Лабараторная работа 5

Клишевич Вадим, M3105

main.cpp

#include <bits/stdc++.h>  
#include "functions.h"  
#include "classes.h"  
  
using namespace std;  
  
int main() {  
 int a = 3;  
 double b = 1.2;  
 cout << pow\_(a, 3) << " : ";  
 cout << pow\_(b, 3) << " : ";  
 cout << pow\_(3.2, 2) << "\n";  
  
 Matrix<int> x(3, 2);  
 try {  
 cout << x.get(1, 1) << "\n";  
 } catch (const exception& ex) {  
 cout << "Exception caught: " << ex.what() << endl;  
 }  
  
 try {  
 cout << x.get(3, 3) << "\n";  
 } catch (const exception& ex) {  
 cout << "Exception caught: " << ex.what() << endl;  
 }  
}

functions.h

//  
// Created by vadim on 7.02.20.  
//  
  
#pragma once  
  
template <typename T>  
T pow\_(const T&, int);  
  
#include "functions.tpp"

functions.tpp

//  
// Created by vadim on 7.02.20.  
//  
  
template <typename T>  
T pow\_(const T& x, const int n) {  
 T a = x;  
 for (int i = 1; i < n; i++) {  
 a \*= x;  
 }  
 return a;  
}

classes.h

//  
// Created by vadim on 7.02.20.  
//  
#include <bits/stdc++.h>  
  
using namespace std;  
  
#pragma once  
  
template <typename T>  
class Matrix {  
private:  
 int n, m;  
 vector<vector<T>> x;  
  
public:  
 Matrix();  
 Matrix(int, int);  
 Matrix(Matrix&);  
 Matrix(vector<vector<T>>&);  
  
 pair<int, int> getSize();  
 T get(int, int);  
 void editField(int, int, T);  
};  
  
class OutOfRangeException : public exception {  
public:  
 const char\* what() const \_GLIBCXX\_TXN\_SAFE\_DYN \_GLIBCXX\_NOTHROW override;  
};  
  
#include "classes.tpp"

classes.cpp

//  
// Created by vadim on 18.03.20.  
//  
#include "classes.h"  
  
const char \*OutOfRangeException::what() const \_GLIBCXX\_NOTHROW {  
 return exception::what();  
}

classes.tpp

//  
// Created by vadim on 7.02.20.  
//  
  
template <typename T>  
Matrix<T>::Matrix() {  
 n = m = 3;  
 x.assign(n, vector<int>(m, 0));  
}  
  
template <typename T>  
Matrix<T>::Matrix(int n\_, int m\_) {  
 n = n\_;  
 m = m\_;  
 x.assign(n, vector<T>(m, 0));  
}  
  
template <typename T>  
Matrix<T>::Matrix(Matrix& x\_) {  
 n = x\_.n;  
 m = x\_.m;  
 x = x\_.x;  
}  
  
template <typename T>  
Matrix<T>::Matrix(vector<vector<T>>& x\_) {  
 n = x\_.size();  
 if (n) {  
 m = x\_[0].size();  
 } else {  
 m = 0;  
 }  
 x = x\_;  
}  
  
template <typename T>  
pair<int, int> Matrix<T>::getSize() {  
 return make\_pair(n, m);  
}  
  
template <typename T>  
T Matrix<T>::get(int i, int j) {  
 if (i < 0 || i >= n || j < 0 || j >= m) {  
 throw OutOfRangeException();  
 }  
 return x[i][j];  
}  
  
template <typename T>  
void Matrix<T>::editField(int i, int j, T a) {  
 if (i < 0 || i >= n || j < 0 || j >= m) {  
 throw OutOfRangeException();  
 }  
 x[i][j] = a;  
}