Раздел 11. Условные конструкции

Условные конструкции в Go - это инструменты, которые позволяют программистам создавать логику, которая выполняется только при определенных условиях. В Go, условные конструкции выполняются при помощи ключевых слов `if`, `else if` и `else`.

Ключевое слово if

Ключевое слово `if` используется для проверки условия и выполнения блока кода, только если условие истинно. Вот пример использования ключевого слова `if`:

```
graph LR
A[Условие] -->|true| B[Действие 1]
A -->|false| C[Действие 2]
```

```
if x > 0 {
    fmt.Println("x is positive")
}
```

В этом примере мы проверяем, больше ли значение переменной `x` нуля. Если это так, мы выводим сообщение "x is positive". Если это не так, мы ничего не делаем.

Ключевое слово else if

Ключевое слово `else if` используется для проверки дополнительных условий, если первое условие ложно. Вот пример использования ключевого слова `else if`:

```
if x > 0 {
    fmt.Println("x is positive")
} else if x < 0 {
    fmt.Println("x is negative")
}
```

В этом примере мы проверяем, больше ли значение переменной `x` нуля. Если это так, мы выводим сообщение "x is positive". Если это не так, мы проверяем, меньше ли значение `x` нуля. Если это так, мы выводим сообщение "x is negative". Если ни одно из этих условий не истинно, мы ничего не делаем.

Ключевое слово else

Ключевое слово `else` используется для выполнения блока кода, если все предыдущие условия были ложными. Вот пример использования ключевого слова `else`:

```
if x > 0 {
    fmt.Println("x is positive")
} else if x < 0 {
    fmt.Println("x is negative")
} else {
    fmt.Println("x is zero")
}
```

В этом примере мы проверяем, больше ли значение переменной `x` нуля. Если это так, мы выводим сообщение "x is positive". Если это не так, мы проверяем, меньше ли значение `x` нуля. Если это так, мы выводим сообщение "x is negative". Если ни одно из этих условий не истинно, мы выводим сообщение "x is zero".

```
graph LR
A[IF Условие] -->|true| B[Действие 1]
```

```
A -->|false| D[else if Условие]
D -->|true| E[Действие 2]
D -->|false| F[else]
F -->G[Действие 3]
```

Пример использования условных конструкций

Вот пример использования условных конструкций для определения, является ли число четным или нечетным:

```
func main() {
    x := 4
    if x%2 == 0 {
        fmt.Println("x is even")
    } else {
        fmt.Println("x is odd")
    }
}
```

В этом примере мы определяем переменную `x` со значением 4 и проверяем, делится ли значение `x` на 2 без остатка. Если это так, мы выводим сообщение "x is even". Если это не так, мы выводим сообщение "x is odd".

```
graph LR
A[IF Условие] -->|true| B[Действие 1]
A -->|false| D[else]
D -->E[Действие 2]
```

Условные конструкции в Go - это инструменты, которые позволяют программистам создавать логику, которая выполняется только при определенных условиях. Они позволяют проверять значения переменных и выполнять определенные действия в зависимости от результата проверки.

Ключевые слова `if`, `else if` и `else` используются для создания условий и выполнения кода при их выполнении.

Ключевое слово switch

Ключевое слово `switch` используется для выполнения блока кода, если значение переменной соответствует одному из нескольких допустимых значений. Вот пример использования ключевого слова `switch`:

```
switch day {
case 1:
    fmt.Println("Monday")
case 2:
    fmt.Println("Tuesday")
case 3:
    fmt.Println("Wednesday")
case 4:
    fmt.Println("Thursday")
case 5:
    fmt.Println("Friday")
case 6:
    fmt.Println("Saturday")
case 7:
    fmt.Println("Sunday")
default:
    fmt.Println("Invalid day")
}
```

В этом примере мы проверяем, равно ли значение переменной `day` одному из семи допустимых значений. Если значение `day` равно 1, мы выводим сообщение "Monday", если равно 2 - "Tuesday", и так далее. Если значение `day` не соответствует ни одному из этих значений, мы выводим сообщение "Invalid day".

```
graph LR
A[switch] -->|case 1| B[Действие 1]
A[switch] -->|case 2| C[Действие 2]
A[switch] -->|case 3| D[Действие 3]
A[switch] -->|case 4| E[Действие 4]
```

Использование switch

Ключевое слово `switch` используется для проверки значения переменной на соответствие одному из нескольких допустимых значений. Каждый возможный вариант значения переменной должен быть определен с помощью ключевого слова `case`. Если значение переменной не соответствует ни одному из значений `case`, выполняется блок кода, определенный ключевым словом `default`. Вот пример использования ключевого слова `switch`:

```
func main() {
    day := 3
    switch day {
    case 1:
        fmt.Println("Monday")
    case 2:
        fmt.Println("Tuesday")
    case 3:
        fmt.Println("Wednesday")
    case 4:
        fmt.Println("Thursday")
    case 5:
        fmt.Println("Friday")
    case 6:
        fmt.Println("Saturday")
    case 7:
        fmt.Println("Sunday")
    default:
        fmt.Println("Invalid day")
```

```
    }
}
```

В этом примере мы определяем переменную `day` со значением 3 и проверяем, равно ли значение `day` одному из семи допустимых значений. Если значение `day` равно 3, мы выводим сообщение "Wednesday". Если значение `day` не соответствует ни одному из этих значений, мы выводим сообщение "Invalid day".

Ключевое слово fallthrough

Ключевое слово `fallthrough` используется для выполнения блока кода следующего `case` в списке, даже если значение переменной не соответствует текущему `case`. Вот пример использования ключевого слова `fallthrough`:

```
switch day {
case 1:
    fmt.Println("Monday")
case 2:
    fmt.Println("Tuesday")
case 3:
    fmt.Println("Wednesday")
    fallthrough
case 4:
    fmt.Println("Thursday")
case 5:
    fmt.Println("Friday")
case 6:
    fmt.Println("Saturday")
case 7:
    fmt.Println("Sunday")
default:
    fmt.Println("Invalid day")
}
```

В этом примере мы проверяем, равно ли значение переменной `day` одному из семи допустимых значений. Если значение `day` равно 1, мы выводим сообщение "Monday". Если же не равно ни одному подходящему значению, мы впадем в `default` кейс.

Домашнее задание

1. **Условный оператор if**: Создайте слайс из чисел. Напишите цикл for-range, который проверяет, является ли каждое число положительным, и выводит соответствующее сообщение.
2. **Условный оператор if-else**: Создайте слайс из чисел. Напишите цикл, который проверяет, больше ли каждое число 10, и выводит разные сообщения для случаев, когда это верно и когда не верно.
3. **Условный оператор if-else**: Создайте слайс из чисел. Напишите цикл, который сравнивает каждое число со значениями 10, 20 и 30, и выводит разные сообщения в зависимости от результата сравнения.
4. **Оператор выбора switch в цикле**: Создайте слайс из чисел от 1 до 7. Напишите цикл, который использует оператор switch для проверки каждого числа как дня недели и выводит сообщение в зависимости от дня.
5. **Оператор выбора switch с несколькими значениями для case**: Создайте слайс из чисел. Напишите цикл, который использует оператор switch для проверки каждого числа и выводит сообщение для четных и нечетных чисел.
6. **Оператор выбора switch без выражения**: Создайте слайс из булевых значений. Напишите цикл, который использует оператор switch без выражения для вывода разных сообщений в зависимости от значения булевой переменной.
7. **Оператор выбора switch с fallthrough (⭐ сложное)**: Создайте слайс из чисел от 1 до 10. Напишите цикл, который использует оператор switch с fallthrough для проверки каждого числа и выводит сообщение для каждого случая.