Final Project 泛型矩阵类库设计

基本要求

本次大作业的目标为设计一个泛型矩阵运算标头库（header only），此库内部含有一个矩阵类（为表述方便，假定名称为`Matrix`）可以完成矩阵的加法、减法和乘法，`Matrix`具有一个模板参数表示其元素类型，他应该支持下述的全部用法和特性

```
// Matrix应该支持无参构造，此时它是一个1*1的矩阵，内部元素值为模板类型的默认值
Matrix<int> mat1;

// 通过矩阵尺寸构造，内部元素值为模板类型的默认值
Matrix<int> mat2(2, 2);
mat2[1][0] = 42; // 可以通过[][]访问内部元素
//mat1[0][1];    // 访问越界时程序行为是未定义的
assert(mat2.at(2) == 42); // 可以使用at访问矩阵按行展开后对应的元素
//mat2.at(5); // 使用at访问越界时应该抛出异常
mat2.at(1) = 1;
// 可以使用push_back()向Matrix添加元素，新添加的元素仅能通过at访问
mat2.push_back(13);
mat2.push_back(14);
assert(mat2.at(4) == 13); // pass
assert(mat2.at(5) == 14); // pass
// 可以使用reshape()修改矩阵尺寸，多余的元素会被删除，缺少的元素会使用默认值填充
mat2.reshape(3, 3);
/* 此时mat2表示的矩阵为
 0  1 42
 0 13 14
 0  0  0
*/

// 使用initializer_list构造Matrix，此时Matrix为一个行向量
Matrix<int> mat3{1, 2, 3, 4};
Matrix<int> mat4{5, 6, 7, 8};
mat3.reshape(2, 3);
mat4.reshape(2, 3);
mat3 + mat4; // 矩阵加法，如果矩阵尺寸不同需抛出异常
mat3 - mat4; // 矩阵减法，如果矩阵尺寸不同需抛出异常
mat3 * mat4; // 矩阵乘法，此处为两个2*3的矩阵相乘，不满足矩阵乘法定义，需要抛出异常
```

其他要求：`Matrix<T>`需要实现移动语义

扩展1：

需要对 `Matrix` 模板参数进行检查，模板参数不能带有 `const/volatile` 修饰且不能是指针或引用类型也不能是 `void` 类型，你需要对模板参数进行判断，并使用 `C++20` 提供的 `Concept` 功能或 `C++11` 提供的 `static_assert` 对不符合条件的参数进行提示并生成编译错误（提示：标准库 `type_traits` 提供了现成的类型检查功能），另外考虑以下代码

```
struct AddOnly
{
    int v;
    AddOnly operator +(AddOnly & rop)
    {
        return AddOnly{v + rop.v};
    }
}

Matrix<AddOnly> mat; // 此处无编译错误!
mat*mat; // 编译错误
```

上述代码编译错误的原因是 `AddOnly` 不支持减法和乘法操作，对于这种情况你也需要使用 `Concept` 或 `static_assert` 进行检查

扩展2：

现在我们想将与矩阵尺寸有关的操作提至编译期执行（包括内存分配），现在矩阵的尺寸通过两个额外的模板参数指定，这两个模板参数均需要大于零，否则产生编译错误，`push_back` 不再是此矩阵类型的成员函数，`reshape` 的尺寸参数通过模板参数给定，你需要实现如下功能（此模板名称应和基本要求中的模板名称一致）：

```
Matrix<int, 2, 3> mat1{1, 2, 3, 4, 5, 6};
// 缺少的元素使用默认值填充
Matrix<int, 3, 2> mat2{1, 0, 0, 1};
Matrix<int, 2, 2> res = mat1 * mat2;
/* res =
1 2
4 5
*/
Matrix<int, 3, 3> reshape_res = res.reshape<3, 3>();

mat1 * mat1; // 编译错误，矩阵尺寸不满足矩阵乘法定义
Matrix<int, 3, 3> temp = mat1 * mat2; // 编译错误，temp类型与结果不匹配
//Matrix<int, 1, 1> mat2{1, 0, 0, 1}; // 程序发生未定义行为
```

上述编译错误的情况均需要使编译器输出人类可读的错误信息，移动语义不需要实现，其他未明确的要求与基本要求和扩展一相同

扩展3：

设计矩阵拼接函数，要求可以指定行拼接或列拼接，具体使用方式如下：

```
// 基本要求的情况
Matrix<int> mat1(2, 2);
Matrix<int> mat2(2, 2);
Matrix<int> mat3(3, 3);
Matrix<int> res = concatenate(mat1, mat2, ROW); // res尺寸为（2，4），ROW为枚举类型，你可以选择其他名称
concatenate(mat1, mat3, ROW); // 抛出异常

// 扩展2的情况
Matrix<int, 2, 2> mat4;
Matrix<int, 2, 2> mat5;
Matrix<int, 3, 3> mat6;
Matrix<int, 2, 4> res = concatenate<ROW>(mat4, mat5);
concatenate<ROW>(mat4, mat6); // 编译错误
```

扩展4：

为了提高代码的可维护性，请保证矩阵运算相关的实现代码仅在作业中出现一次（减少重复代码）

补充知识：

矩阵加减法定义：

两矩阵的尺寸相同时矩阵加减法有定义，结果矩阵中各元素如下面式子得出

$$(A \pm B)_{ij} = a_{ir} \pm b_{rj}$$

矩阵乘法定义：

第一个矩阵的列数和第二个矩阵的行数相同时矩阵乘法才有定义。若A为m×n矩阵，B为n×p矩阵，则他们的乘积AB是一个m×p矩阵。其乘积矩阵的元素如下面式子得出：

$$(AB)_{ij} = \sum_{r=1}^{n} a_{ir} b_{rj}$$

关于 `static_assert` ：

`static_assert` 功能为编译期断言，可以使编译器输出更加友好的编译错误信息，用法请参考CppReference：static_assert declaration - cppreference.com