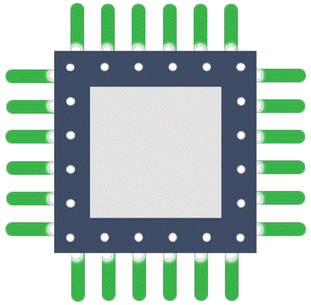
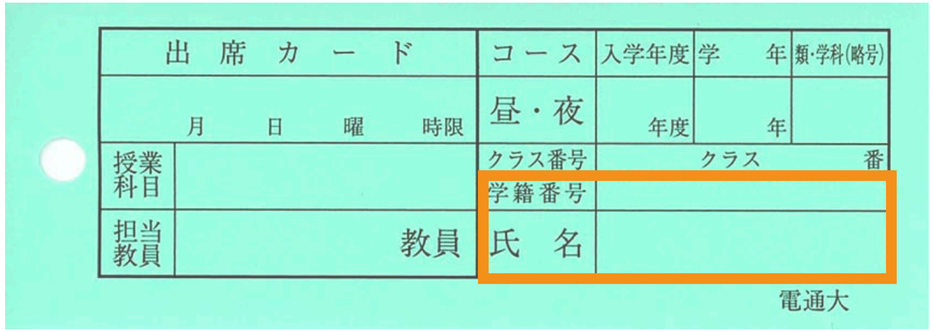


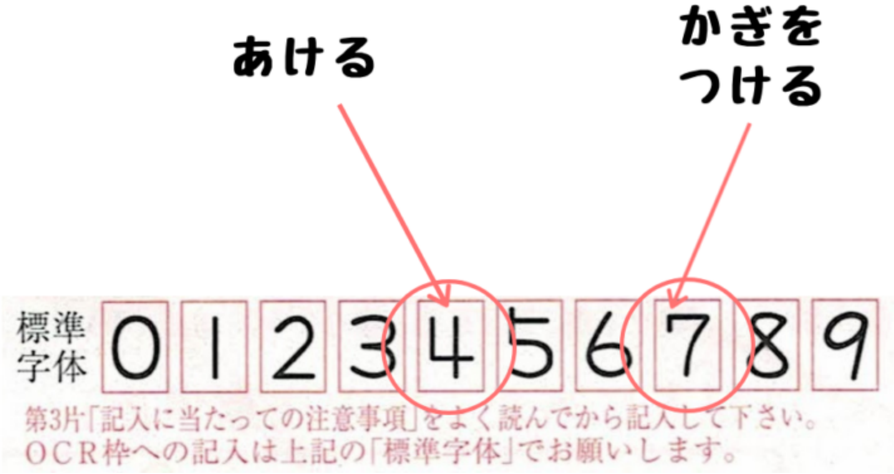
MIPS

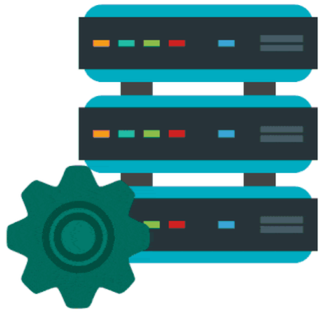
演習資料

情報領域演習第二 C演習 第4回



ここだけ書けば良い
数字は右例のように





クラス２　2024年7月12日1限 担当教員：髙木一幸

再帰的サブルーチンの構成

```
#-----------------------------------------------
#   rcrsv：ある再帰的な関数f(n)の値を計算
#            f(n) := n*f(n-1)
#   引数：$a0：n   (n > 0)
#   戻値：$v0：f(n)
#-----------------------------------------------
rcrsv: addi   $sp, $sp, -8   # スタックに2ワードの領域を確保
       sw     $ra, 0($sp)    # 戻り先アドレスの退避
       sw     $a0, 4($sp)    # $a0(=n)の退避
       addi   $a0, $a0, -1   # $a0 <= $a0-1 (n-1)
       jal    rcrsv          # 再帰呼び出し(f(n-1)の計算)
       lw     $a0, 4($sp)    # $a0(=n)の復元
       mul    $v0, $a0, $v0  # $v0 <= $a0*v0 (n*f(n-1))
       lw     $ra, 0($sp)    # 戻り先アドレスの復元
       addi   $sp, $sp, ...  # スタックに確保した領域の開放
       jr     $ra
```

見通しを良くしてバグを減らす

- コメントにプログラム、サブルーチンの機能を正確かつ簡潔に記述
- 内部処理に使うレジスタの役割分担を明確にする
- 処理ステップ毎にコメント
- 戻り先アドレス$raと引数に用いる$aレジスタ（$a0から順に必要なものだけ）を再帰呼び出し前に退避し、戻り後に復元。内部処理用レジスタも。
- ラベルは個別具体的な名称に
  - loop → nxtint, prdloop
  - exit → extfct, colend

レポート課題

C演習第4回課題

課題１【スタック、サブルーチン】

入力された複数の整数値を入力と逆順に出力

最大値および最小値にそれぞれ "(max)"、"(min)" の注釈

その１：サブルーチン無し版　　その２：サブルーチン有り版

課題２【再帰的アルゴリズム（１）】その１：階乗　　その２：最大公約数

課題３【再帰的アルゴリズム（２）】ハノイの塔

https://ja.wikipedia.org/wiki/ハノイの塔

詳細

MIPS

課題１【スタック、サブルーチン】

入力された複数の整数値を入力と逆順に出力

最大値および最小値にそれぞれ "(max)"、"(min)" の注釈

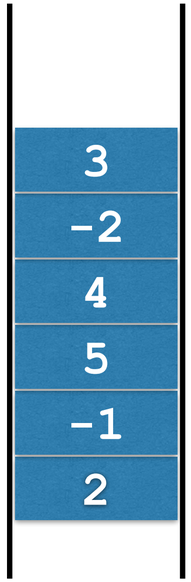
その１：サブルーチン無し版　　その２：サブルーチン有り版

例）2, -1, 5, 4, -2, 3, 0 を入力

```
2
-1
5
4
-2
3
0
3
-2  (min)
4
5  (max)
-1
2
```

入力が0でなければ
最小値、最大値を更新し
スタックに push

スタックから1つずつ pop
1. 数値を表示
2. 最小値、最大値と等しければ
"(max)"、"(min)" を表示



stack

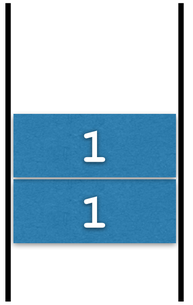
例）-3, 0 を入力。

```
(spim) run
-3
0
-3  (max)  (min)
```

stack

例）1, 1, 0 を入力。

```
(spim) run
1
1
0
1  (max)  (min)
1  (max)  (min)
```



stack

課題２【再帰的アルゴリズム（１）】その１：階乗　その２：最大公約数

定義

階乗

$$
\begin{aligned}
n! &= 1 && n = 0 \\
n! &= n \times (n-1)! && n \geq 1
\end{aligned}
$$

最大公約数　（$a \geq 0$，$b \geq 0$）

$$
\begin{aligned}
g(a,b) &= a && b = 0 \\
g(a,b) &= g(b, a\%b) && \text{otherwise}
\end{aligned}
$$

ユークリッドの互除法

C

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

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

MIPS

```
#-------------------------------------------------
#   fact：階乗計算（再帰的サブルーチン）
#       n!=1,         n=0
#       n!=n*(n-1)!, n>=1
#   引数：$a0：n
#   戻値：$v0：n!
#-------------------------------------------------
fact: addi  $sp, $sp, ... # スタックに...ワードの領域を確保
      sw    $ra, 0($sp)   # 戻り先アドレスの退避
      ...
      jal   fact          # 再帰呼び出し((n-1)!の計算)
      ...
      lw    $ra, 0($sp)   # 戻り先アドレスの復元
      addi  $sp, $sp, ... # スタックに確保した領域の開放
      jr    $ra
```

```
#-------------------------------------------------
#   gcd：最大公約数をユークリッドの互除法で再帰的に計算
#       g(a,b)=a,          b=0
#       g(a,b)=g(b,a%b), otherwise
#   引数：$a0：a, $a1：b
#   戻値：$v0：n!
#-------------------------------------------------
gcd:  addi  $sp, $sp, ... # スタックに...ワードの領域を確保
      sw    $ra, 0($sp)   # 戻り先アドレスの退避
      ...
      jal   gcd           # 再帰呼び出し(g(b,a%b)の計算
      ...
      lw    $ra, 0($sp)   # 戻り先アドレスの復元
      addi  $sp, $sp, ... # スタックに確保した領域の開放
      jr    $ra
```

MIPS

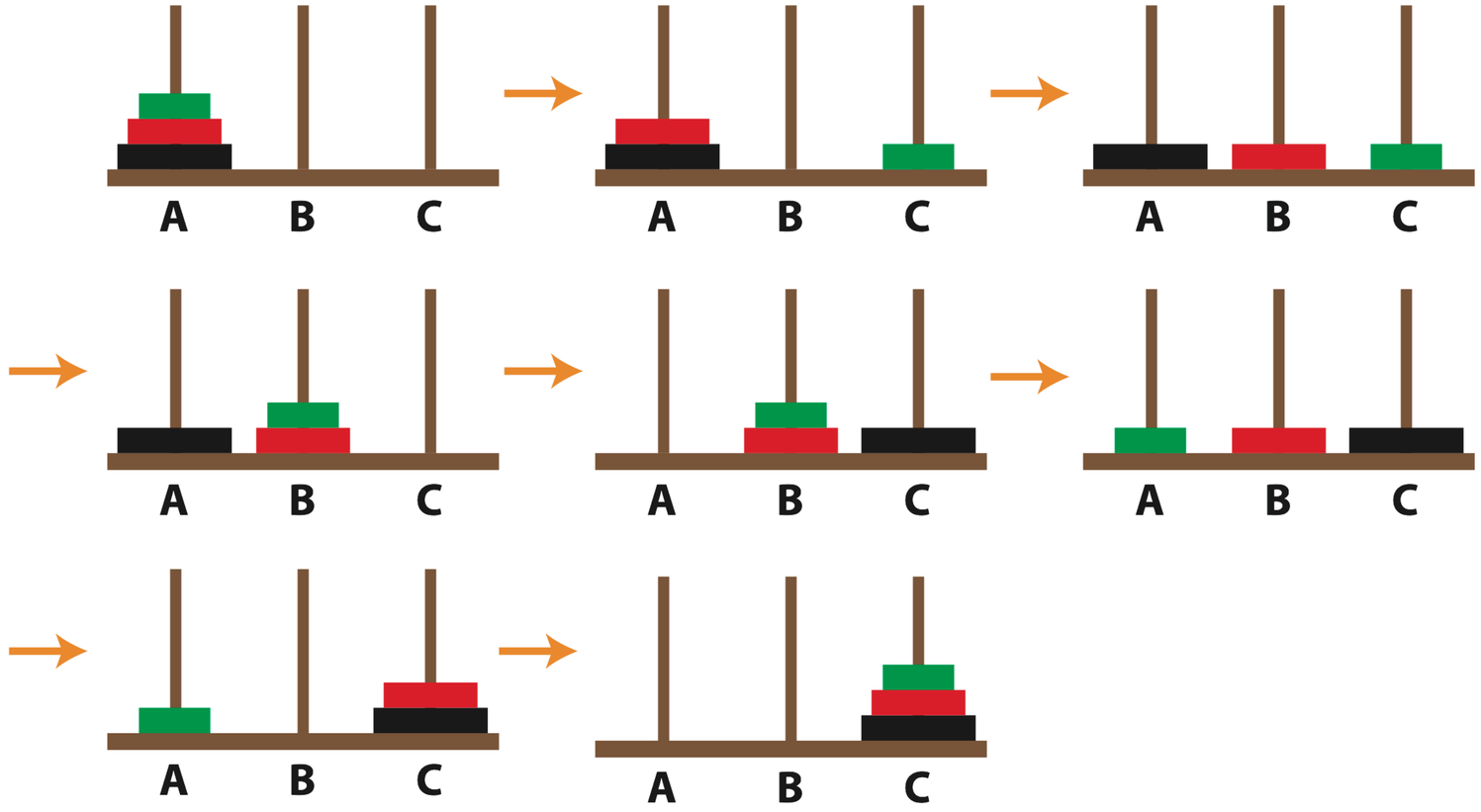
課題３【再帰的アルゴリズム（２）】ハノイの塔

- 3本の杭（A, B, C）と中央に穴の開いた大きさの異なる複数の円盤から構成される。
- 最初はすべての円盤が杭Aに小さいものが上になるように順に積み重ねられている。
- 円盤を1回に1枚ずつ移動させることができるが、小さな円盤の上に大きな円盤を乗せることはできない。

「解」は
円盤の移動手順を
羅列したもの。
”元の杭” -> “先の杭”

解

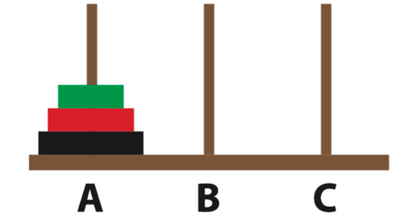
```
A -> C
A -> B
C -> B
A -> C
B -> A
B -> C
A -> C
```

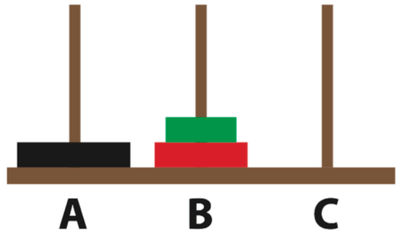


MIPS

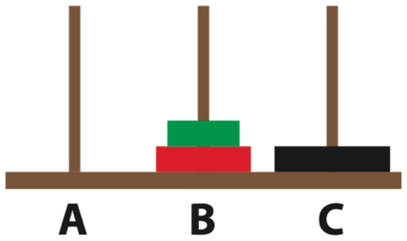
課題３【再帰的アルゴリズム（２）】ハノイの塔

1. 上から n－1 個目までの円盤を
   何らかの方法でAからBに移動。

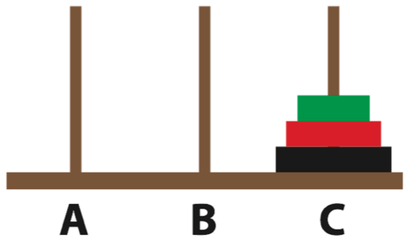




2. 残った1枚をAからCに移動。



3. Bにある円盤を何らかの方法で
   BからCに移動。




```
#include <stdio.h>

void hanoi( int n,              /* 円盤の枚数          */
            char * from,        /* 円盤の移動元の杭    */
            char * to,          /* 円盤の移動先の杭    */
            char * aux )        /* 残りの杭            */
{
    if (1 == n) {
        printf( "%s -> %s\n", from, to );
    }
    else {
        hanoi( n-1, from, aux, to );
        printf( "%s -> %s\n", from, to );
        hanoi( n-1, aux, to, from );
    }
}

int main( void )
{
      int n;

      scanf( "%d", &n );
      hanoi( n, "A", "C", "B" );
      return 0 ;
}
```

メルセンヌ数　Mersenne number

2の冪よりも 1 小さい自然数、すなわち $2^n - 1$（n は自然数）の形の自然数。
これを M_n で表すことが多い。2進数表記では、n 桁の **11…11** となる。

- ハノイの塔の最小移動回数はメルセンヌ数。
- ハノイの塔のゲームでは、1回動かすのに1秒かかるとしたとき、

  塔の板が8枚に増えた場合、$2^8 - 1 = 255$であり、移動所要時間は3分15秒。

- M_{64} =1.8447e+19　　… 約5.849e+11年（5849億年）

（$2^{64} - 1$）回 = 18,446,744,073,709,551,615 回 = 1844京6744兆737億955万1615回

メルセンヌ数のうち素数であるものをメルセンヌ素数という
（例）M_2, M_3, M_5, $M_{77,232,917}$ (December 2017)

メルセンヌ数　Mersenne number

2の冪よりも 1 小さい自然数、すなわち $2^n - 1$（n は自然数）の形の自然数。
これを M_n で表すことが多い。2進数表記では、n 桁の **11…11** となる。

- ハノイの塔の最小移動回数はメルセンヌ数。
- ハノイの塔のゲームでは、1回動かすのに1秒かかるとしたとき、

  塔の板が8枚に増えた場合、$2^8 - 1 = 255$であり、移動所要時間は3分15秒。

- M_{64} =1.8447e+19　　… 約5.849e+11年（5849億年）

（$2^{64} - 1$）回 = 18,446,744,073,709,551,615 回 = 1844京6744兆737億955万1615回

メルセンヌ数のうち素数であるものをメルセンヌ素数という
（例）M_2, M_3, M_5, $M_{77,232,917}$ (December 2017)

課題3 ハノイの塔：データセグメント定義

```
#-------------------------------------------------------------
#
# hanoiTower: ハノイの塔
#
# 情報領域演習第二C演習第4回（2024年7月12日）課題3
#
#-------------------------------------------------------------
#-------------------------------------------------------------
#   データセグメント
#-------------------------------------------------------------
    .data
prmpt: .asciiz     "number of disk = "      # 円盤の数は？”
rodA:  .asciiz     "A"                      # 棒A
rodB:  .asciiz     "B"                      # 棒B
rodC:  .asciiz     "C"                      # 棒C
arrow: .asciiz   " -> "                  # 矢印
newln: .asciiz     "\n"                     # 改行文字
```

レポート課題

C演習第4回課題

課題１【スタック、サブルーチン】

入力された複数の整数値を入力と逆順に出力

最大値および最小値にそれぞれ "(max)"、"(min)" の注釈

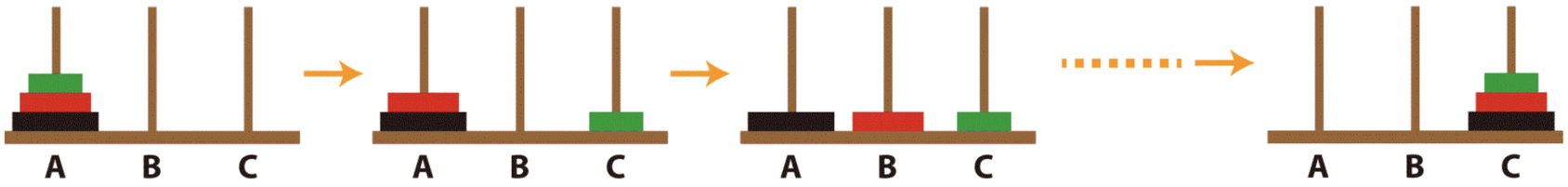
その１：サブルーチン無し版　　その２：サブルーチン有り版

課題２【再帰的アルゴリズム（１）】

その１：階乗　　その２：最大公約数

課題３【再帰的アルゴリズム（２）】

ハノイの塔



C演習最終回なので、提出遅れがないように！

余裕を持って取り組み、期限厳守。

2024年7月25日（木）23:59:59

制作するMIPSプログラムは5種類です。