

计算机程序设计

Computer Programming



控制结构



主讲：吴锋

目录

CONTENTS

C语言语句

选择结构与循环控制结构

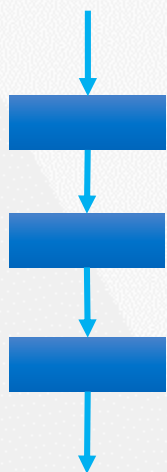
- if, switch

控制转向语句

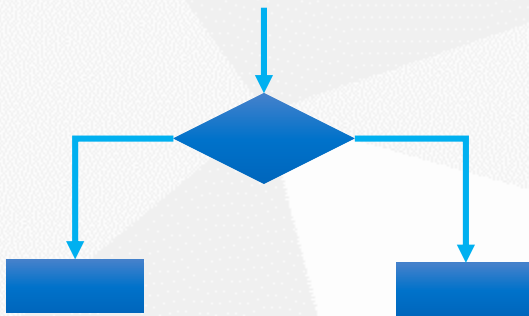
- while, do-while, for
- break, continue, goto

◎ C语言的程序结构

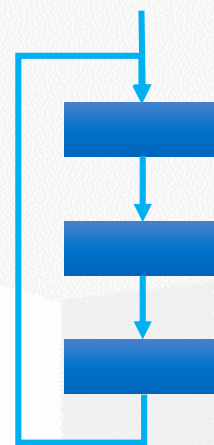
- C是结构化语言，支持三种基本结构



顺序结构

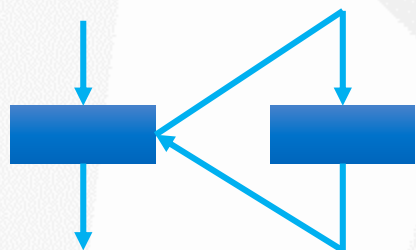


选择（分支）结构



循环结构

- 函数调用



◎ C语言的语句

- 数据声明，编译器一般将它与其它语句区别处理
类型_变量名表；
- 表达式语句
表达式；（将赋值、显示等功能视为“副作用”）
- 函数调用语句
函数名(实参表)；
- 控制语句
选择语句if、switch
循环语句do while、while、for
跳转语句break、continue、return、goto
- 复合语句
{ 一条或多条语句；}
- 空语句
；



◎ 逻辑表达式

- 选择语句/循环语句都需要检测是否满足选择或循环条件

- 关系运算符:

< <= > >= 优先级6

== != 优先级7

- 说明

- 左结合
- 关系表达式的值总是int类型，真为1，假为0
- 注意两边操作数类型不同时会自动转换

char, short → int → unsigned → long → double

↑
float

- 例如 $-1 < 0u$ ，结果不是1，而是0

```
int a=3, b=4, c=5, k;
```

```
a<=b      //值为1
```

```
a+b>c-b    //等价于(a+b)>(c-b), 值为1
```

```
k=3<=c    //等价于k=(3<=c), 值为1
```

```
a==3<=c    //等价于(a)==(3<=c), 值为0
```

```
a<b<c      //等价于(a<b)<c, 值为1
```

```
a==a==a    //等价于(a==a)==a, 值为0
```

```
(i>=j)+(i==j) //根据i, j关系, 得0, 1, 2
```

◎ 逻辑表达式

• 逻辑运算符:

! (逻辑非, 优先级2, 右结合)

&& (逻辑与, 优先级11, 左结合)

|| (逻辑或, 优先级12, 左结合)

◦ 操作数是逻辑值“真”(非0)和“假”(0), 类型int

```
int a=3, b=3, y=2000;
```

```
char c='A';
```

```
double x=0.0, z=7.2, f;
```

```
!(a-b) //值为1
```

```
c&&(a-b)||f==3.2 //等价于(c&&(a-b))||f==3.2, 值为1
```

```
!x*!!z //等价于(!x)*(!(!z)), 值为1
```

```
y%4==0&&y%100||!(y%100)&&!(y%400) //值为1
```

```
(y%100!=0)&&!(y%4) || (y%100==0)&&!(y%400)
```



◎ 逻辑表达式

• 短路运算

- `&&`和`||`都执行短路运算：先计算左操作数的值，若该值已可确定最终结果，则不计算右操作数的值
- 常用于非法或越界运算的保护，例如
 - `(i!=0) && (j/i>0)`
 - `(i<N) && (a[i]>k)`
- 特别注意：右表达式中的副作用未必会发生
 - `if (((a=getchar())=='O') && ((b=getchar())=='K'))`

◎ if 语句

if(表达式) 语句1

if(表达式) 语句1 else 语句2

- 判断给定的条件是否满足，根据结果（真或假）决定执行某个分支操作
- 例： `if (a>0) printf("a>0\n");`
- 例： `if (a>0) printf("a>0\n");`
`else printf("a<=0\n");`
- 注意事项
 - if 括号内的表达式可以是任何符合语法的表达式，if 仅判断表达式的值是否为0，非0认为是逻辑真，为0则认为结果是逻辑假
 - 语句1、语句2可以是简单语句，也可以是复合语句
例 `{ int a=1,b=2,c; c=a+b; printf("c=%d", c); }`
 - if 语句中if和else是一个整体，else 不能单独使用，但可以缺省



◎ if 语句（举例）

- 比较输入的两个数的大小，将大数赋给变量x，小数赋给变量y

```
#include <stdio.h>
void main() {
    int a,b,x,y;
    scanf("%d%d", &a, &b);
    if(a>b){
        x=a;
        y=b;
    }
    else{
        x=b;
        y=a;
    }
    printf("a=%d, b=%d\n x=%d, y=%d\n", a, b, x, y);
}
```

用{}括起来的复合语句

{}的书写没有定式，但应统一

K&R风格

```
if (...) {
    ...
}
```

Allman风格

```
if (...)
{
    ...
}
```

Whitesmiths风格

```
if (...)
{
    ...
}
```

GNU风格

```
if (...)
{
    ...
}
```

◎ 嵌套 if 语句

```
if(表达式1)  
    if(表达式2) 语句1
```

```
if(表达式1)  
    if(表达式2) 语句1  
else 语句2
```

```
if(表达式1)  
    if(表达式2) 语句1  
else 语句2
```



```
if(表达式1)  
    if(表达式2) 语句1  
    else 语句2  
else  
    if(表达式3) 语句3  
    else 语句4
```

```
if(表达式1)  
    { if(表达式2) 语句1 }  
else 语句2
```

else总是与它前面的最近的未配对的if配对，与缩进无关

◎ 嵌套 if 语句（举例）

- 输入三个数a、b、c，输出其中最大者

- 算法描述（自然语言）

如果 $a > b$ 则
 如果 $a > c$ 则a最大，输出a
 否则c大，输出c
否则
 如果 $b > c$ 则b最大，输出b
 否则c大，输出c

记录a,b之中较大的值为max
记录max,c之中较大的值为max

- 算法描述（伪代码）

```
if a > b
    if a > c 输出a
    else 输出c
else
    if b > c 输出b
    else 输出c
```

◎ 嵌套 if 语句（举例）

- 输入三个数a、b、c，输出其中最大者
 - C代码

```
#include <stdio.h>
void main() {
    int a, b, c;
    scanf("%d,%d,%d", &a, &b, &c);
    if (a>b)
        if (a>c) printf("a最大\n");
        else printf("c最大\n");
    else
        if (b>c) printf("b最大\n");
        else printf("c最大\n");
}
```


◎ 嵌套 if 语句（举例）

- 编程求函数值 y ，其中 x 由键盘输入

$$y = f(x) = \begin{cases} x & 0 \leq x < 10 \\ x^2 + 1 & 10 \leq x < 20 \\ x^3 + x^2 + 1 & 20 \leq x \leq 30 \end{cases}$$

注意这里忽略了 $x < 0$ 和 $x > 30$ 的越界情况。

```
#include <stdio.h>
void main() {
    float x, y;
    printf("x=");
    scanf("%f", &x);
    if (0<=x && x<20) {
        if (x>=10) y=x*x+1;
        else y=x;
    }
    else y=x*x*x+x*x+1;
    printf("x=%f, y=%f\n", x, y);
}
```

◎ 多分支 if 语句

- 用多重if语句处理多个分支

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 if(表达式n) 语句n

[else 语句n+1]

注意:

- else和if必须成套出现
- else和if中间有一个空格

- 执行过程: 逐个判断表达式 $i(i=1,2,\dots,n)$, 若条件满足, 则执行相应的语句 i , 否则继续向下判断, 直至执行语句 $n+1$ (未优化时)

◎ 多分支 if 语句（举例）

- 成绩显示为优(≥ 85)、良(84-75)、及格(74~60)、不及格(59~0)

```
#include <stdio.h>
void main()
{
    float score;
    printf("score=" );
    scanf("%d", &score); //输入一个成绩
    if(score>=85) printf("优秀\n");
    else if(score>=75) printf("良好\n");
    else if(score<=59) printf("不及格\n");
    else printf("及格\n");
}
```

59.5分算及格？

```
#include <stdio.h>
void main()
{
    float score;
    printf("score=" );
    scanf("%d", &score); //输入一个成绩
    if(score>=85) printf("优秀\n");
    else if(score>=75) printf("良好\n");
    else if(score>=60) printf("及格\n");
    else printf("不及格\n");
}
```



◎ 条件运算符

$\text{exp1} ? \text{exp2} : \text{exp3}$

- 计算表达式 exp1 的值，并测试是否为0
- 非0则对 exp2 求值，并作为条件表达式的输出
- 0则对 exp3 求值，并作为条件表达式的输出
- exp2 和 exp3 只计算1个，但类型不同时会隐式转换

• 举例

`int x;`

右结合算符

$\text{exp1} ? \text{exp2} : \text{exp3} ? \text{exp4} : \text{exp5}$ 等效于
 $\text{exp1} ? \text{exp2} : (\text{exp3} ? \text{exp4} : \text{exp5})$

`x > 0 ? 1 : x < 0 ? -1 : 0;` //x是正数输出1，负数输出-1，

◎ switch 语句

```
switch (表达式) {  
    case 常量表达式1: 语句序列1  
        break;  
    case 常量表达式2: 语句序列2  
        break;  
    ... ..  
    case 常量表达式n: 语句序列n  
        break;  
    default: 语句n+1  
}
```

```
switch (score/10) {  
    case 9: printf("优秀");  
        break;  
    case 8: printf("良好");  
        break;  
    case 7: printf("中等");  
        break;  
    case 6: printf("及格");  
        break;  
    default: printf("不及格");  
}
```

- 适合于可列举的多分支选择



◎ switch 语句

• 注意事项

- switch后面的表达式用于计算出一个可列值，可以是整型、字符型或枚举型
- case后面的常量(表达式)值必须在switch后的表达式的取值范围中，否则无效
- case只是一个语句标号(带:)，指示出应该跳转到的起始语句
- 一般地，每个case分支，一般应以break;语句结束，以跳出switch；若没有break语句，则会继续执行后续的case分支语句

```
switch (score/10) {  
    case 9: printf("优秀"); break;  
    case 8: printf("良好"); break;  
    case 7: printf("中等"); break;  
    case 6: printf("及格"); break;  
    default: printf("不及格");  
}
```



```
switch (score/10) {  
    case 9: printf("优秀");  
    case 8: printf("良好");  
    case 7: printf("中等");  
    case 6: printf("及格");  
    default: printf("不及格");  
}
```



- 当表达式的值不对应任何一个case常量时，执行default后的语句；若无default部分，则不执行任何语句

◎ switch 语句（举例）

- 成绩显示为优(≥ 90)、良(80-89)、中等(79~70)、及格(60~69)、不及格(59~0)

```
#include <stdio.h>
void main()
{
    int score;
    printf("score=" );
    scanf("%d",&score); //输入一个成绩
    switch (score/10) {
        case 10:
        case 9: printf("优秀\n"); break;
        case 8: printf("良好\n"); break;
        case 7: printf("中等\n"); break;
        case 6: printf("及格\n"); break;
        default: printf("不及格\n");
    }
}
```

◎ switch 语句（举例）

- 成绩显示为优(≥ 85)、良(84-75)、及格(74~60)、不及格(59~0)
 - 需要将实数转换为尽可能少的有限可列值？

```
#include <stdio.h>
void main()
{
    float score;
    printf("score=" );
    scanf("%f", &score); //输入一个成绩
    switch ((int)(score/5)) {
        case 20: case 19: case 18: case 17: printf("优秀\n"); break;
        case 16: case 15: printf("良好\n"); break;
        case 14: case 13: case 12: printf("及格\n"); break;
        default: printf("不及格\n");
    }
}
```



◎ switch 语句（举例）

- 某商场打折促销商品，规则是：购物款 <50 元不打折；50元到150元以内（含50元），折扣率为3%；150元到300元以内（含150元）的折扣率为5%；300元到500元以内（含500元）折扣率为8%；500元以上（含500元）折扣率为12%。编程实现输入购物款、计算并输出实付款数，要求用switch语句编写。
- 解题思路：
 - 根据打折规则的规律，确定表达式形式与case后的常量值；
 - 因折扣范围均是50的整数倍，可考虑用50去除购物款，从而得到若干整数常量值
 - 注意，计算时应把switch后表达式的值强制转换为整型



◎ switch 语句（举例）

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

```
void main()
```

```
{
```

```
    float p, d, f;
```

```
    scanf("%f", &p);
```

```
    switch ( (int) (p/50) ) {
```

```
        case 0: d=0; break;
```

```
        case 1:
```

```
        case 2: d=0.03; break;
```

```
        case 3:
```

```
        case 4:
```

```
        case 5: d=0.05; break;
```

```
        case 6:
```

```
        case 7:
```

```
        case 8:
```

```
        case 9: d=0.08; break;
```

```
        default : d=0.12;
```

```
    }
```

```
    f=p*(1-d);
```

```
    printf("购物款: %8.2f, 折扣率: %5.2f, 实付款: %8.2f\n", p,d,f);
```

```
}
```

若p以万元为单位，此处改为p/0.005，对吗？



◎ switch 语句

- 特别注意：不要将default拼错

- C语言有goto语句，支持标号。switch内容的任何语句都可以加上标号。
default 将被视为一个标号

```
switch (i) {  
    case 5+3: do_again:  
    case 2: printf("I loop unremittingly\n"); goto do_again  
    default: i++;  
    case 3: ;  
}
```

- 编译可以通过，但它相当于没有default子句；肉眼也难以分辨；如果它只在少数特殊条件下才执行，售后服务将是一场恶梦
 - 幸好，现在许多编辑环境支持以彩色标识关键字



◎ 循环结构

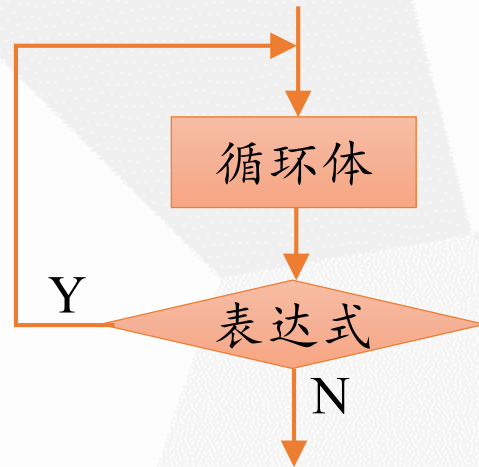
- 为什么要循环结构？

- 需要重复进行某项运算

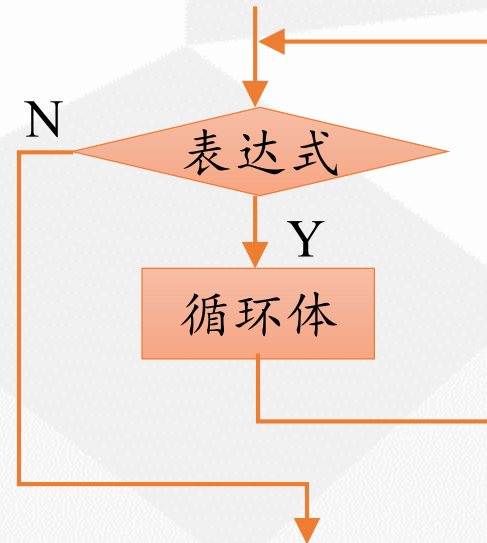
- 重复次数太多，以至于copy-paste也很麻烦，或不便于阅读
 - 需要重复的次数不定

- 两种基本循环方式：

先执行，
再判断



先判断，
再执行



◎ C语言的三种循环语句

A red notepad with a silver clip at the top, containing the text 'while' and '先判断, 再执行'.

while

先判断, 再执行

A red notepad with a silver clip at the top, containing the text 'do - while' and '先执行, 再判断'.

do - while

先执行, 再判断

A red notepad with a silver clip at the top, containing the text 'for' and '最为灵活。适当设置初值, 可以代替前两种'.

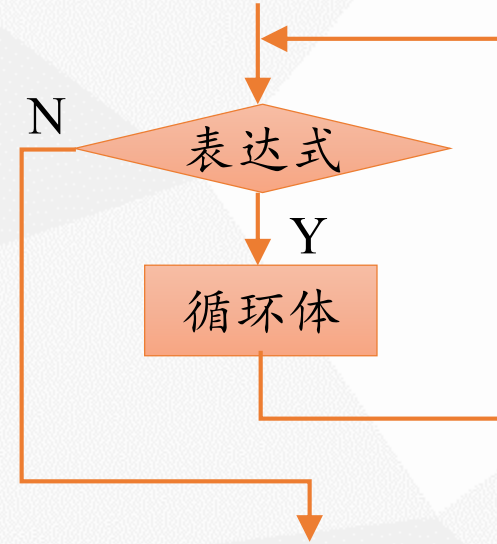
for

最为灵活。适当
设置初值, 可以
代替前两种

都可以用continue、break或goto语句改变循环的执行

◎ while 循环语句

while (表达式)
语句



- 语句称为循环体，可以是简单语句，也可以是复合语句或空语句
- 表达式也称为循环条件，用于控制何时退出循环
- 若表达式值始终非0，则无限制执行循环体（除非使用break、goto等语句主动退出循环），此时称为死循环，一般用于
 - 循环条件复杂，或情况较多
 - 无限期地等待

◎ while 循环语句（举例）

- 计算： $1 + \frac{1}{2} + \frac{1}{3} + \cdots + \frac{1}{n}$ $n = 100$

```
#include <stdio.h>
void main()
{
    int i=1;           //用i作为循环控制变量，置初值为1
    float sum=0;
    while(i<=100) {    //当i小于等于100时执行循环体，否则结束
        sum=sum+1.0/i;
        i++;           //每执行一次循环体，控制变量自增
    }
    printf("sum=%6.2f\n", sum);
}
```

◎ while 循环语句（举例）

- 对若干个乘客计收行李托运费。当输入行李重量小于等于0时结束程序运行。用 w 表示行李重量， d 表示每公斤的托运费率， f 表示托运费。根据行李重量的不同，托运费率规定如下：

行李重量（单位：kg）	托运费率（%）
$w < 20$	免费
$20 \leq w < 60$	0.15
$60 \leq w < 100$	0.20
$100 \leq w < 160$	0.25
$w \geq 160$	0.30

◎ while 循环语句（举例）

```
#include <stdio.h>
void main() {
    double f,d,w;
    printf("输入行李重量: ");
    scanf("%lf", &w);
    while (w>0) {
        switch((int)(w/20)) {
            case 0: printf("免费! \n"); d=0.0; break;
            case 1:
                ...
            default: d=0.30;
        }
        f=w*d;
        printf( ...
        printf( ...
        scanf("%lf", &w);
    }
}
```

行李重量（单位：kg） 托运费率（%）

w<20	免费
20<=w<60	0.15
60<=w<100	0.20
100<=w<160	0.25
w>=160	0.30

分段收费

```
f=0;
switch(...) {
    default:
        case 8: f+=(w-160)*0.30; w=160;
        case 7: case 6: case 5: f+=(w-100)*0.25; w=100;
        case 4: case 3: f+=(w-60)*0.2; w=60;
        case 2: case 1: f+=(w-20)*0.15;
        case 0: f+=0;
}
```

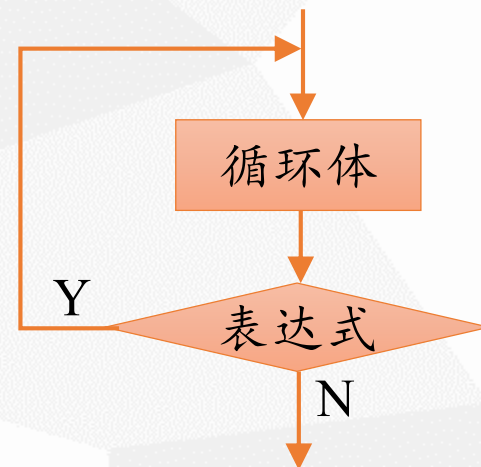
◎ do - while 循环语句

do

语句

while (表达式);

- 先执行，再判断。其它与while语句相同
- 注意最后有个“;”



◎ do - while 循环语句（举例）

- 计算： $1 + \frac{1}{2} + \frac{1}{3} + \dots + \frac{1}{n}$ $n = 100$

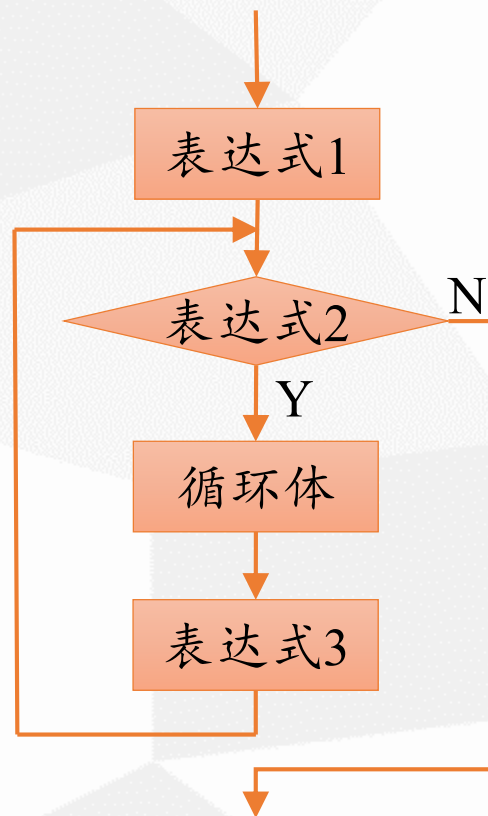
```
#include<stdio.h>
void main() {
    int i=100;           //用i作为循环控制变量，置初值为100
    float sum=0;
    do {
        sum=sum+1.0/i;
        i--;             //每执行一次循环体，控制变量自减
    } while(i>=1);       //当i大于等于1时执行循环体，否则结束
    printf("sum=%6.2f\n",sum);
}
```

为什么 i 不定义成
float 方便后续运算?

◎ for 循环语句

for (表达式1; 表达式2; 表达式3)
语句

- 先判断，再执行
- 表达式1一般用来对循环控制变量进行初始化
- 表达式2用来判断循环是否结束，值为非0则执行循环体，为0则结束循环
- 表达式3用来修改循环控制变量的值（增或减）
- 表达式1中也可以包含其他初始化操作；表达式3中也可以包含其它每次循环后都要进行的操作。注意，用逗号分隔多个表达式时，逗号是左结合的



◎ for 循环语句（举例）

- 计算： $1 + \frac{1}{2} + \frac{1}{3} + \cdots + \frac{1}{n}$ $n = 100$

```
#include<stdio.h>
void main() {
    int i;
    float sum;
    for (sum=0, i=1; i<=100; i++)
        sum=sum+1.0/i;
    printf("sum=%6.2f\n",sum);
}
```

◎ for 循环语句（举例）

- 计算： $n!$

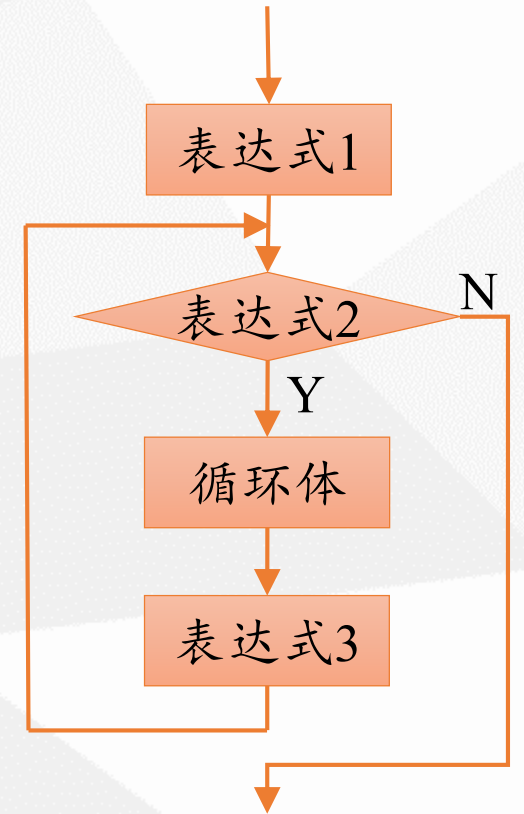
```
#include<stdio.h>
void main() {
    int i,n;
    long fact=1;           //阶乘值较大，用long防止溢出
    scanf("%d", &n);
    for (i=2; i<=n; i++)
        fact=fact*i;
    printf("%d != %ld\n", n, fact);
}
```

```
for (fact=1, i=1; i<=n; fact=fact*(i++));
```



◎ for 循环语句

- 表达式1可以省略，即不设置初值，但表达式1后的分号不能省略。例如: `for(; i<=100; i++)`。此时应在for语句之前给循环变量赋以初值
- 表达式2也可以省略，即不用表达式2来作为循环条件表达式，同样分号不能省略。此时循环无终止地进行下去。此时应在循环内设置退出机制
- 表达式3也可以省略，此时应另外设法保证循环适时退出
- 甚至可以将3个表达式都可省略，即不设初值，不判断条件(认为表达式2为真值)，循环变量也不增值，无终止地执行循环体语句，`for(;;) ↔ while(1)`



◎ 三种循环语句的总结

- 3种循环可以互相代替
- do-while语句的循环体至少执行一次，while和for语句的循环体可以一次也不执行
- for允许在表达式中包含更多运算，功能更强，形式更灵活。
- 凡用while循环能完成的，用for循环都能实现
 - while(条件) 语句 \leftrightarrow for(; 条件;) 语句
 - do 语句 while(条件) \leftrightarrow 语句; for (; 条件;) 语句



◎ 循环的嵌套

- 一个循环体内可以包含一个完整的循环结构，称为循环的嵌套

```
while ( i<100 ) {  
    .....  
    for (j=0; j<10; j++) { ..... }  
    .....  
}
```

```
for (i=0; i<100; i++) {  
    .....  
    for (j=0; j<10; j++) { ..... }  
    .....  
}
```

- {}、() 的配对原则保证了循环嵌套不可以交叉

```
while ( ... ) {  
    .....  
    do {  
        .....  
    }  
} while (...);
```

◎ 内外层循环控制变量一般不应重名

```
for (i=0; i<100; i++) {  
    .....  
    for (i=0; i<10; i++) {  
        .....  
    }  
    .....  
}
```

死循环

◎ 循环控制（举例）

- 求3~100之间的所有素数（按每行8个排列输出）

- 解题思路

- 一个整数 n ，不能被 $2 \sim n-1$ 中任何一个数整除，则为素数
- 为减小计算量，整数 n 只要不能被 $2 \sim \sqrt{n}$ 之间的任何一个数整除即可说明它是素数
- 从3到100，逐个判断是否符合上述条件
- 考虑到偶数肯定不是素数，可以略过

◎ 循环控制（举例）

- 求3~100之间的所有素数（按每行8个排列输出）

- 算法主体

- 需要一个循环，来穷举所有奇数，逐个考察
- 对每一个奇数，再用一个循环，试除2~sqrt(n)
- 若都除不尽，则是素数，显示该素数

```
for (n=3; n<100; n=n+2) {    //逐个测试所有奇数
    for (i=2; i<=sqrt(n); i++) {    //逐个测试可能的因子
        if (能整除) then 不是素数，结束i循环，继续下一个n
    }
    if (都没能整除) then 是素数，显示输出该素数;
}
```



◎ 循环控制（举例）

- 如何判断是循环结束退出，还是中途强制退出？

- 再次判断循环条件：

- 循环变量使得循环条件不满足，说明是循环结束退出；如果循环条件仍满足，说明是中途强制退出

if ($i > \text{sqrt}(n)$) then 是素数, 显示输出该素数;

- 设置标记：

- 循环开始前清除标记，需要中途强制退出时设置标记。通过判断标记状态，就可以知道是何种退出

```
for (flag=0; i=2; i<=sqrt(n); i++) {  
    if (能整除) then 不是素数, flag=1; 结束i循环, 继续下一个n  
}  
if (flag==0) then 是素数, 显示输出该素数;
```



◎ 循环控制（举例）

- 求3~100之间的所有素数（按每行8个排列输出）

```
#include <stdio.h>
#include <math.h>
void main() {
    int n, k, i;
    int m=0;           //计数器，打印换行用
    for (n=3;n<100;n=n+2) {
        k=sqrt(n);
        for (i=2; i<=k; i++)           //嵌套循环
            if (n%i==0) break;         //不是素数
        if (i>=k+1) {                  //是素数
            m++;
            printf("%6d",n);
            if (m%8==0) printf("\n"); //8个换行
        }
    }
}
```

```
#include <stdio.h>
#include <math.h>
void main() {
    int n, k, i;
    int flag, m=0;      //计数器，打印换行用
    for (n=3;n<100;n=n+2) {
        k=sqrt(n);
        for (flag=0, i=2; i<=k; i++)           //嵌套循环
            if (n%i==0) { flag=1; break;}      //不是素数
        if (flag==0) {                          //是素数
            m++;
            printf("%6d",n);
            if (m%8==0) printf("\n"); //8个换行
        }
    }
}
```

◎ 控制转向语句

- **break**: 跳出 **switch** 或 **循环** 语句
- **continue**: 忽略循环体中后续语句，执行循环语句的判断部分
- **goto**: 跳转到标号指定的语句（同一函数内）
 - 都不是结构化控制语句，因其会改变语句执行顺序，为分析程序结构带来困扰
 - 应尽量避免使用 **goto** 语句



◎ break 语句

break;

- 只能用在switch语句或循环语句中，跳出当前结构
 - 注意，多重循环时，break只能跳出当前循环，而不是跳出所有的循环
- 例：某企业当前产值为1千万，编写根据不同的年增长率，计算产值增长一倍所需年数的程序（即产值翻番）
 - 解题思路I：对于年增长率d，
$$(1+d)^x=2 \rightarrow x=\log 2 / \log (1+d)$$



◎ break 语句（举例）

- 某企业当前产值为1千万，编写根据不同的年增长率，计算产值增长一倍所需年数的程序（即产值翻番）

- 解题思路II:

用循环模拟时间的流逝，循环变量代表年，累积计算当年的产值，当产值达到2倍时结束循环，输出结果

```
c1=c;           //设初始产量为c，年增长率为d
for (y=1; ; y++) {
    c1=c1*(1+d);
    if (c1>=2*c) break;
}
printf("d=%lf, year=%d, OutputValue=%lf\n",d,y,c1);
```

◎ continue 语句

continue;

- 有时在循环体中执行一些语句后，其后的语句在本次循环中就不需要再执行了，而是继续下一次循环的判断，则可以使用continue语句，因此该语句有时被称作“**短路**”语句
- 例：编写一个计算 $s1=1+2+3+\cdots+100$ 及 $s2=1+3+5+\cdots+99$ 的累加求和程序，要求用一个循环语句实现

```
for (s1=0, s2=0, i=1; i<=100; i++) {  
    s1=s1+i;  
    if (i%2==0) continue;           //若为偶数，则继续下次循环  
    s2=s2+i;                         //只对奇数求和  
}
```

◎ goto 语句

goto 标号

标号: 语句

- goto语句的功能是使程序执行能无条件地转移到标号所标识的语句去执行（必须在同一函数内）
- 严重影响可读性，建议不使用
- 标号出现在语句前面，可以是任意合法的标识符，甚至可以和变量同名
- 标号一般用作goto语句的转向目标
- goto可以构成循环，也可以从循环中跳出



◎ goto 语句（举例）

- 用近似公式求自然常数/欧拉数 e 的值， $e = 1 + 1/1! + 1/2! + 1/3! + \dots + 1/n!$ ，直到 $1/n!$ 小于 10^{-6} 为止，要求使用goto语句构成循环

```
#include<stdio.h>
void main() {
    float t, e=1;
    int n=1, i=1;
add: t=1.0/n;
    e=e+t;
    i++;
    n=n*i;
    if (t>1e-6) goto add;
    printf ("e=%f\n", e);
}
```

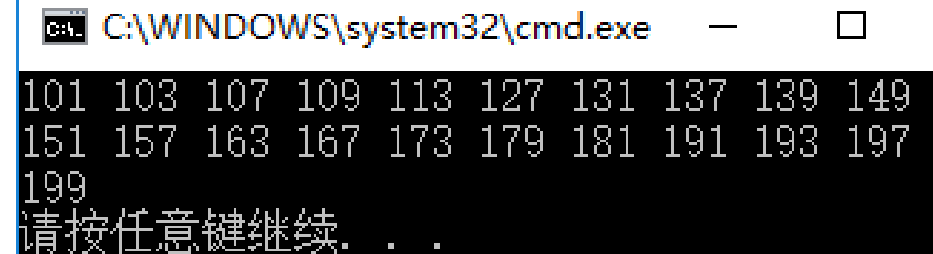


◎ 循环控制（举例）

• 求100~200间的全部素数

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

如果在结果中出现了121, 169,
可能是什么地方出错？



```
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. . .
```



◎ 循环控制（举例）

- 为使电文保密，往往按一定规律将其转换成密文，收报人再按约定的规律将其译回原文。约定明文用小写字母表示，密文用大写字母表示。例如，凯撒密码： $C=m+4$ ，将a变成E，b变成F，...，w变成A，x变成B，y变成C，z变成D，即变成其后的第4个字母。

```
#include <stdio.h>
int main() {
    char c;
    while ((c=getchar())!='\n') { //读入一个字符，若为换行符'\n'则结束
        if (c>='a' && c<='z') { //c如果是小写字母
            c='A'+(c-'a'+4)%26;
        }
        printf("%c",c); //输出已改变的字符
    }
    printf("\n");
    return 0;
}
```



◎ 不恰当的初始化引起的错误（举例）

- 求 $1+2+3+\dots+100$ ，即： $\sum_{n=1}^{100} n$

```
#include<stdio.h>
int main() {
    int i, sum;
    i=1;
    sum=0;
    while (i<=100) {
        sum=sum+i;
        i++;
    }
    printf("%d\n",sum);
    return 0;
}
```



```
#include<stdio.h>
int main() {
    int i, sum;
    i=1;
    while (i<=100) {
        sum=0;
        sum=sum+i;
        i++;
    }
    printf("%d\n",sum);
    return 0;
}
```



```
#include<stdio.h>
int main() {
    int i, sum;
    sum=0;
    while (i<=100) {
        i=1;
        sum=sum+i;
        i++;
    }
    printf("%d\n",sum);
    return 0;
}
```

◎ 不恰当的初始化引起的错误（举例）

- 求 $1+2+3+\dots+100$ ，即： $\sum_{n=1}^{100} n$

```
#include<stdio.h>
int main() {
    int i, sum;
    i=1;
    sum=0;
    while (i<=100) {
        sum=sum+i;
        i++;
    }
    printf("%d\n",sum);
    return 0;
}
```



```
#include<stdio.h>
int main() {
    int i, sum;
    i=1;
    sum=0;
    while (i<=100) {
        i++;
        sum=sum+i;
    }
    printf("%d\n",sum);
    return 0;
}
```


◎ 循环控制处置不当引起的错误（举例）

- 输出100~200之间的不能被3整除的数

```
#include <stdio.h>
int main() {
    int n;
    for (n=100;n<=200;n++) {
        if (n%3==0)
            continue;
        printf("%d ",n);
    }
    printf("\n");
    return 0;
}
```



```
#include <stdio.h>
int main() {
    int n;
    n=100;
    while (n<=200) {
        if (n%3==0)
            continue;
        printf("%d ",n);
        n++;
    }
    printf("\n");
    return 0;
}
```

◎ 循环控制处置不当引起的错误（举例）

- 输出ASCII码 0 ~ 255 对应的字符

```
#include <stdio.h>
int main() {
    char n;
    for (n=0; n<=255; n++)
        printf("%c ",n);
    printf("\n");
    return 0;
}
```

◎ 常用算法

- **递推法**：按照时间、计算步骤的先后顺序，逐步求解。分顺推法和倒推法
- **贪心法（贪婪算法）**：逐步求解，每步都选择当前看起来最优的解，效率高，但有时导致局部最优（非全局最优）
- **枚举法（穷举法）**：遍历问题的所有可能情况，从中寻求结果（显然必须解空间有限），效率低，但可改进
- **递归法**：通过重复将问题分解为同类的子问题而解决问题的方法，效率低，内存消耗大
- **分治法**：将大问题分解为一系列规模较小的相同问题，逐个解决（**动态规划**沿决策的阶段划分子问题）



◎ 常用算法（递推法）

- 求Fibonacci(斐波那契)数列的前20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：

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

- 这是一个有趣的古典数学问题：有一对兔子，从出生后第3个月起每个月都生一对兔子。小兔子长到第3个月后每个月又生一对兔子。假设所有兔子都不死，问每个月的兔子总数为多少？

◎ 常用算法（递推法）

- 例：Fibonacci（斐波那契）数列

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

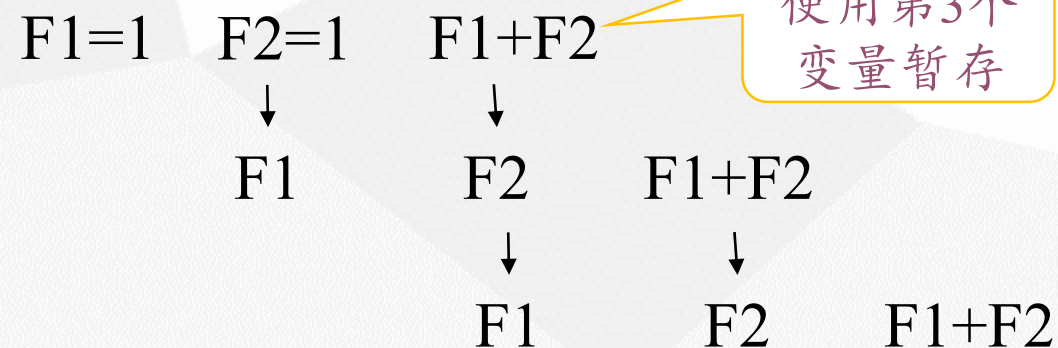
• 递推法

- 设置一个数组，根据公式，从第3个月开始逐个计算

F(1)	F(2)	F(3) =F(1)+F(2)	F(4) =F(2)+F(3)
------	------	--------------------	--------------------	-------

太浪费
存储空间

- 使用两个变量，辗转存储

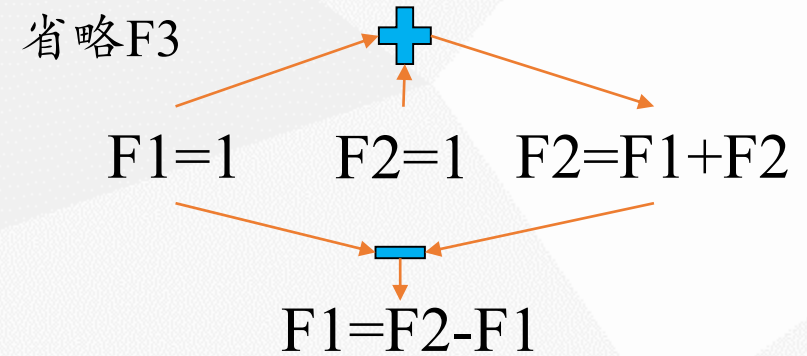


◎ 常用算法（递推法）

- 例：Fibonacci（斐波那契）数列

$F1=1$ $F2=1$ $F3=F1+F2$
 \downarrow \downarrow
 $F1$ $F2$ $F1+F2$

```
#include <stdio.h>
int main() {
    int i, f1=1, f2=1, f3;
    printf("%12d\n%12d\n",f1,f2);
    for (i=1; i<=18; i++) {
        f3=f1+f2;
        printf("%12d\n",f3);
        f1=f2;
        f2=f3;
    }
    return 0;
}
```



```
#include <stdio.h>
int main() {
    int i, f1=1, f2=1;
    printf("%12d\n%12d\n",f1,f2);
    for (i=1; i<=18; i++) {
        f2=f1+f2;
        f1=f2-f1;
        printf("%12d\n",f2);
    }
    return 0;
}
```

◎ 常用算法（递推法）

- 例：Fibonacci（斐波那契）数列

一次计算两个月

F1 F2 $F1=F1+F2$ $F2=F1+F2$

```
#include <stdio.h>
int main() {
    int i,f1=1,f2=1;
    for (i=1; i<=10; i++) {
        printf("%12d %12d ",f1,f2);
        if (i%4==0) printf("\n");
        f1=f1+f2; //计算两个月
        f2=f2+f1;
    }
    return 0;
}
```

◎ 常用算法（贪心法）

- 埃及分解：设a、b为互质正整数，且a<b，要求将分数a/b分解成若干个单位分数（分子为1）之和，又称埃及分数。如 $5/6=1/2+1/3$

◦ 贪心法

- 每次找出一个小于a/b的最大的单位分数
- 令 $b/a = q + r$

$$\begin{aligned}\frac{a}{b} &= \frac{1}{q+1} + \left(\frac{a}{b} - \frac{1}{q+1} \right) \\ &= \frac{1}{q+1} + \frac{a-r}{b(q+1)}\end{aligned}$$

- 令a=a-r，b=b(q+1)。若此时a/b是单位分数，结束。否则，进入下一轮

```
#include <stdio.h>
void main() {
    int a=3, b=7, q, r;
    printf("%d/%d=", a, b);
    while (r=b%a) {
        q=b/a;
        printf("1/%d+", q+1);
        a=a-r;
        b=b*(q+1);
    }
    printf("1/%d\n", b/a);
}
```


◎ 常用算法（枚举法）

- 百鸡问题：“鸡翁一，值钱五；鸡母一，值钱三；鸡雏三，值钱一。百钱买百鸡，问鸡翁、鸡母、鸡雏各几何？”

◦ 枚举法

```
#include <stdio.h>
void main() {
    int a,b,c;
    for (a=0; a<=100; a++)
        for (b=0; b<=100; b++)
            for (c=0; c<=100; c++)
                if ((a+b+c=100) && (5*a+3*b+c/3==100) && (c%3==0))
                    printf("%d,%d,%d\n", a, b, c);
}
```

- 百万次循环！太慢了！



◎ 常用算法（枚举法）

```
#include <stdio.h>
void main() {
    int a,b,c;
    for (a=0; a<=20; a++)
        for (b=0; b<=33; b++){
            c=100-a-b;
            if ((5*a+3*b+c/3==100) && (c%3==0))
                printf("%d,%d,%d\n", a, b, c);
        }
}
```

进一步

约束b的终点

约束b的起点

a=0, b=1 (c是3的倍数)

a=1, b=0

a=2, b=2

.....

```
#include <stdio.h>
void main() {
    int a,b,c;
    for (a=0; a<=20; a++)
        for (b=(4-a%3)%3; b<=(100-a*5)/3; b+=3) {
            c=100-a-b;
            if (5*a+3*b+c/3==100)
                printf("%d,%d,%d\n", a, b, c);
        }
}
```

◎ 常用算法（枚举法）

- 百鸡问题：“鸡翁一，值钱五；鸡母一，值钱三；鸡雏三，值钱一。百钱买百鸡，问鸡翁、鸡母、鸡雏各几何？”

```
#include <stdio.h>
void main() {
    int a,b,c;
    for (a=0; a<=20; a++)
        for (c=0; c<=100-a; c+=3) {
            b=100-a-c;
            if (5*a+3*b+c/3==100)
                printf("%d,%d,%d\n", a, b, c);
        }
}
```

◎ 作业

- 10. 企业发放奖金根据利率提成。（谭书 P109）
- 8. 输出所有的“水仙花数”。（谭书 P137）
- 11. 计算球落地反弹。（谭书 P138）

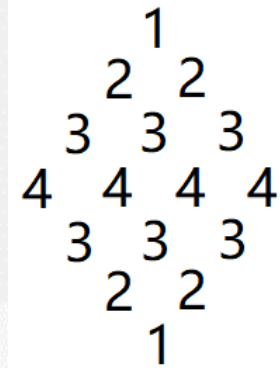


◎ 课外拓展

- 从网上查询C语言的使用手册，例如
<https://en.cppreference.com/w/c>，阅读并学习：
 - 类型：<https://en.cppreference.com/w/c/language/type>
 - 语句：<https://en.cppreference.com/w/c/language/statements>
 - 输入输出：<https://en.cppreference.com/w/c/io>
 - 常用数学库：<https://en.cppreference.com/w/c/numeric/math>



◎ 上机作业



1. 编程输出右侧图形。

其中上半层的高度由键盘输入，不超过9。

2. 编程输出Y年M月的格式化日历。（ $1600 \leq Y \leq 2100$ ）

◦ 提示：用蔡勒（Zeller）公式计算Y年M月第一天是星期几。

Sun	Mon	Tue	Wed	Thu	Fri	Sat
		1	2	3	4	5
6	7	8	9	10	11	12
13	14	15	16	17	18	19
20	21	22	23	24	25	26
27	28	29	30			

3. 编程按以下公式计算 π 的近似值。精度由键盘输入，例如1e-10。

◦ 提示：用math.h库中的pow函数计算 16^k ，如：pow(16, k)

$$\pi = \sum_{k=0}^{\infty} \frac{1}{16^k} \left(\frac{4}{8k+1} - \frac{2}{8k+4} - \frac{1}{8k+5} - \frac{1}{8k+6} \right)$$

◎ 上机作业

4. 哥德巴赫猜想：一个不小于6的偶数可以表示为两个素数之和。
如 $6=3+3$ ， $8=3+5$ ， $10=3+7$ 。要求验证6-1000之间的全部偶数。
 - 提示：利用之前介绍的判断素数的方法，尽可能减少不必要的枚举。
5. 尼克切丝定理：任何一个整数的立方都可以表示成一串连续的奇数之和。设计程序验证尼克切丝定理。
 - 提示：不要简单的枚举连续的奇数，考虑这些奇数满足要求的可能性。
6. 输入5个数与一个结果，设计程序在5个数（顺序固定）中填入运算符（+、-、*、/）使其与结果相等，或回答做不到。
 - 提示：运算符可以重复使用，提前判断减少不必要的枚举。



◎ 上机要求

- 上机前先把程序写好，上机时直接输入，否则浪费时间
- 不该立即向助教提问的问题
 - 编译出错
 - 程序结果不对
- 上机过程中，希望不要一有问题就立刻问助教
 - 先尝试着自行解决，培养解决问题的能力
 - 然后尝试上网搜索，仍无法解决时再问助教
- 不要抄袭上机作业！！！！

