

实验1. C++综合及面向对象程序设计

一. 实验目的

- 1. 复习C++类和对象概念。
- 2. 熟悉继承的基本原理和程序实现方法。
- 3. 熟悉多态的基本原理和程序实现方法。
- 4. 熟悉C++集成开发环境，强化C++程序开发的调试技巧。

二. 实验内容

实验	说明
实验1-1. 2x2矩阵	实现固定尺寸的矩阵类，初步体会C++类作为扩展数据类型
实验1-2. 任意尺寸的矩阵	实现任意尺寸的矩阵类，深入体会C++类作为扩展数据类型
实验1-3. 继承与派生	实现矩阵类的继承派生

三. 实验步骤和结果

实验1-1. 2x2的矩阵

需求说明：

- 1. 实现矩阵类 `Mat2x2`，其宽和高都是2，并实现如下功能：
 - 初始化矩阵
 - 访问（读写）矩阵的元素
 - 显示矩阵内容
 - 矩阵赋值
 - 矩阵运算，主要实现加法
- 2. 类的定义和实现分别存放在 `mat2x2.h` 文件和 `mat2x2.cpp` 文件中，并将 `main` 函数存放在 `main.cpp` 文件中。

💡 类定义名称 `Mat2x2` 中的乘号实际上是 `x`。

实验1-1a. 矩阵的初始化

功能	参考示例 (<code>a, b</code> 是 <code>Mat2x2</code> 类型变量)	说明
初始化	<code>Mat2x2 a(1);</code>	初始化为特定值
缺省初始化	<code>Mat2x2 a;</code>	矩阵各元素需赋值为零
拷贝初始化	<code>Mat2x2 b(a);</code> 或 <code>Mat2x2 b = a;</code>	用另一个矩阵来初始化

实验步骤：

- 1. 在 `mat2x2.h` 中定义矩阵类的成员变量和构造函数。参考示例如下：

```
class Mat2x2 {
public:
    Mat2x2(double init_value=0);    // 以相同值初始化矩阵元素
```

```

    Mat2x2(const double* elements); // 以顺序访问数组的值初始化矩阵元素
    Mat2x2(const Mat2x2& another);  // 以另一个矩阵初始化当前矩阵的元素，实际是元素拷贝
private:
    double m_data[4];              // 矩阵元素。请注意，此例中m_data私有，外部不可访问
};

```

回答问题：

1. 测试以下代码能否执行？并给出运行结果截图。

```

double data[] = {1, 2, 3, 4};
Mat2x2 a(data);
Mat2x2 b(5);
Mat2x2 c = a;
Mat2x2 d;

```

实验1-1b. 访问和修改矩阵元素

参见 实验2-1a. 矩阵的初始化 可知，成员变量 `m_data` 私有，外部不可访。故须定义读写元素的函数接口。

功能	参考示例 (<code>a</code> 是 <code>Mat2x2</code> 类型变量)	说明
读写元素	<code>a.at(y, x) += 1.0;</code>	定义 <code>at</code> 成员函数，以行号和列号为参数，返回元素的引用

实验步骤：

1. 定义成员函数

```

class Mat2x2 {
    ...
    double& at(int y, int x) { // 读和写访问
        return m_data[y * 2 + x];
    }
    const double& at(int y, int x) const { // 只读访问，注意前后两个const
        return m_data[y * 2 + x]; // 实现部分和前一个函数一样
    }
};

```

回答问题：

1. 测试以下代码，看能否执行？并分析每个 `at` 是调用的哪一个成员函数？

```

double data[] = {1, 2, 3, 4};
Mat2x2 a(data);
cout << a.at(0, 0) << endl;
a.at(0, 0) = 9; // 注意此特殊用法，是向元素赋值
cout << a.at(0, 0) << endl;
const Mat2x2& ref_a = a; // 只读引用，因此ref_a只能调用以const结尾的成员函数
cout << ref_a.at(0, 0) << endl;

```

2. 若去掉返回值的引用，即程序改为 `double Mat2x2::at(int row, int col)`，请测试先前示例能否如期执行？以此分析引用(`&`)的用途。

实验1-1c. 矩阵的赋值运算

功能	参考示例 (a,b 是 Mat2x2 类型变量)	说明
赋值	a = b;	重载运算符 =

⚠️需注意： a = b; 和 Mat2x2 a = b; 在本质上不一样！前者是对a赋值为b，调用 operator= 成员函数；后者是用b初始化a，调用拷贝构造函数。

实验步骤：

1. 定义并实现重载 operator= 运算符的成员函数。

```
class Mat2x2 {
    ...
    void operator = (const Mat2x2& b); // 即 a = b, 其中*this即为a
};
```

回答问题：

1. 请用以下测试用例进行测试，给出结果的截屏，并分析程序是否正确。

```
double data[] = {1, 2, 3, 4};
Mat2x2 a(data), b;
b = a;
a.at(0, 0) = 5;
cout << a << endl; // a应为[5, 2; 3, 4] (Matlab语法)
cout << b << endl; // b应为[1, 2; 3, 4]
```

2. 若需要实现如下功能，需如何修改代码？

```
Mat2x2 a;
a = 5; // 将a的所有元素都赋值为5，即 [5, 5; 5, 5]
```

提示：

- 只需循环地拷贝各行和各列的元素。

实验1-1d. 矩阵的加法运算

功能	参考示例 (a,b,c 是 Mat2x2 类型变量)	说明
内联加法	a += b;	重载运算符 += ，无返回值
加法	c = a + b;	重载运算符 + ，返回一个矩阵

实验步骤：

1. 实现矩阵类的加法运算符重载函数。

```
class Mat2x2 {
    ...
    void operator += (const Mat2x2& b); // 即 a += b, 其中*this即为a
    Mat2x2 operator + (const Mat2x2& b) const; // 即 c = a + b,
```

// 其中*this即为a，返回值即为c

```
};
```

2. 调用矩阵类，验证加法功能。

回答问题：

1. 给出两种加法运算符重载函数实现的代码；
2. 分别为两种加法运算符设计2个测试用例（共4个用例），用以验证程序的正确性。并请给出测试用例运行结果的截图。

提示：

- 加法运算是逐个元素进行的。
- 运算 $a += b$ 实质是 $a.operator+=(b)$ ；运算 $c = a + b$ 实质是 $c = a.operator+(b)$ （注意还另有一个赋值运算）；运算 $Mat2x2\ c = a + b$ ；时，除了加法运算，还有一个拷贝构造函数被调用。

实验1-2. 任意尺寸的矩阵

实验1-2a. 任意尺寸矩阵的初始化

需求说明：

1. 实现矩阵类 `Matrix`，其功能要求与实验2-2基本相同，但 `Matrix` 的宽和高可指定为任意值。

实验步骤：

1. 设计 `Matrix` 的初始化

```
class Matrix {
public:
    Matrix(int height, int width, double value=0); // 以固定值初始化所有元素
    Matrix(int height, int width, const double* data); // 以数组顺序访问的值初始化
    Matrix(const Matrix& another); // 拷贝初始化
    ~Matrix(void); // 释放
private:
    int m_height, m_width; // 矩阵的高(行数)与宽(列数)
    double* m_data; // 矩阵元素
};
```

2. 设计矩阵内部元素的**初始化**方法，在 `matrix.cpp` 中实现矩阵类的构造函数。需注意判断参数 `data` 是否为空指针，若为空则初始化所有元素为零，否则进行值拷贝。

```
Matrix::Matrix(int height, int width, const double* data) :
    m_rows(height), m_width(width), m_data(new double[height * width]) {
    ... // 请完善矩阵元素的初始化
}
```



矩阵尺寸不是静态的，而是在程序运行时动态给出，因此需使用动态内存。

3. 在 `matrix.cpp` 中实现矩阵类的**析构函数**。注意释放内存。

回答问题：

1. 请结合程序说明，如何动态地生成任意尺寸的矩阵？
2. 如果不实现析构函数，程序能否执行？有何危害？

实验1-2b. 任意尺寸矩阵的运算（选作）

需求说明：

1. 参照 实验1-1. 2x2的矩阵，实现任意尺寸矩阵的加法运算，分别给出2个测试用例（共4个用例）及运行结果截图。

实验1-3. 继承与派生

实验1-3a. 继承的基本概念

实验内容：


- 新建命令行项目，执行如下代码。

```
#include <iostream>
using namespace std;
class Parent { // 基类
public:
    void f1() {cout << "This is parent-f1" << endl;}
    virtual void f2() {cout << "This is parent-f2" << endl;} // 声明f2是虚函数
}parent;
class Child : public Parent { // 派生类
public:
    void f1() {cout << "This is child-f1" << endl;}
    void f2() {cout << "This is child-f2" << endl;}
} child;
int main(int argc, char* argv[]) {
    Parent* p = new Child; // 定义一个基类指针
    p->f1();                // 执行哪个f1?
    p->f2();                // 执行哪个f2?
    delete p;
    return 0;
}
```

请给出实验结果截图，并分析分别是哪个 f1 和 f2 函数被调用，为什么？

实验1-3b. 矩阵类的派生（选作）

在先前实验中，2x2矩阵和任意尺寸矩阵是矩阵的不同实现，都包含了赋值、加法等矩阵基本运算，而前者速度快，后者功能强大，各具有其适用场合。

 因此，可以考虑“打通”这两个类，将其作为同一个基类的派生类，使其通用接口互相兼容。其中，基类中以纯虚函数定义接口。纯虚函数是以 =0 结尾且无实现的特殊虚函数，格式：`virtual 返回值类型 函数名(参数表)=0;`；含有纯虚函数的类是纯虚类。

 需注意：基类和派生类中相对应的虚函数需要接口完全相同，即**返回值、函数名和参数一致**。

需求说明：

1. 定义矩阵基类 `MatBase`，其中包含了纯虚函数 `at`，以定义统一矩阵元素访问接口。

```
class MatBase { // 定义基类，其中包含了矩阵元素读取的纯虚函数
public:
    virtual double& at(int y, int x)=0; // 定义纯虚函数
    virtual const double& at(int y, int x) const=0; // 定义纯虚函数
};
```

2. 将 Mat2x2 和 Matrix 类改为 MatBase 的派生类，需注意纯虚函数需在派生类中实现。

```
class Mat2x2 : public MatBase { // 修改 Mat2x2，使其为派生类，需实现虚函数功能
...
    virtual double& at(int y, int x); // 定义虚函数
    virtual const double& at(int y, int x) const; // 定义虚函数
...
};
```

```
class Matrix : public MatBase { // 修改 Matrix，使其为派生类，需实现虚函数功能
...
    virtual double& at(int y, int x); // 定义虚函数
    virtual const double& at(int y, int x) const; // 定义虚函数
...
};
```

3. 在测试中，执行如下代码（可自行修改拓展）。可见参照基类调用接口，可自动从派生类接口中输出。

```
void print_element(const MatBase& m, int y, int x) { // 注意参数是基类
    std::cout << m.at(y, x) << std::endl; // 调用虚函数所定义的接口
}
Mat2x2 a(9.0);
Matrix b(2, 2, 7.0);
print_element(a, 0, 0); // 输出 Mat2x2 变量 a 的第一个元素，注意输入是派生类
print_element(b, 0, 0); // 输出 Matrix 变量 b 的第一个元素，注意输入是派生类
```

回答问题：

1. 请给出如前文测试代码运行结果的截图。
2. 执行如下程序，测试含有纯虚函数的基类能否独立使用？分析为什么？

```
MatBase m;
```

3. 若派生类中未实现虚函数 at 的功能（比如，注释掉派生类的 at 函数），能否编译通过？分析为什么？

提示：

- 在互联网上检索 c++ + 虚函数 + 接口。