Практическая работа №5. Матрицы

1. Унаследуйте класс от базового класса Многомерный массив. Убедитесь, что в родительском классе верно выделяется и освобождается память под многомерный массив. В производном классе напишите метод заполнения массива произвольным нетривиальным образом. Введите метод, который решает задачу, указанную в варианте, и возвращает из неё результат (входная матрица остаётся неизменной, на выходе должен быть новый массив или число).
2. Введите систему исключений, указанную на рисунке 1. Встройте собственный тип исключения, которое может возникнуть при решении задачи из Вашего варианта, в общую структуру наследования. Продемонстрируйте обработку исключений с помощью операторов try – throw – catch (исключения генерируются в методах класса Матрица, обрабатываются – в main).

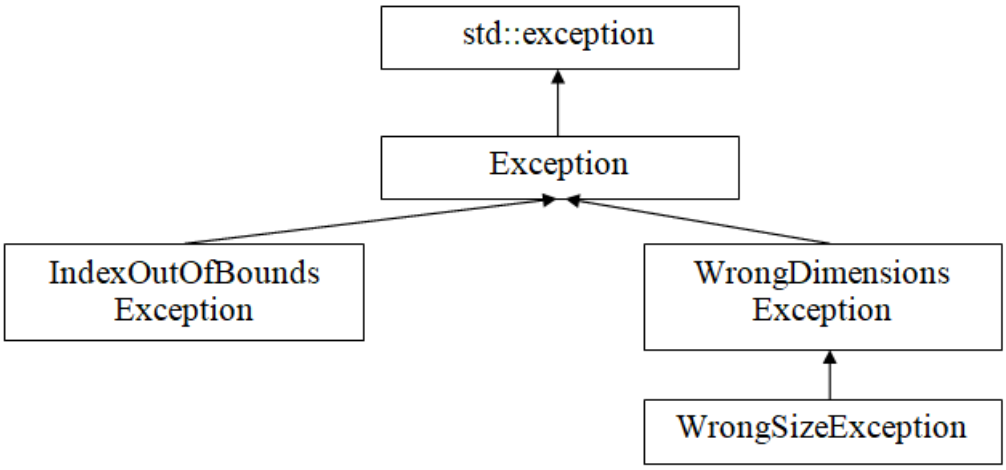

Рисунок 1. Схема наследования классов-исключений

Здесь Exception – базовый класс для исключений, которые могут произойти в классе Матрица.

IndexOutOfBounds – выход за границы массива.

WrongDimensionsException – участие в одной операции операндов (в основном, матриц) неправильных размеров (например, сложение матриц, размеры которых отличаются).

WrongSizeException – использование ошибочных значений при создании объектов класса Матрица.

3. Сохраните результат обработки нескольких матриц в файл, извлеките результаты из файла, выведите их в консоль. Продемонстрируйте, что

результаты, которые записывались в файл, и прочитанные из файла данные не отличаются.

4. Введите конструкторы и методы для чтения матрицы произвольного размера из файла.
5. Преобразуйте классы многомерных массивов к шаблонам классов.

Вариант	Массив, функция
1.	В наличии прямоугольная плоская решётка с шагом 1. В узлах решётки находятся точечные массы, хранящиеся в заданной матрице *M*. Вычислить координаты центра масс решётки.
2.	Получить вектор, в котором содержатся максимальные элементы из каждой строки матрицы.
3.	Считая каждый столбец матрицы вектором, получить новую матрицу, в которой столбцы отсортированы по евклидовой норме.
4.	Построить из двух одинаковых треугольников Паскаля произвольного размера «ромб», симметричный относительно основания треугольника.
5.	Удалить из матрицы все нулевые строки и столбцы.
6.	Построить вектор, в котором компонента равна 1, если в соответствующей строке матрицы нули чередуются с ненулевыми элементами (без учёта кратности чередования), и 0 в противном случае.
7.	Получить вектор, каждая компонента которого содержит сумму элементов из соответствующей строки матрицы.
8.	Проверить, является ли матрица нильпотентной ($A^N = 0$, можно ограничиться некоторой наперёд заданной степенью).
9.	Дана матрица неотрицательных элементов. Получить вектор, каждая компонента которого содержит среднее геометрическое элементов соответствующего столбца матрицы.
10.	Проверить, является ли матрица идемпотентной ($A^N = A$, можно ограничиться некоторой наперёд заданной степенью).
11.	Получить вектор, каждая компонента которого содержит среднее арифметическое элементов из соответствующей строки матрицы.
12.	Привести матрицу к верхнетреугольному виду по методу Гаусса.
13.	В наличии решётка в виде параллелепипеда. В узлах решётки

	находятся точечные массы (с шагом 1 по *x, y, z*), значения которых собраны в заданный массив *M*. Вычислить координаты центра масс решётки.
14.	Даны матрица Грама и два вектора. Найти скалярное произведение векторов.
15.	Удалить все строки матрицы, в которых есть нулевые элементы.
16.	Применить ко всем элементам матрицы некоторую скалярную функцию, переданную по указателю.
17.	Отсортировать элементы во всех строках матрицы по убыванию.
18.	Удалить из матрицы все строки и столбцы, среднее арифметическое которых большие некоторого параметра border.
19.	Элементы входной матрицы считать высотами некоторого участка поверхности. Получить на выходе матрицу, в которой отмечены локальные минимумы карты.
20.	Найти все вхождения некоторого элемента element в матрице. Вернуть вектор, состоящий из их индексов.
21.	Получить вектор, в котором компонента нулевая, если соответствующая строка матрицы не отсортирована, и ненулевая, если все элементы строки расположены в порядке возрастания или убывания.
22.	Удалить из матрицы все строки и столбцы, содержащие элементы, большие некоторого параметра border.
23.	Удалить все строки матрицы, в которых есть некоторый наперёд заданный элемент.
24.	Сгладить все элементы матрицы - заменить их средним двух соседних. Для краевых элементов вторым соседом считать сам элемент.
25.	Написать функцию, возвращающую новую матрицу, которая включает столбцы исходной матрицы, не содержащие ни одного нуля.
26.	Написать функцию, переворачивающую матрицу слева направо (вокруг воображаемой вертикальной оси).
27.	В строках матрицы, в которых есть отрицательные элементы, произвести циклический сдвиг вправо на число этих элементов (для каждой строки оно своё).

28.	Удалить все строки матрицы, в которых нулевых элементов более половины.
29.	Матрица представляет собой один из каналов изображения. На основе зигзаг-сканирования получить вектор (шаг 5: http://www.compression.ru/book/part2/part2__3.htm)
30.	Дана матрица неотрицательных элементов. Получить вектор, каждая компонента которого содержит среднее гармоническое элементов соответствующей строки матрицы.

Пример кода:

```
#include <iostream>
#include <fstream>
#include <typeinfo>

using namespace std;


class Exception: public std::exception
{
protected:
	//сообщение об ошибке
	char* str;
public:
	Exception(const char* s)
	{
		str = new char[strlen(s) + 1];
		strcpy_s(str, strlen(s) + 1, s);
	}
	Exception(const Exception& e)
	{
		str = new char[strlen(e.str) + 1];
		strcpy_s(str, strlen(e.str) + 1, e.str);
	}
	~Exception()
	{
		delete[] str;
	}

	//функцию вывода можно будет переопределить в производных классах, когда будет ясна конкретика
	virtual void print()
	{
		cout << "Exception: " << str;
	}
};

class BaseMatrix
{
```

```
protected:
	double** ptr;
	int height;
	int width;
public:
	BaseMatrix(int Height = 2, int Width = 2)
	{
		if (Height <= 0 || Width <= 0)
			throw Exception("Non-positive size of matrix");
		height = Height;
		width = Width;
		ptr = new double* [height];
		for (int i = 0; i < height; i++)
			ptr[i] = new double[width];
	}

	BaseMatrix(const BaseMatrix& M)
	{
		height = M.height;
		width = M.width;
		ptr = new double* [height];
		for (int i = 0; i < height; i++)
		{
			ptr[i] = new double[width];
			for (int j = 0; j < width; j++)
				ptr[i][j] = M.ptr[i][j];
		}
	}

	~BaseMatrix()
	{
		if (ptr != NULL)
		{
			for (int i = 0; i < height; i++)
				delete[] ptr[i];
			delete[] ptr;
			ptr = NULL;
		}
	}

	void print()
	{
		for (int i = 0; i < height; i++)
		{
			for (int j = 0; j < width; j++)
				cout << ptr[i][j] << " ";
			cout << "\n";
		}
	}

	double& operator()(int row, int column)
	{
		if(row<0 || column<0 || row>=height || column>=width)
```

```
                throw Exception("Index is out of bounds");
            return ptr[row][column];
        }

        friend ostream& operator << (ostream& ustream, BaseMatrix
obj);
        friend istream& operator >> (istream& ustream, BaseMatrix&
obj);

};

ostream& operator << (ostream& ustream, BaseMatrix obj)
{
        //ustream<<my_manip;
        if (typeid(ustream).name() == typeid(ofstream).name())
        {
            ustream << obj.height << " " << obj.width << "\n";
            for (int i = 0; i < obj.height; i++)
            {
                for (int j = 0; j < obj.width; j++)
                    ustream << obj.ptr[i][j] << "\n";
            }
            return ustream;
        }

        for (int i = 0; i < obj.height; i++)
        {
            for (int j = 0; j < obj.width; j++)
                ustream << obj.ptr[i][j] << " ";
            ustream << "\n";
        }

        return ustream;
}

istream& operator >> (istream& ustream, BaseMatrix& obj)
{
        if (typeid(ustream) == typeid(ifstream))
            ustream >> obj.height >> obj.width;

        for (int i = 0; i < obj.height; i++)
            for (int j = 0; j < obj.width; j++)
                ustream >> obj.ptr[i][j];

        return ustream;
}

ostream& my_manip(ostream& s)
{
        s.precision(4);
        s.fill('%');
        s.width(10);
        return s;
```

```
}

int main()
{
	try
	{
		BaseMatrix Wrong(-2, 0);
	}
	catch (Exception e)
	{
		cout << "\nException has been caught: "; e.print();
	}
	cout << "\n";
	BaseMatrix M;
	cin >> M;
	ofstream fout("out.txt");
	if (fout.is_open())
	{
		fout << M;
		fout.close();
	}
	ifstream fin("out.txt");
	BaseMatrix M1;
	if (fin)
	{
		fin>> M1;
		fin.close();
	}
	cout << M1; char c1; cin >> c1;

	return 0;
}
```