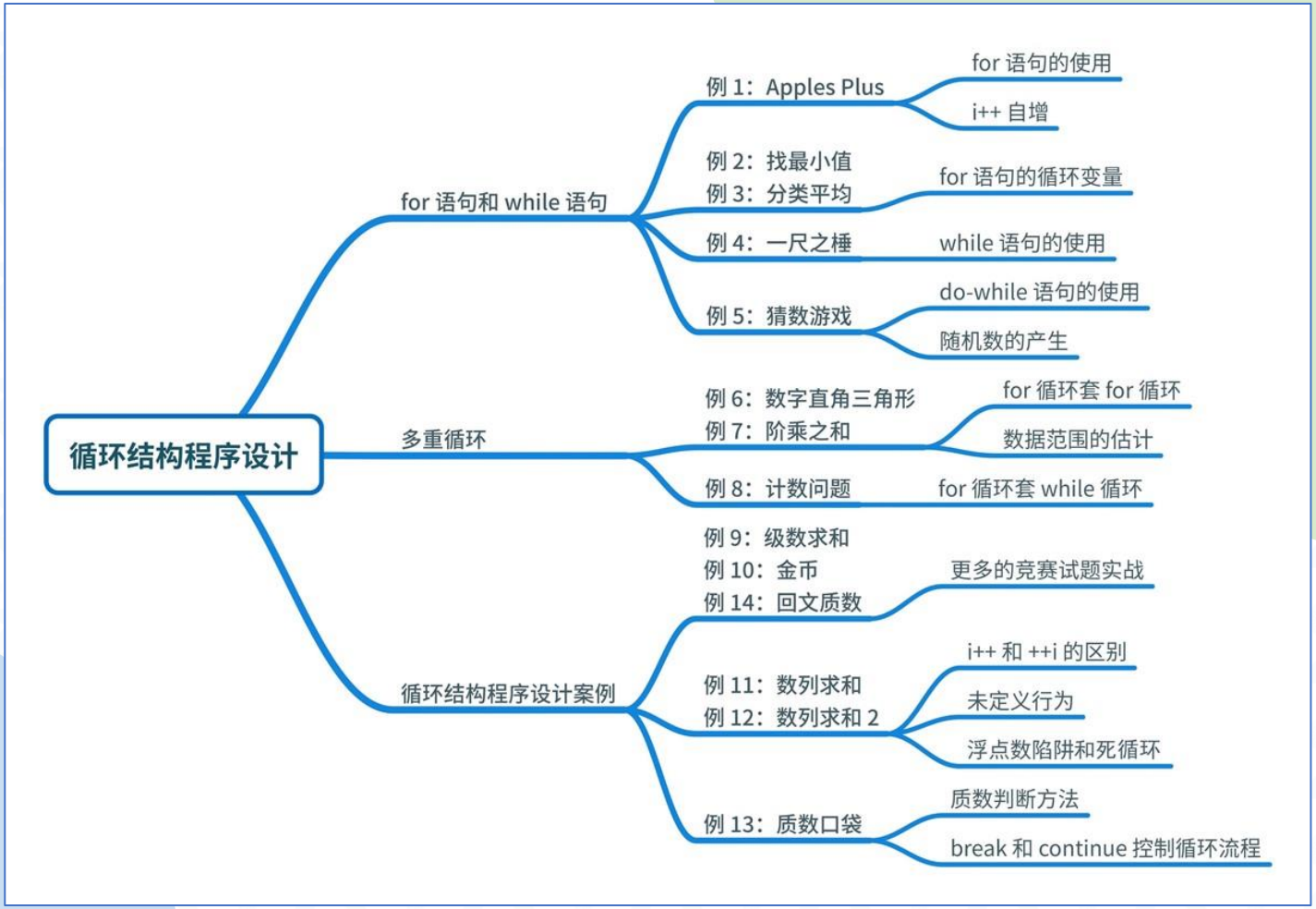
[4] 循环结构程序设计

深入浅出程序设计竞赛
第 1 部分 – 语言入门
V 2021-05

版权声明

- 其它《深基》配套资源、购买本书等请参阅：https://www.luogu.com.cn/blog/kkksc03/IPC-resources
- 如果课件有任何错误，请在这里反馈 https://www.luogu.com.cn/discuss/show/296741

本章知识导图


循环结构程序设计
for 语句和 while 语句
例 1：Apples Plus
for 语句的使用
i++ 自增
例 2：找最小值
例 3：分类平均
for 语句的循环变量
例 4：一尺之棰
while 语句的使用
例 5：猜数游戏
do-while 语句的使用
随机数的产生
多重循环
例 6：数字直角三角形
例 7：阶乘之和
for 循环套 for 循环
数据范围的估计
例 8：计数问题
for 循环套 while 循环
循环结构程序设计案例
例 9：级数求和
例 10：金币
例 14：回文质数
更多的竞赛试题实战
例 11：数列求和
例 12：数列求和 2
i++ 和 ++i 的区别
未定义行为
浮点数陷阱和死循环
例 13：质数口袋
质数判断方法
break 和 continue 控制循环流程

第 4 章 循环结构程序设计

for 语句和 while 语句

多重循环

循环结构程序设计案例

课后习题与实验

for 语句和 while 语句

如果我们让计算机做 100 次运算，真的要写出 100 条语句吗？当然不用，因为重复的工作，只需用循环让计算机重复地去执行就可以了。

请翻至课本 P48

for 语句

用 for 循环进行重复的操作，常用于固定次数的循环。定义一个变量为循环计数器，可用已定义的变量，也可当场定义。

```
for (循环变量初始值; 循环条件; 循环结束操作) {
    循环体执行语句;
}
```

如果符合循环条件，则进入循环。对于每次循环，运行循环体，然后进行每轮循环结束的操作。

```
//使用已经定义好的循环变量
int i;
for (i = 0; i < n; i++) {
    cout << i << endl;
}
```

```
//当场定义的循环变量
for (int i = 0; i < n; ++) {
    cout << i << endl;
}
```

Apples Plus

例 4.1

八尾勇今天又吃掉了 L(1⩽L⩽100) 个苹果。

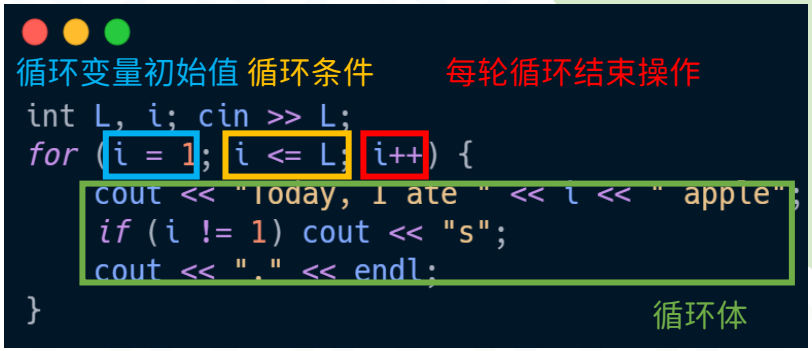
她从第 1 个苹果开始，每吃一个苹果都会在纸上记录下来，在纸上写出一行 Today, I ate x apples.，其中，x 是指吃到第几个苹果；

她吃第一个苹果时，apple 这个单词后面不加 s。

她吃完苹果后，在纸张上记录了什么内容？

Apples Plus

写句子是重复的工作，只需要让计算机重复地去写每一行就可以了。这里使用 for 循环完成重复的工作。



假设读入的 L 是 4，具体的流程如下表所示：

i的值	i<=L成立?	循环体执行	循环结束后
1(初始化)	成立	输出Today, I ate 1 apple.	i++
2	成立	输出Today, I ate 2 apples.	i++
3	成立	输出Today, I ate 3 apples.	i++
4	成立	输出Today, I ate 4 apples.	i++
5	不成立	跳出循环不执行	

找最小值

例 4.2（洛谷 P5718）

给出 $n(n \leq 100)$ 和 n 个整数 a_i $(0 \leq ai \leq 1000)$，求这 n 个整数中最小值是什么。

提示：我们可以使用 min() 函数来获得两个数字中的最小值 。需要使用 algorithm 头文件。

```
minnum = min(minnum, tmp);
maxnum = max(maxnum, tmp);
```

前者 minnum 取 minnum 和 tmp 中 较小 的那个

后者 maxnum 取 maxnum 和 tmp 中 较大 的那个

找最小值

读入数量 n，循环 n 次，每次读入一个数字，共读入 n 个数。

```
int n, tmp, minnum = 100000000;
cin >> n;
for (int i = 0; i < n; i++) {
  cin >> tmp;
  if (tmp<minnum)minnum = tmp;
}
cout << minnum << endl;
```

统计最小值：“打擂台”

定义擂主变量 minnum，挑战者的变量 tmp。

求值过程：

擂主初始值是一个非常大，大于所有输入的数字。这里擂主初始值是 10^8。

每个数字都找擂主打擂。

- 如果新数字小于擂主，那么新数字打擂成功，代替原来擂主；
- 否则打擂失败，擂主不变。

最后留下来的擂主就是最小值。

while 语句

while 循环来进行重复操作，需要确定循环成立的条件。

如果符合循环条件，则继续循环。

如果循环条件不再成立，则停止循环。

while 循环的一般形式如下：

```
while (循环成立条件) {
    循环体
}
```

while 循环常用于没有固定次数的循环。一般来说循环体里有让循环不再成立的语句。

一尺之棰

例 4.4（洛谷 P5720）

《庄子》中说到，“一尺之棰，日取其半，万世不竭”。

第一天有一根长度为 a($a \leqslant 10^9$) 的木棍，从第二天开始，每天都要将这根木棍锯掉一半（每次除 2，向下取整）。

第几天的时候木棍会变为 1?

一尺之棰

for 循环非常适合进行明确知道重复次数的循环。

本例不能很明确知道循环次数，故也可使用 while 循环 解决。

这里使用了逗号表达式，同时增加天数并且减少长度。

当 a>1 时，继续；否则跳出循环。当 a=22，循环流程如下表所示：

```
#include <iostream>
using namespace std;
int main() {
    int a, days = 1;
    cin >> a;
    while (a > 1)
        days++, a /= 2;
    cout << days;
}
```

a的值	a>1是否成立	循环体执行	days	a
22	成立	执行	days++（变为2）	a/=2（变为11）
11	成立	执行	days++（变为3）	a/=2（变为5）
5	成立	执行	days++（变为4）	a/=2（变为2）
2	成立	执行	days++（变为5）	a/=2（变为1）
1	不成立	跳出循环不执行		

do-while 语句

do while 循环，和前面介绍过的 while 循环不同。

无论怎么样 do-while 循环都会直接执行循环体一次，等循环体运行结束后再验证循环成立条件。

do while 的一般形式如下：

```
do {
    循环体
} while(循环成立条件);
```

如果成立就会重新开始循环，否则就退出循环。

do while 循环可以在转化为 while 循环

随机数

如何让计算机产生一个随机数呢?

可以使用 rand() 函数来产生一个 0 到 RAND_MAX 的数字，其中 RAND_MAX 是一个常量，其值与编译器和系统有关，而且别忘了加上头文件cstdlib。

一般来说Windows下其值为32767，而Linux下其值是int的最大值。

可以用 rand()%a 来产生一个 0 到 a-1 的随机数。如果想产生一个 a 到 b 的随机数可以使用 rand()%(b-a+1)+a。

随机数

如果直接这么写，无论运行多少次，输出的结果都是一样的，随机数完全不随机。解决方案是在生成随机数前加上一句srand(time(0))，同时加上头文件ctime。

这个代码是输出一个 1 到 100 的随机数；课本 P52 提供了一个和随机数有关的猜数游戏作为扩展阅读。

```
#include <iostream>
#include <cstdlib>
#include <ctime>
using namespace std;
int main() {
    int ans, guess;
    srand(time(0));
    ans = rand() % 100 + 1;
    cout << ans << endl ;
    return 0;
}
```

多重循环

正如分支条件语句也能嵌套分支条件语句一样，循环语句也能互相嵌套。如果一些需要循环的“大操作”中有数个重复的小操作，就可进行循环嵌套。

请翻至课本 P53

多重循环

循环语句也能互相嵌套。如果需要循环的“大操作”中有好几个重复的小操作，就可以进行循环嵌套。

如果超过一个语句，或者需要调整层级，则必须要大括号，建议全部加上括号。

注意内层循环变量名不能和外层循环变量名冲突。

```
for (;;) {
    for (;;) {
        for (;;) {
            // 内层循环体
        }
    }
    for (;;)
        while (...) {
            // 内层循环体
        }
}
```

数字直角三角形

例 4.6（洛谷 P5721）

给出 n(1≤n≤13)，请输出一个直角边长度是 n 的数字直角三角形。例如，当 n=5 时，应该输出：

```
0102030405
06070809
101112
1314
15
```

提示：可以使用一重循环，在合适的地方换行！

不过，有没有更好思考的方法呢？

数字直角三角形

我们可以把这个任务分成两个层级

1. 大任务：输出每一行。需要一个外层循环，其循环体就是输出一行的内容。循环变量从 1 到 n（也可以 0 到 n-1）。

2. 小任务：输出一行中的每一个数字。需要一个内层循环，用于输出一行中的每一个数字。共需要打出 n 行，所当处理第 i 行时，需要输出 n-i+1 个数字。

定义变量 cnt 来记录输出到那个数字

```
int cnt = 0, n;
scanf("%d", &n);
for(int i = 1; i <= n; i++) {
    for(int j = 1; j <= n - i + 1; j++)
        printf("%02d", ++cnt);
    printf("\n");
}
```

数字直角三角形

循环层级	i 的值	i<=n?	j 的值	j<n-i+1?	循环体执行	循环结束后
外层	**1（初始）**	**成立**			**进入内层循环**	
内层			1（初始）	成立	cnt=1；输出 01	j++（变为 2）
内层			2	成立	cnt=2；输出 02	j++（变为 3）
内层			3	成立	cnt=3；输出 03	j++（变为 4）
内层			4	不成立	退出内层循环	
外层	**1**				**输出换行**	**i++（变为 2）**
外层	**2**	**成立**			**进入内层循环**	
内层			1（初始）	成立	cnt=4；输出 04	j++（变为 2）
内层			2	成立	cnt=5；输出 05	j++（变为 3）
内层			3	不成立	退出内层循环	
外层	**2**				**输出换行**	**i++（变为 3）**
外层	**3**	**成立**			**进入内层循环**	
内层			1（初始）	成立	cnt=6；输出 06	j++（变为 2）
内层			2	不成立	退出内层循环	
外层	**3**				**输出换行**	**i++（变为 4）**
外层	**4**	**不成立**			**退出外层循环**	

这里是 n=3 的例子

010203

010203
0405

010203
0405
06

阶乘之和

例 4.7（洛谷 P1009，有改动）

计算 $S = 1! + 2! + 3! + \cdots + n!\ (n \leq 20)$ 的值，其中 i! 是指 i 的阶乘，即 $i! = 1 \times 2 \times \cdots \times i$。

分析：按计算器可发现，20 的阶乘大约 2.4×10^{18}，所以可以使用 long long 类型存下。

long long 是一种整数类型，可以放下 9×10^{18}的数字，但是比 int 又大又慢。

阶乘之和

外层循环：循环变量 i 从 1 开始，到 n 结束，每次都加上 i 的阶乘，最后统计即可。

内层循环：求 i 的阶乘则需要另一层循环，循环变量 j 从 1 开始，到 i 结束，每次都乘上 j 。

```
long long n, ans = 0;
cin >> n;
for (int i = 1; i <= n; i++) { // 外层循环
    long long factor = 1;
    for (int j = 1; j <= i; j++) // 内层循环
        factor *= j;
    ans += factor;
}
cout << ans << endl;
```

计数问题

例 4.8（洛谷 P1980，NOIP 普及组 2013）

试计算在区间 1 到 n($n \leqslant 10^6$) 的所有整数中，数字 x($0 \leqslant x \leqslant 9$) 共出现了多少次？

当 n 等于 12 时，数字 1 出现了 5 次（1,10,11,12）

提示：如何依次判断一个数字（假设为 tmp）的每一位呢？

计算 tmp%10，得到它的个位数字，判断是不是等于 x，如果是则计数器加 1；然后 tmp/=10，把原来的个位数去掉了；

重复刚才的流程，每次都统计个位数，直到 tmp 为 0 为止。

计数问题

内层循环：对于整数 tmp，获得它的每一位并进行统计。

外层循环：从 1 到 n 枚举 i，然后将 i 的值赋给 tmp 后再统计 tmp 枚举每一位；不能直接处理 i，否则 i 就会被改变，这是外层循环。本例是 for 循环套用 while 循环的一个例子。

```
int n, x, ans = 0;
cin >> n >> x;
for (int i = 1; i <= n; i++) { // 外层循环
    int tmp = i, num;
    while (tmp != 0) { // 内层循环
        num = tmp % 10;
        if (num == x)
            ans++;
        tmp /= 10;
    }
}
cout << ans;
```

循环结构程序设计案例

本节将会介绍更多的关于循环的用法，题目会有一点复杂，前面的知识都会涉及，准备好了吗？

请翻至课本 P56

级数求和

例 4.9（洛谷 P1035，NOIP 普及组 2002）

已知：$S_n=1+1/2+1/3+\cdots+1/n$。显然对于任意一个整数 K，当 n 足够大的时候，S_n 大于 K。现给出一个整数 K（$1\leqslant k\leqslant 15$），要求计算出一个最小的 n，使得$S_n>K$。

提示：我们可以用循环模拟这个过程。

```
int k, ans = 0;
cin >> k;
for (double Sn = 0; Sn <= k; Sn += 1.0 / ans, ans++);
cout << ans;
```

注意这个循环语句，直接以一个分号结尾，它没有循环体！

在 for 循环的第三项每轮结束后的操作就是增加答案和累加倒数操作，使用了逗号表达式。

级数求和

如果 k 可以到 21 呢？程序设计中，每一道题都有时间限制，程序的运行时间需要在这个时间限制内出结果。

当 k 较大时，我们需要打表，即预处理结果，输入之后直接输出。

```
if (k == 21) {
    cout << 740461601;
    return 0;
}
```

这说明：当循环次数较多，运算速度就可能很慢。

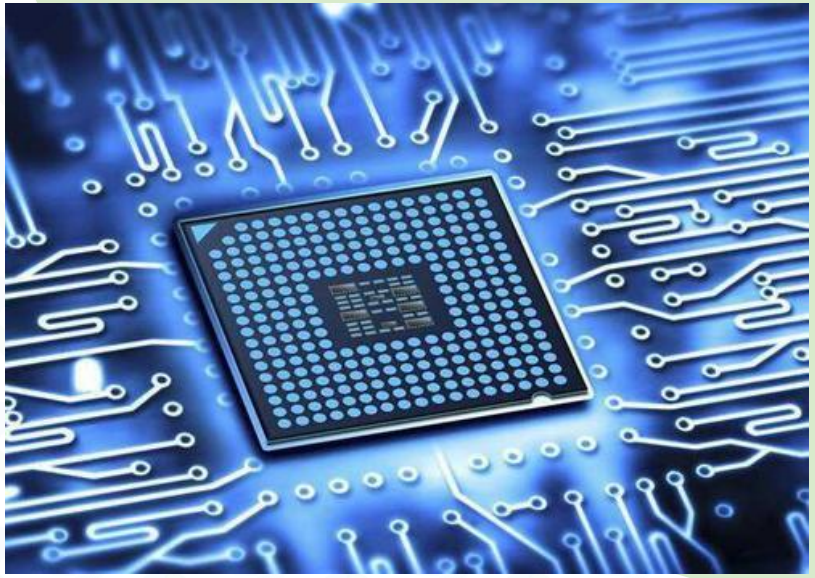
需要想办法提升程序运行的效率，减少循环次数。

计算机运行时间

计算机运行速度很快，但也是有上限的。

一般来说，现在的 CPU 可每秒钟运行 10^8（一亿）次循环。

如果循环次数过多，就会超出时间限制，简称 TLE。



死循环

当一个循环一直无法跳出，一直不断地执行，就会导致死循环。

在比赛或者评测网站中，死循环会导致程序运行之间超时，简称 TLE。

可能的原因：错误的循环终止条件，或者没有正确的修改循环判断变量。

```
while (1) {
    //做一些事情
}
for (i = 0; i < 5; i--) {
    //做一些事情
}
for (i = 0; i < 5; i++) {
    //做一些事情
    i = 1;
}
```

金币

例 4.10（洛谷 P2669，NOIP2015 普及组）

第一天，骑士收到一枚金币；
之后两天（第二天和第三天），每天收到两枚金币；
之后三天（第四、五、六天），每天收到三枚金币；
之后四天（第七、八、九、十天），每天收到四枚金币……；

这种工资发放模式会一直这样延续下去：当连续N天每天收到N枚金币后，骑士会在之后的连续N+1天里，每天收到N+1枚金币。

请计算在前 K 天里，骑士一共获得了多少金币。

例如：前6天中，骑士第 1 天收到 1 枚金币；第 2/3 天，每天收到 2 枚金币；第 4/5/6 天，每天收到 3 枚金币。因此一共收到 1+2+2+3+3+3=14 枚金币。

金币

解法很多。其中的一种简单的思路是枚举“每轮”金币发放。

其中第 i 轮发金币时，每天发 i 个金币，连发 i 天。

- 大任务：发到第 i 轮时；
- 小任务：这轮第 j 天（共 i 天）发 i 个。

每天发金币时记录下骑士收到的金币并累加，同时天数加 1。

等到加到足够天数时输出答案即可。

```cpp
#include<iostream>
using namespace std;
int main() {
  int k, coin = 0, day = 0;
  cin >> k;
  for (int i = 1;; i++)
    for (int j = 1; j <= i; j++) {
      coin += i; day++;
      if (day == k) {
        cout << coin << endl;
        return 0;
      }
    }
  return 0;
}
```

死循环？！

达到目标，可以直接提前退出！

i++与++i

i++ 表示先使用 i 的值运算，然后使 i 增大 1

++i 表示先使 i 增大 1，然后使用增大后的 i 值运算

例如

- i=1; j=i++; 等同于 j=i; i+=1; 此时 i=2 j=1。
- i=1; j=++i; 等同于 i+=1; j=i; 此时 i=2 j=2。

判断练习

请指出下列语句执行后，i 和 j 的值分别是多少？

① i=3; i++; j=i;

② i=3; j=i++;

③ i=3; j=3+i++;

④ i=3; j=++i;

i++与++i

判断练习 – 答案

请指出下列语句执行后，i 和 j 的值分别是多少？

① i=3; i++; j=i;　　i 为 4，j 为 4

② i=3; j=i++;　　i 为 4，j 为 3

③ i=3; j=3+i++;　　i 为 4，j 为 6

④ i=3; j=++i;　　i 为 4，j 为 4

数列求和

例 4.11 （洛谷 P5722）

计算 1+2+3+…+(n-1)+n 的值，其中 n 不大于 100。不得用公式。

解法 1：使用 for 循环直接累加计算。

解法 2：使用 while 循环 n 次。每次循环中 i 增加 1，s 增加 i。

```
int s = 0, i = 0, n;
cin >> n;
```

```
while (n--) s += ++i;
cout << s;
```

- while(n--) 先判断 n 是否非零，再将 n 减去 1。
  while(--n) 先将 n 减 1，然后再判断是否非零。

s += ++i 先将 i 加 1，再加到 s 中。

注意：避免 ++i + i++ 这类可读性极差的代码。

判断质数

因数：一个正整数，能被若干个数整除，那么这若干个数就是这个数的因数。

质数：只有 1 和它本身两个因数的自然数。1 不是质数也不是合数

合数：因数个数大于 2 的自然数。

判断一个数 n 是否为质数：i 从 2 循环到 n-1，若存在 i 能整除 n，则 n 不为质数。

有没有更快的方法？

判断质数

设 $a \times b = n, a \leqslant b$，可知 $a \times a \leqslant n$，所以 $a \leqslant \sqrt{n}$，$b \geqslant \sqrt{n}$。

若不存在符合条件的 a 能整除 n，则必然不存在对应的 b 能整除 n。

比较以下两种写法在时间效率上的区别

```
for (int i = 2; i * i <= n; i++)
for (int i = 2; i <= sqrt(n); i++)
```

乘法速度快于除法，而 sqrt 速度也很慢。

在程序的任意地方加上 return 0; 即可立刻退出程序。

break 和 continue

遇到不想让循环执行完毕的情况下，可以终止这个循环，或者跳过本次循环执行语句。

break 语句：终止本层循环。只能跳出一层循环；

continue 语句：跳过本次循环，直接开始下一次循环。

break 和 continue

判断练习

下列代码，程序会输出什么？

break 语句：终止本层循环。只能跳出一层循环

continue 语句：跳过本次循环，直接开始下一次循环。

```
for (int i = 1; i <= 10; i++) {
 //①
    if (i % 2 == 0)
        continue;
    cout << i << " ";
    if (i >= 5)
        break;
} //②
```

break 和 continue

判断练习 – 答案

下列代码，程序会输出 1 3 5。

break 语句：终止本层循环。跳到位置 ①

continue 语句：跳过本次循环。跳到位置 ②。

```
for (int i = 1; i <= 10; i++) {
 //①
    if (i % 2 == 0)
        continue;
    cout << i << " ";
    if (i >= 5)
        break;
} //②
```

质数口袋

例 4.13（洛谷 P5823）

小A 有一个装质数的口袋。

他从 2 开始，依次判断各个自然数是不是质数，如果是质数就会把这个数字装入口袋。

口袋的负载量就是口袋里的所有数字之和。但口袋的承重量有限，不能装得下总和超过 L(1⩽L⩽100000) 的质数。

给出 L，请问口袋里能装下几个质数？将这些质数从小往大输出，然后输出最多能装下的质数个数，每行输出一个数字。

质数口袋

例 4.13（洛谷 P5823）

装尽量小的质数可以使装的质数的个数最多。

可以从小到大枚举每个数，再判断是否质数，用之前所学方法。

如果当前质数装不下，那么接下来的质数一定装不下，可以直接退出循环。

```
int L, load = 0, ans = 0;
cin >> L;
for (int i = 2;; i++) {
    int is_prime = 1;
    for (int j = 2; j * j <= i;j++)
        if (i % j == 0) {
            is_prime = 0;
            break;
        }
    if (!is_prime) continue;
    if (i + load > L) break;
    cout << i << endl;
    ans++; load += i;
}
cout << ans;
```

一尺之棰

判断练习

如果初始 cnt=0，执行完 for (int i = 1; i <= 100; i += 6)++cnt; 后 cnt 的值是多少？

如果初始 cnt=0,i=1，执行完 while (i <= 100)++cnt, i += 6; 后 cnt 的值是多少？

上面两个结果是否一样？为什么？

一尺之棰

判断练习 – 答案

如果初始 cnt=0 ， 执行完 for (int i = 1; i <= 100; i += 6)++cnt; 后 cnt 的值是 17。

如果初始 cnt=0,i=1， 执行完 while (i <= 100)++cnt, i += 6; 后 cnt 的值是 17。

上面两个结果一样。

计算机运行时间

判断练习

估计以下程序是否会超时，即循环次数超过 10^8。

```
for(i=0;i<20000000;i++)
    ans++;


for(i=0;i<15000000;i+=2)
    ans++;


for(i=0;i<20000;i++)
    ans++;
```

```
for(i=0;i<20000;i++)
    for(j=0;j<20000;j++)
        ans++;


for(i=0;i<20000;i++)
    for(j=0;j<20000;j+=i)
        ans++;
```

计算机运行时间

判断练习 – 答案

估计以下程序是否会超时，即循环次数超过 10^8 。

```
for(i=0;i<20000000;i++)
    ans++;
//超时

for(i=0;i<15000000;i+=2)
    ans++;
//超时

for(i=0;i<20000;i++)
    ans++;
//不超时
```

```
for(i=0;i<20000;i++)
    for(j=0;j<20000;j++)
        ans++;
//超时

for(i=0;i<20000;i++)
    for(j=0;j<20000;j+=i)
        ans++;
//不超时
```

课后习题与实验

学而时习之，不亦说乎。学而不思则罔，思而不学则殆。——孔子

请翻至课本 P64

总结

for 循环

- for 循环格式 / 常用于有固定次数的循环

while 循环

- 循环体运行前验证循环成立条件

do-while 语句

- 循环体运行结束后再验证循环成立条件
- i++ 和 ++i 是有区别的

随机数

- 随机生成一个范围的随机数
- 生成随机数前加上srand(time(0))

总结

数字按位拆分

n % 10 得到 n 的最后一位（即个位）
n / 10 删去 n 的最后一位

多重循环

将任务分解为多个需要循环的子任务
每个子任务中又需要进行循环

计算机运行时间

计算机很快，但也有极限，超过一亿次循环很难一秒钟跑完

判断质数 i 从 2 枚举到 $\sqrt{n}$ 判断是否能整除 n

break 和 continue 跳出这个循环，以及直接开始下一轮循环

课后拓展

习题 4.1：如果想求一个数列的最大值，还要求出是第几个数字是最大的，该如何实现呢？

习题 4.2：小玉在游泳（洛谷 P1423）

小玉在游泳，第一步能游 2 米，可是随着越来越累，力气越来越小，她接下来的每一步都只能游出上一步距离的98%。现在小玉想知道，如果要游到不小于距离 x(x≤100) 米的地方，她需要游多少步呢。

习题 4.3 数字反转（洛谷 P1307，NOIP2011 普及组）

给定一个整数（其绝对值不大于 10^9），请将该数各个位上数字反转得到一个新数。

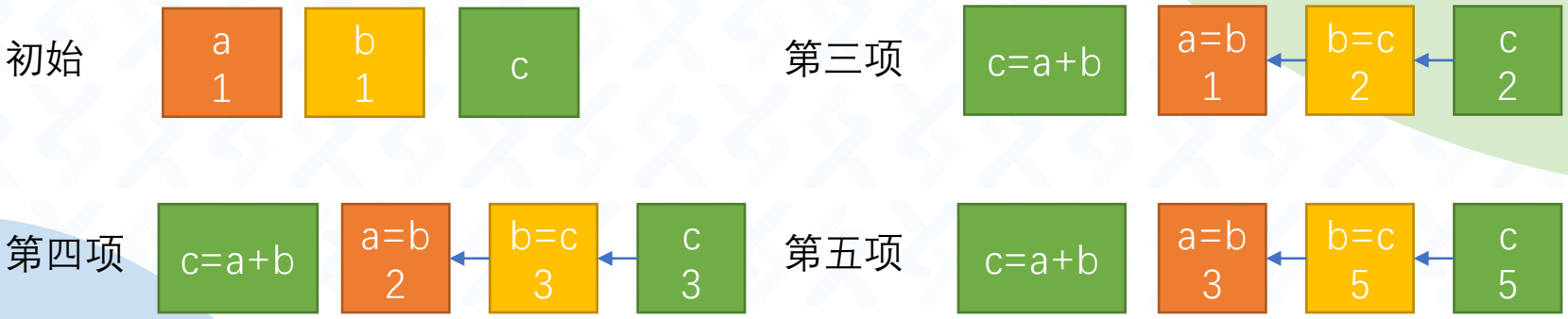
新数也应满足整数的常见形式，即除非给定原数为零，否则反转后的新数的最高位数字不应为零。比如输入是-380，输出-83。

课后拓展

习题 4.4 斐波那契数列（洛谷 P1720 改编）

观察下面的数列：1 1 2 3 5 8 13 21 34 55 89 144 233...

除了最开始的两个数字，后面的数字都是前面两个数字的和，这就是斐波那契数列。请输出第 n(3≤n≤30) 项斐波那契数列。



课后拓展

习题 4.5 求极差（洛谷 P5724）

给出 $n(n \leqslant 100)$ 和 n 个整数 a_i $(0 \leqslant a_i \leqslant 1000)$，求这 n 个整数中的极差是什么。

极差的意思是一组数中的最大值减去最小值的差。

习题 4.6 最长连号（洛谷 P1420）

输入 $n(n \leqslant 10^4)$ 个不超过 10^9 的正整数，要求输出最长的连续自然数子序列的长度。

课后拓展

习 4.10 Davor（洛谷 P4956，COCI2017）

Davor 需要筹集 n 元钱。他打算在每个星期一筹集 x 元，星期二筹集 x+k 元，……，星期日筹集 x+6k 元，并在 52 个星期时筹集完。其中 x,k 为正整数，并且满足 $1 \leqslant x \leqslant 100$。

现在请你帮忙计算 x,k 为多少时，能刚好筹集 n 元。

如果有多个答案，输出 x 尽可能大，k 尽可能小。

参考阅读材料

以下的内容限于课件篇幅未能详细阐述。如果学有余力，可自行翻阅课本作为扩展学习。

- P50 例 4.3：活用 for 循环计算平均值
- P52 例 4.5：猜数游戏；随机数的详细用法和注意事项
- P59 例 4.12：死循环；浮点误差的提示
- P62 例 4.14：比较复杂的枚举题，以及数学证明
- 习题 4.7、4.8、4.9、4.11。