Министерство образования Республики Беларусь

Учреждение образования

«Брестский государственный технический университет»

Кафедра ИИТ

Лабораторная работа №5

По дисциплине: «Языки программирования»

**Тема:** «Структуры данных: Дерево»

Выполнил:

Студент 1-го курса

Группы ПО-7

Лобан К.Ю.

Гункевич И.А.

Проверил:

Анфилец С.В.

Брест 2021

**Ход работы:**

**Текст программы:**

#include <iostream>

#include <string> //работа с классом string

#include <iomanip> //setw для человеческого вывода

#include <fstream> //для работы с текстовым файлом

using namespace std;

struct Tree {

Tree\* parent, //указатель на родителя

\* left, //указатель на левое поддерево

\* right; //указатель на правое поддерево

char x; //для символа

int chislo\_vhodov; //число вхождений

string code\_simvola; //для вывода кода символа

};

Tree\* createTree(char symbol, int chislo\_vhodov, string code\_simvola) {

struct Tree\* node;

node = new Tree; //выделим память

if (node != NULL) {

node->left = node->right = NULL; //инициализация ветвей пустотой

node->x = symbol; //присваивания значений

node->chislo\_vhodov = chislo\_vhodov;

node->code\_simvola = code\_simvola;

}

return node; //возвращаем адрес указанной структуры

}

void addingTree(Tree\* tree, char symbol, int chislo\_vhodov, string code\_simvola) {

struct Tree\* parent;

if (tree == NULL) {

return; //выход если пусто

}

for (parent = tree; tree != NULL; ) {

if (chislo\_vhodov < tree->chislo\_vhodov) { // условие добавление левого потомка

tree = tree->left;

}

else if (chislo\_vhodov > tree->chislo\_vhodov) {// условие добавление правого потомка

tree = tree->right;

}

else {

return; //иначе выход

}

}

}

void Shannon\_Fano(Tree\* tree, Tree\* symbols, int konez, int nachalo = 0,

string vetka = "", const string& fullvetka = "") {

//Теперь нужно разбить массив частот поровну, чтобы в первой группе сумма частот была максимальна равна сумме частот во второй группе

//Символам из первой группы присваиваются нули в их кодах, символам из второй группы - единицы (прибавляем 0 или 1 и сдвигаем влево)

addingTree(tree, symbols[nachalo].x, symbols[nachalo].chislo\_vhodov, vetka); //структура, символ, число вводов, строка

string code\_simvola = fullvetka + vetka;

if (nachalo == konez) { //дошли до символа - выводим значения

symbols[nachalo].code\_simvola = code\_simvola;

return;

}

int i;

int sum\_vhodov = 0; //сумма входов

for (i = nachalo; i <= konez; i++) {

sum\_vhodov = sum\_vhodov + symbols[i].chislo\_vhodov;

}

int sum2;

sum\_vhodov = sum\_vhodov / 2; //идеал равных частот

sum2 = symbols[nachalo].chislo\_vhodov;

//ищем границу раздела массива частот для наиболее равномерного разделения, abs - модуль

for (i = nachalo + 1; abs(sum\_vhodov - (sum2 + symbols[i].chislo\_vhodov)) <

abs(sum\_vhodov - sum2) && i < konez; i++) {

sum2 = sum2 + symbols[i].chislo\_vhodov;

}

string zero = "0";

string one = "1";

Shannon\_Fano(tree, symbols, i - 1, nachalo, zero, code\_simvola);

Shannon\_Fano(tree, symbols, konez, i, one, code\_simvola);

}

int main() {

setlocale(0, "");

string iz\_faila;

ifstream file\_out("text2.txt");

if (!file\_out.is\_open()) {

cout << "Файл не наден." << endl;

}

for (int i = 0; !file\_out.eof(); i++) { //пока не достигнут конец файла

char temp[1024]; //для файла

file\_out.getline(temp, 1024);

iz\_faila = iz\_faila + temp; //запись в строку iz\_faila содержимого файла

}

file\_out.close();

Tree\* symbols = new Tree[iz\_faila.length()]; //массив символов и частот строки

int N = 0; //размер массива

for (int i = 0; i < iz\_faila.length(); i++) { //обрабатываем всю строку

int j = 0;

while (j < N && symbols[j].x != iz\_faila[i]) { //поиск в массиве символа

j++;

}

if (j == N) { //если символ не найден, то добавляем этот символ

//и учитываем, что его частота = 1

symbols[N].x = iz\_faila[i];

symbols[N++].chislo\_vhodov = 1;

}

else { //если найден, увеличиваем значение частоты

symbols[j].chislo\_vhodov = symbols[j].chislo\_vhodov + 1;

}

}

//сортировка массива

bool isSorted = false;

while (!isSorted) {

isSorted = true;

for (int i = 0; i < N - 1; i++) {

if (symbols[i].chislo\_vhodov < symbols[i + 1].chislo\_vhodov) {

swap(symbols[i], symbols[i + 1]); //сортировка swap'ом

isSorted = false;

}

}

}

string parent = "0";

Tree\* tree = createTree(' ', 0, parent); //создание 1-го элемента дерева

Shannon\_Fano(tree, symbols, N - 1); //алгоритм кодирования Шеннона-Фано

ofstream file\_in("Таблица кодов(Ш-Ф).txt");

file\_in << setw(10) << left << "Символ" << setw(10) << "Частота"

<< "Код" << endl; //в файл

cout << setw(10) << left << "Символ" << setw(10) << left << "Частота"

<< "Код" << endl; //консоль

for (int i = 0; i < N; i++) { //символ / кол-во нахождений / код

file\_in << setw(10) << symbols[i].x << setw(10) << symbols[i].chislo\_vhodov

<< symbols[i].code\_simvola << endl; // в файл

cout << setw(10) << symbols[i].x << setw(10) << symbols[i].chislo\_vhodov

<< symbols[i].code\_simvola << endl; // консоль

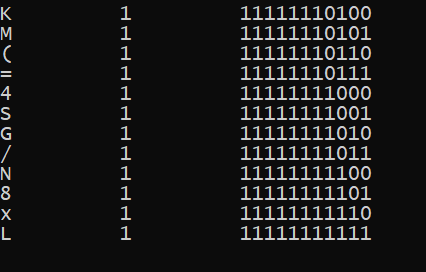
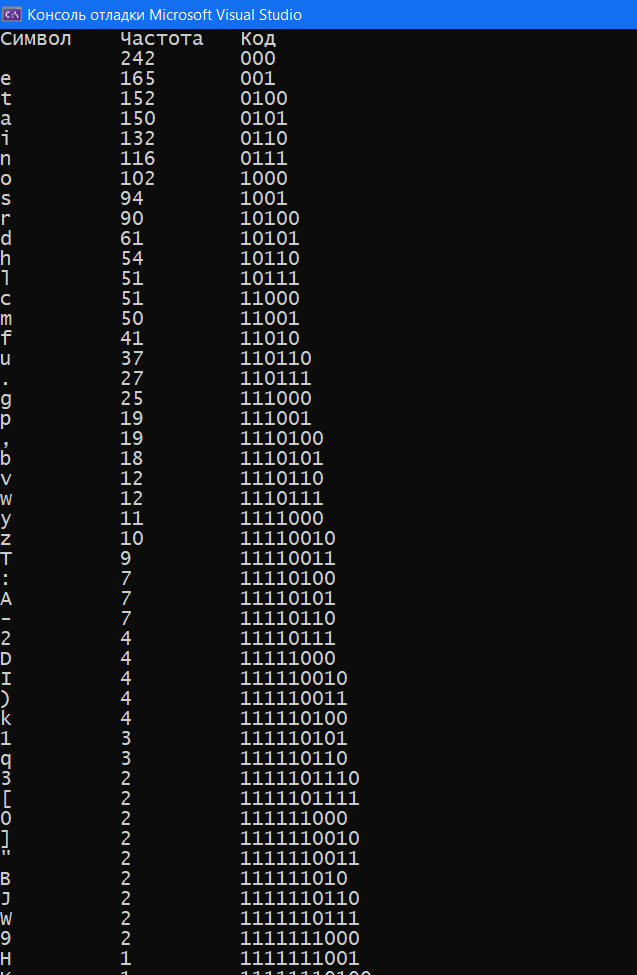
}

cout << endl;

file\_in.close();

}

Результаты работы программы:



**Вывод:** В ходе данной лабораторной работы мы изучили принцип работы алгоритма кодирования Шеннона-Фано и разобрались с построением бинарного дерева.