Учреждение образования

«Белорусский государственный технологический университет»

**Защита информации и надежность информационных систем**

**Лабораторная работа №4**

ИЗБЫТОЧНОЕ КОДИРОВАНИЕ ДАННЫХ

В ИНФОРМАЦИОННЫХ СИСТЕМАХ.

КОД ХЕММИНГА

Выполнил:

Студент 3 курса 2 группы ФИТ

Максимова Вера Владимировна

Проверил:

Ржеутская Надежда Викентьевна

**2022 г.**

1. **Приложение**
2. using System;
3. namespace laba5
4. {
5. class Program
6. {
7. static void Main(string[] args)
8. {
9. string str;
10. do
11. {
12. Console.Clear();
13. Console.Write("Введите строку (не менее трех символов) = ");
14. str = Console.ReadLine();
15. }
16. while (str.Length < 3);
17. int k = str.Length;
18. int r = LenghtHemminga(k);
19. int n = k + r;
20. int[] mas = new int[str.Length + r];
21. int[,] checkMatrix = new int[n, r];
22. int error;
23. //Преобразование строки в массив
24. mas = StrInMas(str, k);
25. Console.Write("\nВходная строка = ");
26. OutMass(mas, k);
27. //Получение проверочной матрицы
28. checkMatrix = CheckMatrix(k);
29. Console.WriteLine("\nПроверочная матрица");
30. OutMass(checkMatrix, k);
31. //Добавление проверочных битов
32. Sindrom(checkMatrix, mas, k);
33. Console.WriteLine("\nПолная строка");
34. OutMass(mas, k);
35. try
36. {
37. Console.WriteLine("Введите место первой ошибки");
38. error = Convert.ToInt32(Console.ReadLine()) - 1;
39. if (mas[error] == 1) mas[error] = 0;
40. else mas[error] = 1;
41. }
42. catch { }
43. try
44. {
45. Console.WriteLine("Введите место второй ошибки");
46. error = Convert.ToInt32(Console.ReadLine()) - 1;
47. if (mas[error] == 1) mas[error] = 0;
48. else mas[error] = 1;
49. }
50. catch { }
51. Console.WriteLine("\nСтрока с ошибкой");
52. OutMass(mas, k);
53. mas = SearchError(mas, checkMatrix, k);
54. Console.WriteLine("\nСтрока без ошибки");
55. OutMass(mas, k);
56. }
57. //Считаем r (кол-во пров. симв.)
58. public static int LenghtHemminga(int k)
59. {
60. int r = (int)(Math.Log(k, 2) + 2.99f);
61. return r;
62. }
63. //Создание пров. матрицы
64. public static int[,] CheckMatrix(int k)
65. {
66. int r = LenghtHemminga(k);
67. int n = r + k;
68. int[,] mas = new int[n, r];
69. double rDouble = r - 1;
70. int rPow = (int)(Math.Pow(2, rDouble));
71. int[,] combinations = new int[rPow, r];
72. //все заполняем нулями
73. for (int i = 0; i < rPow; i++)
74. for (int j = 0; j < r; j++)
75. combinations[i, j] = 0;
76. //генератор бит.мн.
77. for (int segmentLenght = 0; segmentLenght < r - 2; segmentLenght++)
78. {
79. for (int i = 0; i < segmentLenght + 2; i++)
80. {
81. combinations[segmentLenght \* (r - 1), i] = 1;
82. }
83. for (int segmentPositin = 1; segmentPositin < r - 1; segmentPositin++)
84. {
85. for (int i = 0; i < r - 2; i++)
86. {
87. combinations[segmentLenght \* (r - 1) + segmentPositin, i + 1] = combinations[segmentLenght \* (r - 1) + segmentPositin - 1, i];
88. }
89. combinations[segmentLenght \* (r - 1) + segmentPositin, 0] = combinations[segmentLenght \* (r - 1) + segmentPositin - 1, r - 2];
90. }
91. //Заполнение посл. сроки 1-ми
92. if (segmentLenght == r - 2)
93. {
94. for (int i = 0; i < r; i++)
95. {
96. combinations[rPow - 1, i] = 1;
97. }
98. }
99. }
100. for (int i = 0; i < k; i++)
101. {
102. int amount = 0;
103. for (int j = 0; j < r - 1; j++)
104. {
105. if (combinations[i, j] == 1) amount++;
106. }
107. if (amount % 2 == 0)
108. combinations[i, r - 1] = 1;
109. }
110. for (int i = 0; i < k; i++)
111. for (int j = 0; j < r; j++)
112. mas[i, j] = combinations[i, j];
113. for (int i = 0; i < r; i++)
114. mas[i + k, i] = 1;
115. return mas;
116. }
117. //Поиск синдрома
118. public static int[] Sindrom(int[,] CheckMatrix, int[] mas, int k)
119. {
120. int r = LenghtHemminga(k);
121. int n = r + k;
122. int[] sindrom = new int[r];
123. for (int i = 0, l = 0; i < r; i++, l = 0)
124. {
125. for (int j = 0; j < k; j++)
126. {
127. if (CheckMatrix[j, i] == 1 && mas[j] == 1) l++;
128. else sindrom[i] = 0;
129. }
130. if (l % 2 == 1) sindrom[i] = 1;
131. else sindrom[i] = 0;
132. }
133. for (int i = 0; i < r; i++)
134. {
135. mas[i + k] = sindrom[i];
136. }
137. return mas;
138. }
139. //Нахождение ошибок
140. public static int[] SearchError(int[] mas, int[,] checkMatrix, int k)
141. {
142. int r = LenghtHemminga(k);
143. int n = r + k;
144. int[] beforeSindrom = new int[r];
145. //запоминаем проверочные биты
146. for (int i = k; i < n; i++)
147. {
148. beforeSindrom[i - k] = mas[i];
149. }
150. mas = Sindrom(checkMatrix, mas, k);
151. //Складываем синдром по модулю два
152. for (int i = k, j = 0; i < n; i++)
153. {
154. if (beforeSindrom[i - k].Equals(mas[i]))
155. {
156. mas[i] = 0;
157. j++;
158. //если сумма по модулю два все пров. бит равны нулю
159. if (j == r)
160. {
161. for (int l = k; l < n; l++)
162. {
163. mas[l] = beforeSindrom[l - k];
164. }
165. Console.WriteLine("Ошибок нет");
166. return mas;
167. }
168. }
169. else
170. {
171. mas[i] = 1;
172. }
173. }
174. for (int i = 0; i < n; i++)
175. {
176. int l = 0;
177. for (int j = 0; j < r; j++)
178. {
179. if (checkMatrix[i, j].Equals(mas[j + k])) l++;
180. }
181. if (l == r)
182. {
183. mas[i] = (mas[i] + 1) % 2;
184. }
185. if (l != r && i == n - 1)
186. {
187. Console.WriteLine("Ошибок кратно 2 и они исправлены неправильно");
188. }
189. }
190. mas = Sindrom(checkMatrix, mas, k);
191. return mas;
192. }
193. //Преобразование строки в массив
194. public static int[] StrInMas(string str, int k)
195. {
196. int r = LenghtHemminga(k);
197. int[] mas = new int[str.Length + r];
198. for (int i = 0; i < str.Length; i++)
199. {
200. if (str[i] == 49)
201. mas[i] = 1;
202. else mas[i] = 0;
203. }
204. return mas;
205. }
206. //вывод матрицы
207. public static void OutMass(int[,] mas, int k)
208. {
209. int r = LenghtHemminga(k);
210. int n = r + k;
211. for (int i = 0; i < r; i++)
212. {
213. for (int j = 0; j < n; j++)
214. {
215. Console.Write(mas[j, i]);
216. if (j + 1 == k) Console.Write("|");
217. }
218. Console.WriteLine();
219. }
220. }
221. //вывод одномерного массива
222. public static void OutMass(int[] mas, int k)
223. {
224. int n = LenghtHemminga(k) + k;
225. for (int i = 0; i < n; i++)
226. {
227. if (i == k) Console.Write("|");
228. Console.Write(mas[i]);
229. }
230. Console.WriteLine("\n");
231. }
232. }

}

1. **Ответы на контрольные вопросы**

1. В чем заключается цель и функциональная сущность пре-

образования информации на основе избыточного кодирования?

Избыточное кодирование данных в информационных системах. Код Хемминга 47

2. Пояснить зависимость r от длины информационного слова

k. Охарактеризовать относительную избыточность сообщения и

время его передачи по сети.

3. Записать проверочную матрицу кода Хемминга с dmin = 3 и

dmin = 4 для k = 4; 6; 8; 9; 10; 15; 16.

4. Записать проверочную матрицу кода простой четности для

k из вопроса 3. Пояснить на примере определение минимального

кодового расстояния Хемминга для данного кода.

5. Предположим, есть выбор (при построении матрицы кода)

между вектор-столбцами большего и меньшего веса. Какой вари-

ант Вы предпочтете и почему?