# 1.本书讨论的内容：

## 1.1问题一：一组N个数要确定其中第k个最大者。

①将所有数据排序后返回位置k上的元素，②把前k个元素排序，将剩下的元素逐个读入，和第k个元素比较、交换。连个算法都不算好。

#include <iostream>

#include <vector>

#include <string>

#include <memory>

#include<algorithm>

using namespace std;

void BubbleSort(int num[],int n)

{

for(int i = 0;i < n - 1;i++)//因为最后一遍扫描后自动生产最小的，不用扫最后一个

for(int j = 0;j < n - i - 1;j++)

if(num[j] > num[j + 1]) swap(num[j],num[j + 1]);

}

void fun1(const int &k)

{

int num[150],i = 0;

while(cin>>num[i++]);

BubbleSort(num,i);

cout<<num[k - 1]<<endl;

}

void fun2(const int &k)

{

int num[150],i = 0;

while(cin>>num[i++]);

BubbleSort(num,k);

for(int j = k + 1;j < i;j++)

if(num[j] > num[k])

swap(num[k],num[j]);

cout<<num[k]<<endl;

}

int main()

{

int k;

cin>>k;

fun1(k);

fun2(k);

return 0;

}

## 1.2问题二：流行字谜，找出二维字谜中的单词，可以是水平的、垂直的或沿对角线放置的。

①对于单词表中的每个单词，检查每一个有序的三元组（行、列、方向），验证是否有单词存在，②对于每一个尚未进行到字谜最后的有序四元组（行、列、方向、字符数），我们可以测试所指的单词是否在单词表中。

//这种带边界的最操蛋了

#include<iostream>

#include<math.h>

#include<stdlib.h>

#include<stdio.h>

#include<string.h>

using namespace std;

int integer(double before)

{

return (int)before;

}

double decimal(double before)

{

return fabs(before - (int)before);//注意这里是fabs

}

void PrintInteger(int num)

{

if(abs(num) >= 10) PrintInteger(num/10);

if(num < 0 && num > -10)

cout<<num%10;

else cout<<abs(num%10);

}

void PrintDecimal(double num,int bitt)

{

cout<<".";

for(int i = 0;i < bitt;i++)

{

num \*= 10;

cout<<integer(num);

num = num - integer(num);

}

}

int bit(double num)

{

char str[20];

sprintf(str,"%f",num);

return strlen(str) - 2;

}

int main()

{

double num;

cin>>num;

int inte = integer(num);

double deci = decimal(num);

PrintInteger(inte);

if(deci >= 0.0001)

PrintDecimal(deci,bit(deci));

return 0;

}

# 2.数学知识复习：

## 2.1指数：

## 2.2对数：

① ② ③ ④ ⑤ ⑥

## 2.3级数：

2.3.1等比级数（等比数列）：

（等比数列求和）。①如果0<A<1，则，②当N趋于，则趋于，证明方法在书上。

2.3.2算术级数（等差数列）：

①, ②，③(k时可用)

1的平方+2的平方+...+N的平方=n(n+1)(2n+1)/6

1的三次方+2的三次方+...+N的三次方=n^2(n+1)^2/4

1的四次方+2的四次方+...+N的四次方=n(n+1)(2n+1)(3n^2+3n-1)/30

1的五次方+2的五次方+...+N的五次方=n^2(n+1)^2(2n^2+2n-1)/12

2.3.3调和数：

,误差趋向于，该值被称为欧拉常数。

## 2.4模运算：

如果N整除A-B，那么A与B模N同余，记为。

## 2.5证明方法：

证明数据结构分析中的结论的两个最常用的方法是归纳法和反证法。数学归纳法：①证基准情形，②进行归纳假设。假设第k个成立证明第k+1个成立。

证明存在无穷多个素数：

# 3.递归简论：

大多数数学函数是由一个简单公式描述的，有时候数学函数以不太标准的形式来定义，不是所有的数学递归函数都能有效地由C的递归模拟来实现。

F(x) = 2F(x-1)+x\*x的递归（F(1) = 1）：

#include<iostream>

using namespace std;

int f(const int &x)

{

if(x == 1) return 1;//基准情况

else return 2\*f(x - 1) + x \* x;//不断推进

}

int main()

{

int x;cin>>x;

cout<<f(x);

return 0;

}

递归不是循环逻辑，虽然我们定义一个函数用的是这个函数本身，但是我们并没有用函数本身定义该函数的一个特定的实例。通过F(5)来调用F(5)是循环的，通过F(4)得到F(5)不是循环的。

跟踪挂起的函数调用（这些调用已经开始但是正等待着递归调用来完成）以及它们中变量的记录工作都是由计算机自动完成的，调用将反复进行直到基准情形出现。

设计递归程序时，同一问题的所有较小实例均可以假设运行正确，递归程序只需要把这些较小的问题的解结合起来而形成现行问题的解。

打印输出数：

#include<iostream>

using namespace std;

void fun(int x)

{

if(x >= 10) fun(x/10);//每次调用打印输出的下一位

cout<<(x%10);//避免mod操作：N%10=N-└N/10┘\*10;

}

int main()

{

int n;cin>>n;

fun(n);

return 0;

}

用反证法证明：见课本。

递归的四条基本法则：

①基准情形：必须有某些基准情形，它无需递归就能解出

②不断推进：每一次递归调用都必须要使求解状况朝接近基准情形的方向推进

③设计法则：假设所有的递归调用都能运行，不用试图追踪大量的递归调用，递归的主要问题是隐含的薄记开销。

④合法效益法则：在求解一个问题的同一实例时，切勿在不同的递归调用中做重复性工作。