**分组:\_ \_14\_ \_**



信息科学与工程学院课程实验报告

《面向对象程序设计》

姓名： 张学凯

学号： 201711010503

班级： 1701

教师： 张庆科

时间： 2018.11.23

**面向对象程序设计实验报告**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 姓名 | 张学凯 | 班级 | 1701 | 学号 | 201711010503 | 组号 | 14 |
| 时间 | 2017.11.23 | 地点 | 宿舍 | 周次 | 第 12 周 | 页码 | 22 |
| 源码 | □ 无源码 ☑ 文档源码 ☑ 托管源码 | | | | | | |

# 学生信息管理系统:

项目托管网址：https://github.com/zhangxuekai/c-work

实验目的：了解类中数据的共享与保护

实验内容：通过对第四章节的五个代码的练习及思考，充分的了解静态数据成员与函数和常数据成员 昌数据函数 常对象的使用

实验结论：

**代码一：**

P103：（1）将int Student：：total=0；修改为int Student：：total；重新运行后，结果没有变化。原因：静态数据成员的默认值为0。  
（2）int Student：：total=0；删除或注释掉，重新编译后，程序不能运行。原因：total静态数据成员没有初始化。  
（3）static int total；移动到private的下方作为私有成员后，程序不能运行。原因：将total作为私有成员后，不能被类的外部函数访问也不能用对象名直接进行访问，total不能被访问。

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\* 功能 : 静态数据成员

\*\* 作者 : 201711010503@stu.sdnu.edu.cn

\*\* 版本 : 2018-11-25 / 16:22

\*\* 版权 : GNU General Public License(GNU GPL)

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#include <iostream>

#include <cstdlib>

#include <windows.h>

#define *\_CRT\_SECURE\_NO\_WARNINGS*

using namespace *std*;

/\*----------------------------------\*

Class Defination

\*-----------------------------------\*/

class student

{

public:

static int total;

student() { total++; }

~student() { total--; }

student(int n, char \*p = "wang");

void getname();

int getnum();

private:

int num;

char name[20];

};

int student::total = 0;

student::student(int n, char \*p)

{

num = n;

*strcpy*(name, p);

total++;

}

void student::getname()

{

*cout* << name << *endl*;

}

int student::getnum()

{

return num;

}

/\*----------------------------------\*

Main Function

\*-----------------------------------\*/

int main()

{

*cout* << "the number of all students:" << student::total << *endl*;

student \*p = new student(13);

delete p;

*cout* << "the number of all students:" << student::total << *endl*;

*cout* << "the number of all students:" << p->total << *endl*;

*cout* << "the number of all students:" << student::total << *endl*;

student s[2];

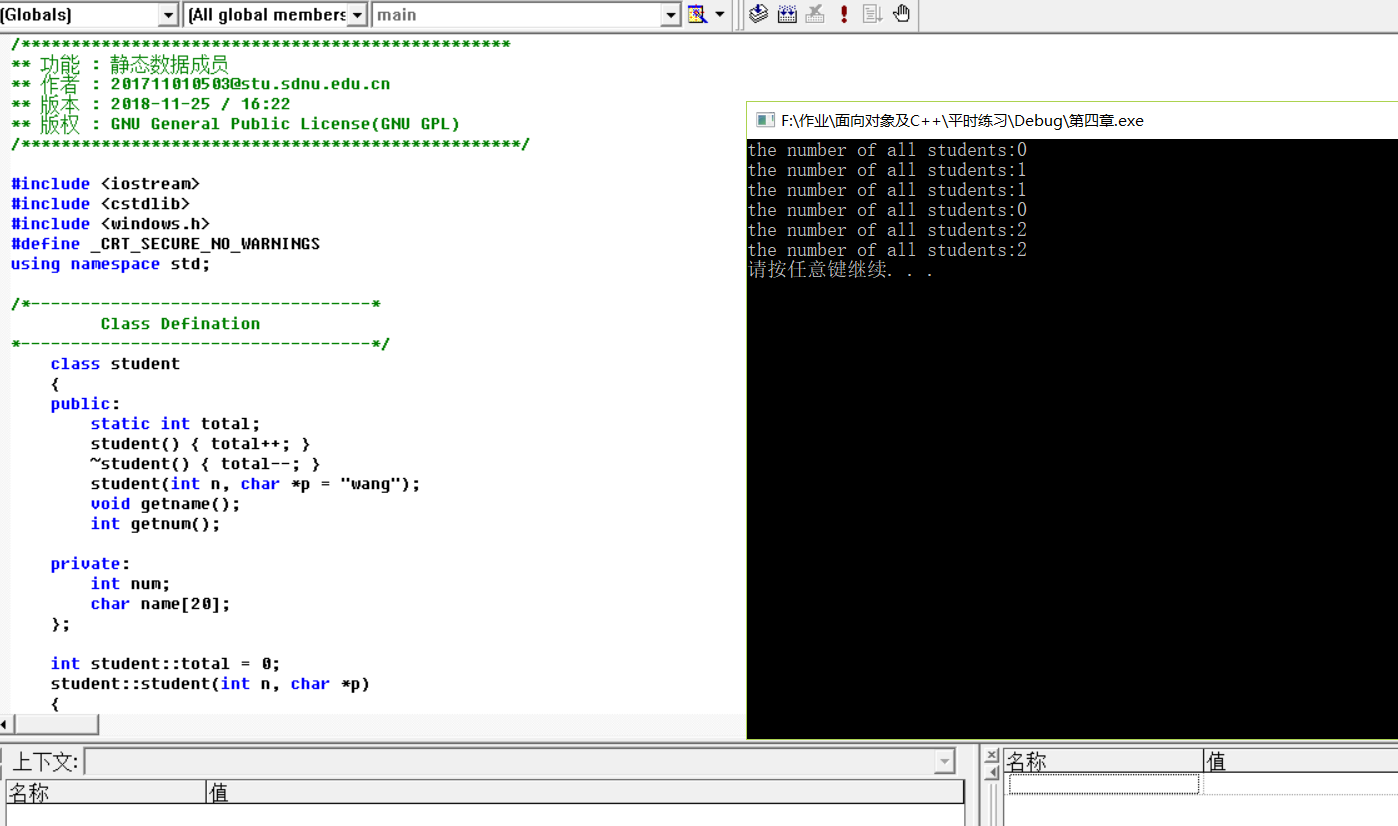
*cout* << "the number of all students:" << s[0].total << *endl*;

*cout* << "the number of all students:" << s[1].total << *endl*;

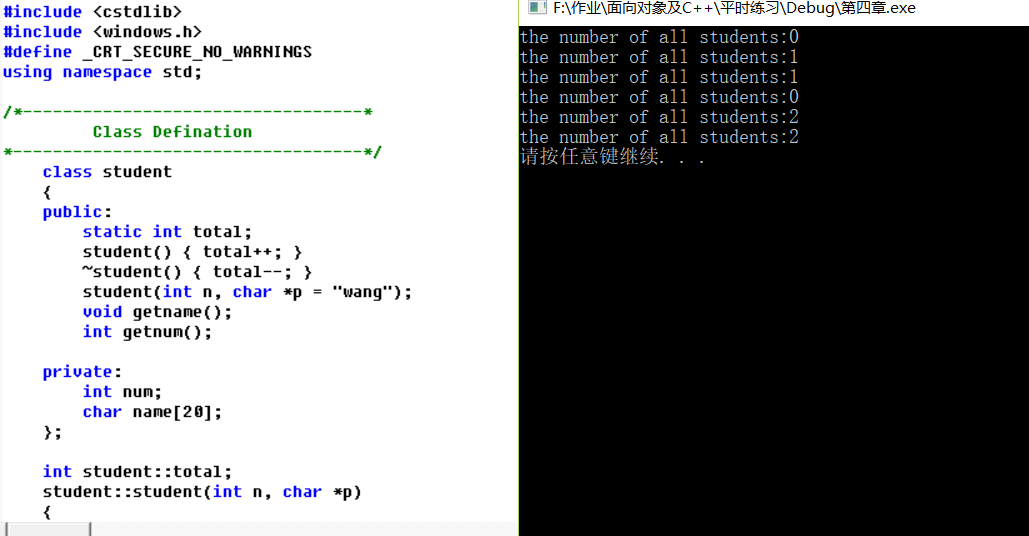
*system*("pause");

return 0;

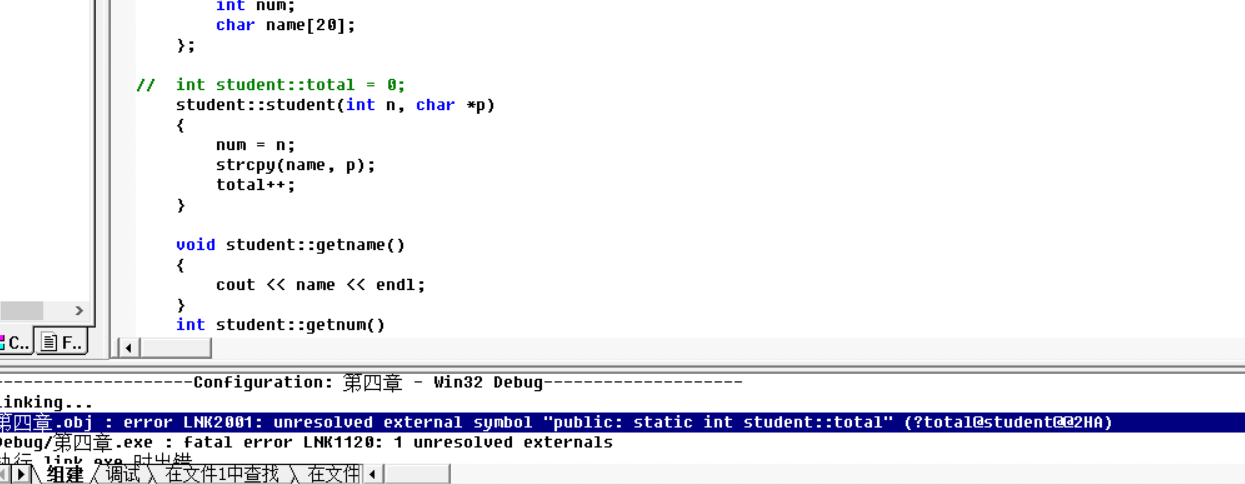
}



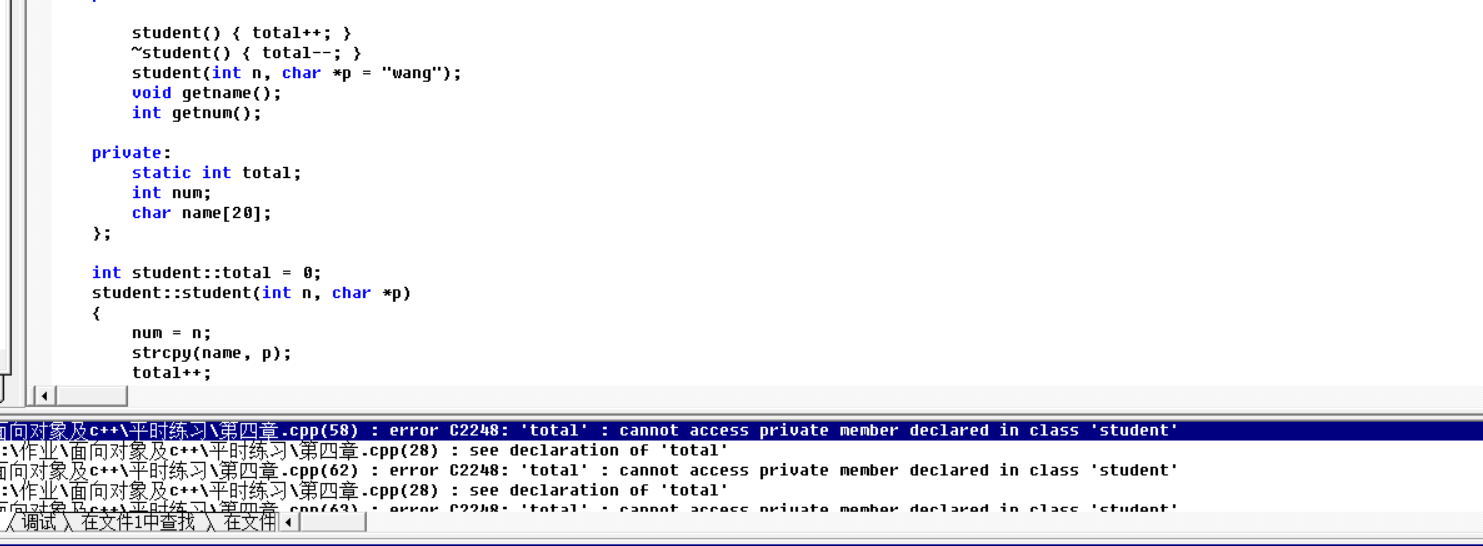
1.



2.



3.



**代码二：**

P105：  
（1）Student::Print()改成p->Print()后能编译运行。原因：因为 delete不会把指针变为空指针，指针还指向已经被删除了的内存，注意置空！  
（2）结果无差异。原因：静态成员函数只能访问静态数据成员，可以通过类名：静态函数名（实际参数表）和对象名.静态成员函数名（实际参数表）进行调用。

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\* 功能 : 静态数据函数

\*\* 作者 : 201711010503@stu.sdnu.edu.cn

\*\* 版本 : 2018-11-25 / 16:22

\*\* 版权 : GNU General Public License(GNU GPL)

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#include <iostream>

#include <cstdlib>

#include <windows.h>

#define \_CRT\_SECURE\_NO\_WARNINGS

using namespace std;

/\*----------------------------------\*

Class Defination

\*-----------------------------------\*/

class student

{

public:

student() { total++; }

~student() { total--; }

student(int n, char \*p = "wang");

void getname();

int getnum();

static void print();

private:

static int total;

int num;

char name[20];

};

int student::total = 0;

student::student(int n, char \*p)

{

num = n;

strcpy(name, p);

total++;

}

void student::getname()

{

cout << name << endl;

}

int student::getnum()

{

return num;

}

void student::print()

{

cout << "the number of all students:" << total << endl;

}

/\*----------------------------------\*

Main Function

\*-----------------------------------\*/

int main()

{

student::print();

student \*p = new student(13);

student::print();

p->print();

delete p;

student::print();

student s[2];

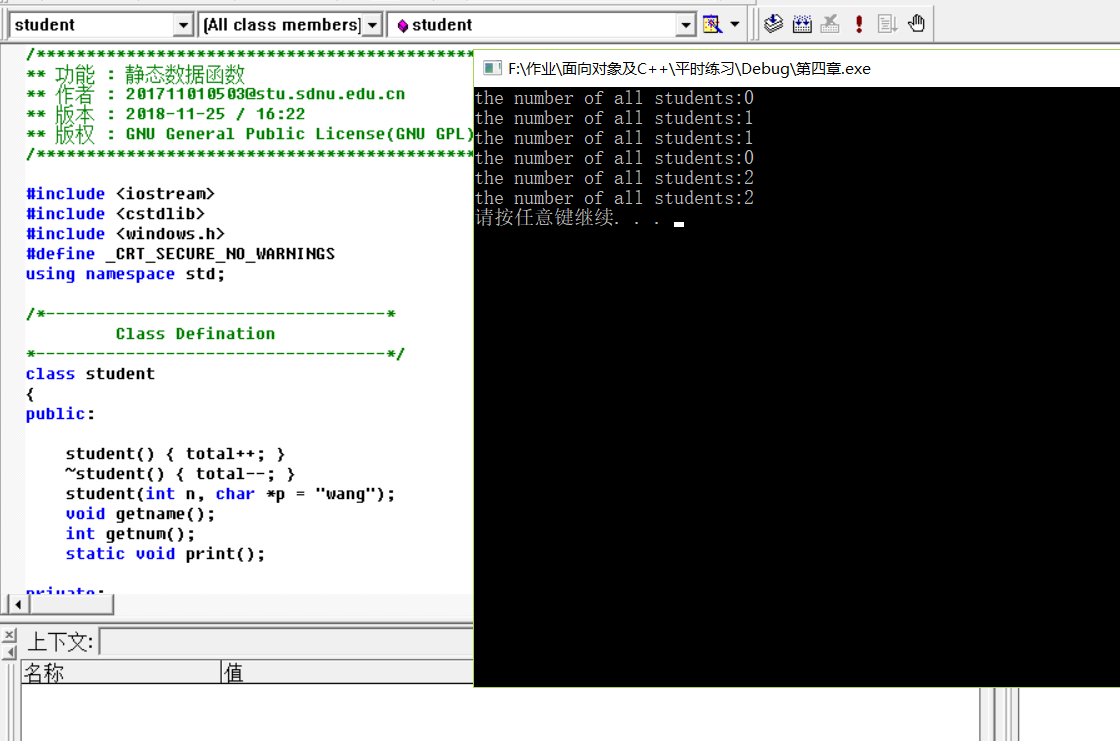
s[0].print();

s[1].print();

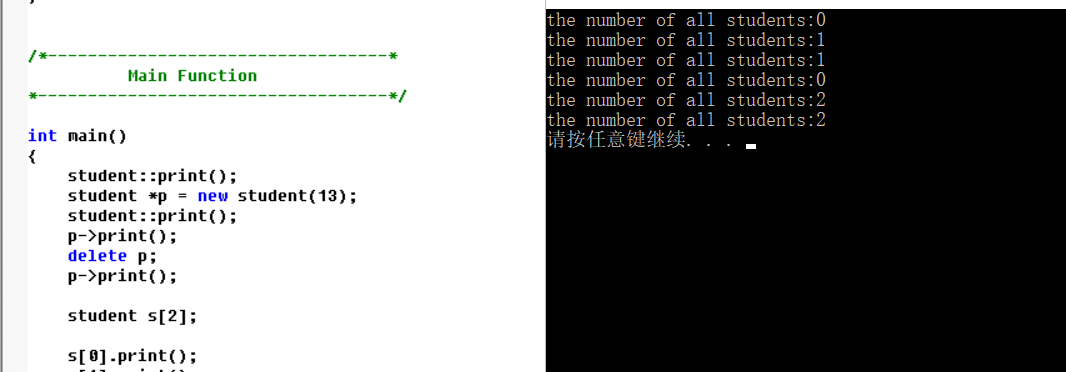
system("pause");

return 0;

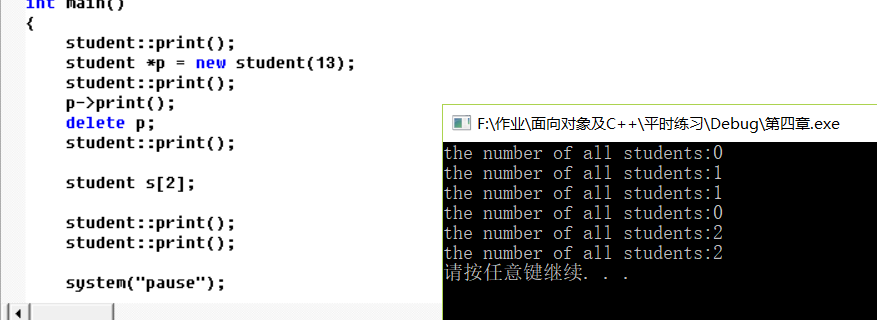
}



1.



2.



**代码三：**

p107：  
（1） 常数据成员的初始化只能在构造函数的初始化列表中进行，不能在函数体内用赋值语句来实现；普通函数两者均可。  
（2） static的常数据成员不能被修改。

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\* 功能 : 常数据成员

\*\* 作者 : 201711010503@stu.sdnu.edu.cn

\*\* 版本 : 2018-11-25 / 17:14

\*\* 版权 : GNU General Public License(GNU GPL)

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#include <iostream>

#include <cstdlib>

#include <windows.h>

using namespace *std*;

/\*----------------------------------\*

Class Defination

\*-----------------------------------\*/

class circle

{

//成员变量

private:

double radius;

const double PI;

//成员函数

public:

circle(double r = 0) :PI(3.1415926)

{

radius = r;

}

double area();

double circumference();

};

double circle::area()

{

return PI \* radius\*radius;

}

double circle::circumference()

{

return 2 \* PI\*radius;

}

/\*----------------------------------\*

Main Function

\*-----------------------------------\*/

int main()

{

circle c1(3.5), c2;

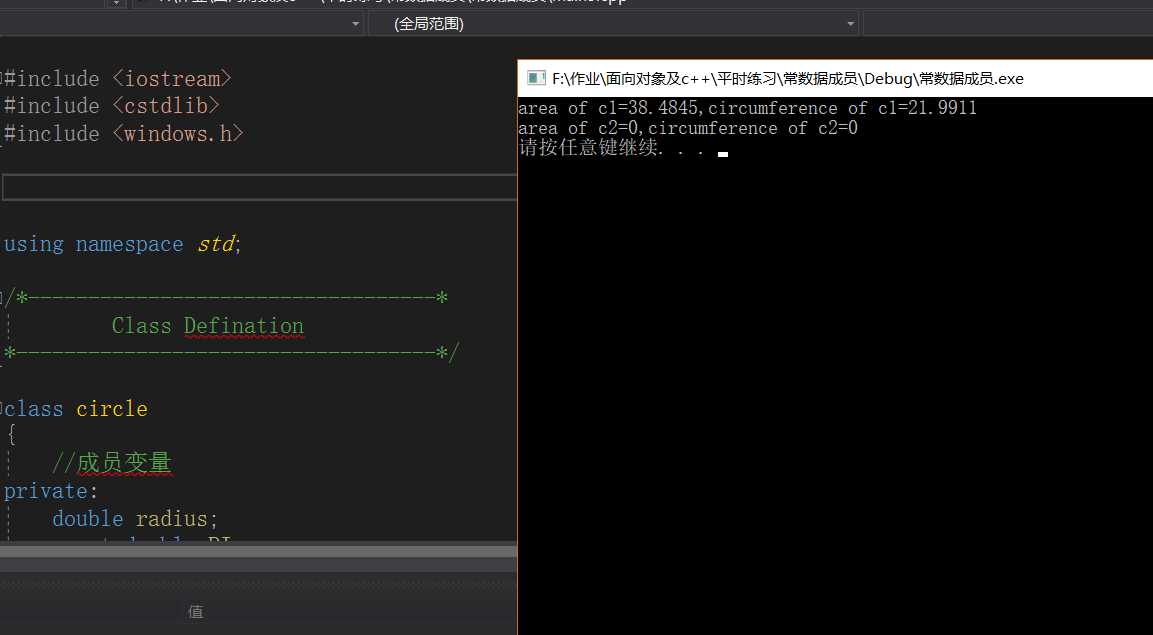
*cout* << "area of c1=" << c1.area() << ",circumference of c1=" << c1.circumference() << *endl*;

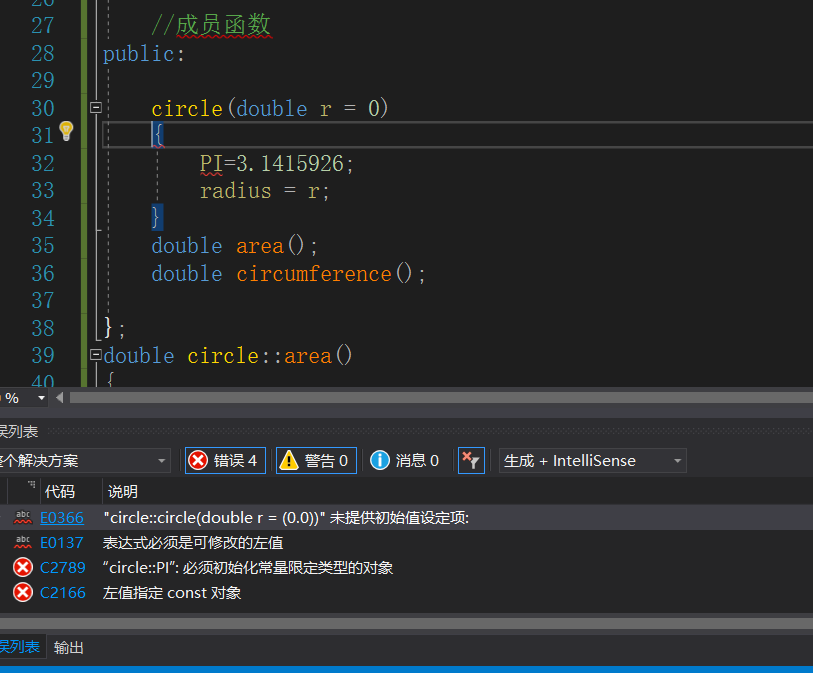
*cout* << "area of c2=" << c2.area() << ",circumference of c2=" << c2.circumference() << *endl*;

*system*("pause");

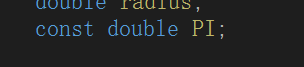
return 0;

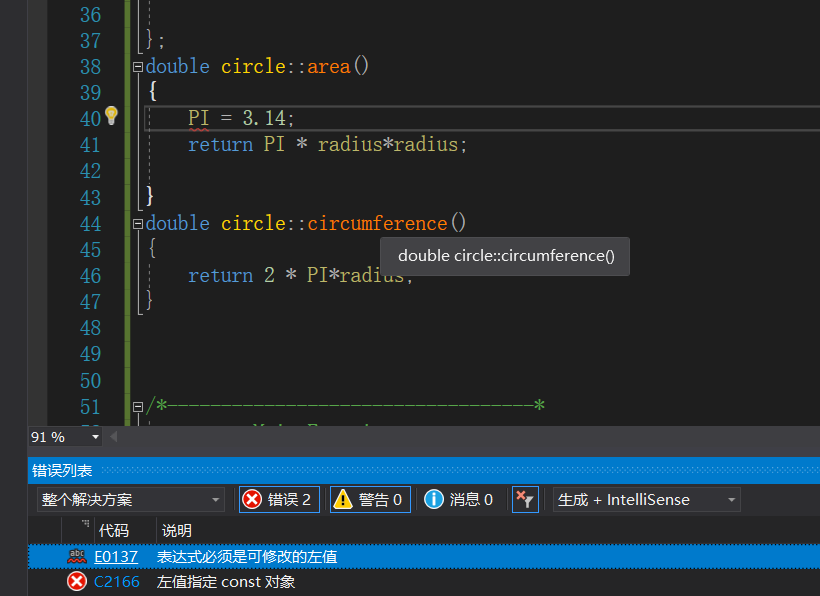
}



**1.**

**2.**





**代码四：**

p109：  
（1） 常成员函数不能修改本类的数据成员  
（2） 常成员函数不能调用未经关键字const修饰的普通成员函数  
（3） 普通函数可以调用常成员函数

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\* 功能 : 常成员函数

\*\* 作者 : 201711010503@stu.sdnu.edu.cn

\*\* 版本 : 2018-11-25 / 17:44

\*\* 版权 : GNU General Public License(GNU GPL)

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#include <iostream>

#include <cstdlib>

#include <windows.h>

using namespace *std*;

/\*----------------------------------\*

Class Defination

\*-----------------------------------\*/

class circle

{

//成员变量

private:

double radius;

const double PI;

//成员函数

public:

circle(double r = 0) :PI(3.1415926)

{

radius = r;

}

double area();

double circumference();

double getradius()const;

};

double circle::area()

{

return PI \* radius\*radius;

}

double circle::circumference()

{

return 2 \* PI\*radius;

}

double circle::getradius()const

{

return radius;

}

/\*----------------------------------\*

Main Function

\*-----------------------------------\*/

int main()

{

circle c1(3.5), c2;

*cout* << "radius of c1=" << c1.getradius()<<*endl*;

*cout* << "area of c1=" << c1.area() << ",circumference of c1=" << c1.circumference() << *endl*;

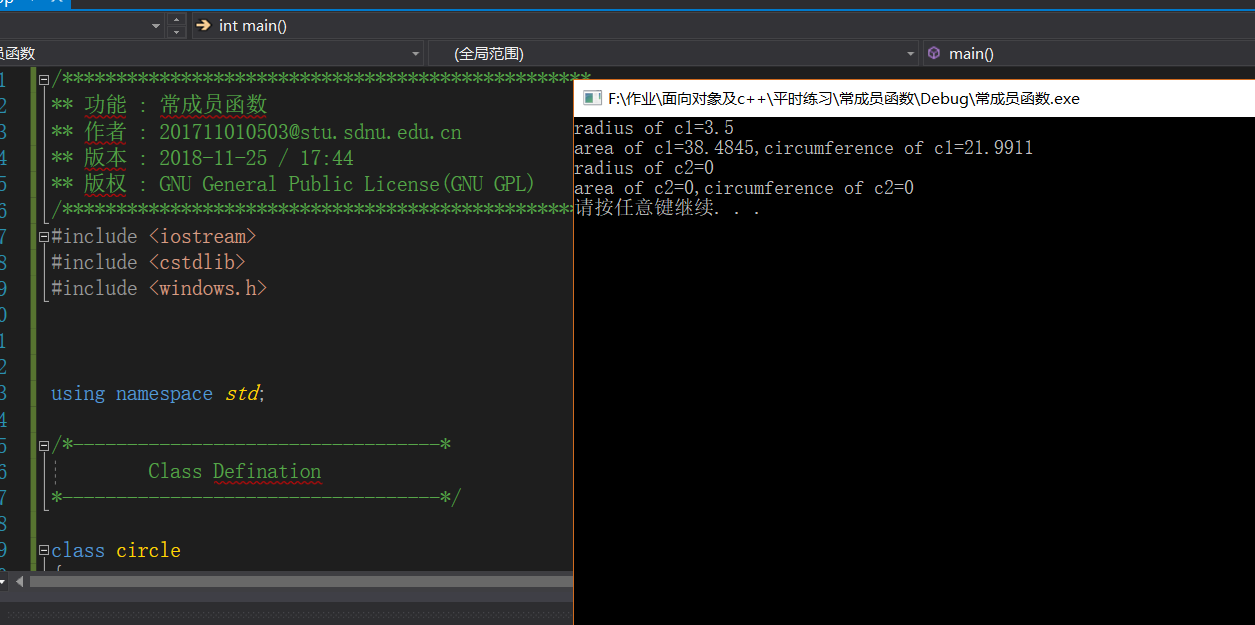
*cout* << "radius of c2=" << c2.getradius()<<*endl*;

*cout* << "area of c2=" << c2.area() << ",circumference of c2=" << c2.circumference() << *endl*;

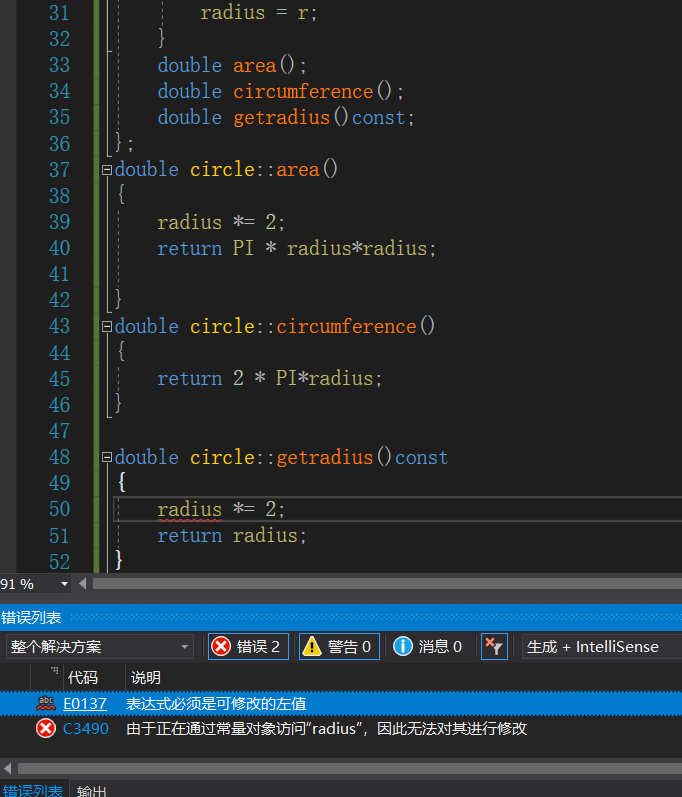
*system*("pause");

return 0;

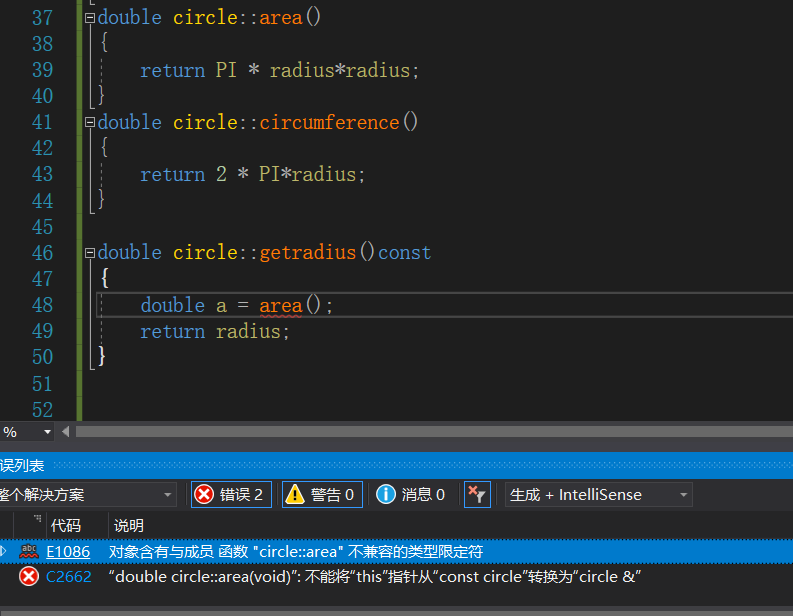
}



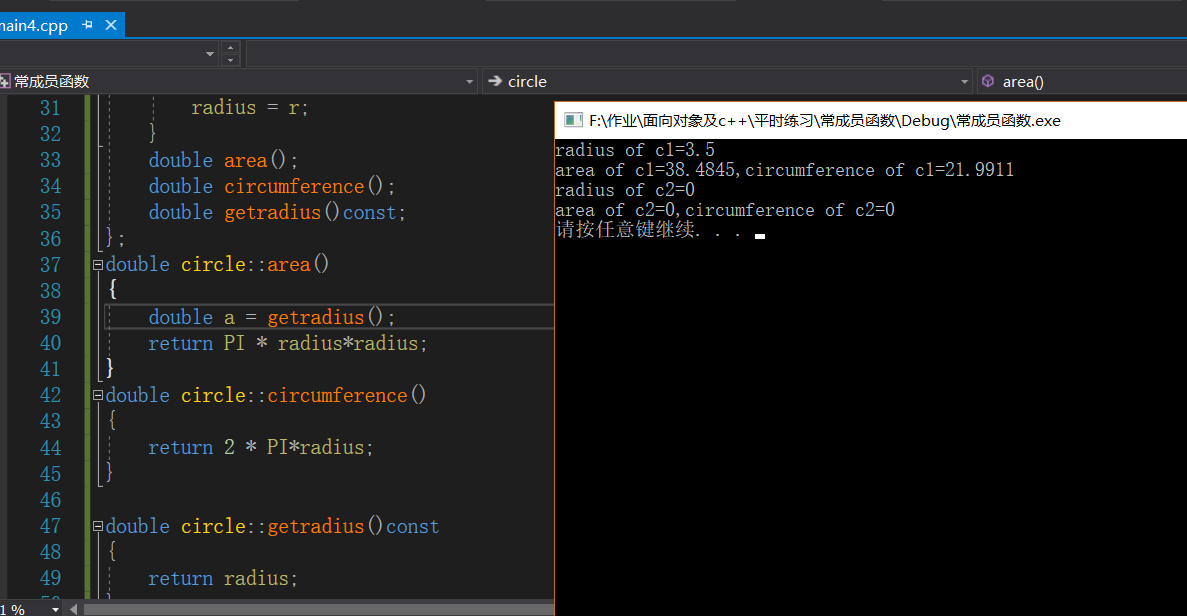
**1.**



**2.**



**3.**



**代码五：**

p110：  
（1） 普通对象也可以调用常成员函数，但先调用重载版本的函数，如果没有，自动调用常成员函数。  
（2） 不能运行，常对象不能调用普通函数。

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\* 功能 : 常对象

\*\* 作者 : 201711010503@stu.sdnu.edu.cn

\*\* 版本 : 2018-11-25 / 17:55

\*\* 版权 : GNU General Public License(GNU GPL)

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#include <iostream>

#include <cstdlib>

#include <windows.h>

#define *\_CRT\_SECURE\_NO\_WARNINGS*

using namespace *std*;

/\*----------------------------------\*

Class Defination

\*-----------------------------------\*/

class person

{

//成员变量

private:

int age;

char \*name;

//成员函数

public:

person(int n = 1,char \*na="zhuli");

~person ();

void print();

void print() const;

void modifyage();

};

person::person(int n, char \*na)

{

age = n;

name = new char[*strlen*(na) + 1];

*strcpy*(name,na);

}

person::~person()

{

delete[]name;

}

void person::print()

{

*cout* << "age:" << age << " name :" << name << *endl*;

*cout* << "this is general print()." << *endl*;

}

void person::print() const

{

*cout* << "age:" << age << " name :" << name << *endl*;

*cout* << "this is const print()." << *endl*;

}

void person::modifyage()

{

age++;

}

/\*----------------------------------\*

Main Function

\*-----------------------------------\*/

int main()

{

const person p1(17, "wu");

*cout* << "output const object p1" << *endl*;

p1.print();

person p2(18, "zahng");

*cout* << "output general object p2" << *endl*;

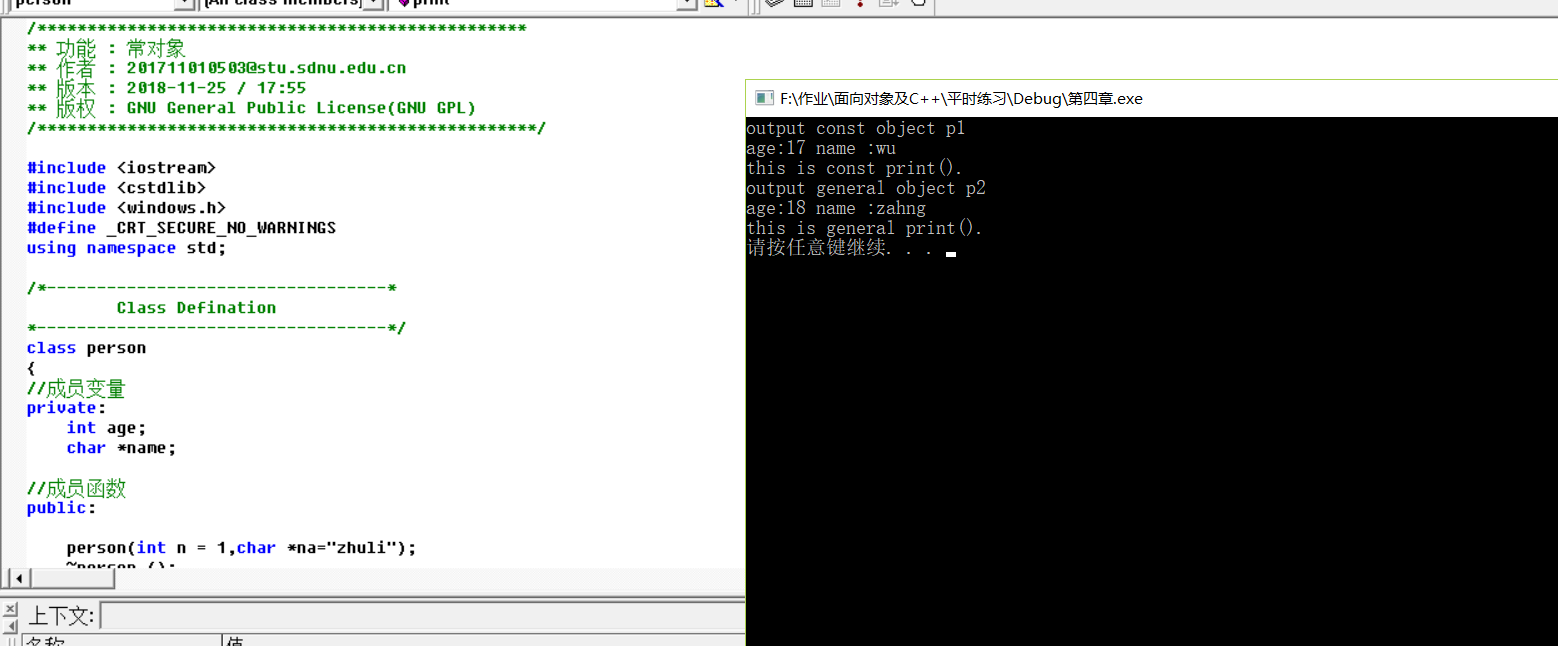
p2.modifyage;

p2.print();

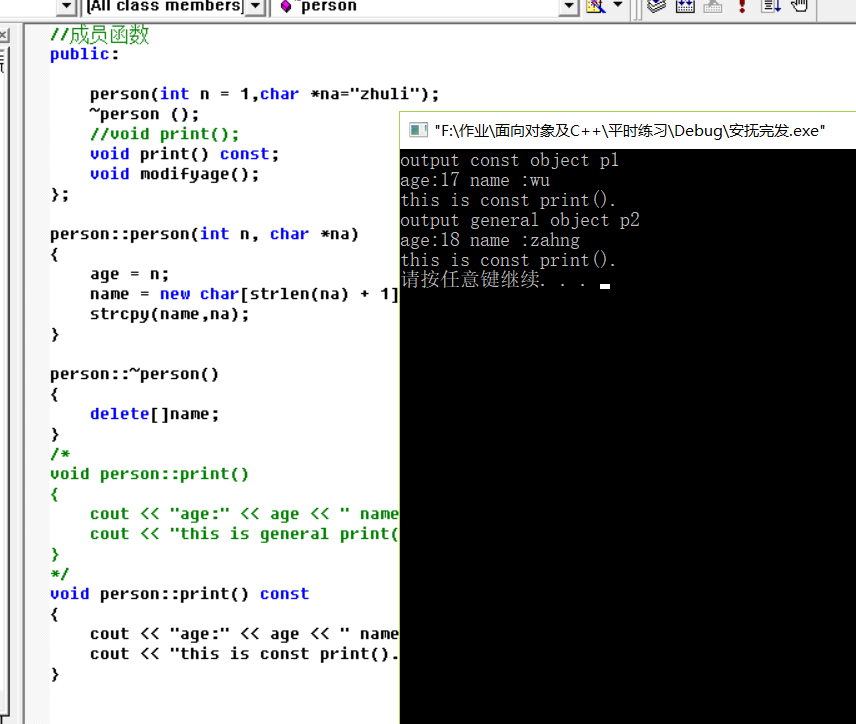
*system*("pause");

return 0;

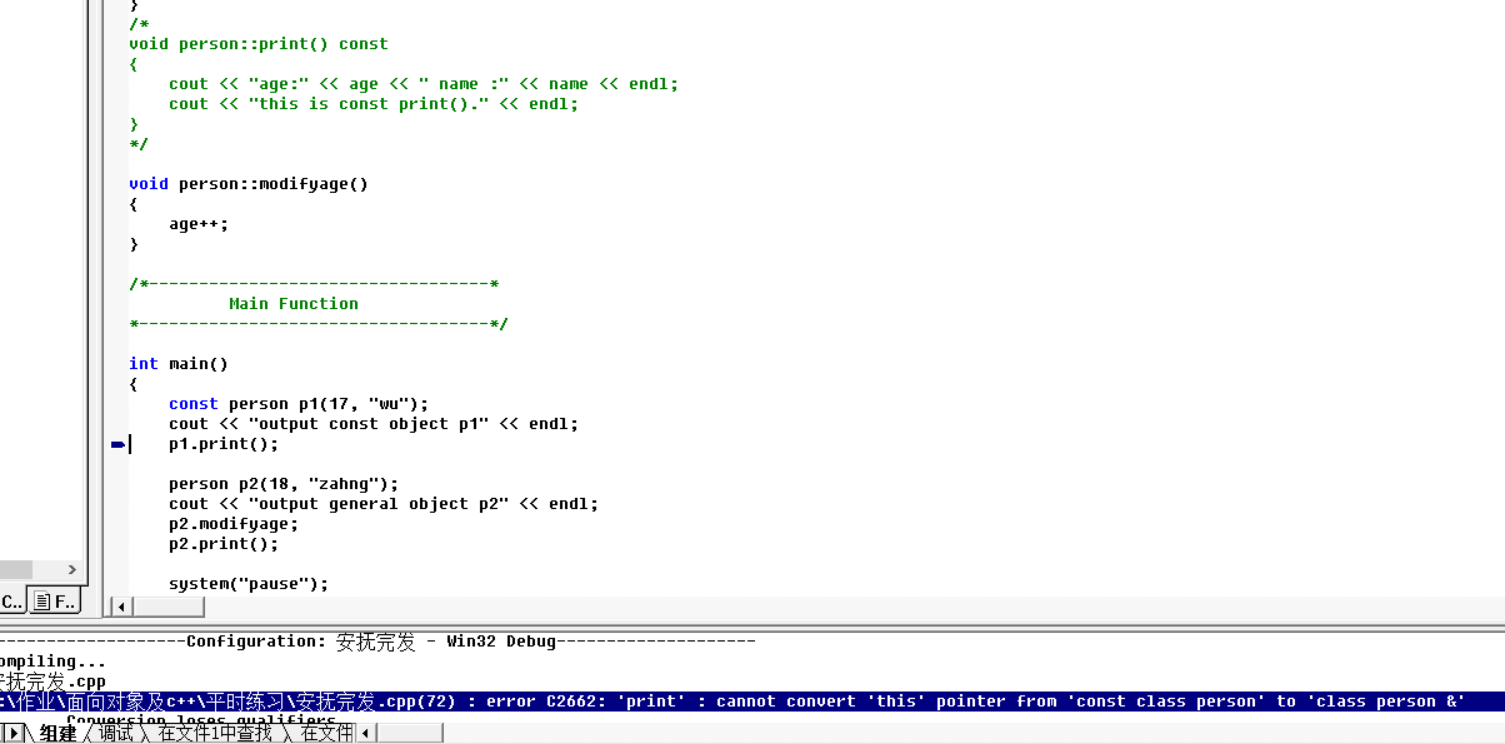
}



1.



2.



**学生信息管理系统：**

// example4\_06\_student.h:学生类的定义

#define *\_CRT\_SECURE\_NO\_WARNINGS*

#ifndef \_STUDENT //条件编译

#define \_STUDENT

#include<iostream>

#include<string>

using namespace *std*;

#define *SIZE* 80

class Student

{

char \*name; //姓名

char ID[19]; //身份证

char number[10]; //学号

char speciality[20]; //专业

int age; //年龄

static int count;//实际有效的学生个数，小于等于对象的个数。

public:

Student(); //无参构造函数

Student(char \*na, char \*id, char \*num, char \* spec, int ag); //带参构造函数

Student(const Student &per); //拷贝构造函数

~Student(); //析构函数

char\* GetName() const; //提取姓名

char\* GetID(); //提取身份证

char\* GetNumber(); //提取学号

char\* GetSpec(); //提取专业

int GetAge() const; //提取年龄

void Display() const; //显示学生信息

void Input(); //输入学生信息

void Insert();

void Delete();

static int GetCount();

};

#endif

//example4\_06\_student.cpp：学生类的实现

int Student::count = 0;

Student::Student()

{

name = *NULL*;

age = 0;

}

//构造函数

Student::Student(char \*na, char \*id, char \*num, char \* spec, int ag)

{

if (na)

{

name = new char[*strlen*(na) + 1]; //避免浅拷贝

*strcpy*(name, na);

}

*strcpy*(ID, id);

*strcpy*(number, num);

*strcpy*(speciality, spec);

age = ag;

count++;

}

Student::Student(const Student &per) //拷贝构造函数，在此例中暂时没有用到

{

if (per.name)

{

name = new char[*strlen*(per.name) + 1];

*strcpy*(name, per.name);

}

*strcpy*(ID, per.ID);

*strcpy*(number, per.number);

*strcpy*(speciality, per.speciality);

age = per.age;

count++;

}

Student::~Student() //析构函数

{

*cout*<<"disCon"<< *endl*;

if (name)

delete[]name;

count--;

}

char\* Student::GetName()const //提取姓名

{

return name;

}

char\* Student::GetID() //提取身份证

{

return ID;

}

int Student::GetAge() const //提取年龄

{

return age;

}

char\* Student::GetSpec() //提取专业

{

return speciality;

}

char\* Student::GetNumber() //提取学号

{

return number;

}

void Student::Display()const //输出数据信息

{

*cout* << "姓 名：" << name << *endl*;

*cout* << "身份证：" << ID << *endl*;

*cout* << "学 号：" << number << *endl*;

*cout* << "专 业：" << speciality << *endl*;

*cout* << "年 龄：" << age << *endl* << *endl*;

}

void Student::Input() //输入数据

{

char na[10];

*cout* << "输入姓 名：";

*cin* >> na;

if (name)

delete[]name;

name = new char[*strlen*(na) + 1];

*strcpy*(name, na);

*cout* << "输入身份证：";

*cin* >> ID;

*cout* << "输入年 龄：";

*cin* >> age;

*cout* << "输入专 业：";

*cin* >> speciality;

*cout* << "输入学 号：";

*cin* >> number;

count++;

}

/\*新增\*/

void Student::Insert()

{

if (!age)

Input();

}

void Student::Delete()

{

age = 0;

count--;

}

int Student::GetCount()

{

return count;

}

//example4\_06.cpp文件：包含主函数及其他一些普通函数

const int N = 10;

/\*

void menu();

void OutputStu(const Student \*array);

void InputStu(Student \*array);

int SearchStu(Student \*array, char \*na);

BOOL InsertStu(Student \*array);

bool DeleteStu(Student \*array, char\* na);

\*/

void menu()

{

*cout* << "\*\*\*\*\*\*\*\*\*\*1.录入信息\*\*\*\*\*\*\*\*\*\*" << *endl*;

*cout* << "\*\*\*\*\*\*\*\*\*\*2.查询信息\*\*\*\*\*\*\*\*\*\*" << *endl*;

*cout* << "\*\*\*\*\*\*\*\*\*\*3.浏览信息\*\*\*\*\*\*\*\*\*\*" << *endl*;

*cout* << "\*\*\*\*\*\*\*\*\*\*4.插入信息\*\*\*\*\*\*\*\*\*\*" << *endl*;

*cout* << "\*\*\*\*\*\*\*\*\*\*5.删除信息\*\*\*\*\*\*\*\*\*\*" << *endl*;

*cout* << "\*\*\*\*\*\*\*\*\*\*0.退 出\*\*\*\*\*\*\*\*\*\*" << *endl*;

}

void OutputStu(const Student \*array) //输出对象数组元素

{

*cout* << "学生总人数=" << Student::GetCount() << *endl*;

for (int i = 0; i < N; i++)

if (array[i].GetAge())

{

array[i].Display();

}

}

int SearchStu(Student \*array, char \*na) //按姓名查询

{

int i, j = N;

for (i = 0; i <N; i++)

if (array[i].GetAge())

{

if (*strcmp*(array[i].GetName(), na) == 0)

j = i;break;

}

return j;

}

void InputStu(Student \*array) //输入对象数组元素

{

char ch;

int i = 0;

do

{

if (Student::GetCount()==N)

{

*cout* << "人数已满，无法继续录入！" << *endl*;

}

if (!array[i].GetAge())

{

array[i].Input();

}

*cout* << "继续输入吗？(Y or N )" << *endl*;

*cin* >> ch;

} while (ch == 'Y');

}

bool InsertStu(Student \*array)

{

if (Student::GetCount()==N)

{

*cout* << "人数已满，无法插入记录!" << *endl*;

return false;

}

for (int i = 0; array[i].GetAge(); i++)

{

array[i].Insert();

}

return true;

}

bool DeleteStu(Student\* array, char \*na)

{

if (Student::GetCount()==0)

{

*cout* << "没有记录，无法删除！" << *endl*;

return false;

}

int i = SearchStu(array, na);

if (i==N)

{

*cout* << "查无此人，无法删除！" << *endl*;

return false;

}

array[i].Delete();

return true;

}

int main()

{

Student array[N];

int choice;

char na[20];

do

{

menu();

*cout* << "Please input your choice:";

*cin* >> choice;

if (choice >= 0 && choice <= 3)

switch (choice)

{

case 1:InputStu(array); break;

case 2:

*cout* << "Input the name searched" << *endl*;

*cin* >> na;

int i;

i = SearchStu(array, na);

if (i == N)

*cout* << "查无此人！\n";

else

array[i].Display();

break;

case 3:OutputStu(array); break;

case 4:if (InsertStu(array))

*cout* << "成功插入一条记录\n";

else

*cout* << "插入失败！\n";

break;

case 5:

*cout* << "Input the name deleted:" << *endl*;

*cin* >> na;

if (DeleteStu(array,na))

*cout* << "成功删除一条记录\n";

else

*cout* << "插入失败！\n";

break;

default:break;

}

} while (choice);

return 0;

}

