离散数学

项目说明文档

矩阵功能集 matrix

作者姓名: 高逸轩

学　　号: 2053385

指导教师: 唐剑锋

学院专业: 软件学院 软件工程

同济大学

Tongji University

1 功能简介

1.1 题目要求

实现一般矩阵、关系矩阵的建立以及部分功能。

1.2 项目需求分析

本项目在实现的过程中，考虑并且满足了以下的需求：

- ✓ 健壮性
  当用户输入的数据不合理时，系统应当给予相应的提示而非直接报错。
- ✓ 代码可读性强
  本项目在实现过程中，将代码根据功能的不同划分为了不同的代码块，同时进行了合理封装。
- ✓ 执行效率高
  对数据量比较大的情况，本系统也应该具有在较短时间内求解出正确答案的能力。

1.3 项目要求

实现基类一般矩阵的建立和内容展示功能，同时保证健壮性；利用基类一般矩阵得到派生类关系矩阵，实现关系矩阵的建立、实现关系矩阵的合成运算、利用关系矩阵得到自反、对称、传递闭包（其中传递闭包包含了 O(n^4)朴素算法和 O(n^3)Warshall 算法两种求解方法），同时保证健壮性。

1.3.1 功能要求

基类一般矩阵：

首先接受矩阵 row 行、column 列的输入，借助 input_tools 功能集保证健壮性。然后接受 row*column 个整型数据输入，作为矩阵元素，其中在每一行都保证当前行的数据合理，实现健壮性。另外实现函数 PrintMatrix()来实现矩阵内容的展示。

派生类关系矩阵：

由于关系矩阵必为方阵，所以仅需输入一个变量 vertexNumber 顶点数量来建立关系矩阵。然后接受$vertexNumber^2$个整型数据(0/1)输入，作为矩阵元素，其中在每一行都保证当前行的数据合理，实现健壮性。另外，根据闭包与对应关系矩阵的关系，实现了自反、对称、传递闭包的求解。

1.3.2 输入格式

以下内容利用 input_tools 函数集实现健壮性

一般矩阵：

矩阵行 row、矩阵列 column　　（整型数据）

矩阵内容 row*column　　（整型数据）

关系矩阵

关系矩阵顶点数目 vertexNumber　（整形数据）

关系矩阵内容 $vertexNumber^2$　　（0/1）

1.3.3 项目简单示例

本函数集功能在后续作业中实现，具体实例请详见其他实验报告。

2 项目实现

本项目核心共两部分：

- ✓ 初始化设置矩阵
- ✓ 实现矩阵类的各类功能

下面将对本项目核心的类进行介绍，部分代码如下：

2.1 基类 一般矩阵

2.1.1 构造函数

```
// 矩阵 基类
class Matrix
{
public:
    // 默认构造函数
    Matrix(const int x = 0, const int y = 0);

    // 复制构造函数
    Matrix(const Matrix& currentMatrix);

    // 赋值构造函数
    Matrix& operator=(const Matrix& currentMatrix);

    // 析构函数
    ~Matrix();

    // 设置矩阵初始信息
    void SetMatrix();

    // 输出矩阵信息
    void PrintMatrix();
protected:
    // 储存矩阵信息的二维数组指针
    int** nearArray = NULL;
private:
    // 矩阵行
    int row = 0;

    // 矩阵列
    int column = 0;
};
```

构造函数共三种：默认构造，复制构造，赋值构造。以复制构造函数为例：

```
// 复制构造函数
Matrix::Matrix(const Matrix& currentMatrix)
{
    if (currentMatrix.nearArray == nearArray) return;   // 复制对象为本身

    row = currentMatrix.row;                            // 行列复制
    column = currentMatrix.column;

    nearArray = new int* [row];                         // 申请空间，深构造
    FOR(i, 0, row)
        nearArray[i] = new int[column];

    FOR(i, 0, row)                                      // 内部信息复制
        FOR(j, 0, column)
        nearArray[i][j] = currentMatrix.nearArray[i][j];
}
```

在动态空间申请时，采取根据传入参数开辟对应空间大小的方式，避免了空间的浪费。同时，在传入矩阵对象为参数是，采用引用方式，减少了另外创建临时对象的开销时间。

```cpp
// 赋值构造函数
Matrix& Matrix::operator=(const Matrix& currentMatrix)
{
    if (currentMatrix.nearArray == nearArray) return *this;

    // 在赋值前先将原来的空间释放，避免发生内存丢失错误
    FOR(i, 0, row)
        delete[] nearArray[i];

    delete[] nearArray;

    row = currentMatrix.row;                                    // 行列赋值
    column = currentMatrix.column;

    nearArray = new int* [row];                                 // 空间申请
    FOR(i, 0, row)
        nearArray[i] = new int[column];


    FOR(i, 0, row)
        FOR(j, 0, column)
        nearArray[i][j] = currentMatrix.nearArray[i][j];// 内部信息复制

    return *this;
}
```

赋值构造函数中，在重新开辟新空间并复制内容之前，需要先将原来的空间 delete 释放,避免原有空间无法释放造成内存泄漏。

2.1.2 矩阵设置函数

```cpp
// 设置初始信息
void Matrix::SetMatrix()
{
    cout << endl << "请输入矩阵：" << endl;
    FOR(i, 0, row)
    {
        cout << endl << "请输入矩阵第" << i << "行(以空格分隔)：" << endl;
        FOR(j, 0, column)
        {
            string errorTips = "第";
            errorTips += char(i + '0');
            errorTips += "行";
            errorTips += char(j + '0');
            errorTips += "列输入错误，请从此处开始重新输入本行元素:";
            errorTips += '\n';// 得到当前矩阵元素输入错误对应提示
            FOR(k, 0, j)
            {
                errorTips += char(nearArray[i][k] + '0');
                errorTips += ' ';
            }
            nearArray[i][j] = getint(INT_MIN, INT_MAX, errorTips);        // 关系矩阵中的值为0/1，通过设置getint自定义函数参数控制
        }
    }
    cout << endl << "设置结束，当前矩阵为：" << endl;
    this->PrintMatrix();
}
```

在设置矩阵信息时，利用了 STL 的 string 类来设置错误输入提示，在保证健壮性的同时，利于用户定位错误输入的位置。

2.2 派生类 关系矩阵

派生类中大部分成员函数功能与基类类似，为实现基类代码的复用率，以复制构造函数为例，使用了显示调用基类函数的方法，其他部分的解释呈现在了代码中。以派生类的复制构造函数为例：

```
// 复制构造函数
RelationMatrix::RelationMatrix(const RelationMatrix& currentMatrix) :Matrix(currentMatrix)
{
    vertexNumber = currentMatrix.vertexNumber;    // 顶点个数复制，其他成员已通过基类复制
}
```

3 项目测试

3.1 关系矩阵信息设置

```
请输入关系矩阵顶点个数(0,1000)：5

请输入关系矩阵：

请输入矩阵第0行(以空格分隔)：
0 0 0 0 1

请输入矩阵第1行(以空格分隔)：
1 1 1 1 1

请输入矩阵第2行(以空格分隔)：
1 1 0 1 0

请输入矩阵第3行(以空格分隔)：
1 1 1 0 1

请输入矩阵第4行(以空格分隔)：
1 1 1 1 1

设置结束，当前矩阵为：
0 0 0 0 1
1 1 1 1 1
1 1 0 1 0
1 1 1 0 1
1 1 1 1 1
```

3.2 健壮性测试

```
请输入关系矩阵顶点个数(0,1000)： 4

请输入关系矩阵：

请输入矩阵第0行(以空格分隔)：
1 0 1 1

请输入矩阵第1行(以空格分隔)：
1 2 3 4
第1行1列输入错误，请从此处开始重新输入本行元素：
1
```

```
请输入关系矩阵顶点个数(0,1000)： 4

请输入关系矩阵：

请输入矩阵第0行(以空格分隔)：
1 1 0 1

请输入矩阵第1行(以空格分隔)：
1 1 1 1

请输入矩阵第2行(以空格分隔)：
ijewqhofwdgi w38q7469b27 tq nigwu
第2行0列输入错误，请从此处开始重新输入本行元素：
1 2 3 4
第2行1列输入错误，请从此处开始重新输入本行元素：
1
```

在后续项目中有展示本函数集求闭包的功能，请参考其他实验报告的内容。

4 心得与总结

在本次矩阵类实现的过程中，需要仔细考虑并设计类，以保证继承与派生后代码的复用率。同时，为提高基类某些函数的利用率，我学习了在派生类函数中显式调用基类函数。在健壮性处理方面，我利用自己的 input_tools 函数集和 STL 的 string 类特性，简单设置了更加完备的错误提示，方便用户定位自己输入错误的具体位置。另外，实现求各类闭包功能的展示放入了其他实验报告中详解，请老师审阅。