# 平成 24 年度 アドバンスト・プログラミング
# (Advanced Programming)

小澤　一文，陳　国躍，中村真輔

木曜 2 限 GI201 教室

# 再帰呼び出しとは

## 階乗の計算

$n!$ は $n! = n \times (n-1)!$ であるから，

まず $(n-1)!$ を計算する関数を呼び出し，

結果が得られたらそれに $n$ を掛ける

という手順で計算できる。

すなわち，$n!$ を計算する関数の中に $(n-1)!$ を計算する関数を埋め込めばよい。

同様に

$(n-1)!$ を計算する関数に $(n-2)!$ を計算する関数を埋め込めばよい

$(n-2)!$ を計算する関数に $(n-3)!$ を計算する関数を埋め込めばよい

……

**これを簡単に行う方法が C にはある**

# 再帰呼び出しのプログラム (1)

```
/*
    階乗の計算
*/
#include<stdio.h>
int fact(int n);
main() {
  int n,n_max=10;
  for (n=0; n<=n_max; n++) {
    printf("%2d!= %d \n",n,fact(n));
  }
}
int fact(int n)
{
  if (n == 0) return (1);
  else
    return (n*fact(n-1));
}
```

$n!$ を計算する関数 fact (n) に $(n-1)!$ を計算する fact (n-1) が埋め込まれている

# 再帰呼び出しのプログラム (2)

実行結果

```
 0!= 1
 1!= 1
 2!= 2
 3!= 6
 4!= 24
 5!= 120
 6!= 720
   ....
10!= 3628800
```

**問題** $S(n) = 1 + 2 + \cdots + n$ を計算する関数を再帰呼び出しを用いて書け。

# 再帰呼び出しの動作原理

階乗を計算するプログラムでは

- “丸投げ” の連続

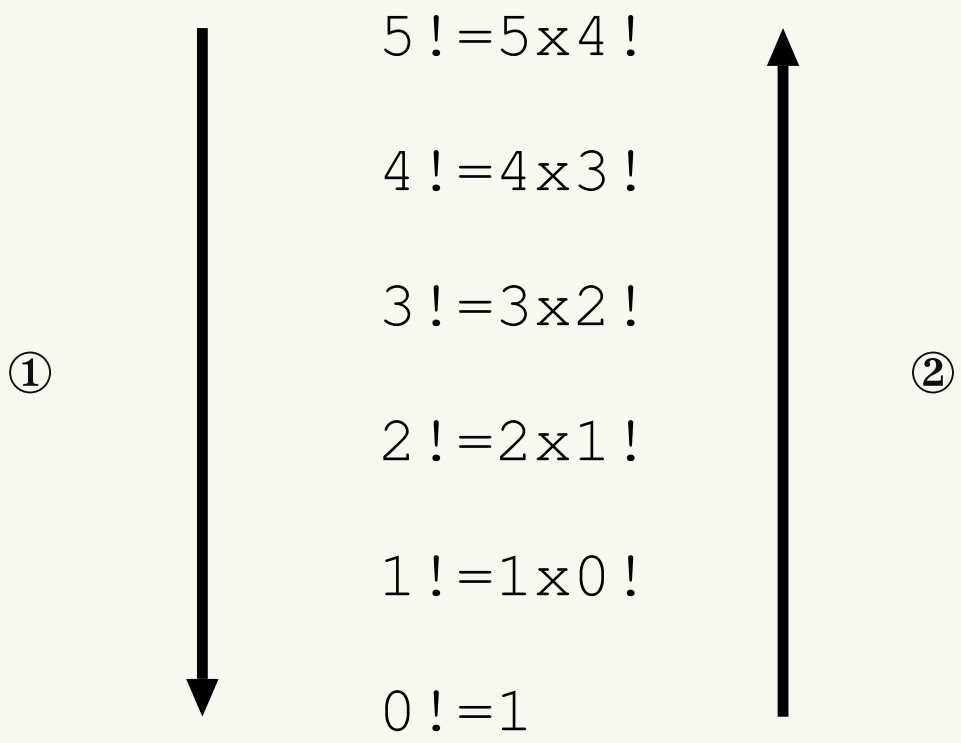
  `fact(n)` は `fact(n-1)` に “丸投げ” し，
  `fact(n-1)` は `fact(n-2)` に “丸投げ” し，
  ……
  ……

  ということを繰り返し行なっている。

- “丸投げ” をどこかで断ち切る仕掛けが必要になる
  その仕掛けとは，いまの場合は

  ```
  if ( n == 0 ) return (1);
  ```

# 5! の計算では




再帰呼び出しを行ったときの計算の流れ

①スタックに次々に仕事を積み上げていき，0! までたどり着いたら，②逆向きに計算を開始する。

$$\longrightarrow$$

$$1 \cdot 1 \cdot 2 \cdot 3 \cdot 4 \cdot 5 = 120$$

# 最大公約数 (gcd) について

整数 $a$ と整数 $b$ の**最大公約数** (Greatest common divisor) を $\gcd(a, b)$ で表す。$a$ を $b(a > b)$ で割ったとき，余りを $r$ とすると次の定理が成り立つ。

**gcd に関する定理**

$$\gcd(a, b) = \gcd(b, r)$$

この定理を繰り返し使っていくと gcd が簡単に求まる。

$r$ は $b$ で割った余りなので，

$$0 \leq r < b$$

である。これより $a > b > r$ なので $\gcd(a, b)$ より $\gcd(b, r)$ を計算するほうが楽である。同じことを繰り返し行う。

# gcd を求めるアルゴリズム (1)

$b$ を $r$ で割った余りを $r_1$ とし，$r$ を $r_1$ で割った余りを $r_2$ とし，$\cdots$，ということを繰り返していくと

$$\gcd(a,b) = \gcd(b,r) = \gcd(r_1,r_2) = \gcd(r_2,r_3) = \cdots$$

であるが

$$r > r_1 > r_2 > r_3 > \cdots \geq 0$$

なので，余りは確実に小さくなり，どこかで必ず 0 になるはず。

もし $r_i = 0$ ならば，$r_{i-2}$ は $r_{i-1}$ で割り切れたことになり，$\gcd(a,b) = r_{i-1}$ である。$\gcd(a,b) = r_{i-1}$ である。

# gcd を求めるアルゴリズム (2)

## ユークリッドの互除法

1. $a > b$ を仮定し，$a$ を $b$ で割り余り $r$ を求める。
2. $r$ の値を調べ
   (a) $r > 0$ なら $a \leftarrow b$, $b \leftarrow r$ とし，1. へ戻る。
   (b) $r = 0$ なら $b$ を返して計算終了。

# ユークリッドの互除法の計算例

例 **1** $\gcd(68, 24)$ の例

$$\gcd(68, 24) = \gcd(24, 20) = \gcd(20, 4) = 4$$

例 **2** $\gcd(480, 320)$ の例

$$\gcd(640, 480) = \gcd(480, 160) = 3$$

問 $\gcd(78, 36)$ および $\gcd(142, 66)$ をユークリッドの互除法で求めよ。

問 $a < b$ のときユークリッドの互除法を用いるとどうなるか。

# gcd のプログラム (1)

```
int gcd(int a, int b) {
  int r;
  r=a%b;
  while( r!= 0 ) {
    a=b; b=r;
    r=a%b;
  }
  return(b);
}
```

かっこよく書くと

```
int gcd(int a, int b) {
  int a,b,r;
  while((r=(a%b))!= 0) {
    a=b; b=r;
  }
  return(b);
}
```

# gcd のプログラム (2)

再帰呼び出しを使うと

```
int gcd(int a, int b) {
  int r;
  r=a%b;
  if ( r==0 )
    return (b);
  else
    return (gcd(b,r));
}
```

# $\gcd(a, b) = \gcd(b, r)$ の証明

- $g = \gcd(a, b)$ とする。このとき $a = g\,l$，$b = g\,m$ と置ける。すると

$$r = a - b\,q = g\,l - g\,m\,q = g\,(l - m\,q), \qquad q \text{ は商}$$

であるから，$g$ は余り $r$ の約数でもある。ということは，$b, r$ の公約数ということになる。

- いま $b, r$ に $g$ より大きい公約数 $g'$ が存在すると仮定する。このとき

$$a = b\,q + r$$

であるから，$g'$ は $a$ の約数になもなっている。すなわち，$g'$ は最大公約数 $g$ よりも大きい $a, b$ の公約数である。しかしこれは明らかに矛盾である。よって $b$ と $r$ の間に $g$ より大きい公約数は存在しない。すなわち

$$g = \gcd(b, r)$$

# 再帰呼び出しを使う上での注意点

## フィボナッチ数列の例

フィボナッチ数列

$$f_{n+2} = f_{n+1} + f_n, \qquad f_0 = f_1 = 1$$

を計算する関数は

```
int fib(int n)
{
  if ((n==0) || (n==1)) return (1);
  return (fib(n-1)+fib(n-2));
}
```

このプログラムを動かすとどうなるか?

# フィボナッチ数列のプログラムを解析すると (1)

$f_n$ を計算するときは，まず $f_{n-1}$ と $f_{n-2}$ を積み上げる。

積み上げられた $f_{n-1}$ は $f_{n-2}$ と $f_{n-3}$ を積み上げる。

積み上げられた $f_{n-2}$ は $f_{n-3}$ と $f_{n-4}$ を積み上げる。

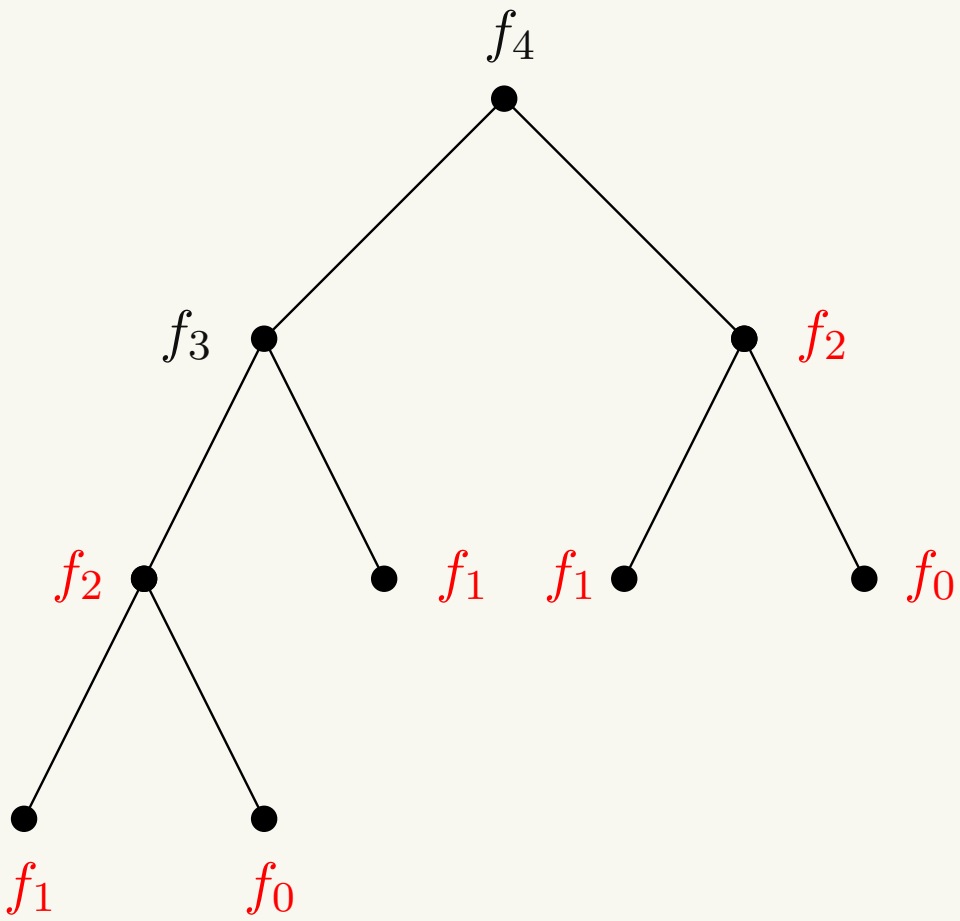
$\cdots$

積み上げられた $f_2$ は $f_1$ と $f_0$ を積み上げる (終了)。

同じ物が何度も積み上げられていく。

# フィボナッチ数列のプログラムを解析すると (2)

$f_4$ の計算では




赤い部分は複数回呼び出されている

# フィボナッチ数列のプログラムを解析すると (3)

$f_n$ を計算するのに必要なメモリ (計算量) を $M(n)$ とする。$f_n$ を計算するためには $f_{n-1}$ と $f_{n-2}$ が必要なので

$$M(n) = M(n-1) + M(n-2)$$

となる。$f_0$ と $f_1$ を計算するの要するメモリ (計算量) を $m$ とすれば，$M(n)$ の値は漸化式

$$M(n) = M(n-1) + M(n-2), \qquad M(0) = M(1) = m$$

で与えられる。すなわち $M(n) = m\, f_n$ で与えられる。$f_n$ の値は

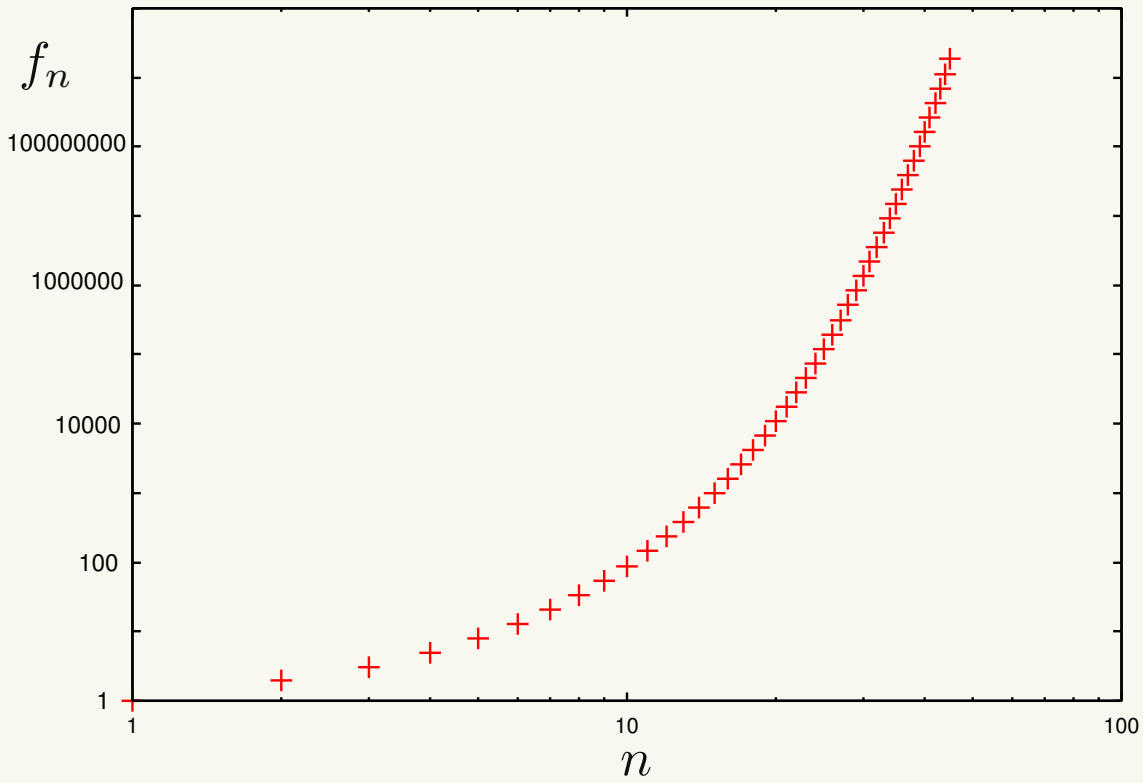
$$f_n = \frac{1}{\sqrt{5}} \left( \frac{1+\sqrt{5}}{2} \right)^{n+1} - \frac{1}{\sqrt{5}} \left( \frac{1-\sqrt{5}}{2} \right)^{n+1} \sim \frac{1}{\sqrt{5}} \left( \frac{1+\sqrt{5}}{2} \right)^{n+1}$$

である。

# フィボナッチ数列のプログラムを解析すると (4)

$f_n$ の値

| $n$ | $f_n$ |
|---|---|
| 0 | 1 |
| 1 | 1 |
| 2 | 2 |
| 3 | 5 |
| ... | ... |
| 10 | 89 |
| ... | ... |
| 20 | 10946 |
| ... | ... |
| 40 | 165580141 |




再帰呼び出しは膨大なメモリ，計算時間を使うことになる (こともある)。

安易に再帰呼び出しを用いないこと

# 演習問題

1. 等比数列の和

$$S_n(r) = 1 + r + r^2 + \cdots + r^{n-1}$$

を 計算する 関数を 再帰呼び出しを 用いて 書け。

2. 整数 $a, b$ の最小公倍数 (Least common multiple) を $\mathrm{lcm}(a, b)$ で表す。$\mathrm{lcm}(a, b)$ と $\gcd(a, b)$ の間に

$$a\,b = \gcd(a, b)\,\mathrm{lcm}(a, b)$$

という 関係がある。この関係を 用いて 整数の最大公約数と 最小公倍数を 同時に 求める 関数を 作れ。

# 演習問題 (続き)

3. $n$ 個のデータ $a_0, a_1, \ldots, a_{n-1}$ の最大値を $m_n$ で表す。そうすると

$$m_n = \max\{m_{n-1}, a_{n-1}\}, \quad m_0 = a_0$$

となる。この考え方を用いて最大値を求める関数を作れ。

4. フィボナッチ数列

$$f_{n+2} = f_{n+1} + f_n, \qquad f_0 = f_1 = 1$$

の解は

$$f_n = \frac{1}{\sqrt{5}}\left(\frac{1+\sqrt{5}}{2}\right)^{n+1} - \frac{1}{\sqrt{5}}\left(\frac{1-\sqrt{5}}{2}\right)^{n+1}$$

で与えられることを示せ。

**Hint:** $\lambda_1 = (1+\sqrt{5})/2$, $\lambda_2 = (1-\sqrt{5})/2$ と置けば，$\lambda_1$, $\lambda_2$ は $\lambda^2 = \lambda + 1$ を満たすことがわかる。このことを用いる。