优雅编码的起始 新知一下

逻辑航线

递归与分治

NOI基础算法系列课程

版本：1.0.0

讲师：孙伟航

01

递归

递归

逻辑航线

递归是很多算法都使用的一种编程方法，是最为重要的编程思想之一。

我们先来通过一个小故事，来对递归做一个初步的了解。

冠军之钥

有一天，上帝给了你们一个巨大的宝箱，并说到，那把能够使你们获得NOI全国冠军的神秘钥匙就藏在里面，你们只需要找到它就可以了。你们兴奋的打开了宝箱，却发现里面竟然又是一堆大小不同的宝箱，甚至有的宝箱里还装着小宝箱。这该怎么办？正在你们一筹莫展的时候，你们的老师——伟大的我出现了，我大手一挥，给出了一个方法：

1、从大宝箱开始。
2、检查宝箱中的每样东西。
3、如果是宝箱就回到第2步。
4、如果是钥匙，就大功告成。

我们可以很容易的把上面的过程，描述成一段精简的代码

```
void lookForKey(Box box){
     for(Item item in box){
          if(item is Box){
                    lookForKey(box); //递归调用
          }
          else{
                    printf("found the key!");
          }
     }
}
```

怎么样，这段代码是不是非常的好理解，递归会使得程序更容易理解。很多算法都使用了递归，所以，我们有必要认真学习。

递归-基线条件和递归条件

由于递归函数是自己调用自己，因此，编写这样的函数时，很容易出错，进而导致无限循环。例如，假设你要编写一个像下面这样倒计时的函数。

3…… 2…… 1

为此，你可以用递归的方式编写，如下图所示：

```
void countDown(int i)
{
      cout << i<<" ";
      countDown(i-1);
}
```

如果你运行上述代码，你将会发现一个问题：这个函数将会持续运行，永不终止！

3…… 2…… 1…… 0 …… -1……

编写程序时，必须告诉他何时停止递归。正因为此，每个递归函数都有两部分：基线条件和递归条件。递归条件指的是函数调用自己，而基线条件指的是函数不再调用自己，从而避免形成无限循环。

现在，我们来给函数**countDown**添加基线条件。

```
void countDown(int i){
      cout << i<<" ";
      if(i<=0){      //基线条件
            return;
      }
      else {         //递归条件
            countDown(i-1);
      }
}
```

现在，函数就运行正常了。

递归-调用栈

本节将介绍一个重要的编程概念——调用栈。

栈是一种后进先出的数据结构，我们可以简单的把栈想象成一摞盘子：后放上去的盘子，会最先被餐厅使用。

计算机在内部是用一种被称为“调用栈”的栈。下面，我们来学习一下计算机是如何使用调用栈的。

示例代码：

```
void greet(string name){
    printf("Hello, %s!\n",name.c_str());
    greet2(name);
    bye(name);
}

void greet2(string name){
    printf("How are you, %s!\n",name.c_str());
}

void bye(string name){
    printf("Bye, %s!\n",name.c_str());
}
```

递归-调用栈

当我们开始调用`greet("Jim")`时，计算机将首先为该函数分配一块内存，并将变量Jim存储到内存中：

greet()	
Name:	Jim

当该函数打印完`"Hello Jim"`时，我们继续调用`greet2("Jim")`,这时，计算机继续在内存中为其分配内存。计算机使用一个栈来存储这些内存快，其中第二个内存位于第一个内存之上。

greet2()	
Name:	Jim
greet()	
Name:	Jim

← 后来者居上

当greet2打印完`"How are you, Jim"` 时，整个函数就执行完毕了，于是该函数被从栈顶弹出。

~~greet2()~~	
~~Name:~~	~~Jim~~

← 被从栈顶弹出，移除。

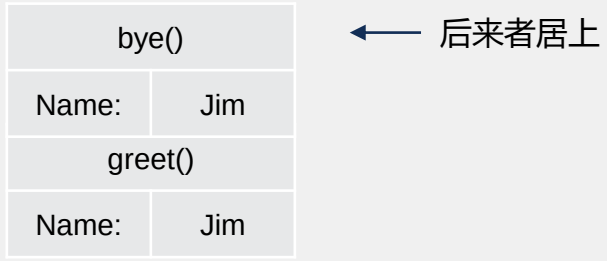
greet()	
Name:	Jim

递归-调用栈

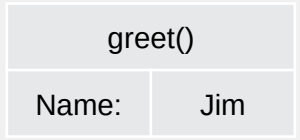
现在，栈顶的内存块是函数**greet**，这意味着你的程序返回了函数**greet**。我们在前面，调用greet2时，函数greet只执行了一部分。

这里有个重要概念：**调用另一个函数时，当前函数暂停并处于未完成状态。**

当前函数所有的信息都依然存储在内存中，执行完函数greet2后，程序又回到了greet，并从离开的地方继续往下执行，即开始调用`bye("Jim")`函数。



于是，当前的greet又停止了下来，bye函数开始执行，打印`"Bye, Jim"`。而后，bye函数执行完毕，被从栈中移除。



函数再一次回到了greet，至此，函数全部执行完毕。

专项练习1-递归

逻辑航线

题目地址：https://www.luogu.com.cn/problem/P5739

解题思路：简单递归

专项练习-递归（AC代码）

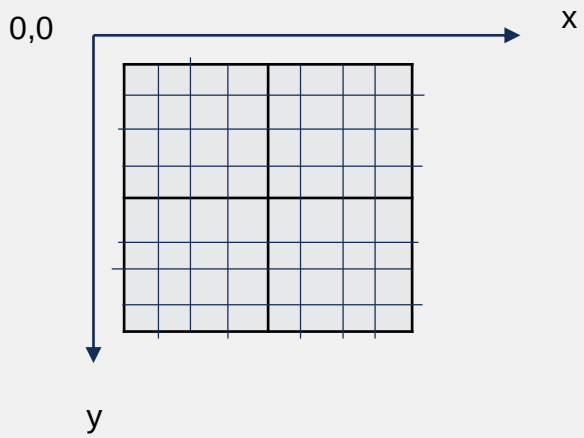
逻辑航线

```
#include <iostream>

using namespace std;

int f(int x)
{
	if(x==1)
	{
		return 1;
	}
	else
	{
		return x*f(x-1);
	}
}

int main(int argc, char** argv)
{
	int n;
	cin>>n;
	cout << f(n) <<endl;
	return 0;
}
```

专项练习2-赦免战俘

题目地址：https://www.luogu.com.cn/problem/P5461

解题思路：递归的基线条件和递归条件



递归条件：length>1。

基线条件：length=1。既不可再分割

专项练习2-赦免战俘（AC代码）

逻辑航线

```cpp
#include <iostream>
#include <cmath>

using namespace std;

int a[1025][1025];

void flag(int x,int y,int length){
    //基线条件，长度为1代表不可再分割。
    if(length==1 ){
        return ;
    }
    //分割长度
    int half = length/2;
    //将左上角的数据标记为可以赦免。
    for(int i=x; i<x + half; i++){
        for(int j=y; j<y+half; j++){
            a[i][j] = 0;
        }
    }
    //递归处理另外三个区域
    flag(x+half, y, half); //右上角
    flag(x, y+half, half); //左下角
    flag(x+half, y+half, half); //左上角
}

int main(int argc, char** argv){
    int n;
    cin>>n;
    int length = pow(2,n);
    //数据初始化，将所有数据标记为不可赦免
    for(int i=0; i<length; i++){
        for(int j=0; j<length; j++){
            a[i][j] = 1;
        }
    }
    //开始递归分析
    flag(0,0,length);
    //输出最终结果
    for(int i=0; i<length; i++){
        for(int j=0; j<length; j++){
            cout << a[i][j] <<" ";
        }
        cout << endl;
    }
    return 0;
}
```

专项练习3-逆波兰表达式

逻辑航线

题目地址：http://ybt.ssoier.cn:8088/problem_show.php?pid=1198

解题思路：计算离一个符号最近的两个数字的值

专项练习3-逆波兰表达式（AC代码）

逻辑航线

```
#include <bits/stdc++.h>

using namespace std;

char str[101];
double hxs;
double exp(){
     scanf("%s",str);
     switch(str[0]){
          case '+':
               hxs = exp() + exp();
               break;
          case '-':
               hxs = exp() - exp();
               break;
          case '*':
               hxs = exp() * exp();
               break;
          case '/':
               hxs = exp() / exp();
               break;
          default:
               //将字符串转为浮点数
               hxs = atof(str);
     }
     return hxs;
}

int main(){
          printf("%f\n",exp());
}
```

专项练习4-斐波那契数列

逻辑航线

题目地址：http://ybt.ssoier.cn:8088/problem_show.php?pid=1201

解题思路：大数递归

斐波那契的第1000项有约1e300，也就是111后面跟着300个0，必须要用高精度

专项练习4-斐波那契数列（AC代码）

逻辑航线

```
#include <iostream>
#include <cstdio>

using namespace std;
int fib(int n){
     if(n==1||n==2){
          return 1;
     }

     return fib(n-1) + fib(n-2);
}
int n,m,tot[2000];

int main(int argc, char** argv)
{
     cin>>n;
     for(int i=0; i<n; i++){
          cin>>m;
          tot[i] = fib(m);
     }
     for(int i=0; i<n; i++){
          cout <<tot[i]<<endl;
     }
     return 0;
}
```

专项练习5-Pell数列

逻辑航线

题目地址：http://ybt.ssoier.cn:8088/problem_show.php?pid=1201

解题思路：简单的递归

专项练习5-Pell数列（AC代码）

逻辑航线

```cpp
#include <iostream>
#include <cstdio>
#define maxN 32767
using namespace std;

int n,m,tot[1000001],input[2000];

int pell(int n){
     if(tot[n] != 0){
          return tot[n];
     }
     if(n==1){
          return tot[1]= 1;
     }
     if(n==2){
          return tot[2]= 2;
     }

     return tot[n] = (2*pell(n-1) + pell(n-2)) % maxN ;
}
```

```cpp
int main(int argc, char** argv){
     cin>>n;
     for(int i=0; i<n; i++){
          cin>>m;
          pell(m);
          input[i] = m;
     }
     for(int i=0; i<n; i++){
          int m = input[i];
          cout <<tot[m]<<endl;
     }
     return 0;
}
```

专项练习6-爬楼梯

题目地址：http://ybt.ssoier.cn:8088/problem_show.php?pid=1204

解题思路：递归思想的应用

当楼梯有一层的时候，只有一种走法。

当楼梯有两层的时候，存在两种走法。

专项练习6-爬楼梯（AC代码）

逻辑航线

```
#include<bits/stdc++.h>

using namespace std;

int calculate(int n)
{
      if(n==1)
            return 1;
      if(n==2)
            return 2;
      return calculate(n-1)+calculate(n-2);
}
int main()
{
      int n;
      while(cin>>n)
      {
            cout<<calculate(n)<<endl;
      }
      return 0;
}
```

专项练习7-最大公约数

逻辑航线

题目地址：http://ybt.ssoier.cn:8088/problem_show.php?pid=1207

解题思路：欧几里得算法

专项练习7-最大公约数（AC代码）

逻辑航线

```cpp
#include <iostream>

using namespace std;

int count(int a,int b)
{
	if(b == 0)
	{
		return a;
	}
	return count(b,a%b);
}



int main(int argc, char** argv)
{
	int a,b;
	cin >> a >> b;
	cout << count(a,b);
	return 0;
}
```

专项练习8-2的幂次方

逻辑航线

题目地址：http://ybt.ssoier.cn:8088/problem_show.php?pid=1208

解题思路：复杂递归调用

每次递归都执行三步：

1、拆数，找出当起数值中包含的2的幂次最大的值

2、表达当前这个最大数值

3、递归调用函数，处理剩下的余数

专项练习8-2的幂次方（AC代码）

逻辑航线

```
void count(int n){
    int num;
    //进行拆数，取出当前2的整数幂
    for(int i=0; i<20; i++){
        if(a[i] > n){
            num = i-1;
            break;
        }
    }
    //处理当前的幂次显示
    cout<<"2";
    switch(num){
        case 0:
            cout<<"(0)";break;
        case 1:
                break;
        case 2:
            cout << "(2)";break;
        default:
            cout << "(";
            count(num);
            cout << ")";
            break;
    }
    //基线条件，处理余数的表示
    int value = n-a[num];
    if(value > 0){
        cout<<"+";
        count(value);
    }
}
```

```
int a[20];

//初始化数据
void initData(){
    a[0] = 1;
    for(int i=1; i<20; i++){
        a[i] = a[i-1] * 2;
    }
}



int main(int argc, char** argv){
    int n;
    initData();
    cin>>n;
    count(n);
    return 0;
}
```

专项练习9-数的计算

题目地址：https://www.luogu.com.cn/problem/P1028

解题思路：记忆化搜索

专项练习9-数的计算（AC代码）

逻辑航线

```
int f[1010];


int Count(int n)
{
     if(f[n] != -1)
     {
          return f[n];
     }
     int ans = 1;
     for(int i=1; i<=n/2; i++)
     {
          ans += Count(i);
     }
     return f[n] = ans;
}




int main(int argc, char** argv)
{
     memset(f,-1,sizeof(f));
     int n;
     f[1] = 1;
     cin >> n;
     cout << Count(n);
     return 0;
}
```

专项练习10-Function

逻辑航线

题目地址：https://www.luogu.com.cn/problem/P1464

解题思路：记忆化搜索

专项练习10-Function（AC代码）

逻辑航线

```
long long f[25][25][25];
long long a,b,c;
long long Cout(long long a,long long b,long long c){
    //条件一、
    if(a<=0 || b<=0 || c<=0){
        return 1;
    }
    //条件二、
    if(a>20 || b>20 || c>20){
        return Cout(20,20,20);
    }
    //检查缓存
    if(f[a][b][c] != 0){
        return f[a][b][c];
    }
    //条件三、
    if(a<b && b<c){
        return f[a][b][c] = Cout(a,b,c-1) +
    Cout(a,b-1,c-1) - Cout(a,b-1,c);
    }
    //条件四、
    else{
        return f[a][b][c] = Cout(a-1,b,c) + Cout(a-
    1,b-1,c) + Cout(a-1,b,c-1) - Cout(a-1,b-1,c-1);
    }
}
```

```
int main(int argc, char** argv)
{

    while(cin>>a>>b>>c)
    {
        if(a==-1 && b==-1 && c==-1)
        {
            break;
        }
        int res = Cout(a,b,c);
        printf("w(%d, %d, %d) = %d\n",a,b,c,res);
    }

    return 0;
}
```

专项练习11-外星密码

题目地址：https://www.luogu.com.cn/problem/P1928

解题思路：复杂递归

专项练习11-外星密码（AC代码）

逻辑航线

```
string Decode()
{
	char temp;
	string s="",x;
	while(cin>>temp)
	{
		switch(temp)
		{
			case '[':
				int d;
				scanf("%d",&d);
				x = Decode();
				for(int i=0; i<d; i++)
				{
					s += x;
				}
				break;
			case ']':
				return s;
				break;
			default:
				s += temp;
				break;
		}
	}
	return s;
}
```

```
int main(int argc, char** argv)
{
	cout << Decode();
	return 0;
}
```

专项练习12-蜜蜂路线

题目地址：https://www.luogu.com.cn/problem/P2437

解题思路：高精度斐波那契

感　谢　观　看

联系地址：河北省廊坊市大厂县孔雀英国宫二期