递归算法

1. 前n项和

给定n(n>=1)，用递归的方法计算1+2+3+...+(n-1)+n。

```
#include <iostream>
using namespace std;

int n;
int f(int);

int main(){

    cin >> n;
    cout << f(n) << endl;

    return 0;
}

int f(int x){
    if(x==1) return 1;
    return x+f(x-1);
}
```

2. 斐波那契数列

满足 $F_1 = F_2 = 1, F_n = F_{n-1} + F_{n-2}$ 的数列称为斐波那契数列(Fibonacci)，它的前若干项是1,1,2,3,5,8,13,21,34,....求词数列第n项(n>=3)。

```
#include <iostream>
using namespace std;

int n;
int f(int);

int main(){

    cin >> n;
    cout << f(n) << endl;

    return 0;
}

int f(int x){
    if(x==1||x==2) return 1;
    return f(x-1)+f(x-2);
}
```

3. 最大公约数

给定两个正整数a,b求它们的最大公约数gcd(a,b)。

```
#include <iostream>
using namespace std;

int a,b;
int gcd(int,int);

int main(){

    cin >> a >> b;
    cout << gcd(a,b) << endl;

    return 0;
}

int gcd(int x,int y){
    if(x%y==0) return y;
    return gcd(y,x%y);
}
```

4. 逆波兰式求解

逆波兰表达式是一种把运算符前置的算术表达式，例如普通的表达式2 + 3的逆波兰表示法为+ 2 3。 逆波兰表达式的优点是运算符之间不必有优先级关系，也不必用括号改变运算次序， 例如(2 + 3) * 4的逆波兰表示法为* + 2 3 4。 本题求解逆波兰表达式的值，其中运算符包括+ - * /四个。

1. 输入：输入为一行，其中运算符和运算数之间都用空格分隔，运算数是浮点数。
2. 输出：输出为一行，表达式的值。可直接用printf("%f\n", v)输出表达式的值v。
3. 样例输入：* + 11.0 12.0 + 24.0 35.0
4. 样例输出：1357.000000
5. 提示：可使用atof(str)把字符串转换为一个double类型的浮点数。atof定义在math.h中。此题可使用函数递归调用的方法求解。

```
#include <iostream>
#include <string>
#include <cmath>
using namespace std;

string s;
double f();

int main(){

    printf("%f",f());

    return 0;
}

double f(){
    cin >> s;
```

```
    if(s=="+")      return f()+f();
    else if(s=="-") return f()-f();
    else if(s=="*") return f()*f();
    else if(s=="/") return f()/f();
    else return atof(s.c_str());
}
```

5. 递归二分

设有n个数已经按从大到小的顺序排列，现在输入x，判断它是否在这n个数中，如果存在则输出“YES”，否则输出“NO”。

```
#include <iostream>
using namespace std;

int a[100],n,x,lft,rgt;
void search(int,int,int);

int main(){
    cin >> n;
    lft = 0;
    rgt = n-1;
    for(int i=0;i<n;i++)
        cin >> a[i];

    cin >>x;
    search(x,lft,rgt);

    return 0;
}

void search(int x,int l,int r){
    if(l<=r){
        int mid = (l+r)/2;
        if(a[mid]==x){
            cout << "YES" << endl;
            return;
        }else{
            if(a[mid]>x)
                search(x,mid+1,r);
            else
                search(x,l,mid-1);
        }
    }else{
        cout << "NO" << endl;
    }
}
```

6. Hanoi 汉诺塔问题

有n个圆盘，依半径大小（半径都不同），自下而上套在a柱上，每次只允许移动最上面一个盘子到另外的柱子上去（除a柱外，还有b柱和c柱，开始时这两个柱子上无盘子），但绝不允许发生柱子上出现大盘子在上小盘子在下的情况。现要求设计将a柱子上n个盘子搬移到c柱去的方法。

```
#include <iostream>
using namespace std;

int k=0,n;
void move(int,char,char, char);

int main(){

    cin >> n;
    move(n,'a','b','c');

    return 0;
}

void move(int n,char a,char c,char b){
    if(n==0) return;
    move(n-1,a,b,c);
    cout << ++k << ": " << a << "->" << n << "->" << c << endl;
    move(n-1,b,a,c);
}
```

7. 集合的划分

设S是一个具有n个元素的集合，$S=\{a_1,a_2,\ldots,a_n\}$，现将S划分成k个满足下列条件的子集合$S_1,S_2,\ldots,S_K$，且满足：

1. $S_i\neq\emptyset$
2. $S_i\bigcap S_j=\emptyset(1\leq i,j\leq k,i\neq j)$
3. $S_1\bigcup S_2\bigcup S_3\bigcup\cdots\bigcup S_k=S$

则称$S_1,S_2,S_3,\cdots,S_k$是集合S的一个划分。他相当于把S集合中的n个元素$a_1,a_2,\cdots,a_n$放入k个(0<k<=n<30)无标号的盒子中，使得没有一个盒子为空。请你确定n个元素$a_1,a_2,\cdots,a_n$放入k个无标号盒子中去的划分数S(n,k)。

- 输入样例： 10 6
- 输出样例： 22827
- 算法分析：
    1. $S(n,k)=S(n-1,k-1)+k*S(n-1,k),(n>k,k>0)$
    2. $S(n,k)=0,(n<k)or(k=0)$
    3. $S(n,k)=1,(k=1)or(k=n)$

```
#include <iostream>
using namespace std;

int n,k;
int s(int,int);

int main(){

    cin >> n >> k;
    cout << s(n,k) << endl;

    return 0;
}
```

```
int s(int n,int k){
    if((n<k)||(k==0))
        return 0;
    if((k==1)||(k==n))
        return 1;
    return s(n-1,k-1) + k*s(n-1,k);
}
```

8. 数的技术(Noip 2001)

我们要求找出具有下列性质数的个数（包括输入的自然数n）。先输入一个自然数n(n<=1000)，然后对此自然数按照如下方法进行处理：

1. 不做任何处理
2. 在它的左边加上一个自然数，但该自然数不能超过原数的一半
3. 加上数后，继续按此规则进行处理，直到不能再加自然数为止

- 输入格式：n(n<=1000)
- 输出格式：满足条件的数
- 样例输入：6
- 样例输出：6
- 样例分析：6,16,26,126,36,136

递归方法

```
#include <iostream>
using namespace std;
int ans=0;

void dfs(int m){
    ans++;
    for(int i=1;i<=m/2;i++){
        dfs(i);
    }
    cout << ",";
}

int main(){
    int n;

    cin >> n;
    dfs(n);
    cout << ans << endl;

    return 0;
}
```

记忆化搜索

```
#include <iostream>
#include <cstring>
using namespace std;
```

```
int h[1001];

void dfs(int m){
    if(h[m]!=-1)
        return;
    h[m]=1;
    for(int i=1;i<=m/2;i++){
        dfs(i);
        h[m]=h[m]+h[i];
    }
}

int main(){
    int n;
    memset(h,-1,sizeof(h));

    cin >> n;
    dfs(n);
    cout << h[n] << endl;

    return 0;
}
```

递推方法一

$h(i) = h(1) + h(2) + \cdots + h(i/2)$

```
#include <iostream>
using namespace std;

int h[10001];

int main(){
    int n;

    cin >> n;
    for(int i=1;i<=n;i++){
        h[i]=1;
        for(int j=1;j<=i/2;j++)
            h[i]+=h[j];
    }
    cout << h[n] << endl;

    return 0;
}
```

递推方法二

已知$h(i) = h(1) + h(2) + \cdots + h(i/2)$，令$s(x) = h(1) + h(2) + \cdots + h(x)$，则
$h(x) = s(x) - s(x-1)$，$h(i) = 1 + s(i/2)$。

```
#include <iostream>
using namespace std;
```

```
int h[1001],s[1001];

int main(){
    int n;

    cin >> n;
    for(int i=1;i<=n;i++){
        h[i]=1+s[i/2];
        s[i]=s[i-1]+h[i];
    }
    cout << h[n] << endl;

    return 0;
}
```

递推方法三

由h(1)=1,h(2)=2,h(3)=2,h(4)=4,h(5)=4,h(6)=6,h(7)=6,h(8)=10,h(9)=10,...,可知:

1. h(i)=h(i-1)，当i为奇数时
2. h(i)=h(i-1)+h(i/2)，当i为偶数时

```
#include <iostream>
using namespace std;

int h[1001];

int main(){
    int n;

    cin >> n;
    h[1]=1;
    for(int i=2;i<=n;i++){
        h[i]=h[i-1];
        if(i%2==0)
            h[i]=h[i-1]+h[i/2];
    }
    cout << h[n] << endl;

    return 0;
}
```