20\_基本情報アルゴリズム⑥

**課題1**

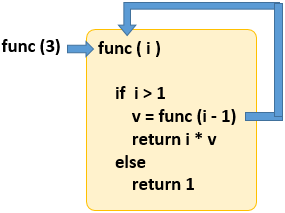
入力した整数の階乗を再帰関数を使用して求め、表示するプログラムを作成しなさい。

また、入力値の範囲は0~10とし、整数以外が入力された場合の例外処理も実装しなさい。

※再帰関数とは

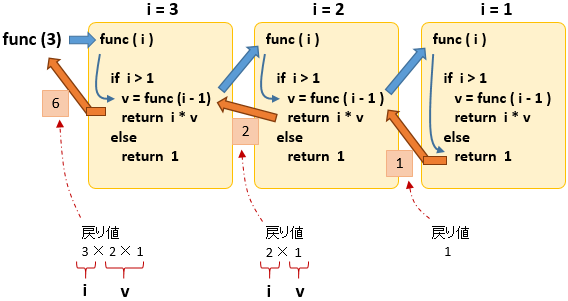
プログラミングの手法の1つで、関数の中に自分自身の関数呼び出しが含まれているもの。

ループ文と同様に、同じ処理を複数回行う場合に利用される。

（例）3の階乗を求める場合

func(3) = 3 x 2 x 1

◆再帰処理の流れ



①func(3)を呼び出す。(i = 3)

②3 > 1なので、自身func(i - 1)を呼び出す。(3 - 1)

③2 > 1なので、同様に自身func(i - 1)を呼び出す。(2 - 1)

④1 > 1ではないので、elseのreturn 1を返す。

⑤返された1が③のvに代入され、return i \* vを返す。（2 \* 1 = 2）

⑥返された2が②のvに代入され、return i \* vを返す。（3 \* 2 = 6）

⑦返された6がfunc(3)の結果となる。

ファイル名：Kad20\_1.java

＜実行結果＞

階乗を求めます(0~10)＞10

10の階乗は3628800です。

＜実行結果＞

階乗を求めます(0~10)＞6

6の階乗は720です。

＜実行結果＞

階乗を求めます(0~10)＞11

入力値が範囲外です。

＜実行結果＞

階乗を求めます(0~10)＞abc

入力値が不正です。

|  |  |  |  |
| --- | --- | --- | --- |
|  | 戻り値 | 名前 | 引数 |
| メソッド | int | func | int i |

◆funcメソッドの処理

引数で受け取った整数の階乗を再帰呼び出しを使用して求め、結果を返す。

例えば5の場合、5×4×3×2×1で最終的に120を返し、6の場合は720を返すことになる。

◆mainメソッドの処理

①階乗計算したい整数を入力し、int型変数inNumに代入する。

この際、整数以外が入力された場合の例外処理をtry-catch文で行い、以降の処理は行わない。

②入力値の範囲チェックを行い、範囲内の場合はfuncメソッドを呼び出し結果を表示する。

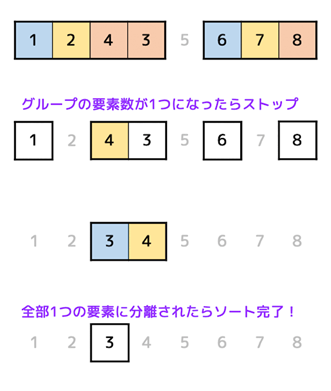
**課題2**

数値配列に対してクイックソートを使用し、昇順に並び替えて表示するプログラムを作成しなさい。

並び替えの途中経過も表示すること。

◆クイックソートの流れ





このようにクイックソートは基準値と相対的に位置を決めて無駄な交換が少ない為、最も速い

ソート法と言われている。ただし、ほぼ整列されているデータ列に対しては効果を発揮しない。

基準値とデータ列の並び方によっては、無駄な比較が増えてしまう可能性もあるので、

時間計算量は最良パターンだと*О(n log n)*だが、最悪パターンだと*О(n²)*になる。

ファイル名：Kad20\_2.java

＜実行結果＞

\*\*\*\*\*\*\*\*\* 元データ \*\*\*\*\*\*\*\*\*

2, 5, 8, 1, 4, 7, 3, 6, 9, 0,

\*\*\*\* クイックソート開始 \*\*\*\*

2, 0, 3, 1, 4, 7, 8, 6, 9, 5,

0, 2, 3, 1, 4, 7, 8, 6, 9, 5,

0, 2, 1, 3, 4, 7, 8, 6, 9, 5,

0, 1, 2, 3, 4, 7, 8, 6, 9, 5,

0, 1, 2, 3, 4, 5, 6, 8, 9, 7,

0, 1, 2, 3, 4, 5, 6, 8, 9, 7,

0, 1, 2, 3, 4, 5, 6, 8, 7, 9,

0, 1, 2, 3, 4, 5, 6, 7, 8, 9,

\*\*\*\* クイックソート完了 \*\*\*\*

|  |  |  |  |
| --- | --- | --- | --- |
|  | 戻り値 | 名前 | 引数 |
| メソッド | void | quickSort | int[] data, int left, int right |

◆quickSortメソッドの処理

引数で受け取った配列の値をクイックソートを使用して昇順に並び替える。

①引数leftがright以上になった場合は以降の処理を行わずに呼び出し元にreturnする。

②以下の変数を宣言する。

//基準値

int pivot = data[(left + right) / 2];

//探索用要素

int searchLeft = left, searchRight = right;

③クイックソートを行う、コメントを記述しながら省略部分を埋めなさい。

//探索用要素が重なるまでループする

while(searchLeft < searchRight){

//基準値以上の値が見つかるまで左から順に探索する

while(data[searchLeft] < pivot){

searchLeft++;

}

//基準値以下の値が見つかるまで右から順に探索する

～省略～

//探索要素が交差した場合は交換せずにループを抜ける

if(searchLeft > searchRight){

break;

}

//探索した基準値より大きい値と小さい値を入れ替える

int tmp = data[searchLeft];

data[searchLeft] = data[searchRight];

data[searchRight] = tmp;

//探索開始位置を更新する

searchLeft++;

～省略～;

}

//配列の値を表示

～省略～;

//基準値より左側を再帰呼び出しでクイックソートする

quickSort(data, left, searchRight);

//基準値より右側を再帰呼び出しでクイックソートする

～省略～;

|  |  |  |  |
| --- | --- | --- | --- |
|  | 戻り値 | 名前 | 引数 |
| メソッド | void | printArray | int[] data |

◆printArrayメソッドの処理

引数で受け取った配列の値を表示する（過去の課題で作成した同じメソッドをコピーし流用する）

◆mainメソッドの処理

①以下の数値配列を宣言する。

int[] numbers = {2, 5, 8, 1, 4, 7, 3, 6, 9, 0};

②数値配列を引数にprintArrayメソッドを呼び出し、ソート前の配列の値を表示する。

System.out.println("\*\*\*\*\*\*\*\*\* 元データ \*\*\*\*\*\*\*\*\*");

printArray(～省略～);

③数値配列, 先頭要素番号, 末尾要素番号を引数にquickSortメソッドを呼び出し、ソートする。

System.out.println("\*\*\*\* クイックソート開始 \*\*\*\*");

quickSort(numbers, 0, ～省略～);

System.out.println("\*\*\*\* クイックソート完了 \*\*\*\*");