リストⅢ

０．目次

２．基本的な操作

２．１　リストから要素の削除

まず、一般的な処理を書き、つぎに、特別な処理を書く。

一般的な処理は、

処理１：リスト中に、削除するデータを見つけ、削除する場合への対応。

特別な処理は、

処理２：先頭のデータを削除する場合への対応。

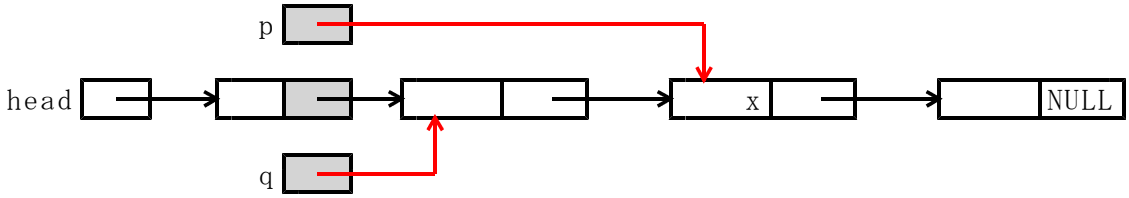
になる。

●処理１

（１）削除前

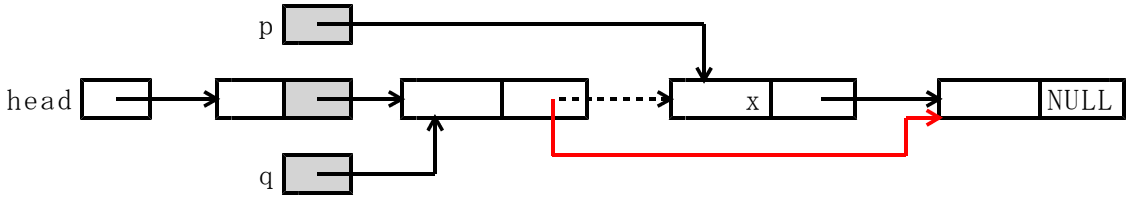
変数x　　：削除データ

（２）削除データxを見つける。



変数p：削除データを指すポインタ変数
変数q：削除データの直前を指すポインタ変数

（３）削除データxを削除するために、ポインタを変更する。

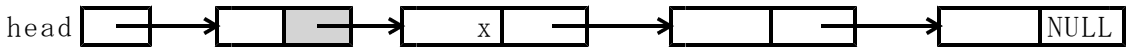



```
q->next = p->next;
```

（４）削除後

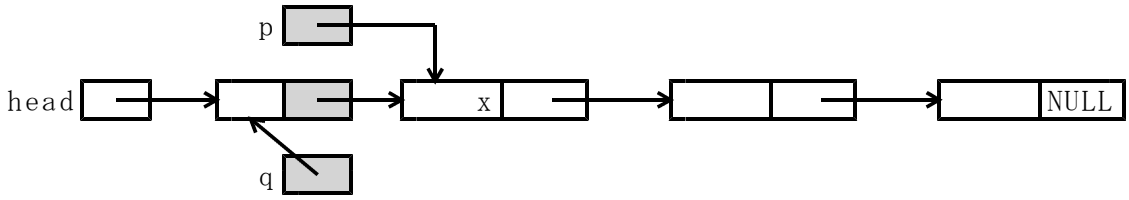
●処理２

（１）削除前



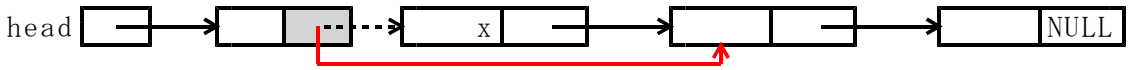
変数x　　：削除データ

（２）削除データxを見つける。



変数p：削除データを指すポインタ変数
変数q：削除データの直前を指すポインタ変数

（３）削除後




```
q->next = p->next;
```

処理１,処理２をまとめて、次のプログラムを得る。

●プログラム（ls211.c）

```
1  /* << ls211.c >> */
2  /* リストからデータの削除。*/
3  #include <stdio.h>
4  #include <stdlib.h>
5  struct NODE {
6    int info;
7    struct NODE *next;
8  };
9  struct NODE *mlist();
10 void slist(struct NODE *p);
11
12 int main() {
13   int x;
14   struct NODE *head,   /* リストの先頭を指すポインタ変数。*/
15               *p,      /* 削除データを指すポインタ変数。*/
16               *q;      /* 削除データの直前を指すポインタ変数。*/
17
18   /* リストの作成。*/
19   head = mlist();
20
21   while( 1 ) {
22     /* リストの表示。*/
23     printf("現在のリスト：");
24     slist(head);
25
26     /* 削除データxの読込。*/
27     scanf("%d",&x);
28     if( x == 0 ) { break; }
29     printf("削除データ：%d ¥n",x);
30
31     /* 削除。*/
32
33     /* 初期設定。*/
34     p = head->next;
35     q = head;
36
37     /* 削除位置を見つける。*/
38     while( p != NULL ) {
39       if( p->info == x ) { break; }
40       q = p;
41       p = p->next;
42     }
43
44     /* 削除データが見つかった場合。*/
45     if( p != NULL ) {
46       q->next = p->next;
```

```
47      }
48    }
49  }
```

実行結果

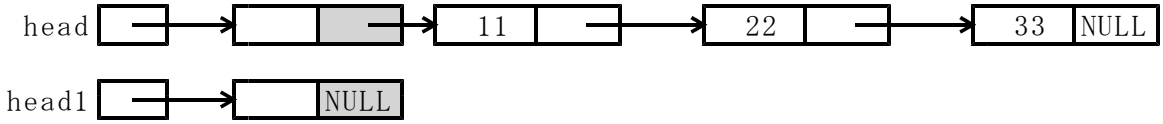
```
% cc ls211.c
% ./a.out
11 22 22 33 0
現在のリスト：11 22 22 33

22
削除データ：22
現在のリスト：11 22 33

22
削除データ：22
現在のリスト：11 33

11
削除データ：11
現在のリスト：33

33
削除データ：33
現在のリスト：
0
```

２．２　リストの複写

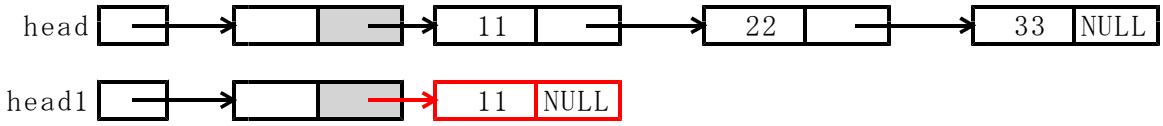
●考え方

元のリストを先頭から末尾へたどりながら、新たなリストを作成していく。

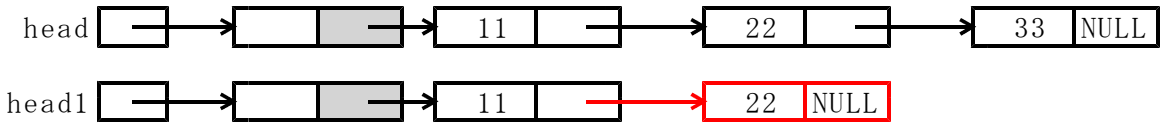
（１）複写前



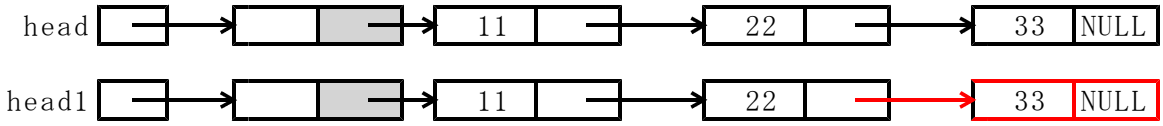
（２）複写中



（３）複写中

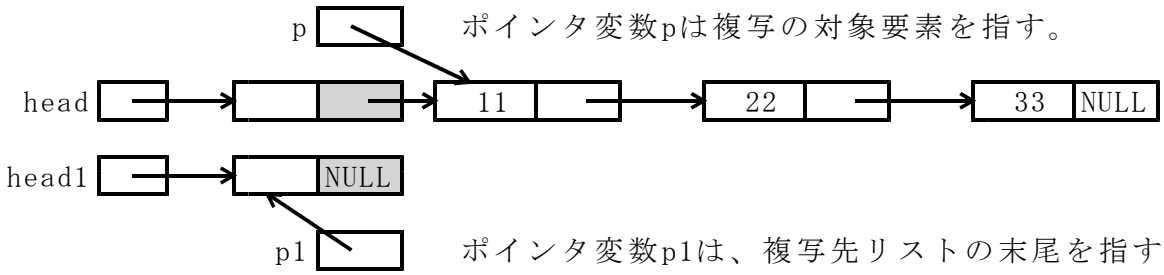


（４）複写後

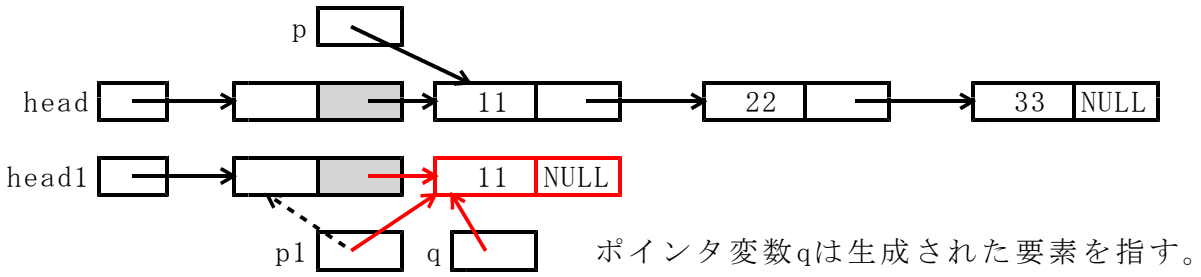


●ひとつずつ要素を複写

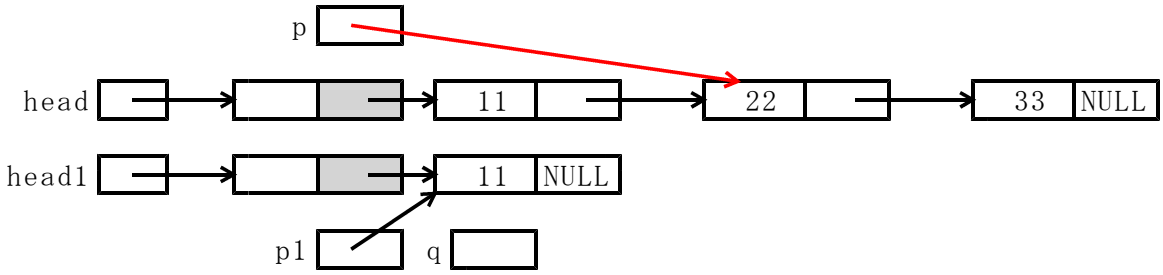
（１）要素11の複写前



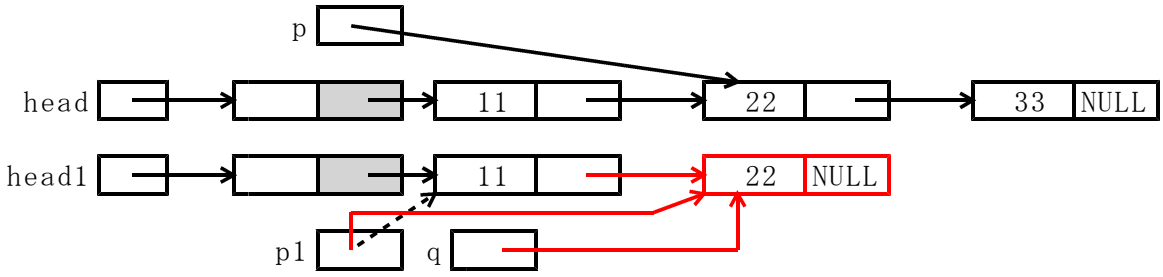
（２）要素11の複写後



（３）要素22の複写前



（４）要素22の複写後



●プログラム（ls221.c）

```
1  /* << ls221.c >> */
2  /* リストの複写。 */
3  #include <stdio.h>
4  #include <stdlib.h>
5  struct NODE {
6    int info;
7    struct NODE *next;
8  };
9  struct NODE *mlist();
10 void slist(struct NODE *p);
11
12 int main() {
13   struct NODE *head,   /* 複写元リストの先頭を指すポインタ変数。*/
14               *p,      /* 複写元の対象要素を指すポインタ変数。*/
15               *q,      /* データを指すポインタ変数。*/
16               *head1,  /* 複写先リストの先頭を指すポインタ変数。*/
17               *t;     /* 複写先リストの末尾を指すポインタ変数。*/
18
19   /* 複写元リストの作成と表示。*/
20   head = mlist();
21   printf("複写元リスト：");
22   slist(head);
23
24   /* 複写。*/
25   /* 複写先リストヘッドの作成。*/
26   head1 = (struct NODE *)malloc(sizeof(struct NODE));
27   head1->next = NULL;
28
29   p = head->next; /* 複写元の対象要素を指す。*/
30   t = head1;      /* 複写先リストの末尾を指す。*/
31
32   while( p != NULL ) {
33     /* 複写先のデータを作成。*/
34     q = (struct NODE *)malloc(sizeof(struct NODE));
35     q->info = p->info;
36     q->next = NULL;
37
38     /* 複写先のリストを作成。*/
39     t->next = q;
40     t = q;
41     p = p->next;
42   }
43
44   /* 複写リストの表示。*/
45   printf("複写先リスト：");
46   slist(head1);
47 }
```

実行結果

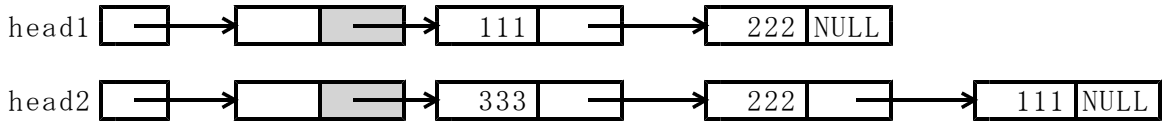
```
% cc ls221.c
% ./a.out
0
複写元リスト：
複写先リスト：

% ./a.out
11 0
複写元リスト：11
複写先リスト：11

% ./a.out
11 22 0
複写元リスト：11 22
複写先リスト：11 22

% ./a.out
11 22 33 44 55 0
複写元リスト：11 22 33 44 55
複写先リスト：11 22 33 44 55
```

２．３　リストの連結

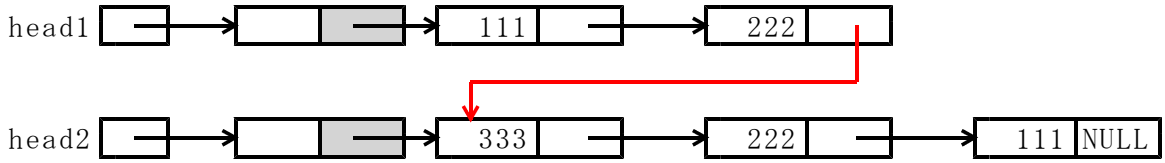
●考え方

一方のリストの末尾と他方のリストの先頭をつなぐ。

（１）連結前



（２）連結後



●プログラム（ls231.c）

```
/* << ls231.c >> */
/* ２つのリストの連結。
   リスト１の末尾とリスト２の先頭をつなげる。*/
#include <stdio.h>
#include <stdlib.h>
struct NODE {
  int info;
  struct NODE *next;
};
struct NODE *mlist();
void slist(struct NODE *p);

int main() {
  struct NODE *head1, /* リスト１の先頭を指すポインタ変数。*/
              *head2, /* リスト２の先頭を指すポインタ変数。*/
              *p,     /* リスト１をたどる作業用ポインタ変数。*/
              *q;     /* リスト１の末尾を指すポインタ変数。*/

  /* リスト１の作成と表示。*/
  head1 = mlist();
  printf("リスト１：");
  slist(head1);

  /* リスト２の作成と表示。*/
  head2 = mlist();
  printf("リスト２：");
  slist(head2);

```

```
29    /* ２つのリストの連結。*/
30    /* 初期設定。*/
31    p = head1->next;
32    q = head1;
33
34    /* リスト１をたどり、末尾の番地をポインタ変数qに保存する。*/
35    while( p != NULL ) {
36      q = p;
37      p = p->next;
38    }
39
40    /* リスト１にリスト２を連結する。*/
41    q->next = head2->next;
42
43    /* 連結リストの表示。*/
44    printf("リスト１＋リスト２：");
45    slist(head1);
46  }
```

実行結果

```
% cc ls231.c
% ./a.out
0
リスト１：
0
リスト２：
リスト１＋リスト２：

% ./a.out
11 22 0
リスト１：11 22
0
リスト２：
リスト１＋リスト２：11 22

% ./a.out
0
リスト１：
11 22 0
リスト２：11 22
リスト１＋リスト２：11 22

% ./a.out
11 22 33 0
リスト１：11 22 33
44 55 66 77 0
リスト２：44 55 66 77
リスト１＋リスト２：11 22 33 44 55 66 77
```

２．４　問題

問題１　リストから指定されたすべての要素を削除するプログラムを作成せよ。

●プログラム（ls241.c）

```
/* << ls241.c >> */
/* リストからデータの削除。*/
#include <stdio.h>
#include <stdlib.h>
struct NODE {
  int info;
  struct NODE *next;
};
struct NODE *mlist();
void slist(struct NODE *p);

int main() {
  int x;
  struct NODE *head, /* リストの先頭を指すポインタ変数。*/
              *p,    /* 削除データを指すポインタ変数。*/
              *q;    /* 削除データの直前を指すポインタ変数。*/

  /* リストの作成。*/
  head = mlist();

  while( 1 ) {
    /* リストの表示。*/
    printf("現在のリスト：");
    slist(head);

    /* 削除データxの読込。*/
    scanf("%d",&x);
    if( x == 0 ) { break; }
    printf("削除データ：%d ¥n",x);

    /* 削除データを見つけながら削除する。*/
    p = head->next;
    q = head;
    while( p != ①      ) {
      if( p->info == x ) {
        q->next = ②          ;
      } else {
        q = ③    ;
      }
      p = ④          ;
    }
  }
}
```

実行結果

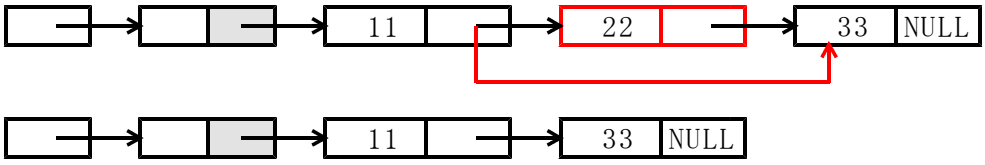
```
% cc ls241.c
% ./a.out
11 22 33 11 22 11 0
現在のリスト：11 22 33 11 22 11

33
削除データ：33
現在のリスト：11 22 11 22 11

22
削除データ：22
現在のリスト：11 11 11

11
削除データ：11
現在のリスト：

0
```

問題２　リストから指定されたk番目の要素を削除するプログラムを作成せよ。
ただし、kは、リストの先頭からk番目を意味する。
また、kが現在のリストの長さより大きい場合、何もしない。

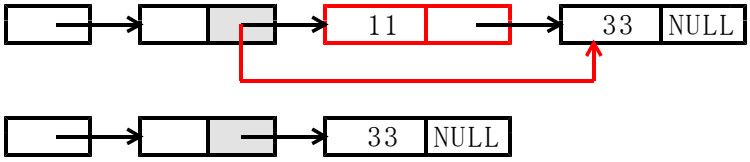
●考え方

（１）元のリスト

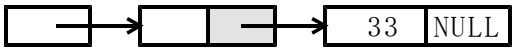
（２）2番目の要素を見つけ削除する。



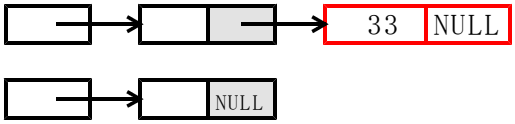
（３）1番目の要素を見つけ削除する。



（４）3番目の要素を見つけ削除する。



（５）1番目の要素を見つけ削除する。



●プログラム（ls242.c）

```
1  /* << ls242.c >> */
2  /* リストからデータの削除。*/
3  #include <stdio.h>
4  #include <stdlib.h>
5  struct NODE {
6    int info;
7    struct NODE *next;
8  };
9  struct NODE *mlist();
10 void slist(struct NODE *p);
11
12 int main() {
13   int count, /* リスト中の要素の個数。*/
14       k;
```

```
15    struct NODE *head, /* リストの先頭を指すポインタ変数。*/
16                *p,    /* 削除データを指すポインタ変数。*/
17                *q;    /* 削除データの直前を指すポインタ変数。*/
18
19    /* リストの作成。*/
20    head = mlist();
21
22    while( 1 ) {
23      /* リストの表示。*/
24      printf("現在のリスト：");
25      slist(head);
26
27      /* 削除位置kの読込。*/
28      scanf("%d",&k);
29      if( k <= 0 ) { break; }
30      printf("削除位置：%d ¥n",k);
31
32      /* 削除。*/
33
34      /* 初期設定。*/
35      p = head->next;
36      q = head;
37      count = 0;
38
39      /* 削除位置を見つけ削除する。*/
40      while( p != ⑤          ) {
41        count++;
42        if( k == count ) {
43          q->next = ⑥          ;
44          break;
45        }
46        q = ⑦      ;
47        p = ⑧          ;
48      }
49
50      /* リストの表示。*/
51      printf("リスト：");
52      slist(head);
53    }
54  }
```

実行結果

```
% cc ls242.c
% ./a.out
11 22 33 44 55 0
現在のリスト：11 22 33 44 55

3
削除位置：3
リスト：11 22 44 55
現在のリスト：11 22 44 55

1
削除位置：1
リスト：22 44 55
現在のリスト：22 44 55

5
削除位置：5
リスト：22 44 55
現在のリスト：22 44 55

3
削除位置：3
リスト：22 44
現在のリスト：22 44

0
```