

循环结构

为什么需要循环?

```
scanf("%f%f%f%f%f",&score1,&score2,&score3,&score4,&score5);
```

```
//输入一个学生5门课的成绩
```

```
aver=(score1+score2+score3+score4+score5)/5;
```

```
//求该学生平均成绩
```

```
printf("%7.2f",aver);
```

```
//输出该学生平均成绩
```

如果全班有50个学生呢?

重复写49个同样的程序段



```
i=1;
```

```
//设置变量i初值为1
```

```
while(i<=50) //当i的值小于或等于50时执行花括号内的语句
```

```
{ scanf("%f%f%f%f%f",&score1,&score2,&score3,&score4,&score5);
```

```
aver=(score1+score2+score3+score4+score5)/5;
```

```
printf("%7.2f",aver);
```

```
i++; //每执行完一次循环使i的值加1
```

C的三种循环方法

`while (表达式) 语句;`

`do{`

`语句 ;`

`} while(表达式);`

`for(表达式1; 表达式2; 表达式3) 语句;`

各有特点，一般情形下可以互相替代

循环方法的共同点

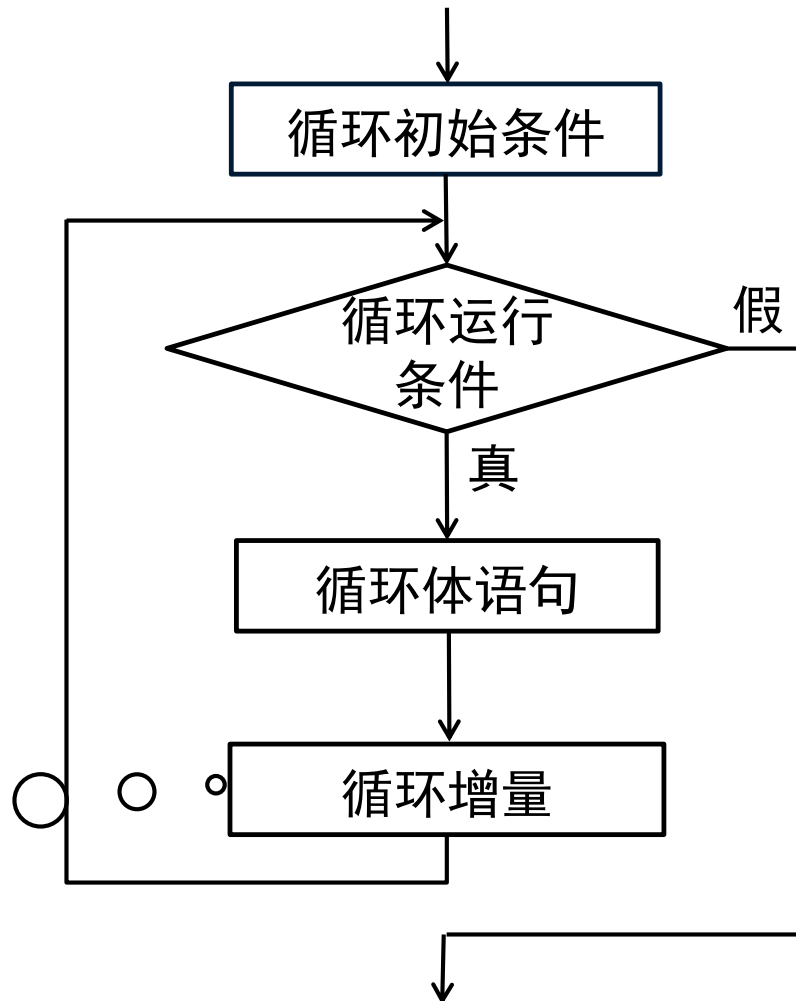
循环三要素

- (1) 循环初始条件：循环开始运行时，循环控制量的初始值；
- (2) 循环运行条件：循环能否继续重复的条件；
- (3) 循环增量：定义循环控制的量，每循环一次后按什么方式变化

循环体

一组被重复执行的语句称之为循环体。

一般循环形式

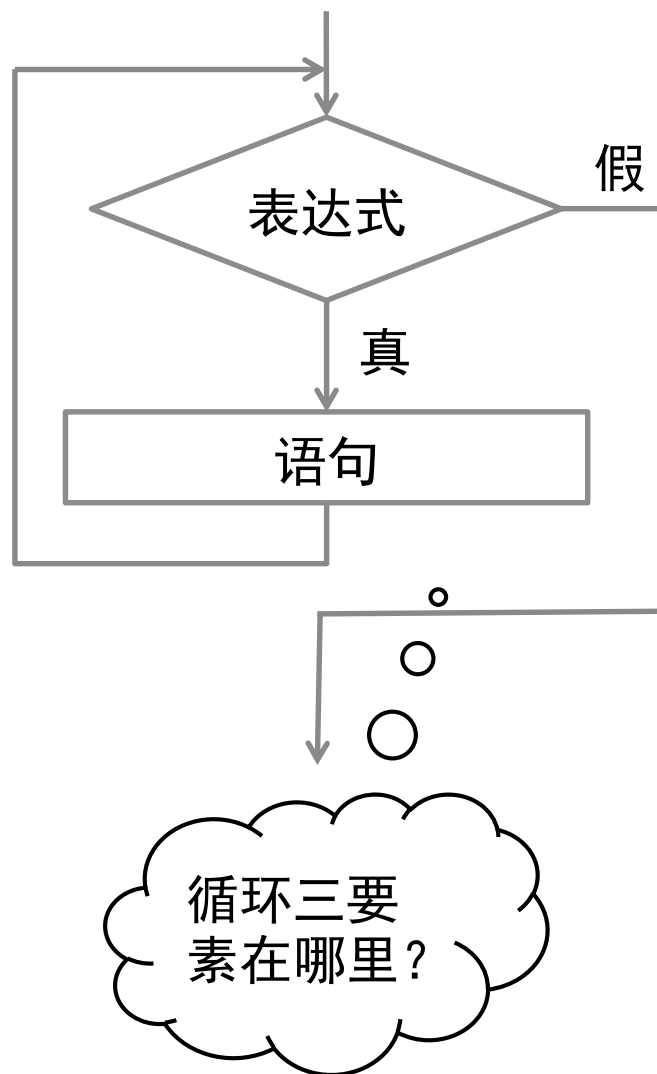


循环增量可
以放在循环
体语句中

当型循环——while语句

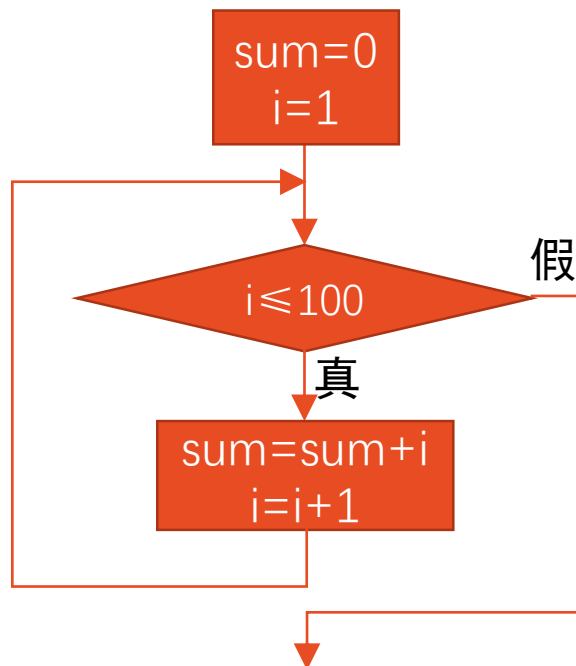
while(表达式) 语句;

只要当循环条件表达式为真(即给定的条件成立), 就执行循环体语句。



while语句实现循环

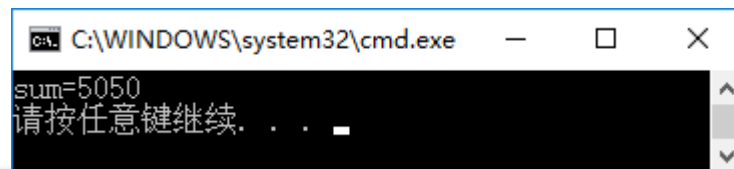
【例7.8】求 $1+2+3+\dots+100$ ，即 $\sum_{n=1}^{100} n$




while语句实现循环

【例7.8】求 $1+2+3+\dots+100$ ，即 $\sum_{n=1}^{100} n$

```
1 #include<stdio.h>
2 int main() {
3     int i=1,sum=0; //定义变量i的初值为1,sum的初值为0
4     while (i<=100) { //当i>100, 条件表达式i<=100的值为假, 不执行循环体
5         //循环体开始
6         sum=sum+i; //第1次累加后, sum的值为1
7         i++; //加完后, i的值加1, 为下次累加做准备
8     } //循环体结束
9     printf("sum=%d\n",sum); //输出1+2+3...+100的累加和
10    return 0;
11 }
```

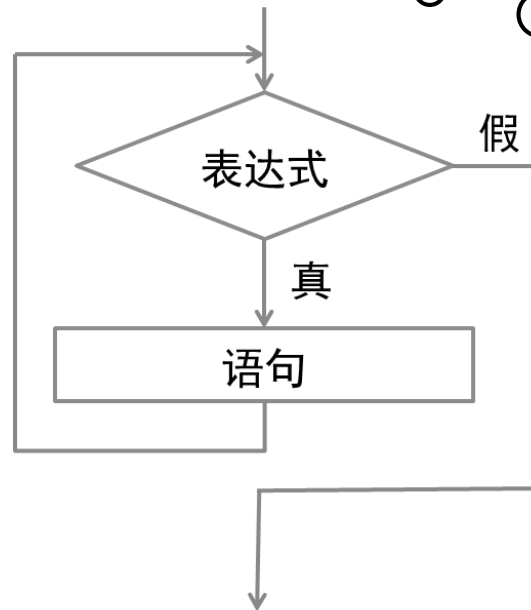


A screenshot of a Windows command prompt window. The title bar shows the path 'C:\WINDOWS\system32\cmd.exe'. The window content displays the output of the program: 'sum=5050' followed by a prompt '请按任意键继续. . . _'.

-  (1) 循环体如果包含一个以上的语句，应该用花括号括起来，作为复合语句出现。
- (2) 不要忽略给i和sum**赋初值**，否则它们的值是不可预测的，结果显然不正确。
- (3) 在循环体中应有使循环趋向于结束的语句。如本例中的“i++；”语句。如果无此语句，则i的值始终不改变，循环永远不结束。

while语句的特例情形

```
while (1) 语句;
```



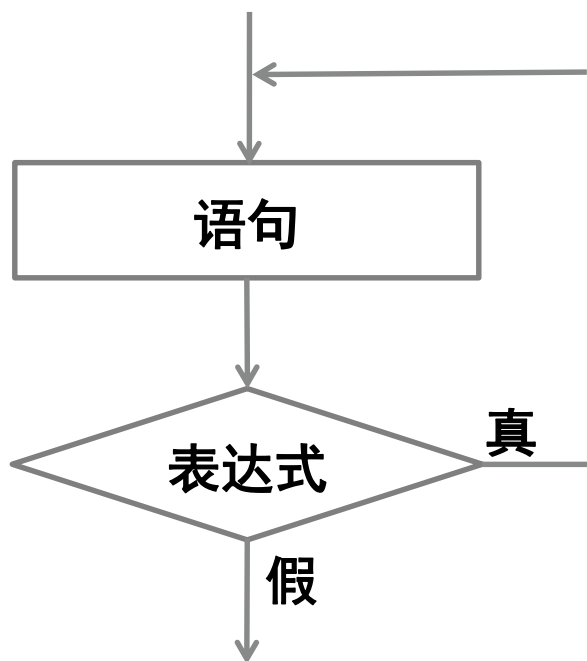
“表达式”为1
时，循环如何运行？

死循环

在编程中，一个无法靠自身的控制终止的循环称为“死循环”或“无限循环”

直到型循环——do while语句

```
do  语句;  
while (表达式);
```



注意

do...while语句的特点是，先无条件地执行循环体，然后判断循环条件是否成立。

直到型循环——do while语句

```
do 语句;  
while (表达式);
```

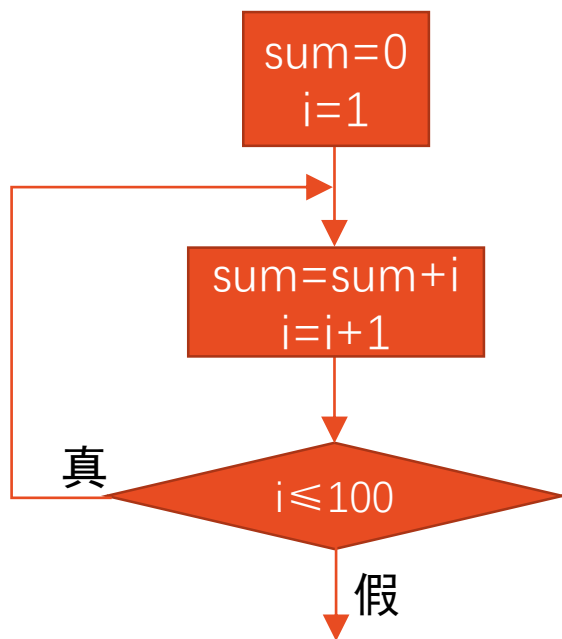
容易与while语句混淆

```
do{  
    语句;  
}while (表达式);
```

建议书写形式

用do...while语句实现循环

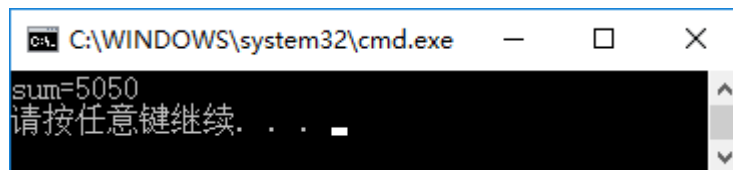
【例7.9】用do...while语句求 $1+2+3+\dots+100$ ，即 $\sum_{n=1}^{100} n$



用do...while语句实现循环

【例7.9】用do...while语句求 $1+2+3+\dots+100$ ，即 $\sum_{n=1}^{100} n$

```
1 #include <stdio.h>
2 int main() {
3     int i=1,sum=0;
4     do {
5         sum=sum+i;
6         i++;
7     } while (i<=100);
8     printf("sum=%d\n",sum);
9     return 0;
10 }
```



A screenshot of a Windows command prompt window. The title bar shows the path 'C:\WINDOWS\system32\cmd.exe'. The window contains the output of the program: 'sum=5050' followed by a prompt '请按任意键继续. . .'. The text is displayed in a black background with white font.

另一种当型循环——for语句

for ([表达式1]; [表达式2]; [表达式3]) 语句;

方括号[]
表示其内
容可缺省

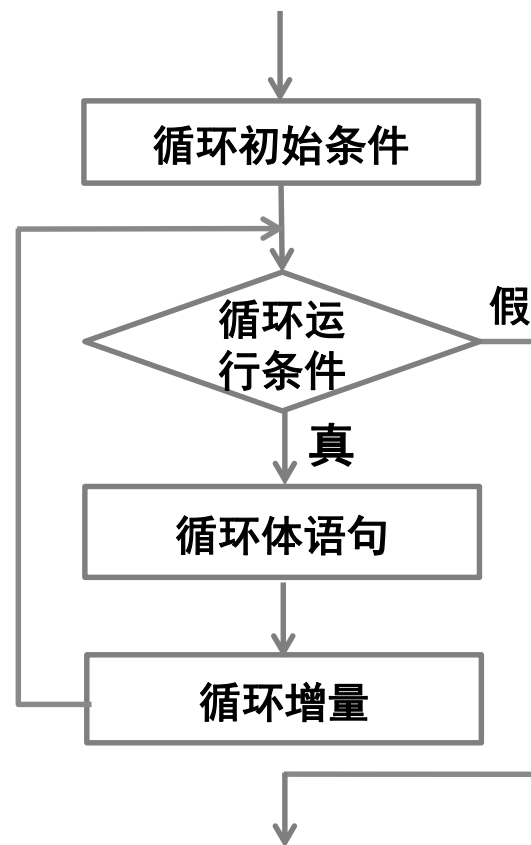
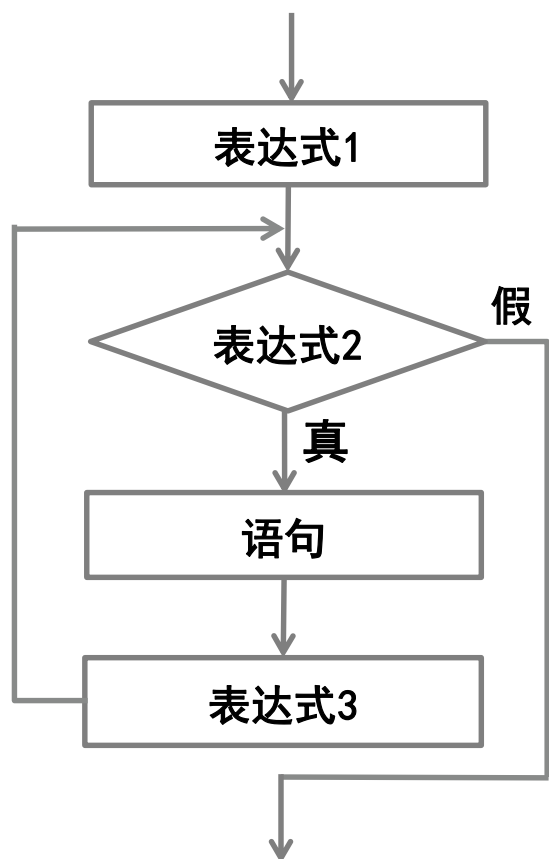
表达式1: 设置初始条件，只执行一次。

表达式2: 是循环条件表达式。在每次执行循环体前先检查此表达式，决定是否继续执行循环。

表达式3: 调整循环变量，它是在执行完循环体后才进行的。

另一种当型循环——for语句

for ([表达式1]; [表达式2]; [表达式3]) 语句;



for ([循环初始条件]; [循环运行条件]; [循环增量]) 语句;

另一种当型循环——for语句

for ([循环初始条件]; [循环运行条件]; [循环增量]) 语句;



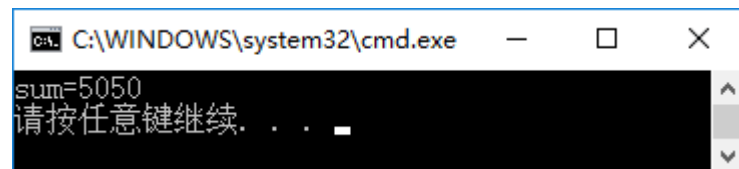
结论

把问题中的循环三要素提炼出来，写出for语句，就是很方便的事情了

用for语句实现循环

【例7.11】 用for语句求 $1+2+3+\dots+100$ ，即 $\sum_{n=1}^{100} n$

```
1 #include <stdio.h>
2
3 int main() {
4     int i, sum = 0;
5     for (i = 1; i <= 100; i++){
6         sum = sum + i;
7     }
8     printf("sum=%d\n", sum);
9     return 0;
10 }
```



A screenshot of a Windows command prompt window titled "C:\WINDOWS\system32\cmd.exe". The window shows the output of the program: "sum=5050" followed by the prompt "请按任意键继续. . .".



三种循环的比较

- (1) 三种循环都可以用来处理同一问题，一般情况下它们可以互相代替。
- (2) 用while和do...while循环时，循环变量初始化的操作应在while和do...while语句之前完成，而for循环可以在表达式1中实现循环变量的初始化。
- (3) 用while和do...while循环时，循环增量的操作应在while和do...while的循环体内完成，而for循环可以在表达式3中实现循环增量。





快速结束循环——break 和 continue 语句

功能

break 和 continue 语句用于改变控制流

语句	出现场合	作用
break	循环语句	跳出循环，使循环提前结束
	switch语句	跳出switch结构
continue	循环语句中	提前结束本次循环

break语句

【例7.12】在100以内的正整数中,求出最大的可被19整除的数。

要求用for语句实现。

```
1 #include <stdio.h>
2
3 int main() {
4     int i;
5     for (i = 100; i > 18; i--) {
6         if (i % 19 == 0)
7             break;
8     }
9     printf("%d\n", i);
10    return 0;
11 }
```

用continue语句提前结束本次循环

```
continue;
```

作用：结束本轮循环，进入下一轮循环的判定

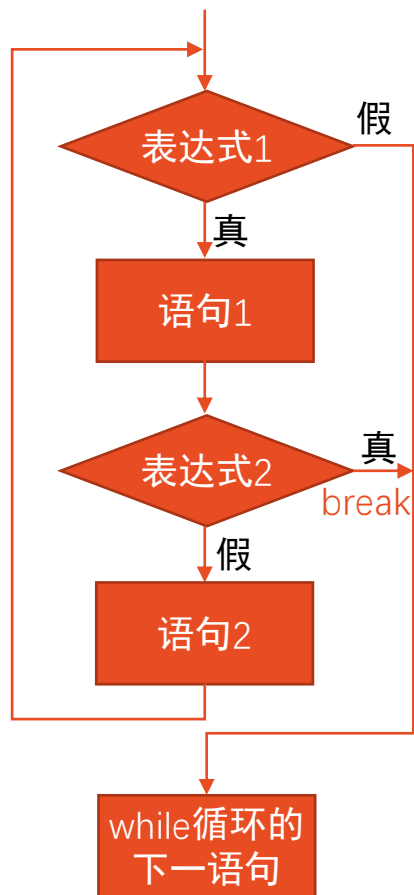
continue语句

【例7.12】输出1到100之间的能被3整除的数，要求一行输出4个数。

```
1 #include <stdio.h>
2 int main() {
3     int i, counter;
4
5     counter = 0;
6     for (i = 1; i <= 100; i++) {
7         if (i % 3 != 0)
8             continue;
9         counter++;
10        printf("%4d", i);
11        if (counter % 4 == 0)
12            printf("\n");
13    }
14    return 1;
15 }
```

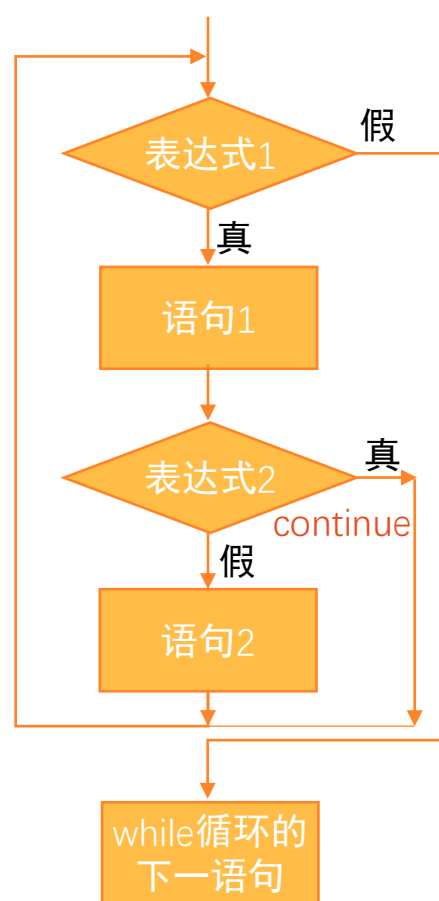
break语句和continue语句的区别

break;



```
while(表达式1)
{
    语句1
    if(表达式2) break;
    语句2
}
```

continue;



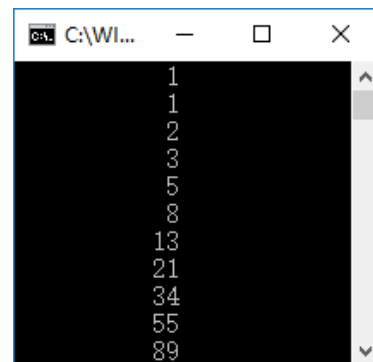
```
while(表达式1)
{
    语句1
    if(表达式2) continue;
    语句2
}
```

continue语句只结束本次循环，而非终止整个循环。
break语句结束整个循环，不再判断执行循环的条件是否成立。

【例7.14】求Fibonacci(斐波那契)数列的前40个数。这个数列有如下特点: 第1, 2两个数为1, 1。从第3个数开始, 该数是其前面两个数之和。即该数列为1,1,2,3,5,8,13,..., 用数学方式表示为:

$$\begin{cases} F_1 = 1 & (n = 1) \\ F_2 = 1 & (n = 2) \\ F_n = F_{n-1} + F_{n-2} & (n \geq 3) \end{cases}$$

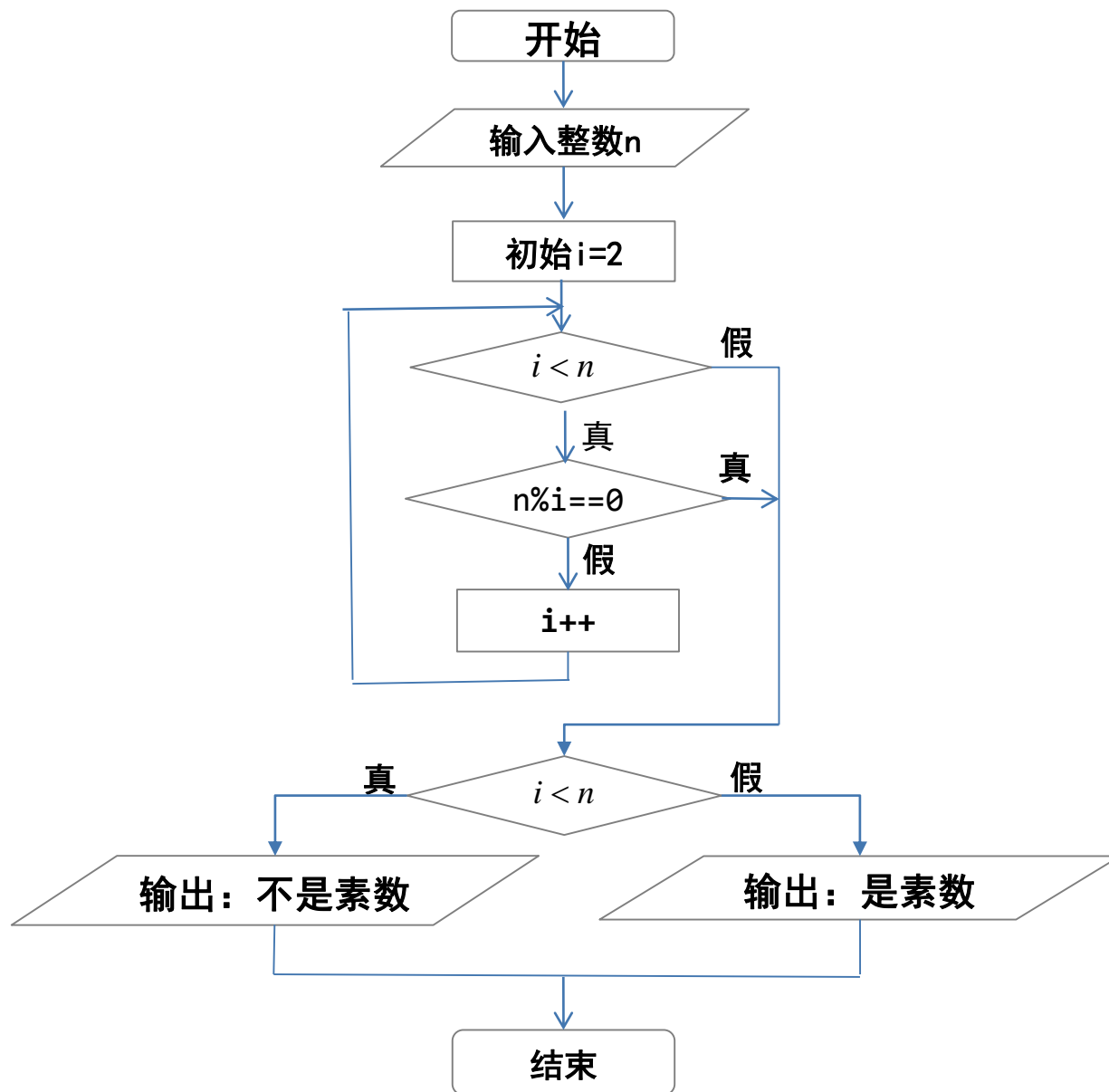
```
1 #include <stdio.h>
2 int main() {
3     int f1=1,f2=1,f3;
4     int i;
5     printf("%12d\n%12d\n",f1,f2);
6     for (i=1; i<=38; i++) {
7         f3=f1+f2;
8         printf("%12d\n",f3);
9         f1=f2;
10        f2=f3;
11    }
12    return 0;
13 }
```



```
C:\WI... 1
          1
          2
          3
          5
          8
         13
         21
         34
         55
         89
```

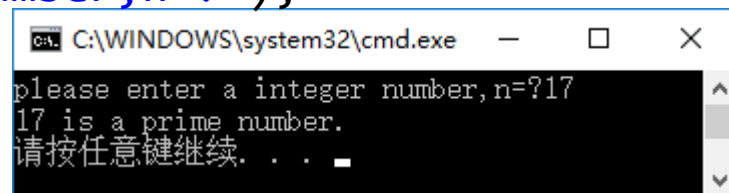
【例7.15】输入一个大于3的整数 n ，判定它是否为素数(prime，又称质数)。

【例7.15】输入一个大于3的整数 n ，判定它是否为素数(prime，又称质数)。



【例7.15】输入一个大于3的整数n，判定它是否为素数(prime，又称质数)。

```
1 #include <stdio.h>
2 int main() {
3     int n,i;
4     printf("please enter a integer number,n=?");
5     scanf("%d",&n);
6     for (i=2; i<n; i++)
7         if (n%i==0) break;
8     if (i<n) printf("%d is not a prime number.\n",n);
9     else printf("%d is a prime number.\n",n);
10    return 0;
11 }
```



若n能被2~(n-1)之间的一个整数整除，则执行break语句，提前结束循环，流程跳转到循环体之外。此时i<n。如果n不能被2~(n-1)之间任何一个的整数整除，则不会执行break语句，循环变量i一直变化到等于n，然后由第1个判断框判定“i<n”条件不成立，从而结束循环。这种正常结束的循环，其循环变量的值必然大于事先指定的循环变量终值(本例中循环变量终值为n-1)。

因此，只要在循环结束后检查循环变量i的值，就能判定循环是提前结束还是正常结束的。从而判定n是否为素数。

【例7.15】输入一个大于3的整数n，判定它是否为素数(prime，又称质数)。

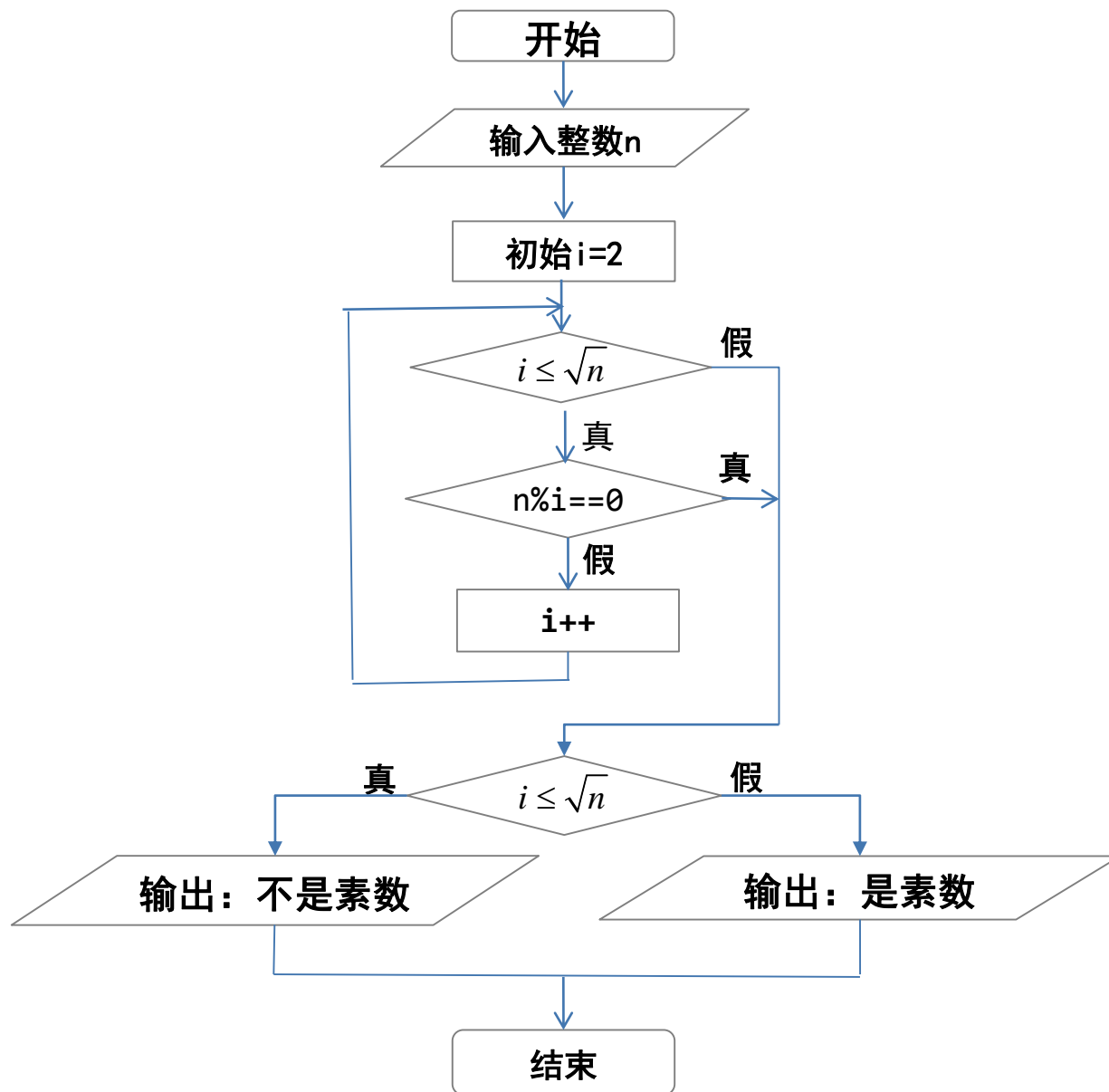
```
1 #include <stdio.h>
2 int main() {
3     int n,i;
4     printf("please enter a integer number,n=?");
5     scanf("%d",&n);
6     for (i=2; i<n; i++)
7         if (n%i==0) break;
8     if (i<n) printf("%d is not a prime number.\n",n);
9     else printf("%d is a prime number.\n",n);
10    return 0;
11 }
```



程序改进：

其实n不必被2~(n-1)范围内的各整数去除，只须将n被2~ \sqrt{n} 之间的整数除即可。因为n的每一对因子，必然有一个小于 \sqrt{n} ，另一个大于 \sqrt{n} 。

【例7.15】输入一个大于3的整数 n ，判定它是否为素数(prime，又称质数)。



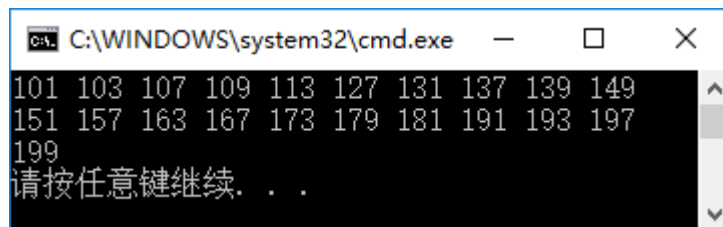
【例7.15】输入一个大于3的整数n，判定它是否为素数(prime，又称质数)。

```
1 #include <stdio.h>
2 #include <math.h>
3 int main() {
4     int n,i,k;
5     printf("please enter a integer number,n=?");
6     scanf("%d",&n);
7     k=sqrt(n);
8     for (i=2; i<=k; i++)
9         if (n%i==0) break;
10    if (i<=k) printf("%d is not a prime number.\n",n);
11    else printf("%d is a prime number.\n",n);
12    return 0;
13 }
```

循环程序举例

【例7.16】求100~200间的全部素数。

```
1 #include<stdio.h>
2 #include<math.h>
3 int main() {
4     int n,k,i,m=0;
5     for (n=101; n<=200; n=n+2) { //n从100变化到200, 对每个奇数n进行判定
6         k=sqrt(n);
7         for (i=2; i<=k; i++)
8             if (n%i==0) break; //如果n被i整除, 终止内循环, 此时i<k+1
9         if (i>=k+1) { //若i>=k+1, 表示n未曾被整除
10             printf("%d ",n); //应确定n是素数
11             m=m+1; //m用来控制换行, 一行内输出10个素数
12         }
13         if (m%10==0) printf("\n"); //m累计到10的倍数, 换行
14     }
15     printf ("\n");
16     return 0;
17 }
```



```
C:\WINDOWS\system32\cmd.exe - □ ×
101 103 107 109 113 127 131 137 139 149
151 157 163 167 173 179 181 191 193 197
199
请按任意键继续. . .
```


循环程序举例

【例7.17】译密码。为使电文保密，往往按一定规律将其转换成密码，收报人再按约定的规律将其译回原文。例如，可以按以下规律将电文变成密码:将字母A变成字母E，a变成e，即变成其后的第4个字母，W变成A，X变成B，Y变成C，Z变成D。字母按照上述规律转换，非字母字符保持原状不变。

高位 低位	000	001	010	011	100	101	110	111
0000	NUL	DLE	SP	0	@	P	`	p
0001	SOH	DC1	!	1	A	Q	a	q
0010	STX	DC2	“	2	B	R	b	r
0011	EXT	DC3	#	3	C	S	c	s
0100	EOT	DC4	\$	4	D	T	d	t
0101	ENQ	NAK	%	5	E	U	e	u
0110	ACK	SYN	&	6	F	V	f	v
0111	BEL	ETB	‘	7	G	W	g	w
1000	BS	CAN	(8	H	X	h	x
1001	HT	EM)	9	I	Y	i	y
1010	LF	SUB	*	:	J	Z	j	z
1011	VT	ESC	+	;	K	[k	{
1100	FF	FS	,	<	L	\	l	
1101	CR	GS	—	=	M]	m	}
1110	SO	RS	.	>	N	^	n	~
1111	SI	US	/	?	O	_	o	DEL

循环程序举例

【例7.17】译密码。为使电文保密，往往按一定规律将其转换成密码，收报人再按约定的规律将其译回原文。例如，可以按以下规律将电文变成密码:将字母A变成字母E，a变成e，即变成其后的第4个字母，W变成A，X变成B，Y变成C，Z变成D。字母按照上述规律转换，非字母字符保持原状不变。

解题思路:

- (1) 判断哪些字符不需要改变，哪些字符需要改变。
- (2) 通过改变字符c的ASCII值的方式将其变为指定的字母。 'A'~'V'或'a'~'v' : $c=c+4$; 'W'~'Z'或'w'~'z' : $c=c-22$ 。

循环程序举例

【例7.17】译密码。为使电文保密，往往按一定规律将其转换成密码，收报人再按约定的规律将其译回原文。例如，可以按以下规律将电文变成密码:将字母A变成字母E，a变成e，即变成其后的第4个字母，W变成A，X变成B，Y变成C，Z变成D。字母按照上述规律转换，非字母字符保持原状不变。

```
1  #include <stdio.h>
2  int main() {
3      char c;
4      c=getchar();//输入一个字符给字符变量c
5      while (c!='\n') { //检查c的值是否为换行符'\n'
6          if ((c>='a' && c<='z') || (c>='A' && c<='Z')) { //c如果是字母
7              if (c>='W' && c<='Z' || c>='w' && c<='z') c=c-22;
8              //如果是26个字母中最后4个字母之一就使c-22
9              else c=c+4; //如果是前面22个字母，c加4，即变成其后第4个字母
10         }
11         printf("%c",c); //输出已改变的字符
12         c=getchar(); //再输入下一个字符给字符变量c
13     }
14     printf("\n");
15     return 0;
16 }
```

程序语句小结——条件语句

单路选择: `if(表达式) 语句A;`

真假

双路选择: `if(表达式) 语句 A;
 else 语句 B;`

真假

多路选择: `if(表达式1) 语句1
 else if(表达式2) 语句2
 else if(表达式3) 语句3
 else 语句4`

真假

多路选择: `switch(表达式){
 case 常量1: 语句1;
 case 常量2: 语句2;
 default: 语句3;
}`

数值

程序语句小结——三种循环方法

while (表达式) 语句;

真假

do{

语句 ;

} while(表达式);

真假

for(表达式1; 表达式2; 表达式3) 语句;

数值

真假

数值

程序语句小结——表达式的种类及计算结果

表达式	结果
算数	数值
关系	真假
逻辑	真假

语句	表达式含义		
if	条件/逻辑		
switch	算术		
while	条件/逻辑		
do-while	条件/逻辑		
for	表达式1	表达式2	表达式3
	赋值	条件/逻辑	算术

程序语句小结——三种循环的比较

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

简单明了，
能用尽量用

```
i=1;  
while (i<=n)  
{ 循环体;  
  i++;  
}
```

循环三要素

初值

条件

增量

```
i=1;  
do  
{ 循环体;  
  i++;  
} while (i<=n);
```