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*

*Клаус В. А. 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.
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 11.** Вывести все числа, у которых сумма цифр состоит из одного десятичного разряда.

Пример: Input: 19 93 44 29 | Output: 44

**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++:**

// регулярные выражения

#include <iostream>

#include <regex>

using namespace std;

int main() {

string str = "1001tf11flb10001"; // строка

regex regular("1(0+)1"); // условие поиска

smatch result; // результат

while (regex\_search(str, result, regular)) {

cout << result[0] << endl; // возвращает первое совпадение с регулярным выражением

str = result.suffix().str(); // возвращает оставшуюся часть строки после последнего соответствия регулярному выражению

}

return 0;

}

OUTPUT: 1001

10001

**Python:**

import re

str = "1001tf11flb10001" # строка

regular = "1[0]+1" # условие поиска

result = re.finditer(regular, str) # поиск условия и присваивание

for match in result: # вывод найденных подстрок

print (match.group(0))

OUTPUT: 1001

10001

**Java:**

import java.util.regex.Matcher;

import java.util.regex.Pattern;

public class zad1 {

public static void main(String[] args) {

String str = "1001tf11flb10001"; // строка

Pattern pattern = Pattern.compile("1(0+)1"); // условие поиска

Matcher matcher = pattern.matcher(str); // результат

while (matcher.find()) {

System.out.println(matcher.group()); // возвращает первое совпадение с регулярным выражением

str = str.substring(matcher.end()); // возвращает оставшуюся часть строки после последнего соответствия регулярному выражению

matcher = pattern.matcher(str);

}

}

}

OUTPUT: 1001

10001

**JavaScript:**

const str = "1001tf11flb10001"; // строка

const regular = new RegExp("1(0+)1", "g"); // условие поиска

const result = str.match(regular); // результат

result.forEach(result => { // поиск совпадения

console.log(result); // вывод

});

OUTPUT: 1001

10001

**C#:**

using System.Text.RegularExpressions;

class Program {

static void Main() {

string str = "1001tf11flb10001"; // строка

Regex regular = new Regex("1(0+)1"); // условие поиска

MatchCollection result = regular.Matches(str); // результат

foreach (Match match in result) {

Console.WriteLine(match.Value); // возвращает совпадение с регулярным выражением

}

}

}

OUTPUT: 1001

10001

**GO:**

package main

import (

"fmt" // пакет для форматирования и вывода

"regexp" // регулярное выражение

)

func main() {

str := "1001tf11flb10001" // строка

regular := regexp.MustCompile("1(0+)1") // условие поиска

result := regular.FindAllString(str, -1) // результат

for \_, match := range result {

fmt.Println(match) // возвращает совпадение с регулярным выражением

}

}

OUTPUT: 1001

10001

**Rust:**

use regex::Regex;

fn main() {

let str = "1001tf11flb10001"; // строка

let regular = Regex::new(r"1[0]+1").unwrap(); // условие поиска

for mat in regular.find\_iter(str) { // вывод найденных подстрок

println!("{}", mat.as\_str());

}

}

OUTPUT: 1001

10001

**Swift:**

import Foundation

let str = "1001tf11flb10001" // строка

let regular = try! NSRegularExpression(pattern: "1(0+)1") // условие поиска

let range = NSRange(str.startIndex..<str.endIndex, in: str)

var results = [String]()

regular.enumerateMatches(in: str, options: [], range: range) { (match, \_, \_) in

if let match = match {

let matchedString = String(str[Range(match.range, in: str)!])

results.append(matchedString)

}

}

for result in results {

print(result) // возвращает первое совпадение с регулярным выражением

}

OUTPUT: 1001

10001

**Kotlin:**

import java.util.regex.Matcher

import java.util.regex.Pattern

fun main() {

val str = "1001tf11flb10001" // строка

val regular = Pattern.compile("1(0+)1") // условие поиска

val matcher = regular.matcher(str) // результат

while (matcher.find()) {

println(matcher.group(0)) // возвращает первое совпадение с регулярным выражением

}

}

OUTPUT: 1001

10001

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

**С++:**

#include <iostream>

#include <map>

using namespace std;

int romanToInt(string s) {

map<char, int> romanNums = { {'I', 1}, {'V', 5}, {'X', 10}, {'L', 50}, {'C', 100}, {'D', 500}, {'M', 1000} };

int result = romanNums[s.back()]; // Получаем численное значение для последнего символа

for (int i = s.length() - 2; i >= 0; i--) {

if (romanNums[s[i]] < romanNums[s[i + 1]]) {

result -= romanNums[s[i]]; // Если меньшая цифра перед большей, вычитаем ее значение

} else {

result += romanNums[s[i]]; // Иначе складываем значения

}

}

return result;

}

int main() {

string input = "XII";

cout << "Число: " << romanToInt(input) << endl;

return 0;

}

OUTPUT: Число: 12

**Python:**

def romanToInt(s):

romanNums = {'I': 1, 'V': 5, 'X': 10, 'L': 50, 'C': 100, 'D': 500, 'M': 1000}

result = romanNums[s[-1]] # получаем числовое значение для последнего символа

for i in range(len(s) - 2, -1, -1):

if romanNums[s[i]] < romanNums[s[i + 1]]:

result -= romanNums[s[i]] # если меньшая цифра стоит перед большей, вычитаем

else:

result += romanNums[s[i]] # в противном случае, складываем

return result

input = "XII"

print("Число:", romanToInt(input))

OUTPUT: Число: 12

**C#:**

using System;

class Program {

static int RomanToInt(string s) {

Dictionary<char, int> romanNums = new Dictionary<char, int> {

{'I', 1}, {'V', 5}, {'X', 10}, {'L', 50}, {'C', 100}, {'D', 500}, {'M', 1000}

};

int result = romanNums[s[^1]]; // Получаем численное значение для последнего символа

for (int i = s.Length - 2; i >= 0; i--) {

if (romanNums[s[i]] < romanNums[s[i + 1]]) {

result -= romanNums[s[i]]; // Если меньшая цифра перед большей, вычитаем ее значение

} else {

result += romanNums[s[i]]; // Иначе складываем значения

}

}

return result;

}

static void Main() {

string input = "XII";

Console.WriteLine("Число: " + RomanToInt(input));

}

}

OUTPUT: Число: 12

**Rust:**

use std::collections::HashMap;

fn roman\_to\_int(s: &str) -> i32 {

let mut roman\_nums = HashMap::new();

roman\_nums.insert('I', 1);

roman\_nums.insert('V', 5);

roman\_nums.insert('X', 10);

roman\_nums.insert('L', 50);

roman\_nums.insert('C', 100);

roman\_nums.insert('D', 500);

roman\_nums.insert('M', 1000);

let mut result = \*roman\_nums.get(&s.chars().last().unwrap()).unwrap(); // Получаем численное значение для последнего символа

for i in (0..s.len() - 1).rev() {

if roman\_nums[&s.chars().nth(i).unwrap()] < roman\_nums[&s.chars().nth(i + 1).unwrap()] {

result -= roman\_nums[&s.chars().nth(i).unwrap()]; // Если меньшая цифра перед большей, вычитаем ее значение

} else {

result += roman\_nums[&s.chars().nth(i).unwrap()]; // Иначе складываем значения

}

}

result

}

fn main() {

let input = "XII";

println!("Число: {}", roman\_to\_int(input));

}

OUTPUT: Число: 12

**Go:**

package main

import (

"fmt"

)

func romanToInt(s string) int {

romanNums := map[byte]int{'I': 1, 'V': 5, 'X': 10, 'L': 50, 'C': 100, 'D': 500, 'M': 1000}

result := romanNums[s[len(s)-1]] // Получаем численное значение для последнего символа

for i := len(s) - 2; i >= 0; i-- {

if romanNums[s[i]] < romanNums[s[i+1]] {

result -= romanNums[s[i]] // Если меньшая цифра перед большей, вычитаем ее значение

} else {

result += romanNums[s[i]] // Иначе складываем значения

}

}

return result

}

func main() {

input := "XII"

fmt.Println("Число:", romanToInt(input))

}

OUTPUT: Число: 12

**Java:**

import java.util.HashMap;

import java.util.Map;

public class zad2 {

public static void main(String[] args) {

String input = "XII";

System.out.println("Число: " + romanToInt(input));

}

public static int romanToInt(String s) {

Map<Character, Integer> romanNums = new HashMap<>();

romanNums.put('I', 1);

romanNums.put('V', 5);

romanNums.put('X', 10);

romanNums.put('L', 50);

romanNums.put('C', 100);

romanNums.put('D', 500);

romanNums.put('M', 1000);

int result = romanNums.get(s.charAt(s.length() - 1)); // Получаем численное значение для последнего символа

for (int i = s.length() - 2; i >= 0; i--) {

if (romanNums.get(s.charAt(i)) < romanNums.get(s.charAt(i + 1))) {

result -= romanNums.get(s.charAt(i)); // Если меньшая цифра перед большей, вычитаем ее значение

} else {

result += romanNums.get(s.charAt(i)); // Иначе складываем значения

}

}

return result;

}

}

OUTPUT: Число: 12

**JavaScript:**

function romanToInt(s) {

const romanNums = {

'I': 1,

'V': 5,

'X': 10,

'L': 50,

'C': 100,

'D': 500,

'M': 1000

};

let result = romanNums[s[s.length - 1]]; // Получаем численное значение для последнего символа

for (let i = s.length - 2; i >= 0; i--) {

if (romanNums[s[i]] < romanNums[s[i + 1]]) {

result -= romanNums[s[i]]; // Если меньшая цифра перед большей, вычитаем ее значение

} else {

result += romanNums[s[i]]; // Иначе складываем значения

}

}

return result;

}

const input = "XII";

console.log("Число: " + romanToInt(input));

OUTPUT: Число: 12

**Kotlin:**

import java.util.\*

fun romanToInt(s: String): Int {

val romanNums = mapOf('I' to 1, 'V' to 5, 'X' to 10, 'L' to 50, 'C' to 100, 'D' to 500, 'M' to 1000)

var result = romanNums[s.last()] ?: 0 // Получаем численное значение для последнего символа

for (i in s.length - 2 downTo 0) {

if (romanNums[s[i]]!! < romanNums[s[i + 1]]!!) {

result -= romanNums[s[i]]!! // Если меньшая цифра перед большей, вычитаем ее значение

} else {

result += romanNums[s[i]]!! // Иначе складываем значения

}

}

return result

}

fun main() {

val input = "XII"

println("Число: ${romanToInt(input)}")

}

OUTPUT: Число: 12

**Swift:**

import Foundation

func romanToInt(\_ s: String) -> Int {

let romanNums: [Character: Int] = ["I": 1, "V": 5, "X": 10, "L": 50, "C": 100, "D": 500, "M": 1000]

var result = romanNums[s.last!] ?? 0 // Получаем численное значение для последнего символа

for i in (0..<s.count-1).reversed() {

if let currentNum = romanNums[s[s.index(s.startIndex, offsetBy: i)]], let nextNum = romanNums[s[s.index(s.startIndex, offsetBy: i+1)]] {

if currentNum < nextNum {

result -= currentNum // Если меньшая цифра перед большей, вычитаем ее значение

} else {

result += currentNum // Иначе складываем значения

}

}

}

return result

}

let input = "XII"

print("Число: \(romanToInt(input))")

OUTPUT: Число: 12

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

**C++:**

#include <iostream>

#include <vector>

using namespace std;

int sum\_of\_digits(int num) { // нахождение суммы цифр числа

int sum = 0;

while (num != 0) {

sum += num % 10;

num /= 10;

}

return sum;

}

int main() {

// ввод чисел

int n;

do {

cout << "Введите кол-во чисел(Больше 0): ";

cin >> n;

} while(n<=0);

vector<double> nums(n);

cout << "Введите числа: ";

for (int i = 0; i < n; ++i) {

cin >> nums[i];

}

cout << "Числа: ";

for (int i = 0; i < nums.size(); ++i) { // перебор каждого числа

if (sum\_of\_digits(nums[i]) < 10) { // проверка условия

cout << nums[i] << " "; // вывод подходящих чисел

}

}

return 0;

}

**Python:**

def sum\_of\_digits(num):

# нахождение суммы цифр числа

sum = 0

while num != 0:

sum += num % 10

num //= 10

return sum

# ввод чисел

n = 0

while n <= 0:

n = int(input("Введите кол-во чисел(Больше 0): "))

nums = []

print("Введите числа:")

for i in range(n):

nums.append(float(input()))

print("Числа:")

for num in nums:

if sum\_of\_digits(num) < 10:

print(num, end=" ")

**C#:**

using System;

class Program {

static int SumOfDigits(int num) { // нахождение суммы чисел

int sum = 0;

while (num != 0) {

sum += num % 10;

num /= 10;

}

return sum;

}

static void Main() {

// ввод чисел

int n;

do {

Console.Write("Введите кол-во чисел(Больше 0): ");

} while (!int.TryParse(Console.ReadLine(), out n) || n <= 0);

List<double> nums = new List<double>(n);

Console.Write("Введите числа: ");

for (int i = 0; i < n; ++i) {

nums.Add(double.Parse(Console.ReadLine()));

}

Console.Write("Числа: ");

foreach (var num in nums) { // перебор каждого числа

if (SumOfDigits((int)num) < 10) { // проверка условия

Console.Write(num + " "); // вывод подходящих чисел

}

}

}

}

**Rust:**

use std::io;

use std::io::Write;

fn sum\_of\_digits(mut num: i32) -> i32 { // нахождение суммы цифр

let mut sum = 0;

while num != 0 {

sum += num % 10;

num /= 10;

}

sum

}

fn main() {

// ввод чисел

let mut n = String::new();

loop {

print!("Введите кол-во чисел (Больше 0): ");

io::stdout().flush().unwrap();

n.clear();

io::stdin().read\_line(&mut n).unwrap();

let n: i32 = match n.trim().parse() {

Ok(num) if num > 0 => num,

\_ => continue,

};

let mut nums = Vec::with\_capacity(n as usize);

print!("Введите числа: ");

io::stdout().flush().unwrap();

for \_ in 0..n {

let mut num = String::new();

io::stdin().read\_line(&mut num).unwrap();

let num: f64 = num.trim().parse().unwrap();

nums.push(num);

}

print!("Числа: ");

for num in &nums { // перебор чисел

if sum\_of\_digits(\*num as i32) < 10 { // проверка условия

print!("{} ", num); // ввод подходящих чисел

}

}

break;

}

}

**Go:**

package main

import (

"fmt"

)

func sumOfDigits(num int) int { // нахождение суммы цифр числа

sum := 0

for num != 0 {

sum += num % 10

num /= 10

}

return sum

}

func main() {

// ввод чисел

var n int

for {

fmt.Print("Введите кол-во чисел (Больше 0): ")

fmt.Scan(&n)

if n > 0 {

break

}

}

nums := make([]int, n)

fmt.Print("Введите числа: ")

for i := 0; i < n; i++ {

fmt.Scan(&nums[i])

}

fmt.Print("Числа: ")

for i := 0; i < len(nums); i++ { // перебор каждого числа

if sumOfDigits(nums[i]) < 10 { // проверка условия

fmt.Print(nums[i], " ") // вывод подходящих чисел

}

}

}

**Java:**

import java.util.Scanner;

import java.util.ArrayList;

public class zad3 {

public static int sumOfDigits(Double double1) { // нахождение суммы цифр числа

int sum = 0;

while (double1 != 0) {

sum += double1 % 10;

double1 /= 10;

}

return sum;

}

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

// ввод чисел

int n;

do {

System.out.print("Введите кол-во чисел(Больше 0): ");

n = scanner.nextInt();

} while (n <= 0);

ArrayList<Double> nums = new ArrayList<>();

System.out.print("Введите числа: ");

for (int i = 0; i < n; ++i) {

nums.add(scanner.nextDouble());

}

System.out.print("Числа: ");

for (int i = 0; i < nums.size(); ++i) { // перебор каждого числа

if (sumOfDigits(nums.get(i)) < 10) { // проверка условия

System.out.print(nums.get(i) + " "); // вывод подходящих чисел

}

}

scanner.close();

}

}

**JavaScript:**

function sum\_of\_digits(num) { // нахождение суммы цифр

let sum = 0;

while (num !== 0) {

sum += num % 10;

num = Math.floor(num / 10);

}

return sum;

}

function main() {

const readline = require('readline');

const rl = readline.createInterface({

input: process.stdin,

output: process.stdout

});

// ввод чисел

rl.question("Введите кол-во чисел(Больше 0): ", function(n) {

n = parseInt(n);

if (n <= 0) {

console.log("Число должно быть больше 0.");

rl.close();

return;

}

let nums = [];

rl.question("Введите числа: ", function(input) {

nums = input.split(' ').map(Number);

console.log("Числа: ");

for (let i = 0; i < nums.length; ++i) { // перебор чисел

if (sum\_of\_digits(nums[i]) < 10) { // проверка условия

process.stdout.write(nums[i] + " "); // вывод подходящих

}

}

rl.close();

});

});

}

main();

**Kotlin:**

import java.util.Scanner

fun sumOfDigits(num: Int): Int { // нахождение суммы цифр

var sum = 0

var number = num

while (number != 0) {

sum += number % 10

number /= 10

}

return sum

}

fun main() {

val scanner = Scanner(System.`in`)

// ввод чисел

var n: Int

do {

print("Введите кол-во чисел(Больше 0): ")

n = scanner.nextInt()

} while (n <= 0)

val nums = DoubleArray(n)

print("Введите числа: ")

for (i in 0 until n) {

nums[i] = scanner.nextDouble()

}

print("Числа: ")

for (i in nums.indices) { // перебор

if (sumOfDigits(nums[i].toInt()) < 10) { // проверка условия

print("${nums[i]} ") // вывод подходящих

}

}

}

**Swift:**

import Foundation

func sumOfDigits(\_ num: Int) -> Int { // нахождение суммы цифр

var sum = 0

var number = num

while number != 0 {

sum += number % 10

number /= 10

}

return sum

}

func main() {

// ввод чисел

var n = 0

repeat {

print("Введите кол-во чисел(Больше 0): ", terminator: "")

if let input = readLine(), let inputInt = Int(input) {

n = inputInt

}

} while n <= 0

var nums = [Double]()

print("Введите числа: ", terminator: "")

for \_ in 0..<n {

if let input = readLine(), let inputDouble = Double(input) {

nums.append(inputDouble)

}

}

print("Числа: ", terminator: "")

for num in nums { // перебор

if sumOfDigits(Int(num)) < 10 { // проверка условия

print("\(num) ", terminator: "") // вывод подходящих

}

}

}

main()

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

**Assembly:**

;------------------------------------------ОБЪЯВЛЕНИЕ ЭКСТЕРНОВ-----------------------------------;

extern scanf ;

extern printf ;

;

;-------------------------------------------ВЫДЕЛЕНИЕ ПАМЯТИ--------------------------------------;

section .data ;

frmt db "%d", 0 ;

frmt2 db 0xA, "%d", 0 ;

num1 dd 0 ;

num2 dd 0 ;

num3 dd 0 ;

num4 dd 0 ;

MSG\_1 db 0x17, "Type 1-st num: " ;

MSG\_2 db 0x17, "Type 2-nd num: " ;

MSG\_3 db 0x17, "Type 3-rd num: " ;

MSG\_4 db 0x17, "Type 4-th num: " ;

MSG\_5 db 0x10, "EXIT CAUSE ERROR " ;

section .bss ;

sum resd 1 ;

;

;-------------------------------------------------ФУНКЦИИ-----------------------------------------;

section .text ;

global main ;

;

sum\_of\_digits: ;

mov eax, edi ; Перемещение аргумента NUM в EAX ;

xor ecx, ecx ; Инициализируем счётчик суммы ;

loop\_digits: ;

test eax, eax ; проверка, равно ли это число нулю ;

jz end\_sum\_of\_digits ; Если ноль, выход из цикла ;

mov edx, 0 ; Очистить EDX для разделения ;

mov ebx, 10 ; Делитель ;

div ebx ; Разделить EAX на 10 ;

add ecx, edx ; Прибавить остаток к сумме ;

jmp loop\_digits ; Повторяем цикл ;

end\_sum\_of\_digits: ;

mov eax, ecx ; Перемещаем сумму в EAX ;

ret ; Возврат из функции ;

;

main: ;

;---------------------------------------------MSG: Type num1 -------------------------------------;

mov rax, 1 ;

mov rdi, 1 ;

mov rsi, MSG\_1 ;

mov rdx, 17 ;

syscall ;

;

;------------------------------------------------READ: num1---------------------------------------;

push rbp ;

lea rdi, [frmt] ; Формат загрузки ;

lea rsi, [num1] ;

call scanf ;

pop rbp ; Восстанавливаем стек ;

;

;-----0---------------------------------------MSG: Type num2 -------------------------------------;

mov rax, 1 ;

mov rdi, 1 ;

mov rsi, MSG\_2 ;

mov rdx, 17 ;

syscall ;

;

;-----------------------------------------------READ: num2----------------------------------------;

push rbp ;

lea rdi, [frmt] ; Формат загрузки ;

lea rsi, [num2] ;

call scanf ;

pop rbp ; Восстанавливаем стек ;

;

;---0-----------------------------------------MSG: Type num3 -------------------------------------;

mov rax, 1 ;

mov rdi, 1 ;

mov rsi, MSG\_3 ;

mov rdx, 17 ;

syscall ;

;

;-----------------------------------------------READ: num3----------------------------------------;

push rbp ;

lea rdi, [frmt] ; Формат загрузки ;

lea rsi, [num3] ;

call scanf ;

pop rbp ; Восстанавливаем стек ;

;

;---------------------------------------------MSG: Type num4 -------------------------------------;

mov rax, 1 ;

mov rdi, 1 ;

mov rsi, MSG\_4 ;

mov rdx, 17 ;

syscall ;

;

;-----------------------------------------------READ: num4----------------------------------------;

push rbp ;

lea rdi, [frmt] ; Формат загрузки ;

lea rsi, [num4] ;

call scanf ;

pop rbp ; Восстанавливаем стек ;

;

;-------------------------------------ОБРАБОТКА И ВЫВОД СУММЫ-------------------------------------;

mov eax, [num1] ;

mov edi, eax ; Передаём num1 в качестве аргумента в sum\_of\_digits ;

call sum\_of\_digits ;

cmp eax, 0x9 ; Проверяем условие суммы ;

jbe .print\_num1 ; Если сумма <= 9, то вывод ;

jmp .check\_num2 ; Иначе, переходим к следующему ;

;

.print\_num1: ;

push rbp ;

mov rdi, frmt2 ;

mov esi, [num1] ;

xor rax, rax ;

call printf ;

pop rbp ;

;

.check\_num2: ;

mov eax, [num2] ;

mov edi, eax ; Передаём num2 в качестве аргумента в sum\_of\_digits ;

call sum\_of\_digits ;

cmp eax, 0x9 ; Проверяем условие суммы ;

jbe .print\_num2 ; Если сумма <= 9, то вывод ;

jmp .check\_num3 ; Иначе, переходим к следующему ;

;

.print\_num2: ;

push rbp ;

mov rdi, frmt2 ;

mov esi, [num2] ;

xor rax, rax ;

call printf ;

pop rbp ;

;

.check\_num3: ;

mov eax, [num3] ;

mov edi, eax ; Передаём num3 в качестве аргумента в sum\_of\_digits ;

call sum\_of\_digits ;

cmp eax, 0x9 ; Проверяем условие суммы ;

jbe .print\_num3 ; Если сумма <= 9, то вывод ;

jmp .check\_num4 ; Иначе, переходим к следующему ;

;

.print\_num3: ;

push rbp ;

mov rdi, frmt2 ;

mov esi, [num3] ;

xor rax, rax ;

call printf ;

pop rbp ;

;

.check\_num4: ;

mov eax, [num4] ;

mov edi, eax ; Передаём num4 в качестве аргумента в sum\_of\_digits ;

call sum\_of\_digits ;

cmp eax, 0x9 ; Проверяем условие суммы ;

jbe .print\_num4 ; Если сумма <= 9, то вывод ;

jmp .exit\_program ; Иначе, переходим к следующему ;

;

.print\_num4: ;

push rbp ;

mov rdi, frmt2 ;

mov esi, [num4] ;

xor rax, rax ;

call printf ;

pop rbp ;

;

.exit\_program: ;

; имитирует return 0 ;

mov rax, 0 ;

ret ;

;-----------------------------------------!!!!!!WIN!!!!!------------------------------------------;

**Haskell:**

**import Data.Char**

**sumOfDigits :: Int -> Int**

**sumOfDigits num = sum $ map digitToInt $ show num**

**main :: IO ()**

**main = do**

**let nums = [92, 91, 44, 17]**

**putStr "Числа: "**

**mapM\_ (\num -> if sumOfDigits num < 10 then putStr (show num ++ " ") else return ()) nums**

**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/JamesSevil/laba2