



ad-2. 双方向リスト

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





双方向リストとは

レコードを次の2つで構成

- 要素を格納する**セル**
- リスト中の次の**レコード**を指す**ポインタ**を格納する**セル**
- リスト中の前の**レコード**を指す**ポインタ**を格納する**セル**

双方向リストと連結リストの違い



リスト内のある**レコード**のアドレス A が分かっていると
とする。

• 連結リスト

A の**次**に要素を挿入することは簡単。

A の**前**に要素を挿入することはできない。

• 双方向リスト

A の**次**に要素を挿入することは簡単。

A の**前**に要素を挿入することも簡単。



1-2. 実習



実習の指示

- 資料： **6** ~ **12**
- 次のことを理解しマスターする
 - 双方向リスト

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<stdlib.h>
2
3  struct Record {
4      int v;
5      struct Record *next;
6      struct Record *prev;
7  };
8
9  int main() {
10     struct Record *r1, *r2, *r3;
11     r1 = (struct Record *)malloc(sizeof(struct Record));
12     r2 = (struct Record *)malloc(sizeof(struct Record));
13     r3 = (struct Record *)malloc(sizeof(struct Record));
14     r1->v = 8;
15     r1->prev = NULL;
16     r1->next = r2;
17     r2->v = 5;
18     r2->prev = r1;
19     r2->next = r3;
20     r3->v = 16;
21     r3->prev = r2;
22     r3->next = NULL;
23     return 0;
24 }
```



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

```
Write code in C (gcc 4.8, Cl1)
1 #include<stdlib.h>
2
3 struct Record {
4     int v;
5     struct Record *next;
6     struct Record *prev;
7 };
8
9 int main() {
10     struct Record *r1, *r2, *r3;
11     r1 = (struct Record *)malloc(sizeof(struct Record));
12     r2 = (struct Record *)malloc(sizeof(struct Record));
13     r3 = (struct Record *)malloc(sizeof(struct Record));
14     r1->v = 8;
15     r1->prev = NULL;
16     r1->next = r2;
17     r2->v = 5;
18     r2->prev = r1;
19     r2->next = r3;
20     r3->v = 16;
21     r3->prev = r2;
22     r3->next = NULL;
23     return 0;
24 }
25
```



```
1 int v;
2 struct Record *next;
3 struct Record *prev;
4 };
5
6 int main() {
7     struct Record *r1, *r2, *r3;
8     r1 = (struct Record *)malloc(sizeof(struct Record));
9     r2 = (struct Record *)malloc(sizeof(struct Record));
10    r3 = (struct Record *)malloc(sizeof(struct Record));
11    r1->v = 8;
12    r1->prev = NULL;
13    r1->next = r2;
14    r2->v = 5;
15    r2->prev = r1;
16    r2->next = r3;
17    r3->v = 16;
18    r3->prev = r2;
19    r3->next = NULL;
20    return 0;
21 }

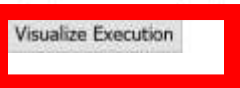
```



```
1 r3->next = NULL;
2 return 0;
3 }

```

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



[Edit this code](#)

that just executed
line to execute

<< First < Prev Next >> Last >>

Step 1 of 14

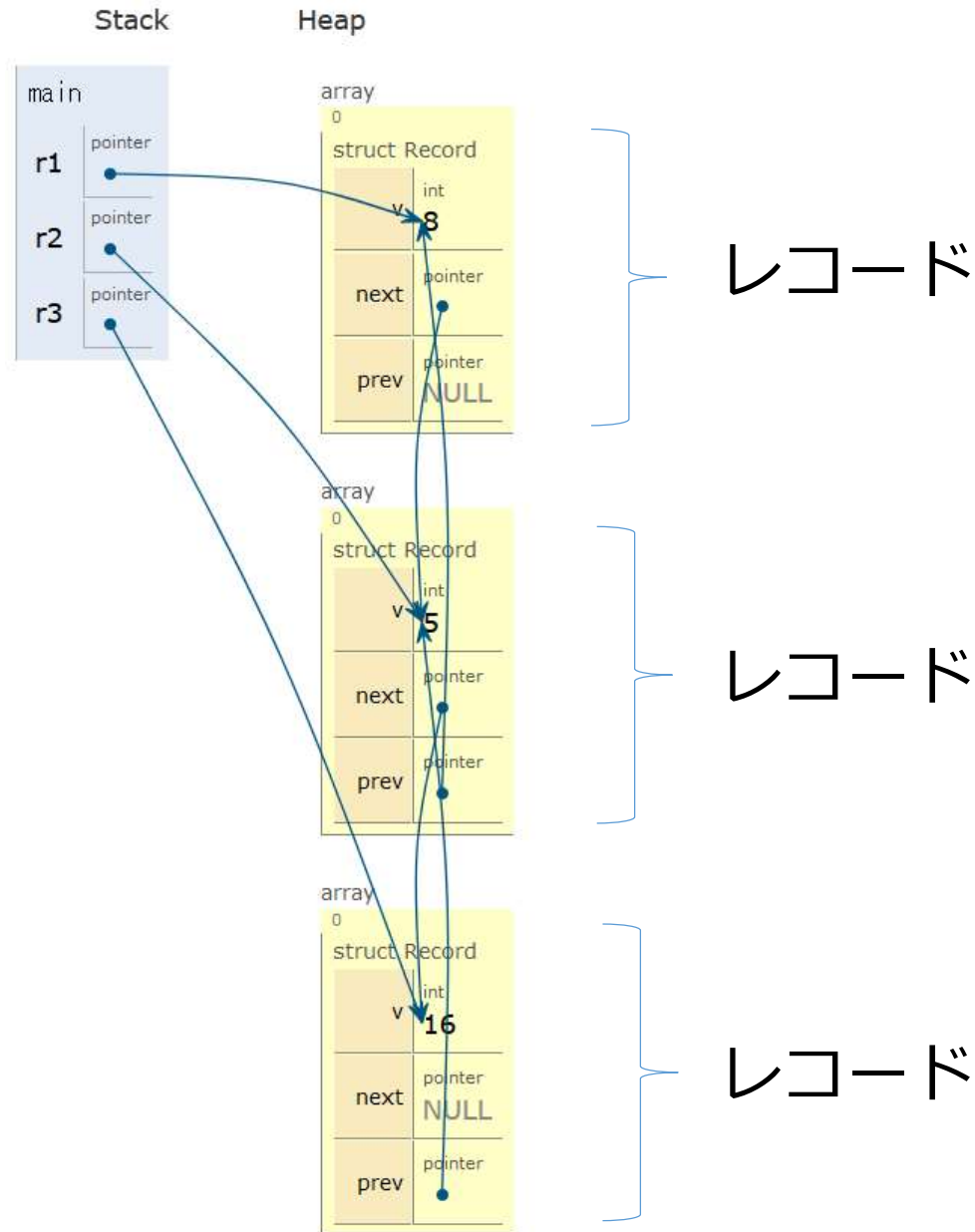
[Edit this code](#)

it just executed
ie to execute

<< First < Prev Next >> Last >>

Done running (14 steps)

実行結果



- いまのプログラムで
末尾の要素（値は16）を削除するプログラム
を書き加えなさい

- いまのプログラムで,
末尾に新しい要素（値は 24）を挿入する
プログラムを書き加えなさい