先说下，选做一个都没写（逃

2021年5月21日　　21:55

圆面积

2021年3月28日 8:50

描述

编写一个圆类Circle，实现半径的输入、面积的计算和输出。要求在Circle类中分别写3个成员函数实现输入半径、计算面积、输出面积。

输入

输入一行，输入圆的半径(double类型)。

输出

输出一行，输出圆的面积(保留小数点后两位)。

输入样例 1

```
3
```

输出样例 1

```
28.27
```

输入样例 2

```
1.2
```

输出样例 2

```
4.52
```

提示

1、在输出面积前使用如下语句：cout<<setiosflags(ios::fixed)<<setprecision(2);来设置输出格式，因此需要包含iomanip头文件

来自 <http://www.bjfuacm.com/contest/173/problem/A>

```
#include<iostream>
#include<iomanip>
#include<cmath>
using namespace std;

const double PI = acos(-1.0);
class Circle {
private:
    double r;
    double s;
public:
    void setr(double xr)
    {
        r = xr;
    }
    void calcu(){
        s = PI * r * r;
    }
    void gets() {
        cout <<setiosflags(ios::fixed)<<setprecision(2)<<s << endl;
    }
};
int main() {
    double r;
    Circle circle;
    cin >> r;
    circle.setr(r);
    circle.calcu();
    circle.gets();
    return 0;
}
//刘璐婷的
```

```
#include <iostream>
#include <iomanip>
#include <cmath>
using namespace std;

const double PI = acos(double(-1.0));

class Circle
{
private:
    double r;

public:
    void get_r( double r )
    {
        this->r = r;
    }
    void get_s()
    {
        double s = r*r*PI;
        cout << setiosflags(ios::fixed)
        << setprecision(2) << s << endl;
    }
};

int main()
{
    double r;

    while( cin >> r )
    {
        Circle C;
        C.get_r(r);
        C.get_s();
    }

    return 0;
}
//参考答案
```

Book类

2021年3月28日 15:26

描述

以下是图书类Book的声明，缺少实现部分，请实现成员函数并编写main函数测试Book类。

```
class Book
{
private:
    char *name;                //书名
    char *author;              //作者
    int sale;                  //销售量
public:
    Book();                               //无参构造函数
    Book(char *a, char *b, int c);         //有参构造函数
    Book(const Book &);                   //拷贝构造函数
    void print();                          //显示数据
    ~Book();                              //析构函数
};
```

输入

在main函数中，我们输入三行数据，第一行是书的名称(长度不超过100，可能含有空格)，第二行是作者的名字(长度不超过100，可能含有空格)，第三行是销量(整数类型)。

类中有三个对应的成员变量，分别为name，author和sale，利用题目中所给的构造函数来实例化对象，需要注意的是，题目中有三个构造函数，分别是有参构造函数和无参构造函数还有拷贝构造函数。在此我们特别声明：

（1）当输入的name，author和sale都为-1的时候，请使用无参构造函数来实例化对象，此时我们将name的默认值设置为"No name"，author的默认值设置为"No author"，sale的默认值设置为0.

（2）当输入都为0的时候，我们使用拷贝构造函数来处理，这种情况具体在main函数中的实现是这样的：

```
 Book bk1;
Book bk2(bk1);
bk2.print();
```

（3）其他情况下一律用有参数的构造函数来构造对象。

输出

使用类中的void print()方法来输出一定格式的字符串，详见样例。

输入样例 1

```
The Art of Computer Programming
Donald Ervin Knuth
1000
```

输出样例 1

```
Name: The Art of Computer Programming Author: Donald Ervin Knuth Sale: 1000
```

提示

1、注意输出格式，每个图书的信息占一行，信息的项目之间用\t分隔，最后以\n换行。冒号后面都有一个空格

2、输入书名和作者时，因为会含有空格，请用cin.getline()函数。**cin.getline(name,sizeof(name))**

3、比较两个字符串是否相等，请用strcmp(s1,s2)函数，如果s1=s2，则返回0。例如strcmp(name, ″-1″)，如果name是“-1”，则返回值为0.

4、必须要用类（class）来实现代码，否则不得分

来自 <http://www.bjfuacm.com/contest/173/problem/B>

```
#include<iostream>
#include<cstring>
using namespace std;
class Book
{
    private:
        char *name;                //书名
        char *author;              //作者
        int sale;                  //销售量
    public:
           Book();                                //无参构造函数
           Book(char *a, char *b, int c);          //有参构造函数
           Book(const Book &);                    //拷贝构造函数
           void print();                           //显示数据
           ~Book();                               //析构函数
 };

 Book::Book()
 {
     name=new char[100];
     author= new char[100];
     strcpy(name,"No name");
     strcpy(author,"No author");
     sale=0;
 }

 Book::Book(char *n,char*a,int c)
 {
    name=new char[strlen(n)];
    author=new char[strlen(a)];
//    cin.getline(n, sizeof(n));
//    cin.getline(a, sizeof(a));
     strcpy(name,n);
    strcpy(author,a);
    sale=c;
 }

 Book::Book(const Book &x)
 {
      /*name = x.name;
    author = x.author;
    sale = x.sale;
    不对啊！这是指针又不是字符串,不能一个样子写啊*/
     name=new char[strlen(x.name)];
     author=new char[strlen(x.author)];
     strcpy(name,x.name);
     strcpy(author,x.author);
     sale=x.sale;
 }
 void Book::print()
 {
     cout<<"Name: "<<name<<"\t"<<"Author: "<<author<<"\t"<<"Sale: "<<sale<<endl;
 }

 Book::~Book()
 {
     delete[] name;
     delete[] author;
 }

 int main()
 {
     char n[100];
      char a[100];
     int num;
     cin.getline(n,100);
     cin.getline(a,100);
      cin>>num;

     if(strcmp(n,"-1")==0&&strcmp(a,"-1")==0&&num==-1)
    {
         Book bk1;
          bk1.print();
    }
    else if(strcmp(n,"0")==0&&strcmp(a,"0")==0&&num==0)
    {
         Book bk1;
         Book bk2(bk1);
         bk2.print();
    }
   else
    {
         Book bk1(n,a,num);
          bk1.print();
    }
   return 0;
}
```

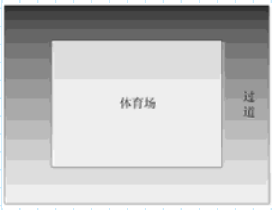
```
    #include <iostream>
#include <cstring>
using namespace std;

class Book
{
private:
    char *name;
    char *author;
    int sale;

public:
    Book()
    {
        name = new char [100];
        strcpy(name, "No name");
        author = new char [100];
        strcpy(author , "No author");
        sale = 0;
    }
    Book(char *a, char *b, int c)
    {
        name = new char [100];
        strcpy(name, a);
        author = new char [100];
        strcpy(author, b);
        sale = c;
    }
    Book(const Book &B)
    {
        this->name = new char [100];
        strcpy(this->name, B.name);
        this->author = new char [100];
        strcpy(this->author, B.author);
        this->sale = B.sale;
    }
    void print()
    {
        cout << "Name: " << name << "\t"
            << "Author: " << author << "\t"
            << "Sale: " << sale << endl;
    }
    ~Book()
    {
        delete[] name;
        delete[] author;

        if( name != NULL )
            name = NULL;
        if( author != NULL )
            author = NULL;
    }
};

int main()
{
    char name[100];
    char author[100];
    int sale;

    cin.getline(name, sizeof(name));
    cin.getline(author, sizeof(author));
    cin >> sale;

    if( strcmp(name, "-1") == 0 && strcmp(author, "-1") == 0 && sale == -1 )
    {
        Book bk1;
        bk1.print();
    }
    else if( strcmp(name, "0") == 0 && strcmp(author, "0") == 0 && sale == 0 )
    {
        Book bk1;
        Book bk2(bk1);
        bk2.print();
    }
    else
    {
        Book bk1( name, author, sale );
        bk1.print();
    }

    return 0;
}
//参考代码
```

栅栏面积和钱

2021年3月28日　　16:20

描述

一矩形体育场如下图所示，现在需在其周围建一矩形过道，并在四周围安上栅栏。栅栏价格为50元/米，过道造价为240元/平方米。过道宽为1.5米，体育场的长宽由键盘输入。



体育场和矩形过道都是如下Rectangle的对象，请实现以下Rectagnle类，编写main函数计算并输出过道和栅栏的造价。

class Rectangle

{

private:

```
  double length; //长
double width; //宽
```

public:

```
  Rectangle(double Length=10.,double Width=5.);
double Area(); //获取面积
double Perimeter();//获取周长
```

};

输入

体育场的长和宽

输出

输出2行

第一行是栅栏的造价

第二行是过道的造价

输入样例 1

```
2.4 1.2
```

输出样例 1

```
960
4752
```

输入样例 2

```
2 1
```

输出样例 2

```
900
4320
```

来自 <http://www.bjfuacm.com/contest/173/problem/C>

```
#include<iostream>
using namespace std;

class Rectangle
{
private:
        double length;
        double width;
public:
        Rectangle(double length, double width);
        double Area(double l, double w);
        double Perimeter(double l, double w);
};

Rectangle::Rectangle(double length, double width)
{
        this->length = length;
        this->width = width;
}
double Rectangle::Area(double l,double w)
{
        double s;
        s = l * w;
        return s;
}
double Rectangle::Perimeter(double l, double w)
{
        double c;
        c = 2 * (l + w);
        return c;
}


int main()
{
        double l;
        double w;
        double w1, w2;
        cin >> l;
        cin >> w;
        Rectangle rect1(l,w);
        Rectangle rect2(l + 3, w + 3);
        rect1.Area(l,w);
        rect2.Area(l+3, w+3);
        rect2.Perimeter(l+3, w+3);
        w1 = (rect2.Area(l + 3, w + 3) - rect1.Area(l, w)) * 240;
        w2 = (rect2.Perimeter(l + 3, w + 3)*50);
        cout <<w2 << endl;
        cout << w1<<endl;
        return 0;
}
```

```
#include <iostream>
using namespace std;

class Rectangle
{
private:
        double length;
        double width;

public:
        Rectangle( double Length = 10., double Width = 5.)
        {
                this->length = Length;
                this->width = Width;
        }
        double Area()
        {
                return length * width;
        }
        double Perimeter()
        {
                return 2.0*(length + width);
        }
};

int main()
{
        double length, width;

        while( cin >> length >> width )
        {
                Rectangle Playground(length, width);
                Rectangle All_Playground(length+3.0, width+3.0);

                double Area = All_Playground.Area() - Playground.Area();
                double Perimeter = All_Playground.Perimeter();

                cout << Perimeter * 50 << endl;

                cout << 240*Area << endl;
        }

        return 0;
}
//参考代码
```

成绩单

2021年3月28日 16:22

描述

为一门课写一个评分程序，评分原则如下：

（1）有两次随堂考试，每次满分50分；

（2）有一次期中考试和一次期末考试，每次满分100分；

（3）期末考试占总评成绩的50%，期中考试占总评成绩的25%，两次随堂考试总共占25%;

（4）总评成绩90~100分为A，80~89分为B，70~79分为C，60~69分为D，低于60分为E;

设计一个Socre类，数据成员如下：

```
string name;//记录学生姓名
double s[4];//存储4次成绩，s[0]和s[1]存储2次随堂考试，s[2]存储期中考试，s[3]存储期末考试
double total;//记录总评成绩
char grade;  //记录对应的等级
```

学生信息由键盘录入，默认总评成绩的等级为B，其他数据项无默认值。计算总评成绩并给出等级，输出某个同学的全部信息。

主函数如下：

```
int main()
{
    Score *s1=new Score;
    s1->Input();
    s1->Evalauate();
    s1->Output();
    return 0;
}
```

输入

输入5行

第1行是学生姓名

第2和3行是两次随堂考试成绩

第4行是期中考试成绩

第5行是期末考试成绩

输出

如果输入的成绩超出了范围，则显示：error

如果输入的成绩在题目要求的范围内，则显示：姓名，总分和等级，详见样例

输入样例 1

```
Jack
90
90
100
100
```

输出样例 1

```
error
```

输入样例 2

```
Mary
50
50
90
100
```

输出样例 2

```
name: Mary, total: 97.5, grade: A
```

提示

冒号和逗号后各有一个空格

来自 <http://www.bjfuacm.com/contest/173/problem/D>

```
#include<iostream>
#include<string>
using namespace std;

class Score
{
private:
        char grade;  //记录对应的等级
        string name;//记录学生姓名
        double s[4];//存储4次成绩，s[0]和s[1]存储2次随堂考试，s[2]存储期中考试，s[3]存储期末考试
        double total;//记录总评成绩
public:
//          Score(char grade, string name, double s[4], double total) ;
        void Input();//应该用一个for循环，输入四次的成绩，然后输入之后做出一个判断（是不是要用到bool类型？类似于flag，在第一次和第二次的时候阈值是50，第三次和第四次阈值是100）
        void Evaluate();//这个函数就是负责计算
        void Output();//这个函数就是负责判断和输出
};//算了，还是输完了再判断吧
void Score::Input()
{
        //double s[4];啊啊啊啊啊啊原来是这里!
        getline(cin,name);
        int i;
        for (i = 0; i < 4; i++)
        {
                cin >> s[i];
        }

}
void Score::Evaluate()
{
        total = (s[0] + s[1]) * 0.25 + s[2] * 0.25 + s[3] * 0.5;
        grade = 'B';
        //不知道用不用。。。先留着8
        if (89 < total && total< 101)
        {
                grade = 'A';
        }
        else if (79 < total && total < 90)
        {
                grade = 'B';
        }
        else if (69 < total && total < 80)
        {
                grade = 'C';
        }
        else if (60 < total && total < 70)
        {
                grade = 'D';
        }
        else
        {
                grade = 'E';
        }
}
void Score::Output()//在这儿判断
{
        //if (s[0] > 50 || s[0] < 0 || s[1] < 0 || s[1] > 50)//能这么写？？？试试
        //不试了，看着别扭
        //{
        //          cout << "error" << endl;
        //}
        //else if (s[2] > 100 || s[2] < 0||s[3]<100|| s[3] > 100)
        //{
        //          cout << "error" << endl;
        //}
        //else
        //{
        //          cout << "name: "<< name << ", "  << "total: " << total << ", "<< "grade: " << grade;
        //}
        int i, flag = 1;
        for (i = 0; i < 4; i++)
        {
                if ((i == 0 || i == 1) && (s[i] < 0 || s[i]>50))
                {
                        cout << "error";
                        flag = 0;
                        break;
                }
                if ((i == 2 || i == 3) && (s[i] < 0 || s[i]>100))
                {
                        cout << "error";
                        flag = 0;
                        break;
                }
        }
        if (flag == 1)
        {
                cout << "name: " << name << ", " << "total: " << total << ", "
                << "grade: " << grade;
        }


}

int main()
{
        Score s1;
        s1.Input();
        s1.Evaluate();
        s1.Output();
        return 0;
}
```

```
#include <iostream>
#include <string>
using namespace std;
class Score
{
private:
string name;//记录学生姓名
double s[4];//存储4次成绩，s[0]和s[1]存储2次随堂考试，s[2]存储期中考试，s[3]存储期末考试
double total;//记录总评成绩
char grade; //记录对应的等级
public:
void Input();
void Evalauate();
void Output();
};
void Score::Input()
{
int i;
getline(cin,name);
for(i=0;i<4;i++)
cin>>s[i];
}
void Score::Evalauate()
{
total=(s[0]+s[1])*0.25+(s[2]*0.25)+(s[3]*0.5);
if(total<60)
grade='E';
else if(total<70)
grade='D';
else if(total<80)
grade='C';
else if(total<90)
grade='B';
else
grade='A';
}
void Score::Output()
{
int i,flag=1;
for(i=0;i<4;i++){
if((i==0||i==1)&&(s[i]<0||s[i]>50))
{
cout<<"error";
flag=0;
break;
}
if((i==2||i==3)&&(s[i]<0||s[i]>100))
{
cout<<"error";
flag=0;
break;
}
}
if(flag==1)
cout<<"name: "<<name<<", "<<"total: "<<total<<", "<<"grade: "<<grade;
}
int main()
{
 Score *s1=new Score;
 s1->Input();
 s1->Evalauate();
 s1->Output();
 return 0;
}
26
```

```
#include <iostream>
#include <cstring>
using namespace std;

class Score
{
private:
   string name;//记录学生姓名
   double s[4];//存储4次成绩，s[0]和s[1]存储2次随堂考试，s[2]存储期中考试，  s[3]存储期末考试
   double total;//记录总评成绩
   char grade;  //记录对应的等级

public:
   void Input();
   void Evalauate();
   void Output();

};
void Score::Input()
{
   cin >> name;
   for (int n = 0; n < 4; n++)
      cin >> s[n];
}
void Score::Evalauate()
{
   total = 0.5 * s[3] + 0.25 * s[2] + 0.25 * (s[1] + s[0]);
   grade = 'B';
   if (90 <= total&&total <= 100)
      grade = 'A';
   else if (80 <= total && total < 90)
      grade = 'B';
   else if (70 <= total && total < 80)
      grade = 'C';
   else if (60 <= total && total < 70)
      grade = 'D';
   else
      grade = 'E';
}
void Score::Output()
{
   if (0 <= s[0] && s[0] <= 50 && 0 <= s[1] && s[1] <= 50 && 0 <= s[2] && s[2] <= 100 && 0 <= s[3] && s[3] <= 100)
      cout << "name: " << name << ", total: " << total << ", grade: " << grade << endl;
   else
      cout << "error" << endl;
}

int main()
{
   Score* s1 = new Score;
   s1->Input();
   s1->Evalauate();
   s1->Output();
   return 0;
}

//白鹭
```

名单类，生日

2021年4月2日　　19:48

```
#include<iostream>
#include<string>
#include<cstring>
using namespace std;

class Date   //日期类
{
private:
   int Date_year;   //年
   int Date_month;   //月
   int Date_day;   //日
public:
   Date(int year = 2000, int month = 1, int day = 1);
   void show();   //以"年-月-日"格式输出年月日
   ~Date();
};
class Croster   //名单类
{
private:
   string name;
   Date birthday;
public:
   Croster();
   Croster(string name, int year, int month, int day);
   Croster(string name, Date date);
   void show();//显示姓名和出生日期
   ~Croster();
};
Date::Date(int year, int month, int day)
{
   Date_year = year;
   Date_month = month;
   Date_day = day;
}
Croster::Croster()
{
   //string类型的数据输入的时候应该怎么输入？我用gets（）可以吗？不行，它不认识;strcpy呢？也不行，不是char*类型的
   name="NULL";//噢！直接赋值就可以了！
   //birthday = (0, 0, 0);
   Date date(0, 0, 0);
   birthday = date;
}//这是输入为0的时候用的那个函数
Croster::Croster(string name, int year, int month, int day)
{
   //cin.getline(name, sizeof(name));记错了吧你
   //getline(cin, name);//使用getline得用string的头文件
   //cin >> year>> month >> day;//出了错找这里。果然是这里。
   this->name = name;
   birthday = Date(year, month, day);
}
Croster::Croster(string name, Date date)
{
   this->name = name;
   //卡住了，怎么向date里面传东西呢？网上说用函数,但是我不太会
   //!!!为什么能这么写?
   birthday = date;
}
Croster::~Croster()
{}//析构函数可以不写东西
Date::~Date()
{}
void Date::show()
{
   //cout >> Date_year >> Date_month >> Date_day >> endl;李在干神魔
   cout << Date_year<<"-"<<Date_month <<"-"<< Date_day;
}
void Croster::show()
{
   //没有与这些操作数匹配的<<运算符?
   //cout << name << birthday << endl;
    cout << "Name: " << name << ", " << "Birthday: ";
   birthday.show();
}
```

描述

已知日期类的定义如下：

```
class Date    //日期类
{
private:
    int Date_year;    //年
    int Date_month;    //月
    int Date_day;    //日
public:
    Date(int year=2000, int month=1, int day=1);
    void show();    //以“年-月-日”格式输出年月日
    ~Date();
};
```

名单类中含有日期类的对象，如下所示：

```
class Croster    //名单类
{
private:
    string name;
    Date birthday;
public:
    Croster();
    Croster(string name,int year,int month,int day);
    Croster(string name, Date date);
    void show();//显示姓名和出生日期
    ~Croster();
};
```

要求实现以上两个类，并在主函数中进行测试。

要求输入为多组数据：

（1）输入为0的时候，直接使用Croster类的无参构造函数（即第1个构造函数）实例化对象，并输出：Name: NULL, Birthday: 0-0-0；

（2）当输入为1时，继续输入姓名和年月日，使用Croster类的含有4个参数的构造函数（即第2个构造函数）实例化对象，并进行输出，详见输出样例2；

（3）当输入为2时，继续输入姓名和年月日，使用Croster类的含有2个参数的构造函数（即第3个构造函数）实例化对象，并进行输出，详见输出样例3；

（4）当输入为-1时，退出程序

来自 <http://www.bjfuacm.com/contest/177/problem/D>

```
int main()
{
    int temp;
    int year, month, day;
    string name;
    Croster c;
    cin >> temp;
    while (temp != -1)
    {
        if (temp == 0)
        {
            c.show();
            //如果不加break就会一直循环哈哈哈哈哈
            break;
        }
        else if (temp == 1)
        {
            cin>>name>> year >> month >> day;
            Croster c(name,year, month, day);
            c.show();
            break;
        }
        else if (temp == 2)
        {
            cin >> name >> year >> month >> day;
            Croster c(name, year, month, day);
            Date birthday(year, month, day);
            c.show();
            break;
        }
    }
          return 0;
}
```

学生类和选课类

2021年4月5日 9:51

```
#include<iostream>
#include<string>
#include<cstring>
using namespace std;
//学生类是选课类的友元类
class Subject   //选课类
{
private:
    double score[3];            //3门课成绩
    const int SMath, SEng, SCpp;    //3门课的学分，分别为4、2、2
    //const类型的数据,使用初始化列表
public:
    Subject(int math = 0, int eng =0, int cpp =0);
    void Input();         //输入3门课的成绩
    friend class Student;    //友元类
};
Subject::Subject(int math, int eng, int cpp) :SMath(4), SEng(2), SCpp(2) {}
void Subject::Input()
{
    for (int i = 0; i < 3; i++)
    {
        cin >> score[i];
    }
}

class Student
{
private:
    string ID;       //学号
    string name;      //姓名
    double GPA;      //平均学分积=（成绩1x学分1+成绩2x学分2+成绩3x学分3）/（学分1+学分2+学分3）
public:
    Student(string id = "00000", string na = "Noname");
    void CalculateGPA(const Subject& sub);    //计算平均学分积
    void Input();                    //输入学号和姓名
    void Show(const Subject& sub)const;      //输出所有信息
};
Student::Student(string id, string na)
{
    ID = id;
    name = na;
}
void Student::CalculateGPA(const Subject& sub)//3门课的学分，分别为4、2、2
{//我没有用指针访问。。。怪不得
 //数组是从0开始啊啊啊啊啊啊
    GPA = (sub.score[0] * sub.SMath + sub.score[1] * sub.SEng + sub.score[2] * sub.SCpp) / (sub.SMath + sub.SEng +
sub.SCpp);
}
void Student::Input()
{
    string id;
    string na;
        cin>>id;
        cin>>na;
        ID=id;
        name=na;
    //getline(cin, name);
    //getline(cin, ID);刚刚建立了类，还没有实体，怎么能直接往里面给东西呢?

}
void Student::Show(const Subject& sub)const
{
    cout << "ID: " << ID << "," << " Name: " << name << "\n";
    cout << "Math " << "Eng " << "Cpp" << "\n";
    cout << sub.score[0] << ' ' << sub.score[1] << ' ' << sub.score[2] << ' ' << "\n";
    cout << "GPA: " << GPA << endl;
}
int main()
{
    int n;      //学生人数
    cin >> n;
    Student* stu = new Student[n];
    Subject* sub = new Subject[n];
    for (int i = 0; i < n; i++)
    {
        stu[i].Input();
        sub[i].Input();
    }
    for (int i = 0; i < n; i++)
    {
        stu[i].CalculateGPA(sub[i]);
        stu[i].Show(sub[i]);
    }
    delete[] stu;
    delete[] sub;
    return 0;
}
```

描述

已知选课类Subject和学生类Student定义如下，学生类是选课类的友元类

```
class Subject    //选课类
{
private:
    double score[3];                //3门课成绩
    const int SMath, SEng, SCpp;    //3门课的学分，分别为4、2、2
public:
    Subject(int math = 0, int eng = 0, int cpp = 0);
    void Input();             //输入3门课的成绩
    friend class Student;     //友元类
};
class Student
{
private:
    string ID;        //学号
    string name;       //姓名
    double GPA;        //平均学分积=（成绩1x学分1+成绩2x学分2+成绩3x学分3）/（学分1+学分2+学分3）
public:
    Student(string id = "00000", string na = "Noname");
    void CalculateGPA(const Subject &sub);    //计算平均学分积
    void Input();                           //输入学号和姓名
    void Show(const Subject &sub)const;        //输出所有信息
};
```

请实现以上两个类，并用如下main函数进行测试：

```
int main()
{
    int n;         //学生人数
    cin >> n;
    Student *stu = new Student[n];
    Subject *sub = new Subject[n];
    for (int i = 0; i < n; i++)
    {
        stu[i].Input();
        sub[i].Input();
    }
    for (int i = 0; i < n; i++)
    {
        stu[i].CalculateGPA(sub[i]);
        stu[i].Show(sub[i]);
    }
    delete[] stu;
    delete[] sub;
    return 0;
}
```

输入

第一行输入学生人数n，下面依次输入n个学生的学号、姓名、3门课的成绩

输出

见输出样例

输入样例 1

```
2
001
Jack
100
89
90
002
Mary
78
69
90
```

输出样例 1

```
ID: 001, Name: Jack
Math Eng Cpp
100 89 90
GPA: 94.75
ID: 002, Name: Mary
Math Eng Cpp
78 69 90
GPA: 78.75
```

输入样例 2

```
3
001
Jack
100
80
90
002
Mary
100
98
99
003
Selina
99
98
98
```

输出样例 2

```
ID: 001, Name: Jack
Math Eng Cpp
100 80 90
GPA: 92.5
ID: 002, Name: Mary
Math Eng Cpp
100 98 99
GPA: 99.25
ID: 003, Name: Selina
Math Eng Cpp
99 98 98
GPA: 98.5
```

提示

输出时，冒号和逗号后各有一个空格，成绩后面有一个空格，课程名称之间有一个空格，“Cpp”后无空格

来自 <http://www.bjfuacm.com/contest/177/problem/E>

计算旅馆人数

2021年4月5日 15:32

```
#include<iostream>
#include<cstring>
using namespace std;

class Hotel
{
public:
	static int getTotal();
	static int total;
	void add(string s);
	void print();
	string getName();
private:
	string m_name;
	int num;
};
int Hotel::total = 0;
string Hotel::getName()
{
	return m_name;
}
void Hotel::print()
{
	cout << num << " " << m_name << " " << total << endl;
}
void Hotel::add(string s)
{
	m_name = s;
	total++;
	num = total;
}
int Hotel::getTotal()
{
	return total;
}
int main()
{
	Hotel h[100];
	h[0].add("Susan");
	h[1].add("Peter");
	h[2].add("John");
	h[3].add("Mary");
	h[4].add("Alice");

	string name;
	cin >> name;

	for (int i = 0; i < Hotel::getTotal(); i++)
	{
		if (h[i].getName() == name)
		{
			h[i].print();
			break;
		}
	}
	return 0;
}
```

描述

编写程序，统计某旅馆住宿客人的总数。要求输入客人的姓名，输出客人的编号（按先后顺序自动生成）、姓名以及总人数。

使用如下main函数对程序进行测试

```
int main(){
Hotel h[100];
h[0].add("Susan");
h[1].add("Peter");
h[2].add("John");
h[3].add("Mary");
h[4].add("Alice");
string name;
cin>>name;
for(int i=0;i<Hotel::getTotal();i++)
{
if(h[i].getName()==name)
{
h[i].print();
break;
}
}
return 0;
}
```

输入

输入一行，输入客人的姓名(不超过100个字符的由英文大小写字母组成的字符串)。

输出

输出一行，输出客人的编号，姓名及总人数，空格分隔。

输入样例 1

```
Peter
```

输出样例 1

```
2 Peter 5
```

提示

请注意，必须要用类（class）来实现代码，否则不得分。

来自 <http://www.bjfuacm.com/contest/177/problem/F>

3.21课堂跟练

2021年4月1日　13:28

现在知道了吗？是sizeof和strlen的使用。
sizeof是计算全部的数组长度，
而strlen是计算实际存在的数组长度。

```
#include<iostream>
using namespace std;
//下边的程序有未知错误!
class Stu
{
private:
	int num;
	char *name;//指针指向谁？没有说。所以这样不行。
	char sex;
	int age;
public:
	void show();
	//below are 构造函数
	Stu();
	Stu(int a,char b[],char c,int d);
	~Stu() //析构函数格式
	//{
	//	cout << num << "," << name << "," << sex << "," << age << endl;
	//}//函数是在类里边定义的
};//大括号就是盒子!
void Stu::show()//Stu::必须加上这个，才认识这是Stu里边的一个函数，你的num，name才会被认识
{
	cout << num << "," << name << "," << sex << "," << age << endl;
}//函数是在类里边定义的
Stu::Stu()//构造函数
{
	num = 423;//int can fuzhi directly
	strcpy_s(name, sizeof("kiana"),"kiana");//name is an arrow that cannot be fuzhi
	directly//review the usage of strcpy!!
	//name is the name of arrow that represents Const and therefore cannot be fuzhi
	directly
	sex = '! ';//单个字符用引号
	age = -1;
}

Stu::Stu(int a, char name[], char sex, int age)
{
	num = a;
	this->name=new char[30];//(37)所以得先申请一块空间，但是还是不行，得释放函数空间吧？用析
	构函数 (16)
	strcpy_s(this->name, sizeof(name),name);//用完之后this就没了，不行
	this->sex = sex;
	this->age = age;
}

Stu::~Stu()
{
	delete[]name;
}
int main()
{
	int i=2;
	cout << i << endl;
	Stu s;
	s.show();
	int a;
	char b[30];
	char c;
	int d;
	cout << "number";
	cin >> a;
	getchar();
	cout << "name";
	//cin >> b;//是不行的，如果遇到空格就输入结束了
	cin.getline(b, sizeof(b));//???
```

```
    //也不行，getline会把上一个回车当成确认，得先输个回车
    //所以52行来个getchar
    cout << "sex";
    cin >> c;
    cout << "age";
    cin >> d;
    Stu s1(a, b, c, d);//如果你不写（）的话那就默认调用不带参数的构造函数
    s1.show();
    return 0;
}
```

————继承和派生————

2021年5月21日 21:50

Table tennis

2021年4月17日 20:53

```
#include<iostream>
using namespace std;
class TableTennisPlayer {
private:
	string firstname;
	string lastname;
	bool hasTable;
public:
	TableTennisPlayer(const string&, const string&, bool);
	string FirstName() const;
	string LastName() const;
	bool HasTable() const;
};
class RatedPlayer :public TableTennisPlayer
{
public:
	RatedPlayer(int rating, const string&, const string&, bool);
	int Rating();
private:
	int m_rating;
};
TableTennisPlayer::TableTennisPlayer(const string&f, const string&l, bool b)
{
	firstname = f;
	lastname = l;
	hasTable = b;
}
string TableTennisPlayer::FirstName() const
{
	return firstname;
}
string TableTennisPlayer::LastName() const
{
	return lastname;
}
bool TableTennisPlayer::HasTable()const
{
	return hasTable;
}
RatedPlayer::RatedPlayer(int rating, const string&f, const string&l, bool b):TableTennisPlayer(f,l,b)
{
	m_rating = rating;
}
int RatedPlayer::Rating()
{
	return m_rating;
}
int main()
{
	string firstname, lastname;
	bool hasTable;
	int rating;
	char flag;
	while (cin >> flag) {
		if (flag == 'T') {
			cin >> firstname >> lastname >> hasTable;
			TableTennisPlayer tp(firstname, lastname, hasTable);
			if (tp.HasTable())
				cout << tp.FirstName() << " " << tp.LastName() << " has a table.\n";
			else
				cout << tp.FirstName() << " " << tp.LastName() << " hasn't a table.\n";
		}
		else if (flag == 'R') {
			cin >> firstname >> lastname >> hasTable >> rating;
			RatedPlayer rp(rating, firstname, lastname, hasTable);
			if (rp.HasTable())
				cout << rp.FirstName() << " " << rp.LastName() << " has a table. The rating is "
				<< rp.Rating() << ".\n";
			else
				cout << rp.FirstName() << " " << rp.LastName() << " hasn't a table. The rating is "
				<< rp.Rating() << "\n";
		}
	}
	return 0;
}
```

编写TableTennisPlayer类和RatedPlayer类（RatedPlayer类继承TableTennisPlayer类
输入多行，每一行以'T'或'R'开头。
'T'表示本行接下来输入一个TableTennisPlayer对象的信息
包括firstname,lastname和hasTable（是否有乒乓球台）；
'R'表示本行接下来输入一个RatedPlayer对象的信息，包括firstname,lastname，hasTable和rating（选
手的得分）。
输出
一行输入对应一行输出
输入样例 1
T Bill Gates 1
输出样例 1
Bill Gates has a table.
输入样例 2
R Jike Zhang 0 19000
输出样例 2
Jike Zhang hasn't a table. The rating is 19000.
bool类型的输入：0表示false，1表示true,bool flag=true;

编写TableTennisPlayer类和RatedPlayer类（RatedPlayer类继承TableTennisPlayer类），其中TableTennisPlayer类的定义如下所示：

```
class TableTennisPlayer{

private:

string firstname;

string lastname;

bool hasTable;

public:

TableTennisPlayer(const string &, const string &, bool);

string FirstName() const;

string LastName() const;

bool HasTable() const;

};
```

实现后，通过以下main函数的测试：

```
int main(){

string firstname, lastname;

bool hasTable;

int rating;

char flag;

while(cin>>flag){

if(flag=='T'){

cin>>firstname>>lastname>>hasTable;

TableTennisPlayer tp(firstname,lastname,hasTable);

if(tp.HasTable())

cout<<tp.FirstName()<<" "<<tp.LastName()<<" has a table.\n";

else

cout<<tp.FirstName()<<" "<<tp.LastName()<<" hasn't a table.\n";

} else if(flag=='R'){

cin>>firstname>>lastname>>hasTable>>rating;

RatedPlayer rp(rating,firstname,lastname,hasTable);

if(rp.HasTable())

cout<<rp.FirstName()<<" "<<rp.LastName()<<" has a table. The rating is "<<rp.Rating()<<".\n";

else
cout<<rp.FirstName()<<" "<<rp.LastName()<<" hasn't a table. The rating is "<<rp.Rating()<<".\n";

}

}

return 0;

}
```

来自 <http://www.bjfuacm.com/contest/178/problem/A>

Vehicle类（虚函数多态）

2021年4月18日 9:40

```
#include<iostream>
using namespace std;

class Vehicle
{
protected:
	string m_name;
	string m_color;
public:
	virtual void display() {};//报错说这里有问题，但是这里又不用做什么，所以改的话只用加一个大括号
};
class Car :public Vehicle
{
private:
	int m_pas;
public:
	Car(string name, string color, int pas);
	void display();
};
Car::Car(string name, string color, int pas)
{
	m_name = name;
	m_color = color;
	m_pas = pas;
}
void Car::display()
{
	cout << "Car name:" << m_name << " Car color:" << m_color << " Car passenger:" << m_pas << endl;
}
class Truck :public Vehicle
{
private:
	double m_cap;
public:
	Truck(string name, string color, double cap);
	void display();
};
Truck::Truck(string name, string color, double cap)
{
	m_name = name;
	m_color = color;
	m_cap = cap;
}
void Truck::display()
{
	cout << "Truck name:" << m_name << " Truck color:" << m_color << " Truck capacity:" << m_cap << endl;

}
int main()
{
	Vehicle* p;
	char type;
	char name[110], color[110];
	int pas;
	double cap;
	while (cin >> type)
	{
		cin >> name >> color;
		if (type == 'C')
		{
			cin >> pas;
			Car car(name, color, pas);
			p = &car;
			p->display();
		}
			else if (type == 'T')
			{
				cin >> cap;
				Truck truck(name, color, cap);
				p = &truck;
				p->display();
			}
	}
	return 0;
}
```

Person Student

2021年4月18日　　10:21

实现一个Person类，再实现一个Student类，要求Student类继承Person类，通过以下测试：

int main()

{

Person * p;

p = new Person;

p->input();

p->display();

delete p;

p = new Student;

p->input();

p->display();

delete p;

return 0;

}

输入

输入包含两行，第一行为一个姓名（不包含空格）；第二行为一个学号和一个姓名（学号、姓名都不包含空格），学号和姓名之间用空格间隔

输出

输出为两行，第一行为一个姓名；第二行为学号和姓名，学号和姓名之间用空格间隔

输入样例 1

```
Mary
001 Mary
```

输出样例 1

```
Mary
001 Mary
```

提示

来自 <http://www.bjfuacm.com/contest/178/problem/B>

```
#include<iostream>
#include<cstring>
using namespace std;

class Person
{
public:
        virtual void input();
        virtual void display();
protected:
        string m_name;
};
void Person::input()
{
        string name;
        cin>>name;
        m_name = name;
}
void Person::display()
{
        cout << m_name << endl;
}
class Student :public Person
{
public:
        void display();
        void input();
private:
        string m_num;
};

void Student::input()
{
        string name;
        string num;
        //char a;
        cin>>name>>num;
        //cin>>a;
        m_name = name;
        m_num = num;
}
void Student::display()
{
        cout << m_name << ' ' << m_num << endl;
}
int main()
{
        Person* p;
        p = new Person;
        p->input();
        p->display();
        delete p;

        p = new Student;
        p->input();
        p->display();
        delete p;
        return 0;
}
```

图书商品

2021年4月18日 10:22

描述

编写两个类，分别是：

```
class Item_base //未打折的图书商品

{

protected:

string ISBN; //图书序列号

double price; //单价

public:

Item_base(const string & book_ISBN = "", double sales_price = 0.0);

string get_ISBN() const;

virtual double net_price(int) const; //返回购买指定数量的图书的总价

virtual ~Item_base();

};
```

第二个类是：

```
class Bulk_Item : public Item_base //根据购买数量打折

{

public:

Bulk_Item(const string & book_ISBN = "", double sales_price = 0.0, int min_qty = 0, double
discount = 0.0);

double net_price(int) const; //返回根据购买数量打折后的总价

private:

int min_qty; // 买够这个数量可以打相应的折扣

double discount; //折扣

};
```

实现以上两个类，通过下面main函数的测试

```
int main()

{

Item_base book("0-001-0001-1", 10.0);

Bulk_Item bulk1("0-001-0001-1",10.0, 5, 0.1);

Bulk_Item bulk2("0-001-0001-1", 10.0, 10, 0.2);

int num;

while (cin >> num)

{

cout << bulk1.get_ISBN() << "\t" << num << "\t";

Item_base * p;

if (num >= 10) p = &bulk2;

else if (num >= 5) p = &bulk1;

else p = &book;

cout << p->net_price(num) << "\n";

}

return 0;

}
```

输入

图书的数量。

输出

输出购买的图书的ISBN,它的数量以及总的价格。(用main函数中输出的形式即可)

输入样例 1

```
2
6
11
```

```
#include<iostream>
using namespace std;

class Item_base //未打折的图书商品
{
protected:
	string ISBN; //图书序列号
	double price; //单价
public:
	Item_base(const string& book_ISBN = "", double sales_price = 0.0);
	string get_ISBN() const;
	virtual double net_price(int a) const; //返回购买指定数量的图书的总价
	virtual ~Item_base() {};
};
Item_base::Item_base(const string& book_ISBN, double sales_price) :ISBN(book_ISBN),price(sales_price){}
string Item_base::get_ISBN()const
{
	return ISBN;
}
double Item_base::net_price(int a) const //返回购买指定数量的图书的总价
{
	return price * a *1.0;
}

class Bulk_Item : public Item_base //根据购买数量打折
{
public:
	Bulk_Item(const string& book_ISBN = "", double sales_price = 0.0, int min_qty = 0, double discount = 0.0);
	double net_price(int n) const; //返回根据购买数量打折后的总价
private:
	int min_qty; // 买够这个数量可以打相应的折扣
	double discount; //折扣
};
Bulk_Item::Bulk_Item(const string& book_ISBN , double sales_price , int min_qty , double
discount ) :Item_base(book_ISBN, sales_price)
{
	this->min_qty = min_qty;
	this->discount = discount;
}
double Bulk_Item::net_price(int n) const
{
	return ((1.0 - discount) * n * 10);//不加括号好像输出的是e。。不知道问题是不是出在这里
}
int main()
{
	Item_base book("0-001-0001-1", 10.0);
	Bulk_Item bulk1("0-001-0001-1", 10.0, 5, 0.1);
	Bulk_Item bulk2("0-001-0001-1", 10.0, 10, 0.2);

	int num;
	while (cin >> num)
	{
		cout << bulk1.get_ISBN() << "\t" << num << "\t";

		Item_base* p;
		if (num >= 10) p = &bulk2;
		else if (num >= 5) p = &bulk1;
		else p = &book;

		cout << p->net_price(num) << "\n";
	}
	return 0;
}
```

输出样例 1

```
0-001-0001-1     2     20
0-001-0001-1     6     54
0-001-0001-1     11    88
```

来自 <http://www.bjfuacm.com/contest/178/problem/C>

表面积和体积（抽象类）

2021年4月18日 10:23

描述

编写程序，计算长方体、圆柱体和球的表面积和体积。要求先定义一个抽象类Shape如下：

```
class Shape {

public:

Shape() {}

virtual double area() = 0;

virtual void input() = 0;

virtual double volume() = 0;

virtual ~Shape() {}

};
```

使用Shape类派生出长方体类、圆柱体类、球类，在这三个类里实现从Shape类继承来的纯虚函数。使用如下代码通过测试。

```
void work(Shape *s) {

s->input();

cout << s->area() << " " << s->volume() << endl;

delete s;

}

int main() {

char c;

while (cin >> c) {

switch (c) {

case 'y':

work(new Cylinder());

break;

case 'c':

work(new Cuboid());

break;

case 'q':

work(new Ball());

break;

default:

break;

}

}

return 0;

}
```

```
#include<iostream>
#include<cmath>
using namespace std;
const double pi=acos(-1);

class Shape
{
public:
Shape() {}
virtual double area() = 0;
virtual void input() = 0;
virtual double volume() = 0;
virtual ~Shape() {}
protected:
	int a,b,c;
	double r,h;
	double R;
};
class Cuboid: public Shape
{
	public:
		void input();
		double area();
		double volume();
};
void Cuboid::input()
{
	cin>>a>>b>>c;
}
double Cuboid::area()
{
	return (a*b+b*c+a*c)*2;
}
double Cuboid::volume()
{
	return a*b*c;
}
class Ball:public Shape
{
	public:
		void input();
		double area();
		double volume();
};
void Ball::input()
	{
		cin>>R;
	}
double Ball::area()
	{
		return 4*pi*R*R;
	}
double Ball::volume()
	{
	return 4.0/3.0*pi*R*R*R;
	}
class Cylinder:public Shape
{
	public:
		void input()
		{
			cin>>r>>h;
		}
		double area()
		{
			return 2*pi*r*r+2*pi*h*r;
		}
		double volume()
		{
			return pi*r*r*h;
		}
};
void work(Shape *s) {
s->input();
cout << s->area() << " " << s->volume() << endl;
delete s;
}
int main()
{
char c;
return 0;

}
```

输入

输入包含多行，每行首先是一个字符'c'，'y'，'q'，分别表示输入长方体、圆柱体或球的信息，接下来是对应的输入。

输出

每行输入对应一行输出，表示该形状的表面积和体积，以空格分隔。

输入样例 1

```
c 3 4 5
y 3 5
q 5
```

输出样例 1

```
94 60
150.796 141.372
314.159 523.599
```

提示

pi的精度要足够，比如使用 `const double pi = acos(-1);`

来自 <http://www.bjfuacm.com/contest/178/problem/E>

```
delete s;
}
int main()
{
char c;
while (cin >> c) {
switch (c) {
case 'y':
work(new Cylinder());
break;
case 'c':
work(new Cuboid());
break;
case 'q':
work(new Ball());
break;
default:
break;
}
}
return 0;
}
```

————运算符重载————

2021年5月21日 21:52

Singer

2021年4月29日 20:14

```
#include<iostream>
#include<string>
using namespace std;

class Singer
{
	public:
	string m_name;
	char m_sex;
	int m_age;
	float m_score;
	public:
		Singer(string name="k",char sex='F',int age=44,float score=87.0);
		string getName();
		friend ostream& operator<<(ostream &os,const Singer s);
		friend istream& operator>>(istream &is,Singer &s);
		int operator>(Singer s);
		int operator==(Singer s);
		~Singer(){};
};
Singer::Singer(string name,char sex,int age,float score)
{
	m_name=name;
	m_sex=sex;
	m_age=age;
	m_score=score;
}
int Singer::operator>(Singer s)
{
	if(this->m_score>s.m_score)
	{
		return 1;
	}
	else return 0;
}
int Singer::operator==(Singer s)
{
	if(this->m_score==s.m_score)
	{
		return 1;
	}
	else return 0;
}

istream& operator>>(istream &is,Singer &s)
{
	is>>s.m_name;
	is>>s.m_sex;
	is>>s.m_age;
	is>>s.m_score;
	return is;
}
ostream& operator<<(ostream &os,Singer s)
{
        os<<s.m_name<<" "<<s.m_sex<<" "<<s.m_age<<" "<<s.m_score;
        return os;
}
string Singer::getName()
{
        return m_name;
}
int main()
{
Singer s1,s2;
cin>>s1>>s2;
cout<<s1<<"\n"<<s2<<endl;

if(s1>s2)
cout<<s1.getName()<<"'s score is higher than "<<s2.getName()<<"'s.\n";
else if(s1==s2)
cout<<s1.getName()<<"'s score is equal to "<<s2.getName()<<"'s.\n";
else
cout<<s1.getName()<<"'s score is lower than "<<s2.getName()<<"'s.\n";
return 0;
}
```

Comlex类

2021年4月30日　　19:34

```
#include<iostream>
#include<string>
using namespace std;

class Complex

{
private:
    double x;
    double y;
public:
    Complex(double x = 0.0, double y = 0.0);
    Complex& operator+=(const Complex&);
    Complex& operator-=(const Complex&);
    Complex& operator*=(const Complex&);
    Complex& operator/=(const Complex&);
    friend Complex operator+(const Complex&, const Complex&);
    friend Complex operator-(const Complex&, const Complex&);
    friend Complex operator*(const Complex&, const Complex&);
    friend Complex operator/(const Complex&, const Complex&);
    friend bool operator==(const Complex&, const Complex&);
    friend bool operator!=(const Complex&, const Complex&);
    friend ostream& operator<<(ostream&, const Complex&);
    friend istream& operator>>(istream&, Complex&);
};
istream & operator>>(istream &is, Complex &a)
{
        is>>a.x>>a.y;
        return is;
}
ostream & operator<<(ostream &os, const Complex &a)
{
        os<<a.x<<" + "<<a.y<<"i";
        return os;
}
Complex operator+(const Complex &a, const Complex &s)
{
        Complex c;
        c.x=(a.x+s.x);//必须加括号！！
        c.y=(a.y+s.y);
        return c;
}
Complex operator-(const Complex &a, const Complex &s)
{
        Complex c;
        c.x=(a.x-s.x);
        c.y=(a.y-s.y);
        return c;
}
Complex operator*(const Complex &a, const Complex &b)
{
        Complex s;
    s.x = a.x * b.x - a.y * b.y;
    s.y = a.x * b.y + a.y * b.x;
    return s;
}
Complex operator/(const Complex &a, const Complex &b)
{
        Complex s;
    s.x = (a.x * b.x + a.y * b.y) / (b.x * b.x + b.y * b.y);
    s.y = (a.y * b.x - a.x * b.y) / (b.x * b.x + b.y * b.y);
        return s;
}
Complex::Complex(double x , double y )
{
        this->x=x;
        this->y=y;
        //构造函数不用返回东西呀
}
bool operator==(const Complex &a, const Complex &s)
{
        if(a.x==s.x||a.y==s.y)
        {
                return true;
        }
        else
                return false;
}
bool operator!=(const Complex &a, const Complex &s)
{
        if(a.x!=s.x||a.y!=s.y)
        {
                return true;
        }
        else return false;
}
Complex &Complex:: operator+=(const Complex &c)
{
```

描述

实现以下复数类Complex，通过运算符重载，实现复数的输入输出以及相关运算。

```
class Complex

{

private:

    double x;

    double y;

public:

    Complex(double x = 0.0, double y = 0.0);

    Complex & operator+=(const Complex &);

    Complex & operator-=(const Complex &);

    Complex & operator*=(const Complex &);

    Complex & operator/=(const Complex &);

    friend Complex operator+(const Complex &, const Complex &);

    friend Complex operator-(const Complex &, const Complex &);

    friend Complex operator*(const Complex &, const Complex &);

    friend Complex operator/(const Complex &, const Complex &);

    friend bool operator==(const Complex &, const Complex &);

    friend bool operator!=(const Complex &, const Complex &);

    friend ostream & operator<<(ostream &, const Complex &);

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

};
```

通过以下主函数测试：

```
int main()

{

    Complex c1, c2;

    cin >> c1 >> c2;

    cout << "c1 = " << c1 << "\n" << "c2 = " << c2 << endl;

    cout << "c1+c2 = " << c1 + c2 << endl;

    cout << "c1-c2 = " << c1 - c2 << endl;

    cout << "c1*c2 = " << c1 * c2 << endl;

    cout << "c1/c2 = " << c1 / c2 << endl;

    cout << (c1 += c2) << endl;

    cout << (c1 -= c2) << endl;

    cout << (c1 *= c2) << endl;

    cout << (c1 /= c2) << endl;

    cout << (c1 == c2) << " " << (c1 != c2) << endl;

    return 0;

}
```

输入

输入有两行，每行输入两个表示复数c1和c2的浮点数。

```
	}
	else return false;
}
Complex &Complex:: operator+=(const Complex &c)
{
	x += c.x;
	y += c.y;
	return *this;
	//其实&加不加没什么区别的，因为都是返回那个复数啊
}
Complex& Complex:: operator-=(const Complex &c)
{
	x -= c.x;
	y -= c.y;
	return *this;
}
Complex& Complex:: operator*=(const Complex &c)
{
	*this=*this *c;
	return *this;
}
Complex &Complex:: operator/=(const Complex &c)

{
	*this=*this/c;
	return *this;
}
int main()
{
Complex c1, c2;
cin >> c1 >> c2;
cout << "c1 = " << c1 << "\n" << "c2 = " << c2 << endl;
cout << "c1+c2 = " << c1 + c2 << endl;
cout << "c1-c2 = " << c1 - c2 << endl;
cout << "c1*c2 = " << c1 * c2 << endl;
cout << "c1/c2 = " << c1 / c2 << endl;
cout << (c1 += c2) << endl;
cout << (c1 -= c2) << endl;
cout << (c1 *= c2) << endl;
cout << (c1 /= c2) << endl;
cout << (c1 == c2) << " " << (c1 != c2) << endl;
return 0;
}
```

输入

输入有两行，每行输入两个表示复数c1和c2的浮点数。

输出

输出一共有11行，分别表示复数之间的各项操作，具体参见主函数和输出样例

输入样例 1

```
-4 6
2 5
```

输出样例 1

```
c1 = -4 + 6i
c2 = 2 + 5i
c1+c2 = -2 + 11i
c1-c2 = -6 + 1i
c1*c2 = -38 + -8i
c1/c2 = 0.758621 + 1.10345i
-2 + 11i
-4 + 6i
-38 + -8i
-4 + 6i
0 1
```

提示

复数加法公式：(a + bi) + (c + di) = (a + c) + (b + d)i

复数减法公式：(a + bi) − (c + di) = (a − c) + (b − d)i

复数乘法公式：(a + bi) ∗ (c + di) = (ac − bd) + (ad + bc)i

复数除法公式：(a + bi) / (c + di) = [(ac + bd) / (c ∗ c + d ∗ d)] + [(bc − ad) / (c ∗ c + d ∗ d)]i

Sales_data

2021年4月30日 20:36

```
#include<iostream>
using namespace std;

class Sales_data {

//依次输入书号、销量和收入

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

//依次输出书号、销量、收入和均价

friend ostream & operator<<(ostream &, const Sales_data &);

friend bool operator==(const Sales_data &, const Sales_data &);

friend bool operator!=(const Sales_data &, const Sales_data &);

// for "+", assume that both objects refer to the same book

friend Sales_data operator+(const Sales_data &, const Sales_data &);

public:

Sales_data(): units_sold(0), revenue(0.0) {};

Sales_data(const string & s, unsigned n, double r): bookNo(s), units_sold(n), revenue(r) {};
//unsigned int类型能存储的正数范围比int大一倍，因为unsigned是无符号，它把int类型那个存储符号的一个位置也用来存储数字了。

string get_bookNo() const;//const函数可以使用类中所有成员变量，但是不能修改它们的值而已

// for "+=", assume that both objects refer to the same book

Sales_data & operator+=(const Sales_data &);

private:

double avg_price() const;  //均价，等于收入除以销量

string bookNo;      //书号

unsigned units_sold; //销量

double revenue;     //收入

};
istream & operator>>(istream&is, Sales_data &s)
{
        is>>s.bookNo>>s.units_sold>>s.revenue;
        return is;
}
ostream & operator<<(ostream &os, const Sales_data &s)
{
        os<<s.get_bookNo()<<" "<<s.units_sold<<" "<<s.revenue<<" "<<s.avg_price();
        return os;
}
bool operator==(const Sales_data &a, const Sales_data &b)
{
        if(a.get_bookNo()==b.get_bookNo())
        {
                return true;
        }
        else return false;
}
bool operator!=(const Sales_data &a, const Sales_data &b)
{
        if(a.get_bookNo()!=b.get_bookNo())
        {
                return true;
        }
        else return false;
}
Sales_data operator+(const Sales_data &a, const Sales_data &b)
{
        Sales_data c;
        c.units_sold=a.units_sold+b.units_sold;
        c.bookNo=a.bookNo;
        c.revenue=a.revenue+b.revenue;
        return c;
}
double Sales_data:: avg_price() const
{
        return revenue/units_sold;
}
string Sales_data:: get_bookNo() const
{
        return bookNo;
}
Sales_data &Sales_data:: operator+=(const Sales_data &a)
{
        bookNo=this->bookNo;
        units_sold=this->units_sold+a.units_sold;
        revenue=this->revenue+a.revenue;
        return *this;
}
int main(){

Sales_data item1,item2;

while(cin>>item1>>item2){
```

描述

实现以下Sales_data类（包括它的友元函数）：

```
class Sales_data {

//依次输入书号、销量和收入

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

//依次输出书号、销量、收入和均价

friend ostream & operator<<(ostream &, const Sales_data &);

friend bool operator==(const Sales_data &, const Sales_data &);

friend bool operator!=(const Sales_data &, const Sales_data &);

// for "+", assume that both objects refer to the same book

friend Sales_data operator+(const Sales_data &, const Sales_data &);

public:

Sales_data(): units_sold(0), revenue(0.0) {}

Sales_data(const string & s, unsigned n, double r): bookNo(s), units_sold(n), revenue(r)
{}

string get_bookNo() const;

// for "+=", assume that both objects refer to the same book

Sales_data & operator+=(const Sales_data &);

private:

double avg_price() const;  //均价，等于收入除以销量

string bookNo;      //书号

unsigned units_sold; //销量

double revenue;     //收入

};
```

通过以下main函数的测试

```
int main(){

Sales_data item1,item2;

while(cin>>item1>>item2){

cout<<item1<<"\n"<<item2<<"\n";

if(item1==item2)

cout<<item1.get_bookNo()<<" equals "<<item2.get_bookNo()<<"\n";

if(item1!=item2)

cout<<item1.get_bookNo()<<" doesn't equal "<<item2.get_bookNo()<<"\n";

cout<<(item1+item2)<<"\n";

item1 += item2;

cout<<item1<<"\n";

}

return 0;

}
```

输入

输入多组数据，每组数据两行，每行表示1个Sales_data对象，依次是书号、销量和收入

输出

对于每组数据，输出5行，具体参见main函数和输出样例

输入样例 1

```
001 10 100.0
001 10 100.0
```

输出样例 1

```
001 10 100 10
001 10 100 10
001 equals 001
001 20 200 10
001 20 200 10
```

```
cout<<item1<<"\n"<<item2<<"\n";

if(item1==item2)

cout<<item1.get_bookNo()<<" equals "<<item2.get_bookNo()<<"\n";

if(item1!=item2)

cout<<item1.get_bookNo()<<" doesn't equal "<<item2.get_bookNo()<<"\n";

cout<<(item1+item2)<<"\n";

item1 += item2;

cout<<item1<<"\n";

}

return 0;

}
```

输入样例 2

```
002 5 250
003 8 400
```

输出样例 2

```
002 5 250 50
003 8 400 50
002 doesn't equal 003
002 13 650 50
002 13 650 50
```

来自 <http://www.bjfuacm.com/contest/180/problem/B>

String

2021年5月2日 19:01

两个编译器都死活输出错误，但是oj过了。。。F2

```
#include<iostream>
#include<cstring>
using namespace std;

class String
{
private:
    char* s;
public:
    String();
    String(const char*);
    String(const String&);
    ~String()
    {
        delete[]s;
    };
    String operator=(const char*);
    String& operator=(const String&);
    String operator+(const char*);
    String operator+(const String&);
    String& operator+=(const char*);
    String& operator+=(const String&);
    friend istream& operator>>(istream&, String&);
    friend ostream& operator<<(ostream&, const String&);
    friend bool operator==(const String&, const char*);
    friend bool operator==(const String&, const String&);
    friend bool operator!=(const String&, const char*);
    friend bool operator!=(const String&, const String&);
};
String::String()
{
    s = new char[100];
}
String::String(const char* a)//!!!strlen
{
    s = new char[strlen(a) + 1];
    strcpy(s, a);
}
String::String(const String &a)
{
    s = new char[strlen(a.s) + 1];
    strcpy(s, a.s);
}
String String::operator=(const char* a)
{
    //return operator=(String(a));
```

//你把a传进去，相当于把a给了this指针，然后a就没了！！ 函数只认this不认a，你也不能return a；
//你新开辟的s没有进行指认（你写的不是string s啊）就默认是类里面那个成员了，而不是一个新的区域！

```
//所以strcpy是给数据成员s赋了值，你返回的时候当然要返回这个数据成员
//this指针没有明写，它是隐式的，回去复习吧
        this->s = new char[strlen(a) + 50];
        strcpy(s, a);
        return *this;
}
String& String::operator=(const String &a)
{
    s = new char[strlen(a.s) + 1];
    strcpy(s, a.s);
    return *this;
}
String String::operator+(const String& a)
{
    return String(s) + a.s;
}
String& String:: operator+=(const String &a)
{
    char* r = new char[strlen(s) + strlen(a.s) + 1];
    r = this->s;
    strcat(r, a.s);
    this->s = r;
    return *this;
}
String String::operator+(const char* a)
{
    return String(s) + String(a);
}
String& String::operator+=(const char* a)//是String !!CAPITAL!!!
{
    //return *this += String(a);//为什么不行
    return operator+=(String(a));
    //   delete [] s;
    //   s = new char[strlen(a)+50];
    //   strcpy(s, a);
    //   return *this;
}
bool operator==(const String& a, const String& b)
{
    if (strcmp(a.s, b.s) == 0)
        return true;
    else
        return false;
}
bool operator!=(const String& a, const String& b)
{
    return (strcmp(a.s, b.s) != 0);
}
bool operator==(const String& a, const char* b)
{
    if (strcmp(a.s, b) == 0)
        return true;
    else
```

```
        return false;
}
bool operator!=(const String& a, const char* b)
{
    return (strcmp(a.s, b) != 0);
}
istream& operator>>(istream& is, String& a)
{
    is >> a.s;
    return is;
}
ostream& operator<<(ostream& os, const String& a)
{
    os << a.s;
    return os;
}

int main()
{
    String s;
    s += "hello";
    cout << s << endl;
    String s1("String1");
    String s2("copy of ");
    s2 += "String1";
    cout << s1 << "\n" << s2 << endl;
    String s3;
    cin >> s3;
    cout << s3 << endl;
    String s4("String4"), s5(s4);
    cout << (s5 == s4) << endl;
    cout << (s5 != s4) << endl;
    String s6("End of "), s7("my string.");
    s6 += s7;
    cout << s6 << endl;
    return 0;
}
```

Checked ptr

2021年5月2日　　20:01

```
#include<iostream>
using namespace std;

class CheckedPtr
{
public:
	CheckedPtr(int* b, int* e) : beg(b), end(e), curr(b) {  };
	CheckedPtr& operator ++(); // prefix ++
	CheckedPtr& operator --(); // prefix --
	CheckedPtr   operator ++(int); // postfix ++
	CheckedPtr   operator --(int); // postfix --
	int* GetBeg();
	int* GetEnd();
	int* GetCurr();
private:
	int* beg;  // pointer to beginning of the array
	int* end;  // one past the end of the array
	int* curr; // current position within the array
};
CheckedPtr& CheckedPtr::operator ++()
{
	curr++;
	return*this;
}
CheckedPtr& CheckedPtr::operator--()
{
	curr--;
	return *this;
}
CheckedPtr CheckedPtr::operator++(int)
{
	CheckedPtr s(*this);
	++* this;
	return *this;
}
CheckedPtr CheckedPtr::operator--(int)
{
	CheckedPtr s(*this);
	--* this;
	return *this;
}
int* CheckedPtr::GetBeg()
{
	return beg;
}
int* CheckedPtr::GetEnd()
{
	return end;
}
int* CheckedPtr::GetCurr()
{
	return curr;
}
int main() {
	int n;
	cin >> n;
	int* array = new int[n];
	for (int i = 0; i < n; i++)
		cin >> array[i];

	CheckedPtr cp(array, array + n);
	for (; cp.GetCurr() < cp.GetEnd(); cp++)
		cout << *cp.GetCurr() << " ";
	cout << endl;
	for (--cp; cp.GetCurr() > cp.GetBeg(); cp--)
		cout << *cp.GetCurr() << " ";
	cout << *cp.GetCurr() << endl;
	delete[] array;
	return 0;
}
```

模板

2021年5月21日 21:42

Swap

2021年5月21日 21:40

描述

用模板函数Swap实现对不同类型的数据进行交换。

```
int main()
{
    int a1, a2;
    std::cin >> a1 >> a2;
    Swap(a1, a2);
    std::cout << a1 << "," << a2 << std::endl;

double b1, b2;
    std::cin >> b1 >> b2;
    Swap(b1, b2);
    std::cout << b1 << "," << b2 << std::endl;

char c1, c2;
    std::cin >> c1 >> c2;
    Swap(c1, c2);
    std::cout << c1 << "," << c2 << std::endl;

return 0;
}
```

```
//注意，本题只需要提交Swap函数代码，头文件和main函数系统已经提供。
template<typename T>
void Swap(T &t1, T &t2)
{
    T temp;
    temp = t1;
    t1 = t2;
    t2 = temp;
}
```

SortFunctionTemplate

2021年5月21日 21:43

描述

用模板函数实现数组的输入、排序和输出。并使用如下主函数测试你的模板

输入

输入包含多组测试数据。每组数据为两行，第一行整数type(0、1、2)。第二行为相应数组的5个元素。

输出

对于每一组测试数据，将其排序后在一行内输出，相邻元素逗号空格分离，最后为换行。

输入样例 1

```
0
3 6 1 4 5
1
A B C B A
```

输出样例 1

```
1, 3, 4, 5, 6
A, A, B, B, C
```

```
#include<iostream>
using namespace std;

template<typename T>
void Input(T arr[],int N)
{
    for (int i = 0; i < N; i++)
    {
        cin >> arr[i];
    }
}

template<typename T>
void Sort(T arr[],int N)
{
    for (int i = 0; i < N - 1; i++)
    {
        for (int j = 0; j < N - i - 1; j++)
        {
            if (arr[j] > arr[j + 1])
            {
                int temp = arr[j];
                arr[j] = arr[j + 1];
                arr[j + 1] = temp;
            }
        }
    }
}
template<typename T>
void Output(T arr[],int N)
{
    for (int i = 0; i < N; i++)
    {
        cout << arr[i]<<", ";
    }
    cout << endl;
}

int main()
{
    const int LEN = 5;
    int type;
    while (std::cin >> type)
    {
        switch (type)
        {
        case 0:
        {
            int a1[LEN];
            Input<int>(a1, LEN); Sort<int>(a1, LEN); Output<int>(a1, LEN);
            break;
        }
        case 1:
        {
            char a2[LEN];
            Input(a2, LEN); Sort(a2, LEN); Output(a2, LEN);
            break;
        }
        case 2:
        {
            double a3[LEN];
            Input(a3, LEN); Sort(a3, LEN); Output(a3, LEN);
            break;
        }
        }
    }
    return 0;
}
```

TVector(好题)

2021年5月22日　21:45

TVector3一定记得多次整理改错，今天没时间了

描述

构造一个模板类（Vector），数据成员如下：

```
template<typename T>
class Vector
{
private:
    T x, y, z;
};
```

完成Vector，并用以下函数测试

```
int main()
{
    double a, b, c;
    std::cin >> a >> b >> c;
    Vector<double> v1(a, b, c), v2(v1), v3, v4;
    double d;
    std::cin >> d;
    v4 = d * v1 + v2;

std::cout << v4 <<std::endl;

Vector<double>  v;
    std::cin >> v;

int flag = (v4 == v);
    std::cout << flag << std::endl;

return 0;
}
```

输入

见样例

输出

见样例

输入样例 1

```
3 4 5
2.2
9.6 12.8 16
```

输出样例 1

```
9.6 12.8 16
1
```

```
//张宇航
#include<cmath>
#define N 1e-14
template<typename T>
class Vector
{
        private:
           T x, y, z;
        public:
                Vector(T _x,T _y,T _z);
                Vector();
                template<typename Q>
                friend Vector<Q> operator +(const Vector<Q> &a,const Vector<Q> &b);
                template<typename Q>
                friend Vector<Q> operator *(const Q &a,const Vector<Q> &b);
                template<typename Q>
                friend bool opera.tor ==(const Vector<Q> &s1,const Vector<Q> &s2);
                template<typename Q>
                friend std::istream & operator >>(std::istream & in,Vector<Q> &c);
                template<typename Q>
                friend std::ostream & operator <<(std::ostream & out, const Vector<Q> &c);
};

template<typename T>
Vector<T>::Vector(T _x,T _y,T _z)
{
        x=_x;
        y=_y;
        z=_z;
}

template<typename T>
Vector<T>::Vector()
{}

template<typename Q>
Vector<Q> operator +(const Vector<Q> &a,const Vector<Q> &b)
{
        return Vector<Q>(a.x+b.x,a.y+b.y,a.z+b.z);
}

template<typename Q>
Vector<Q> operator *(const Q &a,const Vector<Q> &b)
{
        return Vector<Q>(a*b.x,a*b.y,a*b.z);
}

template<typename Q>
bool operator ==(const Vector<Q> &s1,const Vector<Q> &s2)
{
        if((fabs(s1.x-s2.x)<N)&&(fabs(s1.y-s2.y)<N)&&(fabs(s1.z-s2.z)<N))
                return 1;
        return 0;
}

template<typename Q>
std::istream & operator >>(std::istream & in,Vector<Q> &c)
{
        in>>c.x>>c.y>>c.z;
        return in;
}

template<typename Q>
std::ostream & operator <<(std::ostream & out, const Vector<Q> &c)
{
        out<<c.x<<" "<<c.y<<" "<<c.z;
        return out;
}
```

//我的，完整代码

```
#include<iostream>
#include<cmath>
#define N 1e-14
using namespace std;

template<class T>
class Vector
{
private:
   T x, y, z;
public:
   Vector(T x1, T y1, T z1);
   Vector();
   template<class A>
   friend Vector<A> operator *(A a, Vector<A> v1);
   template<class A>
   friend Vector<A> operator+(const Vector<A>& v1, const Vector<A>& v2);
   template<class A>
   friend bool operator==(Vector<A> v1, Vector<A> v2);
   template<class A>
   friend std::ostream& operator<<(std::ostream& os, const Vector<A> &v);
   template<class A>
   friend std::istream& operator>>(std::istream& is, Vector<A>& v);
};
template<class T>
Vector<T>::Vector(T x1, T y1, T z1)
{
   x = x1;
   y = y1;
   z = z1;
}
template<class T>
Vector<T>::Vector()
{}
template<class A>
Vector<A> operator+(const Vector<A>& v1, const Vector<A>& v2)
{
   Vector<A> C;
   C.x = v1.x + v2.x;
   C.y = v1.y + v2.y;
   C.z = v1.z + v2.z;
   return C;
}

template<class A>
Vector<A> operator*(A a, Vector<A> v1)
{
   Vector<A> v2;
   v2.x = v1.x * a;
   v2.y = v1.y * a;
   v2.z = v1.z * a;
   return v2;
}

template<class A>
bool operator==(Vector<A> v1, Vector<A> v2)
{
   /*if ((v1.x == v2.x) && (v1.y == v2.y) && (v1.z == v2.z))
      return 0;
   else return 1;*/
   if ((fabs(v1.x - v2.x) < N) && (fabs(v1.y - v2.y) < N) && (fabs(v1.z - v2.z) < N))
      return 1;
   return 0;
}

template<class A>
std::ostream& operator<<(std::ostream& os, const Vector<A> &v)
{
   os << v.x << " " << v.y << " " << v.z;
   return os;
}

template<class A>
std::istream& operator>>(std::istream& is, Vector<A>& v)
{
   is >> v.x >> v.y >> v.z;
   return is;
}
int main()
{
   double a, b, c;
   std::cin >> a >> b >> c;
   Vector<double> v1(a, b, c), v2(v1), v3, v4;
   double d;
   std::cin >> d;
   v4 = d * v1 + v2;

   std::cout << v4 << std::endl;

   Vector<double>  v;
   std::cin >> v;

   int flag = (v4 == v);
   std::cout << flag << std::endl;

   return 0;
}
```

报错：template error: shadows template parm

使用模板时，不能在嵌套作用域中用相同的名称声明模板参数

```
//Error
template<class T>
class linklist
{
        template<class T>  //错误
        class node
        {
        }
}

//Correct
template<class T>
class linklist
{
        template<class U>
        class node
        {
        }
}
```

解答2：

Code:

```
template<class T> class linkedlist{

        template<class T> class lnode{
```

that should be something like

Code:

```
template<class T> class linkedlist{

        template<class U> class lnode{
```

you cannot redeclare template parameters with the same name in nested scopes.

报错：invalid initialization of non-const reference of type......

原因是什么？

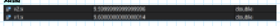
是不是这样的：v1，v2值会存在一个临时变量中，当把这个临时变量传给f时，f的声明中，参数是带&的而不是常量引用。

c++编译器的一个关于语义的限制：如果一个参数是以非const引用传入，c++编译器就有理由认为程序员会在函数中**修改这个值，并且这个被修改的引用在函数返回后要发挥作用。**但如果你把一个临时变量当作非const引用参数传进来，由于临时变量的特殊性，程序员并不能操作临时变量，而且临时变量随时可能被释放掉。所以，一般说来，修改一个临时变量是毫无意义的，据此，c++编译器加入了**临时变量不能作为非const引用**的这个语义限制。

解决办法是在前面加上const或者去掉&符号。

浮点数比较大小的时候不能够直接用==判断

如图



```
v2.x    5.5999999999999996    double
v1.x    5.6000000000000014    double
```

StackClassTemplate

2021年5月25日 17:03

描述

实现一个Stack类模板并测试这一模板

```
template<class T, int SIZE = 20>
class Stack
{
private:
    T    array[SIZE];        //数组，用于存放栈的元素
    int top;                //栈顶位置（数组下标）
public:
    Stack();                //构造函数，初始化栈
    void Push(const T & );  //元素入栈
    T Pop();                //栈顶元素出栈
    void Clear();           //将栈清空
    const T & Top() const;  //访问栈顶元素
    bool Empty() const;     //测试栈是否为空
    bool Full() const;     //测试是否栈满
    int Size();            //返回当前栈中元素个数
};
```

测试函数:

```
int main()
{
    Stack<int, 10> intStack;

int n;
    cin >> n; //n<=10
    for (int i = 0; i < n; i++)
    {
        int temp;
        cin >> temp;
        intStack.Push(temp);
    }

for (int i = 0; i < n; i++)
    {
        cout << intStack.Top() << " ";
        intStack.Pop();
    }
    cout<<endl;

if(intStack.Empty())
        cout<<"Now, intStack is empty."<<endl;

Stack<string,5> stringStack;
    stringStack.Push("One");
    stringStack.Push("Two");
    stringStack.Push("Three");
    stringStack.Push("Four");
    stringStack.Push("Five");
    cout<<"There are "<<stringStack.Size()<<" elements in stringStack."<<endl;
    stringStack.Clear();
    if(stringStack.Empty())
        cout<<"Now, there are no elements in stringStack"<<endl;

return 0;
}
```

输入

参考样例

输出

参考样例

输入样例 1

```
3
1
2
3
```

```
#include<iostream>
using namespace std;

template<typename T, int SIZE = 20>
class Stack
{
private:
    T array[SIZE];     //数组，用于存放栈的元素
    int top;           //栈顶位置（数组下标）
public:
    Stack();           //构造函数，初始化栈
    void Push(const T&);  //元素入栈
    T Pop();           //栈顶元素出栈
    void Clear();      //将栈清空
    const T& Top() const;  //访问栈顶元素
    bool Empty() const;    //测试栈是否为空
    bool Full() const;    //测试是否栈满
    int Size();        //返回当前栈中元素个数
};
template<typename T, int SIZE>
Stack<T, SIZE>::Stack()
{
    top = -1;
}
template<typename T, int SIZE>
bool Stack<T, SIZE>::Empty()const
{
    if (top == -1)
        return true;
    else return false;
}
template<typename T, int SIZE>
void Stack<T, SIZE>::Push(const T&t)
{
    if (top == SIZE - 1)
    {
        cout << "error!" << endl;
    }
    else
    {
        top += 1;
        array[top] = t;
    }
}
template<typename T, int SIZE>
T Stack<T, SIZE>::Pop()
{
    while (top != -1)
    {
        T temp = array[top];
        top -= 1;
        return temp;
    }
}
template<typename T, int SIZE>
void Stack<T, SIZE>::Clear()
{
    while (top != -1)
    {
        T temp = array[top];
        top -= 1;
    }
}
template<typename T, int SIZE>
const T& Stack<T, SIZE>::Top()const
{
    return array[top];
}
template<typename T, int SIZE>
bool Stack<T, SIZE>::Full()const
{
    if (top == 19)
        return true;
    else return false;
}
template<typename T, int SIZE>
int Stack<T, SIZE>::Size()
{
```

输出样例 1

```
3 2 1
Now, intStack is empty.
There are 5 elements in stringStack.
Now, there are no elements in stringStack.
```

```
    return (top+1);
}

int main()
{
    Stack<int, 10> intStack;

    int n;
    cin >> n; //n<=10
    for (int i = 0; i < n; i++)
    {
        int temp;
        cin >> temp;
        intStack.Push(temp);
    }

    for (int i = 0; i < n; i++)
    {
        cout << intStack.Top() << " ";
        intStack.Pop();
    }
    cout << endl;

    if (intStack.Empty())
        cout << "Now, intStack is empty." << endl;

    Stack<string, 5> stringStack;
    stringStack.Push("One");
    stringStack.Push("Two");
    stringStack.Push("Three");
    stringStack.Push("Four");
    stringStack.Push("Five");
    cout << "There are " << stringStack.Size() << " elements in 
stringStack." << endl;
    stringStack.Clear();
    if (stringStack.Empty())
        cout << "Now, there are no elements in stringStack" << endl;

    return 0;
}
```

stl

2021年6月9日星期三　　上午9:30

```
#include<iostream>
#include<string>
#include<vector>
#include<algorithm>
using namespace std;
int main()
{
	int n;
	cin>>n;
	string s;
	vector<string> v;
	for(int i=0;i<n;i++)
	{
		cin>>s;
		sort(s.begin(),s.end());
		v.push_back(s);
	}
	sort(v.rbegin(),v.rend());
	 for(vector<string>::iterator it=v.begin();it!=v.end();it++)
	 {
	 	cout<<*it<<endl;
	 }
	return 0;
}
```

```
#include<iostream>
#include<algorithm>
#include<string>
#include <list>
using namespace std;
struct Student {
    int no;
    string name;
};
void Input(std::list<Student> & li)
{
   int n;
    cin >> n;
    Student s;
    for (int i = 0;i < n;i++)
    {
        cin >> s.no>> s.name;
        li.push_back(s);
    }
}

void Show(list<Student>  li)
{
    for (list<Student>::iterator it = li.begin(); it != li.end(); it++)
        cout << it->no << ", " << it->name << endl;
}
int main()
{
    std::list<Student> li;

    Input(li); //输入
    Show(li); //输出

    return 0;
}
```

```
#include<algorithm>
#include<string>
#include<queue>
#include <list>
#include<iostream>
using namespace std;

int main()
{
	vector<int> v;
	while (1)
	{
		v.clear();/如果不清空的话就会一直往后排呀！！
		int n, q;
		cin >> n >> q;
		for (int i = 0; i < n; i++)
		{
			int temp;
			cin >> temp;
			v.push_back(temp);
		}
		sort(v.begin(), v.end());

		for (int i = 0; i < q; i++)
		{
			int temp;
			cin >> temp;
			vector<int>::iterator iter = v.begin();
			vector<int>::iterator iter2 = lower_bound(v.begin(), v.end(), temp);
			//if (lower_bound(v.begin(), v.end(), temp) != v.end())
			//lower_bound的返回值是下标，而iter（的返回值）是迭代器，这俩根本就不是一个东西，怎么比较呢?
			if(*iter2==temp)
			{
				iter = find(v.begin(), v.end(), temp);
				printf("%d found at %d\n", temp, distance(v.begin(), iter)+1);
			}
			else printf("%d not found \n", temp);
		}
	}
	return 0;
}
```