**Московский государственный технический**

**университет им. Н.Э. Баумана**

Факультет «Информатика и системы управления»

Кафедра ИУ5 «Системы обработки информации и управления»

Отчет по лабораторной работе №9

«**Ввод, сортировка и двоичный поиск в массиве структур**»

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Выполнил:** | |  | **Принял:** | |
| ФИО: | \_Цыпышев Т. А.\_\_\_\_\_ |  | ФИО: | \_**Самохвалов А. Э.**\_\_\_ |
| Группа: | \_ИУ5-11Б\_\_\_\_\_\_\_\_\_\_ |  | Должность: | \_Преподаватель\_\_\_\_ |
| Дата: | \_21.11.2022\_\_\_\_\_\_\_\_ |  | Дата: | \_21.11.2022\_\_\_\_\_\_\_\_ |
| Подпись: | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |  | Подпись: | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |

Москва, 2022 г.

**Постановка задачи**

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.

**Разработка алгоритма**

**Переменные:**

* int item – используется для выбора раздела в меню
* #define max\_len\_word 50 – максимальная длина слов
* #define size\_dictionary 100 – длина словаря
* struct words – структура, состоящая из английского и русского слова
* words dictionary[size\_dictionary] – словарь
* int count – количество слов в словаре
* int last\_position – позиция последнего слова в словаре

**Основные функции:**

* void AddWord() – добавление слова в словарь
* void RemoveWord() – удаление слова из словаря
* void Translate(int mode) – вывод перевода (mode = 0 English -> Russian, = 1 Russian -> English)
* void ViewingDictionary() – вывод словаря в консоль
* void LoggingDictionary() – сохранение словаря в файл
* void Help() – вывод всех команд в консоль

**Вспомогательные функции:**

* bool ComparisonWords(char word1[], char word2[]) – сопоставление двух слов
* int FindWords(char word[], int mode) – поиск слова в словаре (mode = 0 English, = 1 Russian)
* void EditDictionary(int num, char word1[], char word2[]) – изменение словаря
* bool SortingWords(char word1[], char word2[]) – сортировка двух слов
* void SortingDictionary() – сортировка словаря

**Текст программы**

**Main.cpp**

/// Copyright 2022 ttsypyshev [ttsypyshev01@gmail.com](mailto:ttsypyshev01@gmail.com)

#include <iostream>

#include <fstream>

using namespace std;

///global variables

#define max\_len\_word 50

#define size\_dictionary 100

struct words {

char eng[max\_len\_word];

char rus[max\_len\_word];

};

words dictionary[size\_dictionary];

int count = 0, last\_position = 0;

///other fubctions

bool ComparisonWords(char word1[], char word2[]) {

for (int i = 0; i < max\_len\_word; i++) {

if (word1[i] != word2[i])

return false;

if (word1[i] == '\0' && word2[i] == '\0')

return true;

}

return true;

}

int FindWords(char word[], int mode) {

if (mode == 0) {

for (int i = 0; i < count; i++)

if (ComparisonWords(dictionary[i].eng, word))

return i;

} else if (mode == 1) {

for (int i = 0; i < count; i++)

if (ComparisonWords(dictionary[i].rus, word))

return i;

}

return -1;

}

void EditDictionary(int num, char word1[], char word2[]) {

for (int i = 0; i < max\_len\_word; ++i) {

dictionary[num].eng[i] = word1[i];

dictionary[num].rus[i] = word2[i];

}

}

bool SortingWords(char word1[], char word2[]) {

for (int i = 0; word1[i] != '\0' || word2[i] != '\0'; i++)

if (word1[i] > word2[i])

return true;

else if (word1[i] < word2[i])

return false;

}

void SortingDictionary() {

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

for (int j = 0; j < count - i - 1; j++)

if (SortingWords(dictionary[j].eng, dictionary[j + 1].eng))

swap(dictionary[j], dictionary[j + 1]);

}

///main functions

void AddWord() {

char word\_eng[max\_len\_word], word\_rus[max\_len\_word];

cout << "Enter a word in English:";

cin >> word\_eng;

int index = FindWords(word\_eng, 0);

if (index == -1) {

cout << "Enter a word in Russian:";

cin >> word\_rus;

EditDictionary(last\_position, word\_eng, word\_rus);

count++;

last\_position++;

cout << "The word '" << word\_eng << "' (" << word\_rus << ") has been added to the dictionary\n";

} else { cout << "The word '" << word\_eng << "' was added earlier\n"; }

}

void RemoveWord() {

char word\_eng[max\_len\_word];

cout << "Enter a word in English:";

cin >> word\_eng;

int index = FindWords(word\_eng, 0);

if (index == -1) { cout << "The word '" << word\_eng << "' is not in the dictionary\n"; }

else {

cout << "The word '" << word\_eng << "' has been removed from the dictionary\n";

EditDictionary(index, dictionary[last\_position - 1].eng, dictionary[last\_position - 1].rus);

EditDictionary(last\_position - 1, "", "");

last\_position--;

count--;

}

}

void Translate(int mode) {

char word[max\_len\_word];

cout << "Enter a word:";

cin >> word;

int index = FindWords(word, mode);

if (index == -1) { cout << "The word '" << word << "' is not in the dictionary\n"; }

else {

if (mode == 0) { cout << "The word '" << word << "' translates as '" << dictionary[index].rus << "'\n"; }

else { cout << "The word '" << word << "' translates as '" << dictionary[index].eng << "'\n"; }

}

}

void ViewingDictionary() {

for (int i = 0; i < count; i++)

if (dictionary[i].rus[0] != '0')

cout << i + 1 << ". " << dictionary[i].eng << " - " << dictionary[i].rus << endl;

cout << endl;

}

void LoggingDictionary() {

ofstream log\_file("../log.txt", ios\_base::trunc | ios\_base::out);

if (!(log\_file)) { log\_file.open("../log.txt"); }

SortingDictionary();

for (int i = 0; i < count; i++)

if (dictionary[i].rus[0] != '0')

log\_file << i + 1 << ". " << dictionary[i].eng << " - " << dictionary[i].rus << endl;

cout << "The dictionary is written to a file\n";

}

void Help() {

cout << "1 - Adding a word to the dictionary \n"

<< "2 - Deleting a word from the dictionary \n"

<< "3 - Translation of words from English to Russian \n"

<< "4 - Translation of words from Russian to English \n"

<< "5 - Displaying the dictionary on the screen \n"

<< "6 - Dictionary output to a file \n"

<< "7 - Help \n"

<< "8 - Completion of the program \n";

}

int main() {

int item = 7;

while (true) {

if (item == 1) AddWord();

else if (item == 2) RemoveWord();

else if (item == 3) Translate(0);

else if (item == 4) Translate(1);

else if (item == 5) ViewingDictionary();

else if (item == 6) LoggingDictionary();

else if (item == 7) Help();

else if (item == 8) return 0;

else { cout << "Incorrect values have been entered. Enter them again\n"; }

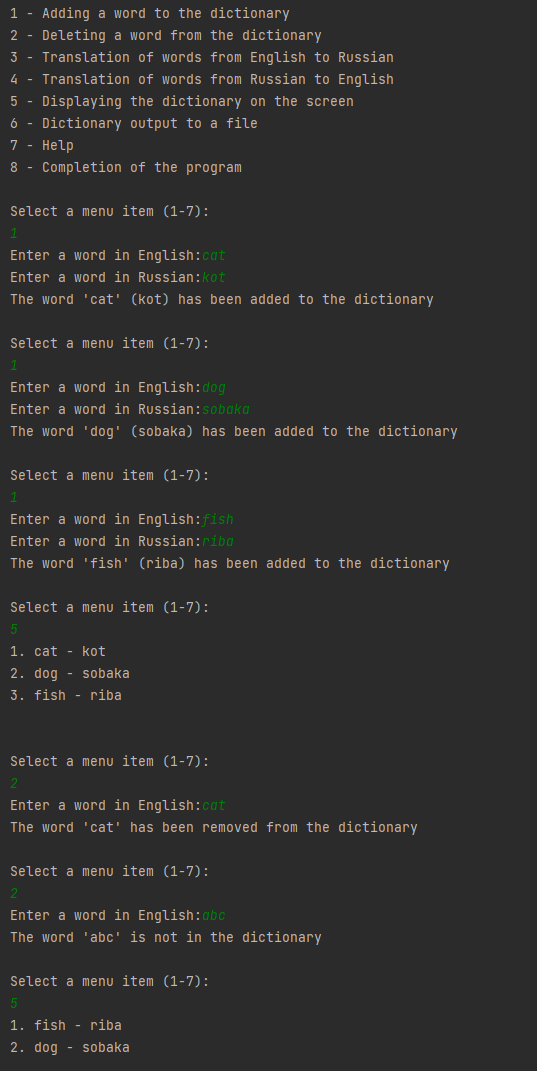
cout << "\nSelect a menu item (1-7):\n";

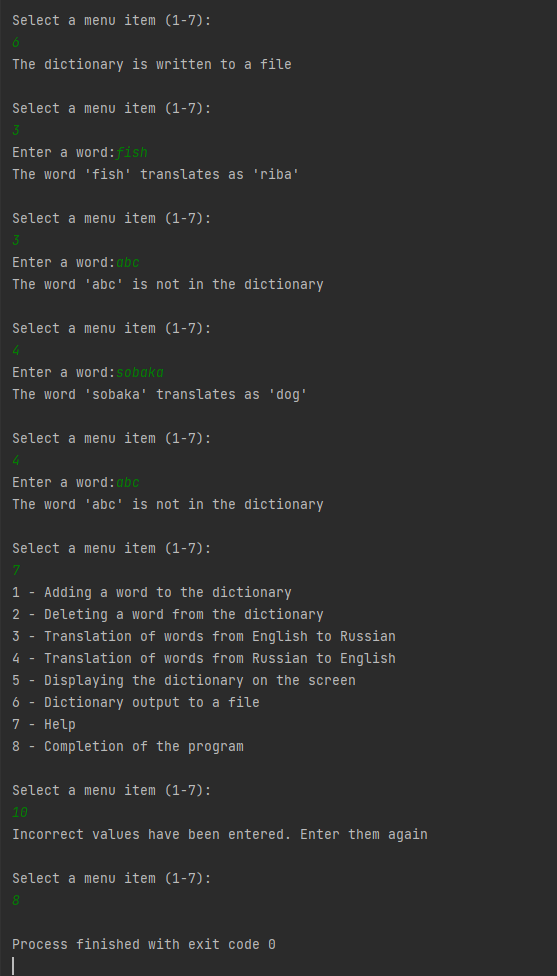
cin >> item;

}

}

**Анализ результатов**

****

****

**Вывод**

Я научился работать с массивами структур