Раздел 10. Методы

Методы в Go - это специальные функции, которые можно привязать к определенному типу данных. В общем, методы похожи на функции, но они имеют специальную "приставку", которая указывает, к какому типу данных они относятся.

Представьте, что у вас есть объект, например, "Автомобиль". У этого объекта могут быть разные характеристики, такие как цвет, модель, год выпуска и т.д. Также у этого объекта могут быть разные действия, которые он может выполнять, например, ехать, останавливаться, поворачивать и т.д. В программировании эти действия представляют собой методы.

В Go методы объявляются с использованием ключевого слова "func" (как и обычные функции), но перед именем функции добавляется определение "приемника". Приемник - это специальная переменная, которая указывает, к какому типу данных привязан метод. Например, вот как может выглядеть метод для типа данных "Автомобиль":

```
type Car struct {
    color string
    model string
    year  int
}

func (c Car) drive() {
    fmt.Println("Driving...")
}
```

В этом примере "(c Car)" - это приемник. Он говорит нам, что метод "drive()" относится к типу данных Car. То есть, только объекты типа Car могут использовать этот метод. В самом методе мы можем использовать переменную "c" для доступа к свойствам объекта.

Таким образом, методы в Go - это способ организации кода и связывания функций с определенными типами данных. Они позволяют делать код более

структурированным и понятным.

Методы в Go - это функции, которые привязаны к определенному типу данных. Они могут использоваться для добавления поведения к типам данных и для упрощения работы с этими типами данных.

Определение методов

Методы в Go определяются как функции, которые привязаны к конкретному типу. Они также могут иметь дополнительные параметры, как и обычные функции.

Вот пример определения метода для структуры `Person`:

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

func (p Person) SayHello() {
    fmt.Printf("Hello, my name is %s and I am %d years old.",
p.Name, p.Age)
}
```

В этом примере мы определяем метод `SayHello` для структуры `Person`. Этот метод принимает параметр типа `Person` и использует его поля для вывода приветствия.

Использование методов

Методы могут быть вызваны для экземпляров типа данных, к которым они привязаны. Для вызова метода используется оператор `.`:

```
p := Person{Name: "Alice", Age: 25}
p.SayHello()
```

В этом примере мы создаем экземпляр структуры `Person` и вызываем метод `SayHello` для этого экземпляра.

Пример использования методов

Вот пример определения структуры `Rectangle` и методов для работы с этой структурой:

```
type Rectangle struct {
    Width  float64
    Height float64
}

func (r Rectangle) Area() float64 {
    return r.Width * r.Height
}

func (r *Rectangle) Scale(factor float64) {
    r.Width *= factor
    r.Height *= factor
}
```

В этом примере мы определяем структуру `Rectangle`, которая содержит поля `Width` и `Height`. Затем мы определяем два метода: `Area`, который возвращает площадь прямоугольника, и `Scale`, который масштабирует прямоугольник на заданный коэффициент.

Простой пример работы структур

```
package main

import "fmt"

type User struct {
    Name  string
    Email string
}

func (u User) Greet() {
    fmt.Printf("Hello, my name is %s and my email is %s.", u.
```

```
Name, u.Email)
}

func main() {
    u := User{Name: "Alice", Email: "alice@example.com"}
    u.Greet()
}
```

В этой программе мы определяем структуру `User`, которая содержит поля `Name` и `Email`. Затем мы определяем метод `Greet`, который принимает экземпляр структуры `User` и использует его поля для вывода приветствия. В `main` функции мы создаем экземпляр структуры `User` и вызываем метод `Greet` для этого экземпляра.

Пример вывода:

```
Hello, my name is Alice and my email is alice@example.com.
```

Домашнее задание

1. **Методы с разными возвращаемыми значениями**: Добавьте к структуре "Rectangle" метод "IsSquare", который возвращает bool, указывающий, является ли прямоугольник квадратом.
2. **Методы, изменяющие значения**: Добавьте к структуре "Rectangle" метод "DoubleSize", который удваивает размеры прямоугольника.
3. **Использование нескольких методов**: Используйте методы "Area" и "IsSquare" на нескольких экземплярах структуры "Rectangle".
4. **Создание структуры с вложенными структурами (⭐ сложное)**: Создайте структуру "Person" с полем "Name" и вложенной структурой "Address" с полями "Street", "City" и "Country". Добавьте метод "FullAddress", который возвращает полный адрес как строку.