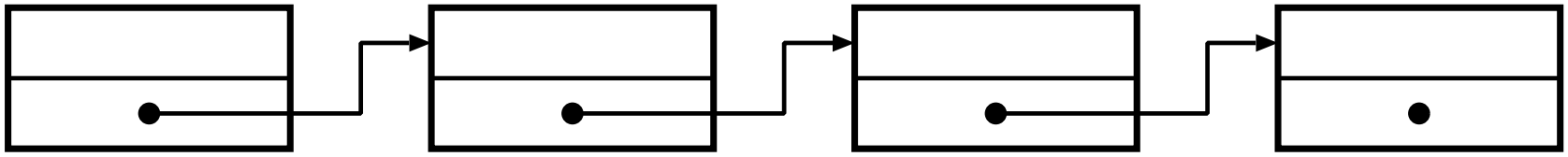
リスト構造 (linked list)

“「値」と「その次を指すもの」との組”
を基本単位としたデータ構造



```
typedef struct node {
    int val;
    struct node *next;
} Node;
```

リストの基本操作

- **新規作成 (末尾への追加)**
- **走査 (traverse)**
- **削除**
- **挿入**

リストの新規作成 (末尾への節点の追加)

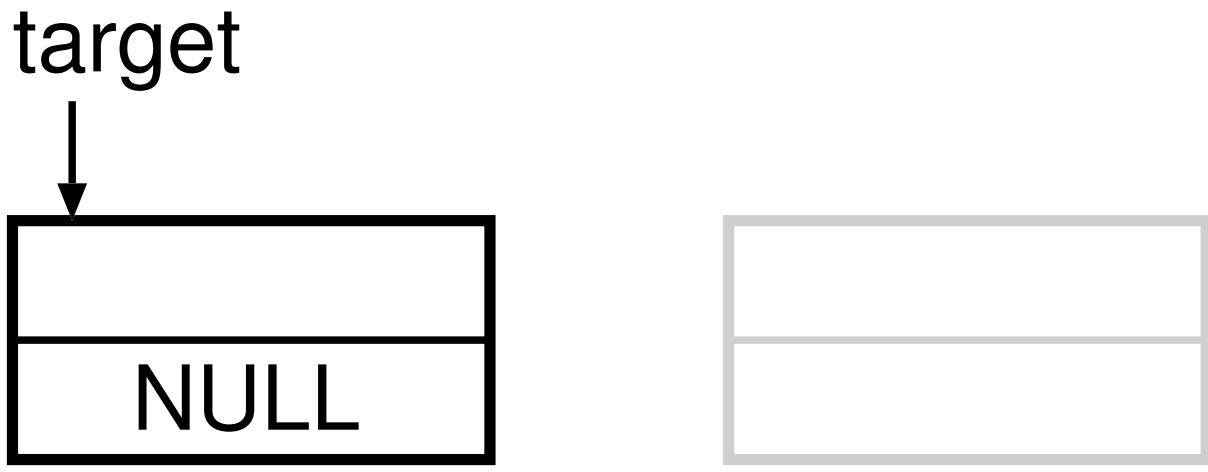
(1) 追跡用ポインタを追加元の節点に向ける

(2) リストに追加したい値がある間、
- それが入るべき新節点を動的に確保
- 追跡用ポインタが指す節点 (現在の末尾) の次へのリンクを、新節点に向ける
- 追跡用ポインタを新節点に進める
- 新節点に値を代入

を繰り返す。

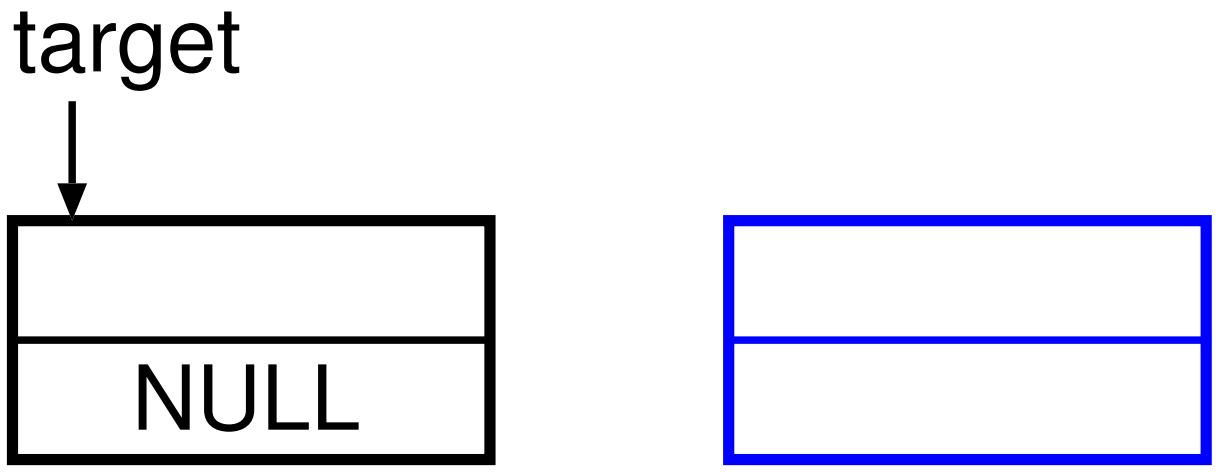
(3) 末尾の節点の次へのポインタの値を NULL に

リストの新規作成 (末尾への節点の追加)



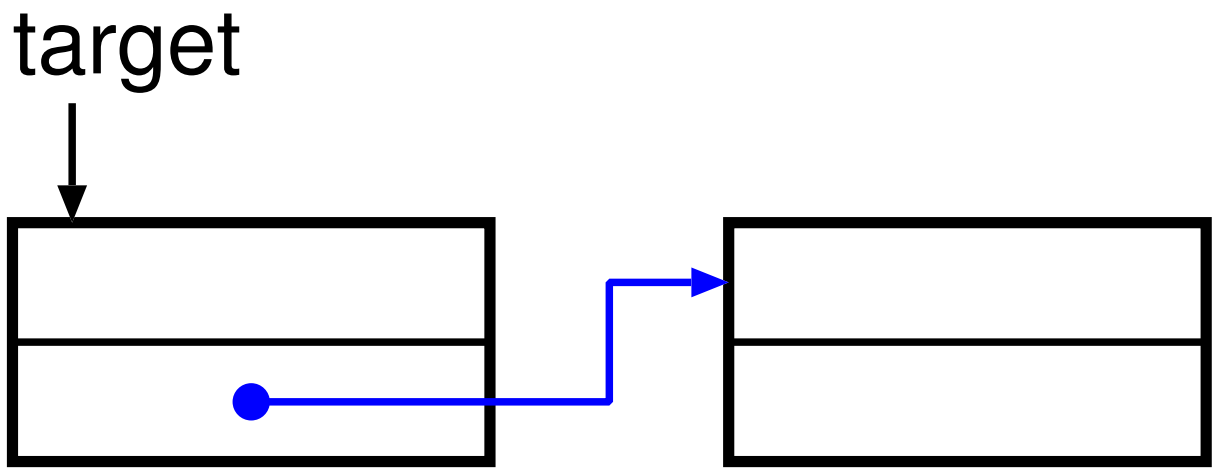
- それが入るべき新たな節点を動的に確保
- 追跡用ポインタが指す節点 (現在の末尾) の次へのリンクを、新節点に向ける
- 追跡用ポインタを新節点に進める
- 新節点に値を代入

リストの新規作成 (末尾への節点の追加)



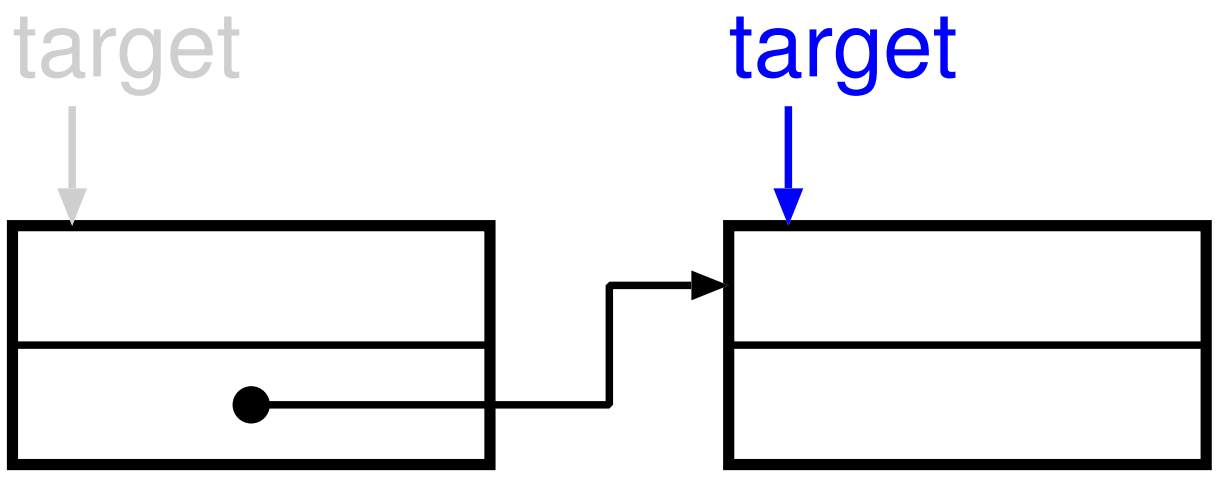
- それが入るべき新たな節点を動的に確保
- 追跡用ポインタが指す節点 (現在の末尾) の次へのリンクを、新節点に向ける
- 追跡用ポインタを新節点に進める
- 新節点に値を代入

リストの新規作成 (末尾への節点の追加)



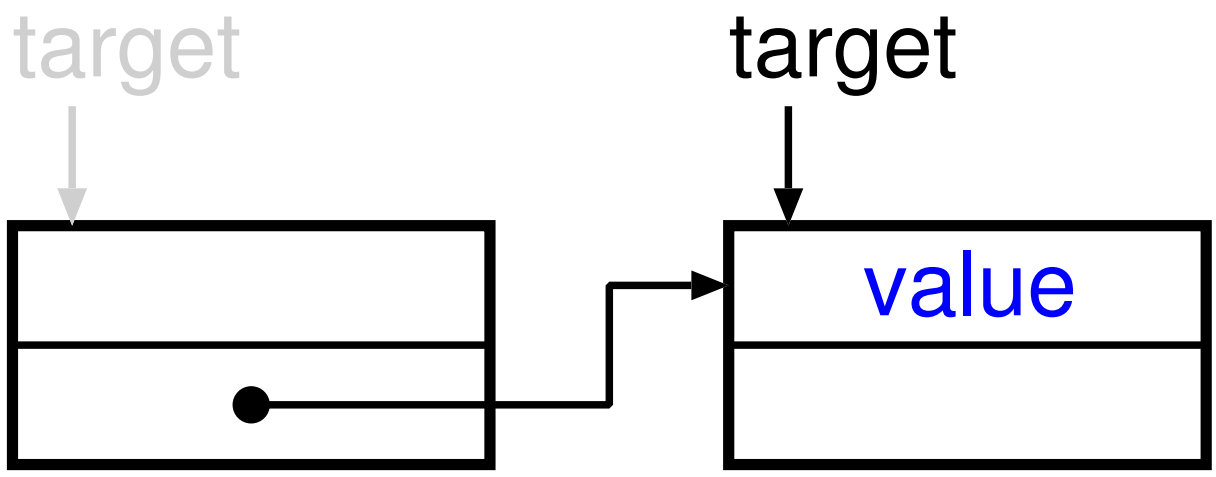
- それが入るべき新たな節点を動的に確保
- 追跡用ポインタが指す節点 (現在の末尾) の次へのリンクを、新節点に向ける
- 追跡用ポインタを新節点に進める
- 新節点に値を代入

リストの新規作成 (末尾への節点の追加)



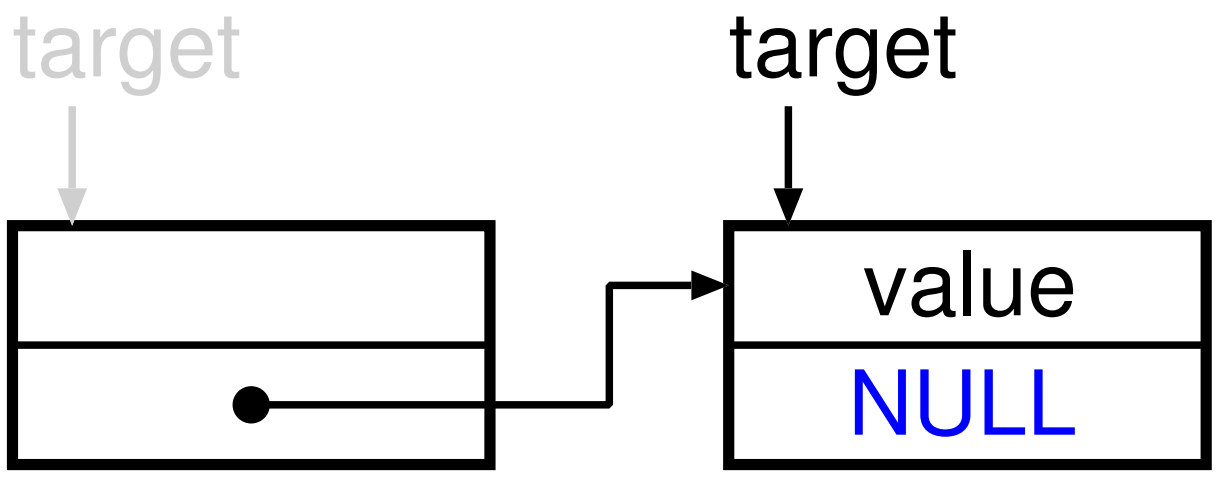
- それが入るべき新たな節点を動的に確保
- 追跡用ポインタが指す節点 (現在の末尾) の次へのリンクを、新節点に向ける
- **追跡用ポインタを新節点に進める**
- 新節点に値を代入

リストの新規作成 (末尾への節点の追加)



- それが入るべき新たな節点を動的に確保
- 追跡用ポインタが指す節点 (現在の末尾) の次へのリンクを、新節点に向ける
- 追跡用ポインタを新節点に進める
- **新節点に値を代入**

リストの新規作成 (末尾への節点の追加)



- それが入るべき新たな節点を動的に確保
- 追跡用ポインタが指す節点 (現在の末尾) の次へのリンクを、新節点に向ける
- 追跡用ポインタを新節点に進める
- 新節点に値を代入

リストの走査 (traverse)

(1) 追跡用ポインタを先頭の節点に向ける

(2) 追跡用ポインタの先に次の節点がある
(次が NULL でない) 間、

- 追跡用ポインタを次の節点に進める
- その節点の要素を処理 (表示など)

を繰り返す

課題 2:

実習 3.1 で、

リストの走査 (表示) の部分を

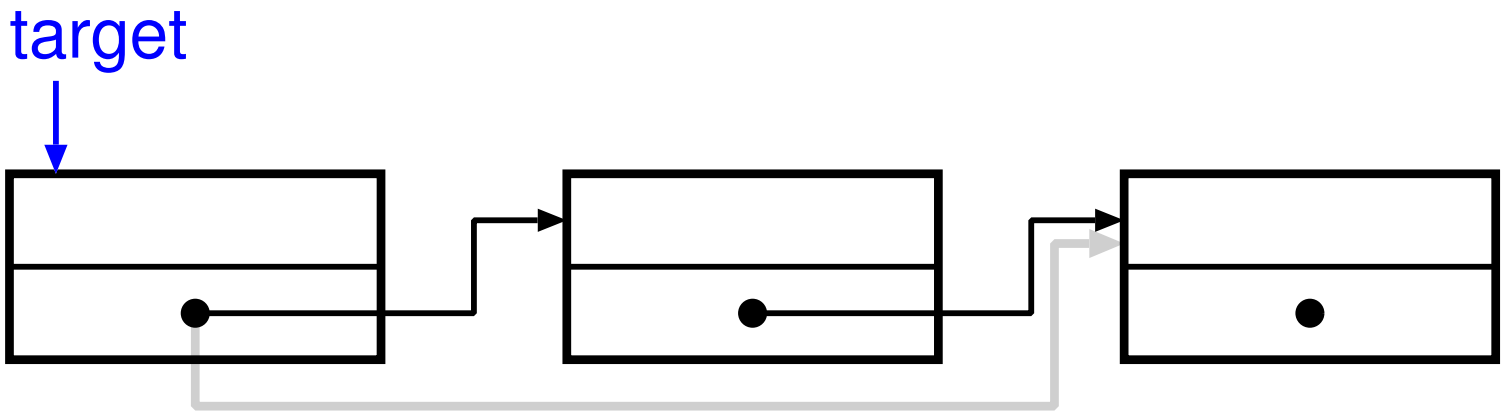
関数 `print_list()` として関数化せよ。

(ヒント: 関数プロトタイプは

```
void print_list(Node *);
```

が適当であろう。)

リストからの節点の削除



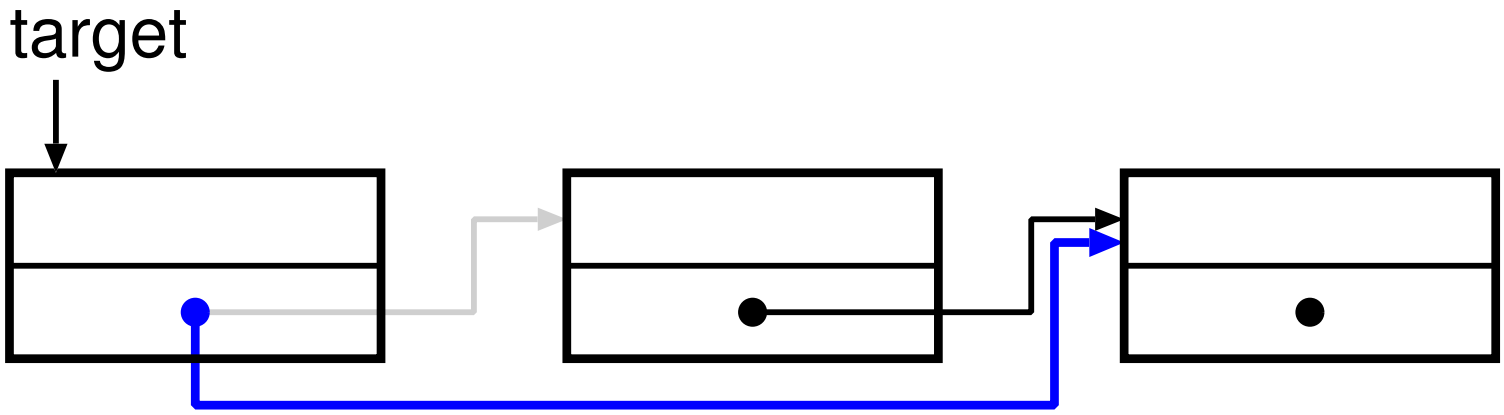
(1) 削除したい節点の手前の節点に、追跡用ポインタを向ける

(2) 削除したい節点を憶えておく

(3) 削除したい節点の次の節点に、追跡用ポインタの指す節点の次へのリンクを向ける

(4) 削除したい節点の領域解放

リストからの節点の削除



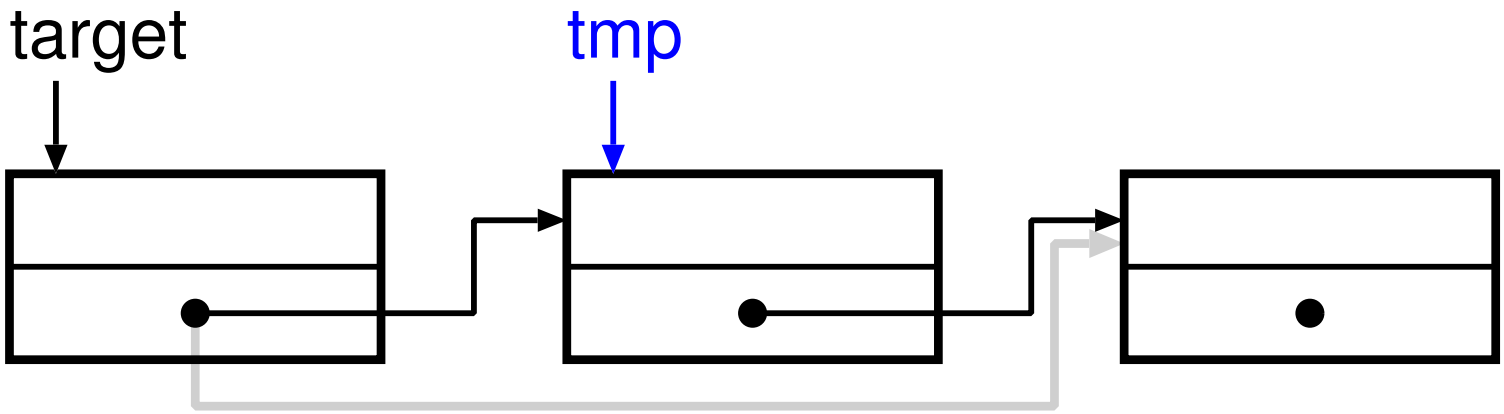
(1) 削除したい節点の手前の節点に、追跡用ポインタを向ける

(2) 削除したい節点を憶えておく

(3) 削除したい節点の次の節点に、追跡用ポインタの指す節点の次へのリンクを向ける

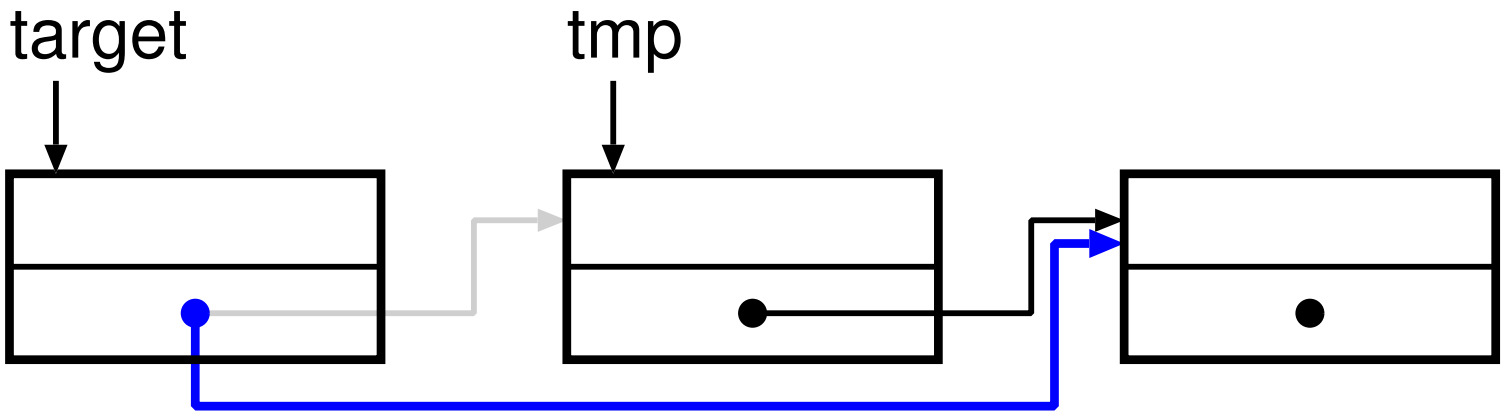
(4) 削除したい節点の領域解放

リストからの節点の削除



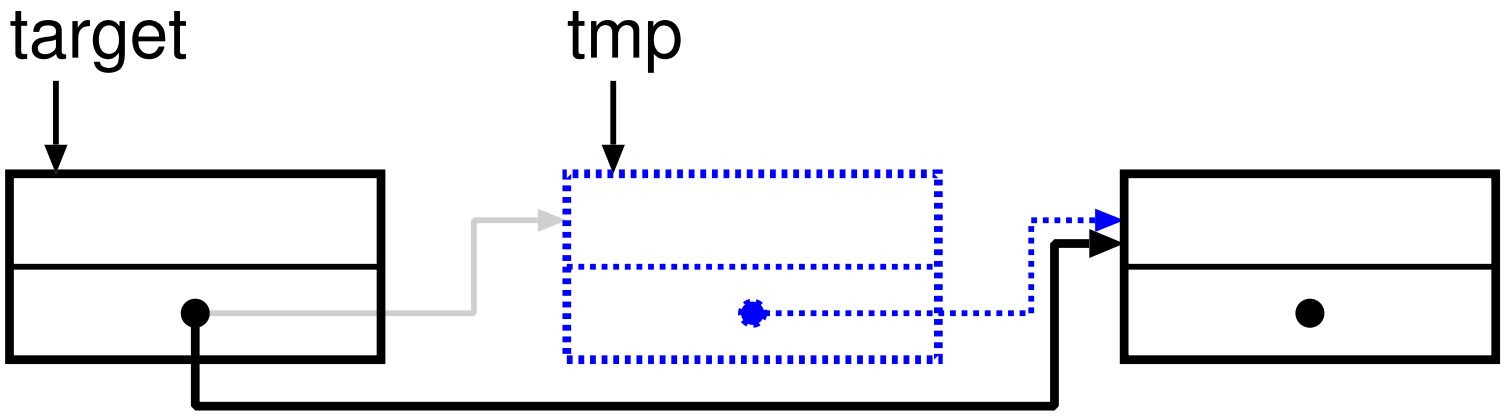
(1) 削除したい節点の手前の節点に、追跡用ポインタを向ける
(2) 削除したい節点を憶えておく
(3) 削除したい節点の次の節点に、追跡用ポインタの指す節点の次へのリンクを向ける
(4) 削除したい節点の領域解放

リストからの節点の削除



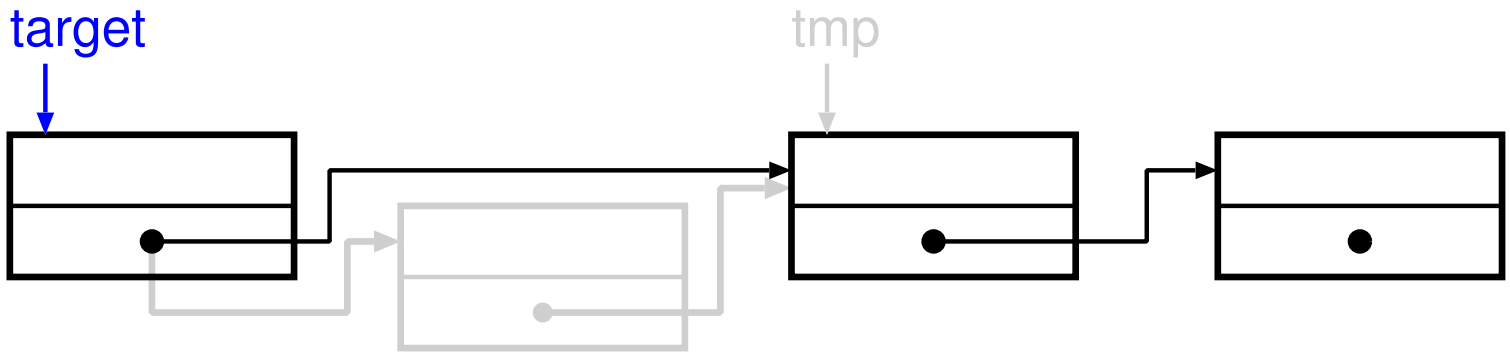
(1) 削除したい節点の手前の節点に、追跡用ポインタを向ける
(2) 削除したい節点を憶えておく
(3) 削除したい節点の次の節点に、追跡用ポインタの指す節点の次へのリンクを向ける
(4) 削除したい節点の領域解放

リストからの節点の削除



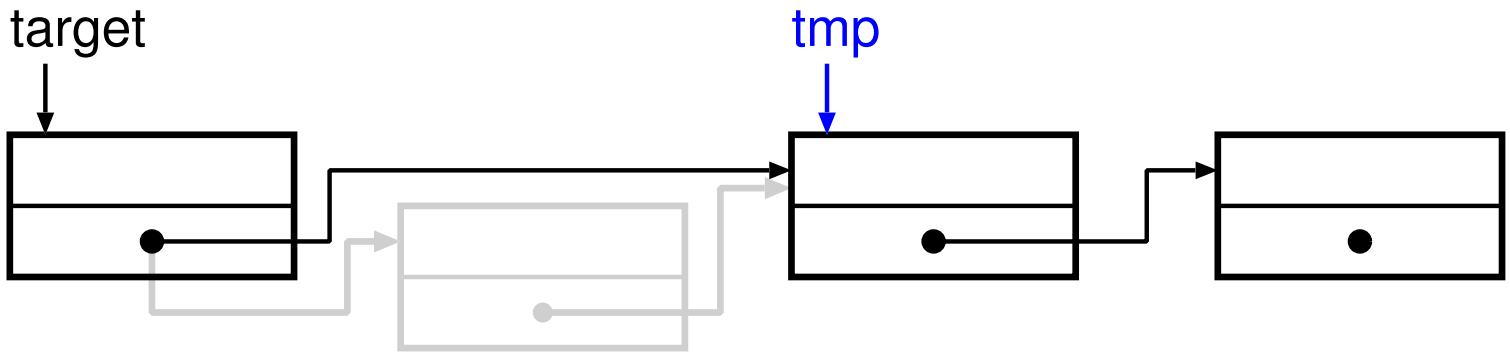
(1) 削除したい節点の手前の節点に、追跡用ポインタを向ける
(2) 削除したい節点を憶えておく
(3) 削除したい節点の次の節点に、追跡用ポインタの指す節点の次へのリンクを向ける
(4) 削除したい節点の領域解放

リストへの節点の挿入



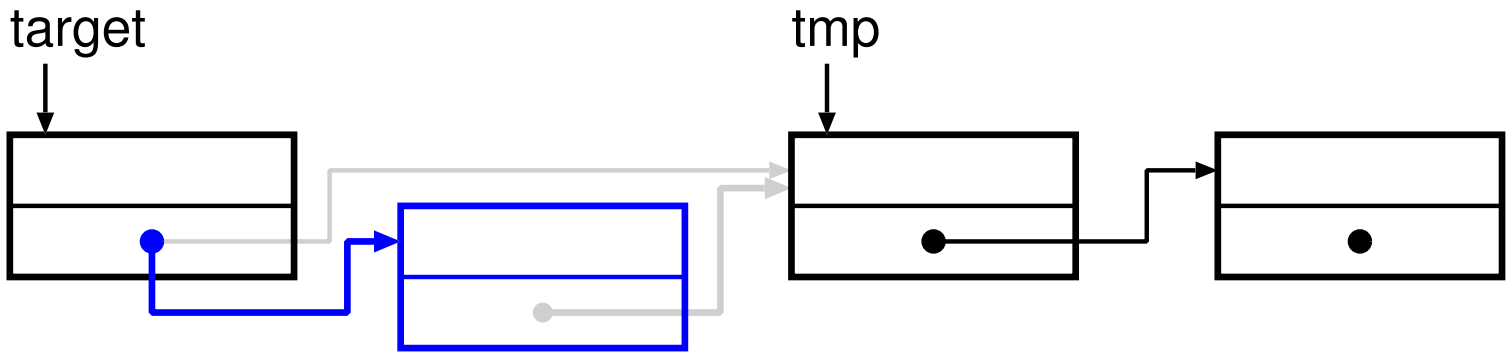
(1) 挿入したい場所の手前の節点に、追跡用ポインタを向ける
(2) 挿入したい場所の次の節点を憶えておく
(3) 挿入する節点を領域確保し、追跡用ポインタの指す節点の次へのリンクを新節点に向ける
(4) 憶えておいた節点に、新節点の次へのリンクを向ける

リストへの節点の挿入



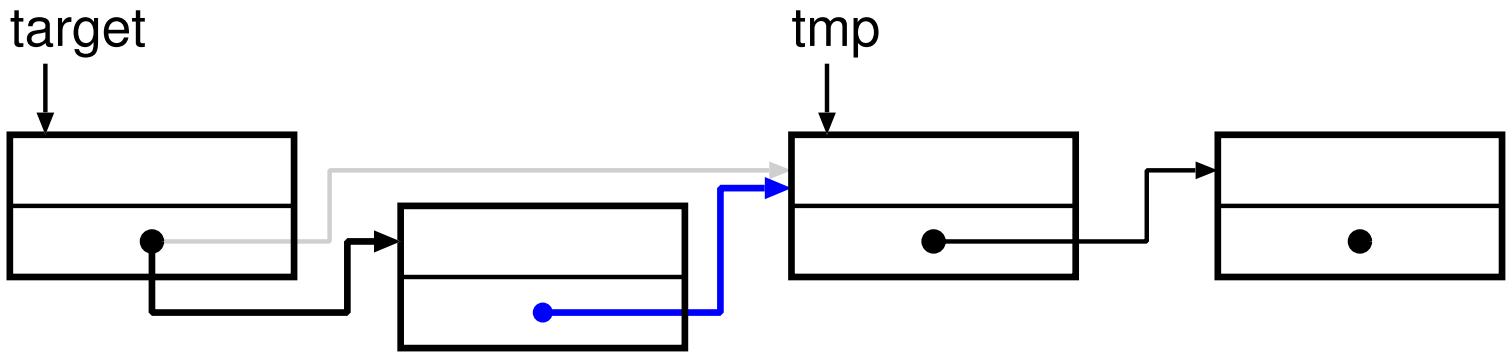
(1) 挿入したい場所の手前の節点に、追跡用ポインタを向ける
(2) 挿入したい場所の次の節点を憶えておく
(3) 挿入する節点を領域確保し、追跡用ポインタの指す節点の次へのリンクを新節点に向ける
(4) 憶えておいた節点に、新節点の次へのリンクを向ける

リストへの節点の挿入



(1) 挿入したい場所の手前の節点に、追跡用ポインタを向ける
(2) 挿入したい場所の次の節点を憶えておく
(3) 挿入する節点を領域確保し、追跡用ポインタの指す節点の次へのリンクを新節点に向ける
(4) 憶えておいた節点に、新節点の次へのリンクを向ける

リストへの節点の挿入



(1) 挿入したい場所の手前の節点に、追跡用ポインタを向ける
(2) 挿入したい場所の次の節点を憶えておく
(3) 挿入する節点を領域確保し、追跡用ポインタの指す節点の次へのリンクを新節点に向ける
(4) 憶えておいた節点に、新節点の次へのリンクを向ける

課題 3:(〆切 11/3(月))

リストを用いて、

次々と入力した整数値を

大きさの順に並べ替えて表示するプログラム

`list2.c` **を作成せよ。**

- **入力した整数を値とする新節点を、リストの適切な場所に挿入し、作成したリストの各節点の値が大きさの順になるようにする**
- `0` **が入力されたら終了**
- **先頭からリストを辿って全ての値を順に表示**