HW5

181220076 周韧哲

一. 概念题

1. 隐式赋值操作可能会将成员指针指向同一块内存区域，可能导致double free或者内存泄漏。可以自定义赋值操作符重载函数。
2. 创建一个新对象时用另一个已存在的同类对象初始化时调用拷贝构造函数；对两个已存在对象，用其中一个对象去改变另一个对象的状态时，调用赋值操作符重载函数。
3. 可以实现堆内存管理，提高堆内存的分配和归还效率。
4. c++编译器会把一个lambda表达式生成一个匿名类的匿名对象，并在类中重载函数调用运算符。

二. 编程题

1.

```
class Array{
    int *data;
    int length;
public:
    Array(){
        length = 0;
    }
    Array(int c){
        set(c);
    }
    ~Array(){
        delete []data;
    }
    void set(int c){
        length = c;
        data = new int[c];
    }
    int &operator[] (int j){
        return data[j];
    }
    friend class Matrix;
};

class Matrix{
    Array *p_data;
    int row, col;
    void init(int r, int c){
        row = r;
        col = c;
        p_data = new Array[r];
        for(int i=0;i<row;i++){
            p_data[i].set(c);
        }
    }
public:
    Matrix(){
```

```
        row = col = 0;
        p_data = NULL;
    }
    Matrix(int r, int c){
        init(r, c);
    }
    ~Matrix(){
        delete []p_data;
    }
    Array &operator[] (int i){
        return p_data[i];
    }
    Matrix &operator= (const Matrix &m){
        if(&m == this) return *this;
        if(row != m.row || col != m.col){
            delete []p_data;
            init(m.row, m.col);
        }
        for(int i=0;i<row;i++){
            for(int j=0;j<col;j++){
                int tmp = m.p_data[i].data[j] ;
                p_data[i].data[j] = tmp;
            }
        }
        return *this;
    }
    bool operator== (const Matrix &m) const{
        if(row != m.row || col != m.col)
            return false;
        for(int i=0;i<row;i++){
            for(int j=0;j<col;j++){
                if(p_data[i][j]!=m.p_data[i][j])
                    return false;
            }
        }
        return true;
    }
    Matrix operator+ (const Matrix &m) const{
        Matrix tmp(row, col);
        for(int i=0;i<row;i++){
            for(int j=0;j<col;j++){
                tmp[i][j] = p_data[i][j] + m.p_data[i][j];
            }
        }
        return tmp;
    }
    Matrix operator* (const Matrix &m) const{
        Matrix tmp(this->row, m.col);
        for(int i=0;i<tmp.row;i++){
            for(int j=0;j<tmp.col;j++){
                int sum = 0;
                for(int p=0;p<col;p++){
                    sum += p_data[i][p] * m.p_data[p][j];
                }
                tmp[i][j] = sum;
            }
        }
        return tmp;
```

```
95      }
96  };
```

2. 在malloc时多申请void *大小指针，当内分配内存delete完后就可以在堆中通过这一指针free这一块内存。