



ad-3. 二分木と走査

アルゴリズムとデータ構造

(スライド, 全11回)

<https://www.kkaneko.jp/cc/ae/index.html>

金子邦彦





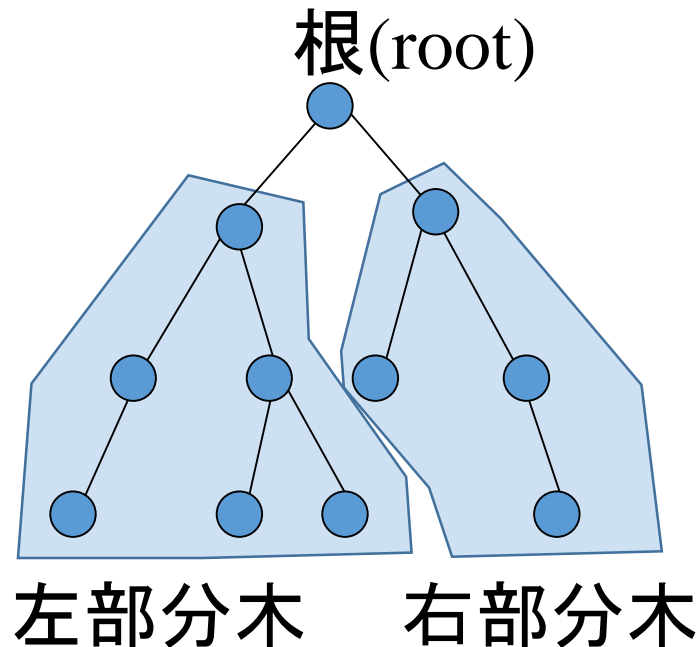
3-1. 二分木と走査



二分木とは

レコードを次の3つで構成

- 要素を格納するセル
- 左部分木を指すポインタを格納するセル
- 右部分木を指すポインタを格納するセル



二分木の走査法



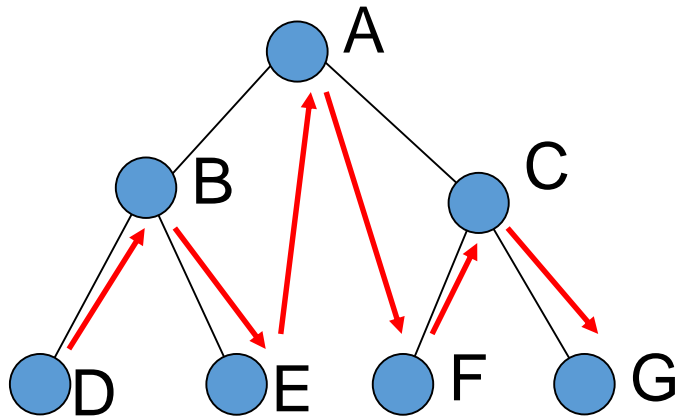
二分木の走査には次の種類がある

- 先行順 (pre-order)
- 中間順 (in-order)
- 後行順 (post-order)

先行順 (pre-order)



- 根ノード, 左部分木, 右部分木の順

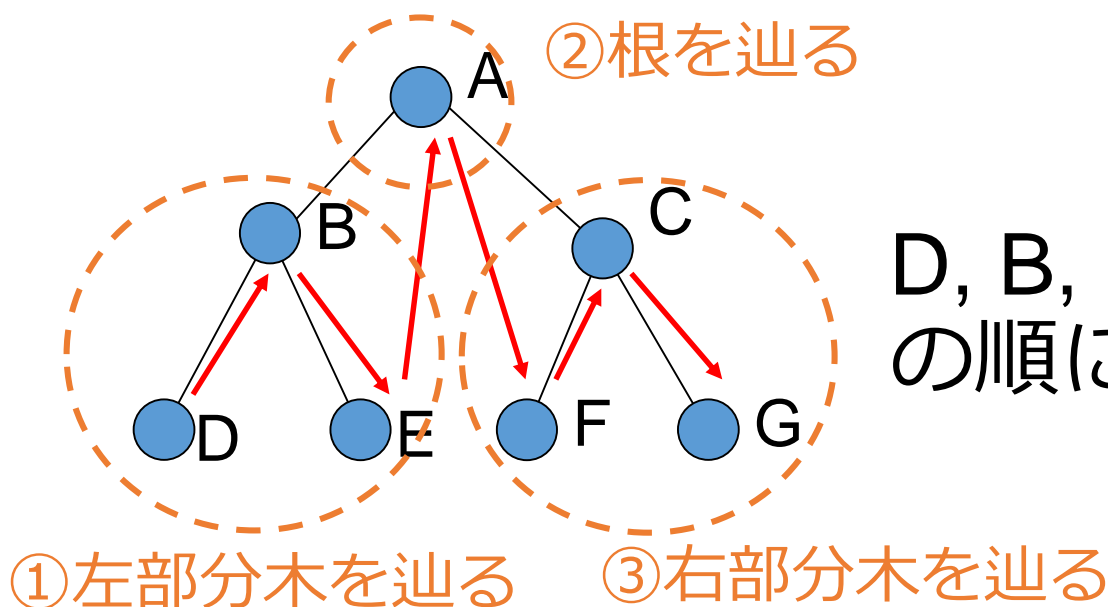


D, B, E, A, F, C, G
の順に処理を行う



中間順 (in-order)

- 左部分木, 根ノード, 右部分木の順

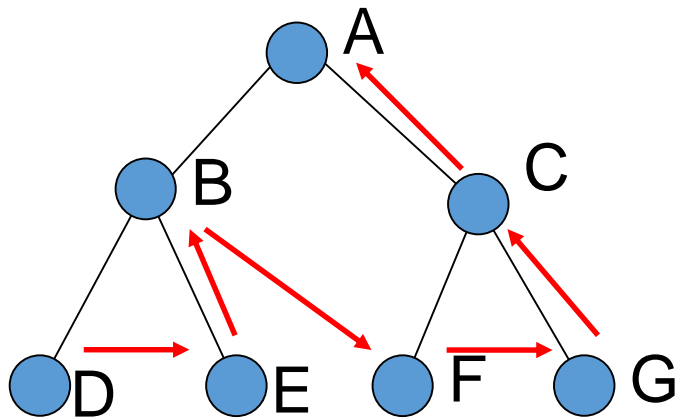


D, B, E, A, F, C, G
の順に処理を行う



後行順 (post-order)

- 左部分木, 右部分木, 根ノードの順



D, E, B, F, G, C, A
の順に処理を行う

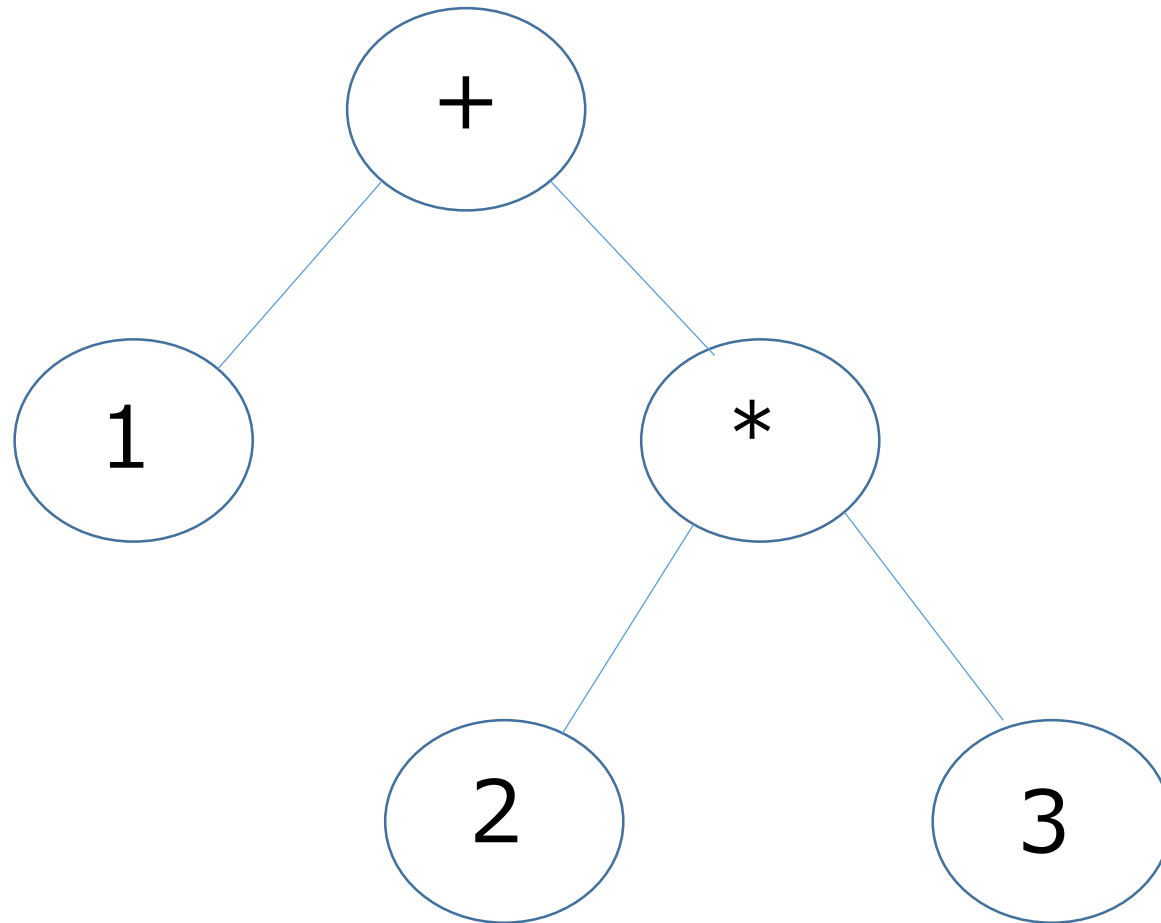


3-2. 実習

実習の指示

- 資料： **10** ~ **17**
- 次のことを理解しマスターする
 - 二分木の作成

ここで作成する木



実習



① **ウェブブラウザ**を起動する

② **C Tutor** を使いたいので, 次の URL を開く

<http://www.pythontutor.com/>

※ Internet Explorer でうまく動かない場合がある

→ うまく動かないときは Google Chrome を試してください

※ 途中で 「Server Busy . . .」 というメッセージが出る
ことがある.

→ 混雑している. 少し (数秒から数十秒) 待つと自動で表示
が変わる (変わらない場合には, 操作をもう一度行ってみ
る)

※ 日本語モードはない. 英語で使う



③ 「C Tutor」をクリック

← → ↻ ⓘ 保護されていない通信 | pythontutor.com ☆ APP 2 🔒 📄 📂 📁

VISUALIZE CODE AND GET LIVE HELP

Learn Python, Java, C, C++, JavaScript, and Ruby

[Python Tutor](#) (created by [Philip Guo](#)) helps people overcome a fundamental barrier to learning programming: understanding what happens as the computer runs each line of code.

Write code in your web browser, see it visualized step by step, and get live help from volunteers.

Related services: [Java Tutor](#), [C Tutor](#), [C++ Tutor](#), [JavaScript Tutor](#), [Ruby Tutor](#)

Over five million people in more than 180 countries have used Python Tutor to visualize over 100 million pieces of code, often as a supplement to textbooks, lectures, and online tutorials.

[Visualize your code and get live help now](#)



Write code in

C (gcc 4.8, C11) ▾

「C (gcc4.8, C11)」になっている

```
1 int main() {  
2  
3     return 0;  
4 }
```

最初から main メソッドの
ひな形が入っている

エディタ

Help improve this tool by completing a [short user survey](#).

Visualize Execution

実行のためのボタン

双方向リストの作成



④ 次のプログラムを使う

```
1  #include<stdio.h>
2  #include<stdlib.h>
3
4  struct Record {
5      char v;
6      struct Record *left;
7      struct Record *right;
8  };
9
10 int main() {
11     struct Record *r1, *r2, *r3, *r4, *r5;
12     r1 = (struct Record*)malloc(sizeof(struct Record));
13     r2 = (struct Record*)malloc(sizeof(struct Record));
14     r3 = (struct Record*)malloc(sizeof(struct Record));
15     r4 = (struct Record*)malloc(sizeof(struct Record));
16     r5 = (struct Record*)malloc(sizeof(struct Record));
```

次ページに続く



```
18     r1->v = '+';
19     r1->left = r2;
20     r1->right = r3;
21     r2->v = '1';
22     r2->left = NULL;
23     r2->right = NULL;
24     r3->v = '*';
25     r3->left = r4;
26     r3->right = r5;
27     r4->v = '2';
28     r4->left = NULL;
29     r4->right = NULL;
30     r5->v = '3';
31     r5->left = NULL;
32     r5->right = NULL;
33
34     return 0;
35 }
```



② 「Visualize Execution」 をクリック.

「Last」 をクリック.

結果を確認する.

「Edit this code」 をクリックして戻る

```
22 r2->left = NULL;
23 r2->right = NULL;
24 r3->v = '*';
25 r3->left = r4;
26 r3->right = r5;
27 r4->v = '2';
28 r4->left = NULL;
29 r4->right = NULL;
30 r5->v = '3';
31 r5->left = NULL;
32 r5->right = NULL;
33
34 return 0;
35 }
```



```
22 r2->left = NULL;
23 r2->right = NULL;
24 r3->v = '*';
25 r3->left = r4;
26 r3->right = r5;
27 r4->v = '2';
28 r4->left = NULL;
29 r4->right = NULL;
30 r5->v = '3';
31 r5->left = NULL;
32 r5->right = NULL;
33
34 return 0;
35 }
```



Help improve this tool by completing a [short user survey](#)

Visualize Execution

[Show example code and courses](#)

[Edit this code](#)

line that just executed
next line to execute

<< First < Prev Next > **Last >>**

Step 1 of 22

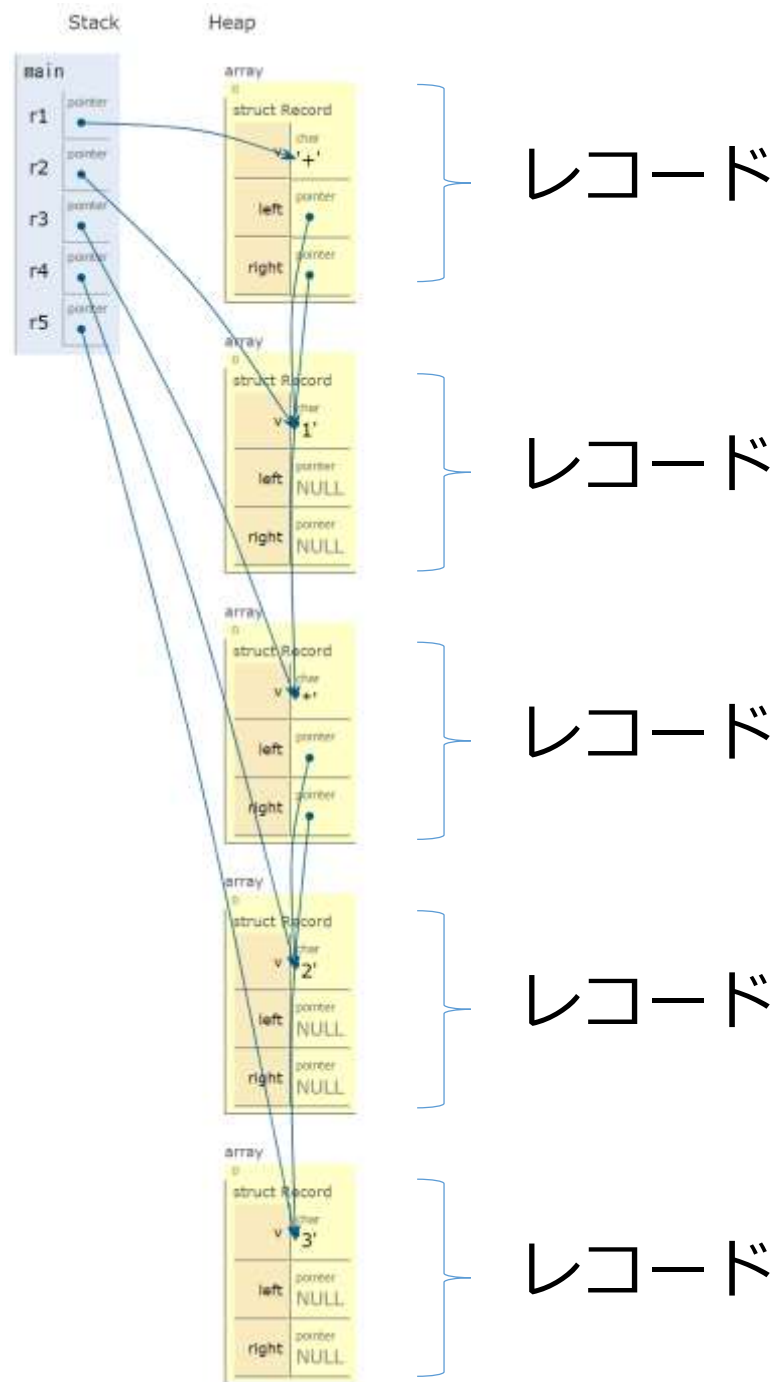
[Edit this code](#)

executed
execute

<< First < Prev Next > **Last >>**

Done running (22 steps)

実行結果





実習の指示

- 資料： **18～27**
- 次のことを理解しマスターする
 - 先行順 (pre-order)
 - 中間順 (in-order)
 - 後行順 (post-order)



先行順 (pre-order)

- プログラムを次のように書き換えなさい

```
1  #include<stdio.h>
2  #include<stdlib.h>
3
4  struct Record {
5      char v;
6      struct Record *left;
7      struct Record *right;
8  };
9
10 void preorder(struct Record *r) {
11     printf("%c\n", r->v);
12     if (r->left != NULL) {
13         preorder(r->left);
14     }
15     if (r->right != NULL) {
16         preorder(r->right);
17     }
18     return;
19 }
```

次ページに続く



```
21 int main() {
22     struct Record *r1, *r2, *r3, *r4, *r5;
23     r1 = (struct Record*)malloc(sizeof(struct Record));
24     r2 = (struct Record*)malloc(sizeof(struct Record));
25     r3 = (struct Record*)malloc(sizeof(struct Record));
26     r4 = (struct Record*)malloc(sizeof(struct Record));
27     r5 = (struct Record*)malloc(sizeof(struct Record));
28
29     r1->v = '+';
30     r1->left = r2;
31     r1->right = r3;
32     r2->v = '1';
33     r2->left = NULL;
34     r2->right = NULL;
35     r3->v = '*';
36     r3->left = r4;
37     r3->right = r5;
38     r4->v = '2';
39     r4->left = NULL;
40     r4->right = NULL;
41     r5->v = '3';
42     r5->left = NULL;
43     r5->right = NULL;
44
45     preorder(r1);
46     return 0;
47 }
```

実行し，結果を確認しなさい

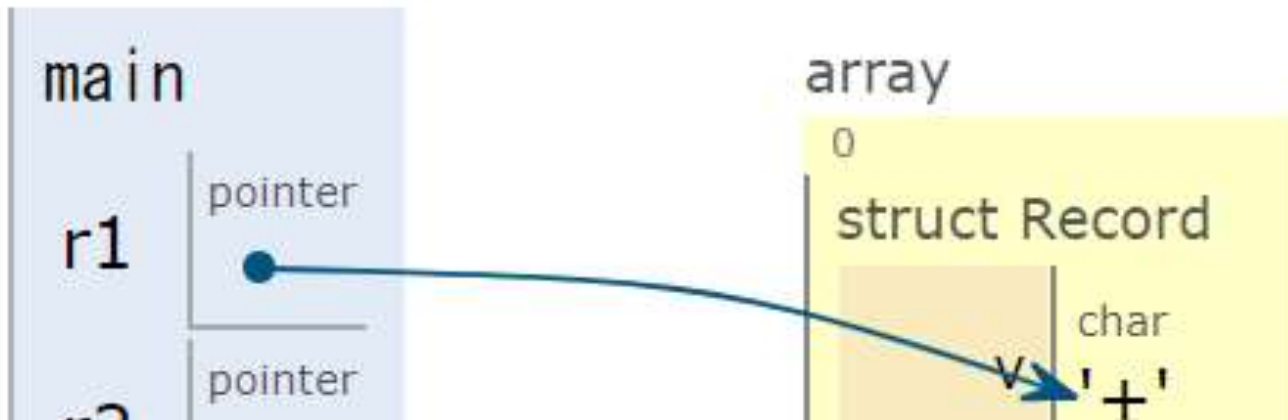


Print output (drag lower right corner to resize)

```
+  
1  
*  
2  
3
```

Stack

Heap





中間順 (in-order)

- プログラムを次のように書き換えなさい

```
1  #include<stdio.h>
2  #include<stdlib.h>
3
4  struct Record {
5      char v;
6      struct Record *left;
7      struct Record *right;
8  };
9
10 void inorder(struct Record *r) {
11     if (r->left != NULL) {
12         inorder(r->left);
13     }
14     printf("%c\n", r->v);
15     if (r->right != NULL) {
16         inorder(r->right);
17     }
18     return;
19 }
```

次ページに続く



```
21 int main() {
22     struct Record *r1, *r2, *r3, *r4, *r5;
23     r1 = (struct Record*)malloc(sizeof(struct Record));
24     r2 = (struct Record*)malloc(sizeof(struct Record));
25     r3 = (struct Record*)malloc(sizeof(struct Record));
26     r4 = (struct Record*)malloc(sizeof(struct Record));
27     r5 = (struct Record*)malloc(sizeof(struct Record));
28
29     r1->v = '+';
30     r1->left = r2;
31     r1->right = r3;
32     r2->v = '1';
33     r2->left = NULL;
34     r2->right = NULL;
35     r3->v = '*';
36     r3->left = r4;
37     r3->right = r5;
38     r4->v = '2';
39     r4->left = NULL;
40     r4->right = NULL;
41     r5->v = '3';
42     r5->left = NULL;
43     r5->right = NULL;
44
45     inorder(r1);
46     return 0;
47 }
```

実行し，結果を確認しなさい

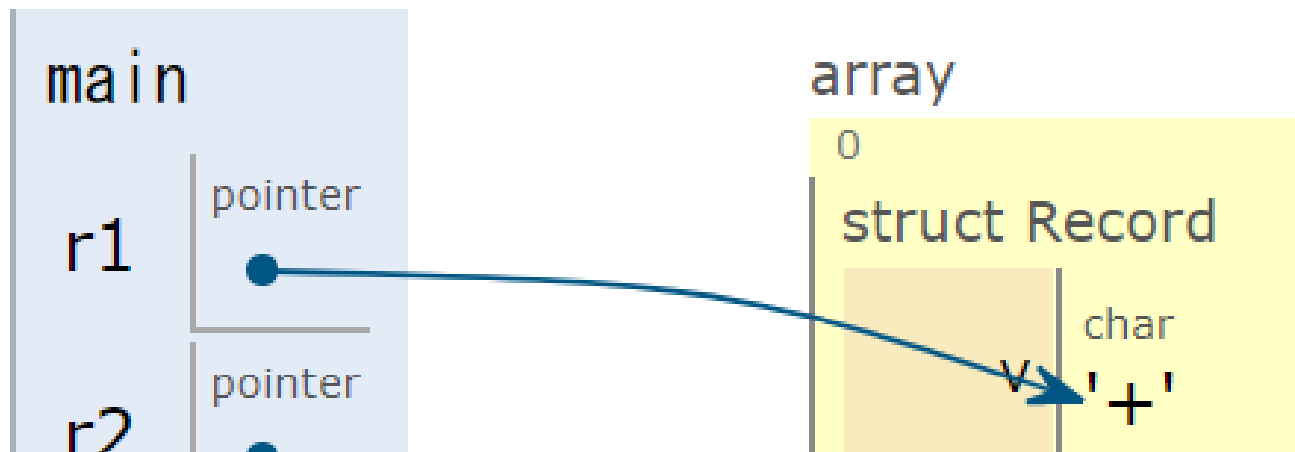


Print output (drag lower right corner to resize)

```
1
+
2
*
3
```

Stack

Heap





後行順 (post-order)

- プログラムを次のように書き換えなさい

```
1  #include<stdio.h>
2  #include<stdlib.h>
3
4  struct Record {
5      char v;
6      struct Record *left;
7      struct Record *right;
8  };
9
10 void postorder(struct Record *r) {
11     if (r->left != NULL) {
12         postorder(r->left);
13     }
14     if (r->right != NULL) {
15         postorder(r->right);
16     }
17     printf("%c\n", r->v);
18     return;
19 }
```

次ページに続く



```
21 int main() {
22     struct Record *r1, *r2, *r3, *r4, *r5;
23     r1 = (struct Record*)malloc(sizeof(struct Record));
24     r2 = (struct Record*)malloc(sizeof(struct Record));
25     r3 = (struct Record*)malloc(sizeof(struct Record));
26     r4 = (struct Record*)malloc(sizeof(struct Record));
27     r5 = (struct Record*)malloc(sizeof(struct Record));
28
29     r1->v = '+';
30     r1->left = r2;
31     r1->right = r3;
32     r2->v = '1';
33     r2->left = NULL;
34     r2->right = NULL;
35     r3->v = '*';
36     r3->left = r4;
37     r3->right = r5;
38     r4->v = '2';
39     r4->left = NULL;
40     r4->right = NULL;
41     r5->v = '3';
42     r5->left = NULL;
43     r5->right = NULL;
44
45     postorder(r1);
46     return 0;
47 }
```

実行し，結果を確認しなさい



Print output (drag lower right corner to resize)

```
1  
2  
3  
*  
+
```

Stack

Heap

