Робота з масивами, зрізами і мапами

Заняття 3

Golang для розробників

Лектор
Дмитро Сташкевич

r_d

План заняття

- Структури
- Масиви
- Зрізи
- Мапи
- Функції `make` та `new`

СТРУКТУРИ

```go
type Point struct {  5 usages
    X float64
    Y float64
}

func add(a, b Point) Point {  1 usage
    return Point{X: a.X + b.X, Y: a.Y + b.Y}
}

func main() {
    point1 := Point{X: 1, Y: 2}
    point2 := Point{X: 3, Y: 4}

    point3 := add(point1, point2)

    fmt.Printf( format: "point3: %+v\n", point3)
}
```

```go
type Config struct {  3 usages
    BaseURL    string
    APIKey     string
    Timeout    time.Duration
    RetryDelay time.Duration
    CacheSize  int
}

func loadConfig() *Config {  1 usage
    return &Config{
        BaseURL: "https://aoi.example.com",
        APIKey:  "123456",
    }
}

func doWithConfig(config *Config) {  1 usage
    fmt.Printf( format: "Config: %+v", config)
}

func main() {
    config := loadConfig()
    doWithConfig(config)
}
```

МАСИВИ

- Розмір масиву має бути відомим під час компіляції

```
type Point struct {  4 usages
    X float64
    Y float64
}

func main() {
    arr := [...]Point{
        {X: 1, Y: 2},
        {X: 3, Y: 4},
        {X: 5, Y: 6},
    }

    fmt.Println(arr)
}
```

```
type Point struct {  6 usages
    X float64
    Y float64
}

func add(points [2]Point) Point {  1 usage
    return Point{
        X: points[0].X + points[1].X,
        Y: points[0].Y + points[1].Y,
    }
}

func main() {
    point1 := Point{ X: 1,  Y: 2}
    point2 := Point{ X: 3,  Y: 4}

    fmt.Println(add([2]Point{point1, point2}))
}
```

```
func main() {
    var arr [2]string
    arr2 := [2]string{"a", "b"}

    fmt.Println(arr)
    fmt.Println(arr2)
}
```

ЗРІЗИ (SLICES)

- Slice — це поінтер на масив
- функція `append`

```go
func add(points []Point) Point {
    sum := Point{}
    for _, p := range points {
        sum.X += p.X
        sum.Y += p.Y
    }

    return sum
}

func main() {
    points := []Point{
        {X: 1, Y: 2},
        {X: 3, Y: 4},
    }

    fmt.Println(add(points))
}
```

```go
func add(points []Point) Point {  4 usages
    sum := Point{}
    for _, p := range points {
        sum.X += p.X
        sum.Y += p.Y
    }

    return sum
}

func main() {
    arr := [...]Point{{X: 1,  Y: 2}, {X: 3,  Y: 4}}
    arr[0].X = 5

    slice := arr[:]
    slice = append(slice, Point{X: 5,  Y: 6})

    fmt.Println(add(slice[1:]))
    fmt.Println(add(slice[:2]))
    fmt.Println(add(slice[:]))
    fmt.Println(add(slice[1:2]))
}
```

МАПИ (MAPS)

- key — value
- функція `delete`

```go
var cache = map[string]string{}  2 usag

func do(a string) string {  1 usage
    // some heavy calculations ....
    return a + a
}

func doWithCache(a string) string {
    if val, ok := cache[a]; ok {
        return val
    }

    result := do(a)
    cache[a] = result
    return result
}
```

```go
func main() {
    countries := map[string]string{
        "US": "United States",
        "JP": "Japan",
        "UK": "United Kingdom",
    }

    if country, ok := countries["US"]; ok {
        fmt.Println(country)
    }
}
```

```go
func main() {
    countries := map[string]string{
        "US": "United States",
        "JP": "Japan",
        "UK": "United Kingdom",
    }

    countries["FR"] = "France"
    delete(countries, key: "UK")

    fmt.Printf( format: "Countries: %+v\n", countries)
}
```

MAKE

- Алоціює та ініціалізує об’єкт типу:
  - slice
  - map
  - chan 


```go
func main() {
    slice1 := make([]int, 10)
    slice2 := make([]int, 0, 10)

    for _, v := range slice1 {
        fmt.Println( a...: "slice1", v)
    }

    for _, v := range slice2 {
        fmt.Println( a...: "slice2", v)
    }
}
```

```go
func main() {
    map1 := make(map[string]Point, 10)
    map2 := make(map[string]string)

    for key, value := range map1 {
        fmt.Println(key, value)
    }

    for key, value := range map2 {
        fmt.Println(key, value)
    }
}
```

NEW

- Алоціює об’єкт будь-якого типу
- Повертає поінтер

```go
func module(point *Point) float64 {  2 usages
    return point.X*point.X + point.Y*point.Y
}

func main() {
    point1 := new(Point)
    point2 := &Point{X: 1.0, Y: 2.0}

    fmt.Println(module(point1))
    fmt.Println(module(point2))
}
```

```go
func square(num *int) int {  3 usages
    if num == nil {
        return -1
    }
    return *num * *num
}

func main() {
    num1 := 10
    num2 := new(int)
    num3 := new(*int)

    fmt.Println(square(&num1))
    fmt.Println(square(num2))
    fmt.Println(square(*num3))
}
```

Q&A

r_d

ЗАВЖДИ Є КУДИ ЗРОСТАТИ