函数

MOOC第6周

教材7.1~7.3节

哈尔滨工业大学

苏小红

sxh@hit.edu.cn

投票

最多可选3项

设置

你为什么留下来？

A 喜欢老师的讲课风格

B 喜欢老师的讲课内容，想多学点

C 只因同学推荐

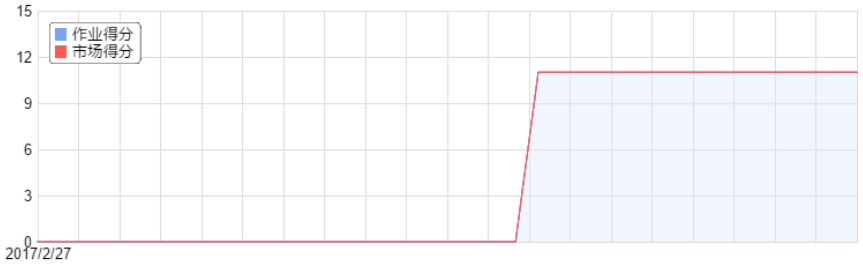
D 只是慕名而来

E 不喜欢也懒得再换了

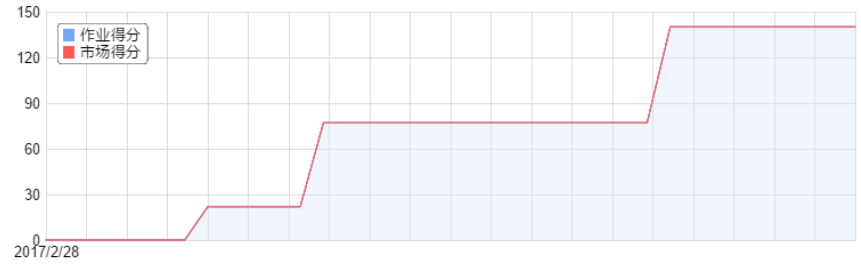
F 以上都不是

提交

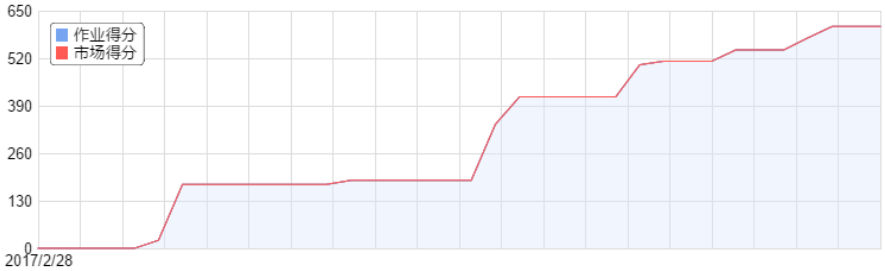
得分情况



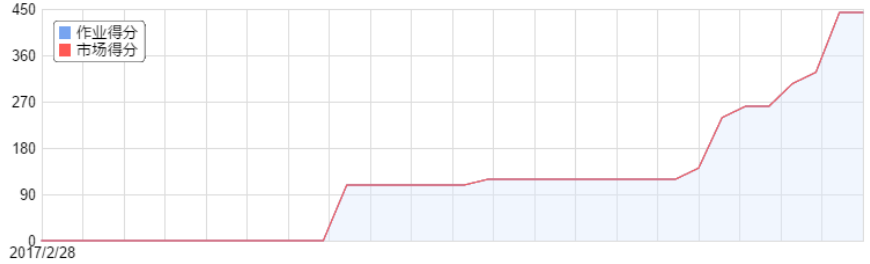
得分情况



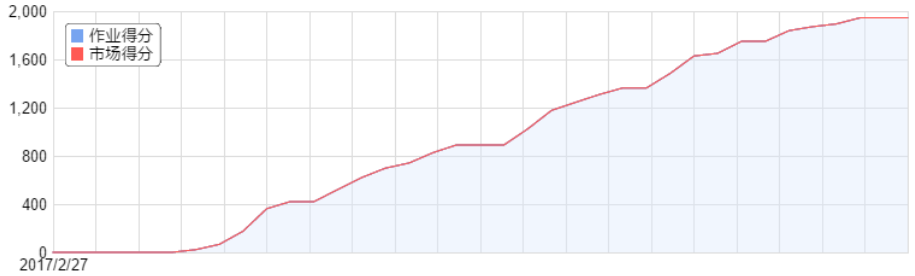
得分情况



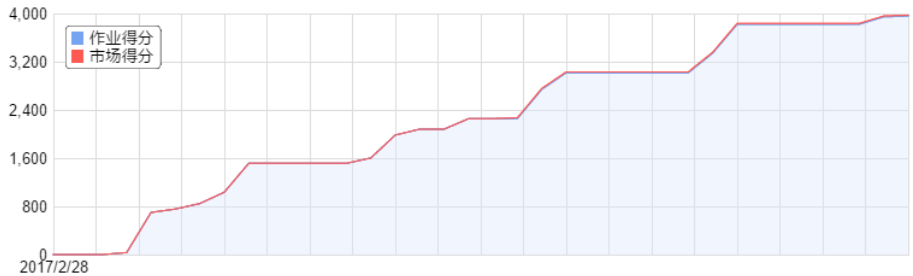
得分情况



得分情况



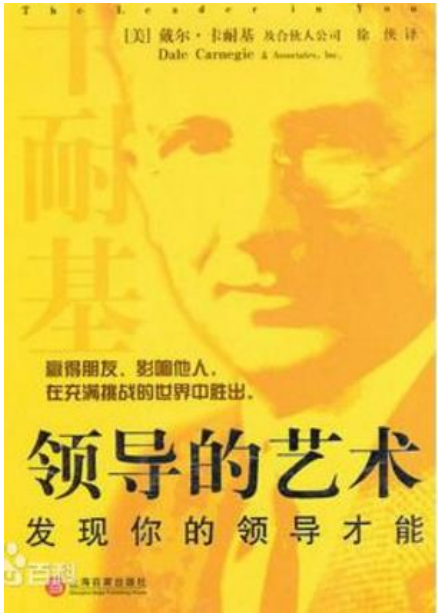
得分情况



一个函数中究竟适合放多少行代码？

- 50行左右（一页显示屏，一张打印纸）
  - 1986年IBM的研究结果：多数有错误的函数大于500行
  - 1991年对148,000行代码的研究表明：函数小于143行更易维护
- 如果把所有代码都写到`main()`函数里……？

大话三国

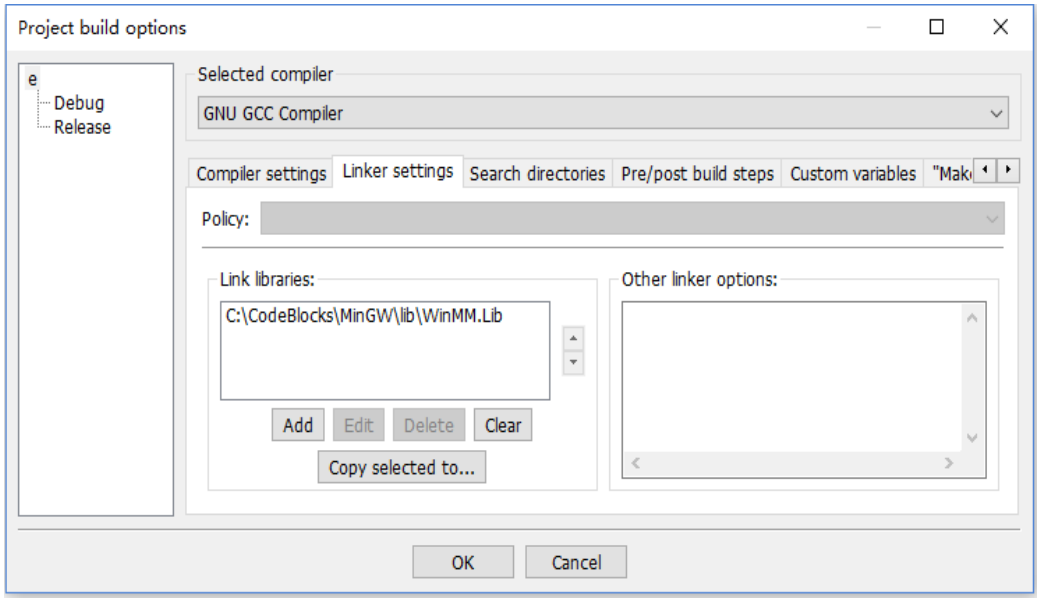


- 分工+协作
  - 分而治之（Divide and Conquer）
- 模块化编程（Modular Programming）
  - 某些功能调用标准库函数或第三方库函数完成

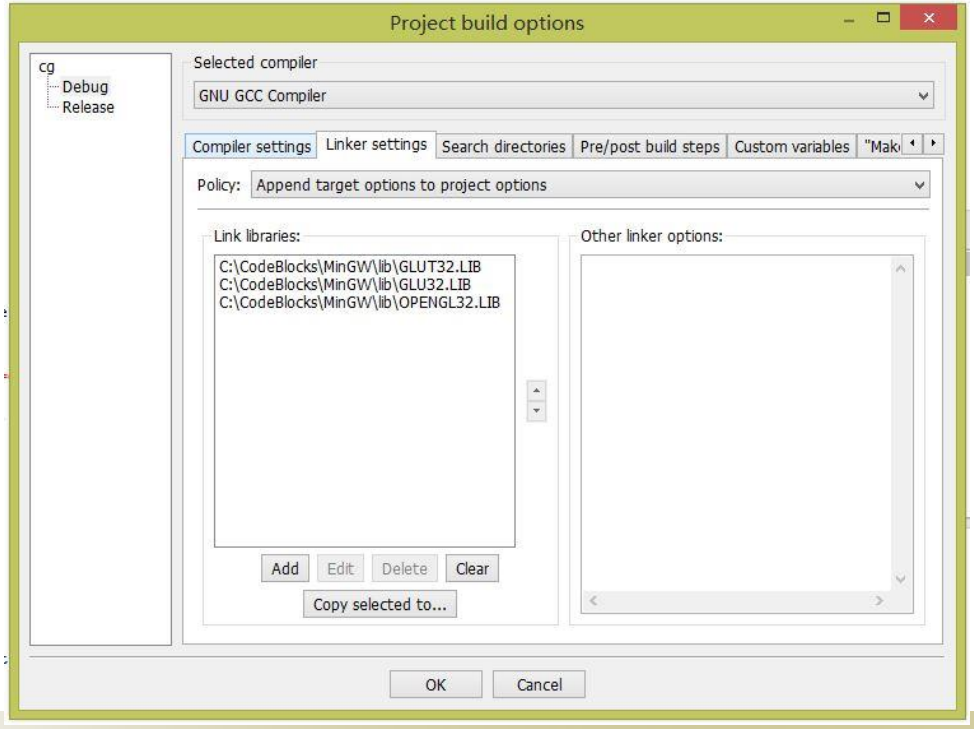
标准库函数

* **ANSI/ISO C定义的标准库函数**
  - 使用时，必须包含定义该函数的头文件
* **第三方库函数**
  - 由其他厂商自行开发的C语言函数库（例如，OpenGL，EasyX）

glutWireTeapot(0.5);



PlaySound()



标准库函数不够用咋办？

- **自定义函数**
  - 用户自己定义的函数
    - 包装后也可成为函数库，供别人使用
- 例如，编写一个函数GetMax()，求两个整数的最大值

函数设计的基本原则

1. 函数规模要适当
2. 函数功能要独立
3. 函数接口定义要清楚

如何调用这个函数？

- 通过函数名调用被调函数
  - 函数定义时的参数，形式参数（Parameter），简称形参
  - 函数调用时的参数，实际参数（Argument），简称实参

```
#include <stdio.h>
int main()
{
    int  a, b, max;
    scanf("%d, %d", &a, &b);
    max = GetMax(a, b);
    printf("max = %d\n", max);
    return 0;
}
```

```
int GetMax(int x,int y)
{
  return x > y ? x : y;
}
```

```
printf("max = %d\n", GetMax(a, b));
```

函数调用的过程

1.保存函数的返回地址，并为函数内的局部变量（包括形参）分配内存

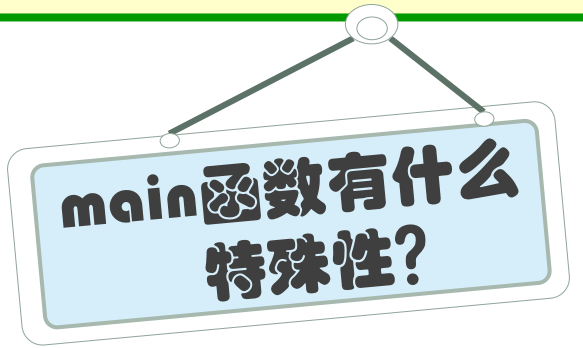
2.把实参值复制一份给形参，**单向传值**（实参→形参）

3.执行函数内语句，执行到return语句或}时，从函数退出，返回到函数调用处

4.函数值返回给主调函数，同时收回给函数内局部变量（包括形参）分配的内存

```
#include <stdio.h>
int main()
{
    int  a, b, max;
    scanf("%d, %d", &a, &b);
    max = GetMax(a, b);
    printf("max = %d\n", max);
    return 0;
}
```

```
int GetMax(int x, int y)
{
    return x > y ? x : y;
}
```



调用一个函数还需什么条件？

```
int GetMax(int x,int y)
{
  return x > y ? x : y;
}
```

```
#include <stdio.h>
int main()
{
    int  a, b, max;
    scanf("%d, %d", &a, &b);
    max = GetMax(a, b);
    printf("max = %d\n", max);
    return 0;
}
```

函数原型的好处是让编译器做参数类型匹配检查

- 隐含的函数声明，默认返回int型

```
int GetMax(int x,int y);
```

```
#include <stdio.h>
int main()
{
    int  a, b, max;
    scanf("%d, %d", &a, &b);
    max = GetMax(a, b);
    printf("max = %d\n", max);
    return 0;
}
```

没有身体不分配内存

```
int GetMax(int x,int y)
{
  return x > y ? x : y;
}
```

单选题

设置

下列说法错误的是：

A 函数内定义的变量称为局部变量，只能在函数内部访问

B 实参和形参同名时占用同一内存

C 函数原型没有函数体，因此编译器不为其分配内存

D 函数原型的好处是让编译器做参数类型匹配检查

E C程序都是从main函数开始执行，并在main函数中结束，除main函数外，其他函数的地位都是平等的

提交

素数探求——判断素数

- 如何判断一个整数x是素数（Prime Number）？
  - 不能被1和x以外的其他数整除的正整数
- 求解算法
  - 用2~x-1之间的整数去试商，看x能否其被整除
  - 用2~x/2之间的整数去试商，看x能否其被整除
  - 用2~sqrt(x)之间的整数去试商，看x能否被其整除
    - x = a * b
    - 若a>=sqrt(x)，则b<=sqrt(x)

素数探求——判断素数

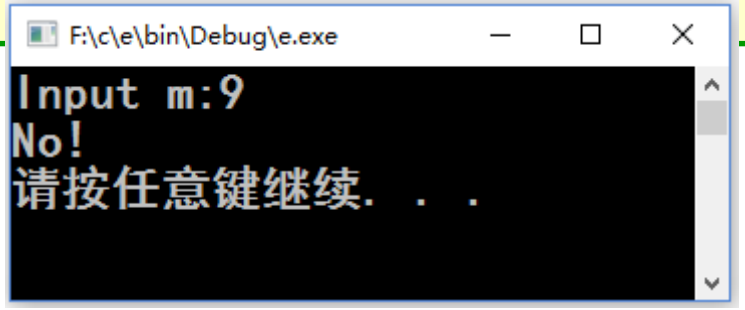
```c
#include <stdio.h>
#include <math.h>
int IsPrime(int x);
int main()
{
    int m;
    printf("Input m:");
    scanf("%d", &m);
    if (IsPrime(m))
    {
        printf("Yes!\n");
    }
    else
    {
        printf("No!\n");
    }
    return 0;
}
```

```c
int IsPrime(int x)
{
  int i, squareRoot;
  if (x <= 1) return 0;
  squareRoot = (int)sqrt(x);
  for (i=2; i<=squareRoot; i++)
  {
     if (x%i == 0)  return 0;
     else return 1;
  }
}
```

素数探求——判断素数

```c
#include <stdio.h>
#include <math.h>
int IsPrime(int x);
int main()
{
    int m;
    printf("Input m:");
    scanf("%d", &m);
    if (IsPrime(m))
    {
        printf("Yes!\n");
    }
    else
    {
        printf("No!\n");
    }
    return 0;
}
```

```c
int IsPrime(int x)
{
  int i, squareRoot;
  if (x <= 1) return 0;
  squareRoot = (int)sqrt(x);
  for (i=2; i<=squareRoot; i++)
  {
     if (x%i == 0)  return 0;
  }
  return 1;
}
```



判断完数

- 如何判断一个整数x是完数（ Perfect Number ）？
  - 它所有的真因子（即除自身以外的约数）的和，恰好等于它本身，即x的所有小于x的不同因子（包括1）加和恰好等于x本身
    - 例如： 6 = 1 + 2 + 3
  - 注意：1没有真因子，所以1不是完全数。
- 求解算法
  - i从1试到x/2，看i是否是x的真因子
  - 若x能被i整除，则累加到sum
  - 累加结束判断x是否等于sum，返回判断结果

判断完数

```
#include <stdio.h>
int IsPerfect(int x);
int main()
{
    int m;
    printf("Input m:");
    scanf("%d", &m);
    if (IsPerfect(m))
    {
      printf("Yes!\n");
    }
    else
    {
        printf("No!\n");
    }
    return 0;
}
```

```
//是完数返回1，否则返回0
int IsPerfect(int x)
{
    int i, sum = 0;
    for (i=1; _________; i++)
    {
       if (x%i == 0)
       {
           _____________;
       }
    }
    return _________;
}
```

- 求解算法
  - i从1试到x/2，看i是否是x的真因子
  - 若x能被i整除，则累加到sum
  - 累加结束判断x是否等于sum，返回判断结果

判断完数

```c
#include <stdio.h>
int IsPerfect(int x);
int main()
{
    int m;
    printf("Input m:");
    scanf("%d", &m);
    if (IsPerfect(m))
    {
      printf("Yes!\n");
    }
    else
    {
        printf("No!\n");
    }
    return 0;
}
```

```c
//是完数返回1，否则返回0
int IsPerfect(int x)
{
    int i, sum = 0;
    for (i=1; i<=x/2; i++)
    {
       if (x%i == 0)
       {
          sum = sum + i;
       }
    }
    return sum==x ? 1 : 0;
}
```

- 求解算法
  - i从1试到x/2，看i是否是x的真因子
  - 若x能被i整除，则累加到sum
  - 累加结束判断x是否等于sum，返回判断结果

挑战速度：输出n（<1000000）以内的所有完数？

素数探求——验证哥德巴赫猜想

- 任何一个大于或等于6的偶数总能表示为两个素数之和。
  - 例如，8=3+5，12=5+7等
  - 请编程从键盘输入[6,2000000000]以内的任意偶数n，验证哥德巴赫猜想
  - 如果n符合“哥德巴赫猜想”，则输出将n分解为两个素数之和的等式，否则输出“n不符合哥德巴赫猜想！”的提示信息。
  - 将n分解为两个奇数之和即n=a+b，然后测试a和b是否均为素数
- 求解算法
  - a从3开始试到n/2，a+=2（测试所有奇数）
  - 若IsPrime(a) && IsPrime(n-a)，则输出n=a+(n-a)

素数探求——验证哥德巴赫猜想

```
int main()
{
    定义变量
    输入n，直到n满足题目要求为止
            //([6,2000000000]以内的偶数n)
    GoldbachTest(n);
    return 0;
}
void GoldbachTest(long n)
{
    for循环，a从3开始循环试到n/2，a的步长为2
    {
        若IsPrime(a) && IsPrime(n-a)，则
        {
            输出分解的等式 n = a + (n-a)
            退出函数调用
        }
    }
    输出n不符合哥德巴赫猜想
    退出函数调用
}
```

- 求解算法
  - a从3开始试到n/2，a+=2（测试所有奇数）
  - 若IsPrime(a) && IsPrime(n-a)，则输出n=a+(n-a)

```
int IsPrime(int x)
{
  int i, squareRoot;
  if (x <= 1) return 0;
  squareRoot = (int)sqrt(x);
  for (i=2; i<=squareRoot; i++)
  {
     if (x%i == 0)  return 0;
  }
  return 1;
}
```

```c
int main()
{
    long n;
    do{
        printf("Input n:");
        scanf("%ld", &n);
    } while (n%2!=0 || n<6 || n>2000000000);
    GoldbachTest(n);
    return 0;
}
```

函数不能嵌套定义，但可嵌套调用

```c
void GoldbachTest(long n)
{
    long a;
    for (a=3; a<=n/2; a+=2)
    {
        if (IsPrime(a) && IsPrime(n-a))
        {
            printf("%ld=%ld+%ld\n", n, a, n-a);
            return; //退出函数调用
        }
    }
    printf("%ld不符合哥德巴赫猜想\n", n);
    return;
}
```

若要输出所有可能的分解式，则应如何修改程序？

- 求解算法
  - a从3开始试到n/2，a+=2（测试所有奇数）
  - 若IsPrime(a) && IsPrime(n-a)，则输出n=a+(n-a)

```c
int IsPrime(int x)
{
  int i, squareRoot;
  if (x <= 1) return 0;
  squareRoot = (int)sqrt(x);
  for (i=2; i<=squareRoot; i++)
  {
     if (x%i == 0)  return 0;
  }
  return 1;
}
```

```
int main()
{
    long n;
    do{
        printf("Input n:");
        scanf("%ld", &n);
    } while (n%2!=0 || n<6 || n>2000000000);
    GoldbachTest(n);
    return 0;
}
```

```
void GoldbachTest(long n)
{
    long a;
    int  flag = 0;
    for (a=3; a<=n/2; a+=2)
    {
        if (IsPrime(a) && IsPrime(n-a))
        {
            printf("%ld=%ld+%ld\n", n, a, n-a);
            flag = 1; //不退出函数调用
        }
    }
    if (!flag)
    {
        printf("%ld不符合哥德巴赫猜想\n", n);
    }
}
```

若要输出所有可能的分解式，则应如何修改程序？

- **求解算法**
  - a从3开始试到n/2，a+=2（测试所有奇数）
  - 若IsPrime(a) && IsPrime(n-a)，则输出n=a+(n-a)

```
int IsPrime(int x)
{
  int i, squareRoot;
  if (x <= 1) return 0;
  squareRoot = (int)sqrt(x);
  for (i=2; i<=squareRoot; i++)
  {
     if (x%i == 0)  return 0;
  }
  return 1;
}
```

```c
int main()
{
  long n, m;
  do{
     printf("Input n:");
     m = scanf("%ld", &n);
     if (m != 1)
     {
       while (getchar()!='\n');
     }
   }while (m != 1);
   GoldbachTest(n);
   return 0;
}
```

防止输入非法字符，提高健壮性

在函数入口处检查参数取值的合法性

```c
void GoldbachTest(long n)
{
   long a;
   if (n%2!=0 || n<6 || n>2000000000)
   {
      printf("Input error!\n");
      return;
   }
   for (a=3; a<=n/2; a+=2)
   {
      if (IsPrime(a) && IsPrime(n-a))
      {
        printf("%ld=%ld+%ld\n", n, a, n-a);
        return;
      }
   }
   printf("%ld不符合哥德巴赫猜想\n", n);
   return;
}
```

常用的错误处理技术

- 检查所有来源于外部的数据，在非法输入中保护你的程序
- （1）终止程序运行
  - 对于非常严重的错误，一旦出现，输出错误信息后立即终止程序的执行
  - 调用错误处理子程序，进行程序结束前的处理（如资源释放等）
- （2）继续让程序运行
  - a.换用下一个正确的数据，直到正确为止
  - b.返回一个错误码
    - 意味着将错误交由其他函数来处理，而不是本函数处理
    - 函数调用结束，需判断函数调用成功与否

素数探求——回文素数

- 苏东坡曾写过一首回文诗，这首诗顺读，读者仿佛看到了从月夜景色到江天破晓的画面
- 而反过来读，仿佛是一幅从黎明晓日，到渔舟唱晚的画卷：

潮随暗浪雪山倾，远浦渔舟钓月明。
桥对寺门松径小，槛当泉眼石波清。
迢迢绿树江天晓，霭霭红霞晚日晴。
遥望四边云接水，碧峰千点数鸥轻。

轻鸥数点千峰碧，水接云边四望遥。
晴日晚霞红霭霭，晓天江树绿迢迢。
清波石眼泉当槛，小径松门寺对桥。
明月钓舟渔浦远，倾山雪浪暗随潮。

素数探求——回文素数

- 对一个素数n，左读和右读都是相同的.
  - 例如11，101，313等
  - 编程计算并输出不超过n（100<=n<=1000）的回文素数，并统计这些回文素数的个数，其中n的值从键盘输入（不考虑非法字符输入的情况）。

```
// 计算并输出不超过n（100<=n<=1000）的回文素数，并返回回文素数的个数
int PalindromicPrime(int n);
```

- 求解算法
  - m从10开始试到n-1
  - 将m分离出个位、十位或百位，计算其右读（逆序）结果t
  - 若(t==m) && IsPrime(m)，则是回文素数
  - m<100和m>=100分两种情况计算逆序数

素数探求——回文素数

```c
#include<stdio.h>
#include<math.h>
int IsPrime(int x);
int PalindromicPrime(int n);
int main()
{
    int n, count;
    do
    {
        printf("Input n:");
        scanf("%d", &n);
    }while (n<100 || n>1000);
    count = PalindromicPrime(n);
    printf("count=%d\n", count);
    return 0;
}
```

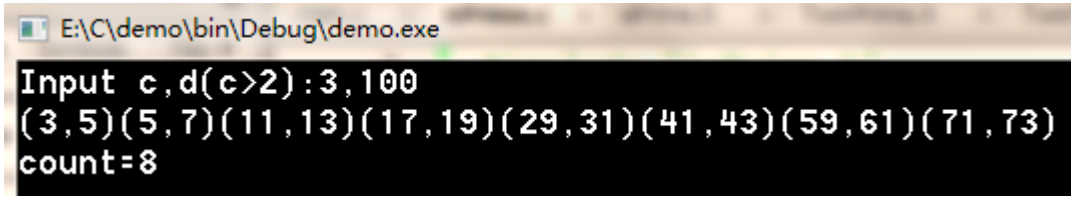
```c
int PalindromicPrime(int n)
{
    int i, j, k, t, m, count = 0;
    for (m=10; m<n; m++)
    {
        i = m / 100;
        j = (m - i * 100) / 10;
        k = m % 10;
        if (m < 100)  //两位数
        {
            t = k * 10 + j;
        }
        else   //三位数
        {
            t = k * 100 + j * 10 + i;
        }
        if (m==t && IsPrime(m))
        {
            printf("%d\t", m);
            count++;
        }
    }
    printf("\n");
    return count;
}
```

素数探求——孪生素数

- 相差为2的两个素数。
  - 例如，3与5，41与43等
  - 编程计算并输出指定区间[c,d]上的所有孪生素数对，并统计这些孪生素数的个数，其中区间上限d和下限c（c>2，d>c）由键盘输入

    //打印[min,max]之间的孪生素数，返回其间孪生素数的个数

    int TwinPrime(int min, int max);



- 求解算法
  - i从min开始试到max-2
  - 若IsPrime(i) && IsPrime(i+2)，则输出这两个孪生素数
  - 加速：若min为偶数，则min++，并且i+=2（测试所有奇数）

课后作业：素数探求——梅森素数

- 形如2^i-1的素数，称梅森素数，或梅森尼数
  - 素数有无穷多个，但目前只发现有极少量的素数能表示成 2^i-1（i为素数）的形式，这就是梅森素数（如3、7、31、127等），以17世纪法国数学家马林·梅森的名字命名。
  - 编程计算并输出指数i在[2,n]中的所有梅森尼数，并统计梅森尼数个数，n值由键盘输入，且n不大于50 。
- 求解算法
  - i从2开始试到n
  - 计算m=2^i-1，若IsPrime(m)，则输出i和m

素数探求——梅森素数

```
int Mensenni(int n)
{
    定义变量
    for (i=2; i<=n; i++)
    {
        迭代计算2^i
        m=2^i-1;
        若m是素数，则
        {
            count++;
            printf("2^%d-1=%ld\n", i, m);
        }
    }
    return count;
}
```

```
F:\c\su\bin\Debug\su.exe
Input n:51
Input n:50
2^2-1=3
2^3-1=7
2^5-1=31
2^7-1=127
2^13-1=8191
2^17-1=131071
2^19-1=524287
2^31-1=2147483647
count=8
```

```
int IsPrime(int x)
{
  int i, squareRoot;
  if (x <= 1) return 0;
  squareRoot = (int)sqrt(x);
  for (i=2; i<=squareRoot; i++)
  {
     if (x%i == 0)  return 0;
  }
  return 1;
}
```

- 求解算法
  - i从2开始试到n
  - 计算m=2^i-1，若m是素数，则输出i和m

使用函数编程的好处

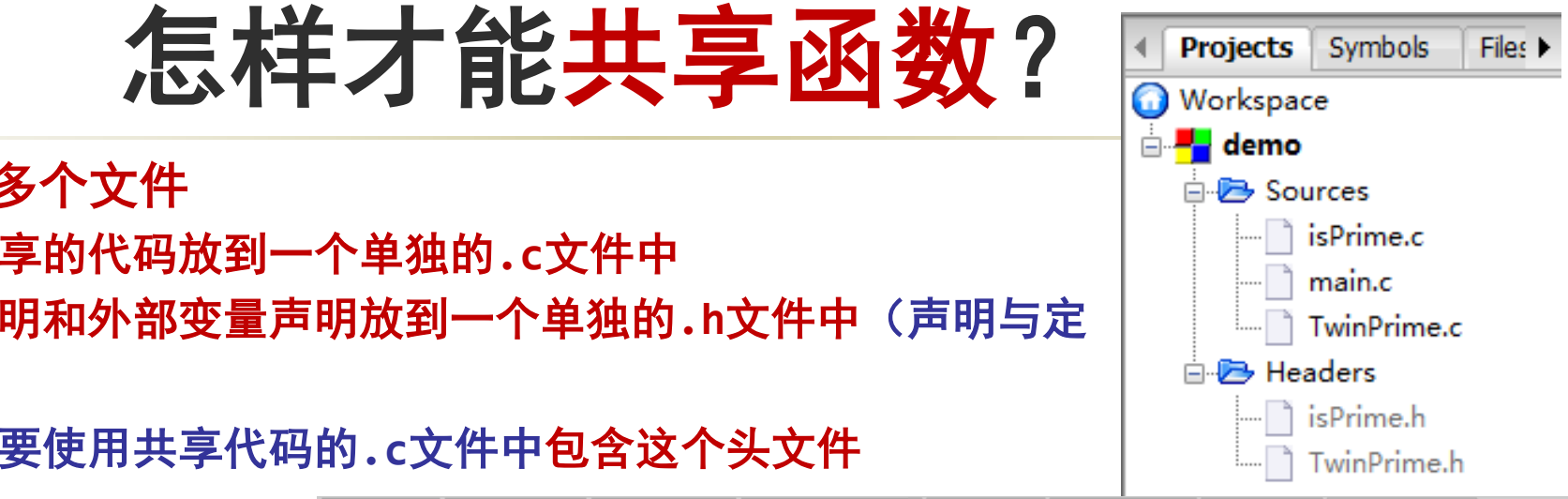
```
int IsPrime(int x)
{
  int i, squareRoot;
  if (x <= 1) return 0;
  squareRoot = (int)sqrt(x);
  for (i=2; i<=squareRoot; i++)
  {
     if (x%i == 0)  return 0;
  }
  return 1;
}
```

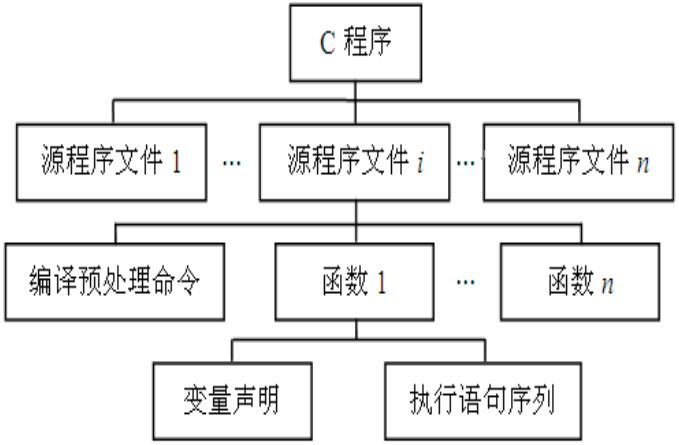
活字印刷
随时拼版
重复使用
易于保管

- 函数封装
  - 把函数内的实现细节对外隐藏，只提供对外接口（Interface）
  - 外界对函数的影响—仅限于入口参数
  - 函数对外界的影响—返回值，…
- 便于实现信息隐藏（Information Hiding）和函数复用（Reuse）

怎样才能共享函数？

- 把代码分成多个文件
  - 把需要共享的代码放到一个单独的.c文件中
  - 把函数声明和外部变量声明放到一个单独的.h文件中（声明与定义分离）
  - 在所有需要使用共享代码的.c文件中包含这个头文件






```c
// main.c
#include <stdio.h>
#include "TwinPrime.h"
int main()
{
    int c, d, n;
    do{
        printf("Input c
        scanf("%d,%d",
    }while (c<=2 || c>=
    n = TwinPrime(c, d)
    printf("count=%d\n"
    return 0;
}
```

```c
// isPrime.h
int IsPrime(int x);
```

```c
// TwinPrime.h
int TwinPrime(int min, int max);
```

```c
// isPrime.c
#include <math.h>
#include <stdio.h>
// 函数功能：判断x是否是素数，若函数返回0，
int IsPrime(i
{
    int i, fl
    int squar
    if (x <=
    for (i=2;
    {
        if (x
    }
    return fl
}
```

```c
// TwinPrime.c
#include <stdio.h>
#include "isPrime.h"
// 函数功能：打印[min,max]之间的孪生
int TwinPrime(int min, int max)
{
    int i, front = 0;
    int count = 0;
    if (min%2 == 0)
    {
        min++;
    }
    for (i=min; i<=max; i+=2)
    {
        if (IsPrime(i))
        {
```

- 构成C语言程序的基本模块

投票

最多可选1项

设置

你能跟上老师现在的讲课进度吗？

A 能跟上

B 基本能跟上

C 有点跟不上，不过我会努力的

D 完全跟不上，郁闷

提交