1. Какой самый эффективный способ конкатенации строк?

strings.builder

1. Что такое интерфейсы, как они применяются в Go?

Тип интерфейса определяет множество методов, называющееся интерфейс. Переменная типа интерфейса может хранить значение любого типа с набором методов, который является любым надмножеством интерфейса. Говорят, что такой тип реализует интерфейс. Значение неинициализированной переменной типа интерфейса равно nil.

Интерфейс можно использовать как template. Функция может принимать структуры, реализующие методы, которые содержит интерфейс, являющийся параметром этой функции.

Также пустой интерфейс, как параметр функции, позволяет передать в неё разные типы данных, при корректной последующей их обработке. Интерфейсы всегда указатели и всегда указывают на конкретный объект.

1. Чем отличаются RWMutex от Mutex?

Mutex имеет методы Lock() и Unlock(), которые позволяют получать доступ к данным для записи или чтения только одной горутине одномоментно, все остальные будут ждать Unlock() для работы с данными.

RWMutex имеет отдельно блок чтения и записи. Rlock() обязывает ожидать своего Unlock() только другие Lock(). остальные Rlock() могут выполняться конкуренто, что позволяет реализовать параллельное чтение данных, но не запись.

1. Чем отличаются буферизированные и не буферизированные каналы?

Буферизированные каналы позволяют записывать в них данные размером буфера без блокировки до ожидания чтения из канала. Это позволяет реализовать очередь из данных.

1. Какой размер у структуры struct{}{}?

Не считая затрат на хранения информации о самом типе данных, пустая структура занимает в памяти 0 байт.

1. Есть ли в Go перегрузка методов или операторов?

Нет, перегрузка методов и операторов в Go отсутствуют.

1. В какой последовательности будут выведены элементы map[int]int?  
     
   *Пример:*

m[0]=1

m[1]=124

m[2]=281

Если печатать map передавая его в fmt.Print(), то элементы будут выведены в порядке возрастания ключей. Если проводить перебор элементов через for range и печатать значение каждого элемента, то стоит помнить – перебор map в for range происходит случайным образом.

1. В чем разница make и new?

Функции 'new' и 'make' различаются тем, что: 'new' возвращает указатель вместо значения возвращаемое функцией 'make'. 'make' имеет опциональный аргумент 'size'. 'make' выделяет память и инциализирует только объекты типов: slice, map, or chan.

1. Сколько существует способов задать переменную типа slice или map?

3 способа: 1) через var 2) через := 3) через make

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)

}

Выведет “1”, потому что происходит глубокое копирование указателя. Внутри функции update присваивается значения адреса переменной b локальному внутри функции указателю p. Для ожидаемого поведения программы, внутри функции надо принимать параметр p \*\*int.

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"**)

}

Выведет цифры от 0 до 5 в случайном порядке, а после “fatal error: all goroutines are asleep - deadlock!”, потом что происходит глубокое копирование объекта sync.WaitGroup и метод Done() применяется к его копии, а не экземплюяру объекта пространства имён функции main().

1. Что выведет данная программа и почему?

**func** main() {

  n := 0

**if *true*** {

     n := 1

     n++

  }

  fmt.Println(n)

}

Выведет “0”, потому что после проверки условия переменная n пересоздаётся в новом пространтсве имён блока if и внутри него происходит работа с дубликатом.

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)

}

Выведет ”100 2 3 4 5”, потому что слайс является структурой, которая содержит указатель на свое начало, длину и емкость. При передаче слайса в функцию не через указатель происходит глубокое копирование. Создается дубликат структуры с тем же указателем на начало, но уже своими длиной и емкостью. Поэтому изменить элемент массива удалось, а добавить элемент не получилось, потому что длина среза изменилась только у дубликата функции.

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)

}

Выведет “b b a” и “a a”, потому что не смотря на то, что в функции происходит копирование указателя на начало оригинального среза, после добавление “a” происходит увелечение capacity, что влечет за собой перевыделение памяти и смены значения указателя (перемещение) копии среза. Поэтому, так как у копии изменился указатель на начало, все последующие действия уже не будут зафиксированы в оригинальном срезе.