

**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ**

**Федеральное государственное автономное
образовательное учреждение высшего образования
«Северо-Кавказский федеральный университет»**

Кафедра инфокоммуникаций

**Отчет по лабораторной работе №1
«Основы языка программирования Go»**

по дисциплине « Программная инженерия»

Выполнил студент группы ПИЖ-б-о-21-1
Ботвинкин Н.С. « » _____ 20__ г.
Подпись студента _____
Работа защищена « » _____ 20__ г.
Проверил Воронкин Р.А. _____
(подпись)

Ставрополь 2024

Постановка задачи

Исследовать назначение и способы установки Go, исследовать типы данных, константы и арифметические операции языка программирования Go

Выполнение

Часть 1. Примеры теоретической части

```
1 package main
2
3 import "fmt"
4
5 func main() {
6 |   fmt.Println("Hello, Go!")
7 }
```

Рисунок 1.1 – Первая программа на Go

```
package main

import "fmt"

func main() {
|   fmt.Println(string("Hello, Go!"[0]))
}
```

Рисунок 1.2 – Пример вывода символа строки через представление в виде строки

```
package main

import "fmt"

func main() {
|   var hello string
|   hello = "Hello Go!"
|   var a int = 2019
|   fmt.Println(hello)
|   fmt.Println(a)
}
```

Рисунок 1.3 – Пример работы с переменными в языке Go

```

package main

import "fmt"

func main() {
    var (
        name string = "Dima"
        age int = 23
    )

    fmt.Println(name)
    fmt.Println(age)
}

```

Рисунок 1.4 – Пример объявления нескольких переменных в блоке var

```

package main

import "fmt"

func main() {
    var name string
    var age int
    fmt.Print("Введите имя: ")
    fmt.Scan(&name)
    fmt.Print("Введите возраст: ")
    fmt.Scan(&age)

    fmt.Println(name, age)
}

```

Рисунок 1.5 – Пример чтения данных с консоли

```

package main

import "fmt"

func main() {
    name := "Ivan"
    age := 27
    fmt.Println("My name is", name, "and I am", age, "years old.")
}

// вывод:
// My name is Ivan and I am 27 years old.

```

Рисунок 1.6 – Пример вывода строк с переменными

```

/*
|   Первая программа
|   на языке Go
*/

package main // определение пакета для текущего файла

import "fmt" // подключение пакета fmt

// определение функции main

func main() {
|   fmt.Println("Hello Go!") // вывод строки на консоль
}

```

Рисунок 1.7 – Пример использования комментариев

```

package main

import (
|   "fmt"
)

const(
|   A int = 45
|   B
|   C float32 = 3.3
|   D
)

func main() {
|   fmt.Println(A, B, C, D) // Вывод: 45 45 3.3 3.3
}

```

Рисунок 1.8 – Пример объявления констант

```

package main

import (
|   "fmt"
)

const (
|   Sunday = iota
|   Monday
|   Tuesday
|   Wednesday
|   Thursday
|   Friday
|   Saturday
)

func main() {
|   fmt.Println(Sunday) // вывод 0
|   fmt.Println(Saturday) // вывод 6
}

```

Рисунок 1.9 – Пример использования идентификатора iota

```
const (  
  c0 = iota // c0 == 0  
  c1 = iota // c1 == 1  
  c2 = iota // c2 == 2  
)
```

Рисунок 1.10 – Представление о том, как работает `iota`

Часть 2. Наиболее часто используемые математические функции пакета `math`

```
result := math.Abs(-5.67)  
// result равен 5.67
```

Рисунок 1.11 – Абсолютное значение числа (`Abs`)

```
result := math.Ceil(5.67)  
// result равен 6
```

Рисунок 1.12 – Округление вверх до ближайшего целого (`Ceil`)

```
result := math.Floor(5.67)  
// result равен 5
```

Рисунок 1.13 – Округление до вниз до ближайшего целого (`Floor`)

```
result := math.Sqrt(16)  
// result равен 4
```

Рисунок 1.14 – Квадратный корень числа (`Sqrt`)

```
result := math.Pow(2, 3)  
// result равен 8
```

Рисунок 1.15 – Степень y числа x (`Pow`)

```
sinValue := math.Sin(math.Pi / 2)  
// sinValue равен 1 (синус 90 градусов)
```

Рисунок 1.16 – Синус угла (`Sin`)

```
result := math.Log(10)
// result равен примерно 2.302585...
```

Рисунок 1.17 – Натуральный логарифм числа (Log)

```
maxVal := math.Max(3, 7)
// maxVal равен 7
```

Рисунок 1.18 – Максимальное значение числа (Max)

```
minVal := math.Min(3, 7)
// minVal равен 3
```

Рисунок 1.19 – Минимальное значение числа (Min)

```
result := math.Mod(10, 3)
// result равен 1
```

Рисунок 1.20 – Остаток от деления x на y (Mod)

```
result := math.Round(5.67)
// result равен 6
```

Рисунок 1.21 – Округление числа к ближайшему целому (Round)

```
result := math.Trunc(5.67)
// result равен 5
```

Рисунок 1.22 – Отбрасывание дробной части числа (Trunc)

```
posInf := math.Inf(1)
// posInf - положительная бесконечность
```

Рисунок 1.23 – Получение положительной или отрицательной бесконечности в зависимости от знака (Inf)

```
nan := math.NaN()
// nan - NaN
```

Рисунок 1.24 – Возвращение числа «Not A Number» (NaN)

```
result := math.Exp(2)
// result равен примерно 7.389056...
```

Рисунок 1.25 – Возвращение экспоненты в степени x (Exp)

```
result := math.Exp2(3)
// result равен 8
```

Рисунок 1.26 – Возвращение 2 в степени x (Exp2)

```
result := math.Expm1(1)
// result равен примерно 1.718281...
```

Рисунок 1.27 – Возвращение e в степени x минус 1 (Expm1)

```
result := math.Log10(100)
// result равен 2
```

Рисунок 1.28 – Возвращение десятичного логарифма (Log10)

```
result := math.Log1p(1)
// result равен примерно 0.693147...
```

Рисунок 1.29 – Возвращение натурального логарифма числа x плюс 1
(Log1p)

```
isNegative := math.Signbit(-5)
// isNegative равен true
```

Рисунок 1.30 – Возвращение true, если x отрицательное или
отрицательный нуль (Signbit)

Часть 3. Практическая часть

Задание 1. Напишите программу, которая выводит "I like Go!"

```
package main

import "fmt"

func main() {
    |   fmt.Println("I like Go!")
}
```

Рисунок 1.31 – Задание 1. Вывод строки

```
I like Go!
PS C:\git\lab1_PI\Examples\Задание 1 - I like Go!> |
```

Рисунок 1.32 – Задание 1. Вывод строки

Задание 2. Напишите программу, которая выведет "I like Go!" 3 раза.

```
package main

import "fmt"

func main() {
    |   var str string = "I like Go!\n"
    |   fmt.Print(str, str, str)
}
```

Рисунок 1.33 – Задание 2. Тройной вывод строки

```
I like Go!
I like Go!
I like Go!
```

Рисунок 1.34 – Задание 2. Вывод

Задание 3. Напишите программу, которая последовательно делает следующие операции с введенным числом:

1. Число умножается на 2;
2. Затем к числу прибавляется 100.

После этого должен быть вывод получившегося числа на экран.

```
package main

import "fmt"

func main() {
    var x int
    fmt.Scan(&x)
    x = x*2 + 100

    fmt.Println(x)
}
```

Рисунок 1.35 – Задание 3. Умножение числа на 2 и добавление 100



```
6
112
```

Рисунок 1.36 – Задание 3. Ввод вывод

Задание 4. Петя торопился в школу и неправильно написал программу, которая сначала находит квадраты двух чисел, а затем их суммирует. Исправьте его программу.

```
package main

import "fmt"

func main(){
    var a int
    fmt.Scan(&a) // считаем переменную 'a' с консоли
    fmt.Scan(&b) // считаем переменную 'b' с консоли
    a = a * a
    b = b * 2
    c = a + b
    fmt.Println(c)
}
```

Рисунок 1.37 – Задание 4. Программа Пети

```

package main

import "fmt"

func main(){
    var a, b, c int
    fmt.Scan(&a) // считаем переменную 'a' с консоли
    fmt.Scan(&b) // считаем переменную 'b' с консоли
    a = a * a
    b = b * b
    c = a + b
    fmt.Println(c)
}

```

Рисунок 1.38 – Задание 4. Исправленная программа

```

9
2
85

```

Рисунок 1.39 – Задание 4. Ввод вывод

Задание 5. По данному целому числу, найдите его квадрат.

```

package main

import "fmt"

func main() {
    var a int
    fmt.Scan(&a)
    fmt.Println(a * a)
}

```

Рисунок 1.40 – Задание 5. Нахождение квадрата введенного числа

```

5
25

```

Рисунок 1.41 – Задание 5. Ввод вывод

Задание 6. Дано натуральное число, выведите его последнюю цифру. На вход дается натуральное число N, не превосходящее 10000. Выведите одно целое число - ответ на задачу.

```
package main

import "fmt"

func main() {
    var n, ans int
    fmt.Scan(&n)
    ans = n % 10

    fmt.Println(ans)
}
```

Рисунок 1.42 – Задание 6. Вывод последней цифры числа

```
1119
9
```

Рисунок 1.43 – Задание 6. Ввод вывод

Задание 7. Дано неотрицательное целое число. Найдите число десятков (то есть вторую цифру справа). На вход дается натуральное число N, не превосходящее 10000. Выведите одно целое число - число десятков.

```
package main

import "fmt"

func main() {
    var n, ans int
    fmt.Scan(&n)
    ans = n % 100 / 10

    fmt.Println(ans)
}
```

Рисунок 1.44 – Задание 7. Нахождение числа десятков

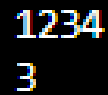


Рисунок 1.45 – Задание 7. Ввод вывод

Задание 8. Часовая стрелка повернулась с начала суток на d градусов. Определите, сколько сейчас целых часов h и целых минут m . На вход программе подается целое число d ($0 < d < 360$). Выведите на экран фразу:

It is ... hours ... minutes.

Вместо многоточий программа должна выводить значения h и m , отделяя их от слов ровно одним пробелом.

```
package main

import "fmt"

func main() {
    var d, mins, hours int
    fmt.Scan(&d)

    hours = d / 30
    mins = (d - hours*30) * 2

    fmt.Println("It is", hours, "hours", mins, "minutes.")
}
```

Рисунок 1.46 – Задание 8. Нахождение времени от градусов



Рисунок 1.47 – Задание 8. Ввод вывод

Задание 9. Уберите лишние комментарии так, чтобы программа вывела число 100.

```
package main

import "fmt"

func main(){
    // a:=44
    /*
    var a2 int = 10
    */
    a2 = a2 * 10
    fmt.Println(a2)
}
```

Рисунок 1.48 – Задание 9. Убрать лишние комментарии

```
package main

import "fmt"

func main() {
    // a:=44
    var a2 int = 10
    a2 = a2 * 10
    fmt.Println(a2)
}
```

Рисунок 1.49 – Задание 9. Результат

```
100
```

Рисунок 1.50 – Задание 9. Вывод

Задание 10. Исправьте ошибку в программе

```
package main

import "fmt"

func main(){
    var a int = 8
    const b int = 10
    a = a + b
    b = b + a
    fmt.Println(a)
}
```

Рисунок 1.51 – Задание 10. Исправить ошибку

```
package main

import "fmt"

func main(){
    var a int = 8
    const b int = 10
    a = a + b
    b = b + a
    fmt.Println(a)
}
```

Рисунок 1.52 – Задание 10. Ошибка

18

Рисунок 1.53 – Задание 10. Вывод

Задание 11. Напишите программу, которая для заданных значений a и b вычисляет площадь поверхности и объем тела, образованного вращением эллипса, заданного уравнением:

$$\frac{x^2}{a^2} + \frac{y^2}{b^2} = 1,$$

Вокруг оси Ox

```
package main

import (
    "fmt"
    "math"
)

func main() {
    var (
        a float32 = 10
        b float32 = 20
    )

    s := 2 * math.Pi * a * b
    v := (4 / 3) * math.Pi * a * b * b

    fmt.Printf("Площадь = %1.2f \nОбъем = %1.2f", s, v)
}
```

Рисунок 1.54 – Задание 10. Вычисление площади и объема

```
Площадь = 1256.64
Объем = 12566.37
```

Рисунок 1.55 – Задание 10. Вывод

Часть 4. Индивидуальные задания (2 вариант)

Задание №1: Задайте координаты двух векторов в трехмерном пространстве. Рассчитайте и выведите угол между ними

```
package main

import (
    "fmt"
    "math"
)

func main() {
    var (
        x1 float64 = 10
        y1 float64 = 0
        z1 float64 = 0
        x2 float64 = 0
        y2 float64 = 0
        z2 float64 = 10
    )

    ans := math.Acos((x1*x2+y1*y2+z1*z2)/
        (math.Sqrt(x1*x1+y1*y1+z1*z1)*
            math.Sqrt(x2*x2+y2*y2+z2*z2))) / math.Pi * 180
    fmt.Printf("Угол между векторами равен %.2f градуса", ans)
}
```

Рисунок 1.54 – Индивидуальное задание 1. Угол между векторами

Угол между векторами равен 90.00 градуса

Рисунок 1.56 – Индивидуальное задание 1. Вывод

Задание №2: Даны стороны прямоугольника. Найти его периметр и длину диагонали.

```
package main

import (
    "fmt"
    "math"
)

func main() {
    var width int = 15
    var height int = 10

    var perimetr = width*2 + height*2
    var diag = math.Sqrt(float64(width*width + height*height))

    fmt.Println("Периметр прямоугольника = ", perimetr)
    fmt.Println("Длина диагонали прямоугольника = ", diag)
}
```

Рисунок 1.55 – Индивидуальное задание 2. Периметр и диагональ
прямоугольника

```
Периметр прямоугольника = 50
Длина диагонали прямоугольника = 18.027756377319946
```

Рисунок 1.56 – Индивидуальное задание 2. Вывод

Выводы

В процессе выполнения лабораторной работы №1 мы рассмотрели базовые основы программирования на языке Go. Язык был самостоятельно установлен на компьютер. Были изучены основные типы данных, констант, арифметических операций языка программирования Go.

Ответы на контрольные вопросы

1. Как объявить переменную типа `int` в Go?

- Переменную типа `int` можно объявить строкой «`var x int`»

2. Какое значение по умолчанию присваивается переменной типа `int` в Go?

- По умолчанию переменной типа `int` присваивается значение 0

3. Как изменить значение существующей переменной в Go?

- Изменить значение существующей переменной можно с помощью оператора присваивания: «`x = 4`»

4. Что такое множественное объявление переменных в Go?

- Множественное объявление переменных в Go означает что в рамках одной конструкции будут обозначены несколько переменных

5. Как объявить константу в Go?

- Константу в Go можно объявить с помощью «`const`»

6. Можно ли изменить значение константы после ее объявления в Go?

- Значения констант нельзя изменить

7. Какие арифметические операторы поддерживаются в Go?

- сложение, вычитание, умножение, деление

8. Какой оператор используется для выполнения операции остатка в Go?

- для выполнения операции остатка используется оператор «`%`»

9. Какой результат выражения `5 / 2` в Go?

- выражение $5/2$ в Go имеет результат 2

10. Как считать строку с консоли в Go?

- для чтения данных с консоли используется метод «`fmt.Scan(&a)`»

11. Как считать целое число с консоли в Go?

- на рисунке 1.56 показан код для считывания целого числа с консоли

```
package main

import "fmt"

func main() {
    var x int
    fmt.Scan(&x)
}
```

Рисунок 1.56 – Считывание целого числа с консоли

12. Как обработать ошибку при считывании данных с консоли в Go?

- нужно создать ошибку функцией «`errors.New`». Эта функция позволяет указывать настраиваемое сообщение об ошибке, которую можно отобразить пользователю: «`err := errors.New("error")`»

13. Как вывести строку в консоль в Go?

- Строку в консоль можно вывести методом «`fmt.Print()`»

14. Как вывести значение переменной типа `int` в консоль?

- вывести значение переменной типа `int` в консоль можно с помощью метода «`fmt.Print()`» с именем переменной.

15. Как форматировать вывод числа с плавающей точкой в Go?

- форматировать вывод числа с плавающей точкой можно с помощью спецификатора `%`. Например «`%6.3f`» - делает ширину числа 6 и точность 3 (`fmt.Printf`).

16. Как объявить переменную типа `byte` и присвоить ей значение 65? Чем отличается оператор `:=` от оператора `=` в Go?

- объявить переменную типа `byte` и присвоить ей значение 65 можно строкой `«var x byte = 65»`. Оператор `:=` автоматически объявляет переменную и присваивает ей значение.

17. Какие типы данных можно использовать для представления чисел с плавающей точкой в Go?

- `float32`, `float64`, `complex64` и `complex128`

18. Как объявить и использовать несколько переменных в Go?

- несколько переменных можно объявить в одном блоке `var`:

```
Var (  
    Name string = "Dima"  
    Age int = 23  
)
```