**设计一个学生类，包含学生学号、姓名、课程、成绩等基本信息，计算学生的平均成绩**

#include <iostream>   
  
#include<string>   
using namespace std;   
  
  
***class Student***   
//定义学生类Student   
{   
public:   
  
**Student(char ID[],char name[],double g1,double g2,double g3)**   
  
{   
  
  
num++;   
  
  
strcpy(this->ID,ID);   
  
  
strcpy(this->name,name);   
  
  
grade1=g1;   
  
  
grade2=g2;   
  
  
grade3=g3;   
  
  
sum1=sum1+g1;   
  
sum2=sum2+g2;   
  
  
sum3=sum3+g3;   
  
}   
  
 **void display()**   
  
{   
  
cout<<ID<<"\t"<<name<<"\t"<<grade1<<"\t"<<grade2<<"\t"<<grade3   
<<endl;   
  
}   
  
**double average1()**   
  
{   
  
return sum1/num;   
  
}   
  
**double average2()**   
  
{   
  
return sum2/num;   
  
}   
  
**double average3()**   
  
{   
  
return sum3/num;   
  
}   
private:   
  
char   
ID[10];   
//学号   
  
char   
name[12];   
//学生姓名   
  
double grade1;   
//功课1成绩   
  
double grade2;   
//功课2成绩   
  
double grade3;   
//功课3成绩   
  
  
static double sum1;   
  
//功课1总分   
  
static double sum2;   
//功课2总分   
  
static double sum3;   
//功课3总分   
  
  
  
static int num;   
  
//学生总人数   
};   
  
**int Student::num=0;**   
**double Student::sum1=0;**   
**double Student::sum2=0;**   
**double Student::sum3=0;**   
  
**int main()**   
{   
  
  
  
Student stu1("200906264","Li Weiwei",88,75,91);   
  
stu1.display();   
  
Student stu2("200902164","Chen Hanfu",86,78,93);   
  
stu2.display();   
  
Student stu3("200908079","Zhan Gaolin",94,69,97);   
  
stu3.display();   
  
  
  
cout<<"The average grade of course1: "<<stu1.average1()<<endl;   
  
cout<<"The average grade of course2: "<<stu2.average2()<<endl;   
  
cout<<"The average grade of course3: "<<stu2.average3()<<endl;   
return 0;   
}

**在一个串中查找一子串，显示出现的次数，将它们全部替换成新串。**

#include <string>   
#include <iostream>   
using namespace std;   
**int main()**   
{   
  
string text="It was Sunday. I never get up early on Sundays. I sometimes   
  
stay in bed until lunch time. Last Sunday I got up very late.";   
  
string OldStr,NewStr;   
  
  
string::iterator itr=text.begin();   
  
int pos1(0),pos2(0);   
//pos1指向文章头,pos2指向找到的字符串   
  
int num(0);   
  
cout<<"Input a string:";   
  
cin>>OldStr;   
  
cout<<"Input a new string:";   
  
cin>>NewStr;   
  
while((pos2=text.find(OldStr,pos1))!=string::npos)   
//如果找到   
  
{   
  
  
text.replace(pos2, OldStr.length(), NewStr);   
  
  
num++;   
  
  
pos1=pos2+NewStr.length();   
  
}   
  
cout<<num<<" string replaced"<<endl;   
  
cout<<"New text is:"<<text<<endl;   
return 0;

}

**统计子字符串substr在字符串str中出现的次数**

#include <iostream>   
using namespace std;   
int SubStrNum (char \*str, char \*substr){   
  
int Num=0; //查找到子字符串的次数   
  
int h1,h2; //分别存储str,substr的长度   
  
int p1,p2; //分别指向str,substr   
  
h1=strlen(str);   
  
h2=strlen(substr);   
  
p1=0;   
  
while(p1<h1)   
// str未到尾   
  
  
{   
  
  
p2=0;   
  
//重新开始比较   
  
while(str[p1]==substr[p2]&&p2<l2&&p1<h1)   
//相等且均没到尾   
  
{   
  
p1++;   
  
p2++;   
  
//指针向后移   
  
}   
  
if(p1==l1&&p2<h2)   
  
//如果str到尾但substr未到尾,结束比较   
  
  
break;   
  
if(p2==h2)   
  
//如果substr到尾,找到一个子串   
  
Num++;   
  
else   
  
// 有字符不相等,指针p1回移   
  
p1=p1-p2+1;   
  
}   
  
return Num;   
}   
int main( ) {   
char str[200];   
  
char substr[20];   
  
cout<<"input source string: ";   
  
cin>>str;   
  
cout<<"input sub string: ";   
  
cin>>substr;   
  
cout<<"match times:"<<SubStrNum (str, substr)<<endl;   
return 0;   
}

**重载数组下标操作符[ ]，使之具有判断与处理下标越界功能。**

#include <iostream>   
using namespace std;   
**class String**   
{   
private:   
  
  
char \*Str;   
  
  
int len;   
public:   
  
**void ShowStr()**   
  
{   
  
cout<<"string:"<<Str<<",length:"<<len<<endl;   
  
}   
  
**String(const char \*p=NULL)**   
  
{   
  
if (p)   
  
{   
  
  
len=strlen(p);   
  
  
Str=new char[len+1];   
  
  
strcpy(Str,p);   
  
} else   
  
{   
  
  
len=0;   
  
  
Str=NULL;   
  
}   
  
}   
  
**~String()**   
  
{   
  
if (Str!=NULL)   
  
  
delete [] Str;   
  
}   
  
**char &operator[](int n)**   
//重载运算符[], 处理String对象   
  
{   
  
  
  
try {   
  
  
if(n<0)   
  
  
throw("Underflow exception!");   
  
  
if(n>=len)   
  
throw("Overflow exception!");   
  
  
else   
  
  
return \*(Str+n);   
  
  
  
  
}   
  
  
catch(const char\*   
s)   
  
  
{   
  
  
cerr<<s<<endl;   
  
  
}   
  
  
return \*Str;   
  
}   
  
};   
**int main()**   
{   
  
  
String S1("0123456789");   
  
S1.ShowStr();   
  
S1[10]='A';   
  
S1[-1]='A';   
  
cout<<" after S1[-1]=A"<<endl;   
S1.ShowStr();   
return 0;   
}

**闰年判断方法**

#include<iostream>   
  
using namespace std;   
**int main( )**   
{   
  
int year,month,day;   
  
bool leap;   
  
cout<<"Input year and month:";   
  
cin>>year>>month;   
  
if((year%400==0)||(year%4==0&&year%100!=0))   
  
leap=true;   
  
else   
  
leap=false;   
  
month=(month%13==0?1:month%13);   
  
switch(month)   
  
{   
  
case 1:   
  
case 3:   
  
case 5:   
  
case 7:   
  
case 8:   
  
case 10:   
  
case 12:   
  
{   
  
day=31;   
  
break;   
  
}   
  
case 4:   
  
case 6:   
  
case 9:   
  
case 11:   
  
{   
  
day=30;   
  
break;   
  
}   
  
case 2:   
  
if(leap)   
//是否闰月   
  
day=29;   
  
  
else   
  
day=28;   
  
break;   
  
}   
  
cout<<"days:"<<day<<endl;   
return 0;   
}

**定义并描述一个人员类Person，它派生出学生类Student和教师类**

#include <iostream>   
  
#include <string>   
using namespace std;   
  
***class Person****//*定义虚基类*Person*   
{   
public:   
  
  
**Person(char \*name, char sex, char \*ID, char \*birthday)**   
  
  
{   
  
  
  
strcpy(this->name,name);   
  
  
  
this->sex=sex;   
  
  
strcpy(this->ID,ID);   
  
  
strcpy(this->birthday,birthday);   
  
  
}   
  
  
**void display()**   
  
  
{   
  
  
  
cout<<"name:"<<name<<endl;   
  
  
  
cout<<"sex:"<<sex<<endl;   
  
  
  
cout<<"ID:"<<ID<<endl;   
  
  
  
cout<<"birthday:"<<birthday<<endl;   
  
  
}   
private:   
  
  
char name[20];   
  
//姓名   
  
  
  
char sex;   
  
  
//性别   
  
  
char ID[20];   
  
  
//身份证号   
  
  
char birthday[12];   
//出生年月   
};   
  
***class Teacher :virtual public Person****//*定义派生类*Teacher*   
{   
public:   
  
  
  
**Teacher(char \*name,char sex,char \*ID,char \*birthday,char \*position)**   
 **:Person(name,sex,ID,birthday)**   
  
  
  
{   
  
  
  
  
strcpy(this->position,position);   
  
  
  
}   
  
  
 **void display()**   
  
  
  
{ cout<<"position:"<<position<<endl;}   
  
private:   
  
  
  
char position[12];   
//职称   
};   
  
***class Student :virtual public Person*** *//*定义派生类*Student*   
{   
public:   
  
  
  
**Student(char \*name,char sex,char \*ID,char \*birthday,char \*major)**   
 **:Person(name,sex,ID,birthday)**   
  
  
  
{   
  
  
  
  
strcpy(this->major,major);   
  
  
  
}   
  
  
  
**void display()**{ cout<<"major:"<<major<<endl; }   
  
private:   
  
  
  
char major[20];   
  
  
  
  
//专业   
};   
  
***class Stu\_Teach:public Teacher,public Student***   
{   
public:   
  
  
  
Stu\_Teach(char \*name,char sex,char \*ID,char \*birthday,char \*position, char \*major)   
  
  
:Person(name,sex, ID, birthday),Student(name,sex,ID,birthday, major),Teacher(name,sex,ID, birthday,position)   
  
  
  
{ }   
  
  
  
void display()   
  
  
  
{   
  
  
  
  
Person::display();   
  
  
  
  
Student::display();   
  
  
  
  
Teacher::display();   
  
  
  
}   
};   
  
**int main()**   
{   
  
  
  
Stu\_Teach ST("Liu\_xiaopeng",'M',"428120198005272487","1976-05-27",   
  
  
  
"Professor", "Computer Science");   
  
  
  
ST.display();   
return 0;   
}

**利用STL算法和迭代器，编程实现堆排序。**

#include <iostream>   
#include <algorithm>   
#include <vector>   
using namespace std;   
  
int main( ) {   
  
const int SIZE=10;   
  
int a[SIZE]={10,3,8,11,20,7,19,5,16,1};   
  
vector <int> v(a,a+SIZE);   
  
ostream\_iterator <int> output(cout," ");   
      
  
cout<<"堆排序之前："<<endl;   
  
copy(v.begin(),v.end(),output);   
      
  
//创建堆   
  
make\_heap(v.begin(),v.end());   
  
cout<<"\n建堆之后："<<endl;   
  
copy(v.begin(),v.end(),output);   
  
  
//堆排序   
  
sort\_heap(v.begin(),v.end());   
  
cout<<"\n堆排序后："<<endl;   
  
copy(v.begin(),v.end(),output);   
  
  
return 0;   
}

**设计一个类模板，其中包括数据成员T a[n]以及对其进行排序的成员函数sort()**

#include <iostream>   
using namespace std;   
**template <class T, int n>**  
***class Array***   
{   
  
private:   
  
T a[n];   
  
  
public:   
  
  
**void sort(double)**   
  
//从小到大排序            
  
  
{   
  
  
for (int i=0;i<n-1;i++)   
  
  
for (int j=i+1;j<n;j++)   
//从待排序序列中选择一个最小的数组元素   
  
  
if (a*>a[j])   
  
  
{   
  
  
  
T t;   
  
  
t=a;   
  
//交换数组元素   
  
  
a=a[j];   
  
  
a[j]=t;   
  
  
}   
  
  
  
}****void sort(char \*)*** *//重载sort()对char \*类型数据排序   
  
  
{   
  
  
for (int i=0;i<n-1;i++)   
  
  
  
for (int j=i+1;j<n;j++)   
//从待排序序列中选择一个最小的数组元素   
  
  
if (a[j]==NULL||(a!=NULL&&strcmp(a,a[j])>0))   
  
  
{//短路求值避免strcmp(NULL)   
  
  
  
T t;   
  
  
t=a;   
  
//交换指针   
  
  
a=a[j];   
  
  
a[j]=t;   
  
  
}   
  
  
  
}* ***void disp()*** *{   
  
  
for (int i=0;i<n-1;i++)   
  
if(a!=NULL)   
//避免对NULL输出   
  
  
cout<<a<<",\t";   
  
else   
  
  
cout<<",\t";   
  
  
if(a!=NULL)   
  
cout<<a<<endl;   
  
  
else   
  
  
cout<<endl;   
  
  
}****Array(T a[])*** *{   
  
  
for(int i=0;i<n;i++)   
  
this->a=a;   
  
  
  
}   
  
};****int main()*** *{   
  
  
float f[]={1.2,2.3,7.7,4,3.4,2.3};   
  
Array<float, 6> a1(f);   
  
a1.sort(f[0]);   
  
a1.disp();   
  
char \*   
a[6]={"a","abc","ABC","abcd"};   
  
Array<char \*   
,6> a2(a);   
  
a2.sort(a[0]);   
  
a2.disp();   
  
char c[]={'a','b','C','2'};   
  
Array<char, 6> a3(c);   
  
a3.sort(c[0]);   
  
a3.disp();   
  
return 0;   
}*

**交通工具类vehicle，将它作为基类派生小车类car、卡车类truck和轮船类boat**

#include <iostream>   
  
using namespace std;   
***class Vehicle***  *//*定义基类*Vehicle*   
{   
public:   
  
**Vehicle()** {   
  
  
cout<<"Vehicle constructor..."<<endl;   
  
}   
  
**~Vehicle()**{   
  
  
cout<<"Vehicle destructor..."<<endl;   
  
}   
  
virtual void display() const=0;   
};   
  
***class Car :public Vehicle***  *//*定义派生类*Car*   
{   
public:   
  
**Car()** {   
  
  
cout<<"Car constructor..."<<endl;   
  
}   
  
**~Car()**{   
  
  
cout<<"Car destructor..."<<endl;   
  
};   
  
**void display() const**   
  
{   
  
  
cout<<"This is a car!"<<endl;   
  
}   
};   
  
***class Truck :public Vehicle*** *//*定义派生类*Truck*   
{   
public:   
  
**Truck()**   
{   
  
  
  
cout<<"Truck constructor..."<<endl;   
  
  
}   
  
**~Truck()** {   
  
  
cout<<"Truck destructor..."<<endl;   
  
}   
  
**void display() const** {   
  
  
cout<<"This is a truck!"<<endl;   
  
}   
};   
  
***class Boat :public Vehicle*** *//*定义派生类*Boat*   
{   
public:   
  
**Boat()** {   
  
  
  
cout<<"Boat constructor..."<<endl;   
  
  
}   
  
**~Boat()**   
{   
  
  
cout<<"Boat destructor..."<<endl;   
  
}   
  
**void display() const**{   
  
  
cout<<"This is a Boat!"<<endl;   
  
}   
};   
  
**int main()**   
{   
  
Vehicle \*V;   
  
V=new Car;   
  
V->display();   
  
delete V;   
  
  
V=new Truck;   
  
V->display();   
  
delete V;   
  
  
V=new Boat;   
  
V->display();   
  
delete V;   
return 0;   
}

**编写一程序，统计一篇英文文章中单词的个数与行数。**

#include <fstream>   
#include <iostream>   
using namespace std;   
**int words(char** \***line)**   
//统计一行文章中的单词数   
{   
  
//单词为含26个大小写英文字母的字符串   
  
int words=0;   
  
bool PreChar=false;   
//前一个字符是否为英文字母   
  
bool CurChar=false;   
//当前字符是否为英文字母   
  
for(int i=0;i<strlen(line);i++)   
  
{   
  
if(line*>='A'&&line<='Z'||line>='a'&&line<='z')   
  
CurChar=true;   
  
else   
  
CurChar=false;   
  
if(!PreChar&&CurChar)   
  
words++;   
  
PreChar=CurChar;   
  
}   
  
return words;   
}****int main()*** *{   
  
const int N=1024;   
  
char line[N];   
//存储文章的一行   
  
int lines=0,TotalWords=0;   
  
  
char filename[40];   
  
ifstream txtfile;   
  
cout<<"text file:"<<endl;   
  
cin>>filename;   
  
txtfile.open(filename);   
  
if(!txtfile) {   
  
cerr<<"File open error!"<<endl;   
  
exit(1);   
  
}   
  
while(txtfile.getline(line,N)) {   
  
TotalWords+=words(line);   
  
lines++;   
  
  
}   
  
txtfile.close();   
  
cout<<"total lines: "<<lines<<endl;   
  
cout<<"total words: "<<TotalWords<<endl;   
return 0;   
}*

## 设计一个程序，两个数相除，精度由小到大输出

#include<iomanip>

#include<iostream>

using namespace std;

int main()

{

double a, b;

while(cin>>a>>b)

{

for(int i = 1; i <= 20; i ++)

cout<<setprecision(i)<<a/b<<endl;

}

return 0;

}