

アルゴリズム論 2&3

概要

- アルゴリズムの定義
- C言語の復習
- データ構造について
- 計算量について
- 再帰呼び出し

再帰呼び出し(再帰処理)

☑再帰とは？

- もともにもどる、繰り返しの意味

☑プログラミングにおける意味

- ある関数の中で自分自身を呼び出す処理
- 処理手順がそれ自身を用いて定義されている
- 自分自身の呼び出しの終了条件が与えられている
- 昔の言語ではサポートされていない(FORTRAN,BASIC)

☑効果

- 繰り返し処理の代用が可能
- プログラムサイズを小さくできる

再帰呼び出しが適用できる例 1

再帰処理基礎1

階乗：

$$1!=1$$

$$2!=2 \times 1 = 2$$

$$3!=3 \times 2 \times 1 = 6$$

$$4!=$$

⋮

⋮

$$n! = n \times (n-1) \times (n-2) \times \cdots \times 2 \times 1 \qquad n! = n \times (n-1)!$$

階乗が階乗の関数で表現できることに注目

数学的な決まり $0!=1$  再帰終了条件

階乗を実現する関数例

(再帰呼び出しを使用しない)

```
int factorial(int n)
{
    int i,x;
    if (n==0)
    {
        return 1;
    }
    else
    {
        x=1;
        for (i=n;i>=1;i--)
        {
            x=x*i;    /* x *= i; */
        }
        return x;
    }
}
```

階乗を実現する関数例

(再帰呼び出しを使用する)

```
int factorial(int n)
{
    if (n==0)
    {
        return 1;
    }
    else
    {
        return n*factorial(n-1);
    }
}
```

自明なケース

自明なケースの処理
再帰の終了条件

return n*factorial(n-1); 自分自身で定義される

プログラム比較

```
int factorial(int n)
{
    int i,x;
    if (n==0)
    {
        return 1;
    }
    else
    {
        x=1;
        for (i=n;i>=1;i--)
        {
            x=x*i;
        }
        return x;
    }
}
```

```
int factorial(int n)
{
    if (n==0)
    {
        return 1;
    }
    else
    {
        return n*factorial(n-1);
    }
}
```

プログラムサイズに注目!!!
効果が理解できる

再帰呼び出しが適用できる例 2

再帰処理基礎2

自然数の和 : $s(n)$

$$s(1)=1$$

$$s(2)=1+2=3$$

$$s(3)=1+2+3=6$$

$$s(4)=$$

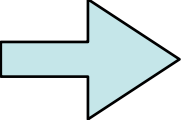
:

:

$$s(n)=1+2+3+\cdots+(n-1)+n \quad s(n)=s(n-1)+n$$

$s(n)$ が $s(n-1)$ の関数で表現できる

再帰 : 自分自身と同じ関数を呼び出す場合に記述可能

$s(1)=1$  再帰終了条件

自然数の和を実現する関数例

(再帰呼び出しを使用しない)

```
int sum(int n)
{
```

```
}
```

30

自然数の和を実現する関数例

(再帰呼び出しを使用する)

```
int sum(int n)
```

```
{
```

```
}
```

演習問題2

課題: 階乗と自然数の和を求める関数について、メイン関数を作成して動作を確認する。

- 各関数のプログラム作成
- メイン関数の機能
 - キーボードから自然数を入力する。
 - 結果を表示する。

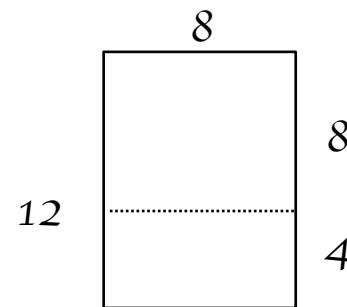
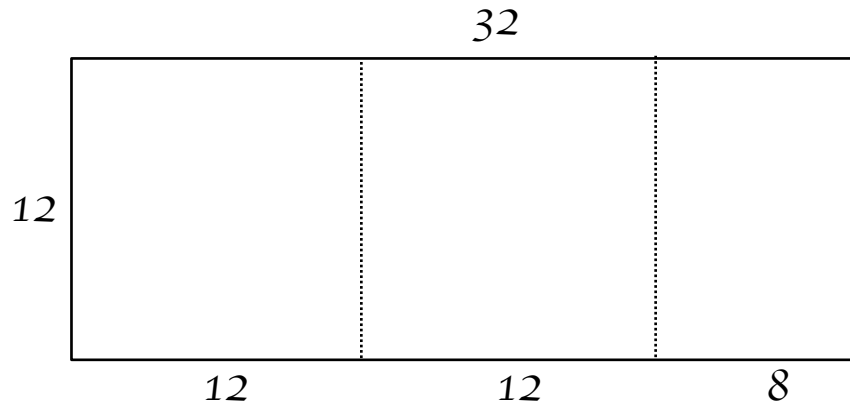
再帰処理応用1

- ユークリッドの互除法：2つの自然数の最大公約数を求めるアルゴリズム
- 問題の置き換え：
 - 長方形を埋め尽くすことのできる正方形を考える
 - 埋め尽くすことが可能な最大の正方形の辺の長さを求める

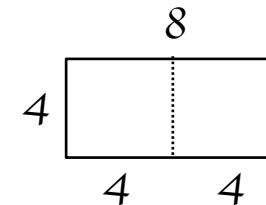
例題

- 32と12の最大公約数を求める

- 長辺が32cm、短編が12cmの長方形を分割



- (1)長い辺を短い辺を1辺とする正方形で分割
- (2)余った長方形に対して繰り返す
- (3)最終的に分割された正方形の1辺が最大公約数



- 1辺4cmの正方形で分割可能 : 最大公約数 4

手順の整理

2つの自然数($a \geq b > 0$)が与えられる

1. **aをbで割った余りをrとする。**

2. **rが0の場合** : 終了条件

割った値が最大公約数

3. **rが0以外**

$a=b, b=r$ として1.にもどる。

赤文字部分を再帰呼び出しで実現する

ユークリッド互除法 ソースコード

```
#include <stdio.h>

int gcd(int a, int b)
{
    if (b==0)
        return(a);
    else
        return(gcd(b,a % b));/* 再帰呼び出し */
}

int main(void)
{
    int a,b;
    printf(" input a :");
    scanf("%d",&a);
    printf(" input b :");
    scanf("%d",&b);
    printf(" gcd is %d ¥n",gcd(a,b));

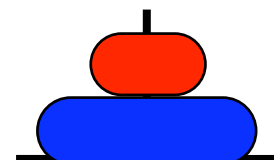
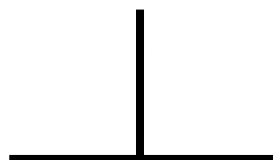
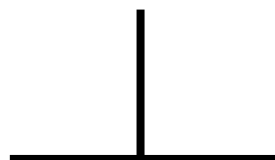
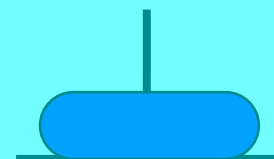
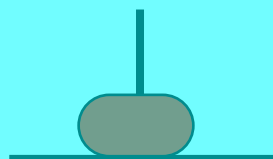
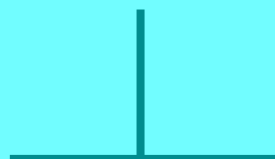
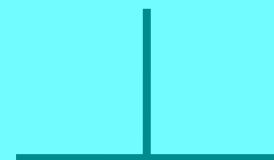
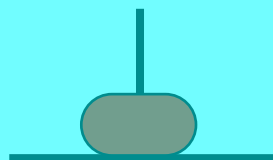
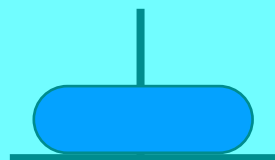
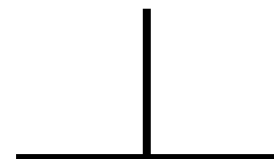
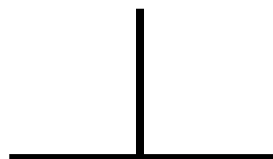
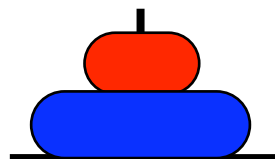
    return(0);
}
```

プログラムを実行して確認すること 36

再帰処理応用 2

- **ハノイの塔：重なった円盤を3本の柱(軸1,2,3)の間で移動する問題**
- **問題の規則：**
 - **最初は軸1に全ての円盤が重ねられている**
 - **円盤は1枚ずつのみ移動できる**
 - **円盤を重ねる場合は、下の円盤よりも小さい円盤のみ重ねることが可能**
 - **全ての円盤が軸3に移動したら終了**

円盤が2枚の場合

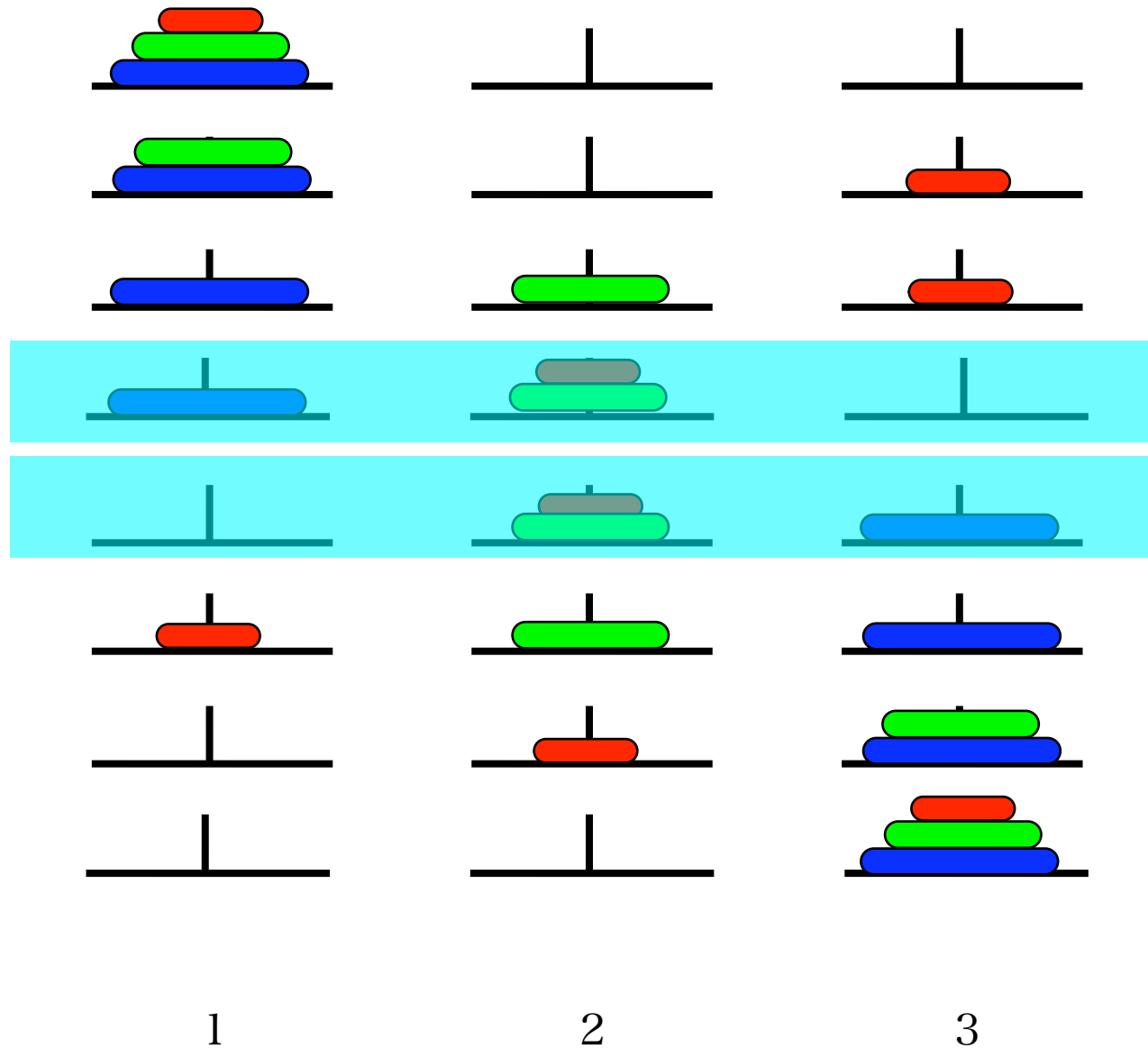


1

2

3

円盤が3枚の場合



円盤の枚数によらない共通的な手順

1. 一番大きな円盤以外を軸 2 に移動
2. 一番大きな円盤を軸 1 から軸 3 に移動
3. 一番大きな円盤以外を軸 3 に移動

1 および 3 を再帰的に実現する

ハノイの塔

ソースファイル

```
#include <stdio.h>

void move(int no, int x, int y)
{
    if (no > 1)
        move(no-1, x, 6-x-y);
    printf("¥n[%d] %d=>%d", no, x, y);
    if (no > 1)
        move(no-1, 6-x-y, y);
}

int main(void)
{
    int n;
    printf(" No of Disc :");
    scanf("%d", &n);
    move(n, 1, 3);
    return(0);
}
```

実行結果

No of Disc :2	No of Disc :3	No of Disc :4
[1] 1 => 2	[1] 1 => 3	[1] 1 => 2
[2] 1 => 3	[2] 1 => 2	[2] 1 => 3
[1] 2 => 3	[1] 3 => 2	[1] 2 => 3
	[3] 1 => 3	[3] 1 => 2
	[1] 2 => 1	[1] 3 => 1
	[2] 2 => 3	[2] 3 => 2
	[1] 1 => 3	[1] 1 => 2
		[4] 1 => 3
		[1] 2 => 3
		[2] 2 => 1
		[1] 3 => 1
		[3] 2 => 3
		[1] 1 => 2
		[2] 1 => 3
		[1] 2 => 3

プログラムを実行して確認すること
41