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 №4

по дисциплине: Теория информации

тема: «Исследование кода Гилберта-Мура»

Выполнил: ст. группы ПВ-233
Ситников Алексей Павлович
Проверил: Твердохлеб Виталий
Викторович

Белгород 2025 г.

Задание 1–2.

Изучить принцип построения кода, используя пример в закрепленном файле «Пример. ЛБ4.pdf».

Построить обработчик, выполняющий компрессию по алгоритму Гилберта-Мура.

Код:

```c
#include <stdio.h>
#include <math.h>
#include <windows.h>

int main() {
    SetConsoleOutputCP(CP_UTF8);

    printf("Введите мощность алфавита\n");
    int n;
    scanf("%d", &n);
    char A[n];
    double p[n];
    double d[n];
    double q[n];
    int log[n];
    char code[n][30];
    getchar();
    for(int i = 0; i < n; i++){
        printf("Введите символ и вероятность\n");
        scanf("%c %lf", &A[i], &p[i]);
        getchar();
    }
    d[0] = 0;
    q[0] = p[0]/2;
    for(int i = 0; i < n; i++){
        if(i != 0){
            double sum = 0;
            for(int j = 0; j < i; j++){
                sum+=p[j];
            }
            d[i] = sum;
            q[i] = d[i] + p[i]/2;
        }
        log[i] = (int)(fabs(log2(p[i])) + 2);
        double num = q[i];

        char temp[log[i]+1];
        char *begin = temp;
        int maxIterations = log[i];
        while (num > 0 && maxIterations > 0) {
            num *= 2;
            if (num >= 1) {
                *begin = '1';
                num -= 1;
            } else {
                *begin = '0';
            }
            maxIterations--;
            begin++;
        }
        *begin = '\0';
        char *b = code[i];
```

```
        begin = temp;
        while (*begin!='\0'){
            *b = *begin;
            b++;
            begin++;
        }
        while (num <= 0 && maxIterations > 0){
            *b = '0';
            b++;
            maxIterations--;
        }
        *b = '\0';
    }

    printf("Введите строку для кодирования\n");
    char buf[10000];
    scanf("%s", buf);
    char *b = buf;
    char ans[1000000];
    char *begin = ans;
    while (*b!='\0'){
        for(int i = 0; i < n; i++){
            if(*b == A[i]){
                char *b_code = code[i];
                while (*b_code!='\0'){
                    *begin = *b_code;
                    b_code++;
                    begin++;
                }
            }
        }
        b++;
    }
    *begin = '\0';

    for(int i = 0; i < n; i++){
        printf("%c %s\n", A[i], code[i]);
    }

    printf("%s", ans);
    return 0;
}
```

Пример программы:

```
C:\Users\admin\CLionProjects\Project17\cmake-build-debug\Project17.exe
Введите мощность алфавита
6
Введите символ и вероятность
b 0.1
Введите символ и вероятность
e 0.4
Введите символ и вероятность
z 0.2
Введите символ и вероятность
m 0.1
Введите символ и вероятность
l 0.1
Введите символ и вероятность
d 0.1
Введите строку для кодирования
mmzebbdlm
Получили
110001100010010100000100001111101101111000
Process finished with exit code 0
```

Задание 3.

Создать генераторы данных, работающих как источники Хартли и Бернулли (в двоичном алфавите).

Хартли:

```c
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include <windows.h>

int main() {
    SetConsoleOutputCP(CP_UTF8);
    int num_bits;
    printf("Введите количество символов\n");
    scanf("%d", &num_bits);
    char ans[num_bits + 1];
    char *begin = ans;
    srand(time(NULL));
    for (int i = 0; i < num_bits; i++) {
        *begin = rand() % 2 == 1 ? '1' : '0';
        begin++;
    }
    *begin = '\0';
    printf("%s", ans);
    return 0;
}
```

Пример вывода:

```
C:\Users\admin\CLionProjects\Project17\cmake-build-debug\Project17.exe
Введите количество символов
10
1111100010
Process finished with exit code 0
```

Бернулли:

```c
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include <windows.h>


int main() {
    SetConsoleOutputCP(CP_UTF8);
    int num_bits;
    double p;
    printf("Введите количество символов и вероятность символа \"1\"\n");
    scanf("%d %lf", &num_bits, &p);
    srand(time(NULL));

    char ans[num_bits+1];
    char *begin = ans;

    for (int i = 0; i < num_bits; i++) {
        double r = (double)rand() / RAND_MAX; // Случайное число от 0 до 1
        *begin = (r < p) ? '1' : '0'; // Генерация бита с вероятностью p
        begin++;
    }
    *begin = '\0';
    printf("%s", ans);
    return 0;
}
```

Пример вывода:

```
C:\Users\admin\CLionProjects\Project17\cmake-build-debug\Project17.exe
Введите количество символов и вероятность символа "1"
15 0.2
000000000010100
Process finished with exit code 0
```

Задание 4.

Сгенерировать 2 цепочки данных длиной 800 символов каждую (соответственно, порожденные по Бернуллевскому принципу и принципу Хартли).

Хартли:

1001001111101010000100001100111001111010101100010101101011001111011
10000000101100001110101011001010010110010100000000000011110010010100
110001010000011100110100100011000011100111110110101111001101110110100
1001010000110101011011010100010100001010100100000000110111000100001
01100100010011101100001011111000110000011110110100011011100001101111
110000100000001011100101111001100001001111110100000000011110101000110
11001011010001011111101011101000110100100110011010100000111111100111
01011111100100000011011011001110000100000100011100101011001000111111
1010101000110011101101111100000010100010111011101110010110000010100
11100010101101101100111101100100001001010111010100000101110110011101
1011011100010110101001010100100110000000011000111111010101101111000
1101110100101011100000101010111001101010100001101000100

Бернулли, вероятность «1» 0,7:

10111101001010001111010101110011010101100111111001111100100001000110
11011101110001010111101011011111111011110110101100101010010111010
1111110101111011010100111111111111111111101001011111111011111110010
0001110011111011001011011011100101011111111111101111111001111011011110
110111011110111010110001111011111111100110011111111111110010000111011
110110111111101110111111111111111110101111110111010101011011101101001
111111100001110111010110011001101111111111011111000100100111010111111
1010111110011111110111101111110101110011111111110011101110001111111
000011101101011101111000111110111111001110111001110011110111101011110
01001101010101111111000011101100111011101111011011010001110110110011
100111011100111000111100110111101101111011111011111010100111110101110
0111111111011110111101111001111111111101100

Задание 5:

Выбрать произвольную текстовую строку на русском языке длиной 100 символов, преобразовать в двоичный формат Unicode/ASCII (на выбор), пользуясь любым соответствующим онлайн-сервисом (на выходе имеем последовательность из 800 символов).

Солнце светит ярко и ветер шелестит листвой. Вдали слышен смех детей. Жизнь прекрасна и удивительна!

Представление в двоичном коде в соответствие с ASCII:
11010000101000011101000010111110110100001011101111010000101111011101
0001100001101101000010110101001000001101000110000001110100001011001
0110100001011010111010001100000101101000010111000110100011000001000

1000001101000110001111110100011000000011010000101110101101000010111
1100010000011010000101110000010000011010000101100101101000010110101
1101000110000010110100001011010111010001100000000010000011010001100
0100011010000101101011101000010111011110100001011010111010001100000
0111010001100000101101000010111000110100011000001000100000110100001
0111011110100001011100011010001100000011101000110000010110100001011
0010110100001011111011010000101110010010111000100000110100001001001
0110100001011010011010000101100001101000010111011110100001011100000
1000001101000110000001110100001011101111010001100010111101000110001
0001101000010110101110100001011110100100000110100011000000111010000
1011110011010000101101011101000110000101001000001101000010110100110
1000010110101110100011000001011010000101101011101000010111001001011
1000100000110100001001011011010000101110001101000010110111110100001
0111101110100011000110000100000110100001011111111010001100000001101
0001011010111010000101110101101000110000000110100001011000011010001
1000000111010000101111011101000010110000001000001101000010111000001
0000011010001100000111101000010110100110100001011100011010000101100
1011010000101110001101000110000010110100001011010111010000101110111
1010001100011001101000010111101110100001011000000100001000000000

```
    SetConsoleOutputCP(CP_UTF8);
    char str[] =
"1101000010100001110100001011111011010000101110111101000010111101110100011000
0110110100001011010100100000110100011000000111010000101100101101000010110101
1010001100000101101000010111000110100011000001000100000110100011000111111010
0110000000110100001011101011010000101111100010000011010000101110000010000011
0100001011001011010000101101011101000110000010110100001011010111010001100000
0010000011010001100010001101000010110101110100001011101111010000101101011010
0011000000111010001100000101101000010111000110100011000001000100000110100001
1110111101000010111000110100011000000111010001100000101101000010110010110100
0101111101101000010111001001011100010000011010000100100101101000010110100110
0000101100001101000010111011110100001011100000100000110100011000000111010000
0111011110100011000101111010001100010001101000010110101110100001011110100100
0011010001100000011101000010111100110100001011010111010001100001010010000011
0100001011010011010000101101011101000110000010110100001011010111010000101110
0010111000100000110100001001011011010000101110001101000010110111110100001011
1011101000110001100001000001101000010111111110100011000000011010000101110101
0100001011101011010001100000001101000010110000110100011000000111010000101111
1110100001011000000100000110100001011100000100000110100011000001111010000101
0100110100001011100011010000101100101101000010111000110100011000001011010000
0110101110100001011101111010001100011001101000010111101110100001011000000100
0100000000";
    char *begin = str;
    char temp_str[9];
    char *temp_pointer;
    node data[800];
    int count = 0;
    int count_el = 0;
    while (*begin!='\0'){
        temp_pointer = temp_str;
        for(int i = 0; i < 8; i++){
            *temp_pointer = *begin;
            temp_pointer++;
            begin++;
        }
        *temp_pointer = '\0';
        int flag_ = 0;
        for(int i = 0; i < count; i++){
            if(equalStr(data[i].letters, temp_str)){
                data[i].p++;
                count_el++;
                flag_ = 1;
            }
        }
        if(flag_==0){
            temp_pointer = temp_str;
            char *b = data[count].letters;
            data[count].p++;
            while (*temp_pointer!='\0'){
                *b = *temp_pointer;
                b++;
                temp_pointer++;
            }
            count_el++;
            count++;
        }
    }

    for(int i = 0; i < count; i++){
        data[i].p /= count_el;
    }
    data[0].d = 0;
    data[0].q = data[0].p / 2;
    data[0].size = (int)(fabs(log2(data[0].p)) + 2);
```

```
for(int i = 0; i < count; i++){\
    if(i!=0) {
        data[i].d = data[i - 1].d + data[i - 1].p;
        data[i].q = (data[i].p / 2) + data[i].d;
        data[i].size = (int) (fabs(log2(data[i].p)) + 2);
    }
    double num = data[i].q;
    begin = data[i].code;
    int maxIterations = data[i].size;
    while (num > 0 && maxIterations > 0) {
        num *= 2;
        if (num >= 1) {
            *begin = '1';
            num -= 1;
        } else {
            *begin = '0';
        }
        maxIterations--;
        begin++;
    }
    while (num <= 0 && maxIterations > 0){
        *begin = '0';
        begin++;
        maxIterations--;
    }
    *begin = '\0';
}
for(int i = 0; i < count; i++){
    printf("%s %s %d\n", data[i].letters, data[i].code, data[i].size);
}
begin = str;
int size_ans = 0;
char ans[count_el*8 + 1];
char *ansB = ans;
while (*begin!='\0'){
    char subStr[9];
    for(int i = 0; i < 8; i++){
        subStr[i] = *begin;
        begin++;
    }
    subStr[8] = '\0';
    int flag = 0;
    for(int i = 0; i < count; i++){
        if(equalStr(data[i].letters, subStr)){
            char *b = data[i].code;
            while (*b!='\0'){
                *ansB = *b;
                ansB++;
                b++;
                size_ans++;
            }
            flag = 1;
        }
    }
    if(flag == 0) {
        printf("Ошибка, символ не найден");
    }
}
*ansB = '\0';
printf("Полученная последовательность:\n%s\n", ans);
printf("Размер: %d\n", size_ans);
printf("Коэффициент сжатия: %f\n", 8.*count_el/size_ans);
```

```
    double sr = 0;
    for(int i = 0; i < count; i++){
        sr+=data[i].p * data[i].size;
    }
    double D = 0;
    for(int i = 0; i < count; i++){
        D+= data[i].p * (data[i].size - sr) * (data[i].size - sr);
    }
    printf("Дисперсия: %f", D);
    return 0;
}
```

Вывод программы:

```
C:\Users\admin\CLionProjects\Project17\cmake-build-debug\Project17.exe
11010000 001 3
10100001 010011101 9
10111110 0101000 7
10111011 010101 6
10111101 0101111 7
11010001 0111 4
10000110 100010010 9
10110101 10010 5
00100000 10100 5
10000001 101100 6
10110010 1011100 7
10000010 110000 6
10111000 110011 6
10001111 110101000 9
10000000 1101011 7
10111010 11011011 8
10001000 11011110 8
10111001 11100001 8
00101110 11100100 8
10010010 111001100 9
10110100 1110100 7
10110000 1110110 7
10001011 111100010 9
10111100 111100101 9
10000101 111101000 9
10010110 111101011 9
10110111 111101101 9
10001100 11111001 8
10111111 111110110 9
10000011 111111001 9
00100001 111111011 9
00000000 111111110 9
Полученная последовательность:
00101001110100101010000010101010010101111011110001001000110010101000111101100001101110000110010011111000000111001101111
00001010001111101010000111110101100111011011001010100010100001110011101000011011100001100100111110000001100100111110101
10100011111011110001100100010101010011001001111011000111110000001110011011111000010100001010101001110011011110110001111
00000011011100000101010000011110000111100100101000011110011000011110100001111011000101010100111001110100011110110000101
01011111110001001111101111000110010001010111101000111101100001111100101001100100111111101000101000011110100001100100111
11000000110010001111000011110010010100001111101011001110011000111110110100101011110111111111001101000011111101100111101
10011001000111011011011111010110011111011001111011000010101111001111011010100001110011101000111111111001001111010000110
11001101110000111001101111100000011001000101010101111111100100101011110011110110111111011111111110
Размер: 938
Коэффициент сжатия: 1.569296
Дисперсия: 3.631734
Process finished with exit code 0
```

Последовательность символов:

```
Полученная последовательность:
000000010000011000100001000001001110011101010111001000000011110001101001000101011011000110111011000000000011010110100010
110000000111100010011110111001011100010001110001000011101110010001101001000101010001001010001110010001100000000001111001
0111001110101110100111101100111000000100101000101011100000100111001111100010001110010000010101110011101010001111110110001
011101100001111000100111110010111001110101111011111010011110001001000111101001011111100110111000111110000010111011100001
11000010001110000101101000101111111001101100101001000000101001000100111000100111101000100011100011111110
Размер: 583
Коэффициент сжатия: 1.372213
Дисперсия: 1.461100
Process finished with exit code 0
```

Входные данные:
1101000010100001110100001011111011010000101110111101000010111101110
1000110000110110100001011010100100000110100011000000111010000101
100101101000010110101110100011000001011010000101110001101000110000
01000100000110100011000111111010001100000001101000010111010110100
00101111100010000011010000101110000010000011010000101100101101000
010110101110100011000001011010000101101011101000110000000000100000
110100011000100011010000101101011101000010111011110100001011010111
01000110000000111010001100000101101000010111000110100011000001000100
0000011010000101110111101000010111000110100011000000111010001100000
0101101000010110010110100001011111011010000101110010010111000100000
0110100001001001011010000101101001101000010110000110100001011101 1
110100001011100000100000110100011000000111010000101110111101000110
001011110100011000100011010000101101011101000010111101001000001101
000110000000111010000101111001101000010110101110100011000010100100 0
0011010000101101001101000010110101110100011000001011010000101101 0
111010000101110010010111000100000110100001001011011010000101110001
101000010110111110100001011110111010001100011000010000011010000101
111111101000110000000110100001011010111010000101110101101000110000
0001101000010110000110100011000000111010000101111011101000010110000
00010000011010000101110000010000011010001100000111101000010110100
110100001011100011010000101100101101000010111000110100011000001011
010000101101011101000010111011110100011000110011010000101111011101
00001011000000010000100000000

Вывод:
Полученная последовательность:
00101001110100101010000010101010010101111011110001001000110010101
000111101100001101110000110010011111000000111001101 1111
000010100011111010100001111101011001110110110010101000101000011100
1110100001101110000110010011111000000110010011111 01011
10100011111011110001100100010101010011001001111011000111110000001 1
100110111110000101000010101010011100110111101100011111

000000110111000010101000001111000011110010010100001111001100001111 0100001111011000101010100111001110100011110110000101010 0101111111000100111110111100011001000101011111010001111011000011111 001010011001001111111010001010000111101000011001001111 1100000011001000111100000111100100101000011111010110011100110011111 011010010101111011111111001101000011111101100111110101 1001100100011101101101111101011001111011001111011000010101111001111 101101010000111001110100011111111001001111010000111000 1100110111000011100110111110000000110010001010101011111111001001010 11110011110110111111011111111110 Размер: 938 Коэффициент сжатия: 1.569296 Дисперсия: 3.631734

Входные данные: Солнце светит ярко и ветер шелестит листвой. Вдали слышен смех детей. Жизнь прекрасна и удивительна!
Вывод: Полученная последовательность: 000000010000011000100001000001001110011101010111001000000001111000 110100100010101101100011011101100000000000110101101000100 1100000001111000100111101110010111000100011100010000111011100100001 101001000101010001001010001110010001100000000000111100 1 01110011101011101001111011001110000001001010001010111000000100111 00 1111100010001110010000101011100111010100011111101100 01 0111011000011110001001111100101110011101011110111110100111100010 01 00011110100101111110011011100011111000000101110111000 01 1100001000111000010110100010111111100110110010100100000101001000 10 01110001001111010001000111000111111 0 Размер: 583 Коэффициент сжатия: 1.372213 Дисперсия: 1.461100

Вывод: по итогу лабораторной работы получена программа способная кодировать методом Гилберта-Мура.