Assignment 12

Fu-yun Wang[1]

191300051

一、概念简答题

1.分析说明C++语言的流类库中为什么要将ios类作为其派生类的虚基类.

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类公有派生出istream和ostream类，而iostream类通过多重继承istream和ostream类而产生的。如果不将ios类作为其派生类的虚基类，那么可能会产生二义性。

2.请简要概述文件缓冲区的作用，并结合回答，程序中为什么要显示的关闭文件.

当用cout和插入运算符“<<”向显示器输出数据时，先将这些数据送到程序中的输出缓冲区保存，直到发生了缓冲区刷新的条件，就将缓冲区中的全部数据送到显示器显示出来。在输入时，从键盘输入的数据先放在键盘缓冲区中，当按回车键时，键盘缓冲区中的数据输入到程序中的输入缓冲区，形成cin流，然后用提取运算符“>>”从输入缓冲区中提取数据送给程序中的有关变量。总之，流是与内存缓冲区相对应的，或者说，缓冲区中的数据就是流。

文件缓冲区也是一样，当我们向文件流插入或者抽取数据时，并没有直接将数据存在文件当中，而是暂时的保存在文件缓冲区中，当发生了缓冲区的更新，才真正存入文件当中。

为什么要显示的关闭文件？

- 如果没有调用close，当程序发生异常时，有可能数据并没有被写入文件中，造成数据的丢失。也就是说如果没有刷新缓冲区的话，数据会被保存在缓冲区当中，就不会被真正的写入到文件中。
- 如果操作系统的资源有限，那么不关闭文件会造成系统资源的浪费。

补充：

下列情况都会引发缓冲区的刷新：

- 关闭文件
- endl可以用来完成换行和刷新缓冲区。
- flush可以直接刷新缓冲区
- ends在输入后加入一个空字符，再刷新缓冲区。

我们可以使用成员函数tie使得流对象发生关联：

- 标准库把cin和cout关联在一起，关联 也就是 当一个输入流被关联到一个输出流时，任何 从输入等待 的读取数据时，都会先刷新被关联的那个 输出流。

二、代码编程题

1. 在第九次作业中，我们实现了复数矩阵的运算，本次作业要求增加一些输入输出 接口。具体要求如下：

```
#include <iostream>
#include <cstdio>
#include <cstring>
#include <cassert>
#include <fstream>
using namespace std;

class Complex
{
private:
    double real, imag;

public:
    Complex(double r = 0, double i = 0) : real(r), imag(i)
    {
    }
    void print()
    {
        if (imag == 0)
        {
            cout << real << endl;
        }
        else if (imag > 0)
        {
            cout << real << "+" << imag << "j" << endl;
        }
        else
        {
            cout << real << imag << "j" << endl;
        }
    }
    double modulus() const
    {
        return real * real + imag * imag;
    }
    Complex operator-() const
    {
        Complex tmp;
        tmp.real = -real;
        tmp.imag = -imag;
        return tmp;
    }
    Complex &operator=(const Complex &c)
    {
```

```
        real = c.real;
        imag = c.imag;
        return *this;
    }
    //但是一般这些我们都会传入const的引用，既可以避免修改原值，又可以加快速度。
    friend bool operator==(const Complex &c1, const Complex &c2);
    friend bool operator!=(const Complex &c1, const Complex &c2);
    friend bool operator>(const Complex &c1, const Complex &c2);
    friend bool operator>=(const Complex &c1, const Complex &c2);
    friend bool operator<(const Complex &c1, const Complex &c2);
    friend bool operator<=(const Complex &c1, const Complex &c2);
    friend Complex operator+(const Complex &c1, const Complex &c2);
    friend Complex operator-(const Complex &c1, const Complex &c2);
    friend Complex operator*(const Complex &c1, const Complex &c2);
    friend Complex operator/(const Complex &c1, const Complex &c2);
    friend ostream &operator<<(ostream &output, const Complex &c);
    friend istream &operator>>(istream &input, Complex &c);
};
bool operator==(const Complex &c1, const Complex &c2)
{
    return c1.real == c2.real && c1.imag == c2.imag;
}
bool operator!=(const Complex &c1, const Complex &c2)
{
    return !(c1 == c2);
}
bool operator>(const Complex &c1, const Complex &c2)
{
    return c1.modulus() > c2.modulus();
}
bool operator>=(const Complex &c1, const Complex &c2)
{
    return c1.modulus() >= c2.modulus();
}
bool operator<(const Complex &c1, const Complex &c2)
{
    return c1.modulus() < c2.modulus();
}
bool operator<=(const Complex &c1, const Complex &c2)
{
    return c1.modulus() <= c2.modulus();
}
Complex operator+(const Complex &c1, const Complex &c2)
{
    return Complex(c1.real + c2.real, c1.imag + c2.imag);
}
Complex operator-(const Complex &c1, const Complex &c2)
{
    return Complex(c1.real - c2.real, c1.imag - c2.imag);
```

```
}
Complex operator*(const Complex &c1, const Complex &c2)
{
    return Complex(c1.real * c2.real - c1.imag * c2.imag, c1.real * c2.imag + c1.imag * c2.real);
}
Complex operator/(const Complex &c1, const Complex &c2)
{
    double d = c2.modulus();
    if (d == 0)
    {
        cerr << "Error in operation / of Complex" << endl;
        exit(-1);
    }
    else
    {
        return Complex((c1.real * c2.real + c1.imag * c2.imag) / d, (c1.imag * c2.real - c1.real * c2.imag) / d);
    }
}
ostream &operator<<(ostream &output, const Complex &c)
{
    int num = c.imag;

    if (num >= 0)
    {
        output << c.real << "+" << c.imag << "i";
    }
    else
    {
        output << c.real << c.imag << "i";
    }

    return output;
}
istream &operator>>(istream &input, Complex &c)
{
    input >> c.real >> c.imag;
}


void testComplex()
{
    Complex a, b, c;
    cin >> a >> b >> c;
    cout << "a<b:" << (a < b) << endl;
    cout << (a * b) << endl;
    cout << (a / b) << endl;
    cout << (a + b) << endl;
```

```
141     cout << (a * c) << endl;
142 
143     //divided by zero exit(-1);
144     // (a / c).print();
145 }
146 
147 template <class T>
148 class Matrix
149 {
150 private:
151     int colNum;
152     int rowNum;
153     T *data;
154 
155 public:
156     Matrix(int r = 0, int c = 0)
157     {
158         colNum = c;
159         rowNum = r;
160         data = new T[colNum * rowNum];
161         memset(data, 0, sizeof(T) * c * r);
162     }
163     Matrix(int r, int c, T a[])
164     {
165         colNum = c;
166         rowNum = r;
167         data = new T[colNum * rowNum];
168         for (int i = 0; i < r * c; i++)
169         {
170             data[i] = a[i];
171         }
172     }
173     void print()
174     {
175         for (int i = 0; i < rowNum; i++)
176         {
177             for (int j = 0; j < colNum; j++)
178             {
179                 cout << data[i * colNum + j] << "  ";
180             }
181             cout << endl;
182         }
183     }
184     Matrix &operator=(const Matrix &m)
185     {
186         colNum = m.colNum;
187         rowNum = m.rowNum;
188         delete data;
189         data = new T[colNum * rowNum];
```

```
        memcpy(data, m.data, sizeof(T) * colNum * rowNum);
        return *this;
    }

    Matrix operator-()
    {
        Matrix m(rowNum, colNum);
        for (int i = 0; i < colNum * rowNum; i++)
        {
            m[i] = -data[i];
        }
    }
    T *operator[](int index)
    {
        return &data[index * colNum];
    }
    template <class Ty>
    friend Matrix<Ty> operator+(const Matrix<Ty> &m1, const
Matrix<Ty> &m2);
    template <class Ty>
    friend Matrix<Ty> operator-(const Matrix<Ty> &m1, const
Matrix<Ty> &m2);
    template <class Ty>
    friend Matrix<Ty> operator*(const Matrix<Ty> &m1, const
Matrix<Ty> &m2);
    template <class Ty>
    friend istream &operator>>(istream &input, Matrix<Ty> &m);
    template <class Ty>
    friend ostream &operator<<(ostream &output, const Matrix<Ty>
&m);
};
template <class T>
Matrix<T> operator+(const Matrix<T> &m1, const Matrix<T> &m2)
{
    assert(m1.colNum == m2.colNum && m1.rowNum == m2.rowNum);
    Matrix<T> tmp(m1.rowNum, m1.colNum);
    for (int i = 0; i < tmp.colNum * tmp.rowNum; i++)
    {
        tmp.data[i] = m1.data[i] + m2.data[i];
    }
    return tmp;
}
template <class T>
Matrix<T> operator-(const Matrix<T> &m1, const Matrix<T> &m2)
{
    assert(m1.colNum == m2.colNum && m1.rowNum == m2.rowNum);
    Matrix<T> tmp(m1.rowNum, m1.colNum);
    for (int i = 0; i < tmp.colNum * tmp.rowNum; i++)
    {
```

```
235         tmp.data[i] = m1.data[i] - m2.data[i];
236     }
237     return tmp;
238 }
239 template <class T>
240 Matrix<T> operator*(const Matrix<T> &m1, const Matrix<T> &m2)
241 {
242     assert(m1.colNum == m2.rowNum);
243     Matrix<T> tmp(m1.rowNum, m2.colNum);
244     for (int i = 0; i < tmp.rowNum * tmp.colNum; i++)
245     {
246         int r = i / tmp.colNum;
247         int c = i % tmp.colNum;
248         for (int j = 0; j < m1.colNum; j++)
249         {
250             tmp.data[i] = tmp.data[i] + m1.data[r * m1.colNum + j] *
    m2.data[j * m2.colNum + c];
251         }
252     }
253     return tmp;
254 }
255 template <class T>
256 istream &operator>>(istream &input, Matrix<T> &m)
257 {
258     int a,b;
259     input>>a>>b;
260     Matrix<T> tmp(a,b);
261     for (int i=0;i<a*b;i++) input>>tmp.data[i];
262     m=tmp;
263     return input;
264 }
265 template <class T>
266 ostream &operator<<(ostream &output, const Matrix<T> &m)
267 {
268     for (int i = 0; i < m.rowNum; i++)
269     {
270         for (int j = 0; j < m.colNum; j++)
271             output << m.data[i * m.colNum + j] << " ";
272         output << endl;
273     }
274     return output;
275 }
276
277 void testMatrix()
278 {
279     Matrix<Complex> m;
280     cin >> m;
281     cout << m;
282 }
```

```
283 void testMatrixFile()
284 {
285     Matrix<Complex> m;
286     ifstream minput("matrix_test.txt",ios::in);
287     if (!minput) exit(-1);
288     minput>>m;
289     cout<<m;
290
291
292 }
293 int main()
294 {
295     // testComplex();
296     // testMatrix();
297     testMatrixFile();
298 }
```

2. 给定一个文件如下

为1-100数字的拼接，首先你需要生成这样的一个文件，然后读取文件，将以0结尾的数字输出出来。

分别使用int型和char型的二进制存储实现了要求。

```
1  #include <iostream>
2  #include <fstream>
3  #include <string>
4  using namespace std;
5
6  //直接以int型保存
7  void testint()
8  {
9      //写入//将1到100的二进制都存在seek.data当中。
10     ofstream out_file("seek.dat",ios::out|ios::binary);
11     if (!out_file)
12     {
13         cerr<<"out_fail!"<<endl;
14         exit(-1);
15     }
16     for (int i=1;i<101;i++)
17     {
18         out_file.write((char*)&i,sizeof(int));
19     }
20     out_file.close();//如果这里不close的话，后面读的就是空文件，因为数据还没有
   从缓冲区传入到文件当中。
21     //读出//将seek.data中的每个数字输出。
22     int num;
23     ifstream in_file("seek.dat",ios::in|ios::binary);
```

```
    if (!in_file)
    {
        cerr<<"is_fail!"<<endl;
        exit(-1);
    }
    // for (int i=0;i<10;i++)
    // {
    //     in_file.seekg((9+i*10)*sizeof(int),ios::beg);
    //     in_file.read((char *)&num,sizeof(int));
    //     cout<<num<<endl;
    // }
    // in_file.close();
    for (int i=0;i<10;i++)
    {
        in_file.seekg(9*sizeof(int),ios::cur);
        in_file.read((char*)&num,sizeof(int));
        cout<<num<<endl;
    }
    in_file.close();
}
void testchar()
{
    string data;
    for (int i=1;i<101;i++)
        data+=std::to_string(i);
    ofstream out_file("seek.dat",ios::out|ios::binary);
    if (!out_file)
    {
        cerr<<"out_fail!"<<endl;
        exit(-1);
    }
    out_file.write(data.c_str(),data.length());
    out_file.close();

    ifstream in_file("seek.dat",ios::in|ios::binary);
    char num[20]="\0";
    for (int i=0;i<9;i++)
    {
        in_file.seekg(9+i*20,ios::beg);
        in_file.read(num,2);
        cout<<num<<endl;
    }
    in_file.seekg(18,ios::cur);
    in_file.read(num,3);
    cout<<num<<endl;


    return ;
}
```

```
int main()
{
    testint();
    testchar();
}
```

3. 某课程G需要将考试成绩录进系统并进行相应的保存，需要实现一个简单的成绩 表管理系统

```
#include <iostream>
#include <fstream>
#include <cstring>
#include <vector>
#include <algorithm>
using namespace std;

enum Gender {Male,Female};
class Grade
{
    private:
    int id;
    char name[10];
    Gender sex;
    int grade;
    public:
    Grade(int i=0,char n[]=NULL,Gender s=Male,int g=0):
    id(i),sex(s),grade(g)
    {
        // strcpy(name,n);
        if (n==NULL)
        {
            name[0]='\0';
        }
        else
        {
            strcpy(name,n);
        }
    }
    int getGrade() const
    {
        return grade;
    }
    Gender getGender() const
    {
        return sex;
    }
    void setGrade(int g)
```

```
    {
        grade=g;
        return ;
    }
    friend istream &operator>>(istream &input,Grade &gd)
    {
        input>>gd.id;
        input>>gd.name;
        int tmp;
        input>>tmp;
        gd.sex=(Gender)tmp;
        input>>gd.grade;
        return input;
    }
    friend ostream &operator<<(ostream &output,const Grade& gd)
    {
        output<<gd.id<<" "<<gd.name<<" "<<gd.sex<<" "
<<gd.grade<<"\n";
        return output;
    }

};
size_t getFileSize(const std::string& file_name){
    std::ifstream in(file_name.c_str(),ios::in|ios::binary);
    in.seekg(0, std::ios::end);
    size_t size = in.tellg();
    in.close();
    return size; //单位是：byte
}
void init()
{
/*输入测试样例
1 lilei 0 90
2 feifei 1 60
3 gugu 0 70
4 hanhan 1 59
5 haha 1 80
*/
    Grade grades[5];
    for (int i=0;i<5;i++)
    {
        cin>>grades[i];
    }
    for (int i=0;i<5;i++)
    {
        cout<<grades[i];
    }
    ofstream out("a.dat",ios::out|ios::binary);
    for (int i=0;i<5;i++)
```

```
87      {
88          //栈中存储结构体。
89          out.write((char*)&grades[i],sizeof(Grade));
90      }
91
92  }
93  void saveTop3(vector<Grade*>);
94  void saveLowerAverage(const vector<Grade *> & grades);
95  void addMakeUp(vector<Grade *>&grades);
96  void change()
97  {
98      vector<Grade*> grades;
99      int num=getFileSize("a.dat")/sizeof(Grade);
100     ifstream in("a.dat",ios::in|ios::binary);
101     for (int i=0;i<num;i++)
102     {
103         Grade *newgrade=new Grade;
104         in.read((char *)newgrade,sizeof(Grade));
105         grades.push_back(newgrade);
106     }
107     in.close();
108     //不改变原来的vector。
109     cout<<"Test1"<<endl;
110     saveTop3(grades);
111     cout<<"Test2"<<endl;
112     saveLowerAverage(grades);
113     cout<<"Test3"<<endl;
114     addMakeUp(grades);
115 }
116 void saveTop3(vector<Grade*> grades)
117 {
118     sort(grades.begin(),grades.end(),[](Grade *gd1,Grade *gd2)->bool
    {if (gd1->getGrade()>gd2->getGrade())return true;return false; });
119     ofstream out("b.dat",ios::binary|ios::out);
120     for_each(grades.begin(),grades.begin()+3,[&out](Grade *gd)->void
    {out.write((char*)gd,sizeof(Grade));});
121     for_each(grades.begin(),grades.begin()+3,[](Grade *gd)-
    >void{cout<<*gd;});
122     out.close();
123 }
124 void saveLowerAverage(const vector<Grade *> & grades)
125 {
126     vector<Grade *>mg,fg;
127
128     copy_if(grades.begin(),grades.end(),back_inserter(mg),[](Grade
    *gd)->bool{return gd->getGender()==Male;});
129     copy_if(grades.begin(),grades.end(),back_inserter(fg),[](Grade
    *gd)->bool{return gd->getGender()==Female;});
```

```
130     int mgave=accumulate(mg.begin(),mg.end(),0,[](double partial,
    Grade *gd)->double{return partial+gd-
    >getGrade();})/(double)mg.size();
131     int fgave=accumulate(fg.begin(),fg.end(),0,[](double partial ,
    Grade*gd)->double{return partial+gd-
    >getGrade();})/(double)fg.size();
132
133     ofstream out("c.dat",ios::binary|ios::out);
134     for_each(mg.begin(),mg.end(),[&out,mgave](Grade *gd)->void{if
    (gd->getGrade()<mgave) out.write((char*)gd,sizeof(Grade));});
135     for_each(fg.begin(),fg.end(),[&out,fgave](Grade *gd)->void{if
    (gd->getGrade()<fgave) out.write((char*)gd,sizeof(Grade));});
136     out.close();
137     for_each(mg.begin(),mg.end(),[mgave](Grade *gd)->void{if (gd-
    >getGrade()<mgave)cout<<*gd;});
138     for_each(fg.begin(),fg.end(),[fgave](Grade *gd)->void{if (gd-
    >getGrade()<fgave)cout<<*gd;});
139 }
140 void addMakeUp(vector<Grade *>&grades)
141 {
142 /*测试样例
143 6 guagua 0 10
144 7 lala  1 20
145 8 sisi 0 30
146 9 sasa 1 40
147 10 tutu 0 50
148
149 */
150     int num=0;
151     cin>>num;
152     for (int i=0;i<num;i++)
153     {
154         Grade *newgrade=new Grade;
155         cin>>*newgrade;
156         newgrade->setGrade(newgrade->getGrade()*0.9);
157         grades.push_back(newgrade);
158     }
159     ofstream out("a.dat",ios::out|ios::binary);
160     for_each(grades.begin(),grades.end(),[&out](Grade *gd)->void
    {out.write((char*)gd,sizeof(Grade));});
161     for_each(grades.begin(),grades.end(),[](Grade *gd)-
    >void{cout<<*gd;});
162 }
163
164 //想要STL不报错，就要重写c_cpp_properties
165 //想要默认用vscode的编辑选项，就要编写task，ctrl+shift+b.
166 //lauch.json用于调试等等。
167 int main()
168 {
```

```
169         init();
170         change();
171     }
```