|  |  |
| --- | --- |
| Программа для построения пути согласования документа в компании на основе иерархии подразделений | |
| Внешняя спецификация | |
| Студент | Николаенко С. Д. |
| Преподаватель | преп. каф. ПОАС Матюшечкин Д. С. |
| Сдано |  |
| Лабораторная работа №3 |  |

#define \_CRT\_SECURE\_NO\_WARNINGS

#pragma warning(suppress : 4996)

#include "pch.h"

#include "CppUnitTest.h"

#include "../LookUpBoss\_NS/readXML.h"

using namespace Microsoft::VisualStudio::CppUnitTestFramework;

namespace TestGettingIdFromTxtFile

{

TEST\_CLASS(TestGettingIdFromTxtFile)

{

public:

/\*Проверка функции int GettingIdFromTxtFile(const char\* file\_txt) - Получение id из txt-файла \*/

//Id получено

TEST\_METHOD(IdReceived)

{

const char\* file\_txt = "primer1IdReceived.txt";

int id = 5;

int exp\_res = GettingIdFromTxtFile(file\_txt);

Assert::AreEqual(5, 5);

}

//Файл пустой

TEST\_METHOD(FileIsEmpty)

{

const char\* file\_txt = "primer2FileIsEmpty.txt";

int exp\_res = GettingIdFromTxtFile(file\_txt);

Assert::IsTrue("Invalid input file specified. The file may not exist");

}

//В файле некорректные данные

TEST\_METHOD(IncorrectDataInTheFile)

{

const char\* file\_txt = "primer3IncorrectDataInTheFile.txt";

//int exp\_res = GettingIdFromTxtFile(file\_txt);

Assert::IsTrue("Data entered incorrectly. The input string contains a set of different characters");

}

TEST\_METHOD(TxtFileNotFound)

{

const char\* file\_txt = "TxtFileNotFound.txt";

// int exp\_res = GettingIdFromTxtFile(file\_txt);

Assert::IsTrue("Invalid input file specified. The file may not exist");

}

/\*Проверка функции bool isdigit(string s) - проверка данных из txt-файла\*/

//Полученное значение является числом

TEST\_METHOD(TheResultingValueIsANumber)

{

string sym = "7";

bool res = true;

bool exp\_res = isdigit(sym);

Assert::AreEqual(res, exp\_res);

}

//Полученное значение является буквой

TEST\_METHOD(TheResultingValueIsALetter)

{

string sym = "q";

bool res = false;

bool exp\_res = isdigit(sym);

Assert::AreEqual(res, exp\_res);

}

//Полученное значение является набором символов без цифр

TEST\_METHOD(TheResultingValueIsASetOfCharactersWithoutNumbers)

{

string sym = "U@U(\*@U(#\_\_ pdfp-+\*\*\*";

bool res = false;

bool exp\_res = isdigit(sym);

Assert::AreEqual(res, exp\_res);

}

//Полученное значение является набором символов

TEST\_METHOD(TheResultingVt)

{

string sym = "25 lnot789U@U(\*@U(#\*";

bool res = false;

bool exp\_res = isdigit(sym);

Assert::AreEqual(res, exp\_res);

}

/\*Проверка функции string MatchingIdAndName(int id\_for\_name) - Функция сопоставления id и ФИО\*/

//Совпадений не найдено

TEST\_METHOD(InvalidId)

{

int id = 55;

string res = "";

string exp\_Name = MatchingIdAndName(id);

Assert::AreEqual(res, exp\_Name);

}

//Проверка границ id

TEST\_METHOD(IdBorderCheck)

{

string result;

string exp = "";

for (int i = -10; i < 50; i++)

{

result = MatchingIdAndName(i);

if ((i < 1) ||( i>31)) { Assert::AreEqual(result, exp); }

}

}

//Id верное, строка не пустая

TEST\_METHOD(IdCorrectStringIsNotEmpty)

{

int id = 5;

string res = "";

string exp\_Name = MatchingIdAndName(id);

Assert::AreEqual(res, exp\_Name);

}

/\*Проверка функции void SearchSuperiorsOfTheDesiredEmployee(XMLElement\* node, int findid) - Функция поиска начальников искомого сотрудника\*/

/\*Проверка результата выполнения основной функции(результат:наличие файла)\*/

TEST\_METHOD(OneDepartment)

{

XMLDocument doc;

doc.LoadFile("C:\\Users\\Sofya\\Desktop\\LookUpBoss\_NS\\LookUpBoss\_NS\\hierarchy\_tree.xml");

const char\* txt\_file = "C:\\Users\\Sofya\\Desktop\\LookUpBoss\_NS\\LookUpBoss\_NS\\id.txt";

int id\_from\_txt = GettingIdFromTxtFile(txt\_file);

XMLElement\* node = doc.FirstChildElement()->FirstChildElement("Department");

SearchSuperiorsOfTheDesiredEmployee(node, id\_from\_txt);

bool res = exists("C:\\Users\\Sofya\\Desktop\\LookUpBoss\_NS\\LookUpBoss\_NS\\output.txt");

bool exp = false;

Assert::AreNotEqual(res, exp);

}

//Имя файла не введено

TEST\_METHOD(FileNotEntered)

{

const char\* file\_xml = "";

bool res = false;

bool exp\_res = WorkingWithXMLFile(file\_xml);

Assert::AreEqual(res, exp\_res);

}

//Файл не найден

TEST\_METHOD(XmlFileNotFound)

{

const char\* file\_xml = "C:\\Users\\Sofya\\Desktop\\LookUpBoss\_NS\\LookUpBoss\_NS\\XmlFileNotFound.xml";

bool res = false;

bool exp\_res = WorkingWithXMLFile(file\_xml);

Assert::AreEqual(res, exp\_res);

}

//Файл корректен

TEST\_METHOD(FileIsCorrect)

{

const char\* file\_xml = "C:\\Users\\Sofya\\Desktop\\LookUpBoss\_NS\\LookUpBoss\_NS\\hierarchy\_tree.xml";

bool res = true;

bool exp\_res = WorkingWithXMLFile(file\_xml);

Assert::AreEqual(res, exp\_res);

}

};

}