信息学课堂笔记10

一、课堂习题

1.用函数实现的计算器(课上代码)

```cpp
#include <bits/stdc++.h>
using namespace std;
bool flag;
double add(double a, double b) {
    return a + b;
}

double sub(double a, double b) {
    return a - b;
}

double pro(double a, double b) {
    return a * b;
}

double chu(double a, double b) {
    //除法中注意除数不能为0!!!!
    if (b == 0) {
        cout << "除数不能为0"<< endl;
        flag = 1;
        return 0;
    }
    else return a / b;
}

double pow1(double a, double b) {
    double ans = 1;
    for (long long i = 0; i < b; i++) {
        ans *= a;
    }
    return ans;
}

long long main() {
    double a, b, ans;
    char c;
    cin >> a >> c >> b;
    if (c == '+') {
        ans = add(a, b);
    } else if (c == '-') {
        ans = sub(a, b);
    } else if (c == '*') {
        ans = pro(a, b);
    } else if (c == '/') {
        ans = chu(a, b);
    } else if (c == '^') {
        ans = pow1(a, b);
```

```
    } else {
        cout <<"Error" <<endl;
    }
    if (!flag) {
        cout << ans << endl;
    }
    return 0;
}
```

2.[B2052]简单计算器

题目描述

复制Markdown 展开

一个最简单的计算器，支持 `+,-,*,/` 四种运算。仅需考虑输入输出为整数的情况，数据和运算结果不会超过 `int` 表示的范围。然而：

1. 如果出现除数为 0 的情况，则输出： `Divided by zero!` 。
2. 如果出现无效的操作符（即不为 `+,-,*,/` 之一），则输出： `Invalid operator!` 。
3. 除号表示整除，结果向 0 取整。

输入格式

输入只有一行，共有三个参数，其中第 $1,2$ 个参数为整数，第 3 个参数为操作符 `(+,-,*,/)` 。

输出格式

输出只有一行，一个整数，为运算结果。然而：

1. 如果出现除数为 0 的情况，则输出： `Divided by zero!` 。
2. 如果出现无效的操作符（即不为 `+,-,*,/` 之一），则输出： `Invalid operator!` 。

输入输出样例

输入 #1	输出 #1
1 2 +	3

输入 #2	输出 #2
2 4 *	8

输入 #3	输出 #3
5 0 /	Divided by zero!

```
#include <bits/stdc++.h>
using namespace std;

bool flag = false;

int add(int a, int b) {
    return a + b;
}

int sub(int a, int b) {
```

```
    return a - b;
}

int pro(int a, int b) {
    return a * b;
}

int chu(int a, int b) {
    if (b == 0) {
        flag = true; //如果除数是0，则把全局变量设置为true
        return 0;
    }
    return a / b;
}

int main() {
    int a, b, ans;
    char c;
    cin >> a >> b >> c;
    if (c == '+') {
        ans = add(a, b);
    } else if (c == '-') {
        ans = sub(a, b);
    } else if (c == '*') {
        ans = pro(a, b);
    } else if (c == '/') {
        ans = chu(a, b);
    } else {
        cout << "Invalid operator!" << endl;
        return 0;  //一定要记得，在判断到符号错误时，将程序退出
    }
    if (flag == 0) {
        cout << ans << endl;
    } else {
        cout << "Divided by zero!" << endl; //当除数为0时，输出Divided by
zero!
    }
    return 0;
}
```

3.[B2064]斐波那契数列（递归做法）

题目描述

复制Markdown 展开

斐波那契数列是指这样的数列：数列的第一个和第二个数都为 1，接下来每个数都等于前面 2 个数之和。

给出一个正整数 a，要求斐波那契数列中第 a 个数是多少。

输入格式

第 1 行是测试数据的组数 n，后面跟着 n 行输入。每组测试数据占 1 行，包括一个正整数 a（$1 \leq a \leq 30$）。

输出格式

输出有 n 行，每行输出对应一个输入。输出应是一个正整数，为斐波那契数列中第 a 个数的大小。

输入输出样例

输入 #1 复制

```
4
5
2
19
1
```

输出 #1 复制

```
5
1
4181
1
```

```cpp
#include <bits/stdc++.h>
using namespace std;
long long fib(long long x) {
    if (x == 1 || x == 2) return 1; //如果当前的x为1或2，则不递归直接return
    return fib(x - 1) + fib(x - 2); //否则return f(x - 1) + f(x - 2);
}
long long main() {
    long long n, x;
    cin >> n;
    while (n--) {  //题目中有n组数据，需要先读入组数n
        cin >> x;
        cout << fib(x) << endl;
    }
    return 0;
}
```

4.[B2077]角谷猜想

题目描述

复制Markdown 展开

所谓角谷猜想，是指对于任意一个正整数，如果是奇数，则乘 3 加 1，如果是偶数，则除以 2，得到的结果再按照上述规则重复处理，最终总能够得到 1。如，假定初始整数为 5，计算过程分别为 16 、 8 、 4 、 2 、 1。

程序要求输入一个整数，将经过处理得到 1 的过程输出来。

输入格式

一个正整数 $N(N \leq 2,000,000)$。

输出格式

从输入整数到 1 的步骤，每一步为一行，每一部中描述计算过程。最后一行输出 `End`。如果输入为 1，直接输出 `End`。

输入输出样例

输入 #1 复制

```
5
```

输出 #1 复制

```
5*3+1=16
16/2=8
8/2=4
4/2=2
2/2=1
End
```

```cpp
#include <bits/stdc++.h>
using namespace std;
void f(long long n) {
    if (n == 1) {
        cout << "End" << endl;
        return;
    }
    if (n % 2 == 1) { //如果是奇数
        cout << n << "*3+1=" << n * 3 + 1 << endl;
        f(n*3+1); //执行下一个函数
    }
    else { //如果是偶数
        cout << n << "/2=" << n / 2 << endl;
        f(n/2);
    }
}
int main() {
    long long n;
    cin >> n;
    f(n);
    return 0;
}
```

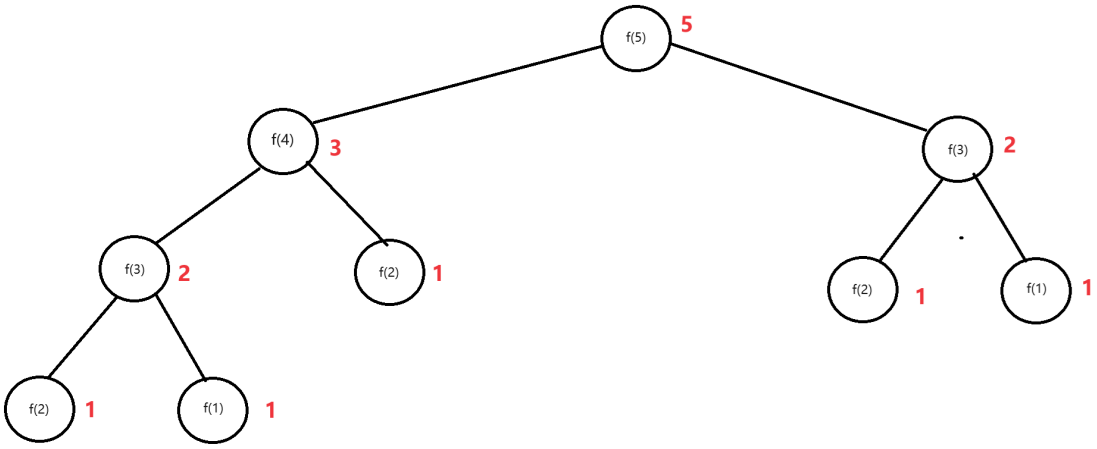
二、 课堂笔记

函数

函数的递归调用

从前有个山，山里有个庙，庙里有个老和尚给小和尚讲故事，讲的是什么呢：从前有个山，山里有个庙.....

> 在编程中，有时我们需要让一个函数调用自己，直到某条件达成为止 ，称之为递归。
>
> 在递归中，我们需要考虑到每次进入函数有哪些不同的情况，根据这些不同的情况让函数做出改变，由上而下的递，由下而上的归。
>
> 就拿课上我们说的斐波那契数列的递归做法举例子，我们可以人为的规定：令f(n) 表示求斐波那契数列的第n项的值
>
> 那无论对于n为任何数时，都有f(n) = f(n - 1) + f(n - 2) 这句公式的含义为斐波那契数列第n项的值等于第n - 1项和第n - 2项的和
>
> 那我们该怎么避免这个递归一直无限走下去呢，那同学们可以发现f(n) = f(n - 1) + f(n - 2)，那么也就说明我们从上往下的时候，总是会碰到**边界**，这个边界就是 n == 1 和n == 2，在这两个情况下我们不会计算f(n) = f(n - 1) + f(n - 2)，因为没有第0项和第-1项，所以我们就称这个为递归边界，一旦递归走到了1或2的时候，就自动返回网上归，这就像是迷宫里的思路，你从一条路走到黑，走到什么时候回头？一个死胡同的时候。
>
> 那对于要求f(5) 时，我们可以画出如下的递归树



> 想求f(5) ，那就让计算机运算f(3) + f(4)，f(3)和f(4)答案是未知的，所以继续往下递归，想求f(3) 那就是f(2) + f(1) ，因为f(2) 和 f(1)都是直接return1 所以f(3) 就等于2，以此类推，那f(4) = f(3) + f(2) 也就是 2 + 1 = 3 最终回到最开始的根，f(5) = f(4) + f(3) = 3 + 2 = 5
>
> 从上往下的过程就是我们求f(5) = f(4) + f(3) = (f(3) + f(2)) + (f(2) + f(1)) 这就是往下递的过程
>
> f(1) f(2)的值给f(3) 再给f(4) 最后回到最初的f(5) 这就是归的过程
>
> 一定是先递后归，在程序中，我们调用fib(n - 1) + fib(n - 2)的时候，就打开了新的函数，函数就反复被使用了，这就是递
>
> 那等fib(n - 1) + fib(n - 2)的值计算出来，程序中return 这个值给上一个函数的时候，就在归了
>
> 递归是一个比较难以理解的算法了，各位同学在学习递归的时候切莫着急，求得速成。

三、作业

> 完成"期末考试"测试卷