<question> Что происходит, если не указать директиву package в Go-файле?

<variant> Произойдет ошибка компиляции

<variant> Скомпилируется, но вызовет предупреждение

<variant> Код скомпилируется, но не запустится

<variant> Будет создан анонимный пакет

<variant> Ничего страшного не произойдёт

<question> Какой тип значения имеет '\t' в Go?

<variant> rune

<variant> bool

<variant> string

<variant> int

<variant> byte

<question> Что делает функция fmt.Println()?

<variant> Выводит аргументы в консоль

<variant> Выполняет арифметическую операцию

<variant> Отправляет запрос в сеть

<variant> Сохраняет данные в файл

<variant> Выполняет сортировку

<question> Что произойдёт при вызове fmt.Println("Hello, Go!") без скобок после Println?

<variant> Ошибка компиляции

<variant> Всё сработает

<variant> Ошибка выполнения

<variant> Вывод будет пустым

<variant> Будет напечатано имя функции

<question> Какое ключевое слово используется для объявления переменной в Go?

<variant> var

<variant> let

<variant> define

<variant> val

<variant> new

<question> Какое значение по умолчанию у переменной int, если ей не присвоено значение?

<variant> 0

<variant> undefined

<variant> 1

<variant> null

<variant> false

<question> Что означает запись := в Go?

<variant> Короткое объявление переменной

<variant> Указание на функцию main

<variant> Условная проверка

<variant> Импорт внешнего пакета

<variant> Вызов сборщика мусора

<question> Какой результат будет у выражения 2 + 2 == 5?

<variant> false

<variant> 4

<variant> 5

<variant> true

<variant> Ошибка

<question> Какой тип данных используется для логических значений в Go?

<variant> bool

<variant> boolean

<variant> logic

<variant> truefalse

<variant> log

<question> Что делает функция reflect.TypeOf(x)?

<variant> Возвращает тип переменной x

<variant> Проверяет тип во время компиляции

<variant> Возвращает значение x

<variant> Меняет тип x

<variant> Удаляет переменную x

<question> Что произойдет при попытке умножить float64 и int без преобразования типов?

<variant> Ошибка компиляции

<variant> Предупреждение

<variant> Автоматическое преобразование

<variant> Результат будет float64

<variant> Произойдёт округление

<question> Что делает команда go fmt?

<variant> Форматирует код по стандартам Go

<variant> Проверяет синтаксис

<variant> Запускает программу

<variant> Удаляет пробелы

<variant> Выводит структуру пакета

<question> Какие имена функций и переменных доступны из других пакетов?

<variant> Только с верхнего регистра

<variant> Только начинающиеся с символа $

<variant> Только с нижнего регистра

<variant> Любые

<variant> Только переменные

<question> Что возвращает метод ReadString('\n')?

<variant> Строку и ошибку

<variant> Только ошибку

<variant> Только строку

<variant> Булевое значение

<variant> Число и строку

<question> Что произойдёт, если переменная в Go объявлена, но не используется?

<variant> Ошибка компиляции

<variant> Выдаст предупреждение

<variant> Будет игнорироваться

<variant> Программа упадёт в рантайме

<variant> Произойдёт утечка памяти

<question> Что делает log.Fatal(err) в Go?

<variant> Завершает программу после вывода ошибки

<variant> Выводит ошибку в лог и продолжает выполнение

<variant> Игнорирует ошибку

<variant> Только выводит ошибку в файл

<variant> Просто компилирует ошибку

<question> Как правильно преобразовать строку в float64 в Go?

<variant> strconv.ParseFloat("2.5", 64)

<variant> strconv.Float("2.5")

<variant> convert("2.5")

<variant> float64("2.5")

<variant> float("2.5")

<question> Какая функция удаляет пробелы и символы новой строки из строки?

<variant> strings.TrimSpace()

<variant> strings.Cut()

<variant> strings.Trim()

<variant> strings.RemoveNewlines()

<variant> strings.Clean()

<question> Какая команда используется для немедленного выхода из цикла?

<variant> break

<variant> continue

<variant> stop

<variant> exit

<variant> return

<question> Какой оператор используется для логического И в Go?

<variant> &&

<variant> and

<variant> &

<variant> &&&

<variant> +

<question> Что делает continue в цикле Go?

<variant> Переходит к следующей итерации цикла

<variant> Переходит к следующей команде после цикла

<variant> Повторяет всю программу

<variant> Прерывает выполнение цикла

<variant> Завершает функцию

<question> Как правильно объявить переменную и сразу присвоить ей значение?

<variant> a := 5

<variant> int a = 5

<variant> var a := 5

<variant> a = int(5)

<variant> define a 5

<question> Что выведет программа?

if true || false {

}

<variant> Yes

<variant> No

<variant> true

<variant> false

<variant> Ошибка компиляции

<question> Почему нужно избегать именования переменной как `error`?\*\*

<variant> Это может замаскировать встроенный тип

<variant> Это вызывает ошибку

<variant> Это зарезервированное слово

<variant> Это влияет на память

<variant> Это имя нельзя использовать

<question> Как объявляется простая функция без параметров и возвращаемых значений в Go?

<variant> func main() {}

<variant> def main():

<variant> function main() {}

<variant> main func() {}

<variant> main := func() {}

<question> Что делает функция fmt.Printf("Value: %0.2f", 3.14159)?

<variant> Выводит число с двумя знаками после запятой

<variant> Вызывает ошибку

<variant> Выводит число с округлением до 0 знаков

<variant> Возвращает строку

<variant> Не форматирует вообще

<question> Какой глагол форматирования используется для вывода строки?

<variant> %s

<variant> %d

<variant> %f

<variant> %b

<variant> %t

<question> Какой тип возвращает функция strconv.ParseFloat помимо числа?

<variant> error

<variant> string

<variant> int

<variant> bool

<variant> nil

<question> Что будет выведено?

fmt.Printf("%#v", "\t")

<variant> "\t"

<variant> #v

<variant> tab

<variant> \t

<variant> Ошибка

<question> Какой тип у значения &myInt, если myInt типа int?

<variant> \*int

<variant> int

<variant> pointer

<variant> &int

<variant> ref int

<question> Как получить значение, на которое указывает указатель ptr?

<variant> \*ptr

<variant> ptr.get()

<variant> ref(ptr)

<variant> ptr\*

<variant> ptr.value

<question> Какой результат работы функции negate(myBool bool) bool с аргументом true?

<variant> false

<variant> 0

<variant> true

<variant> nil

<variant> Ошибка

<question> Какой способ передачи используется в Go по умолчанию?

<variant> По значению

<variant> По имени

<variant> По ссылке

<variant> По типу

<variant> По указателю

<question> Что произойдёт, если в функции не использовать переменную, которой присвоено значение ошибки?

<variant> Ошибка компиляции

<variant> Ничего

<variant> Предупреждение

<variant> Игнорирование

<variant> Значение будет равно nil

<question> Что делает errors.New("example error")?

<variant> Создаёт значение ошибки

<variant> Возвращает nil

<variant> Запускает исключение

<variant> Форматирует строку

<variant> Завершает программу

<question> Как объявить функцию, возвращающую два значения: int и error?

<variant> func f() (int, error)

<variant> func f() int, error

<variant> func f() -> (int, error)

<variant> def f(): int, error

<variant> func f() int error

<question> Что произойдёт, если изменить значение по указателю внутри функции?

<variant> Оригинальное значение изменится

<variant> Изменится только копия

<variant> Ошибка компиляции

<variant> Ничего

<variant> Только лог будет обновлён

<question> Какой способ передачи нужен, чтобы функция изменила оригинальную переменную?

<variant> Передача по указателю

<variant> Передача по типу

<variant> Передача через return

<variant> Передача значения

<variant> Использование глобальных переменных

<question> Какое имя должен иметь файл пакета, чтобы он компилировался как исполняемый?

<variant> Любое имя, но в пакете main

<variant> main.go

<variant> run.go

<variant> start.go

<variant> Только main.go

<question> Что делает команда go install?

<variant> Компилирует и сохраняет бинарник в папке bin

<variant> Создает и запускает программу

<variant> Устанавливает Go

<variant> Удаляет пакет

<variant> Создаёт новый проект

<question> Где в рабочей области Go хранятся исходные коды пакетов?

<variant> В папке src

<variant> В папке lib

<variant> В папке pkg

<variant> В папке bin

<variant> В папке run

<question> Как экспортировать функцию из пакета?

<variant> Назвать функцию с заглавной буквы

<variant> Добавить export перед функцией

<variant> Использовать префикс Exp\_

<variant> Назвать её с маленькой буквы

<variant> Вынести в отдельный файл

<question> Какой правильный способ вызвать функцию Hello() из пакета greeting?

<variant> greeting.Hello()

<variant> call greeting.Hello()

<variant> main.Hello()

<variant> Hello()

<variant> pkg.Hello()

<question> Как импортировать пакет, находящийся по пути github.com/headfirstgo/keyboard?

<variant> import "github.com/headfirstgo/keyboard"

<variant> import "keyboard"

<variant> import github.keyboard

<variant> import keyboard

<variant> use github.com/headfirstgo/keyboard

<question> Что такое GOPATH?

<variant> Переменная среды для рабочей области Go

<variant> Путь к бинарникам Go

<variant> Утилита для компиляции

<variant> Команда запуска

<variant> Путь до исполняемых файлов

<question> Где по умолчанию находится рабочая область Go?

<variant> ~/go

<variant> /etc/go

<variant> /usr/go

<variant> /var/go

<variant> /home/go/go

<question> Что делает команда go get?

<variant> Загружает и устанавливает пакет из интернета

<variant> Удаляет пакет

<variant> Показывает путь к пакету

<variant> Собирает зависимости

<variant> Проверяет версию

<question> Что произойдёт, если в Go-директиве import указать некорректный путь к пакету?

<variant> Произойдёт ошибка компиляции

<variant> Выдаст предупреждение

<variant> Пакет всё равно загрузится

<variant> Произойдёт ошибка выполнения

<variant> Путь будет проигнорирован

<question> Какой префикс используется для документации Go, которую можно увидеть в go doc?

<variant> Комментарии //, перед package или функцией

<variant> @doc

<variant> #

<variant> ///

<variant> doc:

<question> Какой командой можно просмотреть документацию по функции ParseFloat?

<variant> go doc strconv ParseFloat

<variant> \*godoc strconv ParseFloat

<variant> doc strconv.ParseFloat()

<variant> go help ParseFloat

<variant> go run strconv doc

<question> Что делает команда godoc -http=:6060?

<variant> Запускает сервер документации HTML

<variant> Запускает сервер Go Playground

<variant> Запускает Go-модуль

<variant> Показывает структуру каталогов

<variant> Делает fetch последних изменений

<question> Какие функции будут видны вне пакета?

<variant> Только те, что экспортированы (с заглавной буквы)

<variant> Все, без исключения

<variant> Только те, что внутри main()

<variant> Только приватные функции

<variant> Функции, помеченные public

<question> Что произойдёт, если две программы используют одинаковое имя пакета в одной рабочей области?

<variant> Возникнет конфликт

<variant> Используется последний загруженный

<variant> Go создаст дубликат

<variant> Go автоматически переименует

<variant> Всё будет работать нормально

<question> Какой индекс у первого элемента массива в Go?

<variant> 0

<variant> 1

<variant> -1

<variant> Первый индекс зависит от типа

<variant> Go не поддерживает индексацию

<question> Что произойдёт при обращении к элементу массива с недопустимым индексом?

<variant> Паника (runtime error)

<variant> Ошибка компиляции

<variant> Переход к следующему элементу

<variant> Возврат nil

<variant> Возврат пустого значения

<question> Какой тип значения по умолчанию у элементов int в массиве?

<variant> 0

<variant> nil

<variant> undefined

<variant> false

<variant> int64

<question> Как правильно создать массив из 3 строк и сразу присвоить значения?

<variant> var strings = [3]string{"a", "b", "c"}

<variant> strings := [string]{"a", "b", "c"}

<variant> var strings = {"a", "b", "c"}

<variant> strings := "a", "b", "c"

<variant> var strings = string[3]{"a", "b", "c"}

<question> Какая функция возвращает количество элементов в массиве?

<variant> len()

<variant> count()

<variant> size()

<variant> length()

<variant> cap()

<question> Какая форма цикла позволяет безопасно перебирать массивы?

<variant> for...range

<variant> foreach

<variant> for i := range

<variant> while

<variant> loop

<question> Что делает ключевое слово \_ в цикле for \_, value := range arr?

<variant> Позволяет избежать ошибки «переменная не используется»

<variant> Используется как имя массива

<variant> Пропускает значение

<variant> Вызывает функцию

<variant> Преобразует тип

<question> Что произойдёт при чтении строки из файла с помощью bufio.Scanner?

<variant> Читается строка текста

<variant> Возврат os.File

<variant> Сканер вызывает panic

<variant> Возврат []byte

<variant> Строка сразу компилируется

<question> Какой тип возвращает strconv.ParseFloat(str, 64) при ошибке?

<variant> error

<variant> 0.0

<variant> nil

<variant> false

<variant> panic

<question> Какой размер у массива [3]float64?

<variant> Фиксированный - 3

<variant> Определяется во время выполнения

<variant> Неограниченный

<variant> Может изменяться динамически

<variant> Зависит от длины файла

<question> Что произойдёт, если в GetFloats() в файле окажется больше 3 строк?

<variant> Возникнет паника из-за выхода за пределы массива

<variant> Файл будет перезаписан

<variant> Произойдёт ошибка компиляции

<variant> Будет считано только первое значение

<variant> Функция проигнорирует лишние строки

<question> Что произойдёт, если забыть закрыть файл после os.Open()?

<variant> Утечка ресурсов

<variant> Ничего

<variant> Ошибка компиляции

<variant> Файл удалится

<variant> Программа перезапустится

<question> Как можно запустить документацию для всех пакетов в браузере локально?

<variant> godoc -http=:6060

<variant> go doc

<variant> go browser

<variant> go run doc.go

<variant> go info

<question> В чём основное отличие сегмента (slice) от массива (array) в Go?

<variant> Сегмент может изменять размер

<variant> Сегмент нельзя использовать в цикле

<variant> У массива нет нулевых значений

<variant> Массив быстрее

<variant> Сегмент работает только с числами

<question> Как создать сегмент целых чисел длиной 5 с помощью make?

<variant> make([]int, 5)

<variant> int[5]

<variant> make(int, 5)

<variant> []int(5)

<variant> make(slice, 5)

<question> Что делает функция append в Go?

<variant> Добавляет элемент в конец сегмента

<variant> Копирует один массив в другой

<variant> Добавляет элемент в конец массива

<variant> Удаляет элемент

<variant> Заменяет значение по индексу

<question> Что вернёт len(slice) если slice == nil?

<variant> 0

<variant> -1

<variant> ошибка компиляции

<variant> 1

<variant> panic

<question> Что произойдёт при append(nil, 10)?

<variant> Вернётся сегмент с элементом 10

<variant> Вернётся nil

<variant> Ошибка

<variant> Вернётся пустой массив

<variant> Будет panic

<question> Как получить все элементы массива arr, начиная с индекса 2 до конца?

<variant> arr[2:len(arr)]

<variant> arr[2]

<variant> arr[:2]

<variant> arr[0:2]

<variant> arr[-1:]

<question> Что произойдёт, если попытаться получить slice[6] из slice := make([]int, 5)?

<variant> panic

<variant> Ошибка компиляции

<variant> Вернётся 0

<variant> nil

<variant> undefined

<question> Какой тип у переменной os.Args?

<variant> []string

<variant> []int

<variant> []byte

<variant> string

<variant> os.Slice

<question> Что делает os.Args[1:]?

<variant> Возвращает аргументы без имени программы

<variant> Удаляет последний аргумент

<variant> Обрезает строку

<variant> Возвращает длину аргументов

<variant> Ошибка

<question> Как вызвать функцию sum с переменным числом аргументов из сегмента nums?

<variant> sum(nums...)

<variant> sum(...nums)

<variant> sum(nums)

<variant> sum[\*nums]

<variant> sum(...int)

<question> Какая функция позволяет прочитать строки из файла построчно?

<variant> bufio.NewScanner(file)

<variant> os.ScanFile

<variant> strings.ReadLine

<variant> fmt.Scanln(file)

<variant> reader.Lines()

<question> Что делает следующая функция?

func average(nums ...float64) float64 {

sum += n

}

<variant> Возвращает среднее значение

<variant> Делит первое на второе

<variant> Возвращает максимальное число

<variant> Выводит числа

<variant> Считает сумму чисел

<question> Что такое карта (map) в Go?

<variant> Коллекция пар ключ–значение

<variant> Упорядоченный список значений

<variant> Коллекция значений с индексами

<variant> Двоичное дерево

<variant> Массив без размера

<question> Как создать пустую карту с ключами типа string и значениями типа int?

<variant> make(map[string]int)

<variant> new(map[string]int)

<variant> make(map[int]string)

<variant> map[string]int{0}

<variant> []string{}

<question> Как присвоить значение 42 ключу "age" в карте info?

<variant> info["age"] = 42

<variant> info.age = 42

<variant> info[42] = "age"

<variant> set(info, "age", 42)

<variant> append(info, "age", 42)

<question> Что произойдёт при чтении значения по отсутствующему ключу?

<variant> Значение по умолчанию

<variant> Ошибка

<variant> nil

<variant> Компиляция невозможна

<variant> Выводится undefined

<question> Какое нулевое значение у карты map[string]float64?

<variant> nil

<variant> 0.0

<variant> 0

<variant> map{}

<variant> empty

<question> Как проверить, был ли ключ действительно установлен в карте?

<variant> Использовать value, ok := map[key]

<variant> Использовать map[key] != nil

<variant> Использовать двойной индекс map[[key]]

<variant> Использовать len(map[key])

<variant> Это невозможно

<question> Что выведет следующий код?

m := map[string]int{}

m["a"]++

fmt.Println(m["a"])

<variant> 1

<variant> 0

<variant> Ошибка

<variant> nil

<variant> true

<question> Как удалить ключ из карты?

<variant> delete(m, "key")

<variant> delete("key", m)

<variant> m.remove("key")

<variant> m["key"] = nil

<variant> del m["key"]

<question> Какой тип возвращает ok в конструкции value, ok := map[key]?

<variant> bool

<variant> string

<variant> error

<variant> int

<variant> interface{}

<question> Что произойдёт при попытке записи в nil-карту?

<variant> Будет panic во время выполнения

<variant> Ошибка компиляции

<variant> Запишется как есть

<variant> Ничего

<variant> Вернёт nil

<question> Какой из вариантов — корректный литерал карты?

<variant> map[string]int{"a": 1, "b": 2}

<variant> {"a": 1, "b": 2}

<variant> []map{"a": 1}

<variant> map{1: "a", 2: "b"}

<variant> map<string, int>{"a": 1}

<question> Что делает for k, v := range myMap {}?

<variant> Итерация по карте

<variant> Итерация по строке

<variant> Итерация по массиву

<variant> Ошибка

<variant> Сортировка карты

<question> Что нужно сделать, чтобы отсортировать вывод ключей карты?

<variant> Вынести ключи в сегмент и отсортировать

<variant> Сортировать карту напрямую

<variant> Использовать sort.Map()

<variant> Использовать map.Sort()

<variant> Карты упорядочены по умолчанию

<question> Что выведет этот код?

grades := map[string]float64{"Alma": 0, "Carl": 0}

grade, ok := grades["Rohit"]

fmt.Println(grade, ok)

<variant> 0 false

<variant> 0 true

<variant> nil true

<variant> panic

<variant> false 0

<question> Что делает map[string]int{}?

<variant> Создаёт пустую, но готовую к использованию карту

<variant> Создаёт nil-карту

<variant> Создаёт карту с нулевыми ключами

<variant> Не компилируется

<variant> Очищает карту

<question> Для чего используются структуры (struct) в Go?

<variant> Для хранения данных разных типов вместе

<variant> Для хранения строк

<variant> Для создания массивов

<variant> Для повторения значений

<variant> Только для интерфейсов

<question> Как объявляется структура в Go?

<variant> type x struct { field string }

<variant> define struct {}

<variant> type x = struct()

<variant> new struct {}

<variant> x := struct [field string]

<question> Как обратиться к полю rate структуры subscriber?

<variant> subscriber.rate

<variant> subscriber[rate]

<variant> subscriber:rate

<variant> subscriber->rate

<variant> subscriber::rate

<question> Какой способ позволяет определить новый тип на основе структуры?

<variant> type subscriber struct { ... }

<variant> define struct

<variant> new struct

<variant> make struct

<variant> var subscriber struct

<question> Что происходит при передаче структуры в функцию по значению?

<variant> Передаётся копия структуры

<variant> Значения изменяются глобально

<variant> Возникает ошибка

<variant> Функция получает указатель

<variant> Передаётся карта значений

<question> Как изменить значение поля структуры внутри функции?

<variant> Передать указатель на структуру

<variant> Использовать global

<variant> Объявить переменную внутри функции

<variant> Использовать встроенные функции Go

<variant> Это невозможно

<question> Какой синтаксис используется для получения указателя на переменную?

<variant> &var

<variant> pointer var

<variant> \*var

<variant> ref var

<variant> @var

<question> Какой результат даст следующий код?

type car struct { speed int }

func boost(c car) { c.speed += 50 }

func main() {

var mustang car

mustang.speed = 100

boost(mustang)

fmt.Println(mustang.speed)

}

<variant> 100

<variant> 150

<variant> Ошибка

<variant> 0

<variant> nil

<question> Что произойдёт при попытке записи в поле структуры через nil-указатель?

<variant> Произойдёт panic

<variant> Возникнет ошибка компиляции

<variant> Значение будет присвоено

<variant> Будет создана структура

<variant> nil заменится на 0

<question> Как экспортировать тип или поле структуры для использования в другом пакете?

<variant> Использовать имя с заглавной буквы

<variant> Начать имя с символа \_

<variant> Использовать ключевое слово export

<variant> Добавить тег public

<variant> Сначала импортировать поле

<question> Что делает литерал структуры?

<variant> Позволяет создать структуру с инициализацией полей

<variant> Инициализирует структуру без указания полей

<variant> Запускает программу

<variant> Вызывает метод

<variant> Удаляет структуру

<question> Что такое анонимное поле структуры?

<variant> Поле, у которого указан только тип без имени

<variant> Поле с пустым значением

<variant> Поле, у которого указано только имя

<variant> Поле без типа

<variant> Скрытое поле, видимое только в рамках пакета

<question> Что происходит с полями анонимной структуры?

<variant> Они объединяются с основными полями

<variant> Они недоступны

<variant> Их нужно называть через точку

<variant> Они имеют тип interface{}

<variant> Они работают только в main

<question> Как объявить структуру, в которую встроена другая структура Address?

<variant> type S struct { Address }

<variant> type S struct { Address Address }

<variant> type S struct { Address field Address }

<variant> type S struct { } = Address

<variant> type S embed Address

<question> Что такое определяемый тип (custom type) в Go?

<variant> Новый тип, основанный на существующем типе

<variant> Указатель на существующий тип

<variant> Тип, автоматически сгенерированный компилятором

<variant> Тип, объявленный внутри функции

<variant> Только интерфейс

<question> Как создать определяемый тип Gallons, основанный на float64?

<variant> type Gallons float64

<variant> Gallons := float64

<variant> float64 := Gallons

<variant> var Gallons = type(float64)

<variant> struct Gallons float64

<question> Что произойдёт, если попытаться определить две функции с одинаковыми именами и разными параметрами в одном пакете?

<variant> Возникнет ошибка

<variant> Будет использована первая

<variant> Будет использована последняя

<variant> Всё скомпилируется

<variant> Зависит от компилятора

<question> Что отличает метод от обычной функции в Go?

<variant> Метод имеет параметр-получатель

<variant> Метод использует self

<variant> Метод определён внутри функции

<variant> Метод может быть вызван только один раз

<variant> Метод вызывается только через указатель

<question> Какой синтаксис соответствует определению метода для типа Liters?

<variant> func (l Liters) ToGallons() Gallons

<variant> func ToGallons(l Liters) Gallons

<variant> method (l Liters) ToGallons() Gallons

<variant> func Liters.ToGallons() Gallons

<variant> Liters func ToGallons() Gallons

<question> Какой приём используется для передачи по ссылке в метод, чтобы изменить получатель?

<variant> Использование указателя в параметре получателя

<variant> Передача значения как параметра

<variant> Передача значения по значению

<variant> Использование типа interface{}

<variant> Использование return-значения

<question> Что произойдёт при вызове метода с получателем-указателем на литерал (например, MyType("x").Method())?

<variant> Возникает ошибка

<variant> Всё работает

<variant> Вернётся nil

<variant> Будет создана временная переменная

<variant> Будет возвращено значение по умолчанию

<question> Какая команда верна для вызова метода Double() с получателем-указателем?

<variant> value.Double()

<variant> \*value.Double()

<variant> Double(&value)

<variant> value->Double()

<variant> value\*&.Double()

<question> Что делает следующий метод?

func (n Number) Add(x int) {

fmt.Println(n + Number(x))

}

<variant> Выполняет преобразование x в тип Number и складывает

<variant> Сравнивает значения

<variant> Складывает int и Number напрямую

<variant> Ошибка типов

<variant> Умножает значения

<question> Какой результат работы кода?

type Liters float64

func (l Liters) ToMilliliters() Milliliters {

return Milliliters(l \* 1000)

}

<variant> Преобразует Liters в Milliliters

<variant> Ошибка типов

<variant> Возвращает значение типа float64

<variant> Возвращает строку

<variant> Ничего не делает

<question> Какой подход Go использует вместо перегрузки функций?

<variant> Методы с уникальными именами или разными типами получателей

<variant> Интерфейсы

<variant> Инкапсуляция

<variant> Использование self

<variant> Использование inline-функций

<question> Что такое инкапсуляция в Go?

<variant> Сокрытие данных и предоставление доступа через методы

<variant> Сокрытие реализации с помощью struct

<variant> Использование интерфейсов для методов

<variant> Использование структур без методов

<variant> Передача значений по указателю

<question> Какой способ используется в Go для сокрытия полей структуры от других пакетов?

<variant> Начинать имя поля с маленькой буквы

<variant> Использование типа interface{}

<variant> Префикс \_

<variant> Объявление через private

<variant> Использовать @internal

<question> Какой синтаксис соответствует определению get-метода Year?

<variant> func (d \*Date) Year() int

<variant> func (d Date) GetYear() int

<variant> func Year(d \*Date) int

<variant> func Year() (d \*Date) int

<variant> func (Year d) \*Date int

<question> Что произойдёт при попытке обратиться к неэкспортируемому полю из другого пакета?

<variant> Ошибка компиляции

<variant> Значение будет 0

<variant> Возврат nil

<variant> panic

<variant> Значение присвоится автоматически

<question> Зачем в Go используют set-методы?

<variant> Чтобы проверять допустимость данных перед сохранением

<variant> Чтобы увеличить производительность

<variant> Чтобы обойти типизацию

<variant> Чтобы улучшить читаемость

<variant> Для отладки

<question> Какой тип параметра должен иметь set-метод, чтобы изменить значение структуры?

<variant> Указатель на структуру

<variant> struct

<variant> interface{}

<variant> Значение (значение по умолчанию)

<variant> Метод без параметров

<question> Какое имя будет у метода, возвращающего значение поля month?

<variant> Month()

<variant> GetMonth()

<variant> setMonth()

<variant> get\_month()

<variant> MonthGetter()

<question> Что означает «повышение метода» при встраивании структур?

<variant> Метод внешнего типа получает доступ к методам встроенного типа

<variant> Метод удаляется

<variant> Метод скрывается

<variant> Метод заменяется

<variant> Метод становится глобальным

<question> Что произойдёт, если в структуру Event встроен тип Date с неэкспортируемыми полями, но с экспортируемыми методами?

<variant> Только методы будут повышены

<variant> Поля и методы повысятся

<variant> Методы не будут доступны

<variant> Ошибка компиляции

<variant> Методы будут видны только внутри Date

<question> Что произойдёт при вызове метода SetTitle("Очень длинное название...")?

<variant> Вернётся ошибка invalid title

<variant> Название будет установлено

<variant> Название будет обрезано

<variant> Метод вызовет panic

<variant> Возвратится пустая строка

<question> Какой пакет используется для подсчёта символов в строке?

<variant> utf8

<variant> text

<variant> runes

<variant> strings

<variant> math

<question> Какой метод следует определить, чтобы вернуть поле title из Event?

<variant> func (e \*Event) Title() string

<variant> func (e Event) title() string

<variant> func GetTitle() string

<variant> func (Event) Title()

<variant> func Title(e \*Event) string

<question> Какой из примеров демонстрирует вызов метода, повышенного из встроенного типа?

<variant> event.SetYear()

<variant> event.Date.Year()

<variant> event.date.SetYear()

<variant> Date(event).SetYear()

<variant> SetYear(event)

<question> Что такое интерфейс в Go?

<variant> Тип, описывающий набор методов

<variant> Конкретная структура с методами

<variant> Специальный файл проекта

<variant> Тип, описывающий данные

<variant> Метод с указателем

<question> Какой тип поддерживает интерфейс?

<variant> Тот, который содержит все методы, указанные в интерфейсе

<variant> Любой, объявленный через type interface{}

<variant> Тот, у которого нет методов

<variant> Только тип struct

<variant> Только указатели

<question> Какой метод должен быть в типе, чтобы он поддерживал интерфейс error?

<variant> Error() string

<variant> Log() error

<variant> Report(string)

<variant> String() string

<variant> PrintError()

<question> Что произойдёт, если вызвать метод, не определённый в интерфейсе?

<variant> Ошибка компиляции

<variant> Он вызовется

<variant> Ошибка выполнения

<variant> Паника

<variant> Метод вызовется, если тип содержит его

<question> Что произойдёт при использовании утверждения типа без ok-проверки, если тип не совпал?

<variant> Произойдёт паника

<variant> Вернётся значение по умолчанию

<variant> Вернётся nil

<variant> Ошибка компиляции

<variant> Ничего

<question> Как безопасно использовать утверждение типа?

<variant> С val, ok := value.(Type)

<variant> С \_, err := value.(Type)

<variant> С использованием interface{}

<variant> С преобразованием Type(value)

<variant> Только через указатель

<question> Что делает пустой интерфейс interface{}?

<variant> Позволяет использовать любое значение любого типа

<variant> Поддерживается только типом string

<variant> Ничего

<variant> Позволяет использовать только методы без параметров

<variant> Используется только с числами

<question> Что можно сделать со значением типа interface{} без утверждения типа?

<variant> Передать в fmt.Println()

<variant> Преобразовывать без проверки

<variant> Вызывать любые методы

<variant> Выполнить арифметику

<variant> Присвоить nil

<question> Что произойдёт, если тип реализует метод String() string?

<variant> Будет поддерживать fmt.Stringer

<variant> Будет поддерживать error

<variant> Не сможет быть передан в fmt.Println()

<variant> Строки станут неизменяемыми

<variant> Тип станет приватным

<question> Какой результат будет у кода fmt.Println(value) при условии, что тип value реализует Stringer?

<variant> Вызывается метод String()

<variant> Выводится адрес

<variant> Выводится тип

<variant> Выводится значение по умолчанию

<variant> Ошибка

<question> Что означает результат ok == false после утверждения типа?

<variant> Тип не соответствует

<variant> Тип был найден

<variant> Утверждение прошло успешно

<variant> Значение nil

<variant> Метод найден, но не вызывается

<question> Что делает ключевое слово defer в Go?

<variant> Откладывает выполнение функции до конца текущей функции

<variant> Прерывает выполнение функции

<variant> Отменяет выполнение функции

<variant> Повторяет вызов функции

<variant> Выполняет функцию сразу

<question> Когда вызывается отложенная функция?

<variant> После выхода из функции, где она объявлена

<variant> Перед каждой итерацией цикла

<variant> В момент компиляции

<variant> Сразу после объявления

<variant> В начале выполнения main

<question> Что произойдёт, если внутри функции возникнет panic, а до этого был указан defer?

<variant> defer выполнится

<variant> defer не сработает

<variant> panic отменит defer

<variant> Программа зависнет

<variant> Все переменные обнулятся

<question> Что делает функция recover()?

<variant> Останавливает панику и возвращает значение

<variant> Перезапускает программу

<variant> Завершает программу

<variant> Обрабатывает ошибки компиляции

<variant> Перехватывает return

<question> Что произойдёт при вызове recover() вне контекста panic?

<variant> Вернёт nil

<variant> Будет вызвана panic

<variant> Ошибка компиляции

<variant> Вернёт false

<variant> Перезапустит программу

<question> Как можно инициировать panic вручную?

<variant> panic("ошибка")

<variant> raise("ошибка")

<variant> throw "ошибка"

<variant> log.Fatal("ошибка")

<variant> exit("ошибка")

<question> Что происходит с отложенными вызовами при panic?

<variant> Выполняются в обратном порядке

<variant> Выполняются в прямом порядке

<variant> Они удаляются

<variant> Пропускаются

<variant> Сохраняются до перезапуска

<question> Что вернёт recover(), если panic("ошибка") был вызван?

<variant> ошибка

<variant> nil

<variant> panic

<variant> true

<variant> false

<question> Для чего используется утверждение типа внутри recover()?

<variant> Для определения, является ли значение ошибкой

<variant> Для удаления ошибки

<variant> Для преобразования значения в строку

<variant> Для перезапуска

<variant> Для восстановления стек-трейса

<question> Что произойдёт, если в recover() передано значение, не являющееся error?

<variant> Нужно использовать повторную панику

<variant> Возникнет ошибка

<variant> Оно будет преобразовано в строку

<variant> Оно будет проигнорировано

<variant> Вернётся nil

<question> Что произойдёт при вызове scanDirectory, если каталог недоступен, а используется panic и recover?

<variant> Программа завершится корректно, выведя сообщение об ошибке

<variant> Программа зависнет

<variant> Программа завершится с ошибкой

<variant> Ошибка проигнорируется

<variant> Вызовется повторный panic

<question> Что делает ioutil.ReadDir()?

<variant> Читает содержимое каталога

<variant> Возвращает путь к каталогу

<variant> Читает строку из файла

<variant> Записывает в файл

<variant> Закрывает поток

<question> Что делает filepath.Join()?

<variant> Соединяет два сегмента файла с разделителем /

<variant> Объединяет строки через пробел

<variant> Удаляет каталог

<variant> Копирует файл

<variant> Форматирует имя файла

<question> Как рекурсивно обойти все подкаталоги в Go?

<variant> С использованием рекурсивной функции

<variant> Через goto

<variant> С помощью os.ReadAll()

<variant> Через интерфейс

<variant> Это невозможно

<question> Что происходит при вызове panic в рекурсивной функции без recover?

<variant> Вся программа аварийно завершается

<variant> Все уровни рекурсии продолжают работу

<variant> Только текущий вызов прерывается

<variant> Вызовется recover автоматически

<variant> Ошибка будет проигнорирована

<question> Что такое горутина в Go?

<variant> Лёгковесный поток выполнения

<variant> Интерфейс

<variant> Отдельная программа

<variant> Метод структуры

<variant> Ошибка компиляции

<question> Как запустить функцию work() в горутине?

<variant> go work()

<variant> run work()

<variant> work.run()

<variant> start work()

<variant> launch(work)

<question> Что происходит при завершении горутины main?

<variant> Все горутины завершаются

<variant> Программа переходит в спящий режим

<variant> Продолжают работать другие горутины

<variant> Компилятор запускается заново

<variant> Переход к следующей функции

<question> Как можно предотвратить преждевременное завершение main?

<variant> Вызвать time.Sleep()

<variant> Добавить цикл for

<variant> Использовать recover()

<variant> Использовать интерфейс

<variant> Объявить глобальную переменную

<question> Что нельзя делать с функцией, вызываемой через go?

<variant> Возвращать значение

<variant> Использовать указатели

<variant> Импортировать пакеты

<variant> Обрабатывать ошибки

<variant> Использовать интерфейсы

<question> Что делает канал в Go?

<variant> Передает данные между горутинами

<variant> Вызывает методы

<variant> Запускает горутину

<variant> Создает массивы

<variant> Подключает модули

<question> Как объявить канал для передачи int?

<variant> make(chan int)

<variant> channel(int)

<variant> int <- chan

<variant> chan int

<variant> int chan

<question> Как отправить значение 5 в канал ch?

<variant> ch <- 5

<variant> ch = 5

<variant> send(5, ch)

<variant> <-ch = 5

<variant> ch.push(5)

<question> Как получить значение из канала ch?

<variant> <-ch

<variant> get(ch)

<variant> ch <-

<variant> read(ch)

<variant> fetch(ch)

<question> Что произойдёт, если в канале нет значений, а горутина пытается их получить?

<variant> Горутина будет заблокирована

<variant> Программа завершится

<variant> Ошибка компиляции

<variant> Значение будет равно 0

<variant> Значение заменится на nil

<question> Что обеспечивает синхронизацию горутин при использовании каналов?

<variant> Блокировка операций отправки и получения

<variant> Пакет log

<variant> Интерфейсы

<variant> Указатели

<variant> Массивы

<question> Что произойдёт, если в функции main вызвать <-ch, но ни одна горутина не отправит значение в канал ch?

<variant> Возникнет deadlock

<variant> Возникнет ошибка

<variant> Программа выполнится мгновенно

<variant> Значение будет nil

<variant> Программа завершится без вывода

<question> Как можно передать два значения из горутины через канал?

<variant> Все перечисленное

<variant> Использовать send(ch, a, b)

<variant> Использовать []interface{}

<variant> Упаковать значения в структуру

<variant> Дважды вызвать ch <- value

<question> Какой результат даст код: go fmt.Println("hi")?

<variant> hi, если main не завершится раньше

<variant> hi с гарантией

<variant> Ничего

<variant> Ошибка компиляции

<variant> main автоматически приостановится

<question> Что делает этот код?

type Page struct {

}

<variant> Создаёт структуру

<variant> Создаёт определяемый тип

<variant> Создаёт интерфейс

<variant> Создаёт канал

<variant> Создаёт горутину

<question> Зачем нужны автоматизированные тесты?

<variant> Для проверки, что изменения не сломали существующий функционал

<variant> Для создания пользовательского интерфейса

<variant> Для автоматической генерации кода

<variant> Для повышения скорости компиляции

<variant> Для очистки кэша

<question> Какой командой выполняются тесты в Go?

<variant> go test

<variant> run test.go

<variant> go build

<variant> go run test

<variant> test run

<question> Какой суффикс должен иметь файл, содержащий тесты?

<variant> \_test.go

<variant> .test.go

<variant> \_unit.go

<variant> testfile.go

<variant> .go.test

<question> Какой тип должен получать функция теста?

<variant> \*testing.T

<variant> \*Test

<variant> t.Testing

<variant> T.error

<variant> go.test

<question> Какую функцию используют для регистрации ошибки без форматирования?

<variant> t.Error()

<variant> t.FailNow()

<variant> fmt.Print()

<variant> t.Errorf()

<variant> t.Fatal()

<question> Как вызвать функцию с сообщением об ошибке с форматированием?

<variant> t.Errorf()

<variant> t.Fatal()

<variant> t.Error()

<variant> t.Printf()

<variant> fmt.Error()

<question> Какой результат у выражения JoinWithCommas([]string{"apple", "orange", "pear"})?

<variant> "apple, orange, and pear"

<variant> "apple and orange and pear"

<variant> "apple orange pear"

<variant> "apple, orange and pear"

<variant> "apple: orange: pear"

<question> Что такое табличный тест в Go?

<variant> Серия тестов, в которых входные данные и ожидаемые результаты хранятся в структуре

<variant> Тест, оформленный в виде таблицы в HTML

<variant> Отчет о производительности

<variant> Тест, который работает только с базами данных

<variant> Вывод таблицы значений

<question> Что произойдет, если вызвать JoinWithCommas с пустым срезом?

<variant> Вернёт пустую строку

<variant> Возникнет panic

<variant> Вернёт nil

<variant> Вернёт "and"

<variant> Ошибка компиляции

<question> Зачем использовать вспомогательные функции в \_test.go файле?

<variant> Чтобы уменьшить дублирование кода

<variant> Чтобы избежать конфликтов с main

<variant> Для ускорения выполнения

<variant> Для вызова внешних API

<variant> Чтобы убрать необходимость компиляции

<question> Какой параметр используется в флаге go test -run ...?

<variant> Шаблон имени тестовой функции

<variant> Имя файла

<variant> Имя пакета

<variant> Адрес модуля

<variant> Имя переменной

<question> Что делает команда go test -v?

<variant> Выводит подробную информацию о каждом выполненном тесте

<variant> Показывает только завершившиеся неудачей тесты

<variant> Удаляет старые результаты тестов

<variant> Переводит тесты в интерактивный режим

<variant> Обновляет документацию

<question> Почему go build и go install игнорируют файлы \_test.go?

<variant> Потому что они не предназначены для компиляции в конечный исполняемый файл

<variant> Потому что они содержат ошибки

<variant> Потому что они написаны на другом языке

<variant> Потому что они только для Windows

<variant> Потому что они содержат цикл

<question> Какой пакет используется в Go для создания веб-приложений?

<variant> net/http

<variant> fmt/http

<variant> html/net

<variant> web

<variant> io/web

<question> Какой порт использовался в примере для запуска локального веб-сервера?

<variant> 8080

<variant> 3000

<variant> 80

<variant> 443

<variant> 21

<question> Что делает http.ListenAndServe("localhost:8080", nil)?

<variant> Запускает веб-сервер

<variant> Сохраняет HTML-файл

<variant> Запускает браузер

<variant> Устанавливает сетевое соединение

<variant> Проверяет соединение

<question> Что произойдёт при вводе в браузере http://localhost:8080/hello, если сервер запущен?

<variant> Сервер отправит строку “Hello, web!”

<variant> Возникнет ошибка компиляции

<variant> Откроется HTML-файл

<variant> Запустится клиент

<variant> Выполнится goroutine

<question> Какой тип данных используется для передачи ответа браузеру?

<variant> []byte

<variant> int

<variant> string

<variant> interface{}

<variant> json

<question> Как преобразовать строку "Hello" в []byte?

<variant> []byte("Hello")

<variant> string([]byte("Hello"))

<variant> []byte{"Hello"}

<variant> convert("Hello")

<variant> string("Hello")

<question> Что произойдёт, если порт уже занят и вы запустите ListenAndServe с тем же портом?

<variant> Произойдёт ошибка

<variant> Программа зависнет

<variant> Произойдёт перезапуск

<variant> Программа переиспользует порт

<variant> Будет возвращён nil

<question> Как правильно передать функцию sayHi в другую функцию call(fn)?

<variant> call(sayHi)

<variant> call("sayHi")

<variant> call(sayHi())

<variant> call(func sayHi)

<variant> call(\*sayHi)

<question> Что произойдёт, если передать функцию с неподходящей сигнатурой в http.HandleFunc?

<variant> Возникнет ошибка компиляции

<variant> Появится warning

<variant> Всё сработает

<variant> Выполнится с предупреждением

<variant> Запустится тест

<question> Что делает http.HandleFunc("/hello", handler)?

<variant> Связывает путь /hello с функцией handler

<variant> Удаляет ресурс /hello

<variant> Обрабатывает HTML-шаблон

<variant> Запускает маршрутизатор

<variant> Генерирует HTML

<question> Что выведет этот код?

var f func()

f =sayHi

f()

<variant> Вызовется sayHi()

<variant> Ошибка компиляции

<variant> Ничего

<variant> func()

<variant> interface{}

<question> Какая сигнатура у функции, которую можно передать в doMath(func(int, int) float64)?

<variant> func(int, int) float64

<variant> func(a float64, b float64) int

<variant> func() float64

<variant> func(a int, b int)

<variant> func(int, int)

<question> Почему используется localhost в ListenAndServe("localhost:8080", nil)?

<variant> Для ограниченного локального доступа

<variant> Для подключения по HTTPS

<variant> Чтобы запретить подключение с браузера

<variant> Чтобы сервер слушал внешний трафик

<variant> Чтобы использовать JSON

<question> Что такое “функция первого класса” в контексте Go?

<variant> Функция, которая может быть передана или сохранена в переменной

<variant> Функция, вызываемая только из main

<variant> Функция с приоритетом

<variant> Экспортируемая функция

<variant> Только метод интерфейса