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 №2

по дисциплине: Теория информации

тема: «Исследование кодов Шеннона-Фано»

Выполнил: ст. группы ПВ-233

Ситников Алексей Павлович

Проверил: Твердохлеб Виталий

Викторович

Белгород 2025 г.

Цель работы: изучить способ кодирования Шеннона-Фано.

Задание 1.

Построить код для сообщения, содержащего строку панграммы **«в чащах юга жил бы цитрус? Да, но фальшивый экземпляр!».** Для полученного кода рассчитать показатели коэффициента сжатия и дисперсии.

Кодирование 1 буквы char 8 бит, значит размер 8*54 = 432. Рассчитаем количество появлений каждого символа:

```
в 2,  9, ч 1, а 5, щ 1, х 1, ю 1, г 1, ж 1, и 3, л 3, б 1, ы 2, ц 1, т 1, р 2, у 1, с 1, ? 1, Д 1, , 1, н 1, о 1, ф 1,
ь 1, ш 1, й 1, э 1, к 1, з 1, е 1, м 1, п 1, я 1, ! 1,
```

Теперь с помощью Шеннона-Фано получим кодовое предстовления каждого символа:

```
C:\Users\admin\CLionProjects\Project17\cmake-build-debug\Project17.exe
в чащах юга жил бы цитрус? Да, но фальшивый экземпляр!
 : 000
а: 1000
и: 11000
л: 1110
в: 01000
ы: 01100
р: 10100
ч: 010100
щ: 110100
х: 011100
ю: 101100
г: 01011
ж: 111100
б: 01111
ц: 10111
т: 010010
у: 11011
с: 001000
?: 101010
Д: 01101
,: 11111
н: 00110
о: 10010
ф: 010101
ь: 110101
ш: 00101
й: 101101
э: 011101
к: 111101
з: 00111
е: 10011
м: 010011
п: 11001
я: 001001
!: 101011

Process finished with exit code 0
```

Теперь закодируем предложение:

```
0100000001010010001101001000011100000101100010111000000111100110001110000011110110000010111110000100101010011011001000010
1010000001101100011111000000110100100000010101100011101101010010111000010000110010110100001110111110100111100110100111100 11
1100010011010010101 1
Process finished with exit code 0
```

Посчитаем коэффициент сжатия:
изначально насчитали 432 бит, количество бит после сжатия: 260, в таком случае коэффициент сжатия равен $\frac{432}{260}$ = 1,66.

Рассчитаем среднюю длину символа и дисперсию, получим:

```
sr = 4.814815, D = 1.076818
```

Задание 2.

Построить код для сообщения, содержащего строку **«Victoria nulla est, Quam quae confessos animo quoque subjugat hostes»**

Для полученного кода рассчитать показатели коэффициента сжатия и дисперсии.

Закодированное предложение:

```
01010011100101100100100001110111001010000111111000000110001101010000001101100001010010000001011100010101110100001111000010100
11000010110100111111010011011001100010011000000101011111110011101010000000111100001000111100001100001100100000011110011100
00101011010001000001101101001100001001101100
```

Кодирование 1 буквы char 8 бит, значит размер 8*68 = 544. Размер сжатого равен 283, тогда коэффициент сжатия равен $\frac{544}{283}$ = 1,92.

Рассчитаем среднюю длину символа и дисперсию, получим:

```
sr = 4.161765, D = 0.517950
```

Задание 3.

Построить консольное приложение, реализующее процесс кодирования по методу Шеннона-Фано (с возможностью расчета коэффициента сжатия и дисперсии).

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

typedef struct node{
    int count;
    wchar_t  letter;
    char code[30];
}node;

void sort(int n, node *array[n]){
```

```
    for(int i = 0; i < n; i++){
        int is_sort = 1;
        for(int j = n - 1; j > i; j--){
            if(array[j]->count > array[j-1]->count){
                node *temp = array[j];
                array[j] = array[j-1];
                array[j-1] = temp;
                is_sort = 0;
            }
        }
        if(is_sort){
            break;
        }
    }
}

void putChar(char *arr, char c){
    char *begin = arr;
    while (*begin!='\0'){
        begin++;
    }
    *begin = c;
    *(++begin) = '\0';
}

void fano(int size, node *array[size]){
    int sum1 = 0, sum2 = 0;
    node *part1[size];
    node *part2[size];
    int size1 = 0;
    int size2 = 0;
    for(int i = 0; i < size; i++){
        if(sum1 < sum2){
            part1[size1++] = array[i];
            sum1+=array[i]->count;
            putChar(array[i]->code, '1');
        }
        else{
            part2[size2++] = array[i];
            sum2+=array[i]->count;
            putChar(array[i]->code, '0');
        }
    }
    if(size1 > 1){
        fano(size1, part1);
    }
    if(size2 > 1){
        fano(size2, part2);
    }
}


int main() {

    SetConsoleOutputCP(866);
    SetConsoleCP(866);
    wchar_t  str[1000000];
    fgetws(str, sizeof(str) / sizeof(str[0]), stdin);

    wchar_t  *begin = str;
    int size = 0;
    while (*begin!='\n'){
        size++;
        begin++;
```

```
    }
    *begin = '\0';
    begin = str;
    int count = 0;
    node *data[size];
    while (*begin!='\0'){
        int flag = 0;
        for(int i = 0; i < count; i++){
            if(data[i]->letter == *begin){
                flag = 1;
                data[i]->count++;
                break;
            }
        }
        if(flag == 0){
            data[count] = (node *)malloc(sizeof(node));
            data[count]->letter = *begin;
            data[count]->count = 1;
            data[count]->code[0] = '\0';
            count++;
        }
        begin++;
    }
    sort(count, data);
    fano(count, data);

    char answer[size * 8];
    begin = str;
    char *ansBg = answer;
    while (*begin!='\0'){
        for(int i = 0; i < count; i++){
            if(data[i]->letter == *begin){
                char *bg = data[i]->code;
                while (*bg!='\0'){
                    *ansBg=*bg;
                    bg++;
                    ansBg++;
                }
                break;
            }
        }
        begin++;
    }
    *ansBg = '\0';
    printf("\n%s", answer);
    int s = 0;
    ansBg = answer;
    while (*ansBg!='\0'){
        ansBg++;
        s++;
    }
    printf("\nsize: %d\n", s);

    double K = 8*size/(double)s;
    printf("K = %f\n", K);

    double sr = 0;
    for(int i = 0; i < count; i++){
        int len = 0;
        char *bg = data[i]->code;
        while (*bg!='\0'){
            bg++;
            len++;
        }
```

```
        sr+= ((double)data[i]->count/size)*len;
    }
    double d = 0;
    for(int i = 0; i < count; i++){
        int len = 0;
        char *bg = data[i]->code;
        while (*bg!='\0'){
            bg++;
            len++;
        }
        d+= ((double)data[i]->count/size) * (len - sr) * (len - sr);
    }
    printf("D = %f\n", d);
    for(int i = 0; i < count; i++){
        free(data[i]);
    }

    return 0;
}
```

Примеры:

```
C:\Users\admin\CLionProjects\Project17\cmake-build-debug\Project17.exe
в чащах юга жил бы цитрус? Да, но фальшивый экземпляр!

0100000001010010001101001000011100000101100010111000000111100110001110000011110110000010111110000100101010011011001000101010000111000111111000001101001000001010110001110110101001011100001000011001011010000111011111010011110011010011100111100100110100
11000100110100
size: 260
K = 1.661538
D = 1.076818

Process finished with exit code 0
```

```
C:\Users\admin\CLionProjects\Project17\cmake-build-debug\Project17.exe
Victoria nulla est, Quam quae confessos animo quoque subjugat hostes

0101001110010110010010000111011100101000011111000001100011010100000110110000101001000001011100010101110100001111000101001100001011010011111101001101100010011000001010111111100110101000000111100001000111100001100001100100000111110011000010101101000100000110110100110000100110100
size: 283
K = 1.922261
D = 0.517950

Process finished with exit code 0
```

Задание 4.

Программа для кодирования методом Хаффмана.

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

typedef struct node{
    int count;
    wchar_t  letter;
    char code[300];
    struct node* left;
    struct node* right;
```

```
}node;

void sort(int n, node **array){
    for(int i = 0; i < n; i++){
        int is_sort = 1;
        for(int j = n - 1; j > i; j--){
            if(array[j]->count > array[j-1]->count){
                node *temp = array[j];
                array[j] = array[j-1];
                array[j-1] = temp;
                is_sort = 0;
            }
        }
        if(is_sort){
            break;
        }
    }
}

void haffman(node *let, char* code, char *endCode){
    if(let->letter == '\0'){
        *endCode = '0';
        *(endCode+1) = '\0';
        haffman(let->left, code, endCode+1);
        *endCode = '1';
        *(endCode+1) = '\0';
        haffman(let->right, code, endCode+1);
    }
    else{
        char *begin = let->code;
        char *begin_code = code;
        while (*begin_code!='\0'){
            *begin = *begin_code;
            begin++;
            begin_code++;
        }
        *begin = '\0';
    }
}

void huffmanTreeToArray(node **data, int* count, node *root){
    if(root->left==NULL){
        data[*count] = root;
        (*count)+=1;
    }
    else{
        huffmanTreeToArray(data, count, root->left);
        huffmanTreeToArray(data, count, root->right);
    }
}
void free_data(node *root){
    if(root->left!=NULL){
        free_data(root->left);
        free_data(root->right);
    }
    free(root);
}

int main() {

    SetConsoleOutputCP(866);
    SetConsoleCP(866);
    wchar_t  str[1000000];
    fgetws(str, sizeof(str) / sizeof(str[0]), stdin);
```

```
    wchar_t  *begin = str;
    int size = 0;
    while (*begin!='\n'){
        size++;
        begin++;
    }
    *begin = '\0';
    begin = str;
    int count = 0;
    node *data[size];
    while (*begin!='\0'){
        int flag = 0;
        for(int i = 0; i < count; i++){
            if(data[i]->letter == *begin){
                flag = 1;
                data[i]->count++;
                break;
            }
        }
        if(flag == 0){
            data[count] = (node *)malloc(sizeof(node));
            data[count]->letter = *begin;
            data[count]->count = 1;
            data[count]->code[0] = '\0';
            data[count]->left = NULL;
            data[count]->right = NULL;
            count++;
        }
        begin++;
    }
    sort(count, data);
    while (count != 1){
        node *temp = malloc(sizeof(node));
        temp->count = data[count-1]->count + data[count-2]->count;
        temp->letter = '\0';
        temp->code[0] = '\0';
        temp->left = data[count-1];
        temp->right = data[count-2];
        data[count-2] = temp;
        count--;
        sort(count, data);
    }

    node *root = data[0];
    char code[300];
    haffman(root, code, code);

    count = 0;
    huffmanTreeToArray(data, &count, root);
    char answer[size * 100];
    begin = str;
    char *ansBg = answer;
    while (*begin!='\0'){
        for(int i = 0; i < count; i++){
            if(data[i]->letter == *begin){
                char *bg = data[i]->code;
                while (*bg!='\0'){
                    *ansBg=*bg;
                    bg++;
                    ansBg++;
                }
                break;
            }
```

```
        }
        begin++;
    }
    *ansBg = '\0';
    printf("\n%s", answer);
    int s = 0;
    ansBg = answer;
    while (*ansBg!='\0'){
        ansBg++;
        s++;
    }
    printf("\nsize: %d\n", s);
    double K = 8*size/(double)s;
    printf("K = %f\n", K);
    double sr = 0;
    for(int i = 0; i < count; i++){
        int len = 0;
        char *bg = data[i]->code;
        while (*bg!='\0'){
            bg++;
            len++;
        }
        sr+= ((double)data[i]->count/size)*len;
    }
    double d = 0;
    for(int i = 0; i < count; i++){
        int len = 0;
        char *bg = data[i]->code;
        while (*bg!='\0'){
            bg++;
            len++;
        }
        d+= ((double)data[i]->count/size) * (len - sr) * (len - sr);
    }
    printf("D = %f\n", d);
    free_data(root);
    return 0;
}
```

Примеры:

```
C:\Users\admin\CLionProjects\Project17\cmake-build-debug\Project17.exe
в чащах юга жил бы цитрус? Да, но фальшивый экземпляр!

1111000111001010111000010111011001110101001010100010010001101111100100111011110010011001101000010111010000010001110001000101010101100001011111011100010100101011111101000101011011011100111110100011010111010011011111011011000111000011111110001101110110010
size: 258
K = 1.674419
D = 2.395062

Process finished with exit code 0
```

```
C:\Users\admin\CLionProjects\Project17\cmake-build-debug\Project17.exe
Victoria nulla est, Quam quae confessos animo quoque subjugat hostes

1110001101110001011100011100111101111111011101001010000100001111011100001011111100101010110010101111100111011110101011111000100011010011000110000100100000110111111101011011100110001011110101000011101010110010100101001101101101001010101111011110110010000000101111100001
size: 278
K = 1.956835
D = 1.315744

Process finished with exit code 0
```

Размер закодированного сообщения 1 методом Хаффмана на 2 бита меньше, коэффициент сжатия соответственно больше, однако дисперсия у метода Шеннона-Фано меньше, это говорит о том, что символы, закодированные методом Шеннона-Фано более приближённые по размерам, но это не помогло получить более сжатый размер.

Размер закодированного сообщения 2 методом Хаффмана также меньше, на 5 бит, коэффициент сжатия также больше, но вновь дисперсия меньше у метода Шеннона-Фано, значит коды символов более сбалансированные.

Вывод: в ходе проделанной работы я реализовал алгоритм кодирования методом Шеннона-Фано и Хаффмана, после сравнения результатов заметил, что коэффициент сжатия у метода Хаффмана больше, значит сжатие прошло лучше, получили более короткое сообщение, но у метода Шеннона-Фано меньше дисперсия, значит таким методом получаются более сбалансированные коды символов, нет больших скачков.