データ構造とアルゴリズム (第13回)

静岡大学システム工学科

安藤 和敏

2008.02.04

今日のテーマ

リストとその実装

- リストと呼ばれるデータ構造について学び，
- C言語を用いてリストの実装の仕方を学ぶ．

リストの実装のためには，先週までに習った，構造体とメモリの動的割当てを用いる．

「Cによるアルゴリズムとデータ構造」p.26–31

リストとは何か?

要素の列:

$$[a_0, a_1, \cdots, a_{n-1}]$$

をリストと呼ぶ. ここで, 要素というのは整数であったり実数であったり, 文字列であったりする. (数学でいう数列に近い.) 要素がないリスト (つまり, $n = 0$) を空リストと呼ぶ.

例: 以下のものはリストである.

$$[30, 2, 9, 7, 3]$$

$$[\text{“静岡”}, \text{“浜松”}, \text{“磐田”}, \text{“湖西”}, \text{“袋井”}]$$

リストに対する操作

♠ i番目の要素の次にxを挿入:

$$[a_0, a_1, \cdots, a_{i-1}, a_i, \cdots, a_{n-1}]$$
$$\Downarrow$$
$$[a_0, a_1, \cdots, a_{i-1}, x, a_i, \cdots, a_{n-1}]$$

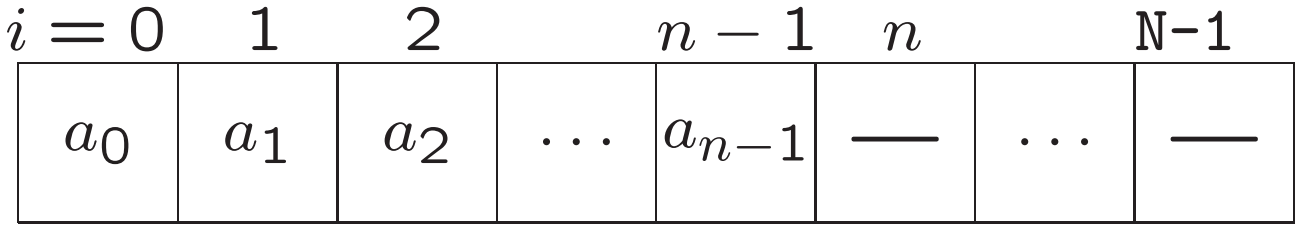
♠ i番目の要素の次の要素を削除:

$$[a_0, a_1, \cdots, a_{i-1}, a_i, a_{i+1} \cdots, a_{n-1}]$$
$$\Downarrow$$
$$[a_0, a_1, \cdots, a_{i-1}, a_{i+1}, \cdots, a_{n-1}]$$

リストの配列を用いた実装

```
int a[N];
```

(int型の配列を考えるが，他の型double，char *などでも同じである.)



リストの配列を用いた実装の欠点

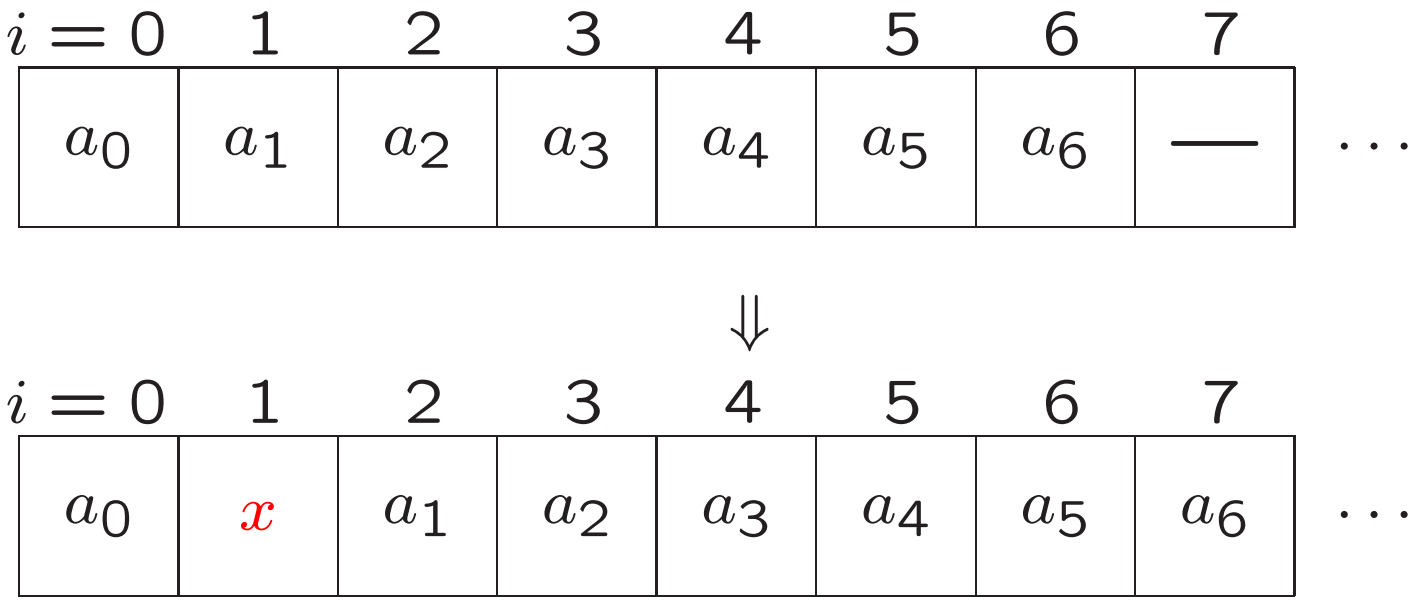
```
elementtype a[N];
```

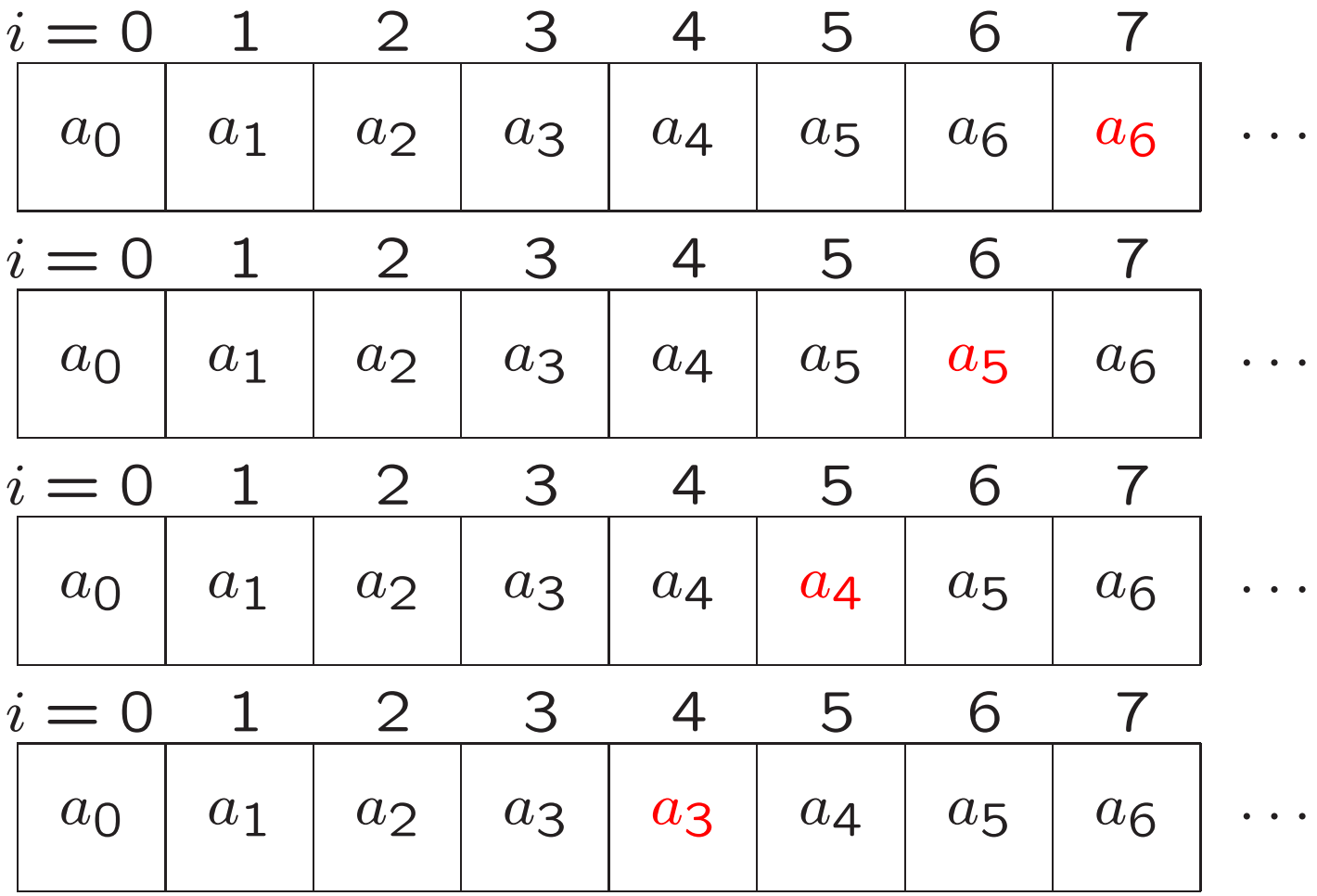
というように宣言されていたら，リストの長さはNを越えることはできない.

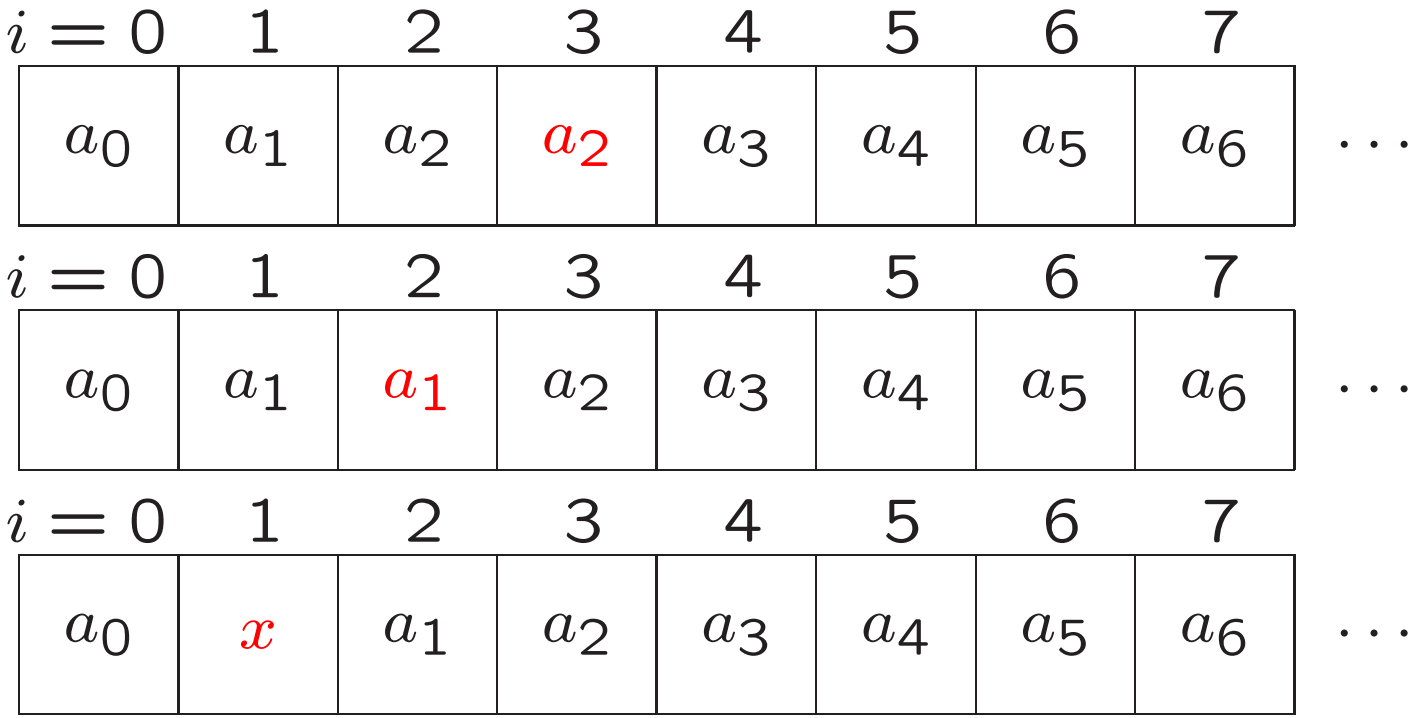
$\Rightarrow$ 前回説明したように，メモリ不足やメモリの無駄使いの問題が生じる.

挿入や削除を実行する計算の手間（時間計算量）は $O(n)$.

♠ 例えば$n = 7$として，a_0の次にxを挿入する場合.




$i = 0$ 1 2 3 4 5 6 7
a_0 a_1 a_2 a_3 a_4 a_5 a_6 a_6 $\cdots$
$i = 0$ 1 2 3 4 5 6 7
a_0 a_1 a_2 a_3 a_4 a_5 a_5 a_6 $\cdots$
$i = 0$ 1 2 3 4 5 6 7
a_0 a_1 a_2 a_3 a_4 a_4 a_5 a_6 $\cdots$
$i = 0$ 1 2 3 4 5 6 7
a_0 a_1 a_2 a_3 a_3 a_4 a_5 a_6 $\cdots$



全部で7回の代入が行われた!

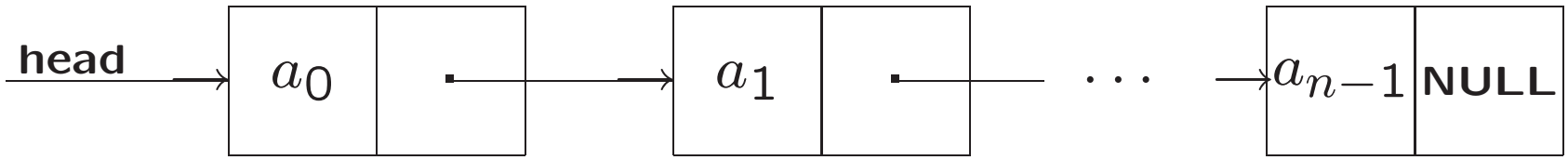
削除についても同様で$O(n)$の計算手間がかかる.

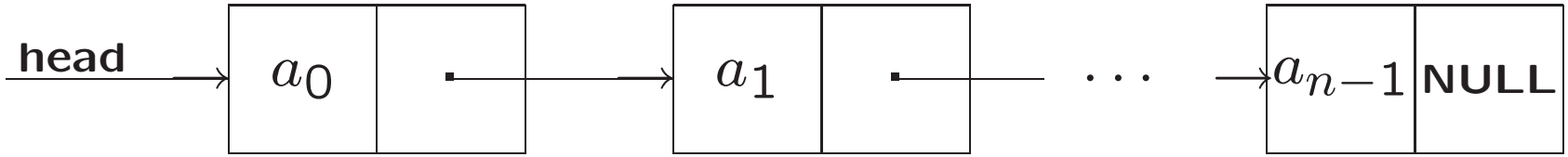
リストのポインタを用いた実装 ── 連結リスト ──

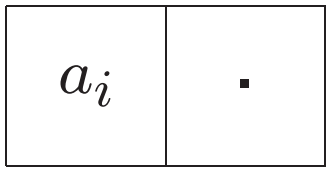
リスト

$$[a_0, a_1, \cdots, a_{n-1}]$$

を連結リストで表現することもできる.





連結リストは，セル 

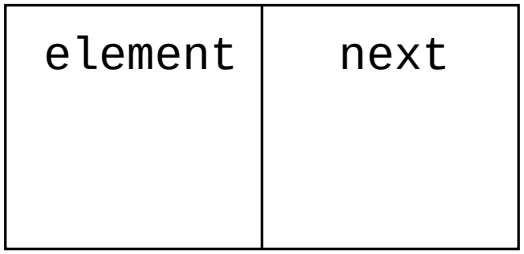
がポインタで継がれたものであり，

各セルは要素と次のセルへのポインタから成る構造体である．

head は，一番最初のセルを指すポインタであり，最後のセルの中のポインタは**NULL**である．

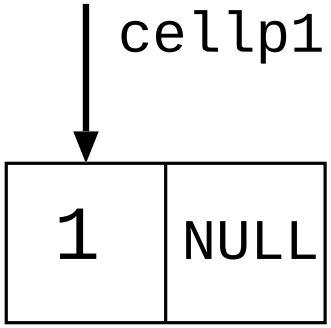
連結リストのCによる実装

♠ セルの型定義



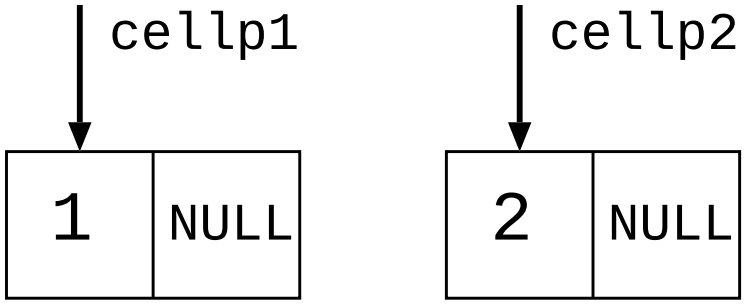

```
typedef struct cell{
  int element;
  struct cell *next;
} cell;
cell *head;
```

♠ セル作る



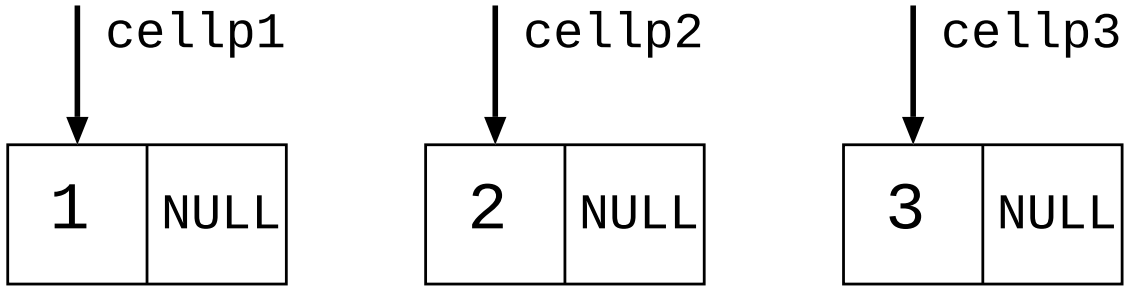

```
cell *cellp1,*cellp2,*cellp3;


cellp1 = (cell *)malloc(sizeof(cell));
cellp1->element=1;
cellp1->next=NULL;
```

♠ もう一個セル作る



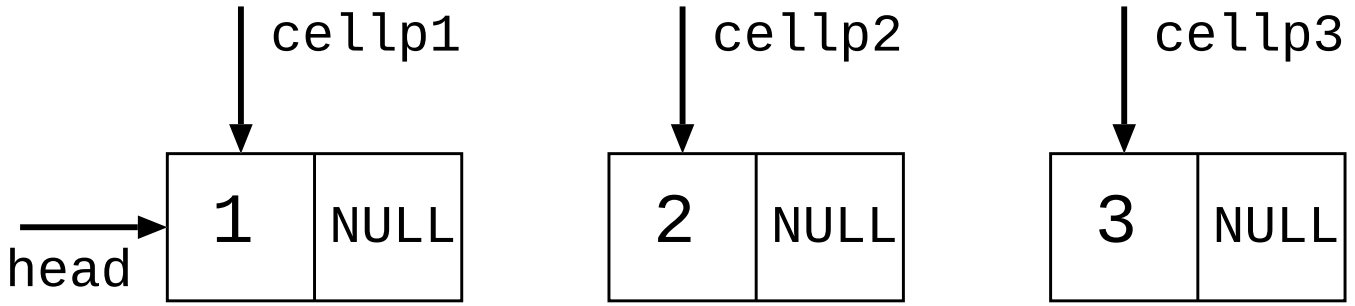

```
cellp2 = (cell *)malloc(sizeof(cell));

cellp2->element=2;

cellp2->next=NULL;
```

♠ さらに，もう一個セル作る



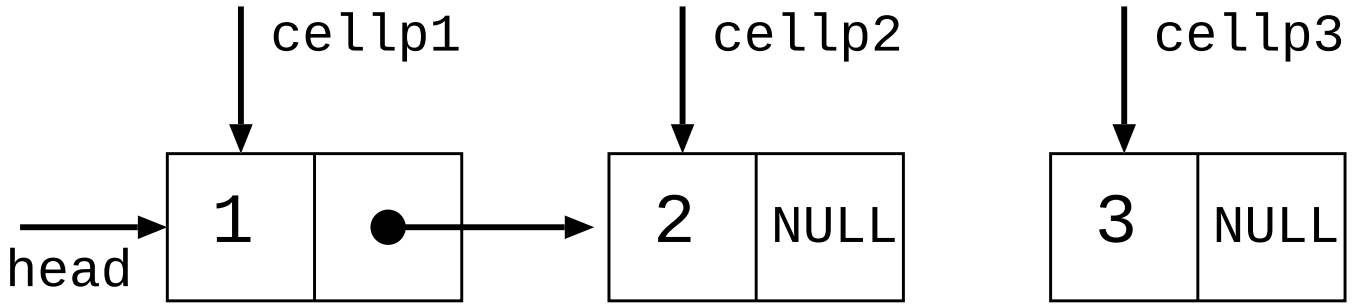

```
cellp3 = (cell *)malloc(sizeof(cell));

cellp3->element=3;

cellp3->next=NULL;
```

♠ セルを継げてみよう



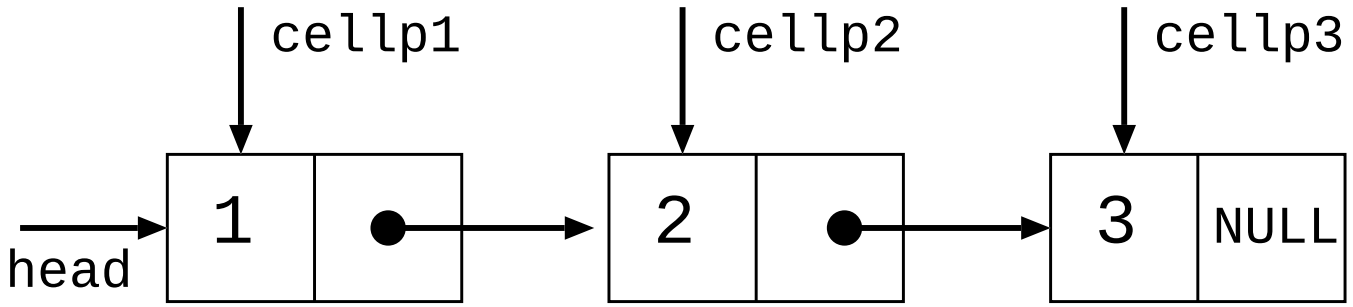

```
head = cellp1;
```

♠ セルを継げてみよう




```
cellp1->next = cellp2;
```

♠ セルを継げてみよう




```
cellp2->next = cellp3;
```

長さ３の連結リストができた.

連結リストの走査

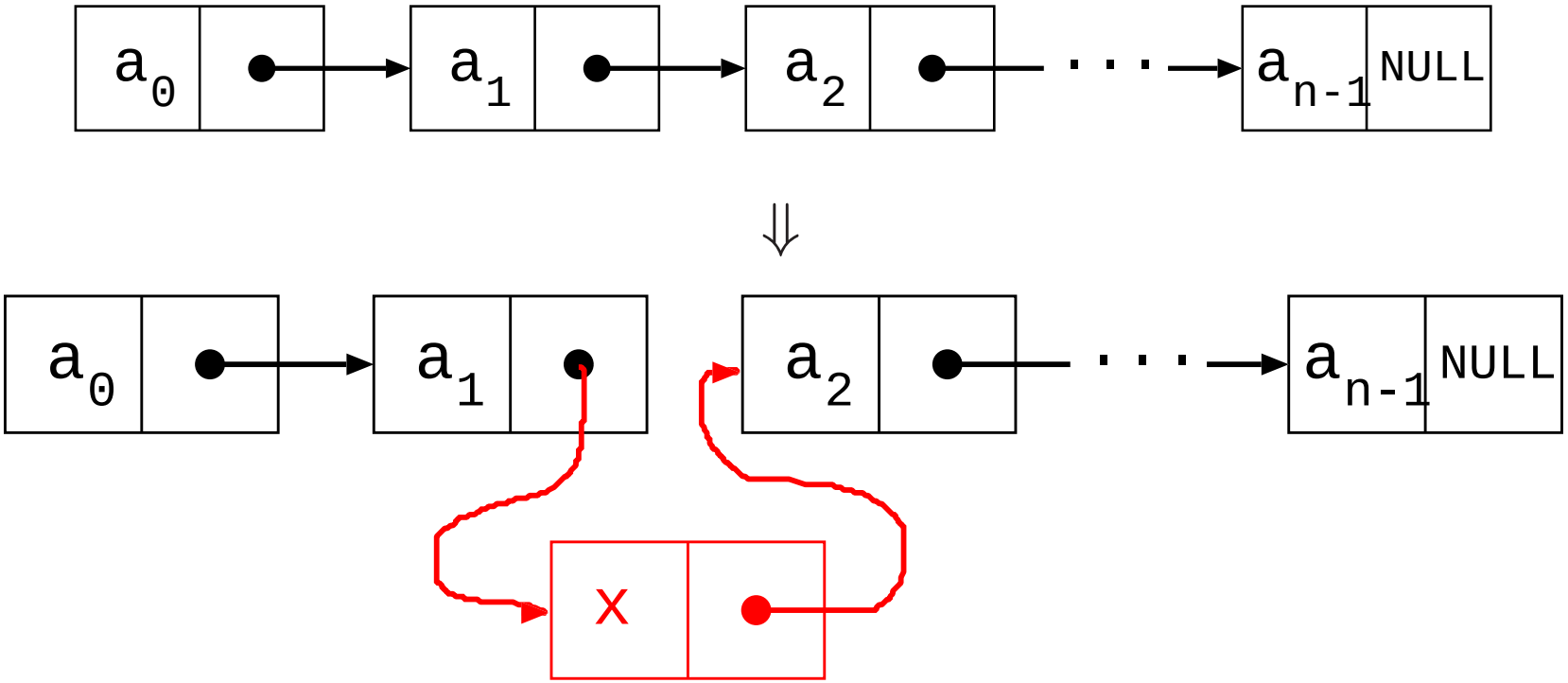
```
void writelist(void) {
  cell *temp=head;
  printf("List: [");
  while (temp != NULL) {
    printf("%5ld", temp->element);
    if (temp->next != NULL) putchar(',');
    temp = temp->next;
  }
  printf("]\n");
}
```

演習の時間

完全なプログラムは，　この講義の **Web** ページに置いてある **(List0.c)**．今説明したことを実行してみなさい．

連結リストにおける挿入と削除

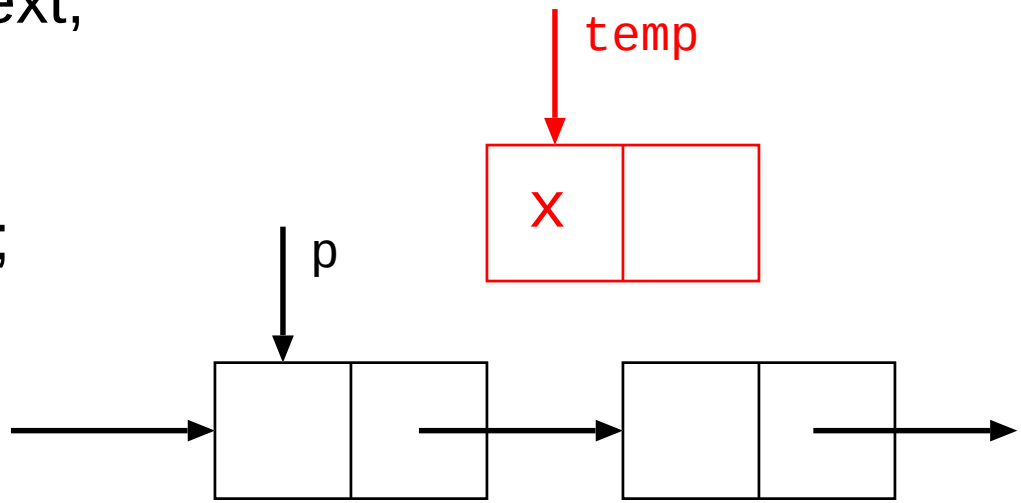
♠ 挿入



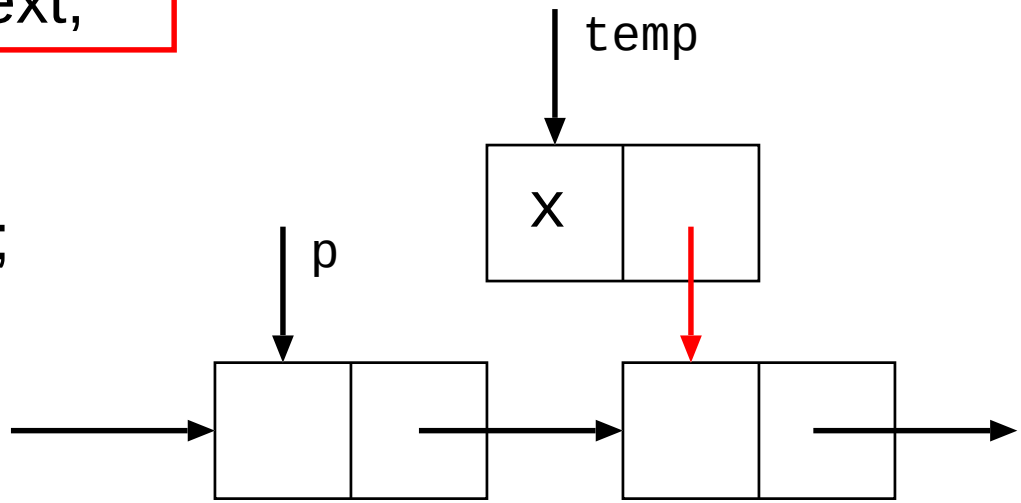
関数 **insert(int x, cell *p)** は，整数xが入ったセルをポインタ **p** が指すセルの次へ挿入する関数．もし，**p** == NULLのときはリストの一番最初に挿入する．

```
void insert(int x, cell *p) {
  cell *temp;
  temp = (cell *)malloc(sizeof(cell));
  temp->element = x;
  if (p != NULL) {
    temp->next = p->next;
    p->next = temp;
  } else {
    temp->next = head;
    head = temp;
  }
}
```

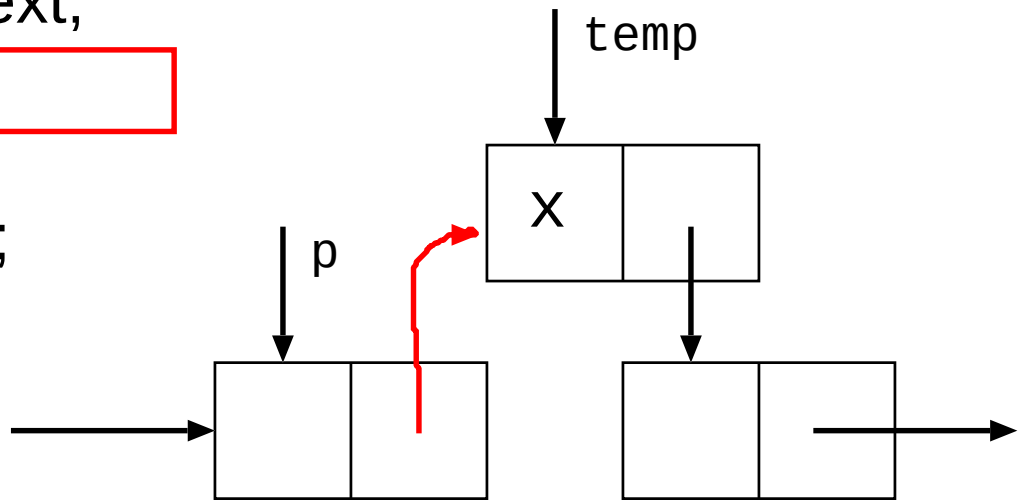
```
void insert(int x, cell *p) {
  cell *temp;
  temp = (cell *)malloc(sizeof(cell));
  temp->element = x;
  if (p != NULL) {
    temp->next = p->next;
    p->next = temp;
  } else {
    temp->next = head;
    head = temp;
  }
}
```



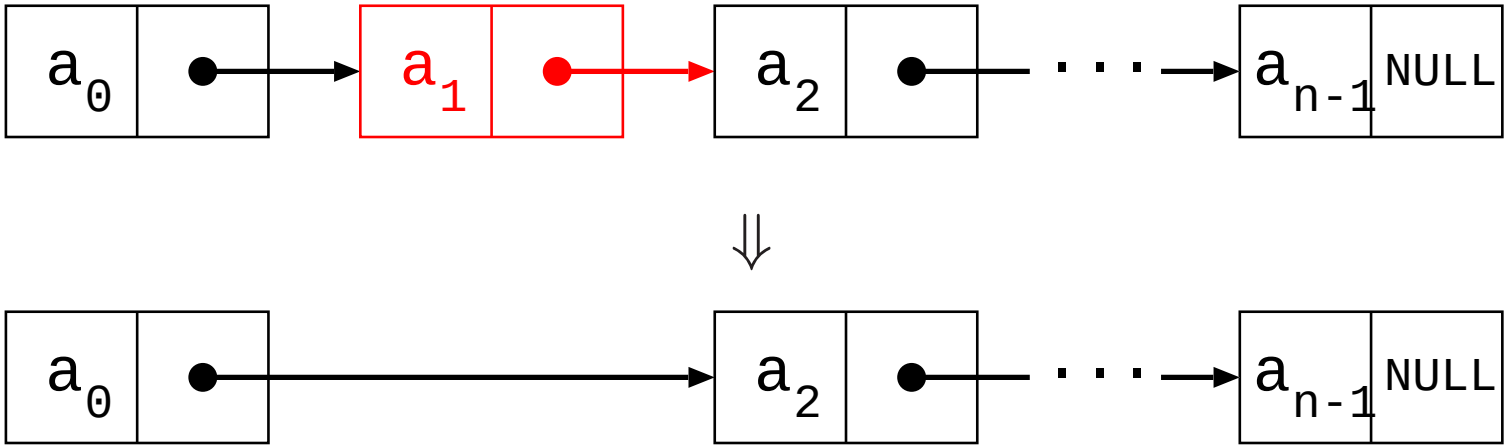
```
void insert(int x, cell *p) {
  cell *temp;
  temp = (cell *)malloc(sizeof(cell));
  temp->element = x;
  if (p != NULL) {
    temp->next = p->next;
    p->next = temp;
  } else {
    temp->next = head;
    head = temp;
  }
}
```



```
void insert(int x, cell *p) {
  cell *temp;
  temp = (cell *)malloc(sizeof(cell));
  temp->element = x;
  if (p != NULL) {
    temp->next = p->next;
    p->next = temp;
  } else {
    temp->next = head;
    head = temp;
  }
}
```



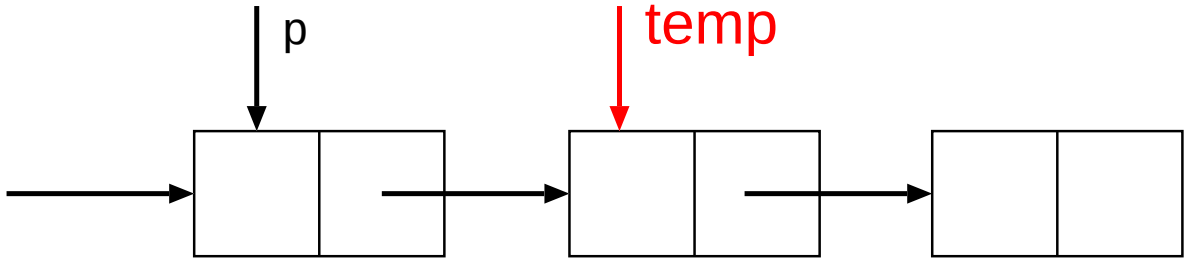
♠ 削除



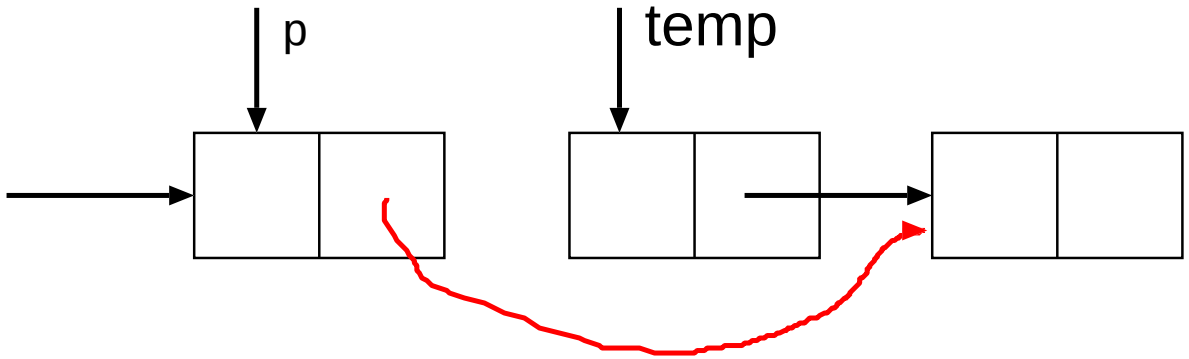
関数 **delete(cell *p)** は，ポインタ **p** が指すセルの次のセルを削除する関数．もし，**p** == NULLのときはリストの一番最初のセルを削除する．

```
void delete(cell *p) {
    cell *temp;
    if (p == NULL) {
      temp = head;
      head = head->next;
      free(temp);
    } else if (p->next != NULL) {
      temp = p->next;
      p->next = p->next->next;
      free(temp);
    }
}
```

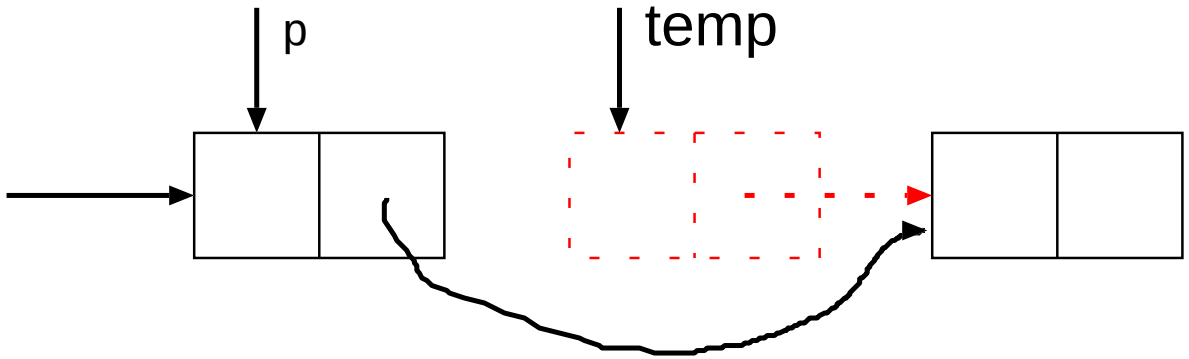
```
void delete(cell *p) {
    cell *temp;
    if (p == NULL) {
      temp = head;
      head = head->next;
      free(temp);
    } else if (p->next != NULL) {
      temp = p->next;
      p->next = p->next->next;
      free(temp);
    }
}
```



```
void delete(cell *p) {
    cell *temp;
    if (p == NULL) {
      temp = head;
      head = head->next;
      free(temp);
    } else if (p->next != NULL) {
      temp = p->next;
      p->next = p->next->next;
      free(temp);
    }
}
```



```
void delete(cell *p) {
    cell *temp;
    if (p == NULL) {
      temp = head;
      head = head->next;
      free(temp);
    } else if (p->next != NULL) {
      temp = p->next;
      p->next = p->next->next;
      free(temp);
    }
}
```



演習の時間

完全なプログラムは，この講義の **Web** ページに置いてある **(List2.c)**．このプログラムを参考にして，関数 **insert** と **delete** を定義し，これらの関数を **main** 関数内で，数回呼びだすようなプログラムを作りなさい．