

Cプログラミング

— 再帰 (1) —

早稲田大学

本日の目標

- 再帰呼び出しを理解する
 - 再帰関数
 - 再帰の例
 - 再帰除去

再帰呼び出し

再帰呼び出しとは…

- 何らかの計算・操作を行いたいとき、それを実現するのがたまたま自分自身と同じ関数であれば、その関数を呼び出すことができる。このような関数の呼び出しを**再帰呼び出し**という。
- 関数の呼び出しがどこかで止まり、大元の呼び出しに戻ってこられるように作らなくてはならない。

再帰呼び出しの長所と短所

- **【長所】** 再帰を使うと、プログラムを簡潔に書くことができる。
- **【短所】** 難しい。
- **【短所】** 実行速度が遅くなり、メモリの消費量も増える。

再帰呼び出しの簡単な例

```
#include <stdio.h>

void recursion( int i ){
    if(i < 10){
        recursion(i+1);
        printf("%d",i);
    }
}

int main(void){
    recursion(0);
    return 0;
}
```

このプログラムは

- 9 8 7 6 5 4 3 2 1 0

を画面に表示する.

プログラムの説明

- このプログラムは、最初に `recursion()` 関数が値 0 を引数として呼び出される。
- 0 は 10 より小さいので、`recursion()` は自分自身を値 $i+1$ (i は現在 0) で呼び出す。これで i は 1 になる。
- その後、`recursion()` は再び自分自身の関数を呼び出すが、今度は引数として値 2 を使う。
- `recursion()` が値 10 で呼び出されるまで同じプロセスが繰り返される。
- 引数の値が 10 に達すると、`recursion()` の再帰呼び出しから戻る。
- `recursion()` は呼び出し元に戻り、そこで `printf()` 文を実行して「9」を表示し、さらに前の呼び出しに戻る。
- その後も同じプロセスが繰り返され、全ての呼び出しが戻った段階でプログラムが終了する。

再帰呼び出しの例 1

階乗の計算

- $1! = 1$
- $2! = 1! \times 2$
- $3! = 2! \times 3 = (1! \times 2) \times 3$

一般には,

$$\begin{aligned}n! &= (n-1)! \times n \\ &= (n-2)! \times (n-1) \times n \\ &= \dots\end{aligned}$$

```
#include <stdio.h>

int factorial( int n ){
    int m;
    if( n==0 || n==1 ){
        printf("1");
        return 1;
    }else{
        printf("%d *(",n);
        m = n * factorial( n-1 );
        printf(")");
        return m;
    }
}

int main(void){
    N= 5;
    factorial(N);
    return 0;
}
```

再帰呼び出しの例 1

階乗の計算

- $1! = 1$
- $2! = 1! \times 2$
- $3! = 2! \times 3 = (1! \times 2) \times 3$

一般には,

$$\begin{aligned}n! &= (n-1)! \times n \\ &= (n-2)! \times (n-1) \times n \\ &= \dots\end{aligned}$$

```
#include <stdio.h>

int factorial( int n ){
    int m;
    if( n==0 || n==1 ){
        printf("1");
        return 1;
    }else{
        printf("%d *(",n);
        m = n * factorial( n-1 );
        printf(")");
        return m;
    }
}

int main(void){
    N= 5;
    factorial(N);
    return 0;
}
```

再帰呼び出しの例 2

フィボナッチ数列

- $f(0) = 0$
- $f(1) = 1$
- $f(2) = f(1) + f(0) = 1$
- $f(3) = f(2) + f(1) = 2$
- $f(n) = f(n-1) + f(n-2)$

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

再帰呼び出しの例 2

フィボナッチ数列

- $f(0) = 0$
- $f(1) = 1$
- $f(2) = f(1) + f(0) = 1$
- $f(3) = f(2) + f(1) = 2$
- $f(n) = f(n-1) + f(n-2)$

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

再帰の必要性

- 階乗の計算やフィボナッチ数列は、再帰を使わなくても計算することができる。
- あらかじめ決められた手順を繰り返すのであれば、繰り返し（ループ）構文で表現することができる。
- 再帰のプログラムをループで書き直すことを再帰除去するという。
- 再帰除去されたプログラムは、再帰呼び出しがなくなるので、一般的に効率が良くなる。

再帰除去の例 1

【再帰を使った例】

```
int factorial( int n ){
    int m;
    if( n==0 || n==1 ){
        printf("1");
        return 1;
    }else{
        printf("%d *(",n);
        m = n * factorial( n-1 );
        printf(")");
        return m;
    }
}
```

【再帰除去した例】

```
int factorial( int n ){
    int i, m=1;

    printf("1");

    for( i=2; i<=n; i++){
        printf(" *%d",i );
        m = m * i;
    }
    return m;
}
```

再帰除去の例 2

【再帰を使った例】

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

【再帰除去した例】

```
int fibonacci( int n ){
    int i, fn, fn1=1, fn2=0;
    if( n==0 ){
        return 0;
    } else if( n == 1){
        return 1;
    } else {
        for( i=2; i<=n;i++){
            fn=fn1+fn2;    fn2=fn1;    fn1=fn;
        }
        return fn;
    }
}
```

演習問題

ファイル名: combination.c

- 異なる n 個の整数から r 個の整数を取り出す組み合わせの数 ${}_nC_r$ を求める関数 combination を作れ. ただし, ${}_nC_r$ は以下のように定義されるものとする.

$${}_nC_r = {}_{n-1}C_{r-1} + {}_{n-1}C_r, \quad {}_nC_0 = {}_nC_n = 1, \quad {}_nC_1 = n$$

- 表示は以下のようにせよ.

Input n : 20

Input r : 3

20 C 3 = 1140

解答例

```
#include <stdio.h>
```

```
int main(void){
```

```
    int n, r;
```

```
    printf("Input n : ");
```

```
    scanf("%d",&n);
```

```
    printf("Input r : ");
```

```
    scanf("%d",&r);
```

```
    printf(" %d C %d =%d \n",n,r,combination(n,r));
```

```
    return 0;
```

```
}
```

解答例

```
#include <stdio.h>
```

```
int main(void){
```

```
    int n, r;
```

```
    printf("Input n : ");
```

```
    scanf("%d",&n);
```

```
    printf("Input r : ");
```

```
    scanf("%d",&r);
```

```
    printf(" %d C %d =%d \n",n,r,combination(n,r));
```

```
    return 0;
```

```
}
```

解答例

```
#include <stdio.h>
```

```
int main(void){
```

```
    int n, r;
```

```
    printf("Input n : ");
```

```
    scanf("%d",&n);
```

```
    printf("Input r : ");
```

```
    scanf("%d",&r);
```

```
    printf(" %d C %d =%d \n",n,r,combination(n,r));
```

```
    return 0;
```

```
}
```

解答例

```
#include <stdio.h>

int combination(int n, int r){

}

int main(void){
    int n, r;

    printf("Input n : ");
    scanf("%d",&n);

    printf("Input r : ");
    scanf("%d",&r);

    printf(" %d C %d =%d \n",n,r,combination(n,r));

    return 0;
}
```

解答例

```
#include <stdio.h>

int combination(int n, int r){
    if(r==0||r==n){ // nC0 = nCn = 1
        return 1;
    }
}

int main(void){
    int n, r;

    printf("Input n : ");
    scanf("%d",&n);

    printf("Input r : ");
    scanf("%d",&r);

    printf(" %d C %d =%d \n",n,r,combination(n,r));

    return 0;
}
```

解答例

```
#include <stdio.h>

int combination(int n, int r){
    if(r==0||r==n){ //  $nC0 = nCn = 1$ 
        return 1;
    }
    if(r==1){        //  $nC1 = n$ 
        return n;
    }
}

int main(void){
    int n, r;

    printf("Input n : ");
    scanf("%d",&n);

    printf("Input r : ");
    scanf("%d",&r);

    printf(" %d C %d =%d \n",n,r,combination(n,r));

    return 0;
}
```

解答例

```
#include <stdio.h>

int combination(int n, int r){
    if(r==0||r==n){ //  $nC0 = nCn = 1$ 
        return 1;
    }
    if(r==1){        //  $nC1 = n$ 
        return n;
    }
    return combination(n-1,r-1)+combination(n-1,r);
}

int main(void){
    int n, r;

    printf("Input n : ");
    scanf("%d",&n);

    printf("Input r : ");
    scanf("%d",&r);

    printf(" %d C %d =%d \n",n,r,combination(n,r));

    return 0;
}
```

まとめ

- 再帰呼び出しを理解する
 - 再帰関数
 - 再帰の例
 - 再帰除去