**Московский государственный технический**

**университет им. Н.Э. Баумана.**

Факультет «Информатика и управление»

Кафедра ИУ5. Курс «Основы программирования»

Отчет по лабораторной работе № 9

«Ввод, сортировка и двоичный поиск в массиве структур»

| Выполнил: Чернев Николай Андреевич |  | Проверил: |
| --- | --- | --- |
| студент группы ИУ5-14Б |  | преподаватель каф. ИУ5 |
|  |  |  |
| Подпись и дата: |  | Подпись и дата: |

Москва, 2023 г.

**Задание**

Англо-русский словарь построен в виде массива структур Dictionary. Структура содержит английское слово и соответствующее ему русское слово. Максимальный размер словаря – 100 пар слов.

Разработать программу, которая:

* обеспечивает формирование словаря (добавление и удаление записей);
* записывает словарь, отсортированный по английским значениям слов, в файл;
* обеспечивает просмотр словаря;
* выполняет перевод слов с английского на русский, используя для поиска слова в словаре метод двоичного поиска в отсортированном массиве;
* выполняет перевод слов с русского на английский, используя для поиска слов в словаре метод перебора;

Программа должна обеспечивать диалог с помощью меню.

Начальное число слов в словаре равно 10.

Для исключения проблем, связанных с вводом кириллицы, вводите русские слова латинскими буквами, например: kot (cat), sobaka(dog) и т.п.

**Разработка алгоритма**

Описание переменных

Структура Dictionary:

* char \*word\_ru - слово на русском
* char \*word\_en - слово на английском
* Метод init:
  + int len\_ru - длина слова на русском
  + int len\_en - длина слова на английском

Функция menu:

* int cmd - номер команды

Функция get\_words\_from\_file:

* std::ifstream fin - переменная для считывания данных из файла
* char word\_en[31] - слово на английском
* char word\_ru[31] - слово на русском

Функция add\_word:

* char word\_ru[31] - новое слово на русском
* char word\_en[31] - новое слово на английском
* int i - счетчик цикла

Функция delete\_word:

* char del\_word[31] - слово, которое хотим удалить(на английском)
* int i - счетчик цикла

Функция translate\_to\_ru:

* int l, r - левая и правая границы бинарного поиска
* int mid - индекс серединного элемента
* char search\_word\_en[31] - искомое слово на английском

Функция translate\_to\_en:

* char search\_word\_ru[31] - искомое слово на русском
* bool is\_exist - отвечает за наличие искомого слова: false - нет слова, true - есть
* int i - счетчик цикла

Функция output\_dict\_to\_file:

* std::ofstream fout - переменная для вывода словаря в файл
* int i - счетчик цикла

**Описание функций**

void Dictionary::init(char new\_word\_ru[31], char new\_word\_en[31]) - принимает слова на русском и английском и записывает их в структуру Dictionary, переводя в динамику, ничего не возвращает

bool is\_equal(char \*s1, char \*s2) - принимает 2 строки s1 и s2, возвращает true, если они равны; false, если не равны

bool is\_bigger(char \*s1, char \*s2) - принимает 2 строки s1 и s2, возвращает true, если s1 > s2, false в противном случае

int menu() - ничего не принимает, выводит список команд, считывает номер выбранной команды, возвращает этот номер

void get\_words\_from\_file(Dictionary \*dict, int &n) - принимает массив структур dict и копию n - кол-ва пар слов в словаре. Считывает из файла “dict.txt” слова и записывает их в dict, ничего не возвращает

void add\_word(Dictionary \*dict, int &n) - принимает массив структур dict и копию n - кол-ва пар слов в словаре. Считывает пару слов(на русском и на английском) и добавляет их в dict, предварительно находя место для них так, чтобы словарь оставался отсортированным. Ничего не возвращает

void delete\_word(Dictionary \*dict, int &n) - принимает массив структур dict и копию n - кол-ва пар слов в словаре. Считывает слово на английском, которое надо удалить, ищет его в dict, если не находит - выводит сообщение об этом, если находит - удаляет эту пару слов из dict, ничего не возвращает

void translate\_to\_ru(Dictionary \*dict, const int &n) - принимает массив структур dict и копию n - кол-ва пар слов в словаре. Считывает слово на английском, ищет бинарным поиском его в dict, если не находит - выводит сообщение об этом, если находит - выводит это слово и русский перевод этого слова, ничего не возвращает

void translate\_to\_en(Dictionary \*dict, const int &n) - принимает массив структур dict и копию n - кол-ва пар слов в словаре. Считывает слово на русском, ищет его в dict, если не находит - выводит сообщение об этом, если находит - выводит это слово и английский перевод этого слова, ничего не возвращает

void print\_dict(Dictionary \*dict, const int &n) - принимает массив структур dict и копию n - кол-ва пар слов в словаре. Выводит в консоль все слова из dict, ничего не возвращает

void output\_dict\_to\_file(Dictionary \*dict, const int &n) - принимает массив структур dict и копию n - кол-ва пар слов в словаре. Выводит в файл “dict.txt” все слова из dict, ничего не возвращает

**Код программы**

header.h

#pragma once

#include <iostream>

#include <fstream>

struct Dictionary{

char \*word\_ru;

char \*word\_en;

void init(char new\_word\_ru[31], char new\_word\_en[31]);

};

bool is\_equal(char \*s1, char \*s2);

bool is\_bigger(char \*s1, char \*s2);

int menu();

void get\_words\_from\_file(Dictionary \*dict, int &n);

void add\_word(Dictionary \*dict, int &n);

void delete\_word(Dictionary \*dict, int &n);

void translate\_to\_ru(Dictionary \*dict, const int &n);

void translate\_to\_en(Dictionary \*dict, const int &n);

void print\_dict(Dictionary \*dict, const int &n);

void output\_dict\_to\_file(Dictionary \*dict, const int &n);

funcs.cpp

#include "header.h"

void Dictionary::init(char new\_word\_ru[31], char new\_word\_en[31]) {

int len\_ru = strlen(new\_word\_ru);

word\_ru = new char[len\_ru];

for (int i = 0; i < len\_ru; i++) {

word\_ru[i] = new\_word\_ru[i];

}

int len\_en = strlen(new\_word\_en);

word\_en = new char[len\_en];

for (int i = 0; i < len\_en; i++) {

word\_en[i] = new\_word\_en[i];

}

}

bool is\_equal(char \*s1, char \*s2){

if (strlen(s1) != strlen(s2))

return false;

for (int i = 0; i < strlen(s1); i++){

if (s1[i] != s2[i])

return false;

}

return true;

}

bool is\_bigger(char \*s1, char \*s2){

for (int i = 0; i < 31; i++){

if (s1[i] > s2[i]){

return true;

}

if (s1[i] < s2[i]){

return false;

}

}

return false;

}

int menu(){

int cmd;

std::cout << "Введите номер команды\n";

std::cout << "1 - добавление слов в словарь\n";

std::cout << "2 - удаление слов из словаря\n";

std::cout << "3 - перевод слов с английского на русский\n";

std::cout << "4 - перевод слов с русского на английский\n";

std::cout << "5 - просмотр словаря\n";

std::cout << "6 - вывод словаря в файл\n";

std::cout << "7 - выход\n";

std::cin >> cmd;

return cmd;

}

void get\_words\_from\_file(Dictionary \*dict, int &n) {

std::ifstream fin("dict.txt");

char word\_en[31], word\_ru[31];

while (fin >> word\_ru) {

fin >> word\_en;

dict[n++].init(word\_ru, word\_en);

fin.close();

}

}

void add\_word(Dictionary \*dict, int &n){

char word\_ru[31], word\_en[31];

std::cout << "Введите сначала слово на русском языке, затем на английском\n";

std::cin >> word\_ru >> word\_en;

if ((is\_bigger(dict[0].word\_en, word\_en)) || (n == 0)) {

for (int i = n - 1; i >= 0; i--) {

dict[i + 1] = dict[i];

}

dict[0].init(word\_ru, word\_en);

}

else {

for (int i = 1; i < n; i++) {

if (is\_bigger(dict[i].word\_en, word\_en)) {

for (int j = n - 1; j >= i; j--) {

dict[j + 1] = dict[j];

}

dict[i].init(word\_ru, word\_en);

break;

}

}

if (is\_bigger(word\_en, dict[n - 1].word\_en)) {

dict[n].init(word\_ru, word\_en);

}

}

n++;

}

void delete\_word(Dictionary \*dict, int &n){

std::cout << "Введите слово на английском, которое хотите удалить\n";

char del\_word[31];

std::cin >> del\_word;

int num = -1;

for (int i = 0; i < n; i++) {

if (is\_equal(dict[i].word\_en, del\_word)) {

num = i;

break;

}

}

if (num == -1) {

std::cout << "Такого слова нет в словаре\n";

}

else {

delete[] dict[num].word\_en;

delete[] dict[num].word\_ru;

for (int i = num; i < n; i++) {

dict[i] = dict[i + 1];

}

n--;

}

}

void translate\_to\_ru(Dictionary \*dict, const int &n){

int l = 0, r = n - 1;

int mid = (l + r) / 2;

char search\_word\_en[31];

std::cout << "Введите искомое слово на английском\n";

std::cin >> search\_word\_en;

while ((!is\_equal(dict[mid].word\_en, search\_word\_en)) && (l < r)){

if (is\_bigger(dict[mid].word\_en, search\_word\_en)){

r = mid;

}

else{

l = mid + 1;

}

mid = (l + r) / 2;

}

if (is\_equal(dict[mid].word\_en, search\_word\_en)) {

std::cout << "Слово " << search\_word\_en << " переводится на русский как " << dict[mid].word\_ru << '\n';

}

else{

std::cout << "Такого слова нет в словаре\n";

}

}

void translate\_to\_en(Dictionary \*dict, const int &n){

char search\_word\_ru[31];

std::cout << "Введите искомое слово на русском\n";

std::cin >> search\_word\_ru;

bool is\_exist = false;

for (int i = 0; (i < n) && (!is\_exist); i++){

if (is\_equal(dict[i].word\_ru, search\_word\_ru)) {

std::cout << "Слово " << search\_word\_ru << " переводится на английский как " << dict[i].word\_en << '\n';

is\_exist = true;

}

}

if (!is\_exist) {

std::cout << "Такого слова нет в словаре\n";

}

}

void print\_dict(Dictionary \*dict, const int &n){

for (int i = 0; i < n; i++){

std::cout << dict[i].word\_ru << ' ' << dict[i].word\_en << '\n';

}

}

void output\_dict\_to\_file(Dictionary \*dict, const int &n){

std::ofstream fout("dict.txt");

for (int i = 0; i < n; i++){

fout << dict[i].word\_ru << ' ' << dict[i].word\_en << '\n';

}

fout.close();

}

main.cpp

#include "header.h"

int main() {

Dictionary dict[100];

int word\_ind = 0;

bool run = true;

get\_words\_from\_file(dict, word\_ind);

while (run){

switch (menu()) {

case 1:

add\_word(dict, word\_ind);

break;

case 2:

delete\_word(dict, word\_ind);

break;

case 3:

translate\_to\_ru(dict, word\_ind);

break;

case 4:

translate\_to\_en(dict, word\_ind);

break;

case 5:

print\_dict(dict, word\_ind);

break;

case 6:

output\_dict\_to\_file(dict, word\_ind);

break;

case 7:

run = false;

break;

default:

std::cout << "Введите число от 1 до 7\n";

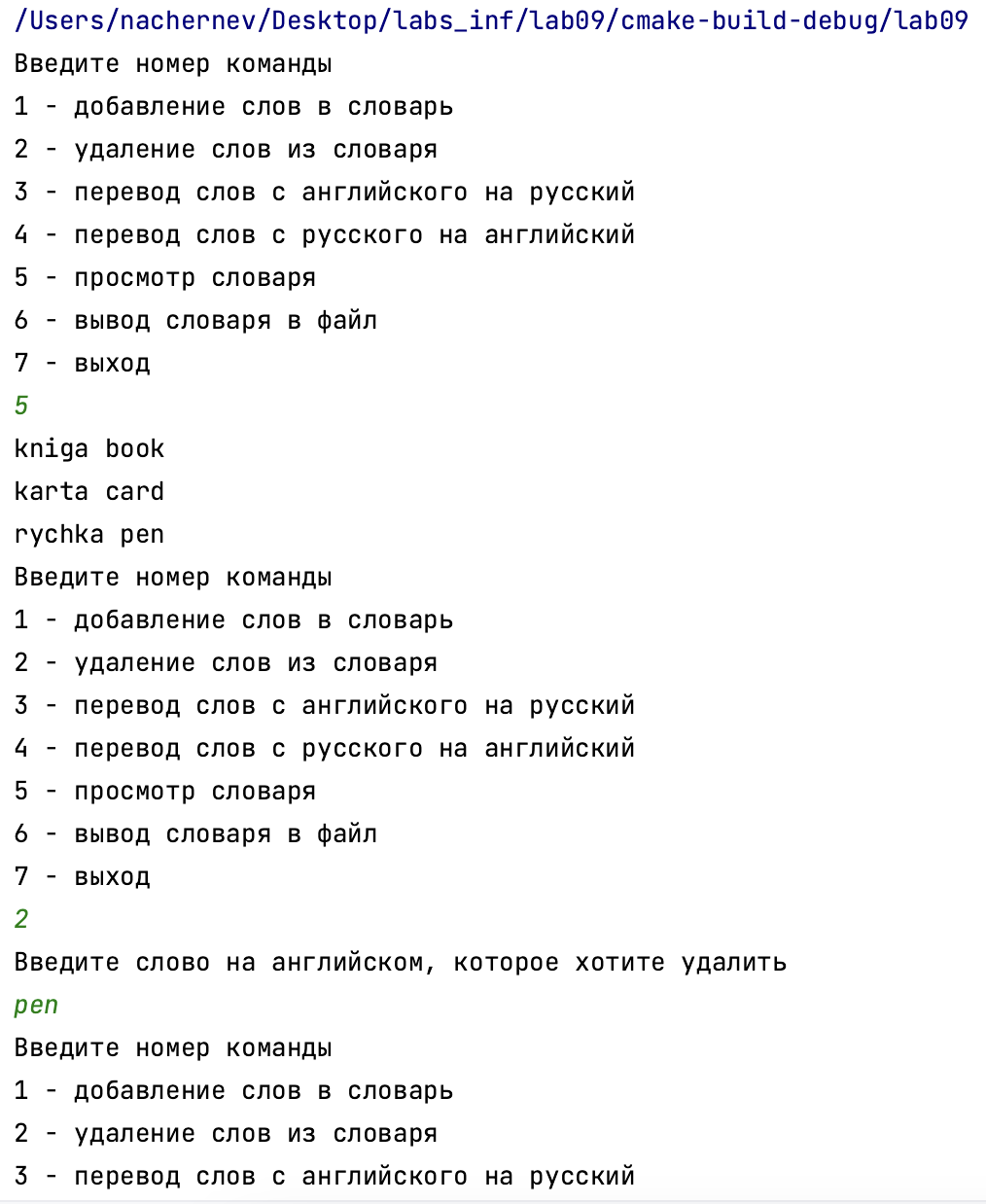
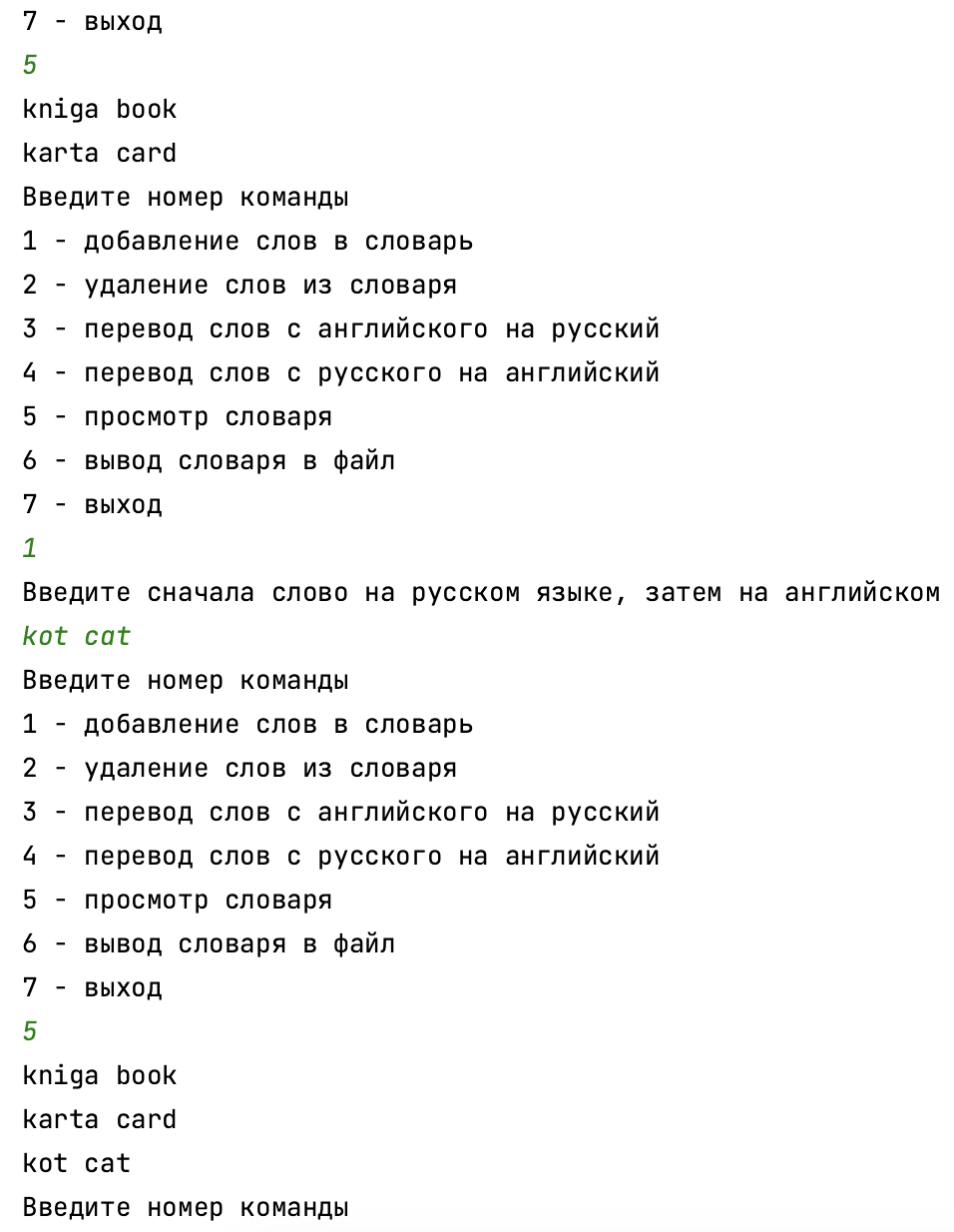
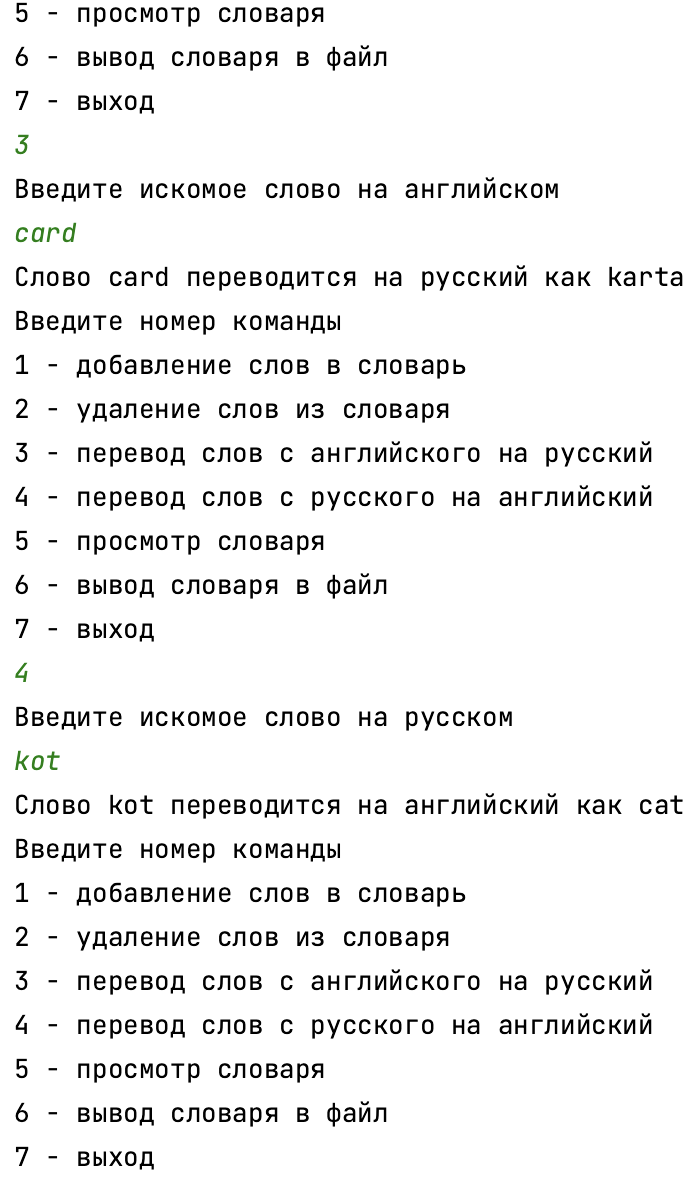
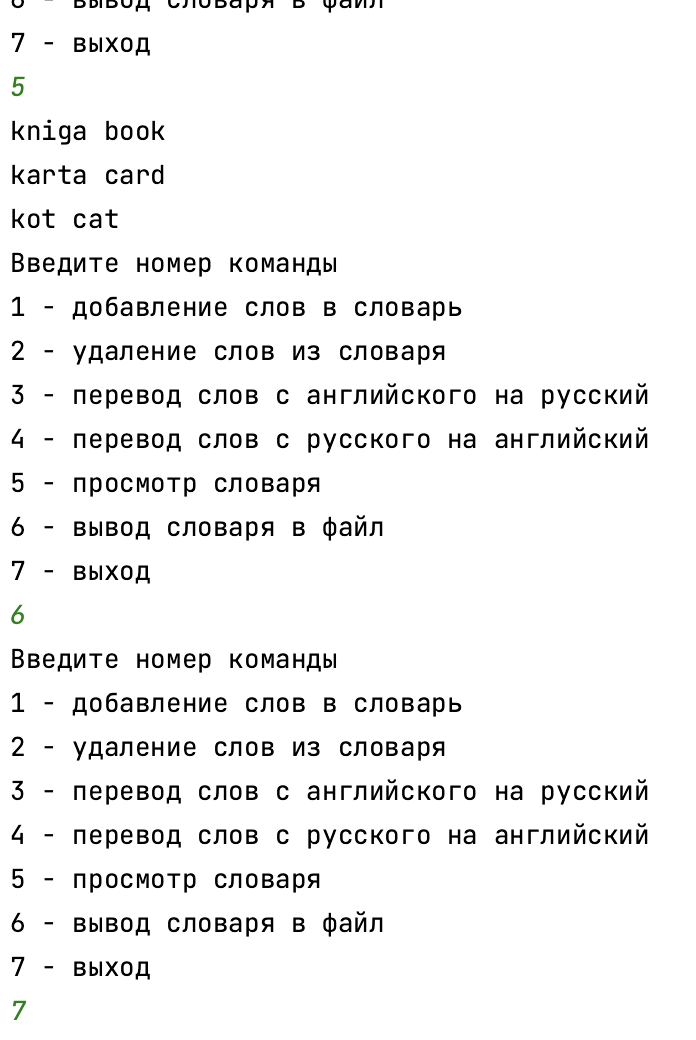
break;

}

}

return 0;

}

Анализ результатов

До запуска программы | После завершения работы программы

