

**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

Зиберов Александр

« » февраля 2024 г.

Подпись студента _____

Работа защищена

« » _____ 20__ г.

Проверил Воронкин Р.А. _____
(подпись)

Ставрополь, 2024

Цель работы:

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

Выполнение работы:

Создан общедоступный репозиторий на GitHub, рисунок 1.

Ссылка: https://github.com/afk552/pi_Lab1

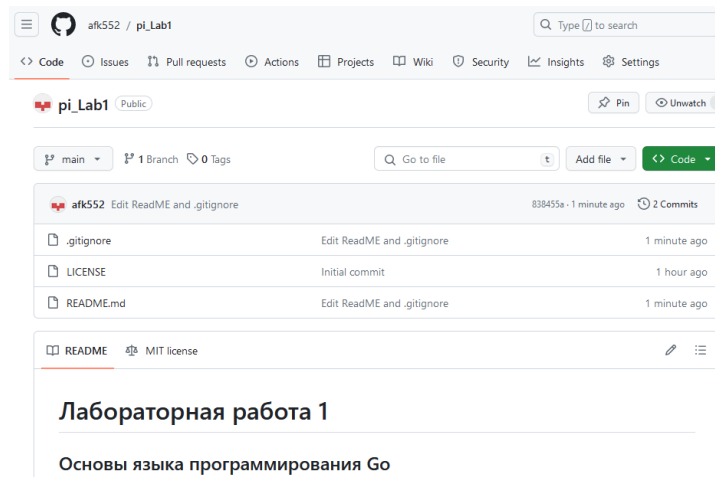


Рисунок 1 – Удаленный репозиторий на GitHub

Дополните файл .gitignore необходимыми правилами для работы с IDE PyCharm, рисунок 2.

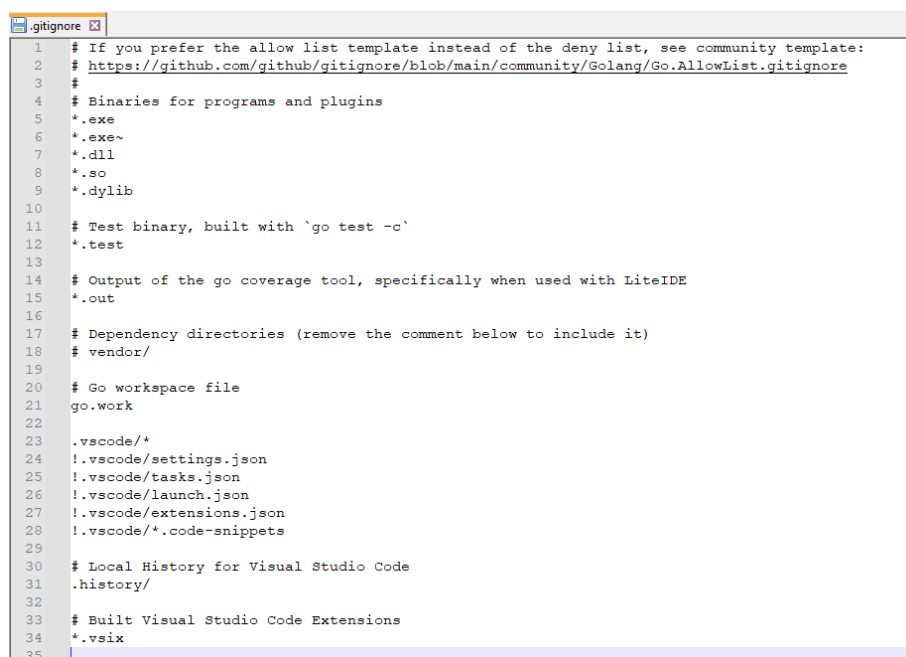


Рисунок 2 – Дополненный файл .gitignore

Настройка IDE для Golang.

1. Установка компилятора языка GO

Установим компилятор языка Go при помощи установщика с официального сайта go.dev (рисунок 3).



Рисунок 3 – Установщик компилятора языка Go

2. Настройка IDE Visual Studio Code (VS Code)

Установим расширение для языка Go, рисунок 4.

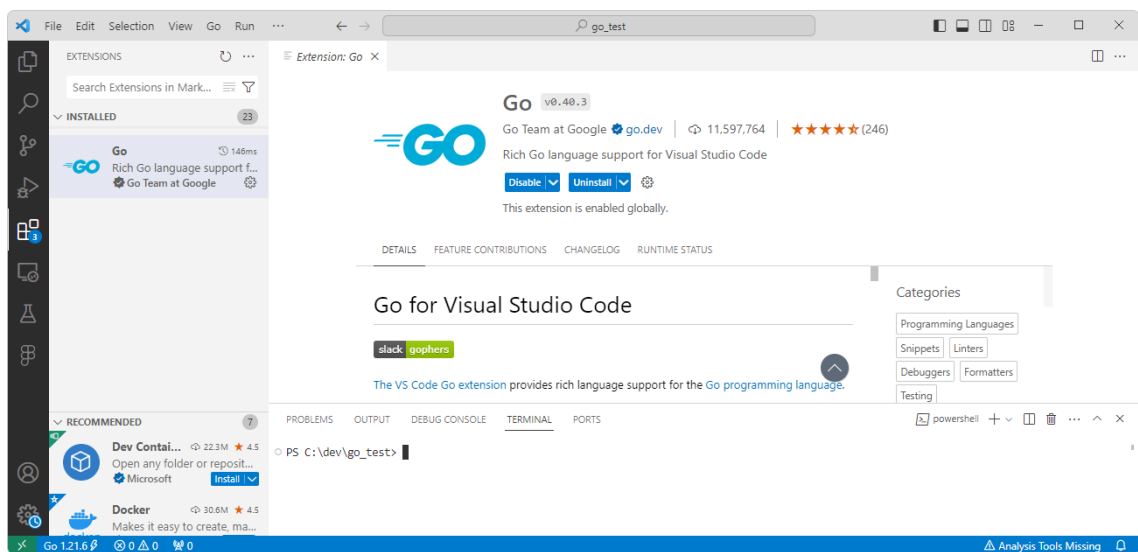


Рисунок 4 – Страница расширения Go для VS Code

Добавим опцию "console": "integratedTerminal" в конфигурацию проекта для удобного запуска программы при помощи PowerShell (в окне Terminal в VS Code) как на рисунке 5.

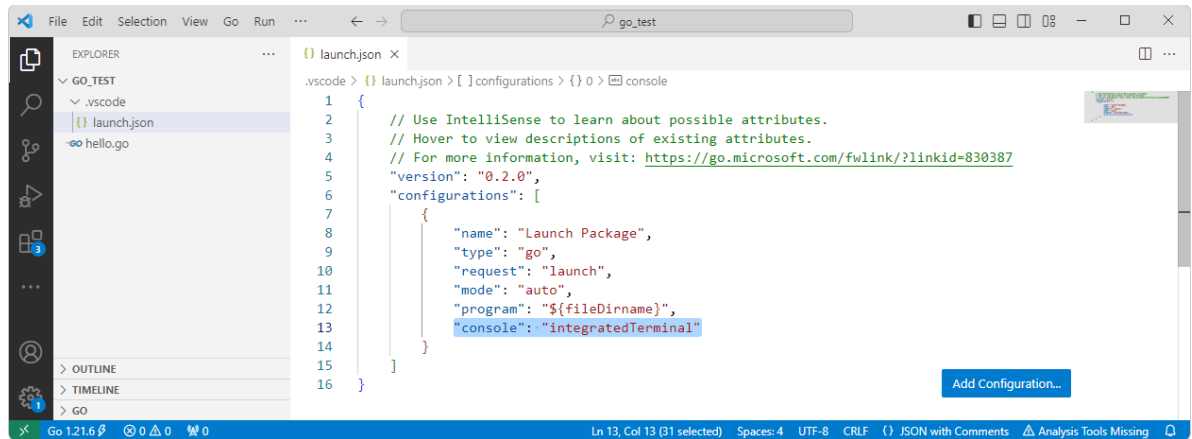


Рисунок 5 – Настройка файла конфигурации отладчика launch.json

Выполнение примеров

Пример 1. Первая программа «Hello, Go!»



Рисунок 6 – Результат выполнения программы примера 1

Пример 2. Попытка вывода символа строки в Go (не выведет, так как байты).

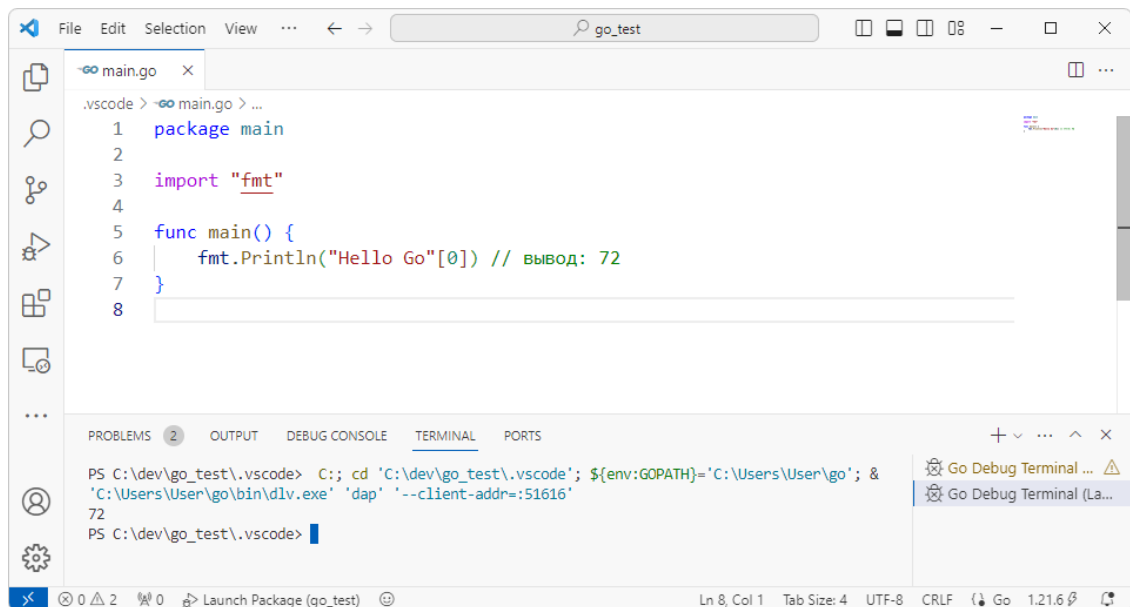


Рисунок 7 – Результат выполнения программы примера 2

Пример 3. Правильный вывод символа строки в Go

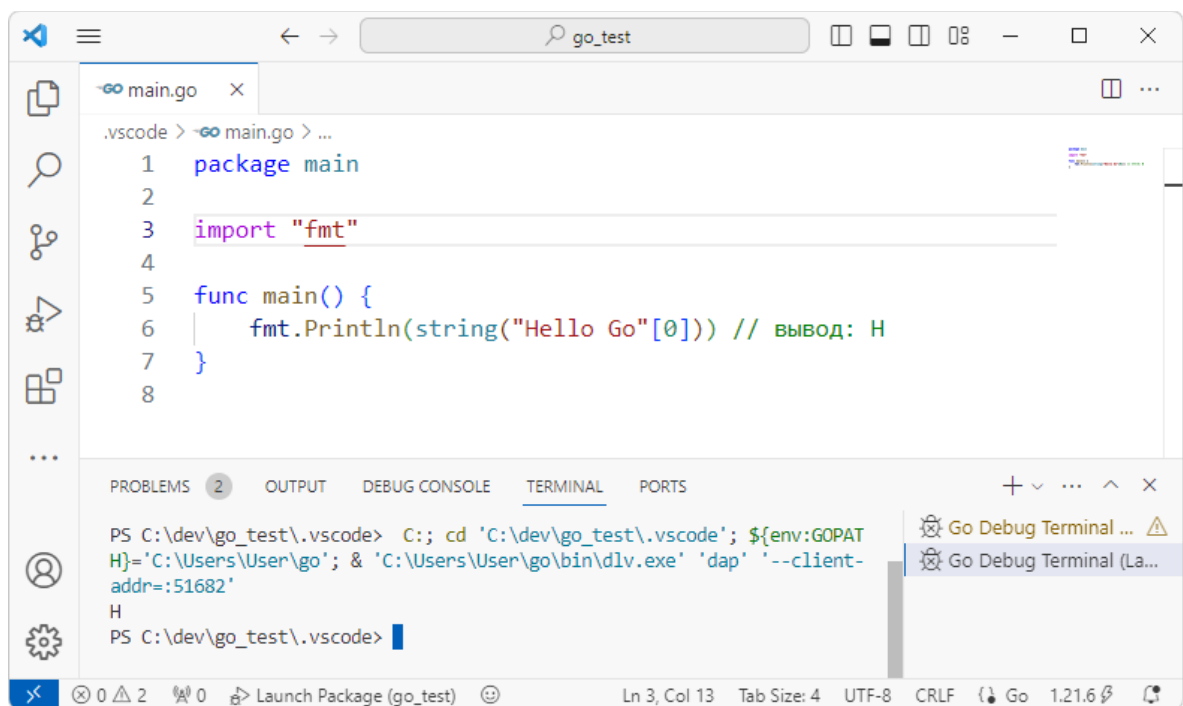
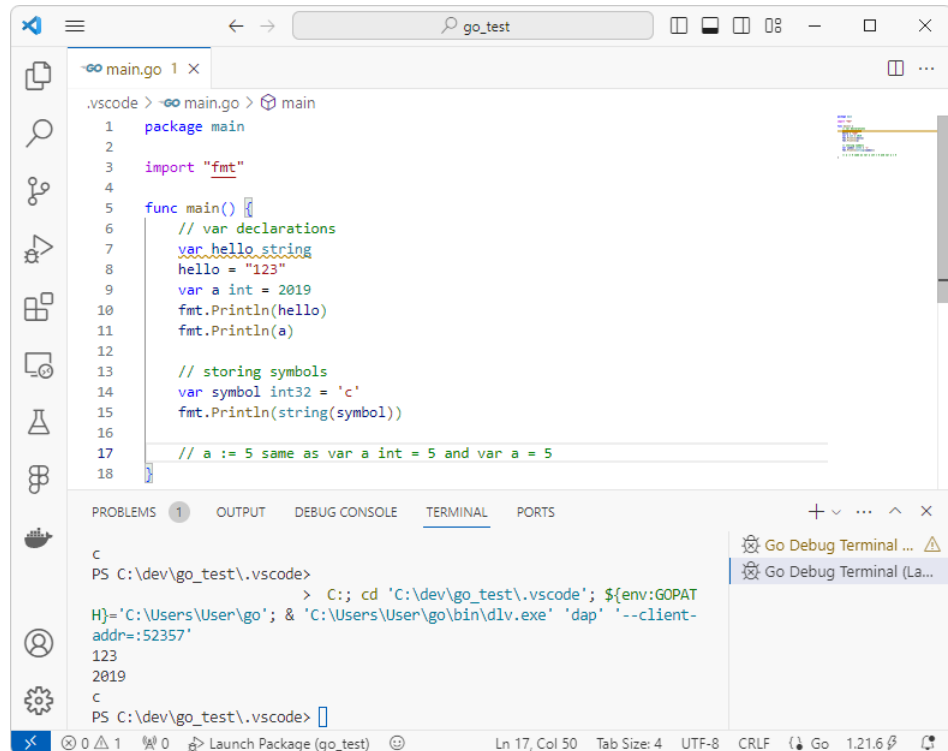


Рисунок 8 – Результат выполнения программы примера 3

Пример 4. Объявление и вывод переменных



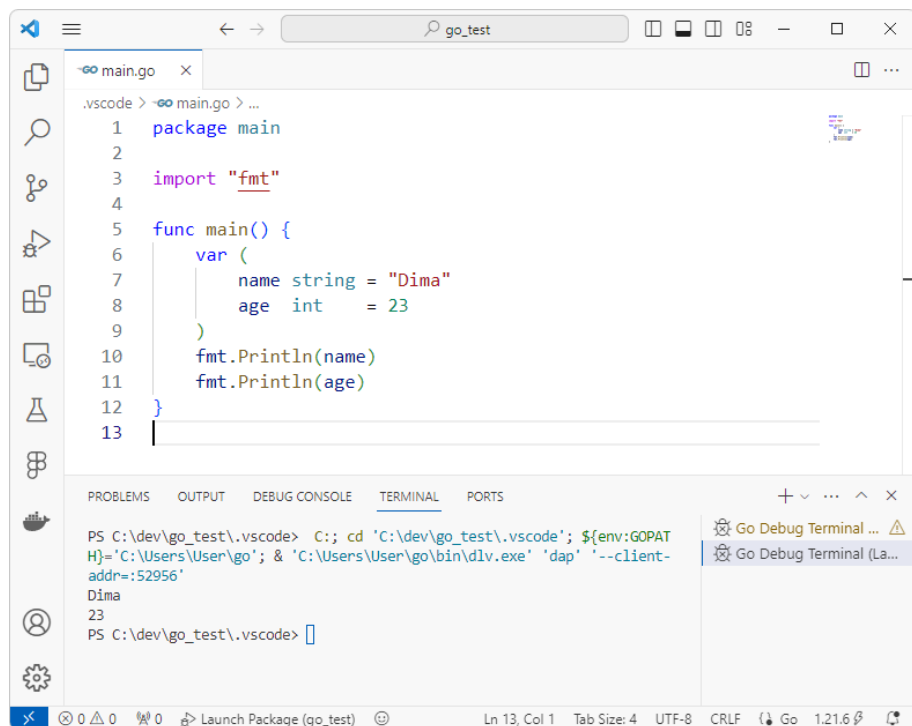
```
main.go 1 x
.vscd > -o main.go > main
1 package main
2
3 import "fmt"
4
5 func main() {
6     // var declarations
7     var hello string
8     hello = "123"
9     var a int = 2019
10    fmt.Println(hello)
11    fmt.Println(a)
12
13    // storing symbols
14    var symbol int32 = 'c'
15    fmt.Println(string(symbol))
16
17    // a := 5 same as var a int = 5 and var a = 5
18
19 }
20
```

PROBLEMS 1 OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS C:\dev\go_test\.vscd>
PS C:\dev\go_test\.vscd> C:; cd 'C:\dev\go_test\.vscd'; ${env:GOPAT
H}='C:\Users\User\go'; & 'C:\Users\User\go\bin\dlv.exe' 'dap' '--client-
addr=:52357'
123
2019
c
PS C:\dev\go_test\.vscd>
```

Рисунок 9 – Результат выполнения программы примера 4

Пример 5. Блоки переменных



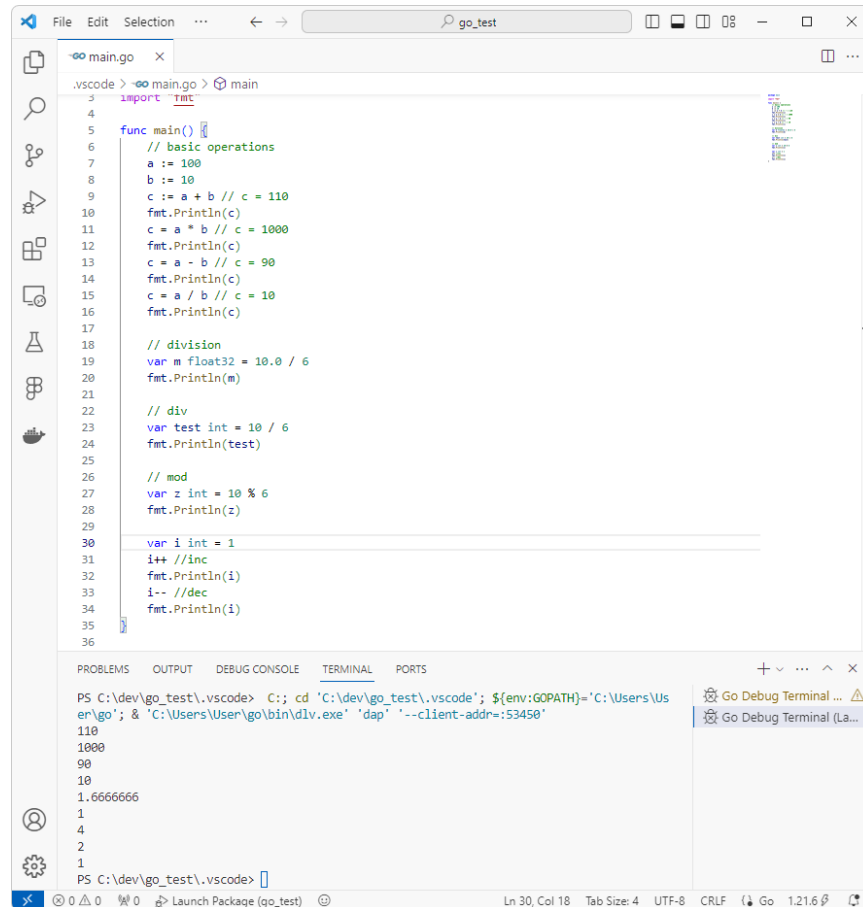
```
main.go x
.vscd > -o main.go > ...
1 package main
2
3 import "fmt"
4
5 func main() {
6     var (
7         name string = "Dima"
8         age int = 23
9     )
10    fmt.Println(name)
11    fmt.Println(age)
12 }
13
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS C:\dev\go_test\.vscd> C:; cd 'C:\dev\go_test\.vscd'; ${env:GOPAT
H}='C:\Users\User\go'; & 'C:\Users\User\go\bin\dlv.exe' 'dap' '--client-
addr=:52956'
Dima
23
PS C:\dev\go_test\.vscd>
```

Рисунок 10 – Результат выполнения программы примера 5

Пример 6. Арифметические операции



The screenshot shows the Visual Studio Code editor with a Go file named `main.go`. The code performs various arithmetic operations and prints the results. The terminal output shows the execution results.

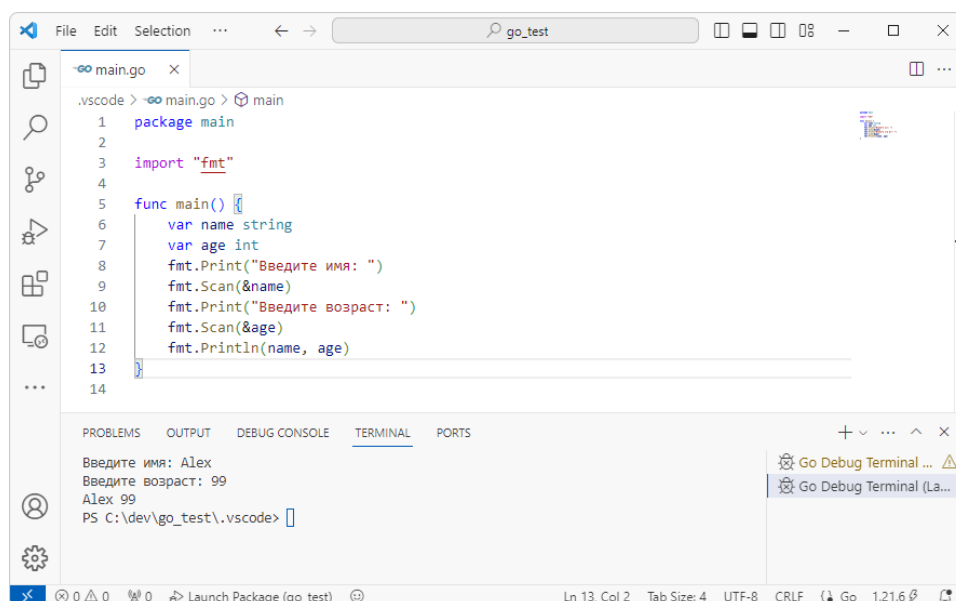
```
3 import "fmt"
4
5 func main() {
6     // basic operations
7     a := 100
8     b := 10
9     c := a + b // c = 110
10    fmt.Println(c)
11    c = a * b // c = 1000
12    fmt.Println(c)
13    c = a - b // c = 90
14    fmt.Println(c)
15    c = a / b // c = 10
16    fmt.Println(c)
17
18    // division
19    var m float32 = 10.0 / 6
20    fmt.Println(m)
21
22    // div
23    var test int = 10 / 6
24    fmt.Println(test)
25
26    // mod
27    var z int = 10 % 6
28    fmt.Println(z)
29
30    var i int = 1
31    i++ //inc
32    fmt.Println(i)
33    i-- //dec
34    fmt.Println(i)
35
36 }
```

Terminal Output:

```
PS C:\dev\go_test\.vscode> C:; cd 'C:\dev\go_test\.vscode'; ${env:GOPATH}='C:\Users\Us
er\go'; & 'C:\Users\User\go\bin\div.exe' 'dap' '--client-addr=:53450'
110
1000
90
10
1.6666666
1
4
2
1
PS C:\dev\go_test\.vscode>
```

Рисунок 11 – Результат выполнения программы примера 6

Пример 7. Чтение данных с консоли



The screenshot shows the Visual Studio Code editor with a Go file named `main.go`. The code prompts the user for their name and age, reads the input, and prints it back. The terminal output shows the user's input.

```
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    fmt.Println(name, age)
13
14 }
```

Terminal Output:

```
Введите имя: Alex
Введите возраст: 99
Alex 99
PS C:\dev\go_test\.vscode>
```

Рисунок 12 – Результат выполнения программы примера 7

Пример 8. Вывод данных с консоли

```
1 package main
2
3 import "fmt"
4
5 func main() {
6     fmt.Println("hello, world")
7     fmt.Print("hello, world")
8     // вывод будет в две строки:
9     // hello, world
10    // hello, world
11
12    fmt.Print("Ivan", 27) // Ivan27
13    fmt.Println("Ivan", 27) // Ivan 27
14
15    name := "Ivan"
16    age := 27
17    fmt.Println("My name is", name, "and I am", age, "years old.")
18
19 }
```

```
PS C:\dev\go_test\.vscode> C:; cd 'C:\dev\go_test\.vscode'; ${env:GOPATH}='C:\Users\User\go'; & 'C:\Users\User\go\bin\dlv.exe' 'dap' '--client-addr=:54160'
hello, world
hello, worldIvan27Ivan 27
My name is Ivan and I am 27 years old.
PS C:\dev\go_test\.vscode>
```

Рисунок 13 – Результат выполнения программы примера 8

Пример 9. Комментарии

```
1 /*
2  Первая программа
3  на языке Go
4  */
5
6 package main
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 }
15
```

```
PS C:\dev\go_test\.vscode> C:; cd 'C:\dev\go_test\.vscode'; ${env:GOPATH}='C:\Users\User\go'; & 'C:\Users\User\go\bin\dlv.exe' 'dap' '--client-addr=:54298'
Hello Go!
PS C:\dev\go_test\.vscode>
```

Рисунок 14 – Результат выполнения программы примера 9

Пример 10. Константы

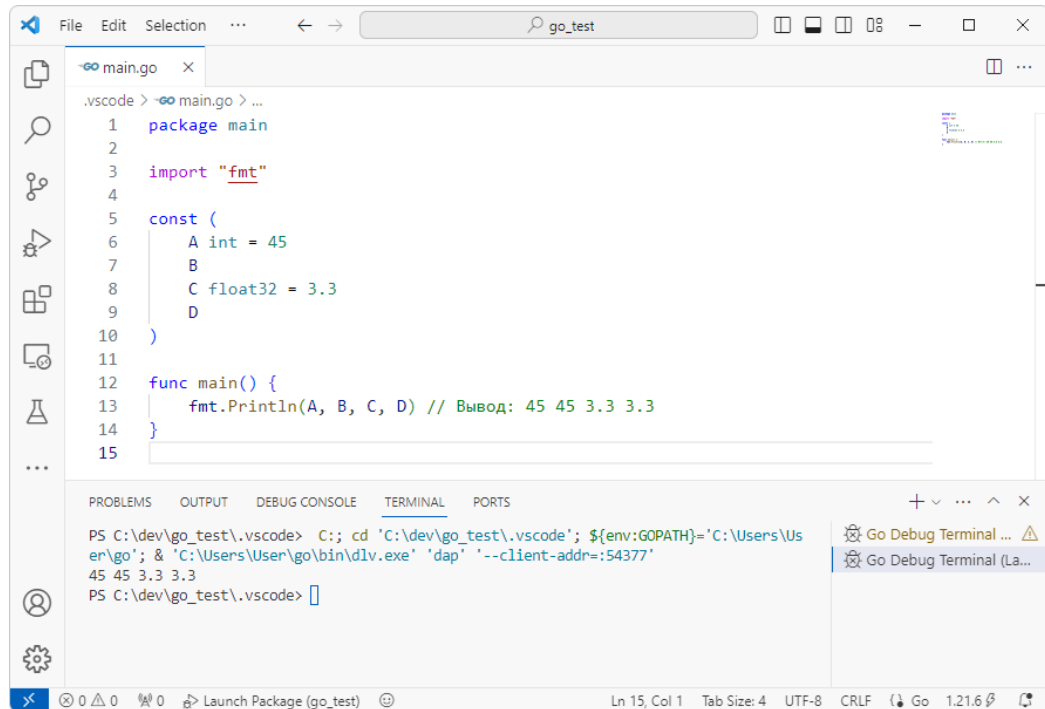


Рисунок 15 – Результат выполнения программы примера 10

Пример 11. Использование `iota` идентификаторов

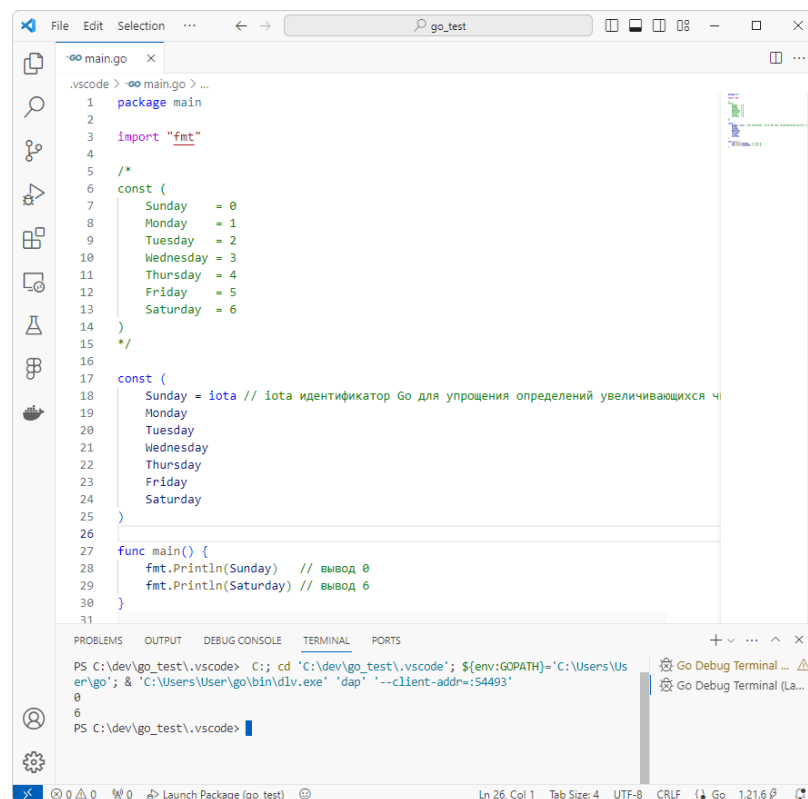


Рисунок 16 – Результат выполнения программы примера 11

Пример 12. Использование и взаимодействие `iota` идентификаторов

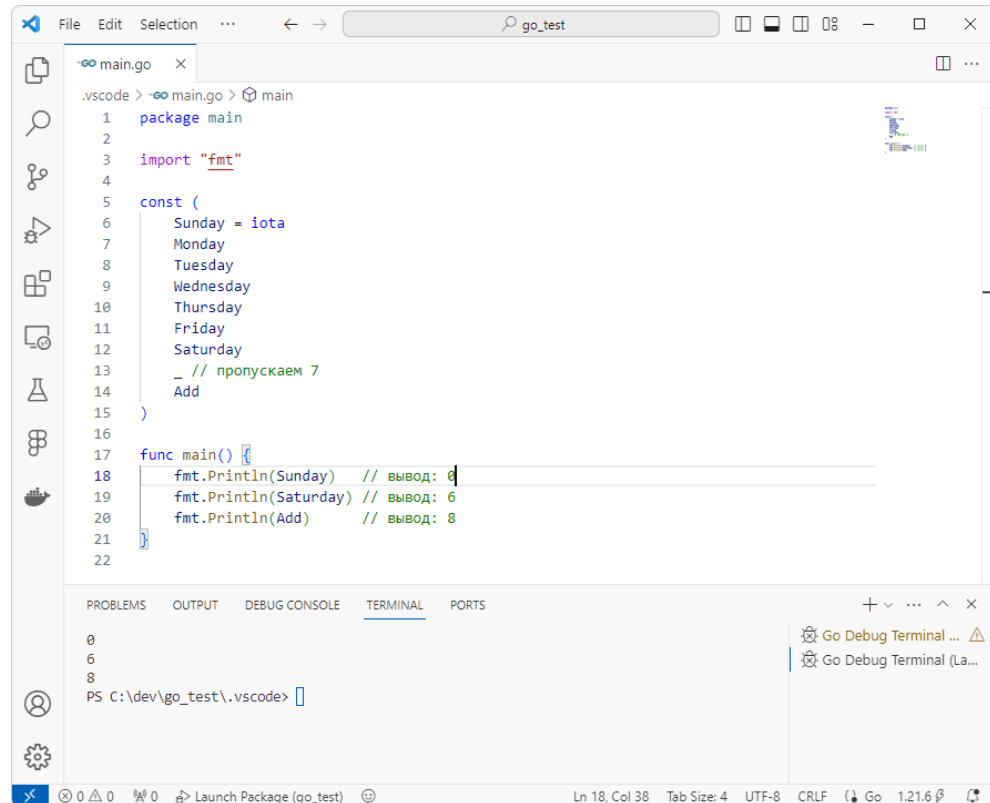


Рисунок 17 – Результат выполнения программы примера 12

Пример 13. Взаимодействие с iota

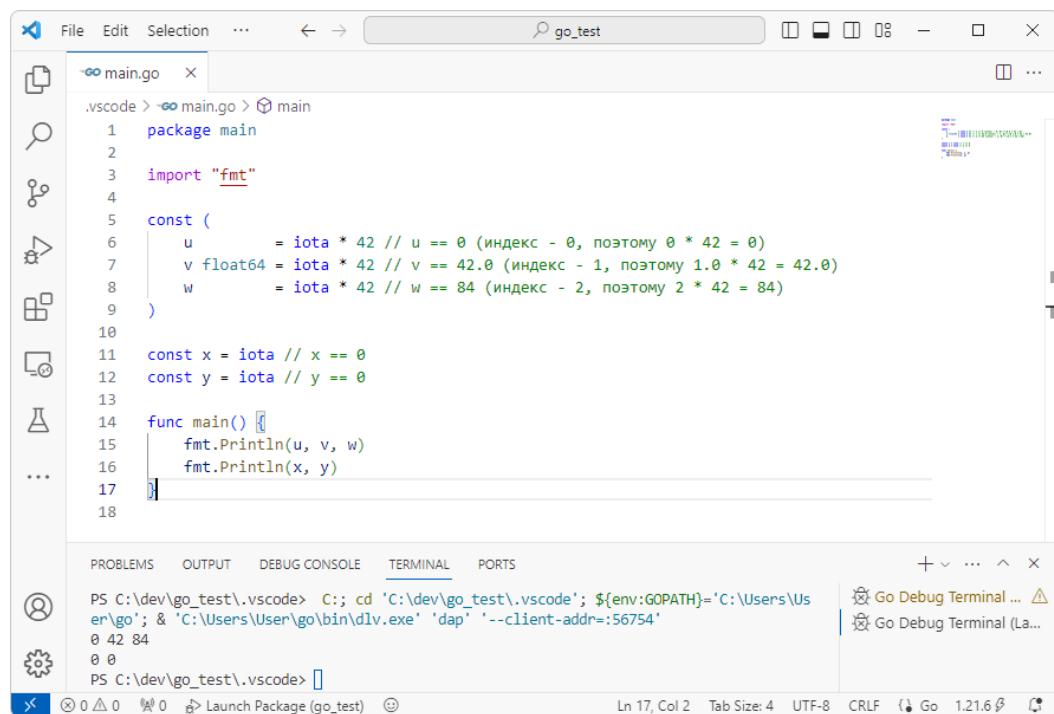
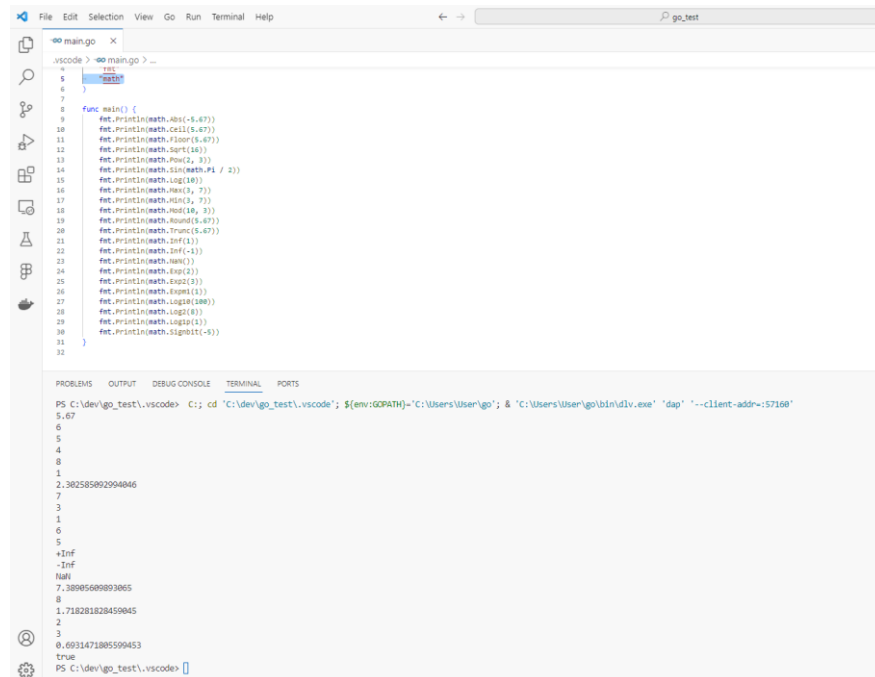


Рисунок 18 – Результат выполнения программы примера 13

Пример 14. Математические функции в Go и их применение



```
1 package main
2
3 import (
4     "fmt"
5 )
6
7 func main() {
8     fmt.Println(math.Abs(-5.67))
9     fmt.Println(math.Ceil(5.67))
10    fmt.Println(math.Floor(5.67))
11    fmt.Println(math.Sqrt(16))
12    fmt.Println(math.Pow(2, 3))
13    fmt.Println(math.Sin(math.Pi / 2))
14    fmt.Println(math.Log(10))
15    fmt.Println(math.Neg(3, 7))
16    fmt.Println(math.Hypot(3, 7))
17    fmt.Println(math.Mod(18, 3))
18    fmt.Println(math.Round(5.67))
19    fmt.Println(math.Trunc(5.67))
20    fmt.Println(math.Pi)
21    fmt.Println(math.Inf(-1))
22    fmt.Println(math.NaN())
23    fmt.Println(math.Exp(2))
24    fmt.Println(math.Exp2(3))
25    fmt.Println(math.Exp10(1))
26    fmt.Println(math.Log10(100))
27    fmt.Println(math.Log(8))
28    fmt.Println(math.Log2(1))
29    fmt.Println(math.Signbit(-5))
30 }
31
32
```

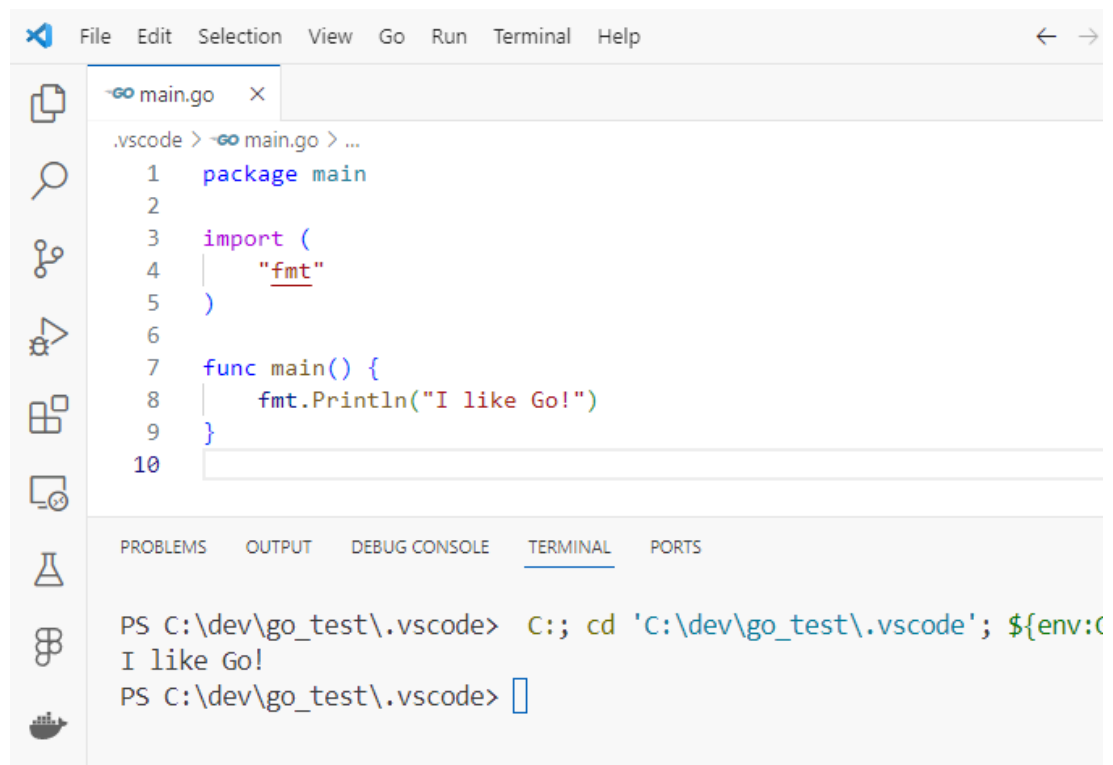
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS C:\dev\go_test\.vscode> cd 'C:\dev\go_test\.vscode'; ${env:GOPATH}="C:\Users\User\go"; & "C:\Users\User\go\bin\div.exe" "dap" "--client-addr=:57168"
5.67
6
5
4
0
1
2.38258982994046
7
3
1
6
5
+Inf
-Inf
NaN
7.38905609893865
8
1.718281828459045
2
3
0.6931471805599453
true
PS C:\dev\go_test\.vscode>
```

Рисунок 19 – Результат выполнения программы примера 14

Практическая часть

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



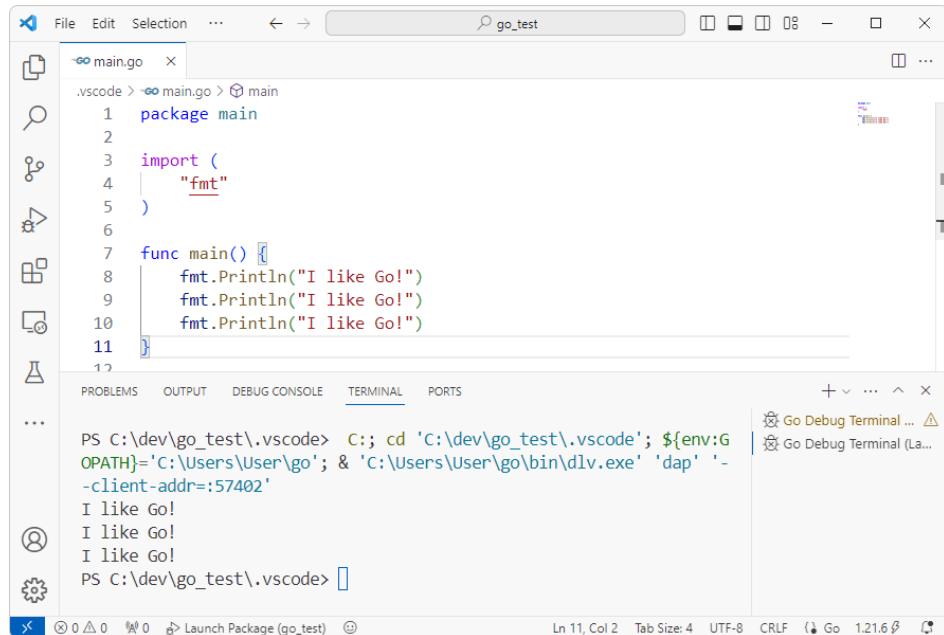
```
1 package main
2
3 import (
4     "fmt"
5 )
6
7 func main() {
8     fmt.Println("I like Go!")
9 }
10
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS C:\dev\go_test\.vscode> C:; cd 'C:\dev\go_test\.vscode'; ${env:GOPATH}="C:\Users\User\go"; & "C:\Users\User\go\bin\div.exe" "dap" "--client-addr=:57168"
I like Go!
PS C:\dev\go_test\.vscode>
```

Рисунок 20 – Результат выполнения программы задания 1

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



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 (
4     "fmt"
5 )
6
7 func main() {
8     fmt.Println("I like Go!")
9     fmt.Println("I like Go!")
10    fmt.Println("I like Go!")
11 }
```

The bottom panel shows the terminal output:

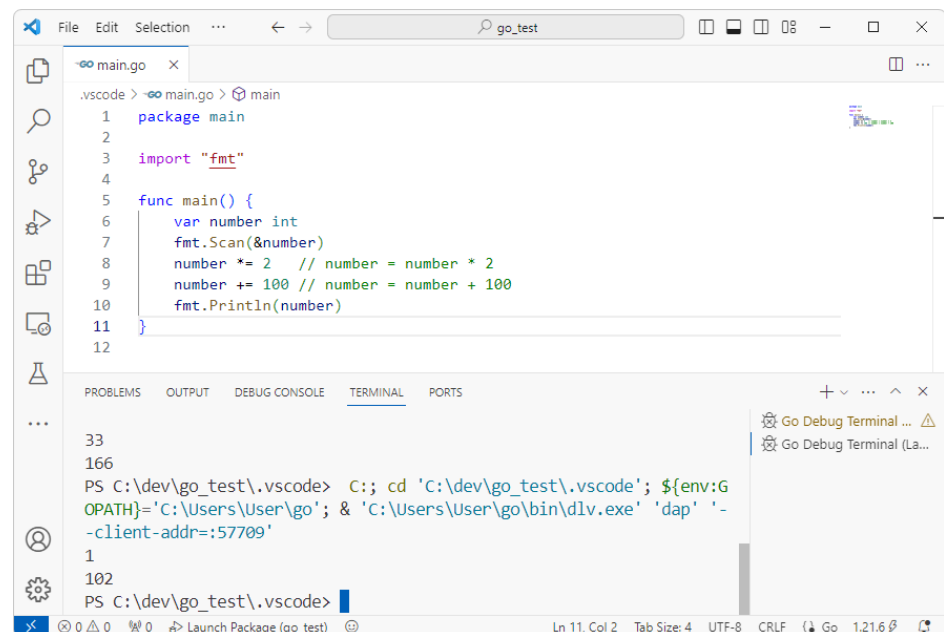
```
PS C:\dev\go_test\.vscode> C:; cd 'C:\dev\go_test\.vscode'; ${env:GOPATH}='C:\Users\User\go'; & 'C:\Users\User\go\bin\dlv.exe' 'dap' '-client-addr=:57402'
I like Go!
I like Go!
I like Go!
PS C:\dev\go_test\.vscode>
```

Рисунок 21 – Результат выполнения программы задания 2

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.



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

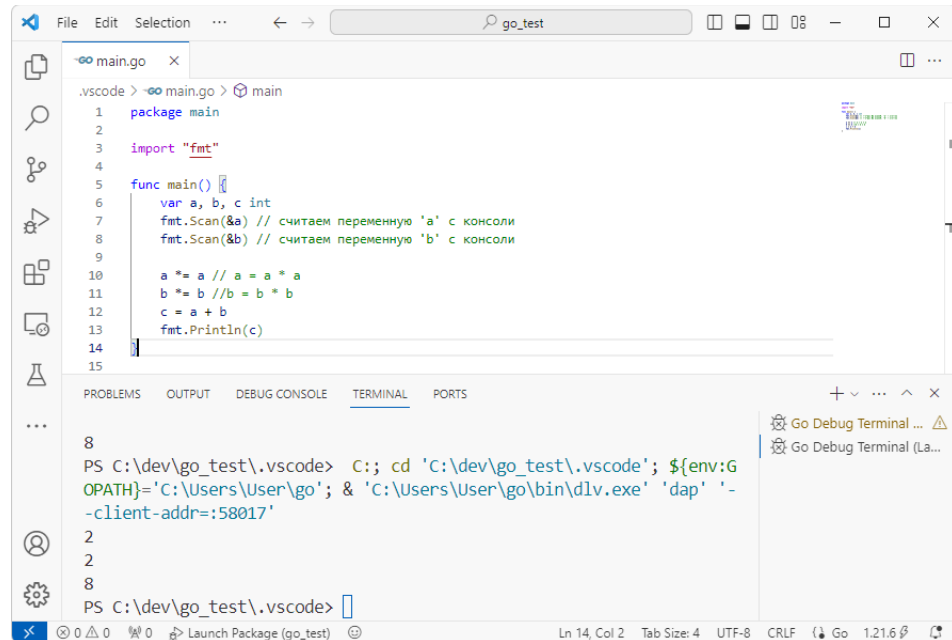
```
1 package main
2
3 import "fmt"
4
5 func main() {
6     var number int
7     fmt.Scan(&number)
8     number *= 2 // number = number * 2
9     number += 100 // number = number + 100
10    fmt.Println(number)
11 }
```

The bottom panel shows the terminal output:

```
33
166
PS C:\dev\go_test\.vscode> C:; cd 'C:\dev\go_test\.vscode'; ${env:GOPATH}='C:\Users\User\go'; & 'C:\Users\User\go\bin\dlv.exe' 'dap' '-client-addr=:57709'
1
102
PS C:\dev\go_test\.vscode>
```

Рисунок 22 – Результат выполнения программы задания 3

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



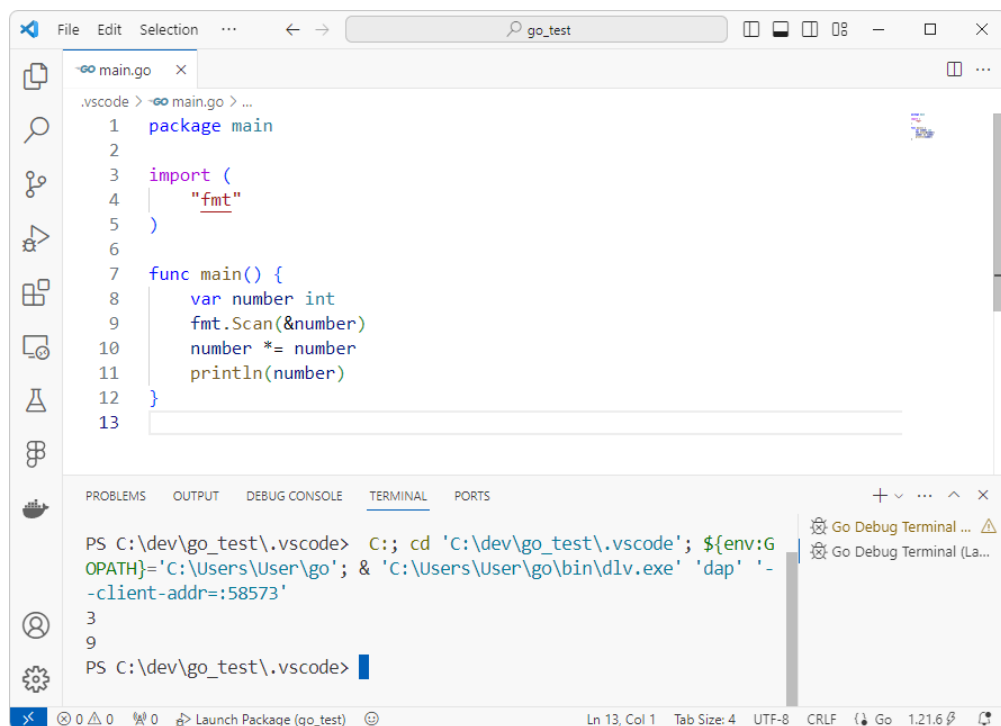
The screenshot shows the VS Code editor with a Go file named `main.go`. The code defines a `main` function that reads two integers `a` and `b` from the console, calculates their squares (`a*a` and `b*b`), and prints the sum (`a + b`). The terminal output shows the command to run the program and the resulting output: `8`.

```
1 package main
2
3 import "fmt"
4
5 func main() {
6     var a, b, c int
7     fmt.Scan(&a) // считаем переменную 'a' с консоли
8     fmt.Scan(&b) // считаем переменную 'b' с консоли
9
10    a *= a // a = a * a
11    b *= b // b = b * b
12    c = a + b
13    fmt.Println(c)
14 }
```

```
PS C:\dev\go_test\.vscode> C:; cd 'C:\dev\go_test\.vscode'; ${env:GOPATH}='C:\Users\User\go'; & 'C:\Users\User\go\bin\dlv.exe' 'dap' '-client-addr=:58017'
2
2
8
PS C:\dev\go_test\.vscode>
```

Рисунок 23 – Результат выполнения программы задания 4

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



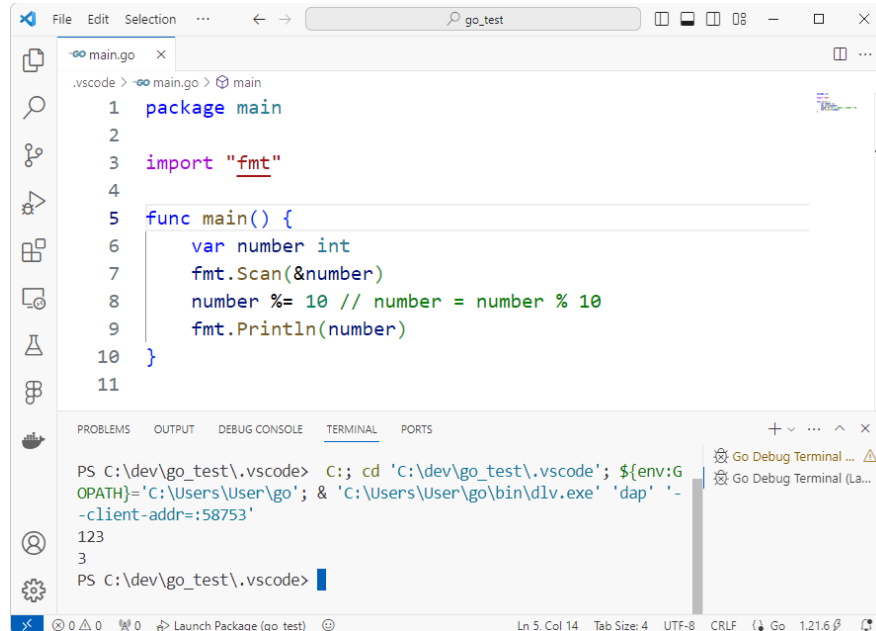
The screenshot shows the VS Code editor with a Go file named `main.go`. The code defines a `main` function that reads an integer `number` from the console, calculates its square (`number * number`), and prints the result. The terminal output shows the command to run the program and the resulting output: `9`.

```
1 package main
2
3 import (
4     "fmt"
5 )
6
7 func main() {
8     var number int
9     fmt.Scan(&number)
10    number *= number
11    println(number)
12 }
13
```

```
PS C:\dev\go_test\.vscode> C:; cd 'C:\dev\go_test\.vscode'; ${env:GOPATH}='C:\Users\User\go'; & 'C:\Users\User\go\bin\dlv.exe' 'dap' '-client-addr=:58573'
3
9
PS C:\dev\go_test\.vscode>
```

Рисунок 24 – Результат выполнения программы задания 5

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.



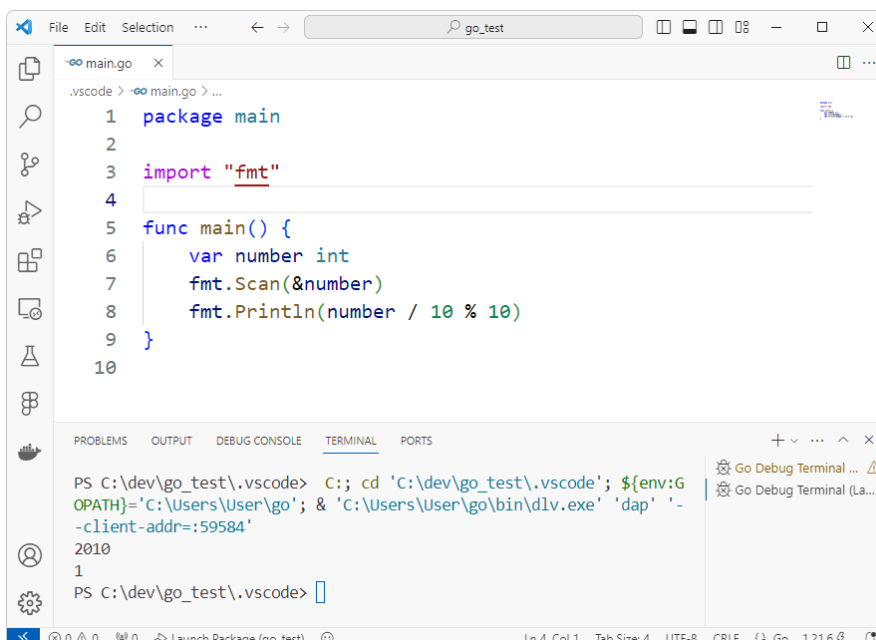
The screenshot shows the Visual Studio Code editor with a Go file named `main.go`. The code defines a `main` package and a `main` function. Inside the function, an integer `number` is declared, and `fmt.Scan(&number)` is used to read input. The last digit is calculated using `number %= 10` (commented as `number = number % 10`), and `fmt.Println(number)` prints the result. The terminal shows the command `go run main.go` being executed, and the output is `123`, indicating the last digit of the input number 1233.

```
1 package main
2
3 import "fmt"
4
5 func main() {
6     var number int
7     fmt.Scan(&number)
8     number %= 10 // number = number % 10
9     fmt.Println(number)
10 }
11
```

```
PS C:\dev\go_test\.vscode> C:; cd 'C:\dev\go_test\.vscode'; ${env:GOPATH}='C:\Users\User\go'; & 'C:\Users\User\go\bin\dlv.exe' 'dap' '-client-addr=:58753'
123
3
PS C:\dev\go_test\.vscode>
```

Рисунок 25 – Результат выполнения программы задания 6

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.



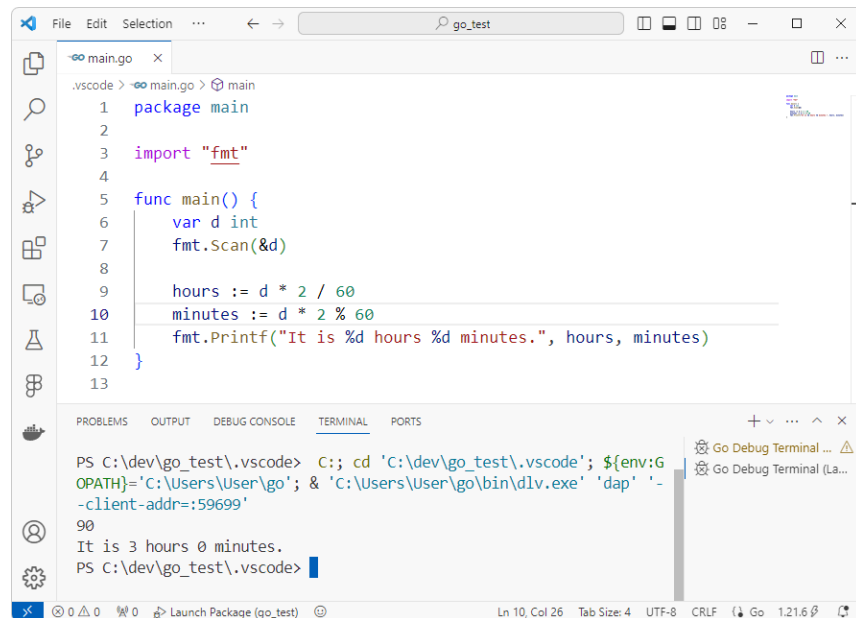
The screenshot shows the Visual Studio Code editor with a Go file named `main.go`. The code defines a `main` package and a `main` function. Inside the function, an integer `number` is declared, and `fmt.Scan(&number)` is used to read input. The number of tens is calculated using `number / 10 % 10`, and `fmt.Println` prints the result. The terminal shows the command `go run main.go` being executed, and the output is `2010`, indicating the number of tens in the input number 20101.

```
1 package main
2
3 import "fmt"
4
5 func main() {
6     var number int
7     fmt.Scan(&number)
8     fmt.Println(number / 10 % 10)
9 }
10
```

```
PS C:\dev\go_test\.vscode> C:; cd 'C:\dev\go_test\.vscode'; ${env:GOPATH}='C:\Users\User\go'; & 'C:\Users\User\go\bin\dlv.exe' 'dap' '-client-addr=:59584'
2010
1
PS C:\dev\go_test\.vscode>
```

Рисунок 26 – Результат выполнения программы задания 7

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.



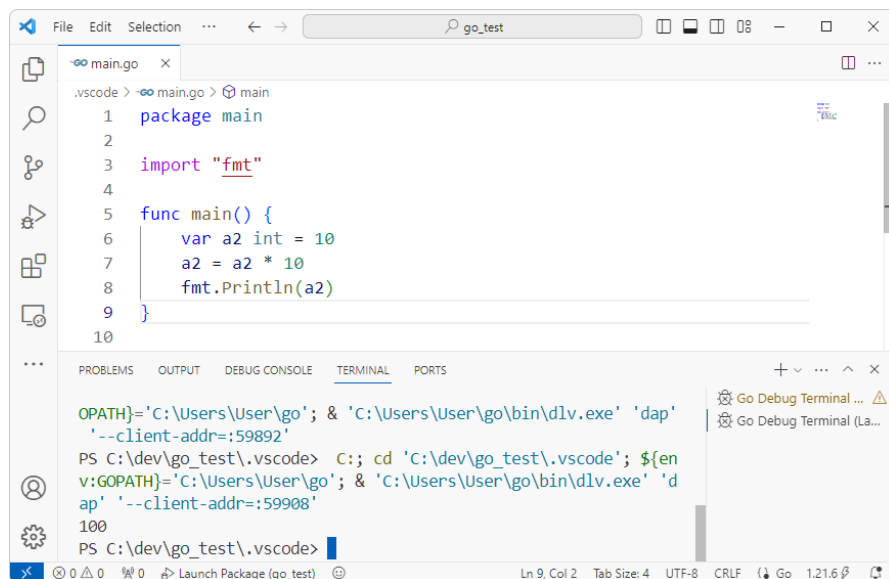
```
.vscode> go main
1 package main
2
3 import "fmt"
4
5 func main() {
6     var d int
7     fmt.Scan(&d)
8
9     hours := d * 2 / 60
10    minutes := d * 2 % 60
11    fmt.Printf("It is %d hours %d minutes.", hours, minutes)
12 }
13
```

```
PS C:\dev\go_test\.vscode> C:; cd 'C:\dev\go_test\.vscode'; ${env:GOPATH}='C:\Users\User\go'; & 'C:\Users\User\go\bin\dlv.exe' 'dap' '-client-addr=:59699'
90
It is 3 hours 0 minutes.
PS C:\dev\go_test\.vscode>
```

Рисунок 27 – Результат выполнения программы задания 8

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

Была раскомментирована строка `var a2 int = 10`.

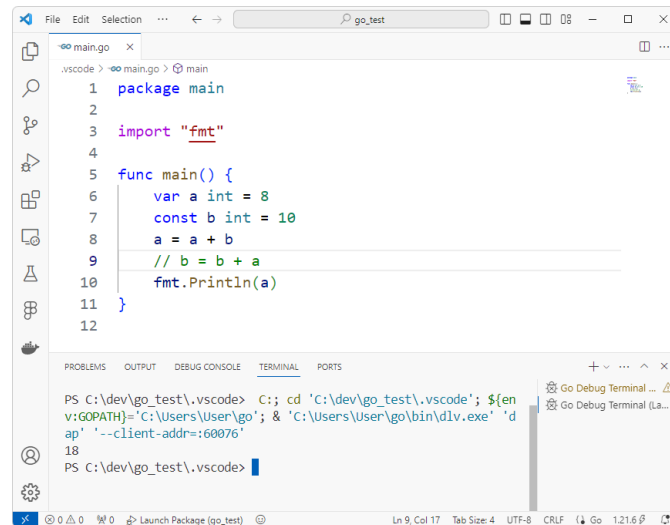


```
.vscode> go main
1 package main
2
3 import "fmt"
4
5 func main() {
6     var a2 int = 10
7     a2 = a2 * 10
8     fmt.Println(a2)
9 }
10
```

```
PS C:\dev\go_test\.vscode> C:; cd 'C:\dev\go_test\.vscode'; ${env:GOPATH}='C:\Users\User\go'; & 'C:\Users\User\go\bin\dlv.exe' 'dap' '-client-addr=:59892'
PS C:\dev\go_test\.vscode> C:; cd 'C:\dev\go_test\.vscode'; ${env:GOPATH}='C:\Users\User\go'; & 'C:\Users\User\go\bin\dlv.exe' 'dap' '-client-addr=:59908'
100
PS C:\dev\go_test\.vscode>
```

Рисунок 28 – Результат выполнения программы задания 9

Задание 10. Задача: Исправьте ошибку в программе ниже
Ошибочная строка была закомментирована в коде.



```
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
```

Terminal output:

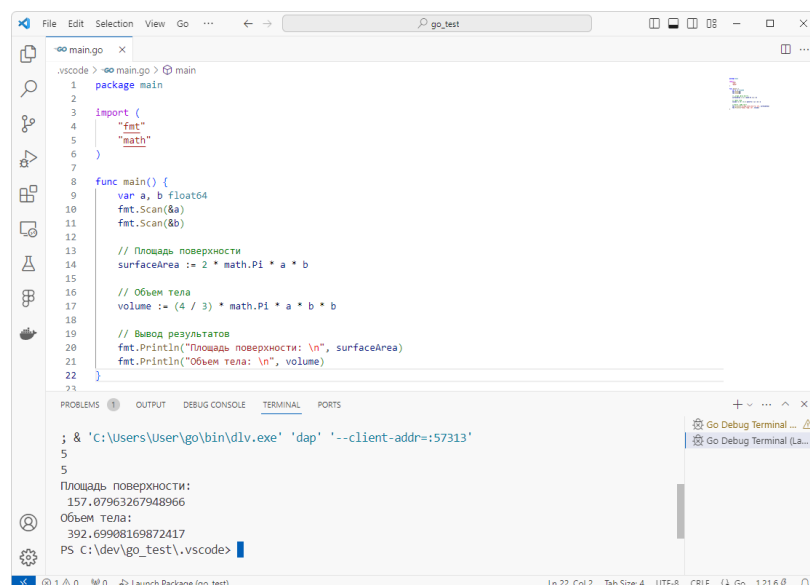
```
PS C:\dev\go_test\.vscode> C:\dev\go_test\.vscode> $(env:GOPATH=C:\Users\User\go; & 'C:\Users\User\go\bin\dlv.exe' 'dap' '--client-addr=:60076')
PS C:\dev\go_test\.vscode>
```

Рисунок 29 – Результат выполнения программы задания 10

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

$$\frac{x^2}{a^2} + \frac{y^2}{b^2} = 1, \text{ вокруг оси } O_x.$$

Для решения были использованы формулы площади поверхности и объема для эллипса.



```
1 package main
2
3 import (
4     "fmt"
5     "math"
6 )
7
8 func main() {
9     var a, b float64
10    fmt.Scan(&a)
11    fmt.Scan(&b)
12
13    // Площадь поверхности
14    surfaceArea := 2 * math.Pi * a * b
15
16    // Объем тела
17    volume := (4 / 3) * math.Pi * a * b * b
18
19    // Вывод результатов
20    fmt.Println("Площадь поверхности: \n", surfaceArea)
21    fmt.Println("Объем тела: \n", volume)
22 }
23
```

Terminal output:

```
; & 'C:\Users\User\go\bin\dlv.exe' 'dap' '--client-addr=:57313'
5
Площадь поверхности:
157.07963267948966
Объем тела:
392.69908169872417
PS C:\dev\go_test\.vscode>
```

Рисунок 30 – Результат выполнения программы задания 11

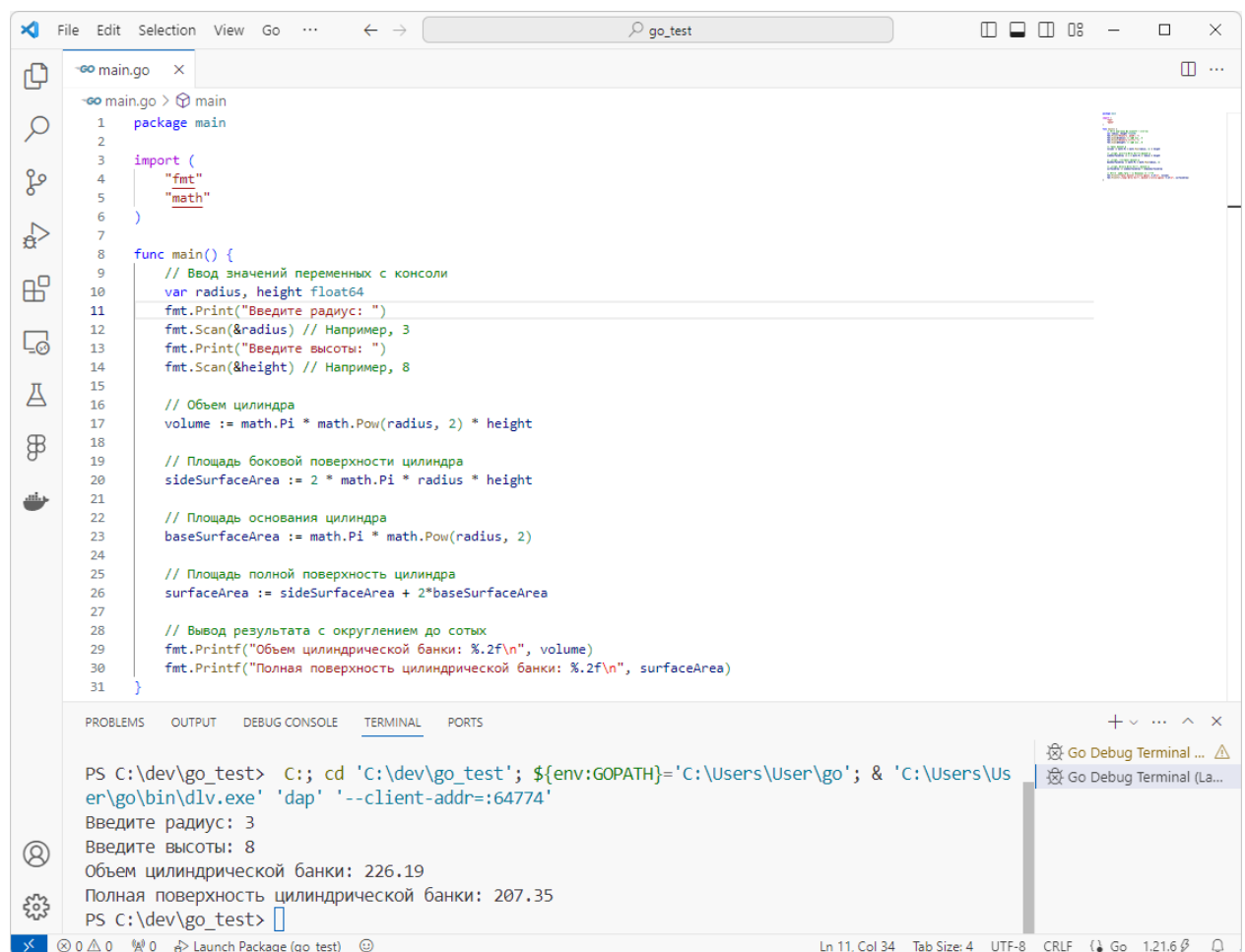
Индивидуальное задание 1 (Вариант 8)

8. Объем и площадь цилиндрической банки: Задайте переменные для радиуса и высоты цилиндрической банки. Рассчитайте и выведите объем и полную поверхность цилиндра.

Использованные формулы:

Объем цилиндра: $V = \pi r^2 h$

Полная поверхность цилиндра: $S_{\text{полн.}} = S_{\text{бок.}} + 2S_{\text{осн.}}$, где $S_{\text{бок.}} = 2\pi r h$ и $S_{\text{осн.}} = \pi r^2$, тогда $S_{\text{полн.}} = 2\pi r h + 2(\pi r^2)$



```
1 package main
2
3 import (
4     "fmt"
5     "math"
6 )
7
8 func main() {
9     // Ввод значений переменных с консоли
10    var radius, height float64
11    fmt.Print("Введите радиус: ")
12    fmt.Scan(&radius) // Например, 3
13    fmt.Print("Введите высоты: ")
14    fmt.Scan(&height) // Например, 8
15
16    // Объем цилиндра
17    volume := math.Pi * math.Pow(radius, 2) * height
18
19    // Площадь боковой поверхности цилиндра
20    sideSurfaceArea := 2 * math.Pi * radius * height
21
22    // Площадь основания цилиндра
23    baseSurfaceArea := math.Pi * math.Pow(radius, 2)
24
25    // Площадь полной поверхности цилиндра
26    surfaceArea := sideSurfaceArea + 2*baseSurfaceArea
27
28    // Вывод результата с округлением до сотых
29    fmt.Printf("Объем цилиндрической банки: %.2f\n", volume)
30    fmt.Printf("Полная поверхность цилиндрической банки: %.2f\n", surfaceArea)
31 }
```

PS C:\dev\go_test> C:; cd 'C:\dev\go_test'; \${env:GOPATH}='C:\Users\User\go'; & 'C:\Users\User\go\bin\dlv.exe' 'dap' '--client-addr=:64774'

Введите радиус: 3
Введите высоты: 8
Объем цилиндрической банки: 226.19
Полная поверхность цилиндрической банки: 207.35
PS C:\dev\go_test>

Рисунок 31 – Результат выполнения программы индивидуального задания 1

Индивидуальное задание 2 (Вариант 8)

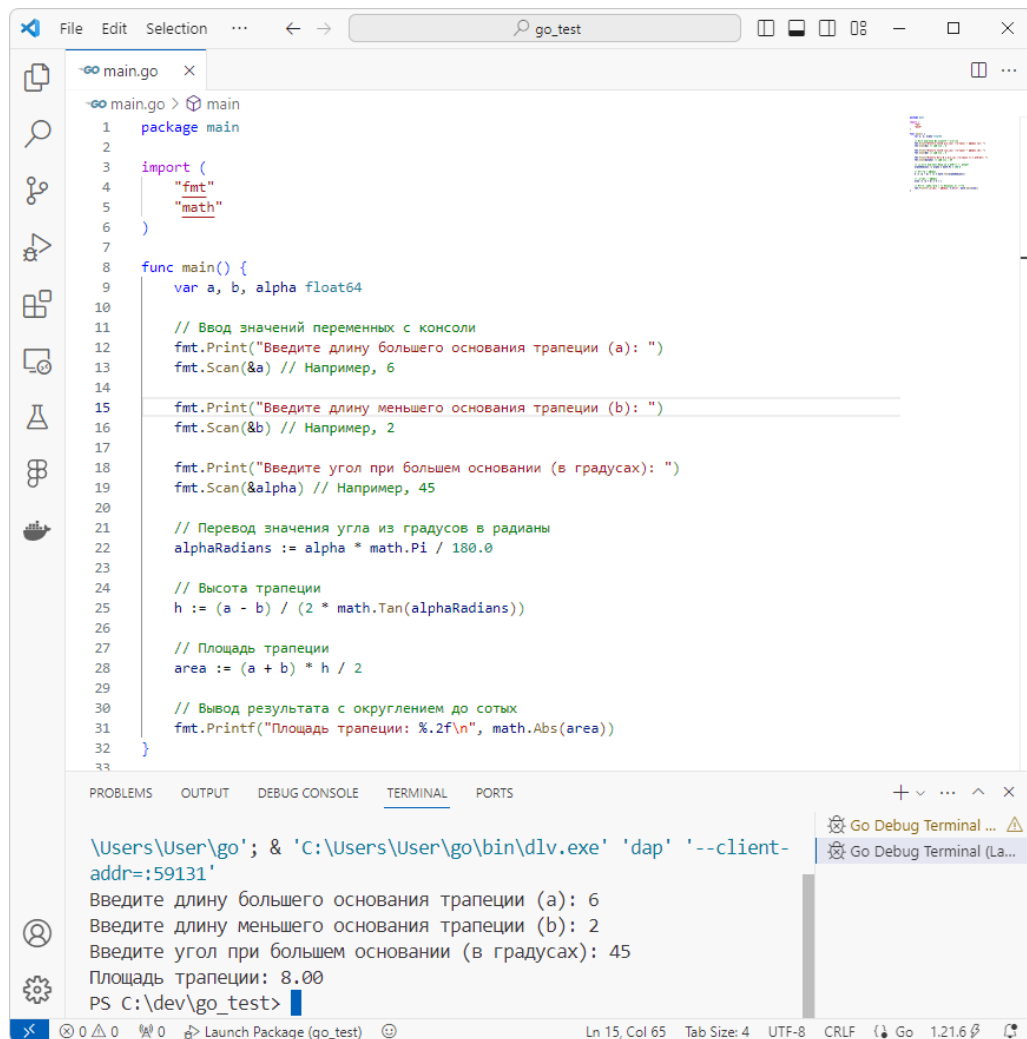
8. Даны основания равнобедренной трапеции и угол при большем основании. Найти площадь трапеции.

a, b – основания (большее и меньше соответственно)

angle – угол между большим основанием трапеции и боковой стороной

Формула площади трапеции: $S = \frac{(a+b) \cdot h}{2}$

Формула высоты трапеции получена из деления трапеции на два прямоугольных треугольника (катетов и тангенса угла): $h = \frac{a \cdot b}{2tg(\alpha)}$



```
1 package main
2
3 import (
4     "fmt"
5     "math"
6 )
7
8 func main() {
9     var a, b, alpha float64
10
11     // Ввод значений переменных с консоли
12     fmt.Print("Введите длину большего основания трапеции (a): ")
13     fmt.Scan(&a) // Например, 6
14
15     fmt.Print("Введите длину меньшего основания трапеции (b): ")
16     fmt.Scan(&b) // Например, 2
17
18     fmt.Print("Введите угол при большем основании (в градусах): ")
19     fmt.Scan(&alpha) // Например, 45
20
21     // Перевод значения угла из градусов в радианы
22     alphaRadians := alpha * math.Pi / 180.0
23
24     // Высота трапеции
25     h := (a - b) / (2 * math.Tan(alphaRadians))
26
27     // Площадь трапеции
28     area := (a + b) * h / 2
29
30     // Вывод результата с округлением до сотых
31     fmt.Printf("Площадь трапеции: %.2f\n", math.Abs(area))
32 }
33
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
\Users\User\go'; & 'C:\Users\User\go\bin\dlv.exe' 'dap' '--client-addr=:59131'
Введите длину большего основания трапеции (a): 6
Введите длину меньшего основания трапеции (b): 2
Введите угол при большем основании (в градусах): 45
Площадь трапеции: 8.00
PS C:\dev\go_test>
```

Ln 15, Col 65 Tab Size: 4 UTF-8 CRLF Go 1.21.6

Рисунок 32 – Результат выполнения программы индивидуального задания 2

Вывод: Изучены основы работы с языком программирования Go, а именно: установка и настройка среды для программирования на этом языке, типы данных, работа с переменными и арифметическими операциями, встроенные функции, а также на основе полученных знания написаны программы.

Контрольные вопросы:

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

```
var name int;
```

Создается переменная `name` типа `int` со значением по умолчанию.

```
var name int = 10
```

При таком объявлении переменной `x` будет назначено начальное значение 10.

Go способен и самостоятельно определить тип переменной:

```
name := 10
```

В таком случае, создается переменная `name` типа `int` и ей присвоится значение 10, используя синтаксис короткого объявления переменной (`:=`).

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

По умолчанию значение переменной `int` в Go: 0.

3. Как изменить значение существующей переменной в Go?

```
var x int = 10
```

```
x = 20
```

4. Что такое множественное объявление переменных в Go?

Язык Go позволяет объявление нескольких переменных сразу, например, инициализация трех переменных одного типа может выглядеть следующим образом:

1) в несколько строк:

```
var a, b, c int // Объявление трех переменных типа int
```

```
a = 1
```

```
b = 2
```

```
c = 3
```

Выше переменные сначала множественно объявляются, а затем изменяются их значения на необходимые.

2) в одну строку:

```
var x, y, z = 4, 5, "text"
```

Здесь же идет и объявление и инициализация трех переменных разных типов сразу: x типа int, y типа int и z типа string.

5. Как объявить константу в Go?

Для определения констант применяется ключевое слово `const`:

```
const pi float64 = 3.1415
```

Константы, как и обычные переменные, можно объявлять в блоке:

```
const (  
    a int = 45  
    b float32 = 3.3  
)
```

Также можно не указывать значение следующей константы по порядку (значение будет скопировано):

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

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

Мы не можем менять значение константы после объявления.

7. Какие арифметические операторы поддерживаются в Go?

В языке Go поддерживаются операторы:

- Сложение (+)
- Вычитание (-)
- Умножение (*)
- Деление (/)
- Остаток от деления (%)
- Инкремент (++)
- Декремент (--)

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

В Go для выполнения операции остатка от деления используется оператор %.

9. Какой результат выражения 5 / 2 в Go?

Результатом выражения будет 2, так как мы получим только целую часть ответа (2.5).

10. Как считать строку с консоли в Go?

Считать строку с консоли можно методом `fmt.Scan(&a)`, где `&a` – ссылка (более точно - адрес) на переменную `a`. Если проще, то введённое число запишется из консоли напрямую в эту переменную и там будет храниться, пока не понадобится её куда-нибудь пристроить или изменить.

Также можно читать с консоли сразу несколько переменных: `fmt.Scan(&a, &b, &c)`.

11. Как считать целое число с консоли в Go?

Необходимо сначала объявить переменную для целого числа до считывания, например, `var num int`. Затем, воспользоваться либо функцией

`fmt.Scan(&num)`, либо `fmt.Scanf("%d", &num)`, где `%d` целые числа в десятичном формате.

Основное различие между `Scan` и `Scanf` заключается в том, что `Scan` используется для сканирования строки без явного форматирования, в то время как `Scanf` используется для сканирования ввода с определенным форматом.

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

Обработать ошибку можно несколькими способами.

Самым простым способом является:

```
var input string
// Считываем строку с консоли
if _, err := fmt.Scan(&input); err != nil {
    fmt.Println("Произошла ошибка:", err)
    return
}
// Выводим данные, если считывание прошло успешно
fmt.Println("Вы ввели:", input)
```

`_` – результат сканирования, `err` – переменная, в которой будет храниться ошибка, если она произойдет во время операции сканирования.

`err != nil` – проверка на наличие ошибки. Если переменная `err` не равна `nil`, значит, произошла ошибка (`nil` используется для представления нулевого значения – `null`).

Более сложным способом обработки ошибки является использование стандартного пакета «errors».

13. Как вывести строку в консоль в Go?

Для вывода данных на консоль мы на данном этапе можно использовать методы, которые присутствуют в пакете `fmt` – `Print()` и `Println()`, например, `fmt.Print("hello, world")`.

Первый метод при выводе нескольких объектов вставляет между ними пробелы, если среди них нет строк.

Второй всегда ставит пробелы между выводимыми объектами, плюс добавляет новую строку. То есть он пригодится, если нам необходимо будет сделать вывод на нескольких строках.

Можно выводить и несколько объектов:

```
fmt.Print("Ivan", 27) // Ivan27
```

```
fmt.Println("Ivan", 27) // Ivan 27
```

```
fmt.Println("My name is", name, "and I am", age, "years old.")
```

Также, для вывода существует функция `fmt.Printf()`. Она принимает строку формата, в которой могут встречаться специальные символы, называемые форматирующими символами, а также аргументы, которые будут вставлены в соответствующие места в этой строке. Например: `fmt.Printf("Привет, %s! Вам %d лет.\n", "Андрей", 25)`. Этот код напечатает строку "Привет, Андрей! Вам 25 лет."

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

```
var number int = 42
```

```
fmt.Println(number)
```

или

```
fmt.Printf("%d", number)
```

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

В Go для форматирования вывода числа с плавающей точкой вы можете использовать спецификатор формата «%f».

```
var value float64 = 3.14159
```

```
fmt.Printf("%.2f", value)
```

Выводом будет 3.14, то есть, мы сделали округление до двух знаков после запятой.

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

Чем отличается оператор := от оператора = в Go?

Переменную типа byte можно объявить так: `var byteValue byte = 65`

Оператор «:=» позволяет одновременно объявить переменную и присвоить ей значение. Он используется внутри функций для создания новых переменных.

Оператор «=» просто присваивает значение переменной. Он используется для изменения значения существующей переменной.

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

`float32` – 32-битное представление числа с плавающей точкой одинарной точности.

`float64` (или просто `float`) – 64-битное представление числа с плавающей точкой двойной точности.

Обычно для большинства ситуаций используется тип `float64`, так как он обеспечивает большую точность по сравнению с `float32`, хотя может занимать больше памяти.

18. Как объявить и использовать несколько переменных в Go?

1) Используя оператор `var`: `var x, y, z int`

2) Используя оператор `:=` для краткого объявления переменных: `x, y, z := 10, 20, 0`

3) Объявление с использованием разных типов данных:

```
var (  
    name  string  
    age   int
```


height float64

)

4) Множественное присваивание

$x, y := 10, 20$

$x, y = y, x$ // Обмен значениями переменных без использования временной переменной

Переменным можно присваивать значения, использовать их в выражениях и операциях, а также выводить их значения или использовать их в других частях программы.