МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное бюджетное учреждение высшего образования
«Чувашский государственный университет И.Н. Ульянова»
Факультет информатики и вычислительной техники
Кафедра вычислительной техники

Системное программирование
Лабораторная работа 3
«Обработка ошибок»

Выполнил:

Студент группы ИВТ-41-20
Галкин Д.С.

Проверил:

Яковлев С.В.

Чебоксары, 2024

Цель работы:

Создать пустое консольное приложение и добавить к нему служебные функции:

1. Добавьте в новый пустой проект служебную функцию **ReportError**, которая выводит сообщение (первый параметр) и завершается с указанным кодом завершения или по return в зависимости от значения второго параметра. Третий параметр определяет, нужно ли отображать системное сообщение об ошибке:

```
VOID ReportError(LPCTSTR UserMessage, DWORD ExitCode, BOOL PrintErrorMsg)
```

2. Проверьте работоспособность функции ReportError с помощью тестовой программы (проекта). Измените текст функций так, чтобы она выдавала сообщение об ошибке на русском языке. Для этого используйте функцию **CharToOem**. Проверьте для случаев, когда имя UNICODE определено и не определено. Возможно, потребуется изменить и функцию PrintStrings.
3. Сравните информацию, выдаваемую функциями **perror** (библиотека C) и **ReportError** для обычных ошибок: 1) открытие несуществующего файла, 2) открытие файла, используемого другой программой на чтение, для записи, 3) открытие файла с атрибутом "только для чтения" на запись, 4) запись в файл, открытый только для чтения, 5) запись в файл, редактируемой другим приложением, 6) чтение из файла, редактируемого другой программой, 7) закрытие файла, используемого другой программой.

Полный текст программы:

error_utils.h

```
//
// Created by Dmitry Galkin on 07.02.2024.
//

#ifndef ERROR_UTILS_H
#define ERROR_UTILS_H

#include <iostream>
#include <string>
#include <cstdlib> // Для использования exit()

const std::string PATH_FILES = "../Lab3/resources/";

// Функция ReportError для вывода сообщений об ошибках и завершения программы
void ReportError(std::string& userMessage, int exitCode, bool printErrorMsg,
std::string rus_error);

void TryOpenFile(const char* filename, const char* mode, std::string rus_error);

void TryOpenFile(const char* filename, const char* mode);

void TryEditFile(const char* filename, const char* mode, std::string rus_error);


#endif //ERROR_UTILS_H
```

Lab3.cpp

```cpp
//
// Created by Dmitry Galkin on 07.02.2024.
//

#include "error_utils.h"


// Функция ReportError для вывода сообщений об ошибках и завершения программы
void ReportError(const std::string& userMessage, int const exitCode, bool const
printErrorMsg, std::string rus_error) {
    std::cerr << userMessage << std::endl; // Вывод пользовательского сообщения об
ошибке
    if (printErrorMsg) {
        // В macOS для вывода системных сообщений об ошибках можно использовать perror
        std::cerr << "Системное сообщение об ошибке: " << rus_error << std::endl <<
std::endl;
    }
    // exit(exitCode); // Завершение программы с указанным кодом завершения
}

void TryOpenFile(const char* filename, const char* mode) {
    std::string full_path = PATH_FILES + filename;
    FILE* file = fopen(full_path.c_str(), mode);
    if (file == nullptr) {
        perror("Ошибка открытия файла через perror");
        ReportError("Ошибка открытия файла через ReportError", 1, true, "rus_error");
    } else {
        std::cout << "Файл успешно открыт: " << filename << std::endl;
        fclose(file);
    }
}


void TryOpenFile(const char* filename, const char* mode, std::string rus_error) {
    std::string full_path = PATH_FILES + filename;
    FILE* file = fopen(full_path.c_str(), mode);
    if (file == nullptr) {
        // perror("Ошибка открытия файла через perror");
        ReportError("Ошибка открытия файла через ReportError", 1, true, rus_error);
    } else {
        // std::cout << "Файл успешно открыт: " << filename << " - " << rus_error <<
std::endl;

        ReportError("Файл успешно открыт через ReportError:", 1, true, rus_error);
        fclose(file);
    }
}


void TryEditFile(const char* filename, const char* mode, std::string rus_error) {
    std::string full_path = PATH_FILES + filename;
    FILE* file = fopen(full_path.c_str(), mode);

    fprintf(file, "Это тестовая строка");
```

```
    perror("Ошибка открытия файла через perror");
    ReportError("Ошибка открытия файла через ReportError", 1, true, rus_error);

    fclose(file);
}
```

main.cpp

```cpp
#include "Lab3/error_utils.h"
#include <iostream>
#include <vector>
#include <sys/termios.h>

using namespace std;

void start_lab3();

int main() {

    start_lab3();
    return 0;
}


void start_lab3() {
    // 1. Открытие несуществующего файла
    // std::cout << "1. Открытие несуществующего файла" << std::endl;
TryOpenFile("empty.txt", "r", "Нет такого файла или каталога");

    // 2. Открытие файла, используемого другой программой на чтение, для записи
    // std::cout << "2. Открытие файла, используемого другой программой на чтение, для записи" << std::endl;    TryOpenFile("test2.txt", "r", "Файл успешно открыт, т.к в Unix системе доступ к файлу, который уже занят, моя программа все равно имеет.");

    // 3. Открытие файла с атрибутом "только для чтения" на запись
    // std::cout << "3. Открытие файла с атрибутом 'только для чтения' на запись" << std::endl;    TryOpenFile("test3.txt", "w", "В разрешении отказано");

    // 4. Запись в файл, открытый только для чтения
    // std::cout << "4. Запись в файл, открытый только для чтения" << std::endl;
TryEditFile("test4.txt", "r", "Неверный файловый дескриптор");

    // 5. Запись в файл, редактируемой другим приложением
    // std::cout << "5. Запись в файл, редактируемой другим приложением" << std::endl;
TryOpenFile("test2.txt", "r", "Файл успешно открыт, т.к в Unix системе доступ к файлу, который уже занят, моя программа все равно имеет.");

    // 6. Чтение из файла, редактируемого другой программой
    // std::cout << "6. Чтение из файла, редактируемого другой программой" << std::endl;    TryOpenFile("test2.txt", "r", "Файл успешно открыт, т.к в Unix системе доступ к файлу, который уже занят, моя программа все равно имеет.");

    // // 7. Закрытие файла, используемого другой программой
    // std::cout << "7. Закрытие файла, используемого другой программой" << std::endl;
TryOpenFile("test2.txt", "r", "Файл успешно закрыт, т.к в Unix системе доступ к файлу, который уже занят, моя программа все равно имеет.");
}
```

Пример работы:

```
Ошибка открытия файла через ReportError
Системное сообщение об ошибке: Нет такого файла или каталога

Файл успешно открыт через ReportError:
Системное сообщение об ошибке: Файл успешно открыт, т.к в Unix системе доступ к файлу, который уже занят, моя программа все равно имеет.

Ошибка открытия файла через ReportError
Системное сообщение об ошибке: В разрешении отказано

Ошибка открытия файла через perror: Bad file descriptor
Ошибка открытия файла через ReportError
Системное сообщение об ошибке: Неверный файловый дескриптор

Файл успешно открыт через ReportError:
Системное сообщение об ошибке: Файл успешно открыт, т.к в Unix системе доступ к файлу, который уже занят, моя программа все равно имеет.

Файл успешно открыт через ReportError:
Системное сообщение об ошибке: Файл успешно открыт, т.к в Unix системе доступ к файлу, который уже занят, моя программа все равно имеет.

Файл успешно открыт через ReportError:
Системное сообщение об ошибке: Файл успешно закрыт, т.к в Unix системе доступ к файлу, который уже занят, моя программа все равно имеет.
```