Федеральное государственное автономное образовательное учреждение высшего образования
«Санкт-Петербургский политехнический университет Петра Великого»

УТВЕРЖДАЮ
Директор ИКНК
_____ Д.П. Зегжда
«17» июня 2025 г.

РАБОЧАЯ ПРОГРАММА ДИСЦИПЛИНЫ (МОДУЛЯ)

«Введение в язык программирования Go»

Разработчик	Высшая школа программной инженерии
Направление (специальность) подготовки	09.03.04 Программная инженерия
Наименование ООП	09.03.04_01 Технология разработки и сопровождения качественного программного продукта
Квалификация (степень) выпускника	**бакалавр**
Образовательный стандарт	**СУОС**
Форма обучения	**Очная**

СОГЛАСОВАНО
Руководитель ОП
_____ А.В. Петров
«01» апреля 2025 г.

Соответствует СУОС
Утверждена протоколом заседания высшей школы "ВШПИ"
от «01» апреля 2025 г. № 1

РПД разработал:
Специалист по учебно-методической работе 1 категории Т.А. Вишневская

1. Цели и планируемые результаты изучения дисциплины

Цели освоения дисциплины

1. 1. Цель изучения дисциплины «Введение в язык программирования Go» направлена на формирование у обучающихся пониманий и знаний теоретических и практических аспектов в основе которых лежат принципы проектирования, разработки, тестирования и сопровождения современного программного обеспечения на языке высокого уровня Go;
2. 2. Также целью изучения дисциплины является подготовка квалифицированных выпускников, умеющих обосновано выбирать, использовать и внедрять библиотеки и сторонние решения, разработанные на языке Go

Результаты обучения выпускника

Код	Результат обучения (компетенция) выпускника ООП
ПК-5	**Способен разрабатывать программное обеспечение с использованием современных тенденций в области операционных систем, сетевых технологий, средств разработки программного интерфейса, систем управления базами данных**
ИД-8 ПК-5	Применяет современные инструментальные средства в процессе разработки и развертывания программного обеспечения

Планируемые результаты изучения дисциплины

знания:

- Знает типы инструментальных средств в процессе разработки и развертывания программного обеспечения

умения:

- Умеет выбирать инструментальные средства разработки программного обеспечения, адекватные решаемой задаче

навыки:

- Владеет навыками использования инструментальных средств разработки и развертывания программного обеспечения при решении профессиональных задач

2. Место дисциплины в структуре ООП

В учебном плане дисциплина «Введение в язык программирования Go» не связана ни с одним модулем учебного плана.

Изучение дисциплины базируется на результатах освоения следующих дисциплин:

- Алгоритмизация и программирование
- Введение в профессиональную деятельность

3. Распределение трудоёмкости освоения дисциплины по видам учебной работы и формы текущего контроля и промежуточной аттестации

3.1. Виды учебной работы

Виды учебной работы	Трудоемкость по семестрам
	Очная форма
Лекционные занятия	16
Практические занятия	30
Самостоятельная работа	22
Промежуточная аттестация (зачет)	4
Общая трудоемкость освоения дисциплины	72, ач
	2, зет

3.2. Формы текущего контроля и промежуточной аттестации

Формы текущего контроля и промежуточной аттестации	Количество по семестрам
	Очная форма
Промежуточная аттестация	
Зачеты, шт.	1

4. Содержание и результаты обучения

4.1 Разделы дисциплины и виды учебной работы

№ раздела	Разделы дисциплины, мероприятия текущего контроля	Очная форма		
		Лек, ач	Пр, ач	СР, ач
1.	Введение в Go и настройка окружения для разработки			
1.1.	Введение в Go	1	0	0
1.2.	Работа с Go	1	0	0
2.	Особенности синтаксиса, структуры данных			
2.1.	Основные типы данных	1	0	0

2.2.	Работа с основными типами данных	0	1	0
2.3.	Составные типы данных	0	1	0
2.4.	Интерфейсы и способы их задания	0	1	0
2.5.	Пакет containers	0	1	0
2.6.	Пакет BTree	0	0	1
2.7.	Пакет Sort для сортировок	0	0	1
2.8.	Сериализация	0	0	1
2.9.	Десериализация	0	0	1
2.10.	Переменные стека или кучи	0	0	1
2.11.	Сборка мусора	0	0	1
3.	Работа с зависимостями и библиотеками в Go			
3.1.	Система модулей Go	1	0	0
3.2.	Публикация модулей	0	1	0
3.3.	Работа с зависимостями	0	0	1
4.	Многопоточность в Go			
4.1.	Горутины и конкурентность	1	0	0
4.2.	Race Conditional и как с ним бороться	0	1	0
4.3.	Общий доступ к разделяемым ресурсам	0	0	1
5.	Планировщик Go			
5.1.	Состояния подпрограммы и life-cycle	1	0	0
5.2.	Планировщик Go 1.0	0	1	0
5.3.	Планировщик M:N:P	0	1	0
5.4.	Асинхронные системные вызовы	0	1	0
5.5.	Синхронные системные вызовы	0	1	0
5.6.	Кража работы или Work stilling	0	0	1
5.7.	Бенчмарки и тестирование	0	0	1
6.	Автоматизация тестирования проекта на языке Go			
6.1.	Что такое тестирование?	1	0	0
6.2.	Тестирование в языке Go	0	0	1

6.3.	Статические анализаторы кода	0	0	1
7.	Continuous Integration, Delivery, Deployment			
7.1.	Что такое CI/CD?	1	0	0
7.2.	Цикл работы CI/CD	0	1	0
7.3.	Преимущества CI/CD	0	1	0
7.4.	Реализация CI/CD	0	1	0
7.5.	Инструмент Jenkins	0	0	1
7.6.	Работа с Jenkins	0	0	1
7.7.	Gitlab CI/CD	0	0	1
8.	Сборка проекта в дистрибутив			
8.1.	Команды сборки	1	0	0
8.2.	Сборка приложений Go для разных операционных систем и архитектур	0	1	0
8.3.	Теги сборки	0	1	0
8.4.	Определение компиляции кода на Go	0	1	0
8.5.	Ассемблер в Go	0	1	0
8.6.	Объектный файл	0	0	1
8.7.	Компоновка и релокация	0	0	1
8.8.	Оптимизация размера исполняемого файла	0	0	1
8.9.	Распространения дистрибутивао	0	0	1
8.10.	Передача аргументов командной строки	0	0	1
8.11.	Troubleshooting	0	0	1
9.	Проектирование REST API на Go			
9.1.	Протокол HTTP	1	0	0
9.2.	Основные понятия	0	1	0
9.3.	Передача и обработка ошибок через HTTP	0	0	1
9.4.	Паттерны проектирования	0	1	0
10.	Внедрение gRPC, buf, proto			
10.1.	Протокол gRPC	1	0	0

10.2.	Формат Protobuf	0	1	0
11.	Метрики Prometheus, профилирование			
11.1.	Профилирование	1	0	0
11.2.	Инструмент Prometheus	0	1	0
11.3.	Инструмент Grafana	0	1	0
12.	Современные подходы к развертыванию приложений			
12.1.	Recreate deployment	1	0	0
12.2.	Rolling deployment	0	1	0
12.3.	Ramped deployment	0	1	0
12.4.	Blue/green deployment	0	1	0
12.5.	A/B Testing	0	1	0
12.6.	Shadow testing	0	1	0
12.7.	Canary deployment	0	1	0
12.8.	Сравнение методов развертывания приложений	1	1	0
13.	Виртуализация и ее типы, контейнеризация, Docker			
13.1.	Виртуализация, Контейнеризация, Docker Compose	1	1	0
14.	Управление Docker кластерами, Kubernetes			
14.1.	Docker, Docker Swarm, Kubernetes	1	0	0
Итого по видам учебной работы:		16	30	22
Зачеты, ач				0
Часы на контроль, ач				0
Промежуточная аттестация (зачет)		4		
Общая трудоёмкость освоения: ач / зет		72 / 2		

4.2. Содержание разделов и результаты изучения дисциплины

Раздел дисциплины	Содержание
1. Введение в Go и настройка окружения для разработки	
1.1. Введение в Go	В разделе рассматриваются основные сведения о языке Go: - Что такое Go; - Актуальность языка Go; - Особенности языка Go. Отличия от других языков (C++, Java, Python); - Область применения языка; - Компиляция программы на языке Go. Зависимость от платформы.
1.2. Работа с Go	В разделе рассматривается работа со средой разработки: - Настройка IDE. Установка; - Структура проекта; - Зависимости; - Тесты; - Debug; - Сборка; - Полезные команды и переменные среды.
2. Особенности синтаксиса, структуры данных	
2.1. Основные типы данных	В разделе рассматриваются названия и размерность целых чисел в Go, числа с плавающей точкой, комплексные числа, функционал пакетов string и bytes, константы и оператор iota.
2.2. Работа с основными типами данных	В разделе рассматриваются способы применения основных типов данных. Предоставляется информация о работе с условными конструкциями, циклами и операторами.
2.3. Составные типы данных	В разделе рассматривается информация об основных составных типах данных: - Массивы; - Срезы (в том числе работа append и копирование элементов внутрь функции); - Отображения (map); - Структуры (в том числе объяснение разницы между new и make и вес структуры).

2.4. Интерфейсы и способы их задания	В разделе рассматриваются понятие интерфейсов в Go и примеры их создания и имплементации. Рассматривается стандартная библиотека io.
2.5. Пакет containers	В разделе рассматривается работа с кучами, списками и кольцами, предоставляемая пакетом containers: - container/list; - container/ring; - container/heap.
2.6. Пакет BTree	В разделе рассматривается бинарное дерево, предоставляемое пакетом BTree. Разбираются основные методы пакета.
2.7. Пакет Sort для сортировок	В разделе рассматриваются функции, предоставляемые пакетом Sort, для сортировки данных. Разбираются основные алгоритмы пакета.
2.8. Сериализация	В разделе рассматриваются понятие сериализации и основные функции для работы с ней. Рассматривается пакет encoding/json для работы с JSON.
2.9. Десериализация	В разделе рассматриваются понятие десериализации и основные функции для работы с ней. Рассматриваются подходы восстановления модели данных из файловых форматов данных (xml, json).
2.10. Переменные стека или кучи	В разделе рассматривается работа стека и кучи в Go. Разбираются примеры определения местоположения переменных в Go.
2.11. Сборка мусора	В разделе рассматривается работа Garbage Collection и основные алгоритмы для сборщиков мусора: - Трехцветный алгоритм; - Incremental Garbage Collector; - Concurrent Garbage Collector; Рассматриваются причины запуска Garbage Collection.
3. Работа с зависимостями и библиотеками в Go	
3.1. Система модулей Go	В разделе рассматриваются понятие модуля, его создание и работа с основными директивами: - Директива module; - Директива go; - Директива require; - Директива exclude; - Директива replace; - Директива retract.

3.2. Публикация модулей	В разделе рассматриваются основные концепции, необходимые для публикации модулей: - Рекомендации по названию модулей, публикуемых в сети; - Зеркало зависимостей; - База контрольных сумм; - Добавление версии модуля.
3.3. Работа с зависимостями	В разделе рассматриваются основные концепции, необходимые для работы с зависимостями: - Настройка источников получения модули; - Добавление зависимости; - Обнаружение доступных обновлений; - Вендоринг; - Синхронизация зависимостей кода; - Удаление зависимости.
4. Многопоточность в Go	
4.1. Горутины и конкурентность	В разделе рассматриваются основные сведения об элементах параллельного программирования в Go: - Запуск горутины; - Создание нескольких горутин; - Каналы; - Channel Axioms; - Использование select в канале горутин; - Deadlock; - Конвейер (pipeline).
4.2. Race Conditional и как с ним бороться	В разделе рассматриваются проблема гонки данных и применение Race Conditional для ее решения. Приводится пример формата выводимого распознавателем отчета, а также ряд опций для его детального формирования.
4.3. Общий доступ к разделяемым ресурсам	В разделе рассматриваются инструменты для синхронизации работы нескольких горутин: - Mutex; - RWMutex; - WaitGroup; - Once; - Метод Do для Once; - Atomic операции.
5. Планировщик Go	

5.1. Состояния подпрограммы и life-cycle	В разделе рассматривается главная проблема при работе с многопоточностью. Разбираются жизненный цикл подпрограммы и состояния, в которых она может находиться в зависимости от инструкций и наличия необходимых ресурсов.
5.2. Планировщик Go 1.0	В разделе рассматривается реализация планировщика Go 1.0. Разбираются его недостатки и преимущества. Рассматриваются пример нарушения честности и локальные очереди запуска.
5.3. Планировщик M:N:P	В разделе рассматриваются реализация планировщика M:N:P и его определение в терминах Go. Разбираются события, при которых планировщик принимает решение о переключении контекста.
5.4. Асинхронные системные вызовы	В разделе рассматриваются понятие асинхронных системных вызовов и использование Net Poller с планировщиком M:N:P.
5.5. Синхронные системные вызовы	В разделе рассматриваются понятие синхронных системных вызовов и передача обслуживания подпрограммы от процессора на отдельный поток.
5.6. Кража работы или Work stilling	В разделе рассматривается ситуация кражи работы. Разбираются основные источники новых подпрограмм. Приводится пример кражи работы из случайно выбранного процессоре и из GRQ.
5.7. Бенчмарки и тестирование	В разделе рассматривается влияние переменной GOMAXPROCS на выполнение одной и той же многопоточной программы.
6. Автоматизация тестирования проекта на языке Go	
6.1. Что такое тестирование?	В разделе рассматриваются понятие тестирования и его видов: - Модульное тестирование; - Компонентное тестирование; - Интеграционное тестирование; - Регрессивное тестирование; - Системное тестирование. Разбираются основные цели проведения этапа тестирования во время разработки ПО.
6.2. Тестирование в языке Go	В разделе рассматривается пакет для автоматического тестирование – testing. Приводится пример написания модульных тестов. Рассматриваются fuzzing и случаи его применения, а также средства для создания бенчмарок, которые можно использовать для проверки производительности кода. Объясняются способы проверки покрытия кода тестами. Разбирается mock-тестирование, его основные принципы и пакеты для работы с ним – sqlmock и mockery.

6.3. Статические анализаторы кода	В разделе рассматриваются основные инструменты, выполняющие статический анализ – линтеры и их локальная установка. Разбирается пример анализа go-файлов путем настройки и запуска линтера.
7. Continuous Integration, Delivery, Deployment	
7.1. Что такое CI/CD?	В разделе рассматриваются основные понятия CI/CD: - Непрерывная интеграция; - Непрерывная доставка; - Непрерывное развертывание. Объясняются основные принципы и цель использования CI/CD.
7.2. Цикл работы CI/CD	В разделе рассматриваются основные этапы цикла работы CI/CD и описываются действия, происходящие во время выполнение каждого из этапов.
7.3. Преимущества CI/CD	В разделе рассматривается ряд преимуществ использования CI/CD для разработчиков и заказчиков программного обеспечения.
7.4. Реализация CI/CD	В разделе рассматриваются CI/CD серверы, которые необходимы для реализации CI/CD: - Jenkins; - GitLab CI/CD; - GitHub Actions; - Travis CI, - CircleCI. Рассматриваются алгоритм работы данных сервисов и файлы конфигурации, в которых описаны задачи по автоматизации процесса разработки.
7.5. Инструмент Jenkins	В разделе рассматривается один их наиболее популярных инструментов для автоматизации CI и CD. Разбираются его преимущества и ключевые возможности.
7.6. Работа с Jenkins	В разделе рассматриваются установка, настройка и работа с Jenkins. Приводится пример добавления нового пайплайна и скрипт CI-пайплайна для репозитория на GitHub с использованием Jenkins в качестве инструмента для непрерывной интеграции.
7.7. Gitlab CI/CD	В разделе рассматривается инструмент для разработки ПО с использованием непрерывных методологий – CI и CD. Приводится пример настройки Gitlab CI/CD путем написания файла .gitlab-ci.yml.
8. Сборка проекта в дистрибутив	

8.1. Команды сборки	В разделе рассматриваются основные команды Go, используемые для сборки и запуска проекта, а также команды и директивы для работы с исходными файлами проекта: - go build [-o output] [-i] [build flags] [packages]; - go install; - go run; - go generate; - go embed.
8.2. Сборка приложений Go для разных операционных систем и архитектур	В разделе рассматривается проблема медленного запуска двоичного файла на платформе с другой операционной системой или архитектурой. Приводится пример просмотра списка платформ путем выполнения команды go tool dist list, разбираются полученные выходные данные.
8.3. Теги сборки	В разделе рассматриваются понятие ограничения сборки и причины использования тегов. Предоставляется пример применения тегов сборки на примере конкретного файла пакета. Рассматривается применение тэга pro.
8.4. Определение компиляции кода на Go	В разделе рассматривается значимость компиляции и разбирается ее процесс на примере конкретной программы. Объясняются фазы компиляции: - Синтаксический разбор; - Построение AST; - SSA; - Генерация ассемблерного кода. Рассматриваются особенности компиляции Go, причины быстрой компиляции.
8.5. Ассемблер в Go	В разделе рассматриваются наиболее важные аспекты ассемблера Go. Предоставляются пути получения ассемблерного кода при помощи интерфейса и команды. Разбираются пример генерации ассемблерного кода и сгенерированный ассемблерный код. Рассматриваются части, из которых состоят строки ассемблерного кода, и отдельные конструкции: - Директива TEXT; - Регистр SB; - Сообщение NOSPLIT; - Символы $0 и $16.

8.6. Объектный файл	В разделе рассматривается результат работы компилятора на примере конкретной программы, задействующей два пакета. Показывается результат процесса компиляции данной программы. Разбираются составные части объектного файла. Представляется формат объектного файла и объясняется значение первых байт: - 0102; - 00dc; - 0100; - dc01; - 0a.
8.7. Компоновка и релокация	В разделе рассматривается процесс сборки исполняемого файла из переданных объектных файлов, которые подаются на вход компоновщику после компиляции. Дается определение релокации, демонстрируется список проиндексированных символов. Приводится пример этапа релокации, на котором всем разделам и инструкциям назначаются виртуальные адреса. Рассматриваются адреса инструкций и основных функций: - main; - fmt.Println.
8.8. Оптимизация размера исполняемого файла	В разделе рассматривается уменьшение размера исполняемого файла путем применение некоторых процессов: - Стриппинг; - Сжатие. Разбираются команды, необходимые для оптимизации размера исполняемого файла, анализируется размер полученного в результате применения операций бинарного файла.
8.9. Распространения дистрибутивао	В разделе рассматривается распространение исполняемого файла, полученного в результате сборки программы. Разбираются пути его распространения: - Передача по сети Интернет в изначальном формате или в формате архива; - Размещение исходного кода в системе контроля версий; - Создание из собранного исполняемого файла Docker-образа и его загрузка в DockerHub.
8.10. Передача аргументов командной строки	В разделе рассматривается возможность обработки аргументов командной строки путем передачи переменной Args из пакета os. Разбирается пример формата передачи аргументов командной строки на примере конкретной программы.

8.11. Troubleshooting	В разделе рассматривается решение наиболее встречающейся у начинающих разработчиков ошибки при выполнении команды go run. Предлагается путь исправления ошибки.
9. Проектирование REST API на Go	
9.1. Протокол HTTP	В разделе рассматривается понятие HTTP-сообщения. Разбирается структура HTTP-запросов и HTTP-ответов: - Стартовая строка; - Метод HTTP-запроса; - Цель запроса; - Версия используемого протокола. Объясняются основные HTTP-методы: - Get; - Post; - Head; - Put; - Delete; - Patch. Рассматриваются основные заголовки запроса и заголовки ответа. Дается определение идемпотентности метода. Разбираются токены идемпотентности и черновики (драфты).
9.2. Основные понятия	В разделе рассматриваются определения следующих терминов: - REST; - API; - REST API. Разбираются основные принципы REST: - Клиент-серверная архитектура; - Отсутствие состояния; - Кэширование; - Единообразие интерфейса; - Слоистая архитектура; - Код по требованию.
9.3. Передача и обработка ошибок через HTTP	В разделе рассматривается передача сгенерированных ошибок с помощью использования функции Error из пакета http. Разбираются пример создания пользовательской ошибки и чтение и использование пользовательских сообщений об ошибках.

9.4. Паттерны проектирования	В разделе рассматривается определение паттернов и необходимость их использования. Разбираются основные группы, на которые делят паттерны: - Архитектурные паттерны; - Паттерны проектирования. Приводятся основные шаблоны, используемые при реализации Restful сервисов: - CRUD; - MVC; - Фасад; - Кэширование.
10. Внедрение gRPC, buf, proto	
10.1. Протокол gRPC	В разделе рассматриваются описание gRPC и причины его использования. Разбираются основные причины высокой производительности gRPC: - HTTP/2; - Мультиплексирование запроса/ответа; - Сжатие заголовка. Объясняются RPC коммуникации между приложениями и различия между gRPC и REST.
10.2. Формат Protobuf	В разделе рассматривается описание Protobuf и варианты его использования. Объясняется разница между Protobuf и JSON. Разбираются варианты использования Protobuf: - Proto2; - Proto3; - gRPC. Описываются основные причины использование Protobuf вместо других форматов данных.
11. Метрики Prometheus, профилирование	
11.1. Профилирование	В разделе рассматривается инструмент для анализа производительности программ – профилирования. Объясняются причины его использования. Описывается встроенный в Go инструмент для профилирования – pprof, определяются его основные возможности: - CPU профилирование; - Память (heap) профилирование; - Горутинное профилирование; - Генерация отчетов. Разбираются работа с pprof и подходы к его использованию.

11.2. Инструмент Prometheus	В разделе рассматривается база данных временных рядов – Prometheus. Разбираются основные типы метрик Prometheus: - Counter; - Guage; - Histogram; - Summary.
11.3. Инструмент Grafana	В разделе рассматривается программная система визуализации данных – Grafana. Объясняются причины ее использование. Описывается принцип работы Grafana в связке с Prometheus.
12. Современные подходы к развертыванию приложений	
12.1. Recreate deployment	В разделе рассматривается базовое развертывание и описывается принцип его работы. Рассматриваются его преимущества и недостатки. Разбираются ситуации использования данной стратегии.
12.2. Rolling deployment	В разделе рассматривается непрерывное развертывание и описывается принцип его работы. Рассматриваются его преимущества и недостатки. Разбираются ситуации использования данной стратегии.
12.3. Ramped deployment	В разделе рассматривается непрерывное развертывание и описывается принцип его работы. Рассматриваются его преимущества и недостатки. Разбираются ситуации использования данной стратегии.
12.4. Blue/green deployment	В разделе рассматривается сине-зеленое развертывание и описывается принцип его работы. Рассматриваются его преимущества и недостатки. Разбираются ситуации использования данной стратегии.
12.5. A/B Testing	В разделе рассматривается A/B тестирование и описывается принцип его работы. Рассматриваются его преимущества и недостатки. Разбираются ситуации использования данной стратегии. Объясняется способ проведения тестов для получения статистически значимого результата. Приводятся примеры метрик: - Конверсия; - Экономические метрики; - Поведенческие факторы.

12.6. Shadow testing	В разделе рассматривается теневое развертывание и описывается принцип его работы. Рассматриваются его преимущества и недостатки. Разбираются ситуации использования данной стратегии.
12.7. Canary deployment	В разделе рассматривается канареечное развертывание и описывается принцип его работы. Рассматриваются его преимущества и недостатки. Разбираются ситуации использования данной стратегии. Описываются основные столпы канареечного развертывания: - Балансировка; - Мониторинг; - Анализ версий; - Автоматизация.
12.8. Сравнение методов развертывания приложений	В разделе рассматривается анализ стратегий развертывания приложений, основанный на следующих критериях: - Zero-downtime; - Real traffic testing; - Targeted users; - Cloud cost; - Rollback duration; - Negative impact on user; - Complexity of setup.
13. Виртуализация и ее типы, контейнеризация, Docker	

13.1. Виртуализация, Контейнеризация, Docker Compose	В разделе рассматривается концепция виртуализации и ее применение. Перечисляются преимущества и недостатки виртуализации и ее типы. Разбирается основополагающая часть концепции виртуализации – типы гипервизоров. В разделе рассматривается метод виртуализации – контейнеризация. Перечисляются достоинства и недостатки данного метода. Приводятся примеры наиболее популярных инструментов, реализующих технологию контейнеризации. Объясняется отличие контейнеров от виртуальных машин.В разделе рассматривается программная платформа для быстрой разработки Docker. Разбирается понятие Docker-образа. Приводится пример работы с Docker с нуля: - Установка Docker; - Dockerfile; - Запуск приложения; - Основные команды Docker; - Docker и CI/CD. В разделе рассматривается инструмент для обработки нескольких контейнеров одновременно – Docker Compose. Разбирается команда запуска всех образов. Демонстрируется содержание конфигурационного файла Docker Compose и пример такого конфигурационного файла. Подробно рассматриваются составляющие конфигурационного файла: - Сервисы; - Тома; - Сеть.
14. Управление Docker кластерами, Kubernetes	

14.1. Docker, Docker Swarm, Kubernetes	В разделе рассматривается система оркестрации контейнеров. Приводится сравнение Docker и других оркестраторов для понимания границ зон ответственности docker и задач оркестратора. Рассматриваются отличия наиболее популярных оркестраторов – Docker Swarm и Kubernetes. В разделе рассматривается работа оркестратора Docker Swarm: - Основные команды; - Запуск приложения на кластере; - Команды состояния стека и сервисов; - Масштабирование и запуск на конкретных нодах; - Web-панель Portainer. В разделе рассматривается работа оркестратора Kubernetes: - Основные компоненты; - Модуль (создание модуля, отправка запроса в модуль, организация модулей); - Контроллер репликаций, набор реплик, набор демонов; - Службы (создание службы, адресация внутри служб, подключение к службам, находящимся за пределами кластера, создание псевдонима служб, предоставление внешним клиентам доступа к службам); - Volume; - Развертывания (использование развертываний для декларативного обновления приложений, создание развертывания, стратегии развертывания); - Ресурсы StatefulSet: развертывание реплицируемых приложений с внутренним состоянием (использование набора реплик ReplicaSet для репликации модуля баз данных, набор модулей с внутренним состоянием, обеспечение стабильного выделенного хранилища, создание набора StatefulSet).

5. Образовательные технологии

1. Лекционные занятий, которые могут проводиться как в очной форме, так и дистанционной. Каждая лекция подкрепляется видео-материалами.
2. Практические занятия по выбранной предметной области, выполняемые студентами
3. Курсовая работа по выбранной предметной области, выполняемая студентами

6. Лабораторный практикум

Не предусмотрено

7. Практические занятия

№ раздела	Наименование практических занятий (семинаров)	Трудоемкость, ач
		Очная форма
1.	Необходимо создать консольное приложение - калькулятор, которое будет выполнять следующие функции: 1. Программа должна запрашивать у пользователя два числа и операцию, которую он хочет выполнить над этими числами (сложение, вычитание, умножение или деление). 2. Программа должна провести выбранную операцию над введенными числами и вывести результат на экран. 3. Программа должна корректно обрабатывать ситуации, когда пользователь вводит некорректные данные, например операцию, которая не поддерживается, или вводит некорректные числа.	3
2.	1. В офисе компании Y у каждого из N отделов есть свой кондиционер, на котором может быть выставлена температура в диапазоне от 15 до 30 градусов. В каждом отделе работает K сотрудников. Каждый сотрудник, приходя в офис, устанавливает желаемое значение температурной границы (не больше или не меньше T). Напишите программу, которая после каждого прибывшего сотрудника будет выводить оптимальную температуру для всего отдела. Если такой температуры нет выведете -1. 2. В компании Y работает сотрудник A, каждый день он ходит на обед в кафе P. В кафе шведский стол, поэтому перед тем, как сделать выбор он оценивает блюда согласно своим предпочтениям. Но самое вкусное он постоянно есть не хочет, поэтому зачастую выбирает k-ое по предпочтению блюдо. Прочитайте мысли сотрудника A и выведете рейтинг выбранного им блюда сегодня.	3
3.	В рамках выполнения упражнения студенту необходимо реализовать приложение, выполняющее загрузку информации о валютах, предоставленную Центральном банком России в формате XML, и произвести автоматизированную обработку и конвертацию полученных данных.	3

4.	В данной лабораторной работе студентам предстоит выполнить следующие задачи: 1. Реализовать структуру с использованием примитивов синхронизации Go. 2. Продемонстрировать работу созданной структуры с использованием выбранного примитива синхронизации. 3. Продемонстрировать работу программы без использования примитива синхронизации. Предполагается, что программа может работать только с его использованием. Пример: написать структуру счетчика людей, проходящих через турникеты в метро. В программе должна использоваться многопоточность.	3
5.	Покрыть заранее предоставленный код тестами, использовать mock-тесты для проверки различных сценариев, а также выполнить анализ кода с помощью golangci-lint и исправить выявленные замечания.	3
6.	Необходимо реализовать автоматический запуск тестов после коммита в ветку удаленного репозитория. Код и тесты можно взять из 6-го упражнения, либо использовать что-то другое.	3
7.	В данной лабораторной работе студентам предстоит выполнить следующие задачи: 1. Сборка проекта. Выбрать любую программу на языке Golang и выполнить для нее следующие шаги: построение AST, получение формы SSA, генерация объектного файла, генерация исполняемого файла. 2. Применение тэгов сборки. Выбрать любую программу (можно ту же, что и в пункте 1), добавить в нее тэги, проверить работоспособность. 3. Применение команды go generate. Воспроизвести любой пример применения go generate (можно использовать пример с генерацией Mock объектов, представленный в лекции). 4. Применение директивы //go:embed: Воспроизвести любой пример применения //go:embed (можно использовать пример, представленный в лекции).	3
8.	Необходимо реализовать REST API сервис по управлению телефонными контактами, а именно: 1. Реализовать подключение к базе данных; 2. Реализовать CRUD-операции; 3. Предусмотреть отправку осмысленных сообщений с достаточным количеством информации о возникших проблемах помимо кодов состояния при генерации ошибок; 4. Предусмотреть проверку отправленных пользователем данных (номер телефона и т.д.).	3

9.	В данной лабораторной работе студентам предстоит выпол-нить следующие задачи: 1. Установить Docker локально (в случае использования студентом ОС Windows может быть установлен Docker Desktop). 2. Найти пример проекта (или использовать собственный), содержащий REST API бэкенд часть и фронтенд часть и создать из них Docker образы. По желанию можно добавить базу данных (можно использовать официальный образ). Сложность проекта не принципиальна, можно выбрать бэкенд часть с одним эндпоинтом, возвращающим текст “hello world”, и фронтенд часть, отображающим его. 3. Создать контейнеры из полученных образов и работоспособности проекта. 4. Достичь того же результата с применением Docker Compose.	3
10.	В данной лабораторной работе необходимо развернуть любой многокомпонентный проект (например, WordPress+PHP+MySql) на двух кластерах: 1. Docker Swarm кластер. Кластер должен содержать как ми-нимум одну мастер ноду и одну воркер-ноду. 2. Развернуть Kubernetes кластер.	3
	Итого часов	30

8. Организация и учебно-методическое обеспечение самостоятельной работы

Примерное распределение времени самостоятельной работы студентов

Вид самостоятельной работы	Примерная трудоемкость, ач
	Очная форма
Текущая СР	
работа с лекционным материалом, с учебной литературой	4
опережающая самостоятельная работа (изучение нового материала до его изложения на занятиях)	0
самостоятельное изучение разделов дисциплины	4
выполнение домашних заданий, домашних контрольных работ	0
подготовка к лабораторным работам, к практическим и семинарским занятиям	0
подготовка к контрольным работам, коллоквиумам	0
Итого текущей СР:	8
Творческая проблемно-ориентированная СР	
выполнение расчётно-графических работ	0
выполнение курсового проекта или курсовой работы	10
поиск, изучение и презентация информации по заданной проблеме, анализ научных публикаций по заданной теме	4
работа над междисциплинарным проектом	0
исследовательская работа, участие в конференциях, семинарах, олимпиадах	0
анализ данных по заданной теме, выполнение расчётов, составление схем и моделей на основе собранных данных	0
Итого творческой СР:	14
Общая трудоемкость СР:	22

9. Учебно-методическое обеспечение дисциплины

9.1. Адрес сайта курса

https://disk.yandex.ru/d/Eo58Any-KSx5pQ

9.2. Рекомендуемая литература

Основная литература

№	Автор, название, место издания, издательство, год (годы) издания	Год изд.	Источник
1	Никифоров И.В. и др. Введение в язык программирования Go: Санкт-Петербург: ПОЛИТЕХ-ПРЕСС, 2025. URL: http://elib.spbstu.ru/dl/2/id25-34.pdf	2025	ЭБ СПбПУ

Дополнительная литература

№	Автор, название, место издания, издательство, год (годы) издания	Год изд.	Источник
1	Ибрагимов И.Д., Коликова Т.В. Прототип сервера авторизации на языке GO для компании «Зенит», 2018. URL: http://elib.spbstu.ru/dl/2/v18-1382.pdf	2018	ЭБ СПбПУ

Ресурсы Интернета

1. УМК в электронном виде: https://disk.yandex.ru/d/Eo58Any-KSx5pQ

9.3. Технические средства обеспечения дисциплины

Для выполнения практических работ студенты используют персональные компьютеры.

По курсу имеется набор электронных документов, подготовленных авторами программы, выдаваемых студентам на электронных носителях, либо высылаемых по электронной почте (по индивидуальным запросам). Эти документы включают:

- электронные презентации по разделам курса;
- электронные книги и другая учебная литература;
- методические указания по курсовому проекту.

10. Материально-техническое обеспечение дисциплины

Для проведение дисциплины дополнительное материально-техническое обеспечение не предусматривается, т.к. для выполнения всех заданий достаточно персонального ноутбука и доступа в Интернет.

11. Критерии оценивания и оценочные средства

11.1. Критерии оценивания

Для дисциплины «Введение в язык программирования Go» формой аттестации является зачёт. Дисциплина реализуется с применением системы индивидуальных достижений.

Текущий контроль успеваемости

Максимальное значение персонального суммарного результата обучения (ПСРО) по приведенной шкале - 100 баллов
Максимальное количество баллов приведенной шкалы по результатам прохождения двух точек контроля - 80 баллов.

Подробное описание правил проведения текущего контроля с указанием баллов по каждому контрольному мероприятию и критериев выставления оценки размещается в СДО в навигационном курсе дисциплины.

Промежуточная аттестация по дисциплине

Максимальное количество баллов по результатам проведения аттестационного испытания в период промежуточной аттестации – 20 баллов приведенной шкалы.

Промежуточная аттестация по дисциплине проводится в соответствии с расписанием.

Для дисциплины «Введение в язык программирования Go» формой аттестации является зачет. Оценивание качества освоения дисциплины производится с использованием рейтинговой системы (СИД).

Уровень освоения дисциплины оценивается по результатам исполнения практических заданий, степени владения студентом понятийным аппаратом, умением применять полученные знания в конкретных ситуациях, предлагаемых для рассмотрения, и результатам выполнения курсовой работы.

В соответствии с рабочим учебным планом, формами контроля является зачет. Форма текущего контроля оценивается по 100-балльной шкале.

Таблица максимального количества баллов, которые студенты могут заработать за каждый раздел:

Выполненные практических работ	66
Выполненный курсовой проект	34
Итого	100

Будут применяться следующие оценочные средства для текущей и итоговой аттестации:

- домашние задания на контроль практических умений репродуктивного уровня;
- задание для курсового проекта;
- проблемы, позволяющие оценить обобщённые профессиональные и общекультурные умения (компетенции) студентов.

Перечень контрольных (экзаменационных) вопросов, позволяющих оценить качество усвоения учебного материала:

1. Особенности языка Go. Область применения языка Go. Настройка окружения для разработки. Структура проектов. Утилиты Go.
2. Основные типы данных. Строки. Константы. Работа с основными типами данных: условные конструкции, циклы.
3. Составные типы данных. Массивы. Срезы. Работа append. Отображения (map). Структуры. Разница между new и make.
4. Интерфейсы. Пакеты containers, Sort. Сериализация. Десериализация.
5. Переменные стека или кучи. Garbage collection.
6. Система модулей Go. Создание модуля. Директивы module, go, require, exclude, replace, retract.
7. Установка внешних зависимостей. Вендоринг. Добавление, обнаружение доступных обновлений, синхронизация зависимостей, удаление.
8. Стандартные библиотеки Go. String, os, io, errors, net/http, database/sql, encoding/json, flag.
9. Команды сборки приложения. Сборка приложения для разных ОС. Теги сборки.
10. Этапы компиляции. Компоновка и релокация. Оптимизация размера исполняемого файла.
11. Возможности бенчмарок и тестов библиотеки testing в Go. Основные методы структур T и B.
12. Линтеры. Основные возможности инструмента golangci-lint.
13. Многопоточность в Go. Горутины и конкурентность. Каналы общения, аксиомы.

14. Объединение горутин. Использование select в канале горутин. Race Conditional и как бороться. Применение, примеры гонок.
15. Общий доступ к разделяемым ресурсам. Основные типы, предоставляемые пакетом sync: Mutex, WaitGroup, Once, Метод Do. Атомарные операции.
16. REST API. Протокол HTTP. Методы HTTP. Идемпотентность.
17. Генерация ошибки. Передача пользовательской ошибки.
18. Чтение и использование пользовательских сообщений об ошибках.
19. Разбор и отображение данных в формате JSON.
20. Подключение к базе данных. Основные функции. Установка пакетов.
21. Паттерны проектирования. Использование шаблонов при создании Restful сервисов.
22. Recreate deployment, Rolling deployment.
23. Ramped deployment, Blue/green deployment.
24. A/B Testing и примеры метрик.
25. Shadow testing и Canary deployment.
26. Виртуализация. Типы. Преимущества и недостатки. Контейнеризация.
27. Docker. Основные команды. Docker Compose. Сервисы. Тома. Сеть.
28. Docker Swarm. Основные команды. Запуск приложения на кластере.
29. Kubernetes.Опишите основные элементы, которые вы будете использовать для развертывания клиент-серверного приложения на трехнодном кластере.
30. CI/CD. Continuous Integration. Continuous Delivery. Continuous Deployment. Отличия Delivery и Deployment.
31. Цикл CI/CD. Преимущества CI/CD. Реализация CI/CD. Инструменты CI/CD.

Результаты промежуточной аттестации, определяются на основе баллов, набранных в рамках применения, СИД

Баллы по приведенной шкале в рамках применения СИД (ПСРО+ ПА)	**Оценка по результатам промежуточной аттестации**
	Экзамен/диф.зачет/зачет
0 - 60 баллов	Неудовлетворительно/не зачтено
61 - 75 баллов	Удовлетворительно/зачтено
76 - 89 баллов	Хорошо/зачтено
90 и более	Отлично/зачтено

11.2. Оценочные средства

Оценочные средства по дисциплине представлены в фонде оценочных средств, который является неотъемлемой частью основной образовательной программы и размещается в электронной информационно-образовательной среде СПбПУ на портале etk.spbstu.ru

12. Методические рекомендации по организации изучения дисциплины

Студентам рекомендуется быть активными – обращаться к преподавателю по организационным, методическим и содержательным вопросам.

Приветствуется инициативность в практических работах и курсовом проекте – если студент хочет реализовать проект сложнее, интереснее, попробовать новую для себя технологию, это усложнение допускается.

Рекомендуется читать 10 страниц книг из рекомендованного списка литературы. Это примерно 3500 страниц в год, что около 6 полных книг. Можно пользоваться сторонними материалами для самостоятельного изучения.

Студентам не стоит ограничиваться книгами из рекомендованного списка литературы. Рекомендуется читать как можно больше дополнительной литературы по языку, регулярно читать статьи, ознакомиться с разными образовательными материалами по Go и изучать материалы не только по самому языку Golang, но и по таким фундаментальным темам, как, например, алгоритмы.

Кроме изучения теоретических материалов рекомендуется как можно больше заниматься практикой. Только на реальном опыте можно понять все тонкости языка и создания программ. Рекомендуется выполнять все практические задания курса, решать задачи на платформе LeetCode, создавать свои пет-проекты. Запускать у себя все примеры из лекций, книг, видео и статей, изучать, как эти примеры работают и почему именно так.

Рекомендуется научиться пользоваться спецификацией языка Golang, уметь находить в ней нужную информацию - именно там находится вся актуальная информация по языку.

Рекомендуется заниматься английским языком – это международный язык в сфере технологий, все документации, дополнения к спецификации, книги, тренажеры и курсы изначально выходят именно на английском языке. Знание английского языка помогает и в написании кода – корректно выбирать названия функций и переменных.

13. Адаптация рабочей программы для лиц с ОВЗ

Адаптированная программа разрабатывается при наличии заявления со стороны обучающегося (родителей, законных представителей) и медицинских показаний (рекомендациями психолого-медико-педагогической комиссии). Для инвалидов адаптированная образовательная программа разрабатывается в соответствии с индивидуальной программой реабилитации.