第 11 回 プログラミング応用レポート

15302114 番　山下尚人

提出日：2018 年 1 月 25 日

課題

- ソースコード

```c
#include <stdio.h>
#include <stdlib.h>
#include <sysexits.h>

typedef struct node {
  struct node *left;
  struct node *right;
  int label;
} node_t;

node_t *allocNode(void);
void setNode(node_t *n, int x, node_t *left, node_t *right);
node_t *insertNode(node_t *n, int x);
void printTree(node_t *n);

int main(void){
  //3,8,6,5,4,9,2,7,1
  node_t *root;
  root = allocNode();
  setNode(root, 3, NULL, NULL);

  insertNode(root,8);
  printf("*****␣TREE:add␣8␣*****\n");
  printTree(root);

  insertNode(root,6);
  printf("*****␣TREE:add␣6␣*****\n");
  printTree(root);

  insertNode(root,5);
  printf("*****␣TREE:add␣5␣*****\n");
  printTree(root);

  insertNode(root,4);
  printf("*****␣TREE:add␣4␣*****\n");
  printTree(root);

  insertNode(root,9);
  printf("*****␣TREE:add␣9␣*****\n");
  printTree(root);

  insertNode(root,2);
  printf("*****␣TREE:add␣2␣*****\n");
  printTree(root);

  insertNode(root,7);
  printf("*****␣TREE:add␣7␣*****\n");
  printTree(root);

  insertNode(root,1);
  printf("*****␣TREE:add␣1␣*****\n");
  printTree(root);

  return EXIT_SUCCESS;
}

node_t *allocNode(void){
  //大きさsizeof(node_t)バイトを1個分をメモリ領域に確保。
  //mallocではなくcallocは、確保した領域を0で初期化する。
  return (node_t *)calloc(1, sizeof(node_t));
}

```

```
63  void setNode(node_t *n, int x, node_t *left, node_t *right){
64    n->left = left;
65    n->right = right;
66    n->label = x;
67  }
68
69  node_t *insertNode(node_t *n, int x){
70    if (NULL == n) {
71      n = allocNode();
72      setNode(n, x, NULL, NULL);
73    } else {
74      if (x < n->label) {
75        n->left = insertNode(n->left, x);
76      } else if (x > n->label) {
77        n->right = insertNode(n->right, x);
78      } else {
79        // x == n->label
80        printf("ERROR:Registered\n");
81      }
82    }
83    return n;
84  }
85
86  void printTree(node_t *n){
87    static int level = 0;
88    int i;
89    level++;
90
91    if (n->right != NULL) {
92      printTree(n->right);
93    }
94
95    printf("%*c␣[%d]\n", 5*level, '␣', n->label);
96
97    if (n->left != NULL) {
98      printTree(n->left);
99    }
100   level--;
101 }
```

- 実行結果

```
1   ***** TREE:add 8 *****
2                 [8]
3         [3]
4   ***** TREE:add 6 *****
5                 [8]
6                       [6]
7         [3]
8   ***** TREE:add 5 *****
9                 [8]
10                      [6]
11                            [5]
12        [3]
13  ***** TREE:add 4 *****
14                [8]
15                      [6]
16                            [5]
17                                  [4]
18        [3]
19  ***** TREE:add 9 *****
20                      [9]
21                [8]
22                      [6]
23                            [5]
24                                  [4]
25        [3]
26  ***** TREE:add 2 *****
27                      [9]
28                [8]
29                      [6]
30                            [5]
31                                  [4]
32        [3]
33                [2]
34  ***** TREE:add 7 *****
35                      [9]
36                [8]
37                            [7]
38                      [6]
39                            [5]
40                                  [4]
41        [3]
42                [2]
43  ***** TREE:add 1 *****
44                      [9]
```

```
          [8]
                    [7]
               [6]
                    [5]
                         [4]
     [3]
          [2]
               [1]
```

- プリントアウト関数の動作説明

  画面の垂直方向 (行) の順番は、ソースの 92,97 行目の再帰呼び出しと 95 行目の printf 関数により決まる。

  右側のノードの再帰呼び出し、ノードの画面出力、左側のノードの再帰呼び出しの順で処理している。

  よって、「そのノードの右の子の末端へ移動し出力。一つ親のノードへ移動して出力し、ノードの左の子へ移動。」を繰り返している。

  91,96 行目の if 文による条件分岐は再帰呼び出しを終わらせるため。

  画面の水平方向 (列) の空白の数は、ソースの 89,100 行目の変数 level の増減により処理し、95 行目の printf 文により実際に出力している。

  printTree 関数内のローカル変数 level は、static 装飾子により printTree 関数が呼ばれるたびに初期化されず、値が保持される。

  printTree 関数は 92,97 行目で再帰呼び出しされているので、level が 1 増えた後に再帰呼び出しされ、呼び出しが終わるごとに level が 1 減っていく。

  これにより、95 行目の printf 文でノードのレベルの 5 倍の空白と、label を出力して改行している。アスタリスク (*) は printf で出力する文字のフィールドの幅を、引数に渡すことで指定できる。