Раздел 9. Структуры

Структуры в Go - это тип данных, который позволяет объединять несколько значений в один объект. Структуры могут содержать поля различных типов данных и методы, которые могут быть вызваны для работы со значениями структуры. Для определения структуры применяются ключевые слова **type** и **struct.**

Структуры в Go - это способ объединения различных типов данных в одну единицу. Это как бы "коробочка", в которую мы можем положить разные вещи - числа, строки, списки и т.д.

Давай представим, что у нас есть игрушечный автомобиль. Этот автомобиль имеет много различных характеристик: цвет, размер, марка, возраст модели и т.д. Вместо того, чтобы каждую характеристику хранить отдельно, мы можем создать структуру под названием "Автомобиль", которая будет содержать все эти характеристики.

В Go это выглядит примерно так:

```
type Cat struct {
   Color string
   Size  int
   Brand string
   Age   int
}
```

Затем мы можем создать конкретный автомобиль, заполнив структуру данными:

```
myCar := Car{
   Color: "Красный",
   Size:  5,
   Brand: "BMW",
   Age:   3,
}
```

Итак, структуры в Go - это как коробочки, в которые мы можем положить много разных вещей, чтобы хранить их вместе и организованно.

Определение структур

Структуры в Go определяются при помощи ключевого слова `type` и ключевого слова `struct`:

```
type Person struct {
    Name string
    Age  int
}
```

В этом примере мы определяем структуру `Person`, которая содержит два поля: `Name` типа `string` и `Age` типа `int`. Поля могут иметь любой тип данных.

Создание переменной с типом структуры

Экземпляры структур можно создавать при помощи оператора new в Go:

```
p := new(Person)
p.Name = "Alice"
p.Age = 25
```

В этом примере создается экземпляр структуры Person при помощи оператора new, а затем устанавливаются значения его полей.

Также структуру можно создать без использования оператора new, задав значения полей прямо при создании:

```
p := Person{
    Name: "Alice",
    Age: 25,
}
```

В этом примере мы создаем экземпляр структуры Person и немедленно задаем значения его полей.

Есть еще третий вариант, когда мы создаем структуру и складываем значения без указания конкретного поля, в этом случае значения устанавливаются в те поля, которые идут по порядку полей указанных в структуре, пример:

```
p := Person{
    "Alice",
     25,
}
```

Доступ к полям структур

Для доступа к полям структур используется оператор `.`:

```
fmt.Println(p.Name)
fmt.Println(p.Age)
```

В этом примере мы выводим значения полей структуры `Person`, используя оператор `.` для доступа к полям.

Домашнее задание

1. **Создание и использование структур**: Создайте структуру, представляющую собой "Book" с полями "Title", "Author" и "Pages". Создайте экземпляр этой структуры, заполните поля и выведите их на экран.
2. **Создание структур с разными типами данных**: Создайте структуру "Person" с полями "Name" (string), "Age" (int) и "IsMarried" (bool). Создайте экземпляр этой структуры, заполните поля и выведите их на экран.
3. **Изменение значений полей структуры**: Создайте экземпляр структуры из предыдущего задания, измените значение одного из полей и выведите все поля на экран.
4. **Создание вложенных структур**: Создайте структуру "Employee" с полем "Name" и вложенной структурой "Job" с полями "Title" и "Salary".

Создайте экземпляр этой структуры, заполните все поля и выведите их на экран.

5. **Сравнение структур**: Создайте два экземпляра одной и той же структуры. Сравните их с помощью оператора == и выведите результат.
6. **Структуры и функции (⭐ сложное)**: Создайте функцию, которая принимает два аргумента типа "Book" и возвращает книгу с большим количеством страниц. Протестируйте эту функцию с несколькими экземплярами структуры "Book".