Go

Начальный курс по Go

ЗНАКОМСТВО

Организационные моменты

Занятия по:

- Пт, Сб, Вс с 09:00 до 13:00

Завтра и послезавтра + 30 минут

Лекции по 55 минут + 5 минут перерыв

Организационные моменты

Вся информация будет на странице по адресу:

Disclaimer

Всё, о чём мы будем говорить:

1. Применимо к разработке бизнес-приложений
2. Может содержать ошибки и неточности
3. По большей степени является мнением автора

Сокращения и обозначения

1. ДЗ – домашнее задание
2. ПЗ – практическое задание (делается в классе)
3. СИ – самостоятельное изучение
4. Demo – демонстрационный пример

BACKGROUND

Небольшой опрос

1. Что-то программировал
2. Видел в глаза Go
3. Что-то делал на Go
4. Программировал на чём-то другом

Что такое программирование и т.д.

ВВЕДЕНИЕ В ПРЕДМЕТ

Программирование

Эффективное решение задачи*:

1. Поиска информации
2. Каталогизации
3. Доставки контента
4. И т.д.

Результат программирования – программа, решающая поставленную задачу

Языки программирования

Языки общения между программистом и компьютером (машиной)

На языках программирования пишутся программы, которые затем исполняются компьютером

Машинный код

Команды в двоичном формате (0 и 1), которые понимает и умеет исполнять машина (их ещё называют инструкциями)

Двоичный формат также называют бинарным, а код – бинарным кодом

История

GO

Go

- Порог входа: высокий (ниже, чем в C++)
- Основные области применения:
  - веб-сервисы
  - сетевые сервисы
  - системные приложения
  - вспомогательные инструменты

Зачем был разработан

- сложный софт и высокие требования к уровню программистов
- медленный процесс внедрения новых функций
- мультиязычность (разные разработчики пишут на разных языках)
- разный уровень разработчиков

История версий

Initial Version - 2007

Version 1 - 2012

Version 1.1 - 2013

Version 1.2 - 2013

…

Version 1.12 - 2019

Version 1.13 - 2019

Что необходимо развивать прямо сейчас

КЛЮЧЕВЫЕ НАВЫКИ

Ключевые навыки

1. Быстрая печать
2. Английский
3. Навык набора кода и использования GoLand

Наконец-то дошли до программирования ☺

GO INTRO

Video

Обязательно смотрите видео в материалах

(иначе ничего не будете успевать)

Hello World

файл main.go:

```
package main

import "fmt"

func main() {
  fmt.Println("hello world")
}
```

Package

package - пакет в системе Go

import - подключение другого пакета

Tools (инструменты)

go – инструмент для управления кодом на Go

Tools

GoLand (IDE) – специальная среда, которая упрощает программирование на Go

Hello World cracking

Посмотреть, что произойдёт, если отойти от шаблона

«Hello world»

А именно:

1. Изменить регистр: fmt на Fmt
2. Убрать фигурную скобку на первой строке

Hello World cracking

Выводы:

1. Регистрозависимость
2. Строгие требования к синтаксису

КОММЕНТАРИИ

Комментарии

Пояснения к тексту кода, не влияющие на итоговый результат

Типы:

1. Строчные
2. Блочные

Комментарии

Строчные:

// всё, что после этих символов (до конца строки) – игнорируется

Комментарии

Блочный:

```
/*

всё, что между этими символами –

игнорируется

*/
```

Комментарии

```
package main
```

GoLand автоматически уберёт import "fmt"

```
func main() {
  // fmt.Println("hello world")
}
```

ПЕРЕМЕННЫЕ

Переменные

Ключевые понятия:

1. Имя
2. Значение
3. Тип

Пример с файлами

Переменные

Синтаксис:

Переменные

Синтаксис (объявление + присваивание):

```
var count int = 10

var count = 10

count := 10
```

старатьcя использовать этот вариант

Переменные

Важно: неинициализированные переменные всегда инициализированы в нулевые значения (например, для целых чисел – 0)

Переменные

```
func main() {
  count := 10
  fmt.Println(count)
}
```

Переменные

```
func main() {
  fmt.Println(count)
  count := 10
  // ниже можно использовать имя count, выше - нет
  fmt.Println(count)
}
```

Имя

использовать

1. Должно быть осмысленным
2. Начинается с буквы (_, $ - не рекомендуется)
3. Содержит буквы (цифры, _, $ - не рекомендуется)
4. Если состоит из двух и более слов пишется в нотации camelCase*
5. Имя на английском языке (без транслита – никаких summaOperacii и подобных!)

использовать

_

_ - don't care (специальное имя, используемое тогда, когда нужно что-то написать, но значение нас не интересует)

Quality Gate

использовать

В большинстве требований приняты стандарты по оформлению кода.

Ваш код не принимается, если вы не соблюдаете стандарты.

Поэтому, домашние работы будут отправляться на доработку, если:

1. Не выполняются правила именования
2. Код не хранится в системе контроля версий (позже)
3. Не выполняются статические проверки (позже)
4. Нет авто-тестов, либо недостаточное покрытие (позже)
5. Не выполнены требования (подробнее см. в требованиях к задаче)

Имя

```
func main() {
  count := 10
  price := 200
  sum := count * price
  discount := 30
  sum := sum * (100 – discount) / 100
  fmt.Println(sum)
}
```

Говорящие имена нам помогают понимать суть программы

ЦЕЛОЧИСЛЕННЫЕ ТИПЫ

запомнить

Целочисленные типы

1. int8 – 1 байт (-128 до 127)
2. int16 – 2 байта (-32768 до 32767)
3. int32 (rune) – 4 байта (-2147483648 до 2147483647)
4. int64 – 8 байт (-9223372036854775808 до 9223372036854775807)

int - по умолчанию для целых чисел будет int32 для 32 битных систем и int64 для 64 битных систем

Целочисленные типы

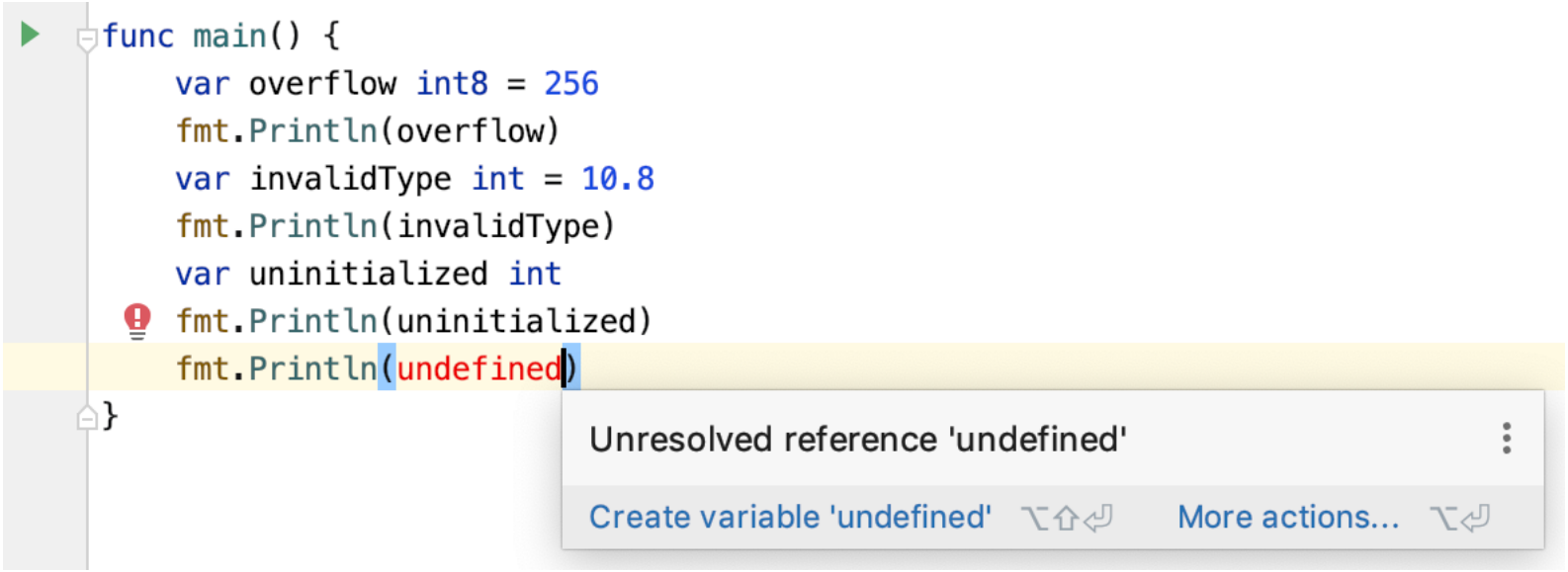
запомнить

1. uint8 (byte) – 1 байт
2. uint16 – 2 байта
3. uint32 – 4 байта
4. uint64 – 8 байт

Те же типы, только без знака (т.е. отрицательные числа там не хранятся)

```
func main() {
        var overflow int8 = 256
        fmt.Println(overflow)
        var invalidType int = 10.8
        fmt.Println(invalidType)
        var uninitialized int
        fmt.Println(uninitialized)
        fmt.Println(undefined)
}
```

Обязательно смотрите на ошибки!



Самостоятельное изучение

ОТРИЦАТЕЛЬНЫЕ ЧИСЛА

Важно

Данный раздел предназначен только для ознакомления, детально разбираться в нём не нужно

Числа со знаком

Старший бит – для хранения знака

$1_{10} = 0000\ 0001_2$

А как же хранить отрицательные?

$-1_{10} = 1000\ 0001_2$

Проблема:

$0000\ 0001_2$

\+

$1000\ 0001_2$

=

$1000\ 0010_2$

Альтернатива

Старший бит – для хранения знака

$1_{10} = 0000\ 0001_2$

А как же хранить отрицательные?

$-1_{10} = 1111\ 1111_2$

Проблемы нет:

$$
\begin{array}{l}
0000\ 0001_2 \\
+ \\
1111\ 1111_2 \\
= \\
0000\ 0000_2
\end{array}
$$

Дополнительный код (-1)

Алгоритм:

1. Модуль числа -1_{10} -> $0000\ 0001_2$
2. Инвертируем все биты -> $1111\ 1110_2$
3. Прибавляем 1 –> $1111\ 1111_2$

Дополнительный код (-128)

Алгоритм:

1. Модуль числа 128_{10} -> $1000\ 0000_2$
2. Инвертируем все биты -> $0111\ 1111_2$
3. Прибавляем 1 –> $1000\ 0000_2$

Обратное преобразование

Алгоритм:

1. Запись в доп.коде -127_{10} -> $1000\ 0001_2$
2. Инвертируем все биты -> $0111\ 1110_2$
3. Прибавляем 1 –> $0111\ 1111_2$ (модуль)

ДЗ

ДЗ: Установка

Посмотреть видео по установке и настройке

GoLand (особенно про Debug – отладку)

ДЗ: GitHub

Нужно зарегистрироваться на

https://github.com и поставить Git (см. видео)

Спасибо за внимание

Ильназ Гильязов

2020г.