**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ**

федеральное государственное автономное образовательное учреждение высшего образования

**НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ**

**ТОМСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ**

Инженерная школа энергетики

Отделение электроэнергетики и электротехники

Направление: 09.04.03 Прикладная информатика

Дисциплина: Программирование и основы алгоритмизации

Отчет

по лабораторной работе №6

**Строки и регулярные выражения**

Вариант 12

|  |  |  |
| --- | --- | --- |
| Выполнил:  студент гр. О-5КМ41 | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись) | Рустамов Р.А. |
| Проверил: | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись)  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (дата) | Гончаров А.С. |
|  | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (балл) |  |

Томск-2025

# ХОД РАБОТЫ

Основной код программы представлен конструкцией Switch case.

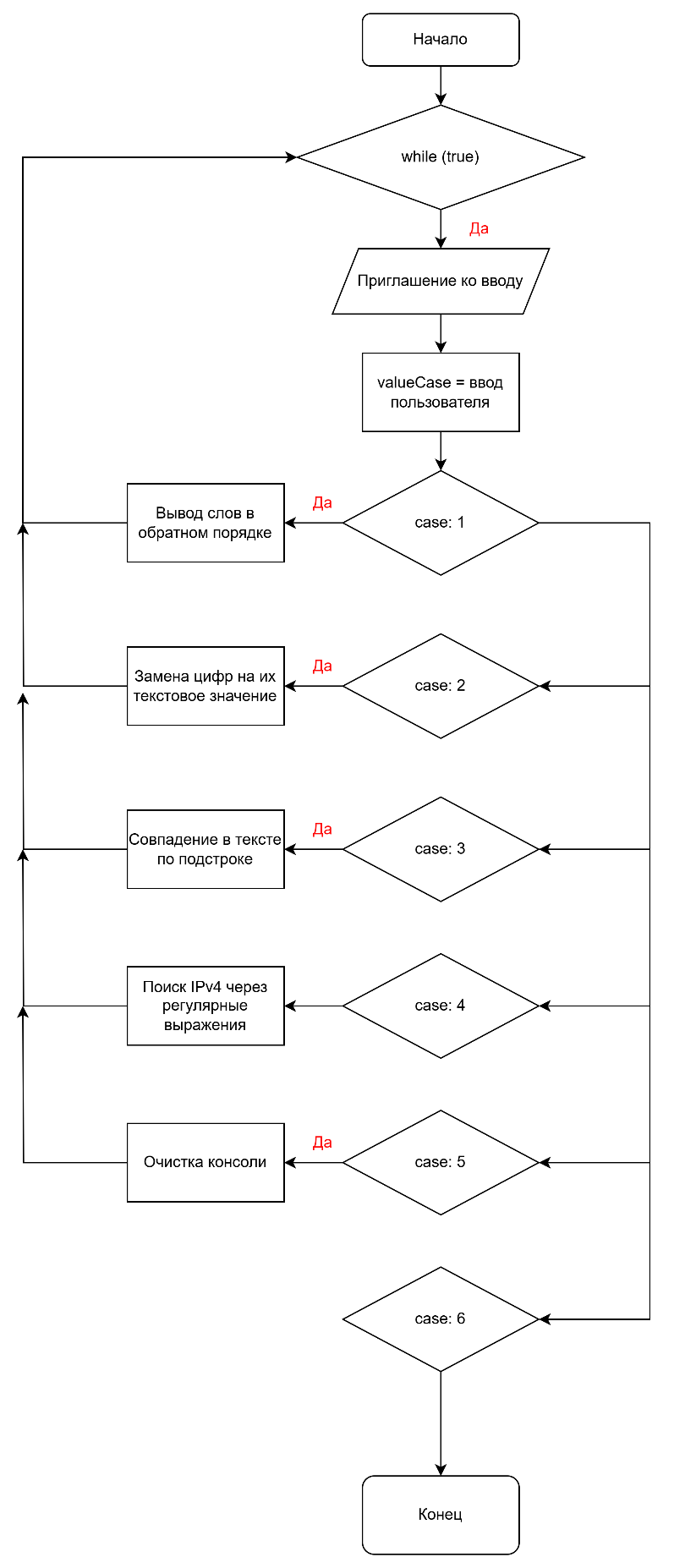


Рисунок – Блок схема основного тела программы

Задание 1: Написать программу, которая вводит текст, состоящий из нескольких предложений, и выводит каждое слово в обратном порядке.

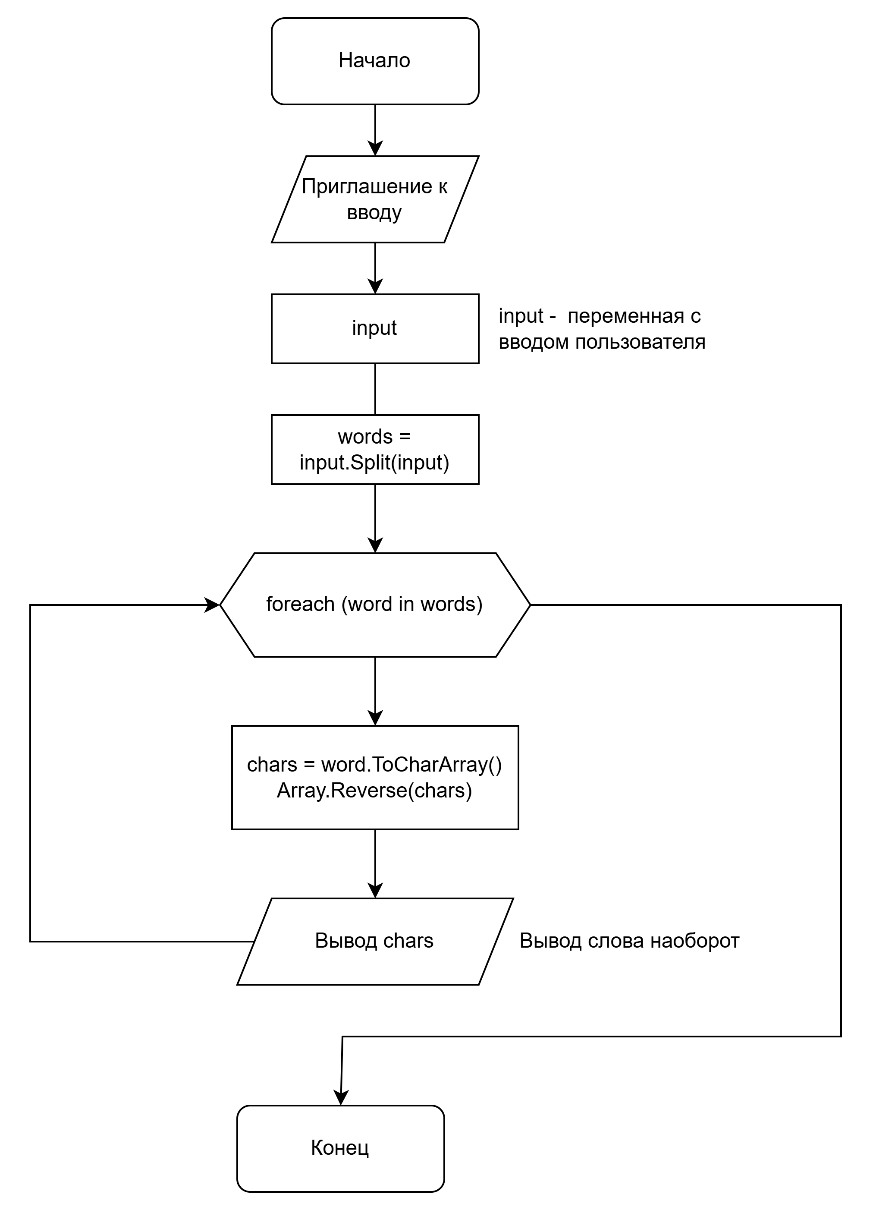


Рисунок – Блок кода по реверсированию символов текста

|  |  |
| --- | --- |
|  |  |

Рисунок 3 – Программная реализация реверсии слов

Программа. Из данной строки сделать новую строку, заменив в ней все цифры на соответствующие слова: "один", "два", "три" и т.д.

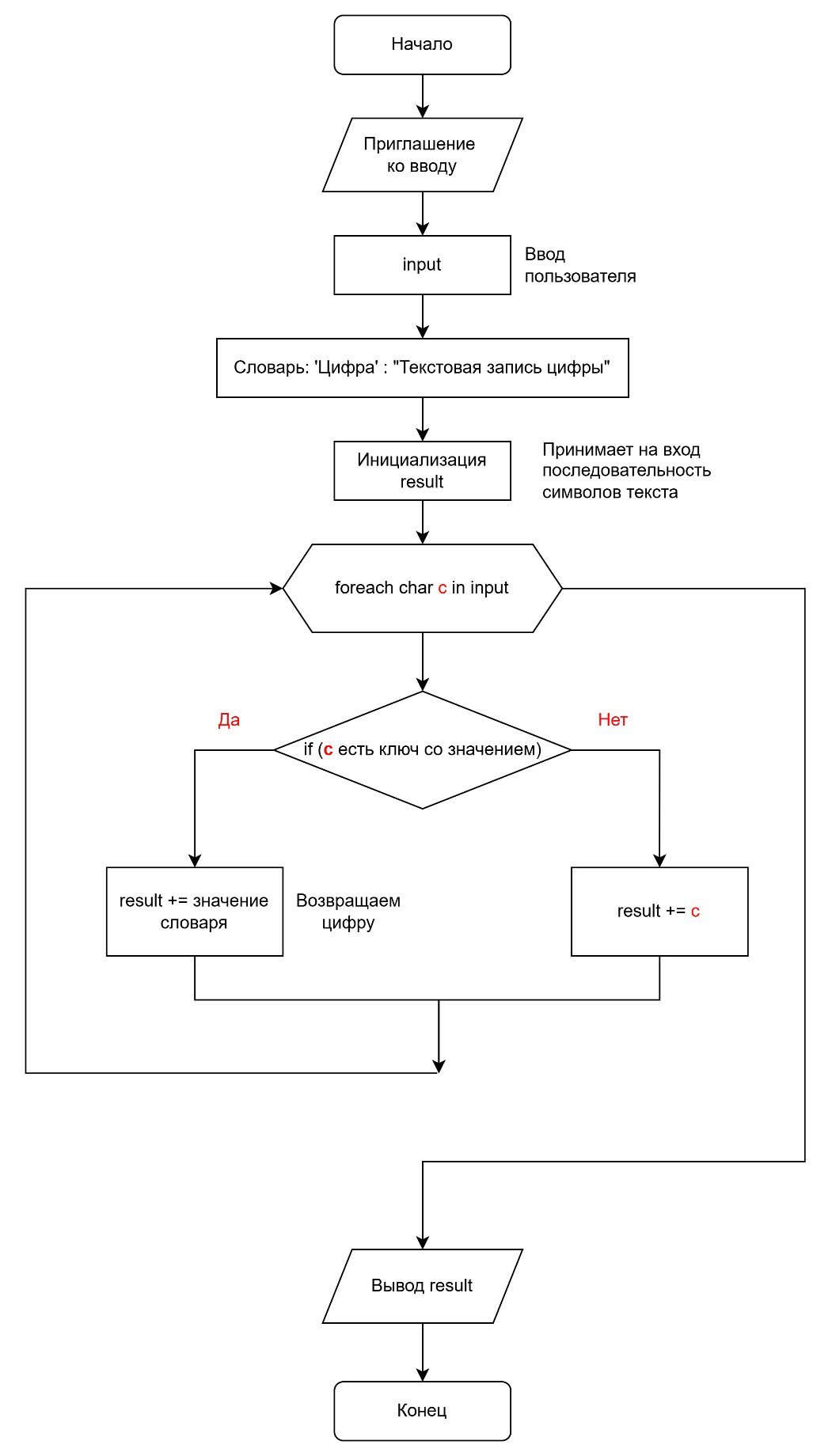


Рисунок – Блок кода программы по замене численных значений цифр на текстовые

|  |  |
| --- | --- |
|  |  |

Рисунок – Пример работы программы по замене численных значений цифр их текстовыми

Задача 2: Дан текст и строка s, Переписать в новый файл g все строки файла f, содержащие значения переменной s в качестве подстроки.

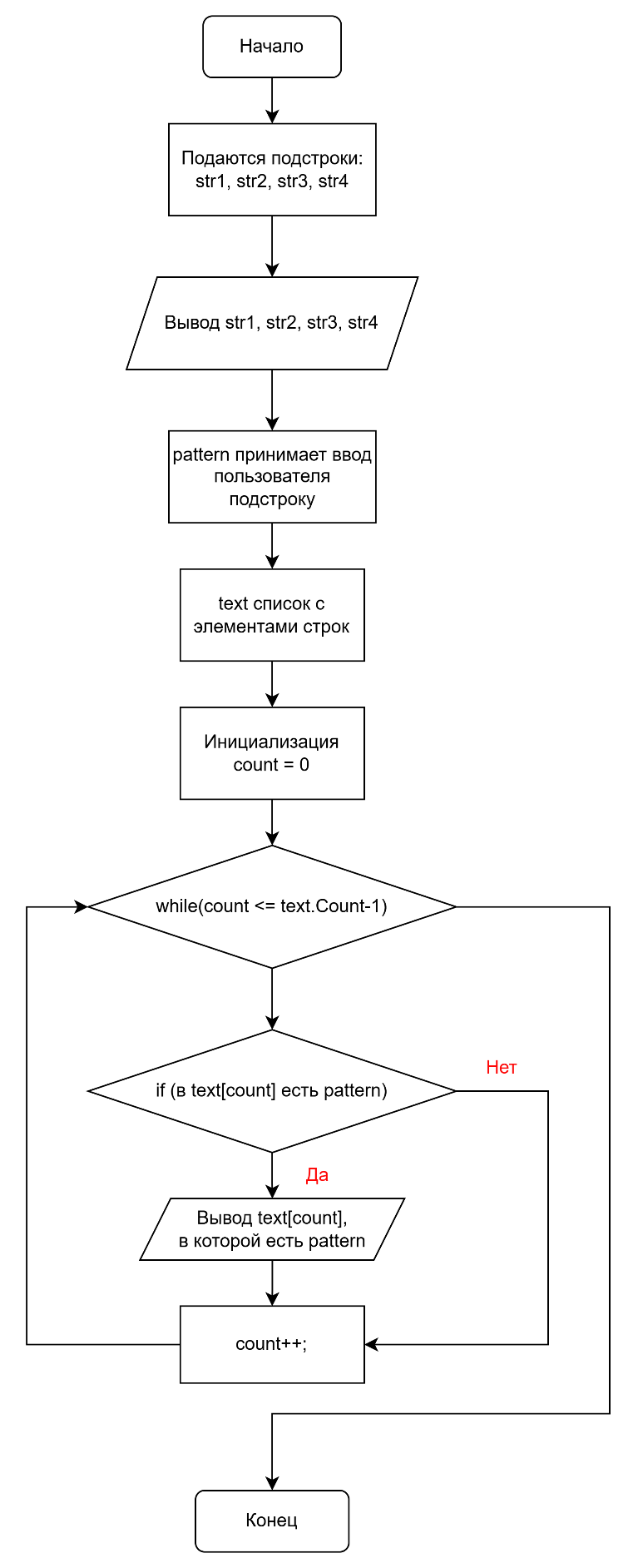


Рисунок – Программа поиска подстроки в других подстроках

Выбрать IPv4 адреса во всех возможных, представлениях: десятичном, шестнадцатеричном и восьмеричном. С точками и без.

|  |  |
| --- | --- |
|  |  |

Рисунок – Пример работы программы по поиску подстроки

Далее внизу представлен псевдокод работы программы с использованием регулярных выражений, позволяющего определить принадлежность IPv4 определенной СИ. С точкой и без.

BEGIN

# 1) Тестовые данные: IPv4 в десятичном/восьмеричном/шестнадцатеричном виде,

# как с точками, так и без (целые 32-битные числа)

textIPv4 ← "<длинная строка из примера>"

PRINT textIPv4

# ---------- A. Поиск адресов с точками ----------

PRINT "\n10-e"

DEC\_DOTTED\_REGEX ←

word\_boundary

( (25[0-5] | 2[0-4]\d | 1\d{2} | [1-9]\d?) \. )

( (25[0-5] | 2[0-4]\d | 1\d{2} | [0-9]\d?) \. ){2}

( (25[0-5] | 2[0-4]\d | 1\d{2} | [0-9]\d?) )

word\_boundary

FOR each match m IN REGEX\_MATCHES(textIPv4, DEC\_DOTTED\_REGEX) DO

PRINT m.value

END FOR

PRINT "\n8-e"

OCT\_DOTTED\_REGEX ←

word\_boundary

( (37[0-7] | 3[0-6][0-7] | 2[0-7]{2} | 1[0-7]{2} | [0-7]{2}[1-7]) \. )

( (37[0-7] | 3[0-6][0-7] | 2[0-7]{2} | 1[0-7]{2} | [0-7]{2}[0-7]) \. ){2}

( (37[0-7] | 3[0-6][0-7] | 2[0-7]{2} | 1[0-7]{2} | [0-7]{3}) )

word\_boundary

FOR each match m IN REGEX\_MATCHES(textIPv4, OCT\_DOTTED\_REGEX) DO

PRINT m.value

END FOR

PRINT "\n16-e"

textLower ← TO\_LOWER(textIPv4)

HEX\_DOTTED\_REGEX ←

( ([a-f]{2} | [a-f][0-9] | [1-9][a-f] | [0-9]{2}) \. )

( ([a-f]{2} | [a-f][0-9] | [0-9][a-f] | [0-9]{2}) \. ){2}

( ([a-f]{2} | [a-f][0-9] | [0-9][a-f] | [0-9]{2}) whitespace )

FOR each match m IN REGEX\_MATCHES(textLower, HEX\_DOTTED\_REGEX) DO

PRINT m.value

END FOR

# ---------- B. Поиск "цельных" представлений без точек ----------

PRINT "IP-адреса без точки: "

# Числа между пробелами: либо десятичные, либо шестнадцатеричные (с опц. префиксом 0x),

# длиной до 8 hex-символов (т.е. до 32 бит)

pattern ←

( whitespace (digits+) whitespace )

OR

( whitespace (optional "0x") [0-9A-Fa-f]{1,8} whitespace )

FOR each match m IN REGEX\_MATCHES(textLower, pattern) DO

found ← FALSE

bases ← [10, 8, 16]

FOR baseX IN bases DO

(isValid, detectedBase) ← Check(m.value, baseX)

IF isValid THEN

PRINT m.value

found ← TRUE

BREAK # из цикла по системам счисления

END IF

END FOR

# (ничего не выводим, если не прошло валидацию ни в одной базе)

END FOR

END

Функция для сортировки значений без точки не принадлежащих IPv4.

FUNCTION Check(rawToken, baseX) RETURNS (bool isValid, int detectedBase)

# 1) Нормализовать строку (обрезать пробелы)

token ← TRIM(rawToken)

# 2) Попробовать интерпретировать token как целое без знака в системе baseX:

# - для baseX=10: только цифры 0–9

# - для baseX=8 : только 0–7 (без недопустимых вроде '8'/'9')

# - для baseX=16: [0–9a–f], допустим опциональный префикс 0x/0X

# Если формат не соответствует — вернуть (FALSE, 0)

ok, n ← TRY\_PARSE\_UNSIGNED(token, baseX)

IF NOT ok THEN

RETURN (FALSE, 0)

END IF

# 3) Проверить диапазон 32-битного IPv4: 0 ≤ n ≤ 0xFFFFFFFF

IF 0 ≤ n ≤ 4294967295 THEN

RETURN (TRUE, baseX)

ELSE

RETURN (FALSE, 0)

END IF

END FUNCTION

Код программы приведен ниже:

string textIPv4 = "192.168.1.10 -dec 300.250.001.012 -oct c0.a8.01.0a -hex 127.0.0.1 -dec 177.000.000.001 -oct 7f.00.00.01 -hex 8.8.8.8 -dec 010.010.010.010 -oct"

+ " 08.08.08.08 -hex 255.255.255.255 -dec 377.377.377.377 -oct ff.ff.ff.ff -hex 999.999.999.999 -wrong 400.1.1.1 -wrong 0x100.0xa8.0x01.0x0a hex-with - 0x"

+ "127.0.0.1\r\n\r\ndec: 2130706433\r\n\r\nhex: 0x7F000001 / 7F000001\r\n\r\noct: 017700000001\r\n\r\n8.8.8.8\r\n\r\n"

+ "dec: 134744072\r\n\r\nhex: 0x08080808 / 08080808\r\n\r\noct: 01010101010\r\n\r\n255.255.255.255\r\n\r\n"

+ "dec: 4294967295\r\n\r\nhex: 0xFFFFFFFF / FFFFFFFF\r\n\r\noct: 037777777777\r\n\r\n192.168.1.10\r\n\r\n"

+ "dec: 500000000000\r\n\r\nhex: 0xFFFFFFFFF / FFFFFFFFF\r\n\r\noct: 04444444445\r\n\r\n292.168.1.10\r\n\r\n" // неправильные IP адреса

+ "dec: 3232235786\r\n\r\nhex: 0xC0A8010A / C0A8010A\r\n\r\noct: 030052000012 ";

;

Console.WriteLine(textIPv4);

Console.WriteLine("\n10-e");

foreach (Match m in Regex.Matches(textIPv4, @"\b((25[0-5]|2[0-4]\d|1\d{2}|[1-9]\d?)**\.**)((25[0-5]|2[0-4]\d|1\d{2}|[0-9]\d?)**\.**){2}((25[0-5]|2[0-4]\d|1\d{2}|[0-9]\d?))\b")) // 10-я система

{

Console.WriteLine(m.Value);

}

Console.WriteLine("\n8-e");

foreach (Match m in Regex.Matches(textIPv4, @"\b((37[0-7]|3[0-6][0-7]|2[0-7]{2}|1[0-7]{2}|[0-7]{2}[1-7])**\.**)((37[0-7]|3[0-6][0-7]|2[0-7]{2}|1[0-7]{2}|[0-7]{2}[0-7])**\.**){2}((37[0-7]|3[0-6][0-7]|2[0-7]{2}|1[0-7]{2}|[0-7]{3}))\b")) // 8-я система

{

Console.WriteLine(m.Value);

}

Console.WriteLine("\n16-e");

foreach (Match m in Regex.Matches(textIPv4.ToLower(), @"(([a-f]{2}|[a-f][0-9]|[1-9][a-f]|[0-9]{2})**\.**)(([a-f]{2}|[a-f][0-9]|[0-9][a-f]|[0-9]{2})**\.**){2}(([a-f]{2}|[a-f][0-9]|[0-9][a-f]|[0-9]{2})\s)")) // 16-я система

{

Console.WriteLine(m.Value);

}

Console.WriteLine("IP-адреса без точки: ");

pattern = @"(\s(\d+)\s)|(\s(?:0x)?[0-9A-Fa-f]{1,8}\s)";

foreach (Match m in Regex.Matches(textIPv4.ToLower(), pattern))

{

//Console.WriteLine();

bool check;

int baseFrom = 0;

int[] array = new int[] { 10, 8, 16 };

foreach (int baseX in array)

{

(check, baseFrom) = Check(m.Value, baseX);

if (check)

{

Console.WriteLine(m.Value);

break;

}

}

}

# Вывод

В ходе выполнения лабораторной работы № 6 были изучены:

* Строковые типы данных и алгоритмы работы с ними.
* Регулярные выражения.ПРИЛОЖЕНИЕ

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Text.RegularExpressions;

using System.Threading; // Библиотека для регулярных выражений

namespace Sixth\_lab

{

internal class Program

{

static int ReadInteger(string text)

{

Console.Write(text + " ");

int intValue;

while (!int.TryParse(Console.ReadLine(), out intValue))

{

Console.ForegroundColor = ConsoleColor.Red;

Console.WriteLine("Введите целое число!");

Console.ResetColor();

Console.Write("\n" + text + " ");

}

//Console.WriteLine();

return intValue;

}

// Отсеиваем числовые значения байт которых больше 32

static (bool, int) Check(string value, int baseFrom)

{

value = value.Trim(new char[] { ' ', '\r' });

try

{

// Преобразование value в СИ baseFrom

var digit = Convert.ToInt64(value, baseFrom);

string binary = Convert.ToString(digit, 2);

if (binary.Length > 32)

{

return (false, 0);

}

//Console.WriteLine(digit + $" {baseFrom}-е число");

//string oct = Convert.ToString(digit, 8);

//string hex = Convert.ToString(digit, 16);

//Console.WriteLine("2:" + binary);

//Console.WriteLine("8:" + oct);

//Console.WriteLine("16:" + hex);

return (true, baseFrom);

}

catch

{

return (false, 0);

}

}

static void Main(string[] args)

{

string input;

while (true)

{

Console.WriteLine("1. Вывести слова в обратном порядке в предложении.\n"

+ "2. Заменить все цифры на их текстовые значения\n"

+ "3. Совпадения в тексте по подстроке\n"

+ "4. IPv4 регулярные выражения\n"

+ "5. Очистить консоль\n"

+ "6. Завершить программу."

);

int valueCase = ReadInteger("Выберите вариант из списка.");

switch (valueCase)

{

case 1:

/\*

Написать программу, которая вводит текст, состоящий из нескольких предложений,

и выводит каждое слово в обратном порядке.

\*/

Console.ForegroundColor = ConsoleColor.Green;

Console.WriteLine("\nВведите текст:");

Console.ResetColor();

input = Console.ReadLine();

string[] words = input.Split(

new char[] { ' ', ',', '!', '?', ';', ':' },

StringSplitOptions.RemoveEmptyEntries

);

Console.ForegroundColor = ConsoleColor.Green;

Console.WriteLine("\nСлова в обратном порядке:");

Console.ResetColor();

foreach (string word in words)

{

char[] chars = word.ToCharArray();

Array.Reverse(chars);

Console.Write(new string(chars) + " ");

}

Console.WriteLine();

break;

case 2:

/\*

\* Из данной строки сделать новую строку, заменив в ней все цифры

\* на соответствующие слова: "один", "два", "три" и т.д.

\*/

Console.WriteLine();

Console.ForegroundColor = ConsoleColor.Green;

Console.WriteLine("Введите строку с цифрами:");

Console.ResetColor();

input = Console.ReadLine();

Dictionary<char, string> digits = new Dictionary<char, string>

{

{'0', "ноль"},

{'1', "один"},

{'2', "два"},

{'3', "три"},

{'4', "четыре"},

{'5', "пять"},

{'6', "шесть"},

{'7', "семь"},

{'8', "восемь"},

{'9', "девять"}

};

string result = "";

foreach (char c in input)

{

if (digits.ContainsKey(c))

result += digits[c] + " ";

else

result += c;

}

Console.WriteLine("\nРезультат:");

Console.WriteLine(result);

break;

case 3:

// Работа с регулярными выражениями

/\*

\* Дан текст и строка s, Переписать в новый файл g все строки файла f,

\* содержащие значения переменной s в качестве подстроки.

\*/

string str1 = "Машина стоит у дома.";

string str2 = "Сегодня солнечный день.";

string str3 = "Дом построен в прошлом году.";

string str4 = "Собака бежит по улице.";

Console.WriteLine($"\n{str1}\n{str2}\n{str3}\n{str4}");

Console.Write("\nВыберете подстроку из текста: ");

string pattern = Console.ReadLine();

List<string> text = new List<string>() {str1, str2, str3, str4};

int count = 0;

while (count <= text.Count - 1)

{

if (Regex.IsMatch(text[count].ToLower(), pattern.ToLower()))

Console.WriteLine(text[count]);

count++;

}

break;

case 4:

/\*

\* Выбрать IPv4 адреса во всех возможных, представлениях: десятичном,

\* шестнадцатеричном и восьмеричном. С точками и без.

\*/

string textIPv4 = "192.168.1.10 -dec 300.250.001.012 -oct c0.a8.01.0a -hex 127.0.0.1 -dec 177.000.000.001 -oct 7f.00.00.01 -hex 8.8.8.8 -dec 010.010.010.010 -oct"

+ " 08.08.08.08 -hex 255.255.255.255 -dec 377.377.377.377 -oct ff.ff.ff.ff -hex 999.999.999.999 -wrong 400.1.1.1 -wrong 0x100.0xa8.0x01.0x0a hex-with - 0x"

+ "127.0.0.1\r\n\r\ndec: 2130706433\r\n\r\nhex: 0x7F000001 / 7F000001\r\n\r\noct: 017700000001\r\n\r\n8.8.8.8\r\n\r\n"

+ "dec: 134744072\r\n\r\nhex: 0x08080808 / 08080808\r\n\r\noct: 01010101010\r\n\r\n255.255.255.255\r\n\r\n"

+ "dec: 4294967295\r\n\r\nhex: 0xFFFFFFFF / FFFFFFFF\r\n\r\noct: 037777777777\r\n\r\n192.168.1.10\r\n\r\n"

+ "dec: 500000000000\r\n\r\nhex: 0xFFFFFFFFF / FFFFFFFFF\r\n\r\noct: 04444444445\r\n\r\n292.168.1.10\r\n\r\n" // неправильные IP адреса

+ "dec: 3232235786\r\n\r\nhex: 0xC0A8010A / C0A8010A\r\n\r\noct: 030052000012 ";

;

Console.WriteLine(textIPv4);

Console.WriteLine("\n10-e");

foreach (Match m in Regex.Matches(textIPv4, @"\b((25[0-5]|2[0-4]\d|1\d{2}|[1-9]\d?)**\.**)((25[0-5]|2[0-4]\d|1\d{2}|[0-9]\d?)**\.**){2}((25[0-5]|2[0-4]\d|1\d{2}|[0-9]\d?))\b")) // 10-я система

{

Console.WriteLine(m.Value);

}

Console.WriteLine("\n8-e");

foreach (Match m in Regex.Matches(textIPv4, @"\b((37[0-7]|3[0-6][0-7]|2[0-7]{2}|1[0-7]{2}|[0-7]{2}[1-7])**\.**)((37[0-7]|3[0-6][0-7]|2[0-7]{2}|1[0-7]{2}|[0-7]{2}[0-7])**\.**){2}((37[0-7]|3[0-6][0-7]|2[0-7]{2}|1[0-7]{2}|[0-7]{3}))\b")) // 8-я система

{

Console.WriteLine(m.Value);

}

Console.WriteLine("\n16-e");

foreach (Match m in Regex.Matches(textIPv4.ToLower(), @"(([a-f]{2}|[a-f][0-9]|[1-9][a-f]|[0-9]{2})**\.**)(([a-f]{2}|[a-f][0-9]|[0-9][a-f]|[0-9]{2})**\.**){2}(([a-f]{2}|[a-f][0-9]|[0-9][a-f]|[0-9]{2})\s)")) // 16-я система

{

Console.WriteLine(m.Value);

}

Console.WriteLine("IP-адреса без точки: ");

pattern = @"(\s(\d+)\s)|(\s(?:0x)?[0-9A-Fa-f]{1,8}\s)";

foreach (Match m in Regex.Matches(textIPv4.ToLower(), pattern))

{

//Console.WriteLine();

bool check;

int baseFrom = 0;

int[] array = new int[] { 10, 8, 16 };

foreach (int baseX in array)

{

(check, baseFrom) = Check(m.Value, baseX);

if (check)

{

Console.WriteLine(m.Value);

break;

}

}

}

break;

case 5:

Console.Clear();

break;

case 6:

return;

}

}

}

}

}