МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«БЕЛГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ им. В. Г. ШУХОВА» (БГТУ им. В.Г. Шухова)

Кафедра программного обеспечения вычислительной техники и автоматизированных систем

Лабораторная работа №3

по дисциплине: «Теория информации»

Выполнил: ст. группы ПВ-211

Чувилко Илья Романович

Проверил:

Твердохлеб В.В.

Белгород 2023 г.

Тема: «Исследование возможности применения методов энтропийного кодирования для обработки двоичных последовательностей»

Вариант 12

Содержание отчета

1. Аналитика касательно построения кодов для исходной двоичной последовательности

2. Примеры кодовой реализации п.3, п.3, п.4 и п.6.

3. Результаты обработки кодов, полученных в п.5.

4. Текстовая последовательность, восстановленная к читаемому виду.

5. Общие выводы

Ход работы:

Задание 1. Открыть файл Лабораторная работа 3 (задание).txt. Рассмотреть возможность построения кода по методам Хаффмана и Шеннона-Фано для бинарной последовательности. Сделать выводы.

Задание 2. Рассмотреть варианты обработки цепочек символов, а именно:

- 2 символа;

- 4 символа;

- 8 символов.

Для этого разработать консольное приложение, разбивающее сплошной массив символов на цепочки заданной длины.

Код модуля:

```
std::vector<simb> breakToParts(std::string &massage, int size) {
    std::vector<simb> res;
    for (int i = 0; i < massage.size(); i += size) {
        std::string latter;

        for (int j = 0; j < size; ++j)
            latter.push_back(massage[j + i]);
        res.push_back(simb(latter));
    }

    return res;
}
```

Задание 3. Рассматривая каждую цепочку (2, 4 и 8 символов длиной) как отдельный символ, построить коды по методу Хаффмана и Шеннона-Фано

Кодировка сообщений при разбиении на 2

```
Shenon
00 |  00 | 750
11 |  01 | 516
01 |  10 | 391
10 |  11 | 390
hoffman
00 |  11 | 750
11 |  10 | 516
01 |  01 | 391
10 |  00 | 390
```

Кодировка сообщения при разбиении на 4:

```
Shenon
1101 |     000 | 247
0000 |     001 | 236
1011 |      01 | 183
0001 |    1000 |  86
1000 |    1001 |  83
0010 |     101 |  70
1110 |   11000 |  28
0101 |   11001 |  21
1100 |    1101 |  21
0011 |  111000 |  14
1010 |  111001 |   9
1111 |   11101 |   9
0111 |  111100 |   5
0110 |  111101 |   5
0100 |  111110 |   4
1001 | 1111110 |   2
10 | 1111111 |   1
```

```
hoffman
1101 |      10 | 247
0000 |      01 | 236
1011 |     111 | 183
0001 |     000 |  86
1000 |    1101 |  83
0010 |    0011 |  70
1110 |   00101 |  28
0101 |   00100 |  21
1100 |  110011 |  21
0011 |  110000 |  14
1010 | 1100100 |   9
1111 | 1100011 |   9
0111 |11001011 |   5
0110 |11001010 |   5
0100 |11000101 |   4
1001 |110001001 |   2
10 |110001000 |   1
```

Кодировка сообщения при разбиении на 8:

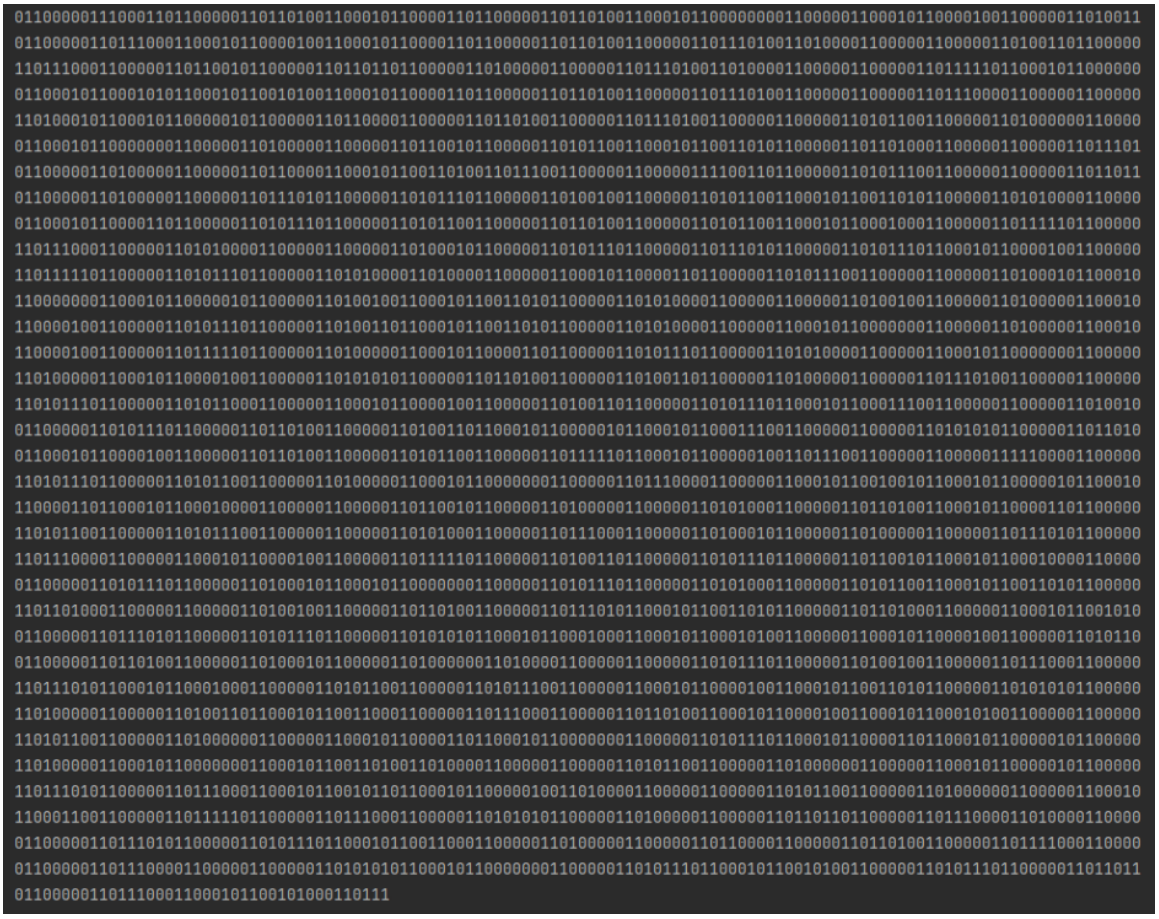
```
Shenon
11010000 |              00 | 163
11010001 |             010 |  68
00100000 |             011 |  41
10111110 |           10000 |  24
10110000 |           10001 |  21
10110101 |           10010 |  18
10111101 |           10011 |  15
10111011 |           10100 |  14
10111000 |           10101 |  13
10000001 |           10110 |  12
10000000 |           10111 |  11
10000010 |          110000 |  10
10001011 |          110001 |   9
10111100 |          110010 |   9
10110010 |          110011 |   8
10000011 |          110100 |   8
00101100 |          110101 |   7
10111010 |           11011 |   7
10110011 |         1110000 |   6
10110001 |         1110001 |   6
10111111 |          111001 |   6
10001100 |          111010 |   5
10110111 |          111011 |   4
10110110 |        11110000 |   4
10001111 |        11110001 |   3
10000101 |         1111001 |   3
10110100 |         1111010 |   3
00101110 |         1111011 |   2
10001110 |        11111000 |   2
10001000 |        11111001 |   2
10010010 |        11111010 |   1
10100010 |        11111011 |   1
10100100 |       111111000 |   1
10000111 |       111111001 |   1
10000110 |        11111101 |   1
10001101 |       111111100 |   1
10111001 |       111111101 |   1
101110 |        11111111 |   1
```

```
hoffman
11010000 |              11 | 163
11010001 |             100 |  68
00100000 |            1011 |  41
10111110 |            0011 |  24
10110000 |            0000 |  21
10110101 |           10100 |  18
10111101 |           01100 |  15
10111011 |           01011 |  14
10111000 |           01001 |  13
10000001 |           00101 |  12
10000000 |           00011 |  11
10000010 |           00010 |  10
10001011 |          101010 |   9
10111100 |          011111 |   9
10110010 |          011100 |   8
10000011 |          011011 |   8
00101100 |          011010 |   7
10111010 |          010101 |   7
10110011 |          010001 |   6
10110001 |          010000 |   6
10111111 |          001001 |   6
10001100 |         1010111 |   5
10110111 |         0111011 |   4
10110110 |         0111010 |   4
10001111 |         0101001 |   3
10000101 |         0101000 |   3
10110100 |         0010001 |   3
00101110 |        01111010 |   2
10001110 |        01111001 |   2
10001000 |        01111000 |   2
10010010 |        00100001 |   1
10100010 |        00100000 |   1
10100100 |       101011011 |   1
10000111 |       101011010 |   1
10000110 |       101011001 |   1
10001101 |       101011000 |   1
10111001 |       011110111 |   1
101110 |       011110110 |   1
```

Задание 4. Составить последовательности из полученных кодов символов для каждого случая

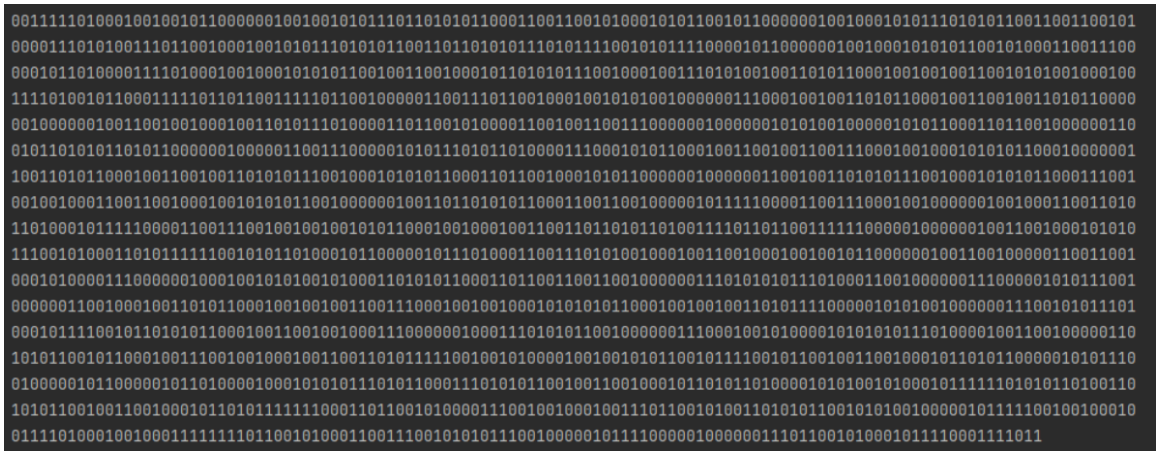
Метод Шеннона-Фано:

Разбиение на 2:



Разбиение на 4:

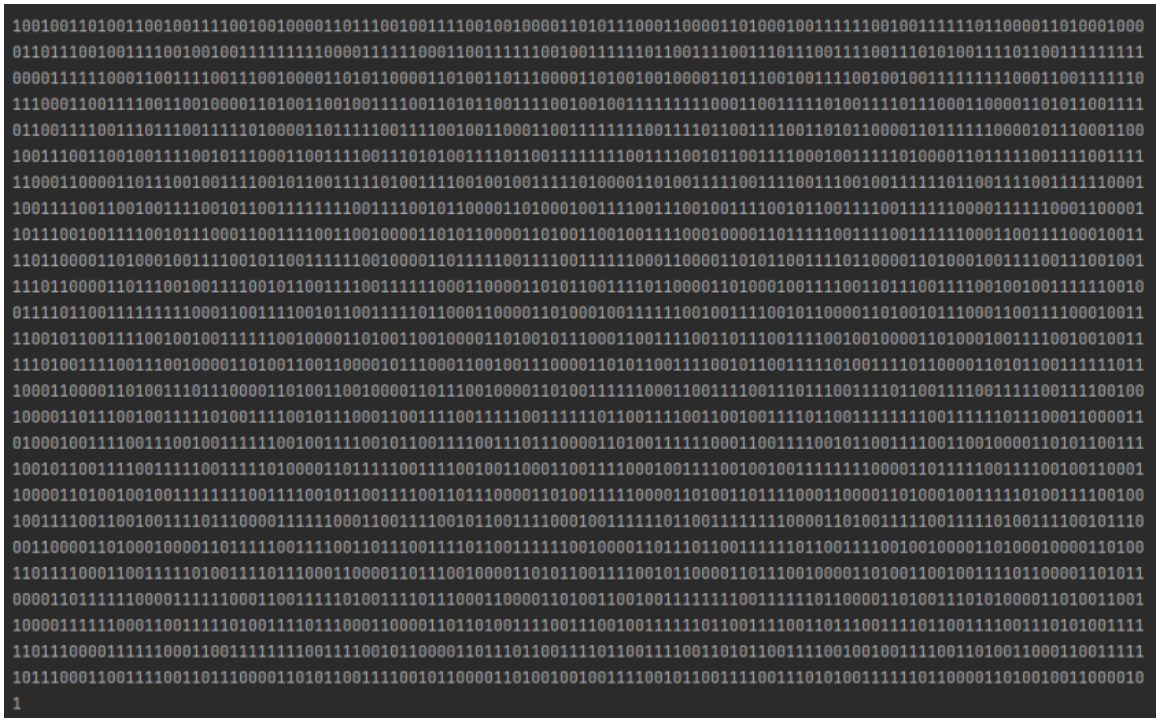
Разбиение на 8:



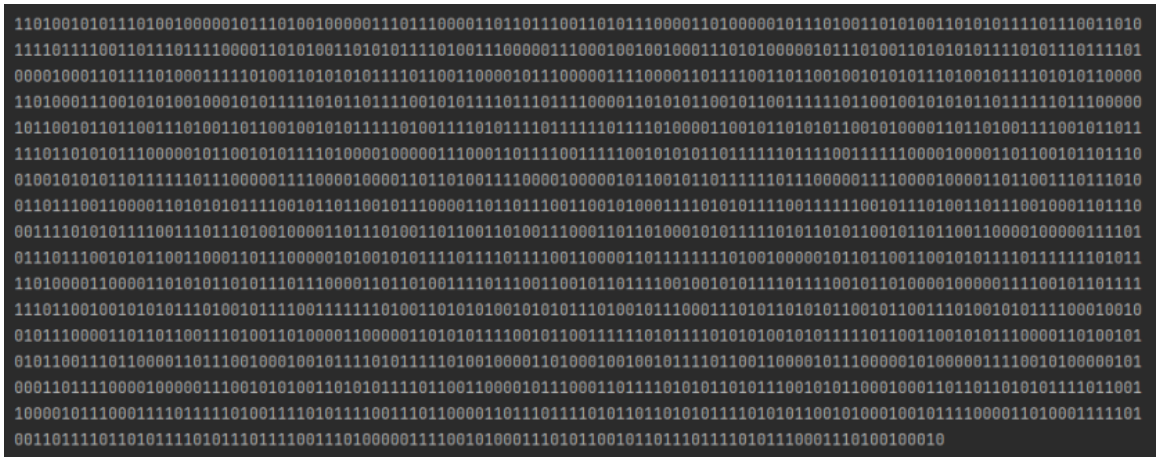
Метод Хаффмана:

Разбиение на 2:

Разбиение на 4:



Разбиение на 8:



Задание 5. По результатам работы в п.3 сделать выводы по поводу полученных результатов для каждого из методов (простота, скорость, полученные результаты (рассчитать коэффициенты сжатия)).

	N = 2	N = 4	N = 8
Хаффмана	1	1.32	2.04
Шеннона-Фано	1	1.23	2.03

Задание 6. Написать программу, восстанавливающую последовательности, полученные в п.3 в исходный вид согласно вариантам, приведенным в п.2.

Код программы:

```
std::vector<simb> createAlp() {
  std::cout << "input size of alp: ";
  int size;
  std::cin >> size;

  std::vector<simb> alp;

  for (int i = 0; i < size; ++i) {
    simb sb;
    std::string s;
    std::cin >> s;

    int j = 0;
    std::string liter;
    while (s[j] != '-') {
      liter.push_back(s[j]);
      j++;
    }
    sb.liter = liter;
    j++;
    liter = "";
    while (s[j] != '-') {
      liter.push_back(s[j]);
      j++;
    }
    sb.code = liter;
    j++;
    int n = 0;
    while (j < s.size()) {
      n += s[j] - '0';
      j++;
    }
    sb.ratio = n;

    alp.push_back(sb);
  }

  return alp;
}

void decodeMessage(std::string &massage, std::vector<simb> &alp) {
  int iStart = 0;
  int iEnd = 0;

  std::string liter;
  while (iStart < massage.size() - 1) {
    liter.push_back(massage[iEnd]);

    int i = -1;
    bool isFound = false;
    while (i != alp.size() - 1 && !isFound) {
      i++;
      if (liter == alp[i].code)
        isFound = true;
    }
```

```
    iEnd++;

    if (isFound) {
      std::cout << alp[i].liter;
      liter = "";
      iStart = iEnd;
    }
  }
}
```

Входные данные:

```
00111110100010010010110000001001001010111011010101100011001100101000101011001011010
input massage: input alph:input size of alp: 37
11010000-00-163
11010001-010-69
00100000-011-41
10111110-10000-24
10110000-10001-21
10110101-10010-18
10111101-10011-15
10111000-10100-14
10111011-10101-14
10000001-10110-12
10000000-10111-11
10000010-110000-10
10001011-110001-9
10111100-110010-9
10110010-110011-8
10000011-110100-8
00101100-110101-7
10111010-11011-7
10110011-1110000-6
10110001-1110001-6
10111111-111001-6
10001100-1110100-5
10110111-1110101-4
10110110-111011-4
10000101-1111000-4
10001111-1111001-3
10110100-1111010-3
00101110-1111011-3
10001110-11111000-2
10001000-11111001-2
10010010-11111010-1
10100010-11111011-1
10100100-111111000-1
10000111-111111001-1
10000110-11111101-1
```

Результат работы программы:

1101000010010010110100001011010111010001100000101101000010110101110100011
1000000000100000110100011000000111010000101100101101000010111000110100011
1000000111010001100000101101000010110101110100001011101100101100001000000
1101000010110010110100001011100011010000101101111101000010110110110100000
1011000011010000101110110010110000100000110100001011101011010001100000000
1101000110001111111010001100001011101000110000010110100001011010111010000101
0111011001000001101000010111000000100000110100001011001111010001100000111101
0100001011010011010000101101011101000010111011001000001101000010111101111
0100001011000000100000110100011000000011010000101100001101000010110111111
0100001011110111010001100010111101000010110101001000001101000010111011110
1000010110000110100001011010011010001100010110010111000100000110100001011
0001011010000101111100010000011010000101101101101000010110000110100001011
1101111010000101111101101000010110001110100001011110111010001100010111101
0000101111000010000011010001100000101101000010111110110100001011110111010
0001011010111010000101111011101000110001100110100001011101011010000101110
0011010000101111000010000011010000101100111101000010111110110100001011101
1110100001011111011010001100000011101000010111010110100001011111011010000
1011110000101100001000001101000110000010110100001011111000100000110100000
1011001111010001100000001101000110000011110100001011000111010001100010111
1010000101111000010000011010000101100011101000010110000110100011000000011
1010000101111101101000010110010110100011000101111010000101111000010000011
0100011000000011010000101100001101000110000001110100001011101011010000010
1100001101000110000010110100001011111011010000101111000010000011010001110
0000001101000010110000110100011000000111010000101111111101000010110101110
1000010110010110100001011000011010000101110110010000011010000101111101110
1000010111101001000001101000110000001110100001011001011010000101111101110
1000110001110001000001101000010110001110100001011111011010000101101011101
0000101100101101000110000011110100011000111000100000110100001011111111010
0001011010111010001100000011101000010110101110100001011110111010000101110
1011010001100000110010111000100000110100001010010011010000101111101101000
0010111101110100001011000011010001100000001101000010111000000100000110100
0110000111110100011000000111101000110000010110100011000110000100000110100
0010110111110100001011000011010000101111001101000010110101110100011000001
0110100001011110111010000101111100010000011010000101111001101000010111000
1101000010110011110100001011000011010000101110111101000010111000000100000
1010001100000011101000010111010110100001011001011010000101111101101000001
0110111110100011000110000100000110100001011111011010000101100111101000110
0000001101000010111110110100001011110011010000101111011101000110001011110
1000010110101001000001101000010110001110100001011010111010000101110111110
1000110001011110100001011010100100000110100011000010111010000101110111110
1000010111110110100001011111111010001100011001101000110001111001000001101
0001100000011101000010111101110100001011010111010000101100111101000010110
0000010110000100000110100001011111011010000101100011101000010111000110100
0001011101111010001100011001101000010111101110100001011111000100000110100
0110000001110100011000101111010000101111111101000010110000110100001011001
0110100011000100011010000101110001101000010110101110100011000000111010000
1100011110010000011010000101111011101000010110000000100000110100011000001
0110100011000000011010000101111101101000110000010110100011000000111101000
0101100001101000110000000110100011000101100101100001000001101000010111110
1110100001011000000100000110100011000001111010000101110111101000010111000
1101000110000110110100011000001100101100001000001101000010111101110100000

1011000000010000011010001100011011101000010111010110100001011100011010000
1011111111010000101100001101000010110110110100001011100000101100001000001
1010000101110111101000010111110110100011000100011010000101100001101000001
0110100110100001011010111010000101110010010000011010000101110000000100000
1101000010111111110100011000000011010000101111101101000110000101110100001
0111110110 10000101101101101000010111000110100011000010100101110

Результат совпадает с исходным сообщением.

Задание 7. Восстановить исходный текст из полученных последовательностей, пользуясь сервисом https://onlineutf8tools.com/convert-binary-to-utf8.

Исходный текст: Ветер свистел, визжал, кряхтел и гудел на разные лады. То жалобным тоненьким голоском, то грубым басовым раскатом распевал он свою боевую песенку. Фонари чуть заметно мигали сквозь огромные белые хлопья снега, обильно сыпавшиеся на тротуары, на улицу, на экипажи, лошадей и прохожих