МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное бюджетное образовательное
учреждение высшего образования
«ДАГЕСТАНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»
Факультет информатики и информационных технологий

МЕТОДИЧЕСКИЕ УКАЗАНИЯ

по организации и проведению учебной (ознакомительной) практики

для студентов направления подготовки
09.03.04 Программная инженерия

Профиль подготовки:
Разработка программно-информационных систем,
Программное обеспечение вычислительной техники и автоматизированных систем
Разработка программного обеспечения и приложений искусственного интеллекта

Махачкала 2026

Содержание

I.Введение 3
1.1.Цель учебной ознакомительной практики 3
1.2. Место проведения практики 4
1.3.Руководство практикой 4
1.4.Права и обязанности студентов в период практики 5
1.6.Требования к оформлению отчета по учебной ознакомительной практике 6
1.7.Подготовка презентационного материала 10
1.8.Аттестация по итогам практики 11
II. Примерная программа учебной практики 12
III. Примерный перечень контрольных вопросов при приеме материалов учебной практики 30
Приложение 1 32

I. Введение

Учебные практики проводятся в соответствии с программами для студентов очной формы обучения по направлению подготовки 09.03.04 программная инженерия, профили – «Разработка программно-информационных систем», «Программное обеспечение вычислительной техники и автоматизированных систем», «Разработка программного обеспечения и приложений искусственного интеллекта».

Методические указания содержат требования к организации и порядку проведения учебной практики студентов, её основное содержание.

В методических указаниях сформулированы права и обязанности практикантов, рассмотрен перечень вопросов для изучения студентами, а также требования к формированию и защите отчета по практике.

Методические указания предназначены для студентов указанного направления подготовки, профессорско-преподавательского состава выпускающих кафедр, руководителей практики от кафедры.

1.1. Цель учебной ознакомительной практики

Целями учебной (ознакомительной) практики по направлению подготовки 09.03.04 «Программная инженерия», направленность (профили) «Разработка программно-информационных систем», «Программное обеспечение вычислительной техники и автоматизированных систем», «Разработка программного обеспечения и приложений искусственного интеллекта» являются закрепление и углубление теоретической подготовки обучающегося и приобретение ими практических навыков и компетенций в сфере профессиональной деятельности.

Кроме того, целями учебной ознакомительной практики по направлению подготовки 09.03.04 Программная инженерия являются также обеспечение непрерывности и последовательности овладения студентами умениями и навыками профессиональной деятельности, закрепление и углубление универсальных и общепрофессиональных компетенций в сфере программной инженерии, сформированных у студентов в процессе обучения и в результате самостоятельной работы по исследованию предметной области.

Основные задачи учебной практики:

Задачами учебной ознакомительной практики по направлению подготовки 09.03.04 «Программная инженерия», направленность (профили)-«Разработка программно-информационных систем», «Программное обеспечение вычислительной техники и автоматизированных систем», «Разработка программного обеспечения и приложений искусственного интеллекта».:

– углубление знаний по дисциплинам, полученным за время обучения на первом курсе, таких как «Программирование на С++», «Технологии и методы программирования», «Основы программной инженерии» и др.

– формирование навыков подготовки и систематизации

необходимых материалов и научно-технической информации для выполнения задания;

– создание условий для практического применения знаний в области общепрофессиональных, специализированных компьютерных и математических дисциплин;

– формирование и совершенствование базовых профессиональных навыков и умений в области применения современных информационных технологий;

– формирование информационной компетентности с целью успешной работы в профессиональной сфере деятельности;

– анализ результатов проведенного исследования, подготовка и составление отчетов;

– приобретение навыков создания отчетов, в том числе и научно-технических, обеспечение успеха дальнейшей профессиональной карьеры.

1.2. Место проведения практики

Учебная (ознакомительная) практика проводится на 1 курсе во 2 семестре, срок проведения – 8 дней.

Местом проведения практики является факультет информатики и информационных технологий, кафедра информационных систем и технологий программирования, а также иные подразделения университета.

План-график прохождения практики разрабатывается руководителями от университета на основе баланса времени и с учетом особенностей базы практики и ее вида.

При наличии обучающихся по данной образовательной программе с ограниченными возможностями здоровья и инвалидов выбор мест прохождения практики осуществляется с учетом состояния здоровья и требований их доступности для данной категории.

1.3. Руководство практикой

Руководство практикой осуществляют факультет информатики и информационных технологий и кафедра информационных систем и технологий программирования (ИСиТП). Кафедра отвечает за выполнение программы и качество проведения учебной практики.

Руководитель практики от кафедры:

– осуществляет методическое руководство учебной (ознакомительной) практикой студентов;

– обеспечивает выполнение подготовительной и текущей работы по организации практики;

– разрабатывает программу учебной практики, индивидуальные задания студентам с указанием основной и дополнительной литературы;

– готовит и проводит совместно с деканатом организационные собрания студентов перед началом практики;

– при необходимости проводит консультации студентов в

период учебной практики и перед защитой отчетов;

– совместно с факультетом в назначенный срок после окончания практики организует и принимает защиту отчетов по практике;

– представляет на кафедру итоги учебной практики;

– подводит итоги практики студентов кафедры;

– организует на кафедре хранение отчетов студентов по практике в течение установленного срока.

1.4. Права и обязанности студентов в период практики

Добавлено примечание ([P1]): Здесь необходимо ссылаться на Положение дГУ о практике https://ndoc.dgu.ru/PDFF/Polojenie_o_praktike_obuch_osvaiv_OOP_VO_2018.pdf
Здесь определены права и обязанности студента

С целью наилучшей подготовки к практике студент обязан ознакомиться с программой и содержанием предстоящих работ, собрать и изучить рекомендованную литературу.

В процессе прохождения практики студенты должны выполнять задания, предусмотренные планом, правила внутреннего распорядка организации, следовать указаниям руководителей практики, собрать материал для отчёта и оформить его. Порядок сбора и обработки материалов согласовывается с руководителем практики.

Перед завершением практики студент должен обсудить результаты своей работы над отчётом с руководителей практики.

Студент при прохождении практики обязан:

- в случае болезни обязательно поставить в известность руководителя практики и предоставить медицинскую справку;
- в течении последней недели практики представить отчёт руководителю практики для проверки и отзыва;
- в последние два дня практики представить руководителю практики от Университета отчёт и после проверки защитить его.

В случае затруднений с выполнением задания практики, связанных с характером работы, студент должен сообщить об этом руководителю практики от Университета.

1.5. Структура и содержание учебной практики»

При формировании отчёта по практике следует соблюдать определённую последовательность расположения материала:

– Титульный лист (см. Приложение 1)

– Содержание (перечень разделов и подразделов отчёта).

– Введение (цель учебной практики, особенности ее прохождения).

– Перечень, особенности и описание практических задач, выполненных студентом во время практики (пункты 2.1.1. и 2.1.2.)

– Выполнение индивидуальной работы (написание реферата).

– Заключение (итоги практики, выводы и предложения студента по практике).

– Список использованной литературы, технической и нормативной документации.

Руководитель практики должен проверить содержание отчета по

практике с целью проверки грамотности его выполнения, допустимости разглашения отдельной информации.

1.6. Требования к оформлению отчета по учебной ознакомительной практике

При оформлении отчёта по практике необходимо соблюдать следующие требования:

– отчёт оформляется на одной стороне листа А4 (210х297 мм), белой бумаги с соблюдением следующих размеров полей: верхнее и нижнее - по 20 мм, левое поле – 30 мм, правое поле – 10 мм;

– таблицы и иллюстрации допускается представлять на листах формата А3 (297х420 мм);

– текст готовится к распечатке на одной стороне листа на компьютере;

– шрифт «Times New Roman» 14 pt, межстрочный интервал – полуторный;

– наименования разделов располагаются по центру строки без точки в конце и набирается прописными буквами без подчёркиваний;

– заголовки подразделов набираются с абзацного отступа строчными буквами (кроме первой) без точки в конце;

– после номера раздела и подраздела в тексте точка не ставится;

– не допускаются переносы в словах разделов и подразделов;

– расстояние между заголовками и текстом должен составлять два интервала;

– страницы следует нумеровать цифрами;

– номер проставляется внизу страницы по центру в пределах рабочего поля; на титульном листе, номер страницы не ставится. Формулы следует выделять в отдельную строку, отделяя её от текста свободной строкой (сверху и снизу). Оформление формул производится при помощи любого редактора формул. Номера формул указываются арабскими цифрами в круглых скобках и имеют сквозную нумерацию;

– при оформлении таблиц и иллюстраций следует руководствоваться следующими правилами.

Все иллюстрации (кроме таблиц) в тексте работы называются **рисунками,** их представляют сразу же после упоминания или на следующей странице.

Выполняют рисунки черными чернилами (пастой) или тушью, если они не могут быть представлены с помощью компьютерной графики или в виде фотографий.

Под изображением рисунка приводится слово «Рисунок» с указанием после него через интервал цифры (номера) далее (через интервал) указывается название рисунка с заглавной буквы. При этом номер и название располагаются на одной и той же строке.

Перед рисунком в тексте оставляют одну свободную строку. Рисунок должен быть расположен по центру страницы, в абзаце, содержащем рисунок, не должно быть заказано отступа или выступа первой строки. Выравнивание названия рисунка производится по левому краю. Нумерация рисунков в тексте должна быть сквозной.

Пример оформления рисунка:



Рис. 1. Функциональная схема котлоагрегата на экране монитора

Уравнения и формулы следует выделять из текста в отдельную строку. В формулах в качестве символов следует применять обозначения, установленные соответствующими государственными стандартами, и отраженные в формате формул. Образец оформления формулы представлен на рисунке 2.

Образец оформления формулы представлен на рисунке 2.

$$S = P \times (1 + \frac{I \times j}{100 \times K})^n$$

Рис. 2. Образец оформления формулы

где S – общая сумма («тело» вклада + проценты), причитающаяся к возврату вкладчику по истечении срока действия вклада;

P – первоначальная величина вклада;

n - общее количество операций по капитализации процентов за весь срок привлечения денежных средств (в данном случае оно соответствует количеству лет);

I – годовая процентная ставка.

K – количество дней в текущем году,

j – количество дней в периоде, по итогам которого банком производится капитализация начисленных процентов (остальные обозначения – как и в предыдущей формуле).

Цифровой материал рекомендуется помещать в тексте в виде **таблиц.** Каждая из них должна иметь заголовок, точно отражающий её содержание и расположенный по центру. Помещается заголовок непосредственно над таблицей по центру. После слова таблица через интервал цифры (номера) далее (через интервал) указывается название таблицы с заглавной буквы. При этом номер и название располагаются на одной и той же строке.

Пример оформления таблицы:

Таблица 1

Доля государственного долга в ВВП в развитых странах G-20
в 2017–2025 гг., %

Страны	2017 (факт)	2019 (факт)	2020 (оценка)	2021 (прогноз)	2025 (прогноз)
G-20 – развитые экономики	77,9	96,9	104,4	108,8	117,1
Япония	187,7	217,7	227,1	234,6	250,0
Италия	103,4	115,8	118,6	120,5	124,7
США	62,1	83,2	92,6	97,4	109,7
Франция	63,8	77,4	84,2	88,6	94,8
Великобритания	44,1	68,2	78,2	84,9	90,6

Источник: IMF World Economic Outlook. 2020. April. [Электронный ресурс]. – Электр. данн. – Режим доступа: http://www.imf.org/external/pubs/ft/weo/

Располагаются таблицы в тексте сразу после первой ссылки на них или на следующей (за первой ссылкой) странице. Оформляются таблицы так, как показано на примере приведенной Таблицы 1. При переносе части таблицы на другую страницу название помещают только над первой её частью, а над второй справа пишут «Продолжение таблицы» с указанием её номера. Допускается располагать таблицу вдоль длинной стороны листа. Графу «Номер по порядку» в таблице не делают.

Если повторяющийся в разных строках таблицы текст состоит из одного слова, то после его первого написания далее допускается заменять его кавычками. Если текст состоит из двух или более слов, то его можно заменять словами «то же», а далее заменять кавычками. Ставить кавычки вместо повторяющихся цифр не допускается.

Если таблица заимствована или рассчитана по данным статистического ежегодника или другого литературного источника, надо обязательно делать ссылку на первоисточник (см. пример Таблица 1).

Если в тексте необходимо сослаться на таблицу, то следует указать номер таблицы и страницу, на которой она расположена.

Приложения помещают за списком литературы в порядке упоминания их в основном тексте. Каждое приложение следует начинать с новой страницы с указанием наверху справа слова «Приложение».

Приложения обозначаются заглавными буквами русского алфавита, начиная с А. После слова «Приложение» должна следовать буква, обозначающая его последовательность.

Разработанные обучающимся документы, имеющие самостоятельное значение и подлежащие исполнению в соответствии с заданием и нормативно-закрепленными требованиями, помещают в прилагаемую к текстовой части папку с соответствующей описью. При наличии аудио - или видеоматериалов и материалов в электронном виде их носители размещают в кармане этой же папки.

Наличие графического материала является необходимым условием её

написания. Этот материал может быть представлен в виде схем, рисунков, таблиц, графиков и диаграмм, которые должны наглядно иллюстрировать и подтверждать основную часть работы. Графический материал может помещаться или в основном тексте, или (если он представлен в формате, большем, чем А4) выноситься в приложения к работе. Перечень графического материала устанавливается в задании на практику.

Список использованной литературы оформляется в соответствии с требованиями ГОСТ Р 7.0.100-2018, ГОСТ 7.32-2017.

Рекомендуется представлять единый список литературы к работе в целом. Список обязательно должен быть пронумерован. Каждый источник упоминается в списке один раз, вне зависимости от того, как часто на него делается ссылка в тексте работы.

Наиболее удобным является алфавитное расположение материала, так как в этом случае произведения собираются в авторских комплексах. Произведения одного автора расставляются в списке по алфавиту заглавий. Официальные документы ставятся в начале списка в определенном порядке: Конституция; Кодексы; Законы; Указы Президента; Постановления Правительства; другие нормативные акты (письма, приказы и т.д.). Внутри каждой группы документы располагаются в хронологическом порядке.

Литература на иностранных языках ставится в конце списка после литературы на русском языке, образуя дополнительный алфавитный ряд. Для каждого документа предусмотрены следующие элементы библиографической характеристики: фамилия автора, инициалы; название; подзаголовочные сведения (учебник, учебное пособие, словарь и т.д.); выходные сведения (место издания, издательство, год издания); количественная характеристика (общее количество страниц в книге).

В каждой группе библиографические записи должны располагаться в алфавитном порядке. Нумерация всего **списка использованных источников** сквозная. Библиографическая запись должна выполняться согласно ГОСТ Р 7.0.100-2018, ГОСТ 7.32-2017. Как правило, на языке выходных сведений: Автор (ФИО). Название источника. – Место издания: Издательство, год издания, количество страниц (общее или же страницы того раздела, который был использован при написании работы). Сведения об используемом издании находятся на обороте титульного листа книги;

- ссылки формируются в квадратных скобках, внутри которых указывается номер источника литературы, который соответствует его порядковому номеру в списке использованных источников. Например, ссылка [5] означает, что ссылка по тексту работы приведена на литературный источник, расположенный под порядковым номером 5 в Списке использованных источников;
- в случае необходимости дословного цитирования фрагмента авторского произведения заимствованный текст должен быть взят в кавычки и снабжён ссылкой на источник, содержащий данный текст. Если в тексте присутствует заключенная в кавычки цитата, то ссылка на источник должна приводиться с указанием не только номера источника в Списке

использованных источников, но и номера страницы источника, где расположена цитата, например, [10, с.37];

– указанные в Списке использованных источников электронные источники должны быть актуальными. Это означает, что дата обращения к электронному ресурсу должна соответствовать текущему месяцу, в котором осуществляются написание и проверка работы;

– запрещается использование в работе множественных ссылок на одно смысловое предложение, что наводит на мысль об их бессистемном, произвольном использовании.

В раздел Приложение включаются материалы, не вошедшие в текст основной части работы. Каждое приложение должно быть пронумеровано в порядке расположения приложений в тексте отчета практики. Заголовок «Приложение» располагается в правом верхнем углу с номером. Каждое приложение начинается с новой страницы.

– каждое Приложение имеет своё название, заголовок которого печатается в центре полужирным шрифтом, размер №16. Слово «Приложение» с номером (без знака № и точки на конце) печатается заглавными буквами в правом верхнем углу страницы, над названием. Каждое приложение начинается с новой страницы. Объем **Приложения** не входит в обязательное количество страниц работы и не должен превышать 1/3 всего текста работы;

– рисунки и таблицы имеют сквозную нумерацию в каждом отдельном Приложении. Например, рисунок 5, расположенный в Приложении В, обозначается: Рисунок В.5, таблица 3, расположенная в Приложении А, обозначается: Таблица А.3;

– работа должна быть написана от третьего лица. Использование в работе личных местоимений запрещается. Необходимо использовать выражения, типа: «известно, что», «существует мнение», «учёные придерживаются следующей точки зрения», «необходимо заметить», «представляет интерес» и т.п. Не допускается сокращение слов, все используемые аббревиатуры необходимо расшифровывать;

В тексте рекомендуется чаще применять красную строку, выделяя законченную мысль в самостоятельный абзац. Не следует приводить слишком много цитат, которые используются как приём аргументации. Текст не должен представлять собой сплошные перечисления. Необходимо чётко и ясно излагать материал, применяя принятую научную терминологию.

1.7. Подготовка презентационного материала

Обучающийся должен подготовить презентационный материал для защиты отчета по практике перед членами кафедры, используя средства Power Point. Структура презентационного материала (5– 10 слайдов):

– титульный лист (фирменный логотип ДГУ; название практики; выполнил обучающийся Ф.И.О., руководитель практики Ф.И.О., с указанием научного звания и степени, Махачкала 202_г»;

- Введение
- результаты практики (реферат)

1.8. Аттестация по итогам практики

Защита учебной практики проводится после окончания практики в сроки, определяемые деканатом.

К защите допускаются студенты, у которых материалы по практике оформлены надлежащим образом и собраны все необходимые документы, указанные в предыдущем подразделе «Требования к оформлению отчета по учебной ознакомительной практике».

Защита студентом учебной практики оценивается комиссией по модульно-рейтинговой системе (отлично, хорошо, удовлетворительно, неудовлетворительно).

Сроки прохождения практики по согласованию с деканатом факультета могут быть продлены студентам, нарушающим трудовую дисциплину.

Полностью оформленный отчет представляется руководителю практики от кафедры «Информационные системы и технологии программирования» для проверки и защиты.

На основании полученного отчета, руководитель практики от кафедры принимает решение о допуске студента к защите отчета. Допуск студента к защите указывается на отчете, который вместе с рецензией руководителя практики от кафедры передается студенту для защиты. При отсутствии отчета с соответствующими рецензиями комиссия вправе не допустить студента к защите.

Защита отчетов проводится на кафедре ИСиТП комиссией, в состав которой входят руководители практик от кафедры, а также другие преподаватели кафедры.

Отчет с указанием даты защиты передается руководителем практики от кафедры ИСиТП ответственному лицу (методисту) по кафедре ИСиТП. Отметки о защите отчета по практике проставляются руководителем практики от кафедры ИСиТП в зачетной книжке и экзаменационной ведомости. Студенты, не прошедшие практику в установленные учебным планом сроки, допускаются к прохождению практики только по решению деканата.

II. Примерная программа учебной практики

2.1. Общая часть

2.1.1. Задания, для развития профессиональных навыков по дисциплине «Программирование на C++»

Каждое задание содержит:

1. Название и цель (акцент на профессиональный навык).
2. Техническое задание (ТЗ).
3. Требования к реализации (кодстайл, безопасность, работа с памятью).

Задание №1. Система верификации банковской карты (Luhn Algorithm)

Цель: Отработка навыков работы с циклами, условными операторами и вводом/выводом данных. Формирование понимания важности валидации входных данных.

Техническое задание:

Разработать консольное приложение, которое запрашивает у пользователя номер банковской карты (строку, содержащую только цифры, длиной 16 символов). Программа должна проверить корректность номера с использованием алгоритма Луна (Luhn) и вывести на экран: «Карта валидна» или «Карта не валидна». Если в строке присутствуют нецифровые символы или длина не соответствует 16, программа должна выдать ошибку и запросить ввод повторно.

Требования:

- Использование функций: bool isNumber(const std::string& s), bool checkLuhn(const std::string& cardNumber).
- Запрещено использовать сторонние библиотеки, кроме <iostream>, <string>, <cctype>.

Задание №2. Генератор паролей с заданной политикой сложности

Цель: Развитие навыков работы со строками, генерацией случайных чисел и обработкой пользовательского ввода.

Техническое задание:

Создать программу-генератор паролей. Пользователь задает длину пароля (от 8 до 32 символов) и выбирает типы символов (флаги): строчные буквы, заглавные буквы, цифры, специальные символы (!@#$%^&*). Программа должна гарантировать, что в сгенерированном пароле присутствует хотя бы один символ из каждого выбранного типа. Результат выводится на экран.

Требования:

- Использование rand() или std::mt19937 (C++11) для генерации.
- Реализация функции std::string generatePassword(int length, bool useLower, bool useUpper, bool useDigits, bool useSpecial).

- Обработка краевых случаев: если пользователь не выбрал ни одного типа, выдать предупреждение и использовать по умолчанию строчные буквы.

Задание №3. Модуль для работы с одномерным массивом (CRUD)

Цель: Освоение работы с массивами фиксированного размера, алгоритмами поиска, сортировки и рефакторинга кода.

Техническое задание:

Реализовать программу для управления массивом целых чисел (размер массива задается пользователем динамически в начале, но не более 100). Программа должна предоставлять текстовое меню:

1. Заполнить массив случайными числами.
2. Вывести массив на экран.
3. Добавить элемент в конец (если есть место).
4. Удалить элемент по индексу.
5. Найти индекс первого вхождения числа.
6. Отсортировать массив (по возрастанию).
7. Выход.

Требования:

- Динамическое выделение памяти (new[] / delete[]) или использование std::vector (по усмотрению преподавателя, но в рамках практики предпочтительнее ручное управление памятью).
- Каждый пункт меню должен быть реализован в отдельной функции.

Задание №4. Библиотека матричных операций (2D массивы)

Цель: Формирование навыков работы с многомерными массивами, вложенными циклами и перегрузкой функций (или шаблонами).

Техническое задание:

Разработать модуль для работы с квадратными матрицами (размер N x N, N вводится пользователем). Реализовать функции:

- Ввод матрицы с клавиатуры.
- Вывод матрицы на экран в отформатированном виде.
- Вычисление суммы элементов на главной диагонали.
- Транспонирование матрицы (создание новой или изменение исходной).
- Умножение двух матриц.
- Требования:
- Использование динамического двумерного массива (массив указателей на массивы).
- Проверка возможности умножения (размерности должны совпадать для квадратных матриц).
- Освобождение памяти в конце работы.

Задание №5. Система управления складом (Структуры и перечисления)

Цель: Отработка навыков объектно-ориентированного мышления на основе структур (struct), работы с перечислениями (enum) и сортировки сложных данных.

Техническое задание:

Создать структуру Product, содержащую поля:
id (int), name (string), quantity (int), category (enum: Electronics, Clothing, Food, Books). Программа должна хранить массив (вектор) товаров. Реализовать функции:

1. Добавление нового товара.
2. Поиск товара по имени.
3. Вывод всех товаров определенной категории.
4. Удаление товара по id.
5. Сохранение текущего списка в текстовый файл warehouse.txt.
6. Загрузка списка из файла при старте программы.

Требования:

- Использование std::vector<Product>.
- Работа с файловыми потоками <fstream>.
- Форматированный вывод данных в файл и их парсинг обратно.

Задание №6. Эмулятор работы банкомата (ООП. Классы)

Цель: Введение в объектно-ориентированное программирование: инкапсуляция, конструкторы, методы.

Техническое задание:

Спроектировать класс BankAccount. Класс должен содержать приватные поля: accountNumber (string), pinCode (int), balance (double). Публичные методы:

- Конструктор для инициализации счета.
- bool authenticate(int pin) — проверка ПИН-кода.
- void deposit(double amount) — пополнение.
- bool withdraw(double amount) — снятие (если достаточно средств).
- double getBalance() — получение баланса (только после аутентификации).

  В main создать один счет, запросить у пользователя ПИН-код (3 попытки) и предоставить меню операций.
- Требования:
- Все данные должны быть скрыты (private).
- Логика аутентификации не должна позволять просматривать баланс без ввода верного ПИН-кода.

Задание №7. Обработчик текстовых логов (Работа со строками и файлами)

Цель: Развитие навыков парсинга строк, использования stringstream и обработки больших объемов текстовой информации.

Техническое задание:

Дан текстовый файл log.txt, содержащий строки в формате: [YYYY-MM-DD HH:MM:SS] LEVEL: MESSAGE, где LEVEL может быть INFO, WARNING, ERROR. Написать программу, которая:

1. Читает файл построчно.
2. Подсчитывает количество сообщений каждого уровня.
3. Создает три отдельных файла: info.log, warning.log, error.log, распределяя сообщения по соответствующим файлам.
4. Находит и выводит на экран все сообщения за указанную пользователем дату.

   Требования:
   - Использование std::getline и std::stringstream для разбора строки.
   - Обработка ситуаций, когда исходный файл не существует.

Задание №8. Реализация простого стека (Динамические структуры данных)

Цель: Понимание принципов динамического выделения памяти, работы с указателями и структуры данных LIFO.

Техническое задание:

Реализовать структуру данных «Стек» для хранения целых чисел *без использования* std::stack и std::vector. Стек должен быть реализован на основе односвязного списка (узел: значение + указатель на следующий). Реализовать функции:

- push(int value) — добавить элемент на вершину.
- pop() — удалить элемент с вершины (если не пуст).
- top() — посмотреть значение на вершине.
- bool isEmpty().
- Деструктор для освобождения всей памяти.
- Требования:
- Строгое соблюдение управления памятью (нет утечек).
- В main продемонстрировать работу стека на примере проверки скобочной последовательности (например, ({[]})).

Задание №9. Программа "Тестирование знаний" (Файловый ввод/вывод + классы)

Цель: Закрепление навыков работы с классами, контейнерами STL и сериализацией данных.

Техническое задание:

Разработать приложение для проведения тестов. Вопросы и варианты ответов хранятся в текстовом файле questions.txt в формате:

text

Вопрос: Сколько бит в байте?

Варианты: 4;8;16;32

Правильный: 2

Программа должна:

1. Загрузить вопросы в вектор объектов класса Question (текст, варианты, индекс правильного ответа).
2. Последовательно задавать вопросы пользователю, вести подсчет баллов.
3. По окончании теста вывести результат (количество правильных ответов) и сохранить историю попыток (имя пользователя, дата, результат) в файл results.csv.

Требования:

- Использование std::vector и std::fstream.
- Обработка ошибок чтения файла вопросов.

Задание №10. Консольный файловый менеджер (Комплексная работа)

Цель: Синтез всех полученных знаний: работа с файловой системой (C++17 filesystem), пользовательским вводом, обработкой строк и архитектурой приложения.

Техническое задание:

Разработать консольное приложение, которое принимает команды пользователя (аналог простой командной строки):

- ls [path] — показать содержимое директории (файлы и папки с указанием размера).
- cd [path] — сменить текущую рабочую директорию.
- copy <source> <dest> — копировать файл.
- move <source> <dest> — переместить/переименовать файл.
- info <file> — вывести информацию о файле (размер, дата создания/изменения).
- exit — выход.

Требования:

- Использование библиотеки <filesystem> (C++17).
- Реализация парсера команд (разбиение строки ввода на команду и аргументы).
- Обработка ошибок (файл не найден, недостаточно прав, неверная команда).

Методические рекомендации для студентов (Памятка):

1. **Система контроля версий:** Все задания рекомендуется выполнять в отдельном репозитории Git. Каждое задание — отдельная папка или ветка. Это формирует навык работы с VCS, критически важный для инженера.
2. **Code Style:** Следовать единому стилю кодирования (например, Google C++ Style Guide или договоренности кафедры). Имена переменных должны быть говорящими (не a, b, c).
3. **Защита памяти:** В заданиях, где используется динамическая память (new), строго следить за наличием delete. Рекомендуется использовать принцип RAII (где возможно).

4. **Документация:** Каждый исходный файл должен начинаться с комментария: ФИО студента, группа, номер задания, краткое описание. Сложные функции должны быть задокументированы.

Задания для развития профессиональных навыков по дисциплине «Основы программной инженерии»

Задание 1. Анализ и «Реверс-инжиниринг» пользовательского интерфейса

Тема: Анализ требований и юзабилити. **Задача:** Выберите популярное приложение (например, Telegram, Spotify, Figma, Notion). Проведите его анализ как программный инженер, а не как пользователь.

1. Составьте **Use Case Diagram** (диаграмму вариантов использования) для 3 ключевых сценариев работы приложения.
2. Опишите один сценарий в формата **User Story**: «Будучи [роль], я хочу [действие], чтобы [ценность]».
3. Найдите и опишите 2 потенциальных нефункциональных требования (производительность, безопасность, масштабируемость), которые предъявлялись к этому приложению.
4. **Результат:** Документ в формате Markdown (или PDF), содержащий диаграмму (скриншот или описание) и текстовые спецификации.

Задание 2. Создание технического задания (ТЗ) на основе чужого кода

Тема: Формирование требований и документации. **Задача:** Найдите на GitHub любой открытый проект на Python/Java/JS (репозиторий должен быть небольшим, до 500 строк кода). Изучив код, составьте обратное техническое задание (Reverse Engineering SRS).

1. Определите основные функциональные требования (что делает программа).
2. Опишите архитектуру на высоком уровне (входные данные -> обработка -> выходные данные).
3. Составьте глоссарий (список терминов, используемых в проекте).
4. **Результат:** Файл SRS.md (Software Requirements Specification) в корневой папке форка этого репозитория.

Задание 3. Введение в Git: История проекта и первый Pull Request

Тема: Системы контроля версий. **Задача:** Создайте личный репозиторий на GitHub. Смоделируйте процесс разработки простого скрипта (калькулятор, конвертер валют).

1. Создайте ветку feature/calculator-logic.
2. Напишите код, сделайте 3 осмысленных коммита с сообщениями по стандарту **Conventional Commits** (feat:, fix:, docs:).
3. Создайте ветку feature/readme-update, обновите README.md (опишите, как запускать проект, примеры использования).

4. Откройте **Pull Request** (или Merge Request) из фича-ветки в main. Напишите понятное описание PR.
5. **Результат:** Ссылка на репозиторий с историей коммитов и закрытым PR.

Задание 4. Моделирование предметной области (UML)

Тема: Объектно-ориентированное проектирование. **Задача:** Спроектируйте систему «Библиотека». Не пишите код, создайте полную диаграмму классов (Class Diagram).

1. Выделите сущности: Книга, Читатель, Библиотекарь, Журнал выдачи.
2. Определите атрибуты и методы (минимум 3-4 на класс).
3. Покажите связи: ассоциация, композиция, наследование (если есть).
4. Используйте инструменты: draw.io, Lucidchart или PlantUML (код диаграммы).
5. **Результат:** Файл диаграммы (png/jpeg) или текстовый файл .puml с описанием логики.

Задание 5. Основы CI/CD: Автоматическая проверка кода

Тема: Непрерывная интеграция. **Задача:** К репозиторию из Задания 3 (или новому) подключите GitHub Actions (или GitLab CI).

1. Напишите простой скрипт на Python (или JS), который проходит по файлам и что-то подсчитывает.
2. Создайте файл .github/workflows/check.yml, который будет автоматически запускаться при пуше в ветку main.
3. В workflow добавьте шаги:
   - Проверка синтаксиса (линтер, например pylint или flake8).
   - Запуск простых unit-тестов (если есть).
4. Намеренно сделайте ошибку в коде, чтобы пайплайн упал (red build), затем исправьте её и добейтесь зелёного билда (green build).
5. **Результат:** Скриншот успешного выполнения CI в интерфейсе GitHub (Actions) и ссылка на репозиторий.

Задание 6. Тестирование: Написание чек-листов и баг-репортов

Тема: Обеспечение качества (QA). **Задача:** Возьмите любое веб-приложение (например, онлайн-редактор кода или форму регистрации на любом сайте).

1. Составьте **чек-лист** для проверки формы регистрации (поля: имя, email, пароль, подтверждение). Минимум 15 пунктов (позитивные и негативные сценарии).
2. Найдите в этом приложении **реальный баг** (если не нашли — придумайте логический баг в воображаемой форме).
3. Оформите **баг-репорт** по стандарту:
   - Заголовок (краткое описание).
   - Шаги воспроизведения.
   - Фактический результат.
   - Ожидаемый результат.
   - Окружение (ОС, браузер).

- Серьезность (Severity) и Приоритет (Priority) с обоснованием.

4. **Результат:** Документ (Google Docs / Markdown) с чек-листом и баг-репортом.

Задание 7. Базы данных: Проектирование схемы и SQL запросы

Тема: Работа с данными.

Задача: Спроектировать базу данных для интернет-магазина.

1. Нарисовать **ER-диаграмму** (сущности: Товары, Категории, Покупатели, Заказы, Корзина).
2. Написать SQL-скрипты создания таблиц (CREATE TABLE) с указанием первичных и внешних ключей.
3. Написать 3 сложных запроса (SELECT) с использованием JOIN, GROUP BY, HAVING (например: «Топ-5 самых продаваемых товаров за последний месяц»).
4. **Результат:** Файл database.sql с комментариями и изображение ER-диаграммы.

Задание 8. Архитектурный паттерн: Реализация простого MVC

Тема: Архитектура приложений.

Задача: Разработать консольное приложение «Список задач» (ToDo List) строго по паттерну **Model-View-Controller (MVC)**.

1. **Model:** Класс Task (id, title, status). Логика хранения (в файле JSON).
2. **View:** Функции отображения меню и вывода списка задач в консоль.
3. **Controller:** Обработка ввода пользователя (add, list, complete, delete), взаимодействие с Model.
4. Код должен быть разнесен по разным файлам/папкам (например, src/models/, src/views/, src/controllers/).
5. **Результат:** Ссылка на репозиторий с кодом и краткое описание того, как разделение ответственности помогло в разработке.

Задание 9. Оценка трудоемкости и планирование (Story Points)

Тема: Управление разработкой (Scrum/Agile).

Задача: Представьте, что вы тимлид. Вам пришло ТЗ на создание простого блога (посты, комментарии, авторизация, админка).

1. Разбейте проект на **пользовательские истории** (User Stories). Минимум 8-10 штук.
2. Оцените каждую историю в **Story Points**, используя последовательность Фибоначчи (1, 2, 3, 5, 8). Обоснуйте оценку (риски, неопределенность, сложность).
3. Составьте план спринта (Sprint Backlog) на 2 недели, выбрав истории, которые команда из 3 человек (условно) может реализовать.
4. **Результат:** Таблица в Excel/Markdown с колонками: ID истории, Название, Описание, SP, Приоритет.

Задание 10. Рефакторинг «Легаси» кода и Code Review

Тема: Работа с техническим долгом.

Задача: Вам предоставлен «плохой» код (на выбор: спагетти-код, отсутствие имен, дублирование). *Преподаватель может выдать свой пример, либо студент находит фрагмент кода на GitHub с меткой «bad practice»*.

1. Проведите **Code Review** этого кода. Напишите список замечаний (минимум 5) по поводу:
    - Именования переменных.
    - Нарушения принципов SOLID (хотя бы одного).
    - Отсутствия обработки ошибок.
    - Дублирования.
2. Выполните **рефакторинг**: перепишите код, применив принципы чистой архитектуры (разделите на функции, уберите дублирование, добавьте понятные имена).
3. Напишите краткий отчет: «Было -> Стало». Объясните, почему ваша версия лучше с точки зрения поддержки (maintainability).
4. **Результат:** Два файла с кодом (old.py и refactored.py) + файл REVIEW.md с комментариями ревьюера.

Рекомендации для преподавателя/руководителя практики:

1. **Сложность:** Задания 1, 2, 4, 6, 9 — больше аналитические и документальные (подходят для студентов, которые еще неуверенно пишут код). Задания 3, 5, 7, 8, 10 — требуют практического применения инструментов (Git, CI, SQL, код).
2. **Вариативность:** Студентам можно предложить на выбор выполнить 6-7 из 10 заданий, чтобы учесть разный уровень подготовки.
3. **Портфолио:** По итогам практики у студента формируется GitHub-портфолио (репозитории с кодом, CI, документацией), что является отличным дополнением к резюме.
4. **Защита:** Каждое задание должно защищаться устно, где студент объясняет свои проектные решения (почему выбрал именно такую структуру БД, почему так оценил Story Points).

Задания для развития профессиональных навыков по дисциплине «Технологии и методы программирования»

Задание 1. Калькулятор арифметических выражений

Название: Калькулятор арифметических выражений

Цель: Научиться работать с вводом/выводом, условными операторами, циклами и базовой обработкой исключений. Развить навыки написания читаемого кода и тестирования на граничных случаях.

Техническое задание:

Разработать программу, которая запрашивает у пользователя два числа и операцию (+, -, *, /, //, %, **). Программа выполняет вычисление и выводит результат. Поддерживается возможность многократных вычислений до тех пор, пока пользователь не введёт команду выхода ("exit" или "quit"). При делении на ноль программа должна выводить сообщение об ошибке и предлагать повторить ввод. При вводе некорректных чисел – обрабатывать исключение ValueError.

Требования к реализации:

Код должен быть разбит на функции: get_number(prompt), get_operation(), calculate(a, b, op), main().

Использовать docstring для каждой функции.

Обрабатывать ошибки ввода с помощью try-except.

Программа не должна завершаться аварийно ни при каких обстоятельствах.

Стиль кодирования – PEP 8.

Задание 2. Анализатор текста

Название: Анализатор текста (статистика)

Цель: Освоить строковые методы, работу со списками, множествами и словарями. Развить навыки анализа данных и составления отчётов.

Техническое задание:

Написать программу, которая принимает на вход текстовый файл (имя файла запрашивается у пользователя) и выводит следующую статистику:

– Общее количество слов.

– Количество уникальных слов.

– Самое длинное слово (если несколько – первое по алфавиту).

– Частоту встречаемости каждого символа (без учёта пробелов и знаков препинания).

– Топ-5 самых частых слов.

– Если файл не существует, программа должна сообщить об ошибке и запросить другое имя.

Требования к реализации:

Использовать with для работы с файлами.

Предусмотреть обработку исключений FileNotFoundError, PermissionError.

Разделить логику на функции: read_file(filename), clean_text(text), count_stats(words), print_report(stats).

Все статистические данные должны быть представлены в виде аккуратно отформатированного отчёта.

Добавить комментарии к сложным фрагментам.

Задание 3. Система управления заметками

Название: Система управления заметками (файловое хранилище)

Цель: Научиться сохранять данные в файл (JSON) и загружать из него, реализовать CRUD-операции. Развить навыки проектирования простого интерфейса командной строки.

Техническое задание:

Разработать консольное приложение для управления заметками. Каждая заметка содержит: id (автоинкремент), заголовок, текст, дату/время создания (автоматически). Программа должна поддерживать команды:

add – добавить новую заметку (запросить заголовок и текст).

list – вывести все заметки в виде таблицы (id, заголовок, дата).

view <id> – показать полную заметку.

delete <id> – удалить заметку.

exit – выход.

Данные должны сохраняться в файл notes.json и автоматически загружаться при запуске.

Требования к реализации:

Использовать модуль json.

При старте, если файла нет, создавать пустой список заметок.

Обрабатывать ситуацию, когда введён неверный id (выводить сообщение).

Функции: load_notes(), save_notes(notes), add_note(notes), list_notes(notes), view_note(notes, note_id), delete_note(notes, note_id). Использовать datetime для меток времени.

Задание 4. Игра «Угадай число» с рекордами

Название: Игра «Угадай число» с таблицей рекордов

Цель: Закрепить циклы, генераторы случайных чисел, работу со списками и сортировку. Развить навыки организации игрового цикла и сохранения прогресса.

Техническое задание:

Компьютер загадывает целое число от 1 до 100. Игрок пытается его угадать, вводя числа. После каждой попытки программа подсказывает «больше» или «меньше». Когда число угадано, программа поздравляет игрока и запрашивает его имя. Количество попыток сохраняется в файл records.txt вместе с именем. После каждой игры показывается таблица топ-5 лучших результатов (меньше попыток – лучше). Предусмотреть возможность просмотра рекордов без игры.

Требования к реализации:

Использовать модуль random.

Функции: generate_number(), play_game(), save_record(name, attempts), show_records().

Файл с рекордами должен быть текстовым (CSV или просто строки вида имя:попытки).

Обрабатывать ввод нечисловых значений.

Программа должна иметь простое меню: 1 – играть, 2 – рекорды, 3 – выход.

Задание 5. Генератор паролей с проверкой сложности

Название: Генератор паролей с проверкой сложности

Цель: Научиться использовать строковые константы, модуль random, функции для валидации. Развить навыки создания утилит командной строки с аргументами.

Техническое задание:

Разработать программу, которая генерирует случайные пароли заданной длины (по умолчанию 12). Пароль должен включать символы из следующих категорий: буквы верхнего и нижнего регистра, цифры, спецсимволы (!@#$%^&*). Программа должна позволять пользователю выбрать длину пароля и набор категорий (можно исключать некоторые). Также реализовать функцию проверки сложности пароля (введённого пользователем), которая возвращает оценку от 1 до 5 и замечания (например, «нет цифр», «короткий»).

Требования к реализации:

Использовать argparse для обработки аргументов командной строки (например: python passgen.py --length 16 --no-digits).

Для интерактивного режима (без аргументов) – запрашивать параметры через input.

Функции: generate_password(length, use_upper, use_lower, use_digits, use_special), check_strength(password), main().

Обеспечить, чтобы при генерации использовалась хотя бы одна категория (если все отключены – выводить ошибку).

Выводить сгенерированный пароль и его оценку сложности.

Задание 6. Телефонная книга (объектно-ориентированная)

Название: Телефонная книга (ООП)

Цель: Освоить основы объектно-ориентированного программирования: классы, объекты, инкапсуляция, методы. Развить навыки проектирования структуры данных и взаимодействия объектов.

Техническое задание:

Реализовать приложение «Телефонная книга», где каждый контакт представлен классом Contact с полями: name, phone, email (опционально). Класс PhoneBook управляет коллекцией контактов и предоставляет методы: add_contact(), remove_contact(name), search_contact(query) (поиск по имени, телефону или email), show_all(), save_to_file(filename), load_from_file(filename). Приложение должно иметь консольное меню для вызова этих операций. Данные хранятся в JSON-файле.

Требования к реализации:

Класс Contact должен иметь метод __str__ для красивого вывода.

В классе PhoneBook использовать приватный список контактов (по соглашению _contacts). Поиск должен возвращать список совпадений (регистронезависимый).

При добавлении проверять, что номер телефона содержит только цифры, знак + и пробелы (простая валидация). Обрабатывать ошибки ввода/вывода при работе с файлами. Код должен быть документирован (docstring для классов и методов).

Задание 7. Интерактивный калькулятор

Название: Интерактивный калькулятор

Цель: Закрепить навыки работы с переменными, арифметическими операциями, условными конструкциями, циклами и обработкой исключений. Развить умение декомпозировать задачу на функции и документировать код.

2. Техническое задание

– Разработать консольное приложение – калькулятор, который:

– Запрашивает у пользователя два числа и операцию (+, -, *, /, **, %).

– Выполняет вычисление и выводит результат.

– При делении на ноль или возведении 0 в отрицательную степень выводится сообщение об ошибке.

– При вводе неверной операции – сообщение об ошибке и повторный запрос.

– Программа работает в бесконечном цикле, пока пользователь не введёт команду выхода (например, exit или q).

– После каждого вычисления программа спрашивает: «Продолжить? (да/нет)».

3. Требования к реализации

Реализовать отдельные функции для каждой арифметической операции.

Создать функцию main(), управляющую ходом программы.

Каждая функция должна содержать docstring (описание назначения, параметров, возвращаемого значения).

Обязательно использовать обработку исключений (try-except) для преобразования ввода в числа.

Код должен быть оформлен согласно PEP 8 (отступы, имена переменных, пробелы).

Допускается использование модуля math для дополнительных операций (например, sqrt – по желанию).

Задание 8. Журнал успеваемости студентов

1. Название и цель

Название: Журнал успеваемости студентов

Цель: Научиться работать со списками, словарями, файловым вводом/выводом, организовать меню взаимодействия с пользователем. Закрепить навыки обработки ошибок и структурирования кода.

2. Техническое задание

Разработать консольное приложение для управления оценками студентов. Программа должна поддерживать следующие действия:

– Добавить студента – запрашивается имя студента.

– Добавить оценку – выбрать студента из списка, ввести оценку (целое число от 2 до 5), добавить к его оценкам.

– Показать успеваемость – вывести список всех студентов и их средний балл.

– Сохранить данные в файл – все данные (имя студента → список оценок) сохраняются в JSON-файл.

– Загрузить данные из файла – при запуске программа автоматически загружает данные из файла (если он существует). Пользователь также может загрузить данные вручную.

– Выход – завершение работы.

3. Требования к реализации

Данные хранить в словаре: ключ – имя студента (str), значение – список оценок (list[int]).

Каждое действие реализовать в отдельной функции.

При сохранении/загрузке использовать модуль json.

Предусмотреть обработку ситуаций:

Попытка добавить оценку несуществующему студенту.

Некорректный ввод оценки (буквы, числа вне диапазона 2–5).

Отсутствие файла при загрузке (выводить сообщение и продолжать работу с пустым словарём).

Для отображения среднего балла реализовать функцию calculate_average(grades: list) -> float.

Программа должна иметь текстовое меню с номерами пунктов.

Соблюдать PEP 8 и документировать функции.

Задание 9. Анализатор текста

1. Название и цель

Название: Анализатор текста

Цель: Освоить работу со строками, файлами, регулярными выражениями (базово). Развить навыки статистического анализа данных и написания отказоустойчивого кода.

2. Техническое задание

Разработать программу, которая анализирует текстовый файл (имя файла вводит пользователь) и выводит следующую статистику:

– Количество символов (включая пробелы).

– Количество слов (слово – последовательность букв, цифр и апострофов, разделённая пробелами/знаками препинания).

– Количество предложений (предложение заканчивается на ., !, ?).

– Самое часто встречающееся слово (или слова, если несколько имеют одинаковую максимальную частоту).

– Среднюю длину слова (в символах).

– Программа также должна предоставлять возможность проанализировать текст, введённый вручную (без файла).

– Результаты анализа выводятся на экран и сохраняются в файл analysis_result.txt.

3. Требования к реализации

Обработка ошибок:

Файл не найден.

Файл пуст.

Для разбиения на слова и предложения использовать методы строк или модуль re (регулярные выражения).

Для подсчёта частоты слов использовать словарь (dict).

При подсчёте самого частого слова игнорировать регистр (приводить к нижнему).

Знаки препинания не должны влиять на подсчёт слов (т.е. "hello," считать как "hello").

Функциональность разбить на логические функции:

read_text(source) – чтение из файла или из строки.

count_characters(text)

count_words(text)

count_sentences(text)

most_frequent_word(text)

average_word_length(text)

save_report(report_dict, filename)

Главная функция организует диалог с пользователем (выбор источника данных, вывод отчёта, сохранение).

Задание 10. Простая библиотечная система (ООП)

1. Название и цель

Название: Простая библиотечная система

Цель: Познакомиться с основами объектно-ориентированного программирования: классы, объекты, инкапсуляция, методы, специальные методы (__str__). Научиться создавать собственные исключения и сохранять объекты в JSON.

2. Техническое задание

Реализовать консольное приложение для управления книгами в библиотеке. Должны быть два класса:

Класс Book

Атрибуты:

- title (название)
- author (автор)
- isbn (уникальный идентификатор, строка)
- is_borrowed (логический флаг, доступна ли книга для выдачи)
- Методы:
- borrow() – выдать книгу (если доступна, иначе вызвать исключение).
- return_book() – вернуть книгу.
- __str__() – возвращает строку вида "Название: ... Автор: ... ISBN: ... Статус: ...".

Класс Library

Атрибут: список книг (list[Book]).

Методы:

add_book(book) – добавить книгу (проверять, нет ли книги с таким же ISBN).

remove_book(isbn) – удалить книгу по ISBN (если не найдена – исключение).

search_by_title(title) – вернуть список книг, содержащих заданную подстроку в названии.

search_by_author(author) – аналогично по автору.

borrow_book(isbn) – найти книгу по ISBN и вызвать её метод borrow().

return_book(isbn) – вернуть книгу.

save_to_file(filename) – сохранить все книги в JSON-файл (сериализовать в словари).

load_from_file(filename) – загрузить книги из JSON-файла.

Консольное меню позволяет выполнять все операции (добавление, удаление, поиск, выдача, возврат, сохранение, загрузка, вывод всех книг).

3. Требования к реализации

Создать собственные классы исключений: BookNotFoundError, BookAlreadyBorrowedError, DuplicateBookError.

Все методы классов должны быть документированы. Для сериализации/десериализации книг написать вспомогательные функции book_to_dict(book) и dict_to_book(data). При сохранении/загрузке использовать модуль json. При поиске по названию/автору регистр не должен учитываться.

Код должен быть структурирован: классы в отдельном модуле (library_models.py), основная программа в main.py.

Следовать PEP 8 и принципам DRY (Don't Repeat Yourself).

Индивидуальные задания по учебной практике

2.2.1.Темы рефератов

1. ИИ в финансовой сфере.
2. ИИ в военном деле.
3. ИИ в медицине.
4. Экспертные системы в финансовой и банковской сфере.
5. Экспертные системы в медицине.
6. Экспертные системы в образовании.
7. Применение ИИ в медицинской диагностике.
8. Применение нейросетей в обработке естественного языка и машинном переводе текстов.
9. Разработка систем ИИ для управления умными домашними устройствами.
10. Самоуправляемые автомобили и транспорт.
11. Компьютерное зрение и распознавание образов.
12. Исследование технологий беспроводных сетей и их применение.
13. Исследование проблем безопасности в сети.
14. Автопилоты.
15. Искусственный интеллект в борьбе с киберпреступлениями и кибербезопасности.
16. Перспективы развития устройств хранения информации.
17. Новые технологии хранения информации.
18. Обзор популярных облачных хранилищ данных.
19. Технологии хранения и передачи информации.
20. Нейронные сети и их применение.
21. Нейронные сети в системах автоматического управления.
22. Блокчейн.
23. Криптовалюта.
24. Провайдеры услуг Интернет в России(Дагестане): сравнительные характеристики.
25. Сравнение разных браузеров. Стандарты создания веб-сайтов.
26. Использование информационных технологий в сфере реабилитации.
27. Робототехника.
28. Интернет вещей.
29. Электронные платежные системы: классификация и сравнительные характеристики.
30. Классификация ЛВС.
31. Проектирование и настройка безопасной беспроводной сети.
32. Среды передачи данных в сети.
33. Мультимедиа-технологии.
34. Технология виртуальной реальности.
35. Технологии обработки графической информации.
36. Возобновляемая энергия.
37. Цифровые кошельки.

38. Телекоммуникационная вычислительная сеть.
39. Телекоммуникационные услуги.
40. Internet.
41. Fidonet.
42. Набор протоколов TCP/IP. Уровни стека протоколов TCP/IP. Назначение протоколов отдельных уровней.
43. Спутниковые сети передачи данных и их характеристика.
44. Технология широкополосного беспроводного доступа WiMAX.
45. Беспилотные автомобили.
46. Геоинформационные системы.
47. Системы дистанционного обучения. Обзор ресурсов. Анализ и классификация.
48. Системы автоматизированного проектирования.
49. Витрины данных.
50. «Развлекательные» ресурсы Интернет. Обзор, анализ, классификация.
51. Платежные системы в сети Интернет. Ресурсы, содержание, перспективы развития.
52. Системы дистанционного обучения. Обзор ресурсов. Анализ и классификация.

III. Примерный перечень контрольных вопросов при приеме материалов учебной практики

1. Общая характеристика языков программирования
2. Структура программного обеспечения
3. Этапы разработки программ.
4. Свойства и способы описания алгоритма.
5. Типы данных.
6. Формат объявления переменной.
7. Область действия идентификатора
8. Синтаксические и семантические ошибки, примеры алгоритмов.
9. Целые, вещественные, символьные и строковые константы.
10. Способы объявления именованных констант.
11. Перечисляемые типы данных.
12. Операторы объявления.
13. Исполняемые операторы.
14. Условный оператор.
15. Оператор множественного выбора.
16. Алгоритм линейной структуры.
17. Операторы безусловного перехода
18. Нахождение макс и мин.
19. Оператор break.
20. Оператор цикла с параметром FOR.
21. Вычисление сумм, произведений.
22. Цикл с предусловием WHILE
23. Цикл с постусловиемDO-WHILE .
24. Безусловный переходgoto.
25. Оператор перехода continue.
26. Функция exit.
27. Объявление и способы инициализации массива.
28. Символьный массив формат объявления.
29. Многомерные массивы.
30. Типовые действие над массивом (Σ,П, !, Σпол. Σотр.,и т.д.).
31. Объявление и способы инициализации массива.
32. Типовые действие над массивом (Σ,П, !, Σпол. Σотр.,и т.д.).
33. Примеры программ на действия с массивами.
34. Многомерные массивы.
35. Формирование псевдодинамических массив.
36. Использование датчика случайных чисел для формирования массива
37. Манипуляторы и форматирование ввода – вывода.
38. Простые методы сортировки. Примеры решения задач.
39. Быстрые методы сортировки. Примеры решения задач.
40. Объявление указателей. Инициализация.
41. Арифметика указателей.
42. Массивы указателей.
43. Указатели на указатели.

44. Указатели на массивы.
45. Операторы new и delete. Примеры составления программ и технология их решения.
46. Символьный массив формат объявления.
47. Формирование одномерных динамических массивов
48. Формирование двумерных динамических массивов
49. Функция malloc, calloc, free.
50. Символьный тип данных. Обработка элементов символьных массивов

Приложение 1

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«ДАГЕСТАНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»
Факультет информатики и информационных технологий

Отчет

по учебной (ознакомительной) практике

Выполнил: студент 1 курса,
очной формы обучения
направления 09.03.04.
Программная инженерия
Магомедов А.А.

Руководитель: к.э.н.,
доцент кафедры ИСиТП
Баммаева Г.А.

Махачкала 2026