

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. 1  template<class Type> class Matrix;
2
3  class Complex{
4      double real, imag;
5  public:
6      Complex(){
7          real = 0;
8          imag = 0;
9      }
10     Complex(double r, double i){
11         real = r;
12         imag = i;
13     }
14     bool operator ==(const Complex& x) const{
15         return (real == x.real) && (imag == x.imag);
16     }
17     bool operator !=(const Complex& x) const{
18         return (real != x.real) || (imag != x.imag);
19     }
20     Complex operator +(const Complex& x){
21         return Complex(real + x.real, imag + x.imag);
22     }
23     Complex operator *(const Complex& x){
24         return Complex(real * x.real- imag * x.imag, real * x.imag +
25             imag * x.real);
26     }
27     friend ostream &operator<<(ostream &output, const Complex &C){
28         output<<C.real<<" "<<C.imag<<"i";
29         return output;
30     }
31     friend istream &operator>>(istream &input, Complex &C){
32         char tmp;
33         input>>C.real>>tmp>>C.imag;
34         return input;
35     }
36 };
37
38 template<class Type> class Array{
39     Type *data;
```

```

39     int length;
40 public:
41     Array(){
42         length = 0;
43     }
44     Array(int c){
45         set(c);
46     }
47     ~Array(){
48         delete []data;
49     }
50     void set(int c){
51         length = c;
52         data = new Type[c];
53     }
54     Type &operator[] (int j){
55         return data[j];
56     }
57     friend class Matrix<Type>;
58 };
59
60 template<class Type> class Matrix{
61     Array<Type> *p_data;
62     int row, col;
63     void init(int r, int c){
64         row = r;
65         col = c;
66         p_data = new Array<Type>[r];
67         for(int i=0;i<row;i++){
68             p_data[i].set(c);
69         }
70     }
71 public:
72     Matrix(){
73         row = col = 0;
74         p_data = NULL;
75     }
76     Matrix(int r, int c){
77         init(r, c);
78     }
79     ~Matrix(){
80         delete []p_data;
81     }
82     Array<Type> &operator[] (int i){
83         return p_data[i];
84     }
85     Matrix<Type> &operator= (const Matrix &m){
86         if(&m == this) return *this;
87         if(row != m.row || col != m.col){
88             delete []p_data;
89             init(m.row, m.col);
90         }
91         for(int i=0;i<row;i++){
92             for(int j=0;j<col;j++){
93                 Type tmp = m.p_data[i].data[j] ;
94                 p_data[i].data[j] = tmp;
95             }
96         }

```

```

97         return *this;
98     }
99     bool operator==(const Matrix &m) const{
100         if(row != m.row || col != m.col)
101             return false;
102         for(int i=0;i<row;i++){
103             for(int j=0;j<col;j++){
104                 if(p_data[i][j]!=m.p_data[i][j])
105                     return false;
106             }
107         }
108         return true;
109     }
110     Matrix<Type> operator+ (const Matrix &m) const{
111         Matrix tmp(row, col);
112         for(int i=0;i<row;i++){
113             for(int j=0;j<col;j++){
114                 tmp[i][j] = p_data[i][j] + m.p_data[i][j];
115             }
116         }
117         return tmp;
118     }
119     Matrix<Type> operator* (const Matrix &m) const{
120         Matrix tmp(this->row, m.col);
121         for(int i=0;i<tmp.row;i++){
122             for(int j=0;j<tmp.col;j++){
123                 Type sum;
124                 for(int p=0;p<col;p++){
125                     if(p==0)
126                         sum = p_data[i][p] * m.p_data[p][j];
127                     else
128                         sum = sum + p_data[i][p] * m.p_data[p][j];
129                 }
130                 tmp[i][j] = sum;
131             }
132         }
133         return tmp;
134     }
135     friend ostream &operator<<(ostream &output, const Matrix<Type> &M){
136         output<<M.row<<" "<<M.col<<endl;
137         for(int i=0;i<M.row;i++){
138             for(int j=0;j<M.col;j++){
139                 cout<<M.p_data[i][j]<<" ";
140             }
141             cout<<endl;
142         }
143         return output;
144     }
145     friend istream &operator>>(istream &input, Matrix<Type> &M){
146         char tmp;
147         input>>M.row>>M.col;
148         M.init(M.row, M.col);
149         Complex c;
150         for(int i=0;i<M.row;i++){
151             for(int j=0;j<M.col;j++){
152                 input>>M.p_data[i][j];
153             }
154         }

```

```

155         return input;
156     }
157 };
158
159 int main(){
160     Complex c;
161     cin>>c;
162     cout<<c<<endl;
163     Matrix<Complex> a;
164     ifstream in_file("./matrix_test.txt",ios::in);
165     if(!in_file) exit(-1);
166     for(int i=0;i<3;i++){
167         in_file>>a;
168         cout<<a;
169     }
170     in_file.close();
171     return 0;
172 }

```

测试用例如下:

```

1  2 3
2  1+1 2+1 3+1
3  2+3 4+2 5+3
4
5  3 2
6  1+1 2+1
7  2+3 4+2
8  3+1 5+3
9
10 1 3
11 1+1 2+1 3+1
12
13 1 1
14 2+3

```

2.

```

1  int main(){
2      int x;
3      ofstream out_file("./number.txt",ios::out);
4      if(out_file.fail()) exit(-1);
5      cout<<"Generated Number ended with 0:"<<endl;
6      for(int i=0;i<1000;i++){
7          x = rand()%100+1;
8          out_file<<x;
9          if(x%10 == 0) cout<<x<<" ";
10     }
11     cout<<endl;
12     out_file<<endl;
13     out_file.close();
14     ifstream file("./number.txt",ios::in| ios::ate);
15     if(file.fail()) exit(-1);
16     char t[5]="\0";
17     int i=0;
18     cout<<"Found Number ended with 0:"<<endl;
19     while(true){
20         file.seekg(i, ios::beg);

```

```

21     file.read(t, 2);
22     if(t[1]=='\n') break;
23     if(t[0]=='0' && t[1]=='0'){
24         file.seekg(i-1, ios::beg);
25         file.read(t, 3);
26         cout<<t<<" ";
27         memset(t, '\0', 5);
28         i+=2;
29     }else if(t[0]=='0'){
30         file.seekg(i-1, ios::beg);
31         file.read(t, 2);
32         cout<<t<<" ";
33         memset(t, '\0', 5);
34         i++;
35     }else{
36         i++;
37     }
38 }
39 cout<<endl;
40 file.close();
41 return 0;
42 }

```

3.

```

1  class Grade{
2      int id;
3      char name[32];
4      char sex[3];
5      double grade;
6  public:
7      double get_grade(){
8          return grade;
9      }
10     void change_grade(double a){
11         grade = a;
12     }
13     string get_sex(){
14         return sex;
15     }
16     friend ostream &operator<<(ostream &output, const Grade &G){
17         output<<G.id<<" "<<G.name<<" "<<G.sex<<" "<<G.grade;
18         return output;
19     }
20     friend istream &operator>>(istream &input, Grade &G){
21         input>>G.id>>G.name>>G.sex>>G.grade;
22         return input;
23     }
24 };
25
26 int main(){
27     //1
28     ofstream out_file("./a.txt",ios::out);
29     if(out_file.fail()) exit(-1);
30     Grade g;
31     cout<<"输入成绩，每条以两个空格分隔："<<endl;
32     while(true){
33         cin>>g;
34         out_file<<g;

```

```

35         if(cin.get() == '\n') break;
36         else out_file<<endl;
37     }
38     out_file.close();
39     //2
40     ifstream in_file("./a.txt",ios::in);
41     if(in_file.fail()) exit(-1);
42     vector<Grade> grades;
43     cout<<"Load Grades from a.txt"<<endl;
44     while(!in_file.eof()){
45         in_file>>g;
46         grades.push_back(g);
47     }
48     in_file.close();
49     //3
50     sort(grades.begin(), grades.end(), [](Grade &g1, Grade &g2){return
g1.get_grade()>g2.get_grade();});
51     cout<<"Write to b.txt"<<endl;
52     ofstream b("./b.txt", ios::out);
53     if(b.fail()) exit(-1);
54     for_each(grades.begin(), grades.end(), [&](Grade &g){b<<g<<endl;});
55     b.close();
56     //4
57     vector<Grade> male, female;
58     copy_if(grades.begin(), grades.end(), back_inserter(male), [](Grade
&g){return g.get_sex()=="男";});
59     copy_if(grades.begin(), grades.end(), back_inserter(female), [](
Grade &g){return g.get_sex()=="女";});
60     double mean_male = (double)accumulate(male.begin(), male.end(), 0,
[](<double partial, Grade &g)->double{ return
partial+g.get_grade();})/male.size();
61     double mean_female = (double)accumulate(female.begin(),
female.end(), 0, [](double partial, Grade &g)->double{ return
partial+g.get_grade();})/female.size();
62     cout<<"Write to c.txt"<<endl;
63     ofstream c("./c.txt", ios::out);
64     if(c.fail()) exit(-1);
65     for_each(male.begin(), male.end(), [&](Grade &g){if(g.get_grade()
<mean_male) c<<g<<endl;});
66     for_each(female.begin(), female.end(), [&](Grade &g)
{if(g.get_grade()<mean_female) c<<g<<endl;});
67     c.close();
68     //5
69     vector<Grade> makeup;
70     cout<<"输入补考成绩，每条以两个空格分隔："<<endl;
71     while(true){
72         cin>>g;
73         makeup.push_back(g);
74         if(cin.get() == '\n') break;
75     }
76     for_each(makeup.begin(), makeup.end(), [](Grade &g)
{g.change_grade(g.get_grade()*0.9);});
77     //6
78     cout<<"Write to a.txt"<<endl;
79     ofstream a("./a.txt",ios::app);
80     if(a.fail()) exit(-1);
81     for_each(makeup.begin(), makeup.end(), [&](Grade &g){a<<endl<<g;});

```

```
82     a.close();
83     return 0;
84 }
```