

# 法律声明

---

- 本课件包括：演示文稿，示例，代码，题库，视频和声音等，小象学院拥有完全知识产权的权利；只限于善意学习者在本课程使用，不得在课程范围外向任何第三方散播。任何其他人或机构不得盗版、复制、仿造其中的创意，我们将保留一切通过法律手段追究违反者的权利。



关注 小象学院

---

# 第一课 编程入门

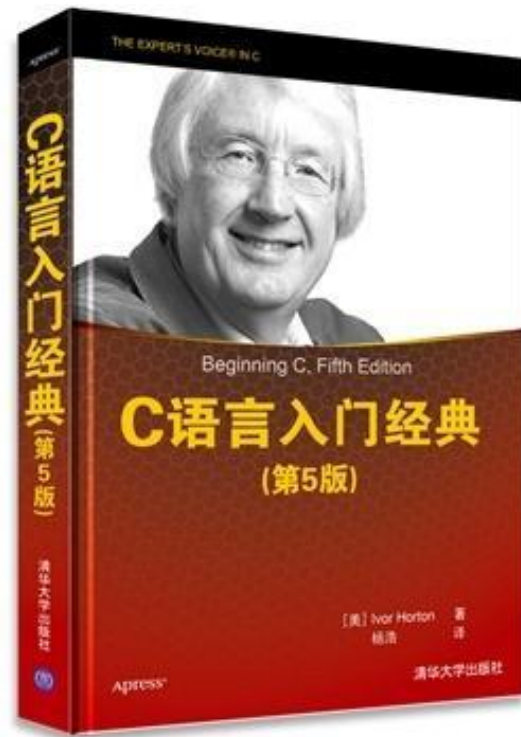
林沐

# 课程目标与建议

目标：掌握C语言程序设计，培养**编程思想**，入门**算法设计**；为后续的计算机相关专业奠定**坚实基础**。

建议：上课学习，跟上**知识点**讲解与**课堂练习**；课后复习，推荐1本C语言教程：《**C语言入门经典**》。

学习**编程**，练习**编写代码**永远比**阅读**编程书籍**更重要**；**编译器**是比**教科书**更好的编程伙伴。



# 内容概述

---

## 第一部分:程序基础知识, **顺序程序**

- 1.内存
- 2.变量、取地址、赋值运算
- 3.变量的类型
- 4.输入与输出
- 5.基本运算
- 6.数学函数
- 7.常量
- 8.例1-计算圆的周长
- 9.类型转换
- 10.无符号类型的变量
- 11.数据溢出

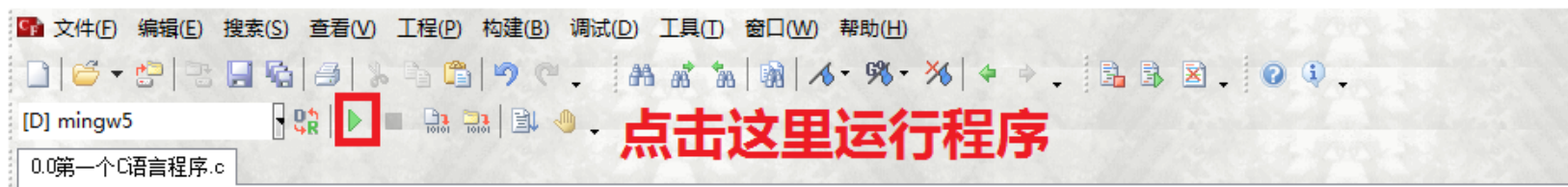
## 第二部分:条件判断语句, **分支程序**

- 1.关系运算符
- 2.逻辑运算符
- 3.运算符的优先级
- 4.if语句
- 5.if-else语句
- 6.嵌套的if-else语句与多项选择
- 7.例2-念数字

## 第三部分:循环语句, **循环程序**

- 1.while循环
- 2.递增与递减运算符
- 3.for循环
- 4.break语句
- 5.continue语句
- 6.for循环的各个写法
- 7.嵌套的循环
- 8.例3-乘法表
- 9.循环总结
- 例4-哥德巴赫猜想

# 编译与运行:第一个C语言程序



推荐编译器:  
C-Free

```
1
2 /* 这是我们学习c语言的第一个程序 */
3
4 #include <stdio.h> // 用于输入输出函数的头文件
5
6 int main() {
7     printf("Just do it !\n");
8
9     return 0;
10 }
11
12
13
```

此刻，我们只需要知道，  
任何C程序都需要有main函数

这里编写代码

## 构建

正在编译 C:\C语言程序设计入门与提高\第1课\0.0第一个C语言程序.c...  
正在连接...

完成构建 0.0第一个C语言程序: 0 个错误, 0 个警告  
生成 C:\C语言程序设计入门与提高\第1课\0.0第一个C语言程序.exe

这里查看编译结果  
根据该结果，修改语法错误

# 第一部分:程序基础知识, 顺序程序

**顺序结构**是最简单的程序结构, 也是**最常用**的程序结构, 只要按照**解决问题的顺序**写出相应的语句就行, 程序语句使用**分号**结束。它的执行顺序是**自上而下**, 依次执行。

例如, 从键盘读入3门成绩, 求这三门成绩的**总成绩**与**平均成绩**。

```
1 #include <stdio.h> //预处理 引用头文件
2
3 int main() {
4     int score1;
5     int score2, score3; //定义三个整型变量
6
7     printf("Please input three scores:\n");
8     scanf("%d %d %d", &score1, &score2, &score3); //输入3个整型变量
9
10    int sum = score1 + score2 + score3;
11    double average = (double) sum / 3; //计算成绩和与平均值
12
13    printf("sum = %d\n", sum);
14    printf("average = %.11f\n", average); //输出结果
15
16    return 0;
17 }
```

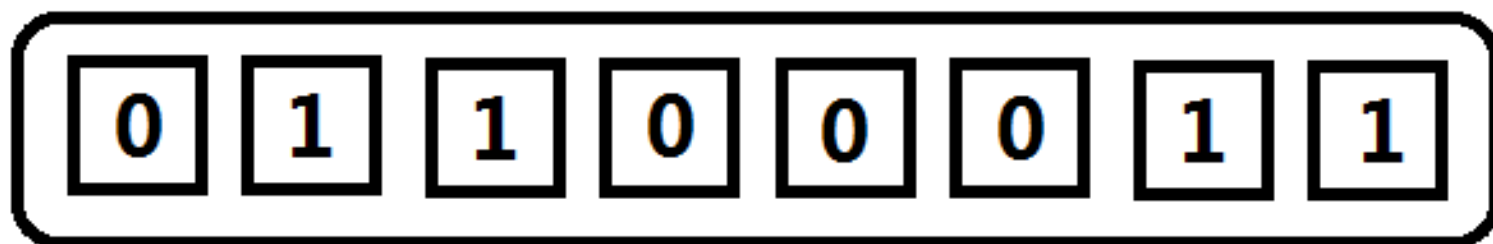
```
Please input three scores:
75 80 90
sum = 245
average = 81.7
请按任意键继续. . .
```

# 内存

计算机**执行程序**时，组成程序的**指令**和程序所操作的**数据**需要**存储**在某个地方，这个地方即为计算机的**内存**。可以**直接操作**的内存的**最小单位**是字节(byte)，一个字节由8个位(bit)组成，每个字节都有**唯一的地址**。如果存储无符号整数，一个字节可以表示0-255范围内的数字。

存储地址: 1201

$$64 + 32 + 2 + 1 = 99$$



128    64    32    16    8    4    2    1

1KB = 1024字节

1MB = 1024KB

1GB = 1024MB

1TB = 1024GB

# 变量、赋值=运算、取地址&运算

**变量**是计算机里一块**特定的内存**，它是有一个或多个**连续的**字节所组成，包括**多种**不同的类型，它们可以存储整数、浮点数、字符等等。每个变量都有一个**名称**，变量名由**1个或多个**字符组成，其中可以包括**大写**(A-Z)、**小写**(a-z)、**数字**(0-9)、**下划线**(\_)，变量名**不可**以数字开头；变量名不可与C语言**关键字**相同。

使用"="为变量进行**赋值**，可以在**变量定义**的时候进行赋值，也可以在后续的语句中为变量赋值。

使用"&"**获取**变量所在的**内存地址**。

```
a = 12345, a_address = 28ff44
A_DOUBLE_variable = 35.778960
请按任意键继续. . .
```

```
#include <stdio.h>
```

```
int main() {
```

**//变量类型** **int** a = 12345; **//变量定义的时候进行赋值**

**+变量名** **double** A\_DOUBLE\_variable;

**定义变量** **char** ch12345;

A\_DOUBLE\_variable = 35.77896; **//后续的语句中为变量赋值**

**int** \*a\_address = &a; **//取变量a的地址&a，将其存储在指针变量a\_address中**

```
printf("a = %d, a_address = %x\n", a, a_address);
```

```
printf("A_DOUBLE_variable = %lf\n", A_DOUBLE_variable);
```

```
return 0;
```

```
}
```



# 变量的类型

变量包括**不同的类型**，使用不同的**定义**方法。它们占用**不同大小**的空间存储**不同类型**的数据。一般来讲，占用**内存大**的类型的变量存储的**数据范围大**。

变量类型	定义方式	取值范围	占用内存
<b>字符型</b>	<b>char</b>	$-2^7 \sim 2^7-1$ (-128~127)	<b>1字节</b>
<b>短整型</b>	short	$-2^{15} \sim 2^{15}-1$	2字节
<b>整型</b>	<b>int</b>	$-2^{31} \sim 2^{31}-1$	<b>4字节</b>
<b>长整型</b>	long	$-2^{31} \sim 2^{31}-1$	4字节
<b>长长整型</b>	long long	$-2^{63} \sim 2^{63}-1$	8字节
<b>单精度</b>	float	$\pm 3.4E \pm 38$ (6位)	4字节
<b>双精度</b>	<b>double</b>	$\pm 1.7E \pm 308$ (15位)	<b>8字节</b>

# 输入与输出

C语言的**输入**(从键盘输入数据)使用**scanf函数**，C语言的**输出**(将数据输出到屏幕上)使用**printf函数**。

当使用它们时，需要在使用它们前，键入**预处理指令**`#include`，**包含头文件**`stdio.h`的语句(文件名写在<>中)  
`#include <stdio.h>`

**scanf()函数**，可以读取一个或多个变量。

使用第一个参数“”中的**转换说明符**控制读入变量的**个数与类型**；将数据读入到哪些变量中，即将这些**变量的地址**写在**后续的参数**中。这些地址与“”中的转换说明符，**按照顺序**一一对应。

例如，`scanf( "%d %d %d" , &a, &b, &c);`

**printf()函数**，可以打印字符串或变量。

使用第一个参数“”中的内容描述打印的内容，“”中的**转换说明符**控制打印的变量；打印哪些变量，即将这些**变量**写在**后续的参数**中。这些变量与“”中的转换说明符，**按照顺序**一一对应。

例如，`printf( "a = %d b = %d c = %d\n" , a, b, c);`

变量类型	转换说明符
char	%c
short	%hd
int	%d
long	%ld
long long	%lld
float	%f
double	%lf

# 输入与输出， 举例1

```
#include <stdio.h>
```

//预处理指令 #include ,  
包含文件stdio.h

```
int main() {  
    int n1, n2, n3;  
    double d1;  
    printf("Please input:\n");  
  
    scanf(" %d", &n1);  
  
    scanf(" %d %d %lf", &n2, &n3, &d1);  
  
    printf("n1 = %d n2 = %d\n", n1, n2);  
    printf("n3 = %d d1 = %lf\n", n3, d1);  
    return 0;  
}
```

变量类型	转换说明符
char	%c
short	%hd
int	%d
long	%ld
long long	%lld
float	%f
double	%lf

```
Please input:  
123 100 1  
23.123  
n1 = 123 n2 = 100  
n3 = 1 d1 = 23.123000  
请按任意键继续. . .
```

# 变量的定义、输入、输出，举例2

```
#include <stdio.h>
```

```
int main(){  
    char a;  
    short b;  
    int c, c1, c2, c3;  
    long d;  
    long long e;  
    float f;  
    double g;
```

```
    scanf("%c", &a);  
    scanf("%hd", &b);  
    scanf("%d", &c);  
    scanf("%ld", &d);  
    scanf("%lld", &e);  
    scanf("%f", &f);  
    scanf("%lf", &g);
```

```
    printf("char takes %d bytes, a = %c.\n", sizeof(a), a);  
    printf("short takes %d bytes, b = %hd.\n", sizeof(b), b);  
    printf("int takes %d bytes, c = %d.\n", sizeof(c), c);  
    printf("long takes %d bytes, d = %ld.\n", sizeof(d), d);  
    printf("long long takes %d bytes, e = %lld.\n", sizeof(e), e);  
    printf("float takes %d bytes, f = %.20f.\n", sizeof(f), f);  
    printf("double takes %d bytes, g = %.20lf.\n", sizeof(g), g);
```

```
    return 0;
```

```
}
```

```
x  
300  
200000000  
200000000  
90000000000000000  
1.01234567890123456789  
1.01234567890123456789  
char takes 1 bytes, a = x.  
short takes 2 bytes, b = 300.  
int takes 4 bytes, c = 200000000.  
long takes 4 bytes, d = 200000000.  
long long takes 8 bytes, e = 90000000000000000.  
float takes 4 bytes, f = 1.01234567165374760000.  
double takes 8 bytes, g = 1.01234567890123460000.  
请按任意键继续. . .
```

# 基本运算

**算术语句**格式: 变量名 = 算术表达式;

= 为**赋值**运算符, 右边的算术表达式可以使用**变量**中存储的值或**明确的数字**, 以及与+, -, \*, /等**运算符**组成的算术表达式。

运算符	动作	举例1 (整型)	举例2 (浮点型)
+	加	5 + 6 (结果为11)	5.3 + 6.9 (结果为12.2)
-	减	7 - 10 (结果为-3)	7.0 - 10 (结果为-3.0)
*	乘	10 * 5 (结果为50)	10 * 5.3 (结果为53.0)
/	除	21 / 8 (结果为2)	21 / 8.0 (结果为2.625)
%	取余	21 % 8 (结果为5)	无

**括号()****运算符**可改变运算的**优先级**, 带括号的**表达式**从内到外计算。

# 基本运算，课堂练习

林沐老师有24块糖果，每块糖果花费3.3，自己吃掉3块，将剩下的糖果平均分给8个小朋友，每个小朋友分得多少块糖果？最终剩余了多少糖果没有分？购买这些糖果需要多少钱？使用程序表达这一过程。

```
Every child can get 2 cookies.  
There are 5 cookies left over.  
To buy those cookies needs 79.20 yuan.
```

```
int main() {  
    int cookies_num = 24;  
    double total_price = 1  
  
    int eaten = 3;  
    int child_num = 8;  
    int cookies_per_child = 0;  
    int cookies_left = 0;  
  
    cookies_per_child = 2  
  
    cookies_left = 3  
  
    printf("Every child can get %d cookies.\n", cookies_per_child);  
    printf("There are %d cookies left over.\n", cookies_left);  
    printf("To buy those cookies needs %.2lf yuan.\n", total_price);  
    return 0;  
}
```

# 基本运算，实现

```
int main() {  
    int cookies_num = 24;  
    double total_price = cookies_num * 3.3;  
  
    int eaten = 3;  
    int child_num = 8;  
    int cookies_per_child = 0;  
    int cookies_left = 0;  
  
    cookies_per_child = (cookies_num - eaten) / child_num;  
  
    cookies_left = (cookies_num - eaten) % child_num;  
  
    printf("Every child can get %d cookies.\n", cookies_per_child);  
    printf("There are %d cookies left over.\n", cookies_left);  
    printf("To buy those cookies needs %.2lf yuan.\n", total_price);  
    return 0;  
}
```

# 数学函数

在程序计算时，往往需要计算**更复杂**的表达式，如表达式中需要计算平方根、自然对数等等，这时就需要调用C语言的**数学函数**了。调用数学函数时，需要引用(include) **数学头文件** math.h。

函数名	含义
<code>fabs(x)</code>	返回x的绝对值
<code>log(x)</code>	返回x的自然对数(底为e)
<code>sqrt(x)</code>	返回x的平方根
<code>exp(x)</code>	返回 $e^x$ 的值
<code>pow(x, y)</code>	返回x的y次幂
<code>sin(x)</code>	返回x(弧度值)的正弦

```
#include <stdio.h>
```

```
#include <math.h> //调用数学函数，要写这一句
```

```
int main() {  
    double num = 100;  
    double root = sqrt(num);  
    double pow_3 = pow(num, 3);  
    printf("%lf %lf\n", root, pow_3);  
    return 0;  
}
```

```
10.000000 1000000.000000  
请按任意键继续. . .
```



# 常量

在计算机程序运行时，往往需要定义一些**不会被程序修改**的量，这些量需要定义为**常量**，例如圆周率PI。C语言中，有**两种方法**定义常量。

方法1，在**预处理**时定义：#define PI 3.14159265

这种定义方法将PI定义为“3.14159265”的**取代符号**，后面的程序中，只要出现PI，编译器会将它**替换**为“3.14159265”。

方法2，定义为**常量变量**：const double PI = 3.14159265;

这种定义方法将PI定义为值**无法修改**的变量，即变量PI增加了const修饰后，后续的程序就**无法**对变量PI进行**修改**了，变量PI就成为了常量PI。

```
#include <stdio.h>
```

```
#define PI1 3.14159265
```

```
int main() {  
    printf("%.10lf\n", PI1);  
    const double PI2 = 3.14159265;  
    printf("%.10lf\n", PI2);  
    return 0;  
}
```

```
#include <stdio.h>
```

```
#define PI1 3.14159265
```

```
int main() {  
    printf("%.10lf\n", PI1);  
    const double PI2 = 3.14159265;  
    printf("%.10lf\n", PI2);
```

```
    PI1 = 0;  
    PI2 = 0;
```

**非法的修改**

```
    return 0;
```

```
}
```

```
3.1415926500  
3.1415926500
```

```
1.4常量.c:10: error: invalid lvalue in assignment
```

```
1.4常量.c:11: error: assignment of read-only variable 'PI2'
```

# 例1-计算圆的周长

---

键盘**输入**圆的**面积**，在屏幕上**打印**出圆的**周长**。

其中Pi取3.1416，圆面积公式  $\pi * r * r$ ，周长公式  $2 * \pi * r$ ，最终结果**保留2位小数**。

程序执行举例：

Please input the area of circle:

100

The diameter of circle is 35.45.

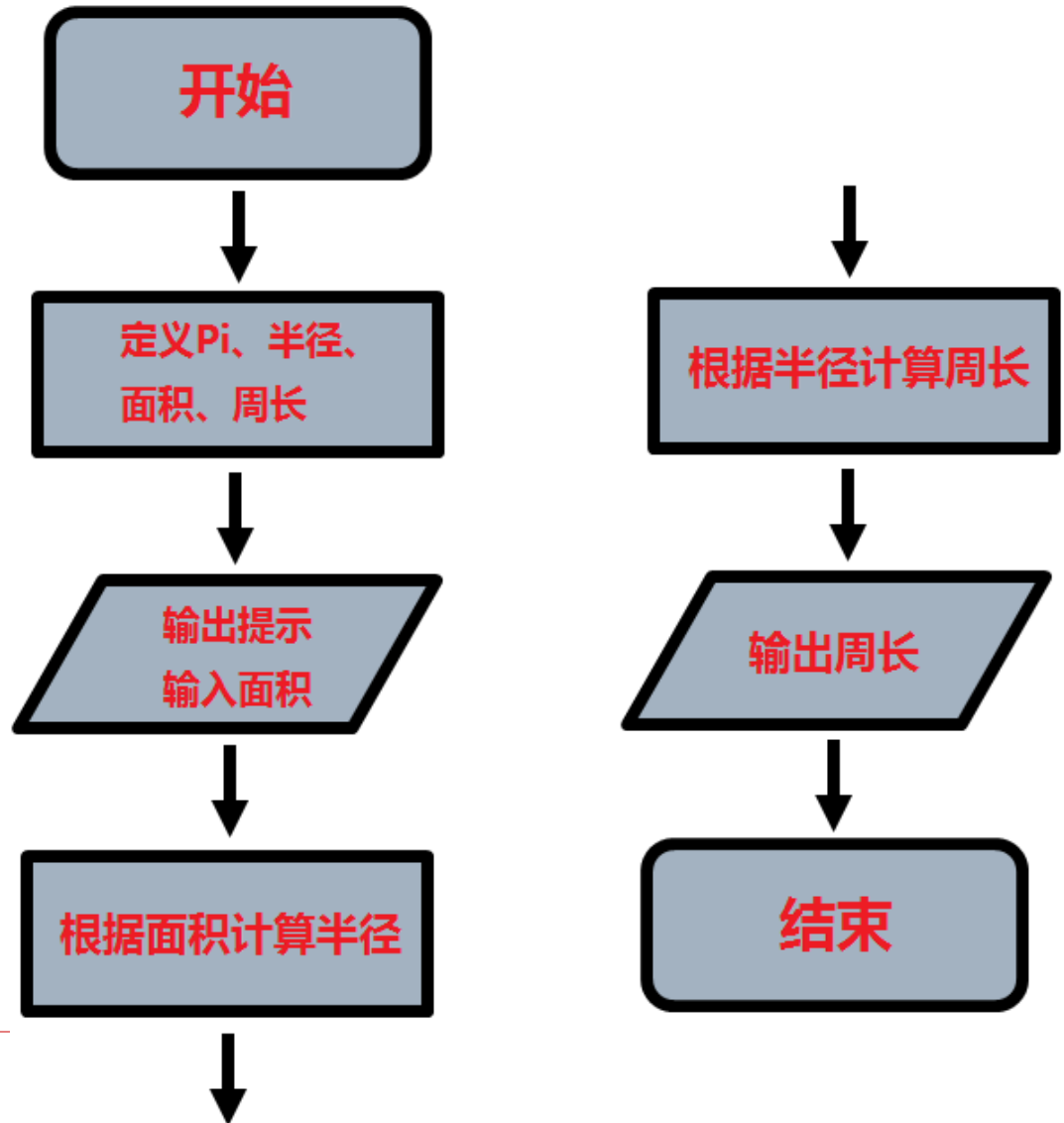
**思考：**

- 1.在这个问题中，需要定义哪些**变量**与**常量**？
- 2.需要调用什么**数学函数**？该库函数需要引用的**头文件**是什么？
- 3.程序的**计算流程**是怎样的？

# 例1-算法设计

设计**算法**是程序设计的**核心**。为了表示一个算法，最常用的是**算法流程图**，算法流程图中，使用特定的**图形符号**加上**说明**来表示算法。

一般使用**圆矩形**表示开始与结束、**菱形**表示输入输出、**矩形**表示执行的程序。



# 例1-课堂练习

```
#include <stdio.h>
```

```
#include
```

1

```
int main() {
```

2

```
Pi = 3.1416;
```

```
double area;
```

```
double radius;
```

```
double diameter;
```

```
printf("Please input the area of circle:\n");
```

```
scanf("%lf",
```

3

```
);
```

```
radius =
```

4

```
diameter =
```

5

```
printf("The diameter of circle is %lf.\n", diameter);
```

```
return 0;
```

```
}
```

3分钟，填写代码，有问题提出！

# 例1-实现与测试

```
Please input the area of circle:
100
The diameter of circle is 35.449118.
```

```
#include <stdio.h>
```

```
#include <math.h>
```

//使用sqrt数学函数需要包含  
<math.h>头文件

```
int main() {
```

```
    const double Pi = 3.1416; //Pi是一个不会改变的常  
                                数，需要定义为常量
```

```
    double area;
```

```
    double radius;
```

```
    double diameter;
```

```
    printf("Please input the area of circle:\n");
```

```
    scanf("%lf", &area); //将浮点数读入area，注意需要有&取地址符号
```

```
    radius = sqrt(area / Pi); //根据 $Pi * r * r = area$  计算半径
```

```
    diameter = 2 * Pi * radius; //计算周长
```

```
    printf("The diameter of circle is %lf.\n", diameter);
```

```
    return 0;
```

```
}
```

# 类型转换

类型转换包括**显式转换的强制类型转换**与**隐式类型转换**,

1)**显示的**强制类型转换:把变量从一种类型**转换**为另一种类型, 将**目标类型**放在变量前面的括号中。

2)隐式类型转换:发生在**二元运算**或**赋值运算**时,

a.在二元运算时, 将**值域较小的**操作数类型**转换**为**值域较大的**操作数类型。

b.在赋值运算时, 当赋值运算符**右边的**表达式值与**左边的**变量类型**不同**时, 即进行隐式类型转换。

```
#include <stdio.h>
```

```
int main() {
```

```
    int total_score = 200;
```

```
    double average1 = total_score / 3; //没有类型转换, 结果错误(丢失小数)
```

```
    double average2 = (double) total_score / 3; //显示的强制类型转换
```

```
    double average3 = total_score / 3.0; //二元运算时的隐式类型转换
```

```
    printf("average1 = %lf\n", average1);
```

```
    printf("average2 = %lf\n", average2);
```

```
    printf("average3 = %lf\n", average3);
```

```
    double test1 = total_score; //赋值时的隐式类型转换, 值域小的整型赋值到值域大的浮点型
```

```
    int test2 = average2; //赋值时的隐式类型转换, 值域大的浮点型赋值到值域小的整型
```

```
    printf("test1 = %lf, test2 = %d\n", test1, test2);
```

```
    return 0;
```

```
}
```

```
average1 = 66.000000
average2 = 66.666667
average3 = 66.666667
test1 = 200.000000, test2 = 66
请按任意键继续. . .
```

# 无符号类型的变量

有些数据总是**非负的**，例如一个班级学生的个数。对于每个**带符号**的整数类型，都有一个对应**无符号**整数类型，同类型的**有符号**与**无符号**数据**占用的内存空间相同**。**同类型**的无符号变量比有符号变量存储的数据范围**大一倍**。浮点类型数据**没有**无符号类型。

变量类型	定义方式	取值范围	占用内存
字符型	unsigned char	$0 \sim 2^8-1$ (0~255)	1字节
短整型	unsigned short	$0 \sim 2^{16}-1$	2字节
整型	unsigned int	$0 \sim 2^{32}-1$	4字节
长整型	unsigned long	$0 \sim 2^{32}-1$	4字节
长长整型	unsigned long long	$0 \sim 2^{64}-1$	8字节

# 数据溢出

当向某变量中存储超过其**存储范围**的数据时，即会发生**数据溢出**现象。

```
#include <stdio.h>
```

```
int main() {  
    char ch1 = 200;  
    unsigned char ch2 = 200;  
    int num1 = 2000000000;  
    int num2 = 40000000000;  
    printf("ch1 = %d ch2 = %d\n", ch1, ch2);  
    printf("num = %d num2 = %d\n", num1, num2);  
    return 0;  
}
```

```
ch1 = -56 ch2 = 200  
num = 2000000000 num2 = 1345294336  
请按任意键继续. . .
```



# 课间休息10分钟！

---

## 有问题提出！

# 第二部分:条件判断语句, 分支程序

程序执行时, 除了**顺序**执行语句, 往往还需要**根据一些情况**(主要是比较表达式的值)的结果, **有选择的**执行某组语句。我们需要在程序中做出**判断**, **选择执行**一组程序语句, 而**不执行**另一组程序语句。

例:键盘输入一个**整型**成绩, 数据范围是0-100, 若**超出该范围**, 屏幕打印"The score is out of range."。

数据范围正确时, **判断成绩的级别**。90-100分是**A级**, 80-89是**B级**, 70-79是**C级**, 60-69是**D级**, 0-59是**E级**。输出该成绩的级别。

```
#include <stdio.h>

int main() {
    int score; //成绩变量
    char level; //成绩等级
    printf("Please input the score:\n");
    scanf("%d", &score); //输入成绩

    //如果成绩范围不合法 逻辑运算 关系运算
    if ( score < 0 || score > 100 ) {
        printf("The score is out of range.\n");
    }
    else {
        if (score >= 90) { //成绩合法时, 判断成绩级别
            level = 'A';
        }
        else if (score >= 80 && score < 90) {
            level = 'B';
        }
        else if (score >= 70 && score < 80) {
            level = 'C';
        }
        else if (score >= 60 && score < 70) {
            level = 'D';
        }
        else if (score < 60) {
            level = 'E';
        }
        //输出成绩级别
        printf("The level of the
                score is : %c.\n", level);
    }
    return 0;
}
```

# 关系运算符

类似+、-、\*、/数学运算，C语言提供了用于**比较两个值的关系运算**，包括6个**关系运算符**。关系运算计算表达式"操作数1 关系运算符 操作数2"，当两操作数**满足关系运算符**时，结果是真(1)，否则为假(0)。

运算符	含义	举例	结果
<	小于	5 < 6	真(1)
<=	小于等于	81 <= 81	真(1)
==	等于	31 == 60	假(0)
!=	不等于	31 != 60	真(1)
>	大于	30 > 60	假(0)
>=	大于等于	30 >= 60	假(0)

```
#include <stdio.h>
```

```
int main() {  
    int compare1 = (5 < 6);  
    int compare2 = (81 <= 81);  
    int a = 31;  
    int b = 60;  
    int compare3 = (a == b);  
    int compare4 = (a != b);  
    int compare5 = (a > b);  
    int compare6 = (a >= b);  
    printf("compare1 = %d\n", compare1);  
    printf("compare2 = %d\n", compare2);  
    printf("compare3 = %d\n", compare3);  
    printf("compare4 = %d\n", compare4);  
    printf("compare5 = %d\n", compare5);  
    printf("compare6 = %d\n", compare6);  
    return 0;  
}
```

```
compare1 = 1  
compare2 = 1  
compare3 = 0  
compare4 = 1  
compare5 = 0  
compare6 = 0
```

# 逻辑运算符

有时执行一个关系测试不足以做出判断，需要合并两个或多个关系运算，这些关系运算可能需要同时为真、某个关系运算为真或者是其他某种组合。此时就需要使用到逻辑运算符。

运算符	含义	举例	结果
&&	逻辑与	(5 < 6) && (5 < 1)	0
	逻辑或	(81 < 81)    (1 < 20)	1
!	逻辑非	!(31 == 60)	1

```
#include <stdio.h>
int main() {
    int logic1 = (5 < 6) && (5 < 1);
    int logic2 = (81 < 81) || (1 < 20);
    int logic3 = !(31 == 60);
    printf("logic1 = %d, logic2 = %d, logic3 = %d\n",
           logic1, logic2, logic3);
    return 0;
}
```

```
logic1 = 0, logic2 = 1, logic3 = 1
请按任意键继续. . .
```

# 运算符的优先级

为了明确运算的**顺序**，我们一般为运算**加上()**，当不使用()**确定运算的优先级**时，运算按照各个运算符的**默认优先级**顺序执行。

优先级	运算符	解释	规则
1	()	括号运算	从左至右
2	+, -	正、负号	从右至左
	!	逻辑非	
	&	取地址	
	sizeof	类型字节数	
	(type)	强制类型转换	
3	*, /, %	乘法、除法、取余	从左至右
4	+, -	加法、减法	从左至右
5	<, <=, >, >=	小于(等于)、大于(等于)	从左至右
6	==, !=	等于、不等于	从左至右
7	&&	逻辑与	从左至右
8		逻辑或	从左至右
9	=	赋值	从右至左

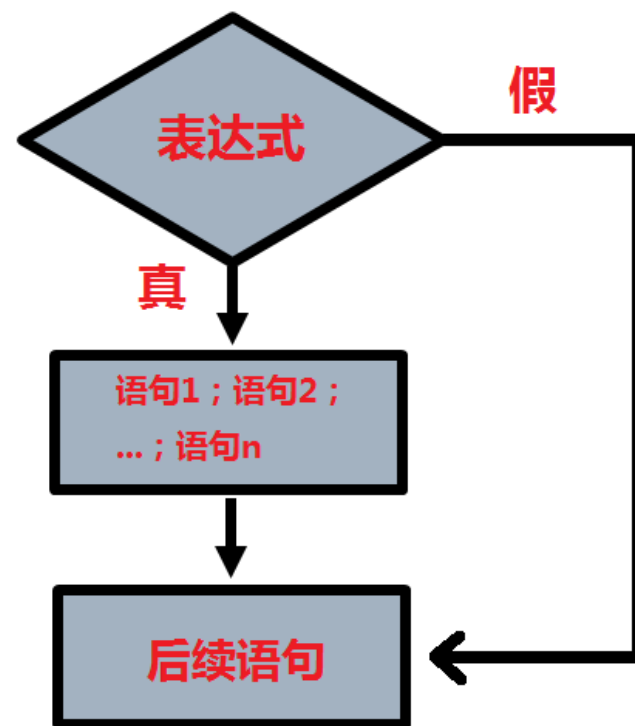
# if语句

有了表达**条件是否成立**的关系运算与逻辑运算后，就可以使用**if语句**进行**条件判断**与**分支程序**的设计了。当if后的**表达式成立**时，执行if后的大括号内的语句。

**if语句的一般形式:**

```
if (表达式){  
    语句1;  
    语句2;  
    ...  
    语句n;  
}
```

```
#include <stdio.h>  
  
int main() {  
    int score;  
    scanf("%d", &score);  
    if (score >= 90 && score <= 100) {  
        printf("Good job !\n");  
    }  
    printf("Your score is %d.\n", score);  
    return 0;  
}
```



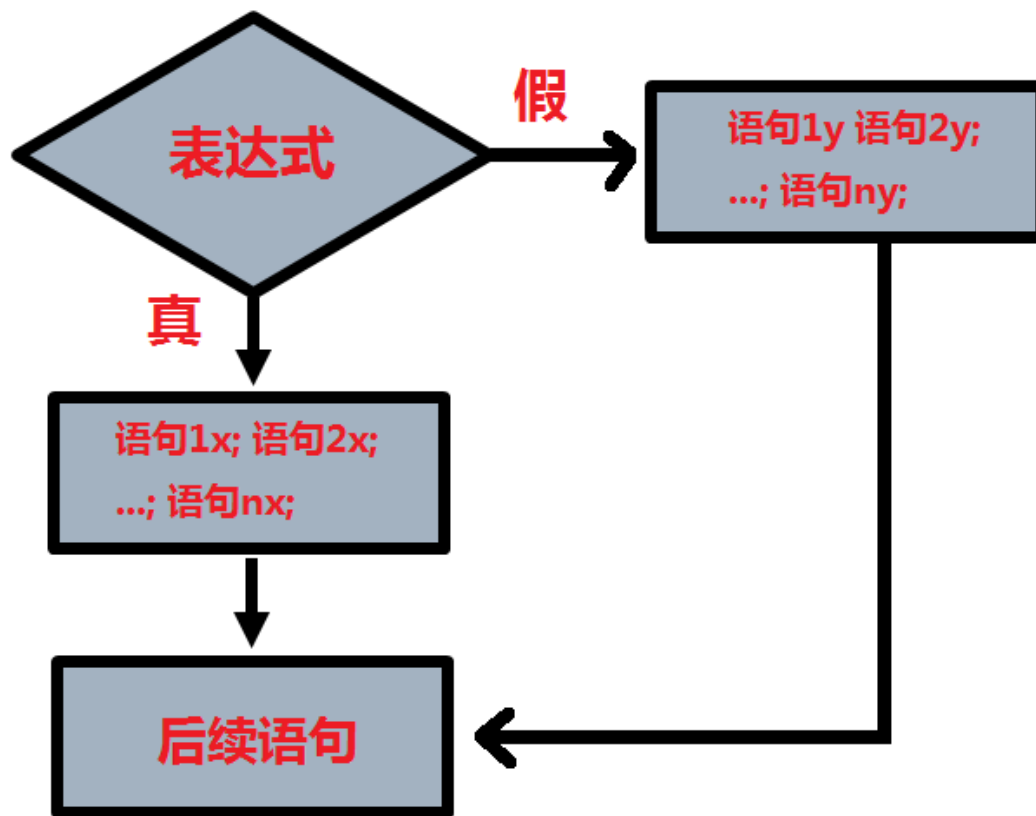
```
95  
Good job !  
Your score is 95.  
请按任意键继续. . .
```

# if-else语句

**拓展**if语句，当if后的表达式**成立时**，**执行if后**的大括号内的语句；若if后的表达式**不成立时**，**执行else后**的大括号内的语句。

**if语句的一般形式:**

```
if (表达式){  
    语句1x;  
    语句2x;  
    ...  
    语句nx;  
}  
else{  
    语句1y;  
    语句2y;  
    ...  
    语句ny;  
}
```



# if-else语句，课堂练习

从键盘输入一个字符，如果该字符**不是大写字母或小写字母**，输出"Your entered wrong letter."后退出程序；如果该字符**是大写字母**('A'-'Z')，将该字符**转换为小写**输出；如果该字符**是小写字母**('a'-'z')，将该字母**转换为大写**输出。

```
#include <stdio.h>
```

```
int main() {
```

```
    char letter;
```

```
    printf("Please enter a lowercase or an uppercase:\n");
```

```
    scanf("%c", &letter);
```

```
    if ( 1 ) {
```

```
        printf("Your entered wrong letter.\n");
```

```
        return 0;
```

```
    }
```

```
    if ( 2 ) {
```

```
        letter = letter - 'A' + 'a';
```

```
    }
```

```
    else {
```

```
        3
```

```
    }
```

```
    printf("The changed letter is %c.\n", letter);
```

```
    return 0;
```

```
}
```

```
Please enter a lowercase or an uppercase:
k
The changed letter is K.
请按任意键继续. . .
```

```
Please enter a lowercase or an uppercase:
9
Your entered wrong letter.
请按任意键继续. . .
```



# if-else语句，课堂练习实现

```
#include <stdio.h>

int main() {
    char letter;
    printf("Please enter a lowercase or an uppercase:\n");
    scanf("%c", &letter);
    if ( !(letter >= 'a' && letter <= 'z' ||  
        letter >= 'A' && letter <= 'Z') ) {
        printf("Your entered wrong letter.\n");
        return 0;
    }
    if ( letter >= 'A' && letter <= 'Z' ) {
        letter = letter - 'A' + 'a';
    }
    else {
        letter = letter - 'a' + 'A';
    }
    printf("The changed letter is %c.\n", letter);
    return 0;
}
```

# 嵌套的if-else语句与多项选择

在判断条件时，往往会遇到条件中**嵌套**条件，即if语句中可以**包含**if语句，这称为**嵌套的if语句**。

同时我们可能遇到**多项选择**问题，即某个表达式有**不同的结果**，不同的结果需要**执行不同的语句**。

```
#include <stdio.h>
int main(){
    int score; //成绩变量
    char level; //成绩等级
    printf("Please input the score:\n");
    scanf("%d", &score); //输入成绩
    //如果成绩范围不合法 逻辑运算 关系运算
    if ( score < 0 || score > 100 ){
        printf("The score is out of range.\n");
    }
    else{
        if (score >= 90){ //成绩合法时，判断成绩级别
            level = 'A';
        }
        else if(score >= 80 && score < 90){
            level = 'B';
        }
        else if(score >= 70 && score < 80){
            level = 'C';
        }
        else if(score >= 60 && score < 70){
            level = 'D';
        }
        else if(score < 60){
            level = 'E';
        } //输出成绩级别
        printf("The level of the
score is : %c.\n", level);
    }
    return 0;
}
```

# switch语句

switch语句会根据一个**整数表达式**的结果，从一组动作中**选择**一个动作。在关键字switch后，括号中integer\_expression表达式的值**确定**执行后续的哪些语句，即integer\_expression与后续的某个case后的**常量**相等时，即**执行**该case后的语句，直到break语句。如果**没有常量**与之相等，执行default后的语句。

```
switch(integer_expression){  
    case constant_1:  
        ...  
        break;  
    case constant_2:  
        ...  
        break;  
    ...  
    case constant_n:  
        ...  
        break;  
    default:  
        ...  
        break;  
}
```

# 例2-念数字

输入一个1到99的数字，将它的**念法(拼音)**打印到屏幕上，如果输入的整数**不在**1-99的范围内，打印"INPUT ERROR!"。

1-99的**念法**如：

YI、ER、SAN、SI、WU、LIU、QI、BA、JIU、SHI；

SHI YI、SHI ER、SHI SAN、SHI SI、SHI WU、SHI LIU、SHI QI、SHI BA、SHI JIU；

ER SHI YI、ER SHI ER、...、ER SHI JIU；

SAN SHI YI、SAN SHI ER、...、SAN SHI JIU；

SI SHI YI、SI SHI ER、...、SI SHI JIU；

WU SHI YI、WU SHI ER、...、WU SHI JIU；

LIU SHI YI、LIU SHI ER、...、LIU SHI JIU；

QI SHI YI、QI SHI ER、...、QI SHI JIU；

BA SHI YI、BA SHI ER、...、BA SHI JIU；

JIU SHI YI、JIU SHI ER、...、JIU SHI JIU。

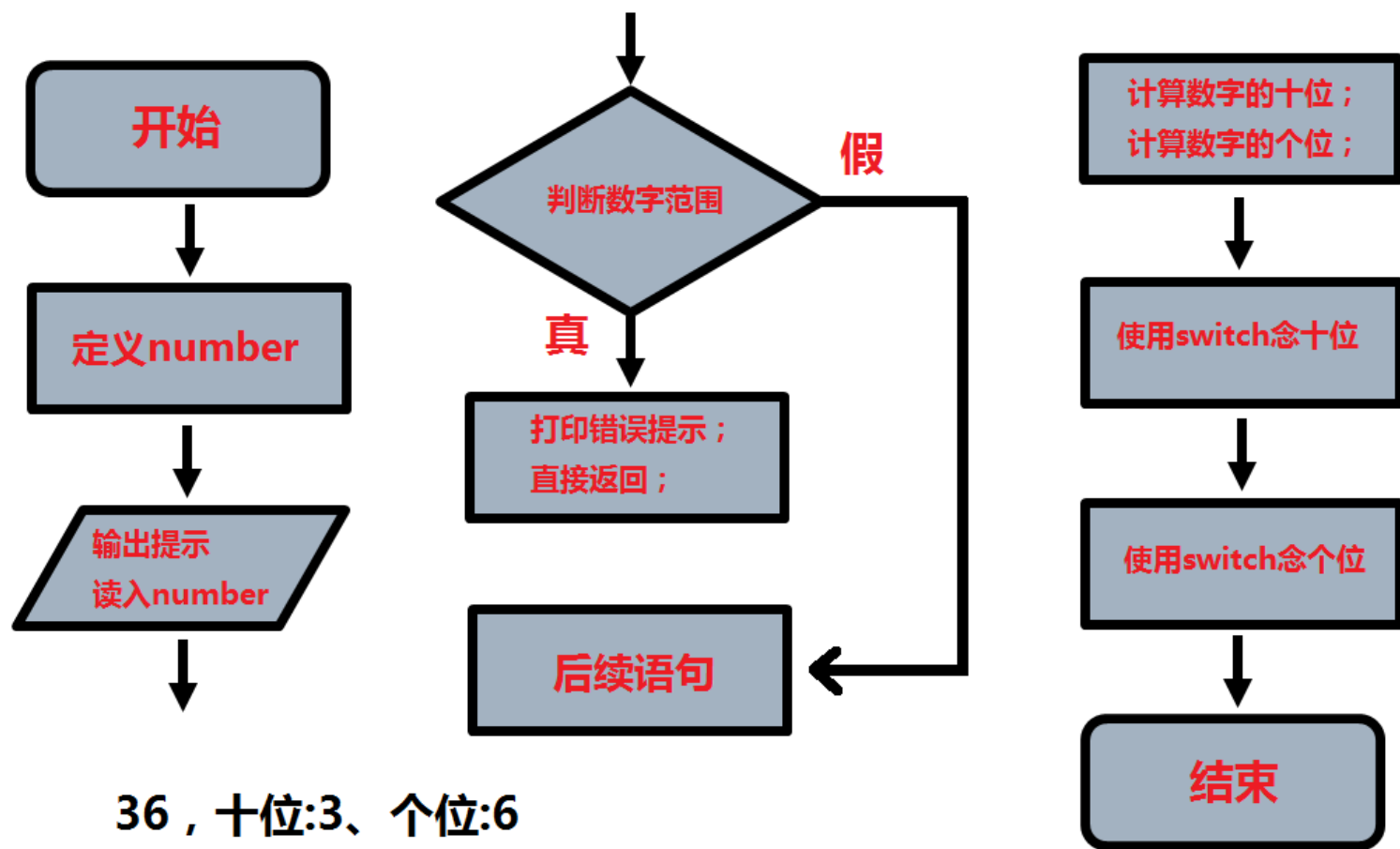
**1分钟**思考算法实现。

```
Please input a number from 1 to 99:
7
QI
请按任意键继续. . .
```

```
Please input a number from 1 to 99:
96
JIU SHI LIU
请按任意键继续. . .
```

```
Please input a number from 1 to 99:
102
INPUT ERROR!
请按任意键继续. . .
```

# 例2-算法设计



36 , 十位:3、个位:6

思考:

如何使用switch语句念(打印)十位?

如何使用switch语句念(打印)个位?

```
#include <stdio.h>
```

## 例2-课堂练习

```
int main() {
    int number;
    printf("Please input a number from 1 to 99:\n");
    scanf("%d", &number);
    if ( 1 ) {
        printf("INPUT ERROR!\n");
        return 0;
    }
    int tens_place = 2;
    int ones_place = 3;
    switch( 4 ) {
        case 0: break;
        case 1: printf("SHI ");
            break;
        case 2: printf("ER SHI ");
            break;
        case 3: printf("SAN SHI ");
            break;
        case 4: printf("SI SHI ");
            break;
        case 5: printf("WU SHI ");
            break;
        case 6: printf("LIU SHI ");
            break;
        case 7: printf("QI SHI ");
            break;
        case 8: printf("BA SHI ");
            break;
        case 9: printf("JIU SHI ");
            break;
    }
}
```

**3分钟**，填写  
代码，有问题  
提出！

```
switch( 5 ) {
    case 0:
        printf("\n");
        break;
    case 1:
        printf("YI\n");
        break;
    case 2:
        printf("ER\n");
        break;
    case 3:
        printf("SAN\n");
        break;
    case 4:
        printf("SI\n");
        break;
    case 5:
        printf("WU\n");
        break;
    case 6:
        printf("LIU\n");
        break;
    case 7:
        printf("QI\n");
        break;
    case 8:
        printf("BA\n");
        break;
    case 9:
        printf("JIU\n");
        break;
}
return 0;
```

```
#include <stdio.h>
```

## 例2-实现

```
int main(){
    int number;
    printf("Please input a number from 1 to 99:\n");
    scanf("%d", &number);
    if (number < 1 || number > 99){
        printf("INPUT ERROR!\n");
        return 0;
    }
    int tens_place = number / 10;

    int ones_place = number % 10;

    switch(tens_place){
        case 0:break;
        case 1:printf("SHI ");
            break;
        case 2:printf("ER SHI ");
            break;
        case 3:printf("SAN SHI ");
            break;
        case 4:printf("SI SHI ");
            break;
        case 5:printf("WU SHI ");
            break;
        case 6:printf("LIU SHI ");
            break;
        case 7:printf("QI SHI ");
            break;
        case 8:printf("BA SHI ");
            break;
        case 9:printf("JIU SHI ");
            break;
    }
}
```

```
switch(ones_place){
    case 0:
        printf("\n");
        break;
    case 1:
        printf("YI\n");
        break;
    case 2:
        printf("ER\n");
        break;
    case 3:
        printf("SAN\n");
        break;
    case 4:
        printf("SI\n");
        break;
    case 5:
        printf("WU\n");
        break;
    case 6:
        printf("LIU\n");
        break;
    case 7:
        printf("QI\n");
        break;
    case 8:
        printf("BA\n");
        break;
    case 9:
        printf("JIU\n");
        break;
}
return 0;
```

# 课间休息10分钟！

---

## 有问题提出！



# 第三部分:循环语句，循环程序

计算机可以**重复执行**一个语句块，直到满足**某个条件**为止，这个过程称为循环。语句块执行的重复次数可以**直接指定**或循环执行直到**满足某个条件**为止。

**输入**一个整数n，计算 $1+2+\dots+n$ 的结果。

```
#include <stdio.h>
int main() {
    int n;
    int sum = 0;
    printf("Please input n:\n");
    scanf("%d", &n);
    int i = 1;
    while (i <= n) {
        sum = sum + i;
        i = i + 1;
    }
    printf("1 + 2 + ... + n = %d\n", sum);
    return 0;
}
```

//循环终止条件

//重复执行的语句块

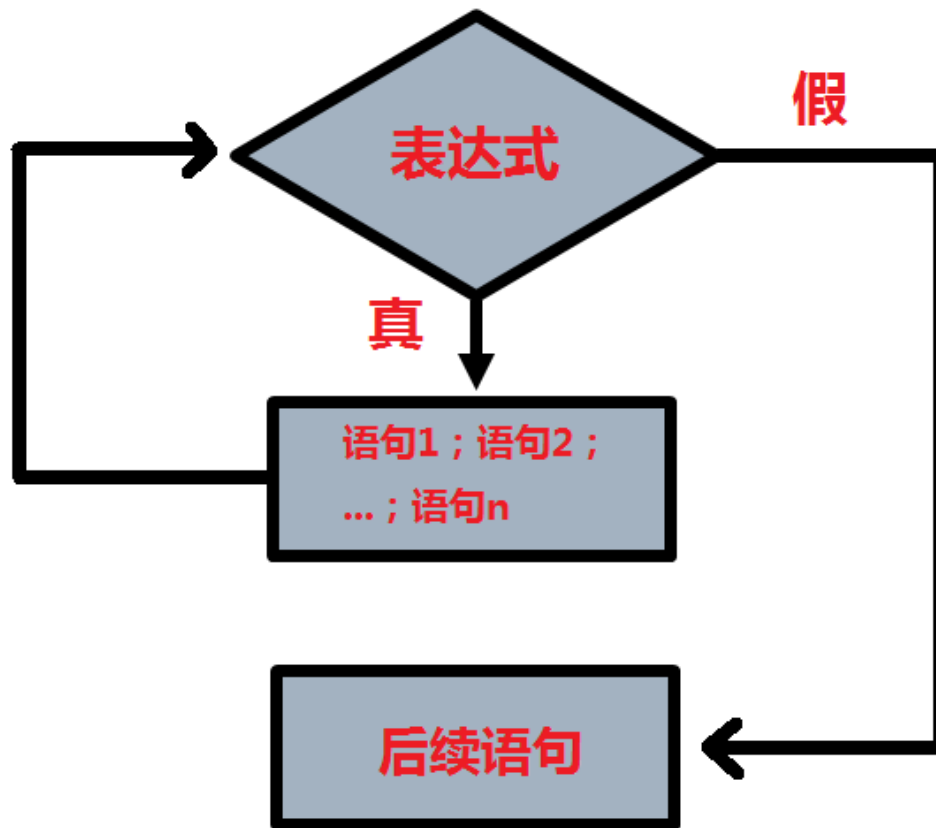
```
Please input n:
100
1 + 2 + ... + n = 5050
请按任意键继续. . .
```

# while循环

在**while循环**中，只要while后()中的**逻辑表达式**等于**true**，就**重复执行**while后的{}中的语句块。

while循环的一般**语法**如下：

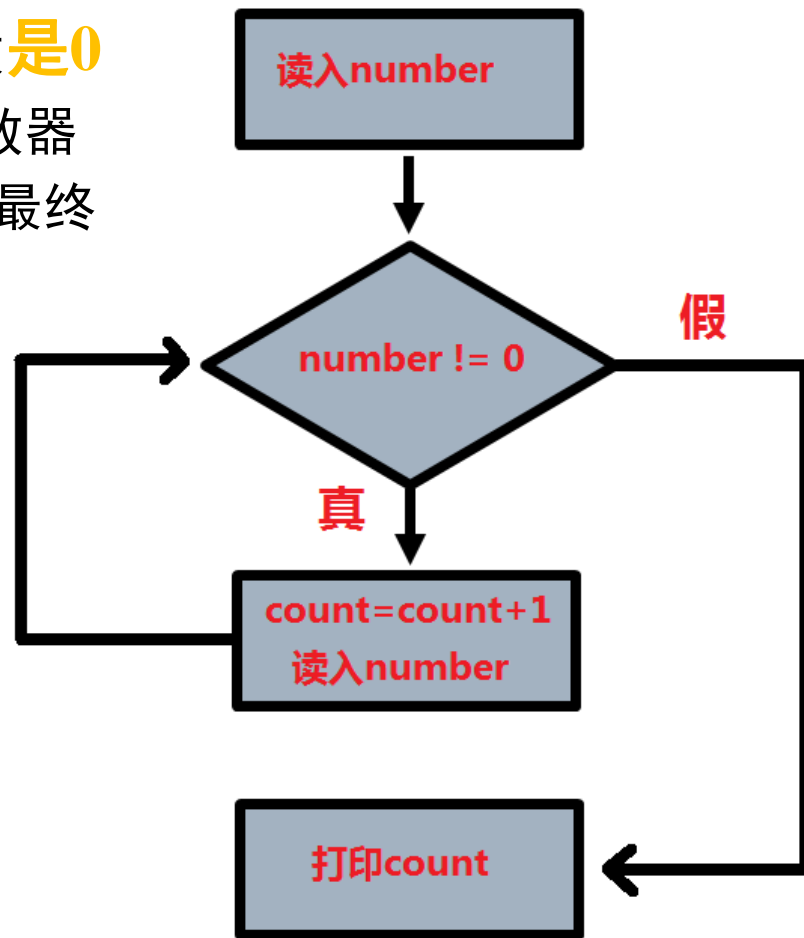
```
while(expression){  
    语句1;  
    语句2;  
    ...  
    语句n;  
}  
后续语句;
```



# while循环，举例

键盘**输入**一个**整数**，**直到**输入的整数**是0**  
**才停止**输入，在这个过程中，使用计数器  
count**记录**输入0前**输入**数字的**次数**，最终  
将输入次数打印出。

```
#include <stdio.h>
int main() {
    int number;
    int count = 0;
    scanf("%d", &number);
    while ( number != 0 ) { //循环的执行条件
        count = count + 1;
        scanf("%d", &number); //循环语句块
    }
    printf("count = %d\n", count);
    return 0;
}
```



# 递增与递减运算符

**递增运算符**(++)和**递减运算符**(--)会将存储在变量中的值**递增**或**递减**1。如果递增或递减运算**独立**成为一个语句，递增或递减运算放在变量**前面**或**后面**的效果是一样的。

如果递增或递减运算出现在**表达式**中，递增或递减运算放在变量前后的**效果不一样**；在程序编写时，为了**增加可读性**，递增或递减运算尽可能独立成为一个语句，**避免**与其他计算放**一起执行**。

```
#include <stdio.h>

int main() {
    int number = 6;
    printf("number = %d\n", number);
    number++;
    printf("number = %d\n", number);
    ++number;
    printf("number = %d\n", number);
    number--;
    printf("number = %d\n", number);
    --number;
    printf("number = %d\n", number);

    int n1 = 10;
    int temp1 = 5 + n1++;
    int n2 = 10;
    int temp2 = 5 + ++n2;
    printf("temp1 = %d temp2 = %d\n", temp1, temp2);
    return 0;
}
```

```
number = 6
number = 7
number = 8
number = 7
number = 6
temp1 = 15 temp2 = 16
请按任意键继续. . .
```

# for循环

一般**明确**循环重复执行的**次数**时，使用**for循环**。for循环的操作由关键字for后面括号中的内容控制。

for循环的一般**语法**如下:

for(循环开始时执行1次; 循环条件;  
每次循环结束后执行){

语句1;

语句2;

...

语句n;

}

后续语句;

```
#include <stdio.h>
```

```
int main() {
```

```
int n;
```

```
int sum = 0;
```

```
printf("Please input n:\n");
```

```
scanf("%d", &n);
```

```
int i;
```

循环开始执行1次

循环条件

每次循环结束时，都要执行1次

```
for ( i = 1 ; i <= n ; i++ ) {
```

```
sum = sum + i;
```

//重复执行的语句块

```
}
```

```
printf("1 + 2 + ... + n = %d\n", sum);
```

```
return 0;
```

```
}
```

```
Please input n:
100
1 + 2 + ... + n = 5050
请按任意键继续. . .
```

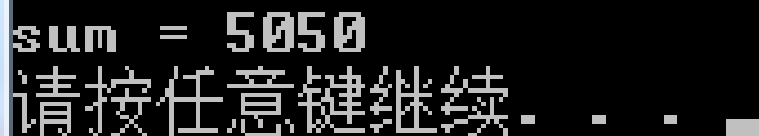
# break语句

**break语句**的作用是**终止循环体**内的语句块。break经常出现在循环中的**条件分支**中，从而在**某条件成立**时，**终止循环**。

```
#include <stdio.h>
```

```
int main() {  
    int sum = 0;  
    int i;  
    for (i = 1; i <= 200; i++) {  
        if (i > 100) {  
            break;  
        }  
        sum += i;  
    }  
    printf("sum = %d\n", sum);  
    return 0;  
}
```

**//当 i == 101时，执行  
break;语句，跳出循环  
不再执行后续的语句**



```
sum = 5050  
请按任意键继续. . .
```

# continue语句

continue语句的作用是**终止**循环体内**当前**的循环，当前循环**continue后的语句**不再执行，直接跳到循环的**下一次**开始。continue经常出现在循环中的条件分支中，从而在某条件成立时，终止**本次**循环。

```
#include <stdio.h>

int main() {
    int sum = 0;
    int i;
    for (i = 1; i <= 100; i++) {
        if (i == 50) {
            continue;
        }
        sum += i;
    }
    printf("sum = %d\n", sum);
    return 0;
}
```

```
sum = 5000
```

```
请按任意键继续. . .
```

**//当  $i == 50$  时，执行continue语句，跳出本次循环，执行后面的循环，故最终结果少了50 ( $1+2+\dots+100 == 5050$ )，最终结果是5000**

# for循环的各个写法

for循环中的(循环开时执行1次; 循环条件; 每次循环结束后执行), **均可以不写**在(;;)中。  
例如:

```
#include <stdio.h>
```

```
int main() {  
    int sum;  
    int i;
```

1

```
    sum = 0;  
    for (i = 1; i <= 100; i++) {  
        sum = sum + i;  
    }  
    printf("sum = %d\n", sum);
```

2

```
    sum = 0;  
    i = 1;  
    for ( ; i <= 100; i++) {  
        sum = sum + i;  
    }  
    printf("sum = %d\n", sum);
```

3

```
    sum = 0;  
    i = 1;  
    for ( ; i <= 100; ) {  
        sum = sum + i;  
        i++;  
    }  
    printf("sum = %d\n", sum);
```

4

```
    sum = 0;  
    i = 1;  
    for ( ; ; ) {  
        sum = sum + i;  
        i++;  
        if (i > 100) {  
            break;  
        }  
    }  
    printf("sum = %d\n", sum);  
    return 0;
```

}

```
sum = 5050  
sum = 5050  
sum = 5050  
sum = 5050
```



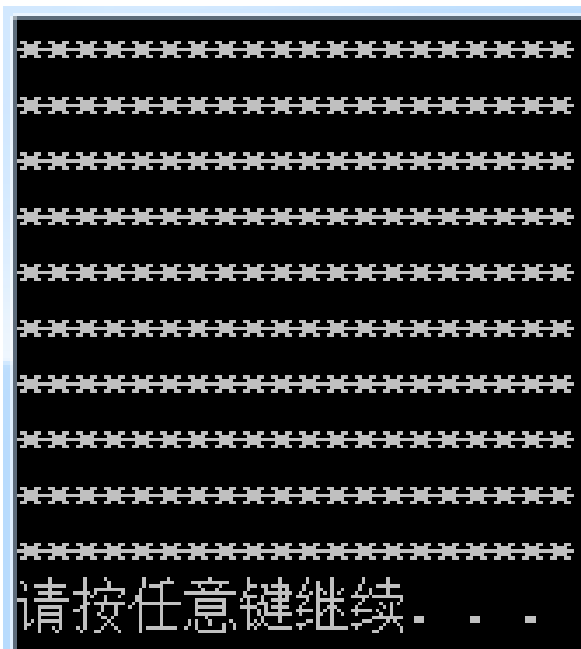
# 嵌套的循环

大多数的情况，一层循环无法满足我们的需求；可以将一个循环放在另一个循环里面，组成嵌套的循环。

例如，我们希望在屏幕上打印一个10 \* 20的，以"\*"组成的矩形:

```
#include <stdio.h>

int main() {
    int i, j;
    for (i = 0; i < 10; i++) {
        for (j = 0; j < 20; j++) {
            printf("*");
        }
        printf("\n");
    }
    return 0;
}
```



# 例3-乘法表

输出**9\*9的乘法表**（形状如直角三角形）。如下所示：

```
1*1=1
2*1=2  2*2=4
3*1=3  3*2=6  3*3=9
4*1=4  4*2=8  4*3=12  4*4=16
5*1=5  5*2=10  5*3=15  5*4=20  5*5=25
6*1=6  6*2=12  6*3=18  6*4=24  6*5=30  6*6=36
7*1=7  7*2=14  7*3=21  7*4=28  7*5=35  7*6=42  7*7=49
8*1=8  8*2=16  8*3=24  8*4=32  8*5=40  8*6=48  8*7=56  8*8=64
9*1=9  9*2=18  9*3=27  9*4=36  9*5=45  9*6=54  9*7=63  9*8=72  9*9=81
```

其中算式与算式之间用tab分隔，最后一个算式后是换行符。

1分钟**思考**算法与如下问题：

需要用**几层循环**实现，如何控制**外层循环**、如何控制**内层循环**？

# 例3-课堂练习

```
#include <stdio.h>
```

```
int main() {
```

```
    int i, j;
```

```
    for (  ;  ; i++) {
```

```
        for (  ;  ; j++) {
```

```
            printf("%d*%d=%d\t", i, j,  );
```

```
        }
```

```
        printf("\n");
```

```
    }
```

```
    return 0;
```

```
}
```

**3分钟**，填写代码  
，有问题提出！

```
1*1=1
2*1=2    2*2=4
3*1=3    3*2=6    3*3=9
4*1=4    4*2=8    4*3=12    4*4=16
5*1=5    5*2=10    5*3=15    5*4=20    5*5=25
6*1=6    6*2=12    6*3=18    6*4=24    6*5=30    6*6=36
7*1=7    7*2=14    7*3=21    7*4=28    7*5=35    7*6=42    7*7=49
8*1=8    8*2=16    8*3=24    8*4=32    8*5=40    8*6=48    8*7=56    8*8=64
9*1=9    9*2=18    9*3=27    9*4=36    9*5=45    9*6=54    9*7=63    9*8=72    9*9=81
```

请按任意键继续. . .

# 例3-实现

```
#include <stdio.h>

int main() {
    int i, j;
    for ( i = 1 ; i <= 9 ; i++) {
        for ( j = 1 ; j <= i ; j++) {
            printf("%d*%d=%d\t", i, j, i*j);
        }
        printf("\n");
    }
    return 0;
}
```

```
1*1=1
2*1=2  2*2=4
3*1=3  3*2=6  3*3=9
4*1=4  4*2=8  4*3=12  4*4=16
5*1=5  5*2=10  5*3=15  5*4=20  5*5=25
6*1=6  6*2=12  6*3=18  6*4=24  6*5=30  6*6=36
7*1=7  7*2=14  7*3=21  7*4=28  7*5=35  7*6=42  7*7=49
8*1=8  8*2=16  8*3=24  8*4=32  8*5=40  8*6=48  8*7=56  8*8=64
9*1=9  9*2=18  9*3=27  9*4=36  9*5=45  9*6=54  9*7=63  9*8=72  9*9=81
```

请按任意键继续. . .

# 循环总结

while循环与for循环**没有本质的区别**，一般来讲，如果**控制循环**(循环条件)比较**复杂**时，使用**while循环**；如果是计数相关操作，**控制循环**(循环条件)**简单**，**循环变量单一**(如只有一个i)，使用**for循环**。

```
for ( 1 ; 2 ; 3 ){  
    需要循环的语句块  
}
```

1.控制循环的变量的初始值

2.循环执行的条件

3.修改控制循环的变量

```
1  
while( 2 ){  
    需要循环的语句块  
    3  
}
```

**while**与**for**语句可以**满足一切循环功能**，除了while与for循环，还可以使用**do-while语句**和**goto语句**控制循环，由于使用较少，可读性不高，不再详述。

# 算法设计入门

---

算法（Algorithm）是指问题求解方案的**准确而完整**的描述，是一系列解决问题的清晰指令，算法一般主要包括**五个**重要的特征：

1. **有穷性**: 算法必须能在执行**有限个步骤**之后终止。
2. **确切性**: 算法的每一步骤必须有**确切的定义**。
3. **输入项**: 如键盘输入、文件输入的数据或算法本身定义的初始条件，以**刻画运算**对象的**初始情况**。
4. **输出项**: 即程序对于输入数据加工后的结果，**没有输出**的算法是**毫无意义**的。
5. **可行性**: 算法中执行的任何计算步骤都是**可执行的操作步**，即每个计算步都可以在**有限时间内完成**。

# 例4-哥德巴赫猜想

哥德巴赫猜想中写到，一个充分大的偶数（大于等于4），它可以分解为两个素数之和。

你的任务是用计算机简单验证哥德巴赫猜想，已知一个偶数 $n$ （大于等于4），将它分解为两个素数的所有可能打印出来。

程序要求：

键盘输入一个偶数 $n$ （ $4 \leq n \leq 10000$ ），代表待验证的偶数。输出偶数分解为两个素数之和的所有可能情况，每种情况占一行。

样例1:

Input:

4

Output:

$4 = 2 + 2$

样例2:

Input:

66

Output:

$66 = 5 + 61$

$66 = 7 + 59$

$66 = 13 + 53$

$66 = 19 + 47$

$66 = 23 + 43$

$66 = 29 + 37$

样例3:

Input:

200

Output:

$200 = 3 + 197$

$200 = 7 + 193$

$200 = 19 + 181$

$200 = 37 + 163$

$200 = 43 + 157$

$200 = 61 + 139$

$200 = 73 + 127$

$200 = 97 + 103$

# 例4-思考

---

1.如何判断一个整数是一个**素数**？

如，13是素数，20不是素数。

2.如何将一个整数**拆解**为两个数的和，找出该整数拆解成两个数的和的**所有可能**？

如，拆解20:

$$20 = 1 + 19$$

$$20 = 2 + 18$$

$$20 = 3 + 17$$

$$20 = 4 + 16$$

$$20 = 5 + 15$$

$$20 = 6 + 14$$

$$20 = 7 + 13$$

$$20 = 8 + 12$$

$$20 = 9 + 11$$

$$20 = 10 + 10$$

**思考3分钟！**

3.如何将一个整数拆解为**两个素数**的和？



# 例4-算法设计，判断素数

一个数是否是素数，即一个数是否有除1与它本身的因子。

需要尝试它是否可以被从2到它的平方根之间的数整除，如果可以则它不是素数，否则是素数。

举例100，可以分解：

$$100 = 2 * 50 = 4 * 25 = 5 * 20 = 10 * 10$$

100的因子是2、4、5、10、20、25、50；100可以被它们整除。

100是否有除了1与100的其他因子，只需要尝试100是否可以被2到10之间的数整除。

超过100的平方根的数，如果是100的因子则一定被尝试过了(有一个更小的数与它相乘是100)，否则一定不是它的因子。

证明，反证法：

假设  $a > 10$  是它的因子，且没被尝试过， $100 = a * b$ 。

$$\therefore 100 = a * b;$$

$\therefore b$ 是它的因子，也没被尝试过。

但  $b < 10$ ，故 **$b$** 一定被尝试过。

所以，假设不成立。

故， $a$ 不是它的因子，或 **$a$** 是它的因子已被尝试过。

# 例4-判断素数，课堂练习

```
#include <stdio.h>
#include <math.h>
int main() {
    int number;
    scanf("%d", &number);
    int prime = number;

    int max_factor = 1 + 1;
    int i;
    for (i = 2; i < max_factor; i++) {
        if (2) {
            prime = 0;
            break;
        }
    }
    if (3) {
        printf("%d is a prime.\n", number);
    }
    else{
        printf("%d is not a prime.\n", number);
    }
    return 0;
}
```

```
1
1 is not a prime.
请按任意键继续. . .
```

```
17
17 is a prime.
请按任意键继续. . .
```

```
30
30 is not a prime.
请按任意键继续. . .
```

# 例4-判断素数，实现

```
#include <stdio.h>
#include <math.h>
int main() {
    int number;
    scanf("%d", &number);
    int prime = number;

    int max_factor = sqrt(prime) + 1;
    int i;
    for (i = 2; i < max_factor; i++) {
        if (prime % i == 0) {
            prime = 0;
            break;
        }
    }
    if (prime > 1) {
        printf("%d is a prime.\n", number);
    }
    else {
        printf("%d is not a prime.\n", number);
    }
    return 0;
}
```

```
1
1 is not a prime.
请按任意键继续. . .
```

```
17
17 is a prime.
请按任意键继续. . .
```

```
30
30 is not a prime.
请按任意键继续. . .
```

# 例4-拆分整数，课堂练习

```
#include <stdio.h>
```

```
int main() {  
    int number;  
    scanf("%d", &number);  
    int i;  
    for (i = 1; i <= 1; i++) {  
        printf("%d = %d + %d\n", number, 2, 3);  
    }  
    return 0;  
}
```

```
20  
20 = 1 + 19  
20 = 2 + 18  
20 = 3 + 17  
20 = 4 + 16  
20 = 5 + 15  
20 = 6 + 14  
20 = 7 + 13  
20 = 8 + 12  
20 = 9 + 11  
20 = 10 + 10
```

# 例4-课堂练习，实现

```
#include <stdio.h>
```

```
int main() {  
    int number;  
    scanf("%d", &number);  
    int i;  
    for (i = 1; i <= number / 2; i++) {  
        printf("%d = %d + %d\n", number, i, number - i);  
    }  
    return 0;  
}
```

```
20  
20 = 1 + 19  
20 = 2 + 18  
20 = 3 + 17  
20 = 4 + 16  
20 = 5 + 15  
20 = 6 + 14  
20 = 7 + 13  
20 = 8 + 12  
20 = 9 + 11  
20 = 10 + 10
```

# 例4-整体算法设计

**外层循环**拆分整数，**内层循环**判断素数。

键盘读入number;

设置循环变量i，将i从2循环至number/2;

如果 i 与 number - i **同时是素数**:

打印  $\text{number} = i + \text{number} - i$

例如  $\text{number} = 20$ ;

$$20 = 2 + 18$$

$$20 = \boxed{3} + \boxed{17}$$

$$20 = 4 + 16$$

$$20 = 5 + 15$$

$$20 = 6 + 14$$

$$20 = \boxed{7} + \boxed{13}$$

$$20 = 8 + 12$$

$$20 = 9 + 11$$

$$20 = 10 + 10$$

# 例4-整体算法，课堂练习

```
#include <stdio.h>
#include <math.h>

int main(){
    int number;
    scanf("%d", &number);
    int i, j;
    for (i = 2; i <= number / 2; i++){
        int prime1 = 1
    }
    int max_factor = sqrt(prime1) + 1;
    for (j = 2; j < max_factor; j++){
        if (prime1 % j == 0){
            prime1 = 0;
        }
    }
}
```

```
int prime2 = 3
max_factor = sqrt(prime2) + 1;
for (j = 2; j < max_factor; j++){
    if (prime2 % j == 0){
        prime2 = 0;
    }
}
if (5){
    printf("%d = %d + %d\n",
           number, prime1, prime2);
}
return 0;
}
```

**3分钟**，填写代码  
，有问题提出！

```
30
30 = 7 + 23
30 = 11 + 19
30 = 13 + 17
```

# 例4-整体算法，实现

```
#include <stdio.h>
#include <math.h>

int main() {
    int number;
    scanf("%d", &number);
    int i, j;
    for (i = 2; i <= number / 2; i++) {
        int prime1 = i;
        int max_factor = sqrt(prime1) + 1;
        for (j = 2; j < max_factor; j++) {
            if (prime1 % j == 0) {
                prime1 = 0;
                break;
            }
        }
    }
}
```

```
int prime2 = number - i;
max_factor = sqrt(prime2) + 1;
for (j = 2; j < max_factor; j++) {
    if (prime2 % j == 0) {
        prime2 = 0;
        break;
    }
}
if (prime1 && prime2) {
    printf("%d = %d + %d\n",
           number, prime1, prime2);
}
return 0;
```

```
30
30 = 7 + 23
30 = 11 + 19
30 = 13 + 17
```



# 例4-整体算法，测试

```
#include <stdio.h>
#include <math.h>

int main() {
    freopen("例4-test.in", "r", stdin);
    freopen("例4-test-my.out", "w", stdout);

    int number;
    scanf("%d", &number);
    int i, j;
    for (i = 2; i <= number / 2; i++) {
        int prime1 = i;
```



2017/12/24 18:06:25 2,632 bytes Everything Else ▾ ANSI ▾ PC	2017/12/24 18:09:17 2,632 bytes Everything Else ▾ ANSI ▾ PC
10000 = 59 + 9941	10000 = 59 + 9941
10000 = 71 + 9929	10000 = 71 + 9929
10000 = 113 + 9887	10000 = 113 + 9887
10000 = 149 + 9851	10000 = 149 + 9851
10000 = 167 + 9833	10000 = 167 + 9833
10000 = 197 + 9803	10000 = 197 + 9803
10000 = 233 + 9767	10000 = 233 + 9767
10000 = 251 + 9749	10000 = 251 + 9749
10000 = 257 + 9743	10000 = 257 + 9743
10000 = 281 + 9719	10000 = 281 + 9719
10000 = 311 + 9689	10000 = 311 + 9689
10000 = 449 + 9551	10000 = 449 + 9551
10000 = 461 + 9539	10000 = 461 + 9539
10000 = 467 + 9533	10000 = 467 + 9533
10000 = 479 + 9521	10000 = 479 + 9521
10000 = 503 + 9497	10000 = 503 + 9497
10000 = 509 + 9491	10000 = 509 + 9491
10000 = 521 + 9479	10000 = 521 + 9479

# poj 2262 Goldbach's Conjecture 原题介绍与poj简介

POJ 即“**北京大学程序在线评测系统**”（Peking University Online Judge）的缩写，主要收录**ACM**国际大学生程序设计竞赛、**NOI**青少年信息学奥林匹克竞赛等各类程序设计竞赛题目，当前共有3000多道。

### Goldbach's Conjecture

Time Limit: 1000MS      Memory Limit: 65536K  
Total Submissions: 47144      Accepted: 17970

#### Description

In 1742, Christian Goldbach, a German amateur mathematician, sent a letter to Leonhard Euler in which he made the following conjecture:

Every even number greater than 4 can be written as the sum of two odd prime numbers.

For example:

$8 = 3 + 5$ . Both 3 and 5 are odd prime numbers.  
 $20 = 3 + 17 = 7 + 13$ .  
 $42 = 5 + 37 = 11 + 31 = 13 + 29 = 19 + 23$ .

Today it is still unproven whether the conjecture is right. (Oh wait, I have the proof of course, but it is too long to write it on the margin of this page.) Anyway, your task is now to verify Goldbach's conjecture for all even numbers less than a million.

Problem	Result	Memory	Time	Language	Code Length
2262	Accepted	176K	360MS	C	460B

# 结束

---

非常感谢大家！

林沐

# 联系我们

---

小象学院：互联网新技术在线教育领航者

— 微信公众号：**小象学院**

