百鸡问题：

C++

//百鸡问题

#include "stdafx.h"

#include<iostream>

using namespace std;

int \_tmain(int argc, \_TCHAR\* argv[])

{

int a,b;

double c;

for(a=0;a<20;a++){

for(b=0;b<(100-a\*5)/3;b++){

c = (100-a\*5-b\*3)\*3;

if((a+b+c)==100){

cout<<"鸡翁："<<a<<"鸡母："<<b<<"鸡雏："<<c<<endl;

}

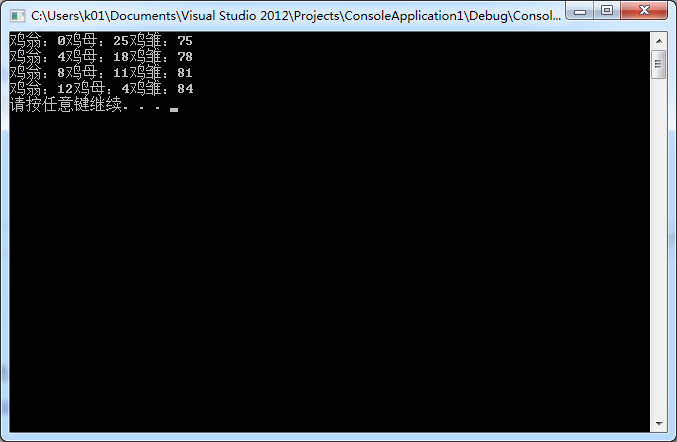
}

}

system("Pause");

return 0;

}



最大公约数（非递归）

C++：

//最大公约数（非递归）

#include "stdafx.h"

#include<iostream>

using namespace std;

int \_tmain(int argc, \_TCHAR\* argv[])

{

int a,b,c;

cin>>a>>b;

if(a<b){

c=a;

a=b;

b=c;

}

while (true)

{

if(a%b==0){

cout<<b;

break;

}

c = a%b;

a=b;

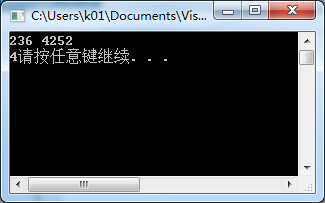
b=c;

}

system("Pause");

return 0;

}



最大公约数（递归）

C++：

//最大公约数（递归）

#include "stdafx.h"

#include<iostream>

using namespace std;

void gcd(int a,int b){

int c;

if(a<b){

c=a;

a=b;

b=c;

}

if(a%b==0){

cout<<b;

return;

}

gcd(b,a%b);

}

int \_tmain(int argc, \_TCHAR\* argv[])

{

int a,b,c;

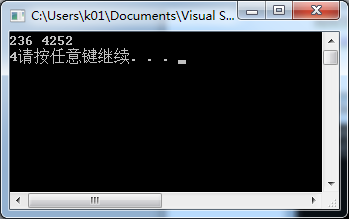
cin>>a>>b;

gcd(a,b);

system("Pause");

return 0;

}



斐波那契数列（非递归）

C++：

//斐波那契数列（非递归）

#include "stdafx.h"

#include<iostream>

using namespace std;

void F(int n){

if(n==1){

cout<<1;

return;

}

if(n==2){

cout<<2;

return;

}

int a=1,b=1,c;

for(int i=1;i<n;i++){

c=a+b;

a=b;

b=c;

}

cout<<c;

}

int \_tmain(int argc, \_TCHAR\* argv[])

{

int n;

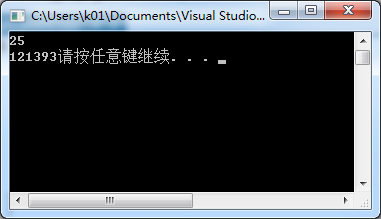
cin>>n;

F(n);

system("Pause");

return 0;

}



斐波那契数列（递归）

C++：

//斐波那契数列（递归）

#include "stdafx.h"

#include<iostream>

using namespace std;

int F(int n){

if(n==1){

return 1;

}

if(n==2){

return 2;

}

return F(n-1)+F(n-2);

}

int \_tmain(int argc, \_TCHAR\* argv[])

{

int n;

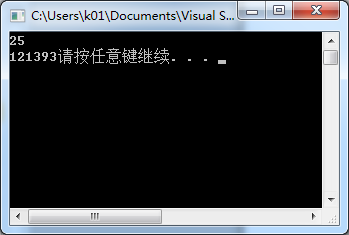
cin>>n;

cout<<F(n);

system("Pause");

return 0;

}



汉诺塔问题（递归）

C++：

//汉诺塔问题

#include "stdafx.h"

#include<iostream>

using namespace std;

void hanoi(int n,char from,char mid,char to){

if(n==1)

cout<<"将第"<<n<<"个从"<<from<<"移到"<<to<<endl;

else{

hanoi(n-1,from,to,mid);

cout<<"将第"<<n<<"个从"<<from<<"移到"<<to<<endl;

hanoi(n-1,mid,from,to);

}

}

int \_tmain(int argc, \_TCHAR\* argv[])

{

int n;

cin>>n;

char from = 'A';

char mid = 'B';

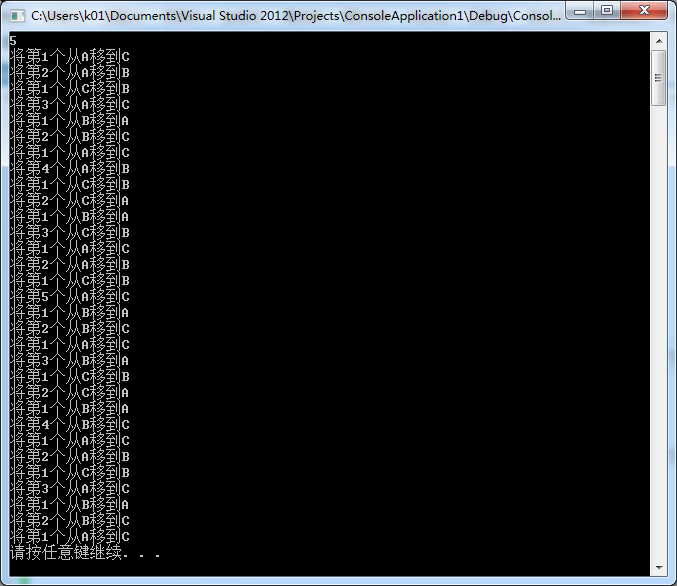
char to = 'C';

hanoi(n,from,mid,to);

system("Pause");

return 0;

}



递归快排+二分法查找

C++：

#include "stdafx.h";

#include <iostream>

using namespace std;

//分治法二分查找

int BinarySearch(int \*A,int n,int k)

{

int start = 0, last = n - 1,mid;

while(start<=last)

{

mid = (int)(start + last) / 2;

if(A[mid]<k)

{

start = mid+1;

}else if(A[mid]>k)

{

last = mid-1;

}else

{

return mid;

}

}

return -1;

}

//分治法快速排序

void Partition(int A[],int start,int end)

{

int i = start, j = end;

int tmp;

if(start<end)

{

tmp = A[start];

while (i!=j)

{

while (j>i && A[j]>=tmp)

{

j--;

}

A[i] = A[j];

while (i<j && A[i]<=tmp )

{

i++;

}

A[j] = A[i];

}

A[i] = tmp;

//左区间递归排序

Partition(A, start, i - 1);

//右区间递归排序

Partition(A, i + 1, end);

}

}

int main()

{

int search;

cin >> search;

int a[20];

for(int i=0;i<20;i++)

{

cin >> a[i];

}

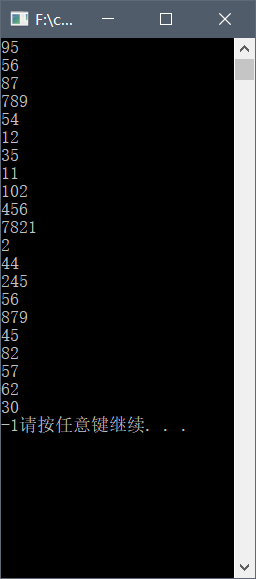
Partition(a, 0, 19);

cout << BinarySearch(a, 20, search);

system("Pause");

return 0;

}



堆排序

C++：

#include "stdafx.h"

#include <iostream>

using namespace std;

void sift(int A[],int low,int high)

{

int i = low;

int j = 2 \* i;

int tmp = A[i];

while (j <= high)

{

if (j+1 <= high && A[j] < A[j + 1])

j++;

if(tmp < A[j])

{

A[i] = A[j];

i = j;

j = 2 \* i;

}else

{

break;

}

A[i] = tmp;

}

}

void HeapSort(int A[],int n)

{

int i,tmp;

for (i = n / 2 ;i >= 1;i--)

sift(A, i, n);

for(i = n;i >= 2;i--)

{

tmp = A[1];

A[1] = A[i];

A[i] = tmp;

sift(A, 1, i - 1);

}

}

int main()

{

int N,A[5000];

cin >> N;

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

{

int num;

cin >> num;

A[i] = num;

}

HeapSort(A, N);

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

{

cout << A[i] << endl;

}

system("Pause");

return 0;

}

