



《计算概论A》课程 程序设计部分

从现实问题到计算机程序

李 戈

北京大学 信息科学技术学院 软件研究所

lige@sei.pku.edu.cn

程序, 是你告诉计算机的话

■ 计算机其实很笨

- ◆ 它可以按照“你告诉它的话去执行”！
- ◆ 它却不能帮你“想出”解决一个问题的办法！

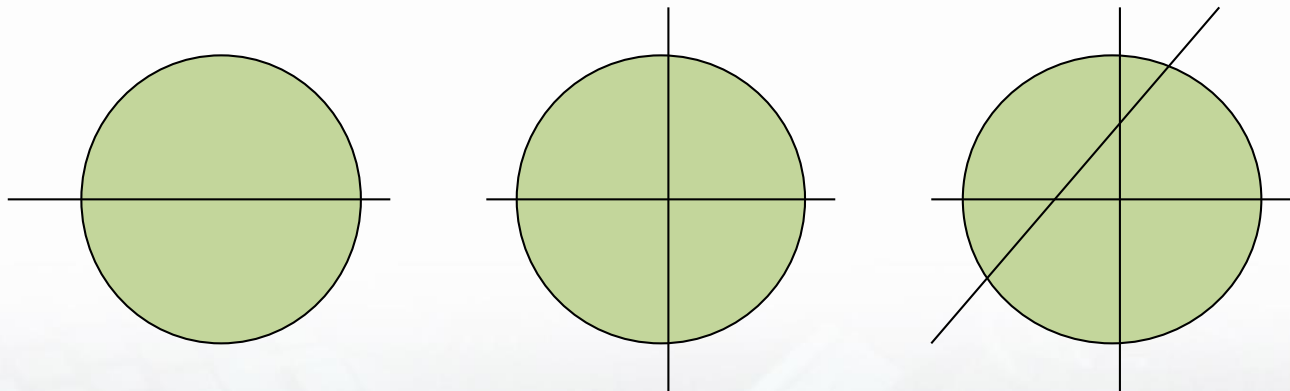


■ 你必须告诉它怎么解决一个问题，它才能去做！

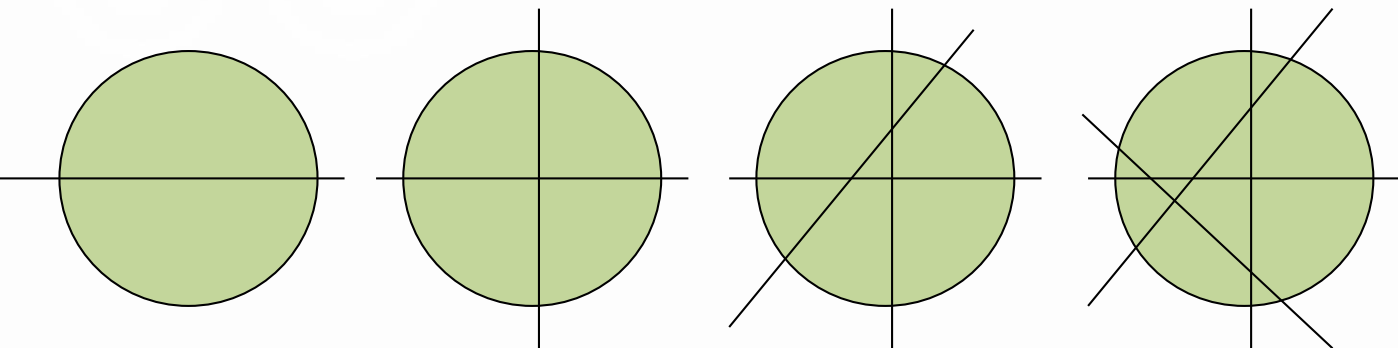
从一个例子开始说起

■ 切饼

- ◆ 假设：有一张很大很大的饼，给你一把足够长的刀。
- ◆ 要求：每次在饼上切一刀。
- ◆ 问题： n 刀，最多能切出多少块饼？



从一个例子开始说起



- $q(1) = 1+1=2$
- $q(2) = 1+1+2=4;$
- $q(3) = 1+1+2+3=7;$
- $q(4) = 1+1+2+3+4=11;$
- $q(n) = q(n-1)+n; \quad q(0)=1;$

因此：

- 第n刀切下去，最多比切之前多出n块；
- 第1刀切下去，得到2块；

从这个例子我们知道

- 在你还没有想到解决方案的时候，不要急着动手去写程序！





从这个例子我们知道

是不是有了解决方案就有程序了？



问题



解决方案

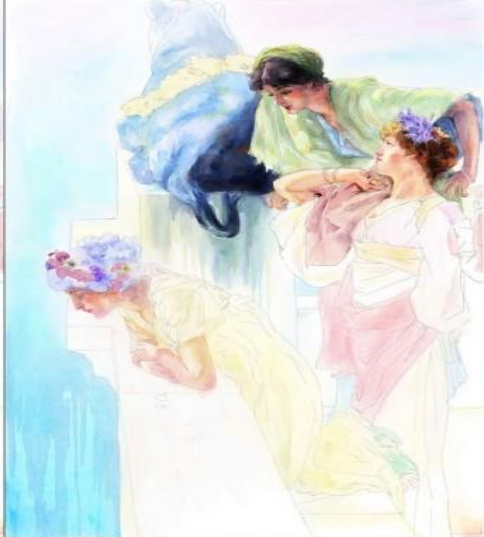


程序

从解决方案到程序



在结构化程序设计中，总是按照“**先粗后细，先抽象后具体**”的办法，对所要描述的解决方案进行穷尽分解，直到分解为顺序、分支、循环三种结构。

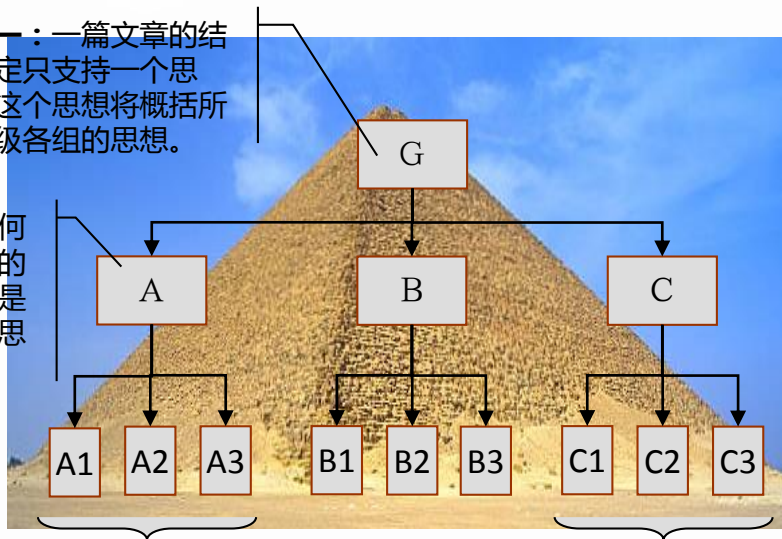




自然语言写作的规律

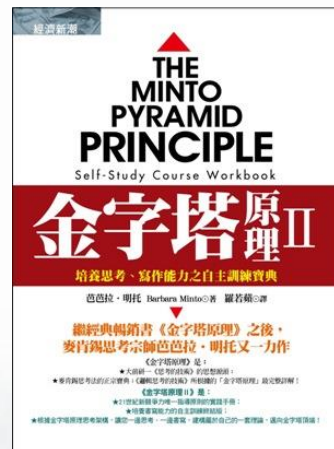
特征一：一篇文章的结构必定只支持一个思想，这个思想将概括所有各级组的思想。

特征二：任何一个层次上的思想都必须是其下一层次思想的概括。

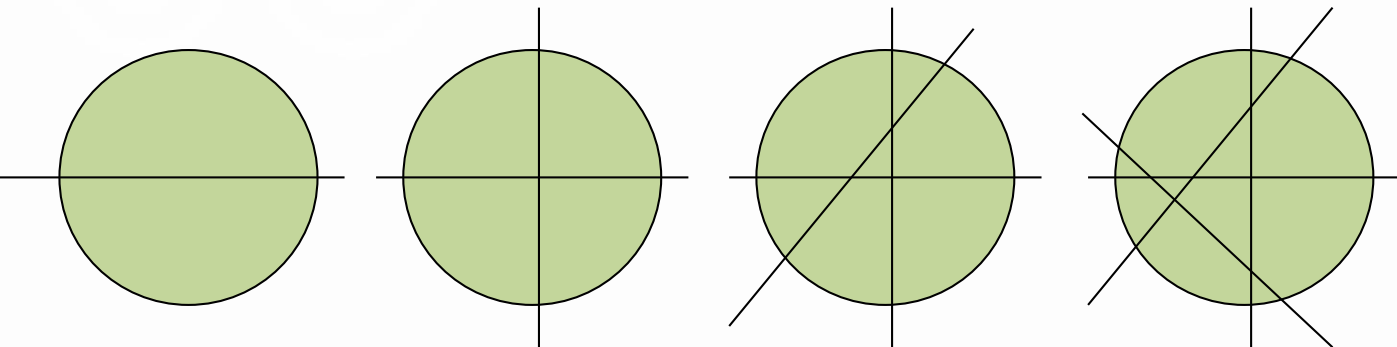


特征三：每组中的思想必须属于同一个范畴。

特征四：每组中的思想都必须按照逻辑顺序组织。

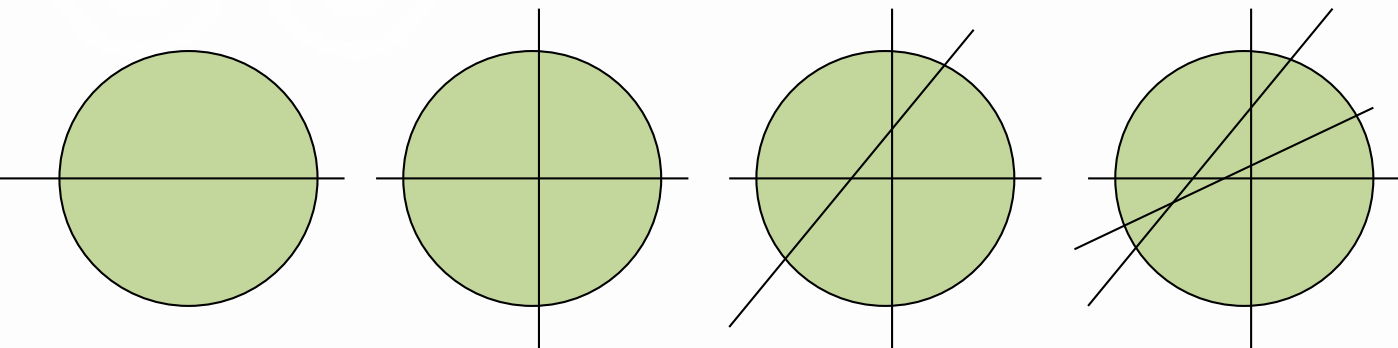


从一个例子开始说起



- $q(1) = 1+1=2$
- $q(2) = 1+1+2=4;$
- $q(3) = 1+1+2+3=7;$
- $q(4) = 1+1+2+3+4=11;$
- $q(n) = q(n-1)+n; \quad q(0)=1;$

从一个例子开始说起



从一个例子开始说起

```
#include <iostream>
using namespace std;
int main()
{
    int blockCount = 1;
    int i = 0, N = 0;

    cin>>N;

    for (i = 1; i <= N; i++)
        blockCount = blockCount + i;

    cout<<"最多可切"<<blockCount<<"块"<<endl;

    return 0;
}
```

从这个例子我们知道



- 没有想好解决方案，不要急于动手写程序？
- 有了解决方案以后，可以按照“先粗后细、先抽象后具体”的办法，先有程序的轮廓，如有必要可以借助“建模工具”画一些图，而后再动手写程序；
- 写程序时，可以**先写出程序轮廓**，而后再补变量定义等细节；

示例1 鸡兔同笼问题

■ 问题描述

- ◆ 一个笼子里面关了鸡和兔子（鸡有2 只脚，兔子有4 只脚，没有例外）。已经知道了笼子里面脚的总数 a ，问笼子里面至少有多少只动物，至多有多少只动物？

■ 输入样例

2

3

20

■ 输出样例

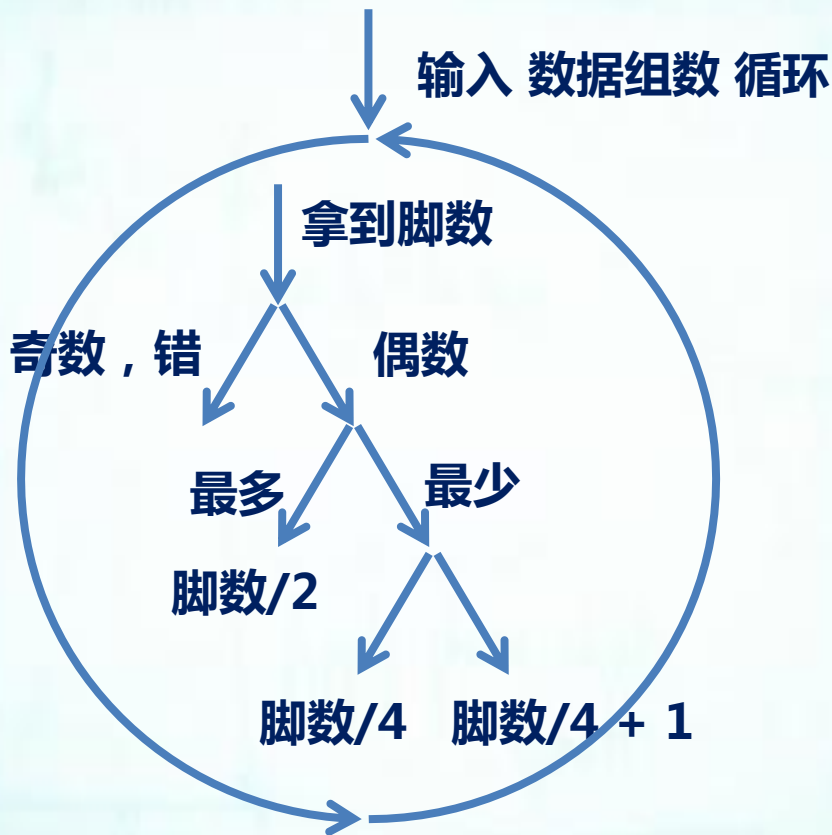
0 0

5 10

示例1 鸡兔同笼问题



示例1 鸡兔同笼问题



```
#include <iostream>
using namespace std;
int main()
{
    int nCases, i, nFeet;
    //nCases 表示输入测试数据的组数， nFeet 表示输入的脚步数。
    cin >> nCases;
    for (i = 0; i < nCases; i++){
        cin >> nFeet;
        if (nFeet % 2 != 0) // 如果有奇数只脚，则输入不正确，
            // 因为不论2只还是4只，都是偶数
            cout << "0 0" << endl;
        else if (nFeet % 4 != 0)
            //若要动物数目最少，使动物尽量有4只脚
            //若要动物数目最多，使动物尽量有2只脚
            cout << nFeet / 4 + 1 << " " << nFeet / 2 << endl;
        else
            cout << nFeet / 4 << " " << nFeet / 2 << endl;
    }
    return 0;
}
```

**在思考解决方案的时候，记得利用
计算机的特性 —— 不怕啰嗦！**

示例2 百元买百鸡问题

■ 问题描述

- ◆ 假定小鸡每只 5 角，公鸡每只2 元，母鸡每只3 元。现在有100 元钱要求买100 只鸡，编程列出所有可能的购鸡方案。

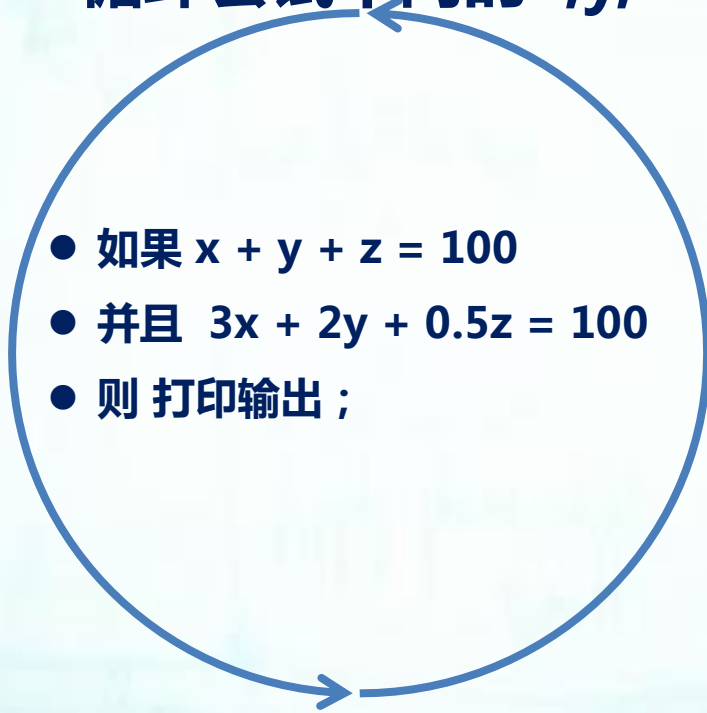
$$\begin{cases} x + y + z = 100 \\ 3x + 2y + 0.5z = 100 \end{cases}$$

穷举:

- ◆ 将可能出现的各种情况——测试，判断是否满足条件；

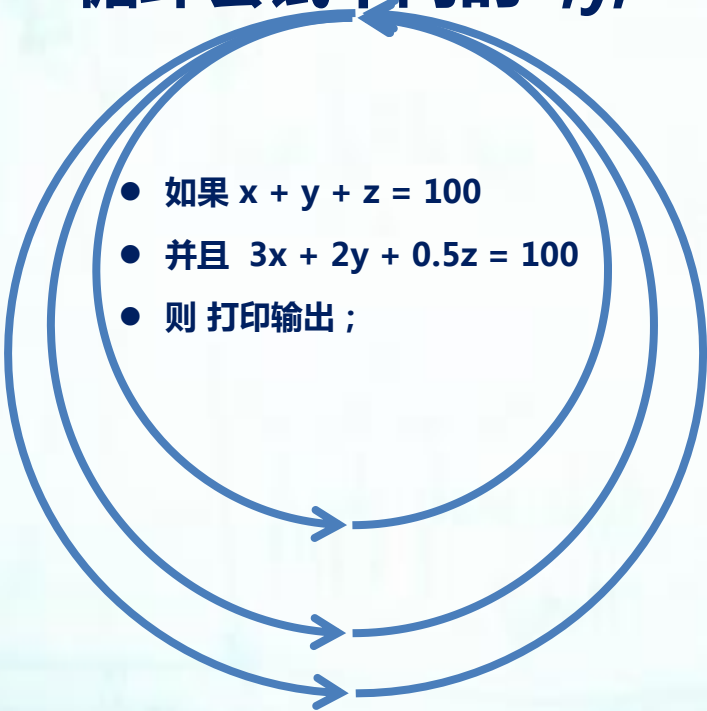
示例2 百元买百鸡问题

循环尝试不同的 x, y, z

- 
- 如果 $x + y + z = 100$
 - 并且 $3x + 2y + 0.5z = 100$
 - 则 打印输出 ;

示例2 百元买百鸡问题

循环尝试不同的 x, y, z

- 如果 $x + y + z = 100$
 - 并且 $3x + 2y + 0.5z = 100$
 - 则 打印输出；
- 



解决方案整理

■最简单的办法：

◆ 循环：对于每个 $X \leq 33$

● 循环：对于每个 $y \leq 50$

◆ 循环：对于每个 $z \leq 100$

●* 如果 $x + y + z = 100$

● 并且 $3x + 2y + 0.5z = 100$

● 则 打印输出；

```
#include <iostream>
using namespace std;
int main()
{
    int x, y, z;
    cout << "\t 母鸡\t\t 公鸡\t\t 小鸡" << endl;
    for (x = 0; x <= 33; x++)
    for (y = 0; y <= 50; y++)
    for (z = 0; z <= 100; z++)
    {
        if ((x + y + z) == 100)
        if ((3 * x + 2 * y + 0.5*z) == 100)
            cout << "\t" << x << "\t\t"
                << y << "\t\t" << z << endl;
    }
    return 0;
}
```




稍作简化的解决方案

- 由于 $x + y + z = 100$, 且 $x \leq 33$, $y \leq 50$, $z < 100$;
- 最简单的办法 :
 - ◆ 循环 : 对于每个 $x \leq 33$
 - 循环 : 对于每个 $y \leq 50$
 - ◆ 对于每个 $z = 100 - x - y$
 - 如果 $3x + 2y + 0.5z = 100$
 - 则 打印输出 ;

```
#include <iostream>
using namespace std;
int main()
{
    int x, y, z;
    cout << "\t 母鸡\t\t 公鸡\t\t 小鸡" << endl;
    for (x = 0; x <= 33; x++)
    for (y = 0; y <= 50; y++)
    {
        z = 100 - x - y;
        if ((3 * x + 2 * y + 0.5*z) == 100)
            cout << "\t" << x << "\t\t"
                << y << "\t\t" << z << endl;
    }
    return 0;
}
```

```
#include <iostream>
using namespace std;
int main()
{
    int x, y, z;
    cout << "\t 母鸡\t\t 公鸡\t\t 小鸡" << endl;
    for (x = 0; x <= 33; x++)
    for (y = 0; y <= 50; y++)
    {
        z = 100 - x - y;
        if ((3 * x + 2 * y + 0.5*z) == 100)
            cout << "\t" << x << "\t\t"
                << y << "\t\t" << z <<
endl;
    }
    return 0;
}
```



示例3 整数奇偶排序

■ Description

输入10个0 ~ 100之间的不同整数，彼此以空格分隔重新排序以后输出(也按空格分隔)，要求：

- 1.先输出其中的奇数,并按从大到小排列；
- 2.然后输出其中的偶数,并按从小到大排列。

■ Input

任意排序的10个整数（0 ~ 100），彼此以空格分隔

■ Output

按照要求排序后输出，由空格分隔

```
#include<iostream>
using namespace std;
int main() {
    //定义变量
    //all为全部十个数: odd记录奇数、even记录偶数, odd、even至多10个
    int all[10], odd[10], even[10];
    //i,j为循环变量
    int i = 0, j = 0;
    //依次输入10个数至all, i为all的下标
    for (; i < 10; i++)
        cin >> all[i];
    //numOdd, numEven分别记录奇数、偶数的个数
    int numOdd = 0;
    int numEven = 0;
    //遍历数组all, 如果当前all[i]为奇数则放入odd[numOdd],
    //偶数放入even[numEven]
    for (i = 0; i < 10; i++){
        if (all[i] % 2 != 0){//奇数
            odd[numOdd] = all[i];
            numOdd++;
        }
        else{//偶数
            even[numEven] = all[i];
            numEven++;
        }
    }
}
```

```

// 对odd冒泡排序
for (i = 0; i < numOdd - 1; i++){
    for (j = 0; j < numOdd - i - 1; j++){
        if (odd[j] > odd[j + 1]){
            //tmp为临时变量
            int tmp = odd[j + 1];
            odd[j + 1] = odd[j];
            odd[j] = tmp;
        }
    }
}

// 对even冒泡排序
for (i = 0; i < numEven - 1; i++){
    for (j = 0; j < numEven - i - 1; j++){
        if (even[j] > even[j + 1]){
            //tmp为临时变量
            int tmp = even[j + 1];
            even[j + 1] = even[j];
            even[j] = tmp;
        }
    }
}

//输出奇数
for (i = 0; i < numOdd; i++){
    cout << odd[i] << " ";
}

//输出偶数
for (i = 0; i < numEven; i++){
    cout << even[i] << " ";
}

cout << endl;
return 0;

```

```

}

```

```

// 对odd冒泡排序
for (i = 0; i < numOdd - 1; i++){
    for (j = 0; j < numOdd - i - 1; j++){
        if (odd[j] > odd[j + 1]){
            //tmp为临时变量
            int tmp = odd[j + 1];
            odd[j + 1] = odd[j];
            odd[j] = tmp;
        }
    }
}

// 对even冒泡排序
for (i = 0; i < numEven - 1; i++){
    for (j = 0; j < numEven - i - 1; j++){
        if (even[j] > even[j + 1]){
            //tmp为临时变量
            int tmp = even[j + 1];
            even[j + 1] = even[j];
            even[j] = tmp;
        }
    }
}

//输出奇数
for (i = 0; i < numOdd; i++){
    cout << odd[i] << " ";
}

//输出偶数
for (i = 0; i < numEven; i++){
    cout << even[i] << " ";
}

cout << endl;
return 0;

```

```

}

```



现实问题

解决方案

计算机程序



探求程序的本质

■ 软件

- ◆ 软件是现实世界在计算机系统映射；
- ◆ 软件 = 程序 + 文档
 - 程序是对计算任务的处理对象和处理规则的描述。

■ 探求程序的本质

- ◆ 程序是现实世界的解决方案在计算机系统映射；

软件开发的最终目的

■ 程序的开发

- ◆ 对所处理的问题域进行正确的理解
- ◆ 把这种正确的理解正确的描述出来

