《算法基础课程设计》指导书

《算法基础课程设计》课程组 编著

上海海洋大学海洋智能信息实验教学示范中心

实验一 温故知新

一、实验目的

回顾在数据结构及程序设计课程的学习内容，熟练掌握：

1、分析问题中数据的逻辑结构，选择数据的存储结构，设计算法

2、编写和调试程序

二、实验环境

1、C、C++、Java、Python 语言均可作为算法实现语言。

2、PTA：在线评测系统

三、实验内容

1.求 n 个数中最大数

知识点：数组、循环

2.对 n 个数进行升序排序

知识点：排序算法（冒泡、选择、插入、归并、快排、堆）、多重循环。并且尝试将冒泡改为递归形式。

3.求两个数的最大公约数（两种方法）

知识点：最大公约数的概念、欧几里得算法（辗转相除法）

4.求解 n!、斐波那契序列和 Hanio（汉诺塔）问题

知识点：递归

四、实验步骤

1. 在开发环境（以 C 语言的 CodeBlocks 为例）中，新建项目 test

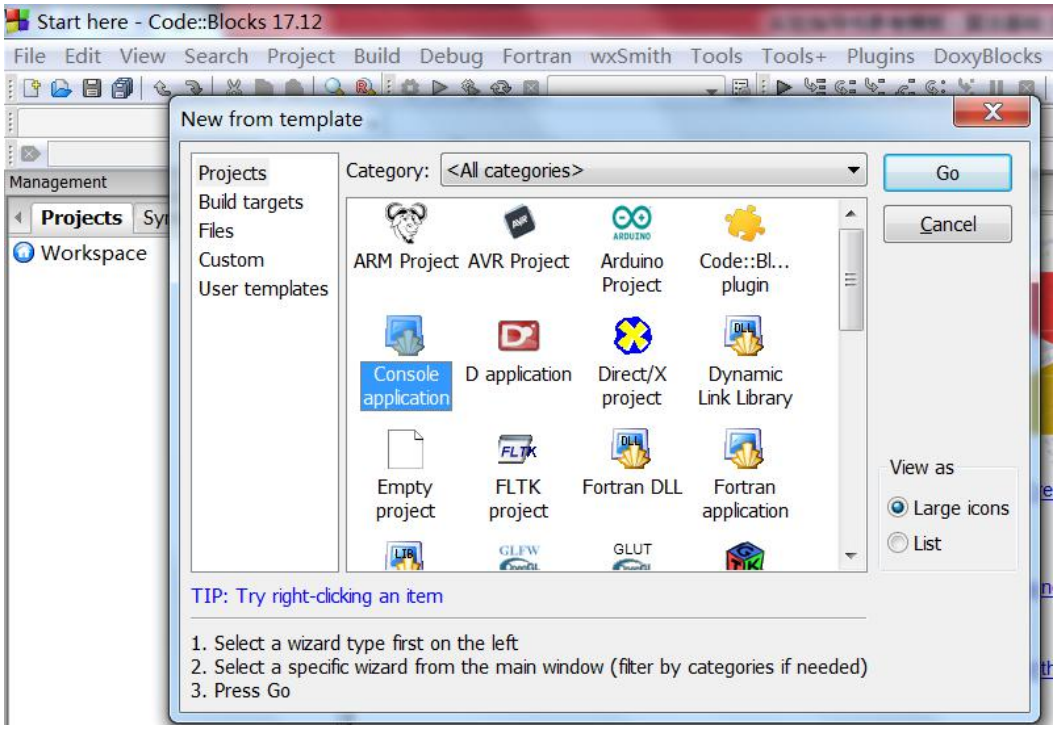

图 1 CodeBlocks 新建项目

2. 输入源代码（以斐波那契序列为例）

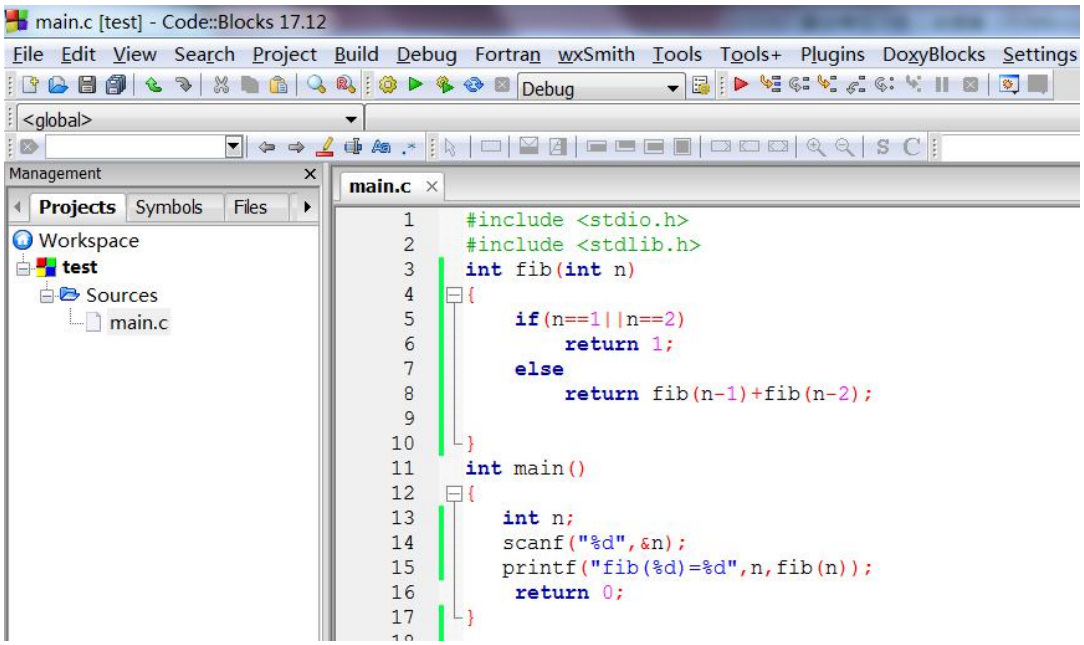

图 2 编写程序

3. 编译运行程序，查看结果

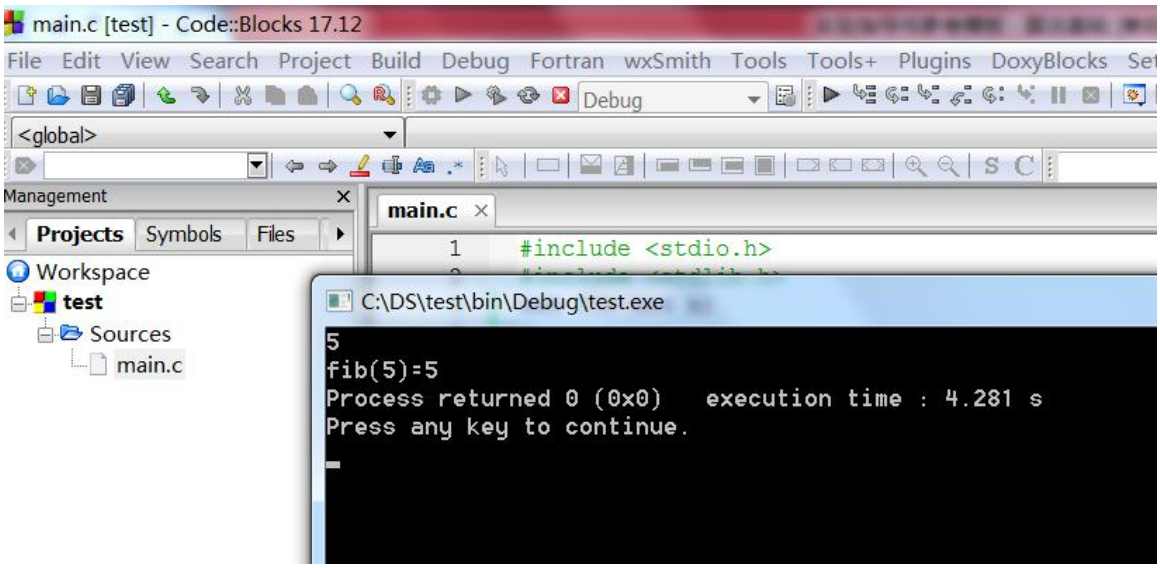

图 3 程序运行结果

五、实验报告要求：

实验报告模板如下（即 附件 1 信息学院实验报告模板）：

实验报告

题目：_____

学 院：信息学院

专 业：

班 级：

学 号：

姓 名：

年 月 日

一、实验目的（宋体四号加粗）

正文（正文 宋体小四，1.5 倍行距）

二、实验环境（宋体四号加粗）

三、实验内容（宋体四号加粗）

四、实验步骤（图文方式叙述）（宋体四号加粗）

五、实验结果及分析（遇到的问题与解决）（宋体四号加粗）

六、实验体会（宋体四号加粗）

实验二 蛮力法

一、实验目的

1、掌握蛮力法的基本思想

2、使用蛮力法解决具体问题（通常，问题规模比较小时，此方法才有意义）

二、实验环境

1、C、C++、Java、Python 语言均可作为算法实现语言。

2、PTA：在线评测系统

三、实验内容

1、蛮力法解决百元百鸡问题、顺序查收、BF、KMP、选择排序、最近点对及凸包问题。可参考教材 C++代码。

2、设计算法，求解问题。教材 P53 习题 3，习题 6 和习题 7。

四、实验步骤

1. 在开发环境（以 C 语言的 CodeBlocks 为例）中，新建项目 test

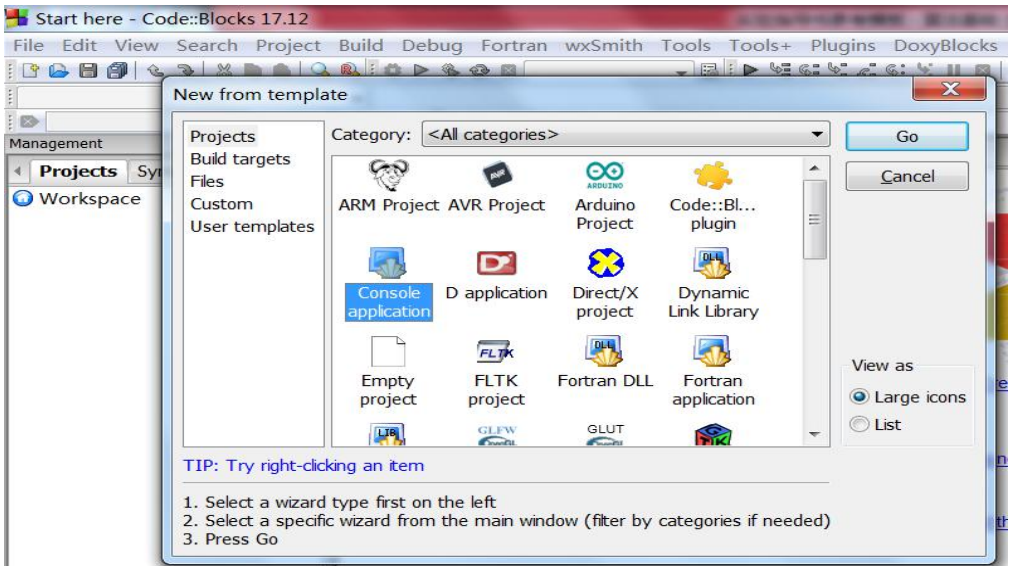

图 1 CodeBlocks 新建项目

2. 输入源代码（以 N 元买 N 鸡为例）

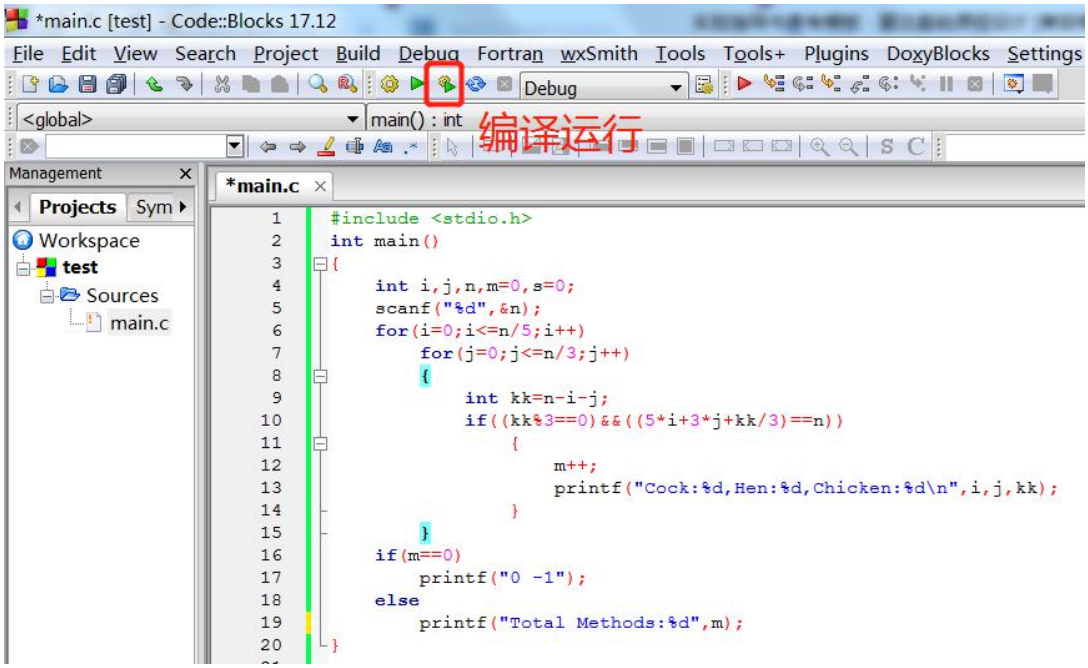

图 2 设计及编写代码算法

3. 编译运行，输出结果

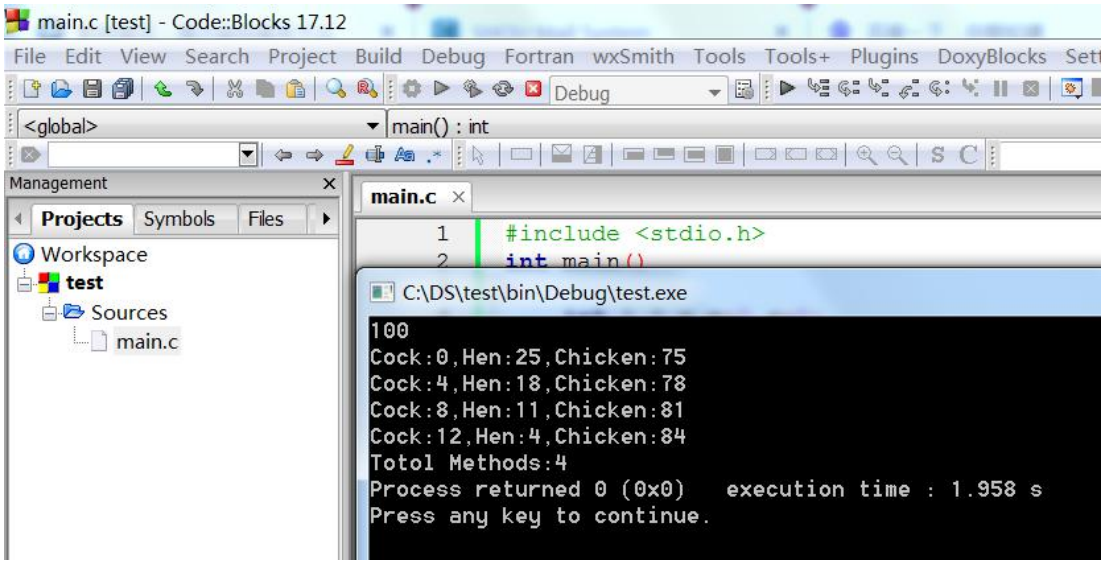

图 3 运行结果

四、实验报告要求

实验报告模板，请参考附件 1 信息学院实验报告

实验三 分治法

一、 实验目的

1、掌握分治法的基本思想

2、使用分治法解决具体问题

二、 实验环境

1、C、C++、Java、Python 语言均可作为算法实现语言。

2、PTA：在线评测系统

三、 实验内容

1、采用分治法的思想，编写程序解决汉诺塔问题 Hanio(n,A,B,C)。

2、分别采用蛮力法和分治法编程计算 a^n。

3、分别采用二路归并、快速排序对序列{23，13，49，6，31，19，28}进行升序

4、设计算法，求解问题。教材 P53 习题 3，习题 6 和习题 7。

四、 实验步骤

1. 在开发环境(以 CodeBlocks 使用 C++语言实现算法为例)中,新建项目 test03，如图 1 CodeBlocks 新建项目 test03。
2. 输入源代码（以快速排序为例），如图 2 设计及编写程序。
3. 编译运行，输入及输出结果，如图 3 运行结果。

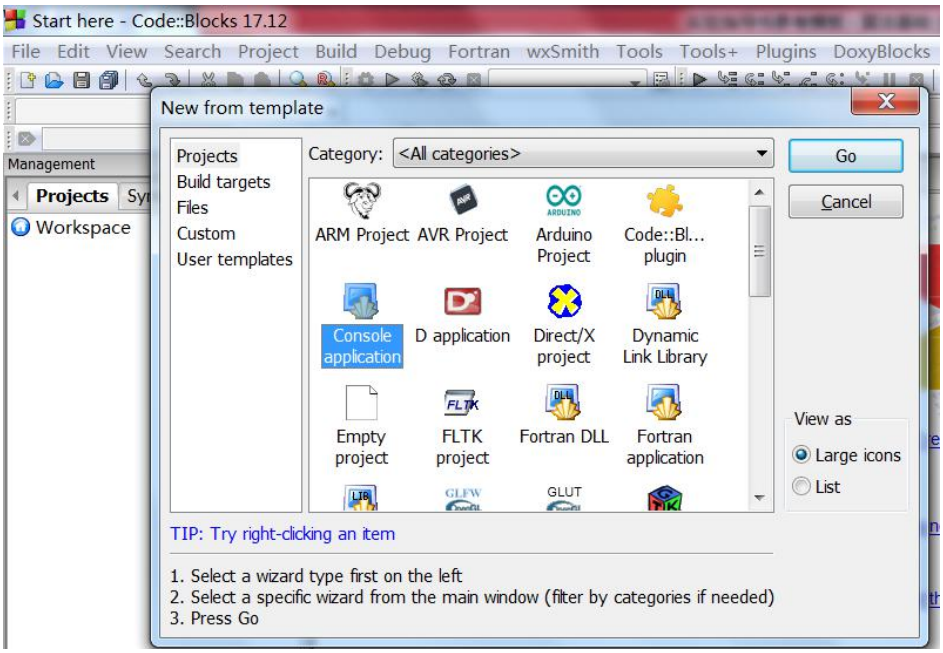

图 1 CodeBlocks 新建项目 test03

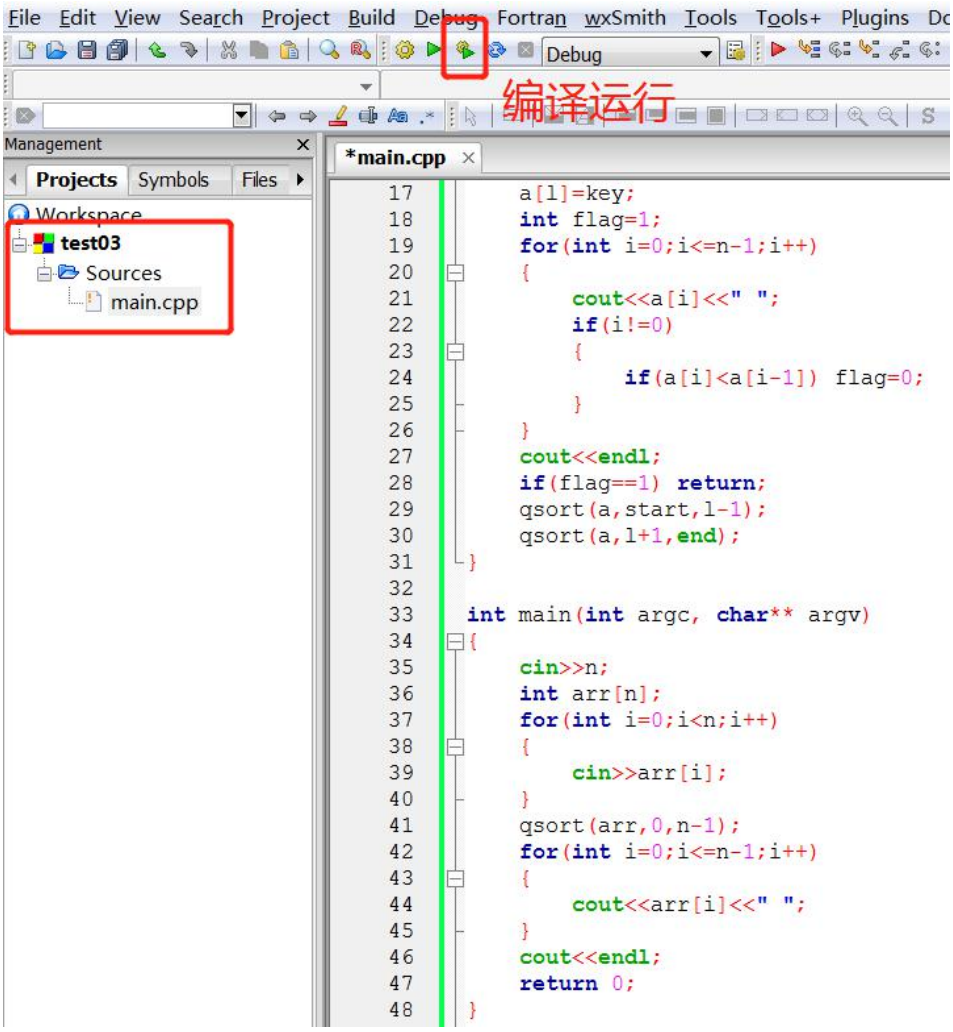

图 2 设计及编写代码程序

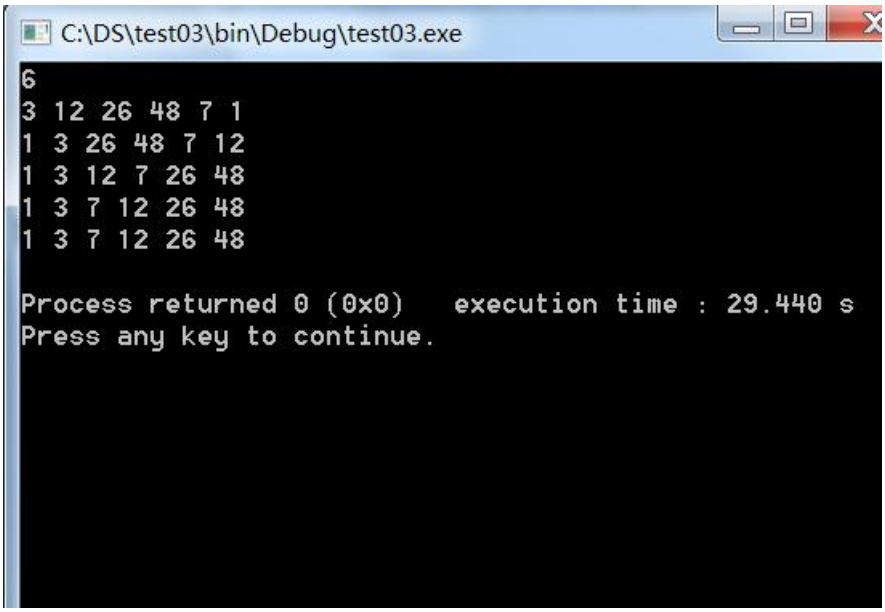

图 3 运行结果

五、实验报告要求

实验报告模板，请参考附件 1 信息学院实验报告

实验四 减治法

一、 实验目的

1、掌握减治法的设计思想

2、使用减治法解决具体问题

二、 实验环境

1、C、C++、Java、Python 语言均可作为算法实现语言。

2、PTA：在线评测系统

三、 实验内容

1、折半查收（二分查找）：利用减治思想，实现在有序序列中查找元素。

1）完成 P94 习题 1；

2）修改为递归算法，完成 P94 习题 2；

3）继续修改，完成 P94 习题 3。

2、堆排序：重点是筛选法调整堆，完成 P94 习题 6

3、查找两个等长有序序列的中位数。

4、计算 a^n

1)蛮力法：n 个 a 相乘

2）分治法：$a^{n/2}*a^{n/2}$

3)减治法：避免重复计算 $a^{n/2}$

4)快速幂：提示将指数 n 转为二进制

参考：https://www.cnblogs.com/nester/p/10878078.html

5、假币问题：选做。解题思路 P93 阅读材料

四、 实验步骤

1. 在开发环境（以 CodeBlocks 使用 C++语言实现算法为例）中，新建项目 test。

2. 输入源代码（以查找两个等长有序序列的中位数为例），代码如下。

```
#include <stdio.h>
#define N 100001
int SearchMid(int a[],int b[],int n)
{    int s1=0,s2=0,e1=n-1,e2=n-1;
     int mid1,mid2;
     while(s1<e1&&s2<e2)
     {
          mid1=(s1+e1)/2;
          mid2=(s2+e2)/2;
          if(a[mid1]==b[mid2]) return a[mid1];
          if(a[mid1]<b[mid2])
           {
               if((s1+e1)%2==0) s1=mid1;
               else
                    s1=mid1+1;
               e2=mid2;
          }
          else
          {
               if((s2+e2)%2==0) s2=mid2;
               else
                    s2=mid2+1;
               e1=mid1;
          }
     }
     if(a[s1]<b[s2]) return a[s1];
     else
          return b[s2];
}
int main()
{
     int n,a[N],b[N],x;
     scanf("%d",&n);
     for(int i=0;i<n;i++)
          scanf("%d",&a[i]);
     for(int i=0;i<n;i++)
          scanf("%d",&b[i]);
     x=SearchMid(a,b,n);
     printf("%d",x);
     return 0;
}
```

3. 编译运行，输入及输出结果，如下图。

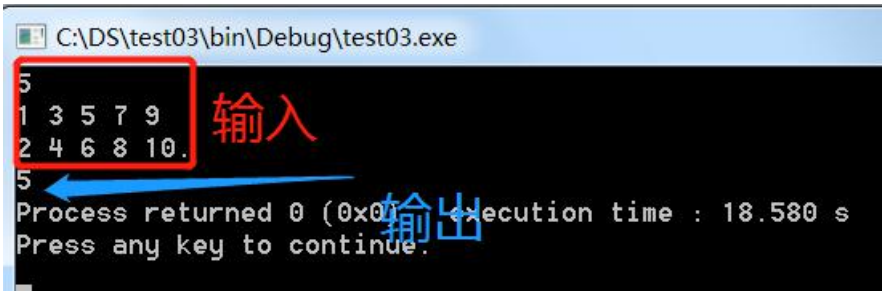

图 1 运行结果

六、实验报告要求

实验报告模板，请参考附件 1 信息学院实验报告

实验五 动态规划法

一、 实验目的

1、掌握动态规划法的设计思想、适用问题、求解步骤。

2、掌握动态规划法在图问题、组合问题和查找问题等中的应用。

二、 实验环境

1、C、C++、Java、Python 语言均可作为算法实现语言。

2、PTA：在线评测系统

三、 实验内容

1、使用动态规划法，解决 0/1 背包问题。

2、动态规划法求解最长公共子序列问题

四、 实验步骤

1、在开发环境（以 CodeBlocks 使用 C++语言实现算法为例）中，新建项目 test

2、DP 法求解 0/1 背包问题。

测试用例： 5 个物品，其重量分别是{2, 2, 6, 5, 4}，价值分别为{6, 3, 5, 4, 6}，背包的容量为 10。输出利用动态规划法构建的表格、背包的最大价值以及装入背包的货物编号。

3、设计算法，编写如下代码：

```
#include <stdio.h>
int max(int m,int n)
      {
          return m>n?m:n;
      }
int knapsack(int n, int w[ ], int v[ ],int C)
{      int i,j,value[101][1001];
       for (i=0; i<=n; i++)
            value[i][0]=0;
       for (j=0; j<=C; j++)
            value[0][j]=0;
       for (i=1; i<=n; i++)
          for (j=1; j<=C; j++)
              if (j<w[i])
                    value[i][j]=value[i-1][j];
              else
                  value[i][j]=max(value[i-1][j], value[i-1][j-w[i]]+v[i]);
      return value[n][C];
}
int main()
{
   int n,kv,i,j,w[101],v[101],C;
```

```
    scanf("%d%d",&n,&C);
    for( i=1;i<=n;i++)
        scanf("%d%d",&w[i],&v[i]);
    kv=knapsack(n,w,v,C);
    printf("%d",kv);
    return 0;
}
```

4、编译运行，输入及输出结果，如下图。

图 1 运行结果

七、实验报告要求

实验报告模板，请参考附件 1 信息学院实验报告

实验六 贪心法

一、 实验目的

1、掌握贪心法的设计思想

2、使用贪心法解决背包问题、最小生成树等具体问题

二、 实验环境

1、C、C++、Java、Python 语言均可作为算法实现语言。

2、PTA：在线评测系统

三、 实验内容

1、使用贪心法，解决背包问题。

2、使用贪心法，解决最小生成树问题、活动安排问题等。

四、 实验步骤

1.在开发环境（以 CodeBlocks 使用 C 语言实现算法为例）中，新建项目 test。

2.根据贪心法在背包问题中的贪心策略，解决背包问题：以 3 个物品，其重量分别是{20，30，10}，价值分别为{60，120，50}，背包的容量为 50。输出背包的最大价值以及装入背包的货物情况。

3.设计算法，编写如下代码：

```
#include <stdio.h>
const int n = 3;
int KnapSack(int w[ ], int v[ ], int n, int C);
int main( )
{
    int w[3] = {20, 30, 10}, v[3] = {60, 120, 50};
    int C = 50;
    int value = KnapSack(w, v, 3, C);
    printf("背包获得的最大价值是：%d",value);
    return 0;
```

```
}

int KnapSack(int w[ ], int v[ ], int n, int C)
{
    int i;
    double x[10] = {0};                 //物品可部分装入
    int maxValue = 0;
    for (int i = 0; w[i] < C; i++)
    {
        x[i] = 1;                       //将物品 i 装入背包
        maxValue += v[i];
        C = C - w[i];                   //背包剩余容量
    }
    x[i] = (double)C/w[i];              //物品 i 装入一部分
    maxValue += x[i] * v[i];
    return maxValue;                    //返回背包获得的价值
}
```

4、编译运行，输入及输出结果，如下图。

图 1 运行结果

五、实验报告要求

实验报告模板，请参考附件 1 信息学院实验报告

实验七 回溯法

一、实验目的

1、掌握回溯法（蛮力法+深度优先搜索）的设计思想

2、使用回溯法解决图着色问题、八皇后等具体问题

二、实验环境

1、C、C++、Java、Python 语言均可作为算法实现语言。

2、PTA：在线评测系统

三、实验内容

采用回溯法的思想，求解图着色问题、4 皇后问题和 0/1 背包问题

四、实验步骤

1.在开发环境(以 CodeBlocks 使用 C++语言实现算法为例)中,新建项目 test03。

2.根据回溯法，求解图着色问题。具体图如下：

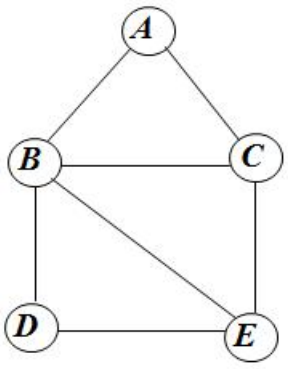

图 1 3 着色问题的图

3.设计算法，编写如下代码：

```
#include <stdio.h>
const int n = 5;
int arc[n][n]={{0,1,1,0,0},{1,0,1,1,1},{1,1,0,0,1},{0,1,0,0,1},{0,1,1,1,0}};
int color[n] = {0};
```

```
void GraphColor(int m);
int Ok(int k);
int main()
{
    GraphColor(3) ;
    return 0;
}
void GraphColor(int m)
{   int i, k;
    for (i = 0; i < n; i++ )                          //将数组 color[n]初始化为 0
        color[i] = 0;
    k = 0;
    while (k >= 0)
    {
        color[k] = color[k] + 1;
        while (color[k] <= m)
            if (Ok(k)) break;
            else color[k] = color[k] + 1;                  //搜索下一个颜色
        if (color[k] <= m && k == n - 1) {             //求解完毕，输出解
            for (i = 0; i < n; i++)
                printf("%d ",color[i]);
            return;
        }
        else if (color[k] <= m && k < n - 1)
                k = k + 1;                        //处理下一个顶点
            else {
                color[k] = 0; k = k - 1;       //回溯
            }
    }
}
int Ok(int k)     //判断顶点 k 的着色是否发生冲突
{
    for (int i = 0; i < k; i++)
        if (arc[k][i] == 1 && color[i] == color[k]) return 0;
    return 1;
}
```

4、编译运行，输入及输出结果，如下图。

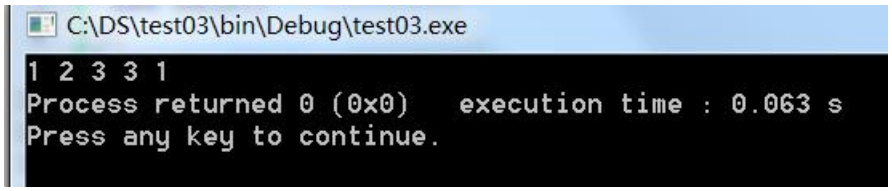

图 1 运行结果

五、实验报告要求

实验报告模板，请参考附件 1 信息学院实验报告

实验八 分支限界法

一、实验目的

1、理解分支限界法（蛮力法+广度优先搜索）的设计思想

2、解决分支限界法的 3 个关键问题：限界函数的确定、pt 表的构造、解向量各个分量的确定

3、对比分支限界法与回溯法，总结异同点

二、实验环境

1、C、C++、Java、Python 语言均可作为算法实现语言。

2、PTA：在线评测系统

三、实验内容

1、采用分支限界法的思想，求解 0/1 背包问题

2、采用分支限界法的思想，求解 TSP 问题

四、实验步骤

1.在开发环境(以 CodeBlocks 使用 C++语言实现算法为例)中,新建项目 test03。

2. 采用队列式分枝限界法求解 0/1 背包问题的算法

3.设计算法，编写如下代码：

```
#include <stdio.h>
#include <queue>
using namespace std;
#define MAXN 101            //最多可能物品数
//问题表示
int n,W;
int w[MAXN];                //重量，下标 0 不用
int v[MAXN];                //价值，下标 0 不用
//求解结果表示
int maxv=-9999;                     //存放最大价值,初始为最小值
```

```
int bestx[MAXN];                    //存放最优解,全局变量
int total=1;                        //解空间中结点数累计,全局变量
struct NodeType                     //队列中的结点类型
{   int no;                         //结点编号
    int i;                          //当前结点在搜索空间中的层次
    int w;                          //当前结点的总重量
    int v;                          //当前结点的总价值
    int x[MAXN];                    //当前结点包含的解向量
    double ub;                      //上界
};
void bound(NodeType &e)             //计算分枝结点 e 的上界
{
    int i=e.i+1;
    int sumw=e.w;
    double sumv=e.v;
    while ((sumw+w[i]<=W) && i<=n)
    {   sumw+=w[i];                 //计算背包已装入载重
        sumv+=v[i];                 //计算背包已装入价值
        i++;
    }
    if (i<=n)
        e.ub=sumv+(W-sumw)*v[i]/w[i];
    else
        e.ub=sumv;
}
void EnQueue(NodeType e,queue<NodeType> &qu)    //结点 e 进队 qu
{
    if (e.i==n)                     //到达叶子结点
    {
        if (e.v>maxv)               //找到更大价值的解
        {
            maxv=e.v;
            for (int j=1;j<=n;j++)
                bestx[j]=e.x[j];
        }
    }
    else qu.push(e);            //非叶子结点进队
}
```

```
void bfs()                          //求 0/1 背包的最优解
{
    int j;
    NodeType e,e1,e2;               //定义 3 个结点
    queue<NodeType> qu;                 //定义一个队列
    e.i=0;                          //根结点置初值，其层次计为 0
    e.w=0; e.v=0;
    e.no=total++;
    for (j=1;j<=n;j++)
        e.x[j]=0;
    bound(e);                       //求根结点的上界
    qu.push(e);                     //根结点进队
    while (!qu.empty())             //队不空循环
    {
        e=qu.front(); qu.pop();     //出队结点 e
        if (e.w+w[e.i+1]<=W)        //剪枝：检查左孩子结点
        {
            e1.no=total++;
            e1.i=e.i+1;             //建立左孩子结点
            e1.w=e.w+w[e1.i];
            e1.v=e.v+v[e1.i];
            for (j=1;j<=n;j++)      //复制解向量
                e1.x[j]=e.x[j];
            e1.x[e1.i]=1;
            bound(e1);              //求左孩子结点的上界
            EnQueue(e1,qu);         //左孩子结点进队操作
        }
        e2.no=total++;              //建立右孩子结点
        e2.i=e.i+1;
        e2.w=e.w; e2.v=e.v;
        for (j=1;j<=n;j++)          //复制解向量
            e2.x[j]=e.x[j];
        e2.x[e2.i]=0;
        bound(e2);                  //求右孩子结点的上界
        if (e2.ub>maxv)             //若右孩子结点可行,则进队,否则被剪枝
            EnQueue(e2,qu);
    }
}
int main()
```

```
{
    scanf("%d%d",&n,&W);
    for(int i=1;i<=n;i++)
    {
        scanf("%d%d",&w[i],&v[i]);
    }
    bfs();                          //调用队列式分枝限界法求 0/1 背包问题
    printf("%d",maxv);
}
```

4、编译运行，输入及输出结果，如下图。

图 1 运行结果

五、实验报告要求

实验报告模板，请参考附件 1 信息学院实验报告

实验九 近似算法

一、实验目的

1、掌握近似算法的设计思想

2、使用近似算法求解具体问题

二、实验环境

1、C、C++、Java、Python 语言均可作为算法实现语言。

2、PTA：在线评测系统

三、实验内容

采用近似算法思想，求解圆周率 PI、装箱问题和子集和问题

四、实验步骤

1.根据近似算法，求解圆周率问题。具体图如下：

2.在开发环境(以 CodeBlocks 使用 C++语言实现算法为例)中,新建项目 test11。

3.设计算法，编写如下代码：

```
#include <iostream>
using namespace std;
#include <math.h>
double Pi(double e);

int main()
{
    double e, PI;
    cout<<"请输入精度要求：";
    cin>>e;
    PI = Pi(e);
    cout<<"Pi="<<PI<<endl;                    //输出π值
    return 0;
}
```

```
double Pi(double e)                                    //给定近似的精度要求
{
    int i = 6;                                         // 从正 6 边形开始
    double   b, x = 1;                                 //2b 为正多边形翻倍之前的边长，x 为翻被之后的边长
    do
    {
        b = x/2;                                       //正 6 边形的边长为 1，即 2b=1
        i = i * 2;                                     //正多边形的边数翻倍
        x = sqrt(2-2*sqrt(1.0-b*b));                   //计算圆内接正多边形翻倍后的边长
    } while (i * x - i * b > e);                       //精度达到要求则停止计算
    cout<<"圆的内接多边形的边数是："<<i<<endl;
    return (i * x)/2;
}
```

4、编译运行，输入及输出结果，如下图。

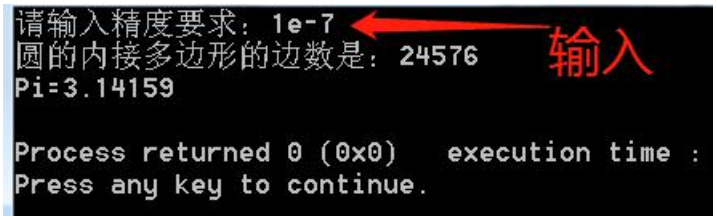

图 1 运行结果

五、实验报告要求

实验报告模板，请参考附件 1 信息学院实验报告

实验十 概率算法

一、实验目的

1、掌握概率算法的设计思想

2、使用概率算法求解具体问题

二、实验环境

1、C、C++、Java、Python 语言均可作为算法实现语言。

2、PTA：在线评测系统

三、实验内容

采用概率算法思想，采用舍伍德型概率算法进行快速排序，采用拉斯维加斯型概率算法求解八皇后问题和整数因子划分问题，采用蒙特卡洛型算法求解主元素问题和素数测试问题。

四、实验步骤

1.根据近似算法，求解圆周率问题。具体图如下：

2.在开发环境（以 CodeBlocks 使用 C++语言实现算法为例）中，新建项目 test。

3.设计算法，编写如下代码：

```
#include <iostream>
#include <stdlib.h>
using namespace std;
const int n = 10;
int Random(int a, int b);
int FermatPrime(int n);

int main()
{
    int x, flag;
    cout<<"请输入一个整数：";
    cin>>x;
```

```
    flag = FermatPrime(x);
    if (flag)
        cout<<x<<"是素数"<<endl;
    else
        cout<<x<<"一定不是素数"<<endl;
    return 0;
}

int FermatPrime(int n)
{
    int a, b = 1;
    a = Random(2, n-1);                    //产生[2, n-1]之间的一个随机整数
    for (int i = 1; i < n; i++)            //计算 an-1 mod n
        b = (b * a) % n;
    if (b == 1) return 1;                  //可能是素数或 Carmichael 数
    else return 0;                         //一定不是素数
}

int Random(int a, int b)
{
    return (rand( )%(b-a) + a);            //rand( )产生[0, 32767]之间的随机整数
}
```

4、编译运行，输入及输出结果，如下图。

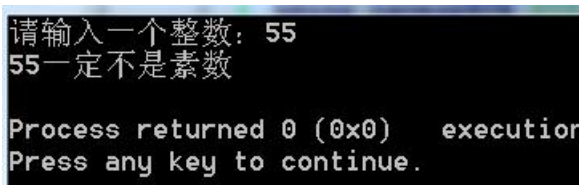

图 1 运行结果

五、实验报告要求

实验报告模板，请参考附件 1 信息学院实验报告