Раздел 6. Функции

Функции - это основные строительные блоки программ в Go. Функция - это блок кода, который может быть вызван из другого места в программе. Функции могут принимать параметры и возвращать значения.

Функции похожи на рецепты для блюд. Когда хотим приготовить что-то вкусное, следуем рецепту с ингредиентами и шагами. Функции используют переменные как "ингредиенты" и команды как "шаги".

Мы уже активно использовали функции из других пакетов, например функцию Println из пакета fmt. Сейчас мы подробнее погрузимся в изучение функций, чтобы научиться не только эффективно использовать готовые, но и писать свои собственные функции.

Функция

Вот пример простой функции, которая не принимает параметров и не возвращает результатов:

```
func sayHello() {
    fmt.Println("Hello!")
}
```

В этом примере мы объявляем функцию `sayHello`, которая не принимает параметров и не возвращает результатов. Когда функция вызывается, она просто выводит строку "Hello!" на экран.

Параметры функций

Параметры функции - это переменные, которые передаются в функцию в момент ее вызова. Они представляют собой "входные данные" для функции, которые она использует для выполнения своих действий. Параметры функции указываются в скобках после имени функции и разделяются запятыми. Каждый параметр имеет имя и тип, который определяет, какие данные может принимать этот параметр.

Вот пример функции, которая принимает два параметра

```
func greet(name string, age int) {
    fmt.Printf("Hello, %s! You are %d years old.\\\\n", name,
age)
}
```

В этом примере мы объявляем функцию `greet`, которая принимает два параметра: `name` типа `string` и `age` типа `int`. Когда функция вызывается, она выводит приветствие, используя значения параметров.

Типы параметров и возможность использования переменного числа параметров

В Go, когда вы объявляете параметры функции, вы должны указать их типы. Это обязательно, так как Go - это статически типизированный язык, и компилятору нужно знать, какого типа значения ожидать. Например, в следующей функции `add` параметры `x` и `y` должны быть типа `int`:

```
func add(x int, y int) int {
    return x + y
}
```

Кроме того, функции в Go могут принимать переменное количество параметров. Это значит, что вы можете передать любое количество аргументов этого типа в функцию. Это достигается с помощью оператора `...`, который следует за типом параметра. Например, следующая функция `sum` может принимать любое количество аргументов типа `int` и возвращает их сумму:

```
func sum(nums ...int) {
    total := 0
    for _, num := range nums {
        total += num
    }
```

```
        fmt.Println(total)
}
```

Переменное число аргументов может быть полезно, когда вы не знаете заранее, сколько значений будет передано в функцию. Это позволяет функции быть более гибкой и повторно использоваться в различных ситуациях. Например, функция для суммирования чисел могла бы принимать любое количество чисел, а функция для форматирования строки могла бы принимать любое количество значений для замены.

Результаты функций

Результаты функций - это значения, которые функция возвращает после своего выполнения. Эти значения могут затем быть использованы в других частях программы. Возвращаемые значения объявляются после списка параметров и перед телом функции.

Пример функции, которая возвращает результат:

```
func add(x int, y int) int {
    return x + y
}
```

В этом примере функция `add` принимает два параметра `x` и `y`, и возвращает их сумму.

Функции в Go могут возвращать несколько результатов. Это удобно, например, когда функция выполняет несколько задач и может вернуть результат каждой из них.

Пример функции, которая возвращает два результата:

```
func divmod(a int, b int) (int, int) {
    return a / b, a % b
}
```

В этом примере функция `divmod` принимает два параметра `a` и `b`, и возвращает частное и остаток от деления `a` на `b`.

Когда функция возвращает несколько результатов, их можно присвоить нескольким переменным:

```
q, r := divmod(7, 3)
fmt.Printf("quotient: %d, remainder: %d\n", q, r)
```

В этом примере мы вызываем функцию `divmod` с аргументами `7` и `3`, и присваиваем ее результаты переменным `q` (quotient - частное) и `r` (remainder - остаток). Затем мы выводим эти значения.

В Go общепринятой практикой является возврат ошибки функцией как последнего значения. Если функция может потенциально вызвать ошибку в процессе своего выполнения, она обычно возвращает ошибку в качестве одного из своих результатов. Таким образом, вызывающий код может проверить, произошла ли ошибка, и соответствующим образом обработать ее. Однако, подробнее о работе с ошибками в Go мы обсудим позже.

Шпаргалка

1. Функции в Go - это основные строительные блоки программы, используются для разбиения программы на более мелкие модули. Зачастую функции переиспользуют, это позволяет избежать лишней дупликации кода.
2. Функции могут принимать параметры, которые представляют собой входные данные для функции. Эти параметры указываются в скобках после имени функции и разделяются запятыми.
3. В Go, когда вы объявляете параметры функции, вы должны указать их типы. Также функции в Go могут принимать переменное количество параметров.
4. Результаты функций - это значения, которые функция возвращает после своего выполнения. Эти значения могут затем быть использованы в других частях программы. Возвращаемые значения объявляются после списка параметров и перед телом функции.
5. Функции в Go могут возвращать несколько результатов. Когда функция возвращает несколько результатов, их **нужно** присвоить нескольким переменным.

6. В Go общепринятой практикой является возврат ошибки функцией как последнего значения.

Домашнее задание

1. **Создание и вызов функции**: Создайте функцию, которая выводит приветствие на экран, и вызовите ее в своей программе.
2. **Функция с параметрами**: Создайте функцию, которая принимает два числа в качестве аргументов и выводит их сумму.
3. **Функция, возвращающая значение**: Создайте функцию, которая принимает два числа и возвращает их произведение.
4. **Использование нескольких значений return**: Создайте функцию, которая принимает два числа и возвращает их сумму и разность как два отдельных значения.
5. **Функция с переменным числом аргументов**: Создайте функцию, которая принимает переменное количество аргументов и выводит их на экран.
6. **Вызов функции из другой функции**: Создайте две функции и вызовите одну из них из другой.
7. **Функция, возвращающая сумму и количество аргументов:** Создайте функцию, которая получает переменное количество аргументов типа int, складывает их, и возвращает их сумму и количество полученных ею аргументов.
8. **Возвращение большего значения из функции:** Создайте функцию, которая принимает 3 переменных типа int, значение передаваемое в функцию должно быть 0 или 1, и возвращает значение, которое повторяется большее количество раз, среди полученных переменных.
9. **Значения по умолчанию для параметров функции (⭐ сложное)**: Исследуйте, можем ли мы в Go установить значения по умолчанию для параметров функции. Если нет, то объясните, как можно достичь похожего результата.
10. **Функция как аргумент другой функции (⭐ сложное)**: Исследуйте, можем ли мы в Go передать функцию как аргумент другой функции. Если нет, то объясните, как можно достичь похожего результата.