プログラミングII

第8回: リスト(その1)

2019年01月15日(水)

筑波大学 情報メディア創成学類
三河 正彦

リンクトリスト

- リスト:順番に並べられたもの
  - 例:配列

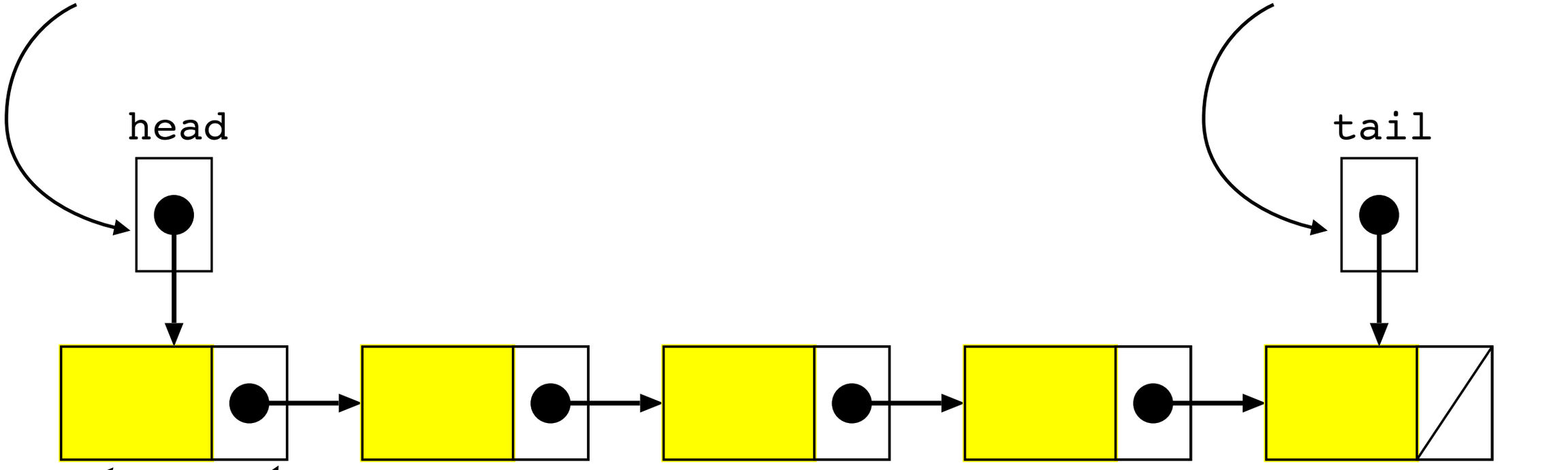
- リンクトリスト
  - データを順番につないだもの
    - 各データは
      - データの要素
      - 次のデータへのポインタ

    を持つ.

単方向リスト

リストの先頭
(特殊なポインタ)

リストの末尾
(特殊なポインタ)



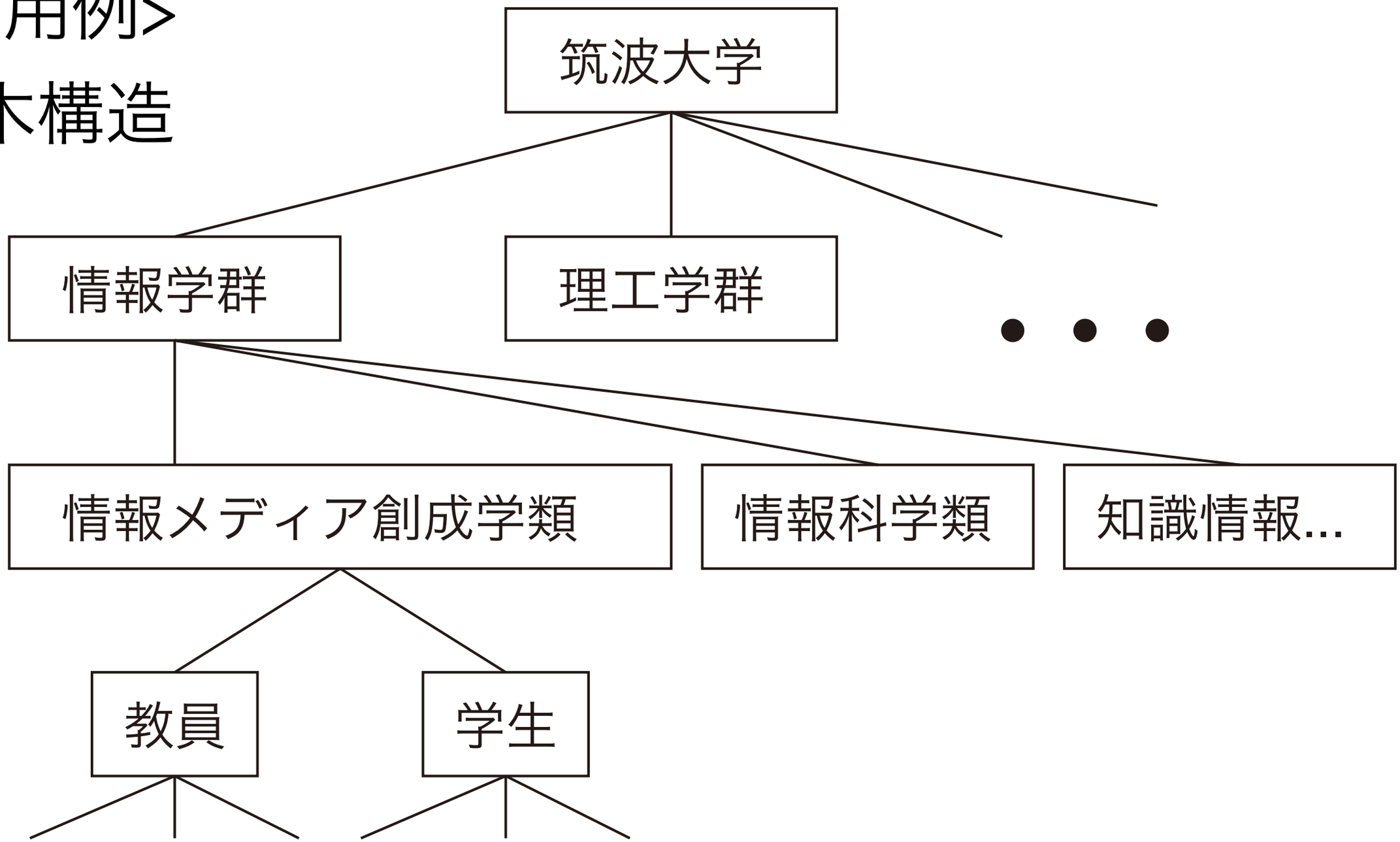
データ要素　次の箱へのポインタ

末尾の箱は
指す場所無し

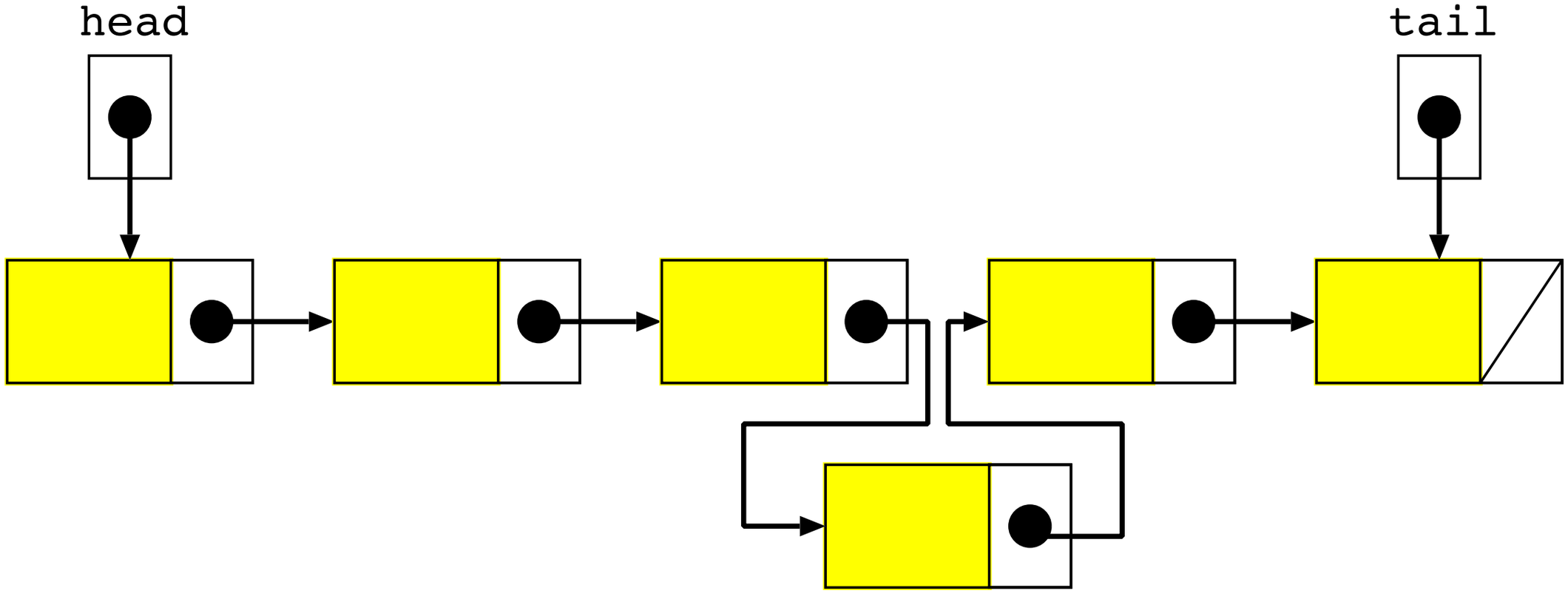
リンクトリスト (連結リスト，linked list)

<応用例>

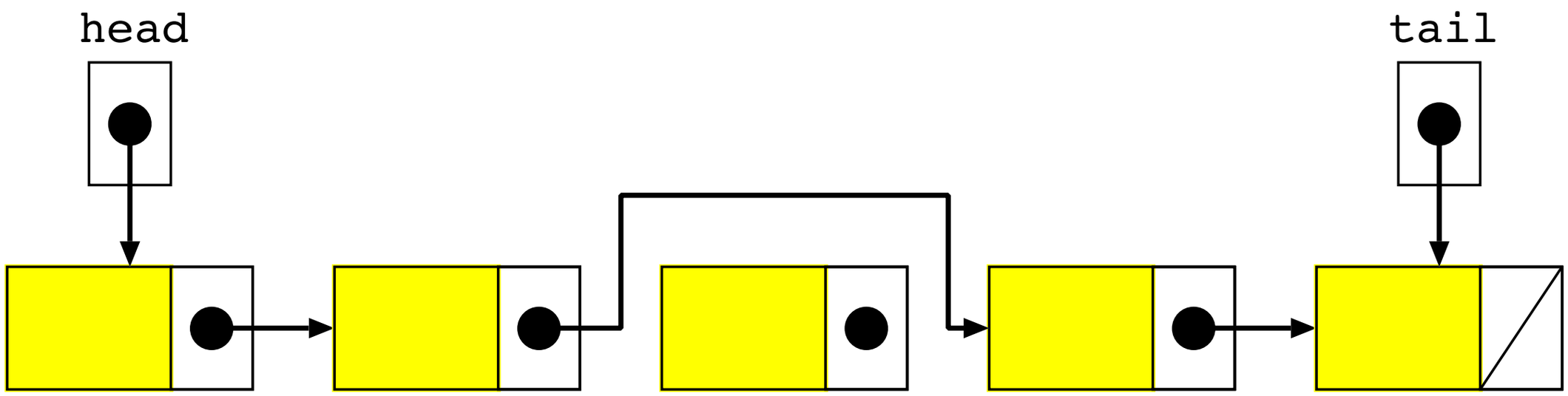
木構造



データ要素の追加



データ要素の削除



llist-func.c (1/4)

```
 1 /* llist-func.c */
 2 #include <stdio.h>
 3 #include <stdlib.h>
 4 #include <string.h>
 5
 6 #define BUFSIZE  1000
 7 #define NAMESIZE 100
 8
 9 struct CELL {
10   char name[NAMESIZE];
11   struct CELL *next;
12 };
13
14 void printdata(struct CELL *);
15 void add_a_data(struct CELL **, struct CELL **);
16 void del_a_data(struct CELL **, struct CELL **);
17
```

```
18 int main(void)
19 {
20   char buf[BUFSIZE];
21   struct CELL *head, *tail;
22
23   head = NULL;
24   tail = NULL;
25
26   while (1) {
27     printf("q) Quit\n");
28     printf("1) Add data\n");
29     printf("2) Delete data\n");
30     printf("3) Show list\n");
31     printf("Input command -> ");
32
33     fgets(buf, BUFSIZE, stdin);
34
35     if (strchr(buf, 'q') != NULL) {
36       break;
37     } else if(strchr(buf, '1') != NULL) {
38       add_a_data(&head, &tail);
39     } else if (strchr(buf, '2') != NULL) {
40       del_a_data(&head, &tail);
41     } else if (strchr(buf, '3') != NULL) {
42       printdata(head);
43     }
44     printf("\n");
45   }
46 }
```

llist-func.c (2/4)

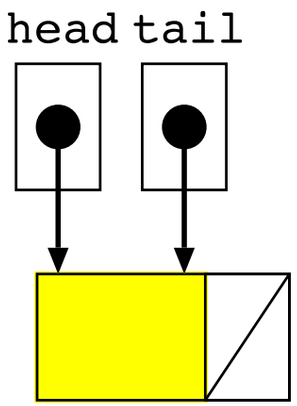
```
47
48 void printdata(struct CELL *h) /* リストの先頭を受け取る */
49 {
50   struct CELL *tmpc;
51
52   printf("------------------------------\n");
53   for (tmpc = h; tmpc != NULL; tmpc = tmpc->next) {
54     printf("%s\n", tmpc->name);
55   }
56   printf("------------------------------\n");
57 }
58
```

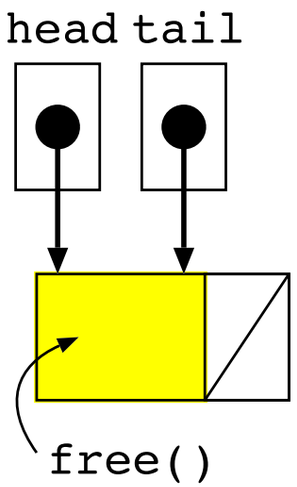
llist-func.c (3/4)

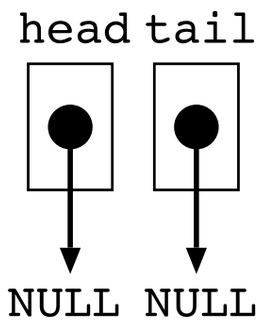
```
85 void del_a_data(struct CELL **h, struct CELL **t) /* 先頭と末尾を受け取る */
86 {
87   struct CELL *tmpc;
88
89   if (*h != NULL) {              /* セルがあることの確認 */
90     if ((*h)->next == NULL) {    /* セルが1つだけなら */
91       *t = NULL;                 /* tailも処理 */
92     }
93     tmpc = (*h)->next;           /* 先頭の次のセルへのポインタを保存 */
94     free(*h);
95     *h = tmpc;                   /* headをつなぎ直す */
96   }
97 }
```

llist-func.c (3/4)

```
85 void del_a_data(struct CELL **h, struct CELL **t) /* 先頭と末尾を受け取る */
86 {
87   struct CELL *tmpc;
88
89   if (*h != NULL) {              /* セルがあることの確認 */
90     if ((*h)->next == NULL) {    /* セルが1つだけなら */
91       *t = NULL;                 /* tailも処理 */
92     }
93     tmpc = (*h)->next;           /* 先頭の次のセルへのポインタを保存 */
94     free(*h);
95     *h = tmpc;                   /* headをつなぎ直す */
96   }
97 }
```





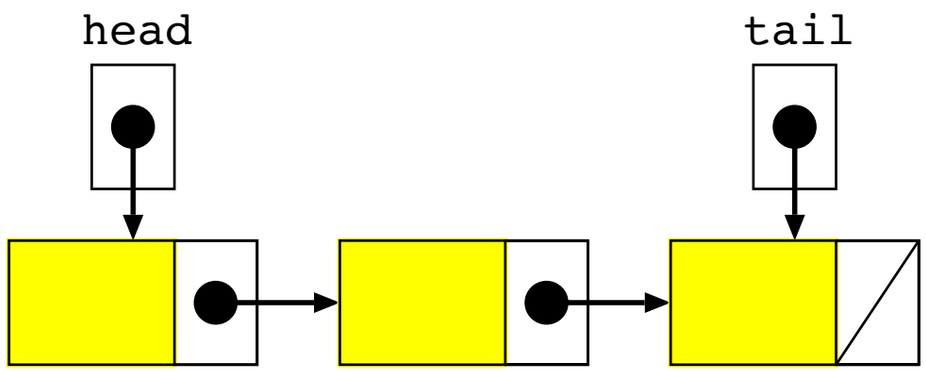


llist-func.c (3/4)

```
85 void del_a_data(struct CELL **h, struct CELL **t) /* 先頭と末尾を受け取る */
86 {
87   struct CELL *tmpc;
88
89   if (*h != NULL) {              /* セルがあることの確認 */
90     if ((*h)->next == NULL) {    /* セルが1つだけなら */
91       *t = NULL;                 /* tailも処理 */
92     }
93     tmpc = (*h)->next;           /* 先頭の次のセルへのポインタを保存 */
94     free(*h);
95     *h = tmpc;                   /* headをつなぎ直す */
96   }
97 }
```

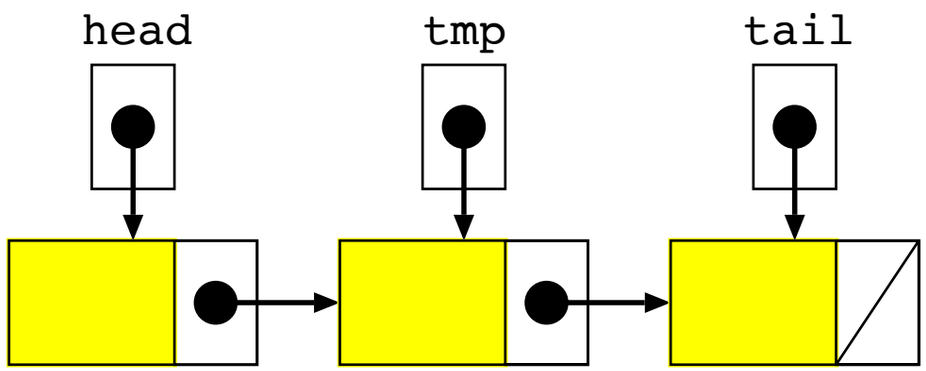
llist-func.c (3/4)

```
85 void del_a_data(struct CELL **h, struct CELL **t) /* 先頭と末尾を受け取る */
86 {
87   struct CELL *tmpc;
88
89   if (*h != NULL) {              /* セルがあることの確認 */
90     if ((*h)->next == NULL) {    /* セルが1つだけなら */
91       *t = NULL;                 /* tailも処理 */
92     }
93     tmpc = (*h)->next;           /* 先頭の次のセルへのポインタを保存 */
94     free(*h);
95     *h = tmpc;                   /* headをつなぎ直す */
96   }
97 }
```



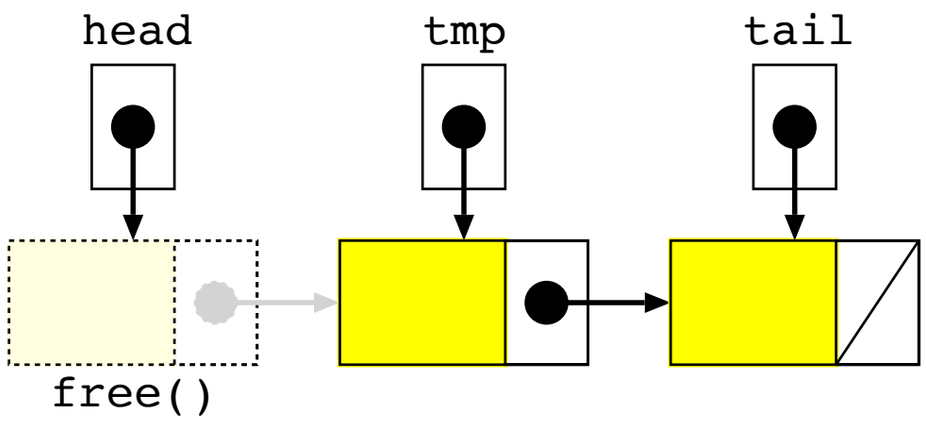
llist-func.c (3/4)

```
85 void del_a_data(struct CELL **h, struct CELL **t) /* 先頭と末尾を受け取る */
86 {
87   struct CELL *tmpc;
88
89   if (*h != NULL) {              /* セルがあることの確認 */
90     if ((*h)->next == NULL) {    /* セルが1つだけなら */
91       *t = NULL;                 /* tailも処理 */
92     }
93     tmpc = (*h)->next;           /* 先頭の次のセルへのポインタを保存 */
94     free(*h);
95     *h = tmpc;                   /* headをつなぎ直す */
96   }
97 }
```



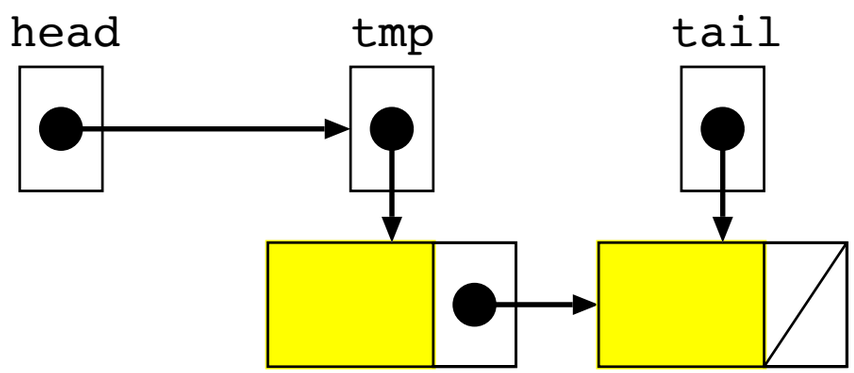
llist-func.c (3/4)

```
85 void del_a_data(struct CELL **h, struct CELL **t) /* 先頭と末尾を受け取る */
86 {
87   struct CELL *tmpc;
88
89   if (*h != NULL) {              /* セルがあることの確認 */
90     if ((*h)->next == NULL) {    /* セルが1つだけなら */
91       *t = NULL;                 /* tailも処理 */
92     }
93     tmpc = (*h)->next;           /* 先頭の次のセルへのポインタを保存 */
94     free(*h);
95     *h = tmpc;                   /* headをつなぎ直す */
96   }
97 }
```



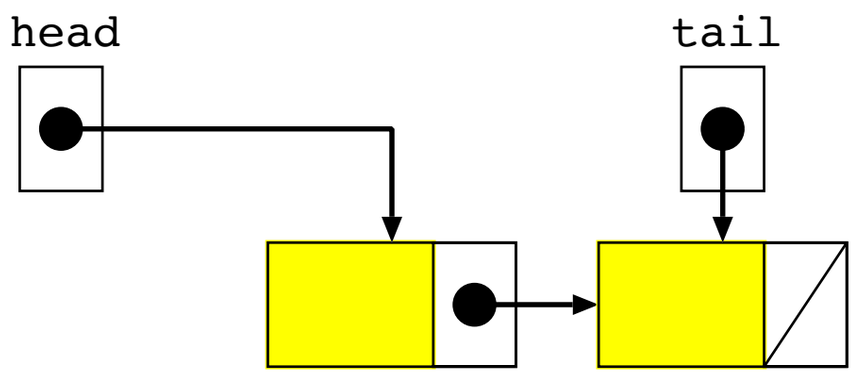
llist-func.c (3/4)

```
85 void del_a_data(struct CELL **h, struct CELL **t) /* 先頭と末尾を受け取る */
86 {
87   struct CELL *tmpc;
88
89   if (*h != NULL) {              /* セルがあることの確認 */
90     if ((*h)->next == NULL) {    /* セルが1つだけなら */
91       *t = NULL;                 /* tailも処理 */
92     }
93     tmpc = (*h)->next;           /* 先頭の次のセルへのポインタを保存 */
94     free(*h);
95     *h = tmpc;                   /* headをつなぎ直す */
96   }
97 }
```



llist-func.c (3/4)

```
85 void del_a_data(struct CELL **h, struct CELL **t) /* 先頭と末尾を受け取る */
86 {
87   struct CELL *tmpc;
88
89   if (*h != NULL) {              /* セルがあることの確認 */
90     if ((*h)->next == NULL) {    /* セルが1つだけなら */
91       *t = NULL;                 /* tailも処理 */
92     }
93     tmpc = (*h)->next;           /* 先頭の次のセルへのポインタを保存 */
94     free(*h);
95     *h = tmpc;                   /* headをつなぎ直す */
96   }
97 }
```

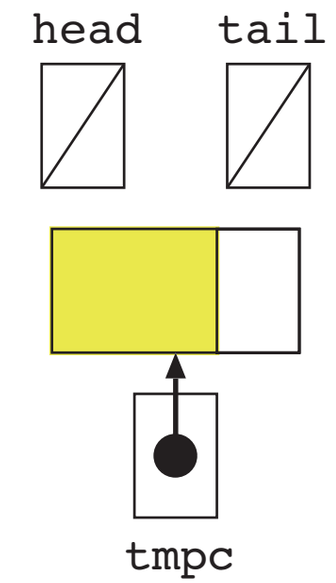


llist-func.c (4/4)

```
59 void add_a_data(struct CELL **h, struct CELL **t)
60 {
61   char buf[BUFSIZE];
62   struct CELL *tmpc;
63
64   printf("Input name -> ");
65   fgets(buf, BUFSIZE, stdin);
66   buf[strlen(buf)-1] = '\0';
67
68   tmpc = (struct CELL *)malloc(sizeof(struct CELL));
69   if (tmpc == NULL) {
70     printf("memory allocation error!!\n");
71     exit(EXIT_FAILURE);
72   }
73
74   strcpy(tmpc->name, buf);
75
76   if (*h == NULL) {
77     *h = tmpc;
78     (*h)->next = NULL;
79   } else {
80     (*t)->next = tmpc;
81   }
82   (*t) = tmpc;
83   (*t)->next = NULL;
84 }
```

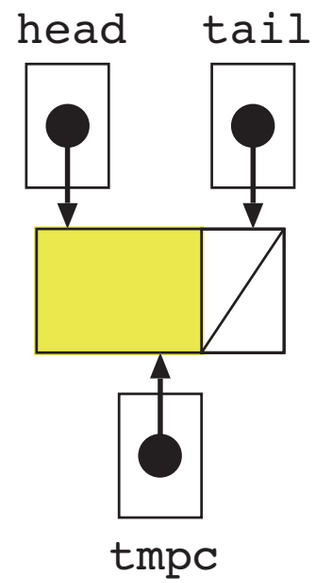
llist-func.c (4/4)

最初の要素の追加



headとtailは
NULLで初期化されている.

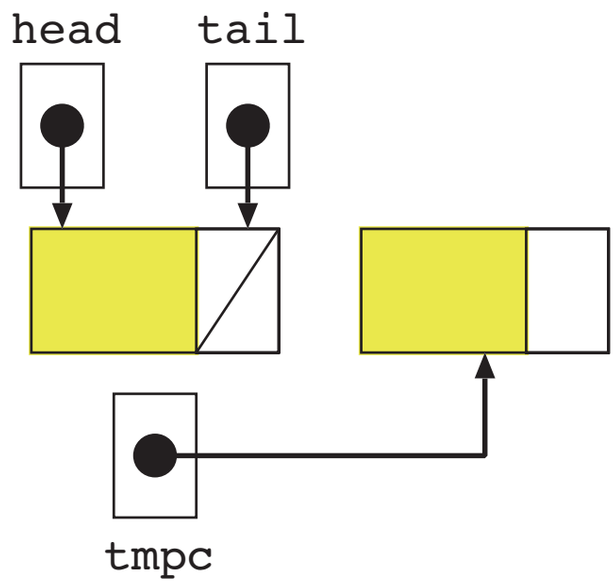
malloc()によってセルを確保し,
それをtmpcが指すようにする.



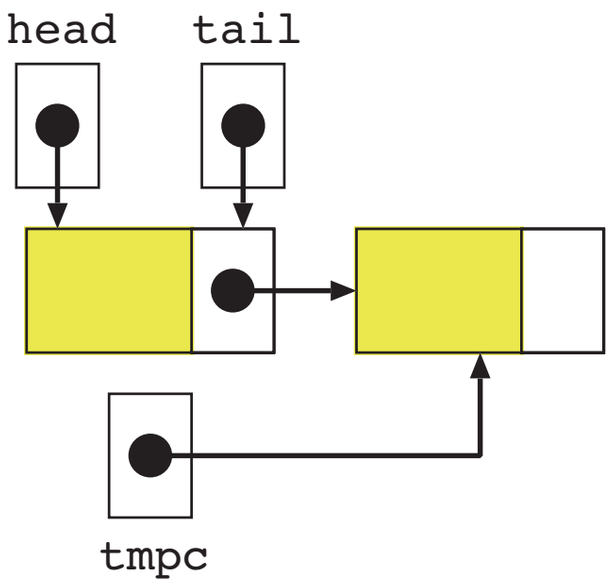
tmpcが指しているセルを
headとtailが指すようにする.
)

tailが指しているセルのnextに
NULLを代入する.

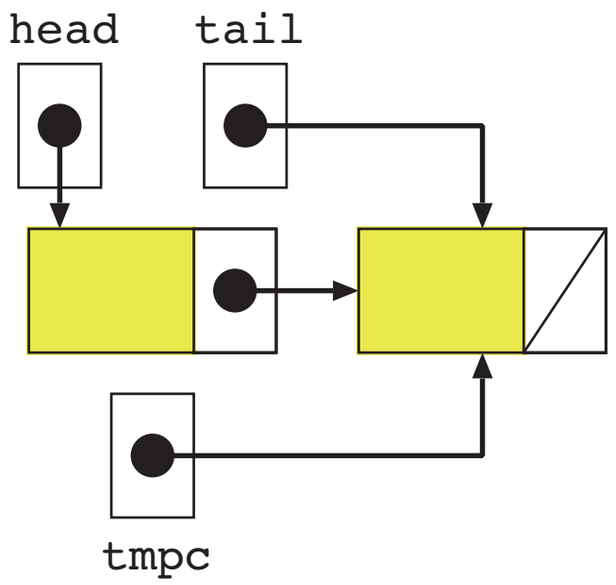
次の要素の追加



malloc()によってセルを確保し,
それをtmpcが指すようにする.



tailが指すセルのnextが,
tmpcが指すセルを指すようにする.



tmpcの指すセルをtailが指すようにし,
tailが指すセルのnextに
NULLを代入する.

おわり

筑波大学情報メディア創成学類 三河 正彦