МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ

ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ

ВЫСШЕГО ОБРАЗОВАНИЯ

«НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Кафедра защиты информации



**ОТЧЁТ**

**по лабораторной работе №2**

**«**Классификация языков программирования**»**

**по дисциплине: «***Программирование***»**

Выполнил:Проверил:

Студент гр. «АБс-324», «АВТФ» *Ассистент ЗИ*

*Петров М. И. Исаев Г. А.*

«\_\_» \_\_\_\_\_\_ 2024 г.«\_\_\_» \_\_\_\_\_\_ 2024 г.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись) (подпись)

Новосибирск 2024

**Цели и задачи работы**: изучение функций ввода-вывода данных, программирования вычисления значения выражения.

**Задание к работе**:

1. Реализовать линейный вычислительный процесс. Самостоятельно решить задачу в соответствии с индивидуальным вариантом. Реализовать представленные задачи на языках программирования JavaScript, Go (Golang), Swift, С#, Rust, C++, Python, Java, Kotlin, Ruby.
2. Реализовать линейный вычислительный процесс любого задания на языке программирования Assembler, Haskell.
3. Представленные задачи можно реализовать на каждом языке в одной программе с последовательным выполнением.

**Задания:**

1. **Вариант 1.** Дана строка S. Необходимо найти ее подстроки, которые удовлетворяют шаблону вида 1[0]1, где [0] – любое количество нулей (минимально - один). В последовательности [0] не может быть никаких других символов, кроме 0.

Пример: str=”101tf11flb10001”. Результат: 101, 10001.

1. **Вариант 1.** Римские цифры. Необходимо написать программу по преобразованию римского числа в натуральное число.

Пример: XV. Результат: 10 + 5 = 15.

1. **Вариант 21.** Вывести номера чисел из последовательности, у которых сумма цифр меньше их произведения (например, у числа 87 сумма цифр равна 15, а произведение 56).

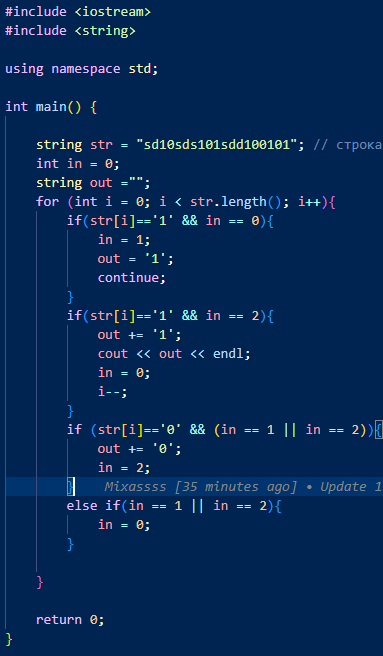
Пример. Вход: 3 27 14 99 Выход: 1 3.

**Методика выполнения работы**:

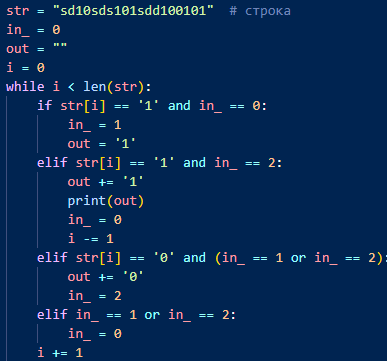
1. Определить типы используемых в программе данных.
2. Описать переменные.
3. Написать функции ввода-вывода.
4. Разработать алгоритм решения задачи по индивидуальному заданию.
5. Написать и отладить программу с вводом-выводом информации
6. Протестировать работу программы на различных исходных данных.
7. Изменить формат вывода, проверить работу программы при другом формате вывода.

**Задание 1. Текст программы с комментариями и вывод:**

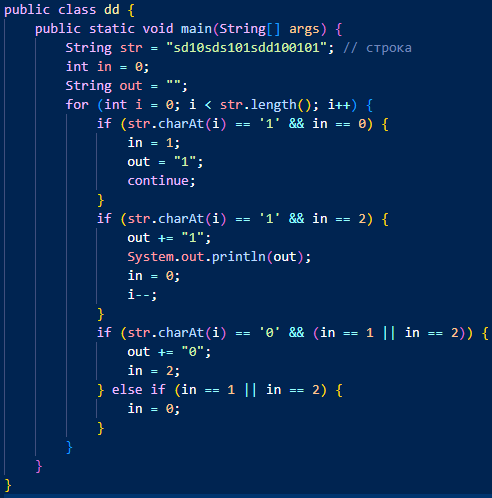
**C++:**



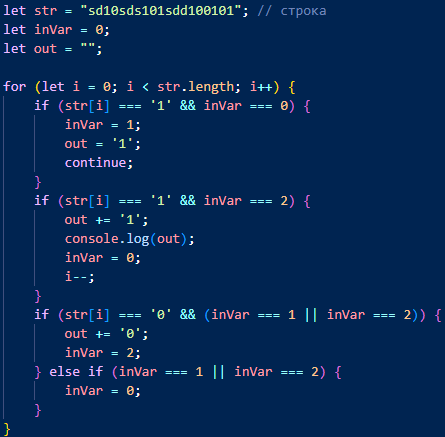
**Python:**



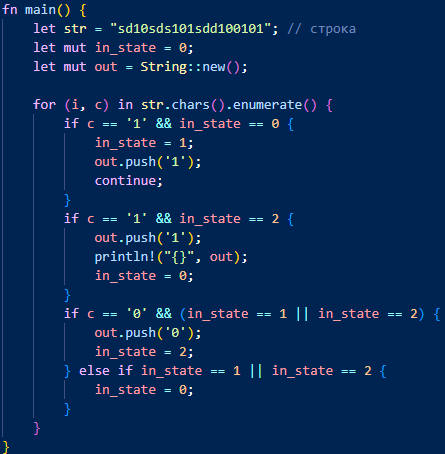
**Java:**



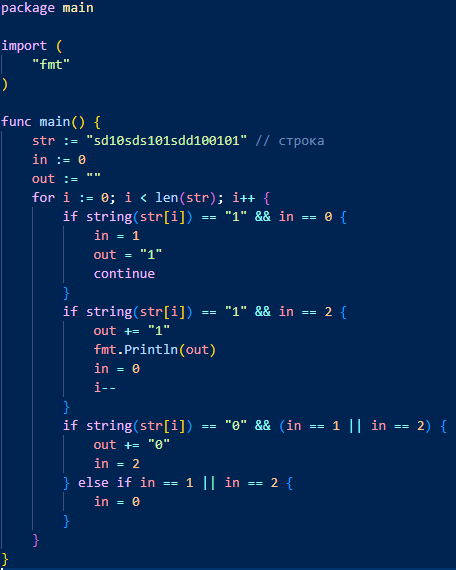
**JavaScript:**



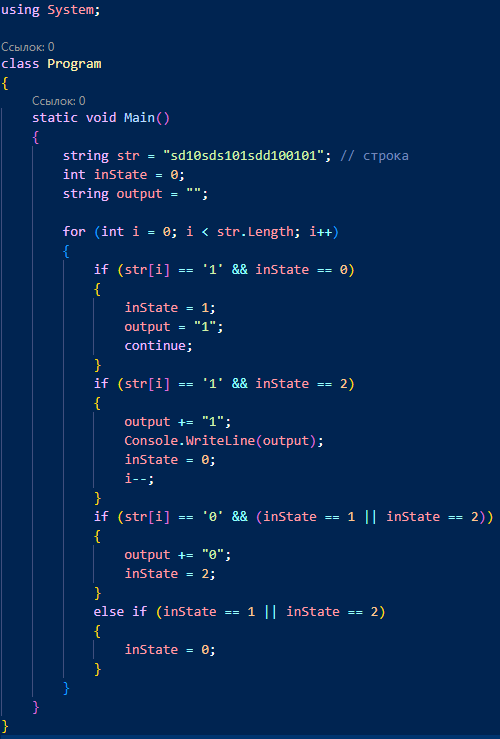
**Rust:**



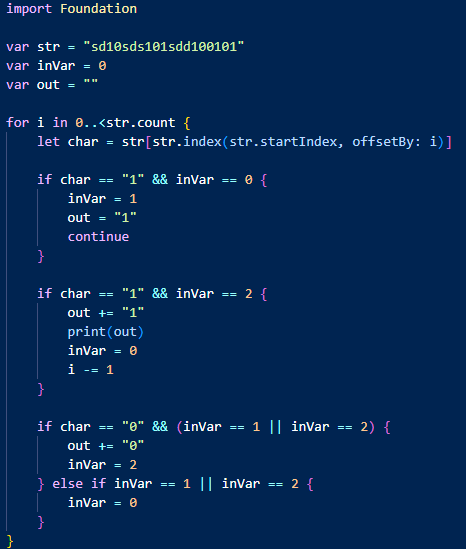
**GO:**



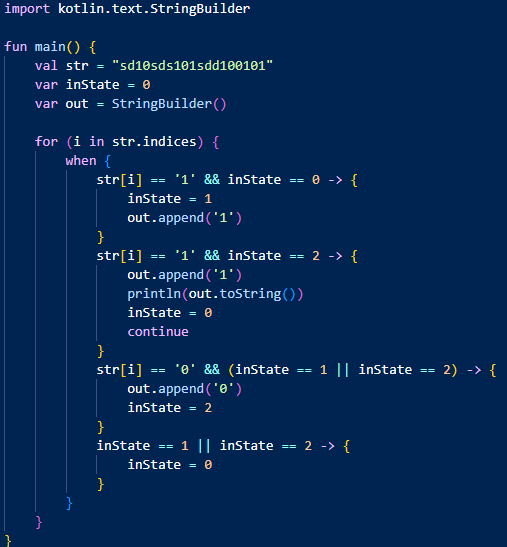
**C#:**



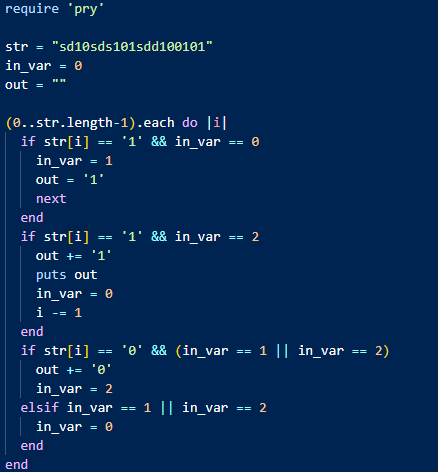
**Swift:**



**Kotlin:**

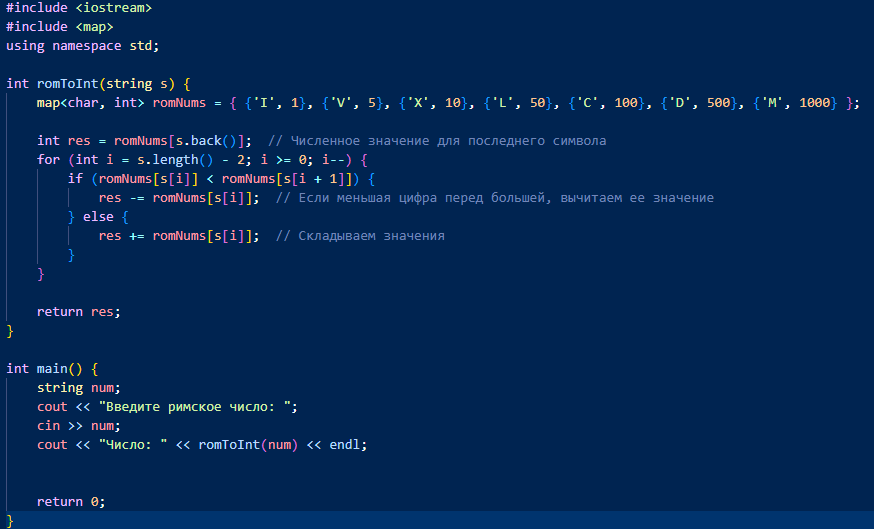


**Ruby:**

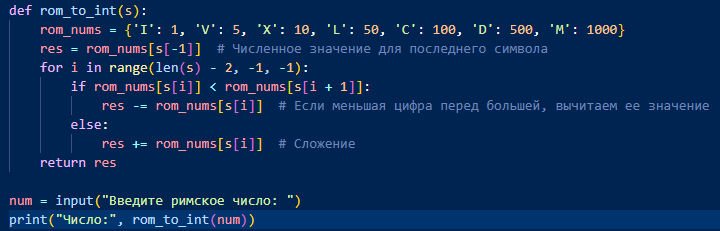


**Задание 2. Текст программы с комментариями и вывод.**

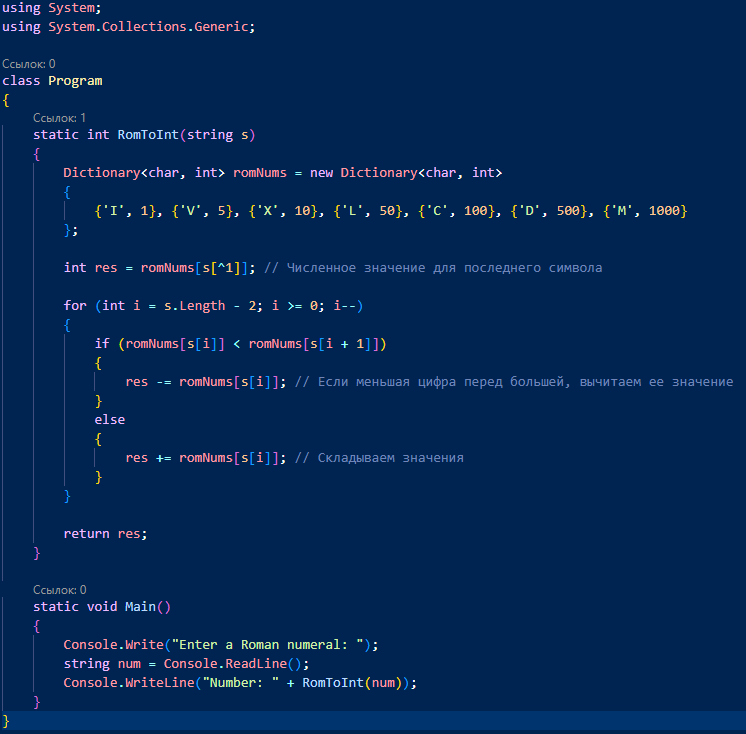
**С++:**



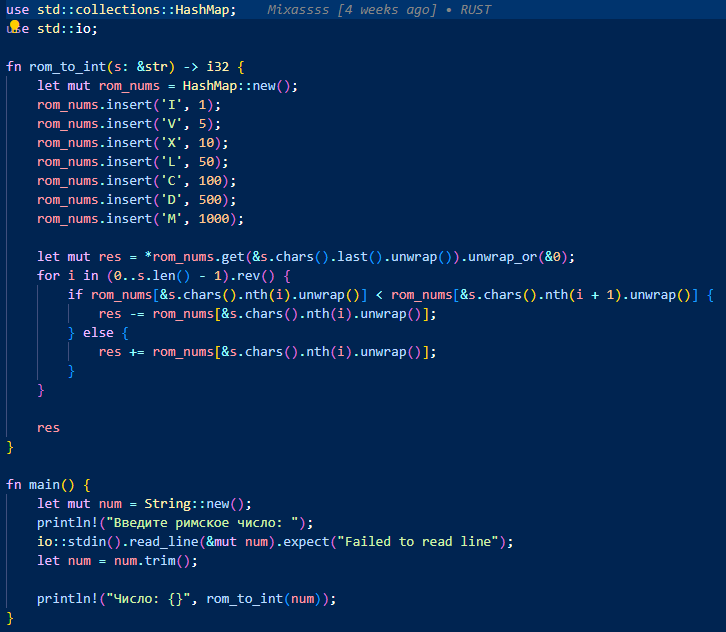
**Python:**



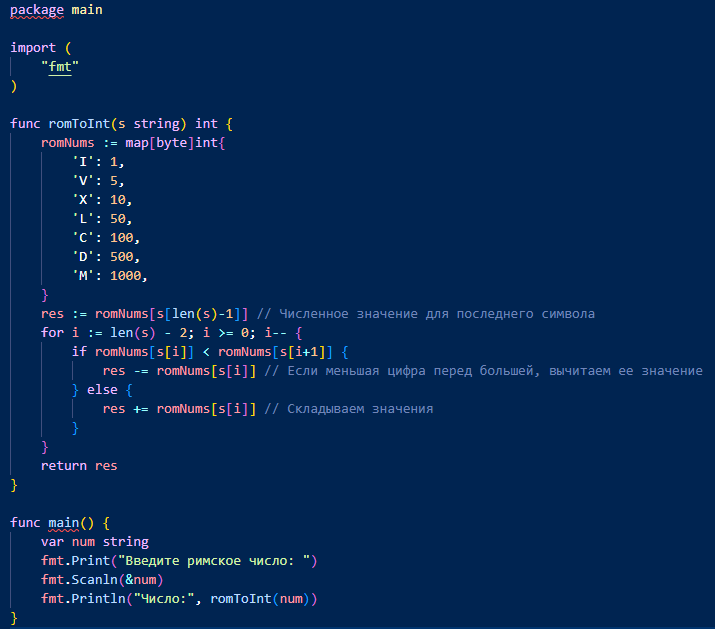
**C#:**



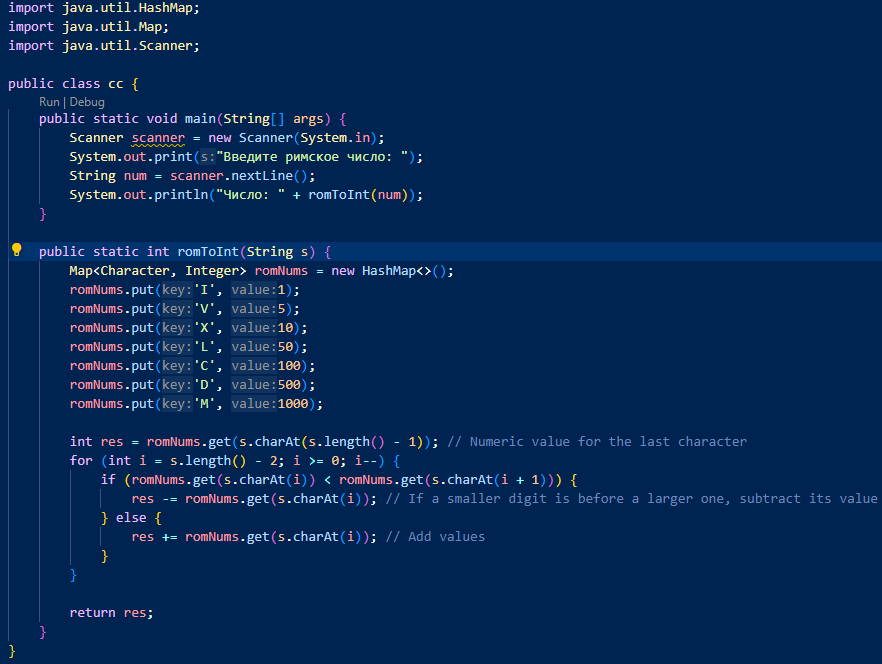
**Rust:**



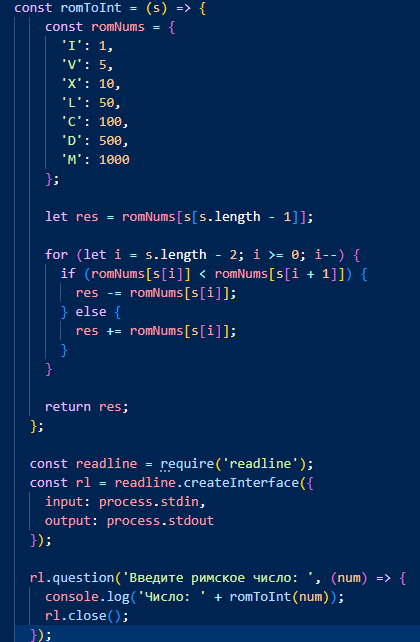
**Go:**



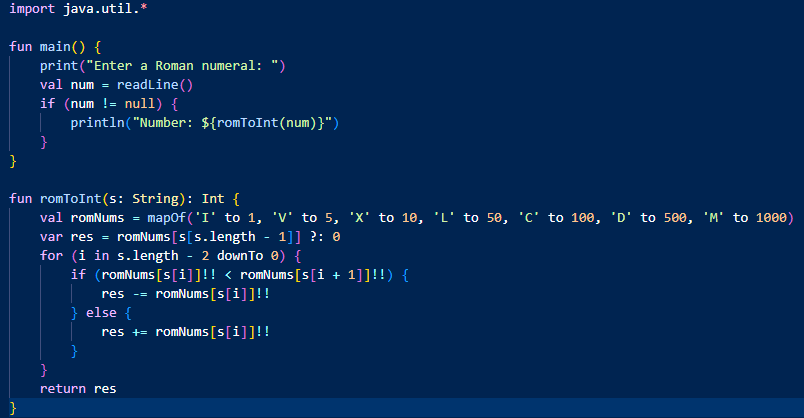
**Java:**



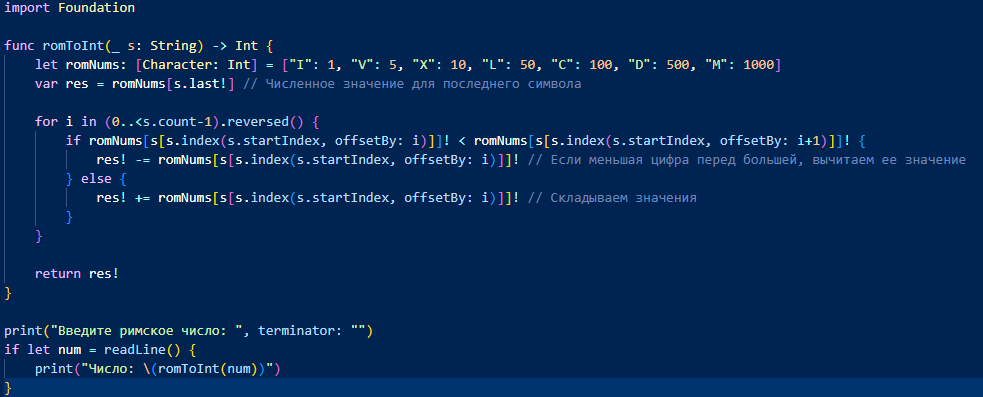
**JavaScript:**



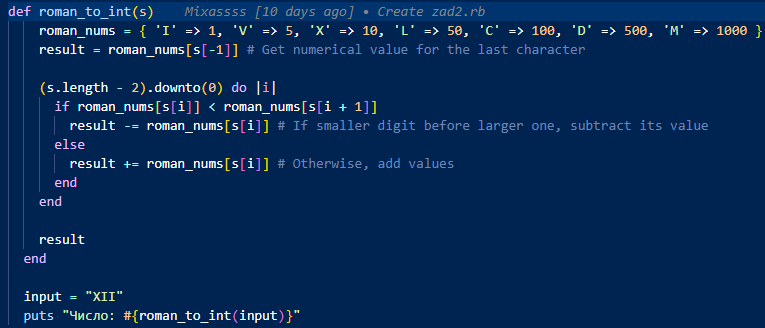
**Kotlin:**



**Swift:**

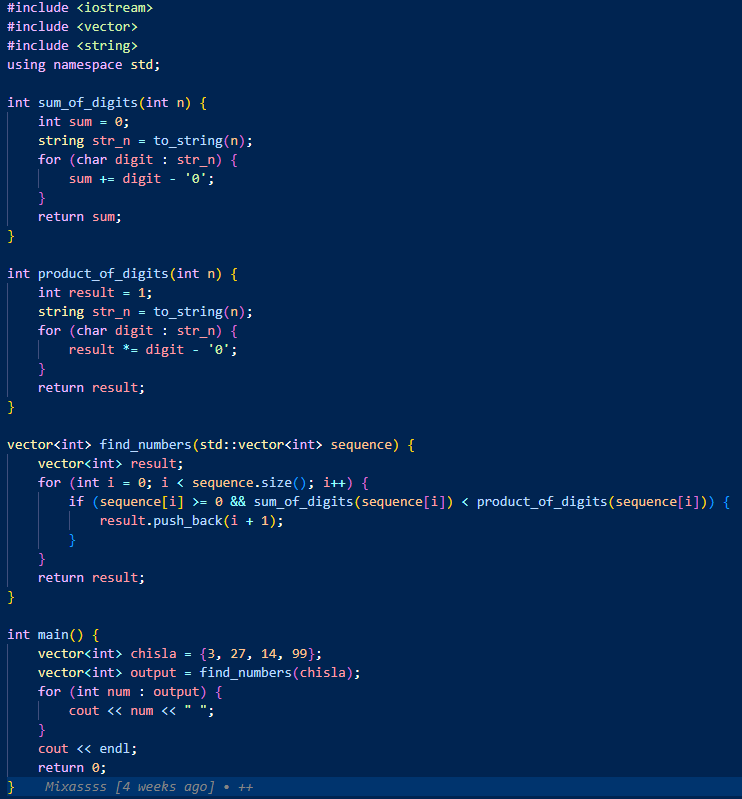


**Ruby:**

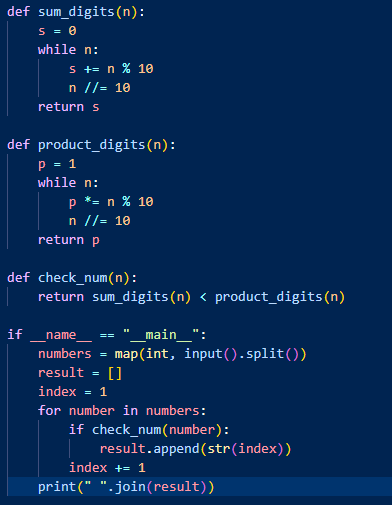


**Задание 3. Текст программы с комментариями и вывод.**

**C++:**



**Python:**



**C#:**

**using System;**

**using System.Collections.Generic;**

**class Program**

**{**

**static int SumDigits(int n)**

**{**

**int s = 0;**

**while (n != 0)**

**{**

**s += n % 10;**

**n /= 10;**

**}**

**return s;**

**}**

**static int ProductDigits(int n)**

**{**

**int p = 1;**

**while (n != 0)**

**{**

**p \*= n % 10;**

**n /= 10;**

**}**

**return p;**

**}**

**static bool CheckNum(int n)**

**{**

**return SumDigits(n) < ProductDigits(n);**

**}**

**static void Main()**

**{**

**string[] numbersStr = Console.ReadLine().Split();**

**List<int> numbers = new List<int>();**

**foreach (string numStr in numbersStr)**

**{**

**numbers.Add(int.Parse(numStr));**

**}**

**List<string> result = new List<string>();**

**int index = 1;**

**foreach (int number in numbers)**

**{**

**if (CheckNum(number))**

**{**

**result.Add(index.ToString());**

**}**

**index++;**

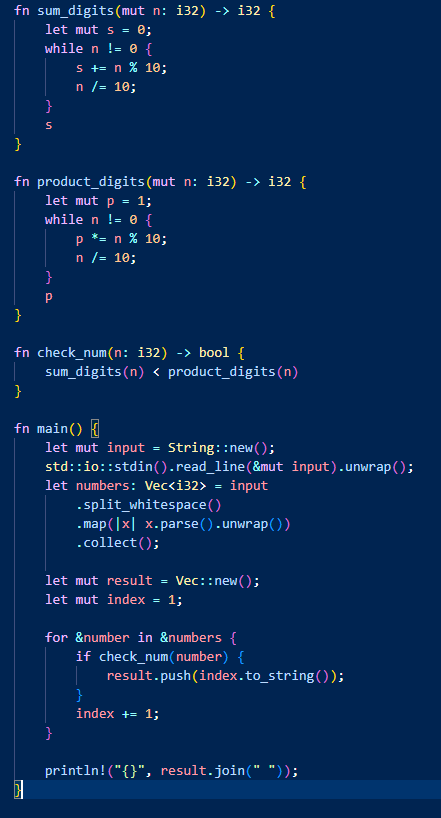
**}**

**Console.WriteLine(string.Join(" ", result));**

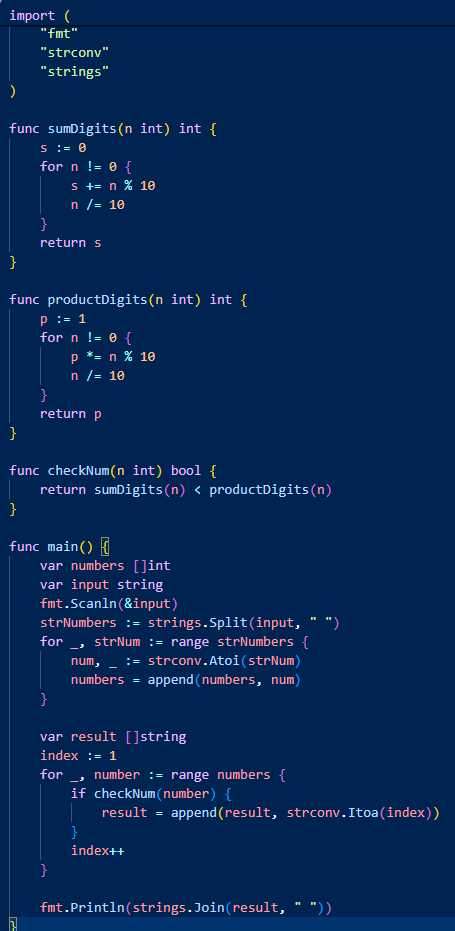
**}**

**}**

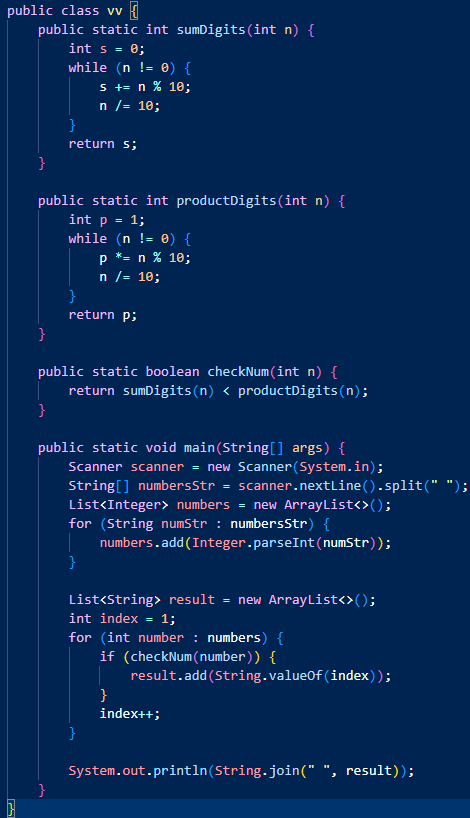
**Rust:**



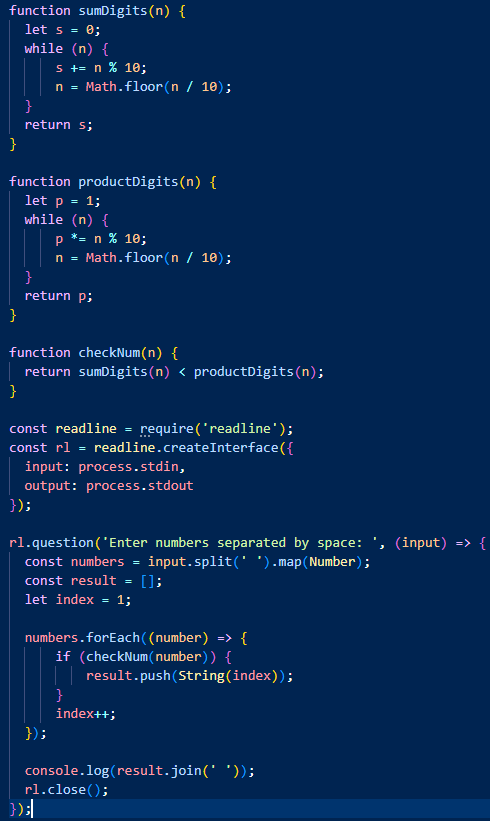
**Go:**



**Java:**



**JavaScript:**



**Kotlin:**

**fun sumDigits(n: Int): Int {**

**var s = 0**

**var num = n**

**while (num != 0) {**

**s += num % 10**

**num /= 10**

**}**

**return s**

**}**

**fun productDigits(n: Int): Int {**

**var p = 1**

**var num = n**

**while (num != 0) {**

**p \*= num % 10**

**num /= 10**

**}**

**return p**

**}**

**fun checkNum(n: Int): Boolean {**

**return sumDigits(n) < productDigits(n)**

**}**

**fun main() {**

**val numbers = readLine()!!.split(" ").map { it.toInt() }**

**val result = mutableListOf<String>()**

**var index = 1**

**for (number in numbers) {**

**if (checkNum(number)) {**

**result.add(index.toString())**

**}**

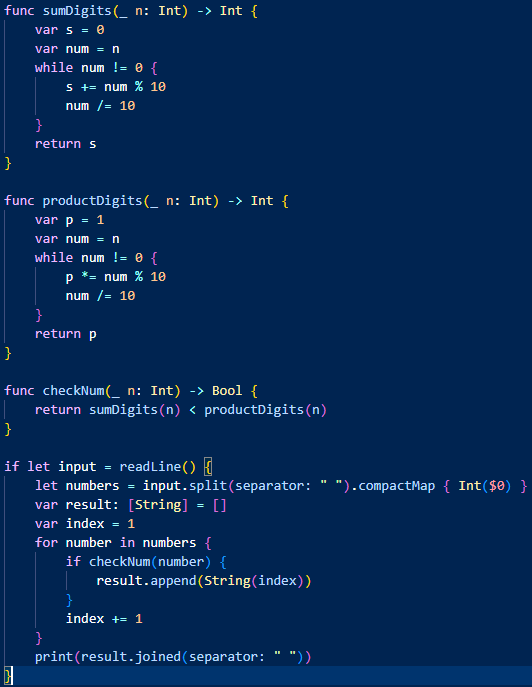
**index++**

**}**

**println(result.joinToString(" "))**

**}**

**Swift:**



**Ruby:**

**def sum\_digits(n)**

**s = 0**

**while n != 0**

**s += n % 10**

**n /= 10**

**end**

**s**

**end**

**def product\_digits(n)**

**p = 1**

**while n != 0**

**p \*= n % 10**

**n /= 10**

**end**

**p**

**end**

**def check\_num(n)**

**sum\_digits(n) < product\_digits(n)**

**end**

**if \_\_FILE\_\_ == $PROGRAM\_NAME**

**numbers = gets.chomp.split.map(&:to\_i)**

**result = []**

**index = 1**

**numbers.each do |number|**

**result << index.to\_s if check\_num(number)**

**index += 1**

**end**

**puts result.join(" ")**

**end**

**Задание 3 на языках Assembly и Haskell.**

**Assembly:**

;sumpro6.asm

.model small

.stack 100h

.data

buffer db 250,0,254 dup(?) ; буфер для ввода строки Первое число = к-во чисел массива

sum dw 0 ; сумма цифр числа

product dw 0 ; произведение цифр числа

num\_count dw 0 ; счетчик чисел

cnt2 dw 0

kol dw 0

num\_list dw 100 dup(?) ; массив для хранения номеров чисел с необходимым условием

prompt db 'Vvedite stroky chisel (razdelennix probelami): ',13,10,'$'

result\_header db 13,10, "Nomera Chisel y kotorix summa men`she proizvedenia: $"

.code

start:

mov ax, @data

mov ds, ax

; выводим приглашение на ввод строки

mov ah, 09h

lea dx, prompt

int 21h

; считываем строку

mov ah, 0Ah

lea dx, buffer

int 21h

; берем первое число = количество чисел в массиве (=kol)

mov si, offset buffer+2 ; указатель на начало введенной строки

xor ax,ax

xor cx,cx

go1:

mov al,[si] ; берем символ из массива

sub al,30h ; перевод в символьный вид

add cx,ax ; в сх будет само число

cmp byte ptr[si+1],20h ; если число однозначное,

je go2 ; то берем его в kol

mov bx,10 ; если число двух-значное

mul bl ; умножаем дестки на 10

mov cx,ax ; и сохраняем в сх произведение

inc si ; берем вторую цифру

jmp go1

go2:

mov kol,cx ; kol = количество чисел в массиве

inc si

; счетчик чисел устанавливаем в 0

mov num\_count, 0

check\_numbers:

; =========== ЦИКЛ =========

check\_next\_number:

mov bx,cnt2

cmp bx,kol

jbe go5

jmp print\_numbers ; цикл закончен. все числа проверены

go5:

mov al, [si] ; берем символ числа

cmp al,13 ; конец строки?

jne go9

jmp print\_numbers ; конец строки

go9: cmp al, 20h ; если это пробел, значит число закончилось

je next\_number ; переходим к следующему числу

push si ; сохраняем текущее значение SI в стеке

mov ah, 0 ; сбрасываем регистр ah перед использованием функции для конвертации

sub al, '0' ; конвертируем символ в число

mov bl, al ; сохраняем число в bl для дальнейших вычислений

mov sum, 0 ; обнуляем сумму цифр числа

calc\_sum:

xor bh,bh

add sum, bx ; добавляем текущую цифру к сумме

inc si ; переходим к следующей цифре

mov al, [si] ; берем следующий символ

cmp al, 20h ; если это пробел, значит число закончилось

je calc\_product ; если конец числа, переходим к вычислению произведения

sub al, '0' ; конвертируем символ в число

mov bl, al ; сохраняем число в bl

jmp calc\_sum ; продолжаем вычисления

calc\_product:

pop si

mov bh,0

mov bl,[si]

sub bl,30h

mov product, 1 ; устанавливаем произведение в 1 перед вычислением

calc\_product\_loop:

mov ax,product

mul bl ; умножаем текущую цифру на произведение

mov product,ax

inc si ; переходим к следующей цифре

mov bl, [si] ; берем следующий символ

cmp bl, 13 ; если это конец строки, значит число закончилось

je compare ; если конец числа, переходим к сравнению

cmp bl,20h

je compare

sub bl, '0' ; конвертируем символ в число

jmp calc\_product\_loop ; продолжаем вычисления

compare:

inc cnt2 ; инкремент количества обработанных чисел

cmp sum, ax ; bx product сравниваем сумму и произведение

jge next\_number ; если сумма не меньше произведения, переходим к следующему числу

; если сумма меньше произведения, добавляем номер числа в массив

mov di, num\_count

shl di,1

mov bx,cnt2

mov num\_list[di],bx

inc num\_count

next\_number:

inc si ; переходим к следующему числу

jmp check\_next\_number ; проверяем следующее число

print\_numbers:

lea dx,result\_header

mov ah,9

int 21h

; выводим номера чисел с необходимым условием

mov cx, num\_count

cmp cx,0

je exi

mov si, offset num\_list

print\_loop:

mov ax, [si]

aam

add ax, 3030h

mov bx, ax

mov ah, 02

mov dl, bh

int 21h

mov dl, bl

int 21h

mov dl,20h

mov ah,2

int 21h

add si,2

loop print\_loop

exi:

end\_program:

mov ax, 4C00h ; завершаем программу

int 21h

end start

**Haskell:**

сумма\_цифр :: Int -> Int

сумма\_цифр n = go n 0

where

go 0 acc = acc

go n acc = go (n `div` 10) (acc + n `mod` 10)

произведение\_цифр :: Int -> Int

произведение\_цифр n = go n 1

where

go 0 acc = acc

go n acc = go (n `div` 10) (acc \* (n `mod` 10))

проверка\_числа :: Int -> Bool

проверка\_числа n = сумма\_цифр n < произведение\_цифр n

main :: IO ()

main = do

let числа = [3, 27, 14, 99]

let результаты = filter (\число -> проверка\_числа число) числа

mapM\_ (\(номер, \_) -> putStr (show номер ++ " ")) (zip [1..] результаты)

**Вывод:**

В ходе выполнения лабораторной работы я узнал о множестве других языков программирования, получил по ним базовые знания и навыки в работе. Также научился читать написанные программы на данных языках, что в будущем однозначно мне пригодится.

**Ссылка на репозиторий: https://github.com/Mixassss/laba-2.git**