**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 №7

**Структуры и файлы**

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. В программе используются структуры Mash и Date, которые расположены в namespace SeventhLab

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

Рисунок 1 – Блок схема основного тела программы

Задание 1:

Дана структура, задающая дату вида:

struct date

{

int day;

int month;

int year;

};

Пользуясь таким структурным типом, составить программу, определяющую:

а) дату следующего (относительно сегодняшнего) дня.

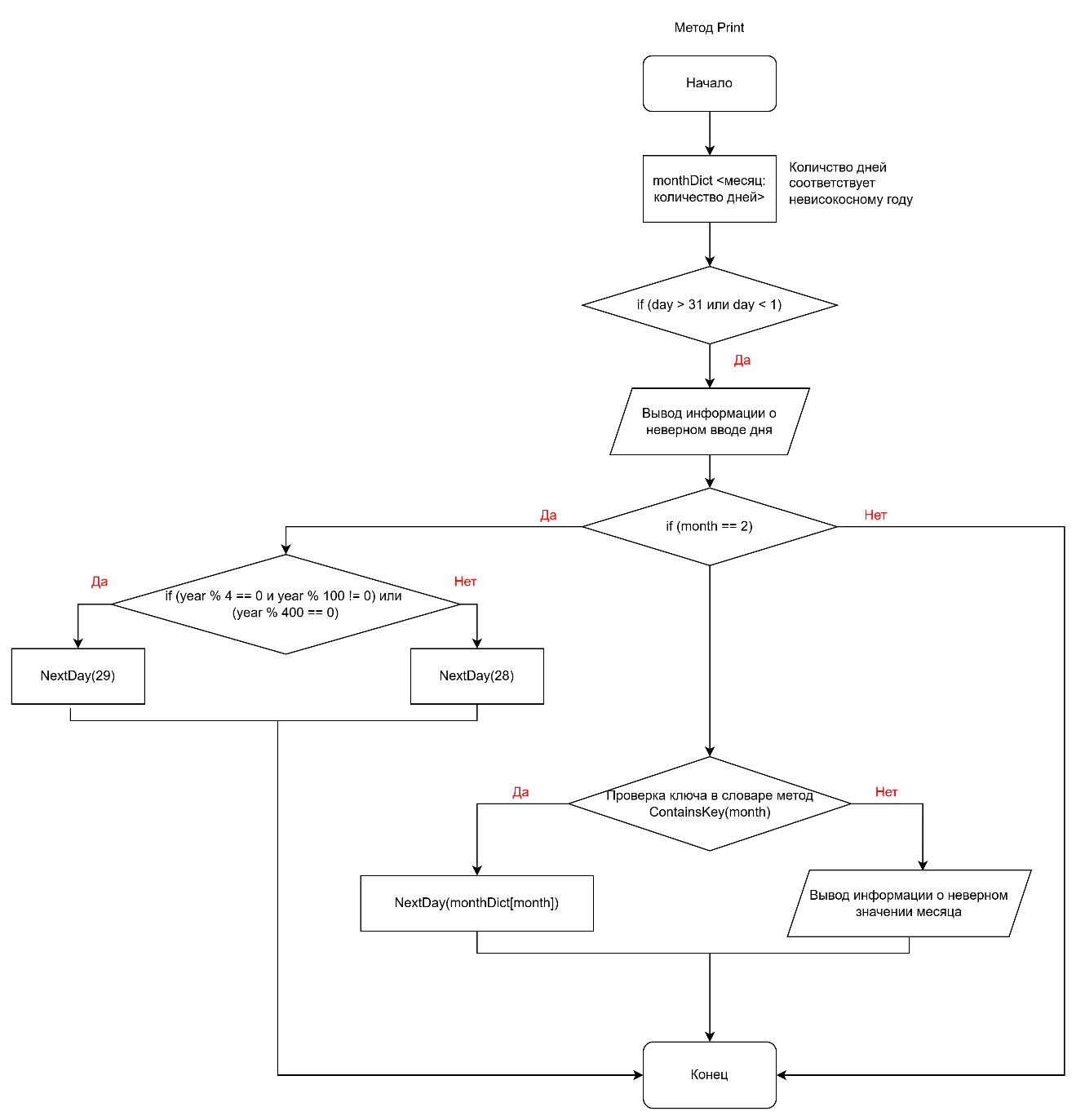


Рисунок 2 – Реализация метода Print в структуре Date

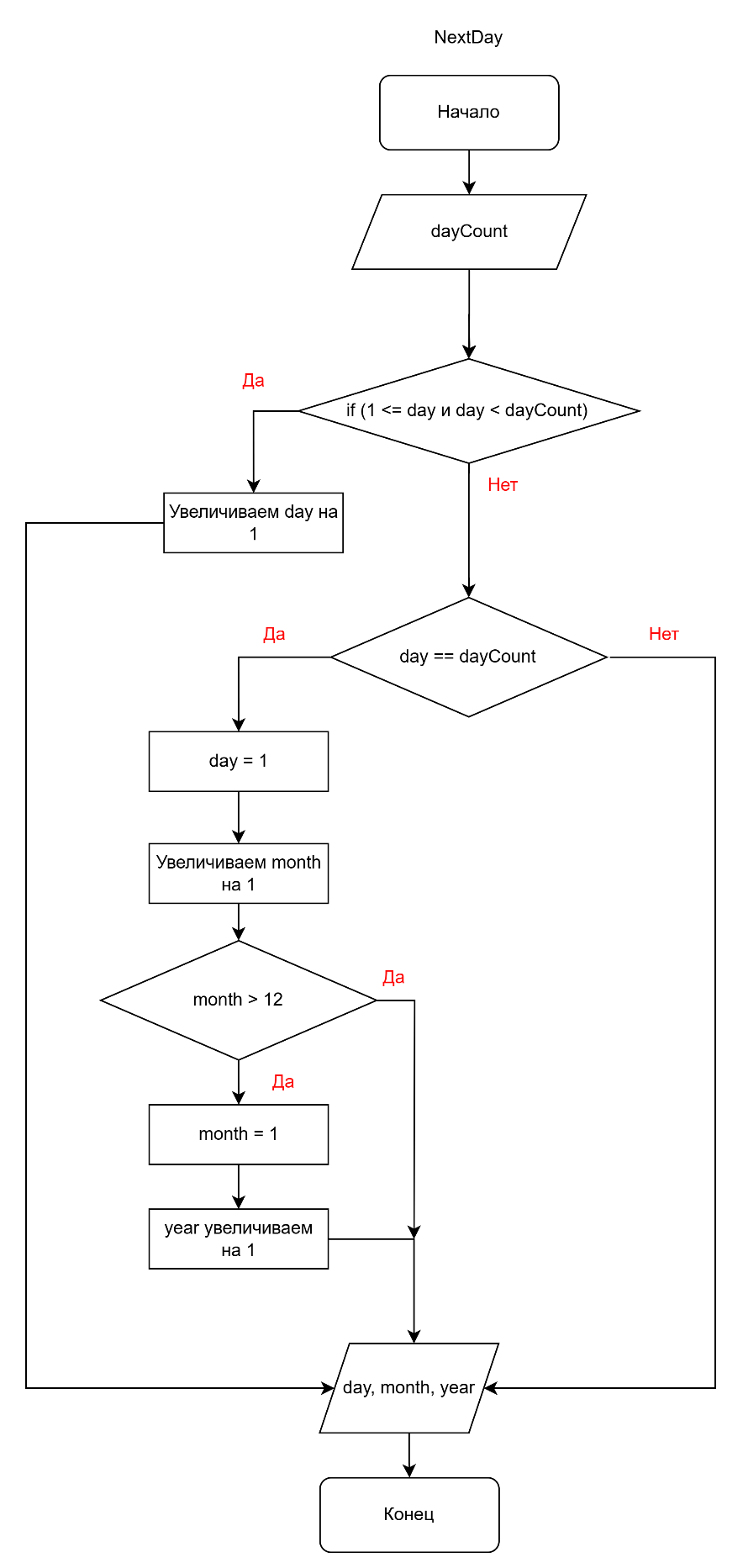


Рисунок 3 – Реализация метода NextDay в структуре Date

|  |  |
| --- | --- |
|  |  |
| а | б |
|  |  |
| в | г |

Рисунок 4 – Программный вывод по предсказанию следующего дня

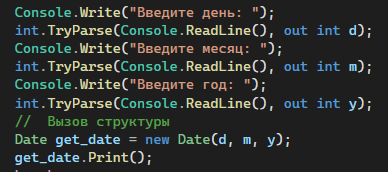


Рисунок 5 – Вызов программы в основном теле

Задача 2: Дана структура с именем MARSH, содержащая следующие поля:

* Название начального пункта назначения.
* Название конечного пункта назначения.
* Номер маршрута.

Написать программу, которая выполняет следующие действия:

* Ввод с клавиатуры данных в массив, состоящий из 8 элементов типа MARSH, и занесение их в файл данных.
* Чтение данных из файла и вывод их на экран.
* Вывод на экран информации о маршруте, номер которого введен с клавиатуры (если таких нет — вывести об этом сообщение).
* Список должен быть упорядочен по номерам маршрутов.

|  |  |
| --- | --- |
|  |  |
| а | б |

Рисунок 6 – Блок кода по поиску маршрута в основном теле программы

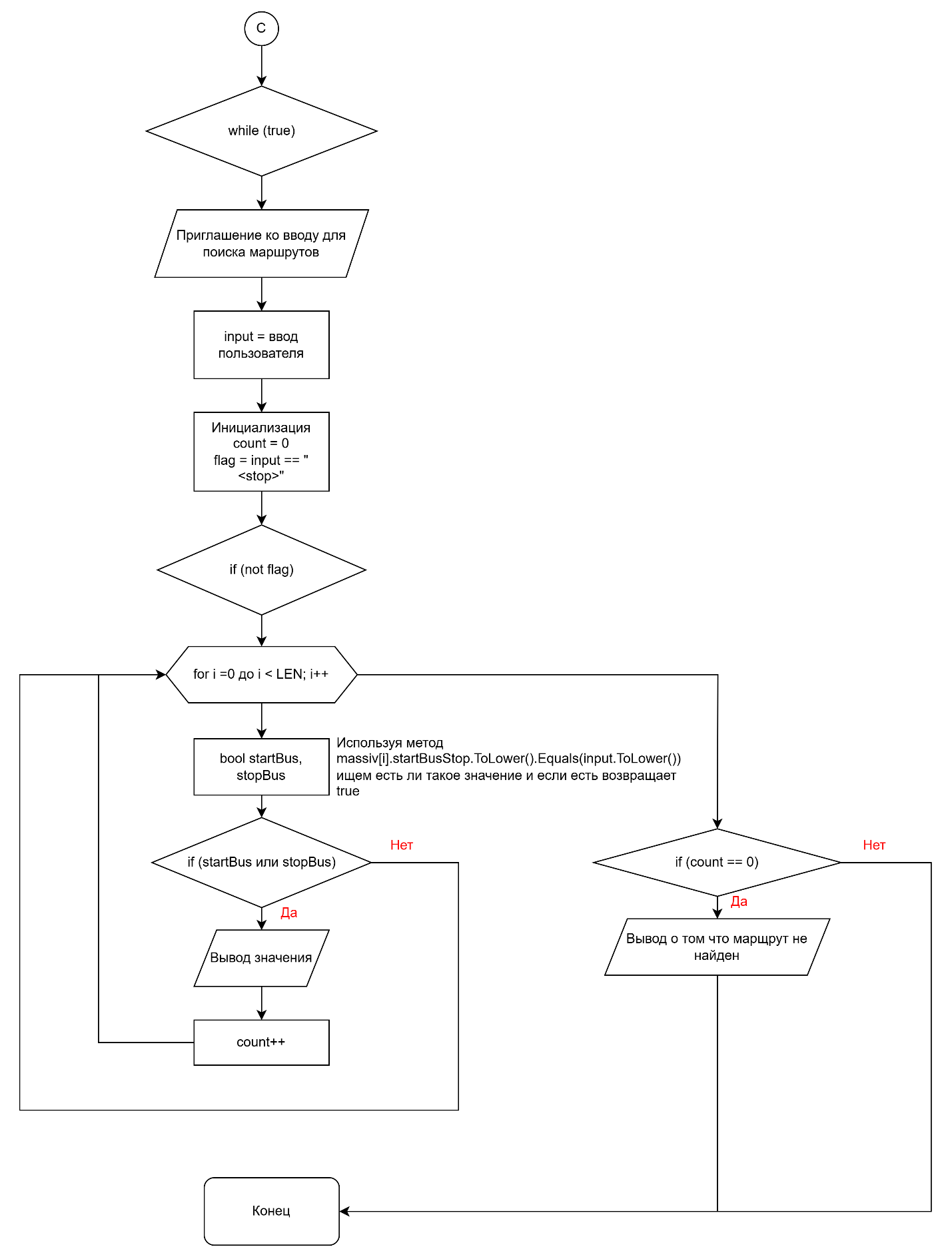


Рисунок 7 – Блок кода по поиску маршрута завершение

# 

Рисунок 8 –Блок кода метода структуры Mash по созданию массива с номером маршрута, начала и конца

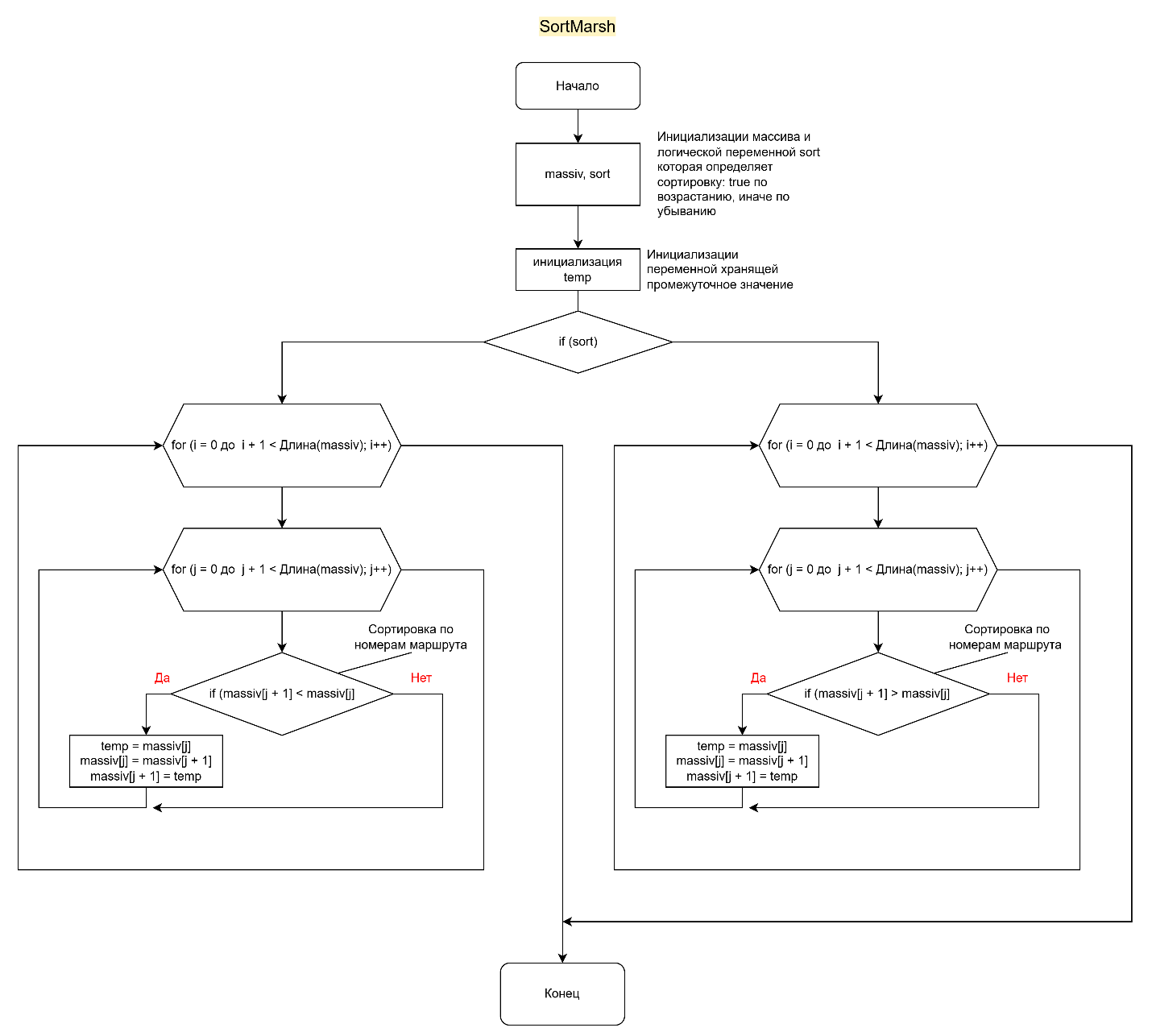


Рисунок 9 –Блок кода сортировки пузырьком массива с маршрутами по их номерам

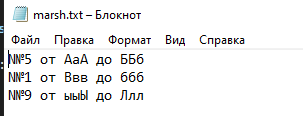


Рисунок 10 –Открытие процесса блокнот с записанными данными

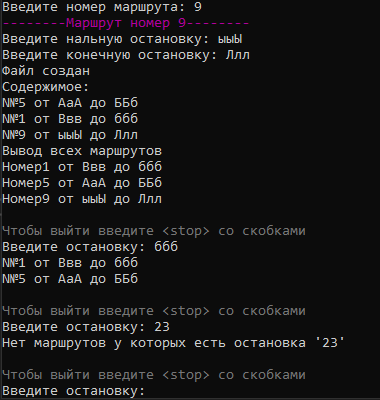


Рисунок 11 – Пример работы программы

Ниже представлен блок кода основного тела программы:

int count;

int LEN = 3; // Количество маршрутов

Marsh[] massiv = Marsh.Create(LEN);

string path = "marsh.txt"; // Имя и тип файлу куда писать маршруты и остановки

Marsh.CreateFileMarsh(path);

Marsh.AllMarsh(massiv, path);

// Читаем содержимое файла

Console.WriteLine("Содержимое:");

string text = File.ReadAllText(path);

Console.WriteLine(text);

// Открываем файл через блокнот

Process.Start("notepad.exe", path);

// Заполняем файл текстом из файла path

string[] strings = text.Split(new char[] { '\n' });

var pattern = @"^N№\s\*(?<route>\d+)\s+от\s+(?<from>.+?)\s+до\s+(?<to>.+)$";

var rx = new Regex(pattern, RegexOptions.IgnoreCase | RegexOptions.CultureInvariant);

count = 0;

foreach (string value in strings)

{

var match = rx.Match(value);

string from = match.Groups["from"].Value.Trim();

string to = match.Groups["to"].Value.Trim();

string route = match.Groups["route"].Value.Trim();

Marsh object\_marsh = new Marsh(from, to, int.Parse(route));

massiv[count] = object\_marsh;

count++;

}

// Сортировка массива

massiv = Marsh.SortMarsh(massiv, true);

Marsh.PrintAll(massiv);

// Поиск маршрутов по остановкам

while(true)

{

Console.ForegroundColor = ConsoleColor.DarkGray;

Console.WriteLine("\nЧтобы выйти введите <stop> со скобками");

Console.ResetColor();

Console.Write("Введите остановку: ");

string input = Console.ReadLine();

bool flag = input == "<stop>";

count = 0;

if (!flag)

{

for (int i = 0; i < LEN; i++)

{

bool startBus = massiv[i].startBusStop.ToLower().Equals(input.ToLower());

bool stopBus = massiv[i].endBusStop.ToLower().Equals(input.ToLower());

if (startBus || stopBus)

{

Console.WriteLine(massiv[i].Print());

count++;

continue;

}

}

if (count == 0)

{

Console.WriteLine($"Нет маршрутов у которых есть остановка '{input}'");

}

}

else

break;

}

# Вывод

В ходе выполнения лабораторной работы № 7 были изучены:

* Структурированный тип данных.
* Алгоритмы работы с файлами.

.ПРИЛОЖЕНИЕ А

using System;

using System.Diagnostics;

using System.IO;

using System.Text.RegularExpressions;

namespace SeventhLabs

{

internal class Program

{

// Обработка ввода, если пользователь ввел значение не целочисленного типа, то вывести информацию об этом

static int ReadInputInteger(int startCom, int endCom)

{

while (true)

{

Console.Write("Введите число из меню, для выбора задания: ");

if (int.TryParse(Console.ReadLine(), out int input))

{

if (input < startCom || input > endCom)

{

Console.ForegroundColor = ConsoleColor.Red;

Console.WriteLine("Введите от {0} до {1}", startCom, endCom);

Console.ResetColor();

}

else return input;

}

else

{

Console.ForegroundColor = ConsoleColor.Red;

Console.WriteLine("Ошибка, необходимо целое число");

Console.ResetColor();

continue;

}

}

}

static void Main(string[] args)

{

while (true)

{

int caseValue;

Console.WriteLine("1. Задание №1 - Структуры."

+ "\n2. Задание №2 - Файлы;"

+ "\n3. Очистить;"

+ "\n4. Выход из программы."

);

caseValue = ReadInputInteger(1, 4);

switch (caseValue)

{

case 1:

/\*

Дана структура, задающая дату вида:

struct date

{

int day;

int month;

int year;

};

Пользуясь таким структурным типом, составить программу, определяющую:

а) дату следующего (относительно сегодняшнего) дня,

\*/

// Ввод от пользователя

Console.Write("Введите день: ");

int.TryParse(Console.ReadLine(), out int d);

Console.Write("Введите месяц: ");

int.TryParse(Console.ReadLine(), out int m);

Console.Write("Введите год: ");

int.TryParse(Console.ReadLine(), out int y);

// Вызов структуры

Date get\_date = new Date(d, m, y);

get\_date.Print();

break;

case 2:

// Задание 2

// Дана структура с именем MARSH, содержащая следующие поля:

// Название начального пункта назначения.

// Название конечного пункта назначения.

// Номер маршрута.

// Написать программу, которая выполняет следующие действия:

// Ввод с клавиатуры данных в массив, состоящий из 8 элементов типа MARSH, и занесение их в файл данных.

// Чтение данных из файла и вывод их на экран.

// Вывод на экран информации о маршруте, номер которого введен с клавиатуры(если таких нет — вывести об этом сообщение).

// Список должен быть упорядочен по номерам маршрутов.

int count;

int LEN = 3; // Количество маршрутов

Marsh[] massiv = Marsh.Create(LEN);

string path = "marsh.txt"; // Имя и тип файлу куда писать маршруты и остановки

Marsh.CreateFileMarsh(path);

Marsh.AllMarsh(massiv, path);

// Читаем содержимое файла

Console.WriteLine("Содержимое:");

string text = File.ReadAllText(path);

Console.WriteLine(text);

// Открываем файл через блокнот

Process.Start("notepad.exe", path);

// Заполняем файл текстом из файла path

string[] strings = text.Split(new char[] { '\n' });

var pattern = @"^N№\s\*(?<route>\d+)\s+от\s+(?<from>.+?)\s+до\s+(?<to>.+)$";

var rx = new Regex(pattern, RegexOptions.IgnoreCase | RegexOptions.CultureInvariant);

count = 0;

foreach (string value in strings)

{

var match = rx.Match(value);

string from = match.Groups["from"].Value.Trim();

string to = match.Groups["to"].Value.Trim();

string route = match.Groups["route"].Value.Trim();

Marsh object\_marsh = new Marsh(from, to, int.Parse(route));

massiv[count] = object\_marsh;

count++;

}

// Сортировка массива

massiv = Marsh.SortMarsh(massiv, true);

Marsh.PrintAll(massiv);

// Поиск маршрутов по остановкам

while(true)

{

Console.ForegroundColor = ConsoleColor.DarkGray;

Console.WriteLine("\nЧтобы выйти введите <stop> со скобками");

Console.ResetColor();

Console.Write("Введите остановку: ");

string input = Console.ReadLine();

bool flag = input == "<stop>";

count = 0;

if (!flag)

{

for (int i = 0; i < LEN; i++)

{

bool startBus = massiv[i].startBusStop.ToLower().Equals(input.ToLower());

bool stopBus = massiv[i].endBusStop.ToLower().Equals(input.ToLower());

if (startBus || stopBus)

{

Console.WriteLine(massiv[i].Print());

count++;

continue;

}

}

if (count == 0)

{

Console.WriteLine($"Нет маршрутов у которых есть остановка '{input}'");

}

}

else

break;

}

break;

case 3:

Console.Clear();

break;

case 4:

return;

}

}

}

}

}

ПРИЛОЖЕНИЕ Б

using System;

using System.Collections.Generic;

using System.IO;

using System.Linq;

using System.Text;

using System.Threading;

using System.Threading.Tasks;

namespace SeventhLabs

{

struct Marsh

{

public string startBusStop;

public string endBusStop;

public int numberBus;

public Marsh(string startBusStop, string endBusStop, int numberBus)

{

this.startBusStop = startBusStop;

this.endBusStop = endBusStop;

this.numberBus = numberBus;

}

// Данный метод выводит: номер : начало : конец - заданного маршрута

public string Print()

{

string writeMarsh = "N№" + numberBus + " от " + startBusStop + " до " + endBusStop;

return writeMarsh;

}

// Данный метод создает массив 1D с типом Marsh, принимает на вход количество маршрутов

public static Marsh[] Create(int len)

{

int count = 0;

Marsh[] massiv = new Marsh[len];

while (count < len)

{

Console.Write("Введите номер маршрута: ");

int.TryParse(Console.ReadLine(), out int numberBus);

Console.ForegroundColor = ConsoleColor.Magenta;

Console.WriteLine($"--------Маршрут номер {numberBus}--------");

Console.ResetColor();

Console.Write("Введите нальную остановку: ");

string startBusStop = Console.ReadLine();

Console.Write("Введите конечную остановку: ");

string endBusStop = Console.ReadLine();

Marsh objectMarsh = new Marsh(startBusStop, endBusStop, numberBus);

massiv[count] = objectMarsh;

count++;

}

return massiv;

}

// Сортировка массива пузырьком

// sort = True по возрастанию номера, иначе по убыванию номера.

public static Marsh[] SortMarsh(Marsh[] massiv, bool sort)

{

Marsh temp;

if (sort)

{

for (int i = 0; i + 1 < massiv.Length; i++)

{

for (int j = 0; j + 1 < massiv.Length - i; j++)

{

if (massiv[j + 1].numberBus < massiv[j].numberBus)

{

temp = massiv[j];

massiv[j] = massiv[j + 1];

massiv[j + 1] = temp;

}

}

}

return massiv;

}

else

{

for (int i = 0; i + 1 < massiv.Length; i++)

{

for (int j = 0; j + 1 < massiv.Length - i; j++)

{

if (massiv[j + 1].numberBus < massiv[j].numberBus)

{

temp = massiv[j];

massiv[j] = massiv[j + 1];

massiv[j + 1] = temp;

}

}

}

return massiv;

}

}

// Запись введеных маршрутов в файл

public static void AllMarsh(Marsh[] massiv, string path)

{

bool flag = true;

foreach (Marsh value in massiv)

{

if (flag)

{

File.AppendAllText(path, $"{value.Print()}");

flag = false;

}

else

File.AppendAllText(path, $"\n{value.Print()}");

}

}

public static string CreateFileMarsh(string path)

{

//string path = "marsh.txt";

using (FileStream fs = File.Create(path))

{

Console.WriteLine("Файл создан");

}

return path;

}

// Вывод всех элементов

public static void PrintAll(Marsh[] massiv)

{

Console.WriteLine("Вывод всех маршрутов");

foreach (Marsh value in massiv)

{

Console.WriteLine("Номер" + value.numberBus + " от " + value.startBusStop + " до " + value.endBusStop);

}

}

}

}

ПРИЛОЖЕНИЕ В

using System;

using System.Collections.Generic;

namespace SeventhLabs

{

// Структура Date

struct Date

{

// Поля структуры

public int day; // 4 bytes

public int month; // 4 bytes

public int year; // 4 bytes

// Конструктор с параметрами

public Date(int day, int month, int year)

{

this.day = day;

this.month = month;

this.year = year;

}

// Метод по определению следующего дня

private void NextDay(int dayCount)

{

// Основное тело метода

if (1 <= day && day < dayCount)

{

day++;

}

else if (day == dayCount)

{

day = 1;

month++;

if (month > 12)

{

month = 1;

year++;

}

}

Console.WriteLine(day + "." + month + "." + year);

return;

}

public void Print()

{

var monthDict = new Dictionary<int, int>

{

{1, 31 },

{2, 28 },

{3, 31 },

{4, 30 },

{5, 31 },

{6, 30},

{7, 31},

{8, 31},

{9, 30},

{10, 31},

{11, 30},

{12, 31}

};

// Проверка корректности введеных данных пользователем

if (day > 31 || day < 1)

{

Console.ForegroundColor = ConsoleColor.DarkRed;

Console.WriteLine("Введите корректное числовое значение дня\n");

Console.ResetColor();

return;

}

if (month == 2)

{

// Проверка, если год високосный

if ((year % 4 == 0 && year % 100 != 0) || (year % 400 == 0))

{

NextDay(29);

}

else

{

NextDay(28);

}

}

else if (monthDict.ContainsKey(month))

{

NextDay(monthDict[month]);

}

else

{

Console.ForegroundColor = ConsoleColor.DarkRed;

Console.WriteLine("Введите корректное числовое значение месяца\n");

Console.ResetColor();

return;

}

}

}

}