МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ государственное БЮДЖЕТНОЕ

образовательное учреждение

высшего образования

«НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

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

Кафедра защиты информации



**ЛАБОРАТОРНАЯ РАБОТА №2**

**«Классификация языков программирования»**

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

Выполнил:Проверил:

Студент гр. «АБ-420», «АВТФ» *Ассистент кафедры ЗИ*

*Твердохлеб Андрей Сергеевич Исаев Г. А.*

«\_\_» \_\_\_\_\_\_ 2025г«\_\_\_» \_\_\_\_\_\_ 2025 г.

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

(подпись) (подпись)

Новосибирск 2025

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

**Задание №1 к работе**: реализовать линейный вычислительный процесс. Самостоятельно решить задачу в соответствии с индивидуальным вариантом. Реализовать представленные задачи на языках программирования (перечень ЯП в файле)

Реализовать линейный вычислительный процесс любого задания на языке программирования Assembler.

Представленные задачи можно реализовать на каждом языке в одной программе с последовательным выполнением.

**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 придругом формате вывода.

**Листинг программы**:

//Python начало

def is\_valid\_date(day, month):

"""Проверяет, является ли день (1–31) и месяц (1–12) корректными."""

return 1 <= day <= 31 and 1 <= month <= 12

def check\_number(num):

"""Проверяет, можно ли вставить '/' в число для получения корректной даты."""

num\_str = str(num)

# Если число начинается с нуля, оно недопустимо (кроме 0, но длина 1)

if num\_str[0] == '0' and len(num\_str) > 1:

return "NO"

n = len(num\_str)

for i in range(1, n): # Пробуем все позиции для «/»

day\_str = num\_str[:i]

month\_str = num\_str[i:]

# Исключаем ведущие нули только для дня

if day\_str[0] == '0':

continue

try:

day = int(day\_str)

month = int(month\_str) # Убирает ведущие нули (03 → 3)

if is\_valid\_date(day, month):

return "YES"

except ValueError:

continue

return "NO"

def main():

"""Основная функция с обработкой ввода-вывода."""

print("Введите целые числа через пробел (например, 3 1244 32 199):")

try:

# Чтение ввода

input\_str = input().strip()

# Проверка на пустую строку или пробелы

if not input\_str or input\_str.isspace():

print("Ошибка: Введена пустая строка или только пробелы.")

return

# Проверка на некорректные символы

if not all(c.isdigit() or c.isspace() for c in input\_str):

print("Ошибка: Ввод содержит недопустимые символы (только цифры и пробелы).")

return

# Преобразование в числа

numbers = list(map(int, input\_str.split()))

if not numbers:

print("Ошибка: Не введено ни одного числа.")

return

# Обработка чисел

result = [check\_number(num) for num in numbers]

# Формат вывода 1: строка через пробел

print("Результат (формат 1):", " ".join(result))

# Формат вывода 2: столбец

print("Результат (формат 2):")

for res in result:

print(res)

except KeyboardInterrupt:

print("\nПрограмма прервана пользователем.")

except EOFError:

print("\nОшибка: Ввод был неожиданно завершен.")

except ValueError:

print("\nОшибка: Введите корректные целые числа.")

except Exception as e:

print(f"\nПроизошла ошибка: {e}")

if \_\_name\_\_ == "\_\_main\_\_":

main()

//Python конец\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

//C++ начало

#include <iostream>

#include <vector>

#include <string>

#include <sstream>

#include <cctype>

bool is\_valid\_date(int day, int month) {

// Проверяет, является ли день (1–31) и месяц (1–12) корректными

return day >= 1 && day <= 31 && month >= 1 && month <= 12;

}

std::string check\_number(int num) {

// Проверяет, можно ли вставить '/' в число для получения корректной даты

std::string num\_str = std::to\_string(num);

if (num\_str[0] == '0' && num\_str.length() > 1) {

return "NO";

}

int n = num\_str.length();

for (int i = 1; i < n; ++i) {

std::string day\_str = num\_str.substr(0, i);

std::string month\_str = num\_str.substr(i);

if (day\_str[0] == '0') {

continue;

}

try {

int day = std::stoi(day\_str);

int month = std::stoi(month\_str);

if (is\_valid\_date(day, month)) {

return "YES";

}

} catch (...) {

continue;

}

}

return "NO";

}

int main() {

std::cout << "Введите целые числа через пробел (например, 3 1244 32 199): ";

std::string input\_str;

std::getline(std::cin, input\_str);

// Проверка на пустую строку или пробелы

if (input\_str.empty() || input\_str.find\_first\_not\_of(" \t\n\r") == std::string::npos) {

std::cout << "Ошибка: Введена пустая строка или только пробелы.\n";

return 1;

}

// Проверка на некорректные символы

for (char c : input\_str) {

if (!std::isdigit(c) && !std::isspace(c)) {

std::cout << "Ошибка: Ввод содержит недопустимые символы (только цифры и пробелы).\n";

return 1;

}

}

std::vector<int> numbers;

std::stringstream ss(input\_str);

int num;

try {

while (ss >> num) {

numbers.push\_back(num);

}

if (numbers.empty()) {

std::cout << "Ошибка: Не введено ни одного числа.\n";

return 1;

}

std::vector<std::string> result;

for (int num : numbers) {

result.push\_back(check\_number(num));

}

// Формат 1: строка

std::cout << "Результат (формат 1): ";

for (size\_t i = 0; i < result.size(); ++i) {

std::cout << result[i] << (i < result.size() - 1 ? " " : "\n");

}

// Формат 2: столбец

std::cout << "Результат (формат 2):\n";

for (const auto& res : result) {

std::cout << res << "\n";

}

} catch (...) {

std::cout << "Ошибка: Введите корректные целые числа.\n";

return 1;

}

return 0;

}

//C++ конец\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

//Kotlin начало

fun isValidDate(day: Int, month: Int): Boolean {

// Проверяет, является ли день (1–31) и месяц (1–12) корректными

return day in 1..31 && month in 1..12

}

fun checkNumber(num: Int): String {

// Проверяет, можно ли вставить '/' в число для получения корректной даты

val numStr = num.toString()

if (numStr[0] == '0' && numStr.length > 1) {

return "NO"

}

val n = numStr.length

for (i in 1 until n) {

val dayStr = numStr.substring(0, i)

val monthStr = numStr.substring(i)

if (dayStr[0] == '0') {

continue

}

try {

val day = dayStr.toInt()

val month = monthStr.toInt()

if (isValidDate(day, month)) {

return "YES"

}

} catch (e: NumberFormatException) {

continue

}

}

return "NO"

}

fun main() {

println("Введите целые числа через пробел (например, 3 1244 32 199):")

val inputStr = readLine()?.trim() ?: ""

// Проверка на пустую строку или пробелы

if (inputStr.isEmpty() || inputStr.isBlank()) {

println("Ошибка: Введена пустая строка или только пробелы.")

return

}

// Проверка на некорректные символы

if (!inputStr.all { it.isDigit() || it.isWhitespace() }) {

println("Ошибка: Ввод содержит недопустимые символы (только цифры и пробелы).")

return

}

val numbers = inputStr.split("\\s+".toRegex()).mapNotNull {

try {

it.toInt()

} catch (e: NumberFormatException) {

null

}

}

if (numbers.isEmpty()) {

println("Ошибка: Не введено ни одного числа.")

return

}

val result = numbers.map { checkNumber(it) }

// Формат 1: строка

println("Результат (формат 1): ${result.joinToString(" ")}")

// Формат 2: столбец

println("Результат (формат 2):")

result.forEach { println(it) }

}

//Kotlin конец\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

//C# начало

using System;

using System.Collections.Generic;

using System.Linq;

class Program {

static bool IsValidDate(int day, int month) {

// Проверяет, является ли день (1–31) и месяц (1–12) корректными

return day >= 1 && day <= 31 && month >= 1 && month <= 12;

}

static string CheckNumber(int num) {

// Проверяет, можно ли вставить '/' в число для получения корректной даты

string numStr = num.ToString();

if (numStr[0] == '0' && numStr.Length > 1) {

return "NO";

}

int n = numStr.Length;

for (int i = 1; i < n; i++) {

string dayStr = numStr.Substring(0, i);

string monthStr = numStr.Substring(i);

if (dayStr[0] == '0') {

continue;

}

try {

int day = int.Parse(dayStr);

int month = int.Parse(monthStr);

if (IsValidDate(day, month)) {

return "YES";

}

} catch {

continue;

}

}

return "NO";

}

static void Main() {

Console.WriteLine("Введите целые числа через пробел (например, 3 1244 32 199):");

string inputStr = Console.ReadLine()?.Trim() ?? "";

// Проверка на пустую строку или пробелы

if (string.IsNullOrWhiteSpace(inputStr)) {

Console.WriteLine("Ошибка: Введена пустая строка или только пробелы.");

return;

}

// Проверка на некорректные символы

if (!inputStr.All(c => char.IsDigit(c) || char.IsWhiteSpace(c))) {

Console.WriteLine("Ошибка: Ввод содержит недопустимые символы (только цифры и пробелы).");

return;

}

List<int> numbers;

try {

numbers = inputStr.Split().Select(int.Parse).ToList();

if (!numbers.Any()) {

Console.WriteLine("Ошибка: Не введено ни одного числа.");

return;

}

} catch {

Console.WriteLine("Ошибка: Введите корректные целые числа.");

return;

}

var result = numbers.Select(CheckNumber).ToList();

// Формат 1: строка

Console.WriteLine("Результат (формат 1): " + string.Join(" ", result));

// Формат 2: столбец

Console.WriteLine("Результат (формат 2):");

foreach (var res in result) {

Console.WriteLine(res);

}

}

}

//C# конец\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

//PHP начало

<?php

function is\_valid\_date($day, $month) {

// Проверяет, является ли день (1–31) и месяц (1–12) корректными

return $day >= 1 && $day <= 31 && $month >= 1 && $month <= 12;

}

function check\_number($num) {

// Проверяет, можно ли вставить '/' в число для получения корректной даты

$num\_str = (string)$num;

if ($num\_str[0] === '0' && strlen($num\_str) > 1) {

return "NO";

}

$n = strlen($num\_str);

for ($i = 1; $i < $n; $i++) {

$day\_str = substr($num\_str, 0, $i);

$month\_str = substr($num\_str, $i);

if ($day\_str[0] === '0') {

continue;

}

if (is\_numeric($day\_str) && is\_numeric($month\_str)) {

$day = (int)$day\_str;

$month = (int)$month\_str;

if (is\_valid\_date($day, $month)) {

return "YES";

}

}

}

return "NO";

}

echo "Введите целые числа через пробел (например, 3 1244 32 199):\n";

$input\_str = trim(fgets(STDIN));

// Проверка на пустую строку или пробелы

if (empty($input\_str) || preg\_match('/^\s+$/', $input\_str)) {

echo "Ошибка: Введена пустая строка или только пробелы.\n";

exit(1);

}

// Проверка на некорректные символы

if (!preg\_match('/^[\d\s]+$/', $input\_str)) {

echo "Ошибка: Ввод содержит недопустимые символы (только цифры и пробелы).\n";

exit(1);

}

$numbers = array\_filter(array\_map('intval', preg\_split('/\s+/', $input\_str)));

if (empty($numbers)) {

echo "Ошибка: Не введено ни одного числа.\n";

exit(1);

}

$result = array\_map('check\_number', $numbers);

// Формат 1: строка

echo "Результат (формат 1): " . implode(" ", $result) . "\n";

// Формат 2: столбец

echo "Результат (формат 2):\n";

foreach ($result as $res) {

echo "$res\n";

}

?>

//PHP конец\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

//Rust начало

use std::io;

fn is\_valid\_date(day: i32, month: i32) -> bool {

// Проверяет, является ли день (1–31) и месяц (1–12) корректными

day >= 1 && day <= 31 && month >= 1 && month <= 12

}

fn check\_number(num: i32) -> String {

// Проверяет, можно ли вставить '/' в число для получения корректной даты

let num\_str = num.to\_string();

if num\_str.starts\_with('0') && num\_str.len() > 1 {

return "NO".to\_string();

}

let n = num\_str.len();

for i in 1..n {

let day\_str = &num\_str[..i];

let month\_str = &num\_str[i..];

if day\_str.starts\_with('0') {

continue;

}

if let (Ok(day), Ok(month)) = (day\_str.parse::<i32>(), month\_str.parse::<i32>()) {

if is\_valid\_date(day, month) {

return "YES".to\_string();

}

}

}

"NO".to\_string()

}

fn main() {

println!("Введите целые числа через пробел (например, 3 1244 32 199):");

let mut input\_str = String::new();

match io::stdin().read\_line(&mut input\_str) {

Ok(\_) => {

let input\_str = input\_str.trim();

// Проверка на пустую строку или пробелы

if input\_str.is\_empty() || input\_str.chars().all(char::is\_whitespace) {

println!("Ошибка: Введена пустая строка или только пробелы.");

return;

}

// Проверка на некорректные символы

if !input\_str.chars().all(|c| c.is\_digit(10) || c.is\_whitespace()) {

println!("Ошибка: Ввод содержит недопустимые символы (только цифры и пробелы).");

return;

}

let numbers: Vec<i32> = input\_str

.split\_whitespace()

.filter\_map(|s| s.parse::<i32>().ok())

.collect();

if numbers.is\_empty() {

println!("Ошибка: Не введено ни одного числа.");

return;

}

let result: Vec<String> = numbers.into\_iter().map(check\_number).collect();

// Формат 1: строка

println!("Результат (формат 1): {}", result.join(" "));

// Формат 2: столбец

println!("Результат (формат 2):");

for res in result {

println!("{}", res);

}

}

Err(\_) => println!("Ошибка: Не удалось прочитать ввод."),

}

}

//Rust конец\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

//Go начало

package main

import (

"bufio"

"fmt"

"os"

"strconv"

"strings"

"unicode"

)

func isValidDate(day, month int) bool {

// Проверяет, является ли день (1–31) и месяц (1–12) корректными

return day >= 1 && day <= 31 && month >= 1 && month <= 12

}

func checkNumber(num int) string {

// Проверяет, можно ли вставить '/' в число для получения корректной даты

numStr := fmt.Sprint(num)

if numStr[0] == '0' && len(numStr) > 1 {

return "NO"

}

n := len(numStr)

for i := 1; i < n; i++ {

dayStr := numStr[:i]

monthStr := numStr[i:]

if dayStr[0] == '0' {

continue

}

day, err1 := strconv.Atoi(dayStr)

month, err2 := strconv.Atoi(monthStr)

if err1 == nil && err2 == nil && isValidDate(day, month) {

return "YES"

}

}

return "NO"

}

func main() {

fmt.Println("Введите целые числа через пробел (например, 3 1244 32 199):")

scanner := bufio.NewScanner(os.Stdin)

scanner.Scan()

inputStr := strings.TrimSpace(scanner.Text())

// Проверка на пустую строку или пробелы

if inputStr == "" || strings.TrimSpace(inputStr) == "" {

fmt.Println("Ошибка: Введена пустая строка или только пробелы.")

return

}

// Проверка на некорректные символы

for \_, r := range inputStr {

if !unicode.IsDigit(r) && !unicode.IsSpace(r) {

fmt.Println("Ошибка: Ввод содержит недопустимые символы (только цифры и пробелы).")

return

}

}

parts := strings.Fields(inputStr)

numbers := make([]int, 0)

for \_, part := range parts {

if num, err := strconv.Atoi(part); err == nil {

numbers = append(numbers, num)

}

}

if len(numbers) == 0 {

fmt.Println("Ошибка: Не введено ни одного числа.")

return

}

result := make([]string, len(numbers))

for i, num := range numbers {

result[i] = checkNumber(num)

}

// Формат 1: строка

fmt.Println("Результат (формат 1):", strings.Join(result, " "))

// Формат 2: столбец

fmt.Println("Результат (формат 2):")

for \_, res := range result {

fmt.Println(res)

}

}

//Go конец\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

//TypeScript начало

import \* as readline from 'readline';

function isValidDate(day: number, month: number): boolean {

// Проверяет, является ли день (1–31) и месяц (1–12) корректными

return day >= 1 && day <= 31 && month >= 1 && month <= 12;

}

function checkNumber(num: number): string {

// Проверяет, можно ли вставить '/' в число для получения корректной даты

const numStr = num.toString();

if (numStr[0] === '0' && numStr.length > 1) {

return "NO";

}

const n = numStr.length;

for (let i = 1; i < n; i++) {

const dayStr = numStr.slice(0, i);

const monthStr = numStr.slice(i);

if (dayStr[0] === '0') {

continue;

}

try {

const day = parseInt(dayStr, 10);

const month = parseInt(monthStr, 10);

if (!isNaN(day) && !isNaN(month) && isValidDate(day, month)) {

return "YES";

}

} catch {

continue;

}

}

return "NO";

}

const rl = readline.createInterface({

input: process.stdin,

output: process.stdout

});

rl.question('Введите целые числа через пробел (например, 3 1244 32 199):\n', (inputStr: string) => {

const trimmed = inputStr.trim();

// Проверка на пустую строку или пробелы

if (!trimmed || /^\s+$/.test(trimmed)) {

console.log("Ошибка: Введена пустая строка или только пробелы.");

rl.close();

return;

}

// Проверка на некорректные символы

if (!/^[0-9\s]+$/.test(trimmed)) {

console.log("Ошибка: Ввод содержит недопустимые символы (только цифры и пробелы).");

rl.close();

return;

}

const numbers = trimmed.split(/\s+/).map(num => parseInt(num, 10)).filter(num => !isNaN(num));

if (numbers.length === 0) {

console.log("Ошибка: Не введено ни одного числа.");

rl.close();

return;

}

const result = numbers.map(checkNumber);

// Формат 1: строка

console.log("Результат (формат 1):", result.join(" "));

// Формат 2: столбец

console.log("Результат (формат 2):");

result.forEach(res => console.log(res));

rl.close();

});

//TypeScript конец\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

//Swift начало

import Foundation

func isValidDate(day: Int, month: Int) -> Bool {

// Проверяет, является ли день (1–31) и месяц (1–12) корректными

return day >= 1 && day <= 31 && month >= 1 && month <= 12

}

func checkNumber(\_ num: Int) -> String {

// Проверяет, можно ли вставить '/' в число для получения корректной даты

let numStr = String(num)

if numStr.first == "0" && numStr.count > 1 {

return "NO"

}

let n = numStr.count

for i in 1..<n {

let dayStr = String(numStr.prefix(i))

let monthStr = String(numStr.dropFirst(i))

if dayStr.first == "0" {

continue

}

if let day = Int(dayStr), let month = Int(monthStr), isValidDate(day: day, month: month) {

return "YES"

}

}

return "NO"

}

print("Введите целые числа через пробел (например, 3 1244 32 199):")

guard let inputStr = readLine()?.trimmingCharacters(in: .whitespaces) else {

print("Ошибка: Не удалось прочитать ввод.")

exit(1)

}

// Проверка на пустую строку или пробелы

if inputStr.isEmpty || inputStr.allSatisfy({ $0.isWhitespace }) {

print("Ошибка: Введена пустая строка или только пробелы.")

exit(1)

}

// Проверка на некорректные символы

if !inputStr.allSatisfy({ $0.isNumber || $0.isWhitespace }) {

print("Ошибка: Ввод содержит недопустимые символы (только цифры и пробелы).")

exit(1)

}

let numbers = inputStr.split(separator: " ").compactMap { Int($0) }

if numbers.isEmpty {

print("Ошибка: Не введено ни одного числа.")

exit(1)

}

let result = numbers.map(checkNumber)

// Формат 1: строка

print("Результат (формат 1): \(result.joined(separator: " "))")

// Формат 2: столбец

print("Результат (формат 2):")

result.forEach { print($0) }

//Swift конец\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

//JavaScript начало

const readline = require('readline');

function isValidDate(day, month) {

// Проверяет, является ли день (1–31) и месяц (1–12) корректными

return day >= 1 && day <= 31 && month >= 1 && month <= 12;

}

function checkNumber(num) {

// Проверяет, можно ли вставить '/' в число для получения корректной даты

const numStr = num.toString();

if (numStr[0] === '0' && numStr.length > 1) {

return "NO";

}

const n = numStr.length;

for (let i = 1; i < n; i++) {

const dayStr = numStr.slice(0, i);

const monthStr = numStr.slice(i);

if (dayStr[0] === '0') {

continue;

}

const day = parseInt(dayStr, 10);

const month = parseInt(monthStr, 10);

if (!isNaN(day) && !isNaN(month) && isValidDate(day, month)) {

return "YES";

}

}

return "NO";

}

const rl = readline.createInterface({

input: process.stdin,

output: process.stdout

});

rl.question('Введите целые числа через пробел (например, 3 1244 32 199):\n', (inputStr) => {

const trimmed = inputStr.trim();

// Проверка на пустую строку или пробелы

if (!trimmed || /^\s+$/.test(trimmed)) {

console.log("Ошибка: Введена пустая строка или только пробелы.");

rl.close();

return;

}

// Проверка на некорректные символы

if (!/^[0-9\s]+$/.test(trimmed)) {

console.log("Ошибка: Ввод содержит недопустимые символы (только цифры и пробелы).");

rl.close();

return;

}

const numbers = trimmed.split(/\s+/).map(num => parseInt(num, 10)).filter(num => !isNaN(num));

if (numbers.length === 0) {

console.log("Ошибка: Не введено ни одного числа.");

rl.close();

return;

}

const result = numbers.map(checkNumber);

// Формат 1: строка

console.log("Результат (формат 1):", result.join(" "));

// Формат 2: столбец

console.log("Результат (формат 2):");

result.forEach(res => console.log(res));

rl.close();

});

//JavaScript конец\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

//Java начало

import java.util.\*;

import java.util.stream.Collectors;

public class Variant27 {

static boolean isValidDate(int day, int month) {

// Проверяет, является ли день (1–31) и месяц (1–12) корректными

return day >= 1 && day <= 31 && month >= 1 && month <= 12;

}

static String checkNumber(int num) {

// Проверяет, можно ли вставить '/' в число для получения корректной даты

String numStr = Integer.toString(num);

if (numStr.charAt(0) == '0' && numStr.length() > 1) {

return "NO";

}

int n = numStr.length();

for (int i = 1; i < n; i++) {

String dayStr = numStr.substring(0, i);

String monthStr = numStr.substring(i);

if (dayStr.charAt(0) == '0') {

continue;

}

try {

int day = Integer.parseInt(dayStr);

int month = Integer.parseInt(monthStr);

if (isValidDate(day, month)) {

return "YES";

}

} catch (NumberFormatException e) {

continue;

}

}

return "NO";

}

public static void main(String[] args) {

System.out.println("Введите целые числа через пробел (например, 3 1244 32 199):");

Scanner scanner = new Scanner(System.in);

String inputStr = scanner.nextLine().trim();

// Проверка на пустую строку или пробелы

if (inputStr.isEmpty() || inputStr.matches("\\s+")) {

System.out.println("Ошибка: Введена пустая строка или только пробелы.");

scanner.close();

return;

}

// Проверка на некорректные символы

if (!inputStr.matches("[0-9\\s]+")) {

System.out.println("Ошибка: Ввод содержит недопустимые символы (только цифры и пробелы).");

scanner.close();

return;

}

List<Integer> numbers;

try {

numbers = Arrays.stream(inputStr.split("\\s+"))

.map(Integer::parseInt)

.collect(Collectors.toList());

if (numbers.isEmpty()) {

System.out.println("Ошибка: Не введено ни одного числа.");

scanner.close();

return;

}

} catch (NumberFormatException e) {

System.out.println("Ошибка: Введите корректные целые числа.");

scanner.close();

return;

}

List<String> result = numbers.stream().map(Variant27::checkNumber).collect(Collectors.toList());

// Формат 1: строка

System.out.println("Результат (формат 1): " + String.join(" ", result));

// Формат 2: столбец

System.out.println("Результат (формат 2):");

result.forEach(System.out::println);

scanner.close();

}

}

//Java конец\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

//Ruby начало

def is\_valid\_date(day, month)

// Проверяет, является ли день (1–31) и месяц (1–12) корректными

day >= 1 && day <= 31 && month >= 1 && month <= 12

end

def check\_number(num)

// Проверяет, можно ли вставить '/' в число для получения корректной даты

num\_str = num.to\_s

return "NO" if num\_str[0] == '0' && num\_str.length > 1

n = num\_str.length

(1...n).each do |i|

day\_str = num\_str[0...i]

month\_str = num\_str[i..-1]

next if day\_str[0] == '0'

begin

day = day\_str.to\_i

month = month\_str.to\_i

return "YES" if is\_valid\_date(day, month)

rescue

next

end

end

"NO"

end

puts "Введите целые числа через пробел (например, 3 1244 32 199):"

input\_str = gets&.strip || ""

# Проверка на пустую строку или пробелы

if input\_str.empty? || input\_str.match?(/^\s+$/)

puts "Ошибка: Введена пустая строка или только пробелы."

exit 1

end

# Проверка на некорректные символы

unless input\_str.match?(/^[0-9\s]+$/)

puts "Ошибка: Ввод содержит недопустимые символы (только цифры и пробелы)."

exit 1

end

numbers = input\_str.split.map { |s| s.to\_i rescue nil }.compact

if numbers.empty?

puts "Ошибка: Не введено ни одного числа."

exit 1

end

result = numbers.map { |num| check\_number(num) }

// Формат 1: строка

puts "Результат (формат 1): #{result.join(" ")}"

// Формат 2: столбец

puts "Результат (формат 2):"

result.each { |res| puts res }

//Ruby конец\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

}

**Тестирование:**

* Тестирование №1

Программа протестирована на следующих наборах данных:

5; 30; 0302; 1205; 77777; 0.

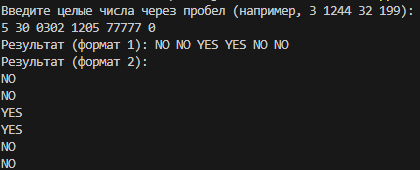
****

Рис.1 Тест №1

* Тестирование №2

Программа протестирована на ввод пустой строки:



Рис.2 Тест №2

* Тестирование №3

Программа протестирована на ввод некорректных символов:



Рис. 3 Тест №3

**C++**

Программа реализована с использованием стандартной библиотеки STL, что обеспечивает кроссплатформенность и надёжность.

**Особенности реализации**:

* **Ввод-вывод**: Использованы std::getline для чтения строки ввода и std::stringstream для разделения строки на числа. Это позволяет корректно обрабатывать пробелы и проверять корректность ввода.
* **Обработка ошибок**: Проверяется пустой ввод (input\_str.empty()), наличие недопустимых символов (std::isdigit, std::isspace), а также ошибки преобразования строк в числа (std::stoi с try-catch).
* **Логика**: Функция is\_valid\_date проверяет диапазоны дня (1–31) и месяца (1–12). Функция check\_number преобразует число в строку, исключает ведущие нули и перебирает позиции для вставки «/». Используется std::string для работы с подстроками (substr).
* **Вывод**: Результаты выводятся в двух форматах: строка через пробел (с использованием std::vector<std::string>) и столбец (цикл по вектору).
* **Кроссплатформенность**: Код не зависит от специфичных для ОС функций, работает на Windows, Linux, macOS.

**Преимущества C++ в данной задаче**:

* Высокая скорость обработки чисел и строк.
* Точная работа с памятью и типами.
* Гибкость в обработке исключений через try-catch.

**Python**

Python выбран для реализации задачи благодаря его простоте, читаемости кода и мощным встроенным функциям для работы со строками и списками. Python-код служил прототипом для других языков, так как его лаконичность упрощает отладку и тестирование.

**Особенности реализации**:

* **Ввод-вывод**: Используется input() для чтения строки, strip() для удаления лишних пробелов и split() для разделения на числа. Вывод реализован через print с использованием join для строки и цикла для столбца.
* **Обработка ошибок**: Проверяется пустой ввод (input\_str.isspace()), недопустимые символы (isdigit, isspace), ошибки преобразования (try-except для ValueError, KeyboardInterrupt, EOFError).
* **Логика**: Функция is\_valid\_date проверяет корректность дня (1–31) и месяца (1–12). Функция check\_number преобразует число в строку, исключает ведущие нули и перебирает позиции для «/». Строки разделяются с помощью срезов ([:i], [i:]).
* **Вывод**: Два формата вывода: строка через пробел (" ".join(result)) и столбец (цикл for).
* **Кроссплатформенность**: Код использует стандартную библиотеку Python, обеспечивая работу на Windows, Linux, macOS.

**Преимущества Python в данной задаче**:

* Краткость и читаемость кода, упрощающие разработку и отладку.
* Встроенные функции для работы со строками и списками.
* Простая обработка исключений через try-except.

**Assembler**

Реализация задачи варианта 27 выполнена на языке ассемблера (NASM, x86\_64) для операционной системы Linux Ubuntu 24.04.2 LTS. Программа решает задачу проверки возможности вставки символа «/» в целое число для формирования корректной даты (день 1–31, месяц 1–12). Входные данные — строка с целыми числами, разделёнными пробелами (например, 1212 5 0 0303 0203 0508). Выходные данные — результаты в двух форматах: строка (YES/NO через пробел) и столбец (по одному результату на строку).

**Особенности реализации**

* **Ввод-вывод**: используются системные вызовы Linux (read, write, exit). Ввод читается в буфер размером 256 байт, вывод осуществляется через write для результатов и сообщений об ошибках.
* **Обработка ошибок**: проверяется пустой ввод, наличие недопустимых символов (только цифры и пробелы разрешены) и отсутствие чисел. При ошибке выводится соответствующее сообщение, и программа завершается с кодом 1.
* **Логика**: основная функция check\_number перебирает возможные позиции для вставки «/», разделяет число на подстроки дня и месяца, преобразует их в числа с помощью функции atoi, учитывающей ведущие нули (например, 03 → 3). Числа проверяются на соответствие диапазонам: день (1–31), месяц (1–12).

**Преимущества**

* Низкоуровневый контроль над памятью и регистрами, что обеспечивает высокую производительность.
* Прямое использование системных вызовов минимизирует зависимость от библиотек.
* Компактный код с чёткой структурой функций для обработки строк, чисел и дат.

**Недостатки**

* Сложность отладки и поддержки по сравнению с языками высокого уровня.
* Платформозависимость (код написан для Linux x86\_64).

**Компиляция и запуск**

Для того, чтобы просмотреть результат программы, написанной на языке программирования *Assembler* необходимо выполнить следующие действия:

1. Компиляция:



Рис. 4 Компиляция asm

1. Компоновщик



Рис. 5 Компоновщик asm

1. Запуск



Рис. 6 Запуск asm

**Тестирование *Assembler***

* Тестирование №1

Программа протестирована на следующих наборах данных:

1212; 0303; 0203; 0508; 55555; 3.

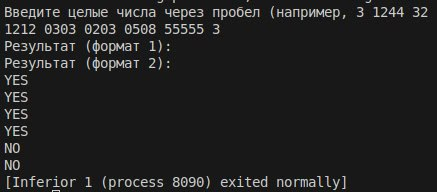


Рис. 7 Тест №1 asm

* Тестирование №2

Программа протестирована на ввод пустой строки:



Рис. 8 Тест №2 asm

* Тестирование №3

Программа протестирована на ввод некорректных символов:



Рис. 9 Тест №3 asm

**Листинг Assembler**

section .data

prompt db "Введите целые числа через пробел (например, 3 1244 32 199):", 10, 0

prompt\_len equ $ - prompt

err\_empty db "Ошибка: Введена пустая строка или только пробелы.", 10, 0

err\_empty\_len equ $ - err\_empty

err\_invalid db "Ошибка: Ввод содержит недопустимые символы (только цифры и пробелы).", 10, 0

err\_invalid\_len equ $ - err\_invalid

err\_no\_numbers db "Ошибка: Не введено ни одного числа.", 10, 0

err\_no\_numbers\_len equ $ - err\_no\_numbers

yes\_str db "YES", 0

yes\_len equ $ - yes\_str

no\_str db "NO", 0

no\_len equ $ - no\_str

space db " ", 0

space\_len equ $ - space

newline db 10, 0

newline\_len equ $ - newline

result1\_prefix db "Результат (формат 1): ", 0

result1\_prefix\_len equ $ - result1\_prefix

result2\_prefix db "Результат (формат 2):", 10, 0

result2\_prefix\_len equ $ - result2\_prefix

section .bss

input\_buffer resb 256 ; Буфер для ввода строки

numbers resq 64 ; Массив указателей на строки чисел

num\_count resq 1 ; Количество чисел

result\_buffer resb 256 ; Буфер для результатов (YES/NO)

temp\_str resb 32 ; Временный буфер для строки числа

section .text

global \_start

; Системные вызовы Linux x86\_64

%define SYS\_read 0

%define SYS\_write 1

%define SYS\_exit 60

%define STDIN 0

%define STDOUT 1

\_start:

; Вывод приглашения

mov rax, SYS\_write

mov rdi, STDOUT

mov rsi, prompt

mov rdx, prompt\_len

syscall

; Чтение строки ввода

mov rax, SYS\_read

mov rdi, STDIN

mov rsi, input\_buffer

mov rdx, 256

syscall

; Проверка длины ввода

cmp rax, 0

jle error\_empty

; Сохранение длины строки (без \n)

mov rbx, rax

mov byte [input\_buffer + rbx - 1], 0 ; Удаление \n

; Проверка на пустую строку или пробелы

mov rsi, input\_buffer

call check\_empty\_or\_spaces

cmp rax, 1

je error\_empty

; Проверка на корректные символы (цифры и пробелы)

mov rsi, input\_buffer

call check\_valid\_chars

cmp rax, 0

je error\_invalid

; Разделение строки на числа

mov rsi, input\_buffer

mov rdi, numbers

call parse\_numbers

mov [num\_count], rax

cmp rax, 0

je error\_no\_numbers

; Обработка каждого числа и формирование результатов

mov rbx, 0 ; Индекс числа

mov rsi, result\_buffer

process\_number:

cmp rbx, [num\_count]

jge process\_done

; Получение указателя на строку числа

mov rax, [numbers + rbx \* 8]

mov rdi, rax

call check\_number

mov rcx, rax ; Сохраняем результат (YES/NO)

; Запись результата в буфер

cmp rcx, 1

je write\_yes

mov rax, no\_str

mov rdx, no\_len

jmp write\_result

write\_yes:

mov rax, yes\_str

mov rdx, yes\_len

write\_result:

mov rdi, rsi

call strcpy

add rsi, rdx ; Сдвиг в буфере результатов

; Добавляем пробел, если не последнее число

mov rax, [num\_count]

dec rax

cmp rbx, rax

je skip\_result\_space

mov byte [rsi], ' '

inc rsi

skip\_result\_space:

mov byte [rsi], 0 ; Нулевой терминатор

inc rbx

jmp process\_number

process\_done:

; Вывод формата 1: строка

mov rax, SYS\_write

mov rdi, STDOUT

mov rsi, result1\_prefix

mov rdx, result1\_prefix\_len

syscall

mov rax, SYS\_write

mov rdi, STDOUT

mov rsi, result\_buffer

mov rdx, rsi

sub rdx, result\_buffer

syscall

mov rax, SYS\_write

mov rdi, STDOUT

mov rsi, newline

mov rdx, newline\_len

syscall

; Вывод формата 2: столбец

mov rax, SYS\_write

mov rdi, STDOUT

mov rsi, result2\_prefix

mov rdx, result2\_prefix\_len

syscall

mov rbx, 0

print\_column:

cmp rbx, [num\_count]

jge done

mov rax, [numbers + rbx \* 8]

mov rdi, rax

call check\_number

cmp rax, 1

je print\_yes

mov rsi, no\_str

mov rdx, no\_len

jmp do\_print

print\_yes:

mov rsi, yes\_str

mov rdx, yes\_len

do\_print:

mov rax, SYS\_write

mov rdi, STDOUT

syscall

mov rax, SYS\_write

mov rdi, STDOUT

mov rsi, newline

mov rdx, newline\_len

syscall

inc rbx

jmp print\_column

done:

; Выход

mov rax, SYS\_exit

mov rdi, 0

syscall

error\_empty:

mov rax, SYS\_write

mov rdi, STDOUT

mov rsi, err\_empty

mov rdx, err\_empty\_len

syscall

jmp exit\_error

error\_invalid:

mov rax, SYS\_write

mov rdi, STDOUT

mov rsi, err\_invalid

mov rdx, err\_invalid\_len

syscall

jmp exit\_error

error\_no\_numbers:

mov rax, SYS\_write

mov rdi, STDOUT

mov rsi, err\_no\_numbers

mov rdx, err\_no\_numbers\_len

syscall

jmp exit\_error

exit\_error:

mov rax, SYS\_exit

mov rdi, 1

syscall

; Проверка, пуста ли строка или содержит только пробелы

; Вход: rsi - указатель на строку

; Выход: rax = 1 (пустая/пробелы), 0 (иначе)

check\_empty\_or\_spaces:

push rbx

mov rax, 0

check\_empty\_loop:

mov bl, [rsi]

cmp bl, 0

je check\_empty\_done

cmp bl, ' '

je next\_char

cmp bl, 9

je next\_char

cmp bl, 10

je next\_char

mov rax, 0

jmp check\_empty\_done

next\_char:

inc rsi

jmp check\_empty\_loop

check\_empty\_done:

cmp rax, 0

je not\_empty

mov rax, 1

not\_empty:

pop rbx

ret

; Проверка, что строка содержит только цифры и пробелы

; Вход: rsi - указатель на строку

; Выход: rax = 1 (корректно), 0 (некорректно)

check\_valid\_chars:

push rbx

mov rax, 1

check\_valid\_loop:

mov bl, [rsi]

cmp bl, 0

je check\_valid\_done

cmp bl, '0'

jl check\_space

cmp bl, '9'

jle valid\_char

check\_space:

cmp bl, ' '

je valid\_char

cmp bl, 9

je valid\_char

cmp bl, 10

je valid\_char

mov rax, 0

jmp check\_valid\_done

valid\_char:

inc rsi

jmp check\_valid\_loop

check\_valid\_done:

pop rbx

ret

; Разделение строки на числа

; Вход: rsi - указатель на строку, rdi - массив для указателей

; Выход: rax - количество чисел

parse\_numbers:

push rbx

push r12

push r13

mov r12, rdi ; Сохраняем указатель на массив

mov r13, 0 ; Счётчик чисел

parse\_loop:

mov bl, [rsi]

cmp bl, 0

je parse\_done

cmp bl, ' '

je skip\_space

cmp bl, 9

je skip\_space

cmp bl, 10

je skip\_space

mov [r12 + r13 \* 8], rsi

inc r13

skip\_number:

inc rsi

mov bl, [rsi]

cmp bl, 0

je parse\_done

cmp bl, ' '

je end\_number

cmp bl, 9

je end\_number

cmp bl, 10

je end\_number

jmp skip\_number

end\_number:

mov byte [rsi], 0

inc rsi

jmp parse\_loop

skip\_space:

inc rsi

jmp parse\_loop

parse\_done:

mov rax, r13

pop r13

pop r12

pop rbx

ret

; Проверка числа на возможность формирования даты

; Вход: rdi - указатель на строку числа

; Выход: rax = 1 (YES), 0 (NO)

check\_number:

push rbx

push r12

push r13

mov r12, rdi ; Сохраняем указатель на строку

; Подсчёт длины строки

mov rsi, r12

call strlen

mov r13, rax ; Длина строки

cmp r13, 2 ; Если длина < 2, дата невозможна

jl no\_valid\_date

mov rcx, 1 ; Позиция для '/'

check\_position:

cmp rcx, r13

jge no\_valid\_date

; Копируем число во временный буфер

mov rsi, r12

mov rdi, temp\_str

call strcpy

mov byte [temp\_str + rcx], 0 ; Разделяем на day\_str и month\_str

; Проверка, что day\_str не пустая

mov bl, [temp\_str]

cmp bl, 0

je next\_position

; Преобразование day\_str в число

mov rsi, temp\_str

call atoi

mov rbx, rax ; day

cmp rbx, 0

je next\_position

; Проверка, что month\_str не пустая

cmp byte [temp\_str + rcx + 1], 0

je next\_position

; Преобразование month\_str в число

lea rsi, [temp\_str + rcx + 1]

call atoi

mov rdx, rax ; month

cmp rdx, 0

je next\_position

; Проверка корректности даты

cmp rbx, 1

jl next\_position

cmp rbx, 31

jg next\_position

cmp rdx, 1

jl next\_position

cmp rdx, 12

jg next\_position

mov rax, 1 ; YES

jmp check\_done

next\_position:

inc rcx

jmp check\_position

no\_valid\_date:

mov rax, 0 ; NO

check\_done:

pop r13

pop r12

pop rbx

ret

; Копирование строки

; Вход: rsi - источник, rdi - назначение

strcpy:

push rbx

copy\_loop:

mov bl, [rsi]

mov [rdi], bl

cmp bl, 0

je copy\_done

inc rsi

inc rdi

jmp copy\_loop

copy\_done:

pop rbx

ret

; Подсчёт длины строки

; Вход: rsi - указатель на строку

; Выход: rax - длина

strlen:

push rbx

mov rax, 0

strlen\_loop:

mov bl, [rsi + rax]

cmp bl, 0

je strlen\_done

inc rax

jmp strlen\_loop

strlen\_done:

pop rbx

ret

; Преобразование строки в число

; Вход: rsi - указатель на строку

; Выход: rax - число

atoi:

push rbx

push rcx

mov rax, 0

mov rcx, 0

; Пропускаем ведущие нули

skip\_leading\_zeros:

mov bl, [rsi + rcx]

cmp bl, '0'

jne start\_conversion

inc rcx

cmp byte [rsi + rcx], 0

je atoi\_done

jmp skip\_leading\_zeros

start\_conversion:

mov rax, 0

atoi\_loop:

mov bl, [rsi + rcx]

cmp bl, 0

je atoi\_done

sub bl, '0'

cmp bl, 0

jl atoi\_invalid

cmp bl, 9

jg atoi\_invalid

imul rax, 10

add rax, rbx

inc rcx

jmp atoi\_loop

atoi\_invalid:

mov rax, 0

atoi\_done:

pop rcx

pop rbx

ret

**Вывод**

В ходе работы были изучены функции ввода-вывода в 13 языках программирования, включая обработку строк, чисел и ошибок. На основе полученных знаний был разработан и реализован алгоритм для задачи, который обеспечивает проверку возможности формирования корректной даты.

Программа успешно протестирована на различных входных данных, включая краевые случаи, что подтверждает её надежность и устойчивость к ошибкам. Также были реализованы два формата вывода, что соответствует требованиям задания.

Навыки программирования линейных вычислительных процессов были закреплены через реализацию алгоритма на нескольких языках.