递归算法

1. 编写把十进制正整数转换为S进制(S=2，8，16)数输出的递归算法。

#include <stdio.h>

//n:原数(10进制) to:将转换成的进制

char**\*** fn**(**int n**,** int to**)**

**{**

//定义静态变量：为了在多层递归里面使用的是同一个值，且static只执行一次

static char buf**[**100**]** **=** ""**;**

//定义进制表

static char BinHex**[]** **=** **{**'0'**,**'1'**,**'2'**,**'3'**,**'4'**,**'5'**,**'6'**,**'7'**,**'8'**,**'9'**,**'A'**,**'B'**,**'C'**,**'D'**,**'E'**,**'F'**};**

//进行n == 0 和 n < 0 的情况处理,static只执行一次

static int i**;**

**if(**n**>=**0**)**

**{**

buf**[**0**]** **=** '0'**;**

i **=** 0**;**

**}**

**else**

**{**

buf**[**0**]** **=** '-'**;**

n **=** **-**n**;**

i **=** 1**;**

**}**

//正整数递归，进行进制转换.

**if** **(**n **!=** 0**)**

**{**

//递归入栈

fn**(**n**/**to**,**to**);**

//递归弹栈

buf**[**i**++]** **=** BinHex**[**n **%** to**];**

**}**

**return** buf**;**

**}**

int main**()**

**{**

char **\***buf **=** **NULL;**

int n**,** to**;**

cout **<<** "Input data to transform:" **<<**"\t"**<<** "Radix to transform to:"**<<**endl**;**

cin**>>** n **>>** to**;**

//进制转换

buf **=** fn**(**n**,**to**);**

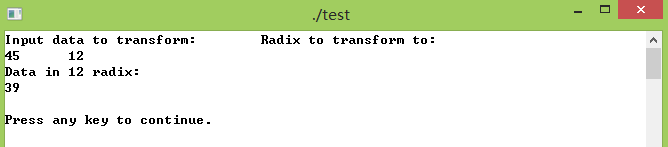
cout **<<** "Data in " **<<** to **<<** " radix:" **<<** endl**;**

puts**(**buf**);**

**return** 0**;**

**}**

测试结果：



1. 编写出计算Fib(n)的递归算法。并分析递归算法和非递归算法它们的时间复杂度和空间复杂度。

分析：斐波那契数列有个规律：从第三个数开始，每个数是前两个数之和，比如：

1 1 2 3 5 8 13 21 34 55 ……

算法描述：

/\*\*

\* 斐波那契数列算法，从第三个数开始，每个数是前两个数之和：1 1 2 3 5 8 13 21 34 55

\* 求第N个数的两种算法，分递归和非递归两种

\*/

#include <iostream>

#include <stdlib.h>

**using** **namespace** std**;**

//递归方式

int f**(**int n**)**

**{**

//参数合法性验证

**if** **(**n **<** 1**)** **{**

cerr **<<** "n must larger than 1 !"**;**

exit**(**1**);**

**}**

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

**return** 1**;**

**else**

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

**}**

//非递归方式

int fx**(**int n**)**

**{**

//参数合法性验证

**if** **(**n **<** 1**)** **{**

cerr **<<** "n must larger than 1 !"**;**

exit**(**1**);**

**}**

//n为1或2时候直接返回值

**if** **(**n **==** 1 **||** n **==** 2**)** **return** 1**;**

//n>2时候循环求值

int result **=** 0**;**

int a **=** 1**;**

int b **=** 1**;**

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

**{**

result **=** a **+** b**;**

a **=** b**;**

b **=** result**;**

**}**

**return** result**;**

**}**

int main**()**

**{**

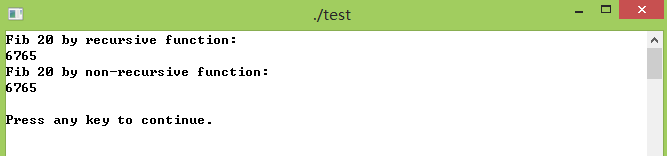
cout **<<** f**(**20**)** **<<** endl**;**

cout **<<** fx**(**20**)** **<<** endl**;**

**return** 0**;**

**}**

测试截图：



递归算法复杂度：

非递归算法复杂度：

1. 杨辉三角形的递归实现。

//杨辉三角递归实现

#include <iostream>

**using** **namespace** std**;**

int fun**(**int**,**int**);**

int main**()**

**{**

int i**,**j**,**k**,**n**;**

cout **<<** "Input line number:" **<<**endl**;**

cin **>>** n**;**

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

**{**

**for(**j**=**0**;** j**<=**2**\***n**-**2**\***i**;** j**++)**

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

**for(**k**=**0**;** k**<=**i**;** k**++)**

cout **<<**" " **<<** fun**(**i**,**k**)** **<<**" "**;**

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

**}**

**return** 0**;**

**}**

int fun**(**int i**,**int j**)**

**{**

**if(**j**==**0**)**

**return** 1**;**

**else**

**return** fun**(**i**,**j**-**1**)\*(**i**-**j**+**1**)/**j**;**

**}**

测试截图：

