МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное бюджетное образовательное учреждение
высшего профессионального образования
«Вятский государственный университет»
Факультет автоматики и вычислительной техники
Кафедра электронных вычислительных машин

Лабораторная работа № 4 по курсу «Технологии программирования»

Выполнил студент группы ИВТ-21 _____/Рзаев А. Э./
Проверил доцент кафедры ЭВМ _____/Долженкова М. Л./

Киров 2017

1 Задание

Написать DLL-библиотеку, содержащую функцию поиска различий по словам в двух текстовых файлах. Использовать явное и неявное связывание.

2 Результат работы

Экранные формы приведены в приложении А.

3 Листинг программы

Листинг библиотеки приведен в приложении Б.

4 Вывод

В ходе выполнения лабораторной работы была написана DLL-библиотека, в которой содержалась функция поиска различий по словам в двух текстовых файлах.

Приложение А
(обязательное)
Экранные формы

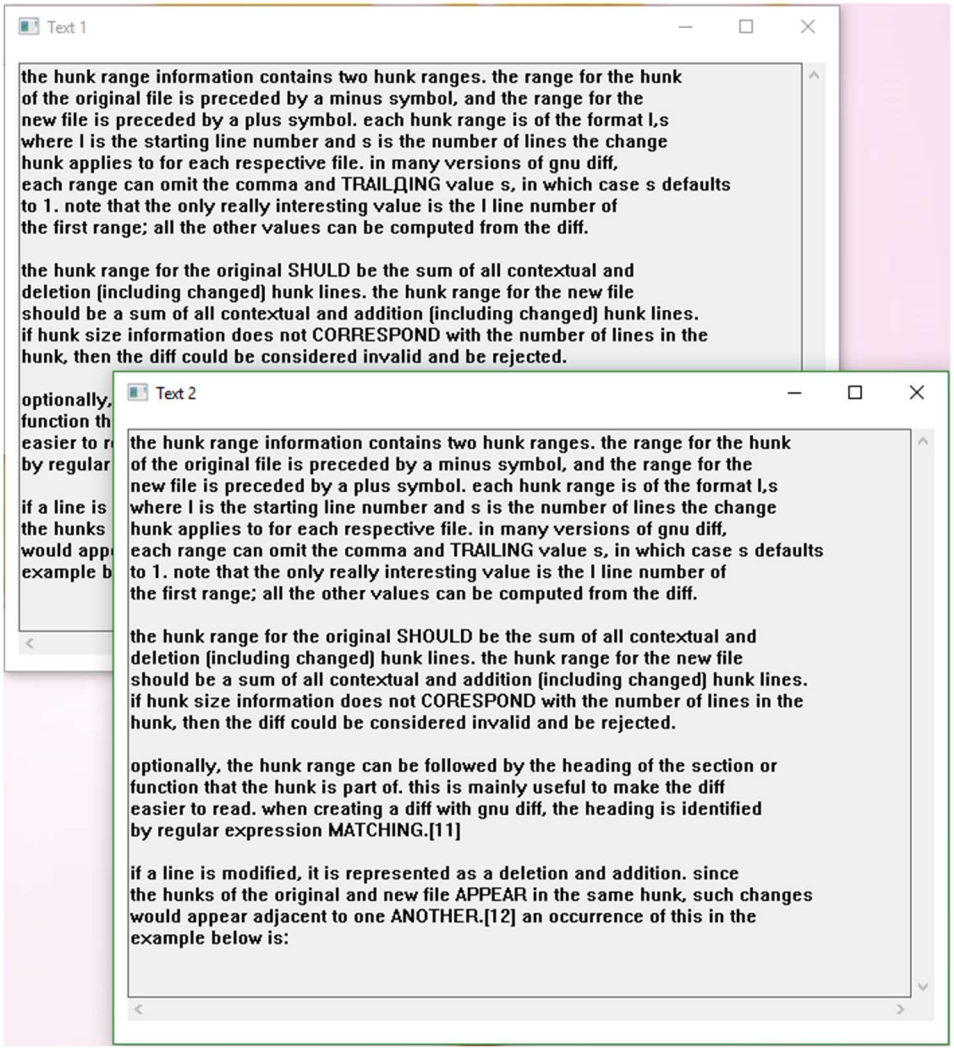

Рисунок 1 – Результат работы функции

Приложение Б
(обязательное)
Листинг библиотеки

Diff.h

```
#pragma once

#include <Windows.h>
#include <bits\stdc++.h>

__declspec(dllimport) std::string
LoadTextFromFile(const std::string&
name);

__declspec(dllimport) int
ansi_tolower(int ch);

__declspec(dllimport) int
ansi_toupper(int ch);

__declspec(dllimport) std::pair<
      std::string,
      std::string
> ShowDifferences(const std::string&
text1, const std::string& text2);
```

Diff.cpp

```
#pragma once

#include "Diff.h"

#define mt std::make_tuple
#define mp std::make_pair

std::string LoadTextFromFile(const
std::string& name) {
std::ifstream file(name);
std::string text;
std::string line;
while (std::getline(file, line)) {
      text += line; text += "\r\n";
}
return text;}

int ansi_tolower(int ch) {
int ruA = 'À';
int ruZ = 'ß';
int rua = 'à';
int enA = 'A';
int enZ = 'Z';
int ena = 'a';
if (ruA <= ch && ch <= ruZ)
      return ch + (rua - ruA);
else if (enA <= ch && ch <= enZ)
      return ch + (ena - enA);
else
      return ch;}

int ansi_toupper(int ch) {
int ruA = 'À';
int rua = 'à';
int ruz = 'ÿ';
int enA = 'A';
int enz = 'z';
int ena = 'a';
if (rua <= ch && ch <= ruz)
      return ch - (rua - ruA);
else if (ena <= ch && ch <= enz)
      return ch - (ena - enA);
else
      return ch;}

namespace internal {
using pii = std::pair < int, int >;
using std::tuple;
using std::get;
using std::tie;

struct word_t {
using hash_t = std::hash<std::string>;
std::string str;
size_t hash;

word_t(const std::string& str, size_t
hash) : str(str), hash(hash) {}
bool operator==(const word_t& rhs) const
{ return hash == rhs.hash; }
bool operator!=(const word_t& rhs) const
{ return hash != rhs.hash; }
word_t& operator=(const word_t& rhs) =
default;
word_t to_upper() const {
std::string upp_str = str;
for (char& c : upp_str)
      c = ansi_toupper(c);
return
word_t{upp_str,hash_t()(upp_str)};}
word_t to_lower() const {
std::string low_str = str;
for (char& c : low_str)
      c = ansi_tolower(c);
return
word_t{low_str,hash_t()(low_str)};}};
std::vector<word_t> SplitLines(const
std::vector<std::string>& lines) {
```

```
std::vector<word_t> words;
std::hash<std::string> hash;
for (const auto& line : lines) {
      std::stringstream stream{ line };
      std::string word;
      while (stream >> word)
      words.emplace_back(word,hash(word));
      words.emplace_back("\r\n",
hash("\r\n"));}
return words;}

std::vector<std::string> GetLines(const
std::string& str) {
std::vector<std::string> lines;
std::stringstream stream{ str };
std::string line;
std::hash<std::string> hash;
while (std::getline(stream, line))
      lines.emplace_back(line);
      return lines;
}
std::string JoinLines(const
std::vector<word_t>& lines) {
std::string text;
for (const auto& word : lines) {
text += word.str;
if (word.str != "\r\n") text += " ";
}
return text;}

class Differences {
private:
      std::map<pii, int>& d;
      std::map<pii, pii>& p;
public:
      Differences(std::map<pii, int>& d,
std::map<pii, pii>& p) : d(d), p(p) {}
tuple<int, int, int>
operator()(
      const std::vector<word_t>& result1,
      const std::vector<word_t>& result2,
      int n1, int n2) {
auto key = mp(n1, n2);
if (n1 < 0 || n2 < 0)
return mt(0, -1, -1);
else if (d.find(key) != d.end()) {
int i1, i2; tie(i1, i2) = p[key];
if (result1[n1] != result2[n2])
      return mt(d[key], i1, i2);
else
      return mt(d[key], n1, n2);
}
if (result1[n1] == result2[n2]) {
int i1, i2, res;
tie(res, i1, i2) = operator()(result1,
result2, n1 - 1, n2 - 1);
res += 1;
d[key] = res;
p[key] = mp(i1, i2);
return mt(res, n1, n2);
} else {
int i1, i2, res;
auto t1 = operator()(result1, result2, n1
- 1, n2);
auto t2 = operator()(result1, result2,
n1, n2 - 1);
if (get<0>(t1) > get<0>(t2))
      tie(res, i1, i2) = t1;
else
      tie(res, i1, i2) = t2;
d[key] = res;
p[key] = mp(i1, i2);
return mt(res, i1, i2); }}};}

std::pair<std::string, std::string>
ShowDifferences(const std::string& text1,
const std::string& text2) {
using namespace internal;
auto result1 =
SplitLines(GetLines(text1));
auto result2 =
SplitLines(GetLines(text2));
std::map<pii, int> d;
std::map<pii, pii> p;
Differences diffs(d, p);

int i1, i2;
std::tie(std::ignore, i1, i2) =
diffs(result1, result2, result1.size() -
1, result2.size() - 1);
result1[i1] = result1[i1].to_lower();
result2[i2] = result2[i2].to_lower();
for (int f = 0, s = 0;;) {
      std::tie(f, s) = p[mp(i1, i2)];
      if (f < 0 || s < 0) break;
      result1[f] = result1[f].to_lower();
      result2[s] = result2[s].to_lower();
      i1 = f, i2 = s;
}
return mp(JoinLines(result1),
JoinLines(result2));
}
#undef mt
#undef mp
```

Неявное связывание

```
#include "Diff.h"

auto text1 = LoadTextFromFile(filePathEdit1->text());
auto text2 = LoadTextFromFile(filePathEdit2->text());

for (auto& c : text1) {
      c = ansi_toupper(c);
}
for (auto& c : text2) {
      c = ansi_toupper(c);
}

if (text1 == text2) {
      MessageBox(
            mainWindow->hwnd(),
            "Тексты совпадают",
            "Информация",
            MB_ICONINFORMATION);
      return;
}

auto diffs = ShowDifferences(text1, text2);
```

Явное связывание

```
std::string (*LoadTextFromFile)(const std::string&);
int (*ansi_tolower)(int);
int (*ansi_toupper)(int);
std::pair<
      std::string,
      std::string
> (*ShowDifferences)(const std::string&, const std::string&);

HINSTANCE dll = LoadLibrary("DiffDLL.dll");

LoadTextFromFile = (std::string(*)(const std::string&)) GetProcAddress(dll,
"LoadTextFromFile");
ansi_tolower = (int(*)(int)) GetProcAddress(dll, "ansi_tolower");
ansi_toupper = (int(*)(int)) GetProcAddress(dll, "ansi_toupper");
ShowDifferences = (std::pair<std::string, std::string>(*)(const std::string&, const
std::string&)) GetProcAddress(dll, "ShowDifferences");
```