コンピュータとプログラミング C言語編

7. データ構造

阿部洋丈 / ABE Hirotake
habe@cs.tsukuba.ac.jp

期末試験について

- 詳しくは manaba のコースコンテンツを参照
- 日時： 2025年7月25日 3限
  （4限は授業なし。計算機室は使用可。）
- 場所： 3A204
- 実施方法： 筆記試験（60分間）
- この授業の配布資料を印刷したもの（書き込み可）、および、自筆のノートの持ち込みおよび閲覧は認める

データ構造とは

- データに対するアクセスの傾向は、データの内容や処理の内容によって様々
  - 読み込みが多い、既存項目への変更が多い、新規項目の追加が多い、それぞれが一定の割合で混ざっている、…
  - 時間的局所性、空間的局所性、意味的局所性、…
- データ構造は、それらの処理を効率的に行うために、データを組織化するための方式である。
  - より詳しくは、秋学期の「アルゴリズムとデータ構造」にて

代表的なデータ構造の種類

- 配列
- リスト（連結リスト、スタック、キュー、スキップリスト）
- ハッシュ表（開番地法、連結法）
- 木（二分木、B木、トライ木、その他にも多数）
- グラフ（無向、有向（単方向、双方向））
- その他多数

配列とリスト

- 配列は、計算機のメモリそのもの
  - 利点： インデックスが分かっている場合の処理が早い
    （ベースアドレス＋オフセットの計算一発でアクセス可能）
  - 欠点： 要素の増減、特に途中挿入や削除が不得意
    （多くの場合、配列の作り直しが必要）
- リストは、上記の欠点を補うために用いられる。
  ただし、配列のメリットは逆に損なわれる。

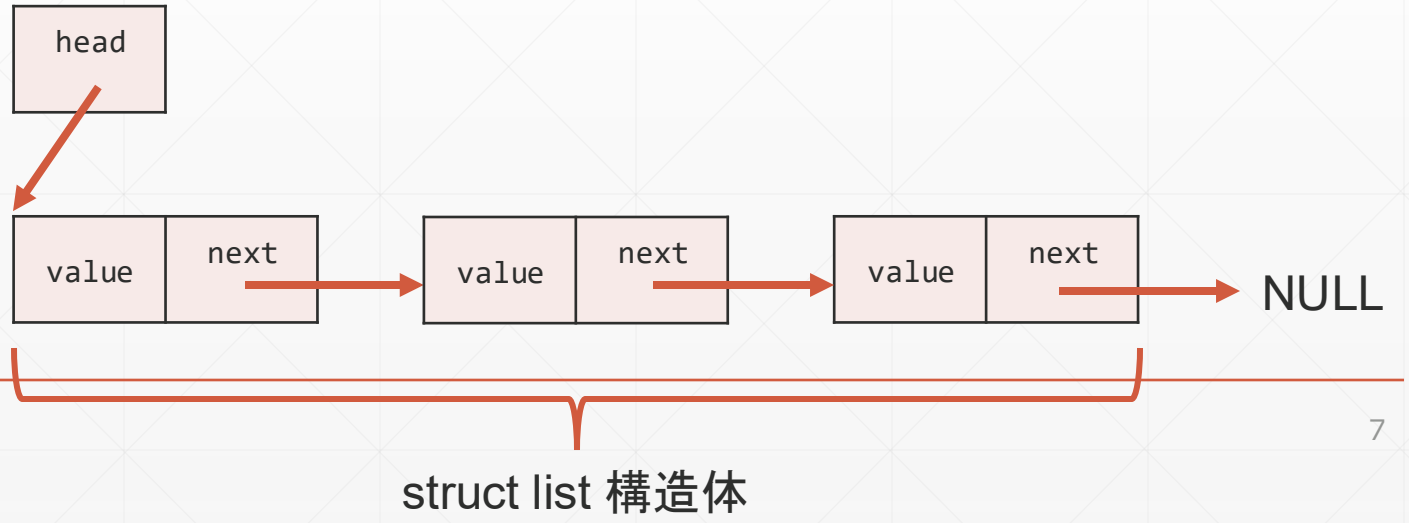
連結リスト (Linked List)

- 各要素をポインタで連結することで、末尾追加・途中挿入・削除を簡単に実現可能にしている
  - 単方向と双方向がある。
- その代わり、先頭以外の要素に辿り着くためには、他の要素を経由する必要がある
  - 末尾にアクセスするためには、他のすべての要素を経由しなければならない
    - 数十程度なら良いが、数千万や数億あったら....？

単方向連結リストの実現方法

- 自己参照構造体を使う（第4週で既出）
  - リストの先頭を指すためのポインタ（下の例では head）を確保し、そこが常にリストを指しているようにする。
  - 各構造体は、自分の次の要素を表す構造体へのアドレスをポインタに格納する。リストの末尾の場合は NULL を格納する。

```
struct list {
  int value;   // 扱うデータの型を指定
  struct list *next;
};
```



連結リストに対する操作

- ループでも再帰でも複雑さはそれほど変わらないが、この授業では基本的に再帰を使うことを推奨
  - 木やグラフでは再帰を多用するので、慣れておくと良い
- リストに対する再帰処理は以下のように考える：
  - 現在注目している要素が操作の対象かどうかを検査
  - 対象であった場合は、処理を行い、適切な値を return する
  - 対象でなかった場合は、 next 以降の「部分リスト」に対して再帰呼び出しを行う

リスト内の要素の探索

- 前のページの説明そのまま。

```
list_t *search(int value, list_t *list) {
  if (list == NULL) { return NULL; }
  if (value == list->value) {
    return list;
  } else {
    return search(value, list->next);
  }
}
```

停止条件

対象かどうかを検査

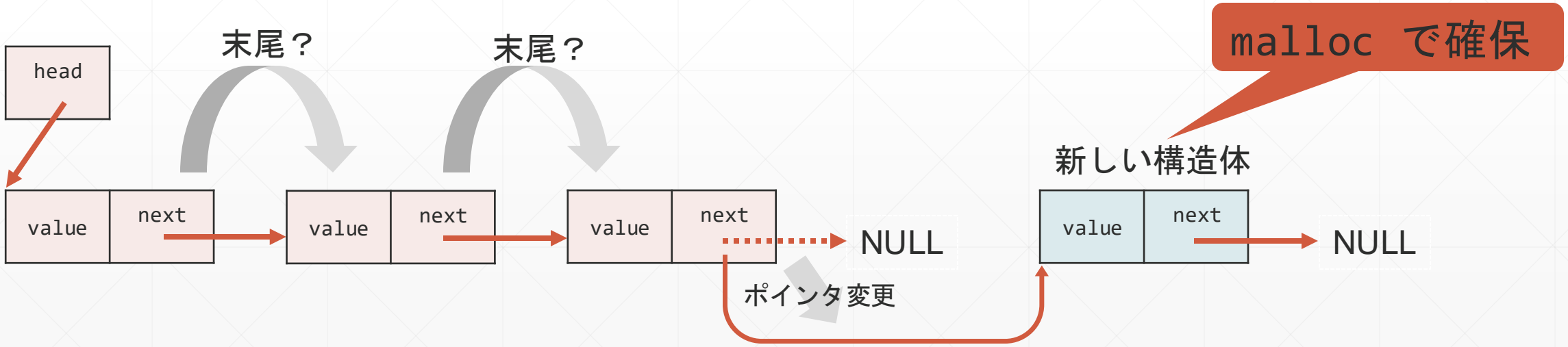
何もせずreturn

残りのリストに対して再帰呼び出し

注: typedef struct list list_t; だと思って読んでください

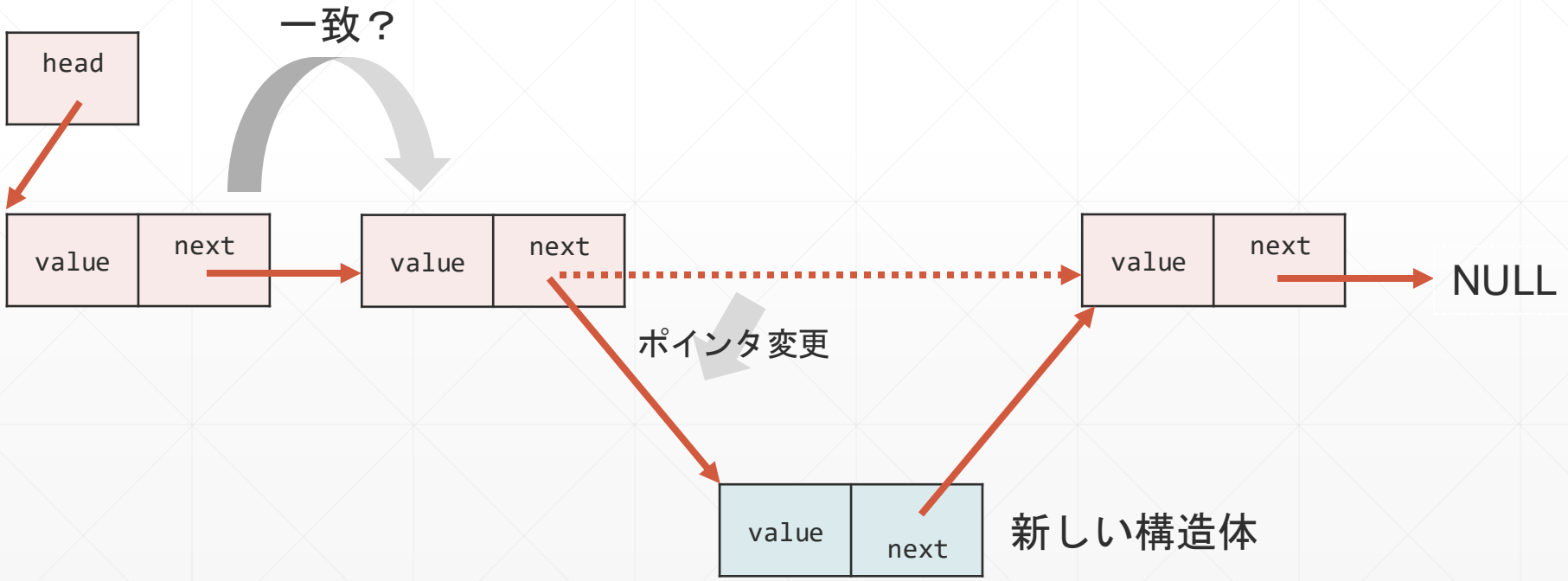
リストへの末尾追加

- 操作の対象は、末尾の要素、つまり next==NULL の要素
- 新しい構造体を確保し、そのアドレスを next に代入する。新しい構造体の next は NULL にしておく



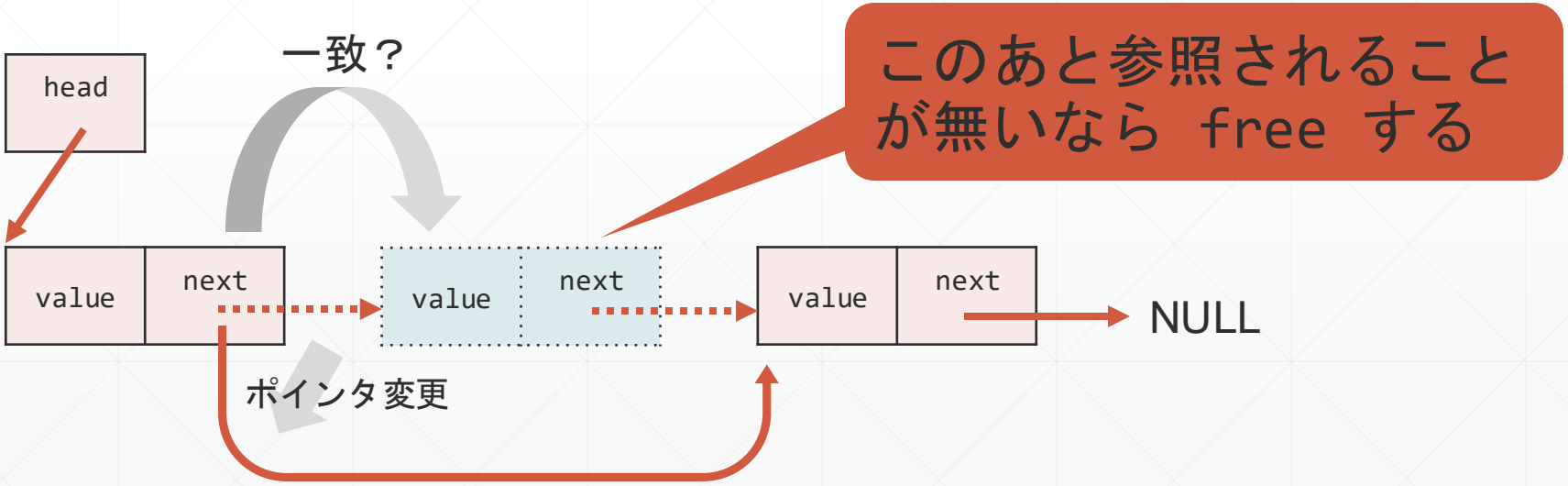
リストへの途中挿入

- 探索で追加すべき箇所を見つけ、ポインタの張り替えを行う



リストからの削除

- 途中挿入の逆をやれば良い
  - ただし、変更が必要なのは一つ前の構造体であることに注意
  - 必要に応じ、不要になった構造体のfreeを忘れずに



リストを使った応用例： ワードカウント

- 複数の単語からなる自然言語の文章を読み込み、その中に出現する各単語の出現回数を数える
- 右図のような構造体を使ってリストを構成
  - key は単語を表す文字列を指す
  - value には、keyが指す単語の出現回数が入る
- readline で行を読み込み、tokenize で単語を切り出す
- 単語を1個読み込む度にリスト内を探索する。もし見つかったらvalueを1増やす。見つからなかったら末尾にその単語を追加する
- 最後に結果の一覧を表示する
  - 場合によっては、a や the など、頻出すぎてあまり意味の無い単語を取り除く

```
typedef struct list {
  char *key;
  int value;
  struct list *next;
} list_t;
```

今日の演習

- 以下の課題のいずれか一方を選択して行うこと
  - 課題A: 講義の内容に沿って、連結リストを使ったワードカウントプログラムを完成させよ (appendとdeleteを実現)
  - 課題B（課題Aを包含）: 通常の連結リストに加え、スキップリストを用いたワードカウントを作成し、連結リスト版と実行時間を比較せよ（単語数を変化させた場合、単語の種類数を変化させた場合、etc.）
    - 参考： https://ja.wikipedia.org/wiki/スキップリスト