МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное бюджетное образовательное учреждение высшего образования «Кабардино-Балкарский государственный университет им. Х.М. Бербекова» (КБГУ)

ИНСТИТУТ ИСКУССТВЕННОГО ИНТЕЛЛЕКТА И ЦИФРОВЫХ ТЕХНОЛОГИЙ
КАФЕДРА ПРИКЛАДНОЙ МАТЕМАТИКИ И ИНФОРМАТИКИ

СОГЛАСОВАНО

Руководитель образовательной программы _____ **М.М. Лафишева**

«12» 04 **2023г.**

УТВЕРЖДАЮ
Директор института
_____ **А.Х. Шанеиков**

«12» 04 **2023**

ФОНД ОЦЕНОЧНЫХ СРЕДСТВ (ОЦЕНОЧНЫХ МАТЕРИАЛОВ) ПО ДИСЦИПЛИНЕ (МОДУЛЮ)

«ПРОФЕССИОНАЛЬНАЯ РАЗРАБОТКА ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ GOLANG»

02.03.02 Фундаментальная информатика и информационные технологии
(код и наименование направления подготовки)

«Проектирование систем искусственного интеллекта»
(наименование профиля подготовки)

Бакалавр
Квалификация (степень) выпускника

Очная
Форма обучения

Нальчик - 2023

Оглавление

1. Перечень компетенций и этапы их формирования

Карта компетенции

Шифр и название компетенций:
профессиональные (ПКС):

Коды	Содержание компетенций
ПКС-2	Способен создавать и исследовать новые математические модели в естественных науках, промышленности и бизнесе, с учетом возможностей современных информационных технологий, программирования и компьютерной техники

2. Методические материалы и типовые контрольные задания, необходимые для оценки знаний, умений, навыков опыта деятельности, характеризующих этапы формирования компетенций в процессе освоения образовательной программы

1.1. Этапы формирования компетенций и средства оценивания

Результаты обучения (компетенции)	Индикаторы достижения компетенции	Освоенные показатели оценки результатов обучения	Виды оценочного материала, обеспечивающий формирование компетенций
ПКС-2. Способен создавать и исследовать новые математические модели в естественных науках, промышленности и бизнесе, с учетом возможностей современных информационных технологий, программирования и компьютерной техники	**ПКС-21.** Способен использовать основные методы проектирования и производства программного продукта и программных комплексов, их сопровождения, администрирования и развития (эволюции)	**ПКС-2.1.** З-1. Знает арсенал и области применения современных научных методов и информационных технологий, необходимых для решения задач, имеющих естественно-научное содержание и возникающих при выполнении профессиональных функций **ПКС-2.1.** У-1. Умеет описывать проблемы и ситуации профессиональной деятельности на основе знаний математического аппарата и естественнонаучных дисциплин и формулировать задачу профессиональной деятельности в области прикладной математики и информатики аппарата и естественнонаучных дисциплин	Типовые оценочные материалы для устного опроса (п. 5.1.1); типовые оценочные материалы для контрольной работы (п. 5.2.1); типовые тестовые задания (п. 5.2.2); типовые оценочные материалы к экзамену (п. 5.2.3)

		ПКС-2.1. В-1. Владеет навыками производить статистические расчеты с применением соответствующих математических методов и информационных технологий, а также проводить последующую аналитическую работу с полученными данными	
	ПКС-2.2. Способен использовать методы проектирования и производства программного продукта, принципы построения, структуры и приемы работы с инструментальными средствами, поддерживающими создание программного продукта	**ПКС-2.2.** З-1. Знает методологии разработки программного обеспечения и технологии программирования **ПКС-2.2.** У-1. Умеет использовать существующие типовые решения и шаблоны проектирования программного обеспечения, применять методы и средства проектирования программного обеспечения, структур данных, баз данных, программных интерфейсов **ПКС-2.2.** В-1. Владеет навыками программирования элементов компьютерной графики и навыками создания правильных, геометрических и реалистичных изображений на экране компьютера	

1.2. Критерии формирования оценок на различных этапах их формирования

Текущий и рубежный контроль

Этап (уровень)	**Первый этап (уровень)**	**Второй этап (уровень)**	**Третий этап (уровень)**
Баллы	36-50 баллов	51-60 баллов	61-70 баллов
Характеристика	Полное или частичное посещение аудиторных занятий. Частичное выполнение домашнего задания. Частичное выполнение заданий контрольных работ, тестовых заданий на оценку «удовлетворительно».	Полное или частичное посещение аудиторных занятий. Полное выполнение домашнего задания. Выполнение заданий на коллоквиуме на оценку «хорошо».	Полное посещение аудиторных занятий. Полное выполнение домашнего задания, заданий контроль-ных работ. Выполнение заданий на коллоквиуме на оценку «отлично».

На первом (начальном) этапе формирования компетенции формируются знания, умения и навыки, составляющие базовую основу компетенции, без которой невозможно ее дальнейшее развитие. Обучающийся воспроизводит термины, факты, методы, понятия, принципы и правила; решает учебные задачи по образцу.

На втором (основном) этапе формирования компетенции приобретается опыт деятельности, когда отдельные компоненты компетенции начинают «работать» в комплексе и происходит выработка индивидуального алгоритма продуктивных действий, направленных на достижение поставленной цели. На этом этапе обучающийся осваивает аналитические действия с предметными знаниями по конкретной дисциплине, способен самостоятельно решать учебные задачи, внося коррективы в алгоритм действий, осуществляя координирование хода работы, переносит знания и умения на новые условия.

Третий (завершающий) этап – это овладение компетенцией. Обучающийся способен использовать знания, умения, навыки при решении задач повышенной сложности и в нестандартных условиях. По результатам этого этапа обучающийся демонстрирует итоговый уровень сформированности компетенции.

Промежуточная аттестация
8 семестр-экзамен

Семестр	**Шкала оценивания**			
	Неудовлетвори-тельно (36-60 баллов)	**Удовлетвори-тельно (61-80 баллов)**	**Хорошо (81-90 баллов)**	**Отлично (91-100 баллов)**
1	Студент имеет 36-60 баллов по итогам текущего и рубежного контроля, на экзамене не дал полного ответа ни на один вопрос, не сделал пример. Студент имеет 36-45 баллов по итогам текущего и рубежного контроля, на экзамене дал полный ответ только на один вопрос, а пример сделан неправильно.	Студент имеет 36-50 баллов по итогам текущего и рубежного контроля, на экзамене дал полный ответ на один вопрос и частично (полностью) ответил на второй, а пример сделан не верно. Студент имеет 46-60 баллов по итогам текущего и рубежного контроля, на экзамене дал полный ответ на один вопрос или частично ответил на оба вопроса, а пример не сделан. Студент имеет по итогам текущего и рубежного контроля 61-70 баллов на экзамене	Студент имеет 51-60 баллов по итогамтекущего и рубежного контроля, на экзамене дал полный ответ на один вопроси частично (полностью) ответил на второй. Пример сделан верно. Студент имеет 61 – 65 баллов по итогам текущего и рубежного контроля, на экзамене дал полный ответ на один вопроси частично ответил на второй, и в примере есть недочеты, которые не повлияли на ответ. Студент имеет	Студент имеет 61-70 баллов по итогам текущего и рубежного контроля, на экзамене дал полный ответ на один вопрос и частично (полностью) ответил на второй, и пример сделан правильно. Или же студент на оба вопроса ответил верно, а в задаче, есть неточности, которые не повлияли на ответ.

		не дал полного ответа ни на один вопрос. В решении примера есть грубая ошибка, которая повлияла на ответ, вследствие чего пример сделан не верно	66-70 баллов по итогам текущего и рубежного контроля, на экзамене дал полный ответ только на один вопрос. В примере есть неточности, которые не повлияли на ответ.	

3. Перечень контрольных заданий и иных материалов, необходимых для оценки знаний, умений, навыков и опыта деятельности

3.1. Вопросы для коллоквиумов

Вопросы для оценки компетенции «ПКС-2»:

Тема 1. Строгая статическая типизация (утиная типизация в случае интерфейсов).
Тема 2. Полноценная поддержка юникода.
Тема 4. Прочие фишки функционального программирования - лямбды, замыкания и т.п.
Тема 5. Указатели Go, но над которыми нельзя выполнять арифметические операции, как в C/C++/D.
Тема 6. Легковесные потоки goroutines - для управления которыми и взаимодействия между которыми в Go предусмотрены удобные средства.
Тема 7. Отсутствие ООП- фанатизма, по большому счету Go является процедурным языком с поддержкой интерфейсов.
Тема 8. Нет поддержки исключений, вместо них предлагается использовать интерфейс error и возможность функций возвращать несколько значений;
Тема 9. Язык является интерпретируемым и компилируемым, при разработке Go особое внимание уделяется скорости компиляции.

Практические задания на коллоквиум

1. Верните несколько значений функции.
2. Инициализируйте структуру.
3. Отсортируйте элементы слайса целых чисел.
4. Сравните слайсы.
5. Каков будет вывод этого кода?

```
package main
  import "fmt"
   const (
        i = 8
        j
        k
   )
  func main() {
   fmt.Println(i, j, k)
}
```

6. Определите тип объекта.
7. Дано: два неупорядоченных среза.
а) a := []int{37, 5, 1, 2} и b := []int{6, 2, 4, 37}.
б) a = []int{1, 1, 1} и b = []int{1, 1, 1, 1}.
Верните их пересечение.
8. Напишите генератор случайных чисел.
9. Дано: строка a := «mfgah134517095aldrfgvh8h».
Вырежьте из строки всё, кроме чисел.
10. Проверьте наличие ключа в map.
11. Дано: срезы
a := []int{1, 2, 3}
b := []int{4, 5, 6}.
Выполните конкатенацию двух срезов.
12. Отложите вызов функции с возвращаемым значением.
13. Преобразуйте строку в число.
14. Проверьте, что файл существует.
15. Создайте две горутины, чтобы числа из одного канала читались по мере поступления, возводились в квадрат и результат записывался во второй канал.

3.2. Тестовые задания по дисциплине «Профессиональная разработка программного обеспечения Golang»(контролируемая компетенция ПКС-2):

Что выведет код ниже?
```
var happiness = 0.04
var isHappy = true
if happiness >= 0.5 || isHappy {
  fmt.Println("Happy ")
}
fmt.Println(":)")
```
-: :)
-: Happy
+: Будет выведена ошибка
Что выведет код ниже?
```
var happiness = 0.04
var isHappy = true
if happiness >= 0.5 || isHappy {
  fmt.Println("Happy ")
}

fmt.Println(":)")
```
:)
-: Happy
-: Будет выведена ошибка
+: Happy :)

Обязательно ли писать «package»?
-: Нет, не обязательно
-: Обязательно только для главного файла
+: Всегда обязательно

Где верно указан комментарий?

-: # Комментарий
-: /* Комментарий */
-: ## Комментарий
+: // Комментарий
Ближайший конкурент по скорости является
-: C++
-: COBOL
-: Java
-: Таковых нет
+: Си

Существует ли в языке Go оператор switch case?
-: Нет, не существует
-: Существует его аналог
+: Да, существует

Какая компания создала Golang?
-: Компания FaceBook
-: Компания Apple
-: Группа независимых разработчиков
+: Компания Google

Где верно создана переменная?
-: var age int = 12
-: var age = 12
-: var isDone bool = true
-: Нет верного варианта
+: Все варианты верные

Где верно выведена переменная?
package main
import "fmt"
var bob = "Bob"
Println(bob)
println(bob)
log(bob)
console(bob)
fmt.Println(bob)
Верно ли ниже указан импорт пакетов?
import ("fmt"; "os"; "net/http")
-: Не верно указаны названия модулей
-: Нужно прописывать через запятую
-: Круглые скобки не требуются
-: Все варианты ошибок верны
+: Код указан верно

Язык Go является...
-: интерпретируемым
+: компилируемым

Создатели Go взяли за основу языка C и Python. В C им нравилась универсальность, а в Python’e — простота, но и в том, и в другом языке были вещи, которые разработчики

хотели убрать, и в первых заметках о создании языка была фраза: «Нужно исправить некоторые очевидные недостатки языка C, убрать…
+: ...возможность писать небезопасный код».
-: ...кое-какой очевидный хлам».
-: ...сборщик мусора».

В Go есть горутина — goroutine. С помощью goroutine вы можете конкурентно (не то же самое, что параллельно) выполнять функции в том же адресном пространстве. Что-то вроде запуска функции в «легковесном» треде. С версии 1.4 запуск одной goroutine занимает на старте…
+: ...2048 байт.
-: ...512 байт.
-: ...8192 байта.

Как думаете, где допущена опечатка?
```
func main() {
d := time.Now().Add(1 * time.Millisecond)
ctx, cancel := context.WithDeadline(context.Background(), d)
defer cancel()
select {
case <-time.After(1 * time.Second):
fmt.Println("Hi")
case <-ctx.Dead():
fmt.Println("How are you?")
}
}
```

-: вместо "context.WithDeadline" должно быть "context.Deadline".
+: вместо "ctx.Dead()" должно быть "ctx.Done()".
-: вместо "context.WithDeadline" должно быть "context.WithCancel".

В Go, если значение переменной не было задано, оно будет взято по умолчанию. Для неинициализированного слайса значение по умолчанию будет nil.
Например, для var a []string значение будет nil. Также можно создать пустой слайс var b []string{}. А какой json мы получим из этих слайсов?
```
var a []string
b := []string{}
var (
  err error
  aa []byte
  bb []byte
)
aa, err = json.Marshal(a)
bb, err = json.Marshal(b)
fmt.Printf("aa: %#v; bb: %#vn", string(aa), string(bb))
```

+: ... aa: "null"; bb: "[]"
-: ... aa: ""; bb: ""
-: ... aa: "[]"; bb: "[]"

Почти во всех языках есть конструкция «return», которая призвана вернуть контекст выполнения из функции/процедуры — то, что мы привыкли считать «возвращаемым

значением». В Golang для возврата значения тоже можно воспользоваться «return», но в отличие от других языков значение или имя переменной указывать не обязательно. Можно деклалировать имя или имена переменных при объявлении, а после необходимых операций в функции просто написать return и значения данных будут возвращены из функции.

А ещё в Go есть способ обмануть этот механизм возврата значений. Как?

```
func foo() (a *int) {
   return nil
}
```

```
func foo() (a int) {
   defer func() {
   a = 777
}()
   return 123
}
```

```
func foo() (a *int) {
   var v int
   return &v
}
```

-: Вариант 1
+: Вариант 2
-: Вариант 3

Популярность Go настолько велика, что трудно придумать, где он не используется. Угадаете топ-5 сфер применения Go?

+: веб разработка, базы данных, DevOps, системное программирование, безопасность.
-: веб разработка, e-commerce, DevOps, системное программирование, безопасность.
-: веб разработка, IoT, DevOps, DataScience, ML/AI.

В Go есть привычные массивы, но есть и другое понятие — слайс байт. Слайсы аналогичны массивами в других языках программирования, но со своими особенностями. Строка в Go — это тоже слайс байт. А вот для работы с одним символом используется тип…

-: ...char.
+:...rune.
-: ...string.

В Go есть бенчмарки для тестирования производительности программы или функции. Но бывает, что недостаточно иметь исходный код программы на языке Go, и нужно узнать, что происходит на более низком уровне — на уровне ассемблерного кода. Получить ассемблерный код из исходного кода программы на Go…

+...можно с помощью команды go tool compile -S main.go.
-: ....нельзя.
-: ...можно, но только из уже скомпилированного бинарного файла.

Go славится своей простотой, но для некоторых базовых вещей отсутствует элегантное решение. Например, для работы со слайсами не добавлены «базовые» операции над ним.

```
sl := []int{1, 3, 4}
n := 1
```

Поэтому, чтобы удалить второй элемент в этом слайсе, нужно использоваться код…

+: res := append(sl[:n], sl[n+1:]...)
-: sl.delete(n)
-: sl[n] = nil

Когда вы компилируете программу, вам может понадобиться зафиксировать некоторую информацию в итоговом бинарном файле. Например указать хеш коммита и ветку Git, а также версию собранной программы. Для этого в Go можно устанавливать непосредственно переменные пакета на этапе компиляции:
+: go build -ldflags -X package-import-path.val=foo'
-: это невозможно.
-: go build package-import-path.val=foo GOOS=linux GOARCH=arm

Компилятор Go может производить escape analysis, чтобы определить, будут ли данные размещены на heap. Для этого используется флаг -m: go tool compile -m main.go.
В каком из вариантов x определена на heap?
-: func foo() int {
  x := new(int)
  return *x
}

+: func bar() *int {
  x := new(int)
  return x
}

Критерии формирования оценок по тестовым заданиям:

По итогам выполнения тестовых заданий оценка производится по пятибалльной шкале. При правильных ответах на:

- 89-100% заданий – «5» (баллов);
- 70-88% заданий – «4» баллов);
- 50-69% заданий – «3» (балла);
- 30-49% заданий – «2» (балла);
- 10-29% заданий – «1» (балл);
- менее 10% заданий – «0» (баллов).

4. Экзаменационные вопросы «Профессиональная разработка программного обеспечения Golang»(контролируемая компетенция ПКС-2).

1. Начало программирования в Go — что нужно знать?
2. Что такое Go? Go Playground, пакеты, функции и скобки в Golang.
3. Основы: Работа с числами, форматирование строк, переменные и константы.
4. Цикл for, if-else-switch, True-False и операторы сравнения в Golang
5. Оператор switch в Golang.
6. Область видимости переменных в Golang.
7. Создание программы для покупки билетов в Golang.
8. Вещественные числа в Golang — float64 и float32.
9. Целые числа integer в Golang — выбор верного типа.
10. Пакет Big — Крупные числа в Golang и примеры их использования.
11. Работа со строками в Golang.
12. Конвертирование типов данных в Golang.

13. Создаем Шифр Виженера на Golang.
14. Функции в Golang на примерах.
15. Методы в Go — Создание и использование методов в Golang.
16. Функции первого класса, замыкания и анонимные функции в Golang.
17. Программа для перевода температуры из Цельсия в Фаренгейты.
18. Создание и итерация массива в Golang.
19. Срез массива в Golang.
20. Работа с массивами и срезами в Golang — append() и make().
21. Карта — ассоциативный массив в Golang.
22. Создание игры «Жизнь» в Golang.
23. Структуры в Golang — Экспорт структур в JSON.
24. Структуры и методы — объектно-ориентированный подход в Golang.
25. Композиция и встраивание методов в Golang.
26. Интерфейсы в Golang.
27. Создание игры-симулятора фермы в Golang.
28. Указатели в Golang.
29. Значение nil в Golang.
30. Обработка ошибок в Golang.
31. Создание игры Судоку в Golang.
32. Горутины и конкурентность — Многопоточность в Go.
33. Изучаем конкурентность и параллелизм в Golang.
34. Конкурентность в Go на примере создания игры для изучения Марса.
35. JSON в Golang — сериализация и десериализация.
36. Применение языка Go .
37. Плюсы языка Go. Минусы языка Go.
38. Переменные и арифметические операции, ввод/вывод данных
39. Комментарии. Константы
40. Условные выражения. Условные конструкции
41. Циклы. Форматированный вывод.
42. Массивы и срезы.

Форма экзаменационного билета
по учебной дисциплине

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное бюджетное образовательное учреждение высшего образования «Кабардино-Балкарский государственный университет им. Х.М. Бербекова» (КБГУ)

Кафедра– Прикладной математики и информатики
Дисциплина – «Профессиональная разработка программного обеспечения Golang»
Направление подготовки – 02.03.02 Фундаментальная информатика и информационные технологии 4 курс

Экзаменационный билет №1

1.
2.
3.

Руководитель ОПОП
к.ф.-м.н., доцент _____ М.М. Лафишева

И.о. зав. кафедрой ПМ и И
к.ф.-м.н., доцент _____ А.Р. Бечелова