深蓝学院
shenlanxueyuan.com

泛型矩阵类库设计思路讲解

主讲人 葛磊

纲要

- 第一部分：整体思路
- 第二部分：方法

整体思路

- 本题目主要考察的是c++的泛型编程的相关知识点。其中模板就是泛型编程的基础。
- 本次作业的主要目的就是设计一个简单的矩阵运算的头文件，可以对矩阵进行初始化，并进行一些简单的矩阵运算。

纲要

- 第一部分：概述
- **第二部分：方法**

基本要求

- 基本要求实现的是动态内存分配的矩阵，对于这个结构体我们可以设置如下的成员变量。

```
template <typename T>
class Matrix {

  int row_;
  int col_;
  std::vector<T> elements_;
};
```

基本要求

- Reshape
  - 更改row和col的值
  - 为elements_增加0或者删除多余的元素

```
template <typename T>
class Matrix {

  int row_;
  int col_;
  std::vector<T> elements_;
};
```

基本要求

- Initializer_list构造矩阵

```
Matrix(std::initializer_list<T> l) : elements_(l) {
  //      ref: cppreference initializer_list
  col_ = 1;
  row_ = elements_.size();
}
```

基本要求

- operator[]的重载

```
//  operator []
typename std::vector<T>::iterator operator[](std::size_t idx) {
  return elements_.begin() + (idx * col_);
}
```

基本要求

- 操作符重载
  - 类模板与成员函数模板

扩展1

- 扩展1主要是进行模板参数的检查。在作业中也写明了提示，可以用type_traits来进行类型检查
- 要注意报错的位置。

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

扩展1

深蓝学院
shenlanxueyuan.com

- 参考代码
  - 类模板与成员函数模板

```
template <typename T> concept Addable = requires(T a, T b) { a + b; };
```

```
// add()
friend auto operator+(const Matrix<T> &lhs, Matrix<T> &rhs) {
  static_assert(Addable<T>, "is not add able");
```

扩展2

- 对于扩展2来说，是完成一个静态内存分配的矩阵。
- 操作符重载需要使用类模板的成员函数模板的方法实现

```
template <typename T, size_t T_row, size_t T_col>
struct MatrixStaticData {
  static constexpr size_t row_=T_row;
  static constexpr size_t col_=T_col;
  std::vector<T> elements_;
}
```

扩展3

- 使用函数模板来实现矩阵拼接函数

扩展4

- 减少重复代码
  - 函数模板
  - 模板继承

扩展4

- 函数模板

```
template <typename L, typename R>
auto operator+(const L& lhs, const R& rhs) {
  L res(lhs.get_row(), rhs.get_col());
  auto size = lhs.get_row() * lhs.get_col();
  for (int i = 0; i < size; ++i) {
    res.at(i) = lhs.at(i) + rhs.at(i);
  }
  return res;
}
```

扩展4

- 模板继承
  - 第一层基类中有数据成员，保存矩阵的数据。这里有两种基类，一种是静态内存分配，一种是动态内存分配

```
template <typename T>
struct MatrixDynamicData {
  size_t row_;
  size_t col_;
  std::vector<T> elements_;
};

template <typename T, size_t T_row, size_t T_col>
struct MatrixStaticData {
  static constexpr size_t row_=T_row;
  static constexpr size_t col_=T_col;
  std::vector<T> elements_;
}
```

扩展4

- 模板继承
  - 第二层的结构体会根据不同的情况来继承不同的数据基类。一些基础的计算将会在这层进行。

```
template<class T>
struct MatrixBase : public T
{
  template<class T1, class T2>
  void add(const T1& rhs, T2& res)
  {
    for(size_t i=0; i<T::row_*T::col_;++i)
      res.elements_[i]=T::elements_[i]+rhs.elements_[i];
  }

}
```

扩展4

- 模板继承
  - 第三层的结构为最后的子类。会根据情况来选择继承哪种基类。用特化的方法来实现。
  - 感谢助教-云行月下提供的第二种方法

代码框架和思路

```
template <typename T, size_t... sizes> struct Matrix;

template <typename T>
struct Matrix<T> : public MatrixBase<MatrixDynamicData<T>> {

  Matrix operator+(const Matrix &rhs) {
    Matrix res(this->get_row(), this->get_col());
    this->add(rhs, res);
    return res;
  }
};

template <typename T, size_t T_row, size_t T_col>
struct Matrix<T, T_row, T_col>
    : public MatrixBase<MatrixStaticData<T, T_row, T_col>> {

  Matrix operator+(const Matrix &rhs) {
    Matrix res;
    this->add(rhs, res);
    return res;
  }
};
```

在线问答

深蓝学院
shenlanxueyuan.com

Q&A

深蓝学院
shenlanxueyuan.com

感谢各位聆听！

Thanks for Listening