Выполнили:

студенты группы 19ВВ2

Гусев Д.О.

Кубасов И.М.

Приняли:

Синев М.П.

Дорошенко И.Н.

Пенза 2022

ОТЧЕТ

по лабораторной работе №11

по курсу «Технологии программирования»

на тему «Исключения и пространство имен»

Министерство образования Российской Федерации

Пензенский государственный университет

Кафедра «Вычислительная техника»

### Цель работы

Изучение пространства имен и механизмов обработки исключительных ситуаций.

### Лабораторное задание

Требуется разработать набор классов описания исключительной ситуации. Для этого необходимо реализовать базовый класс **CMyException** и два производных класса (**CMyException1** и **CMyException2**), детализирующих исключительную ситуацию.

Требуется разработать три функции: **Func1**, **Func2** и **Func3**, которые сгенерируют исключения в зависимости от заданного условия (условие определяется вариантом задания).

Функция **Func1** должна открывать файл с указанным именем и возвращать файловый идентификатор. Функция может генерировать два исключения (**CMyException** и **CMyException1**). Исключения генерируются при проверке имени файла.

Функция **Func2** должна читать содержимое файла в буфер. Функция может генерировать одно исключение (**CMyException2**). Исключение генерируется при проверке прочитанных данных.

Функция **Func3** должна закрывать файловый идентификатор. Функция может генерировать одно исключение (**CMyException**). Исключение генерируется при проверке файлового указателя.

Требуется разработать функцию **FuncHandle**, реализующую обработку исключительных ситуаций. В данной функции внутри блока **try** должны поочередно вызываться функции **Func1**, **Func2** и **Func3**. В функции должны быть реализованы несколько блоков **catch**, для каждого возможного типа исключения. Внутри блока необходимо **catch** вывести информацию по ошибке (функция, описание и т.д.).

В основной функции необходимо несколько раз вызвать функцию **FuncHandle** задав такие условия, чтобы поочередно были сгенерированы все возможные исключения. В том числе должен быть такой вызов **FuncHandle**, который не генерирует исключения.

Требования к классам CMyException:

* Классы должны содержать сведения о функции, в которой сгенерировано исключение (текстовое имя).
* Классы должны содержать общее описание ошибки (текстовая строка).
* Производные классы должны содержать дополнительные индивидуальные данные по ошибке.

Все классы и функции (кроме функции **main**) должны быть реализованы внутри собственного пространства имен.



### Листинг

Source.cpp:

#include <iostream>

#include "MyException.h"

#include <conio.h>

#include <locale.h>

#define SIZE\_OF\_BUFFER 2056

#define OFFSET 0x400

#define BYTE\_READ\_BEGIN 0x10

#define BYTE\_READ\_END 0x20

namespace Func1 {

FILE\* Func1(const char fileName[], FILE\*& file) {

char temp1[8];

char temp2[32];

const char\* ind = strrchr(fileName, int('.'));

int index = ind - fileName;

int i = 0;

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

temp2[i] = fileName[i];

}

temp2[index] = 0;

while (fileName[index] != '\0') {

temp1[i] = fileName[index];

i++;

index++;

}

temp1[i] = 0;

fopen\_s(&file, fileName, "r");

//Проверка на валидность расширения

if (!strcmp(temp1, ".xml")) throw CMyException::CMyException("Func1");

//Проверка на валидность имени файла

if (!strcmp(temp2, "name")) throw CMyException1::CMyException1("Func1");

return file;

}

}

namespace Func2 {

char\* Func2(char\* buffer, FILE\*& file) {

char tempbuff[256];

char\* pBuff;

while (1) {

pBuff = fgets(tempbuff, sizeof(tempbuff), file);

if (pBuff == NULL) {

if (feof(file) != 0) {

cout << "Чтение файла завершено" << endl;

break;

}

else {

cout << "Ошибка чтения из файла" << endl;

break;

}

}

strcat\_s(buffer, SIZE\_OF\_BUFFER, tempbuff);

if ((ftell(file) > BYTE\_READ\_BEGIN) and (ftell(file) < BYTE\_READ\_END)) throw CMyException2::CMyException2("Func2");

}

return buffer;

}

}

namespace Func3 {

void Func3(FILE\*& file) {

long long offset;

fseek(file, OFFSET, SEEK\_SET);

offset = ftell(file);

if (offset > OFFSET) throw CMyException::CMyException("Func3");

fclose(file);

}

}

namespace FuncHandle {

void FuncHandle(const char fileName[], char\* buffer, FILE\* file) {

try

{

Func1::Func1(fileName, file);

}

catch (CMyException1::CMyException1& ex)

{

cout << ex.nameOfFunction << endl;

cout << ex.additionalData << endl;

}

catch (CMyException::CMyException& ex)

{

cout << ex.nameOfFunction << endl;

cout << ex.additionalData1 << endl;

}

try

{

Func2::Func2(buffer, file);

}

catch (CMyException2::CMyException2& ex)

{

cout << ex.nameOfFunction << endl;

cout << ex.additionalData << endl;

}

try

{

Func3::Func3(file);

}

catch (CMyException::CMyException& ex)

{

cout << ex.nameOfFunction << endl;

cout << ex.additionalData2 << endl;

}

}

}

void main(void) {

setlocale(LC\_ALL, "rus");

const char fileName[] = "file.txt";

FILE\* file = nullptr;

char\* buffer = (char\*)malloc(SIZE\_OF\_BUFFER);

buffer[0] = 0;

FuncHandle::FuncHandle(fileName, buffer, file);

cout << buffer;

free(buffer);

\_getch();

}

MyException:

#pragma once

#include <string>

using namespace std;

namespace CMyException {

class CMyException {

public:

CMyException(string nameOfFunction) : nameOfFunction(nameOfFunction) {}

CMyException() : nameOfFunction("Неизвестная функция") {}

const string additionalData1 = "Расширение файла - «xml»!";

const string additionalData2 = "Указатель располагается дальше смещения 0x400";

const string nameOfFunction;

};

}

namespace CMyException1 {

class CMyException1 : public CMyException::CMyException {

public:

CMyException1(string nameOfFunction) : nameOfFunction(nameOfFunction) {}

const string additionalData = "Имя файла - «name»!";

string nameOfFunction;

};

}

namespace CMyException2 {

class CMyException2 : public CMyException::CMyException {

public:

CMyException2(string nameOfFunction) : nameOfFunction(nameOfFunction) {}

const string additionalData = "Чтение по смещению в файле в диапазоне 0x10 – 0x20";

const string nameOfFunction;

};

}

### Результаты работы программы



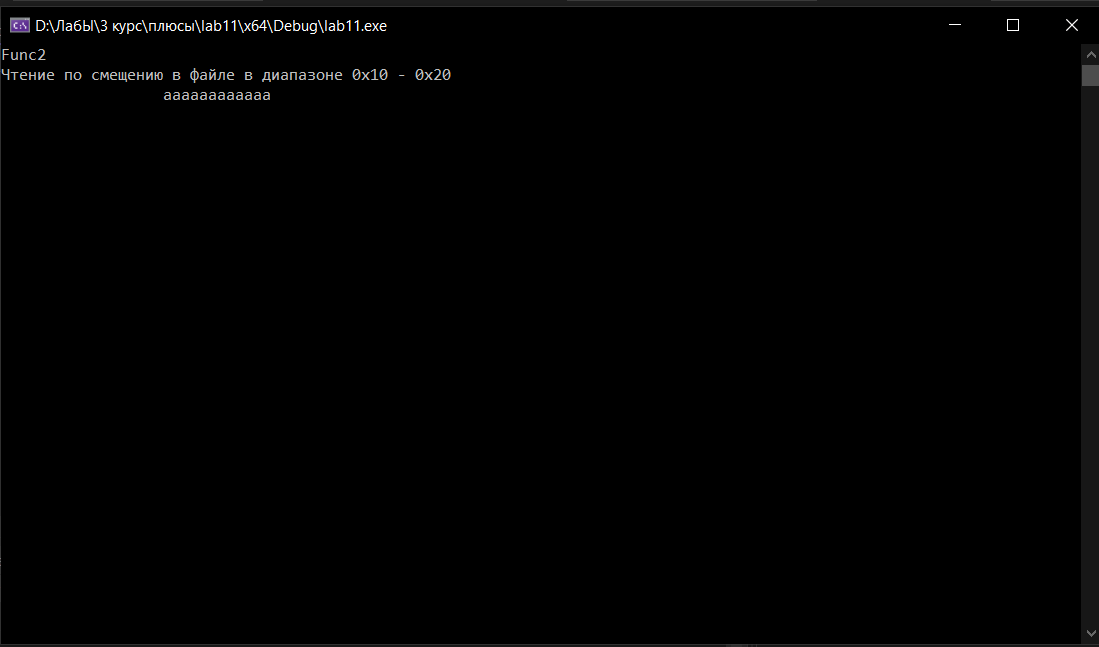


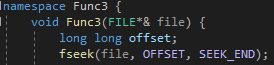


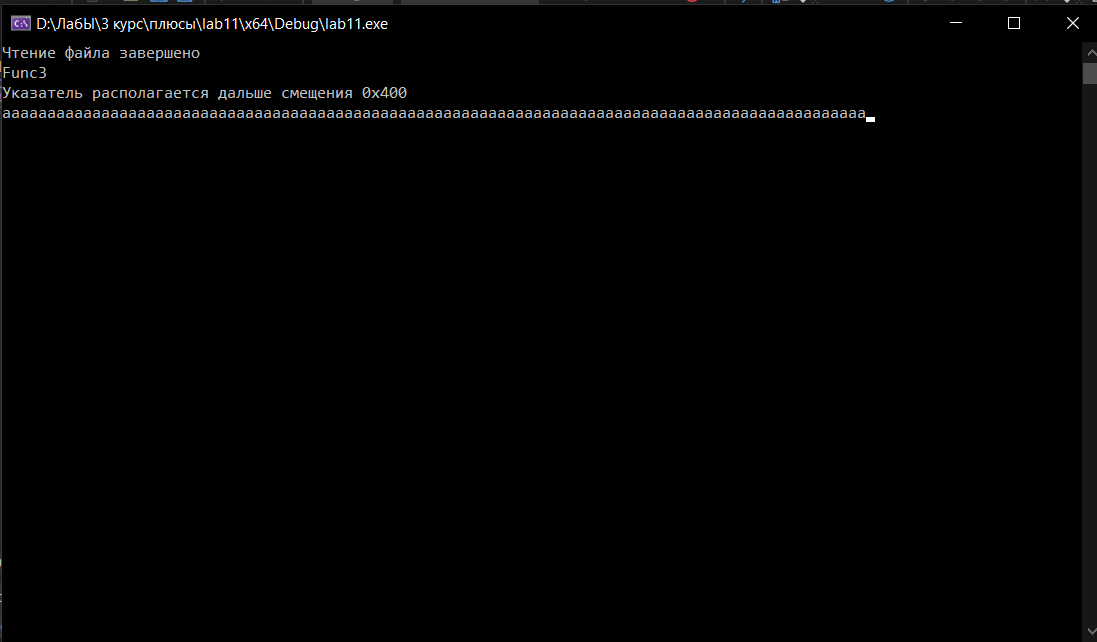












Работы программы без исключений:



### Вывод

Мы изучили пространства имен и механизмов обработки исключительных ситуаций.