МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ государственное БЮДЖЕТНОЕ

образовательное учреждение

высшего образования

«НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

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

Кафедра защиты информации

**

**ЛАБОРАТОРНАЯ РАБОТА №2**

**по дисциплине: «Программирование»**

**на тему: «**Классификация языков программирования**»**

|  |  |
| --- | --- |
| Выполнил:  Студент гр. «АБс-424», «АВТФ»,  *Подземельный Юрий Евгеньевич*  «11» \_\_04\_\_ 2025г  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись) | Проверил:  *Доцент кафедры ЗИ*  *Исаев Глеб Андреевич*  «\_\_\_» \_\_\_\_\_\_ 2025г  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись) |

Новосибирск 2025

СОДЕРЖАНИЕ

[Введение 3](#_Toc191566155)

[Ход выполнения 4](#_Toc191566156)

[Вывод 29](#_Toc191566158)

# ВВЕДЕНИЕ

**Цели и задачи работы**: изучение функций ввода-вывода данных,

программирования вычисления значения выражения.

**Задание к работе:**

1. Реализовать линейный вычислительный процесс. Самостоятельно

решить задачи в соответствии с индивидуальным вариантом. Реализовать

представленные задачи на языках программирования C++, Kotlin, С#, PHP,

Rust, Go (Golang), TypeScript, Swift, JavaScript, Python, Java, Ruby.

2. Реализовать линейный вычислительный процесс любого задания на

языке программирования Assembler.

3. Представленные задачи можно реализовать на каждом языке в

одной программе с последовательным выполнением.

**Ход выполнения**

10. Для заданного натурального числа N найдите количество

конечных нулей в факториале числа N (то есть в записи N!).

Пример: N=25

Результат: 6

C++

#include <iostream>

using namespace std;

int main()

{

int N;

cin >> N; //Чтение N

int Nf = 0; //Счётчик нулей

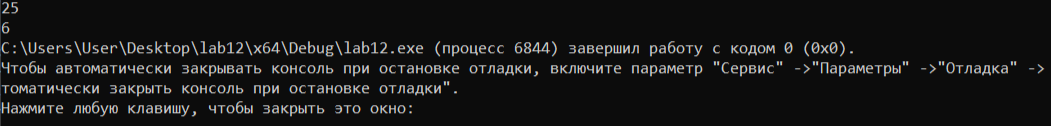
for (int i = 5; i <= N; i \*= 5) { //Считаем, сколько раз 5 входит в разложение

Nf += N / i;

}

cout << Nf;

}



GO

package main

import "fmt"

func main() {

var N int

fmt.Scan(&N)

Nf := 0

for i := 5; i <= N; i \*= 5 {

Nf += N / i

}

fmt.Println(Nf)

}

Java

import java.util.Scanner;

public class Main {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

int N = scanner.nextInt();

int Nf = 0;

for (int i = 5; i <= N; i \*= 5) {

Nf += N / i;

}

System.out.println(Nf);

}

}

JS

const N = parseInt(prompt("Введите число N:"));

let Nf = 0;

for (let i = 5; i <= N; i \*= 5) {

Nf += Math.floor(N / i);

}

console.log("Количество нулей в факториале:", Nf);

Kotlin

fun main() {

val N = readLine()!!.toInt()

var Nf = 0

var i = 5

while (i <= N) {

Nf += N / i

i \*= 5

}

println(Nf)

}

PHP

<?php

$N = (int)fgets(STDIN);

$Nf = 0;

for ($i = 5; $i <= $N; $i \*= 5) {

$Nf += (int)($N / $i);

}

echo $Nf;

?>

Python

n = int(input())

total = 0

i = 5

while i <= n:

total += n // i

i \*= 5

print(total)

Ruby

n = gets.to\_i

nf = 0

i = 5

while i <= n

nf += n / i

i \*= 5

end

puts nf

Rust

use std::io;

fn main() {

let mut input = String::new();

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

let n: i32 = input.trim().parse().unwrap();

let mut nf = 0;

let mut i = 5;

while i <= n {

nf += n / i;

i \*= 5;

}

println!("{}", nf);

}

Swift

import Foundation

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

var Nf = 0

var i = 5

while i <= N {

Nf += N / i

i \*= 5

}

print(Nf)

}

TypeScript

import \* as readline from 'readline';

const rl = readline.createInterface({

input: process.stdin,

output: process.stdout

});

rl.on('line', (input) => {

const N = parseInt(input);

let Nf = 0;

for (let i = 5; i <= N; i \*= 5) {

Nf += Math.floor(N / i);

}

console.log(Nf);

rl.close();

});

С#

using System;

class Program

{

static void Main()

{

int N = int.Parse(Console.ReadLine());

int Nf = 0;

for (int i = 5; i <= N; i \*= 5)

{

Nf += N / i;

}

Console.WriteLine(Nf);

}

}

Assembly

section .data

input\_format db "%d", 0

output\_format db "%d", 10, 0

section .bss

N resd 1

Nf resd 1

i resd 1

section .text

global main

extern scanf, printf

main:

push rbp

mov rbp, rsp

mov rdi, input\_format

mov rsi, N

xor eax, eax

call scanf

mov dword [Nf], 0

mov dword [i], 5

.loop:

mov eax, [i]

cmp eax, [N]

jg .end\_loop

mov eax, [N]

xor edx, edx

div dword [i]

add [Nf], eax

mov eax, [i]

imul eax, 5

mov [i], eax

jmp .loop

.end\_loop:

mov rdi, output\_format

mov esi, [Nf]

xor eax, eax

call printf

xor eax, eax

mov rsp, rbp

pop rbp

ret

6. Необходимо написать программу по преобразования римского числа в

натуральное число.

Пример 1:

Input: LXVI

Output: 66

I = 1 V = 5 X = 10 L = 50

Пример 2:

Input: LXXXI

Output: 81

I = 1 X = 10 L = 50

C++

#include <string>

#include <map>

#include <iostream>

using namespace std;

int main()

{

string A;

cin >> A; //Ввод числа

map<char, int> M = { //Соответствие римских цифр и их значений

{'I', 1},

{'V', 5},

{'X', 10},

{'L', 50},

{'C', 100},

{'D', 500},

{'M', 1000}

};

int B = 0; //Результат

//Проходим по каждой цифре в римском числе

for (int k = 0; k < A.size(); k++) {

//Если текущая цифра меньше следующей, то вычитаем её значение

if (k < A.size() - 1 && M[A[k]] < M[A[k + 1]]) {

B -= M[A[k]];

}

//Иначе прибавляем

else {

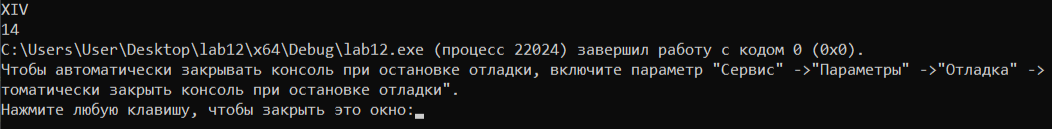
B += M[A[k]];

}

}

cout << B;

}



GO

package main

import "fmt"

func main() {

var A string

fmt.Scan(&A)

M := map[byte]int{

'I': 1,

'V': 5,

'X': 10,

'L': 50,

'C': 100,

'D': 500,

'M': 1000,

}

B := 0

for k := 0; k < len(A); k++ {

if k < len(A)-1 && M[A[k]] < M[A[k+1]] {

B -= M[A[k]]

} else {

B += M[A[k]]

}

}

fmt.Println(B)

}

Java

import java.util.Scanner;

import java.util.HashMap;

public class Main {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

String A = scanner.next(); // Аналог cin >> A

HashMap<Character, Integer> M = new HashMap<>() {{

put('I', 1);

put('V', 5);

put('X', 10);

put('L', 50);

put('C', 100);

put('D', 500);

put('M', 1000);

}};

int B = 0;

for (int k = 0; k < A.length(); k++) {

if (k < A.length() - 1 && M.get(A.charAt(k)) < M.get(A.charAt(k + 1))) {

B -= M.get(A.charAt(k));

} else {

B += M.get(A.charAt(k));

}

}

System.out.println(B);

}

}

JS

{

const A = prompt("Введите римское число:")?.toUpperCase() || "";

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

let B = 0;

for (let k = 0; k < A.length; k++) {

if (k < A.length - 1 && M[A[k]] < M[A[k + 1]]) {

B -= M[A[k]];

} else {

B += M[A[k]];

}

}

console.log("Результат:", B);

}

Kotlin

fun main() {

val A = readLine()!!

val M = mapOf(

'I' to 1,

'V' to 5,

'X' to 10,

'L' to 50,

'C' to 100,

'D' to 500,

'M' to 1000

)

var B = 0

for (k in A.indices) {

if (k < A.length - 1 && M[A[k]]!! < M[A[k + 1]]!!) {

B -= M[A[k]]!!

} else {

B += M[A[k]]!!

}

}

println(B)

}

PHP

<?php

$A = readline();

$M = [

'I' => 1,

'V' => 5,

'X' => 10,

'L' => 50,

'C' => 100,

'D' => 500,

'M' => 1000

];

$B = 0;

for ($k = 0; $k < strlen($A); $k++) {

if ($k < strlen($A) - 1 && $M[$A[$k]] < $M[$A[$k + 1]]) {

$B -= $M[$A[$k]];

} else {

$B += $M[$A[$k]];

}

}

echo $B;

?>

Python

A = input()

M = {

'I': 1,

'V': 5,

'X': 10,

'L': 50,

'C': 100,

'D': 500,

'M': 1000

}

B = 0

for k in range(len(A)):

if k < len(A) - 1 and M[A[k]] < M[A[k + 1]]:

B -= M[A[k]]

else:

B += M[A[k]]

print(B)

Ruby

A = gets.chomp

M = {

'I' => 1,

'V' => 5,

'X' => 10,

'L' => 50,

'C' => 100,

'D' => 500,

'M' => 1000

}

B = 0

(0...A.length).each do |k|

if k < A.length - 1 && M[A[k]] < M[A[k + 1]]

B -= M[A[k]]

else

B += M[A[k]]

end

end

puts B

Rust

use std::collections::HashMap;

use std::io;

fn main() {

let mut input = String::new();

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

let a = input.trim();

let mut m: HashMap<char, i32> = HashMap::new();

m.insert('I', 1);

m.insert('V', 5);

m.insert('X', 10);

m.insert('L', 50);

m.insert('C', 100);

m.insert('D', 500);

m.insert('M', 1000);

let mut b = 0;

let chars: Vec<char> = a.chars().collect();

for k in 0..chars.len() {

if k < chars.len() - 1 && m[&chars[k]] < m[&chars[k + 1]] {

b -= m[&chars[k]];

} else {

b += m[&chars[k]];

}

}

println!("{}", b);

}

Swift

import Foundation

let A = readLine()!

let M: [Character: Int] = [

"I": 1,

"V": 5,

"X": 10,

"L": 50,

"C": 100,

"D": 500,

"M": 1000

]

var B = 0

for k in 0..<A.count {

let currentChar = A[A.index(A.startIndex, offsetBy: k)]

if k < A.count - 1 {

let nextChar = A[A.index(A.startIndex, offsetBy: k + 1)]

if M[currentChar]! < M[nextChar]! {

B -= M[currentChar]!

} else {

B += M[currentChar]!

}

} else {

B += M[currentChar]!

}

}

print(B)

TypeScript

import \* as readline from 'readline';

const rl = readline.createInterface({

input: process.stdin,

output: process.stdout

});

rl.question('', (A: string) => {

const M: Record<string, number> = {

'I': 1,

'V': 5,

'X': 10,

'L': 50,

'C': 100,

'D': 500,

'M': 1000

};

let B = 0;

for (let k = 0; k < A.length; k++) {

if (k < A.length - 1 && M[A[k]] < M[A[k + 1]]) {

B -= M[A[k]];

} else {

B += M[A[k]];

}

}

console.log(B);

rl.close();

});

C#

using System;

using System.Collections.Generic;

class Program

{

static void Main()

{

string A = Console.ReadLine().ToUpper();

Dictionary<char, int> M = new Dictionary<char, int>

{

{'I', 1},

{'V', 5},

{'X', 10},

{'L', 50},

{'C', 100},

{'D', 500},

{'M', 1000}

};

int B = 0;

for (int k = 0; k < A.Length; k++)

{

if (k < A.Length - 1 && M[A[k]] < M[A[k + 1]])

{

B -= M[A[k]];

}

else

{

B += M[A[k]];

}

}

Console.WriteLine(B);

}

}

Assembly

section .data

roman\_map:

db 'I', 1, 0, 0, 0

db 'V', 5, 0, 0, 0

db 'X', 10, 0, 0, 0

db 'L', 50, 0, 0, 0

db 'C', 100, 0, 0, 0

db 'D', 500, 0, 0, 0

db 'M', 1000, 0, 0, 0

roman\_map\_size equ ($ - roman\_map) / 5

input\_prompt db "Введите римское число: ", 0

input\_buffer times 256 db 0

output\_format db "%d", 10, 0

section .text

global main

extern printf

extern scanf

extern strlen

main:

push rbp

mov rbp, rsp

sub rsp, 32

lea rdi, [input\_prompt]

xor eax, eax

call printf

lea rdi, [input\_buffer]

lea rsi, [input\_buffer]

xor eax, eax

call scanf

lea rdi, [input\_buffer]

call strlen

mov r12, rax

mov r13, 0

mov r14, 0

convert\_loop:

cmp r14, r12

jge convert\_done

lea rbx, [input\_buffer]

add rbx, r14

movzx r15, byte [rbx]

mov rcx, roman\_map

mov rdx, 0

find\_char:

cmp dl, byte [rcx]

je found\_char

add rcx, 5

inc dh

cmp dh, roman\_map\_size

jb find\_char

jmp next\_iter

found\_char:

movzx r8, byte [rcx+1]

mov r9, r14

inc r9

cmp r9, r12

jge add\_value

lea rbx, [input\_buffer]

add rbx, r9

movzx r10, byte [rbx]

mov rcx, roman\_map

mov rdx, 0

find\_next\_char:

cmp dl, byte [rcx]

je found\_next\_char

add rcx, 5

inc dh

cmp dh, roman\_map\_size

jb find\_next\_char

jmp add\_value

found\_next\_char:

movzx r9, byte [rcx+1]

cmp r8, r9

jge add\_value

sub r13, r8

jmp next\_iter

add\_value:

add r13, r8

next\_iter:

inc r14

jmp convert\_loop

convert\_done:

mov rdi, output\_format

mov rsi, r13

xor eax, eax

call printf

mov rsp, rbp

pop rbp

xor eax, eax

ret

26. Все числа из последовательности, которые составлены из цифр,

идущих по возрастанию необходимо перевернуть и вывести на экран.

Пример: 4 87 129 33 45

Результат: 921 54

C++

#include <iostream>

using namespace std;

int main() {

int n;

while (cin >> n) { //Читаем числа из ввода

int rev = 0 //Хранит правильное перевёрнутое число

int t = n //Рабочая копия числа

int prev = 10 //Предыдущая цифра

int ok = 1; //Флаг

//Проверяем, что цифры числа строго убывают справа налево

while (t) {

if (t % 10 >= prev) {

ok = 0;

break;

}

prev = t % 10; //Запоминаем текущую цифру как предыдущую

t /= 10; //Убираем последнюю цифру из числа

}

//Если цифры убывают и число не 0

if (ok && n != 0) {

t = n; //Снова делаем копию числа

rev = 0; //Сбрасываем перевернутое число

//Переворачиваем число

while (t) {

rev = rev \* 10 + t % 10;

t /= 10;

}

cout << rev << " ";

}

}

return 0;

}



GO

package main

import "fmt"

func main() {

var n int

for {

\_, err := fmt.Scan(&n)

if err != nil {

break

}

rev, t, prev, ok := 0, n, 10, true

for t > 0 {

if t%10 >= prev {

ok = false

break

}

prev = t % 10

t /= 10

}

if ok && n != 0 {

t = n

rev = 0

for t > 0 {

rev = rev\*10 + t%10

t /= 10

}

fmt.Print(rev, " ")

}

}

}

Java

import java.util.Scanner;

public class Main {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

while (scanner.hasNextInt()) {

int n = scanner.nextInt();

int rev = 0, t = n, prev = 10;

boolean ok = true;

while (t > 0) {

if (t % 10 >= prev) {

ok = false;

break;

}

prev = t % 10;

t /= 10;

}

if (ok && n != 0) {

t = n;

rev = 0;

while (t > 0) {

rev = rev \* 10 + t % 10;

t /= 10;

}

System.out.print(rev + " ");

}

}

scanner.close();

}

}

JS

(function() {

let input = prompt("Введите числа через пробел:");

if (input) {

let numbers = input.trim().split(/\s+/).map(Number);

let output = [];

for (let n of numbers) {

let rev = 0, t = n, prev = 10, ok = 1;

let temp = t;

while (temp > 0) {

if (temp % 10 >= prev) ok = 0;

prev = temp % 10;

temp = Math.floor(temp / 10);

}

if (ok && prev < 10) {

rev = 0;

temp = n;

while (temp > 0) {

rev = rev \* 10 + temp % 10;

temp = Math.floor(temp / 10);

}

output.push(rev);

}

}

console.log(output.join(" "));

}

})();

Kotlin

fun main() {

val input = readLine() ?: return

val numbers = input.split(" ").mapNotNull { it.toIntOrNull() }

for (n in numbers) {

var t = n

var prev = 10

var ok = true

while (t != 0) {

if (t % 10 >= prev) {

ok = false

break

}

prev = t % 10

t /= 10

}

if (ok && n != 0) {

t = n

var rev = 0

while (t != 0) {

rev = rev \* 10 + t % 10

t /= 10

}

print("$rev ")

}

}

}

PHP

<?php

while (fscanf(STDIN, "%d", $n) == 1) {

$rev = 0;

$t = $n;

$prev = 10;

$ok = 1;

while ($t != 0) {

if ($t % 10 >= $prev) {

$ok = 0;

break;

}

$prev = $t % 10;

$t = (int)($t / 10);

}

if ($ok && $n != 0) {

$t = $n;

$rev = 0;

while ($t != 0) {

$rev = $rev \* 10 + $t % 10;

$t = (int)($t / 10);

}

echo $rev . " ";

}

}

?>

Python

import sys

for line in sys.stdin:

numbers = map(int, line.split())

for n in numbers:

rev = 0

t = n

prev = 10

ok = 1

while t:

if t % 10 >= prev:

ok = 0

break

prev = t % 10

t = t // 10

if ok and n != 0:

t = n

rev = 0

while t:

rev = rev \* 10 + t % 10

t = t // 10

print(rev, end=' ')

Ruby

while (n = gets.to\_i) != 0

t = n

prev = 10

ok = true

while t > 0

if t % 10 >= prev

ok = false

break

end

prev = t % 10

t /= 10

end

if ok && n != 0

t = n

rev = 0

while t > 0

rev = rev \* 10 + t % 10

t /= 10

end

print "#{rev} "

end

end

Rust

use std::io;

use std::io::BufRead;

fn main() {

let stdin = io::stdin();

for line in stdin.lock().lines() {

if let Ok(line) = line {

for word in line.split\_whitespace() {

if let Ok(n) = word.parse::<u32>() {

let mut t = n;

let mut prev = 10;

let mut ok = true;

while t > 0 {

if t % 10 >= prev {

ok = false;

break;

}

prev = t % 10;

t /= 10;

}

if ok && n != 0 {

t = n;

let mut rev = 0;

while t > 0 {

rev = rev \* 10 + t % 10;

t /= 10;

}

print!("{} ", rev);

}

}

}

}

}

}

Swift

import Foundation

let input = readLine()?.split(separator: " ").compactMap { Int($0) } ?? []

for n in input {

var rev = 0

var t = n

var prev = 10

var ok = true

while t != 0 {

if t % 10 >= prev {

ok = false

break

}

prev = t % 10

t /= 10

}

if ok && n != 0 {

t = n

rev = 0

while t != 0 {

rev = rev \* 10 + t % 10

t /= 10

}

print(rev, terminator: " ")

}

}

TypeScript

function main(): void {

const readline = require('readline');

const rl = readline.createInterface({

input: process.stdin,

output: process.stdout,

terminal: false

});

rl.on('line', (line: string) => {

const numbers = line.trim().split(/\s+/).filter(s => s.length > 0);

for (const numStr of numbers) {

const n = parseInt(numStr, 10);

if (isNaN(n)) continue;

let t = n;

let prev = 10;

let ok = true;

while (t > 0) {

const digit = t % 10;

if (digit >= prev) {

ok = false;

break;

}

prev = digit;

t = Math.floor(t / 10);

}

if (ok && n !== 0) {

t = n;

let rev = 0;

while (t > 0) {

rev = rev \* 10 + t % 10;

t = Math.floor(t / 10);

}

process.stdout.write(rev + " ");

}

}

});

}

main();

C#

using System;

using System.Linq;

class Program

{

static void Main()

{

string line;

while ((line = Console.ReadLine()) != null)

{

var numbers = line.Split(new[] { ' ', '\t' }, StringSplitOptions.RemoveEmptyEntries);

foreach (var numStr in numbers)

{

if (int.TryParse(numStr, out int n))

{

int rev = 0, t = n, prev = 10, ok = 1;

while (t != 0)

{

if (t % 10 >= prev)

{

ok = 0;

break;

}

prev = t % 10;

t /= 10;

}

if (ok != 0 && n != 0)

{

t = n;

rev = 0;

while (t != 0)

{

rev = rev \* 10 + t % 10;

t /= 10;

}

Console.Write(rev + " ");

}

}

}

}

}

}

Assembly

.section .rodata

input\_format: .asciz "%d"

output\_format: .asciz "%d "

.section .text

.globl main

main:

pushq %rbp

movq %rsp, %rbp

subq $16, %rsp

read\_loop:

leaq -4(%rbp), %rsi

leaq input\_format(%rip), %rdi

xorl %eax, %eax

call scanf@PLT

cmpl $1, %eax

jne exit

movl -4(%rbp), %ebx

testl %ebx, %ebx

je read\_loop

movl %ebx, %ecx

movl $10, %edx

movl $1, %esi

check\_digits:

testl %ecx, %ecx

je check\_ok

movl %ecx, %eax

movl $10, %edi

xorl %r8d, %r8d

divl %edi

cmpl %edx, %edx

jl digit\_ok

movl $0, %esi

jmp check\_ok

digit\_ok:

movl %edx, %edx

movl %eax, %ecx

jmp check\_digits

check\_ok:

testl %esi, %esi

je read\_loop

movl %ebx, %ecx

xorl %r9d, %r9d

reverse\_loop:

testl %ecx, %ecx

je print\_rev

movl %r9d, %eax

movl $10, %edi

imull %edi

movl %ecx, %r8d

xorl %edx, %edx

divl %edi

addl %edx, %eax

movl %eax, %r9d

movl %ecx, %eax

xorl %edx, %edx

divl %edi

movl %eax, %ecx

jmp reverse\_loop

print\_rev:

movl %r9d, %esi

leaq output\_format(%rip), %rdi

xorl %eax, %eax

call printf@PLT

jmp read\_loop

exit:

xorl %eax, %eax

leave

ret

Вывод:

В ходе лабораторной работы по классификации языков программирования я изучил особенности разных языков и написал на них простые программы.