「アルゴリズムとデータ構造」資料5
連結リスト

鴨 浩靖

2009年10月27日 初版
2011年10月25日 第二版改
2013年10月22日 第三版
2019年10月30日 第四版

配列の長所と短所

配列の長所

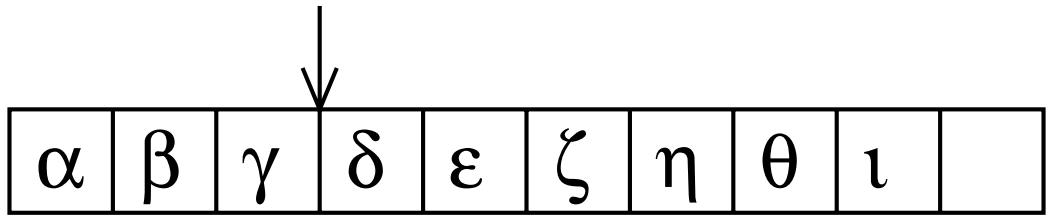
- どの要素もほぼ同じ時間でアクセスできる
- メモリ効率が良い

配列の短所

- 挿入・削除に時間がかかる
- 途中で大きさが変更できない

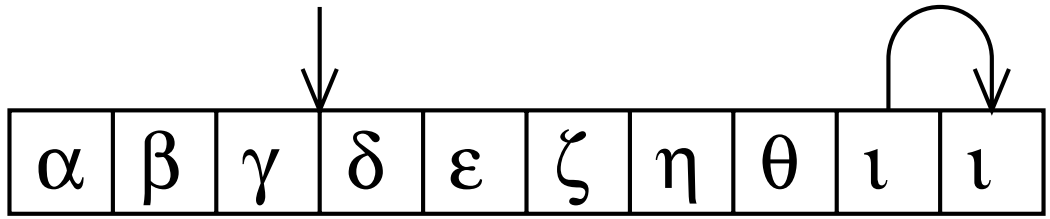
配列への要素の挿入

矢印の位置に挿入したい



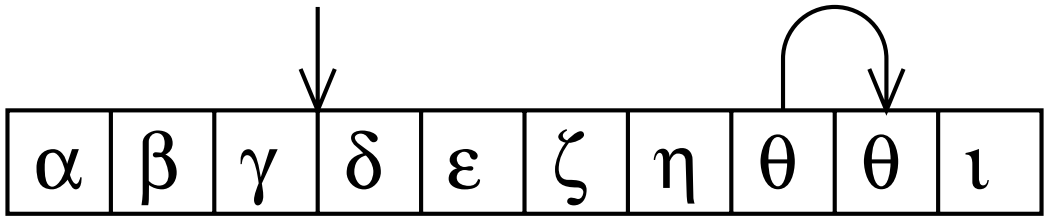
配列への要素の挿入 (つづき 1)

まず、最後の要素をずらす



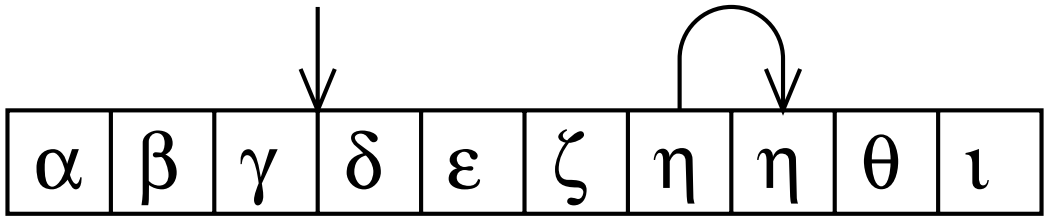
配列への要素の挿入 (つづき 2)

次に、最後の一つ前の要素をずらす



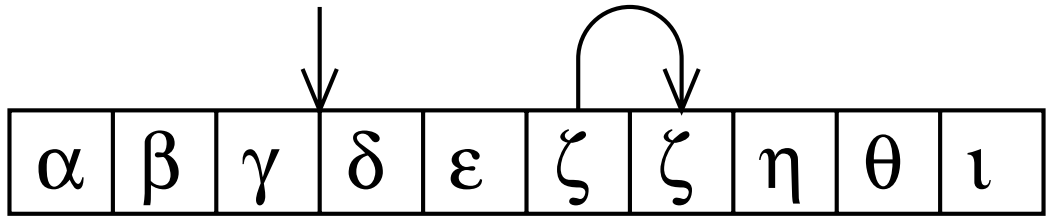
配列への要素の挿入 (つづき 3)

さらに、要素をずらす



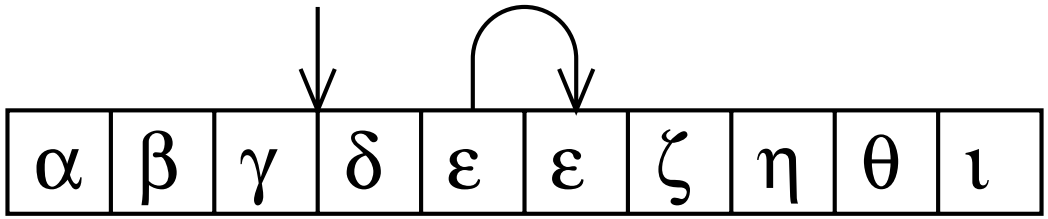
配列への要素の挿入 (つづき 4)

さらに、要素をずらす



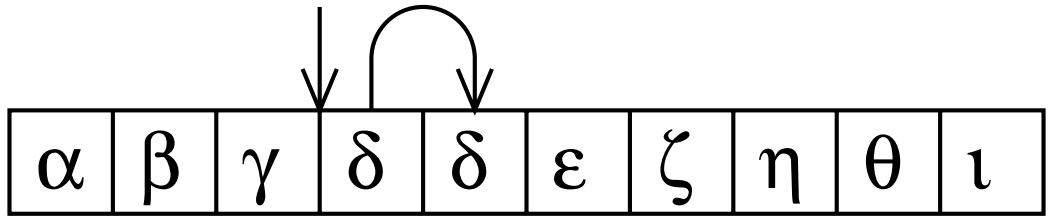
配列への要素の挿入 (つづき 5)

さらに、要素をずらす



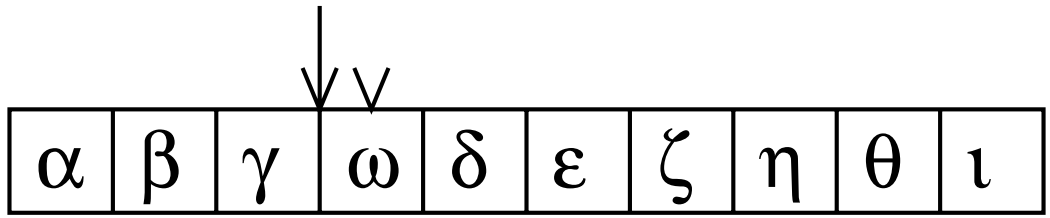
配列への要素の挿入 (つづき 6)

さらに、要素をずらす



配列への要素の挿入 (つづき 7)

最後に、空いたところに入れる



配列への要素の挿入の時間計算量

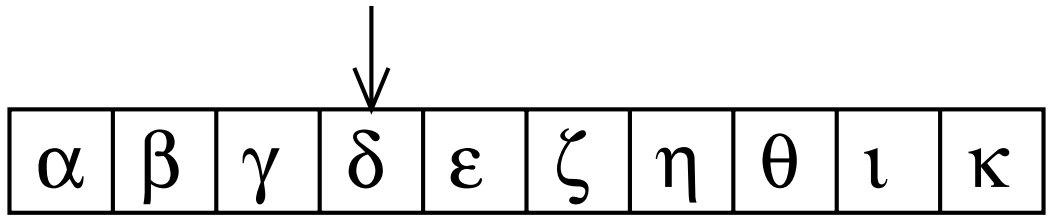
最悪　$O(n)$

平均　$O(n)$

ただし、配列の大きさを n とする。

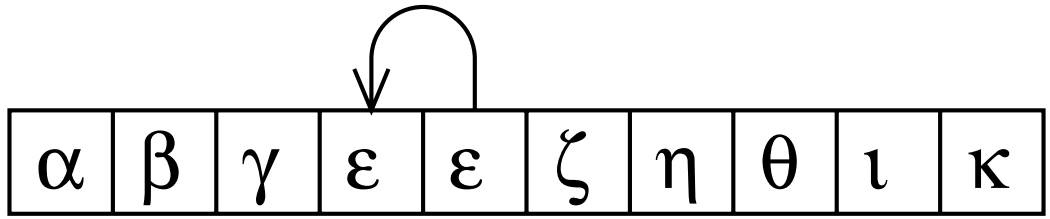
配列からの要素の削除

矢印の位置の要素を削除したい



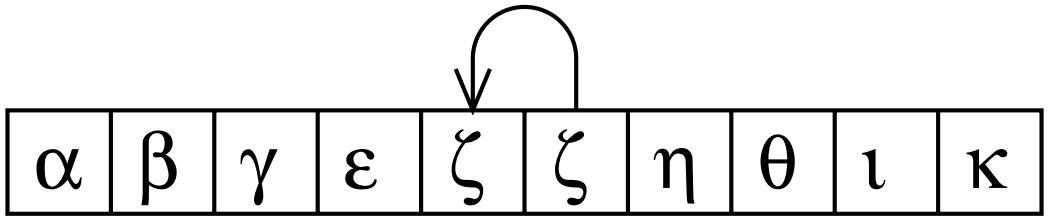
配列からの要素の削除 (つづき 1)

矢印の位置に次の要素をずらす



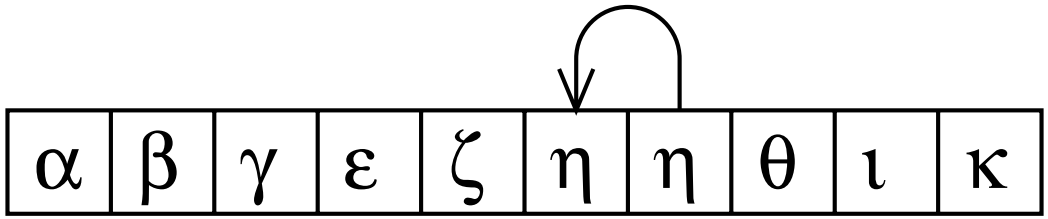
配列からの要素の削除 (つづき 2)

さらに、要素をずらす



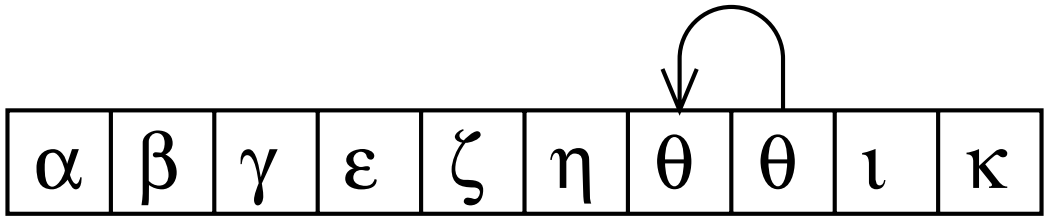
配列からの要素の削除 (つづき 3)

さらに、要素をずらす



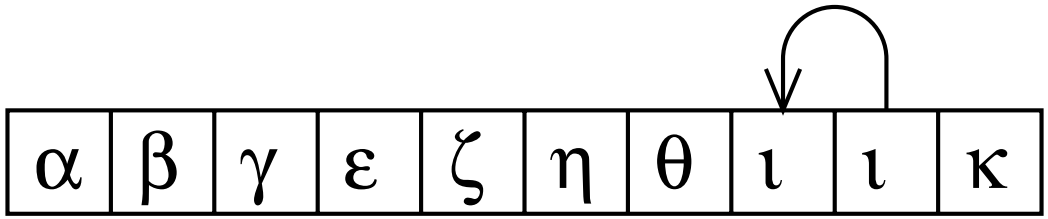
配列からの要素の削除 (つづき 4)

さらに、要素をずらす



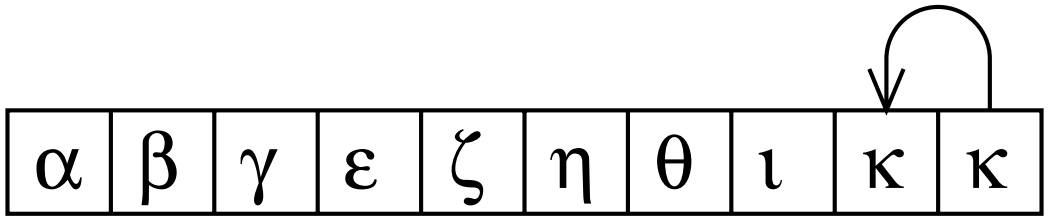
配列からの要素の削除 (つづき 5)

さらに、要素をずらす



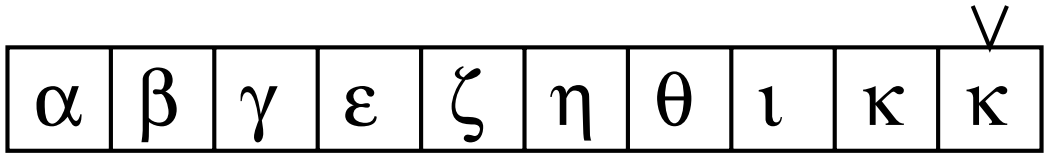
配列からの要素の削除 (つづき 6)

さらに、要素をずらす



配列からの要素の削除 (つづき 7)

ゴミが残るのは放置



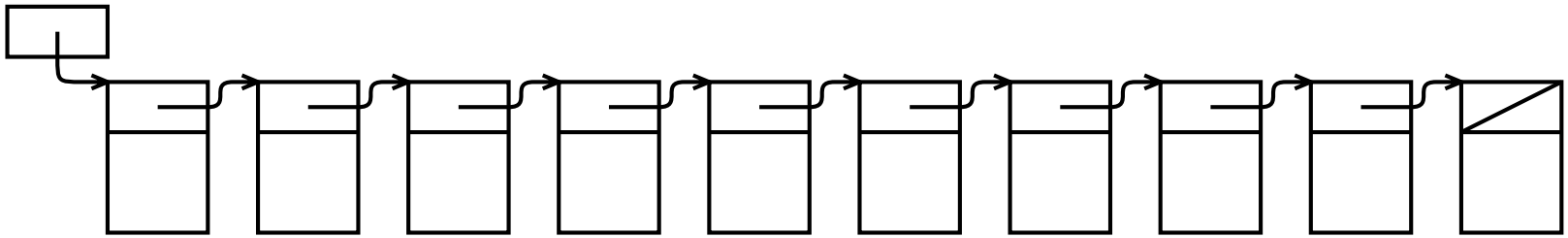
配列からの要素の削除の時間計算量

最悪　$O(n)$

平均　$O(n)$

ただし、配列の大きさを n とする。

単方向線形リストの長所と短所



単方向線形リストの長所

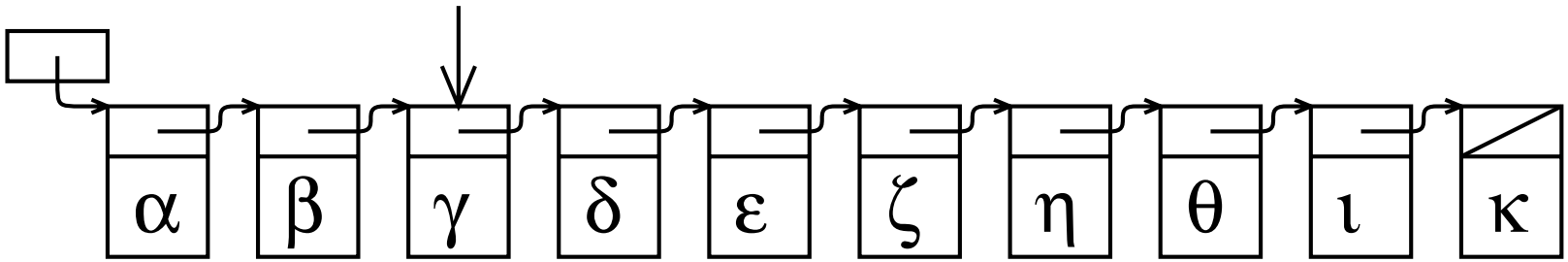
- 挿入・削除が一定の時間でできる
- 大きさが自由に変更できる

単方向線形リストの短所

- ランダムアクセスができない
- 逆向きアクセスができない
- メモリのオーバヘッドがちょっとある

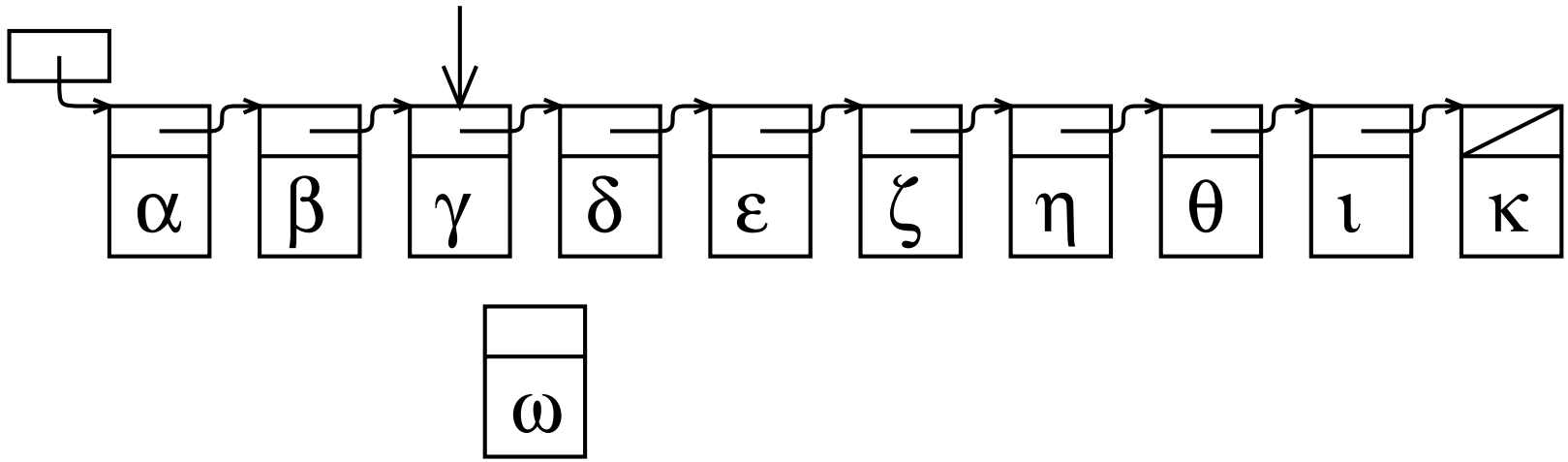
単方向線形リストへの要素の挿入

矢印のセルの次に挿入したい



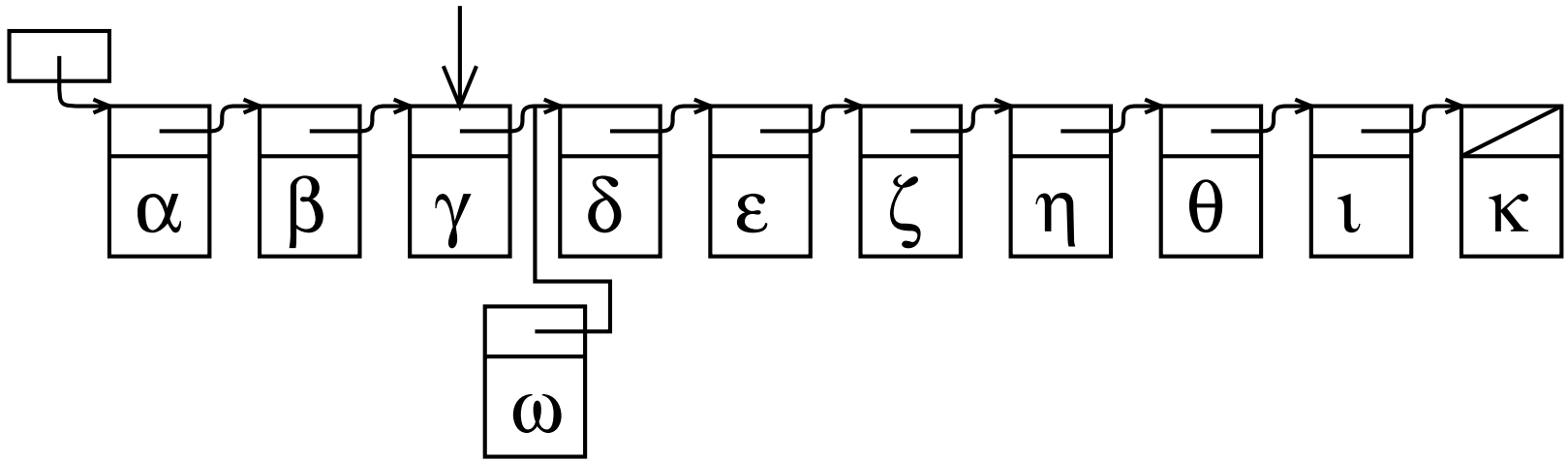
単方向線形リストへの要素の挿入 (つづき 1)

新しいセルを生成する



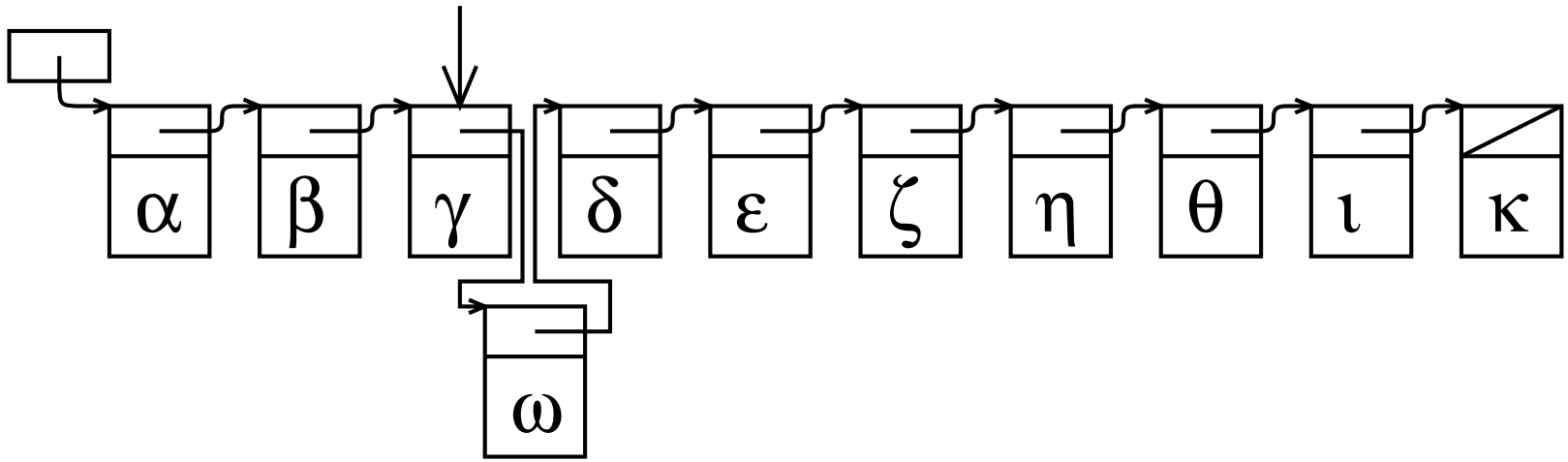
単方向線形リストへの要素の挿入 (つづき 2)

リンクをはる



単方向線形リストへの要素の挿入 (つづき 3)

リンクをつけかえる



単方向線形リストへの要素の挿入の時間計算量

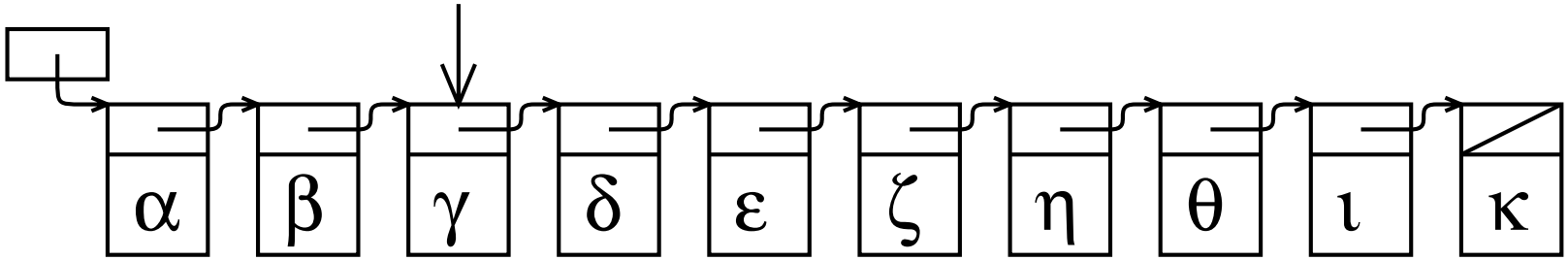
最悪　$O(1)$

平均　$O(1)$

ただし、リストの大きさを n とする。

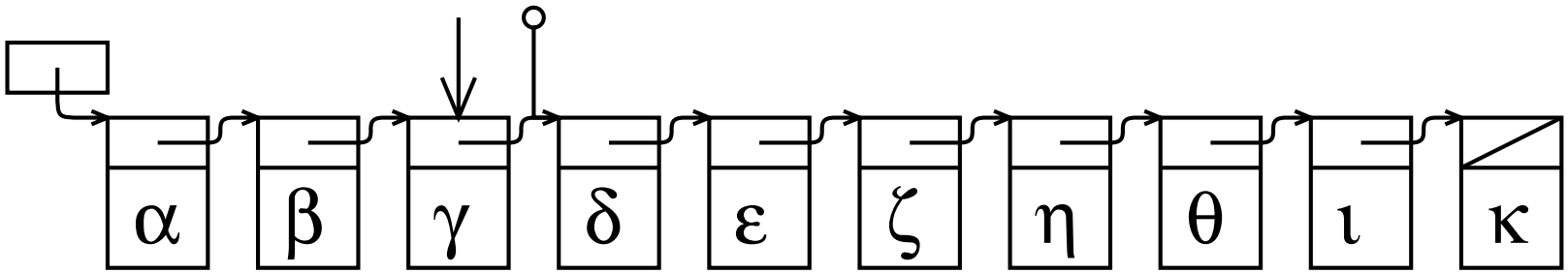
単方向線形リストからの要素の削除

矢印のセルの次のセルを削除したい



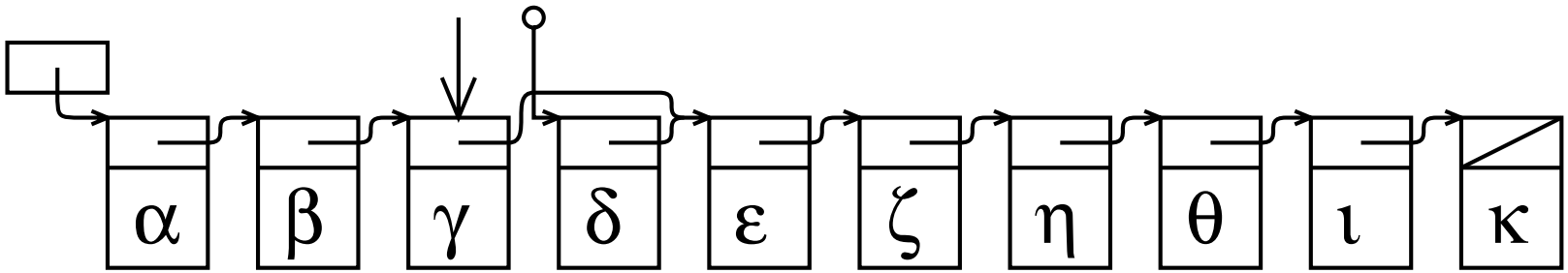
単方向線形リストからの要素の削除 (つづき 1)

リンクを退避する



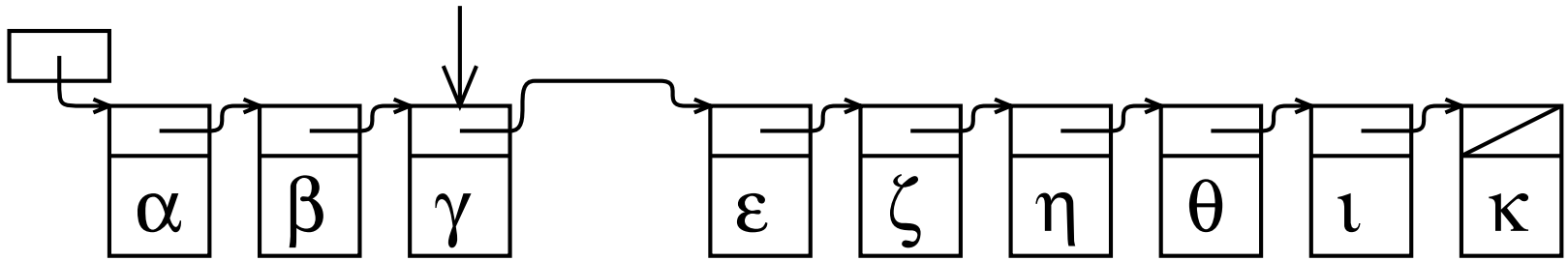
単方向線形リストからの要素の削除 (つづき 2)

リンクをつけかえる



単方向線形リストからの要素の削除 (つづき 3)

不要なセルを消去する



単方向線形リストからの要素の削除の時間計算量

最悪　$O(1)$

平均　$O(1)$

ただし、リストの大きさを n とする。

単方向線形リストの先頭への挿入・先頭の削除

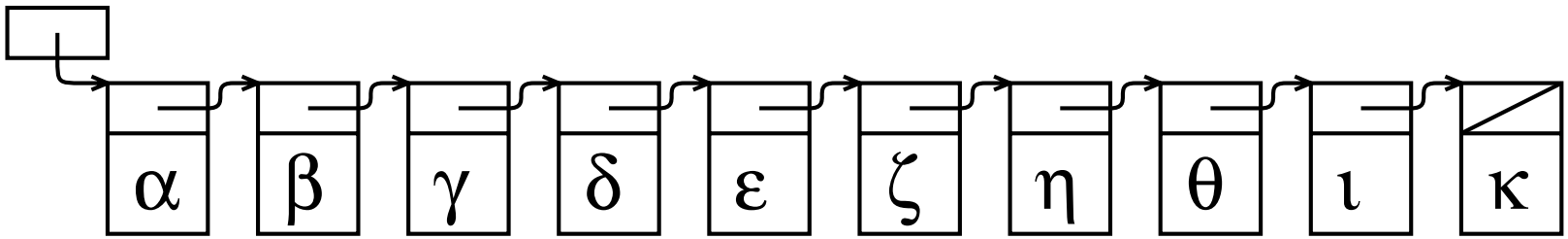
説明した方法では、先頭への挿入や先頭の削除はできない。
良く使われる対策は以下のとおり：

- ★ 先頭に挿入する関数と先頭を削除する関数を別途作成する。
- ★ 先頭のセルはダミーとする。
- ★ ポインタへのポインタを巧妙に使う。

状況に応じて使い分けること。

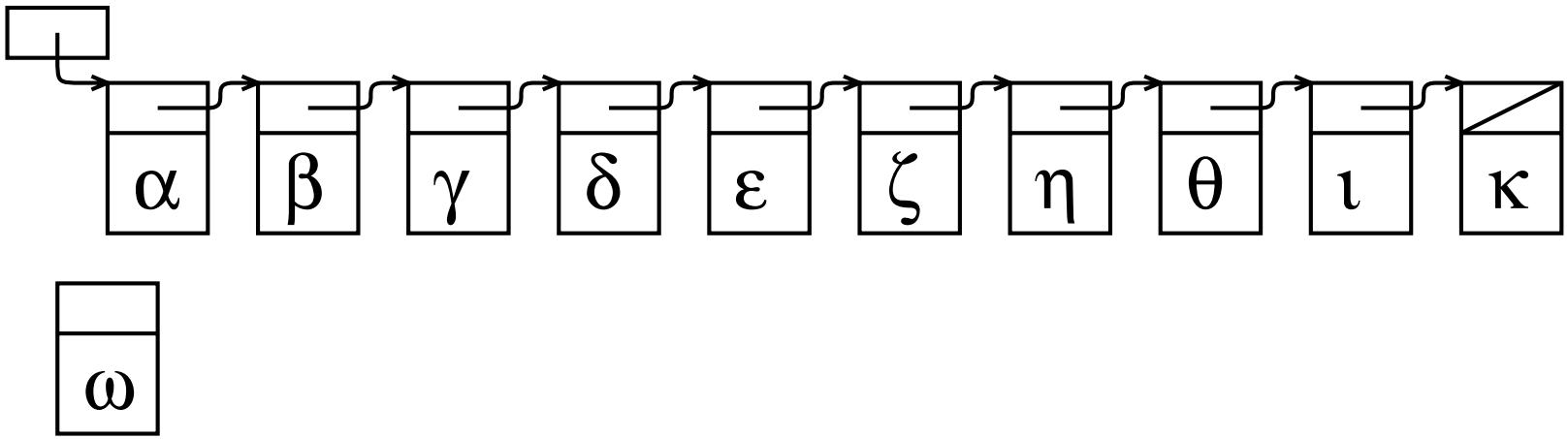
単方向線形リストの先頭への要素の挿入

先頭に挿入したい



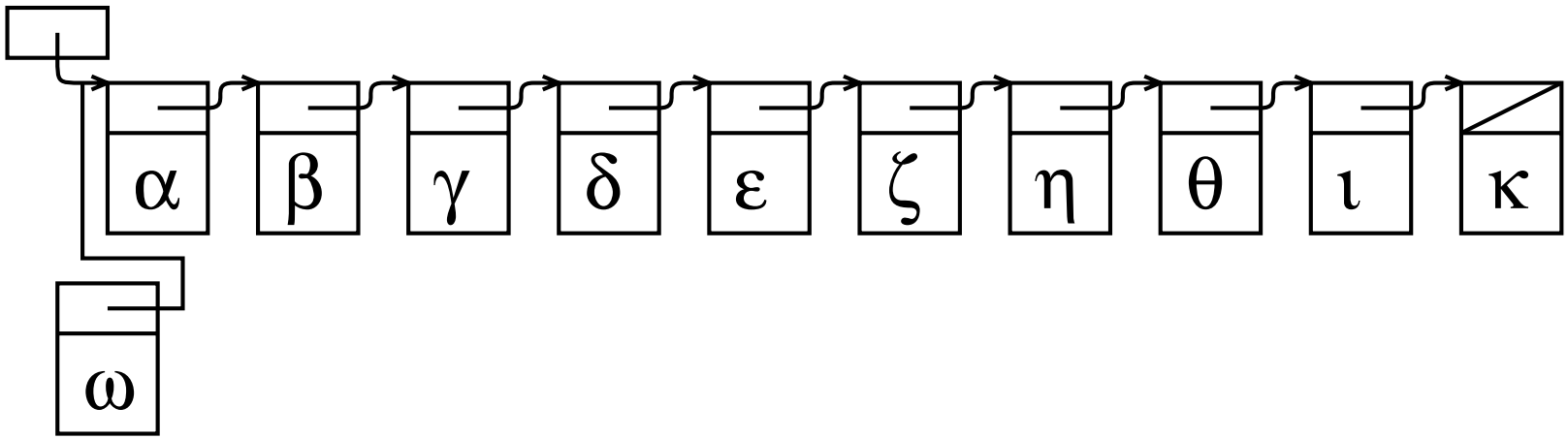
単方向線形リストの先頭への要素の挿入 (つづき 1)

新しいセルを生成する



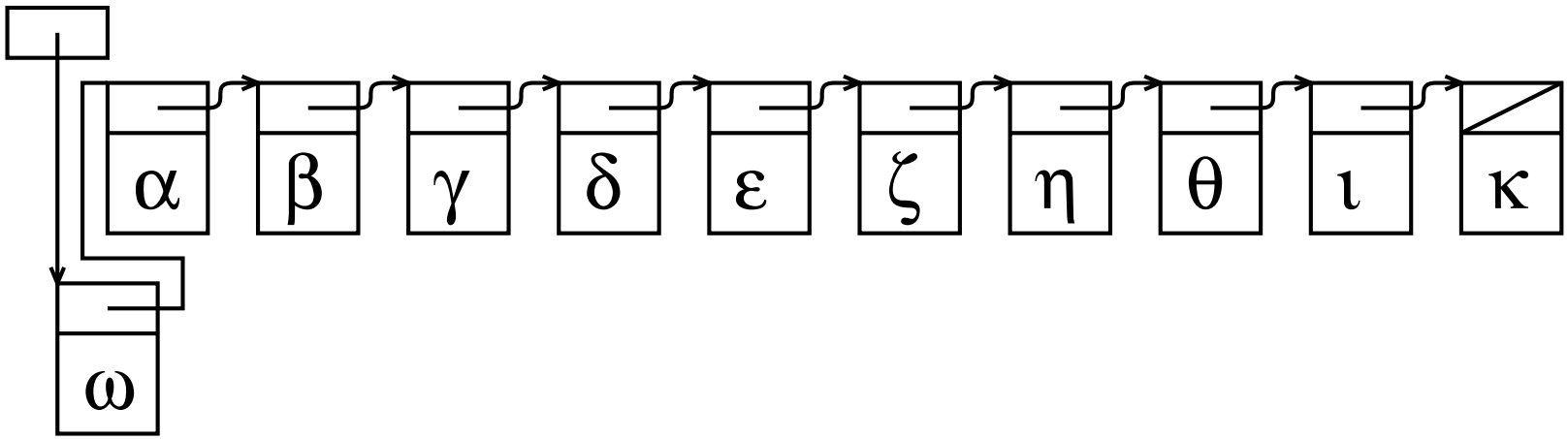
単方向線形リストの先頭への要素の挿入 (つづき 2)

リンクをはる



単方向線形リストの先頭への要素の挿入 (つづき 3)

リンクをつけかえる



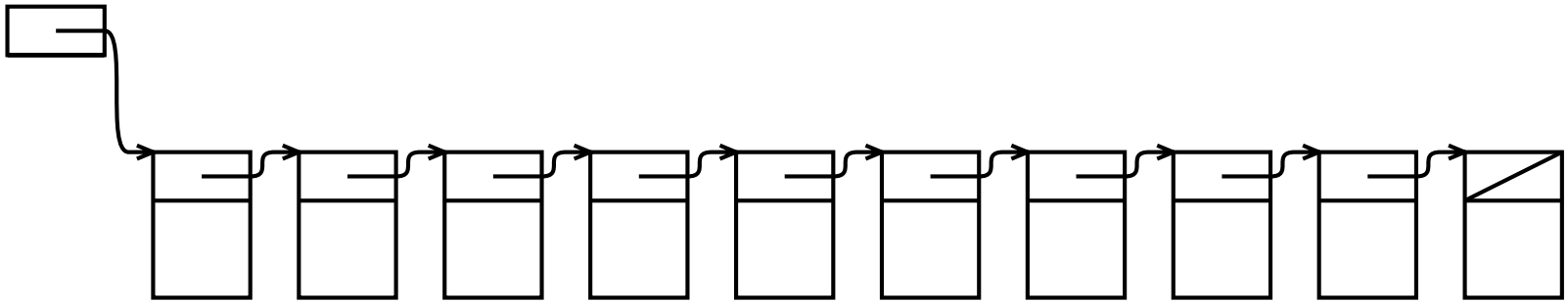
実装例 1

```
struct head {
        struct cell     *first;
};
struct cell {
        struct cell     *next;
        int             index;
        double          value;
};
```

実装例2

```
struct datum {
        int             index;
        double          value;
};
struct head {
        struct cell     *first;
};
struct cell {
        struct cell     *next;
        struct datum    *datum_ptr;
};
```

単方向線形リスト



実装例

```
struct head {
        struct cell     *first;
};
struct cell {
        struct cell     *next;
        int             index;
        double          value;
};
```

単方向線形リスト—指定されたセルの次に挿入 実装例

```
struct cell *
slist_insert_after(struct slist *list, struct cell *p,
                   int index, double value)
{
        struct cell     *q = malloc(sizeof(struct cell));
                             /* 新しいセルを生成する */
        q->index = index;    /* データをセットする */
        q->value = value;    /* データをセットする */
        q->next = p->next;   /* リンクをはる */
        p->next = q;         /* リンクをつなぎかえる */
        return q;
}
```

単方向線形リスト—指定されたセルの次のセルを削除 実装例

```
void
slist_delete_after(struct slist *list, struct cell *p)
{
        struct cell     *q = p->next;
                                /* リンクを退避する */
        p->next = q->next;      /* リンクをつなぎかえる */
        free(q);                /* 不要なセルを削除する */
}
```

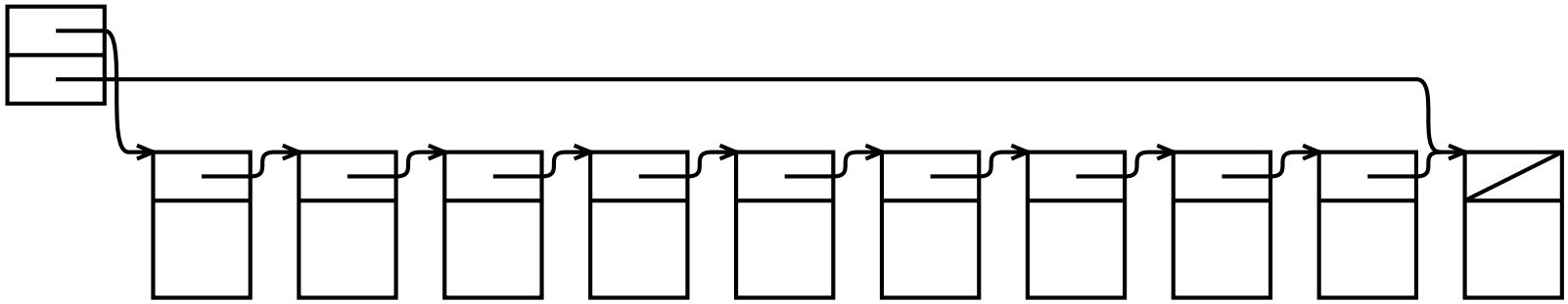
単方向線形リスト—先頭に挿入 実装例

```
struct cell *
slist_insert_first(struct slist *list, int index, double va
{
        struct cell     *q = malloc(sizeof(struct cell));
                                /* 新しいセルを生成する */
        q->index = index;   /* データをセットする */
        q->value = value;   /* データをセットする */
        q->next = list->first; /* リンクをはる */
        list->first = q;    /* リンクをつなぎかえる */
        return q;
}
```

単方向線形リスト—先頭のセルを削除 実装例

```
void
slist_delete_after(struct slist *list)
{
        struct cell     *q = list->first;
                                /* リンクを退避する */
        list->first = q->next; /* リンクをつなぎかえる */
        free(q);                /* 不要なセルを削除する */
}
```

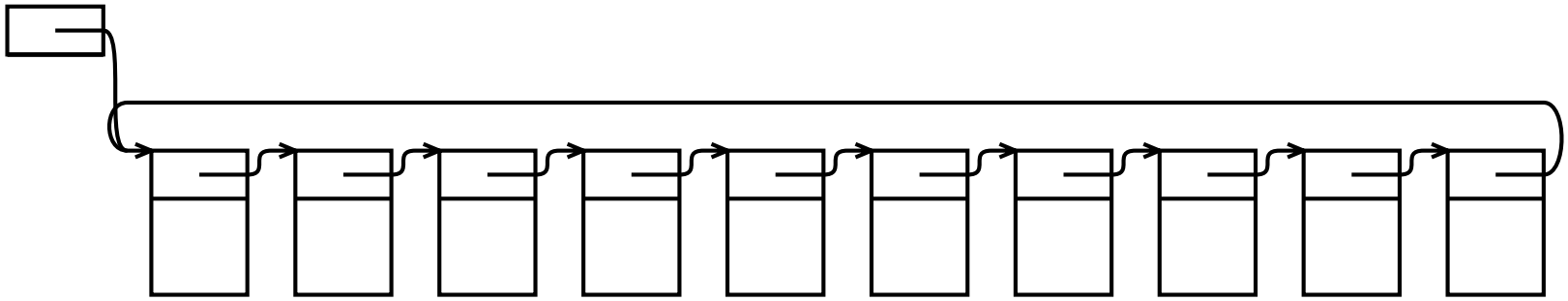
単方向線形リストの変種—末尾つき単方向線形リスト



実装例

```
struct head {
        struct cell     *first;
        struct cell     *last;
};
struct cell {
        struct cell     *next;
        int             index;
        double          value;
};
```

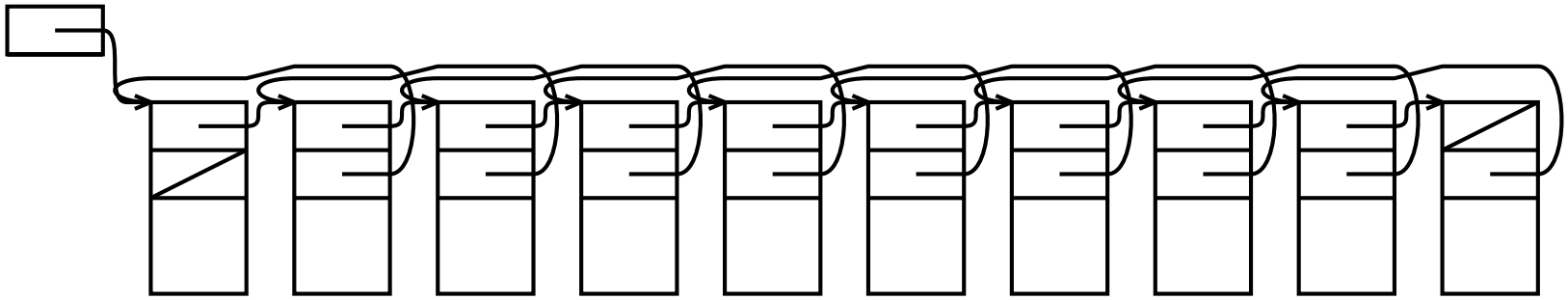
単方向線形リストの変種—単方向循環リスト



実装例

```
struct head {
        struct cell     *first;
};
struct cell {
        struct cell     *next;
        int             index;
        double          value;
};
```

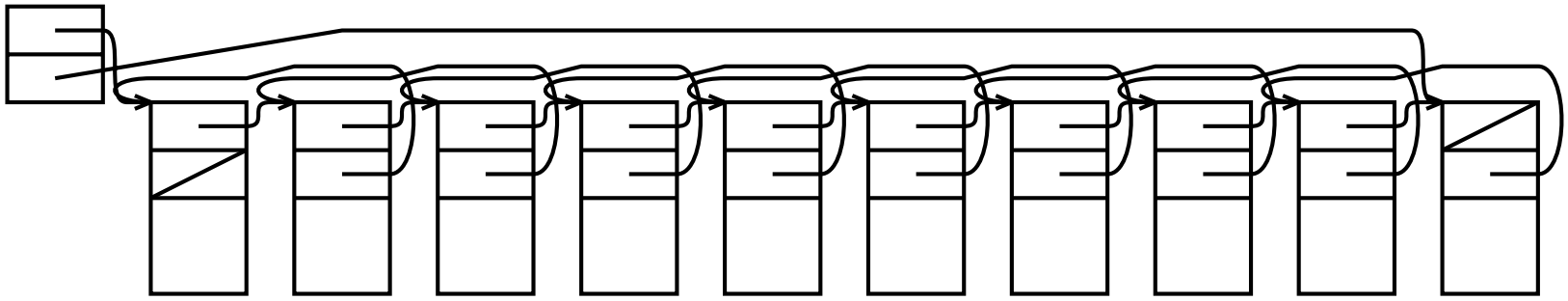
双方向線形リスト



実装例

```
struct head {
        struct cell     *first;
};
struct cell {
        struct cell     *next;
        struct cell     *prev;
        int             index;
        double          value;
};
```

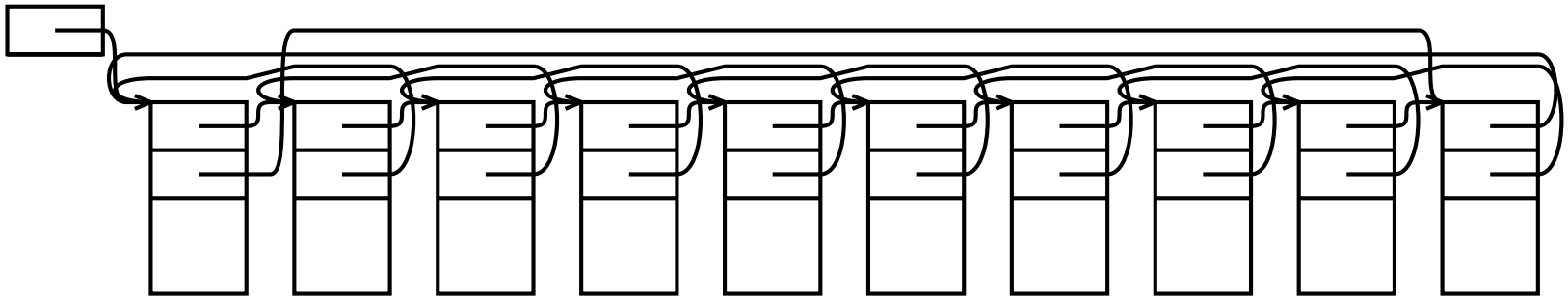
双方向線形リストの変種—末尾つき



実装例

```
struct head {
        struct cell     *first;
        struct cell     *last;
};
struct cell {
        struct cell     *next;
        struct cell     *prev;
        int             index;
        double          value;
};
```

双方向線形リストの変種—循環



実装例

```
struct head {
        struct cell     *first;
};
struct cell {
        struct cell     *next;
        struct cell     *prev;
        int             index;
        double          value;
};
```

まとめ

- 単方向線形リスト
- 末尾つき単方向線形リスト
- 単方向循環リスト
- 双方向線形リスト
- 末尾つき双方向線形リスト
- 双方向循環リスト