

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

Кувшин Ирина Анатольевна

---

(Подпись)

**Проверил:**

Доцент, кандидат технических наук,  
доцент кафедры инфокоммуникаций  
института цифрового развития

Воронкин Роман Александрович

---

(Подпись)

Работа защищена с оценкой:

---

Ставрополь, 2024

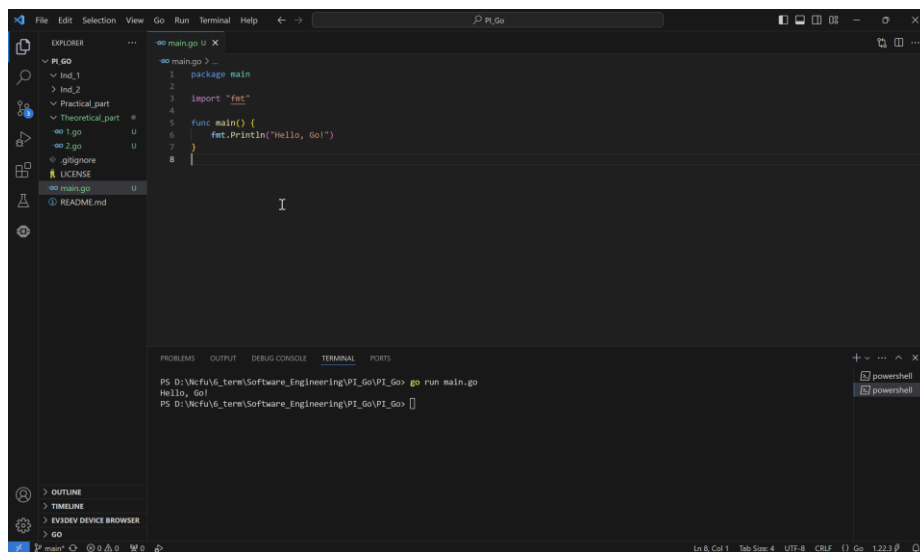
## Цель:

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

## Ход работы:

### Проработка примеров теоретической части:

Вывод в консоль «Hello world!» (рис. 1). В Go fmt - это пакет, который предоставляет функциональность для форматирования и вывода данных в стандартный поток вывода (консоль) или в любой другой поток вывода.



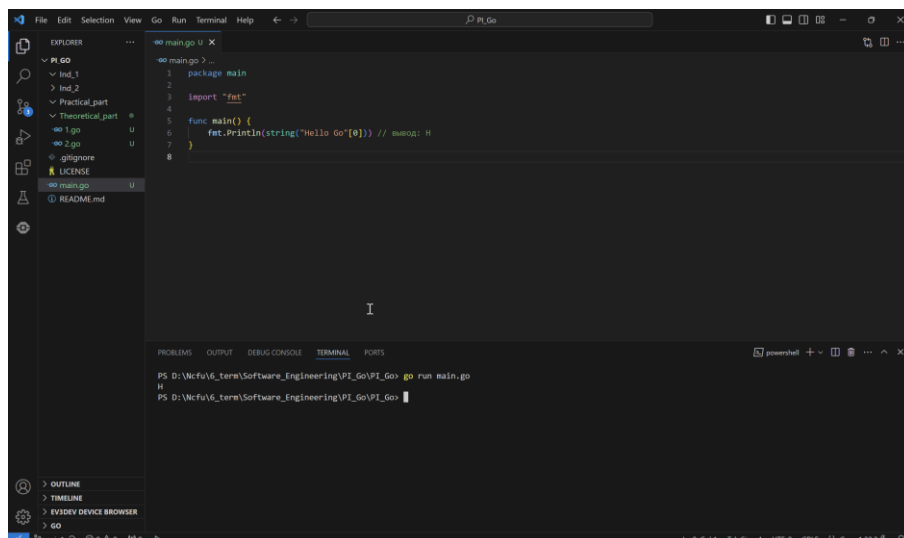
The screenshot shows the Visual Studio Code interface with a Go file named `main.go` open in the editor. The code is as follows:

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

The terminal at the bottom shows the command `go run main.go` being executed, resulting in the output `Hello, Go!`.

Рисунок 1 — Результат выполнения программы

Представим байты в виде строки (рис. 2):



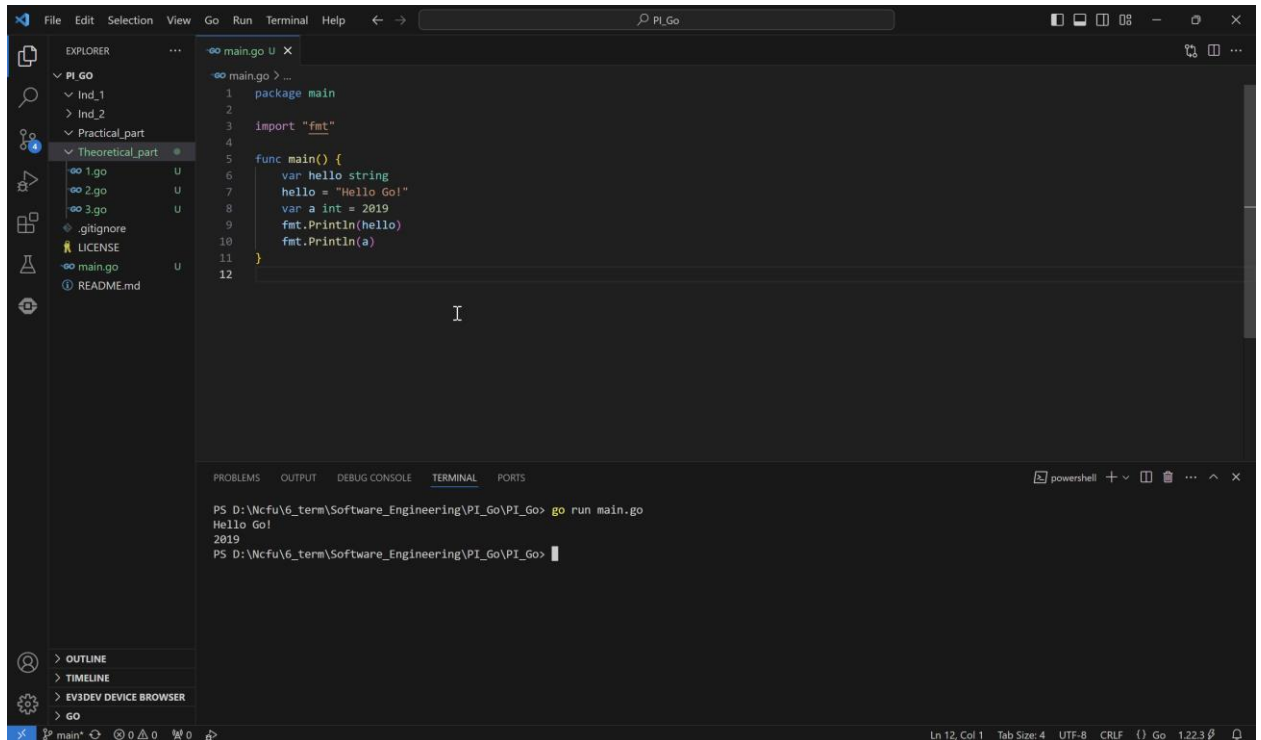
The screenshot shows the Visual Studio Code interface with a Go file named `main.go` open in the editor. The code is as follows:

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

The terminal at the bottom shows the command `go run main.go` being executed, resulting in the output `H`.

Рисунок 2 — Результат выполнения программы

Теперь мы можем объявить переменную типа `string` или `int`, присвоить ей любое значение, а затем вывести (рис. 3).



The screenshot shows the Visual Studio Code interface with a Go file named `main.go` open. The code defines a `main` package and a `main` function. Inside the function, it declares a string variable `hello` with the value "Hello Go!", an integer variable `a` with the value 2019, and then prints both using `fmt.Println`. The terminal at the bottom shows the command `go run main.go` being executed, resulting in the output "Hello Go!" followed by "2019".

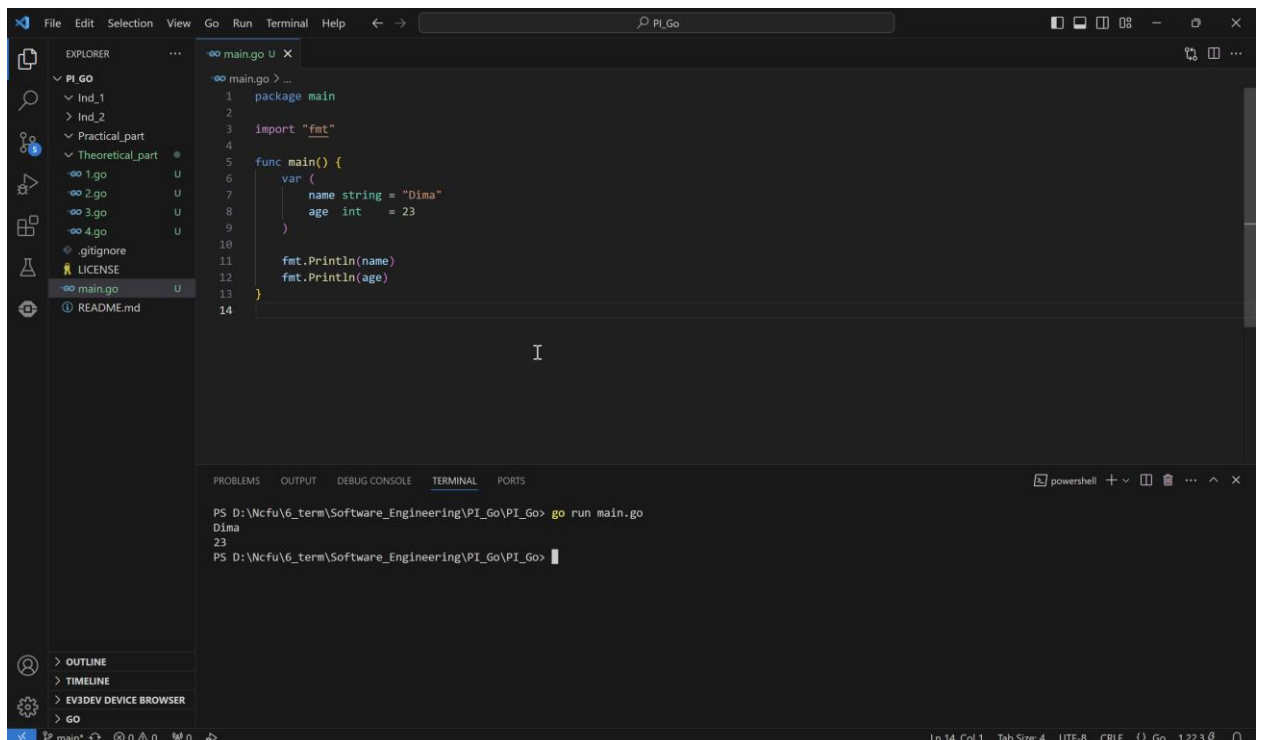
```
1 package main
2
3 import "fmt"
4
5 func main() {
6     var hello string
7     hello = "Hello Go!"
8     var a int = 2019
9     fmt.Println(hello)
10    fmt.Println(a)
11 }
12
```

Terminal output:

```
PS D:\Wcfu\6_term\Software_Engineering\PI_Go\PI_Go> go run main.go
Hello Go!
2019
PS D:\Wcfu\6_term\Software_Engineering\PI_Go\PI_Go>
```

Рисунок 3 — Результат выполнения программы

Также можно объявить сразу несколько переменных в одном блоке `var` (рис. 4):



The screenshot shows the Visual Studio Code interface with a Go file named `main.go` open. The code defines a `main` package and a `main` function. Inside the function, it declares two variables, `name` of type `string` and `age` of type `int`, with values "Dima" and 23 respectively. It then prints both variables using `fmt.Println`. The terminal at the bottom shows the command `go run main.go` being executed, resulting in the output "Dima" followed by "23".

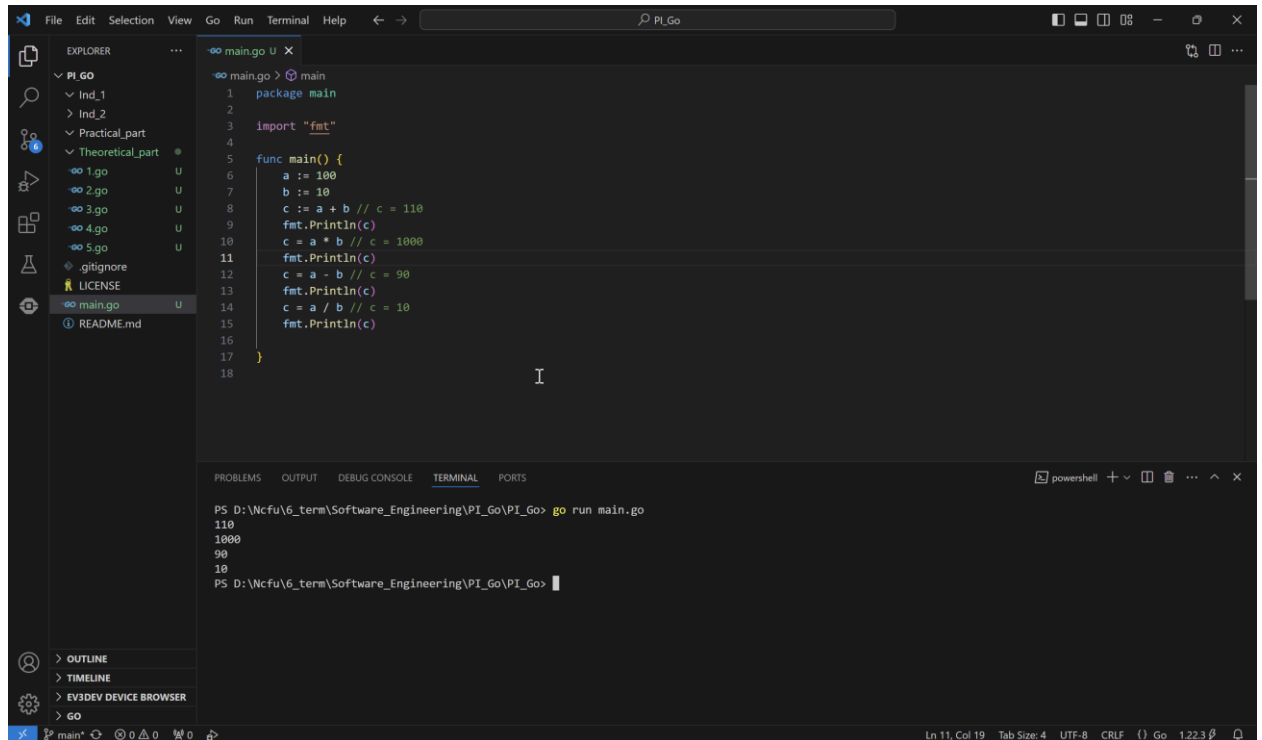
```
1 package main
2
3 import "fmt"
4
5 func main() {
6     var (
7         name string = "Dima"
8         age  int    = 23
9     )
10
11    fmt.Println(name)
12    fmt.Println(age)
13 }
14
```

Terminal output:

```
PS D:\Wcfu\6_term\Software_Engineering\PI_Go\PI_Go> go run main.go
Dima
23
PS D:\Wcfu\6_term\Software_Engineering\PI_Go\PI_Go>
```

Рисунок 4 — Результат выполнения программы

Примеры арифметических операций (рис. 5):



The screenshot shows the Visual Studio Code editor with a Go file named `main.go`. The code defines a `main` function that performs several arithmetic operations and prints the results using `fmt.Println`.

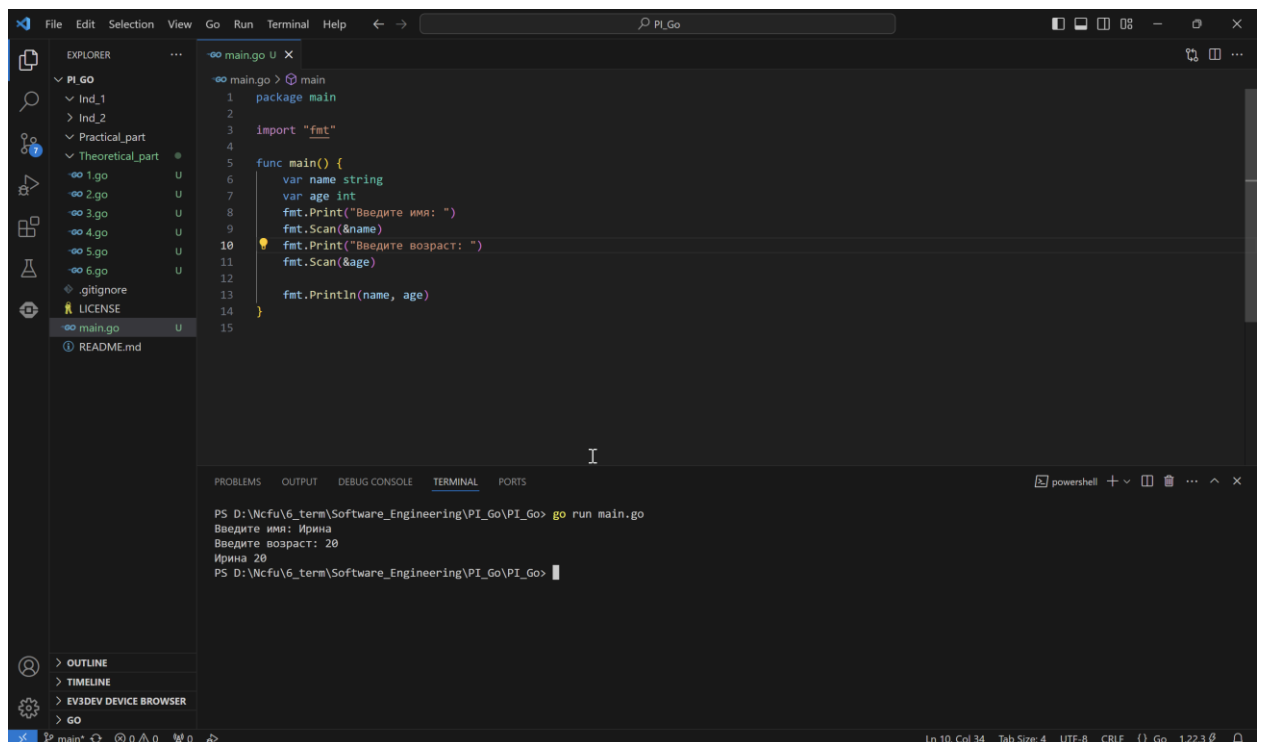
```
1 package main
2
3 import "fmt"
4
5 func main() {
6     a := 100
7     b := 10
8     c := a + b // c = 110
9     fmt.Println(c)
10    c = a * b // c = 1000
11    fmt.Println(c)
12    c = a - b // c = 90
13    fmt.Println(c)
14    c = a / b // c = 10
15    fmt.Println(c)
16 }
17
18
```

The terminal output shows the results of these operations:

```
PS D:\Ncfu\6_term\Software_Engineering\PI_Go\PI_Go> go run main.go
110
1000
90
10
PS D:\Ncfu\6_term\Software_Engineering\PI_Go\PI_Go>
```

Рисунок 5 — Результат выполнения программы

Чтение данных с консоли (рис. 6):



The screenshot shows the Visual Studio Code editor with a Go file named `main.go`. The code defines a `main` function that reads user input for a name and age, and then prints them using `fmt.Println`.

```
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 }
15
```

The terminal output shows the user input and the program's response:

```
PS D:\Ncfu\6_term\Software_Engineering\PI_Go\PI_Go> go run main.go
Введите имя: Ирина
Введите возраст: 20
Ирина 20
PS D:\Ncfu\6_term\Software_Engineering\PI_Go\PI_Go>
```

Рисунок 6 — Результат выполнения программы

Еще пример вывода, используя строки и переменные (рис. 7):

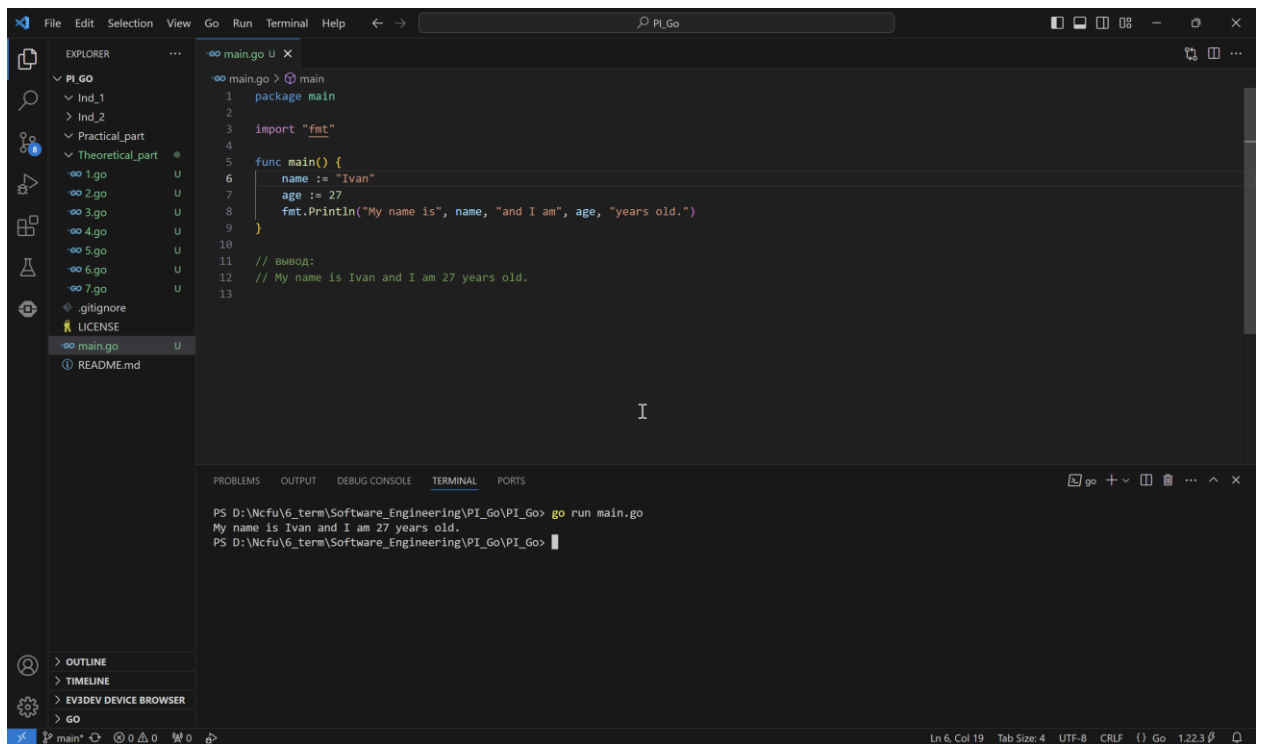


Рисунок 7 — Результат выполнения программы

Константы (рис. 8):

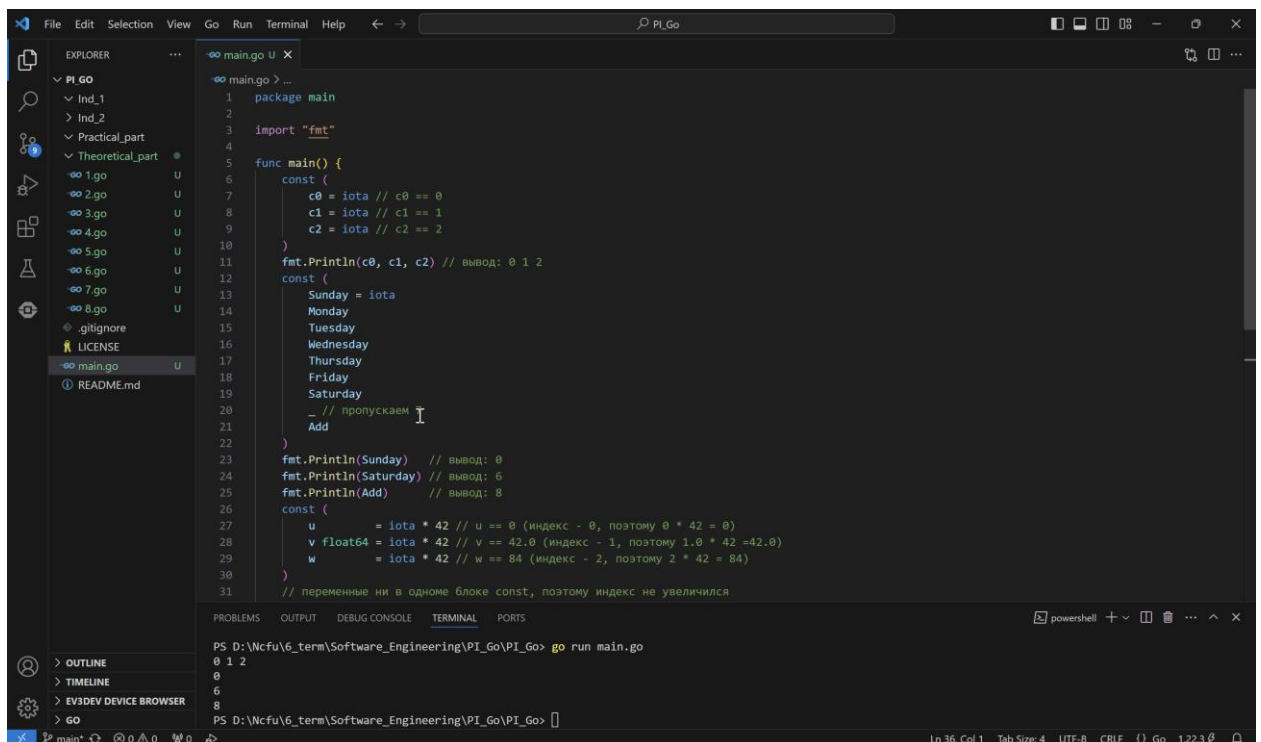


Рисунок 8 — Результат выполнения программы

## Решение задач практической части:

### 1. Задача: Напишите программу, которая выводит "I like Go!" (рис. 9)

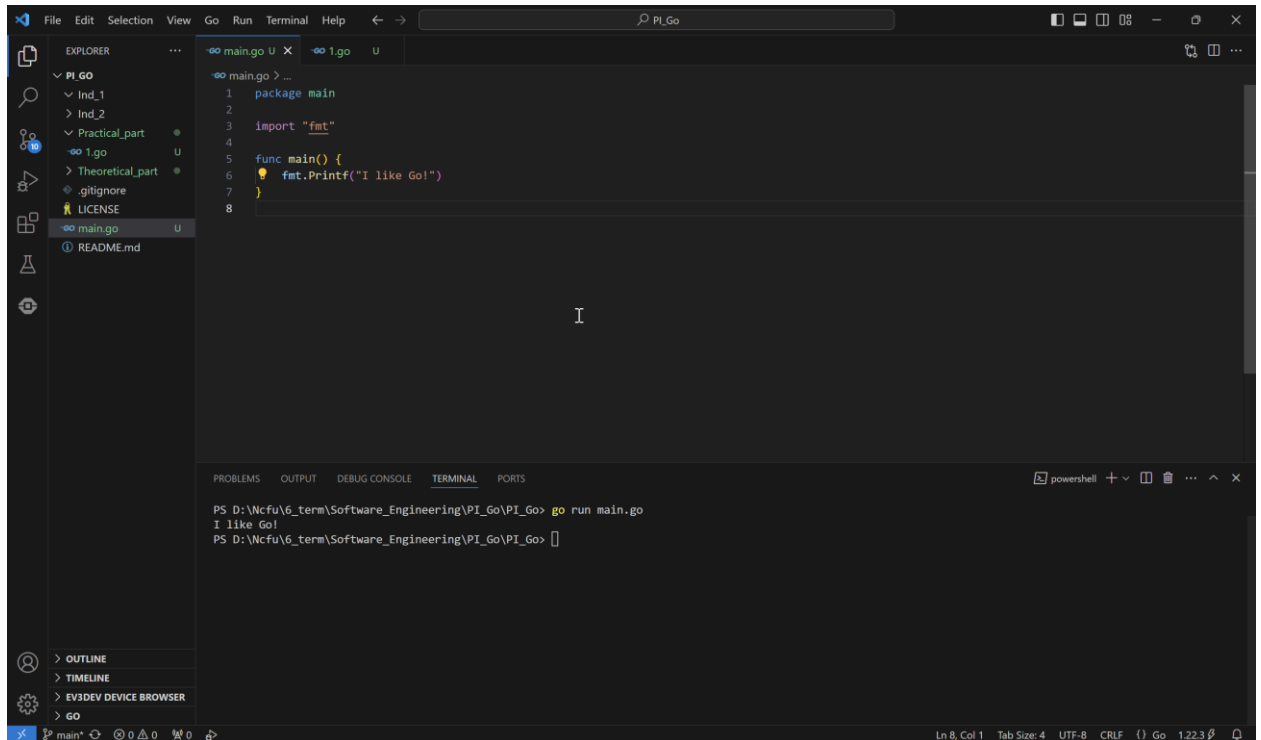


Рисунок 9 — Результат выполнения программы

### 2. Задача: Напишите программу, которая выведет "I like Go!" 3 раза (рис. 10):

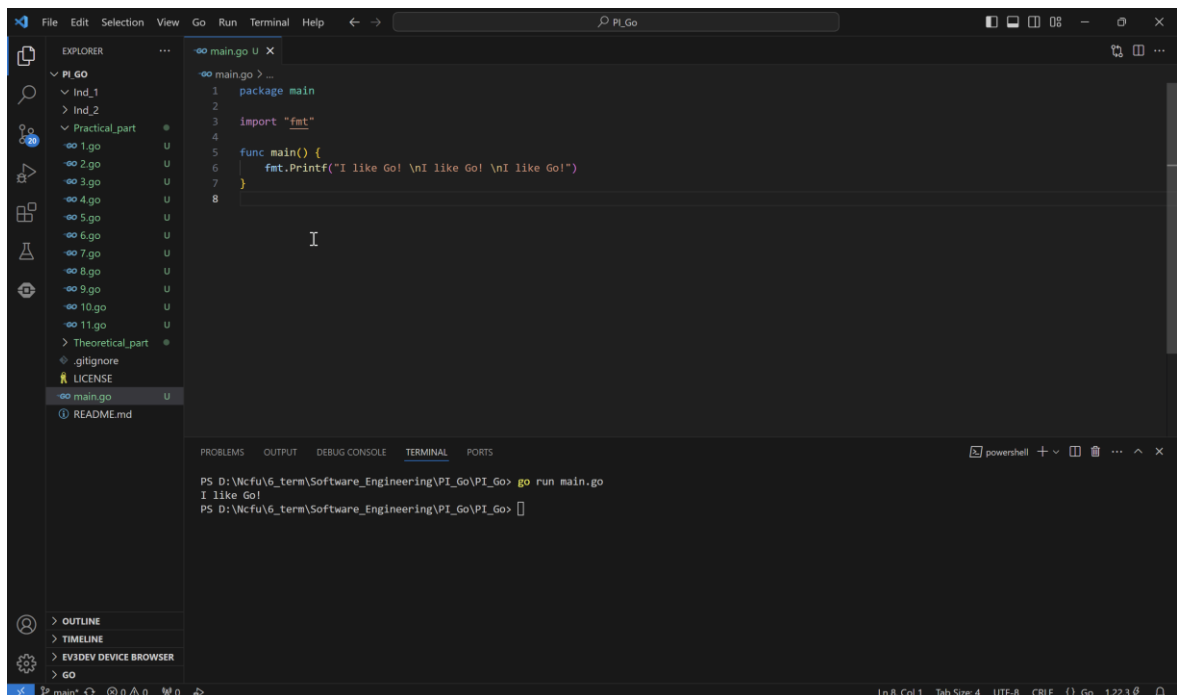


Рисунок 10 — Результат выполнения программы

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

Число умножается на 2;

Затем к числу прибавляется 100.

После этого должен быть вывод получившегося числа на экран (рис. 11):

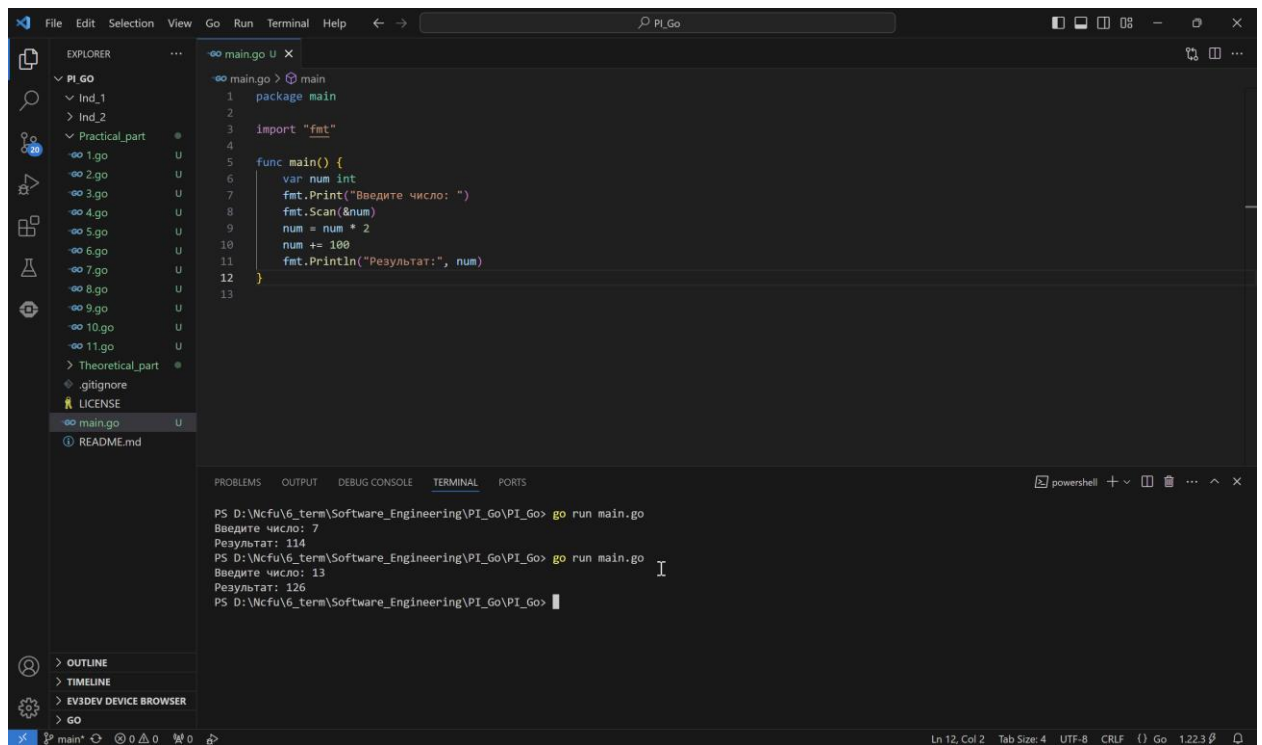


Рисунок 11 — Результат выполнения программы

4. Задача: Петя торопился в школу и неправильно написал программу, которая сначала находит квадраты двух чисел, а затем их суммирует. Исправьте его программу (рис. 12):

```
package main

import "fmt"

func main(){

    var a int

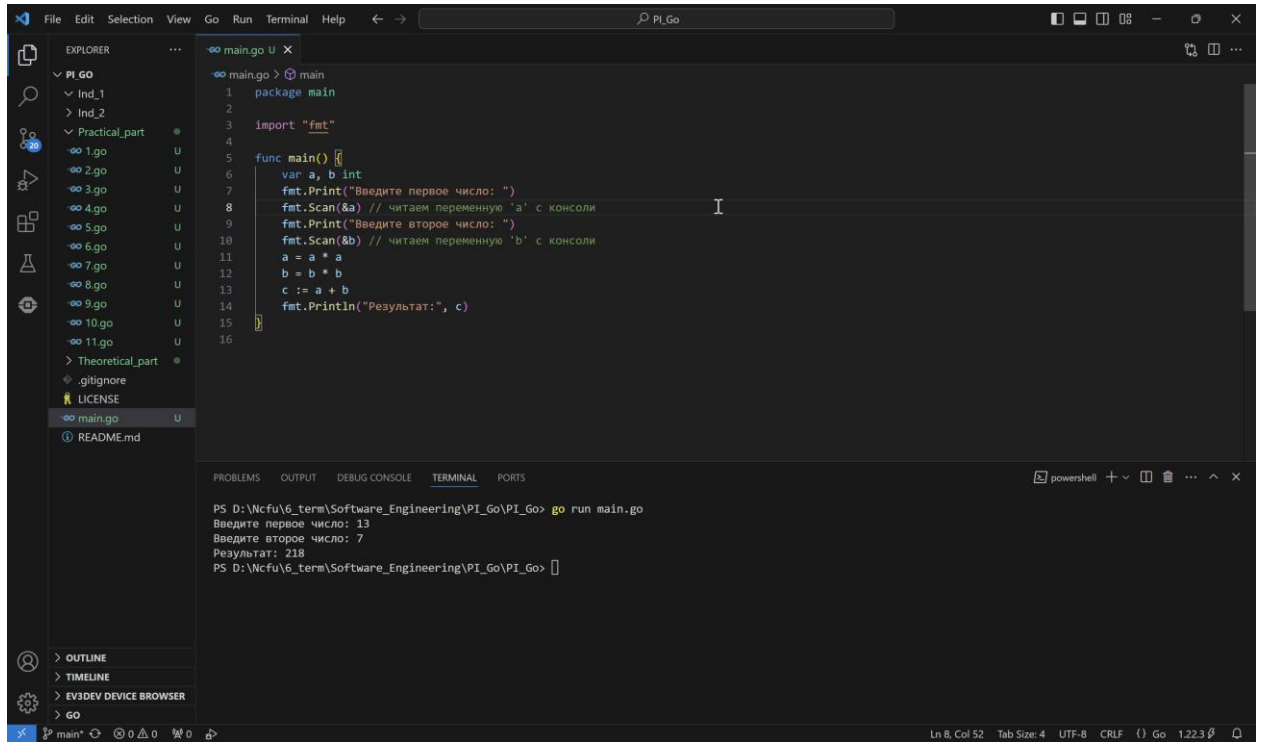
    fmt.Scan(&a) // считаем переменную 'a' с консоли
    fmt.Scan(&b) // считаем переменную 'b' с консоли

    a = a * a
```

```

b = b * 2
c = a + b
fmt.Println(c)
}

```



```

1 package main
2
3 import "fmt"
4
5 func main() {
6     var a, b int
7     fmt.Print("Введите первое число: ")
8     fmt.Scan(&a) // читаем переменную "a" с консоли
9     fmt.Print("Введите второе число: ")
10    fmt.Scan(&b) // читаем переменную "b" с консоли
11    a = a * a
12    b = b * b
13    c := a + b
14    fmt.Println("Результат:", c)
15 }

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

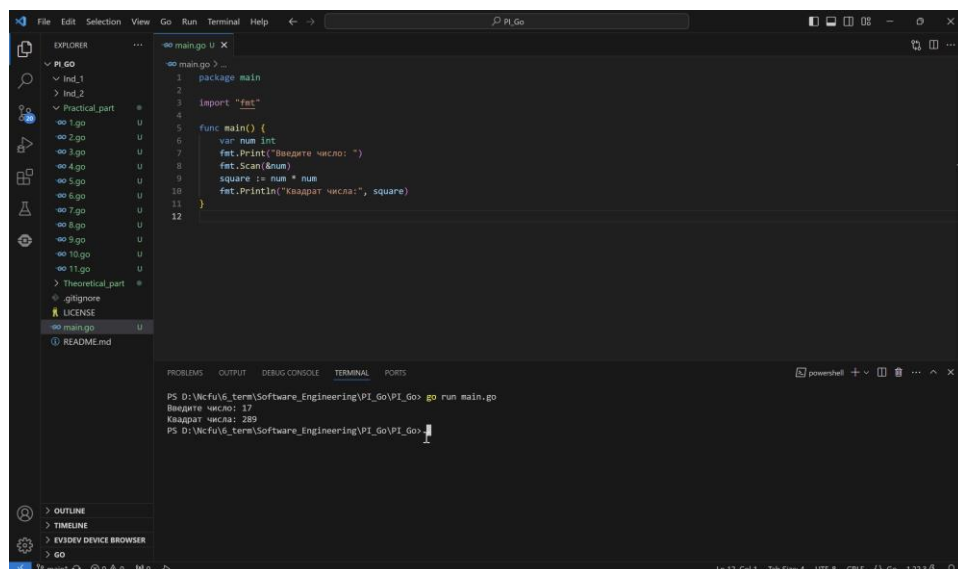
```

PS D:\McFu6_term\Software_Engineering\PI_Go\PI_Go> go run main.go
Введите первое число: 13
Введите второе число: 7
Результат: 218
PS D:\McFu6_term\Software_Engineering\PI_Go\PI_Go>

```

Рисунок 12 — Результат выполнения программы

5. Задача: По данному целому числу, найдите его квадрат (рис. 13):



```

1 package main
2
3 import "fmt"
4
5 func main() {
6     var num int
7     fmt.Print("Введите число: ")
8     fmt.Scan(&num)
9     square := num * num
10    fmt.Println("Квадрат числа:", square)
11 }
12

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```

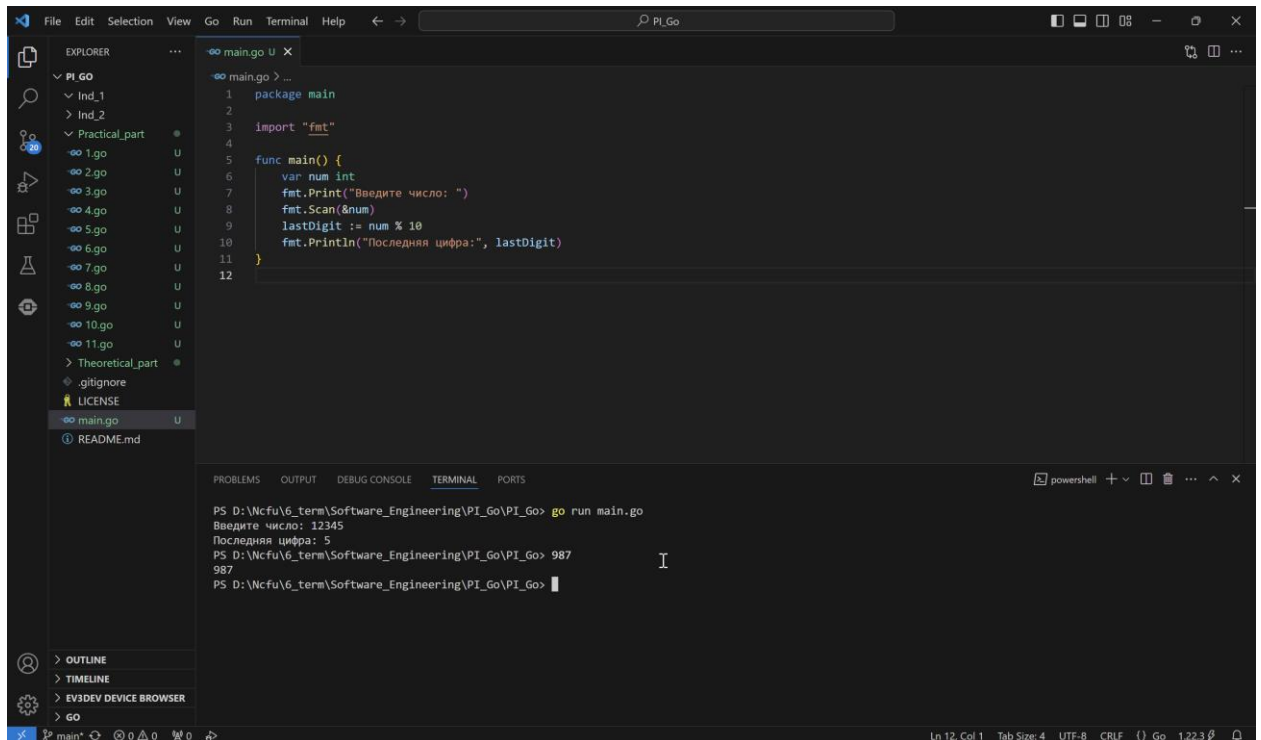
PS D:\McFu6_term\Software_Engineering\PI_Go\PI_Go> go run main.go
Введите число: 17
Квадрат числа: 289
PS D:\McFu6_term\Software_Engineering\PI_Go\PI_Go>

```

Рисунок 13 — Результат выполнения программы



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 (рис. 14):



The screenshot shows the Visual Studio Code editor with a Go file named `main.go`. The code is as follows:

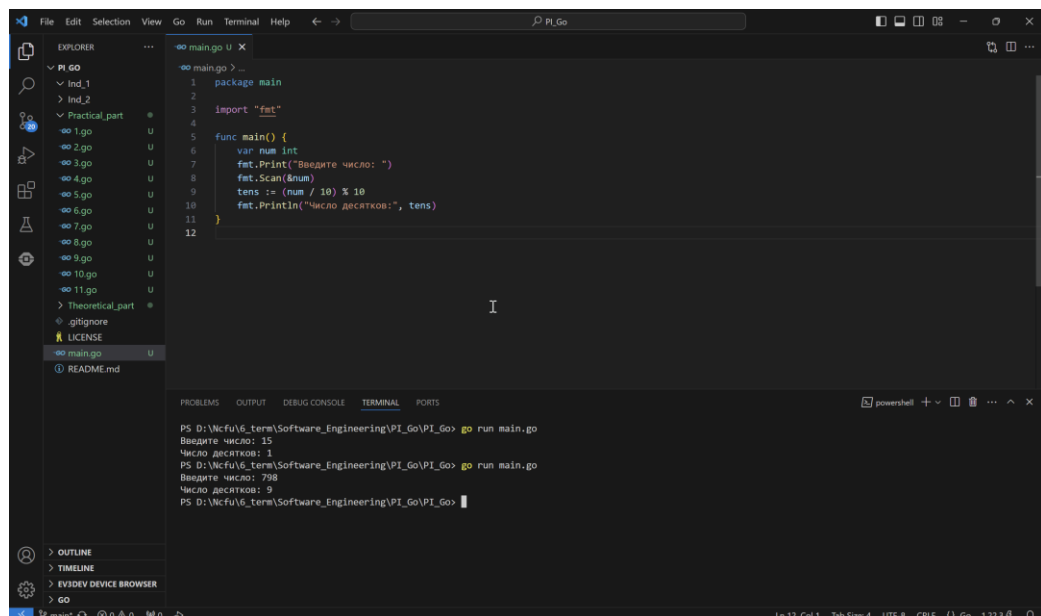
```
1 package main
2
3 import "fmt"
4
5 func main() {
6     var num int
7     fmt.Print("Введите число: ")
8     fmt.Scan(&num)
9     lastDigit := num % 10
10    fmt.Println("Последняя цифра:", lastDigit)
11 }
12
```

The terminal output shows the program being run twice. In the first run, the input is 12345 and the output is 5. In the second run, the input is 987 and the output is 7.

```
PS D:\Wcfu\6_term\Software_Engineering\PI_Go\PI_Go> go run main.go
Введите число: 12345
Последняя цифра: 5
PS D:\Wcfu\6_term\Software_Engineering\PI_Go\PI_Go> go run main.go
Введите число: 987
Последняя цифра: 7
PS D:\Wcfu\6_term\Software_Engineering\PI_Go\PI_Go>
```

Рисунок 14 — Результат выполнения программы

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 (рис. 15):



The screenshot shows the Visual Studio Code editor with a Go file named `main.go`. The code is as follows:

```
1 package main
2
3 import "fmt"
4
5 func main() {
6     var num int
7     fmt.Print("Введите число: ")
8     fmt.Scan(&num)
9     tens := (num / 10) % 10
10    fmt.Println("Число десятков:", tens)
11 }
12
```

The terminal output shows the program being run twice. In the first run, the input is 15 and the output is 1. In the second run, the input is 798 and the output is 9.

```
PS D:\Wcfu\6_term\Software_Engineering\PI_Go\PI_Go> go run main.go
Введите число: 15
Число десятков: 1
PS D:\Wcfu\6_term\Software_Engineering\PI_Go\PI_Go> go run main.go
Введите число: 798
Число десятков: 9
PS D:\Wcfu\6_term\Software_Engineering\PI_Go\PI_Go>
```

Рисунок 15 — Результат выполнения программы

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 (рис. 16):

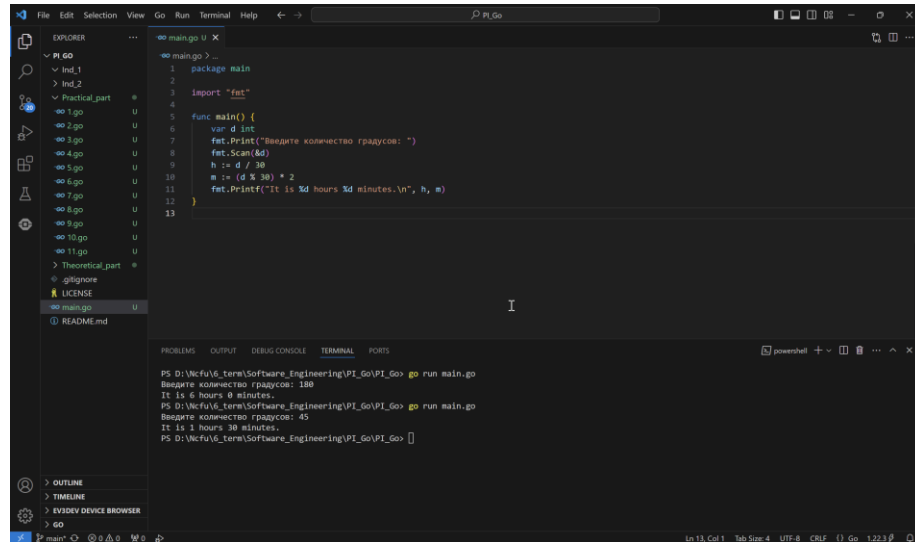


Рисунок 16 — Результат выполнения программы

9. Задача: Уберите лишние комментарии так, чтобы программа вывела число 100 (рис. 17):

```
package main

import "fmt"

func main(){

    // a:=44

    /*

    var a2 int = 10

    */

    a2 = a2 * 10

    fmt.Println(a2)

}
```

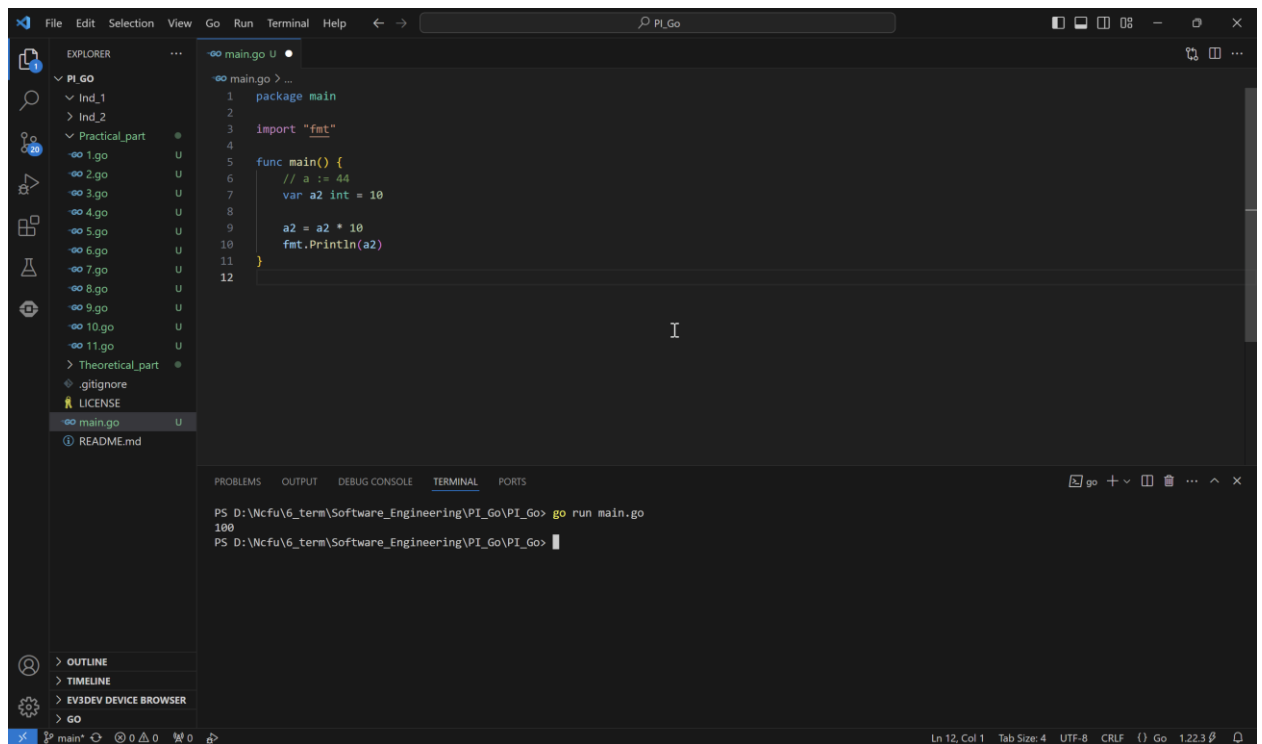


Рисунок 17 — Результат выполнения программы

10. Задача: Исправьте ошибку в программе ниже (рис. 18):

```
package main

import "fmt"

func main(){
    var a int = 8

    const b int = 10

    a = a + b

    b = b + a

    fmt.Println(a)
}
```

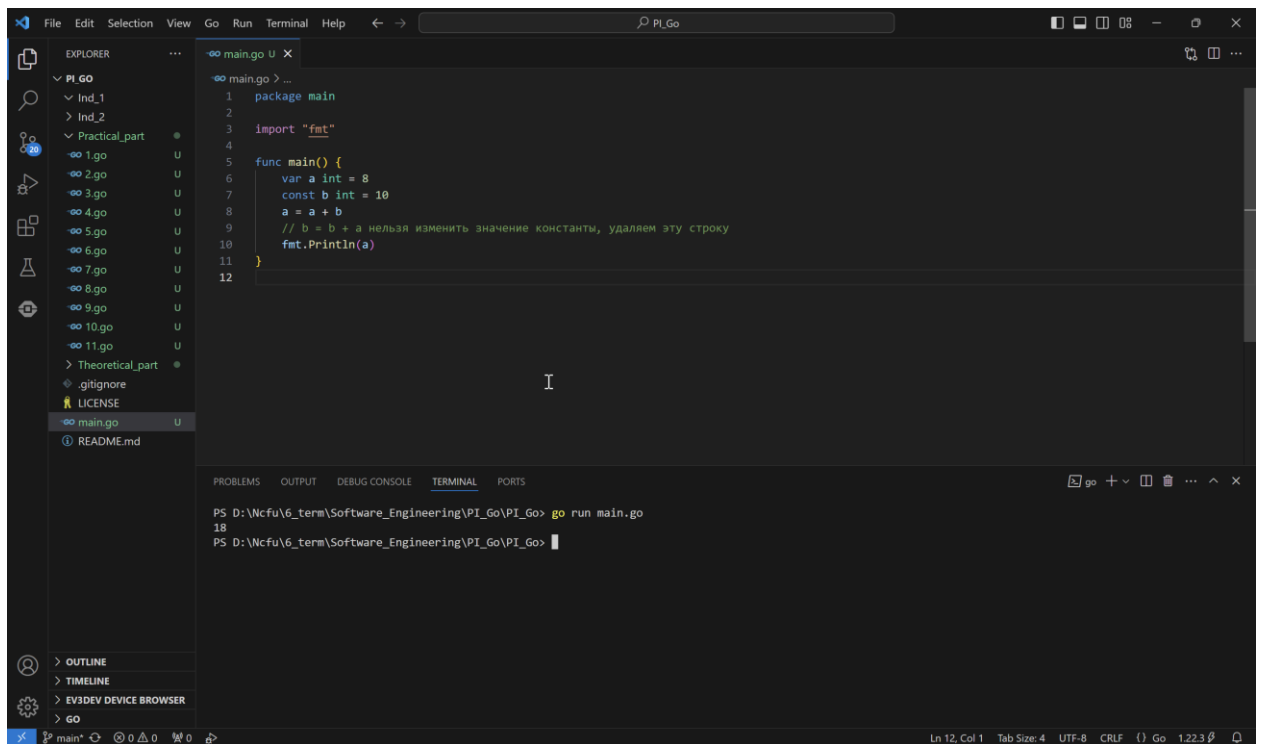


Рисунок 18 — Результат выполнения программы

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  $Ox$  (рис. 19):

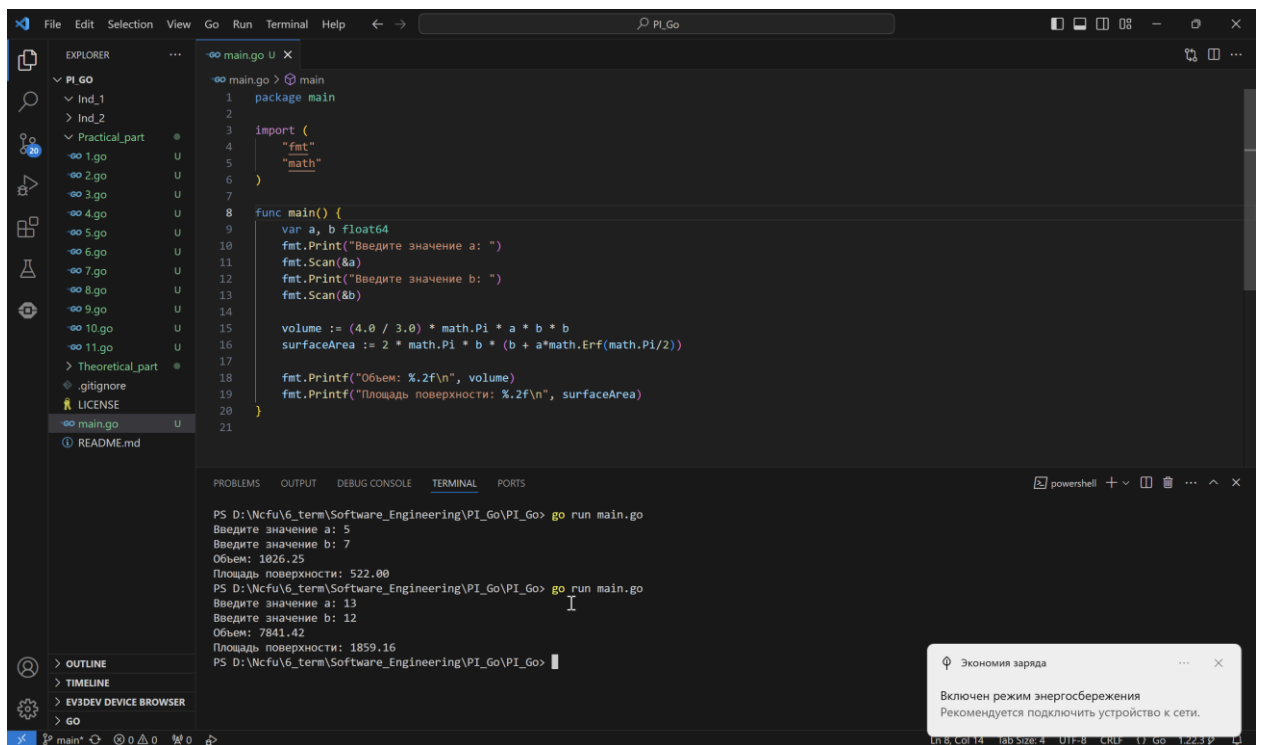
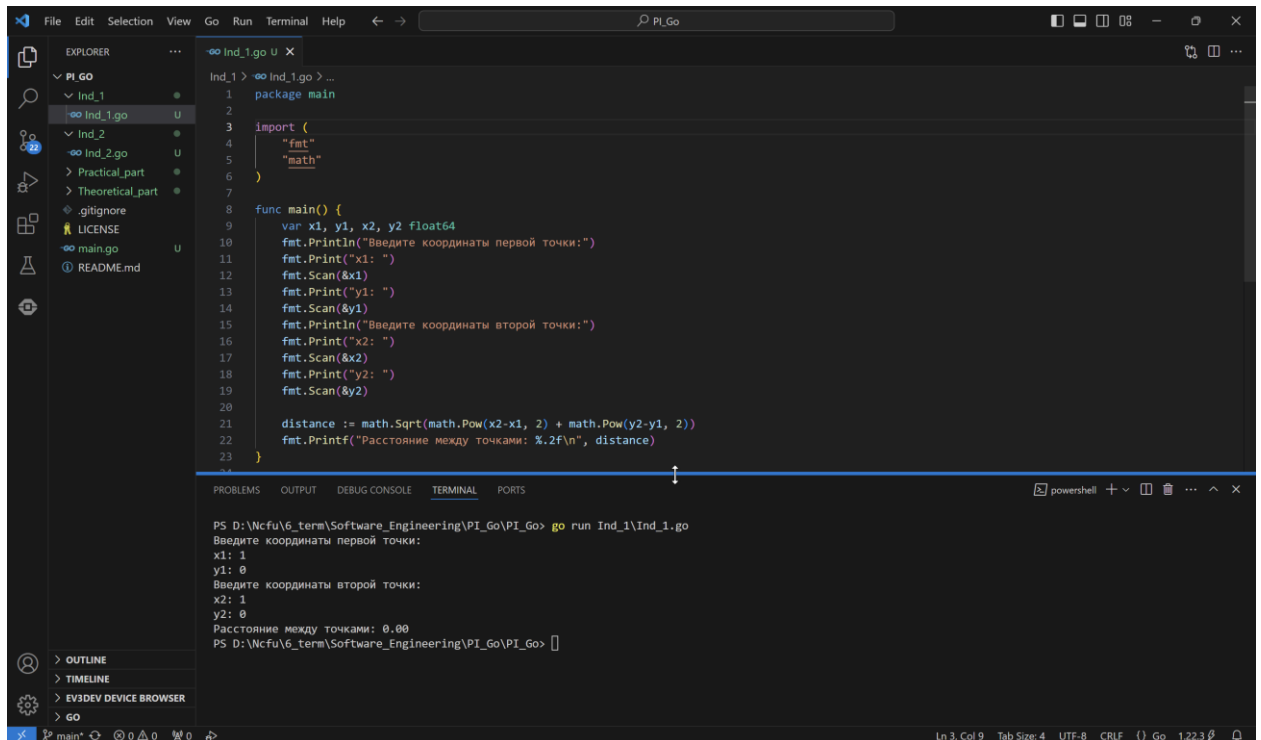


Рисунок 19 — Результат выполнения программы

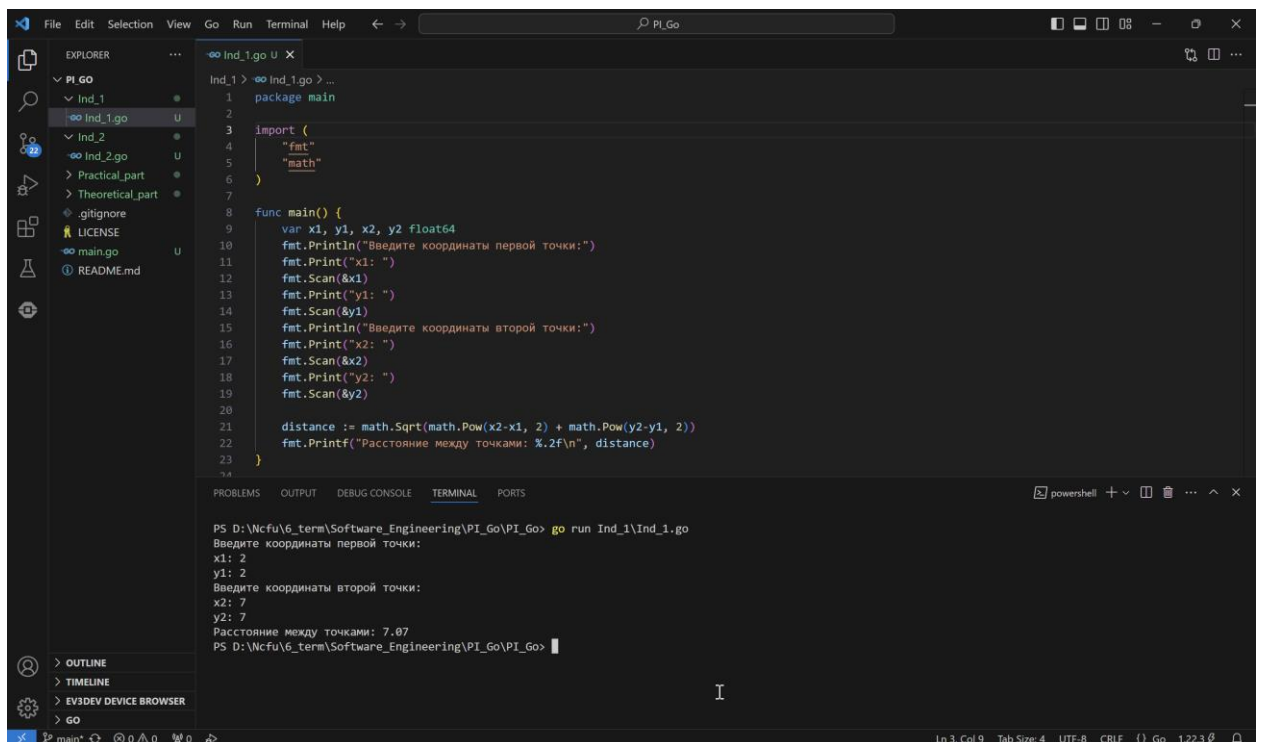
## Индивидуальное задание 1:

11. Длина отрезка на плоскости: Задайте координаты двух точек на плоскости. Рассчитайте и выведите расстояние между этими точками (рис. 20).



```
1 package main
2
3 import (
4     "fmt"
5     "math"
6 )
7
8 func main() {
9     var x1, y1, x2, y2 float64
10    fmt.Println("Введите координаты первой точки:")
11    fmt.Print("x1: ")
12    fmt.Scan(&x1)
13    fmt.Print("y1: ")
14    fmt.Scan(&y1)
15    fmt.Println("Введите координаты второй точки:")
16    fmt.Print("x2: ")
17    fmt.Scan(&x2)
18    fmt.Print("y2: ")
19    fmt.Scan(&y2)
20
21    distance := math.Sqrt(math.Pow(x2-x1, 2) + math.Pow(y2-y1, 2))
22    fmt.Printf("Расстояние между точками: %.2f\n", distance)
23 }
```

PS D:\Ncfu\6\_term\Software\_Engineering\PI\_Go\PI\_Go> go run Ind\_1\Ind\_1.go  
Введите координаты первой точки:  
x1: 1  
y1: 0  
Введите координаты второй точки:  
x2: 1  
y2: 0  
Расстояние между точками: 0.00  
PS D:\Ncfu\6\_term\Software\_Engineering\PI\_Go\PI\_Go>



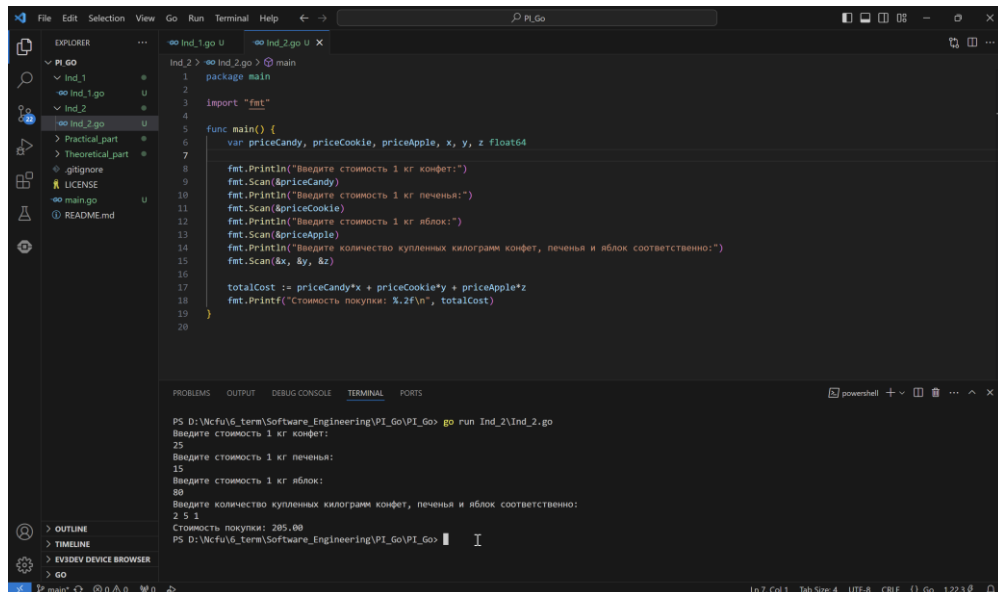
```
1 package main
2
3 import (
4     "fmt"
5     "math"
6 )
7
8 func main() {
9     var x1, y1, x2, y2 float64
10    fmt.Println("Введите координаты первой точки:")
11    fmt.Print("x1: ")
12    fmt.Scan(&x1)
13    fmt.Print("y1: ")
14    fmt.Scan(&y1)
15    fmt.Println("Введите координаты второй точки:")
16    fmt.Print("x2: ")
17    fmt.Scan(&x2)
18    fmt.Print("y2: ")
19    fmt.Scan(&y2)
20
21    distance := math.Sqrt(math.Pow(x2-x1, 2) + math.Pow(y2-y1, 2))
22    fmt.Printf("Расстояние между точками: %.2f\n", distance)
23 }
```

PS D:\Ncfu\6\_term\Software\_Engineering\PI\_Go\PI\_Go> go run Ind\_1\Ind\_1.go  
Введите координаты первой точки:  
x1: 2  
y1: 2  
Введите координаты второй точки:  
x2: 7  
y2: 7  
Расстояние между точками: 7.07  
PS D:\Ncfu\6\_term\Software\_Engineering\PI\_Go\PI\_Go>

Рисунок 20 — Результат выполнения программы

## Индивидуальное задание 2:

11. Известна стоимость 1 кг конфет, печенья и яблок. Найти стоимость всей покупки, если купили x кг конфет, y кг печенья и z кг яблок (рис. 21).



```
1 package main
2
3 import "fmt"
4
5 func main() {
6     var priceCandy, priceCookie, priceApple, x, y, z float64
7
8     fmt.Println("Введите стоимость 1 кг конфет:")
9     fmt.Scan(&priceCandy)
10    fmt.Println("Введите стоимость 1 кг печенья:")
11    fmt.Scan(&priceCookie)
12    fmt.Println("Введите стоимость 1 кг яблок:")
13    fmt.Scan(&priceApple)
14    fmt.Println("Введите количество купленных килограмм конфет, печенья и яблок соответственно:")
15    fmt.Scan(&x, &y, &z)
16
17    totalCost := priceCandy*x + priceCookie*y + priceApple*z
18    fmt.Printf("Стоимость покупки: %.2f\n", totalCost)
19 }
20
```

PS D:\McFul6\_term\Software\_Engineering\PI\_Go\PI\_Go> go run Ind\_2\Ind\_2.go

Введите стоимость 1 кг конфет:  
25

Введите стоимость 1 кг печенья:  
15

Введите стоимость 1 кг яблок:  
80

Введите количество купленных килограмм конфет, печенья и яблок соответственно:  
2 5 1

Стоимость покупки: 285.00

PS D:\McFul6\_term\Software\_Engineering\PI\_Go\PI\_Go>

Рисунок 21 — Результат выполнения программы

## **Вывод:**

В данной лабораторной работе были исследованы основные аспекты языка программирования Go, включая его назначение, способы установки, базовые типы данных, константы и арифметические операции. Был создан и настроен репозиторий на GitHub. Были проработаны теоретические примеры и решены практические задачи, а также выполнено индивидуальное задание. Отчет по выполненной работе был оформлен и добавлен в репозиторий.

## **Контрольные вопросы:**

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

`var x int`

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

Значение по умолчанию для типа `int` в Go — это 0.

3. Как изменить значение существующей переменной в Go?

`x = 10 // Присваивание нового значения переменной`

4. Что такое множественное объявление переменных в Go?

Множественное объявление переменных позволяет объявить несколько переменных в одной строке: `var x, y, z int` (`var a, b, c int = 1, 2, 3` // с присвоением)

5. Как объявить константу в Go?

```
const pi = 3.14
```

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

Нет, значение константы нельзя изменить после её объявления.

7. Какие арифметические операторы поддерживаются в Go?

+ (сложение)

- (вычитание)

\* (умножение)

/ (деление)

% (остаток от деления)

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

Оператор %.

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

Так как оба операнда — целые числа, результатом будет целое число: 2.

10. Как считать строку с консоли в Go?

```
var input string
fmt.Scanln(&input)
```

11. Как считать целое число с консоли в Go?

```
var number int
fmt.Scanln(&number)
```

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

```
var number int
_, err := fmt.Scanln(&number)
if err != nil {
    fmt.Println("Ошибка при считывании:", err)
```

```
}
```

13. Как вывести строку в консоль в Go?

```
fmt.Println("Hello, World!")
```

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

```
x := 10
```

```
fmt.Println(x)
```

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

```
y := 3.14159
```

```
fmt.Printf("%.2f\n", y) // Вывод с двумя знаками после точки
```

16. Как объявить переменную типа `byte` и присвоить ей значение 65?

```
var b byte = 65
```

17. Чем отличается оператор `:=` от оператора `=` в Go?

`:=` используется для объявления и инициализации переменной в одном выражении: `x := 10`

`=` используется для присвоения значения уже объявленной переменной:

```
var y int
```

```
y = 20
```

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

```
float32
```

```
float64
```

19. Как объявить и использовать несколько переменных в Go

```
var a, b, c int = 1, 2, 3
```

```
fmt.Println(a, b, c)
```

```
var d, e, f = 4.5, "hello", true
```

```
fmt.Println(d, e, f)
```

```
g, h, i := 6, "world", false
```

```
fmt.Println(g, h, i)
```