

正規表現・分割統治と再帰

リカーシブ

- ① リカーシブ(再帰呼出)は
プログラムの中から
自分自身を呼び出すことを再帰呼出という。
- ② 再帰呼出には、
自分自身を定義するのに自分自身よりも
1次低い集合を用いる。
- ③ その部分集合はより低次の部分定義を用いて
定義することを繰り返して表現する。

再帰からの脱出

① 再帰呼出には

- ① 再帰からの脱出口がなければならない。
- ② 脱出口がない場合、再帰呼出は永遠に続く。

② 再帰呼出を行うときは、

- ① 呼んだ自分自身から制御が戻ってきたときに、
- ② 前の状態で引き続き実行できるように、
- ③ 実行アドレスや引数、使用していた変数などの情報を保管しておく。

③ 後入先出の特徴を持つスタックを使用する。

再帰呼び出しの具体例

① フィボナッチの数列

1 1 2 3 5 8 13 21 34 55 89

② フィボナッチの数列の定義

$$F_n = F_{n-1} + F_{n-2} \quad n \geq 3$$

$$F_1 = F_2 = 1 \quad n = 1 \text{ または } 2$$

フィボナッチ数列の処理

- ① フィボナッチの数列は再帰呼び出しを使用して処理ができる。
- ② 処理手順
 - ① 定義した関数 F_n をスタックに格納、
 - ② 関数 F_n の中で使用している関数 F_{n-1} を次に格納、
 - ③ 順次1次低い定義関数を格納していく。
 - ④ 最後に既知の値 $F_2=1$ 、 $F_1=1$ にたどり着き、
 - ⑤ 再帰呼び出しから脱出して、
 - ⑥ 格納した順序とは逆に各関数の値を求める。

スタックを利用した解答手順

- ① 次の各式を逐次スタックにPUSHする。

$$F_n = F_{n-1} + F_{n-2}$$

$$F_{n-1} = F_{n-2} + F_{n-3}$$

.....

$$F_4 = F_3 + F_2$$

$$F_3 = F_2 + F_1$$

② $F_2 = F_1 = 1$ であるから、
 $F_2 + F_1 = 2$ となり、再帰呼出しから脱出する。

③ スタックの各式をPOPLして値を求める。

$$F_3 = F_2 + F_1 = 2$$

$$F_4 = F_3 + F_2 = 3$$

.....

$$F_{n-1} = F_{n-2} + F_{n-3}$$

④ $F_n = F_{n-1} + F_{n-2}$ を利用して
 F_n を求めることができる。

バックス記法とは

- ① バックス記法は言語を定義するための言語である。
- ② 文章をどのようにして構成するかという
構文規則を示している。
- ③ バックス記法の「::=」や「|」を超記号と呼ぶ。
- ④ 「<」と「>」で囲まれたものを構文要素と呼び、
これを構文則という。

構文則によって定義される文章

- ① <文>から始まる。
- ② 構文要素をその右辺のものと置き換えて、
構文要素を一つも含まなくなったものである。

表記法


① $\langle \text{数字} \rangle ::= 0 \mid 1 \mid 2 \mid 3 \mid 4 \mid 5 \mid 6 \mid 7 \mid 8 \mid 9$

② $\langle \text{英字} \rangle ::= A \mid B \mid C \mid D \mid E \mid F \mid G \mid H \mid I \mid J$
 $\quad \quad \quad \mid K \mid L \mid M \mid N \mid O \mid P \mid Q \mid R \mid S$
 $\quad \quad \quad \mid T \mid U \mid V \mid W \mid X \mid Y \mid Z$

③ $\langle \text{名前} \rangle ::= \langle \text{英字} \rangle \mid \langle \text{名前} \rangle \langle \text{英字} \rangle$
 $\quad \quad \quad \mid \langle \text{名前} \rangle \langle \text{数字} \rangle$

表記法の内容説明

- ① 表記法の①は
「<数字>は0~9のどれかである」
- ② 表記法の②は「<英字>はA~Zのどれかである」
- ③ 「|」は「または」という意味である。
- ④ 表記法の③の最初の<名前> ::= <英字>は
「英字は名前である」と読む。

- 
- ⑤ Aという英字は名前として使うことができる。
 - ⑥ | <名前><英字>は
「または、名前の次に英字をつけたものも名前である」と読む。
 - ⑦ Aは名前であるから、
その次に英字AをつけたものAAも名前である。
 - ⑧ AAは名前であるから、
その次に英字YをつけたものAAYも名前である。

- ⑨ | <名前><数字>は
「または、名前の次に数字をつけたものも名前である」と読む。
- ⑩ Bは名前であるから、
その次に数字をつけたB3は名前である。
- ⑪ B3は名前であるから、
その次に数字9をつけたものB39も名前である。
- ⑫ 名前は英字一つまたは英字で始まり
その後英字又は数字をいくつかつけたものである。

バックス記法の具体例

日本語の文章は主部と述部からなり、主部は名詞と助詞からなり、述部は動詞からなるとする。名詞としては、私、君、犬、助詞としては、は、が、も、動詞としては、遊ぶ、泳ぐ、走る、があるものとし、これをバックス記号で表すと次のようになる。

- <文> ::= <主部> <述部>
- <主部> ::= <名詞> <助詞>
- <述部> ::= <動詞>
- <名詞> ::= 私 | 君 | 犬
- <助詞> ::= は | が | も
- <動詞> ::= 遊ぶ | 泳ぐ | 走る

「私は遊ぶ」という文章

- ① <文> ::= <主部> <述部>
 - | <名詞> <助詞> <述部>
 - | <名詞> <助詞> <動詞>
 - | 私 <助詞> <動詞>
 - | 私は <動詞>
 - | 私は遊ぶ

- ② 「私」、「は」、「遊ぶ」を
 終端記号、構文要素を非終端記号という。

正規表現とは

- ① 正規表現は、
字句をパターンの集合で表す式で
定義する表現方法である。
- ② 文字列に対し、
メタキャラクタといわれる特殊な記号を使用し、
組み合わせて任意の文字列を表現する。

主な正規表現

記号	意味	例
^	行の先頭	^A 行頭がAの文字で始まる
\$	行の末尾	Z\$ 行末がZの文字で終わる
.	任意の1文字	M..N Mで始まりNで終わる4文字
[]	[]内に含まれる任意の1文字	[A-Z] A~Zの範囲の文字
*	直前文字の0回以上の繰返し	ABX* Xを0回以上繰返す
+	直前文字の1回以上の繰返し	ABX+ Xを1回以上繰返す

分割統治とは

① 分割統治法は、

- ① 解くべき問題を小規模な問題に分割する。
- ② 各部分問題の解を結合することによって、
- ③ 全体の解を求めようとする解決法である。

② 分割というのは、

- ① 対象とする変数の集合をいくつかに分けたり、
- ② 定義領域を分けることであり、
- ③ 多くの場合、再帰的に反復して利用する。

分割統治の考え方の利用

- ① クイックソートやマージソートは
分割統治の考え方を利用したアルゴリズムであり、
システム分析やプログラム設計にも
この考え方が用いられている。
- ② モジュール分割は
大きなプログラムを主たる部分と
データ入出力やデータチェック部分に分割し、
各モジュールを個別に作成し、テストし、
各モジュールを結合して完全なプログラムを作成する。