# 第十次作业：

## 实现了所需要的功能，可能在细节上存在一定的偏差，但整体的功能是有了。

## 存在的问题：类的封装和设计不够好，没能够用到合适的宏来对不同的类型的变量进行。

具体实现：

这里采用了将图像化处理和文件处理分开的方式，主要设计了interface类和VectorOfSome类之间的操作，再连接上之前的People，Student，Graduate类进行实现。

但依旧有很多可以优化的地方没有处理到位。

提交内容：

各部分的．ｃｐｐ文件和．ｈ文件

测试用例ｍａｉｎ＿ｔｅｓｔ．ｃｐｐ

//interface.h

#pragma once

#ifndef INTERFACE\_H

#define INTERFACE\_H

#include<iostream>

#include<string>

#include"Personnel\_Controller.h"

using namespace std;

class Core\_interface

{

private:

int interface\_state;

VectorOfSome vector\_temp;

public:

//构造函数与默认构造函数`

//Core\_interface(int state):interface\_state(state){}

Core\_interface() { interface\_state = 0; }

//打印界面的函数

void interface\_Init();

//界面控制的链接函数

void interface\_controller();

void interface\_controller(int);

};

//有关打印的一些函数（内容有待完善）

void interface\_00();

void interface\_01();

void interface\_02();

void interface\_03();

#endif // !INTERFACE\_H

//interface.cpp

#include "interface.h"

//打印不同界面的图案

void Core\_interface::interface\_Init()

{

switch (interface\_state)

{

case(0):

system("cls"); interface\_00(); break;

case(1):

{

system("cls");

interface\_01();

interface\_state = vector\_temp.Select\_type();

break;

}

case(2):

{

system("cls");

interface\_02();

interface\_state = vector\_temp.Search\_info();

break;

}

case(3):

{

system("cls");

interface\_03();

interface\_state = vector\_temp.Change\_info();

break;

}

default:

break;

}

}

//实现界面的转换

void Core\_interface::interface\_controller()

{

interface\_state = cin.get() - '0';

interface\_Init();

}

void Core\_interface::interface\_controller(int temp)

{

interface\_state = temp;

interface\_Init();

}

void interface\_00()

{

cout << "+++++++++++++++++++++++++++++++++++++++" << endl << endl;

cout << " 华清大学人事管理系统 " << endl << endl;

cout << "+++++++++++++++++++++++++++++++++++++++" << endl << endl;

cout << "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" << endl << endl;

cout << " 主菜单 " << endl;

cout << " 1、数据录入 " << endl;

cout << " 2、数据查询 " << endl;

cout << " 3、数据修改 " << endl;

cout << " 4、人员查询 " << endl;

cout << " 5、退出 " << endl;

cout << " 请选择序号（1-4） " << endl << endl;

cout << "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" << endl << endl;

}

void interface\_01()

{

}

void interface\_02()

{

}

void interface\_03()

{

}

//Personnnel\_Controller.h

#pragma once

#ifndef PERSONNEL\_CONTROLLER\_H

#define PERSONNEL\_CONTROLLER\_H

#define \_CRT\_SECURE\_NO\_WARNINGS

#include<iostream>

#include<vector>

#include<Windows.h>

#include<assert.h>

#include<string>

#include <fstream>

#include <sstream>

#include <cstdio>

using namespace std;

//#include"interface.h"

class Core\_interface;

class Birthday

{

private:

public:

int year, month, day;

Birthday() {}

Birthday(int yy, int mm, int dd)

{

year = yy;

month = mm;

day = dd;

}

void BirthdayPrint()

{

cout << "生日：" << year << "/" << month << "/" << day << endl;

}

};

class People

{

public:

Birthday birthday;

friend ostream& operator<<(ostream&, People&);

friend istream& operator>>(istream&, People&);

People() {}

People(string nn, string num, string ss, int yy, int mm, int dd, string id) : birthday(yy, mm, dd)

{

name = nn;

number = num;

sex = ss;

ID = id;

}

friend void save\_people(vector<People> temp\_people, int temp\_num, const string& save\_name);

//private内容的调用

string GetPrivate\_people(string content);

private:

string name;

string number;

string sex;

string ID;

};

class Student:public People

{

private:

string classNo;

public:

friend ostream& operator<<(ostream&, Student&);

friend istream& operator>>(istream&, Student&);

Student() {}

Student(string nn, string num, string ss, int yy, int mm, int dd, string id, string classno) :People(nn, num, ss, yy, mm, dd, id), classNo(classno) {}

friend void save\_student(vector<Student> temp\_student, int temp\_num, const string& save\_name);

//private内容的调用

string GetPrivate\_student(string content);

};

class Graduate :public Student

{

private:

string subject;

string teacher\_advisor;

public:

friend ostream& operator<<(ostream&,Graduate&);

friend istream& operator>>(istream&, Graduate&);

Graduate() {}

Graduate(string nn, string num, string ss, int yy, int mm, int dd, string id, string classno,string sub,string ta) :Student(nn, num, ss, yy, mm, dd, id,classno)\

,subject(sub),teacher\_advisor(ta){}

friend void save\_graduate(vector<Graduate> temp\_graduate, int temp\_num, const string& save\_name);

};

//使用Vector进行数据处理的类

class VectorOfSome

{

private:

string vector\_type;

int vector\_number;

Core\_interface\* core\_interface;

public:

VectorOfSome() {}

VectorOfSome(Core\_interface\* cointerface,string type,int number):vector\_type(type),vector\_number(number),core\_interface(cointerface) {}

//与数据录入相关联的

int Select\_type();

//与数据查询相关联的

int Search\_info();

//与数据修改相关联的

int Change\_info();

void Number\_check(const string &search\_name,string searchname);

int target\_num\_1 = 0;

int target\_num\_2 = 0;

void Change\_Line(const string& search\_name, string searchname,int stack);

//vectorOfSome中可能会用的创建函数

//vector构建函数：

void construct\_people();

void construct\_student();

void construct\_graduate();

};

//静态空变量

static People EMPTY\_people = { "empty","empty","empty",0,0,0,"empty" };

static Student EMPTY\_student = { "empty","empty","empty",0,0,0,"empty","empty" };

static Graduate EMPTY\_graduate = { "empty","empty","empty",0,0,0,"empty","empty","empty","empty" };

//定义全局的宏变量，设定为各个类中变量的参数

//文件处理有关的函数

//（PS：这里偷懒就没有采用类来进行封装）

string readFile(const string&);

string checkFile(const string&);

string LastWord(string); //用于得到单行末尾的单词

int LineCount(const string&); //用于计算文件中的行数

void ModifyLineData(const string& fileName, int lineNum, char\* lineData);

string CharToStr(char\* contentChar);

#endif // !PERSONNEL\_CONTROLLER\_H

//Personnel\_Controller.cpp

#define \_CRT\_SECURE\_NO\_WARNINGS

#include"interface.h"

#include<fstream>

//People类对于cin、cout的重载

ostream& operator<<(ostream&, People& people)

{

cout << "姓名：" << people.name << endl;

cout << "编号：" << people.number << endl;

cout << "性别：" << people.sex << endl;

people.birthday.BirthdayPrint();

cout << "ID：" << people.ID << endl;

return cout;

}

istream& operator>>(istream&, People& people)

{

cout << "请输入姓名：" << endl;

cin >> people.name;

cout << "请输入性别：" << endl;

cin >> people.sex;

cout << "请输入编号：" << endl;

cin >> people.number;

cout << "请输入生日：" << endl;

int yy, mm, dd;

cin >> yy >> mm >> dd;

people.birthday = { yy,mm,dd };

cout << "请输入ID：" << endl;

cin >> people.ID;

return cin;

}

//调取People类中的内容

string People::GetPrivate\_people(string content)

{

if (content == "name")

return name;

else if (content == "sex")

return sex;

else if (content == "number")

return number;

else if (content == "ID")

return ID;

else { cout << "没有这个成分" << endl; exit(0); }

}

//student类对于cin、cout的重载

ostream& operator<<(ostream&, Student& student)

{

cout << (People&)student;

cout << "班号：" << student.classNo << endl;

return cout;

}

istream& operator>>(istream&, Student& student)

{

cin >> (People&)student;

cout << "请输入班号：" << endl;

cin >> student.classNo;

return cin;

}

//调取Student类中的内容

string Student::GetPrivate\_student(string content)

{

if (content == "classNo")

return classNo;

else { cout << "没有这个成分" << endl; exit(0); }

}

//graduate类对于cin、cout的重载

ostream& operator<<(ostream&, Graduate& graduate)

{

cout << (Student&)graduate;

cout << "专业：" << graduate.subject << endl;

cout << "导师：" << graduate.teacher\_advisor << endl;

return cout;

}

istream& operator>>(istream&, Graduate& graduate)

{

cin >> (Student&)graduate;

cout << "请输入专业：" << endl;

cin >> graduate.subject;

cout << "请输入导师名：" << endl;

cin >> graduate.teacher\_advisor;

return cin;

}

//筛选要形成的vector的类型，并选择是否进行后续操作

int VectorOfSome::Select\_type()

{

cout << "请输入索要创建的人员清单的类型：" << endl;

cout << "（提示：目前已包括的类型有people、student）" << endl;

cin >> vector\_type;

cout << "请输入所要创建的清单中成员的数量：" << endl;

cin >> vector\_number;

system("cls");

//筛选vector\_type

//cout << vector\_number;

if (vector\_type == "people") construct\_people();

else if (vector\_type == "student")construct\_student();

else if (vector\_type == "graduate")construct\_graduate();

else { cout << "暂不支持该种类型的成员！" << endl; }

return 0;

}

int VectorOfSome::Search\_info()

{

cout << "请输入要打开的文件的名：" << endl;

cout << "（提示：文件名均为.txt格式）" << endl;

//cout << "(提示：输入quit以返回主界面)" << endl;

string search\_name;

cin >> search\_name;

ifstream search((const string)search\_name);

//判断是否能够打开相对应的文件

if(search.good())

{

//search.close();

cout << endl << "成功打开目标文件 " << search\_name << endl;

cout << "是否读取该文件中的内容" << endl;

cout<< "(Tips:输入y读取，输入n放弃读取)" << endl;

getchar();

if (cin.get() == 'y')

{

cout << readFile((const string)search\_name) << endl;

cout << "（按任意键返回）" << endl;

getchar();

getchar();

}

else{}

}

else

{

cout << "所找的文件不存在" << endl; Sleep(1000);

}

;

return 0;

}

int VectorOfSome::Change\_info()

{

cout << "请输入要打开的文件的名：" << endl;

cout << "（提示：文件名均为.txt格式）" << endl;

string search\_name;

cin >> search\_name;

ifstream search((const string)search\_name);

//判断是否能够打开相对应的文件

if (search.good())

{

//search.close();

cout << endl << "成功打开目标文件 " << search\_name << endl;

cout << "是否读取该文件中的内容" << endl;

cout << "(Tips:输入y读取，输入n放弃读取)" << endl;

getchar();

if (cin.get() == 'y')

{

cout << readFile((const string)search\_name) << endl;

string search\_type = checkFile((const string)search\_name);

cout << search\_type << endl; //测试是否能够检测到search\_type

Number\_check(search\_name, search\_type);

}

else {}

}

else

{

cout << "所找的文件不存在" << endl; Sleep(1000);

}

;

return 0;

}

void VectorOfSome::Number\_check(const string& search\_name,string search\_type)

{

int stack;

string a[100];

if (search\_type == "people.") stack = 5;

else if (search\_type == "student.")stack = 6;

else if (search\_type == "graduate.")stack = 8;

else

{

cout << "无法对该种类型的文件进行修改" << endl;

exit(0);

}

int linecount = LineCount(search\_name);

//录入相对应的编号进入表格中：

int i = 0; //使用i来进行行数的统计

int j = 0; //使用j来进行编号的统计

string temp;

ifstream fin(search\_name);

while(getline(fin,temp))

{

i++;

if ((i - 4) % (stack + 1) == 0)

{

a[j] = LastWord(temp);

j++;

}

}

fin.close();

cout << "其中成员的编号有：" << endl;

for (int k = 0; k < j; k++)

{

cout << a[k] << endl;

}

//进行相对应编号的确定

cout << "请输入所要修改的目标的编号" << endl;

string search\_num;

cin >> search\_num;

for (int k = 0; k < j; k++)

{

if (search\_num == a[k])

{

target\_num\_1 = k;

break;

}

}

system("cls");

//打印出所选的target所对应的部分的信息

ifstream Fin(search\_name);

int I = 0,I\_head= target\_num\_1 \* (stack + 1) + 2,I\_end= (target\_num\_1 + 1) \* (stack + 1) + 2;

while (getline(Fin, temp))

{

I++;

if ((I > I\_head) && (I < I\_end))

{

cout << I-I\_head<<". " << temp << endl;

}

}

cout << "请选择所要修改的内容：" << endl;

cout << "（提示：输入所需要修改的内容之前的编号）" << endl;

cin >> target\_num\_2;

Change\_Line(search\_name, search\_type, I\_head);

}

void VectorOfSome::Change\_Line(const string& search\_name, string search\_type,int head)

{

cout << "请输入修改后的内容：" << endl;

cout << "（提示：请注意加上之前的标头，如：姓名：）" << endl;

char Changestack[100];

getchar();

cin.getline(Changestack, 100);

ModifyLineData(search\_name, (head+ target\_num\_2), Changestack);

}

void VectorOfSome::construct\_people()

{

//创建临时的vector<people>用于存储

vector<People> people;

cout << "创建成功!" << endl;

cout << "(Tips:输入y开始填写资料，输入n取消创建)" << endl;

getchar();

if (cin.get() == 'y')

{

for (int i = 0; i < vector\_number; i++)

{

cout << "对于第" << i + 1 << "个员工" << endl;

People Instance = EMPTY\_people;

cin >> Instance;

people.push\_back(Instance);

system("cls");

}

cout << "填写完毕!" << endl;

cout << "(Tips:输入y开始资料的保存，输入n取消保存)" << endl;

getchar();

if (cin.get() == 'y')

{

cout << "请输入所要存储的文件名：" << endl;

cout << "（Tips：文件将被自动保存为.txt格式）" << endl;

string save\_name;

cin >> save\_name;

save\_people(people, vector\_number, (const string)save\_name);

cout << "保存成功！" << endl;

Sleep(500);

//core\_interface->interface\_controller(0);

}

else if (cin.get() == 'n') { delete& people; core\_interface->interface\_controller(0); }

}

else if (cin.get() == 'n') { delete& people; core\_interface->interface\_controller(0); }

}

void VectorOfSome::construct\_student()

{

//创建临时的vector<student>用于存储

vector<Student> student;

cout << "创建成功!" << endl;

cout << "(Tips:输入y开始填写资料，输入n取消创建)" << endl;

getchar();

if (cin.get() == 'y')

{

for (int i = 0; i < vector\_number; i++)

{

cout << "对于第" << i + 1 << "个学生" << endl;

Student Instance = EMPTY\_student;

cin >> Instance;

student.push\_back(Instance);

system("cls");

}

cout << "填写完毕!" << endl;

cout << "(Tips:输入y开始资料的保存，输入n取消保存)" << endl;

getchar();

if (cin.get() == 'y')

{

cout << "请输入索要存储的文件名：" << endl;

cout << "（Tips：文件将被自动保存为.txt格式,请记得在末尾输入.txt）" << endl;

string save\_name;

cin >> save\_name;

save\_student(student, vector\_number, (const string)save\_name);

cout << "保存成功！" << endl;

Sleep(500);

//core\_interface->interface\_controller(0);

}

else if (cin.get() == 'n') { delete& student; core\_interface->interface\_controller(0); }

}

else if (cin.get() == 'n') { delete& student; core\_interface->interface\_controller(0); }

}

void VectorOfSome::construct\_graduate()

{

//创建临时的vector<graduate>用于存储

vector<Graduate> graduate;

cout << "创建成功!" << endl;

cout << "(Tips:输入y开始填写资料，输入n取消创建)" << endl;

getchar();

if (cin.get() == 'y')

{

for (int i = 0; i < vector\_number; i++)

{

cout << "对于第" << i + 1 << "个学生" << endl;

Graduate Instance = EMPTY\_graduate;

cin >> Instance;

graduate.push\_back(Instance);

system("cls");

}

cout << "填写完毕!" << endl;

cout << "(Tips:输入y开始资料的保存，输入n取消保存)" << endl;

getchar();

if (cin.get() == 'y')

{

cout << "请输入索要存储的文件名：" << endl;

cout << "（Tips：文件将被自动保存为.txt格式）" << endl;

string save\_name;

cin >> save\_name;

save\_graduate(graduate, vector\_number, (const string)save\_name);

cout << "保存成功！" << endl;

Sleep(500);

//core\_interface->interface\_controller(0);

}

else if (cin.get() == 'n') { delete& graduate; core\_interface->interface\_controller(0); }

}

else if (cin.get() == 'n') { delete& graduate; core\_interface->interface\_controller(0); }

}

void save\_people(vector<People> temp\_people, int temp\_num, const string& save\_name)

{

ofstream out(save\_name);

if (out.is\_open())

{

//声明存储的内容

out << "This a list of "<<temp\_num <<" people." << endl;

for (int i = 0; i < temp\_num; i++)

{

out << "People " << i+1 << endl;

out <<"姓名： "<< temp\_people[i].name << endl;

out <<"编号： "<< temp\_people[i].number << endl;

out <<"性别： "<< temp\_people[i].sex << endl;

out <<"生日： "<< temp\_people[i].birthday.year << "/" << temp\_people[i].birthday.month << "/" << temp\_people[i].birthday.day << endl;

out <<"ID： "<< temp\_people[i].ID << endl;

}

out.close();

}

else { cout << "存储失败" << endl; }

}

void save\_student(vector<Student> temp\_student, int temp\_num, const string& save\_name)

{

ofstream out(save\_name);

if (out.is\_open())

{

//声明存储的内容

out << "This a list of "<<temp\_num<<" student." << endl;

for (int i = 0; i < temp\_num; i++)

{

out << "Student " << i + 1 << endl;

out <<"姓名： " << temp\_student[i].GetPrivate\_people("name") << endl;

out <<"编号： " << temp\_student[i].GetPrivate\_people("number") << endl;

out <<"性别： " << temp\_student[i].GetPrivate\_people("sex") << endl;

out <<"生日： " << temp\_student[i].birthday.year << "/" << temp\_student[i].birthday.month << "/" << temp\_student[i].birthday.day << endl;

out <<"ID： "<< temp\_student[i].GetPrivate\_people("ID") << endl;

out << "学号：" << temp\_student[i].classNo << endl;

}

out.close();

}

else { cout << "存储失败" << endl; }

}

void save\_graduate(vector<Graduate> temp\_graduate, int temp\_num, const string& save\_name)

{

ofstream out(save\_name);

if (out.is\_open())

{

//声明存储的内容

out << "This a list of "<<temp\_num<<" graduate." << endl;

for (int i = 0; i < temp\_num; i++)

{

out << "Graduate " << i+1 << endl;

out << "姓名： " << temp\_graduate[i].GetPrivate\_people("name") << endl;

out << "编号： " << temp\_graduate[i].GetPrivate\_people("number") << endl;

out << "性别： " << temp\_graduate[i].GetPrivate\_people("sex") << endl;

out << "生日： " << temp\_graduate[i].birthday.year << "/" << temp\_graduate[i].birthday.month << "/" << temp\_graduate[i].birthday.day << endl;

out << "ID： " << temp\_graduate[i].GetPrivate\_people("ID") << endl;

out << "学号： " << temp\_graduate[i].GetPrivate\_student("classNo") << endl;

out << "专业： " << temp\_graduate[i].subject << endl;

out << "导师： " << temp\_graduate[i].teacher\_advisor << endl;

}

out.close();

}

else { cout << "存储失败" << endl; }

}

//文件处理函数：

//文件读取

string readFile(const string& search\_name)

{

ifstream in(search\_name);

stringstream buffer;

if (in.is\_open())

{

buffer << in.rdbuf();

in.close();

}

else {

cout << "无法读取目标文件 " << search\_name << " 中的内容" << endl;

}

return buffer.str();

}

//文件类型识别

string checkFile(const string& search\_name)

{

ifstream in(search\_name);

string check\_line;

if (in.is\_open())

{

getline(in,check\_line);

in.close();

}

return LastWord(check\_line);

}

//末端字符读取

string LastWord(string line)

{

char separator = ' ';

int i = 0;

// Temporary string used to split the string.

string s;

while (line[i] != '\0')

{

if (line[i] != separator)

{

// Append the char to the temp string.

s += line[i];

}

else

{

s.clear();

}

i++;

}

return s;

}

//读取文件中所包含的行数的函数

int LineCount(const string& search\_name)

{

ifstream fin(search\_name);

int i = 0;

string temp;

while (getline(fin, temp))

{

i++;

};

fin.close();

return i;

}

//修改具体的某一行的函数

void ModifyLineData(const string& fileName, int lineNum, char\* lineData)

{

ifstream in;

in.open(fileName);

string strFileData = "";

int line = 1;

char tmpLineData[1024] = { 0 };

while (in.getline(tmpLineData, sizeof(tmpLineData)))

{

if (line == lineNum)

{

strFileData += CharToStr(lineData);

strFileData += "\n";

}

else

{

strFileData += CharToStr(tmpLineData);

strFileData += "\n";

}

line++;

}

in.close();

//写入文件

ofstream out;

out.open(fileName);

out.flush();

out << strFileData;

out.close();

}

//char\* 和string 之间的转化（PS：不知道是否已经存在该种功能的函数）

string CharToStr(char\* contentChar)

{

string tempStr;

for (int i = 0; contentChar[i] != '\0'; i++)

{

tempStr += contentChar[i];

}

return tempStr;

}

//Main\_test.cpp

#include"interface.h"

#include"Personnel\_Controller.h"

int main()

{

Core\_interface personnel\_interface;

VectorOfSome personnel\_vector(&personnel\_interface,"null",0);

personnel\_interface.interface\_Init();

while (true)

{

personnel\_interface.interface\_controller();

//system("cls");

personnel\_interface.interface\_Init();

}

return 0;

}