ad-3. 二分木と走査

（C 言語によるアルゴリズムとデータ構造）　（全６回）

URL: https://www.kkaneko.jp/pro/ad/index.html

金子邦彦

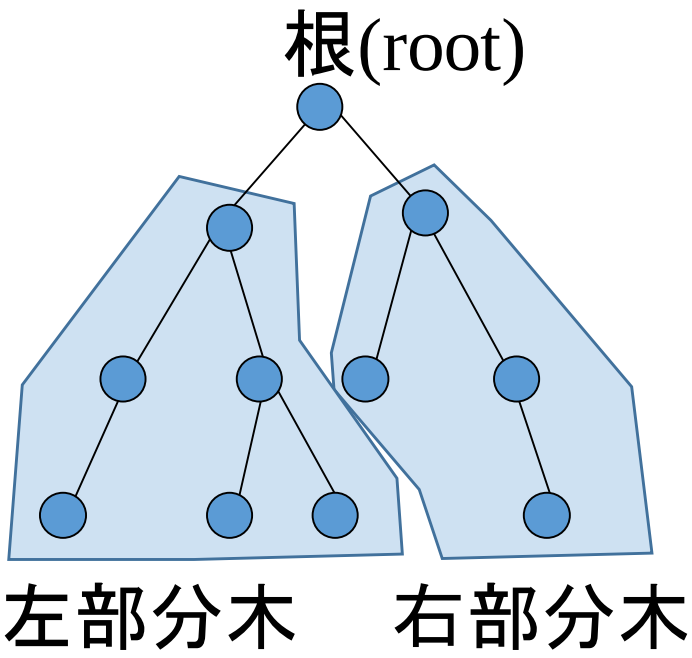
CC BY NC SA

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**



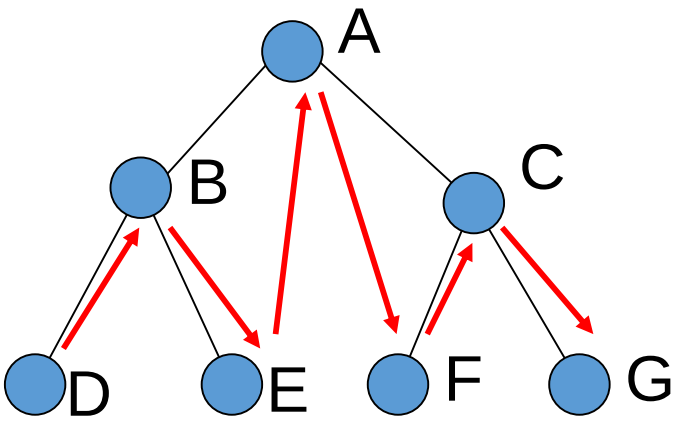
二分木の走査法

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)

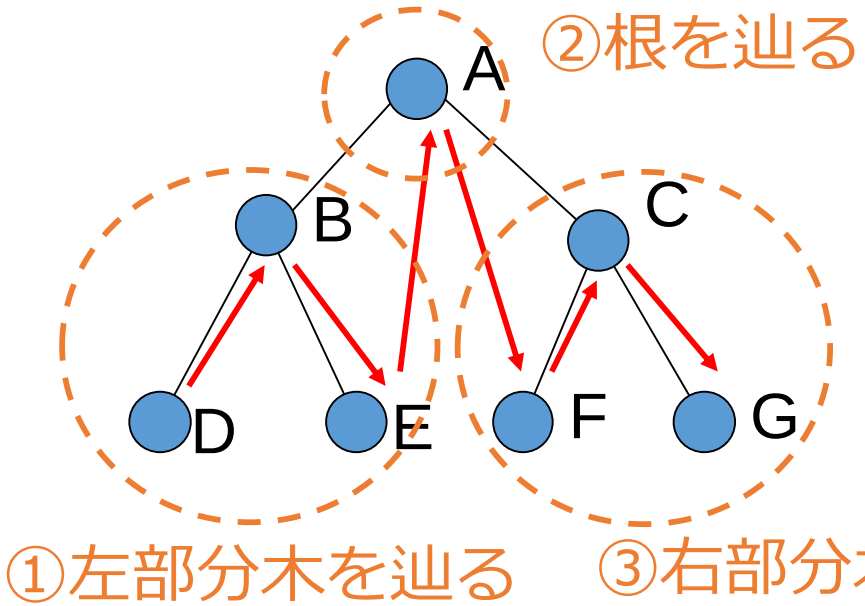
- 根ノード，左部分木，右部分木の順



D, B, E, A, F, C, G
の順に処理を行う

中間順 (in-order)

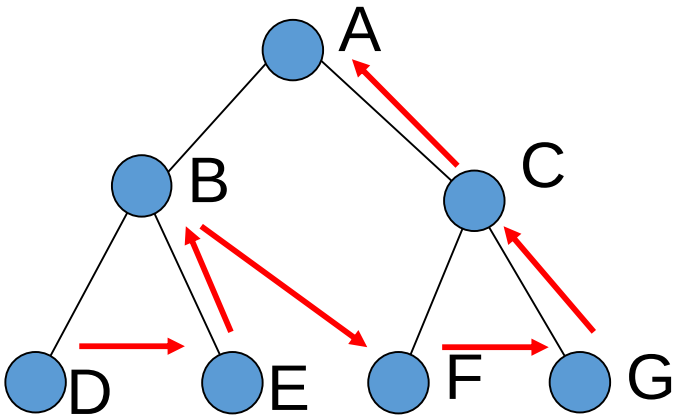
- 左部分木，根ノード，右部分木の順



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



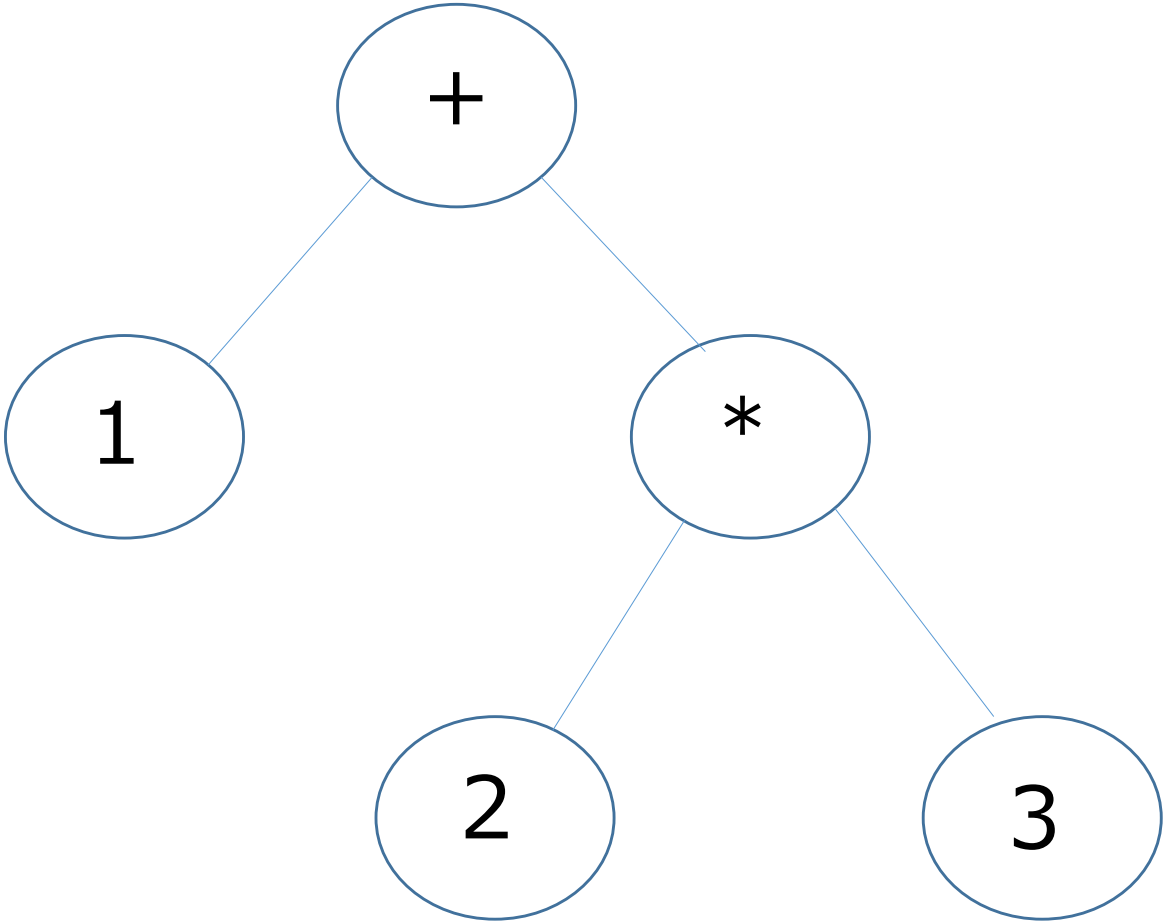
D, E, B, F, G, C, A
の順に処理を行う

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

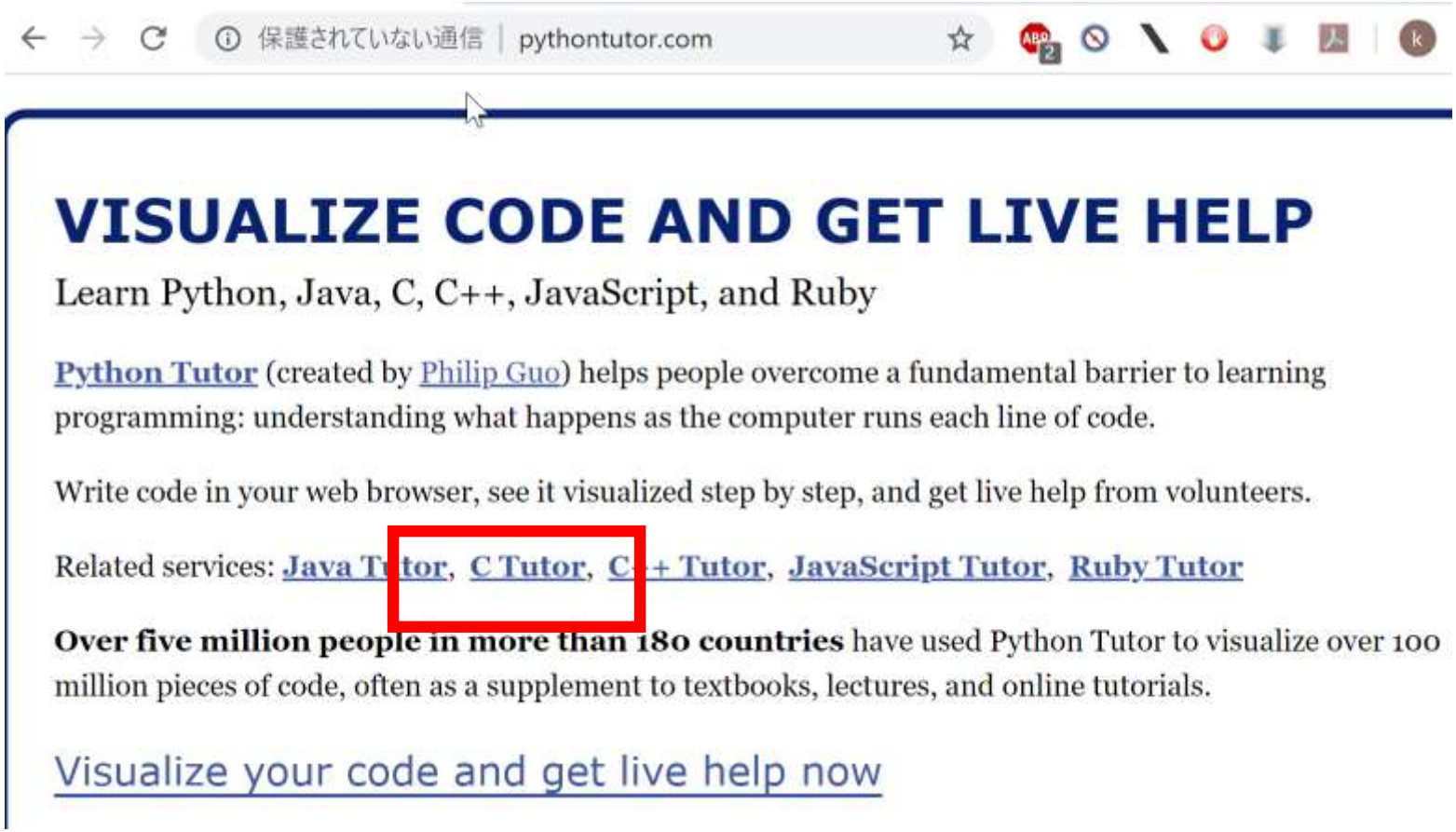
→ うまく動かないときは **Google Chrome** を試してください

※ 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



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

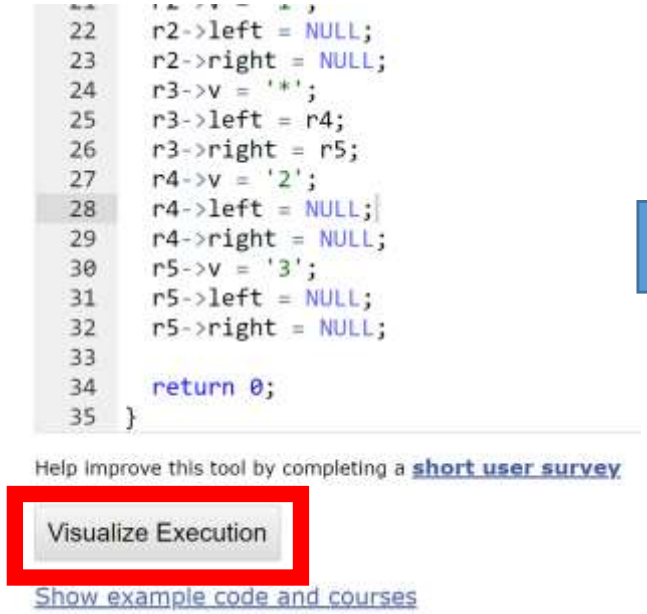
```
18    r1->v = '+';
19    r1->left = r2;
20    r1->right = r3;
21    r2->v = '1';
22    r2->left = NULL;
23    r2->right = NULL;
24    r3->v = '*';
25    r3->left = r4;
26    r3->right = r5;
27    r4->v = '2';
28    r4->left = NULL;
29    r4->right = NULL;
30    r5->v = '3';
31    r5->left = NULL;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

② 「**Visualize Execution**」をクリック.

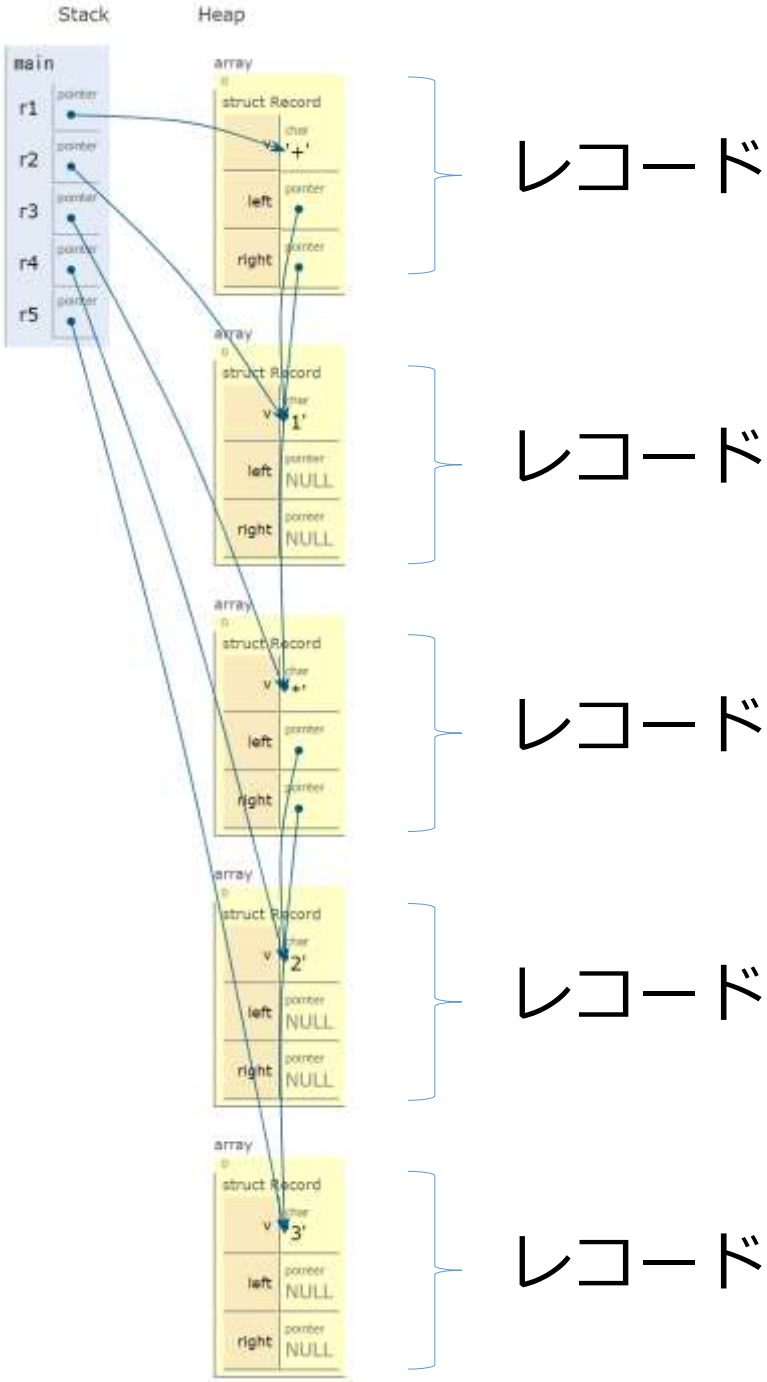
「**Last**」をクリック.

結果を確認する.

「**Edit this code**」をクリックして戻る



実行結果


Stack
Heap
main
r1
r2
r3
r4
r5
array
struct Record
v
'+'
left
right
'1'
NULL
'2'
'3'
レコード
レコード
レコード
レコード
レコード

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 }
```

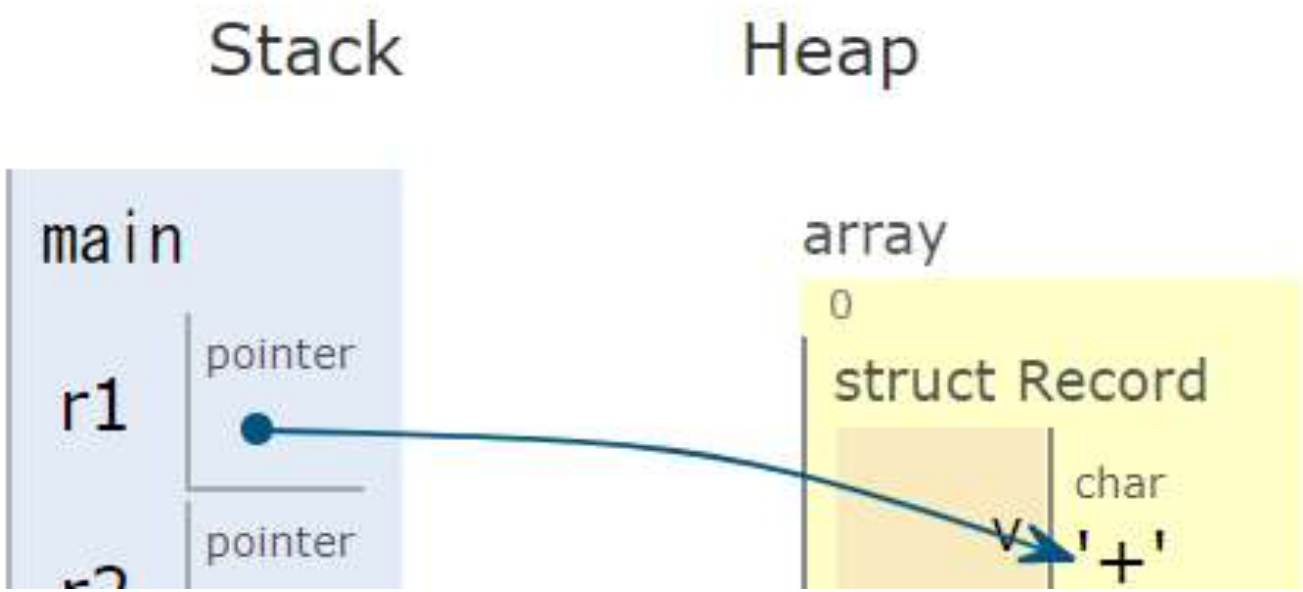
次ページに続く

```c
int main() {
  struct Record *r1, *r2, *r3, *r4, *r5;
  r1 = (struct Record*)malloc(sizeof(struct Record));
  r2 = (struct Record*)malloc(sizeof(struct Record));
  r3 = (struct Record*)malloc(sizeof(struct Record));
  r4 = (struct Record*)malloc(sizeof(struct Record));
  r5 = (struct Record*)malloc(sizeof(struct Record));

  r1->v = '+';
  r1->left = r2;
  r1->right = r3;
  r2->v = '1';
  r2->left = NULL;
  r2->right = NULL;
  r3->v = '*';
  r3->left = r4;
  r3->right = r5;
  r4->v = '2';
  r4->left = NULL;
  r4->right = NULL;
  r5->v = '3';
  r5->left = NULL;
  r5->right = NULL;

  preorder(r1);
  return 0;
}
```

実行し，結果を確認しなさい

Print output (drag lower right corner to resize)

```
+
1
*
2
3
```



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 }
```

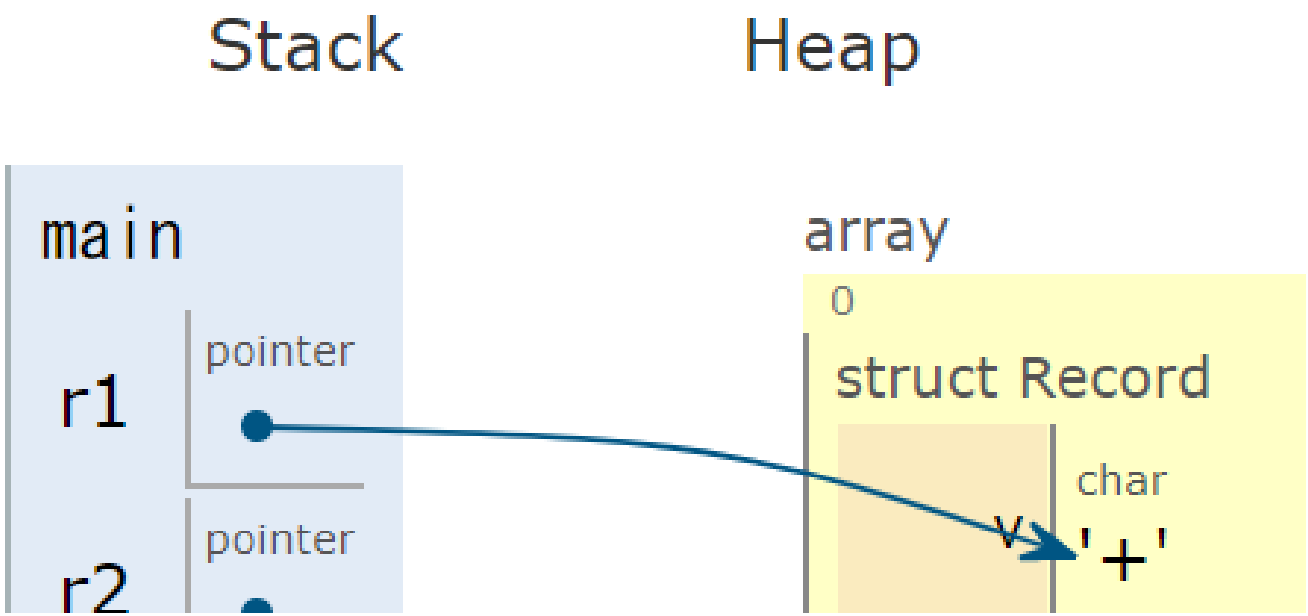
次ページに続く

```c
int main() {
  struct Record *r1, *r2, *r3, *r4, *r5;
  r1 = (struct Record*)malloc(sizeof(struct Record));
  r2 = (struct Record*)malloc(sizeof(struct Record));
  r3 = (struct Record*)malloc(sizeof(struct Record));
  r4 = (struct Record*)malloc(sizeof(struct Record));
  r5 = (struct Record*)malloc(sizeof(struct Record));

  r1->v = '+';
  r1->left = r2;
  r1->right = r3;
  r2->v = '1';
  r2->left = NULL;
  r2->right = NULL;
  r3->v = '*';
  r3->left = r4;
  r3->right = r5;
  r4->v = '2';
  r4->left = NULL;
  r4->right = NULL;
  r5->v = '3';
  r5->left = NULL;
  r5->right = NULL;

  inorder(r1);
  return 0;
}
```

実行し，結果を確認しなさい

Print output (drag lower right corner to resize)

```
1
+
2
*
3
```



後行順 (post-order)

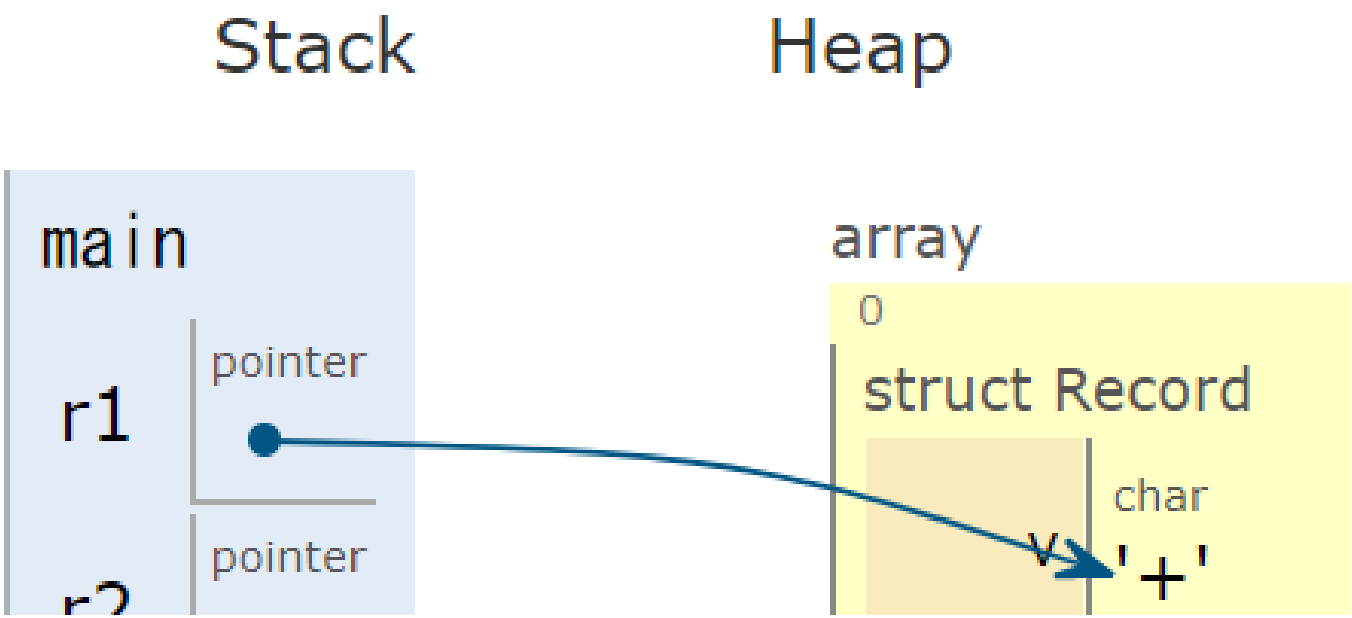
- プログラムを次のように書き換えなさい

```
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

```c
int main() {
  struct Record *r1, *r2, *r3, *r4, *r5;
  r1 = (struct Record*)malloc(sizeof(struct Record));
  r2 = (struct Record*)malloc(sizeof(struct Record));
  r3 = (struct Record*)malloc(sizeof(struct Record));
  r4 = (struct Record*)malloc(sizeof(struct Record));
  r5 = (struct Record*)malloc(sizeof(struct Record));

  r1->v = '+';
  r1->left = r2;
  r1->right = r3;
  r2->v = '1';
  r2->left = NULL;
  r2->right = NULL;
  r3->v = '*';
  r3->left = r4;
  r3->right = r5;
  r4->v = '2';
  r4->left = NULL;
  r4->right = NULL;
  r5->v = '3';
  r5->left = NULL;
  r5->right = NULL;

  postorder(r1);
  return 0;
}
```

実行し，結果を確認しなさい

Print output (drag lower right corner to resize)

```
1
2
3
*
+
```