WB Tech: level # 1 (Golang)

# Как делать задания

В заданиях никаких устных решений — только код. Одно решение — один файл с хорошо откомментированным кодом. Каждое решение или невозможность решения надо объяснить.

Разрешается и приветствуется использование любых справочных ресурсов, привлечение сторонних экспертов и т.д. и т.п.

Основной критерий оценки — четкое понимание «как это работает». Некоторые задачи можно решить несколькими способами, в этом случае требуется привести максимально возможное количество вариантов.

Можно задавать вопросы, как по условию задач, так и об их решении. Идеальный вариант — продемонстрировать свои решения и получить максимальный фидбэк от опытных разработчиков Wildberries.

# Задания

1. Дана структура Human (с произвольным набором полей и методов). Реализовать встраивание методов в структуре Action от родительской структуры Human (аналог наследования).
2. Написать программу, которая конкурентно рассчитает значение квадратов чисел взятых из массива (2,4,6,8,10) и выведет их квадраты в stdout.
3. Дана последовательность чисел: 2,4,6,8,10. Найти сумму их квадратов(22+32+42….) с использованием конкурентных вычислений.
4. Реализовать постоянную запись данных в канал (главный поток). Реализовать набор из N воркеров, которые читают произвольные данные из канала и выводят в stdout. Необходима возможность выбора количества воркеров при старте.  
     
   Программа должна завершаться по нажатию Ctrl+C. Выбрать и обосновать способ завершения работы всех воркеров.
5. Разработать программу, которая будет последовательно отправлять значения в канал, а с другой стороны канала — читать. По истечению N секунд программа должна завершаться.
6. Реализовать все возможные способы остановки выполнения горутины.
7. Реализовать конкурентную запись данных в map.
8. Дана переменная int64. Разработать программу которая устанавливает i-й бит в 1 или 0.
9. Разработать конвейер чисел. Даны два канала: в первый пишутся числа (x) из массива, во второй — результат операции x\*2, после чего данные из второго канала должны выводиться в stdout.
10. Дана последовательность температурных колебаний: -25.4, -27.0 13.0, 19.0, 15.5, 24.5, -21.0, 32.5. Объединить данные значения в группы с шагом в 10 градусов. Последовательность в подмножноствах не важна.

Пример: -20:{-25.0, -27.0, -21.0}, 10:{13.0, 19.0, 15.5}, 20: {24.5}, etc.

1. Реализовать пересечение двух неупорядоченных множеств.
2. Имеется последовательность строк - (cat, cat, dog, cat, tree) создать для нее собственное множество.
3. Поменять местами два числа без создания временной переменной.
4. Разработать программу, которая в рантайме способна определить тип переменной: int, string, bool, channel из переменной типа interface{}.
5. К каким негативным последствиям может привести данный фрагмент кода, и как это исправить? Приведите корректный пример реализации.

**var** justString string

**func** someFunc() {

v := createHugeString(1 << 10)

justString = v[:100]

}

**func** main() {

someFunc()

}

1. Реализовать быструю сортировку массива (quicksort) встроенными методами языка.
2. Реализовать бинарный поиск встроенными методами языка.
3. Реализовать структуру-счетчик, которая будет инкрементироваться в конкурентной среде. По завершению программа должна выводить итоговое значение счетчика.
4. Разработать программу, которая переворачивает подаваемую на ход строку (например: «главрыба — абырвалг»). Символы могут быть unicode.
5. Разработать программу, которая переворачивает слова в строке.   
   Пример: «snow dog sun — sun dog snow».
6. Реализовать паттерн «адаптер» на любом примере.
7. Разработать программу, которая перемножает, делит, складывает, вычитает две числовых переменных a,b, значение которых > 2^20.
8. Удалить i-ый элемент из слайса.
9. Разработать программу нахождения расстояния между двумя точками, которые представлены в виде структуры Point с инкапсулированными параметрами x,y и конструктором.
10. Реализовать собственную функцию sleep.
11. Разработать программу, которая проверяет, что все символы в строке уникальные (true — если уникальные, false etc). Функция проверки должна быть регистронезависимой.

Например:   
abcd — true

abCdefAaf — false  
 aabcd — false

# Устные вопросы

1. Какой самый эффективный способ конкатенации строк?

strings.Builder:

func concat2builder(x, y string) string {

var builder strings.Builder

builder.Grow(len(x) + len(y)) // Только эта строка выделяет память

builder.WriteString(x)

builder.WriteString(y)

return builder.String()

}

1. Что такое интерфейсы, как они применяются в Go?  
   Интерфейс — это набор методов, представляющих стандартное поведение для различных типов данных, абстрактное описание методов.
2. Чем отличаются RWMutex от Mutex?

RWMutex в отличие от Mutex не блокирует параллельное чтение объекта

1. Чем отличаются буферизированные и не буферизированные каналы?  
   наличием или отсутствием буфера определенного размера))) Однако в закрытый не буферизованный канал нельзя писать и читать из него, а в закрытый буферизованный канал нельзя писать, но читать из него можно. Также в буферизованный не до конца заполненный канал можно писать, не дожидаясь чтения из него.
2. Какой размер у структуры struct{}{}?  
   размер struct{}{} не существует
3. Есть ли в Go перегрузка методов или операторов?
4. нет
5. В какой последовательности будут выведены элементы map[int]int?  
     
   *Пример:*

m[0]=1

m[1]=124

m[2]=281

В произвольном порядке

1. В чем разница make и new?  
   make создает и инициализирует объект только типов slice, map или chan. new возвращает указатель на тип, переданный в качестве аргумента
2. Сколько существует способов задать переменную типа slice или map?

var slice []int // создает переменную типа слайс, но не инициализирует ее

slice := []int{1,2,3} // создает и инициализирует переменную типа слайс с определенным значением элементов массива

slice := make([]int, l, c) // создает и инициализирует слайс с длиной l и ёмкостью c

var m map[int]int // создает переменную типа map, но не инициализирует ее (пустая map)

m := map[int]int{1:1, 2:2, 3:3} // создает и инициализирует переменную типа map с определенным значением ключей и значений

m := make(map[int]int, l) // создает пустую map длиной l

1. Что выведет данная программа и почему?

**func** update(p \*int) {

b := 2

p = &b

}

**func** main() {

**var** (

a = 1

p = &a

)

fmt.Println(\*p)

update(p)

fmt.Println(\*p)

}

1 1 потому что внутри update будет создана копия указателя p, о котором функции main ничего не будет известно, однако если мы хотим изменить значение переменной, на которую указывает p, то следует изменить код следующим образом:

func update(p \*int) {  
 b := 2  
 \*p = b  
}  
  
func main() {  
 var (  
 a = 1  
 p = &a  
 )  
 fmt.Println(\*p)  
 update(&a)  
 fmt.Println(\*p)  
}

результат – 1 2

1. Что выведет данная программа и почему?

**func** main() {

wg := sync.WaitGroup{}

**for** i := 0; i < 5; i++ {

wg.Add(1)

**go func**(wg sync.WaitGroup, i int) {

fmt.Println(i)

wg.Done()

}(wg, i)

}

wg.Wait()

fmt.Println(**"exit"**)

}

Приведет к дедлоку, потому что исполняемые горутины создадут свои копии wg и соответственно wg.Done будет применятся именно к копии wg. Правильная реализация:

func main() {  
 wg := sync.WaitGroup{}  
 for i := 0; i < 5; i++ {  
 wg.Add(1)  
 go func(wg \*sync.WaitGroup, i int) {  
 fmt.Println(i)  
 wg.Done()  
 }(&wg, i)  
 }  
 wg.Wait()  
 fmt.Println("exit")  
}

1. Что выведет данная программа и почему?

**func** main() {

n := 0

**if *true*** {

n := 1

n++

}

fmt.Println(n)

}

0, потому что внутри if будет создана своя переменная n

1. Что выведет данная программа и почему?

**func** someAction(v []int8, b int8) {

v[0] = 100

v = append(v, b)

}

**func** main() {

**var** a = []int8{1, 2, 3, 4, 5}

someAction(a, 6)

fmt.Println(a)

}

[100 2 3 4 5]. Функция someAction создаст свою копию слайса, т.е. копию указателя на массив. Поэтому сначала в функции someAction у нас будет доступ к элементу массива с индексом 0 и его можно изменить. Однако после вызова append вернется новый указатель на массив, так как capacity данного массива недостаточно для вставки нового элемента но так как в функции someAction у нас создана копия слайса, то функция main не будет знать, что указатель на массив (слайс) изменился. Для добавления элемента в слайс следует переписать данный код на следующий:

func someAction(v \*[]int8, b int8) {  
 (\*v)[0] = 100  
 (\*v) = append((\*v), b)  
 fmt.Println(\*v)  
}  
  
func main() {  
 var a = []int8{1, 2, 3, 4, 5}  
 someAction(&a, 6)  
 fmt.Println(a)  
}

1. Что выведет данная программа и почему?

**func** main() {

slice := []string{**"a"**, **"a"**}

**func**(slice []string) {

slice = append(slice, **"a"**)

slice[0] = **"b"**

slice[1] = **"b"**

fmt.Print(slice)

}(slice)

fmt.Print(slice)

}  
[b b a][a a] потому что анонимная функция создаст внутри себя копию слайса, которая не поменяет исходный слайс. Для того, чтобы его изменить следует переписать код следующим образом:

func main() {  
 slice := []string{"a", "a"}  
  
 func(slice \*[]string) {  
 \*slice = append(\*slice, "a")  
 (\*slice)[0] = "b"  
 (\*slice)[1] = "b"  
 fmt.Print(\*slice)  
 }(&slice)  
 fmt.Print(slice)  
}

однако если переписать код следующим образом:

func main() {  
 slice := []string{"a", "a"}  
 func(slice []string) {  
 slice[0] = "b"  
 slice[1] = "b"  
 a := &slice  
 slice = append(slice, "a")  
 a = &slice  
 fmt.Print(\*a)  
 }(slice)  
 fmt.Print(slice)  
}

то вывод будет [b b a][b b] по причине того, что внутри анонимной функции сначала у нас есть доступ к элементам слайса, однако после append создаётся новый указатель на массив.