# HW12

*181220076 周韧哲*

## 一. 概念题

1. 从流类库的基本结构可以看到，ios类是istream 类和ostream 类的基类，从ios头公有派生istream 和ostream两个类， 而iostream 类通过多重继承istream 类和ostream类而产生的。如果不将ios类作为其派生类的虚基类，可能会产生二义性。
2. 文件缓冲区是为了让低速的输入输出设备和高速的用户程序能够协调工作，并降低输入输出设备的读写次数。显式地关闭文件，正是为了将缓冲区写入文件中。

## 二. 编程题

1.
```cpp
template<class Type> class Matrix;

class Complex{
    double real, imag;
public:
    Complex(){
        real = 0;
        imag = 0;
    }
    Complex(double r, double i){
        real = r;
        imag = i;
    }
    bool operator ==(const Complex& x) const{
        return (real == x.real) && (imag == x.imag);
    }
    bool operator !=(const Complex& x) const{
        return (real != x.real) || (imag != x.imag);
    }
    Complex operator +(const Complex& x){
        return Complex(real + x.real, imag + x.imag);
    }
    Complex operator *(const Complex& x){
        return Complex(real * x.real- imag * x.imag, real * x.imag + imag * x.real);
    }
    friend ostream &operator<<(ostream &output, const Complex &C){
        output<<C.real<<"+"<<C.imag<<"i";
        return output;
    }
    friend istream &operator>>(istream &input, Complex &C){
        char tmp;
        input>>C.real>>tmp>>C.imag;
        return input;
    }
};

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

template<class Type> class Matrix{
    Array<Type> *p_data;
    int row, col;
    void init(int r, int c){
        row = r;
        col = c;
        p_data = new Array<Type>[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<Type> &operator[] (int i){
        return p_data[i];
    }
    Matrix<Type> &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++){
                Type 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<Type> 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<Type> 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++){
                Type sum;
                for(int p=0;p<col;p++){
                    if(p==0)
                        sum = p_data[i][p] * m.p_data[p][j];
                    else
                        sum = sum + p_data[i][p] * m.p_data[p][j];
                }
                tmp[i][j] = sum;
            }
        }
        return tmp;
    }
    friend ostream &operator<<(ostream &output, const Matrix<Type> &M){
        output<<M.row<<" "<<M.col<<endl;
        for(int i=0;i<M.row;i++){
            for(int j=0;j<M.col;j++){
                cout<<M.p_data[i][j]<<" ";
            }
            cout<<endl;
        }
        return output;
    }
    friend istream &operator>>(istream &input, Matrix<Type> &M){
        char tmp;
        input>>M.row>>M.col;
        M.init(M.row, M.col);
        Complex c;
        for(int i=0;i<M.row;i++){
            for(int j=0;j<M.col;j++){
                input>>M.p_data[i][j];
            }
        }
        return input;
    }
};

int main(){
    Complex c;
    cin>>c;
    cout<<c<<endl;
    Matrix<Complex> a;
    ifstream in_file("./matrix_test.txt",ios::in);
    if(!in_file) exit(-1);
    for(int i=0;i<3;i++){
        in_file>>a;
        cout<<a;
    }
    in_file.close();
    return 0;
}
```

测试用例如下:

```
2 3
1+1 2+1 3+1
2+3 4+2 5+3

3 2
1+1 2+1
2+3 4+2
3+1 5+3

1 3
1+1 2+1 3+1

1 1
2+3
```

2.
```cpp
int main(){
    int x;
    ofstream out_file("./number.txt",ios::out);
    if(out_file.fail()) exit(-1);
    cout<<"Generated Number ended with 0:"<<endl;
    for(int i=0;i<1000;i++){
        x = rand()%100+1;
        out_file<<x;
        if(x%10 == 0) cout<<x<<" ";
    }
    cout<<endl;
    out_file<<endl;
    out_file.close();
    ifstream file("./number.txt",ios::in| ios::ate);
    if(file.fail()) exit(-1);
    char t[5]="\0";
    int i=0;
    cout<<"Found Number ended with 0:"<<endl;
    while(true){
        file.seekg(i, ios::beg);
        file.read(t, 2);
        if(t[1]=='\n') break;
        if(t[0]=='0' && t[1]=='0'){
            file.seekg(i-1, ios::beg);
            file.read(t, 3);
            cout<<t<<" ";
            memset(t, '\0', 5);
            i+=2;
        }else if(t[0]=='0'){
            file.seekg(i-1, ios::beg);
            file.read(t, 2);
            cout<<t<<" ";
            memset(t, '\0', 5);
            i++;
        }else{
            i++;
        }
    }
    cout<<endl;
    file.close();
    return 0;
}
```

3.

```
class Grade{
    int id;
    char name[32];
    char sex[3];
    double grade;
public:
    double get_grade(){
        return grade;
    }
    void change_grade(double a){
        grade = a;
    }
    string get_sex(){
        return sex;
    }
    friend ostream &operator<<(ostream &output, const Grade &G){
        output<<G.id<<" "<<G.name<<" "<<G.sex<<" "<<G.grade;
        return output;
    }
    friend istream &operator>>(istream &input, Grade &G){
        input>>G.id>>G.name>>G.sex>>G.grade;
        return input;
    }
};

int main(){
    //1
    ofstream out_file("./a.txt",ios::out);
    if(out_file.fail()) exit(-1);
    Grade g;
    cout<<"输入成绩，每条以两个空格分隔: "<<endl;
    while(true){
        cin>>g;
        out_file<<g;
```

```cpp
        if(cin.get() == '\n') break;
        else out_file<<endl;
    }
    out_file.close();
    //2
    ifstream in_file("./a.txt",ios::in);
    if(in_file.fail()) exit(-1);
    vector<Grade> grades;
    cout<<"Load Grades from a.txt"<<endl;
    while(!in_file.eof()){
        in_file>>g;
        grades.push_back(g);
    }
    in_file.close();
    //3
    sort(grades.begin(), grades.end(), [](Grade &g1, Grade &g2){return g1.get_grade()>g2.get_grade();});
    cout<<"Write to b.txt"<<endl;
    ofstream b("./b.txt", ios::out);
    if(b.fail()) exit(-1);
    for_each(grades.begin(), grades.size()>3? grades.begin()+3:grades.end(), [&](Grade &g){b<<g<<endl;});
    b.close();
    //4
    vector<Grade> male, female;
    copy_if(grades.begin(), grades.end(), back_inserter(male), [](Grade &g){return g.get_sex()=="男";});
    copy_if(grades.begin(), grades.end(), back_inserter(female), [](Grade &g){return g.get_sex()=="女";});
    double mean_male = (double)accumulate(male.begin(), male.end(), 0, [](double partial, Grade &g)->double{ return partial+g.get_grade();})/male.size();
    double mean_female = (double)accumulate(female.begin(), female.end(), 0, [](double partial, Grade &g)->double{ return partial+g.get_grade();})/female.size();
    cout<<"Write to c.txt"<<endl;
    ofstream c("./c.txt", ios::out);
    if(c.fail()) exit(-1);
    for_each(male.begin(), male.end(), [&](Grade &g){if(g.get_grade()<mean_male) c<<g<<endl;});
    for_each(female.begin(), female.end(), [&](Grade &g){if(g.get_grade()<mean_female) c<<g<<endl;});
    c.close();
    //5
    vector<Grade> makeup;
    cout<<"输入补考成绩，每条以两个空格分隔："<<endl;
    while(true){
        cin>>g;
        makeup.push_back(g);
        if(cin.get() == '\n') break;
    }
    for_each(makeup.begin(), makeup.end(), [](Grade &g){g.change_grade(g.get_grade()*0.9);});
    //6
    cout<<"Write to a.txt"<<endl;
    ofstream a("./a.txt",ios::app);
    if(a.fail()) exit(-1);
    for_each(makeup.begin(), makeup.end(), [&](Grade &g){a<<endl<<g;});
    a.close();
    return 0;
}
```