МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ «РЯЗАНСКИЙ ГОСУДАРСТВЕННЫЙ РАДИОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ ИМЕНИ В.Ф. УТКИНА»

Кафедра «Вычислительная и прикладная математика»

ОЦЕНОЧНЫЕ МАТЕРИАЛЫ ПО ДИСЦИПЛИНЕ

«Основы программной инженерии**»**

Направление подготовки
09.03.04 «Программная инженерия»

Направленность (профиль) подготовки
«Программное обеспечение систем искусственного интеллекта»

Уровень подготовки – бакалавриат

Квалификация выпускника – бакалавр

Форма обучения – очная

Срок обучения – 4 года

Рязань 2023 г.

1. ОБЩИЕ ПОЛОЖЕНИЯ

Оценочные материалы – это совокупность учебно-методических материалов и процедур, предназначенных для оценки качества освоения обучающимися данной дисциплины как части основной образовательной программы.

Цель – оценить соответствие знаний, умений и уровня приобретенных компетенций, обучающихся целям и требованиям основной образовательной программы в ходе проведения текущего контроля и промежуточной аттестации.

Основная задача – обеспечить оценку уровня сформированности компетенций и индикаторов их достижения, приобретаемых обучающимися в соответствии с этими требованиями.

Контроль знаний обучающихся проводится в форме текущего контроля и промежуточной аттестации.

Текущий контроль успеваемости и промежуточная аттестация проводятся с целью определения степени усвоения учебного материала, своевременного выявления и устранения недостатков в подготовке обучающихся, организации работы обучающихся в ходе учебных занятий и оказания им индивидуальной помощи.

К контролю текущей успеваемости относятся проверка знаний, умений и навыков обучающихся на практических занятиях по результатам выполнения и защиты обучающимися индивидуальных заданий, по результатам выполнения контрольных работ и тестов, по результатам проверки качества конспектов лекций и иных материалов.

В качестве оценочных средств на протяжении семестра используется устные и письменные ответы студентов на индивидуальные вопросы, письменное тестирование по теоретическим разделам курса, реферат. Дополнительным средством оценки знаний и умений студентов является отчет о выполнении практических заданий и его защита.

По итогам курса обучающиеся сдают экзамен. Форма проведения – устный ответ с письменным подкреплением по утвержденным билетам, сформулированным с учетом содержания дисциплины. В билет для экзамена включается два теоретических вопроса и задача. В процессе подготовки к устному ответу студент должен составить в письменном виде план ответа.

1. Перечень компетенций с указанием этапов их формирования

При освоении дисциплины формируются следующие компетенции: ОПК-4 (индикаторы ОПК-4.2, ОПК-4.1), ОПК-9 (индикаторы ОПК-9.1, ОПК-9.2).

Указанные компетенции формируются в соответствии со следующими этапами:

– формирование и развитие теоретических знаний, предусмотренных указанными компетенциями (лекционные занятия, самостоятельная работа студентов);

– приобретение и развитие практических умений, предусмотренных компетенциями (практические занятия, самостоятельная работа студентов);

– закрепление теоретических знаний, умений и практических навыков, предусмотренных компетенциями, в ходе решения конкретных задач на занятиях, выполнения индивидуальных заданий на практических занятиях и их защиты, а так же в процессе сдачи экзамена.

2. Показатели и критерии оценивания компетенций (*результатов*) на различных этапах их формирования, описание шкал оценивания

Сформированность каждой компетенции в рамках освоения данной дисциплины оценивается по трехуровневой шкале:

– пороговый уровень является обязательным для всех обучающихся по завершении

освоения дисциплины;

– продвинутый уровень характеризуется превышением минимальных характеристик сформированности компетенций по завершении освоения дисциплины;

– эталонный уровень характеризуется максимально возможной выраженностью компетенций и является важным качественным ориентиром для самосовершенствования.

При достаточном качестве освоения более 80% приведенных знаний, умений и навыков преподаватель оценивает освоение данной компетенции в рамках настоящей дисциплины на эталонном уровне, при освоении более 60% приведенных знаний, умений и навыков – на продвинутом, при освоении более 40% приведенных знаний умений и навыков – на пороговом уровне. При освоении менее 40% приведенных знаний, умений и навыков компетенция в рамках настоящей дисциплины считается неосвоенной.

Уровень сформированности каждой компетенции на различных этапах ее формирования в процессе освоения данной дисциплины оценивается в ходе текущего контроля успеваемости и представлено различными видами оценочных средств.

Оценке сформированности в рамках данной дисциплины подлежат компетенции/индикаторы:

Показатели достижения планируемых результатов обучения и критерии их оценивания на разных уровнях формирования компетенций приведены в таблице 1.

Таблица 1. Показатели достижения индикаторов компетенции

1	2	3	4
Компетенция: код по ФГОС 3++, формулировка	**Индикаторы**	**Этап**	**Наименование оценочного средства**
ОПК-4 (09.03.04) Способен участвовать в разработке стандартов, норм и правил, а также технической документации, связанной с профессиональной деятельностью	**ОПК-4.1. Понимает суть и следует требованиям нормативно-регулирующих документов, связанных с профессиональной деятельностью** ЗНАТЬ - основные стандарты оформления технической документации на различных стадиях жизненного цикла информационной системы. УМЕТЬ - применять стандарты оформления технической документации на различных стадиях жизненного цикла информационной системы. ВЛАДЕТЬ - требованиями нормативно-регулирующих документов, связанных с профессиональной деятельностью. **ОПК-4.2. Разрабатывает и использует стандарты, нормы и правила, а также техническую документацию, связанную с профессиональной деятельностью**	1	Зачёт

1	2	3	4
	ЗНАТЬ -стандарты, нормы и правила, а также техническую документацию, связанную с профессиональной деятельностью. УМЕТЬ -разрабатывать и использовать стандарты, нормы и правила, а также техническую документацию, связанную с профессиональной деятельностью. ВЛАДЕТЬ - навыками разработки и использования стандартов, норм и правил, а также технической документации, связанной с профессиональной деятельностью.		
ОПК-9 (09.03.04) Способен анализировать, разрабатывать, внедрять и выполнять организационно-технические и экономические процессы с применением технологий и систем искусственного интеллекта	**ОПК-9.1. Использует знание рынка информационных систем и информационно-коммуникационных технологий, методов математического моделирования и искусственного интеллекта для анализа и разработки организационно-технических и экономических процессов** ЗНАТЬ - рынок информационных систем и информационно-коммуникационных технологий, автоматизирующих организационно-технические и экономические процессы. УМЕТЬ - выбирать рациональные решения в области информационных технологий и систем искусственного интеллекта при построении организационно-технических и экономических процессов. ВЛАДЕТЬ - знаниями по используемым информационно-коммуникационным технологиям, методам математического моделирования и искусственного интеллекта для анализа и разработки организационно-технических и экономических процессов	1	Зачёт

1	2	3	4
	ОПК-9.2. Решает задачи по построению организационно-технических и экономических процессов с применением информационных технологий и систем искусственного интеллекта ЗНАТЬ - способы моделирования и построения организационно-технических и экономических процессов с использованием информационно-коммуникационных технологий и систем искусственного интеллекта. УМЕТЬ - разрабатывать и внедрять организационно-технические и экономические процессы с применением информационных технологий и систем искусственного интеллекта. ВЛАДЕТЬ - навыками решения задач с применением информационных технологий и систем искусственного интеллекта для построения организационно-технических и экономических процессов.		

Преподавателем оценивается содержательная сторона и качество материалов, приведенных в отчетах студента по практическим занятиям. Кроме того, преподавателем учитываются ответы студента на вопросы по соответствующим видам занятий при текущем контроле:

– контрольные опросы;

– задания для практических занятий.

Принимается во внимание **знания** обучающимися:

- понятия и стандарта программной инженерии;
- основных этапов разработки программ, их назначения и характеристики;
- систем управления версиями, системы отслеживания ошибок;
- подходов к проектированию программного обеспечения;
- структурного проектирования и программирования;
- конфликтов и способов их устранения;
- правил декомпозиции.

наличие **умений**:

- обнаруживать ошибки в программе;
- отслеживать ошибки в программе;
- проектировать, реализовывать и тестировать программное обеспечение различного масштаба;
- тестировать ПО различными способами

обладание навыками**:**

- использования программы GitLab;
- использования современных инструментов и технологий программной инженерии в разработке ПО
- участия в работе совместной команды разработчиков в области программной инженерии

Критерии оценивания уровня сформированности компетенции в процессе выполнения практических работ:

41%-60% правильных ответов соответствует пороговому уровню сформированности компетенции на данном этапе ее формирования;

61%-80% правильных ответов соответствует продвинутому уровню сформированности компетенции на данном этапе ее формирования;

81%-100% правильных ответов соответствует эталонному уровню сформированности компетенции на данном этапе ее формирования.

Сформированность уровня компетенций не ниже порогового является основанием для допуска обучающегося к промежуточной аттестации по данной дисциплине.

Формой промежуточной аттестации по данной дисциплине является зачет, оцениваемый по принятой в ФГБОУ ВО «РГРТУ» системе: «зачтено» и «не зачтено».

Критерии оценивания промежуточной аттестации представлены в таблице.

Шкала оценивания	Критерии оценивания
«зачтено»	**оценки «зачтено»** заслуживает обучающийся, продемонстрировавший полное знание материала изученной дисциплины, усвоивший основную литературу, рекомендованную рабочей программой дисциплины; выполнивший все практические задания; показавший систематический характер знаний по дисциплине, ответивший на все вопросы билета или допустивший погрешность в ответе вопросы, но обладающий необходимыми знаниями для их устранения под руководством преподавателя;
«не зачтено»	**оценки «не зачтено»** заслуживает обучающийся, не выполнивший практические задания, продемонстрировавший серьезные пробелы в знаниях основного материала изученной дисциплины, не ответивший на все вопросы билета и дополнительные вопросы. Оценка «не зачтено» ставится обучающимся, которые не могут продолжить обучение по образовательной программе без дополнительных занятий по соответствующей дисциплине (формирования и развития компетенций, закрепленных за данной дисциплиной).

3. Типовые контрольные задания или иные материалы

ФОС по дисциплине содержит следующие оценочные средства, позволяющие оценить знания, умения и уровень приобретенных компетенций при текущем контроле и промежуточной аттестации, разбитые по модулям дисциплины:

- перечни экзаменационных вопросов;
- макеты билетов к экзамену.

Средства для оценки различных уровней формирования компетенций по категориям «знать», «уметь», «владеть» обеспечивают реализацию основных принципов кон-

троля, таких, как объективность и независимость, практико-ориентированность, междисциплинарность.

С учетом этого, контрольные вопросы (задания, задачи,) входящие в ФОС, для различных категорий и уровней освоения компетенций имеют следующий вид:

Уровень ЗНАТЬ

Дескрипторы	Пример задания из оценочного средства
правила разработки нормативных документов различного назначения	1. Что такое репозиторий? Что отличает репозиторий от сетевой папки, к которой имеют доступ все участники проекта? 2. Какие «смысловые части» выделяет Git в директории проекта? Для чего они нужны? 3. Дайте определение сторон функционального блока, используемого для визуального представления структуры программы в методологии IDEF0.
основные требования ГОСТов к составу и содержанию нормативных документов различного назначения	
основы управления проектами по созданию и развитию технологий и систем искусственного интеллекта на стадиях их жизненного цикла	
как решать задачи управления проектами по созданию и развитию технологий и систем искусственного, интеллекта на стадиях их жизненного цикла	

Уровень УМЕТЬ

Дескрипторы	Пример задания из оценочного средства
разрабатывать стандарты, инструкции, нормы, методические материалы и техническую документацию, связанную с профессиональной деятельностью	1. Найдите тест, на котором программа будет работать неверно. Оформите его как функцию, которая не принимает никаких параметров, а возвращает пару вида: список, количество элементов. Добавьте вызов новой тестовой функции. Сохраните ваши изменения. 2. Создайте отдельную ветвь для исправления ошибки (для этого используется команда git branch). Ветвь должна называться fix. Переключитесь на нее. 3. Выполните декомпозицию нижеприведенной задачи на подзадачи. Результат декомпозиции представьте в формате IDEF0-диаграмм, используя приложение Ramus Educational. Диаграммы оформите в соответствии с требованиями стандарта IDEF0.
разрабатывать и внедрять организационно-технические и экономические процессы с применением информационных технологий и систем ИИ	
управлять проектами по созданию и развитию технологий и систем искусственного интеллекта на стадиях их жизненного цикла	
решать задачи управления проектами по созданию и развитию технологий и систем искусственного интеллекта на стадиях их жизненного цикла	

Перечни вопросов к зачёт и макеты зачётного билета

2 семестр

Перечень лабораторных работ

1.1 Система управления версиями Git

Цель работы: знакомство с системой управления версиями Git.

Задание:

1. Запустите командную оболочку MSYS.
2. Создайте отдельную папку для выполнения лабораторной работы.
3. Создайте локальный репозиторий, воспользовавшись командой **“git init”.**
4. С помощью команды **git config --list** проверьте параметры конфигурации. Если имя и адрес электронной почты не заданы, установите их с помощью переменных **user.name** и **user.email**.
5. Поместите в папку для выполнения лабораторной работы исходный код программы из архива. Убедитесь в том, что программа работает на тестовом примере.
6. В папке рабочей копии с помощью текстового редактора Notepad++ создайте файл, который будет содержать строку: ****/__pycache__/**** и сохраните его под именем **.gitignore** (имя файла начинается с точки, никакого расширения у файла быть не должно).
7. Проанализируйте состояние репозитория, используя команду **git status.**
8. Поставьте файл **.gitignore** под версионный контроль с помощью команды **git add** и зафиксируйте изменения командой **git commit –m “gitignore was added.”**
9. Добавьте под версионный контроль саму программу **(**файлы **iarray.py** и **main.py)** и зафиксируйте изменения командой **git commit –m “Initial version of program was added.”**
10. Опишите назначение программы и каждой ее функции.
11. Найдите тест, на котором программа будет работать неверно. Оформите его как функцию, которая не принимает никаких параметров, а возвращает пару список, количество элементов. Добавьте вызов новой тестовой функции (вызов старой тестовой процедуры оставьте, потому что количество тестов должно только увеличиваться). Сохраните ваши изменения.
12. Проанализируйте ваши изменения с помощью команд **git status** и **git diff**. Результат анализа приведите в отчете.
13. Зафиксируйте ваши изменения с помощью команды **git commit**. В отчете приведите номер ревизии и комментарий, которым вы сопроводили фиксацию.

1.2 GitLab как система управления проектами по разработке ПО

Цель работы: Освоение возможностей системы GitLab для отслеживания ошибок в программном обеспечении и документирования проекта.

Задание:

Исходными данными для выполнения лабораторной работы является исходный код программы, которая решает некоторую задачу. Программа состоит из модуля, реализующего основные функции, и основной программы. Основная программа содержит функцию, которая формирует массив с тестовыми данными, и вызов функций из модуля. Программа успешно работает на простых тестовых данных, т.е. находится в том состоянии, когда ее можно поместить под версионный контроль.

В программе допущена ошибка. Эту ошибку вам надо найти и исправить.

1. Создайте отчет об ошибке (**issue**) в GitLab, заполнив поля “**Title**” (заголовок) и “**Description**” (полное описание). В полном описании обязательно укажите номер ревизии, в которой ошибка была найдена.
2. Назначьте ошибку на себя.
3. Исправьте ошибку и убедитесь в правильности ваших исправлений.
4. Перед фиксацией изменений проанализируйте их с помощью команд **git status** и **git diff**. Результаты анализа приведите в отчете.
5. Зафиксируйте ваши изменения командой **git commit**.
6. Закройте отчет об ошибке. При этом в комментарии укажите суть исправлений, приведите номер ревизии.
7. Проанализируйте историю изменений в репозитории с помощью команды **git log**, команды **git log -- name-status**.
8. Сравните две версии одного и того же файла, но разных ревизий. Результаты сравнения зафиксируйте в отчете.
9. Создайте в GitLab wiki-страницу с отчетом по лабораторной работе.

1.3 Работа с ветвями в системе управления версиями Git

Цель работы: знакомство со схемами ветвления в *Git*. Приобретение навыков создания ветвей, переключения между ветвями и слияния ветвей.

Задание:

1. Распакуйте репозиторий.
2. Проанализируйте историю изменений в этом репозитории:
    a. Сколько ветвей в нем есть? Как они называются?
    b. Сколько пользователей работали с этим репозиторием? Какие имена у этих пользователей?
    c. Сколько файлов находится в репозитории? Кто и в какой последовательности вносил изменения в эти файлы?
3. Результаты анализа зафиксируйте в отчете.
4. Перенесите изменения из всех ветвей в ветвь master. Выполняя объединение изменений добейтесь расположения абзацев в логически правильной последовательности.

1.4 Конфликты и способы их устранения.

Целью работы является освоение навыков разрешения конфликтов, возникающих в системе управления версиями Git.

Задание:

При выполнении данного задания моделируется ситуация, когда вы одновременно работаете над двумя лабораторными работами, а первой принимают лабораторную работу, mege request для которой был создан позже. В отчете приведите подробные ответы с обоснованием причин возникновения конфликта и способа его разрешения.

1. Получите копию удаленного репозитория.
2. Перейдите в рабочую директорию.
3. На основе ветви **master** создайте ветви **lab_04_a** и **lab_04_b.**
4. Переключитесь на ветвь **lab_04_a.**
5. Добавьте под версионный контроль:

a. текстовый файл **lab_04_a.txt**, который содержит текст “lab_04_a”;

b. файл .**gitignore** для игнорирования исполняемых файлов.

6. Отправьте изменения в удаленный репозиторий и создайте merge request с названием «A».
7. Переключитесь на ветвь **lab_04_b**.
8. Добавьте под версионный контроль:

a. текстовый файл **lab_04_b.txt**, который содержит текст “lab_04_b”;

b. файл .**gitignore** для игнорирования объектных файлов.

9. Отправьте изменения в удаленный репозиторий и создайте merge request с названием «B».
10. Попросите преподавателя принять merge request B.
11. Проанализируйте состояние merge request A. Объясните, почему возник конфликт.
12. Разрешите конфликт.

1.5 Программные средства визуального проектирования программ. Нотация IDEF0

Целью лабораторной работы является знакомство студентов с методологией IDEF0 для проведения декомпозиции сложной задачи на подзадачи.

Задание:

1. Выполните декомпозицию нижеприведенной задачи на подзадачи.
2. Результат декомпозиции представьте в формате IDEF0-диаграмм, используя приложение Ramus Educational. Диаграммы оформите в соответствии с требованиями стандарта IDEF0.
3. Файлы *.rsf (результат работы Ramus Educational) поместите в репозиторий в папку **Lab_05**.
4. В Wiki создайте отдельную страницу, на которой разместите эти же решения, но уже как png-файлы (пункт меню «Диаграмма» / «Экспортировать как рисунки» в Ramus Educational).

Условие задачи: Исправить ошибку в программе по ее указанному описанию. Входными данными являются адрес репозитория Git и номер ошибки, указанный в Issue в GitLab, выходными – измененный репозиторий и закрытая ошибка.

1.6 Структурное проектирование. Правила декомпозиции, используемые при проектировании.

Целью лабораторной работы является освоение навыков проектирования «сверху-вниз», разработки внешних спецификаций модулей и формулирования допущений.

Задание: Для задачи, условие которой приведено ниже, выполните следующие действия:

1. Выполните декомпозицию нижеприведенной задачи на подзадачи. Результат

декомпозиции представьте в формате IDEF0-диаграмм.

2. Для каждой выделенной подзадачи разработайте спецификацию, используя следующий формат:
    a. имя модуля;
    b. функция модуля;
    c. список параметров;
    d. входные данные;
    e. выходные данные;
    f. внешние эффекты.
3. Сформулируйте допущения.

Условие задачи: Задан текстовый файл, на каждой строке которого располагаются целые числа. Необходимо вывести на печать те строки этого файла, которые содержат хотя бы одно число Фибоначчи.

1.7 Структурное программирование

Цель работы: освоение навыков оформления логически завершенных фрагментов алгоритма в виде подпрограмм.

Задание:

1. Выполните декомпозицию задачи на подзадачи. Результат декомпозиции представьте в формате IDEF0-диаграмм.
2. Для каждой выделенной подзадачи разработайте прототип функции для ее реализации.
3. Предложите несколько вариантов интерфейса для обмена данными между функцией main() и каждой подпрограммой.
4. Проанализируйте, где целесообразнее проводить проверку корректности данных, в подпрограмме или в функции main(). Результаты анализа – преимущества и недостатки каждого подхода, отразите в отчете.

Условие задачи: Задан текстовый файл, на каждой строке которого располагаются целые числа. Вывести на печать числа-палиндромы.

1.8 Псевдокод как способ описания алгоритма.

Цель работы: изучение различных нотаций описания алгоритмов решения задач.

Задание:

1. Для задачи из лабораторной работы №7 разработайте и запишите на псевдокоде алгоритм решения для каждой подзадачи в соответствии с произведенной декомпозицией.
2. Разработайте псевдокод для основной программы, указав входные и выходные данные и форматы их представления.
3. Напишите программу на языке С для решения данной задачи на основе подготовленного псевдокода.

2.1 Модульное тестирование ПО. Тестирование «черным ящиком».

Цель работы: изучение метода «черного ящика» для тестирования программ.

Задание: При выполнении данного задания используется условие задачи и схема декомпозиции из лабораторной работы №6.

1. Подготовьте тестовые данные для каждой подпрограммы, предварительно выделив классы эквивалентности.
2. Тестовые наборы для каждого класса эквивалентности дополните данными на основе метода анализа граничных значений.
3. В соответствии с допущениями, сделанными в лабораторной работе № 6, разработайте тесты, демонстрирующие корректность работы программы в целом.
4. Создайте отдельный проект для автоматизированного тестирования программы на основе разработанных тестовых данных.

2.2 Модульное тестирование ПО. Тестирование «белым ящиком».

Цель работы: изучение метода «белого ящика» для тестирования программ.

Задание: При выполнении данного задания используется условие задачи и схема декомпозиции из лабораторной работы №6.

1. Для любой из выделенных подзадач подготовьте тестовые данные, используя метод покрытия операторов.
2. Дополните предложенный тестовый набор данными на основе метода покрытия решений.
3. Расширьте тестовые данные за счет применения метода покрытия условий.
4. Выполните программу с использованием разработанных тестовых данных и оцените полноту тестового покрытия с помощью утилиты **Gcov**. Вывод относительно полноты тестового покрытия отразите в отчете.

2.3 Функциональное тестирование ПО.

Цель работы: освоение навыков автоматизации функционального тестирования с использованием командных пакетных файлов.

Задание: При выполнении данного задания используется условие задачи и схема декомпозиции из лабораторной работы №6.

1. Используя тестовые данные, разработанные в лабораторной работе №10 для демонстрации корректности работы программы в целом, подготовьте входные и выходные файлы для перенаправления ввода-вывода.

2.4 Bat-файлы.

Цель работы: освоение навыков автоматизации функционального тестирования с использованием командных пакетных файлов.

Задание: При выполнении данного задания используется условие задачи и схема декомпозиции из лабораторной работы №6.

1. Создайте bat-файл для автоматизированного выполнения функциональных тестов.
2. Проверьте работоспособность программы на заданном наборе тестовых данных. Сделанные выводы отразите в отчете.

2.5 Защитное программирование. Обработчики ошибок.

Цель работы: изучение методов защитного программирования и приобретение навыков их использования при разработке программного обеспечения.

Задание: При выполнении данного задания используется условие задачи и схема декомпозиции из лабораторной работы №7.

1. Определите внешние и внутренние данные, корректность которых должна проверяться при выполнении программы.
2. Разработайте сценарии поведения системы в случае обнаружения неверных данных.
3. Добавьте в программный код обработчики ошибок и оцените правильность их работы.
4. Сделанные выводы относительно предложенных сценариев обработки ошибок отразите в отчете.

2.6 Защитное программирование. Утверждения (Assert-ы) и правила их использования.

Цель работы: изучение методов защитного программирования и приобретение навыков их использования при разработке программного обеспечения.

Задание: При выполнении данного задания используется условие задачи и схема декомпозиции из лабораторной работы №7.

1. Определите те подзадачи (функции), при реализации которых целесообразно использовать утверждения для проверки сделанных внутренних допущений.
2. Добавьте утверждения в программный код выделенных функций и оцените правильность их работы.

2.7 Защитное программирование. Условная компиляция.

Цель работы: изучение методов защитного программирования и приобретение навыков их использования при разработке программного обеспечения.

Задание: При выполнении данного задания используется условие задачи и схема декомпозиции из лабораторной работы №7.

1. Используйте директивы условной компиляции для создания отладочной и релизной версии программного кода.
2. Результаты работы отразите в отчете.

2.8 Документирование программ

Цель работы: изучение функциональных возможностей системы Doxygen для автоматической генерации документации к разработанному программному обеспечению.

Задание: При выполнении данного задания используется условие задачи и программный код из лабораторной работы №7.

1. Изучите правила инсталляции и использования системы Doxygen.

2. Добавьте комментарии Doxygen в программный код, используя специальные параметры для краткого и подробного описания выделенных подзадач и используемых ими данных.
3. Используя Doxygen в качестве генератора документации, создайте html-файл с текстом документа и разместите его в GitLab.
4. Результаты работы отразите в отчете.

2.9 Выбор модели жизненного цикла ПО

Цель работы: приобретение навыков обоснования модели жизненного цикла на основе характеристик программного продукта.

Задание: Для задачи, условие которой приведено ниже, обоснуйте модель жизненного цикла.

1. Проведите выбор модели жизненного цикла на основе характеристик требований.
2. Проведите выбор модели жизненного цикла на основе характеристик команды разработчиков.
3. Проведите выбор модели жизненного цикла на основе характеристик коллектива пользователей.
4. Проведите выбор модели жизненного цикла на основе характеристик типа проекта и рисков.
5. Обоснуйте итоговый выбор модели жизненного цикла, приведите не менее 5-ти аргументов, подтверждающих правильность вашего выбора, и результаты отразите в отчете.

Условие задачи: Постановка задачи: В связи с переходом на новую программную платформу логистическая компания планирует внести изменения в свою информационную систему в части баз данных и учета складских запасов. Остальные модули программы останутся неизменными. Для выполнения этой задачи планируется привлечь команду высокопрофессиональных программистов из 6 человек, которые разрабатывали предыдущую версию системы. Какую модель жизненного цикла можно было бы применить при разработке системы?

-