Министерство образования Республики Беларусь

Учреждение образования

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет компьютерных систем и сетей

Кафедра электронных вычислительных машин

Лабораторная работа № 4

«Бинарные деревья»

Проверил: Выполнил:

асс. каф. ЭВМ ст. гр. 250505

И.Г. Скиба Бакунович Н.А.

МИНСК 2023

# 1 Цель и задачи лабораторной работы

Цель работы – освоить принципы работы с бинарными деревьями.

**Задача:** написать программу, которая должна определять загаданный пользователем объект. Для этого она должна управлять двоичным деревом, в котором хранится информация о загадываемых объектах. Работа проходит по следующему сценарию:

1. Пользователь загадывает объект.
2. Программа задает вопросы, на которые пользователь отвечает да/нет.
3. Каждый ответ приводит к очередному вопросу из дерева. Так до тех пор, пока не будет достигнут конец (лист) с ответом.
4. Если ответ неверный, программа должна предложить ввести данные о загаданном объекте (отличительный вопрос) и запомнить новый объект.

# 2 Результаты выполнения лабораторной работы

На рисунке 1.1 приведена блок-схема алгоритма для выполнения задания

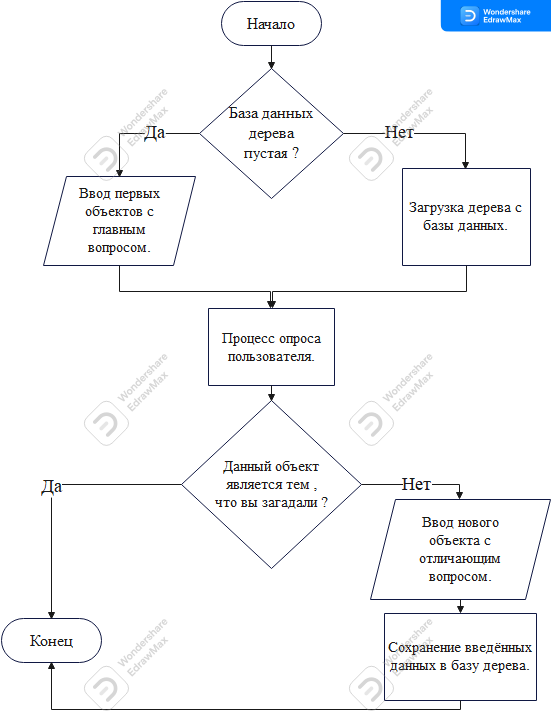


Рисунок 1.1 - Блок схема программы

**1.2** Листинг компьютерной программы

#define \_CRT\_SECURE\_NO\_WARNINGS

#define CONST256 256

#include <stdio.h>

#include <stdlib.h>

#include < time.h >

#include <string.h>

#include <windows.h>

#include "Source.h"

#include "tree.h"

int main()

{

log\_message("Program started");

char\* buffer = (char\*)malloc(128 \* sizeof(char));

tree\* tree = NULL;

FILE\* read = fopen("questions.txt", "r");

int c = fgetc(read);

if (c == EOF) {

log\_message("Creating new tree");

printf("Insert first object\n");

char object[CONST256];

scanf("%s", object);

printf("Insert question for this object: %s \n", object);

char question[CONST256];

int c;

while ((c = getchar()) != '\n' && c != EOF) {}

fgets(question, CONST256, stdin);

strtok(question, "\n");

tree = create(question, read);

tree->mark = question;

tree->right = create(object, read);

printf("Insert second object\n");

scanf("%s", object);

tree->left = create(object, read);

log\_message("Finished creating new tree");

}

else {

log\_message("Loading tree");

fseek(read, 0, SEEK\_SET);

tree = create(buffer, read);

tree = insert(read);

log\_message("Finished loading tree from file");

}

ask(tree, read);

read = fopen("questions.txt", "w");

if (read == NULL) {

printf("Cant open file!");

return 0;

}

log\_message("Saving tree to file");

save\_tree(read, tree);

fclose(read);

log\_message("Finished saving tree to file");

log\_message("Program finished");

FILE\* logfile = fopen("log.txt", "a");

fprintf(logfile, "- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - \n");

return 0;

}

tree\* create(char\* w, FILE\* file) {

tree\* node = (tree\*)malloc(sizeof(tree));

node->mark = (char\*)malloc(strlen(w) \* sizeof(char) + 1);

strcpy(node->mark, w);

int i = 0;

while (w[i] != '\n' && w[i] != '\0') {

i++;

}

//strncpy(node->mark, w, i);

node->mark[i] = '\0';

node->right = NULL;

node->left = NULL;

return node;

}

tree\* insert(FILE\* file)

{

char\* buffer = (char\*)malloc(128 \* sizeof(char));

if (fgets(buffer, strlen(buffer), file) == NULL)

return NULL;

if (strcmp(buffer, "^\n") == 0)

return NULL;

tree\* tree = create(buffer, file);

tree->right = insert(file);

tree->left = insert(file);

return tree;

}

void add(tree\* tree, char\* question, char\* object, FILE\* file) {

char\* old\_question = tree->mark;

tree->mark = question;

tree->left = create(old\_question, file);

tree->right = create(object, file);

log\_message("Added new question and object to tree");

}

void ask(tree\* tree, FILE\* file) {

if (tree == NULL) {

printf("NULL\n");

return;

}

if (tree->right == NULL && tree->left == NULL) {

printf("It can be.... %s\n", tree->mark);

printf("Am I right?\n");

char answer[4];

scanf("%s", answer);

if (strcmp(answer, "yes") == 0) {

printf("I guess!\n");

log\_message("Akinator guessed right.");

return;

}

else

{

printf("I didn't guess which object you made up?\n");

char object[CONST256];

int c;

while ((c = getchar()) != '\n' && c != EOF) {}

fgets(object, CONST256, stdin);

strtok(object, "\n");

printf("What is a question that distinguishes %s from %s?\n", object, tree->mark);

char question[CONST256];

fgets(question, CONST256, stdin);

strtok(question, "\n");

add(tree, \_strdup(question), \_strdup(object), file);

printf("Thank you! I'll remember that for next time.\n");

log\_message("A new object has been added .");

log\_message\_object("Object", object);

return;

}

}

printf("%s\n", tree->mark);

char answer[4];

scanf("%s", answer);

if (strcmp(answer, "yes") == 0) {

ask(tree->right, file);

}

else if (strcmp(answer, "no") == 0) {

ask(tree->left, file);

}

else {

printf("Answer must be 'yes' or 'no'\n");

ask(tree, file);

}

}

void save\_tree(FILE\* file, tree\* current\_node) {

if (current\_node == NULL) {

fprintf(file, "^\n");

return;

}

fprintf(file, "%s\n", current\_node->mark);

save\_tree(file, current\_node->right);

save\_tree(file, current\_node->left);

}

void log\_message(char\* message) {

FILE\* file = fopen("log.txt", "a");

time\_t now = time(NULL);

char\* time\_str = ctime(&now); //преобразования значения time\_t в строку, содержащую текстовое представление времени и даты.

strtok(time\_str, "\n"); // remove newline character

fprintf(file, "[%s] %s\n", time\_str, message);

fclose(file);

}

void log\_message\_object(char\* message, char\* object) {

FILE\* file = fopen("log.txt", "a");

time\_t now = time(NULL);

char\* time\_str = ctime(&now); //преобразования значения time\_t в строку, содержащую текстовое представление времени и даты.

strtok(time\_str, "\n"); // remove newline character

fprintf(file, "[%s] %s : %s\n", time\_str, message, object);

fclose(file);

}

typedef struct tree {

char\* mark;

struct tree\* left;

struct tree\* right;

}tree;

tree\* create(char\* w, FILE\* file);

void add(tree\* current\_node, char\* new\_question, char\* new\_object, FILE\* file);

tree\* insert(FILE\* file);

void ask(tree\* tree, FILE\* file);

tree\* load\_tree(FILE\* file);

void save\_tree(FILE\* file, tree\* current\_node);

void log\_message(char\* message);

void log\_message\_object(char\* message, char\* object);

**1.3** На рисунках 1.2 представлен скриншот выполнения программы

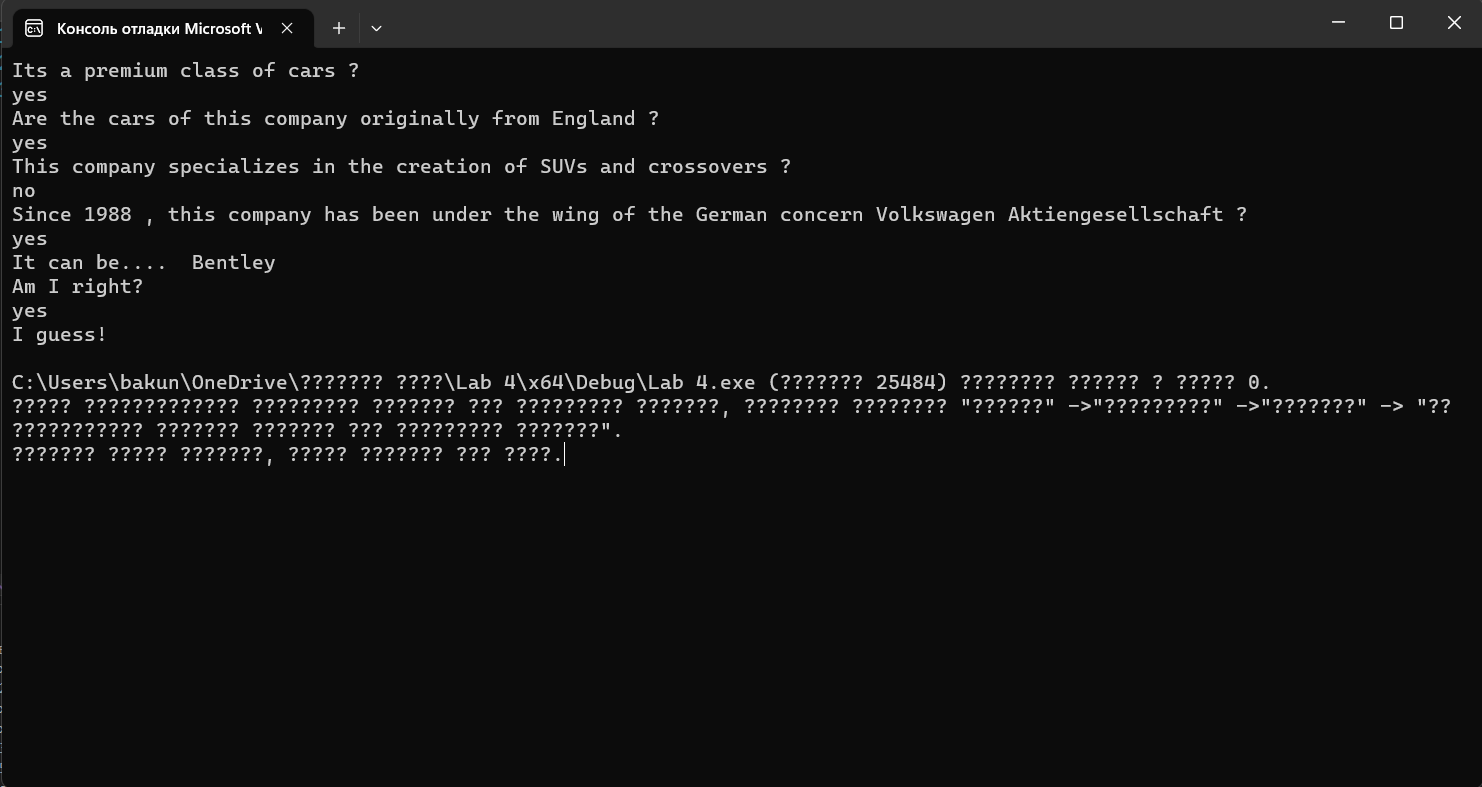


Рисунок 1.2 – Скрншот результата выполнения компьютерной программы