GO-02 02: Основы языка. Указатели в Go

Описание:

Go изначально проектировался как простой и очень быстрый язык программирования, и авторы старались взять самое лучшее от его предшественников. Поэтому в нем реализованы указатели, как и в языке C. Но в отличие от C, здесь нет адресной арифметики, т.е. нельзя выполнять арифметические операции над указателями для получения других однотипных значений.
Все переменные представляют собой именованную область памяти, зарезервированную для хранения значения указанного типа. То есть при работе с переменными мы ссылаемся на значения посредством имени переменной. Но блок памяти, в котором хранится значение, имеет адрес и к нему можно обращаться по этому адресу.
В Go, как и в других языках программирования, переменные передаются по значению. Передача по значению фактически означает копирование переменной, то есть выполняются дополнительные операции по выделению памяти и копированию значений. Как вы поняли, копирование — это довольно дорогая операция, особенно если речь идет о больших объемах данных. Реализация возможности передавать по ссылке позволяет избежать копирования данных, так как в этом случае передается не значение, а ссылка, т.е. указатель на область памяти, где находится значение.
Таким образом, в Go выделяется еще один тип данных — указатели. Указатель — это значение адреса блока памяти, в котором хранится значение переменной. Значением по умолчанию для указателей является nil.
Получить адрес переменной, то есть указатель, можно при помощи оператора & (амперсанд).
var x int

```
pntr := &x
fmt.Println(pntr)     // 0xc00001c080
```

Для того чтобы получить значение по известному адресу, нужно выполнить обратную операцию — разыменовать указатель. Для этого используется оператор *.
var x int

```
pntr := &x
x = 64
fmt.Println(*pntr)    // 64
```

В примере выше запись *pntr возвращает значение переменной x, указатель на которую хранится в переменной pntr.

Можно присвоить новое значение переменной, используя указатель.

```
*pntr = 31
fmt.Println(*pntr)    // 31
fmt.Println(x)        // 31
```

В Go также выделяется отдельный тип данных uintptr, который представляет собой целочисленный тип данных. Его размер достаточен для хранения значения указателя.

```
var pntr1 *int
var pntr2 uintptr

fmt.Println(pntr1)    // <nil>
fmt.Println(pntr2)    // 0
```

Как было сказано выше, значением по умолчанию для указателя является nil. Это видно на примере, когда мы объявили переменную как var pntr1 *int. Мы явно описали переменную как указатель на int.
При использовании типа данных uintptr значением по умолчанию является 0, так как это целочисленный тип данных и используется для хранения указателя. Этот тип данных используется не так часто, и в основном сферой его применения являются какие-то низкоуровневые алгоритмы. В своей практике вы намного чаще будете использовать первый вариант работы с указателями.
Функция new()
Для удобства в Go реализована функция для создания переменных new. Эта функция на вход принимает тип данных, не значение. Далее она выделяет память, которой будет достаточно для хранения переданного типа, устанавливает значение по умолчанию и возвращает указатель на созданную переменную.
Функция new является не какой-либо фундаментальной возможностью языка, а всего лишь синтаксическим сахаром для объявления переменных. Таким образом явное объявление переменной с последующим получением указателя и применение функции new полностью схожи в своем поведении. Функцию new удобно применять при создании локальных переменных.

```
var intValue int
pntr := &intValue
fmt.Println(pntr)     //0xc00001c080

pntr2 := new(int)
fmt.Println(pntr2)    //0xc00001c088
```

Полезные ссылки:

- Зачем в Go амперсанд и звездочка
- Pointers in Go

- Pointers in Go

Задание:

1. Создайте в проекте module02 новую ветку 02_task.
2. Создайте новую директорию и в ней файл main.go. Напишите код, в котором:

- объявите переменную A как указатель на int;
- переменную B — целочисленную с произвольным значением;
- присвойте в переменную A указатель на целочисленную переменную B и выведите на экран значение путем разыменовывания указателя A;
- присвойте целочисленной переменной B новое произвольное значение через указатель A и выведите на экран новое значение переменной B.

3. Создайте новую директорию и файл main.go. Напишите код, в котором:

- найдите радиус окружности, если её длина равна 35.
- радиус окружности R объявите как указатель на float64.
- вычислите площадь круга, используя при расчёте разыменовывание указателя R.
- При необходимости дробные значения округлите до двух знаков после запятой.

4. Создайте новый коммит с вашими решениями и отправьте в удаленный репозиторий вашего проекта.
5. В ответе пришлите ссылку на merge request в ветку master своего проекта ветки 02_task.