素数

判断某个正整数是否是素数（试除法）

问题描述：

从键盘输入一个正整数 x，编写程序判断 x 是否为素数，若是输出“Yes”，否则输出“No”。

问题分析：

素数是指除了 1 和本身，没有其他因子的自然数。例如：2，3，5，7，11，…等都是素数。判断一个数是否为素数通常要对该数所有可能的因子进行判断。在此我们介绍一种最普通的算法：试除法。

试除法的思想是：除了 1 和 x 本身外，用 2～x-1 中的每个数去试除 x，如果有一次被整除，则说明 x 不是素数(因为在 2～x-1 区间中至少存在一个 x 的因子)；如果整个区间都不存在 x 的因子，则说明 x 是素数。但如果 x 很大时，必须试除的次数很多，导致算法效率很低。为了提高试除的效率，可以从 2 循环到 $\sqrt{x}$ 即可判断 x 是否为素数。为什么呢？因为对于任何一个数 x，它的因子是成对出现的，如果到它的算术平方根都找不到 x 的因子，那么这个数不可能再有因子。

程序清单：

```
#include "stdio.h"
#include "math.h"
int prime(int n) // 判断 n 是否为素数，若是，函数返回 1，否则返回 0
{
  int i;
  for(i=2;i<=sqrt(n);i++) // 采用试除法判断 n 是否为素数
    if(n%i==0) return 0;
  return 1;
}

main()
{
  int x,f;
  printf("Input x(x>=2):");
  scanf("%d",&x);
  if(prime(x)==1)  // 若 prime 函数的返回值为 1，说明 x 是素数，否则不是素数
    printf("Yes");
  else
    printf("No");
  getch();
}
```

求某个区间内的所有素数（筛选法）

问题描述：

用筛选法求 10000 以内的素数，输出格式为：每行 15 个，最后一行输出 10000 内素数的总个数。

算法分析：

筛选法求 2 至 n 之间的素数的方法大致如下：在 2 到 n 之间划去 2 的倍数(不包括 2)，再划去 3 的倍数(不包括 3)，（由于在划去 2 的倍数时，4 已经被划去）再划去 5 的倍数……直至划去不超过 $\sqrt{n}$ 的数的倍数(因为数学中已经证明过，任一个合数 n 的不等于 1 的最小正因数不大于 $\sqrt{n}$)，剩下的数都是素数。

S1:建立一个 2 到 n 的集合 s={2..n}。这里的“集合”指的是数学中的集合概念。

S2:p=2，从 2 开始，准备划去集合 s 中所有 2 的倍数。

S3:若 p>$\sqrt{n}$，则 s6;

S4:在集合中划去 p 的倍数(不包括 p);

S5:在 p 的后继数中找出第一个未被划去的数作为数 p，转 s3;

S6:集合 s 中的所有合数均被划去，集合中剩下的元素数全部为素数，算法结束。

利用筛选法求某区间范围内的素数，较之于试除法求素数，要简单得多，算法所需的时间也会少得多，特别是 n 的值很大时，这种方法的优越性越发明显。

程序清单：

```
#include "stdio.h"
#define N 10001
main()
{
  int a[N],i,k;
  for(i=2;i<N;i++) a[i]=1;
  for(i=2;i<=N/2;i++)
  {
    if(a[i]==1)
    {
      k=2;
      while(i*k<N)
      {  a[i*k]=0;  k++; }
    }
  }
  k=0;
  for(i=2;i<N;i++)
    if(a[i]==1)
    {
      printf("%5d",i);
      k++;
      if(k%15==0) printf("\n");
    }
  printf("\nTotal=%d",k);
}
```

哥德巴赫猜想（偶数分解）

问题描述：

哥德巴赫猜想说“每个大于 2 的偶数都可以表示成两个质数的和”。编程用 100 内的所有偶数来验证此猜想。

输出样例：

4=2+2

6=3+3

8=3+5

……

100=3+97

注意输出的格式。

程序清单：

```
#include "stdio.h"
#include "math.h"
#define M 101
int a[M];

void prime()                    // 筛选法求 100 内所有的素数
{
  int i,k;
  a[0]=0; a[1]=0;               // 屏蔽 0 和 1
  for(i=2;i<M;i++) a[i]=1;      // 从 2～100 全部置 1
  for(i=2;i<=sqrt(M);i++)       // 将 2～100 间全部素数筛选出来
  {
    if(a[i]==1)
    {
      k=2;
      while(i*k<M)
      { a[i*k]=0; k++; }
    }
  }
}

void sushu()                    //  将偶数分解为两个素数之和
{
  int i,x,y;
  for(i=4;i<M;i+=2)
  {
    x=2;
    while(1)
    {
      y=i-x;
```

```
        if(a[x]==1&&a[y]==1)  // 此表达式也可写成：if(a[x]&&a[y])
        {
          printf("%d=%d+%d\n",i,x,y);
          break;
        }
        x++;
      }
    }
}

main()
{
  prime();
  sushu();
  getch();
}
```

哥德巴赫猜想（奇数分解）

问题描述：

任一奇数 X(X>=7)都可以分解为三个素数之和。使用 7-100 间的奇数验证此猜想。

输出样例：

7=2+2+3

9=3+3+3

11=2+2+7

……

注意输出的格式。

程序清单：

```
#include "stdio.h"
#include "math.h"

int prime(int n)
{
  int i;
  for(i=2;i<=sqrt(n);i++)
    if(n%i==0) return 0;
  return 1;
}

main()
{
  int x,a,b,c;
  for(x=7;x<=100;x+=2)
  {
```

```
    a=2;
    while(1)
    {
      b=a;
      c=x-a-b;
      if(prime(a) && prime(b) && prime(c))
      {
        printf("%d=%d+%d+%d\n",x,a,b,c);
        break;
      }
      a++;
    }
  }
  getch();
}
```