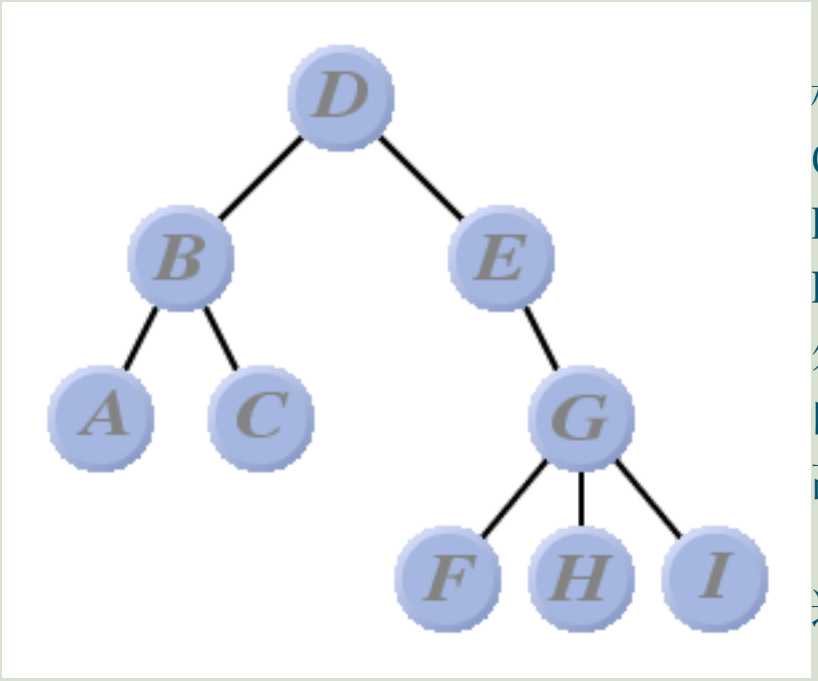
アルゴリズムとデータ構造
演習第 6 回
ツリー 1 （解析木）

今回は木 （アルゴリズム C 第 1 巻 p.41） についてです。木の構造や、辿り方について理解してください。

問題 1 ［印刷用 PostScript］

(1) 次の木の名称が、図のどの節点（ノード）に該当するかを答えなさい。



根（root）：
G の親（parent）：
B の子（child）：
H の兄弟（sibling）：
外部節点（external node）：
内部節点（internal node）：
高さ（height）：
C のレベル（level）：
道長（path length）：

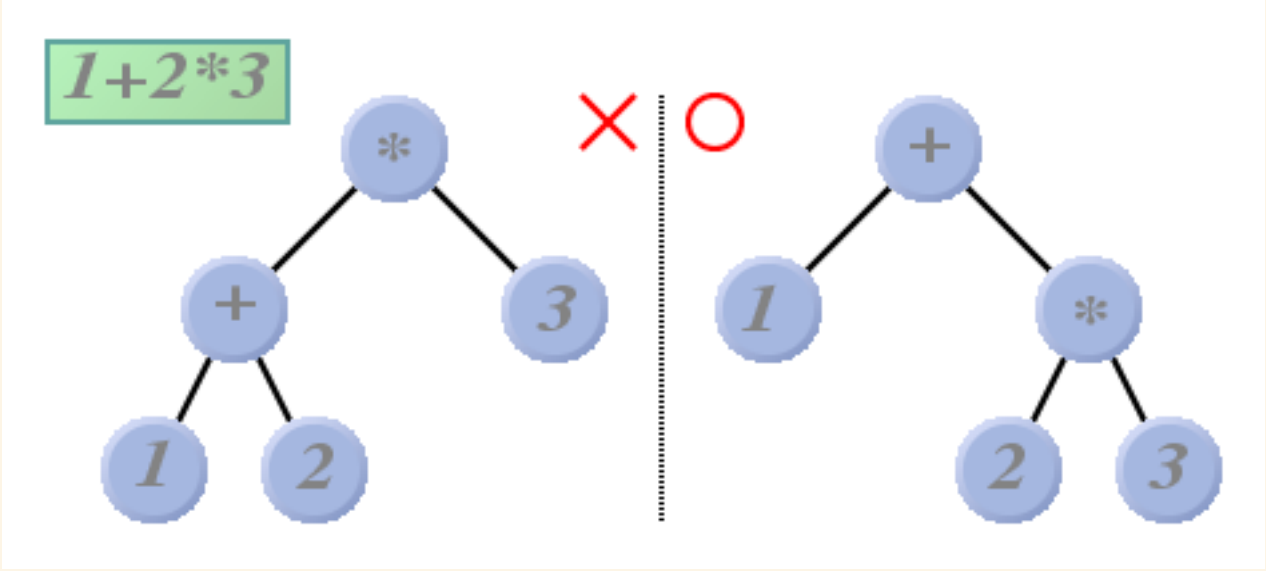
(2) 次の中置記法で書かれた式の解析木を書きなさい。

```
4*(3+2)+1
```

(3) (2) で作成した木を、preorder、inorder、postorder でトラバースした結果を書きなさい。

(1) は用語 （アルゴリズム C 第 1 巻 p.42） に関する問題です。木は用語が多いので、教科書で用語の意味をよく確認して答えてください。

(2) は解析木 （アルゴリズム C 第 1 巻 p.47） を作る問題です。例えば、1+2*3 の解析木は このようになります。



左側の木は間違いです。この木で計算すると答えが 9 になってしまいます。 演算の優先順位に気をつけて気を作ってください。

(3) はトラバース （アルゴリズム C 第 1 巻 p.51） する問題です。三種類の辿り方によって、ノードを通る順番が 変わります。先程の解析木の例では、このようになります。

```
   preorder: +1*23
    inorder: 1+2*3
  postorder: 123*+
```

preorder、inorder、postorder の結果はそれぞれ、 前置記法、（括弧のない）中置記法、後置記法になります。

問題 2

ノードの中身を表示しながらトラバースする関数

```
void preorder(NodePointer node);
void inorder(NodePointer node);
void postorder(NodePointer node);
```

を再帰を使って書きなさい。ファイルは ex06-2-skel.c を使用すること。

実行例：

```
% ./a.out
preorder:  + 1 * 2 3
inorder:   1 + 2 * 3
postorder: 1 2 3 * +
```

演習第 2 回のリストを応用して、木を作ります。 ex06-2-skel.c をよく読んでから、 関数を作成してください。

まず構造体 node は次のようになっています。

```
struct node {
  struct node *right;
  char key;
  struct node *left;
};
```

right が右のノードへのポインタ、 *left* が左のノードへのポインタ、 *key* がノードの持つ

値です。 実際にノードを作るのは makenode 関数で、malloc をして、左右には tail ノード （何も繋っていない状態）を代入しています。

main 関数の前半で 1+2*3 の解析木を作成しています。 どのように木が繋がっているかをよく確認してください。 通常はこのような木の作り方はせず、演習第７回でやるように、木に繋ぐ関数を作って繋いでいきます。

トラバースする関数は再帰を用いて作成します。 左右に辿るときに tail かどうかを調べて、 tail でなければ再帰して木を辿ります。

問題３

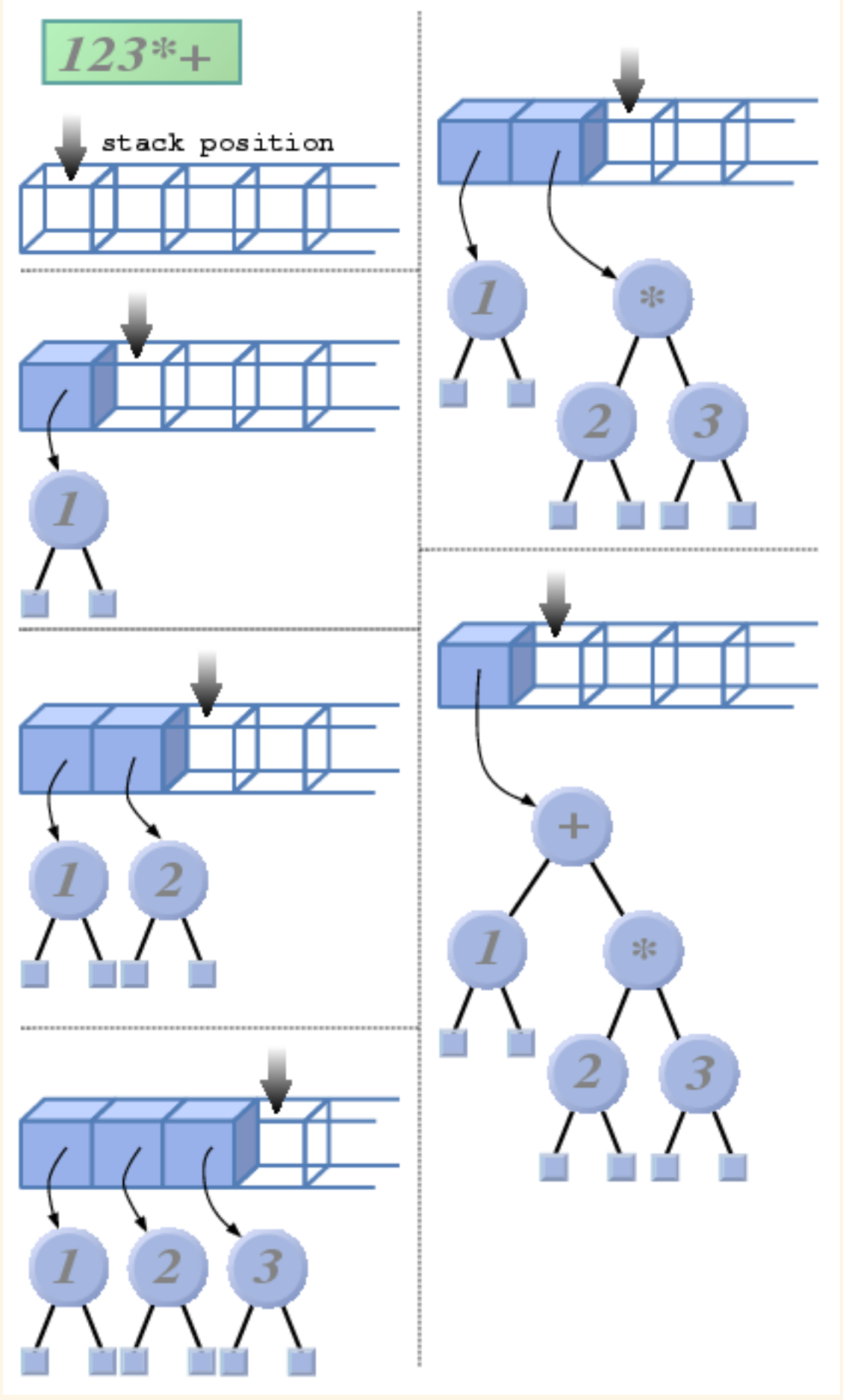
後置記法から解析木を作成するプログラムを書きなさい。 必要であれば、演習第３回で使用した関数などを 用いても良い。

実行例：

```
% ./a.out
Input data by Reverse Polish Notation: 123*+
preorder:  + 1 * 2 3
inorder:   1 + 2 * 3
postorder: 1 2 3 * +
```

解析木を作るには、演習第３回でやったスタック を使います。処理の流れはほぼ同じです。計算を行なう代わりに木として 繋げていきます。

1. 文字を読みこんだら、ノードを作成して、その文字をノードに入れる
2. 1. その文字が数字ならば、そのノードをスタックにプッシュする
   2. その文字が記号ならば、二回ポップし （一回目にポップしたものを記号のノードの右に、 二回目にポップしたものを記号のノードの左に繋ぐ）、 その記号のノードを再びプッシュする



演習第３回の問題２ と大きく異なるのはスタックの扱うデータの型です。演習第３回ではスタックには int 型を入れていましたが、ここでは NodePointer 型を入れることになります。 配列を宣言するときの型や、push 関数や pop 関数の引数や返り値の型に 気を付けてください。

Written by わかまつなおき