补充知识1：

定义结构体使用关键字struct，例如：（通俗的理解：我们定义了一种类型！）

C++ 代码

|  |
| --- |
| struct test //定义一种类型，类型名为test  {      float a;      int b;  }; |

上面的代码就定义了一个名为test的结构体，它的数据类型就是test，它包含两个成员a和b，成员a的数据类型为浮点型，成员b的数据类型为整型。

由于结构体本身就是自定义的数据类型，定义结构体变量的方法和定义普通变量的方法一样。

test pn1; //定义pn1为test类型

这样就定义了一test结构体数据类型的结构体变量pn1，结构体成员的访问通过点操作符进行，

pn1.a=10 就对结构体变量pn1的成员a进行了赋值操作，

注意:结构体声明的时候本身不占用任何内存空间，只有当你用你定义的结构体类型定义结构体变量的时候计算机才会分配内存。

结构体，同样是可以定义指针的，那么结构体指针就叫做结构指针。

结构指针通过->符号来访问成员，下面我们就以上所说的看一个完整的例子：

C++ 代码

|  |  |
| --- | --- |
|  | #include<iostream>  #include<string>  **using** **namespace** std;  **struct** test;//定义一个名为test的结构体  {  **int** a;//定义结构体成员a  **int** b;//定义结构体成员b  };  **int** main()  {      test pn1;//定义结构体变量pn1      test pn2;//定义结构体变量pn2      pn2.a = 10;//通过成员操作符.给结构体变量pn2中的成员a赋值      pn2.b = 3;//通过成员操作符.给结构体变量pn2中的成员b赋值      pn1 = pn2;//把pn2中所有的成员值复制给具有相同结构的结构体变量pn1      cout << pn1.a << "|" << pn1.b << endl;      cout << pn2.a << "|" << pn2.b << endl;      return 0；  } |

补充知识2：

快速排序的使用

C++  qsort函数

头文件：#include <algorithm>

默认的sort函数是按升序排。

sort(begin,end)，表示一个范围，默认从小到大排序！

例如：#include<iostream>

#include<algorithm>

using namespace std;

int main()

{

　　//sort函数第三个参数采用默认从小到大

　　int a[]={45,12,34,77,90,11,2,4,5,55};

　　sort(a+0,a+10);

for(int i=0;i<10;i++)cout<<a[i]<<" ";

}

运行结果：2 4 5 11 12 34 45 55 77 90

若要从大大小？

sort(begin,end，comp)

（1）第一个参数begin：是要排序的数组的起始地址。

（2）第二个参数end：是结束的地址（最后一个数据的后一个数据的地址）

（3）第三个参数comp是排序的方法：可以是从升序也可是降序。如果第三个参数不写，则默认的排序方法是从小到大排序。

#include<iostream>

#include<algorithm>

using namespace std;

bool cmp(int a,int b);

int main(){

　　//sort函数第三个参数自己定义，实现从大到小

　　int a[]={45,12,34,77,90,11,2,4,5,55};

　　sort(a+0,a+10,cmp);

　　for(int i=0;i<10;i++)　cout<<a[i]<<" ";

}

//自定义函数

bool cmp(int a,int b)

{

　　return a>b;

}

又有例子2：

1 #include<iostream>

2 #include<algorithm>

3 #include"cstring"

4 using namespace std;

5 typedef struct student{

6 　　char name[20];

7 　　int math;

8 　　int english;

9 }Student;

10 bool cmp(Student a,Student b);

11 main(){

12 　　//先按math从小到大排序，math相等，按english从大到小排序

13 　　Student a[4]={{"apple",67,89},{"limei",90,56},{"apple",90,99}};

14 　　sort(a+0,a+3,cmp);

15 　　for(int i=0;i<3;i++)

16 　　cout<<a[i].name <<" "<<a[i].math <<" "<<a[i].english <<endl;

17 }

18 bool cmp(Student a,Student b){

19 　　if(a.math >b.math )

20 　　return a.math <b.math ;//按math从小到大排序

21 　　else if(a.math ==b.math )

22 　　　　　　return a.english>b.english ; //math相等，按endlish从大到小排序

23

24 }

递推：

1. 简单模型：
2. 普通数列

1,2,3,4,5,6.

递推式子:a[n]=a[n-1]+1; n>1

递推边界：a[1]=1 n=1

2,4,6,8,10

递推式子:a[n]=a[n-1]+2; n>1

递推边界：a[1]=2 n=1

2,4,8,16,32,64,128,256,512

递推式子:a[n]=a[n-1]\*2; n>1

递推边界：a[1]=2 n=1

1. 斐波那契数列 1,1,2,3,5,8,13,21,34,55

递推式子:a[n]=a[n-1]+a[n-2]; n>2

递推边界：a[1]=a[2]=1 n=1,2

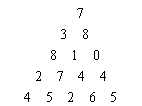
3、倒递推：赶鸭子：一个人赶着鸭子去每个村庄卖，每经过一个村子卖去所赶鸭子的一半又一只。这样他经过了n个村子后还剩x只鸭子，问他出发时共赶多少只鸭子？经过每个村子卖出多少只鸭子？

a[n]=a[n-1]/2-1 所以 a[n-1]=(a[n]+1)\*2

递推式子:a[n-1]=(a[n]+1)\*2; n>0

递推边界：a[n]=x n=n

1. **【例1】数字三角形。如下所示为一个数字三角形。请编一个程序计算从顶到底的某处的一条路径，使该路径所经过的数字总和最大。只要求输出总和。**
2. 1、 一步可沿左斜线向下或右斜线向下走；
3. 2、 三角形行数小于等于100；
4. 3、 三角形中的数字为0，1，…，99；
5. 测试数据通过键盘逐行输入，
6. 如上例数据应以如下所示格式输入：
7. 5
8. 7
9. 3 8
10. 8 1 0
11. 2 7 4 4
12. 4 5 2 6 5
13. **【算法分析】**
14. 此题解法有多种，从递推的思想出发，设想，当从顶层沿某条路径走到第i层向第i+1层前进时，我们的选择一定是沿其下两条可行路径中最大数字的方向前进，为此，我们可以采用倒推的手法，设a[i][j]存放从i,j 出发到达n层的最大值，则a[i][j]=max{a[i][j]+a[i+1][j]，a[i][j]+a[i+1][j+1]}，a[1][1] 即为所求的数字总和的最大值。



【参考程序】

#include<iostream>

using namespace std;

int main()

{

int n,i,j,a[101][101];

cin>>n;

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

for (j=1;j<=i;j++) cin>>a[i][j]; //输入数字三角形的值

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

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

{

if (a[i+1][j]>=a[i+1][j+1]) a[i][j]+=a[i+1][j]; //路径选择

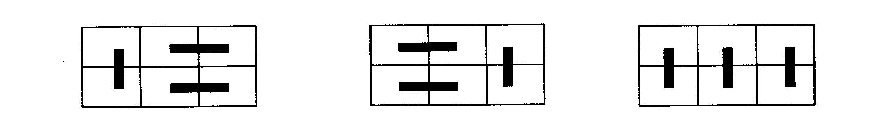
else a[i][j]+=a[i+1][j+1];

}

　 cout<<a[1][1]<<endl;

}

【例2】 有 2χn的一个长方形方格，用一个1\*2的骨牌铺满方格。



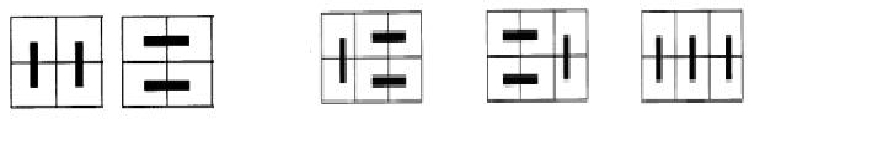
编写一个程序，试对给出的任意一个n(n>0), 输出铺法总数。

【算法分析】

　（1）面对上述问题，如果思考方法不恰当，要想获得问题的解答是相当困难的。可以用递推方法归纳出问题解的一般规律。

　（2）当n=1时，只能是一种铺法，铺法总数有示为x1=1。

　（3）当n=2时：骨牌可以两个并列竖排，也可以并列横排，再无其他方法，如下左图所示，因此，铺法总数表示为x2=2；



（4）当n=3时：骨牌可以全部竖排，也可以认为在方格中已经有一个竖排骨牌，则需要在方格中排列两个横排骨牌（无重复方法），若已经在方格中排列两个横排骨牌，则必须在方格中排列一个竖排骨牌。如上右图，再无其他排列方法，因此铺法总数表示为x3=3。

　　由此可以看出，当n=3时的排列骨牌的方法数是n=1和n=2排列方法数的和。

  （5）推出一般规律：对一般的n，要求xn可以这样来考虑，若第一个骨牌是竖排列放置，剩下有n-1个骨牌需要排列，这时排列方法数为xn-1；若第一个骨牌是横排列，整个方格至少有2个骨牌是横排列（1\*2骨牌），因此剩下n-2个骨牌需要排列，这是骨牌排列方法数为xn-2。从第一骨牌排列方法考虑，只有这两种可能，所以有：

             xn=xn-1+xn-2 （n>2）

             x1=1

             x2=2

xn=xn-1+xn-2就是问题求解的递推公式。任给n都可以从中获得解答。例如n=5，

           x3=x2+x1=3

           x4=x3+x2=5

           x5=x4+x3=8

下面是输入n，输出x1~xn的c++程序：

#include<iostream>

using namespace std;

int main()

{

int n,i,j,a[101];

cout<<"input n:"; //输入骨牌数

cin>>n;

a[1]=1;a[2]=2;

cout<<"x[1]="<<a[1]<<endl;

cout<<"x[2]="<<a[2]<<endl;

for (i=3;i<=n;i++) //递推过程

{

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

cout<<"x["<<i<<"]="<<a[i]<<endl;

　 }

}

  下面是运行程序输入 n=30，输出的结果：

   input n: 30

      x[1]=1

      x[2]=2

      x[3]=3

　　........

      x[29]=832040

      x[30]=1346269

【例4】昆虫繁殖

【问题描述】

科学家在热带森林中发现了一种特殊的昆虫，这种昆虫的繁殖能力很强。每对成虫过x个月产y对卵，每对卵要过两个月长成成虫。假设每个成虫不死，第一个月只有一对成虫，且卵长成成虫后的第一个月不产卵(过X个月产卵)，问过Z个月以后，共有成虫多少对？0=<X<=20,1<=Y<=20,X=<Z<=50

【输入格式】

x,y,z的数值

【输出格式】

过Z个月以后，共有成虫对数

【输入样例】

1 2 8

【输出样例】

37

【参考程序】

#include<iostream>

using namespace std;

int main()

{

long long a[101]={0},b[101]={0},i,j,x,y,z;

cin>>x>>y>>z;

for(i=1;i<=x;i++){a[i]=1;b[i]=0;}

for(i=x+1;i<=z+1;i++) //因为要统计到第z个月后，所以要for到z+1

{

b[i]=y\*a[i-x];

a[i]=a[i-1]+b[i-2];

}

cout<<a[z+1]<<endl;

return 0;

}

1、走楼梯

楼梯有N级台阶，上楼可以一步上一阶，也可以一步上二阶。编一递归程序，计算共有多少种不同走法？

【输入样例】Stairs.in

3

【输出样例】Stairs.out

3

2、兔子繁殖

有一种兔子，出生后一个月就可以长大，然后再过一个月一对长大的兔子就可以生育一对小兔子且以后每个月都能生育一对。现在，我们有一对刚出生的这种兔子，那么，n个月过后，我们会有多少对兔子呢？假设所有的兔子都不会死亡。

【输入格式】

输入文件仅一行，包含一个自然数n。

【输出格式】

输出文件仅一行，包含一个自然数，即n个月后兔子的对数。

【输入样例】Rabbit.in

5

【输出样例】Rabbit.out

5