Министерство науки и высшего образования Российской Федерации

Федеральное государственное автономное образовательное учреждение

высшего образования

«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт цифрового развития

Кафедра инфокоммуникаций

**ОТЧЕТ**

**ПО ПРАКТИЧЕСКОЙ РАБОТЕ №7**

**дисциплины «Алгоритмизация»**

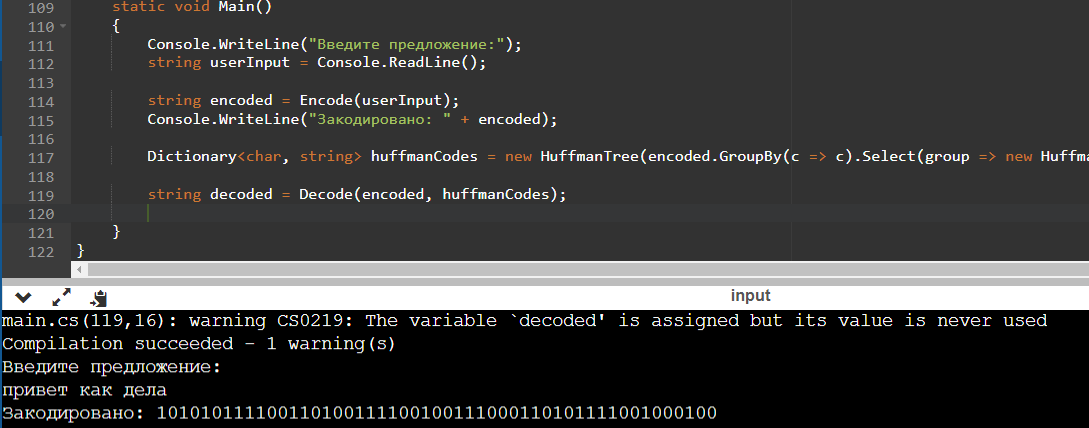
|  |  |  |  |
| --- | --- | --- | --- |
|  | | Выполнил:  Лейс Алексей Вячеславович  2 курс, группа ИВТ-б-о-22-1,  09.03.01 «Информатика и вычислительная техника», направленность (профиль) «Программное обеспечение средств вычислительной техники и автоматизированных систем»  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись) | |
|  | | Руководитель практики кандидат тех. наук доцент кафедры инфокоммуникаций: Воронкин Р.А  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись) | |
|  | |  | |

Отчет защищен с оценкой \_\_\_\_\_\_\_\_\_\_\_ Дата защиты\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Ставрополь, 2023 г.

**Порядок выполнения работы:**

Реализация алгоритма Хаффмана:



Предложение которое ввводит пользователь кодируется по спецальному алгоритму Хаффмана выдавая выходное значение 01.. строку в которой закодированы все буквы.

Код:

*using System;*

*using System.Collections.Generic;*

*using System.Linq;*

*class HuffmanNode*

*{*

*public char? Symbol { get; set; }*

*public int Frequency { get; set; }*

*public HuffmanNode Left { get; set; }*

*public HuffmanNode Right { get; set; }*

*}*

*class HuffmanTree*

*{*

*private readonly List<HuffmanNode> nodes;*

*public HuffmanTree(List<HuffmanNode> nodes)*

*{*

*this.nodes = nodes;*

*BuildTree();*

*}*

*private void BuildTree()*

*{*

*while (nodes.Count > 1)*

*{*

*List<HuffmanNode> orderedNodes = nodes.OrderBy(node => node.Frequency).ToList();*

*if (orderedNodes.Count >= 2)*

*{*

*List<HuffmanNode> taken = orderedNodes.Take(2).ToList();*

*HuffmanNode parentNode = new HuffmanNode*

*{*

*Symbol = null,*

*Frequency = taken[0].Frequency + taken[1].Frequency,*

*Left = taken[0],*

*Right = taken[1]*

*};*

*nodes.Remove(taken[0]);*

*nodes.Remove(taken[1]);*

*nodes.Add(parentNode);*

*}*

*}*

*}*

*public Dictionary<char, string> BuildHuffmanCodes()*

*{*

*if (nodes.Count == 0)*

*return null;*

*HuffmanNode root = nodes.Single();*

*Dictionary<char, string> huffmanCodes = new Dictionary<char, string>();*

*EncodeNode(root, "", huffmanCodes);*

*return huffmanCodes;*

*}*

*private void EncodeNode(HuffmanNode node, string code, Dictionary<char, string> huffmanCodes)*

*{*

*if (node.Symbol.HasValue)*

*huffmanCodes.Add(node.Symbol.Value, code);*

*else*

*{*

*if (node.Left != null)*

*EncodeNode(node.Left, code + "0", huffmanCodes);*

*if (node.Right != null)*

*EncodeNode(node.Right, code + "1", huffmanCodes);*

*}*

*}*

*}*

*class HuffmanCoding*

*{*

*public static string Encode(string input)*

*{*

*Dictionary<char, int> frequencies = input.GroupBy(c => c).ToDictionary(group => group.Key, group => group.Count());*

*List<HuffmanNode> nodes = frequencies.Select(pair => new HuffmanNode { Symbol = pair.Key, Frequency = pair.Value }).ToList();*

*HuffmanTree tree = new HuffmanTree(nodes);*

*Dictionary<char, string> huffmanCodes = tree.BuildHuffmanCodes();*

*char[] inputChars = input.ToCharArray();*

*string encoded = new string(inputChars.Select(c => huffmanCodes[c]).Aggregate((current, next) => current + next).ToArray());*

*return encoded;*

*}*

*public static string Decode(string encoded, Dictionary<char, string> huffmanCodes)*

*{*

*string currentCode = "";*

*string decoded = "";*

*foreach (char bit in encoded)*

*{*

*currentCode += bit;*

*if (huffmanCodes.ContainsValue(currentCode))*

*{*

*char symbol = huffmanCodes.First(pair => pair.Value == currentCode).Key;*

*decoded += symbol;*

*currentCode = "";*

*}*

*}*

*return decoded;*

*}*

*static void Main()*

*{*

*Console.WriteLine("Введите предложение:");*

*string userInput = Console.ReadLine();*

*string encoded = Encode(userInput);*

*Console.WriteLine("Закодировано: " + encoded);*

*Dictionary<char, string> huffmanCodes = new HuffmanTree(encoded.GroupBy(c => c).Select(group => new HuffmanNode { Symbol = group.Key, Frequency = group.Count() }).ToList()).BuildHuffmanCodes();*

*string decoded = Decode(encoded, huffmanCodes);*

*}*

*}*

Рисунок 1 – Запуск программы и главный алгоритм