

**Федеральное государственное бюджетное образовательное учреждение
высшего профессионального образования
"Уфимский государственный авиационный технический университет"**

Кафедра Высокопроизводительных вычислительных технологий и систем

Дисциплина: Программирование

Отчет по лабораторной работе № 9

Тема: «Обработка исключительных ситуаций»

| | | | | |
|----------------|-------------------|---------|------|--------|
| Группа МКН-113 | Фамилия И.О. | Подпись | Дата | Оценка |
| Студент | Шамаев И.Р. | | | |
| Принял | Гайнетдинова А.А. | | | |

Уфа 2020

Цель: изучить средства языка C++ для обработки исключительных ситуаций.

Ответы на контрольные вопросы

- 1. Как создается защищённый блок кода? Как описывается процедура обработки конкретного исключения? .**

```
try{  
  
<Код, который может вызвать исключительную ситуацию>  
  
}  
  
catch (индикатор){  
  
<Обработка исключения>  
  
}
```

Инструкция, генерирующая исключение, должна исполняться внутри блока try. Всякое исключение должно быть перехвачено инструкцией catch, которая непосредственно следует за инструкцией try, сгенерировавшей исключение.

Чтобы исключение, сгенерированное в одном блоке кода, могло найти нужный обработчик, находящийся в другом блоке, при генерации исключения выбрасывается индикатор исключения. Индикатором служит объект или переменная некоторого конкретного типа. При

возникновении исключения будет выбран тот обработчик, в описании которого указан тот же тип ожидаемого индикатора. Обработчики различают исключения по типам данных индикатора, и поэтому в наиболее распространённом случае в качестве индикатора

указывают объект некоторого класса, специально предусмотренного для этой цели.

- 2. Как генерируется исключение?** Для генерации исключения используется оператор throw после него указывается индикатор.
- 3. Какие стандартные классы исключений вы знаете?** exception, runtime_error, invalid_argument, bad_alloc, range_error, logic_error, out_of_range

4. **Какой класс является базовым для всех стандартных классов исключений?** `exception`
5. **Как можно ограничить список исключений, которые могут генерироваться в функции?** Для того, чтобы ограничить список исключений, которые могут генерироваться в функции, необходимо добавить к определению функции ключевое слово `throw` (список типов).

Индивидуальное задание №1

Задание:

1. Реализовать функцию:

$$y(x, n) = \sqrt[n]{\lg(x)}.$$

При задании некорректных параметров функции, а именно:

- вычисление корня четного порядка от отрицательного числа,
- недопустимое значение аргумента логарифма,

выполнить генерацию соответствующего исключения в виде текстовой строки или целочисленного кода ошибки.

Написать приложение, использующее в вычислениях данную функцию. Предусмотреть обработку всех возможных исключений в виде вывода текста соответствующей ошибки.

Исходный код программы:

Main:

```
#include <iostream>
#include <iostream>
#include <cmath>

using namespace std;

float Foo(float v_n, float v_x)
{
    float y;
    int a = static_cast<int>(v_n);
    if (v_x <= 0)
    {
        throw "Ошибка! X не может быть меньше 0!";
    }
    if ((a % 2 == 0) && (log10(v_x) < 0))
    {
        throw "Ошибка! Вы пытаетесь вычислить корень четного порядка от отрицательного числа! ";
    }
    y = pow(log10(v_x), (1 / v_n));
    return y;
}

int main()
{
    setlocale(LC_ALL, "Russian");
    float s, x;
    float n;
    try
    {
        cout << "Введите n и x" << endl;
        cin >> n >> x;
        s = Foo(n, x);
        cout << s << endl;
    }
    catch (const char *ex)
    {
        cout << ex;
    }

    return 0;
}
```

```
}
```

Пример выполнения программы:

```
Введите п и х
3
2
0.670198

C:\Users\MSI\source\repos\ConsoleApplication37\Debug\ConsoleApplication37.exe (процесс 8708) завершил работу с кодом 0.
Чтобы автоматически закрывать консоль при остановке отладки, включите параметр "Сервис" ->"Параметры" ->"Отладка" -> "Автоматически закрыть консоль при остановке отладки".
Нажмите любую клавишу, чтобы закрыть это окно...
```

```
Введите п и х
4
0.5
Ошибка! Вы пытаетесь вычислить корень четного порядка от отрицательного числа!
C:\Users\MSI\source\repos\ConsoleApplication37\Debug\ConsoleApplication37.exe (процесс 14212) завершил работу с кодом 0.
Чтобы автоматически закрывать консоль при остановке отладки, включите параметр "Сервис" ->"Параметры" ->"Отладка" -> "Автоматически закрыть консоль при остановке отладки".
Нажмите любую клавишу, чтобы закрыть это окно...
■
```

```
Введите п и х
3
-2
Ошибка! X не может быть меньше 0!
C:\Users\MSI\source\repos\ConsoleApplication37\Debug\ConsoleApplication37.exe (процесс 12340) завершил работу с кодом 0.
Чтобы автоматически закрывать консоль при остановке отладки, включите параметр "Сервис" ->"Параметры" ->"Отладка" -> "Автоматически закрыть консоль при остановке отладки".
Нажмите любую клавишу, чтобы закрыть это окно...
```

Индивидуальное задание №2

2. Доработать классы, написанные в лабораторной работе 2 «[Классы и наследование](#)», следующим образом:

- добавить функцию вывода информации в заданный пользователем файл,
- добавить в методы классов (там, где это необходимо) генерацию исключений, описанных ниже.

Найти в стандартной библиотеке классы-исключения или создать исключение самостоятельно (производное от стандартного исключения) для следующих исключительных ситуаций:

- ошибка открытия файла – необходимо с помощью класса исключения передать в обработчик имя файла,
- некорректное задание параметров (в конструкторах и set-методах) – необходимо с помощью класса исключения передать в обработчик имя параметра (поля), на котором возникла ошибка, и некорректное значение.

Создать демонстрационное приложение и продемонстрировать корректную работу всех методов, а также обработку всех исключительных ситуаций.

Исходный код программы

main:

```
#define _CRT_SECURE_NO_WARNINGS

#include <iostream>
#include <string>
#include <cstdio>
#include <fstream>
#include <iomanip>
#include <sstream>

using namespace std;

class File {
protected:
    char* Name; // Имя
    char* Expansion; // Расширение
    int Size; // Размер
    float k = 0.3;

public:
    File(); // Пустой конструктор
    File(const char* NName, const char* NExpansion, int NSize);
    virtual ~File(); // Деструктор
    int compression(); // Функция сжатия

    const char* Get_Name() const;
    void Set_Name(const char* NName);

    const char* Get_Expansion() const;
    void Set_Expansion(const char* NExpansion);

    int Get_Size() const;
```

```

        void Set_Size(int NSize);
        virtual void OutputFile(string file_name);
        void Print_compression();
        //void Test();

};

class Sound_File : public File
{
protected:
    int Biterate;
public:
    Sound_File();
    Sound_File(const char* NName, const char* NExpansion, int Size, int
NBiterate);

    int Get_Biterate() const;
    void Set_Biterate(int NBiterate);
    void OutputFile(std::string file_name);
};

class Myexception :public std::exception
{
public:
    Myexception(const char* msg, const std::string str);

    // Возвращает неправильное значение
    const std::string Get_Name() const;
    //const std::string Get_Expansion() const;

protected:
    std::string f_str;
};

class InputNumError : public std::exception
{
public:
    InputNumError(const char* msg, int value);

    // Возвращает неправильное значение
    int getVal() const;

protected:
    int f_value;
};

const char* Name;
const char* Expansion;
int Size;
File::File()
{
    Name = new char[strlen("Unnamed") + 1];
    strcpy(Name, "Unnamed");
    Expansion = new char[strlen("Unnamed") + 1];
    strcpy(Expansion, "Unnamed");
    Size = 569;
} //Пустой Конструктор
File::File(const char* NName, const char* NExpansion, int NSize)
{
    if (!(strlen(NName) <= 20 && strlen(NName) >= 3))
    {

```

```

        throw Myexception("Name", NName);
    }
    if (!(strlen(NExpansion) <= 4 && strlen(NExpansion) >= 2))
    {
        throw Myexception("Expansion", NExpansion);
    }
    if (!(NSize < 10 && NSize>1000))
    {
        throw InputNumError("Size", NSize);
    }
    Name = new char[strlen(NName) + 1];
    strcpy(Name, NName);
    Expansion = new char[strlen(NExpansion) + 1];
    strcpy(Expansion, NExpansion);
    Size = NSize;
} //заполненный конструктор

void File::Set_Name(const char* NName) {
    if (!(strlen(NName) <= 20 && strlen(NName) >= 3))
    {
        throw Myexception("Name", NName);
    }
    delete[] Name; // удаление старых данных
    Name = new char[strlen(NName) + 1];
    strcpy(Name, NName);
}

void File::Set_Expansion(const char* NExpansion) {
    if (!(strlen(NExpansion) <= 4 && strlen(NExpansion) >= 2))
    {
        throw Myexception("Expansion", NExpansion);
    }
    delete[] Expansion;
    Expansion = new char[strlen(NExpansion) + 1];
    strcpy(Expansion, NExpansion);
}

void File::Set_Size(int NSize) {
    if (!(NSize < 10 && NSize>1000))
    {
        throw InputNumError("Size", NSize);
    }
    Size = NSize;
}

const char* File::Get_Name() const
{
    return Name;
}

const char* File::Get_Expansion() const
{
    return Expansion;
}

int File::Get_Size() const
{
    return Size;
}

void File::OutputFile(string file_name)
{
    ofstream file(file_name, ofstream::out);
    if (!file.is_open())
    {

```



```

        throw std::runtime_error(file_name);;
    }
    file << "Имя файла : " << Name << "\nРасширение файла : " << Expansion << "\nРазмер : " << Size << endl;
    file.close();
}
void Sound_File::OutputFile(string file_name)
{
    ofstream file(file_name, ofstream::out);
    if (!file.is_open())
    {
        throw std::runtime_error(file_name);;
    }
    file << "Имя файла : " << Name << "\nРасширение файла : " << Expansion << "\nРазмер : " << Size << "\nБитрейт : " << Biterate << endl;
    file.close();
}
/*
void File::Print_compression()
{
    int R = 0;
    R = Size * k;
    ofstream file("Output.txt", ofstream::app);
    file << "Размер файла с заданным коэффициентом сжатия : " << k << " составляет : " << R << "\n" << endl;
}
*/
Sound_File::Sound_File() : File()
{
    Biterate = 88;
}
Sound_File::Sound_File(const char* NName, const char* NExpansion, int NSize, int NBiterate)
{
    if (!(strlen(NName) <= 20 && strlen(NName) >= 3))
    {
        throw Myexception("Name", NName);
    }
    if (!(strlen(NExpansion) > 1 && strlen(NExpansion) < 5))
    {
        throw Myexception("Expansion", NExpansion);
    }
    if (!(NSize > 10 && NSize < 1000))
    {
        throw InputNumError("Size", NSize);
    }
    if (!(NBiterate < 1000 && NBiterate > 9))
    {
        throw InputNumError("Biterate", NBiterate);
    }
    Name = new char[strlen(NName) + 1];
    strcpy(Name, NName);
    Expansion = new char[strlen(NExpansion) + 1];
    strcpy(Expansion, NExpansion);
    Size = NSize;
    Biterate = NBiterate;
}
void Sound_File::Set_Biterate(int NBiterate)
{
    if (!(NBiterate < 1000 && NBiterate > 9))
    {
        throw InputNumError("Biterate", NBiterate);
    }
    Biterate = NBiterate;
}

```

```

int Sound_File::Get_Biterate() const
{
    return Biterate;
}
/*
void Sound_File::OutputFile(string file_name)
{
    File::OutputFile(file_name);
    ofstream file(file_name, ofstream::app);
    file << "Битрейт : " << Biterate << "\n" << endl;
}
*/

File::~~File()
{
    delete[] Name;
    delete[] Expansion;
}

Myexception::Myexception(const char* msg, std::string str) : std::exception(msg)
{
    f_str = str;
}

InputNumError::InputNumError(const char* msg, int value) : std::exception(msg)
{
    f_value = value;
}

const std::string Myexception::Get_Name() const
{
    return f_str;
}

int InputNumError::getVal() const
{
    return f_value;
}

int main()
{
    setlocale(LC_ALL, "Russian");
    string fileName;
    Sound_File Music("Imagine", "mp3", 700, 800);

    try
    {
        File Word("Лабораторная работа", "docx", 0);
        //Word.OutputFile("Output.txt");
        //Word.Test();
        //Word.Print_compression();
    }
    catch (const Myexception & ex)
    {
        cout << "Ошибка! Некорректно введено название " << ex.what() << " " <<
ex.Get_Name() << endl;
    }
    catch (const InputNumError & ex)

```

```

    {
        cout << "Ошибка! Некорректно введено значение " << ex.what() << " " <<
ex.getVal() << endl;
    }

    try
    {
        Music.Set_Name("Ff");
    }
    catch (const Myexception & ex)
    {
        cout << "Ошибка! Некорректно введено название " << ex.what() << " " <<
ex.Get_Name() << endl;
    }
    fileName = "Output.txt";


    try
    {
        Music.OutputFile(fileName);
    }
    catch (const runtime_error & ex)
    {
        cout << ex.what() << " Невозможно открыть" << endl;
    }
    /*
    puts("\n");
    Sound_File Music("Imagine Dragons-believer", "mp3", 700, 800);
    Music.OutputFile();
    */
    return 0;
}

```

Пример выполнения программы:

```
Ошибка! Некорректно введено значение Size 0
Ошибка! Некорректно введено название Name Ff

C:\Users\MSI\source\repos\ConsoleApplication37\Debug\ConsoleApplication37.exe (процесс 9280) завершил работу с кодом 0.
Чтобы автоматически закрывать консоль при остановке отладки, включите параметр "Сервис" ->"Параметры" ->"Отладка" -> "А
томатически закрыть консоль при остановке отладки".
Нажмите любую клавишу, чтобы закрыть это окно...
```

 Output – Блокнот

Файл Правка Формат Вид Справка

Имя файла : Imagine

Расширение файла : mp3

Размер : 700

Битрейт : 800

Вывод: изучил на практике средства языка C++ для обработки исключительных ситуаций.