Пенза 2020

Министерство образования Российской Федерации

Пензенский государственный университет

Кафедра «Вычислительная техника»

Выполнил:

студенты группы 17ВВ1

Кокин Денис

Беленков Никита

Принял:

Дорошенко И.Н.

**ОТЧЁТ**

по лабораторной работе №11

по курсу «Технологии программирования»

на тему «Исключения и пространство имен»

**Цель работы:** 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 программы**

#include <windows.h>

#include <string>

#include <iostream>

#include <exception>

#define FIRST\_COMPARABLE\_VALUE 0x200

#define SECOND\_COMPARABLE\_VALUE 0x400

const int BUFFER\_SIZE = 0x1000;

char mFileBuffer[BUFFER\_SIZE];

DWORD mNumberOfBytesRead;

namespace CMyExceptionSpace {

class CMyException : public std::exception {

protected:

std::string mEMessage;

std::string mEDesciption;

public:

CMyException() {}

CMyException(std::string pEMessage, std::string pEDesciption) {

mEMessage = pEMessage;

mEDesciption = pEDesciption;

}

virtual const char\* what() const throw()

{

std::cout << "Message: " << this->mEMessage << std::endl;

std::cout << "Description: " << this->mEDesciption << std::endl;

return NULL;

}

};

}

namespace CMyException2Space {

class CMyException2 : public CMyExceptionSpace::CMyException{

protected:

std::string mEEXDesciption;

public:

CMyException2(std::string pEMessage, std::string pEDesciption, std::string pEEXDesciption) {

mEMessage = pEMessage;

mEDesciption = pEDesciption;

mEEXDesciption = pEEXDesciption;

}

virtual const char\* what() const throw()

{

CMyExceptionSpace::CMyException::what();

std::cout << "Extra Description: " << this->mEEXDesciption << std::endl;

return NULL;

}

};

}

namespace CMyException3Space {

class CMyException3 : public CMyExceptionSpace::CMyException {

protected:

std::string mEEXDesciption;

public:

CMyException3(std::string pEMessage, std::string pEDesciption, std::string pEEXDesciption) {

mEMessage = pEMessage;

mEDesciption = pEDesciption;

mEEXDesciption = pEEXDesciption;

}

virtual const char\* what() const throw()

{

CMyExceptionSpace::CMyException::what();

std::cout << "Extra Description: " << this->mEEXDesciption << std::endl;

return NULL;

}

};

}

namespace Func1Space {

HANDLE Func1(std::string pFilePath) {

if (pFilePath.substr(0, pFilePath.find\_last\_of(".")) == "source")

throw CMyExceptionSpace::CMyException(

"Program was interrupted by Func1",

"Cause's file name contains forbiden substrign"

);

if(pFilePath.substr(pFilePath.find\_last\_of(".") + 1) == "h")

throw CMyException2Space::CMyException2(

"Program was interrupted by Func1",

"Cause's file name contains forbiden substrign",

"Unhadled file's extention"

);

HANDLE mHFile;

mHFile = CreateFile((std::wstring(pFilePath.begin(), pFilePath.end())).c\_str(),

GENERIC\_READ,

FILE\_SHARE\_READ,

NULL,

OPEN\_EXISTING,

FILE\_ATTRIBUTE\_NORMAL,

NULL);

return mHFile;

}

}

namespace Func2Space {

void Func2(HANDLE pHFile) {

if (ReadFile(pHFile, &::mFileBuffer, BUFFER\_SIZE, &mNumberOfBytesRead, NULL)

&& (mNumberOfBytesRead > FIRST\_COMPARABLE\_VALUE)) {

throw CMyException3Space::CMyException3(

"Program was interrupted by Func2",

"Read file exception" ,

"A count of read bytes greater than " + FIRST\_COMPARABLE\_VALUE

);

}

}

}

namespace Func3Space {

void Func3(HANDLE pHFile) {

if (pHFile < (HANDLE)SECOND\_COMPARABLE\_VALUE) {

CloseHandle(pHFile);

throw CMyExceptionSpace::CMyException(

"Program was interrupted by Func2",

"File pointer exception"

);

}

CloseHandle(pHFile);

}

}

namespace FuncHadlerSpace {

void FuncHandler(std::string pFilePath) {

try {

HANDLE mHFile = Func1Space::Func1(pFilePath);

Func2Space::Func2(mHFile);

Func3Space::Func3(mHFile);

}

catch (CMyExceptionSpace::CMyException & Exception){

Exception.what();

}

catch (CMyException2Space::CMyException2 & Exception) {

Exception.what();

}

catch (CMyException3Space::CMyException3 & Exception) {

Exception.what();

}

}

}

void main(){

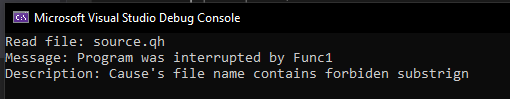
std::string mFilePath = "qsource.qh";

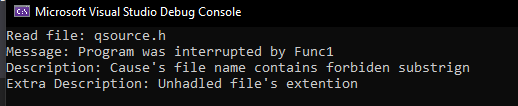
FuncHadlerSpace::FuncHandler(mFilePath);

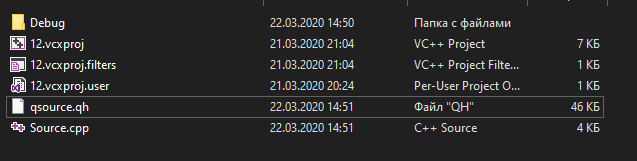
std::cin;

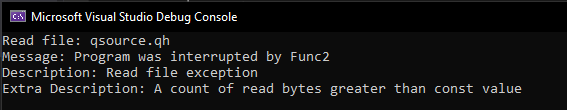
}

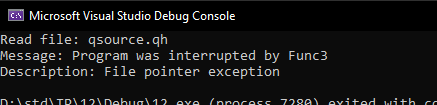
**Результаты работы программы**

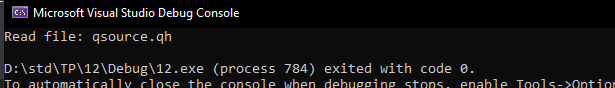












**Вывод:** студенты изучили механизмы сохранения и восстановления объектов данных. Получили навыки разработки самовосстанавливающихся структур данных.