

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ
ФГАОУ ВО «СЕВЕРО-КАВКАЗСКАЯ ФЕДЕРАЛЬНЫЙ
УНИВЕРСИТЕТ» ИНСТИТУТ ЦИФРОВОГО РАЗВИТИЯ
КАФЕДРА ИНФОКОММУНИКАЦИЙ

Дисциплина: Программная инженерия

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №1

Основы языка программирования Go

Выполнил:

студент 3 курса направления
подготовки «Программная
инженерия»

группы ПИЖ-б-о-21-1

Трушева Вероника Олеговна

(Подпись)

Проверил:

доцент кафедры инфокоммуникаций
института цифрового развития

Воронкин Роман Александрович

(Подпись)

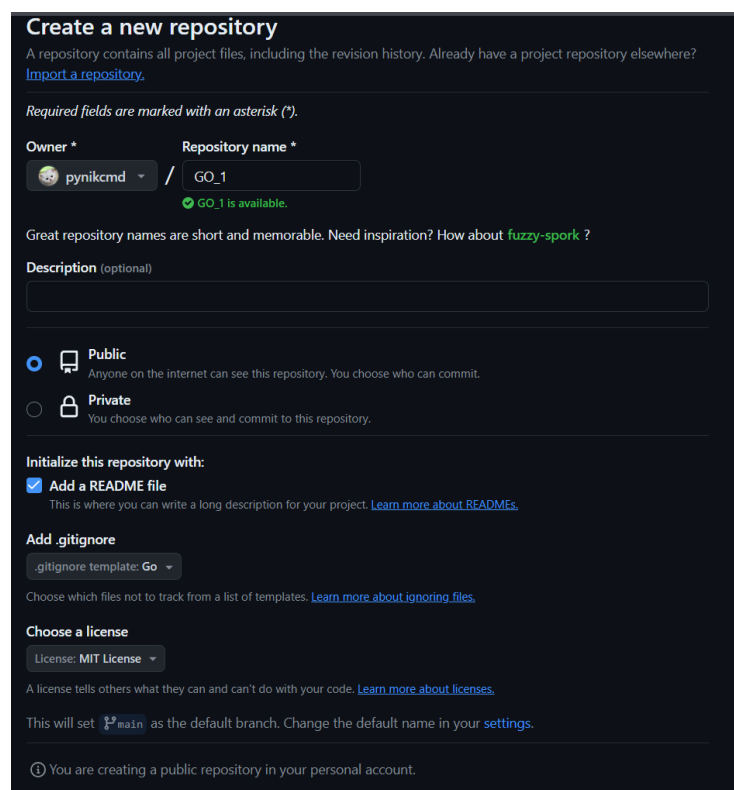
Работа защищена с оценкой:

Ставрополь, 2024

Цель: исследование назначения и способов установки Go, исследование типов данных, констант и арифметических операций языка программирования Go.

Ход работы

1. Изучить теоретический материал работы.
2. Создать общедоступный репозиторий на GitHub (или ином сервисе), в котором будет использована лицензия MIT и язык программирования Go.



Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)

Required fields are marked with an asterisk (*).

Owner * / Repository name *
pynikcmd / GO_1
GO_1 is available.

Great repository names are short and memorable. Need inspiration? How about [fuzzy-spork](#) ?

Description (optional)

☒ Public
Anyone on the internet can see this repository. You choose who can commit.

☐ Private
You choose who can see and commit to this repository.

Initialize this repository with:

☒ Add a README file
This is where you can write a long description for your project. [Learn more about READMEs.](#)

Add .gitignore
gitignore template: Go

Choose which files not to track from a list of templates. [Learn more about ignoring files.](#)

Choose a license
License: MIT License

A license tells others what they can and can't do with your code. [Learn more about licenses.](#)

This will set `main` as the default branch. Change the default name in your [settings](#).

① You are creating a public repository in your personal account.

Рисунок 1 – Создание репозитория

3. Выполните клонирование созданного репозитория.

```
D:\fgit>git clone https://github.com/pynikcmd/GO_1.git
Cloning into 'GO_1'...
remote: Enumerating objects: 14, done.
remote: Counting objects: 100% (14/14), done.
remote: Compressing objects: 100% (12/12), done.
remote: Total 14 (delta 3), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (14/14), 4.74 KiB | 2.37 MiB/s, done.
Resolving deltas: 100% (3/3), done.
```

Рисунок 2 – Клонирование репозитория

4. Дополните файл .gitignore необходимыми правилами для работы с используемой IDE (Visual Studio Code и т. д.).



Рисунок 3 – Дополнение файла .gitignore

5. Проработайте примеры теоретической части.

1 Пример:

```
Primers > -go 1.go > ...
1  package main
2
3  import "fmt"
4
5  func main() {
6      fmt.Println("Hello, Go!")
7  }
8
```

Рисунок 4 – Код программы 1 примера

```
PS D:\fgit\GO_1\Primers> go run 1.go
Hello, Go!
```

Рисунок 5 – Результат работы программы 1 примера

2 Пример:

```
Primers > -go 2.go > main
1  package main
2
3  import "fmt"
4
5  func main() {
6      fmt.Println("Hello Go"[0]) // вывод: 72
7  }
8
```

Рисунок 6 – Код программы 2 примера

```
PS D:\fgit\GO_1\Primers> go run 2.go
72
```

Рисунок 7 – Результат работы программы 2 примера

3 Пример:

```

Primers > -go 3.go > main
1  package main
2
3  import "fmt"
4
5  func main() {
6      fmt.Println(string("Hello Go"[0])) // вывод: H
7  }
8

```

Рисунок 8 – Код программы 3 примера

```

PS D:\fgit\GO_1\Primers> go run 3.go
H

```

Рисунок 9 – Результат работы программы 3 примера

4 Пример:

```

Primers > -go 4.go > ...
1  package main
2
3  import "fmt"
4
5  func main() {
6      var x int
7      var y float64
8      var max, min int
9      x = 10
10     var c string = "Hello World!"
11     var z float64 = 1.045
12     var a = 12
13     var hello = "Hello"
14
15     fmt.Printf("x: %v, type: %T\n", x, x)
16     fmt.Printf("y: %v, type: %T\n", y, y)
17     fmt.Printf("max: %v, type: %T\n", max, max)
18     fmt.Printf("min: %v, type: %T\n", min, min)
19     fmt.Printf("c: %v, type: %T\n", c, c)
20     fmt.Printf("z: %v, type: %T\n", z, z)
21     fmt.Printf("a: %v, type: %T\n", a, a)
22     fmt.Printf("hello: %v, type: %T\n", hello, hello)
23 }

```

Рисунок 10 – Код программы 4 примера

```

x: 10, type: int
y: 0, type: float64
max: 0, type: int
min: 0, type: int
c: Hello World!, type: string
z: 1.045, type: float64
a: 12, type: int
hello: Hello, type: string

```

Рисунок 11 – Результат работы программы 4 примера

5 Пример:

```
Primers > 5.go > ...
1 package main
2
3 import "fmt"
4
5 func main() {
6     var hello string
7     hello = "Hello Go!"
8     var a int = 2019
9     fmt.Println(hello)
10    fmt.Println(a)
11 }
```

Рисунок 12 – Код программы 5 примера

```
PS D:\fgit\GO_1\Primers> go run 5.go
Hello Go!
2019
```

Рисунок 13 – Результат работы программы 5 примера

6 Пример:

```
Primers > 6.go > ...
1 package main
2
3 import "fmt"
4
5 func main() {
6     var symbol int32 = 'c'
7     fmt.Println(string(symbol))
8 }
```

Рисунок 14 – Код программы 6 примера

```
c
PS D:\fgit\GO_1\Primers> go run 6.go
```

Рисунок 15 – Результат работы программы 6 примера

7 Пример:

```
Primers > 7.go > ...
1 package main
2
3 import "fmt"
4
5 func main() {
6     var (
7         name string = "Dima"
8         age int = 23
9     )
10
11    fmt.Println(name)
12    fmt.Println(age)
13 }
```

Рисунок 16 – Код программы 7 примера

```
PS D:\fgit\GO_1\Primers> go run 7.go
Dima
23
```

Рисунок 17 – Результат работы программы 7 примера

8 Пример:

```
Primers > go 8.go > ...
1  package main
2
3  import "fmt"
4
5  func main() {
6      a := 100
7      b := 10
8      c := a + b
9      fmt.Println("c =", c) // Вывод: c = 110
10
11     c = a * b
12     fmt.Println("c =", c) // Вывод: c = 1000
13
14     c = a - b
15     fmt.Println("c =", c) // Вывод: c = 90
16
17     c = a / b
18     fmt.Println("c =", c) // Вывод: c = 10
19
20     c = a % 3
21     fmt.Println("c =", c) // Вывод: c = 1
22
23     var d int = 1
24     d++
25     fmt.Println("d =", d) // Вывод: d = 2
26
27     var e int = 10
28     e--
29     fmt.Println("e =", e) // Вывод: e = 9
30 }
```

Рисунок 18 – Код программы 8 примера

```
PS D:\fgit\GO_1\Primers> go run 8.go
c = 110
c = 1000
c = 90
c = 10
c = 1
d = 2
e = 9
```

Рисунок 19 – Результат работы программы 8 примера

9 Пример:

```
Primers > go 9.go > ...
1  package main
2
3  import "fmt"
4
5  func main() {
6      var name string
7      var age int
8      fmt.Print("Введите имя: ")
9      fmt.Scan(&name)
10     fmt.Print("Введите возраст: ")
11     fmt.Scan(&age)
12
13     fmt.Println(name, age)
14 }
```

Рисунок 20 – Код программы 9 примера

```
PS D:\fgit\GO_1\Primers> go run 9.go
Введите имя: Вероника
Введите возраст: 22
Вероника 22
```

Рисунок 21 – Результат работы программы 9 примера

10 Пример:

```
Primers > go 10.go > ...
1 package main
2
3 import "fmt"
4
5 func main() {
6     var a, b, c int
7
8     fmt.Println("Введите три целых числа через пробел:")
9
10    fmt.Scan(&a, &b, &c)
11
12    fmt.Println("Вы ввели:")
13    fmt.Println("a =", a)
14    fmt.Println("b =", b)
15    fmt.Println("c =", c)
16 }
```

Рисунок 22 – Код программы 10 примера

```
PS D:\fgit\GO_1\Primers> go run 10.go
Введите три целых числа через пробел:
4 5 2
Вы ввели:
a = 4
b = 5
c = 2
```

Рисунок 23 – Результат работы программы 10 примера

11 Пример:

```
1 package main
2
3 import "fmt"
4
5 func main() {
6     name := "Ivan"
7     age := 27
8     fmt.Println("My name is", name, "and I am", age, "years old")
9 }
```

Рисунок 24 – Код программы 11 примера

```
PS D:\fgit\GO_1\Primers> go run 11.go
My name is Ivan and I am 27 years old.
```

Рисунок 25 – Результат работы программы 11 примера

12 Пример:

```
1  /*
2   Первая программа
3   на языке Go
4   */
5
6  package main // определение пакета для текущего файла
7
8  import "fmt" // подключение пакета fmt
9
10 // определение функции main
11
12 func main() {
13     fmt.Println("Hello Go!") // вывод строки на консоль
14 }
```

Рисунок 26 – Код программы 12 примера

```
PS D:\fgit\GO_1\Primers> go run 12.go
Hello Go!
```

Рисунок 27 – Результат работы программы 12 примера

13 Пример:

```
1  package main
2
3  import "fmt"
4
5  const (
6      A int = 45
7      B
8      C float32 = 3.3
9      D
10 )
11
12 func main() {
13     fmt.Println(A, B, C, D) // Вывод: 45 45 3.3 3.3
14 }
15
```

Рисунок 28 – Код программы 13 примера

```
PS D:\fgit\GO_1\Primers> go run 13.go
45 45 3.3 3.3
```

Рисунок 29 – Результат работы программы 13 примера

14 Пример:

```
1  package main
2
3  import "fmt"
4
5  const (
6      Sunday    = 0
7      Monday    = 1
8      Tuesday   = 2
9      Wednesday = 3
10     Thursday  = 4
11     Friday    = 5
12     Saturday  = 6
13 )
14
15 func main() {
16     fmt.Println(Sunday) // ВЫВОД 0
17     fmt.Println(Saturday) // ВЫВОД 6
18 }
```

Рисунок 30 – Код программы 14 примера

```
PS D:\fgit\GO_1\Primers> go run 14.go
0
6
```

Рисунок 31 – Результат работы программы 14 примера

```
1  package main
2
3  import "fmt"
4
5  const (
6      Sunday    = iota
7      Monday    = 1
8      Tuesday   = 2
9      Wednesday = 3
10     Thursday  = 4
11     Friday    = 5
12     Saturday  = 6
13 )
14
15 func main() {
16     fmt.Println(Sunday) // ВЫВОД 0
17     fmt.Println(Saturday) // ВЫВОД 6
18 }
```

Рисунок 32 – Код программы 14 примера

```
PS D:\fgit\GO_1\Primers> go run 14.go
0
6
```

Рисунок 33 – Результат работы программы 14 примера

15 Пример:

```
1 package main
2
3 import "fmt"
4
5 const (
6     c0 = iota // c0 == 0
7     c1 = iota // c1 == 1
8     c2 = iota // c2 == 2
9
10 )
11
12 func main() {
13     fmt.Println(c0, c1, c2) // вывод: 0 1 2
14 }
15
```

Рисунок 34 – Код программы 15 примера

```
PS D:\fgit\GO_1\Primers> go run 15.go
0 1 2
```

Рисунок 35 – Результат работы программы 15 примера

16 Пример:

```
1 package main
2
3 import "fmt"
4
5 const (
6     u = iota * 42 // u == 0 (индекс - 0, поэтому 0 * 42 = 0)
7     v float64 = iota * 42 // v == 42.0 (индекс - 1, поэтому 1.0 * 42 = 42.0)
8     w = iota * 42 // w == 84 (индекс - 2, поэтому 2 * 42 = 84)
9
10 )
11
12 // переменные ни в одной блоке const, поэтому индекс не увеличился
13 const x = iota // x == 0
14 const y = iota // y == 0
15
16 func main() {
17     fmt.Println(x, y) // вывод: 0 0
18     fmt.Println(u, v, w) // вывод: 0 42 84
19 }
20
```

Рисунок 36 – Код программы 16 примера

```
PS D:\fgit\GO_1\Primers> go run 16.go
0 0
0 42 84
```

Рисунок 37 – Результат работы программы 16 примера

Математические примеры

1. `Abs(x float64) float64` : Возвращает абсолютное значение числа.

```
1 package main
2
3 import (
4     "fmt"
5     "math"
6 )
7
8 func main() {
9     result := math.Abs(-5.67)
10    fmt.Println(result)
11 }
```

Рисунок 38 – Код программы

```
PS D:\fgit\GO_1\Primers> go run 17.go
5.67
```

Рисунок 39 – Результат работы программы

2. `Ceil(x float64) float64` : Округляет число вверх до ближайшего целого.

```
1 package main
2
3 import (
4     "fmt"
5     "math"
6 )
7
8 func main() {
9     result := math.Ceil(5.67)
10    fmt.Println(result)
11 }
```

Рисунок 40 – Код программы

```
PS D:\fgit\GO_1\Primers> go run 17.go
6
```

Рисунок 41 – Результат работы программы

3. `Floor(x float64) float64` : Округляет число вниз до ближайшего целого.

```
1 package main
2
3 import (
4     "fmt"
5     "math"
6 )
7
8 func main() {
9     result := math.Floor(5.67)
10    fmt.Println(result)
11 }
```

Рисунок 42 – Код программы

```
PS D:\fgit\GO_1\Primers> go run 17.go
5
```

Рисунок 43 – Результат работы программы

4. Sqrt(x float64) float64 : Возвращает квадратный корень числа.

```
1 package main
2
3 import (
4     "fmt"
5     "math"
6 )
7
8 func main() {
9     result := math.Sqrt(16)
10    fmt.Println(result)
11 }
```

Рисунок 44 – Код программы

```
PS D:\fgit\GO_1\Primers> go run 17.go
4
```

Рисунок 45 – Результат работы программы

5. Pow(x, y float64) float64 : Возводит число x в степень y.

```
1 package main
2
3 import (
4     "fmt"
5     "math"
6 )
7
8 func main() {
9     result := math.Pow(2, 3)
10    fmt.Println(result)
11 }
```

Рисунок 46 – Код программы

```
PS D:\fgit\GO_1\Primers> go run 17.go
8
```

Рисунок 47 – Результат работы программы

6. Sin(x float64) float64 , Cos(x float64) float64 , Tan(x float64) float64 :

Возвращают синус, косинус и тангенс угла в радианах соответственно.

```
1 package main
2
3 import (
4     "fmt"
5     "math"
6 )
7
8 func main() {
9     sinValue := math.Sin(math.Pi / 2)
10    fmt.Println(sinValue)
11 }
```

Рисунок 48 – Код программы

```
PS D:\fgit\GO_1\Primers> go run 17.go
1
```

Рисунок 49 – Результат работы программы

7. Log(x float64) float64 : Возвращает натуральный логарифм числа.

```
1 package main
2
3 import (
4     "fmt"
5     "math"
6 )
7
8 func main() {
9     result := math.Log(10)
10    fmt.Println(result)
11 }
12
```

Рисунок 50 – Код программы

```
PS D:\fgit\GO_1\Primers> go run 17.go
2.302585092994046
```

Рисунок 51 – Результат работы программы

8. Max(x, y float64) float64 , Min(x, y float64) float64 : Возвращают максимальное и минимальное значение из двух чисел соответственно.

```
1 package main
2
3 import (
4     "fmt"
5     "math"
6 )
7
8 func main() {
9     maxVal := math.Max(3, 7)
10    minVal := math.Min(3, 7)
11
12    fmt.Println(maxVal)
13    fmt.Println(minVal)
14 }
```

Рисунок 52 – Код программы

```
PS D:\fgit\GO_1\Primers> go run 17.go
7
3
```

Рисунок 53 – Результат работы программы

9. `Mod(x, y float64) float64` : Возвращает остаток от деления `x` на `y`.

```
1  package main
2
3  import (
4      "fmt"
5      "math"
6  )
7
8  func main() {
9      result := math.Mod(10, 3)
10
11      fmt.Println(result)
12  }
```

Рисунок 54 – Код программы

```
PS D:\fgit\GO_1\Primers> go run 17.go
1
```

Рисунок 55 – Результат работы программы

10. `Round(x float64) float64` : Округляет число к ближайшему целому.

```
1  package main
2
3  import (
4      "fmt"
5      "math"
6  )
7
8  func main() {
9      result := math.Round(5.67)
10
11      fmt.Println(result)
12  }
```

Рисунок 56 – Код программы

```
PS D:\fgit\GO_1\Primers> go run 17.go
6
```

Рисунок 57 – Результат работы программы

11. `Trunc(x float64) float64` : Отбрасывает дробную часть числа.

```
1 package main
2
3 import (
4     "fmt"
5     "math"
6 )
7
8 func main() {
9     result := math.Trunc(5.67)
10
11     fmt.Println(result)
12 }
```

Рисунок 58 – Код программы

```
PS D:\fgit\GO_1\Primers> go run 17.go
5
```

Рисунок 59 – Результат работы программы

12. `Inf(sign int) float64` : Возвращает положительную или отрицательную бесконечность в зависимости от знака.

```
1 package main
2
3 import (
4     "fmt"
5     "math"
6 )
7
8 func main() {
9     posInf := math.Inf(1)
10     // posInf - положительная бесконечность
11     negInf := math.Inf(-1)
12     // negInf - отрицательная бесконечность
13
14     fmt.Println(posInf)
15     fmt.Println(negInf)
16 }
```

Рисунок 60 – Код программы

```
PS D:\fgit\GO_1\Primers> go run 17.go
+Inf
-Inf
```

Рисунок 61 – Результат работы программы

13. NaN() float64 : Возвращает "Not a Number" (NaN).

```
1 package main
2
3 import (
4     "fmt"
5     "math"
6 )
7
8 func main() {
9     nan := math.NaN()
10
11     fmt.Println(nan)
12 }
```

Рисунок 62 – Код программы

```
PS D:\fgit\GO_1\Primers> go run 17.go
NaN
```

Рисунок 63 – Результат работы программы

14. Exp(x float64) float64 : Возвращает экспоненту (e) в степени x.

```
1 package main
2
3 import (
4     "fmt"
5     "math"
6 )
7
8 func main() {
9     result := math.Exp(2)
10    fmt.Println(result)
11 }
```

Рисунок 64 – Код программы

```
PS D:\fgit\GO_1\Primers> go run 17.go
7.38905609893065
```

Рисунок 65 – Результат работы программы

15. Exp2(x float64) float64 : Возвращает 2 в степени x.

```
1 package main
2
3 import (
4     "fmt"
5     "math"
6 )
7
8 func main() {
9     result := math.Exp2(3)
10    fmt.Println(result)
11 }
```

Рисунок 66 – Код программы


```
PS D:\fgit\GO_1\Primers> go run 17.go
8
```

Рисунок 67 – Результат работы программы

16. `Expn1(x float64) float64` : Возвращает e в степени x минус 1.

```
1 package main
2
3 import (
4     "fmt"
5     "math"
6 )
7
8 func main() {
9     result := math.Exp1(1)
10    fmt.Println(result)
11 }
```

Рисунок 68 – Код программы

```
PS D:\fgit\GO_1\Primers> go run 17.go
1.718281828459045
```

Рисунок 69 – Результат работы программы

17. `Log10(x float64) float64` : Возвращает десятичный логарифм числа.

```
1 package main
2
3 import (
4     "fmt"
5     "math"
6 )
7
8 func main() {
9     result := math.Log10(100)
10    fmt.Println(result)
11 }
```

Рисунок 70 – Код программы

```
PS D:\fgit\GO_1\Primers> go run 17.go
2
```

Рисунок 71 – Результат работы программы

18. `Log2(x float64) float64` : Возвращает двоичный логарифм числа.

```
1 package main
2
3 import (
4     "fmt"
5     "math"
6 )
7
8 func main() {
9     result := math.Log2(8)
10    fmt.Println(result)
11 }
```

Рисунок 72 – Код программы

```
PS D:\fgit\GO_1\Primers> go run 17.go
3
```

Рисунок 73 – Результат работы программы

19. `Log1p(x float64) float64` : Возвращает натуральный логарифм числа `x` плюс 1.

```
1 package main
2
3 import (
4     "fmt"
5     "math"
6 )
7
8 func main() {
9     result := math.Log1p(1)
10    fmt.Println(result)
11 }
```

Рисунок 74 – Код программы

```
PS D:\fgit\GO_1\Primers> go run 17.go
0.6931471805599453
```

Рисунок 75 – Результат работы программы

20. `Signbit(x float64) bool` : Возвращает `true`, если `x` отрицательное или отрицательный ноль.

```
1 package main
2
3 import (
4     "fmt"
5     "math"
6 )
7
8 func main() {
9     isNegative := math.Signbit(-5)
10    fmt.Println(isNegative)
11 }
```

Рисунок 76 – Код программы

```
PS D:\fgit\GO_1\Primers> go run 17.go
true
```

Рисунок 77 – Результат работы программы

6. Решите все задания практической части.

Задание 1

Задача: Напишите программу, которая выводит "I like Go!"

```

1  package main
2
3  import "fmt"
4
5  func main() {
6      fmt.Println("I like Go!")
7  }

```

Рисунок 78 – Код программы 1 задания

```

PS D:\fgit\GO_1\Primers> go run 1.go
I like Go!

```

Рисунок 79 – Результат работы программы 1 задания

Задание 2

Задача: Напишите программу, которая выведет "I like Go!" 3 раза.

```

1  package main
2
3  import "fmt"
4
5  func main() {
6      fmt.Printf("%s\n%s\n%s\n", "I like Go!", "I like Go!", "I like Go!")
7  }

```

Рисунок 80 – Код программы 2 задания

```

1  package main
2
3  import "fmt"
4
5  func main() {
6      for i := 0; i < 3; i++ {
7          fmt.Println("I like Go!")
8      }
9  }

```

Рисунок 81 – Код программы 2 задания

```

PS D:\fgit\GO_1\Primers> go run 2.go
I like Go!
I like Go!
I like Go!

```

Рисунок 82 – Результат работы программы 2 задания

Задание 3

Задача: Напишите программу, которая последовательно делает следующие операции с введённым числом:

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

После этого должен быть вывод получившегося числа на экран.

```
1 package main
2
3 import "fmt"
4
5 func main() {
6     var number int
7
8     fmt.Println("Введите число:")
9     fmt.Scan(&number)
10
11     result := number * 2
12     result += 100
13
14     fmt.Println("Результат:", result)
15 }
```

Рисунок 83 – Код программы 3 задания

```
PS D:\fgit\GO_1\Primers> go run 3.go
Введите число:
5
Результат: 110
```

Рисунок 84 – Результат работы программы 3 задания

Задание 4

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

```
1 package main
2
3 import "fmt"
4
5 func main() {
6     var a int
7     fmt.Scan(&a) // считаем переменную 'a' с консоли
8     fmt.Scan(&b) // считаем переменную 'b' с консоли
9     a = a * a
10    b = b * 2
11    c = a + b
12    fmt.Println(c)
13 }
```

Рисунок 85 – Неверный код программы 4 задания

```

1  package main
2
3  import "fmt"
4
5  func main() {
6      var a, b, c int
7
8      fmt.Println("Введите два числа:")
9      fmt.Scan(&a)
10     fmt.Scan(&b)
11
12     a = a * a
13     b = b * b
14     c = a + b
15
16     fmt.Println(c)
17 }

```

Рисунок 86 – Код программы 4 задания

```

PS D:\fgit\GO_1\Primers> go run 4.go
Введите два числа:
5
4
41

```

Рисунок 87 – Результат работы программы 4 задания

Задание 5

Задача: По данному целому числу, найдите его квадрат.

```

1  package main
2
3  import "fmt"
4
5  func main() {
6      var number int
7
8      fmt.Println("Введите целое число:")
9      fmt.Scan(&number)
10
11     fmt.Println(number * number)
12 }

```

Рисунок 88 – Код программы 5 задания

```

PS D:\fgit\GO_1\Primers> go run 5.go
Введите целое число:
5
25

```

Рисунок 89 – Результат работы программы 5 задания

Задание 6

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

```
1 package main
2
3 import "fmt"
4
5 func main() {
6     var number int
7
8     // Ввод числа с клавиатуры
9     fmt.Println("Введите натуральное число:")
10    fmt.Scan(&number)
11
12    // Находим последнюю цифру числа
13    result := number % 10
14
15    // Выводим последнюю цифру
16    fmt.Println(result)
17 }
```

Рисунок 90 – Код программы 6 задания

```
PS D:\fgit\GO_1\Primers> go run 6.go
Введите натуральное число:
78
8
```

Рисунок 91 – Результат работы программы 6 задания

Задание 7

Задача: 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.

```
1 package main
2
3 import "fmt"
4
5 func main() {
6     var number int
7
8     // Ввод числа с клавиатуры
9     fmt.Println("Введите неотрицательное целое число:")
10    fmt.Scan(&number)
11
12    // Находим число десятков
13    result := (number / 10) % 10
14
15    // Выводим число десятков
16    fmt.Println(result)
17 }
```

Рисунок 92 – Код программы 7 задания

```
PS D:\fgit\GO_1\Primers> go run 7.go
Введите неотрицательное целое число:
4567
6
```

Рисунок 93 – Результат работы программы 7 задания

Задание 8

Задача: 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.

```
1 package main
2
3 import "fmt"
4
5 func main() {
6     var degrees int
7
8     // Ввод числа градусов с клавиатуры
9     fmt.Println("Введите число градусов:")
10    fmt.Scan(&degrees)
11
12    // Переводим градусы в часы и минуты
13    hours := degrees / 30
14    minutes := (degrees % 30) * 2
15
16    // Выводим результат
17    fmt.Printf("It is %d hours %d minutes.\n", hours, minutes)
18 }
19
```

Рисунок 94 – Код программы 8 задания

```
PS D:\fgit\GO_1\Primers> go run 8.go
Введите число градусов:
180
It is 6 hours 0 minutes.
```

Рисунок 95 – Результат работы программы 8 задания

Задание 9

Задача: Уберите лишние комментарии так, чтобы программа вывела число 100.

```

1  package main
2
3  import "fmt"
4
5  func main() {
6      // a:=44
7      /*
8      | var a2 int = 10
9      */
10     a2 = a2 * 10
11     fmt.Println(a2)
12 }

```

Рисунок 96 – Неверный код программы 9 задания

```

1  package main
2
3  import "fmt"
4
5  func main() {
6      var a2 int = 10
7      a2 = a2 * 10
8      fmt.Println(a2)
9  }

```

Рисунок 97 – Код программы 9 задания

```

PS D:\fgit\GO_1\Primers> go run 9.go
100

```

Рисунок 98 – Результат работы программы 9 задания

Задание 10

Задача: Исправьте ошибку в программе ниже:

```

1  package main
2
3  import "fmt"
4
5  func main() {
6      var a int = 8
7      const b int = 10
8      a = a + b
9      b = b + a
10     fmt.Println(a)
11 }
12

```

Рисунок 99 – Неверный код программы 10 задания

В языке Go, объявление констант не позволяет их изменять после инициализации.


```

1  package main
2
3  import "fmt"
4
5  func main() {
6      var a int = 8
7      const b int = 10
8      a = a + b
9      fmt.Println(a)
10 }

```

Рисунок 100 – Код программы 10 задания

```

PS D:\fgit\GO_1\Primers> go run 10.go
18

```

Рисунок 101 – Результат работы программы 10 задания

Задание 11

Задача: 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 .

```

4      "fmt"
5      "math/rand"
6  )
7
8  func main() {
9      // Задаем значения полуосей эллипса
10     var a, b float64
11     fmt.Println("Введите значение a:")
12     fmt.Scan(&a)
13     fmt.Println("Введите значение b:")
14     fmt.Scan(&b)
15
16     // Количество случайных точек
17     numPoints := 10000
18
19     // Переменные для подсчета точек внутри эллипса
20     pointsInside := 0
21
22     // Генерируем случайные точки и подсчитываем количество точек внутри эллипса
23     for i := 0; i < numPoints; i++ {
24         x := rand.Float64()*a*2 - a
25         y := rand.Float64()*b*2 - b
26         if x*x/(a*a)+y*y/(b*b) <= 1 {
27             pointsInside++
28         }
29     }
30
31     // Оцениваем площадь поверхности эллипса
32     surfaceArea := float64(pointsInside) / float64(numPoints) * (4 * a * b)
33
34     // Оцениваем объем тела, образованного вращением эллипса вокруг оси Oх
35     volume := surfaceArea * (2.0 / 3.0) * a
36
37     // Выводим результат
38     fmt.Printf("Площадь поверхности: %.2f\n", surfaceArea)
39     fmt.Printf("Оценка объема тела: %.2f\n", volume)

```

Рисунок 102 – Код программы

```

PS D:\fgit\GO_1\Primers> go run 11.go
Введите значение a:
4
Введите значение b:
9
Площадь поверхности: 113.01
Оценка объема тела: 301.36

```

Рисунок 103 – Результат работы программы 11 задания

7. Решите индивидуальное задание согласно варианта (номер варианта необходимо уточнить у преподавателя).

Вариант – 23

Индивидуальное задание 1

Объем пирамиды: Задайте переменные для длины, ширины основания и высоты пирамиды. Рассчитайте и выведите объем пирамиды.

Формула для расчета объема пирамиды:

$$V = \frac{1}{3} S_{\text{осн}} h$$

```

1  package main
2
3  import "fmt"
4
5  func main() {
6      // Переменные для длины, ширины основания и высоты пирамиды
7      var length, width, height float64
8
9      // Вводим значения
10     fmt.Println("Введите длину основания пирамиды:")
11     fmt.Scan(&length)
12
13     fmt.Println("Введите ширину основания пирамиды:")
14     fmt.Scan(&width)
15
16     fmt.Println("Введите высоту пирамиды:")
17     fmt.Scan(&height)
18
19     // Объем пирамиды
20     volume := (length * width * height) / 3.0
21
22     // Результат
23     fmt.Printf("Объем пирамиды: %.2f\n", volume)
24 }

```

Рисунок 104 – Код программы 1 задания

```

PS D:\fgit\GO_1\Primers> go run 1.go
Введите длину основания пирамиды:
5
Введите ширину основания пирамиды:
8
Введите высоту пирамиды:
9
Объем пирамиды: 120.00

```

Рисунок 105 – Результат работы программы 1 задания

Индивидуальное задание 2

Даны катеты прямоугольного треугольника. Найти его периметр.

Периметр треугольника равен сумме длины его трех сторон:

$$P_{\triangle ABC} = a + b + c$$

Если катеты a и b , то гипотенуза c . Где гипотенузу можно найти с помощью теоремы Пифагора:

$$a^2 + b^2 = c^2$$

$$c = \sqrt{a^2 + b^2}$$

```

1  package main
2
3  import (
4      "fmt"
5      "math"
6  )
7
8  func main() {
9      // Задаем значения катетов прямоугольного треугольника
10     var a, b float64
11     fmt.Println("Введите длину первого катета:")
12     fmt.Scan(&a)
13     fmt.Println("Введите длину второго катета:")
14     fmt.Scan(&b)
15
16     // Вычисляем гипотенузу
17     c := math.Sqrt(a*a + b*b)
18
19     // Вычисляем периметр
20     per := a + b + c
21
22     // Выводим результат
23     fmt.Printf("Периметр прямоугольного треугольника: %.2f\n", per)
24 }

```

Рисунок 106 – Код программы 2 задания

```

PS D:\fgit\GO_1\Primers> go run 2.go
Введите длину первого катета:
45
Введите длину второго катета:
78
Периметр прямоугольного треугольника: 213.05

```

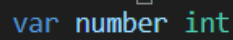
Рисунок 107 – Результат работы программы 1 задания

8. Используйте для каждой задачи (проекта) отдельную папку.
9. Добавьте отчет по лабораторной работе в формате PDF в папку doc репозитория. Зафиксируйте изменения.
10. Выполните слияние ветки для разработки с веткой main/master.
11. Отправьте сделанные изменения на сервер GitHub (или иной используемый сервис).
12. Отправьте адрес репозитория GitHub (или иного используемого сервиса) на электронный адрес преподавателя.

Вывод: в ходе выполнения лабораторной работы были исследованы назначения и способы установки Go, исследование типов данных, констант и арифметических операции языка программирования Go.

Ответы на контрольные вопросы

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



```
var number int
```

Рисунок 108 – Объявление переменной

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

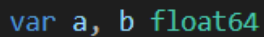
Присваивается нулевое значение, которое для типа `int` равно 0.

3. Как изменить значение существующей переменной в Go?

Для изменения значения существующей переменной в Go используется оператор присваивания `=`.

4. Что такое множественное объявление переменных в Go?

Объявление нескольких переменных в одной строке.

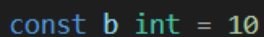


```
var a, b float64
```

Рисунок 109 – Объявление переменной

5. Как объявить константу в Go?

Используется ключевое слово `const`.



```
const b int = 10
```

Рисунок 110 – Константа

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

Нет, значение константы нельзя изменить после ее объявления в Go.

7. Какие арифметические операторы поддерживаются в Go?

В Go поддерживаются следующие арифметические операторы:

Сложение (+) – для сложения двух значений.

Вычитание (-) – для вычитания одного значения из другого.

Умножение (*) – для умножения двух значений вместе.

Деление (/) – для деления одного значения на другое.

Модуль (%) – для вычисления остатка от операции деления.

Инкремент (++) – для увеличения значения переменной на единицу.

Уменьшение (--) – для уменьшения значения переменной на единицу.

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

Оператор % используется для выполнения операции остатка в Go.

9. Какой результат выражения 5 / 2 в Go?

Целое число 2, так как в Go деление целых чисел возвращает целое число, а дробная часть отбрасывается.

10. Как считать строку с консоли в Go?

Используется пакет fmt и функция Scan().

11. Как считать целое число с консоли в Go?

Для считывания целого числа с консоли в Go также используется пакет fmt и функция Scan() вместе с оператором &.

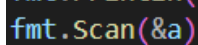
A screenshot of a code editor showing the Go function call `fmt.Scan(&a)`. The text is highlighted in a dark background with syntax coloring: `fmt` is blue, `.` is white, `Scan` is red, `(` is white, `&a` is green, and `)` is white.

Рисунок 111 – функция Scan()

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

Для обработки ошибок при считывании данных с консоли в Go можно использовать механизм обработки ошибок с помощью функции Scanln() и проверки возвращаемого значения.

13. Как вывести строку в консоль в Go?

Для вывода строки в консоль в Go используется функция Println() из пакета fmt.

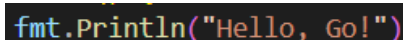
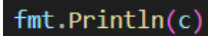
A screenshot of a code editor showing the Go function call `fmt.Println("Hello, Go!")`. The text is highlighted in a dark background with syntax coloring: `fmt` is blue, `.` is white, `Println` is red, `(` is white, `"Hello, Go!"` is green, and `)` is white.

Рисунок 112 – Операция вывода

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

Для вывода значения переменной типа `int` в консоль также используется функция `Println()`.

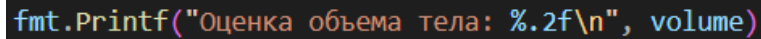


```
fmt.Println(c)
```

Рисунок 113 – Операция вывода

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

Для форматирования вывода числа с плавающей точкой в Go можно использовать функцию `Printf()` из пакета `fmt`.

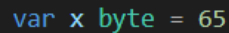


```
fmt.Printf("Оценка объема тела: %.2f\\n", volume)
```

Рисунок 114 – Операция форматирования вывода

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

В Go оператор `=` используется для присваивания значения переменной, а оператор `:=` используется для объявления и присваивания значения новой переменной.



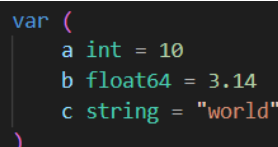
```
var x byte = 65
```

Рисунок 115 – Тип `byte`

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

В Go для представления чисел с плавающей точкой можно использовать следующие типы данных: `float32` и `float64`.

18. Как объявить и использовать несколько переменных в Go



```
var (  
    a int = 10  
    b float64 = 3.14  
    c string = "world"  
)
```

Рисунок 116 – Способ объявления переменных