//编写一个C++的程序，输入你的年龄，然后在屏幕上输出年龄，最后将源程序以myprog.cpp文件名存盘

#include <iostream.h>

void main()

{

int i;

cout<<"How old are you?\n"; // 用C++特有的方式输出数据

cin>>i; // 用C++特有的方式输入数据

cout<<"You are "<<i<<" years old.\n";

}

//2-4-1编写程序，提示用户输入三角形的三条边长，判断该三角形是否为直角三角形，若是输出结果以及三角形面积。

#include <iostream.h>

void main()

{

float a,b,c,side1,side2,area; //side1和side2为两条直角边长度

cout<<"please input length of three sides of the triangle: \n"; // 输出提示信息

cin>>a>>b>>c; // 输入三角形三边长度

if ((a\*a==b\*b+c\*c)||(c\*c==a\*a+b\*b)||(b\*b==a\*a+c\*c))

//判断是否为直角三角形

{

if (a\*a==b\*b+c\*c)

//判断三边中哪两条边为直角边,并存储到side1和side2中

{

side1=b;

side2=c;

}else

{

if (c\*c==a\*a+b\*b)

{

side1=a;

side2=b;

}else

{

side1=a;

side2=c;

}

}

area=side1\*side2/2;//计算直角三角形的面积

cout<<"It is a right-angled triangle and the area is "<<area<<".\n";//输出判断结果及直角三角形面积

}else

cout<<"It is not a right-angled triangle.\n";

}

//2-4-2编写程序，求解各种数据类型的存储长度并显示出来，在其中找出存储长度最大的和最小的两种数据类型并输出。

#include <iostream.h>

void main()

{

int length[7];

int max=0,min=0;

cout<<"data type\tmemory used(bytes)";

length[0]=sizeof(short int); //获取短整型长度

cout<<"\nshort int\t"<<length[0]<<"\t";

length[1]=sizeof(int); //获取整型长度

cout<<"\ninteger \t"<<length[1];

length[2]=sizeof(long); //获取长整型长度

cout<<"\nlong integer\t"<<length[2];

length[3]=sizeof(char); //获取字符型长度

cout<<"\nchar \t" <<length[3];

length[4]=sizeof(float); //获取单浮点型长度

cout<<"\nfloat \t"<<length[4];

length[5]=sizeof(double); //获取双浮点型长度

cout<<"\ndouble \t"<<length[5];

length[6]=sizeof(bool); //获取布尔型长度

cout<<"\nbool \t"<<length[6]<<endl;

for(int i=0;i<7;i++)

{

if (length[i]>length[max])//求取长度最大的类型的存取位置

max=i;

if (length[i]<length[min])//求取长度最小的类型的存取位置

min=i;

}

cout<<"The longest length is from ";

switch (max)

{

case 0:cout<<"short int."<<endl;

break;

case 1:cout<<"int."<<endl;

break;

case 2:cout<<"long."<<endl;

break;

case 3:cout<<"char."<<endl;

break;

case 4:cout<<"float."<<endl;

break;

case 5:cout<<"double."<<endl;

break;

case 6:cout<<"bool."<<endl;

break;

}

cout<<"The shortest length is from ";

switch (min)

{

case 0:cout<<"short int."<<endl;

break;

case 1:cout<<"int."<<endl;

break;

case 2:cout<<"long."<<endl;

break;

case 3:cout<<"char."<<endl;

break;

case 4:cout<<"float."<<endl;

break;

case 5:cout<<"double."<<endl;

break;

case 6:cout<<"bool."<<endl;

break;

}

}

//2-4-3（3）编写程序输入一个华氏温度，将其转换为摄氏温度并输出。

#include <iostream.h>

void main()

{

float C; //变量C为摄氏温度

float F; //变量F为华氏温度

cout<<"请输入华氏温度(float类型):\n";

cin>>F; //输入华氏温度

C=(F-32)\*5/9; //华氏温度转换为摄氏度

cout<<"转换为摄氏温度为 "<<C<<endl;

}

//2-4-4编写程序输入一个十进制表示的正整数，将其转化为二进制表示并输出结果。

#include <iostream.h>

void main()

{

int a; //变量C为摄氏温度

int b[20]; //数组存储转换后的二进制数

int i;

i=0;

cout<<"请输入一个十进制的数字:\n";

cin>>a; //输入十进制数

while (a>0) //转换过程

{

b[i]=a%2;

i++;

a=(int)(a/2);

}

i--;

cout<<"转换的二进制数为： ";

while (i>=0) //输出二进制结果

{

cout<<b[i];

i--;

}

cout<<endl;

}

//3-4-1编写程序，计算1到100中所有3的倍数的数的和。

#include <iostream.h>

void main()

{

int sum=0; //sum变量为3的倍数的和

for(int i=1;i<=100;i++)

{

if (i%3==0) //判断i是否是3的倍数

sum+=i;

}

cout<<"1到100中所有3的倍数的数的和为:"<<sum<<endl;

}

//编写程序，用户输入一些整数，该程序分别计算出所有奇数和所有偶数之和，并输出它们。

#include <iostream.h>

void main()

{

int sum\_odd=0; //变量为所有奇数的和

int sum\_even=0; //变量为所有偶数的和

int b; //变量为输入的数字

int N; //变量为数字个数

cout<<"输入数字的总的个数为";

cin>>N;

cout<<"请输入数字(数字以空格隔开)"<<endl;

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

{

cin>>b;

if (b%2==0) //判断数字是否为偶数

sum\_even+=b;

else

sum\_odd+=b;

}

cout<<"所有奇数的和为"<<sum\_odd<<endl;

cout<<"所有偶数的和为"<<sum\_even<<endl;

}

//求解输入两个正整数的最大公约数和最小公倍数。

#include <iostream.h>

void main()

{

int a,b; //输入的两个正整数

int min,max; //最小值和最大值

cout<<"请输入两个正整数(数字以空格隔开)"<<endl;

cin>>a>>b;

if (a>=b)//找出两个数中的最大值和最小值

{

min=b;

max=a;

}

else

{

min=a;

max=b;

}

//最大公约数一定不大于两个数中的最小值

while (min>0)

{

if ((a%min==0)&&(b%min==0))

break;

else

min--;

}

cout<<a<<"和"<<b<<"的最大公约数为"<<min<<endl;

//最小公倍数一定不小于两个数中的最大值

while ((max%a!=0)||(max%b!=0))

{

max++;

}

cout<<a<<"和"<<b<<"的最小公倍数为"<<max<<endl;

}

//编写程序计算10个正整数的平均值、方差以及标准方差。

#include <iostream.h>

#include <math.h>

#define number 10

void main()

{

int a[number]; //输入的10个正整数

int average,sum=0; //平均值和总和

int variance=0; //方差

int mvariance; //均方差

cout<<"请输入"<<number<<"个整数："<<endl;

for (int i=0;i<number;i++) //求总和

{

cin>>a[i];

sum+=a[i];

}

i--;

average=sum/number; //求均值

cout<<"average="<<average<<endl;

while (i>=0) //求方差

{

variance += abs(a[i]\*a[i]-average\*average);

i--;

}

mvariance=variance/number; //求均方差

cout<<"variance="<<variance<<endl;

cout<<"mvariance="<<mvariance<<endl;

}

//输入4个字母，并反向显示这些字母。

#include <iostream.h>

void main()

{

char a[4]; //输入4个字符

cout<<"请输入4个字符："<<endl;

for (int i=0;i<4;i++)

cin>>a[i];

cout<<"反向输出4个字符为:"<<endl;

for (i=3;i>=0;i--)

cout<<a[i];

cout<<endl;

}

//输出所有的"水仙花数"。

//"水仙花数"是指一个3位数，其各位数字的立方和等于该数本身。

#include <iostream.h>

void main()

{

int N,M;

int a[3]; //存储三位数的个十百位

cout<<"水仙花数为:"<<endl;

for (N=100;N<=999;N++)

{

M=N;

for (int j=0;j<3;j++) //提取数字的个十百位

{

a[j]=M%10;

M/=10;

}

if (a[0]\*a[0]\*a[0]+a[1]\*a[1]\*a[1]+a[2]\*a[2]\*a[2]==N)

cout<<N<<" ";

}

}

//求 1!+2!+…+50!。

#include <iostream.h>

void main()

{

int sum1=1;//记录i!的结果

int sum2=0;//记录i!累加的结果

for (int i=1;i<=50;i++)

{

for (int j=1;j<=i;j++) //计算i!

sum1\*=j;

sum2+=sum1;

}

cout<<"1!+2!+…+50!="<<sum2<<endl;

}

//编写程序求一元二次方程ax\*x+bx+c=0的解。

#include <iostream.h>

#include <math.h>

void main()

{

float a,b,c;//方程的系数

float s1,s2;//解

float temp;

cout<<"Input a,b,c:"<<endl;

cin>>a>>b>>c;

if (a==0)

{

cout<<"不是一元二次方程!"<<endl;

}else

{

temp=b\*b-4\*a\*c;

if (temp<0) //无根的情况

cout<<"无实根!"<<endl;

else if (temp==0) //只有一个实根的情况

{

s1=-b/(2\*a);

cout<<"方程有一个实根,为"<<s1<<endl;

}

else //两个实根的情况

{

s1=(-b+sqrt(temp))/(2\*a);

s2=(-b-sqrt(temp))/(2\*a);

cout<<"方程有两个实根,为"<<s1<<"和"<<s2<<endl;

}

}

}

//3-4-9编写程序，用循环语句打印如下图案。

#include <iostream.h>

#include <math.h>

#include <iomanip.h>

void main()

{

for (int i=1;i<=7;i++)

{

cout<<setw(abs(4-i)+1)<<"@";//显示第一个@

if ((i==1)||(i==7)) //第一行和最后一行仅显示一个@

{

cout<<endl;

continue;

}

for (int m=abs(4-i)+2;m<=6-abs(4-i);m++) //显示中间的\*

cout<<"\*";

cout<<"@"<<endl; //中间几行显示第二个@

}

}

//3-4-10编写程序，输入年月日信息，并输出这一天为这一年的第几天，注意闰年问题。

#include <iostream.h>

void main()

{

int year,month,day;

int num=0;

int length;

cout<<"Input year:";

cin>>year;

cout<<"Input month:";

cin>>month;

cout<<"Input day:";

cin>>day;

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

{

if((i==1)||(i==3)||(i==5)||(i==7)||(i==8)||(i==10)||(i==12))

//大月31天

length=31;

else if ((i==4)||(i==6)||(i==9)||(i==11))//小月30天

length=30;

else

if (((year%100==0)&&(year%4==0))||((year%100!=0)&&(year%4==0)))//闰年判断条件

length=29;//闰年的二月29天

else

length=28;//非闰年的二月28天

num+=length;

}

num+=day;

cout<<year<<"年"<<month<<"月"<<day<<"日为这一年的第:"<<num<<"天."<<endl;

}

//3-4-11编写程序，由用户输入x值，计算函数值并输出y。函数如下所示：

#include <iostream.h>

void main()

{

int x,y;

cout<<"Input X(int):";

cin>>x;

if (x<0) //x<0的情况

y=x;

else if (x>10) //x>10的情况

y=4\*x\*x\*x-x\*x;

else //其余的情况

y=x\*x;

cout<<"函数结果y为:"<<y<<endl;

}

//3-4-12鸡兔同笼问题。若鸡兔共有100只脚，利用循环计算鸡兔各几只。

#include <iostream.h>

void main()

{

int rabbit,chicken;

for (rabbit=100/4;rabbit>=0;rabbit--)//兔子最多有25只,最少没有

{

chicken=(100-rabbit\*4)/2;

cout<<"兔子有"<<rabbit<<"只,鸡有"<<chicken<<"只"<<endl;

}

}

//4-5-1编写一个函数，计算直角坐标系中点a(x0,y0)到点b(x1,y1)的距离。

#include <iostream.h>

#include <math.h>

float length(float x0,float y0,float x1,float y1)//求两点距离的子函数

{

return sqrt((x0-x1)\*(x0-x1)+(y0-y1)\*(y0-y1));;

}

void main()

{

float x0,x1,y0,y1;

cout<<"Input x and y of a point:";

cin>>x0>>y0;

cout<<"Input x and y of b point:";

cin>>x1>>y1;

cout<<"The length from a to b is "<<length(x0,y0,x1,y1)<<endl;

}

//4-5-2求a!+b!+c!的值，其中求n!要用一个函数实现，通过主函数输入a、b和c的值，并在主函数中输出计算的结果。

#include <iostream.h>

int factorial(int n)//求两点距离的子函数

{

int sum=1;

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

sum\*=i;

return sum;

}

void main()

{

int a,b,c;

cout<<"Input a,b,c:";

cin>>a>>b>>c;

cout<<"a!+b!+c!="<<factorial(a)+factorial(b)+factorial(c)<<endl;

}

//4-5-3编写一个函数。该函数读入一个整数，并判断这个整数是否为一个回文数字。例如4，44，434，4334，43534都是回文数字。

#include <iostream.h>

bool palindrome(int n)//判断n是否为回文的子函数

{

int a[20];

int m=n;

int i=0;

int temp,num;

while (m>0)//将数字的各位反向放置在数组a中

{

a[i]=m%10;

m=m/10;

i++;

}

temp=i/2;//需要比较的次数

num=--i;//数字的最高位存储在a[num]中

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

{

if (a[j]!=a[num-j]) return false;//如果出现不相等的情况立即退出

}

return true;//若比较的各个位置都相等,则为回文

}

void main()

{

int number;

cout<<"Input the number:";

cin>>number;

if (palindrome(number))

cout<<number<<" is a palindrome."<<endl;

else

cout<<number<<" is not a palindrome."<<endl;

}

//4-5-4编写一个程序，为选修3、4和5门课程的学生计算平均分，其中求平均分要用重载函数实现

#include <iostream.h>

float average(float a,float b,float c)//求三门课程平均分子函数

{

return (a+b+c)/3;

}

float average(float a,float b,float c,float d)//求四门课程平均分子函数

{

return (a+b+c+d)/4;

}

float average(float a,float b,float c,float d,float e)//求五门课程平均分子函数

{

return (a+b+c+d+e)/5;

}

void main()

{

int number;

float s1,s2,s3,s4,s5;

cout<<"The total of student's courses is ";

cin>>number;

if (number==3)

{

cout<<"Input three score of student:";

cin>>s1>>s2>>s3;

cout<<"The average is "<<average(s1,s2,s3)<<endl;

}

else if (number==4)

{

cout<<"Input four score of student:";

cin>>s1>>s2>>s3>>s4;

cout<<"The average is "<<average(s1,s2,s3,s4)<<endl;

}

else

{

cout<<"Input three score of student:";

cin>>s1>>s2>>s3>>s4>>s5;

cout<<"The average is "<<average(s1,s2,s3,s4,s5)<<endl;

}

}

//4-5-5用递归方法将一个整数n转换成字符串。

#include <iostream>

#include <string>

using namespace std;

string convert(int n);

string numstring("");

void main()

{

int number;

cout<<"Input the number: ";

cin>>number;

cout<<"converted string is ";

cout<<convert(number)<<endl;

}

string convert(int n)//递归函数

{

if ((n/10==0)&&(n==0))

return "";

else

numstring=convert(n/10).append(string(1,(n%10+48)));//将各位数转换为字符后创建字符串，加到最后

return numstring;

}

//4-5-6编写一个函数。该函数读入一个整数，然后将这个整数上每个位的数字按照相反的顺序输出。

//例如输入的整数为12345，输出结果为54321。

#include <iostream.h>

void reverse(int n)//反向显示整数n

{

int a[20];//存储n的各位，最多20位

int m=n;

int i=0;

while (m>0)//提取n的各位

{

a[i]=m%10;

m=m/10;

i++;

}

cout<<"reversed number is ";

for (int j=0;j<=i-1;j++)//反向输出各位

cout<<a[j];

cout<<endl;

}

void main()

{

int number;

cout<<"Input the number: ";

cin>>number;

reverse(number);

}

//4-5-7输入三个数字，数字可以为整形或浮点型，分别编写函数来求解三个数字的最大值、最小值和平均值，要求在主函数中完成数字的输入和计算结果的输出。

#include <iostream.h>

#include <iomanip.h>

float fmax,fmin,faverage;

int max,min,average;

void calculate(float fa,float fb,float fc) //浮点型处理函数

{

fmax=fa>=fb?fa:fb;

fmax=fmax>=fc?fmax:fc;

fmin=fa<=fb?fa:fb;

fmin=fmin<=fc?fmin:fc;

faverage=(fa+fb+fc)/3;

}

void calculate(int a,int b,int c) //整型处理函数

{

max=a>=b?a:b;

max=max>=c?max:c;

min=a<=b?a:b;

min=min<=c?min:c;

average=(a+b+c)/3;

}

void main()

{

int s1,s2,s3;

float f1,f2,f3;

int type;

cout<<"Input the type: float(1) or int(0):";

cin>>type;

while ((type!=0)&&(type!=1)) //输入类型错误时再次输入

{

cout<<"Please input 1 or 0:";

cin>>type;

}

cout<<"Input three numbers:";

if (type==1) //浮点型数据

{

cin>>f1>>f2>>f3;

calculate(f1,f2,f3);

cout<<setw(8)<<"max"<<setw(8)<<"min"<<setw(12)<<"average"<<endl;

cout<<setw(8)<<fmax<<setw(8)<<fmin<<setw(12)<<faverage<<endl;

}

else //整型数据

{

cin>>s1>>s2>>s3;

calculate(s1,s2,s3);

cout<<setw(5)<<"max"<<setw(8)<<"min"<<setw(8)<<"average"<<endl;

cout<<setw(5)<<max<<setw(8)<<min<<setw(8)<<average<<endl;

}

}

//5-4-1编写函数，完成指定二维数组（3\*3）的转置，即行列对换。

#include <iostream.h>

#include <iomanip.h>

void main()

{

int i,j,temp,row=0,colum=0;

int a[3][3];

cout<<"输入一个3×3的整型矩阵："<<endl;

for (i=0;i<=2;i++) //从第0行~第2行

for (j=0;j<=2;j++) //从第0列~第2列

cin>>a[i][j]; //输入矩阵元素

for (i=0;i<=2;i++)

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

{

//将a[i][j]与a[j][i]的值互换

temp=a[i][j];

a[i][j]=a[j][i];

a[j][i]=temp;

}

for (i=0;i<=2;i++)

{

for (j=0;j<=2;j++)

cout<<setw(8)<<a[i][j]; //输入矩阵元素

cout<<endl;

}

}

//5-4-2编写一个程序，要求当输入一个数字月份时，程序输出该月的英文名称。例如输入5时，程序输出"May"，要求用指针数组实现。

#include <iostream.h>

#include <string>

void main()

{

//定义指针数组

char \*month[12]={"January","February","March","April","May","June","July","August","September","October","November","December"};

int n;

cout<<"Input the month(1-12)：";

cin>>n;

while ((n>12)||(n<1))

{

cout<<"Input the month(1-12):";

cin>>n;

}

cout<<"The month is "<<\*(month+n-1)<<endl;

}

//5-4-3编写一个程序，要求分别输入5个学生的3科成绩，并输出平均成绩最高的学生的姓名及各科成绩。要求用结构体数组实现。

#include <iostream.h>

#include <string>

struct student

{

char name[20]; //姓名

float course1; //第一科成绩

float course2; //第二科成绩

float course3; //第三科成绩

float average; //平均成绩

};

struct student stu[5];

void main()

{

float temp=-100.0;

int j,k=0;

char c;

for (int i=0;i<5;i++)

{

cout<<"Input NO."<<i+1<<" student's name: ";

cin>>stu[i].name;

cout<<"Input NO."<<i+1<<" student's three scores: ";

cin>>stu[i].course1>>stu[i].course2>>stu[i].course3;//输入三科成绩

stu[i].average=(stu[i].course1+stu[i].course2+stu[i].course3)/3;//求平均成绩

if (temp<stu[i].average)

{

temp=stu[i].average; //temp存放最高平均分

k=i; //k存放该同学的序号

}

}

cout<<"The student is "<<stu[k].name<<", three scores are "<<stu[k].course1<<", "<<stu[k].course2<<" and "<<stu[k].course3<<endl;

}

//5-4-4编写一个程序，实现方程式的相加。

#include <iostream.h>

#include <string>

struct coefficient

{

bool pn;

int num;

}a[3],b[3],c[3],d[3];

void main()

{

char equation[2][50],ch,z[8];

int m,j;

cout<<"Input NO. 1 Equation:"<<endl;

cin>>equation[0];

cout<<"Input NO. 2 Equation:"<<endl;

cin>>equation[1];

for (int i=0; i<2;i++)

{

j=0;

ch=equation[i][0];

if (ch=='-')

{

a[i].pn=false;

j++;

ch=equation[i][j];

}

else

a[i].pn=true;

if(ch>='0'&&ch<='9') /\* 提取a的系数 \*/

{

m=0;

do

{

z[m]=ch;

j++;

m++;

ch=equation[i][j];

}while(ch>='0'&&ch<='9');

z[m]=0;

if (a[i].pn)

a[i].num=atoi(z);

else

a[i].num=-1\*atoi(z);

}

j++;

ch=equation[i][j];

if (ch=='-')

b[i].pn=false;

else

b[i].pn=true;

j++;

ch=equation[i][j];

if(ch>='0'&&ch<='9') /\* 提取b的系数 \*/

{

m=0;

do

{

z[m]=ch;

j++;

m++;

ch=equation[i][j];

}while(ch>='0'&&ch<='9');

z[m]=0;

if (b[i].pn)

b[i].num=atoi(z);

else

b[i].num=-1\*atoi(z);

}

j++;

ch=equation[i][j];

if (ch=='-')

c[i].pn=false;

else

c[i].pn=true;

j++;

ch=equation[i][j];

if(ch>='0'&&ch<='9') /\* 提取c的系数 \*/

{

m=0;

do

{

z[m]=ch;

j++;

m++;

ch=equation[i][j];

}while(ch>='0'&&ch<='9');

z[m]=0;

if (c[i].pn)

c[i].num=atoi(z);

else

c[i].num=-1\*atoi(z);

}

j++;

j++;

ch=equation[i][j];

if (ch=='-')

{

d[i].pn=false;

j++;

ch=equation[i][j];

}

else

d[i].pn=true;

if(ch>='0'&&ch<='9') /\* 提取=右侧的常数 \*/

{

m=0;

do

{

z[m]=ch;

j++;

m++;

ch=equation[i][j];

}while(ch>='0'&&ch<='9');

z[m]=0;

if (d[i].pn)

d[i].num=atoi(z);

else

d[i].num=-1\*atoi(z);

}

}

//计算相加后a,b,c的系数和常数的结果

a[2].num=a[0].num+a[1].num;

b[2].num=b[0].num+b[1].num;

c[2].num=c[0].num+c[1].num;

d[2].num=d[0].num+d[1].num;

cout<<"The added equation is :"<<endl;

if (a[2].num!=0)

cout<<a[2].num<<"a";

if (b[2].num>0)

cout<<"+"<<b[2].num<<"b";

else if (b[2].num<0)

cout<<b[2].num<<"b";

if (c[2].num>0)

cout<<"+"<<c[2].num<<"c";

else if (c[2].num<0)

cout<<c[2].num<<"c";

cout<<"="<<d[2].num<<endl;

}

//5-4-5编写一个函数，利用指针将一个字符串反向输出。编写相应的主函数对其进行调用，在主函数中完成原始数据的输入和结果输出。

#include <iostream.h>

void reverse(char \*a, int n)//a为字符串首地址,n+1为字符串长度

{

char temp;

for(int k=0;k<((n+1)/2);k++)//a[k]和a[n-k]交换数据

{

temp=\*(a+k);

\*(a+k)=\*(a+n-k);

\*(a+n-k)=temp;

}

}

void main()

{

char a[100]; //最多输入100个字符

cout<<"Input the string："<<endl;

cin.getline(a,100);

for (int i=0;a[i]!=0;)//求字符串的长度

i++;

reverse(a,i-1);

cout<<"The reversed string is :";

for(int j=0;j<=i-1;j++)//输出反向后的字符串

cout<<\*(a+j);

cout<<endl;

}

//5-4-6编写一个函数，统计并返回一行文字中所有字母、数字、空格及其他字符的个数，同时编写相应的主函数，在主函数中能接收用户输入的一行文本，然后调用上面定义的函数，并在主函数中输出各种字符的个数。

#include <iostream.h>

void main()

{

char a[100]; //最多输入100个字符

cout<<"Input the string："<<endl;

cin.getline(a,100);

int num[4]={0,0,0,0};//分别存储数字、空格、字母和其它字符的个数

int i=0;

while(a[i]!=0)//求字符串的长度

{

if ((a[i]>=48)&&(a[i]<=57)) //数字

num[0]+=1;

else if (a[i]==32) //空格

num[1]+=1;

else if (((a[i]>=65)&&(a[i]<=90))||((a[i]>=97)&&(a[i]<=122)))//大小写字母

num[2]+=1;

else //其它字符

num[3]+=1;

i++;

}

cout<<"The Statistic result is :"<<endl;//输出统计结果

cout<<"number:"<<num[0]<<" blank:"<<num[1]<<" character:"<<num[2]<<" other:"<<num[3]<<endl;

}

//6-6-1

#include <iostream.h>

#define PI 3.14159

class Circle

{

public:

void input()

{

cout<<"请输入圆的半径：";

cin>>r;

}

double cal()

{

return PI\*r\*r;

}

void output()

{

cout<<"圆的半径为："<<r<<"，面积为："<<cal()<<"。"<<endl;

}

private:

int r;

};

void main()

{

Circle c;

c.input();

c.output();

}

//6-6-2

#include<iostream>

#include<cstring>

#include<iomanip>

using namespace std;

class Date //定义一个日期类

{

private:

int year;

int month;

int day;

public:

Date() //无参构造函数定义

{ }

Date(int y,int m,int d) //有参构造函数定义

{ year=y;

month=m;

day=d;

cout<<"Date类有参构造函数被执行!"<<endl;

}

Date(const Date &p) //拷贝构造函数定义

{ year=p.year;

month=p.month;

day=p.day;

cout<<"Date类拷贝构造函数被执行!"<<endl;

}

int Getyear()

{ return year;

}

int Getmonth()

{ return month;

}

int Getday()

{ return day;

}

};

class Student //定义一个学生类

{

private:

int number;

char name[20];

Date birth;

public:

Student();

Student(int n,char \*m,Date d);

Student(const Student &s);

~Student();

void setnumber(int n);

void setname(char \*s);

void setbirth(Date d);

void printstudent();

};

Student::Student()//无参构造函数定义

{ }

Student::Student(int n,char \*m,Date d):birth(d)//有参构造函数定义

{ number=n;

strcpy(name,m);

cout<<"Student类有参构造函数被执行!"<<endl;

}

Student::Student(const Student &s):birth(s.birth)//拷贝构造函数定义

{ number=s.number;

strcpy(name,s.name);

cout<<"Student类拷贝构造函数被执行!"<<endl;

}

Student::~Student()//析构函数定义

{ }

void Student::setnumber(int n)

{

number=n;

}

void Student::setname(char \*s)

{

strcpy(name,s);

}

void Student::setbirth(Date d)

{

birth=d;

}

void Student::printstudent()//输出全部学生的信息

{

cout<<"学生学号"<<setw(15)<<"学生姓名"<<setw(15)<<"出生日期"<<endl;

cout<<number<<setw(15)<<name<<setw(15)<<birth.Getyear()<<"-"<<birth.Getmonth()<<"-"<<birth.Getday()<<endl;

}

int main()

{ int setnumber; //用于设定学生人数

int q;

int xnumber; //定义学号

char xname[20]; //定义姓名

int xyear,xmouth,xday;//定义年,月,日

Student student[100];//定义一个Student对象数组,调用Student类中的无参构造函数,但是会先调用子对象birth的无参构造函数

cout<<"请输入学生人数(小于100): ";

cin>>setnumber;

cout<<endl;

for(q=0;q<setnumber;q++)

{ cout<<"学生学号: ";

cin>>xnumber;

cout<<"学生姓名: ";

cin>>xname;

cout<<"出生日期"<<endl;

cout<<"年: ";

cin>>xyear;

cout<<"月: ";

cin>>xmouth;

cout<<"日: ";

cin>>xday;

cout<<endl;

Date xdate(xyear,xmouth,xday); //调用Date类有参构造函数

Student temp(xnumber,xname,xdate); //调用Student类的有参构造函数,但会先执行Date类的拷贝构造函数

Student stu(temp); //调用Student类的拷贝构造函数,但先执行Date类的拷贝构造函数

student[q]=stu;

}

for(q=0;q<setnumber;q++)

{ student[q].printstudent();//输出学生信息

}

Student stu;

cout<<"学生学号: ";

cin>>xnumber;

cout<<"学生姓名: ";

cin>>xname;

cout<<"出生日期"<<endl;

cout<<"年: ";

cin>>xyear;

cout<<"月: ";

cin>>xmouth;

cout<<"日: ";

cin>>xday;

cout<<endl;

Date date1(xyear,xmouth,xday);

stu.setnumber(xnumber);

stu.setname(xname);

stu.setbirth(date1);//调用Date类的拷贝构造函数

stu.printstudent();

return 0;

}

//6-6-3设计一个点类Point，其中包含点的坐标x和y两个数据成员，并设计两个友元函数，分别计算两点间的水平距离和垂直距离。

#include<iostream.h>

#include<math.h>

class Point

{ private:

int x,y;

public:

Point(int a=0,int b=0) //有参构造函数

{ x=a;

y=b;

}

/\* void set(int a,int b) //赋值函数

{ x=a;

y=b;

}\*/

friend void distX(Point &p1,Point &p2) //友元函数，计算两点之间的水平距离

{ cout<<"两点间的水平距离是："<<fabs(p1.x-p2.x)<<endl;

}

friend void distY(Point &p1,Point &p2) //友元函数，计算两点之间的垂直距离

{ cout<<"两点间的垂直距离是："<<fabs(p1.y-p2.y)<<endl;

}

};

void main()

{ Point d1(3,4),d2(15,10);

/\* Point d1,d2;

int x,y;

cout<<"输入第一个点的水平坐标和垂直坐标：";

cin>>x>>y;

d1.set(x,y);

cout<<"输入第二个点的水平坐标和垂直坐标：";

cin>>x>>y;

d2.set(x,y);\*/

distX(d1,d2);

distY(d1,d2);

}

//7-6-1

#include <iostream>

#include <string>

using namespace std;

class Date //定义日期类

{

private:

int year;

int month;

int day;

public:

Date(int y,int m,int d) //有参构造函数

{

cout<<"调用Date构造函数"<<endl;

year=y;

month=m;

day=d;

}

void show() //显示函数

{

cout<<year<<"年"<<month<<"月"<<day<<"日";

}

};

class Time //定义时间类

{

private:

int hour;

int minute;

int second;

public:

Time(int h,int m,int s) //有参构造函数

{

cout<<"调用Time构造函数"<<endl;

hour=h;

minute=m;

second=s;

}

void show() //显示函数

{

cout<<hour<<"点"<<minute<<"分"<<second<<"秒"<<endl;

}

};

class Schedule:public Date,public Time //派生类

{

private:

int number;

public:

Schedule(int num,int a,int b,int c,int d,int e,int f):Date(a,b,c),Time(d,e,f) //派生类构造函数

{

cout<<"调用Schedule构造函数"<<endl;

number=num;

}

void show()

{

cout<<"序号"<<number<<":";

Date::show();

Time::show();

}

};

int main()

{

Schedule obj(1,2008,8,8,0,0,0);

obj.show();

return 0;

}

//7-6-2

#include<iostream.h>

#include<string.h>

class Person

{ protected:

char \*name;

int age;

public:

Person();

Person(char \*sname,int sage);

~Person();

void show();

};

Person::Person()

{ name=new char[10];

strcpy(name,"no name");

age=0;

cout<<"Person类的无参构造函数已经执行！"<<endl;

}

Person::Person(char \*sname,int sage)

{ name=new char[strlen(sname)+1];

strcpy(name,sname);

age=sage;

cout<<"Person类的有参构造函数已经执行！"<<endl;

}

Person::~Person()

{ delete[]name;

cout<<"Person类的析构函数已经执行！"<<endl;

}

void Person::show()

{ cout<<"姓名："<<name<<"\t"<<"年龄："<<age<<"\t";

}

class Student:virtual public Person

{ protected:

int no;

public:

Student();

Student(char \*sname,int sage,int sno);

~Student();

void show();

};

Student::Student()

{ no=0;

cout<<"Student类的无参构造函数已经执行！"<<endl;

}

Student::Student(char \*sname,int sage,int sno):Person(sname,sage)

{ no=sno;

cout<<"Student类的有参构造函数已经执行！"<<endl;

}

Student::~Student()

{ cout<<"Student类的析构函数已经执行！"<<endl;

}

void Student::show()

{ Person::show();

cout<<"学号："<<no<<endl;

}

class Teacher:virtual public Person

{ protected:

char \*title;

public:

Teacher();

Teacher(char \*sname,int sage,char \*stitle);

~Teacher();

void show();

};

Teacher::Teacher()

{ title=new char[10];

strcpy(title,"no title");

cout<<"Teacher类的无参构造函数已经执行！"<<endl;

}

Teacher::Teacher(char \*sname,int sage,char \*stitle):Person(sname,sage)

{ title=new char[strlen(stitle)+1];

strcpy(title,stitle);

cout<<"Teacher类的有参构造函数已经执行！"<<endl;

}

Teacher::~Teacher()

{ delete[]title;

cout<<"Teacher类的析构函数已经执行！"<<endl;

}

void Teacher::show()

{ Person::show();

cout<<"职称："<<title<<endl;

}

class Teacher\_Student:public Student,public Teacher

{

public:

Teacher\_Student(char \*pname=" ",int page=0,int pno=0,char \*ptitle=" ");

void show();

};

Teacher\_Student::Teacher\_Student(char \*pname,int page,int pno,char \*ptitle):Student(pname,page,pno),Teacher(pname,page,ptitle),Person(pname,page)

{ }

void Teacher\_Student::show()

{

Student::show();

cout<<"职称："<<title<<endl;

}

void main()

{ Person p1;

p1.show();

cout<<endl;

Person p2("zhao",30);

p2.show();

cout<<endl<<endl;

Student s1;

s1.show();

Student s2("qian",18,5001);

s2.show();

cout<<endl;

Teacher t1;

t1.show();

Teacher t2("sun",50,"professor");

t2.show();

cout<<endl;

Teacher\_Student ts1;

ts1.show();

Teacher\_Student ts2("li",35,8002,"lecturer");

ts2.show();

}

//7-6-3

#include <iostream.h>

class Shape //抽象类Shape，表示形状

{

protected:

double d;

public:

void setShape(double i)

{ d=i;

}

virtual void area()=0;

virtual void volume()=0;

};

class Cylinder: public Shape //圆柱体类

{

private:

double height;

public:

void setCylinder(double i,double j)

{

setShape(i);

height=j;

}

void area()

{ cout<<"圆柱体的表面积："<<2\*3.14\*d\*height+2\*3.14\*d\*d<<endl;

}

void volume()

{ cout<<"圆柱体的体积："<<3.14\*d\*d\*height<<endl;

}

};

class Sphere:public Shape //球体类

{

public :

void setSphere(double i)

{

setShape(i);

}

void area()

{

cout<<"球体的表面积："<<4\*3.14\*d\*d<<endl;

}

void volume()

{

cout<<"球体的体积："<<4/3\*3.14\*d\*d\*d<<endl;

}

};

class Cube:public Shape //正方体类

{

public:

void setCube(double i)

{

setShape(i);

}

void area()

{

cout<<"正方体的表面积："<<d\*d\*6<<endl;

}

void volume()

{

cout<<"正方体的体积："<<d\*d\*d<<endl;

}

};

void main()

{ Shape \*p0;

double p ;double q;double r;

Cylinder cy1;

Sphere sp1;

Cube cu1;

p0=&cy1;

cout<<"分别输入圆柱体的半径和高："<<endl;

cin>>p>>q;

cy1.setCylinder(p,q);

p0->area();

p0->volume();

p0=&sp1;

cout<<endl<<"输入球体的半径："<<endl;

cin>>r;

sp1.setSphere(r);

p0->area();

p0->volume();

p0=&cu1;

cout<<endl<<"输入正方体的边长："<<endl;

cin>>p;

cu1.setCube(p);

p0->area();

p0->volume();

}

//8-4-1

#include <iostream.h>

class Complex

{

private:

double r; //表示复数的实部

double i; //表示复数的虚部

public:

Complex(double x=0.0,double y=0.0) //定义构造函数

{ r=x;

i=y;

}

Complex operator+(Complex &c);

Complex operator-(Complex &c);

Complex operator\*(Complex &c);

Complex operator/(Complex &c);

void print();

};

Complex Complex::operator +(Complex &c)

{

Complex temp;

temp.r=r+c.r;

temp.i=i+c.i;

return temp;

}

Complex Complex::operator -(Complex &c)

{

Complex temp;

temp.r=r-c.r;

temp.i=i-c.i;

return temp;

}

Complex Complex::operator \*(Complex &c)

{

Complex temp;

temp.r=r\*c.r-i\*c.i;

temp.i=i\*c.r+r\*c.i;

return temp;

}

Complex Complex::operator /(Complex &c)

{

Complex temp;

temp.r=(r\*c.r+i\*c.i)/(c.r\*c.r+c.i\*c.i);

temp.i=(i\*c.r-r\*c.i)/(c.r\*c.r+c.i\*c.i);

return temp;

}

void Complex::print()

{ cout<<'('<<r<<','<<i<<')'<<endl;

}

void main()

{

Complex c1(1.0,2.0),c2(3.0,4.0),c3=c1+c2,c4=c1-c2,c5=c1\*c2,c6=c1/c2;

c1.print();

c2.print();

c3.print();

c4.print();

c5.print();

c6.print();

}

//8-4-2

#include <iostream>

using namespace std;

class Time

{

public:

Time()

{

minute=0;

sec=0;

}

Time(int m,int s):minute(m),sec(s)

{ }

Time operator++(); //声明前置白增运算符“++”重载函数

Time operator++(int); //声明后置自增运算符“++”重载函数

void display()

{ cout<<minute<<":"<<sec<<endl;

}

private:

int minute;

int sec;

};

Time Time::operator++() //定义后置自增运算符“++”重载函数

{ if(++sec>=60)

{

sec-=60;

++minute;

}

return \*this;

} //返回的是自加前的对象

Time Time::operator++(int)

{

Time temp(\*this);

sec++;

if(sec>=60)

{

sec-=60;

++minute;

}

return temp;

}

int main()

{

Time time1(34,59),time2;

cout<<" time1 : ";

time1.display();

++time1;

cout<<"++time1: ";

time1.display();

time2=time1++; //将自加前的对象的值赋给time2

cout<<"time1++: ";

time1.display();

cout<<" time2 : "; //输出time2对象的值

time2.display();

return 0;

}

/\*

\*

\* 文件名称：Xiti9\_4\_1.cpp

\* 摘 要：在例9-4的基础上，继续实现单向链表类模版中能够实现节点的插入、删除、排序等的成员函数

\* 完成日期：2009.1

\*

\*/

#include<iostream.h>

template<class T>

class List

{

private:

T data;

List \*next;

static List \*curnode; //当前节点指针，静态数据成员，为List类的所有对象所共有

static List \*head; //头指针，静态数据成员，为List类的所有对象所共有

public:

List():next(NULL) //无参构造函数，用于构造头节点

{

head=curnode=this;

}

List(T newdata):data(newdata),next(NULL) //有参构造函数，用于构造新节点

{

}

void create(T newnode); //创建链表函数

void display(); //输出函数

//所添加的节点的插入、删除、排序等的成员函数//////////

int insert(int insert\_number,T newdata); //插入函数

int del(int del\_number); //删除函数

void sequence(); //排序函数

//////////////////////////////////////////////////////

};

template<class T>

List<T>\*List<T>::curnode;

template<class T>

List<T>\*List<T>::head;

template<class T>

void List<T>::create(T newdata) //创建链表函数

{

curnode->next=new List(newdata);

curnode=curnode->next;

}

template<class T>

void List<T>::display() //输出函数

{

curnode=head->next;

while(curnode!=NULL)

{

cout<<curnode->data<<endl;

curnode=curnode->next;

}

}

////////////////////////////////////////////

//插入函数 //

//实现在链表的某一位置插入节点的功能 //

//其中： //

//insert\_number：要插入的位置 //

//insert\_data：要插入的新节点中存储的数据 //

////////////////////////////////////////////

template<class T>

int List<T>::insert(int insert\_number,T insert\_data)

{

List \*temp=new List(insert\_data);

curnode=head;

for(int i=1;i<insert\_number;i++) //找到插入位置

{

curnode=curnode->next;

}

if(curnode==NULL) //如果没有找到插入位置，则插入失败，返回-1

{

return -1;

}

else

{

temp->next=curnode->next; //通过移动指针，实现插入操作，插入成功，返回0

curnode->next=temp;

return 0;

}

}

////////////////////////////////////////////

//删除函数 //

//实现在链表的某一位置删除节点的功能 //

//其中： //

//del\_number：要删除节点的位置 //

////////////////////////////////////////////

template<class T>

int List<T>::del(int del\_number)

{

curnode=head;

if(del\_number==1) //第一种情况：要删除的是头节点

{

head->next=(curnode->next)->next; //通过移动指针，实现删除操作，删除成功，返回0

return 0;

}

else //第二种情况：要删除的是一般节点

{

for(int i=1;i<del\_number;i++) //找到要删除的节点的位置

{

curnode=curnode->next;

}

if(curnode==NULL)

{

return -1; //如果没有找到删除位置，则删除失败，返回-1

}

else

{

curnode->next=(curnode->next)->next; //通过移动指针，实现删除操作，删除成功，返回0

return 0;

}

}

}

////////////////////////////////////////////

//排序函数 //

//实现对链表节点的排序功能（从小到大排序）//

////////////////////////////////////////////

template<class T>

void List<T>::sequence()

{

List \*temp\_node=new List(1); //新建一个临时节点（由于只做临时应用故随意给其赋了一个值）

List \*tail; //定义指针表示链表的结尾

tail=NULL;

temp\_node->next=head; //通过指针的变换，将新节点设为最开始节点

head=temp\_node;

while(tail!=head->next) //循环进行排序

{

temp\_node=head;

curnode=temp\_node->next;

while(tail!=curnode->next)

{

if(temp\_node->next->data > curnode->next->data) //如果后一个节点储存的值小于前一个节点，则移动指针对链表进行重新排列

{

temp\_node->next=curnode->next;

curnode->next=curnode->next->next;

temp\_node->next->next=curnode;

}

temp\_node=temp\_node->next;

curnode=temp\_node->next;

}

tail=curnode;

}

temp\_node=head;

head=head->next; //最后通过指针移动将临时的最开始节点删除

}

void main()

{

cout<<"建立两个链表并输出"<<endl;

cout<<"第一个链表："<<endl;

List<int> list1;

list1.create(2);

list1.create(0);

list1.create(0);

list1.create(6);

list1.display();

cout<<"第二个链表："<<endl;

List<char> list2;

list2.create('O');

list2.create('K');

list2.create('!');

list2.display();

cout<<"向第二个链表的第2个位置添加字符“\*”"<<endl;

char a='\*';

list2.insert(2,a);

list2.display();

cout<<"删除第二个链表的第3个位置的节点元素"<<endl;

list2.del(3);

list2.display();

cout<<"新建立一个链表为："<<endl;

List<int> list3;

list3.create(2);

list3.create(3);

list3.create(5);

list3.create(7);

list3.create(10);

list3.create(9);

list3.create(1);

list3.create(4);

list3.create(6);

list3.create(8);

list3.display();

cout<<"对其进行排序并输出："<<endl;

list3.sequence();

list3.display();

}#include<iostream.h>

template<class T> //定义二分查找函数模板

int find(T data[],int length,T object) //数组data 为有序数组，length 为数组长度，object 为待查找元素

{

int max,min,mid;

min=0;

max=n-1;

while(min<=max)

{

mid=(min+max)/2;

if(object==data[mid])

return mid;

else if(object>data[mid])

min=mid+1;

else

max=mid-1;

}

return -1;

}

int const n=5;

void main()

{

int i; //i为下标

int f1,a[n],x1; //int型测试所需数据

int f2; //char型测试所需数据

char b[n];

char x2;

int f3; //double型测试所需数据

double c[n];

double x3;

cout<<"请输入"<<n<<"个依次增大的整数"<<endl; //用int型进行测试

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

cin>>a[i];

cout<<"请输入一个要查找的整数:"<<endl;

cin>>x1;

f1=find(a,n,x1); //函数模板实例化

if(f1==-1)

cout<<"查找"<<x1<<"失败"<<endl;

else

cout<<"查找"<<x1<<"成功"<<"下标为"<<f1<<endl;

cout<<"请输入"<<n<<"个依次增大的字符"<<endl; //用char型进行测试

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

cin>>b[i];

cout<<"请输入一个要查找的字符:"<<endl;

cin>>x2;

f2=find(b,n,x2); //函数模板实例化

if(f2==-1)

cout<<"查找"<<x2<<"失败"<<endl;

else

cout<<"查找"<<x2<<"成功"<<"下标为"<<f2<<endl;

cout<<"请输入"<<n<<"个依次增大的双精度浮点型数据"<<endl;//用double型进行测试

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

cin>>c[i];

cout<<"请输入一个要查找的双精度浮点型数据:"<<endl;

cin>>x3;

f3=find(c,n,x3); //函数模板实例化

if(f3==-1)

cout<<"查找"<<x3<<"失败"<<endl;

else

cout<<"查找"<<x3<<"成功"<<"下标为"<<f3<<endl;

}

/\*

\*

\* 文件名称：Xiti10\_4\_1.cpp

\* 摘 要： 编写程序，打开指定的一个文本文件，并能在其中每一行的前面加上行号和一个空格符。

\* 完成日期：2009.1

\*

\*/

#include "iostream.h"

#include "fstream.h"

void main()

{

int i=1;

char a[30];

fstream f1,f2;

f1.open("From\_Xiti\_10\_4\_1.txt",ios::in); //以读方式打开文件From\_Xiti\_10\_4\_1.txt

if(!f1) //判断是否成功打开

{

cout<<"can't open this file.";

return;

}

f2.open("To\_Xiti\_10\_4\_1.txt",ios::out); //以写方式打开文件To\_Xiti\_10\_4\_1.txt

if(!f2) //判断是否成功打开

{

cout<<"can't open this file.";

return;

}

while(!f1.eof())

{

f1.getline(a,sizeof(a)); //从From\_Xiti\_10\_4\_1.txt文件中读出一行数据到数组a[30]

f2<<i<<" "<<a<<'\n'; //将其前面加上行号和一个空格符，之后再写入f2

i++;

}

cout<<"程序执行成功!\nTo\_Xiti\_10\_4\_1.txt被保存在项目文件夹下"<<endl;

f1.close(); //关闭文件

f2.close();

}/\*

\*

\* 文件名称：Xiti10\_4\_2.cpp

\* 摘 要：

\* 编写程序，统计一篇英文文章中英文单词的个数以及文章的总行数

\* 其中要假定文章要用正常格式输入。即单词与单词之间要有空格。并且每一行的结尾要有回车。

\* 完成日期：2009.1

\*

\*/

#include "iostream.h"

#include "fstream.h"

void main()

{

int word\_number=1; //单词个数

int essay\_line=1; //行数

int sign=-1; //标志位，判断txt是否一个单词也没有。

char a;

fstream f;

f.open("Xiti\_10\_4\_2.txt",ios::in); //打开文件

if(!f)

{

cout<<"can't open this file.";

return;

}

while(!f.eof()) //循环得到文件中的每个字符，并判断空格数和回车数，由此得出单词数和行数

{

f.get(a);

if(a==' ')

{

word\_number++;

sign=1;

}

else if(a=='\n')

{

essay\_line++;

word\_number++;

sign=1;

}

}

if(sign==-1) //如果标志位是-1，则说明文件中单词数、行数均为0

{

essay\_line=0;

word\_number=0;

}

cout<<"程序执行成功!\n"<<"这篇英文文章中单词的个数是:"<<word\_number<<'\n'<<"文章的总行数为:"<<essay\_line<<'\n';

f.close(); //关闭文件

}/\*

\*

\* 文件名称：Xiti10\_4\_3.cpp

\* 摘 要：编写程序，完成两个文件的连接操作

\* 完成日期：2009.1

\*

\*/

#include "iostream.h"

#include "fstream.h"

#define N 30

void main()

{

int i=1;

char a[N];

fstream f1,f2;

f1.open("From\_Xiti\_10\_4\_3.txt",ios::in|ios::out); //以读写方式打开文件

if(!f1)

{

cout<<"can't open this file.";

return;

}

f2.open("To\_Xiti\_10\_4\_3.txt",ios::in|ios::out|ios::app); //以读写方式打开文件，并使新的内容添加到文件的末尾

if(!f2)

{

cout<<"can't open this file.";

return;

}

while(!f1.eof()) //得到From\_Xiti\_10\_4\_3.txt中的数据并将添加到To\_Xiti\_10\_4\_3.txt的末尾，以实现文件的连接

{

f1.getline(a,sizeof(a));

f2<<a<<'\n';

i++;

}

cout<<"两个文件连接完成!\n结果被保存在To\_Xiti\_10\_4\_3.txt中"<<'\n';

f1.close(); //关闭文件

f2.close();

}/\*

\*

\* 文件名称：Xiti11\_4\_1.cpp

\* 摘 要：编写程序，利用string类提供的成员函数来完成字符串的查找与替换

\* 完成日期：2009.1

\*

\*/

#include <iostream>

#include <string>

#include <vector>

#include <iterator>

using namespace std;

void main()

{

cout<<"在字符串a中查找出字符串b，然后以字符串c替换"<<endl;

string aa("");

string bb("");

string cc("");

cout<<"输入字符串a"<<endl;

cin>>aa;

cout<<"输入字符串b"<<endl;

cin>>bb;

cout<<"输入字符串c"<<endl;

cin>>cc;

int npos=aa.find(bb); //查找字符串b在a的位置

int bb\_length=bb.length(); //得到b的长度

cout<<"字符串b的长度为："<<bb\_length<<endl;

aa.replace(npos,npos+bb\_length-1,cc); //在字符串a中查找出字符串b，然后以字符串c替换

cout<<"替换之后的字符串a为：\n"<<aa<<endl;

}/\*

\*

\* 文件名称：Xiti 11\_4\_2.cpp

\* 摘 要：编写程序，利用string类完成一个字符串的逆序输出

\* 完成日期：2009.1

\*

\*/

#include "iostream"

#include <string>

#include <vector>

#include <iterator>

using namespace std;

void main()

{

cout<<"字符串的逆序输出"<<endl;

string dd(""); //dd用来保存输入的字符串

cout<<"输入字符串:"<<endl;

cin>>dd;

cout<<"逆序输出字符串：";

for(int i=dd.length()-1;i>=0;i--) //实现逆序输出字符串

{

cout<<dd.substr(i,1);

}

cout<<endl;

}/\*

\*

\* 文件名称：Xiti 11\_4\_3.cpp

\* 摘 要：编写程序，利用vector容器输入若干个string类数据元素，将其排序后输出

\* 完成日期：2009.1

\*

\*/

#include "iostream"

#include <string>

#include <vector>

#include <iterator>

using namespace std;

void main()

{

cout<<"输入若干个string类数据元素，然后排序后输出"<<endl;

vector <string> str\_vector;

string stradd("");

//输入//////////////////////////////

cout<<"向容器中添加字符串，以“0000”结束添加"<<endl;

cin>>stradd;

while (stradd.compare("0000")!=0)

{

str\_vector.push\_back(stradd); //将字符串加入容器

cin>>stradd;

}

//排序////////////////////////////

int size=str\_vector.size(); //得到容器长度

int i,j;

for(i=0;i<size;i++) //冒泡排序

{

for(j=0;j<size-1-i;j++)

{

if(str\_vector[j]>str\_vector[j+1])

{

str\_vector[j].swap(str\_vector[j+1]);

}

}

}

//输出//////////////////////////

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

{

cout<<str\_vector[i]<<endl;

}

}/\*

\*

\* 文件名称：Xiti11\_4\_4.cpp

\* 摘 要：编写程序，利用输入、输出迭代器，将一个文件中的内容添加到另一个文件的尾部

\* 完成日期：2009.1

\*

\*/

#include<iostream>

#include<fstream>

#include<string>

#include<vector>

#include<iterator>

using namespace std;

void main()

{

typedef vector<string> string\_vector;

typedef istream\_iterator<string> input;

typedef ostream\_iterator<string> output;

typedef back\_insert\_iterator<string\_vector> ins;

string\_vector vec; //定义string类型容器对象vec

ifstream from\_file("file1.txt");

if(from\_file.fail())

{

cout<<"文件打开失败"<<endl;

exit(0);

}

copy(input(from\_file),input(),ins(vec)); //从file1.txt文件中提取元素插入到容器vec中

cout<<"file1.txt文件的内容是："<<endl;

copy(vec.begin(),vec.end(),output(cout," ")); //将容器vec中的元素插入到屏幕输出流

cout<<endl;

ofstream to\_file("file2.txt",ios::app); //使用ios::app方式实现一个文件中的内容添加到另一个文件的尾部

copy(vec.begin(),vec.end(),output(to\_file," "));//将容器vec中的元素插入到file2.txt文件的尾部，分隔符为空格

cout<<"文件的内容已成功插入到file2.txt文件的尾部!!!"<<endl;

}

//编写一个程序，采用异常处理的方法，对数组的下标错误进行显示。

#include <iostream.h>

void main()

{ int a[20],m;

try //在try语句块中包含要检查的函数

{

for (int i=0;i<20;i++)

a[i]=i;

cout<<"现有一维数组a[20]，请输入要查询的下标:";

cin>>m;

if (m<0)

throw 3;

else if (m>=20)

throw 'a';

else

cout<<"a["<<m<<"]="<<a[m]<<endl;

}

catch(int) //用catch捕捉int类型的异常并进行处理

{

cout << "数组下标不能小于零!";

}

catch(char)

{

cout << "数组下标不能大于或等于20!";

}

}

//编写一个程序，采用异常处理的方法，在指定文件不存在时显示出错信息，文件存在时显示文件的基本信息。

#include<iostream.h>

#include<fstream.h>

#include<stdlib.h>

#include <stdio.h>

void main()

{

FILE \*fp; //定义文件指针fp

try

{

//以读的方式打开文件exception.txt，并判断文件是否打开成功

if ((fp=fopen("c:\\exception.txt","r"))==NULL)

throw 1;//打开不成功

else

throw fp;//打开成功

fclose(fp);//关闭原文件

}

catch (int)

{

cout<<"Can't find file!!"<<endl;

}

catch (FILE \* Exceptionfile)//用catch捕捉FILE类型的异常并获取抛出的值

{

char ch;

ch=fgetc(Exceptionfile);//读取文件的第一个字符

while(ch!=EOF)

{ cout<<ch;

ch=fgetc(Exceptionfile); //读取文件的下一个字符

}

}

}