Министерство науки и высшего образования
Российской Федерации
Министерство образования и науки Кыргызской Республики

Государственное образовательное учреждение
высшего профессионального образования
Кыргызско-Российский Славянский университет имени первого президента
Российской Федерации Б.Н. Ельцина

Естественно-технический факультет

Кафедра Информационных и вычислительных технологий

Фонд
оценочных средств

по дисциплине «Современные языки программирования»

Уровень высшего образования

БАКАЛАВРИАТ

Направление подготовки
15.03.03 - РФ, 650500 - КР Прикладная механика
(код и наименование направления подготовки)

Квалификация
бакалавр

Фонд оценочных средств предназначен для контроля знаний обучающихся по направлению подготовки 15.03.03 - РФ, 650500 - КР Прикладная механика по дисциплине «Современные языки программирования»

Фонд оценочных средств рассмотрен и утвержден на заседании кафедры Информационных и вычислительных технологий

Заведующий кафедрой
д.т.н., проф. Лыченко Н.М.

Исполнитель (разработчик):

ст. преподаватель Джалилова Т.Я.

СОГЛАСОВАНО:
и.о. декана Н.М. Комарцов

Раздел 1. Перечень компетенций, с указанием этапов их формирования в процессе освоения дисциплины/практики

Формируемые компетенции	Планируемые результаты обучения по дисциплине, характеризующие этапы формирования компетенций	Виды оценочных средств/ шифр раздела в данном документе
ОПК-14: Способен разрабатывать алгоритмы и компьютерные программы, пригодные для практического применения.	**<u>Знать:</u>** Синтаксис Python и стандарты оформления кода (PEP 8). Алгоритмы численных методов, ветвления, циклы и структуры данных. Основы ООП для моделирования механических систем. Принципы построения графических интерфейсов (GUI).	**Блок А –** задания репродуктивного уровня - Устный опрос - Контрольная работа - Тестирование
	<u>Уметь</u>: Формализовать инженерные задачи в виде программного кода. Создавать расчетные модули с автоматизацией ввода-вывода (файлы TXT/CSV). Разрабатывать интерактивные приложения с оконным интерфейсом. Отлаживать и оптимизировать код для вычислений.	**Блок В** – задания реконструктивного уровня - Практическая контрольная работа
	<u>Владеть:</u> Навыками разработки прикладного ПО для задач вычислительной механики. Методикой модульного проектирования расчетных систем. Техникой интеграции Python-скриптов в инженерные CAE-пакеты. Культурой разработки надежного и документированного кода.	**Блок С** – задания практико-ориентированного и/или исследовательского уровня - Практические задания

Компетенции обучающегося, формируемые в результате освоения дисциплины

В результате освоения дисциплины (модуля) обучающийся должен

Знать

Синтаксис и библиотеки: базовые конструкции Python, стандартную библиотеку и основы научных пакетов (NumPy).

Алгоритмизацию: методы построения линейных, разветвляющихся и циклических алгоритмов для инженерных задач.

ООП и GUI: принципы объектно-ориентированного подхода и теорию построения графических интерфейсов пользователя.

Уметь

Программировать: переводить математические модели механики в программный код.

Работать с данными: автоматизировать чтение и запись данных через файлы (TXT, CSV) для связи с CAE-системами.

Создавать интерфейсы: разрабатывать оконные приложения для визуализации и управления расчетами.

Иметь навыки и (или) опыт деятельности

Разработки ПО: создания практически применимых программ для вычислительной механики и инжиниринга.

Отладки: поиска и устранения логических и синтаксических ошибок в расчетном коде.

Проектирования: использования ООП для создания расширяемых иерархий физических объектов и материалов.

Автоматизации инженерного анализа: написания скриптов для пакетной обработки результатов конечно-элементного моделирования.

Работы в IDE: использования сред разработки для контроля версий и документирования инженерного ПО.

Раздел 2. Технологическая карта дисциплины

Дисциплина:	Современные языки программирования
Группа:	ЕМ-1-23
Курс/семестр:	3/6
Количество кредитов (ЗЕ):	3
Отчетность:	**Зачет с оценкой**

Название модулей дисциплины согласно РПД	**Контроль**	**Форма контроля**	**зачетный минимум**	**зачетный максимум**	**график контроля**
Модуль 1					
Основы синтаксиса и алгоритмические конструкции	Текущий контроль	Посещение занятий. Активность. Решение типовых задач - Выполнение практических заданий.	5	10	30

	Рубежный контроль	Контрольная работа.	5	10	
Модуль 2					
Коллекции, функции и основы ООП	Текущий контроль	Посещение занятий. Активность. Решение типовых задач - Выполнение практических заданий.	10	15	34
	Рубежный контроль	Контрольная работа	5	10	
Модуль 3					
Графический интерфейс (GUI) и работа с файлами	Текущий контроль	Посещение занятий. Активность. Решение типовых задач - Выполнение практических заданий.	10	15	38
	Рубежный контроль	Контрольная работа	5	10	
ВСЕГО за семестр			40	70	
Промежуточный контроль (Зачет с оценкой)			20	30	
Семестровый рейтинг по дисциплине			60	100	

Раздел 3. Типовые контрольные задания и иные материалы, необходимые для оценки планируемых результатов обучения по дисциплине (оценочные средства)

Вопросы для оценки качества освоения дисциплины.

Модуль 1. Основы синтаксиса и алгоритмические конструкции

Знать:

1. В чем принципиальное различие между компилируемыми (C++) и интерпретируемыми (Python) языками?
2. Какие типы данных в Python являются изменяемыми, а какие — нет?
3. Как работают операторы деления (`/`), целочисленного деления (`//`) и остатка (`%`)?
4. Каков порядок приоритета арифметических и логических операций?

5. Что такое динамическая типизация и как она влияет на выделение памяти под переменные?
6. Каков синтаксис и правила использования функции `range()` в циклах?
7. В чем разница между операторами сравнения == и оператором идентичности `is`?
8. Как работают инструкции `break` и `continue` в сложных циклах?
9. Что такое «магические числа» и почему их следует заменять константами?
10. Какие стандарты оформления кода (PEP 8) являются обязательными для читаемости программы?

Уметь:

1. Реализовать ввод данных с клавиатуры и приведение их к вещественному типу (`float`).
2. Составить программу для решения квадратного уравнения с обработкой отрицательного дискриминанта.
3. Оформить вывод результатов расчета с использованием f-строк (округление до нужного знака).
4. Написать цикл для вычисления суммы ряда (например, разложение функции в ряд Тейлора).
5. Использовать математические функции модуля `math` (sin, cos, exp) в расчетах.
6. Реализовать проверку входных параметров на физическую корректность (например, площадь > 0).
7. Написать вложенный цикл для вывода таблицы значений функции двух переменных.
8. Преобразовать строковые данные, содержащие числа, в список числовых значений.
9. Создать алгоритм поиска наибольшего общего делителя для оптимизации шага сетки.
10. Использовать тернарный оператор для краткой записи простых условий.

Владеть:

1. Навыками отладки линейных программ в современных IDE (PyCharm, VS Code).
2. Методикой перевода сложных математических формул в безошибочный программный код.
3. Техникой безопасного ввода данных с защитой от некорректных типов.
4. Приемами оптимизации логических условий в разветвляющихся алгоритмах.
5. Навыками использования векторизованных операций (на базовом уровне) для замены простых циклов.
6. Способностью оценивать точность вычислений при работе с числами с плавающей точкой.
7. Методикой построения блок-схем для визуализации алгоритмов инженерных задач.
8. Навыками документирования кода через аннотации типов и комментарии.
9. Техникой работы с константами для управления глобальными параметрами модели.
10. Навыками самоконтроля кода на соответствие стандартам PEP 8.

Модуль 2. Коллекции, функции и основы ООП

Знать:

1. В чем различие между списком (`list`) и кортежем (`tuple`) в контексте использования памяти?
2. Как работает механизм «срезов» (`slices`) в списках и строках?
3. Что такое словарь (`dict`) и почему он эффективен для хранения свойств материалов?
4. Какова роль аргумента `self` в методах класса Python?
5. Что такое рекурсия и в каких задачах механики (например, фракталы или деревья) она применима?
6. В чем разница между локальными и глобальными переменными (правило LEGB)?
7. Что такое Lambda-функции и когда их использование оправдано?
8. Для чего используется метод `__init__` в классах?
9. Каковы принципы инкапсуляции данных в Python (публичные и приватные атрибуты)?
10. Как работают функции высшего порядка `map()`, `filter()` и `zip()`?

Уметь:

1. Создать список координат узлов расчетной сетки и выполнить его обход.
2. Реализовать функцию, принимающую произвольное число аргументов (`*args`, `**kwargs`).
3. Разработать класс «Материал» с атрибутами (модуль Юнга, коэффициент Пуассона).
4. Использовать генераторы списков (`list comprehensions`) для создания матриц.
5. Написать рекурсивную функцию для расчета чисел Фибоначчи или факториала.
6. Реализовать метод класса, вычисляющий напряжение по заданным силе и площади.
7. Выполнить сортировку списка объектов по заданному критерию (например, по цене или массе).
8. Создать словарь, связывающий ID узла с его координатами.
9. Написать функцию cDocstrings, описывающими входные параметры и возвращаемое значение.
10. Использовать множества (`set`) для поиска уникальных элементов в массиве данных.

Владеть:

1. Методикой декомпозиции инженерной задачи на отдельные функции и классы.
2. Навыками объектно-ориентированного проектирования простейших механических систем.
3. Техникой работы со вложенными структурами данных (списки списков для матриц).
4. Навыками создания модульных программ, состоящих из нескольких файлов.
5. Приемами функционального программирования для обработки массивов данных.
6. Навыками управления состоянием объектов в процессе выполнения программы.
7. Техникой реализации интерфейсов взаимодействия между объектами классов.
8. Методами профилирования функций для выявления «узких мест» в расчетах.
9. Опытом работы с итераторами и генераторами для экономии памяти.
10. Навыками применения паттернов проектирования (на базовом уровне) в ООП.

Модуль 3. Графический интерфейс (GUI) и работа с файлами

Знать:

1. Какова иерархия обработки исключений (`try-except-finally`) в Python?
2. Какие режимы открытия файлов существуют (`r`, `w`, `a`, `b`)?
3. Что такое контекстный менеджер `with` и почему он обязателен при работе с файлами?
4. В чем разница между менеджерами геометрии `pack`, `grid` и `place` в Tkinter?
5. Что такое «событийно-ориентированное программирование» в контексте GUI?
6. Как реализовать механизм наследования классов для расширения функционала виджетов?
7. Как работают переменные управления (`StringVar`, `IntVar`) в графическом интерфейсе?
8. Что такое сериализация данных и форматы CSV/JSON?
9. Как организовать диалоговое окно для выбора файла на диске?
10. В чем преимущество Модульения логики расчета и кода интерфейса (паттерн MVC)?

Уметь:

1. Написать программу, считывающую матрицу жесткости из текстового файла.
2. Создать графическое окно с полями ввода для параметров балки и кнопкой расчета.
3. Реализовать обработку ошибки `ZeroDivisionError` при расчетах через GUI.
4. Организовать запись результатов моделирования в CSV-файл для Excel.
5. Создать выпадающий список (OptionMenu/Combobox) для выбора типа материала.
6. Написать обработчик события нажатия кнопки, который запускает расчетный модуль.
7. Создать класс-исключение для специфических инженерных ошибок.
8. Реализовать графическую метку (Label), цвет которой меняется в зависимости от результата.
9. Организовать чтение данных из JSON-файла конфигурации проекта.
10. Разместить элементы интерфейса в виде таблицы (Grid) для создания калькулятора.

Владеть:

1. Навыками проектирования пользовательских интерфейсов для инженерных расчетов.
2. Техникой создания отказоустойчивых программных комплексов.
3. Навыками интеграции Python-скриптов с внешними данными (например, экспорт в CAE).
4. Методикой тестирования интерфейса на корректность пользовательского ввода.
5. Навыками работы с бинарными файлами для хранения больших объемов данных.
6. Техникой визуализации прогресса вычислений в графическом окне.
7. Опытом разработки кроссплатформенных приложений на Python.
8. Навыками связывания сложной логики на базе ООП с элементами управления GUI.
9. Техникой обработки системных прерываний и ошибок ввода-вывода.
10. Методами подготовки программного продукта к передаче конечному пользователю.

Раздел 1. Основы синтаксиса и алгоритмические конструкции

Теоретические вопросы:

1. Особенности управления памятью в Python: как работает счетчик ссылок и сборщик мусора?

2. Сравнение точности вычислений типов `float` и `decimal` при решении инженерных задач.
3. Логические операторы и «ленивые вычисления» (short-circuit evaluation) в Python.
4. Стандарты оформления кода PEP 8: основные правила именования переменных и структур программы.

Практические задания:

1. **Расчет физических величин:** Написать программу для перевода мер давления (Паскали, Бары, Технические атмосферы, PSI) друг в друга с использованием форматированного вывода.
2. **Алгоритм итераций:** Реализовать нахождение квадратного корня числа методом итераций Ньютона без использования встроенного модуля `math`.
3. **Анализ условий:** Написать алгоритм, определяющий вид напряженного состояния в точке по введенным главным напряжениям (растяжение, сжатие, чистый сдвиг).

Раздел 2. Структуры данных, функции и основы ООП

Теоретические вопросы:

1. Механизм работы хэш-таблиц в словарях (`dict`): почему ключи должны быть неизменяемыми?
2. Отличия между «глубоким» (`copy.deepcopy`) и «поверхностным» копированием списков.
3. Понятие замыкания функции (closure) и область применения декораторов.
4. Принципы SOLID применительно к проектированию классов в инженерном ПО.

Практические задания:

1. **Работа с матрицами:** Реализовать сложение и умножение двух матриц (представленных как списки списков) без использования сторонних библиотек.
2. **Функциональное программирование:** Написать программу, которая с помощью `filter` и `lambda` выбирает из списка узлов сетки те, которые попадают в заданную геометрическую область (например, круг или прямоугольник).
3. **ООП Моделирование:** Разработать класс `Beam` (Балка). Параметры: длина, модуль упругости, момент инерции. Методы: расчет изгибной жесткости и определение прогиба под действием сосредоточенной силы.

Раздел 3. Графический интерфейс (GUI) и работа с файлами

Теоретические вопросы:

1. Жизненный цикл графического приложения: как работает событийная петля `mainloop()`?
2. Сравнение форматов хранения инженерных данных: CSV, JSON и XML — когда и что выбирать?
3. Иерархия стандартных исключений Python: для чего нужно создавать собственные классы ошибок (User-defined Exceptions)?
4. Методы сериализации объектов: использование модуля `pickle` для сохранения состояния расчетной модели.

Практические задания:

1. **Парсинг данных:** Написать скрипт, который считывает из текстового файла координаты точек (формат: `x;y;z`), вычисляет расстояние от начала координат до каждой точки и записывает результат в новый файл.
2. **Разработка интерфейса:** Создать окно на `Tkinter` для подбора параметров материала. Пользователь выбирает название из выпадающего списка (`Combobox`), а программа отображает в метках его физические свойства из заранее подготовленного JSON-файла.
3. **Обработка ошибок в GUI:** Дополнить расчетную программу блоками `try-except`, которые выводят предупреждающее окно (`messagebox`), если пользователь ввел отрицательное значение модуля Юнга или оставил поля пустыми.

Раздел 1. Основы синтаксиса и алгоритмические конструкции

Теоретические вопросы:

1. Особенности управления памятью в Python: как работает счетчик ссылок и сборщик мусора?
2. Сравнение точности вычислений типов `float` и `decimal` при решении инженерных задач.
3. Логические операторы и «ленивые вычисления» (short-circuit evaluation) в Python.
4. Стандарты оформления кода PEP 8: основные правила именования переменных и структур программы.

Практические задания:

1. **Расчет физических величин:** Написать программу для перевода мер давления (Паскали, Бары, Технические атмосферы, PSI) друг в друга с использованием форматированного вывода.
2. **Алгоритм итераций:** Реализовать нахождение квадратного корня числа методом итераций Ньютона без использования встроенного модуля `math`.
3. **Анализ условий:** Написать алгоритм, определяющий вид напряженного состояния в точке по введенным главным напряжениям (растяжение, сжатие, чистый сдвиг).

Раздел 2. Структуры данных, функции и основы ООП

Теоретические вопросы:

1. Механизм работы хэш-таблиц в словарях (`dict`): почему ключи должны быть неизменяемыми?
2. Отличия между «глубоким» (`copy.deepcopy`) и «поверхностным» копированием списков.
3. Понятие замыкания функции (closure) и область применения декораторов.
4. Принципы SOLID применительно к проектированию классов в инженерном ПО.

Практические задания:

1. **Работа с матрицами:** Реализовать сложение и умножение двух матриц (представленных как списки списков) без использованя сторонних библиотек.

2. **Функциональное программирование:** Написать программу, которая с помощью `filter` и `lambda` выбирает из списка узлов сетки те, которые попадают в заданную геометрическую область (например, круг или прямоугольник).
3. **ООП Моделирование:** Разработать класс `Beam` (Балка). Параметры: длина, модуль упругости, момент инерции. Методы: расчет изгибной жесткости и определение прогиба под действием сосредоточенной силы.

Раздел 3. Графический интерфейс (GUI) и работа с файлами

Теоретические вопросы:

1. Жизненный цикл графического приложения: как работает событийная петля `mainloop()`?
2. Сравнение форматов хранения инженерных данных: CSV, JSON и XML — когда и что выбирать?
3. Иерархия стандартных исключений Python: для чего нужно создавать собственные классы ошибок (User-defined Exceptions)?
4. Методы сериализации объектов: использование модуля `pickle` для сохранения состояния расчетной модели.

Практические задания:

1. **Парсинг данных:** Написать скрипт, который считывает из текстового файла координаты точек (формат: `x;y;z`), вычисляет расстояние от начала координат до каждой точки и записывает результат в новый файл.
2. **Разработка интерфейса:** Создать окно на `Tkinter` для подбора параметров материала. Пользователь выбирает название из выпадающего списка (`Combobox`), а программа отображает в метках его физические свойства из заранее подготовленного JSON-файла.
3. **Обработка ошибок в GUI:** Дополнить расчетную программу блоками `try-except`, которые выводят предупреждающее окно (`messagebox`), если пользователь ввел отрицательное значение модуля Юнга или оставил поля пустыми.

Рекомендуемые темы рефератов/докладов для СРС:

- «Сравнение производительности Python и C++ в задачах вычислительной механики».
- «Обзор библиотек Python для визуализации научных данных (Matplotlib, Plotly)».
- «Автоматизация создания расчетных сеток с использованием Python-скриптов».
- «Применение Python для написания макросов в CAE-системах (Abaqus, ANSYS)».

Примерные вопросы и задания для самостоятельных работ:

Раздел 1. Основы синтаксиса и алгоритмические конструкции

Теоретические вопросы:

1. Особенности управления памятью в Python: как работает счетчик ссылок и сборщик мусора?
2. Сравнение точности вычислений типов `float` и `decimal` при решении инженерных задач.
3. Логические операторы и «ленивые вычисления» (short-circuit evaluation) в Python.
4. Стандарты оформления кода PEP 8: основные правила именования переменных и структур программы.

Практические задания:

1. **Расчет физических величин:** Написать программу для перевода мер давления (Паскали, Бары, Технические атмосферы, PSI) друг в друга с использованием форматированного вывода.
2. **Алгоритм итераций:** Реализовать нахождение квадратного корня числа методом итераций Ньютона без использования встроенного модуля `math`.
3. **Анализ условий:** Написать алгоритм, определяющий вид напряженного состояния в точке по введенным главным напряжениям (растяжение, сжатие, чистый сдвиг).

Раздел 2. Структуры данных, функции и основы ООП

Теоретические вопросы:

1. Механизм работы хэш-таблиц в словарях (`dict`): почему ключи должны быть неизменяемыми?
2. Отличия между «глубоким» (`copy.deepcopy`) и «поверхностным» копированием списков.
3. Понятие замыкания функции (closure) и область применения декораторов.
4. Принципы SOLID применительно к проектированию классов в инженерном ПО.

Практические задания:

1. **Работа с матрицами:** Реализовать сложение и умножение двух матриц (представленных как списки списков) без использования сторонних библиотек.
2. **Функциональное программирование:** Написать программу, которая с помощью `filter` и `lambda` выбирает из списка узлов сетки те, которые попадают в заданную геометрическую область (например, круг или прямоугольник).
3. **ООП Моделирование:** Разработать класс `Beam` (Балка). Параметры: длина, модуль упругости, момент инерции. Методы: расчет изгибной жесткости и определение прогиба под действием сосредоточенной силы.

Раздел 3. Графический интерфейс (GUI) и работа с файлами

Теоретические вопросы:

1. Жизненный цикл графического приложения: как работает событийная петля `mainloop()`?
2. Сравнение форматов хранения инженерных данных: CSV, JSON и XML — когда и что выбирать?
3. Иерархия стандартных исключений Python: для чего нужно создавать собственные классы ошибок (User-defined Exceptions)?
4. Методы сериализации объектов: использование модуля `pickle` для сохранения состояния расчетной модели.

Практические задания:

1. **Парсинг данных:** Написать скрипт, который считывает из текстового файла координаты точек (формат: `x;y;z`), вычисляет расстояние от начала координат до каждой точки и записывает результат в новый файл.
2. **Разработка интерфейса:** Создать окно на `Tkinter` для подбора параметров материала. Пользователь выбирает название из выпадающего списка (`Combobox`), а программа отображает в метках его физические свойства из заранее подготовленного JSON-файла.
3. **Обработка ошибок в GUI:** Дополнить расчетную программу блоками `try-except`, которые выводят предупреждающее окно (`messagebox`), если пользователь ввел отрицательное значение модуля Юнга или оставил поля пустыми.

Рекомендуемые темы рефератов/докладов для СРС:

- «Сравнение производительности Python и C++ в задачах вычислительной механики».
- «Обзор библиотек Python для визуализации научных данных (Matplotlib, Plotly)».
- «Автоматизация создания расчетных сеток с использованием Python-скриптов».
- «Применение Python для написания макросов в CAE-системах (Abaqus, ANSYS)».

ПЕРЕЧЕНЬ ВОПРОСОВ К ЗАЧЕТУ

1. История возникновения и философия языка Python. Преимущества и недостатки перед компилируемыми языками (C++).
2. Интерпретируемость и динамическая типизация: понятия и особенности реализации в Python.
3. Основные типы данных (int, float, bool, str). Приведение типов и встроенные функции преобразования.
4. Арифметические операции в Python: особенности деления нацело (`//`) и нахождения остатка (`%`). Приоритеты операций.
5. Логические выражения и операторы сравнения. Особенности работы операторов `and`, `or`, `not`.
6. Условный оператор `if-elif-else`: синтаксис и правила вложенности.
7. Цикл с условием `while`. Бесконечные циклы и способы их предотвращения.
8. Цикл со счетчиком `for` и функция `range()`. Итерирование по различным объектам.
9. Управляющие инструкции в циклах: `break`, `continue` и блок `else` для циклов.
10. Списки (`list`): создание, индексация и методы изменения (`append`, `extend`, `remove`, `pop`).
11. Срезы (slices) в списках и строках: синтаксис, шаг и использование для инверсии данных.
12. Кортежи (`tuple`): отличия от списков, преимущества использования и область применения.
13. Словари (`dict`): структура «ключ-значение», методы работы и правила формирования ключей.
14. Множества (`set`): теоретико-множественные операции (объединение, пересечение) и удаление дубликатов.
15. Понятие функции в Python. Определение `def`, передача аргументов и оператор `return`.
16. Позиционные и именованные аргументы функций. Значения аргументов по умолчанию.

17. Произвольное количество аргументов: использование `*args` и `**kwargs`.
18. Области видимости переменных: локальные, глобальные переменные и правило LEGB.
19. Анонимные функции (`lambda`): синтаксис и совместное использование с `map` и `filter`.
20. Генераторы списков (List Comprehensions) как эффективный инструмент создания коллекций.
21. Работа с текстовыми файлами: режимы открытия, методы `read`, `write` и контекстный менеджер `with`.
22. Обработка исключений: иерархия ошибок, блоки `try`, `except`, `finally` и `else`.
23. Понятие объектно-ориентированного программирования. Определение класса, объекта и экземпляра.
24. Конструктор класса `__init__` и назначение параметра `self`.
25. Инкапсуляция в Python: соглашения об именовании атрибутов (публичные, защищенные, приватные).
26. Наследование: создание дочерних классов и использование функции `super()`.
27. Полиморфизм в Python: переопределение методов и «утиная типизация».
28. Основы графического интерфейса (библиотека Tkinter): создание главного окна и цикл `mainloop`.
29. Основные виджеты GUI: `Label`, `Button`, `Entry`. Менеджеры геометрии `pack` и `grid`.
30. Событийно-ориентированное программирование: связывание функций с событиями виджетов.

Раздел 4. Методические материалы, определяющие процедуры оценивания знаний, умений, навыков и (или) опыта деятельности, характеризующих этапы формирования компетенций

Описание показателей и критериев оценивания компетенций, описание шкал оценивания

Применяемые оценочные средства:

- Сдача практических и контрольных работ, прохождение тестирования на практических занятиях в соответствии с технологической картой дисциплины (текущая и рубежная аттестация)
- Письменный опрос по экзаменационным билетам (промежуточная аттестация - зачет с оценкой),

Все виды оценочных средств оцениваются в соответствии со шкалами оценивания.

Устный опрос на практических занятиях по отдельным темам проводится в течение всего периода обучения дисциплине. Результаты опроса учитываются при оценивании практических работ.

ШКАЛА ОЦЕНИВАНИЯ УСТНОГО ОПРОСА

В рамках дисциплины «Дискретная математика» опрос проводится фронтальным методом в устной форме беседы с группой, сочетая его с повторением пройденной темы, как средство для закрепления знаний. Вопросы ставятся таким образом, чтобы ответ имел краткую форму, чтобы последующий вопрос был продолжением предыдущего, для того, чтобы раскрыть все вопросы изученной темы. В результате в активную умственную работу

вовлекаются почти все студенты группы, оценка ставится всем участвующим в обсуждении в зависимости от активности каждого и правильности и глубины ответов.

(**промежуточный контроль** – «ЗНАТЬ»)

Отметкой (7-10- баллов) оценивается ответ, который показывает прочные знания теоретических основ текущей темы дисциплины, понимание и правильное применение терминологии, правильные ответы на 75-100% вопросов

Отметкой (5-7 баллов) оценивается ответ, который показывает знание теоретических основ текущей темы дисциплины, но неполное понимание и не всегда правильное применение терминологии, даны правильные ответы на 50-74% вопросов, в ответах допущено некоторое количество неточностей.

Отметкой (3-4 баллов) оценивается ответ, свидетельствующий о знакомстве с некоторыми теоретическими основами текущей темы дисциплины. Даны правильные ответы на 25-49% вопросов, допущены неточности и ошибки.

Отметкой (2 балла) оценивается ответ, обнаруживающий незнание теоретических основ текущей темы дисциплины. Отмечается отсутствие логичности и последовательности в ответе. Менее 25% правильных ответов. Допущены серьезные ошибки в содержании ответа.

Отметкой (0-1 балл) оценивается ответ, при котором студент демонстрирует непонимание поставленных вопросов, или нет ответа.

ШКАЛА ОЦЕНИВАНИЯ ПРАКТИЧЕСКИХ ЗАДАНИЙ

(**промежуточный контроль** – «УМЕТЬ и ВЛАДЕТЬ»)

Отметкой (8-10 баллов) оценивается ответ, при котором студент правильно отвечает на поставленные вопросы, Демонстрирует полное понимание проблемы. Все требования, предъявляемые к заданию, выполнены.

Отметкой (5-7 баллов) оценивается ответ, при котором студент в основном правильно отвечает на поставленные вопросы. Демонстрирует значительное понимание проблемы. Большинство требований, предъявляемых к заданию, выполнены.

Отметкой (2-4 баллов) оценивается ответ, при котором студент в основном не правильно отвечает на поставленные вопросы. Демонстрирует частичное или небольшое понимание проблемы. Многие требования, предъявляемые к заданию, не выполнены.

Отметкой (0 -1 балл) оценивается ответ, при котором студент демонстрирует непонимание проблемы или нет ответа и даже не было попытки решить задачи.

ШКАЛА ОЦЕНИВАНИЯ ПРАКТИЧЕСКИХ РАБОТ

(текущий контроль)

- 85-100 % - Демонстрирует полное понимание проблемы. Все требования, предъявляемые к заданию, выполнены.
- 70-84 % - Демонстрирует значительное понимание проблемы. Все требования, предъявляемые к заданию, выполнены.
- 60-69 % - Демонстрирует частичное понимание проблемы. Большинство требований, предъявляемых к заданию выполнены.
- 31-60 % - Демонстрирует небольшое понимание проблемы. Многие требования, предъявляемые к заданию не выполнены.
- 0-30 % - Демонстрирует непонимание проблемы и даже не было попытки решить задачу.

ШКАЛА ОЦЕНИВАНИЯ КОНТРОЛЬНЫХ РАБОТ (рубежный контроль)

- 85-100 % - Демонстрирует полное понимание проблемы. Все задания выполнены.
- 70-84 % - Демонстрирует значительное понимание проблемы. Все задания выполнены, но содержат некоторые неточности.
- 60-69 % - Демонстрирует частичное понимание проблемы. Большинство требований, предъявляемых к заданию, выполнены.
- 31-60 % - Демонстрирует небольшое понимание проблемы. Многие требования, предъявляемые к заданию, не выполнены.
- 0-30 % - Демонстрирует непонимание проблемы или нет ответа и даже не было попытки решить задачу.

ШКАЛА ОЦЕНИВАНИЯ ТЕСТИРОВАНИЯ (рубежный контроль)

Тестирование проводится с помощью онлайн сервиса «Online Test Pad».
На тестирование отводится 40 минут. Каждый вариант тестовых заданий включает 4-5вопросов. За каждый правильный ответ на вопрос дается 0,15-0,17 баллов.

85 - 100 % – 5 баллов
70 - 84 % – 4 балла
60 - 69 % – 3 балла
менее 60 % – 2 балла

ШКАЛА ОЦЕНИВАНИЯ ПИСЬМЕННОГО ОПРОСА

(**промежуточный контроль** – «ЗНАТЬ»)
Отметкой (7-10- баллов) оценивается ответ, который показывает прочные знания теоретических основ дисциплины, понимание и правильное применение терминологии, правильные ответы на 75-100% вопросов
Отметкой (5-7 баллов) оценивается ответ, который показывает знание теоретических основ дисциплины, но неполное понимание и не всегда правильное применение терминологии, даны правильные ответы на 50-74% вопросов, в ответах допущено некоторое количество неточностей.
Отметкой (3-4 баллов) оценивается ответ, свидетельствующий о знакомстве с некоторыми теоретическими основами дисциплины. Даны правильные ответы на 25-49% вопросов, допущены неточности и ошибки.
Отметкой (2 балла) оценивается ответ, обнаруживающий незнание теоретических основ дисциплины. Отмечается отсутствие логичности и последовательности в ответе. Менее 25% правильных ответов. Допущены серьезные ошибки в содержании ответа.
Отметкой (0-1 балл) оценивается ответ, при котором студент демонстрирует непонимание поставленных вопросов, или нет ответа.

В экзаменационный билет включены два теоретических вопроса и одно практическое задание, соответствующие содержанию формируемых компетенций. Зачет с оценкой проводится в письменной форме. На ответ и решение задачи студенту отводится 80 минут. За ответ на теоретические вопросы студент может получить максимально 20 баллов, за выполнение практического задания - 10 баллов.

По итогам прохождения дисциплины и с учетом шкал оценивания все набранные в результате текущей, рубежной и промежуточной аттестаций баллы суммируются и выставляется оценка .

Перевод баллов в оценку:

85 - 100 баллов – «отлично»

70 - 84 баллов – «хорошо»

60 - 69 баллов – «удовлетворительно»

менее 60 баллов – «неудовлетворительно»

Раздел 5. Методические указания для обучающегося по освоению дисциплины и выполнению контрольных заданий

5.1. Общие рекомендации к организации самостоятельной работы

Текущий контроль осуществляется в течение семестра в виде защиты практических работ. Методические указания по выполнению практических работ представлены в электронной папке преподавателя (локальная сеть кафедры Информационных и вычислительных технологий КРСУ) и на корпоративной платформе MSTeams. Выполнение практических работ №№3-7 завершается оформлением отчета, в котором приводится теория по теме работы, блок-схемы алгоритмов, результаты расчетов и графики. Рубежный контроль осуществляется в виде сдачи контрольных работ. Вопросы по контрольным работам представлены в разделе 3 (блок А). Контрольные работы оформляются в письменном виде

5.2. Подготовка к практическим занятиям

Перед посещением практического занятия необходимо проработать конспект лекций по теме практического занятия. Оформление отчётов должно производиться по представленному образцу после окончания работы. Для подготовки к защите отчёта следует проанализировать экспериментальные результаты, обобщать результаты исследований в виде выводов, подготовить ответы на вопросы.