/\*求n的阶乘\*/

#include <iostream>

using namespace std;

int f(int n)

{

if(n==1)

return 1;

else

return n\*f(n-1);

}

int main()

{

int n;

cin>>n;

cout<<f(n)<<endl;

return 0;

}

#include <iostream>

using namespace std;

int main()

{

int n,sum=1;

cin>>n;

for(int i=n;i>=1;i--)

sum\*=i;

cout<<sum<<endl;

return 0;

}

/\*斐波那契第n项的值\*/

#include <iostream>

using namespace std;

int main()

{

int n;

long long a=1,b=1,c=1;

cin>>n;

for(int i=3;i<=n;i++)

{

c=a+b;

a=b;

b=c;

}

cout<<c<<endl;

return 0;

}

#include<iostream>

using namespace std;

int f(long long n)

{

if(n==1||n==2)

return 1;

else

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

}

int main()

{

long long n;

cin>>n;

cout<<f(n);

return 0;

}

#include <iostream>

using namespace std;

int main()

{

int i;

int f[20]={1,1};

for(i=2;i<20;i++)

f[i]=f[i-1]+f[i-2];

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

{

if (i%5==0)//五个一排

cout<<endl;

cout<<" "<<f[i];

}

return 0;

}

/\*p12Ackerman\*/

#include<iostream>

using namespace std;

int Ackerman(int n,int m)

{

if(n==1&&m==0)

return 2;

if(n==0&&m>=0)

return 1;

if(n>=2&&m==0)

return n+2;

if(n>=1&&m>=1)

returnAckerman(Ackerman(n-1,m),m-1);

}

int main()

{

cout<<Ackerman(3,3);

cout<<Ackerman(4,3);

return 0;

}

/\*整数划分\*/

#include <iostream>

using namespace std;

int q(int n,int m)

{

if((n<1)||(m<1))

return 0;

if(n==1||m==1)

return 1;

if(n<m)

return q(n,n);

if(n==m)

return q(n,m-1)+1;

if(n>m)

return q(n,m-1)+q(n-m,m);

}

int main()

{

int n,m;

cin>>n;

m=n;

cout<<q(n,m)<<"种划分";

return 0;

}

/\*递归求数组累加和\*/

#include<iostream>

using namespace std;

int s(int a[],int n)

{

if(n==0)

return 0;

else

return s(a,n-1)+a[n-1];

}

int main()

{

int a[]={1,2,3,4,5,6,7,8,9,10}

cout<<s(a,10)<<endl;

return 0;

}

/\*递归求乘法表\*/

#include <iostream>

using namespace std;

void table(int i)

{

if(i==1)

cout<<endl;

else

table(i-1);

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

cout<<j<<"\*"<<i<<"="<<j\*i<<" ";

cout<<endl;

}

int main()

{

int i;

cin>>i;

table(i);

return 0;

}

/\*用循环和递归编写一个打印下面的星号\*/

#include <iostream>

using namespace std;

void table(int i)

{

if(i==1)

cout<<endl;

else

table(i-1);

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

cout<<"\*"<<" ";

cout<<endl;

}

int main()

{

int i;

cin>>i;

table(i);

return 0;

}

#include <iostream>

using namespace std;

int main()

{

int i,j=1;

for(j=1;j<=9;j++)

{

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

{

cout<<"\*"<<" ";

}

cout<<endl;

}

return 0;

}

/\*求数组中的最大值\*/

#include <iostream>

using namespace std;

int max(int A[],int l,int r)

{

if(r==l)

return A[r];

else

{

int a,b;

a=A[l];

b=max(A,l+1,r);

if(a>b)

return a;

else

return b;

}

}

int main()

{

int a[]={77,49};

for(int i=0;i<=1;i++)

cout<<a[i]<<" ";

cout<<endl;

cout<<max(a,0,1)<<endl;

return 0;

}

/\*求全排列\*/

#include <iostream>

using namespace std;

void Swap(int &a,int &b)

{

int temp=a;

a=b;

b=temp;

}

void Perm(int list[],int k,int n)

{

int i;

if(k==n)

{

for(i=0; i<=n; i++)

cout<<" "<<list[i];

cout<<endl;

}

else

for(i=k; i<=n; i++) {

Swap(list[i],list[k]);

Perm(list,k+1,n);

Swap(list[i],list[k]);

}

}

int main()

{

int n=0,i,list[10];

cout<<"请输入需要全排列的数: "<<endl;

cin>>n;

for(i=0;i<n;i++)

list[i]=i+1;

Perm(list,0,n-1);

}

/\*大整数乘法\*/

#include<iostream>

#include<cmath>

#include<cstring>

using namespace std;

long mult(long x,long y,int n)

{

long a,b,c,d,s;

if(n=1)

return x\*y;

else

{

a=long(x/pow(10,n/2));//取x的左半边

b=long(x%long(pow(10,n/2)));//取x的右半部分

c=long(y/pow(10,n/2));//取y的左半部分

d=long(y%long(pow(10,n/2)));//取y的右半部分

s=mult(a,c,n/2)\*pow(2,n)+(mult((a-b),(d-c),n/2)+mult(a,c,n/2)+mult(b,d,n/2))\*pow(2,n/2)+mult(b,d,n/2);//书上的公式

return s;

}

}

int main()

{

int x,y,n;

cout<"输入位数和两个数";

cin>>n>>x>>y;

cout<<mult(x,y,n);

return 0;

}

/\*二分搜索找数\*/

#include <iostream>

using namespace std;

int BinarySearch(int a[],int first,int last,int k)

{

int index;

if(last<first)

index=-1;// 未找到x

else

{

int mid=(first+last)/2;

if(k==a[mid])

index=mid;

else

if(k<a[mid]) index=BinarySearch(a,first,mid-1,k);

else

index=BinarySearch(a,mid+1,last,k);

return index;

}

}

int main()

{

int array[]={0, 1, 2, 7, 13,19};

int m;

m=BinarySearch(array,0,5,13);

cout<<"m ="<<m<<endl;

}

/\*二分搜索找数递归\*/

#include <iostream>

using namespace std;

int BinarySearch(int a[], int first,int last, int k)

{

int index;

if(last<first)

index=-1;// 未找到x

else

{

int mid=(first+last)/2;

if(k==a[mid])

index=mid;

else

if(k<a[mid])

index=BinarySearch(a,first,mid-1,k);

else

index=BinarySearch(a,mid+1,last,k);

return index;

}

}

int main()

{

int array[] = {0, 1, 2, 7, 13,19};

int m;

m=BinarySearch(array,0,5,13);

cout<<"m ="<<m<<endl;

}

/\*分治最小元素\*/

#include <iostream>

using namespace std;

int min(int a,int b)

{

if(a<b)

return a;

else

return b;

}

int searchmin(int s[],int left,int right)

{

int min1,min2,middle; if(left==right||(left+1)==right)

{

min1=s[left];

min2=s[right];

return min(min1,min2);

}

else

{ middle=(left+right)/2; min1=searchmin(s,left,middle);

min2=searchmin(s,middle+1,right);

return (min(min1,min2));

}

}

int main()

{

int i,num,n,z,min;

int a[100];

cout<<"输入数字个数:"<<endl;

cin>>n;

cout<<"输入数字:"<<endl;

for(i=0;i<n;i++)

{

cin>>num;

a[i]=num;

}

min=searchmin(a,0,n);

cout<<"最小值是:";

cout<<min<<endl;

return 0;

}

/\*分治最小数选择排序\*/

#include <iostream>

using namespace std;

void SelectSort(int a[],int s,int n)

{

if(s+1>n)//递归的出口

return;

else

{

int temp,i,min;

min=s;

for(i=s+1;i<n;i++)

{

if(a[min]>a[i])

min=i;

}

temp=a[s];

a[s]=a[min];

a[min]=temp;

SelectSort(a,s+1,n);//递归找下个

}

}

int main()

{

int j,n,a[200],i,b[200];

cout<<"输入数组个数:"<<endl;

cin>>n;

cout<<"输入数组的数:"<<endl;

for(i=0;i<n;i++)

cin>>a[i];

for(i=0;i<n;i++)

b[i]=a[i];

SelectSort(a,0,n);

for(i=0;i<n;i++)

{

if(b[i]==a[0])

{

j=i;

break;

}

}

cout<<"最小的数下标为:"<<j<<endl;

return 0;

}

/\*快速排序\*/

#include <iostream>

#include <stdlib.h>

#include <time.h>

using namespace std;

template <class Type>

void Swap(Type &x,Type &y)

{

Type temp = x;

x = y;

y = temp; }

int Random(int x, int y)

{

srand((unsigned)time(0));

int ran\_num = rand() % (y - x) + x;

return ran\_num;

}

template <class Type>

int Partition(Type a[],int p,int r)

{

int i = p,j = r + 1;

Type x = a[p];

while(true)

{

while(a[++i]<x && i<r);

while(a[--j]>x);

if(i>=j) break;

Swap(a[i],a[j]);

}

a[p] = a[j];

a[j] = x;

return j;

}

template<class Type>

intRandomizedPartition(Type a[],int p,int r)

{

int i = Random(p,r);

Swap(a[i],a[p]);

return Partition(a,p,r);

}

template <class Type>

voidRandomizedQuickSort(Type a[],int p,int r)

{

if(p<r)

{ int q = RandomizedPartition(a,p,r);

RandomizedQuickSort(a,p,q-1);

RandomizedQuickSort(a,q+1,r);

}

}

int main()

{

int a[10];

a[0]=15;a[1]=2;a[2]=4;a[3]=8;a[4]=3;a[5]=7;a[6]=1;a[7]=5;a[8]=6;a[9]=2;

RandomizedQuickSort(a,0,9);

for(int i=0; i<10; i++)

cout<<a[i]<<" ";

cout<<endl;

return 0;

}

/\*合并排序\*/

#include <iostream>

using namespace std;

int const N=10;

//合并

template <class Type>

void Merge(Type c[],Type d[],int l,int m,int r)

{

//合并c[l:m]和c[m+1:r]到d[l:r]

int i=l,j=m+1,k=l;

while((i<=m)&& (j<=r))

if(c[i]<=c[j])

d[k++]=c[i++];

else

d[k++]=c[j++];

if(i>m)

for(int q=j;q<=r;q++)

d[k++]=c[q];

else

for(int q=i;q<=m;q++)

d[k++]=c[q];

}

template <class Type>

void copy(Type \*S, Type \*T, int s, int t)

{

int i;

for (i = s;i<=t; i++)// 拷贝T到S数组

S[i] = T[i];

}

template <class Type>

void MergeSort(Type a[], int left, int right)// 归并排序

{

int i;

Type b[N];

if (left< right)

{

i = (right+ left) / 2;

MergeSort(a, left, i);// 对前半部分进行排序

MergeSort(a, i + 1, right);// 对后半部分进行排序

Merge(a,b, left, i, right);// 合并前后两部分

copy(a,b,left, right);

}

}

int main()

{ int a[N];

a[0]=15;a[1]=2;a[2]=4;a[3]=8;a[4]=3;a[5]=7;a[6]=1;a[7]=5;a[8]=6;a[9]=2;

MergeSort(a,0,9);

for(int i=0;i<10;i++)

cout<<a[i]<<" ";

return 0;

}

/\*找数组中第k小\*/

#include <iostream>

#include <stdlib.h>

#include <time.h>

using namespace std;

template<class Type>

void Swap(Type &x,Type &y)

{

Type temp = x;

x = y;

y = temp;

}

int Random(int x, int y)

{

srand((unsigned)time(0));

int ran\_num = rand() % (y - x) + x;

return ran\_num;

}

template <class Type>

int Partition(Type a[],int p,int r)

{ int i = p,j = r + 1;

Type x = a[p];

while(true)

{

while(a[++i]<x && i<r);

while(a[--j]>x);

if(i>=j) break;

Swap(a[i],a[j]);

}

a[p] = a[j];

a[j] = x;

return j;

}

template<class Type>

int RandomizedPartition(Type a[],int p,int r)

{

int i = Random(p,r);

Swap(a[i],a[p]);

return Partition(a,p,r);

}

template<class Type>

Type RandomizedSelect(Type a[],int p,int r,int k)

{

if (p==r) return a[p];

int i=RandomizedPartition(a,p,r),

j=i-p+1;

if (k<=j)

return RandomizedSelect(a,p,i,k);

else

return RandomizedSelect(a,i+1,r,k-j);

}

int main()

{

int num,n,k;

int a[10];

cout<<"输入数组个数："<<endl;

cin>>n;

cout<<"输入数："<<endl;

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

{

cin>>num;

a[i]=num;

}

cout<<"输入想查第k小数字:"<<endl;

cin>>k;

cout<<RandomizedSelect(a,0,n-1,k);

cout<<endl;

return 0;

}

/\*求最大最小分治\*/

#include <iostream>

using namespace std;

void MaxMin(int a[], int low, int high, int \*pmax,int \*pmin)

{

if(high-low <=1) //为两个元素

{

if(a[low] <= a[high])

{

\*pmax = a[high];

\*pmin = a[low];

}

else

{

\*pmin = a[high];

\*pmax = a[low];

}

}

else

{

int min1, min2, max1, max2;

int mid = (low+high)/2;

MaxMin(a, low, mid, &max1, &min1);

MaxMin(a, mid+1, high, &max2, &min2);

\*pmin = min(min1,min2);

\*pmax = max(max1, max2);

}

}

int main()

{

int a[] = {1, 3, 5, 7, 9, 0, 2, 4, 6, 8};

int max,min;

MaxMin(a, 0, 9, &max, &min);

cout<<"max="<<max<<" "<<"min="<<min<<endl;

return 0;

}

/\*骨牌号\*/

#include <iostream>

using namespace std;

int tile = 1; //L型骨牌的编号

const int Maxnum = 1 << 10; //棋盘的尺寸大小最大为2的10次方

int Board[Maxnum][Maxnum]; //定义一个二维数组表示这个棋

void ChessBoard(int tr,int tc,int dr,int dc,int size) //tr , tc分别代表左上角行号和列号,dr和dc分别代表特殊方块的行号和列号,size是棋盘的尺寸

{

if(size==1)

return;

int t=tile++; //tile是全局变量，代表L型骨牌的编号

int s=size/2;

if(dr<tr+s&&dc<tc+ s) //左上

ChessBoard(tr,tc,dr,dc,s);

else

{

Board[tr+s-1][tc+s-1]=t; //用t号L型骨牌填充右下角

ChessBoard(tr,tc,tr+s-1,tc+s-1,s); //递归

}

if(dr>=tr+s&&dc<tc+s) //左下

ChessBoard(tr+s,tc,dr,dc,s);

else

{

Board[tr+s][tc+s-1]=t; //用t号L型骨牌填充右上角

ChessBoard(tr+s,tc,tr+s,tc+s-1,s); //递归

}

if(dr<tr+s&&dc>=tc+s) //右上

ChessBoard(tr,tc+s,dr,dc,s);

else

{

Board[tr+s-1][tc+s]=t; //用t号L型骨牌填充左下角

ChessBoard(tr,tc+s,tr+s-1,tc+s,s); //递归

}

if(dr>=tr+s&&dc>=tc+s) //右下角

ChessBoard(tr+s,tc+s,dr,dc,s);

else

{

Board[tr+s][tc+s]=t; //用t号L型骨牌填充左上角

ChessBoard(tr+s,tc+s,tr+s,tc+s,s); //递归

}

}

int main()

{

int x, y, size; //x , y分别代表特殊方块的位置,size代表棋盘的大小

cout << "请输入特殊方块的位置(x,y)和棋盘的尺寸大小size: ";

cin >> x >> y >> size;

//特殊方块赋值为0

Board[x][y] = 0;

//分治法填满棋盘

ChessBoard(0, 0, x, y, size);

//输出棋盘

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

{

for (int j = 0; j < size; j++)

{

cout << Board[i][j] << "\t";

}

cout << endl;

}

return 0;

}

/\*合并\*/

#include <iostream>

using namespace std;

void Merge(int a[],int N1, int b[], int N2,int d[])

{

int i=0;

int j=0;

int k=0;

while(i<N1&&j<N2)

{

if(a[i]<b[j])

d[k++]=a[i++];

else

d[k++]=b[j++];

}

while(i<N1)

{

d[k++]=a[i++];

}

while(j<N2)

{

d[k++]=a[j++];

}

int z;

for(z=0;z<8;z++)

cout<<d[z]<<" ";

cout<<endl;

}

int main()

{

int i;

int a[]={1,2,10,12};

int b[]={3,4,7,8};

int d[8];

Merge(a,4,b,4,d);

return 0;

}