Функции шпаргалка

Функция - это блок кода, который выполняет какую-то задачу. Функцию можно представить как черный ящик с входом и выходом. Мы передаем туда какие-то данные, но не знаем, что происходит внутри, мы можем только взять обработанные данные из ящика.

То что мы передаем называется - **аргумент**, то что функция нам отдает - **возвращаемое значение.**

Каждый аргумент имеет свой тип данных, который определяется после его имени, разделенного пробелом от типа данных.

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

Аргументы одного типа следующие друг за другом не требуют указания типа для каждого:

```
func add(x, y int) int {
  return x + y // можем возвращать сразу выражение
}
```

Функция может возвращать обработанные данные с помощью ключевого слова `return`. Данные будут использоваться в месте вызова

```
func main() {
  sum := add(42, 13)
}
```

Из одной функции мы можем вернуть несколько значений, нужно указать типы всех возвращаемых значений

```
func divide(a, b int) (int, int) {
    integer := a / b
    remainder := a % b
    return integer, remainder
}
```

Рекурсия

Рекурсия - это когда функция вызывает саму себя.

Пример рекурсии

```
func factorial(n int) int {
    if n <= 1 {
        return 1
    }
    return n * factorial(n-1)
}
```

Эта функция вычисляет факториал числа `n` (т.е. произведение всех чисел от `n` до `1`). Обратите внимание, что функция вызывает саму себя с аргументом `n-1`.

Выход из рекурсии

Каждый рекурсивный вызов должен привести к условию выхода из рекурсии. В примере выше это условие - когда `n <= 1`. Без него функция будет бесконечно вызывать саму себя, что приведет к переполнению стека.

Глубина рекурсии

Глубина рекурсии - это количество рекурсивных вызовов, которые функция может сделать перед тем, как достигнуть условия выхода из рекурсии.

Вызов функции	Глубина	Возврат
`factorial(3)`	1	`3 * factorial(2)`
`factorial(2)`	2	`2 * factorial(1)`
`factorial(1)`	3	`1`

Конечный возврат значения происходит, когда достигается условие выхода из рекурсии. В данном случае условие выхода - это когда `n <= 1`.

Итак, начинаем с `factorial(3)`:

1. `factorial(3)` вызывает `factorial(2)`.
2. `factorial(2)` вызывает `factorial(1)`.
3. `factorial(1)` достигает условия выхода из рекурсии и возвращает `1` в `factorial(2)`.

4. `factorial(2)` умножает `2` на `1` (то что вернула `factorial(1)`) и возвращает результат `2` в `factorial(3)`.
5. `factorial(3)` умножает `3` на `2` (то что вернула `factorial(2)`) и возвращает результат `6`.

Таким образом, итоговый результат для `factorial(3)` - это `6`.