**Московский государственный технический**

**университет им. Н.Э. Баумана**

Факультет «Информатика и системы управления»

Кафедра ИУ5 «Системы обработки информации и управления»

Курс «Парадигмы и конструкции языков программирования»

Отчет по домашнему заданию

«Telegram-Bot»

|  |  |  |
| --- | --- | --- |
| Выполнил: |  | Проверил: |
| студент группы ИУ5-31Б |  | преподаватель каф. ИУ5 |
| Кузнецов А.Д. |  | Нардид А.Н. |
|  |  |  |

Москва, 2024 г.

**Описание задания**

Разработать телеграм-бота на языке Go, который выполняет функции калькулятора для различных математических задач. Бот должен реагировать на команды и ввод пользователя, предлагая следующие возможности:

1. **Нахождение корней квадратного уравнения**:
   * Пользователь вводит коэффициенты a, b, c через пробел.
   * Бот рассчитывает дискриминант и корни уравнения по формулам:

В случае D < 0, бот сообщает об отсутствии действительных корней.

1. **Вычисление площади треугольника**:
   * Пользователь вводит длины сторон треугольника a, b, c.
   * Бот проверяет существование треугольника (неравенство треугольника) и вычисляет площадь по формуле Герона: .
2. **Нахождение простых множителей числа**:
   * Пользователь вводит целое число n.
   * Бот разлагает число на простые множители, используя последовательное деление, и выводит результат.

**Дополнительные требования**:

* Бот должен использовать библиотеку [go-telegram-bot-api](https://github.com/go-telegram-bot-api/telegram-bot-api) версии 5.5.1.
* Реализовать обработку команд и пользовательского ввода с использованием кнопок для выбора функций.
* Программа должна быть структурирована с использованием отдельных функций для выполнения каждой из задач.
* Обеспечить корректную обработку неверного ввода.

**Текст программы**

Файл *go.sun*

github.com/go-telegram-bot-api/telegram-bot-api/v5 v5.5.1 h1:wG8n/XJQ07TmjbITcGiUaOtXxdrINDz1b0J1w0SzqDc=

github.com/go-telegram-bot-api/telegram-bot-api/v5 v5.5.1/go.mod h1:A2S0CWkNylc2phvKXWBBdD3K0iGnDBGbzRpISP2zBl8=

Файл *go.mod*

module mytelegrambot

go 1.23.1

require github.com/go-telegram-bot-api/telegram-bot-api/v5 v5.5.1

Файл *main.go*

package main

import (

    "fmt"

    "log"

    "math"

    "strconv"

    "strings"

    tgbotapi "github.com/go-telegram-bot-api/telegram-bot-api/v5"

)

var currentFunction string

func main() {

    bot, err := tgbotapi.NewBotAPI("7885796272:AAEyoAOmCs3Et5u2RWtVydF\_B5RccM\_kgsg")

    if err != nil {

        log.Panic(err)

    }

    bot.Debug = true

    log.Printf("Authorized on account %s", bot.Self.UserName)

    u := tgbotapi.NewUpdate(0)

    u.Timeout = 60

    updates := bot.GetUpdatesChan(u)

    // Создание кнопок

    rootButton := tgbotapi.NewKeyboardButton("Найти корни квадратного уравнения")

    areaButton := tgbotapi.NewKeyboardButton("Найти площадь треугольника")

    factorsButton := tgbotapi.NewKeyboardButton("Найти простые множители")

    keyboard := tgbotapi.NewReplyKeyboard(

        tgbotapi.NewKeyboardButtonRow(rootButton),

        tgbotapi.NewKeyboardButtonRow(areaButton),

        tgbotapi.NewKeyboardButtonRow(factorsButton),

    )

    for update := range updates {

        if update.Message == nil {

            continue

        }

        msg := tgbotapi.NewMessage(update.Message.Chat.ID, "")

        switch update.Message.Text {

        case "/start":

            msg.Text = "Выберите функцию калькулятора:"

            msg.ReplyMarkup = keyboard

            bot.Send(msg)

        case "Найти корни квадратного уравнения":

            currentFunction = "roots"

            msg.Text = "Введите коэффициенты a, b и c через пробел (например: 1 -3 2):"

            bot.Send(msg)

        case "Найти площадь треугольника":

            currentFunction = "triangle"

            msg.Text = "Введите длины сторон треугольника a, b и c через пробел (например: 3 4 5):"

            bot.Send(msg)

        case "Найти простые множители":

            currentFunction = "factors"

            msg.Text = "Введите число для поиска его простых множителей:"

            bot.Send(msg)

        default:

            // Обработка ввода в зависимости от выбранной функции

            switch currentFunction {

            case "roots":

                // Разбор коэффициентов и вызов функции для нахождения корней

                input := strings.Fields(update.Message.Text)

                if len(input) == 3 {

                    a, \_ := strconv.ParseFloat(input[0], 64)

                    b, \_ := strconv.ParseFloat(input[1], 64)

                    c, \_ := strconv.ParseFloat(input[2], 64)

                    msg.Text = calculateRoots(a, b, c)

                } else {

                    msg.Text = "Неверный формат. Пожалуйста, введите три коэффициента через пробел."

                }

            case "triangle":

                // Разбор сторон треугольника и вызов функции для нахождения площади

                input := strings.Fields(update.Message.Text)

                if len(input) == 3 {

                    a, \_ := strconv.ParseFloat(input[0], 64)

                    b, \_ := strconv.ParseFloat(input[1], 64)

                    c, \_ := strconv.ParseFloat(input[2], 64)

                    msg.Text = calculateTriangleArea(a, b, c)

                } else {

                    msg.Text = "Неверный формат. Пожалуйста, введите длины сторон через пробел."

                }

            case "factors":

                // Разбор числа и вызов функции для нахождения простых множителей

                n, err := strconv.Atoi(update.Message.Text)

                if err == nil && n > 1 {

                    msg.Text = calculateFactors(n)

                } else {

                    msg.Text = "Неверный формат. Пожалуйста, введите целое число больше 1."

                }

            default:

                msg.Text = "Пожалуйста, выберите функцию из предложенных вариантов."

            }

            bot.Send(msg)

        }

    }

}

func calculateRoots(a, b, c float64) string {

    D := b\*b - 4\*a\*c

    if D < 0 {

        return "Уравнение не имеет действительных корней"

    }

    x1 := (-b + math.Sqrt(D)) / (2 \* a)

    x2 := (-b - math.Sqrt(D)) / (2 \* a)

    return fmt.Sprintf("Корни уравнения: x1 = %.2f, x2 = %.2f\n", x1, x2)

}

func calculateTriangleArea(a, b, c float64) string {

    if a+b > c && a+c > b && b+c > a {

        p := (a + b + c) / 2

        area := math.Sqrt(p \* (p - a) \* (p - b) \* (p - c))

        return fmt.Sprintf("Площадь треугольника: %.2f\n", area)

    } else {

        return fmt.Sprintf("Треугольник с такими сторонами не существует.")

    }

}

func calculateFactors(n int) string {

    if n > 1 {

        factors := primeFactors(n)

        return fmt.Sprintf("Простые множители числа %d: %v\n", n, factors)

    } else {

        return fmt.Sprintf("Введите число больше 1.")

    }

}

func primeFactors(n int) []int {

    var factors []int

    for n%2 == 0 {

        factors = append(factors, 2)

        n /= 2

    }

    for i := 3; i\*i <= n; i += 2 {

        for n%i == 0 {

            factors = append(factors, i)

            n /= i

        }

    }

    if n > 2 {

        factors = append(factors, n)

    }

    return factors

}

**Изображение выглядит как текст, снимок экрана, астрономия, Вселенная

Автоматически созданное описаниеИзображение выглядит как Вселенная, Космическое пространство, снимок экрана, Астрономический объект

Автоматически созданное описаниеЭкранные формы с примерами выполнения программы**

**Изображение выглядит как текст, снимок экрана

Автоматически созданное описание**

**Изображение выглядит как текст, снимок экрана

Автоматически созданное описание**

**Изображение выглядит как текст, снимок экрана

Автоматически созданное описание**