深蓝学院
shenlanxueyuan.com

C++ 基础与深度解析
Project6-矩阵库 思路提示
2023-05-05

主讲人 天哲

纲要

- 整体思路
- 扩展
- 扩展内容
- 其他

基本设计

- 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。参数包和折叠表达式

基本设计

- 基本的就是考虑需要行数，列数，还有一个vector来存储数据

```
template <compuatable T>
class Matrix
{
    static_assert((!std::is_const<T>()), "inputType cant be const");

private:
    //default value
    T defval;
    //use a single vector to store. []operator uses span to provide view in matrix
    std::vector<T> data;
    //size data
    uint16_t rowNum, colNum;
}
```

基本设计

- 部分的初始化方式，填充默认值或者使用初始化列表

```
Matrix(uint16_t rows, uint16_t cols) : defval(0), data(rows * cols,
defval), rowNum(rows), colNum(cols)
    {
        if (rows == 0 || colNum == 0)
            throw std::invalid_argument("Matrix::(uint16_t rows,
uint16_t cols), arguments cant be 0");
    }
    //mainly copy vector data, init size value
    Matrix(std::initializer_list<T> list) : data(list), rowNum(1),
colNum(list.size())
    {
        if (list.size() == 0)
            *this = Matrix();
    }
```

基本设计-加法

- 这里使用友元

```
//matrix calculations
    friend Matrix operator+(const Matrix &l, const Matrix &r)
    {
        if ((l.rowNum != r.rowNum) || (l.colNum != r.colNum))
            throw std::length_error("Matrix::operator+,
length does not match");

        Matrix ret(l.rowNum, l.colNum);
        //cant use data.len due to push_back
        const uint16_t len = l.rowNum * l.colNum;
        for (auto i = 0; i < len; ++i)
            ret.data[i] = l.data[i] + r.data[i];
        return ret;
    }
```

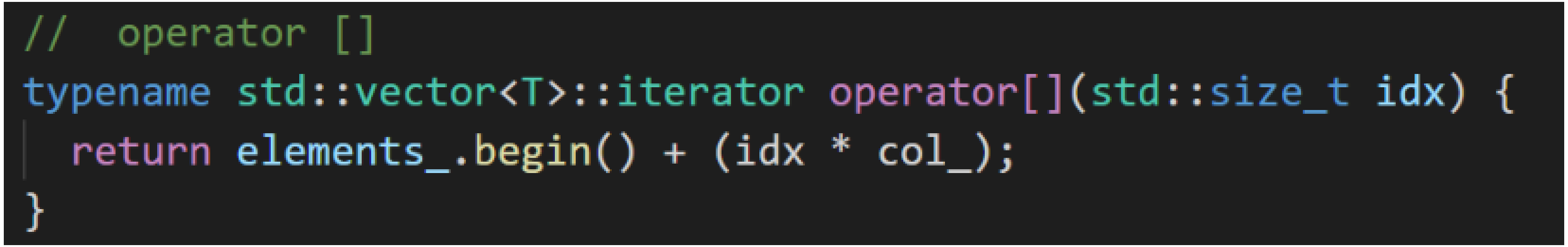
基本设计-乘法

```
Matrix operator*(const Matrix &m)
    {

        if ((this->colNum != m.rowNum))
            throw std::length_error("Matrix::operator*, length does not match");

        uint16_t newRow = this->rowNum, newCol = m.colNum;
        Matrix ret(newRow, newCol);
        myPrintf(LOG_DEBUG, "Debugging operator *(const Matrix &m)");
        //cant use data.len due to push_back
        for (auto i = 0; i < newRow; ++i)
        {
            for (auto j = 0; j < newCol; ++j)
            {
                for (auto k = 0; k < m.rowNum; ++k)
                {
                    //debug use
                    //std::cout << this->getVal(i, k) << ' ' << m.getVal(k, j) << std::endl;
                    ret[i][j] += (this->getVal(i, k)) * (m.getVal(k, j));
                }
            }
        }
        return ret;
    }
```

基本设计-[]运算符



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

```
std::span<T> operator[](uint16_t rowId)
    {
        if (rowId > (rowNum - 1))
            throw std::range_error("Matrix::operator[], access 
invalid row");

        const auto beginIt = data.begin() + rowId * colNum;
        return std::span(beginIt, colNum);
    }
```

扩展内容 参数检查

- 这里使用concept和requires进行检查

```
template <typename T>
concept typecheck = (!std::is_const<T>::value) &&
                    (!std::is_void<T>::value) && (!std::is_pointer<T>::value) &&
                    (!std::is_volatile<T>::value) &&
(!std::is_lvalue_reference<T>::value) && (!std::is_rvalue_reference<T>::value);
template <typename T>
concept computable = requires(T a)
{
    a + a;
    a *a;
    a - a;
}
&&typecheck<T>;
template <computable T, uint16_t rowNum, uint16_t colNum>
requires(rowNum > 0 && colNum > 0) class Matrix;
```

扩展内容 内存分配

```
template <computable T, uint16_t rowNum, uint16_t colNum>
requires(rowNum > 0 && colNum > 0)
class Matrix
{
    static_assert((!std::is_const<T>()), "inputType cant be const");

private:
    //default value
    T defval;
    constexpr static uint16_t msize = rowNum * colNum;
    std::array<T, msize> data;
}
```

扩展内容 矩阵拼接

- 先确定拼接完的类型

```
template <uint16_t op, computable T, uint16_t rowL, uint16_t colL, uint16_t rowR, uint16_t colR>
//requires((op == ROW && colL == colR) || (op == COL && rowL == rowR))
struct concatenateType
{
    using type = std::conditional<(op == ROW), Matrix<T, rowL + rowR, colL>, Matrix<T, rowL, colL
+ colR>>::type;
};
template <uint16_t op, computable T, uint16_t rowL, uint16_t colL, uint16_t rowR, uint16_t colR>
requires((op == ROW && colL == colR) || (op == COL && rowL == rowR))
    concatenateType<op, T, rowL, colL, rowR, colR>::type concatenate(const Matrix<T, rowL, colL>
&l, const Matrix<T, rowR, colR> &r);
```

在线问答

Q&A

深蓝学院
shenlanxueyuan.com

感谢各位聆听！

Thanks for Listening