МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

**Липецкий государственный технический университет**

Факультет автоматизации и информатики

Кафедра автоматизированных систем управления

Лабораторная работа

по программированию №8

“Обработка исключений”

Студент \_\_\_\_\_\_\_\_\_\_\_\_ Станиславчук С. М.

(подпись, дата)

Группа АС-21-1

Руководитель \_\_\_\_\_\_\_\_\_\_\_\_ Ведищев В. В.

(подпись, дата)

Липецк 2022 г.

Содержание

2. Задание

3. Цель

4. Код программы

5. Пример выполнения

6. Вывод

2. Задание:

Для класса, полученного в результате лабораторной работы №2, перегрузить ещё как минимум три операции. Создать собственную иерархию класса обработки исключений. В ситуациях, при которых может возникать ошибка, генерировать собственное исключение и его обрабатывать. В программе должны быть использованы блоки try, catch. В лабораторной работе описать все обрабатываемые в программе исключительные ситуации.

3. Цель работы:

Научиться создавать и использовать собственные события для возможности отслеживания другими классами наступления отдельных условий в текущем классе.

4. Код программы:

#include <iostream>

using namespace std;

class Invalid : public exception {

public:

Invalid(const char\* message) : exception(message) {}

};

class Zero : public exception {

public:

Zero(const char\* message) : exception(message) {}

};

class Scanner {

public:

void get(int& x) {

while (1) {

try

{

cin >> x;

if (cin.fail())

throw Invalid("Wrong value of !");

if (x == 0)

throw Zero("Value == 0!");

break;

}

catch (Zero& ex)

{

printf("%s", ex.what());

}

catch (Invalid& ex) {

cin.clear();

cin.ignore(INT\_MAX, '\n');

printf("%s", ex.what());

}

}

}

};

class Matrix

{

private:

size\_t rows\_, cols\_;

int\*\* arr;

void copy(Matrix& M)

{

size\_t t = rows\_; rows\_ = M.rows\_; M.rows\_ = t;

t = cols\_; cols\_ = M.cols\_; M.cols\_ = t;

int\*\* p = arr; arr = M.arr; M.arr = p;

}

public:

Matrix(const size\_t rows = 0, const size\_t cols = 0)

{

rows\_ = rows; cols\_ = cols; arr = nullptr;

if (rows\_ && cols\_)

{

arr = new int\* [rows\_];

for (size\_t i = 0; i < rows\_; i++)

arr[i] = new int[cols\_];

}

}

Matrix(const Matrix& M)

{

rows\_ = M.rows\_; cols\_ = M.cols\_; arr = nullptr;

if (rows\_ && cols\_)

{

arr = new int\* [rows\_];

for (size\_t i = 0; i < rows\_; i++)

{

arr[i] = new int[cols\_];

for (size\_t j = 0; j < cols\_; ++j)

arr[i][j] = M.arr[i][j];

}

}

}

Matrix& operator = (const Matrix& M)

{

Matrix tmp(M);

copy(tmp);

return \*this;

}

~Matrix()

{

if (arr != nullptr)

{

for (size\_t i = 0; i < rows\_; i++)

delete arr[i];

delete[] arr;

}

}

size\_t rows() const

{

return rows\_;

};

size\_t cols() const

{

return cols\_;

};

int\* operator[](size\_t i) { return arr[i]; }

const int\* operator[](size\_t i) const { return arr[i]; }

};

void Print(Matrix& M) {

for (size\_t i = 0; i < M.rows(); i++)

{

for (size\_t j = 0; j < M.cols(); j++)

cout << M[i][j] << ' ';

cout << endl;

}

}

void Input(Matrix& M) {

for (size\_t i = 0; i < M.rows(); i++)

{

for (size\_t j = 0; j < M.cols(); j++)

cin >> M[i][j];

}

}

Matrix operator + (const Matrix& M, int m)

{

Matrix temp(M.rows(), M.cols());

for (size\_t i = 0; i < M.rows(); i++)

{

for (size\_t j = 0; j < M.cols(); j++)

{

temp[i][j] = M[i][j] + m;

}

}

return temp;

}

Matrix operator - (const Matrix& M, int m)

{

Matrix temp(M.rows(), M.cols());

for (size\_t i = 0; i < M.rows(); i++)

{

for (size\_t j = 0; j < M.cols(); j++)

{

temp[i][j] = M[i][j] - m;

}

}

return temp;

}

Matrix operator \* (const Matrix& M, int m)

{

Matrix temp(M.rows(), M.cols());

for (size\_t i = 0; i < M.rows(); i++)

{

for (size\_t j = 0; j < M.cols(); j++)

{

temp[i][j] = M[i][j] \* m;

}

}

return temp;

}

Matrix operator / (const Matrix& M, int m)

{

Matrix temp(M.rows(), M.cols());

for (size\_t i = 0; i < M.rows(); i++)

{

for (size\_t j = 0; j < M.cols(); j++)

{

temp[i][j] = M[i][j] / m;

}

}

return temp;

}

int main()

{

Scanner\* scan = new Scanner;

int a, b;

int m;

cout << "Enter size of matrix A(a, b): ";

scan->get(a);

scan->get(b);

cout << "Enter m-value: ";

scan->get(m);

Matrix A(a, b), B;

cout << "A = \n";

Input(A);

cout << "\nA = \n";

Print(A);

cout << "\nA + m = B\n";

B = A + m;

Print(B);

cout << endl;

cout << "A - m = B\n";

B = A - m;

Print(B);

cout << endl;

cout << "\nA \* m = B\n";

B = A \* m;

Print(B);

cout << endl;

cout << "\nA / m = B\n";

B = A / m;

Print(B);

cout << endl;

return 0;

}

Вывод: научился использовать встроенное в язык средство обработки исключительных ситуаций для предотвращения аварийного завершения работы программы в случае ошибок в данных.