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

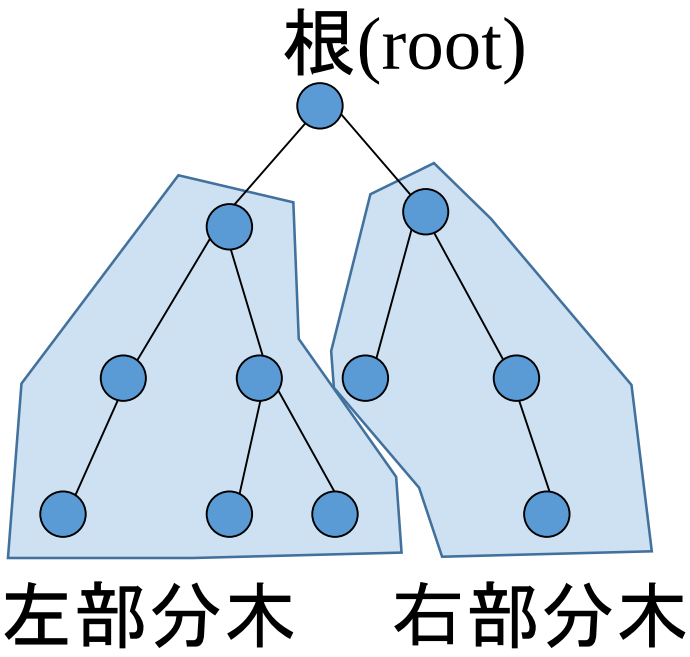
金子邦彦

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**



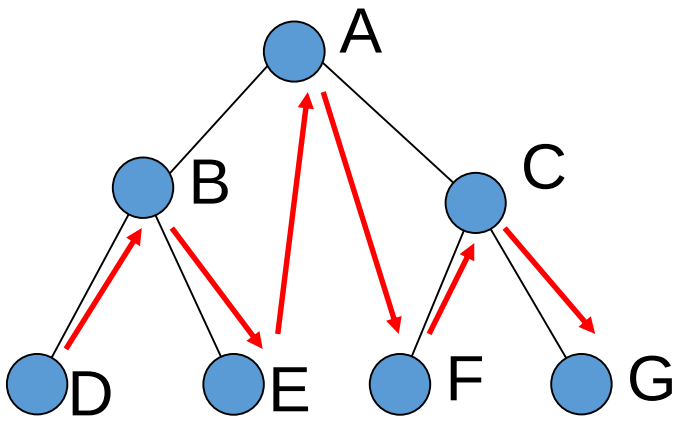
二分木の走査法

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)

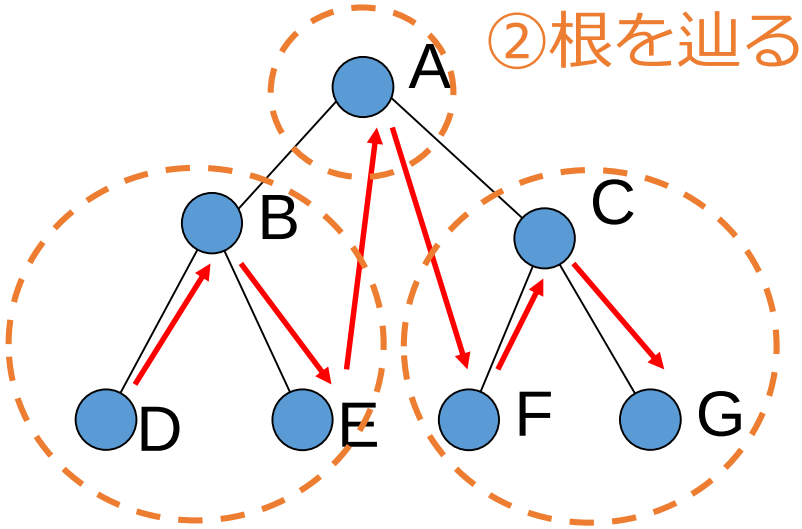
- 根ノード，左部分木，右部分木の順



D, B, E, A, F, C, G
の順に処理を行う

中間順 (in-order)

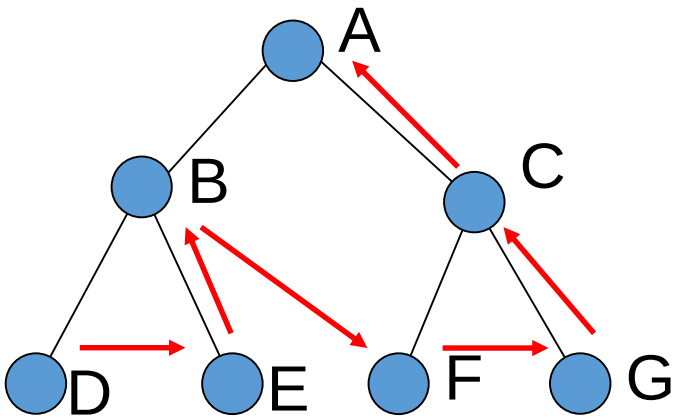
- 左部分木，根ノード，右部分木の順



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



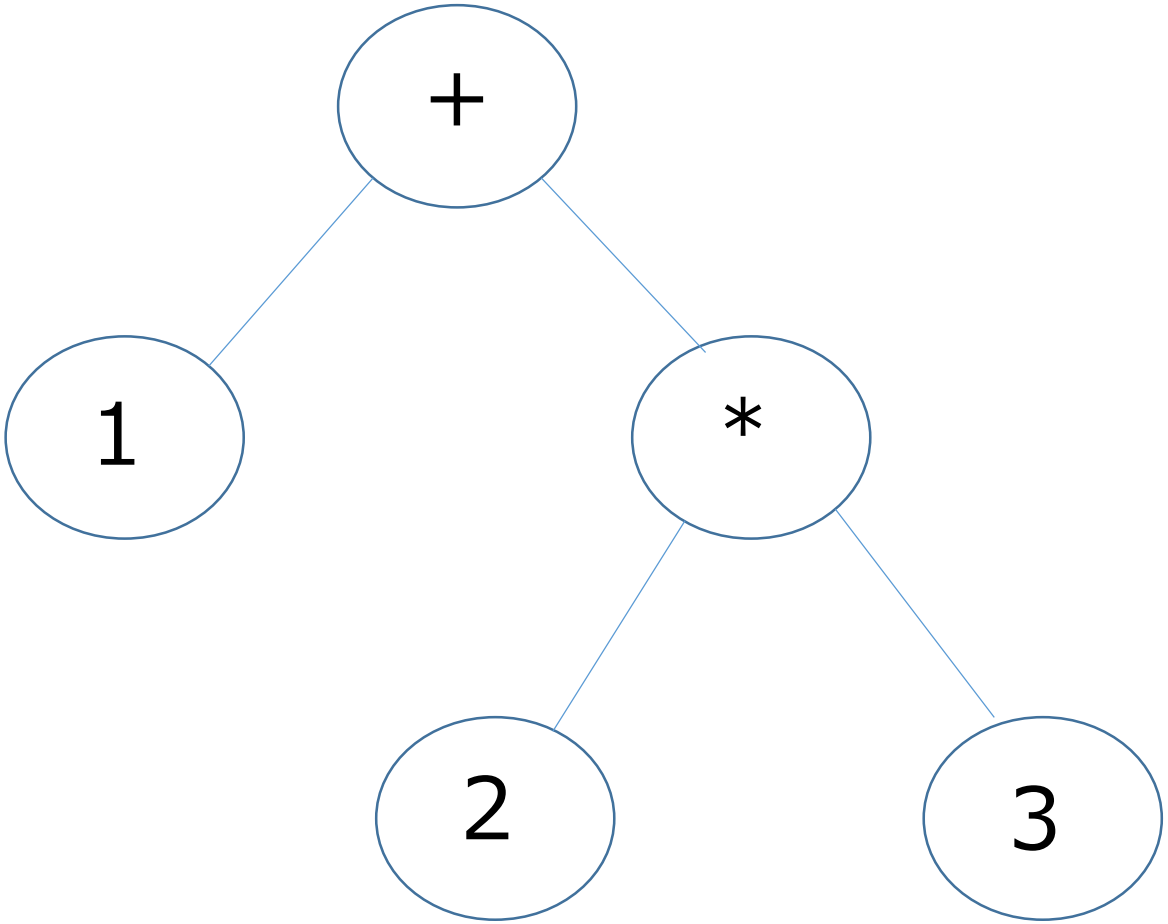
D, E, B, F, G, C, A
の順に処理を行う

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

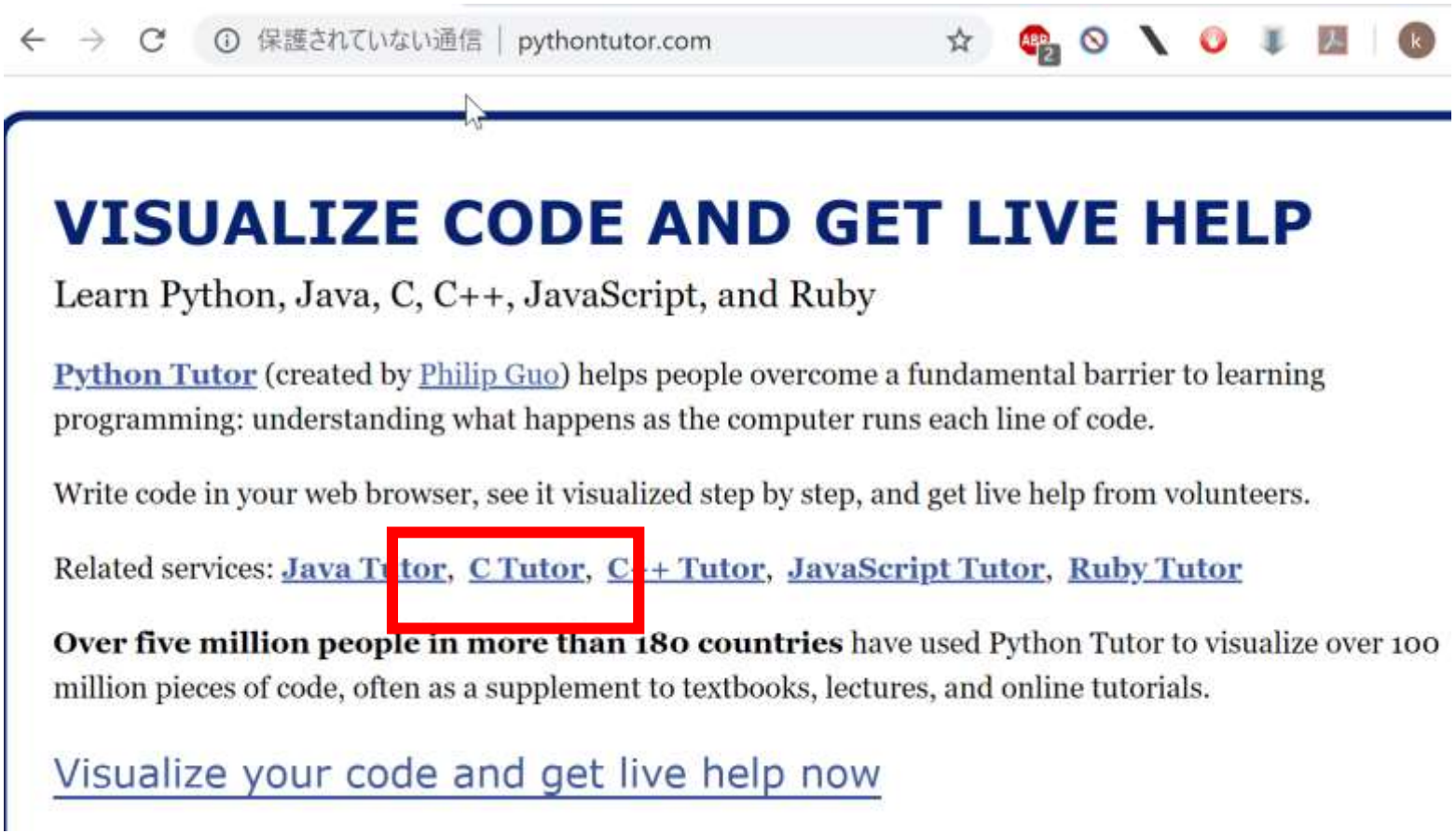
→ うまく動かないときは **Google Chrome** を試してください

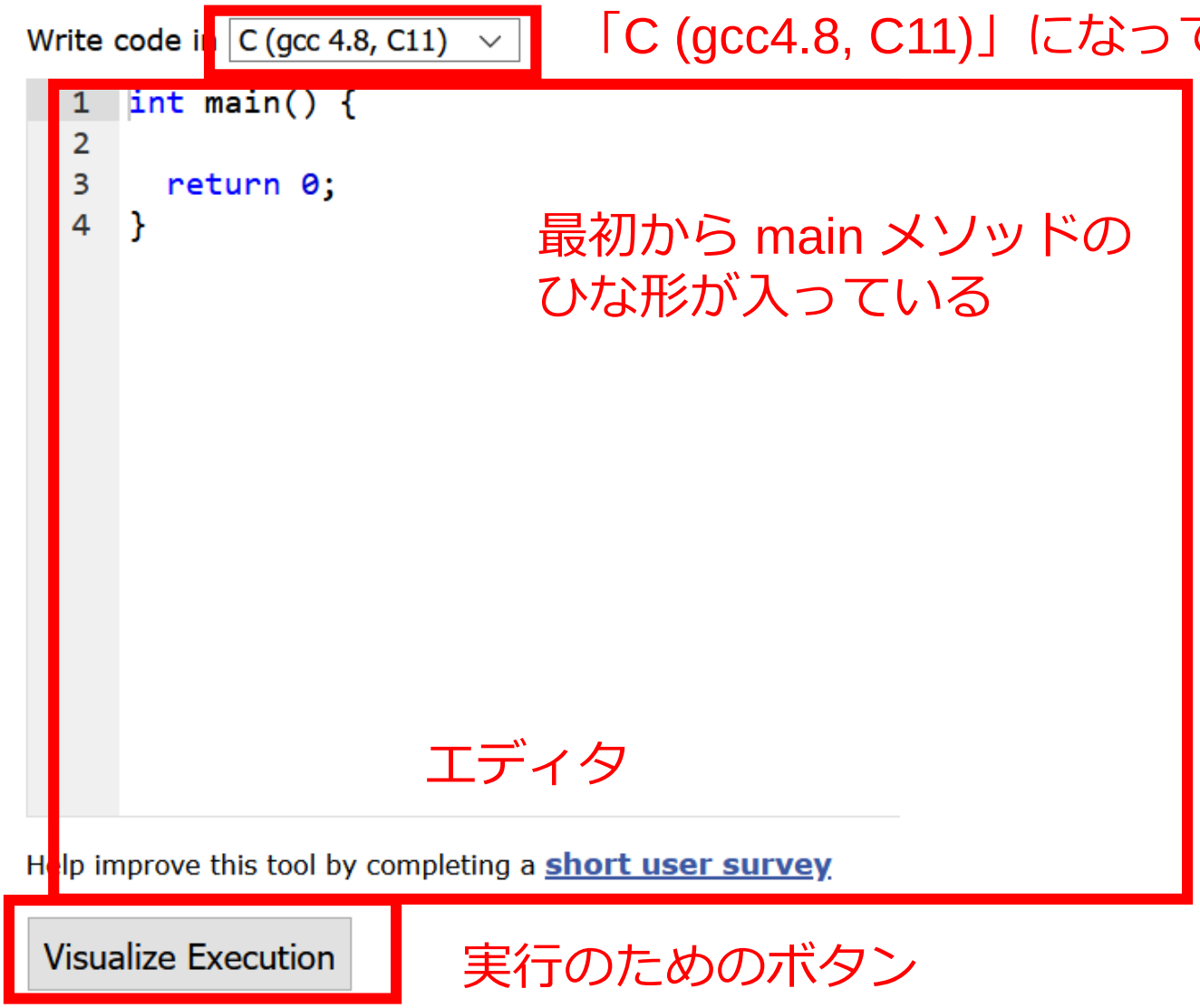
※ 途中で**「Server Busy・・・」というメッセージが出る**ことがある.

→ 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




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  }
```

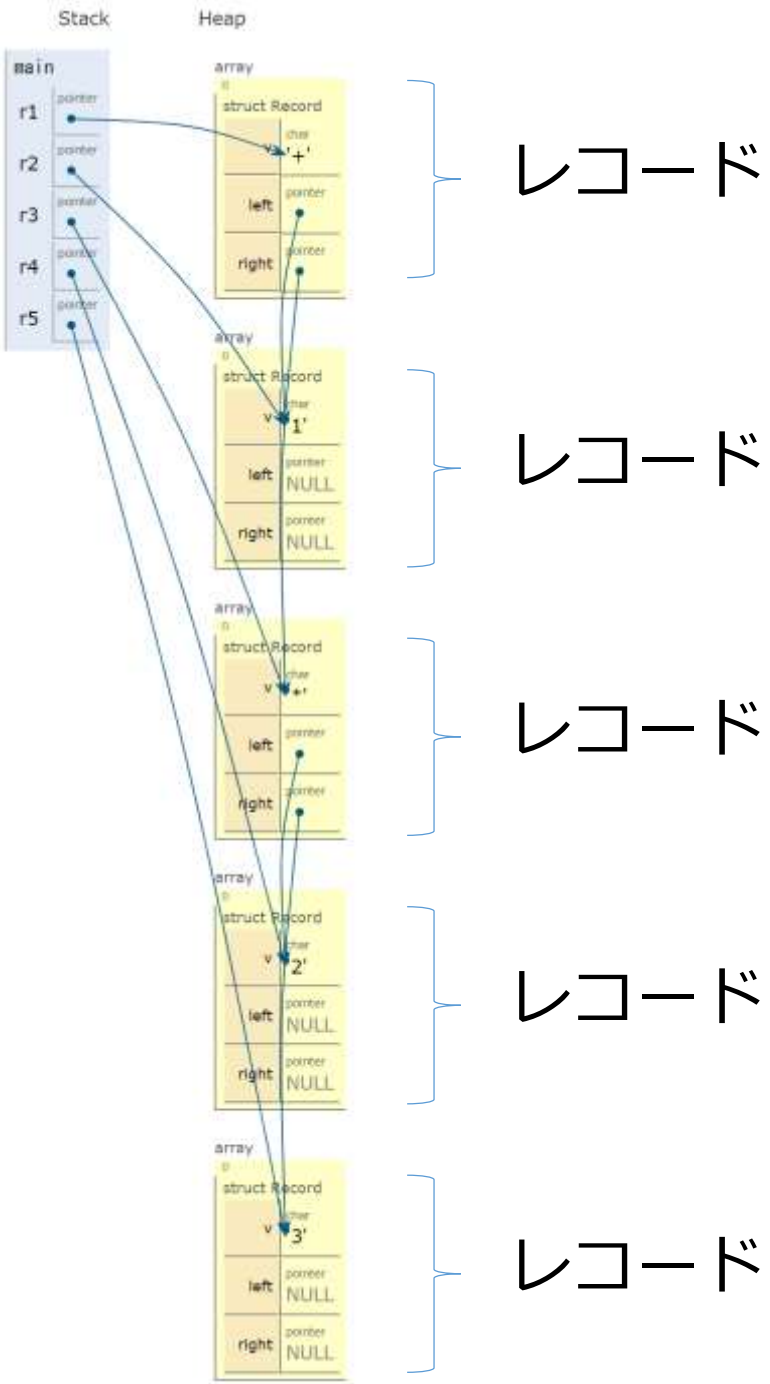
②「**Visualize Execution**」をクリック.

「**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
pointer
array
struct Record
v
'+'
left
right
NULL
'1'
'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 }
```

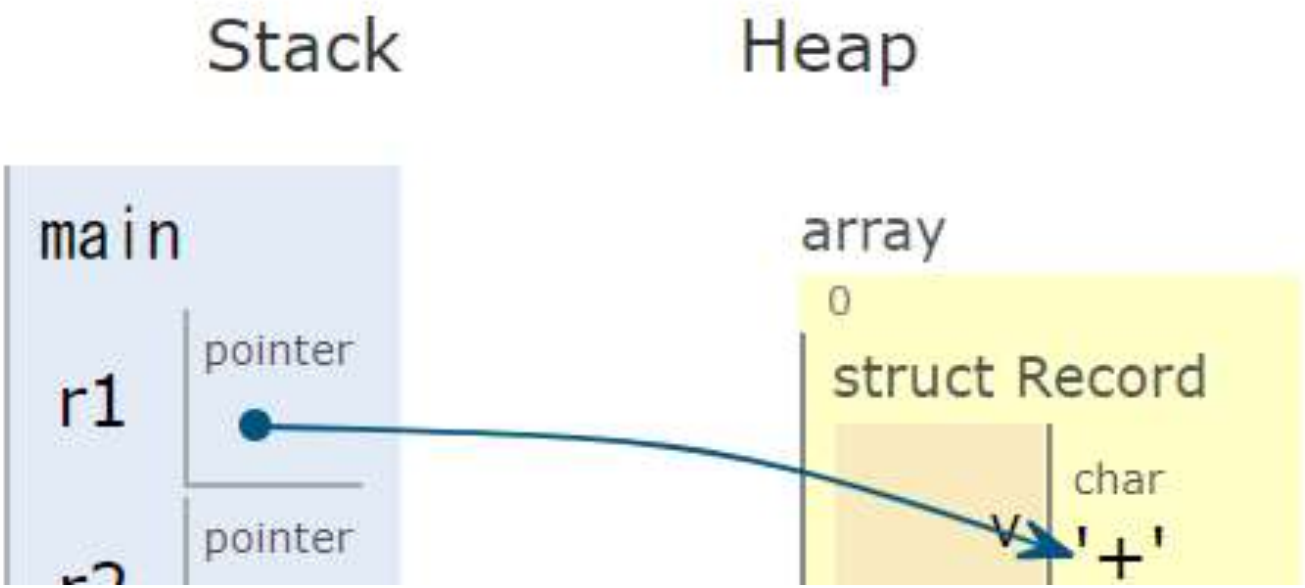
次ページに続く

```
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
```



中間順 (in-order)

- プログラムを次のように書き換えなさい

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

struct Record {
  char v;
  struct Record *left;
  struct Record *right;
};

void inorder(struct Record *r) {
  if (r->left != NULL) {
    inorder(r->left);
  }
  printf("%c\n", r->v);
  if (r->right != NULL) {
    inorder(r->right);
  }
  return;
}
```

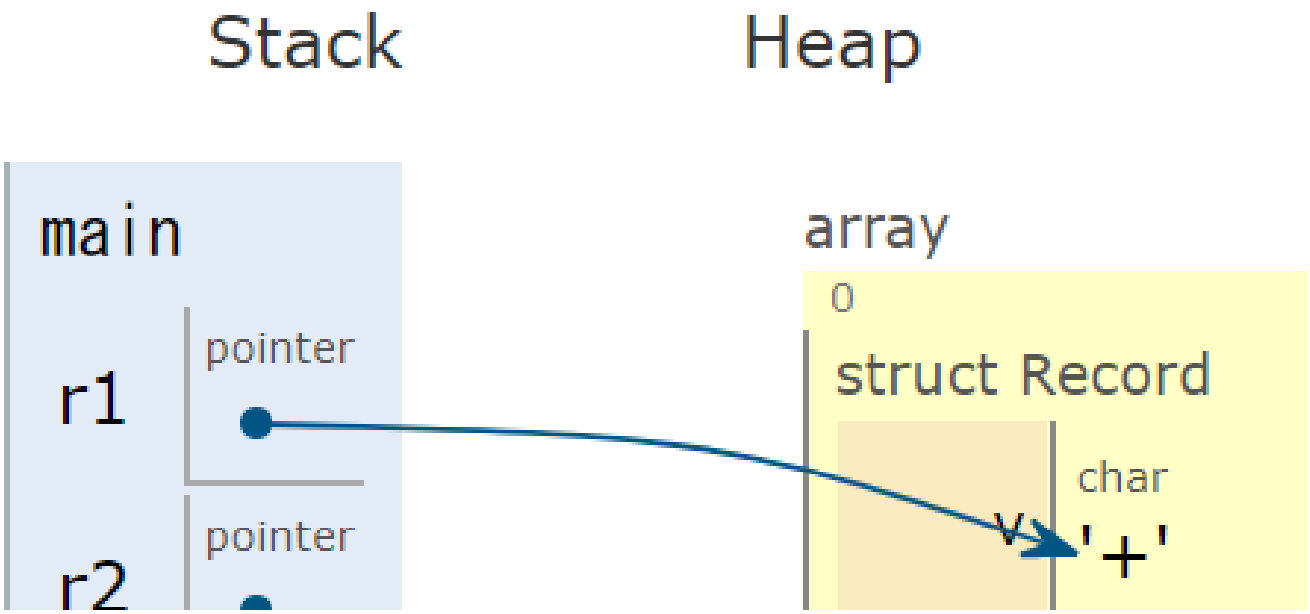
次ページに続く

```
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
```



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
*
+
```