12.3-函数的应用实例(一)

代码示例1：编写程序，利用结构体和函数完成复数的加减法运算

#include<iostream>

using namespace std;

struct complex

{

double a;

double b;

};

void in(complex &c)

{

cout<<"请输入实部和虚部:";

cin>>c.a>>c.b;

return;

}

/\*

这是一个没有返回值的函数，类型void为“空型”或“无值型”

\*/

complex add(complex c1,complex c2){

complex tmp;

tmp.a=c1.a+c2.a;

tmp.b=c1.b+c2.b;

return tmp;

}

/\*

当函数要返回的值不止一个、情况比较复杂时，它可以是结构或类的形式，也可以数组或对象指针类型来实现返回。

\*/

complex sub(complex c1,complex c2){

complex tmp;

tmp.a=c1.a-c2.a;

tmp.b=c1.b-c2.b;

return tmp;

}

void out(complex c)

{

cout<<"("<<c.a<<"+"<<c.b<<"i)";

return;

}

void out\_add(complex c1,complex c2)

{

cout<<"("<<c1.a<<"+"<<c1.b<<"i)+"<<"("<<c2.a<<"+"<<c2.b<<"i)=";

out(add(c1,c2));

return;

}

void out\_sub(complex c1,complex c2)

{

cout<<"("<<c1.a<<"+"<<c1.b<<"i)-"<<"("<<c2.a<<"+"<<c2.b<<"i)=";

out(sub(c1,c2));

return;

}

int main()

{

complex cA[3];

for(int i=1;i<=2;i++)

{ in(cA[i]); }

out\_add(cA[1],cA[2]);

cout<<endl;

out\_sub(cA[1],cA[2]);

return 0;

}

代码示例2:

编写程序，求排列数和组合数的值

A(n,m)=n!/(n-m)!

C(n,m)=A(n,m)/m!=n!/( (n-m)!\*m! )

第一行输入两个整数n,m，求排列数和组合数的值；若输入的m或n为非正整数或n<m，则输出“Wrong!Please input again:”，然后重新输入n,m,直到输入合法为止

输出：两行，分别为排列数和组合数的计算结果

样例输入：

2 3

0 1

5 2

样例输出：

Wrong!Please input again:

Wrong!Please input again:

20

10

#include<iostream>

using namespace std;

long long factorial(int x)

{

//求x!

long long res=1;

while(x>1)

{

res\*=x;

x--;

}

return res;

}

double permutation(int n,int m)

{

//求排列数

return factorial(n)/factorial(n-m);

}

double combination(int n,int m)

{

//求组合数

return factorial(n)/factorial(n-m)/factorial(m);

}

int main()

{

int m,n;

bool judge=true;

while(judge)

{

cin>>n>>m;

if(n<=0 || m<=0 || m>n)

{

cout<<"Wrong!Please input again:"<<endl;

}

else

{

judge=false;

}

}

cout<<permutation(n,m)<<endl;

cout<<combination(n,m)<<endl;

return 0;

}

代码示例3：

著名的哥德巴赫猜想：任何大于6的偶数都能分解成两个素数之和。当然，很多偶数能够分解成多对素数之和，例如：20=3+17=7+13

输入：

第一行：一个整数N，表示有N组测试数据

接下来的N行，每行输入一个正偶数（小于10000）

输出：

N行， 对于每个输入的偶数，求出其能够分解成多少对不同的素数(A+B和B+A算一对)

#include<iostream>

#include<math.h>

using namespace std;

int judge\_prime(int m)

{

//判断一个整数是否为素数,返回值为0不是素数，1是素数

int i,isprime=1;

for(i=2;i<=sqrt(double(m));i++)

{

if(m%i==0)

{

isprime=0;

break;

}

}

return isprime;

}

int main()

{

int N;

cin>>N;

int i,j;

int m,count;

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

{

cin>>m;

count=0; //记录素数对的个数

for(j=2;j<=m/2;j++)

{

if( judge\_prime(j)==1 && judge\_prime(m-j)==1)

{

//j和m-j都是素数，则找到一对素数对

count++;

}

}

cout<<count<<endl;

}

return 0;

}