квадрокоптера (например, компаса и гиро-

[1]Количество случаев может варьироваться. В случае изменения количества баллы за все случаи делятся на их количество. При «–» общее количество случаев неизвестно, баллы за все случаи делятся на их количество.

[2]Сигнализирование дронопорта:

- Функция реализована при помощи светодиодов — 50% баллов.
- Функция реализована при помощи светодиодной ленты — 80% баллов.
- Функция реализована при помощи светодиодной LED-матрица — 100% баллов.

скопа) согласно рекомендациям производителя.

2. Настроить карту aruco-маркеров в соответствии с реальным полем на площадке.
3. Убедиться, что все системы квадрокоптера, включая полетный контроллер, Raspberry Pi, камеру, систему навигации по aruco-маркерам, работают исправно. Для этого можно воспользоваться инструментом `self-check`, согласно инструкции на сайте https://clover.coex.tech/ru/selfcheck.html, а также `web_video_server`, который позволяет выполнить визуальную проверку.

Для решения задачи 1.1 можно разбить процесс на несколько ключевых шагов, каждый из которых имеет важное значение для успешного выполнения миссии:

1. С использованием координат четырех углов застроенного микрорайона экспериментальным путем разработать маршрут, который обеспечивает полное покрытие зоны мониторинга.
2. Разработать систему детектирования зданий по цветам крыш. Для этого может использоваться как простая система с детекцией цвета и выравнивания к нему по двухосевому (x и y) ПИД-регулятору или более сложная с использованием геометрии изображения с камеры (см. PnP-задача: https://waksoft.susu.ru/2020/02/23/geometriya-formirovaniya-izobrazhenij/).

Кроме того, можно постоянно считывать показания с дальномера и ждать, пока наша текущая высота квадрокоптера, определенная системой навигации, будет отличаться от показаний дальномера — это означает, что в данный момент мы находимся над зданием. После этого использовать те же методы работы с изображением.

```
import rospy
from sensor_msgs.msg import Image, CameraInfo, Range
from cv_bridge import CvBridge
from clover import srv

import os

from geometry_msgs.msg import Quaternion, PoseStamped
from visualization_msgs.msg import Marker, MarkerArray

import tf
import tf.transformations as t

import numpy as np
import math
import cv2

from min_max_map import GetMinMaxXYMap

import pprint
from threading import Thread, Event
import threading


class GetMinMaxXYMap:
    def __init__(self) -> None:
        self.dynamic_reconfigure = __import__('dynamic_reconfigure.client')

    def get_minmax_xy_map(self, pogr=0.2):
        map_client = self.dynamic_reconfigure.client.Client('aruco_map')
        with open(map_client.get_configuration()['map']) as f:
```

```
32              n = list(map(lambda x: x.strip().split(), f.readlines()))
33          n = n[1:]
34          self.max_x_map = max(map(lambda x: float(x[2]), n)) + pogr
35          self.max_y_map = max(map(lambda x: float(x[3]), n)) + pogr
36          self.min_x_map, self.min_y_map = -pogr, -pogr
37
38
39  class ObjectDetection:
40      def __init__(self, objects, transform_method='pnp', init=True, dev='virtual',
        ↪  cnt_method='default'):
41          self.STOP_ALL = False
42          self.cnt_pub = False
43          self.mask_pub = False
44          self.rectify_anytime = False
45          self.max_cnt = False
46          if dev == 'from_virtual_to_clover':
47              os.environ['ROS_MASTER_URI'] = 'http://192.168.11.1:11311'
48
49          if init:
50              rospy.init_node('a', disable_signals=True)
51
52          self.init_max_min_map()
53          self.objects = objects
54          self.publishers = {}
55          for name in self.objects.keys():
56              self.publishers[name] = rospy.Publisher(f'~{name}', Image,
                ↪  queue_size=1)
57
58          self.masks = {}
59          for name in self.objects.keys():
60              self.masks[name] = None
61
62          self.img_from_topic = None
63          self.img_for_visualisation = None
64
65          self.bridge = CvBridge()
66          self.listener = tf.TransformListener()
67
68          self.blue_pub = rospy.Publisher("~blue_viz", MarkerArray, queue_size=1)
69          self.contours_publisher = rospy.Publisher('~cnts', Image, queue_size=1)
70
71          # Словарь с  усредненными найденными точками {"index": [кол-во точек,
            ↪  усредненная точка]
72          self.Points = {}
73          self.lastIndex = 0
74
75          self.transform_method = transform_method
76
77          # Условное получение информации о камере на основе среды разработки
78          if dev == 'virtual':
79              self.camera_info = rospy.wait_for_message('main_camera/camera_info',
                ↪  CameraInfo)
80              self.distortion = np.zeros(5, dtype="float64").flatten()
81              #self.camera_matrix = np.array([[ 332.47884746146343, 0., 320.0], [0.,
                ↪  333.1761847948052, 240.0], [0., 0., 1.]], dtype="float64")
82              self.camera_matrix = np.float64(self.camera_info.K).reshape(3, 3)
83          elif dev == 'clover' or dev == 'from_virtual_to_clover':
84              self.camera_info = rospy.wait_for_message('main_camera/camera_info',
                ↪  CameraInfo)
85              self.camera_matrix = np.float64(self.camera_info.K).reshape(3, 3)
```

```
86              self.distortion = np.float64(self.camera_info.D).flatten()
87
88          # Установите смещение для метода PNP
89          if transform_method == 'pnp':
90              self.OFFSET = [61, 35]
91          elif transform_method == 'rectify_point':
92              self.range_sub = rospy.Subscriber('rangefinder/range', Range,
                ↪  self.range_callback, queue_size=1)
93              self.tolerance_of_recognise_inPixels = 100
94              self.range = None
95              self.get_telemetry = rospy.ServiceProxy('get_telemetry',
                ↪  srv.GetTelemetry)
96
97          # Подписка на тему изображения камеры с троттлингом
98          self.image_sub = rospy.Subscriber('main_camera/image_raw_throttled',
            ↪  Image, self.image_callback, queue_size=1)
99
100         print(self.max_x_map, self.max_y_map, self.min_x_map, self.min_y_map)
101
102         # Запускаем отдельный поток для визуализации изображений
103         if dev != 'clover':
104             thread = threading.Thread(target=self.show, daemon=True)
105             thread.start()
106
107     def show(self):
108         while not rospy.is_shutdown():
109             #print(self.__dict__)
110             if self.img_from_topic is not None:
111                 cv2.imshow('Image', self.img_from_topic)
112                 cv2.imshow('Cnt', self.img_for_visualisation)
113                 for name, mask in self.masks.items():
114                     cv2.imshow(name, mask)
115                 cv2.waitKey(1)
116
117     def init_max_min_map(self):
118         """Инициализируем объект GetMinMaxXYMap и получаем данные карты min-max"""
119         m = GetMinMaxXYMap()
120         m.get_minmax_xy_map()
121         self.max_x_map, self.max_y_map, self.min_x_map, self.min_y_map =
            ↪  m.max_x_map, m.max_y_map, m.min_x_map, m.min_y_map
122
123     def default_cnt(self, mask):
124         conturs, _ = cv2.findContours(mask, cv2.RETR_TREE,
            ↪  cv2.CHAIN_APPROX_SIMPLE) #conturs of all objects
125         # Если размер контуров больше 20, то мы их оставляем
126         conturs = list(filter(lambda x: cv2.contourArea(x) > 50, conturs))
127         return conturs
128
129     def resolve_method(self, cnt, name):
130         obj = self.objects[name] # Получаем данные объекта из словаря
131         xy = self.get_center_of_mass(cnt) # Вычисляем центр масс контура
132
133         # Отрисовка контуров для визуализации
134         cv2.drawContours(self.img_for_visualisation, [cnt], 0, obj[3], 2)
135         cv2.putText(self.img_for_visualisation,name, (int(xy[0]),int(xy[1])),
            ↪  cv2.FONT_HERSHEY_SIMPLEX, 1, obj[3], 1)
136
137         # Выполняем преобразование на основе self.transform_method
138         if self.transform_method == 'pnp':
139             self.pnp(cnt, obj, name)
```

```
140             elif self.transform_method == 'rectify_point':
141                 self.rectify(cnt, name)
142 
143     def image_callback(self, msg):
144         if self.STOP_ALL == True:
145             return
146         self.img_from_topic = self.bridge.imgmsg_to_cv2(msg, 'bgr8')
147         # Конвертировать сообщение изображения ROS в изображение OpenCV
148         self.img_for_visualisation = self.img_from_topic.copy()
149         for name, obj in self.objects.items():
150             mask = self.image_mask(name, obj[0], obj[1], obj[2])
151 
152             conturs = self.default_cnt(mask)
153             # Выбираем самый большой контур, если max_cnt имеет значение True
154             if len(conturs) != 0:
155                 if self.max_cnt == True:
156                     conturs = [max(conturs, key=cv2.contourArea)]
157             for cnt in conturs:
158                 self.resolve_method(cnt, name)
159 
160         # Опубликовать визуализированное изображение, если cnt_pub имеет значение
            ↪ True
161         if self.cnt_pub:
162             self.contours_publisher.publish(
                ↪ self.bridge.cv2_to_imgmsg(self.img_for_visualisation, 'bgr8'))
163         self.publish_markers_blue()
164 
165     def img_xy_to_point(self, xy):
166         """
167         Преобразует координаты изображения в 3D-точки в кадре камеры, используя
            ↪ параметры камеры.
168 
169         Аргументы:
170             xy (кортеж): координаты (x, y) на изображении.
171 
172         Возврат:
173             кортеж: (x, y, z) координаты в кадре камеры.
174         """
175         xy = cv2.undistortPoints(xy, self.camera_matrix, self.distortion,
            ↪ P=self.camera_matrix)[0][0]
176 
177         # Сместим координаты в центр изображения (при условии, что начало
            ↪ координат находится вверху слева)
178         xy -= self.camera_info.width // 2, self.camera_info.height // 2
179         # Рассчитать координаты x, y и z в кадре камеры на основе фокусных
            ↪ расстояний и диапазона
180         fx = self.camera_matrix[0, 0]
181         fy = self.camera_matrix[1, 1]
182 
183         return xy[0] * self.range / fx, xy[1] * self.range / fy, self.range
184 
185     def rectify(self, cnt, name):
186         xy = self.get_center_of_mass(cnt)
187         if xy is None:
188             return
189         if self.transform_method == 'rectify_point':
190             if abs(xy[0] - 160) > self.tolerance_of_recognise_inPixels or abs(120
                ↪ - xy[1]) > self.tolerance_of_recognise_inPixels:
191                 return
192 
```

```
193         # calculate and publish the position of the circle in 3D space
194         if self.range is not None:
195             if self.rectify_anytime == False:
196                 if self.range > 1.8:
197                     return
198             cx_cam1, cy_cam1, cz_cam1 = self.img_xy_to_point(xy)
199             cx_map, cy_map, cz_map, rvec = self.transform_xyz_yaw(cx_cam1,
                ↪  cy_cam1, cz_cam1, (0,0,0), "main_camera_optical", "aruco_map",
                ↪  self.listener)
200 
201             x_ar, y_ar, z_ar, _yaw = self.transform_xyz_yaw(0, 0, 0, (0,0,0),
                ↪  "main_camera_optical", "aruco_map", self.listener)
202 
203             if math.sqrt( (x_ar - cx_map) ** 2 + (y_ar - cy_map) ** 2) < 0.2:
204                 self.magic((cx_map, cy_map, cz_map), (0,0,0), name)
205 
206     def image_mask(self, name, hsv_range, bitwise=False, hsv_range_list=None):
207         img_hsv = cv2.cvtColor(self.img_from_topic, cv2.COLOR_BGR2HSV)
208         mask = cv2.inRange(img_hsv, hsv_range[0], hsv_range[1])
209 
210         if bitwise:
211             for hsv in hsv_range_list:
212                 mask2 = cv2.inRange(img_hsv, hsv[0], hsv[1])
213                 mask = cv2.bitwise_or(mask, mask2)
214 
215         self.masks[name] = mask
216         if self.publishers[name].get_num_connections() > 0:
217             if self.mask_pub:
218                 self.publishers[name].publish(self.bridge.cv2_to_imgmsg(mask,
                    ↪  'mono8'))
219             else:
220                 pass
221 
222         return mask
223 
224     def pnp(self, cnt, obj, name):
225         approx = cv2.approxPolyDP(cnt, 0.04 * cv2.arcLength(cnt, True), True)
226         MARKER_SIDE1_SIZE = obj[4] # in m
227         MARKER_SIDE2_SIZE = obj[4] # in m
228 
229         #Реальные размеры цветной метки
230         self.objectPoint =
231             np.array([(-MARKER_SIDE1_SIZE / 2, -MARKER_SIDE2_SIZE / 2, 0),
232                       (MARKER_SIDE1_SIZE / 2, -MARKER_SIDE2_SIZE / 2, 0),
233                       (MARKER_SIDE1_SIZE / 2, MARKER_SIDE2_SIZE / 2, 0),
234                       (-MARKER_SIDE1_SIZE / 2, MARKER_SIDE2_SIZE / 2, 0)])
235 
236         #Поиск минимального ограничивающего прямоугольника и его площади
237         rect = cv2.minAreaRect(cnt)
238         box = cv2.boxPoints(rect)
239         box = np.int0(box)
240         box_area = cv2.contourArea(box)  + 1e-7
241         c_area = cv2.contourArea(cnt)  + 1e-7
242 
243         if len(approx) == 4:
244             points_img = np.array([np.array(p[0]) for p in approx]) #Извлечение
                ↪  угловых точек
245         else:
246             points_img = box
247 
```

```
248
249         # Проверка положения углов в пределах границ изображения
250         image_shape=(240, 320, 3)
251         minx1 = points_img[:, 0].min() > self.OFFSET[0]
252         miny1 = points_img[:, 1].min() > self.OFFSET[1]
253         minx2 = image_shape[1] - points_img[:, 0].max() > self.OFFSET[0]
254         miny2 = image_shape[0] - points_img[:, 1].max() > self.OFFSET[1]
255
256         if minx1 and minx2 and miny1 and miny2:
257             # извлечение координат объекта в системе координат камеры
258             retval, rvec, tvec = cv2.solvePnP(np.array(self.objectPoint,
                ↪  dtype="float32"), np.array(points_img, dtype="float32"),
                ↪  self.camera_matrix, self.distortion)
259
260             cx_cam1=tvec[0][0]
261             cy_cam1=tvec[1][0]
262             cz_cam1=tvec[2][0]
263
264             #Преобразование координат объекта в системе координат камеры в систему
                ↪  координат аруко карты
265             cx_map, cy_map, cz_map, _ = self.transform_xyz_yaw(cx_cam1, cy_cam1,
                ↪  cz_cam1, rvec, "main_camera_optical", "aruco_map", self.listener)
266             self.magic((cx_map, cy_map, cz_map), rvec, name)
267
268     def magic(self, point, rot, name, radius=0.5):
269         """Функция усредняет координаты в словаре с определенным расстоянием"""
270         if point[0] > self.max_x_map or point[0] < self.min_x_map or point[1] >
            ↪  self.max_y_map or point[1] < self.min_y_map:
271             return
272         if len(self.Points) == 0:
273             self.Points[self.lastIndex] = [1, point, rot, name]
274         else:
275             distances1 = []
276             for index, obj in self.Points.items():
277                 pnts = obj[1]
278                 kol = obj[0]
279
280                 #Находим дистанцию между всеми точками
281                 distance = math.sqrt((pnts[0] - point[0]) ** 2 + (pnts[1] -
                    ↪  point[1]) ** 2)# + (pnts[2] - point[2]) ** 2)
282                 distances1.append((distance, index))
283             #Берем минимальную дистанцию
284             miDST = min(distances1, key=lambda x: x[0])
285
286             if miDST[0] <= radius:
287                 #Текущий список с точкой, который ближе всего к point
288                 check = self.Points[miDST[1]]
289                 kol, pnts, rot1 = check
290
291                 rotx, roty, rotz = rot1
292                 x, y, z = pnts
293
294                 #В kol лежит кол-во точек в x, y координаты точки, которая ближе
                    ↪  всего к point
295                 #Добавляем нашу точку, усредняя ее с предыдущей
296                 self.Points[miDST[1]][1] = ((x * kol + point[0]) / (kol + 1),  (y
                    ↪  * kol + point[1]) / (kol + 1), (z * kol + point[2]) / (kol +
                    ↪  1))
297                 self.Points[miDST[1]][2] = ((rotx * kol + rot[0]) / (kol + 1),
                    ↪  (roty * kol + rot[1]) / (kol + 1), (rotz * kol + rot[2]) /
                    ↪  (kol + 1))
```

```
298                 self.Points[miDST[1]][0] += 1
299             else:
300                 #Если point находится на расстоянии больше radius от всех точек,
                    ↪  то мы добавляем новую
301                 self.Points[self.lastIndex + 1] = [1, point, rot, name]
302                 self.lastIndex += 1
303 
304     def get_center_of_mass(self, mask) -> tuple:
305         M = cv2.moments(mask)
306         if M['m00'] == 0:
307             return None
308         return M['m10'] // M['m00'], M['m01'] // M['m00']
309 
310     def transform_xyz_yaw(self, x, y, z, yaw, framefrom, frameto, listener):
311         p = PoseStamped()
312         p.header.frame_id = framefrom
313         p.pose.position.x = x
314         p.pose.position.y = y
315         p.pose.position.z = z
316         p.pose.orientation = self.orientation_from_euler(yaw[0], yaw[1], yaw[2])
317         #print(p.pose.orientation)
318         pose_local = listener.transformPose(frameto, p)
319         target_x = pose_local.pose.position.x
320         target_y = pose_local.pose.position.y
321         target_z = pose_local.pose.position.z
322         target_yaw = self.euler_from_orientation(pose_local.pose.orientation)
323         return target_x, target_y, target_z, target_yaw
324 
325     def orientation_from_quaternion(self, q):
326         return Quaternion(*q)
327 
328     def orientation_from_euler(self, roll, pitch, yaw):
329         q = t.quaternion_from_euler(roll, pitch, yaw)
330         return self.orientation_from_quaternion(q)
331 
332     def quaternion_from_orientation(self, o):
333         return o.x, o.y, o.z, o.w
334 
335     def euler_from_orientation(self, o):
336         q = self.quaternion_from_orientation(o)
337         return t.euler_from_quaternion(q)
338 
339     def range_callback(self, msg):
340         self.range = msg.range
341 
342     def publish_markers_blue(self):
343         result = []
344         iddd = 0
345 
346         for fs in self.Points.values():
347             m = fs[1]
348             rot = fs[2]
349 
350             marker = Marker()
351             marker.header.frame_id = "aruco_map"
352             marker.header.stamp = rospy.Time.now()
353             marker.ns = "color_markers"
354             marker.id = iddd
355             marker.type =  Marker.CUBE
356             marker.action = Marker.ADD
```

```
357
358             # Позиция и ориентация
359             marker.pose.position.x = m[0]
360             marker.pose.position.y = m[1]
361             marker.pose.position.z = 0
362             marker.pose.orientation = self.orientation_from_euler(rot[0], rot[1],
                ↪  rot[2])
363
364             # Масштаб
365             if self.objects[fs[3]][6] == 'korob':
366                 marker.scale.x = 0.2
367                 marker.scale.y = 0.2
368                 marker.scale.z = m[2]
369                 marker.pose.position.z = m[2]/2
370             else:
371                 marker.scale.x = 0.2
372                 marker.scale.y = 0.2
373                 marker.scale.z = 0.001
374
375
376             # Цвет
377             marker.color.a = 0.8
378             marker.color.r = self.objects[fs[3]][5][0]
379             marker.color.g = self.objects[fs[3]][5][1]
380             marker.color.b = self.objects[fs[3]][5][2]
381
382             result.append(marker)
383             iddd += 1
384
385         # Публикуем маркеры
386         self.blue_pub.publish(MarkerArray(markers=result))
```

3. Используя предварительно запрограммированные команды, выполнить взлет и следовать по запланированному маршруту к застраиваемому микрорайону, придерживаясь заданных высоты и скорости полета.
4. С высокой точки выполнить фотографирование всей зоны для общего обзора.
5. С использованием алгоритма из пункта 2, сканировать зону на предмет наличия зданий, определять их координаты, цвет крыши и количество этажей по высоте и сохранять в память.
6. Используя `RViz` или аналог, визуализировать обнаруженные здания, окрашивая их в соответствии с целевой этажностью и отображая реальные размеры.
7. Сформировать и отправить запрос на сервер с информацией о найденных зданиях и получить координаты здания для детального анализа.
8. На основе полученных координат подлететь к выбранному зданию и сделать фотографии с необходимой высоты для детального анализа.
9. После выполнения всех задач квадрокоптер должен автономно вернуться в зону «H» для посадки.

Пример кода полета квадрокоптера

```
1  # Импортируем библиотеки
2  from detection import *
3  import requests
4
5  #Объявление цветовых диапазонов и прочих параметров
6  objects = {
```

```
    'blue': ( [(93, 125, 124), (120, 255, 253)], False, None, (0, 0, 255), 0.26, (0,0,255), 'korob' ),
    'red': ( [(170, 111, 86), (180, 255, 255)], False, None, (255, 0, 0), 0.125*2, (255,0,0), 'korob' ),
    'green': ( [(53, 68, 115), (74, 255, 255)], False, None, (255, 0, 255), 0.125*2, (0,255,0), 'korob' ),
    'yellow': ( [(21, 83, 131), (34, 255, 255)], False, None, (255, 255, 0), 0.125*2, (255,255,0), 'korob' )

}

# Определяем словарь для хранения высот объектов
heights = {
    'red': 4,
    'blue': 3,
    'green': 2,
    'yellow': 1
}
# Импорт, связанный с ROS
from std_srvs.srv import Trigger

#объявление класса для работы распознавания
mark = ObjectDetection(objects, transform_method='rectify_point', dev='from_virtual_to_clover', cnt_method='default')
mark.STOP_ALL = True
mark.max_cnt = True
get_telemetry = rospy.ServiceProxy('get_telemetry', srv.GetTelemetry)
navigate = rospy.ServiceProxy('navigate', srv.Navigate)
land = rospy.ServiceProxy('land', Trigger)

# Функция перехода к определенному месту и ожидания прибытия
def navigate_wait(x=0, y=0, z=0, yaw=float('nan'), speed=0.6, frame_id='', auto_arm=False, tolerance=0.2):
    # Вызов функцию навигации, чтобы перейти к нужным координатам
    navigate(x=x, y=y, z=z, yaw=yaw, speed=speed, frame_id=frame_id, auto_arm=auto_arm)
# Постоянно проверяем, достиг ли дрон целевого местоположения
    while not rospy.is_shutdown():
        telem = get_telemetry(frame_id='navigate_target')
        if math.sqrt(telem.x ** 2 + telem.y ** 2 + telem.z ** 2) < tolerance:
            break
        rospy.sleep(0.2)

# Функция посадки дрона и ожидания его снятия с охраны
def land_wait():
    land()
    while get_telemetry().armed:
        rospy.sleep(0.2)

#Логика полета
navigate(x=0, y=0, z=2, yaw=float('nan'), speed=0.7, frame_id='body', auto_arm=True)

rospy.sleep(2)
# Переход к серии путевых точек, используя Navigation_wait с идентификатором кадра, установленным на «aruco_map».
# - Первая путевая точка: x=6, y=0,5, z=2 (yaw=-pi для взгляда в определенном направлении)
# - Вторая путевая точка: x=5,5, y=2,5, z=2,5 (yaw=-pi для взгляда в определенном направлении)
```

```
navigate_wait(x=6,y=0.5, z=2, yaw=float('nan'), frame_id='aruco_map')
navigate_wait(x=5.5,y=2.5, z=2.5, yaw=-(np.pi), frame_id='aruco_map')
# Захват изображение с основной камеры и сохраните его как «foto_all.jpg».
img = mark.bridge.imgmsg_to_cv2(rospy.wait_for_message('main_camera/image_raw', Image), 'bgr8')
cv2.imwrite('foto_all.jpg', img)

x0, y0, x1, y1, shag_y = 0, 0, 1, 2, 0.2
x0, y0, x1, y1, shag_y = 4, 1, 7, 3, 0.25
print(y1 / shag_y)

mark.STOP_ALL = False

# Генерируем список точек полета для сеточного узора внутри прямоугольника
fly_pnts = []
for i in range(0, int(y1 // shag_y)):
    if i % 2 == 0:
        fly_pnts.append([x0, y0+shag_y*i])
        fly_pnts.append([x1, y0+shag_y*i])
    else:
        fly_pnts.append([x1, y0+shag_y*i])
        fly_pnts.append([x0, y0+shag_y*i])
# Летим к каждой точке списка Fly_pnts, используя Navigation_wait
for pt in fly_pnts:
    navigate_wait(x=pt[0],y=pt[1], z=2, yaw=-(np.pi), frame_id='aruco_map')
# Потенциально остановить любые текущие задачи обработки изображений
mark.STOP_ALL = True
# Вывести содержимое mark.Points
print(mark.Points)
# Инициализируем пустой список для хранения сообщений для сервера
server_message = []
# Перебираем каждое значение в mark.Points
for value in mark.Points.values():
    msg = {}
    msg['coords'] = [value[1][0], value[1][1]]
    msg['target_height'] = heights[value[3]]
    msg['real_height'] = int(round(value[1][2]/0.25))
    if (msg['target_height'] == msg['real_height']):
        msg['match'] = 'true'
    else:
        msg['match'] = 'false'
    server_message.append(msg)
# Вывод сообщения сервера для целей отладки
pprint.pprint(server_message)
headers = {'content-type': 'application/json', "Authorization": "Bearer 2d96a545-4e3d-479d-b799-eec389463ea6"}
r = requests.post('http://65.108.156.108/buildings', json=server_message, headers=headers)
print(r.json())
xy_build = r.json()['coords']

navigate_wait(x=xy_build[0],y=xy_build[1], z=2, yaw=-(np.pi), frame_id='aruco_map')
print((2-mark.range+0.5))
navigate_wait(x=xy_build[0],y=xy_build[1], z=2-mark.range+0.5, yaw=-(np.pi), frame_id='aruco_map')
navigate_wait(x=xy_build[0],y=xy_build[1], z=2, yaw=-(np.pi), frame_id='aruco_map')

img = mark.bridge.imgmsg_to_cv2(rospy.wait_for_message('main_camera/image_raw', Image), 'bgr8')
```

```
110  cv2.imwrite('foto_build.jpg', img)
111
112  #rospy.spin()
113  navigate_wait(x=6,y=0.5, z=2, yaw=-(np.pi), frame_id='aruco_map')
114  navigate_wait(x=6,y=0.5, z=1, yaw=-(np.pi), frame_id='aruco_map')
115
116  mark.image_sub.unregister()
117  #посадка
118  land_wait()
```

Для решения задачи 1.2 разобьем процесс работы с линией на две ключевых подзадачи:

1. Следование по линии:
   - 1.1. Поиск контуров: на бинарном изображении ищутся контуры, среди которых отбираются те, чья площадь превышает минимально допустимую (`self.min_area`).
   - 1.2. Выбор главного контура: определяется контур соответствующей линии путем выбора контура, расположенного ниже всех на изображении (предполагается, что это и есть искомая линия).
   - 1.3. Вычисление прямоугольника: для выбранного контура вычисляется описанный вокруг него прямоугольник (`minAreaRect`), который представляет собой наиболее подходящую аппроксимацию линии.
   - 1.4. Анализ положения линии и корректировка движения: исходя из положения центра прямоугольника, определяется, насколько линия смещена от центра изображения, и корректируется траектория полета квадрокоптера. Это делается путем установки скорости движения вперед и скорости смещения в сторону (v_y), а также корректировки высоты (v_z), чтобы поддерживать целевую высоту полета.
2. Работа с отрезками линии:
   - 2.1. Применение цветового фильтра для выделения дорожного полотна: использование функции `cv2.inRange()` для создания бинарного изображения (маски).
   - 2.2. Очистка нижней части изображения: применение `cv2.rectangle()` для закрашивания нижней части изображения (это делается для исключения возможных участка дороги, расположенного в нижней части кадра).
   - 2.3. Поиск контуров: выделение контуров на бинарном изображении с помощью `cv2.findContours()`; отфильтровываются контуры, площадь которых меньше заданной минимальной (`self.min_area`), что позволяет исключить мелкие и несущественные объекты.
   - 2.4. Анализ наибольшего контура: определение наибольшего контура, предполагаемо соответствующего дорожному полотну, и вычисление его минимального описанного прямоугольника (`cv2.minAreaRect()`); получение вершин этого прямоугольника (`cv2.boxPoints()`).
   - 2.5. Классификация сторон прямоугольника: расчет длины сторон прямоугольника для определения, какие из них соответствуют длине дороги, а какие — ширине (это позволяет понять ориентацию дорожного полотна на изображении).
   - 2.6. Вычисление векторов преобразования: на основе полученных сторон прямоугольника вычисляются векторы, которые будут использоваться для преобразования координат точек дороги в глобальные координаты сце-

ны.

2.7. Подсчет пикселей дороги и определение сегментов: используя линейное пространство (`np.linspace`), метод проходит вдоль ширины дороги, определяя наличие дорожного полотна на основе бинарной маски (это позволяет оценить количество полос и их приблизительные размеры).

2.8. Определение глобальных координат сегментов: с помощью преобразования координат (через матрицу перехода) и функции `tf_function`, которая используется для перевода локальных координат точек в глобальные координаты сцены.

2.9. Анализ изменений в ширине дорожного полотна: если обнаруживается значительное изменение в ширине между последовательными измерениями, считается, что начался новый сегмент дороги; это изменение фиксируется и для каждого такого сегмента вычисляются его начальные и конечные точки, а также средняя ширина.

```
import rospy
from clover import srv
from sensor_msgs.msg import Image
from cv_bridge import CvBridge, CvBridgeError
from sensor_msgs.msg import Range

import cv2

from enum import Enum
from math import fmod, pi

import numpy as np

import time

class LineFollower:
    class States(Enum):
        Init = 0
        Reverse = 1
        Follow = 2
        End = 3

    def __init__(self,
                 line_threshold,
                 bridge,
                 target_height = 1.0,
                 debug_publisher = True,
                 k_velocity_y = -0.006,
                 line_velocity = 0.12,
                 k_velocity_z = 0.05):
        self.state = self.States.Init
        self.debug_publisher = debug_publisher
        self.bridge = bridge
        self.blur_kernel = (5, 5)
        self.morph_kernel = np.ones((5, 5), np.uint8)

        assert len(line_threshold) == 2
        self.threshold = line_threshold

        # коэффициент движения по оси Y за линией
        self.k_velocity_y = k_velocity_y
        # скорость движения за линией
```

```
43          self.line_velocity = line_velocity
44          # коэффициент движения по оси Z
45          self.k_velocity_z = k_velocity_z
46
47          self.y_roi = (120, 20)
48          self.min_area = 300
49
50          self.target_yaw = None
51          self.target_height = target_height
52
53          self.end_time = 3.0 # sec
54          self.last_line_seen = None
55
56          self.is_enable = False
57
58          self.height = self.target_height
59
60          self.get_telemetry = rospy.ServiceProxy('get_telemetry', srv.GetTelemetry)
61          self.navigate = rospy.ServiceProxy('navigate', srv.Navigate)
62          self.set_velocity = rospy.ServiceProxy('set_velocity', srv.SetVelocity)
63
64          self.range_sub = rospy.Subscriber('rangefinder/range', Range,
            ↪  self.range_cb)
65
66          if self.debug_publisher:
67              self.debug_pub = rospy.Publisher("/a/line_debug", Image, queue_size=1)
68
69      def range_cb(self, msg):
70          self.height = msg.range
71
72      def ang_norm(self, ang):
73          ang = fmod(fmod(ang, 2.0 * pi) + 2.0 * pi, 2.0 * pi)
74          if ang > pi:
75              ang -= 2.0 * pi
76          return ang
77
78      def enable(self):
79          self.is_enable = True
80
81      def disable(self):
82          self.is_enable = False
83
84      def update(self, image: np.ndarray, hsv: np.ndarray):
85          debug = self.on_frame(image, hsv)
86          if debug is not None:
87              self.debug_pub.publish(self.bridge.cv2_to_imgmsg(debug, "bgr8"))
88
89      # распознавание линии и повреждений
90      def on_frame(self, image: np.ndarray, hsv: np.ndarray):
91          debug = None
92          if self.debug_publisher: debug = image.copy()
93
94          if not self.is_enable:
95              return debug
96
97          if self.state == self.States.End:
98              return debug
99
100         if self.state == self.States.Reverse:
101             pose = self.get_telemetry(frame_id = 'aruco_map')
```

```
102             ang_min = self.ang_norm(self.reverse_yaw - 0.15)
103             ang_max = self.ang_norm(self.reverse_yaw + 0.15)
104 
105             if pose.yaw >= ang_min and pose.yaw <= ang_max:
106                 self.state = self.States.Follow
107             else:
108                 return debug
109 
110         height, width, _ = image.shape
111 
112         # бинаризуем изображение из пространства HSV
113         # в этом пространстве легче выделить желтый цвет
114         hsv_blur = cv2.GaussianBlur(hsv, self.blur_kernel, 0)
115         binary = cv2.inRange(hsv, self.threshold[0], self.threshold[1])
116 
117         binary = binary[(height // 2) - self.y_roi[0]:(height // 2) +
            ↪  self.y_roi[1], :]
118         binary = cv2.erode(binary, self.morph_kernel)
119         binary = cv2.dilate(binary, self.morph_kernel)
120 
121         if self.debug_publisher:
122             debug = cv2.rectangle(debug, (0, (height // 2) - self.y_roi[0]),
                ↪  (width, (height // 2) + self.y_roi[1]), (0, 255, 0), 2)
123 
124         # ищем контуры линии
125         contours, _ = cv2.findContours(binary, cv2.RETR_TREE,
            ↪  cv2.CHAIN_APPROX_SIMPLE)
126         contours = list(filter(lambda c: cv2.contourArea(c) > self.min_area,
            ↪  contours))
127 
128         major_contour = None
129         if len(contours) != 0: major_contour = min(contours, key = lambda c:
            ↪  np.int0(cv2.boxPoints(cv2.minAreaRect(c)))[0][1])
130 
131         if major_contour is not None:
132             rect = cv2.minAreaRect(major_contour)
133             (x_min, y_min), (w_min, h_min), _ = rect
134 
135             if self.debug_publisher:
136                 box = cv2.boxPoints(rect)
137                 box = np.int0(box)
138 
139                 box = [[p[0], p[1] + (height // 2) - self.y_roi[0]] for p in box]
140                 box = np.array(box)
141                 debug = cv2.drawContours(debug, [box], 0, (255, 120, 0), 2)
142                 debug = cv2.circle(debug, (int(x_min), int(y_min + (height // 2) -
                    ↪  self.y_roi[0])), 5, (255, 120, 0), -1)
143 
144             y_min += (height // 2) - self.y_roi[0]
145             thr_low = (height // 2) - 15
146 
147             if y_min >= thr_low and self.state == self.States.Init:
148                 pose = self.get_telemetry(frame_id = 'aruco_map')
149                 self.target_yaw = self.ang_norm(pose.yaw + pi)
150 
151                 print(f'Reverse clover to follow the line...')
152 
153                 self.navigate(
154                     x = pose.x,
155                     y = pose.y,
```

```
156                     z = pose.z,
157                     yaw = self.target_yaw,
158                     frame_id='aruco_map'
159                 )
160                 self.target_yaw = need_yaw
161                 self.state = self.States.Reverse
162 
163             else:
164                 self.state = self.States.Follow
165                 error = x_min - (width / 2)
166 
167                 self.set_velocity(
168                     vx = self.line_velocity,
169                     vy = error * self.k_velocity_y,
170                     vz = (self.target_height - self.height) * self.k_velocity_z,
171                     yaw = float('nan'),
172                     frame_id = 'body'
173                 )
174 
175             self.last_line_seen = None
176 
177         else:
178             now = time.time()
179             if self.last_line_seen is None:
180                 self.last_line_seen = now
181             elif (now - self.last_line_seen) >= self.end_time:
182                 self.state = self.States.End
183                 self.set_velocity(
184                     vx = 0.0,
185                     vy = 0.0,
186                     vz = 0.0,
187                     yaw = float('nan'),
188                     frame_id = 'body'
189                 )
190 
191         return debug
192 
193 import cv2
194 import numpy as np
195 import tf
196 import tf2_ros
197 import rospy
198 import tf2_geometry_msgs
199 from geometry_msgs.msg import Point32, Vector3Stamped, Vector3, Point
200 from visualization_msgs.msg import Marker, MarkerArray
201 
202 from utils import *
203 
204 from typing import Tuple, Optional
205 
206 import math
207 
208 class LineProcessor:
209     def __init__(self,
210                  hsv_threshold,
211                  cm: np.ndarray,
212                  dc: np.ndarray,
213                  tf_buffer = None,
214                  step_x: int = 1,
215                  threshold: float = 10,
```

```
216                 min_segment_length = 20):
217         self.hsv_threshold = hsv_threshold
218         self.step_x = step_x
219         self.threshold = threshold
220         self.min_segment_length = min_segment_length
221 
222         self.min_area = 200
223 
224         self.cm = cm
225         self.dc = dc
226         self.tf_buffer = tf_buffer
227 
228         self.is_enable = False
229 
230         self.width_array = []
231         self.start = None
232         self.segments = []
233         self.max_width = 0
234         self.all_xs = []
235 
236         self.road_pub = rospy.Publisher("/a/road_viz", MarkerArray, queue_size=1)
237 
238     def tf_function(self, vec: np.ndarray) -> np.ndarray:
239         vec[0] = max(min(vec[0], 319), 0.0)
240         vec[0] = max(min(vec[0], 239), 0.0)
241 
242         centers = [vec]
243         centers = np.array(centers).astype(np.float64)
244         pnt_img_undist = cv2.undistortPoints(centers.reshape(-1, 1, 2), self.cm,
            ↪  self.dc, None, None).reshape(-1, 2).T
245         ray_v = np.ones((3, pnt_img_undist.shape[1]))
246         ray_v[:2, :] = pnt_img_undist
247         ray_v /= np.linalg.norm(ray_v, axis=0)
248 
249         if self.tf_buffer is not None:
250             try:
251                 transform = self.tf_buffer.lookup_transform("aruco_map",
                    ↪  "main_camera_optical", rospy.Time())
252             except tf2_ros.ConnectivityException:
253                 print("LookupException")
254                 return None
255             except tf2_ros.LookupException:
256                 print("LookupException")
257                 return None
258 
259             t_wb = np.array([transform.transform.translation.x,
                ↪  transform.transform.translation.y,
                ↪  transform.transform.translation.z])
260 
261             ray_v = np.array([unpack_vec( tf2_geometry_msgs.do_transform_vector3(
                ↪  Vector3Stamped(vector=Vector3(v[0], v[1], v[2])), transform)) for
                ↪  v in ray_v.T])
262             ray_o = t_wb
263 
264             pnts = [intersect_ray_plane(v, ray_o) for v in ray_v]
265             if pnts[0] is not None:
266                 return pnts[0][:2]
267 
268         return None
269 
```

```
270     def publish_road(self):
271         segments = self.calculate_segments()
272         if segments is None or len(segments) <= 0:
273             return
274 
275         result = []
276 
277         y = (segments[0][1][1] + segments[-1][1][1]) / 2.0
278 
279         prev_end = None
280         for idx, (start, end, _, width) in enumerate(segments):
281             marker = Marker()
282             marker.header.frame_id = "aruco_map"
283             marker.header.stamp = rospy.Time.now()
284             marker.ns = "color_markers"
285             marker.id = idx
286             marker.type =  Marker.CUBE
287             marker.action = Marker.ADD
288 
289             if prev_end is not None:
290                 start = prev_end
291             prev_end = end
292 
293             w = width * 0.1
294 
295             center_pos = [(start[0] + end[0]) / 2.0, y]
296             length = math.sqrt(pow(start[0] - end[0], 2))
297 
298             # Позиция и ориентация
299             marker.pose.position.x = center_pos[0]
300             marker.pose.position.y = center_pos[1]
301             marker.pose.position.z = 0.04
302             marker.pose.orientation.x = 0
303             marker.pose.orientation.y = 0
304             marker.pose.orientation.z = 0
305             marker.pose.orientation.w = 1
306 
307             # Масштаб
308             marker.scale.x = length
309             marker.scale.y = w
310             marker.scale.z = 0.04
311 
312             # Цвет
313             marker.color.a = 1.0
314 
315             marker.color.r = 0.7
316             marker.color.g = 0.5
317             marker.color.b = 0.0
318 
319             result.append(marker)
320 
321         # Публикуем маркеры
322         self.road_pub.publish(MarkerArray(markers=result))
323 
324     def flow(self, image: np.ndarray, hsv: np.ndarray):
325         self.publish_road()
326 
327         if not self.is_enable:
328             return
329 
```

```
330             self.realtime_line_segmentation(tf_function=self.tf_function, image=image,
                ↪  hsv=hsv)
331
332         def enable(self):
333             self.is_enable = True
334
335         def disable(self):
336             self.is_enable = False
337
338         def dist(self, a, b):
339             return math.sqrt(pow(a[0] - b[0], 2) + pow(a[1] - b[1], 2))
340
341         def realtime_line_segmentation(self, tf_function, image: np.ndarray, hsv:
                ↪  np.ndarray):
342             '''Определение сегментов дороги'''
343             bin = cv2.inRange(hsv, self.hsv_threshold[0], self.hsv_threshold[1])
344             bin = cv2.rectangle(bin, (0, 120+30), (320, 240), 0, -1)
345
346             # нахождение контуров
347             cnts, _ = cv2.findContours(bin, cv2.RETR_EXTERNAL,
                    ↪  cv2.CHAIN_APPROX_SIMPLE)
348             cnts = [c for c in cnts if cv2.contourArea(c) > self.min_area]
349
350             if len(cnts) == 0:
351                 return None
352
353             cnt = max(cnts, key=cv2.contourArea)
354
355             # нахождение минимального описанного прямоугольника
356             rect = cv2.minAreaRect(cnt)
357             box = cv2.boxPoints(rect)
358             box = np.int0(box)
359
360             # классификация отрезков прямоугольника
361             if self.dist(box[0], box[1]) > self.dist(box[1], box[2]):
362                 walls = ((box[0], box[1]), (box[3], box[2]))
363                 between = ((box[1], box[2]), (box[3], box[0]))
364             else:
365                 walls = ((box[1], box[2]), (box[0], box[3]))
366                 between = ((box[2], box[3]), (box[0], box[1]))
367
368             corner = walls[0][0]
369
370             wall_height = self.dist(*walls[0])
371             wall_width = self.dist(*between[0])
372
373             # нахождение векторов для матрицы перехода
374             vec_y = np.array([
375                 walls[0][1][0] - walls[0][0][0],
376                 walls[0][1][1] - walls[0][0][1]
377             ]) / wall_height
378
379             vec_x = np.array([
380                 between[0][1][0] - between[0][0][0],
381                 between[0][1][1] - between[0][0][1]
382             ]) / wall_width
383
384             origin = np.array([
385                 [1, 0],
386                 [0, 1]
```

```
387         ])
388
389         dest = np.array([
390             vec_x,
391             vec_y
392         ])
393
394         l = np.vstack([origin[0].T, origin[1].T])
395         r = np.vstack([dest[0].T, dest[1].T])
396
397         # нахождение матрицы перехода
398         T = np.linalg.solve(l, r)
399         T_inv = np.linalg.inv(T)
400
401         space_x = np.linspace(0, int(wall_width), int(wall_width) // self.step_x)
402
403         c = 0
404         xs = []
405         for x in space_x:
406             # Переход от локальных координат дороги к глобальным координатам кадра
407             local_vec = np.array([x, 0]).T
408             global_vec = T_inv @ local_vec
409
410             coordinates = global_vec.T + corner
411
412             # Подсчет пикселей дороги
413             point = bin[np.clip(int(coordinates[1]), 0, hsv.shape[0] - 1),
                    ↪  np.clip(int(coordinates[0]), 0, hsv.shape[1] - 1)]
414             if point > 127:
415                 c += 1
416                 xs.append(x)
417
418         self.all_xs.append(np.array(xs).mean() if len(xs) > 0 else 0)
419         print(c)
420         if self.start is None:
421             start_vec = np.array([0, wall_height])
422             frame_start_vec = (T_inv @ start_vec.T).T + corner
423             print(frame_start_vec)
424             map_start_vec = tf_function(frame_start_vec)
425
426             self.start = map_start_vec
427
428         elif abs(c - self.width_array[-1]) > self.threshold:
429             end_vec = np.array([0, 0])
430
431             frame_end_vec = (T_inv @ end_vec.T).T + corner
432             print(frame_end_vec)
433             map_end_vec = tf_function(frame_end_vec)
434
435             is_allowed = True
436             if len(self.segments) > 0:
437                 dst = self.dist(self.segments[-1][1], map_end_vec)
438                 is_allowed = (dst >= 0.5)
439             else:
440                 dst = self.dist(self.start, map_end_vec)
441                 is_allowed = (dst >= 0.5)
442
443             if is_allowed:
444                 segment = (self.start, map_end_vec,
                        ↪  np.array(self.width_array).mean())
```

```
445
446                 self.segments.append(segment)
447                 self.width_array = []
448
449         self.width_array.append(c * self.step_x)
450         if c > self.max_width:
451             self.max_width = c
452
453     def calculate_segments(self):
454         if len(self.segments) <= 0:
455             return None
456
457         min_width = min(s[2] for s in self.segments) + pow(10, -3)
458
459         return [(*s, round(s[2] / min_width)) for s in self.segments]
460
461     def calc_inverse_transform_matrixes(self, segments):
462         invTs = []
463         for s in segments:
464             dest = np.array([
465                 np.array([s[1][0] - s[0][0], s[1][1] - s[0][1]]) / self.dist(s[1],
                    ↪  s[0]),
466                 np.array([-(s[1][1] - s[0][1]), s[1][0] - s[0][0]]) /
                    ↪  self.dist(s[1], s[0]),
467             ])
468             origin = np.array([
469                 [1, 0],
470                 [0, 1]
471             ])
472             l = np.vstack([origin[0].T, origin[1].T])
473             r = np.vstack([dest[0].T, dest[1].T])
474
475             T = np.linalg.solve(l, r)
476
477             invTs.append(np.linalg.inv(T) if invert else T)
478
479         return invTs
480
481     def is_worker_working(self, x, y, distance, segments, invTs):
482         """Аргументы:
483             x (int): координата x (карта)
484             y (int): координата y (карта)
485             distance (int): расстояние, при котором работник считается работающем
                ↪  (карта)
486             segments (List[...]): список подотрезков
487             invTs (List[T]) список обратных матриц перехода для кадого отрезка
488
489             Возвращает:
490             bool, int - 1) работает ли 2) подотрезок, над которым работает
491         """
492
493         for s, invT in zip(segments, invTs):
494             # переход к базису сегмента
495             arr = np.array([x, y]) - s[0]
496             vec = arr.T
497             local_vec = (invT @ vec).T[::-1]
498
499             # проверка на работу
500             print(local_vec, s[2], distance)
501             if local_vec[0] >= -distance and local_vec[0] <= s[2] + distance and 0
                ↪  <= local_vec[1] <= self.dist(s[1], s[0]):
```

```
502                 return True
503         return False
```

Для решения задачи 1.3 контроля доступности строительной спецтехники с использованием дрона (квадрокоптера) предлагается использование методов машинного обучения или глубокого обучения, например, сверточные нейронные сети (CNN) для классификации типов спецтехники. Наиболее простой в использовании является модель YOLO (https://github.com/ultralytics/ultralytics). К сожалению, она велика для работы непосредственно на дроне, поэтому предлагается использовать инструменты ROS (топики и сервисы) для передачи изображения на мощный компьютер, где будет производиться запуск нейросети.

Инженерное задание

Перед началом разработки устройства необходимо ознакомиться с техническим заданием и обратить внимание на имеющиеся ограничения:

- Временные рамки на проектирование, изготовление и доработку устройства.
- Электронную компонентную базу.
- Работу с 3D-принтером:
  - для минимизации использования поддержек необходимо выбрать правильное направление и ориентацию деталей, а также можно использовать геометрические формы, требующие минимальное число поддержек;
  - во избежание деформации и провисания пластика во время печати необходимо обеспечить достаточную толщину стенок деталей и их правильное расположение;
  - для гарантированного соединения деталей между собой необходимо расположить детали на печатном столе так, чтобы поверхности и отверстия, с помощью которых детали соединяются, имели максимальную точность;
  - для минимизации времени печати и использования материала необходимо подобрать оптимальное расположение деталей на печатной платформе (закладывать минимальное расстояние между деталями; сторону детали, имеющую максимальный линейный размер, располагать горизонтально на поверхности стола и т. п.);
  - для обеспечения правильного прижима первого слоя путем последовательной проверки зазора между соплом и столом по нескольким противолежащим точкам в углах стола перед началом работы с принтером необходимо произвести калибровку печатной платформы;
  - для настройки температуры перед запуском основной печати необходимо произвести печать тестовой модели из необходимого пластика;
  - использовать подходящие настройки скорости печати для конкретных деталей; чтобы определить оптимальную скорость перемещений на холостом ходу для имеющегося принтера необходимо распечатать тестовую модель при различных скоростях движения, начиная со 100 мм/с и изменяя ее с приростом 5 мм/с, скорость печати можно увеличивать, если качество поверхности приемлемое, и уменьшать, если качество 3D-печати ухудшается (необходимо обращать внимание на такие дефекты, как несовпадение слоев).
- На работу с лазерным станком с ЧПУ для резки фанеры:

– при проектировании частей устройства и необходимо учитывать количество материала;
– для минимизации использования материала необходимо подобрать оптимальное расположение деталей на каждом листе:
– детали размещаются на листе с зазором не менее 1 мм.

При сборке, тестировании и эксплуатации устройства необходимо соблюдать технику безопасности.

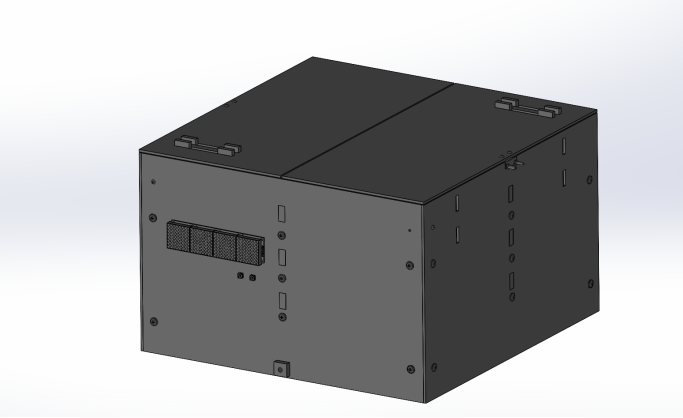


Рис. VI.2.6. Вид в изометрии

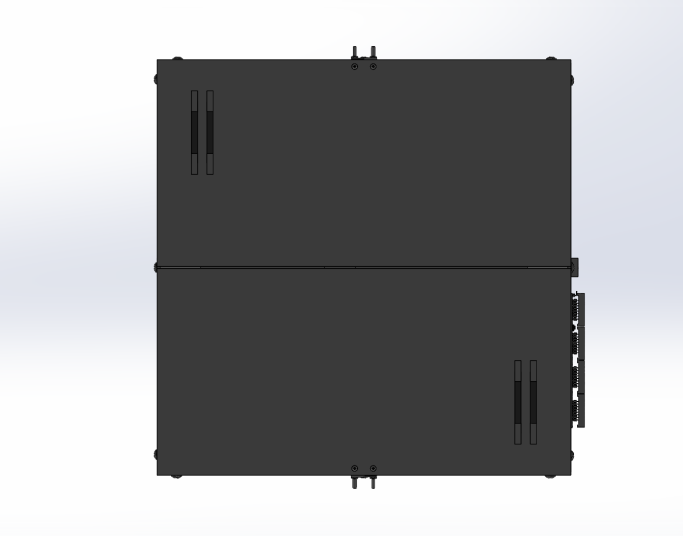


Рис. VI.2.7. Вид сверху

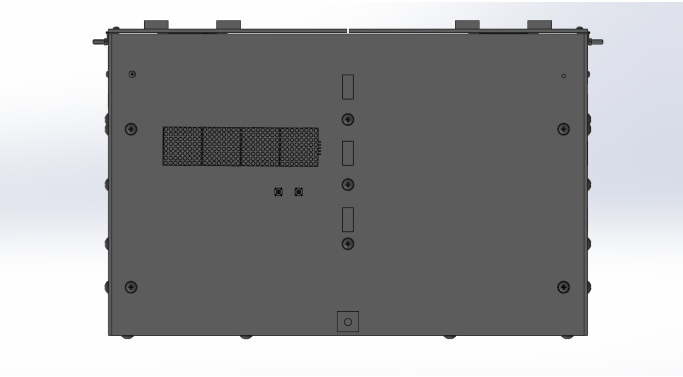


Рис. VI.2.8. Вид сбоку

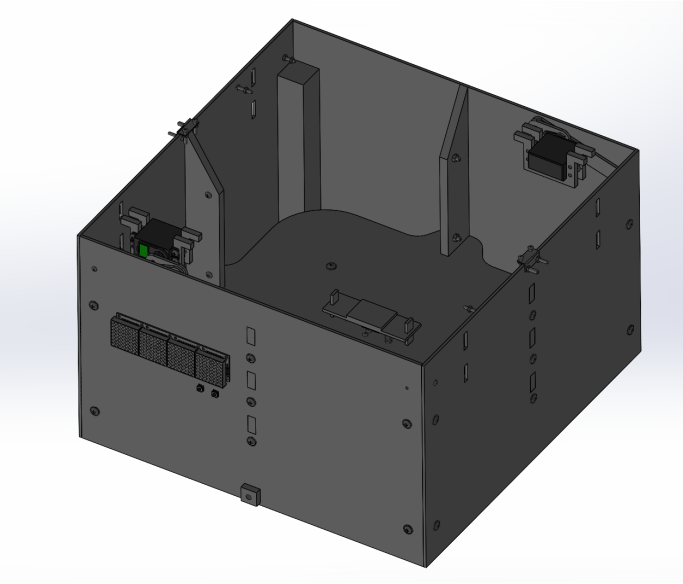


Рис. VI.2.9. Устройство без крышки

Программный код для эксплуатации устройства (Arduino)

```
#include <GyverMAX7219.h>
#include "SoftwareSerial.h"
#include <Servo.h>

//инициализация серв, матрицы и блютуза
Servo S1,S2;
MAX7219 < 1, 1, 4, 5, 6 > mtrx;
  SoftwareSerial BTserial(8, 9);
bool is_open=0, izm=0;
```

```
unsigned long time=0;
int is_pow=1;

void setup() {
  // put your setup code here, to run once:
  mtrx.begin();
  mtrx.print("Charg");
  mtrx.update();
  Serial.begin(9600);
  BTserial.begin(9600);
  pinMode(10, INPUT);
  pinMode(11, INPUT);
  S1.attach(A4);
  S2.attach(A5);
  is_open=0;
}

void loop() {
  if (is_open==0 && izm==1)//закрытие короба
  {
    mtrx.clear();
    mtrx.print("Close");
    mtrx.update();
    S1.write(0);
    S2.write(0);
    delay(1000);
    izm=0;
  }
  if (is_open==1 && izm==1)//открытие короба
  {
    mtrx.clear();
    mtrx.print("Open");
    mtrx.update();
    S1.write(100);
    S2.write(100);
    delay(1000);
    izm=0;
  }
  if (izm==0)//Сообщений открытия и закрытия короба
  {
     if (BTserial.available())
    {
      int p=BTserial.read();
      if(p==99)
      {
        if (is_open==1) izm=1;
        is_open=0;
      }
      if(p==11)
      {
        if (is_open==0) izm=1;
        is_open=1;
      }
      if ( digitalRead(10)==HIGH)
      {
        is_open=!is_open;
        izm=1;
      }
    }
  }
```

```
70    if ( digitalRead(11)==LOW )//Детектим взлет
71    {
72          is_pow=0;
73          time=0;
74    }
75    if ( digitalRead(11)==LOW && izm==0)//никого нет
76    {
77            mtrx.clear();
78            mtrx.print("NoIn");
79            mtrx.update();
80    }
81    if (is_pow==0 && izm==0)//Логика подключения питания
82    {
83      if ( digitalRead(11)==HIGH && time==0)
84      {
85            mtrx.clear();
86            mtrx.print("InNoP");
87            mtrx.update();
88            is_pow=0;
89            time=millis();
90      }
91      if ( digitalRead(11)==HIGH && time!=0 && millis()-time>5000)
92      {
93            mtrx.clear();
94            mtrx.print("Charg");
95            mtrx.update();
96            is_pow=1;
97            time=0;
98      }
99    }
100 }
```

Материалы для подготовки

- Статьи, видеолекции по тематике профиля на сайте документации платформы «Клевер»: https://clover.coex.tech/.
- Видеокурс по сборке, настройке и программированию квадрокоптера: https://www.youtube.com/watch?v=GJvucA2igME&list=PLpWUdRMCiBWaSy0ZvTUttDJkVnUF3p2TD.
- Среда симуляции платформы «Клевер»: https://clover.coex.tech/ru/simulation.html.
- Работа с OpenCV при помощи Python: https://github.com/sfalexrog/coex_kb/blob/master/kb014_opencv_python.md.
- Программирование на Python: https://stepik.org/course/67/promo.
- Введение в ROS: https://stepik.org/course/3222/promo.
- Введение в Git: https://ru.hexlet.io/courses/intro_to_git.

Критерии определения победителей и призеров

Первый отборочный этап

В первом отборочном этапе участники решали задачи предметного тура по двум предметам: информатике и физике 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 не менее 30 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 85 баллов, независимо от уровня.

Заключительный этап

Индивидуальный предметный тур

- информатика — максимально возможный балл за все задачи — 100 баллов;
- физика — максимально возможный балл за все задачи — 100 баллов.

Командный инженерный тур

Команды заключительного этапа получали за командный инженерный тур от 0 до 100 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 предметам (информатика, физика) с весом $K_1 = 0,2$ каждый предмет и баллы за командное решение задач инженерного тура с весом $K_2 = 0,6$.

Итоговый балл определяется по формуле:

$$S = K_1 \cdot (S_1 + S_2) + K_2 \cdot S_3,$$

где S_1 — балл первой части заключительного этапа по информатике (предметный тур) в стобалльной системе ($S_{1\ \text{макс}} = 100$);

S_2 — балл первой части заключительного этапа по физике (предметный тур) в стобалльной системе ($S_{2\ \text{макс}} = 100$);

S_3 — итоговый балл инженерного командного тура в стобалльной системе ($S_{3\ \text{макс}} = 100$).

Итого максимально возможный индивидуальный балл участника заключительного этапа = 100 баллов.

Критерий определения победителей и призеров

Чтобы определить победителей и призеров (независимо от класса) на основе индивидуальных результатов участников, был сформирован общий рейтинг всех участников заключительного этапа. С начала рейтинга были выбраны 2 победителя и 6 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	37,98 и выше
Призеры	От 33,60 до 34,75

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.

Важная задача наставника — превратить неудачу в инструмент будущего успеха. Для этого необходимо вместе с учениками наметить план развития компетенций и подготовки к будущему сезону Олимпиады. Подробные материалы о проведении рефлексии представлены в курсе «Наставник НТО»: https://academy.sk.ru/events/310.

Наставнику важно проинформировать руководство образовательного учреждения, если его учащиеся стали финалистами, призерами и победителями. Публичное признание высоких результатов дополнительно повышает мотивацию.

В процессе рефлексии с учениками, не ставшими призерами или победителями, рекомендуется уделить особое внимание особенностям командной работы: распределению ролей, планированию работы, возникающим проблемам. Для этого могут использоваться опросники для самооценки собственной работы и взаимной оценки участниками других членов команды (P2P). Такие опросники 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.

Еще одним направлением работы наставника после НТО может стать создание кружка по направлению профилей или по формированию необходимых компетенций: программирование, электроника, робототехника, 3D-моделирование и т. п. Формат подобного кружка может быть различным: короткие модули, дополнительные курсы, факультативы, группы дополнительного образования. Для создания кружков можно воспользоваться образовательными программами, опубликованными на сайте НТО: https://ntcontest.ru/mentors/education-programs/.

Важным фактором успешного участия в следующих сезонах НТО может стать поддержка родителей учеников. Знакомство с родителями помогает наставнику продемонстрировать им 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».