2-2 "全副武装"——应用性补充

解决编程问题的一般方法/模拟算法

1. 在本章节主要在于讲述编程的中的思考方式和一些常用的技巧。
2. 模拟算法是一种通过模拟真实场景或系统行为来解决问题的算法。用一句老话说，就是“照着葫芦画瓢”，根据题目表述进行筛选提取关键要素，按需求书写代码解决实际问题。
3. 实际上，所有的编程问题都是广义上的模拟算法，不过当其使用其他特定的算法设计思路或者编程技术时，我们不以模拟算法命名
4. 而如果其他思想和技术不明确，则称其为模拟算法（我们暂时遇到的问题都可以归类到模拟算法中去）。

5. 面对编程问题时，我们如果觉得无从下手，可以考虑从以下步骤思考如何解决：

分析问题，理解题意

1. 搞清楚题目目的，问两个问题
    a. “求的是什么”【题目含义】
    b. “考的是什么”【考察内容】
2. 搞清楚题目输入输出，问三个问题
    a. “输入输出是什么”【数据含义】
    b. “输入输出是怎么样的”【数据形式】
    c. “如何存储输入输出"【数据存储】

设计算法，编写代码

1. 朴素模拟法：对于题目描述的情景进行“起点—>终点”的模拟。采用“先……，再……”的逻辑
2. 逆向工作法:对于题目描述的情景进行“起点<—终点”的模拟。采用“要……，就要……”的逻辑。这个目标如果能够直接实现，就用C/C++实现，不行就继续分解。
3. **思考的时候执果索因，做的时候由因导果**

调试、编译、运行、测试

1. 算法竞赛中，选手按照要求编写完程序后，统一提交并进行评测。测评的过程如下：在测评结束后，测试点通常有以下几种情况：
   a. AC：Accepted，意为答案正确。
   b. WA：Wrong Answer，意为答案错误。即选手的程序和标准输出的答案不一致。
   c. TLE：Time Limit Exceeded，意为超出时间限制。最常见的原因是算法复杂度不够优或者死循环
   d. RE：Runtime Error，意为运行时错误。最经常出现的原因是数组越界。
   e. MLE：Memory Limit Exceeded，意为超出内存限制。最有可能的原因是因为开了非常大的数组。
   f. CE：Compile Error，意为编译错误。**初学阶段较易出现这种错误，应尝试去阅读编译错误信息并进行更正**

分析循环问题的一般方法/循环分析法

面对循环问题时，我们如果觉得无从下手，可以考虑从以下步骤思考如何解决：

这是一个什么循环/不同类型循环语句的应用场景

1. 知道起点终点【区间循环】：for
2. 知道循环次数【次数循环】：for
3. 只知道甚至不知道循环结束/继续的条件【条件循环】：while

循环变量的分析（“跑步的人”）

1. 区间循环
   a. 起点（“从哪来开始跑”）：第一个可以进入循环的数是?

b. 终点（“跑到哪里结束”）：最后一个可以进入循环的数是？第一个不可以进入循环的数是？

c. 循环条件（“什么情况下继续跑”）：[第一个可以进入循环的数，最后一个可以进入循环的数]

d. 逼近操作/靠近操作（起点如何跑向终点）

2. 次数循环：for(int i=0;i<n;i++)或者for(int i=1;i<=n;i++)可以循环n次
3. 条件循环：确定循环结束/继续的条件

循环体的分析

1. 第一种情况:循环体不用到循环变量。循环变量只是控制循环的次数
2. 第二种情况:循环体需要用到循环变量。循环变量还需要参与循环体的操作

下面是使用循环分析法分析问题的例子：B3839,求和问题。

常用技巧

取数位

在我们编程过程中，常常遇到需要将数的各位拿出来单独判断的情况，这就是我们的取数位问题：

1. 取数位的核心思想：使用取余%获取数位，使用除法/删除不需要的位

📌【取数位的核心思想】

12345 =1×10000+2×1000+3×100+4×10+5×1

=12340+5

=1234*10+5

因此，12345%10可以得到5，12345/10得到1234。同理，1234%10得到4，1234/10得到123……

也可以通过1234/1000得到1，得到1234/100%10=2（1234/100=12，12%10=2……）

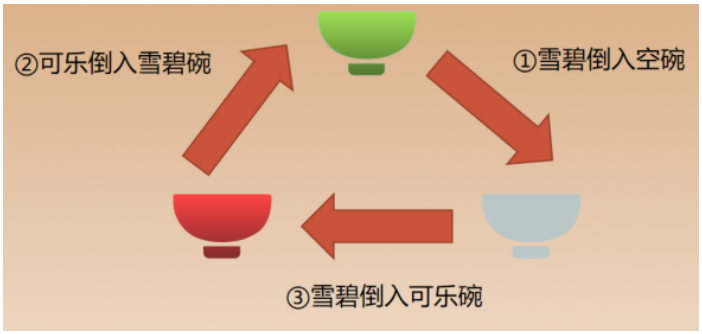
1. 数的位数固定（一直是3/4/5……位）——B2028 反向输出一个三位数
2. B2028，反向输出一个三位数、B2082数字统计、B3699

两个变量的值交换、三个变量地排序与求最值

1. 两个变量的值交换

“设给你一杯可乐，一杯雪碧，如何把这两杯饮料交换？”我们通常会采用这样的方法：准备第三个碗，然后进行下面三个步骤。

1. 将第一个碗的东西放进第三个碗
2. 然后将第二个碗的东西放入第一个碗
3. 最后把第三个碗的东西放回第二个碗



2. 交换两个变量的值也是如此，需要引入一个临时变量tmp（临时的temporary缩写），作为“第三个碗”，代码如下，也可以使用自带的swap函数实现：：

代码块

```
int a,b,tmp;
cin>>a>>b;
tmp=a;
a=b;
b=tmp;
```

代码块

```
int a,b;
cin>>a>>b;
swap(a,b);//swap是英文交换，实现两个参数的交换，没有返回值
```

3. 三个变量的排序——P5715 三位数排序

如果需要对三个变量a、b、c进行排序（以从小到大为例），我们需要进行如下操作：

1. 比较a和b的值：如果a>b，表示a和b当前的顺序不对，交换a和b；如果a<=b，表示a和b当前的顺序正确，不动
2. 比较c和b的值（此时b中存放a和b中较大的值）：如果c>=b，表示c和b当前的顺序正确，不动；如果c<b，表示c和b当前的顺序不对，交换c和b的值。
3. 比较a和b的值（此时c中存放了a、b、c中最大的那个值，a和b存放另外两个）：如果a>b，表示a和b当前的顺序不对，交换a和b；如果a<=b，表示a和b当前的顺序正确，不动。

代码块

```
#include<bits/stdc++.h>
using namespace std;
int main(){
    int a,b,c;
    cin>>a>>b>>c;
    if(a>b)
        swap(a,b);
    if(b>c)
        swap(b,c);
    if(a>b)
        swap(a,b);

    cout<<a<<" "<<b<<" "<<c;

    return 0;
}
```

4. 三个变量求最值——B2049 最大数输出

   a. 在我们完成对三个变量的从小到大排序后，实际上也已经实现了对a、b、c求最值的过程。

b. 其中a是最小值，c是最大值。

c. 但如果变量数增加，交换变量的过程会变得复杂，如果我们只需要最值，可以使用“打擂法”直接求最值：

d. 当有abcd四个变量时排序过程如下：a-b，b-c，c-d。a-b，b-c。a-b。

5. 💡打擂台求解

1. 打擂法求最值的思想源自现实生活中的擂台赛：
2. 选手依次和擂主比赛，选手若赢，成为新擂主（旧擂主退场）
3. 下一位选手挑战新擂主；选手若输，选手退场，下一位选手再挑战擂主。

1. 实现的代码如下：
2. 其中，INT_MIN是常数，与之对应的还有常数INT_MAX。INT_MAX是int能存的最大值，INT_MIN是int能存的最小值。
3. C++还有自带的判断两个变量哪个更大/小的函数max与min，因此上面的代码等价于：

打擂台

```
#include<bits/stdc++.h>
using namespace std;
int main(){
    int a,b,c;
    cin>>a>>b>>c;
    int max=INT_MIN;
    if(a>max)
        max=a;
    if(b>max)
        max=b;
    if(c>max)
        max=c;
    cout<<max;

    return 0;
}
```

代码块

```
#include<bits/stdc++.h>
using namespace std;
int main(){
    int a,b,c;
    cin>>a>>b>>c;
    int maxx=INT_MIN;
    maxx=max(maxx,a);//max(a,b)返回a和b中较大的那个
    maxx=max(maxx,b);
    maxx=max(maxx,c);
    cout<<maxx;

    return 0;
}
```

图形打印

在学习完输出输入后，我们已经学会如何来打印最简单的图形了。但是更复杂的情况我们需要通过循环来控制图形打印的过程。一般的方法是在纸上画出方格，然后标出几行几列的标号（类似后面的二维数组），并需要关注以下关键点：

1. 使用一个for循环控制需要打印几行i
2. 使用一个for循环控制每行打印几列j
3. 使用一个变量控制输出什么内容x
4. i、j、x之间是什么关系。“第i行第j列输出的是……”
5. 处理是否存在第几行第几列需要特殊处理
6. 见题库练习

简单数学思维

统一单位思想

1. 假设我们需要计算两个时间点之间的时间间隔。时间A为14:30，时间B为16:45。我们希望得到它们之间的时间间隔，以分钟为单位。
2. 时间是以小时和分钟为单位表示的。为了计算时间间隔，**我们需要将时间转换为统一的单位**，例如分钟。将两个时间点转换为从午夜（00:00）开始经过的总分钟数，然后计算它们的差值。
3. 计算得到的差值就是时间间隔，以分钟为单位表示。
4. 计算时间A的总分钟数：时间A的小时和分钟分别是14小时和30分钟。时间A的总分钟数 = 14 * 60 + 30 = 840 + 30 = 870分钟。计算时间B的总分钟数：时间B的小时和分钟分别是16小时和45分钟。时间B的总分钟数 = 16 * 60 + 45 = 960 + 45 = 1005分钟。时间间隔 = 时间B的总分钟数 - 时间A的总分钟数 = 1005分钟 - 870分钟 = 135分钟。
5. B3838

基础数列问题

1. 数列
    a. 数列是一列有序的数。用{a(n)}表示。数列中的每一个数都叫做这个数列的项。排在第一位的数，称为这个数列的第1项（通常也叫做首项），通常用a(1)表示，排在第二位的数，称为这个数列的第2项，通常用a(2)表示……排在第n位的数，称为这个数列的第n项，通常用a(n)表示。
    b. 例如{a(n)}=1，2，3，4，5，6，7。则这是一个七个数组成的数列，第一项是1，第二项是2……
2. 特殊的数列类别
    a. **项数有限的数列为“有穷数列”；项数无限的数列为“无穷数列”**

b. 从第2项起，每一项都大于它的前一项的数列叫做递增数列；每一项都小于它的前一项的数列叫做递减数列

c. 其它较复杂的数列类型此处暂不讲解，均以数列实际规律为准

3. **数列的公式**
4. **数列的第n项a(n)与项的序数n之间**，如果存在着计算的关系，**这种关系我们称之为通项公式**。有些数列的通项公式可以有不同形式，即不唯一；有些数列没有通项公式。
5. **数列的第n项a(n)与前一项或者几项之间**，如果存在着计算的关系，**这种关系我们称之为递推公式**。有些数列的递推公式可以有不同形式，即不唯一；有些数列没有递推公式；有递推公式也不一定有通项公式。
6. 数列的前n项求和也通常存在公式，这种关系我们称之为前n项和公式。下面介绍两个特殊的数列的通项公式和前n项和公式。
7. **等差数列**

a. 如果一个数列从第2项起，每一项与它的前一项的差等于同一个常数，这个数列就叫做**等差数列**，这个常数叫做**等差数列的公差**，公差通常用字母d表示。

b. 等差数列的通项公式和递推公式如下：a(n)=a1+(n-1)d=a(n-1)+d

【著名的高斯公式】

等差数列的前n项和公式的推导过程如下：其中a即a1

公式证明如下：

将等差数列和写作以下两种形式：

$$S_n = a + (a+d) + (a+2d) + \cdots + [a+(n-2)d] + [a+(n-1)d]$$
$$S_n = [a_n-(n-1)d] + [a_n-(n-2)d] + \cdots + (a_n-2d) + (a_n-d) + a_n$$

将两公式相加来消掉公差 d，可得

$$2S_n = n(a+a_n)$$

整理可得第一种形式。

代入 $a_n = a+(n-1)d$，可得第二种及第三种形式。

证明高斯求和

```cpp
#include<bits/stdc++.h>
using namespace std;
int main(){
    int a1,a2;
    cin>>a1>>a2;
    int d=a2-a1;//公差

    int n;
    cin>>n;
    int an;//第n项
    for(int i=1;i<n;i++)//累加公差
        an=an+d;
    an=a1+(n-1)*d;//通项公式

    int sn=0;//前n项和
    for(int i=a1;i<=an;i=i+d)//累加第i项
        sn=sn+i;
    sn=(a1+an)*n/2;//前n项和公式
```

```
19        return 0;
20    }
```

8. **等比数列**
9. 如果一个数列从第2项起，每一项与它的前一项的比等于同一个常数，这个数列就叫做**等比数列**。这个常数叫做等比数列的公比），公比通常用字母q表示。
    a. 等比数列的通项公式和递推公式如下：a(n)=a1*q(n-1)=a(n-1)*q。

等比数列的前n项和公式的推导过程如下：其中a即a1，r即q

公式证明如下：

将等比数列和写作以下形式：

$$S_n = a + ar + ar^2 + \cdots + ar^{n-1} \quad \text{.....(1)}$$

将两边同乘以公比 r，有：

$$rS_n = ar + ar^2 + \cdots + ar^n \quad \text{.....(2)}$$

(1)式减去(2)式，有：

$$(1-r)S_n = a - ar^n$$

当$r \neq 1$时，整理后得证。

当$r = 1$时，可以发现：

$$\begin{aligned} S_n &= a + ar + ar^2 + \cdots + ar^{n-1} \\ &= \underbrace{a + a + a + \cdots + a}_{n} \\ &= n \times a \\ &= an \end{aligned}$$

综上所述，等比数列的求和公式为：

$$S_n = \begin{cases} \frac{a(1-r^n)}{1-r} & r \neq 1 \\ an & r = 1 \end{cases}$$

代码块

```
#include <bits/stdc++.h>
using namespace std;

int main() {
    int a1, a2;
    cin >> a1 >> a2;
    int q = a2 / a1; // 公比

    int n;
    cin >> n;
    int an = 1; // 第n项
    for (int i = 1; i < n; i++) // 累乘公比
        an = an * q;
    // an = a1 * pow(q, n - 1); // 通项公式

    int sn = 0; // 前n项和
    // 累加第i项
    for (int i = a1; i <= an; i = i * q)
        sn = sn + i;
 // sn = a1 * ((1 - pow(q, n)) / (1 - q)); // 前n项和公式

    cout << an << endl << sn;
    return 0;
}
```