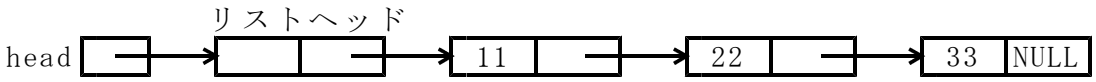
リストⅡ

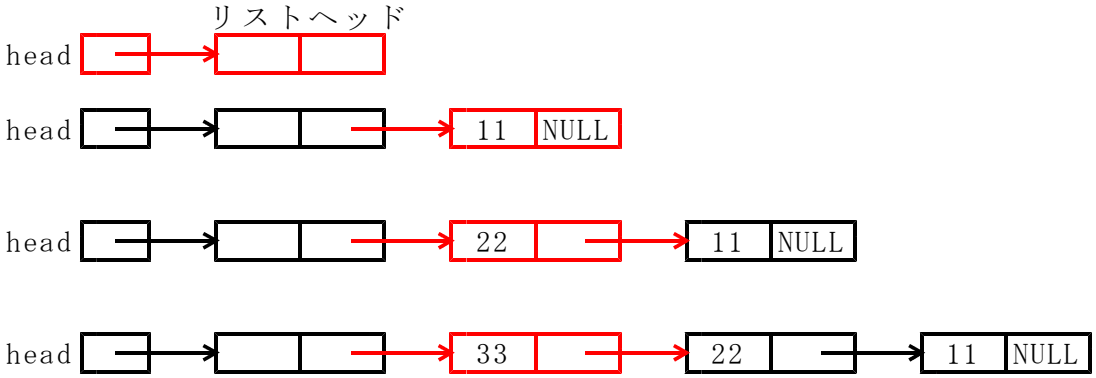
０．目次

１．３　リストの作成(リストヘッドの利用)

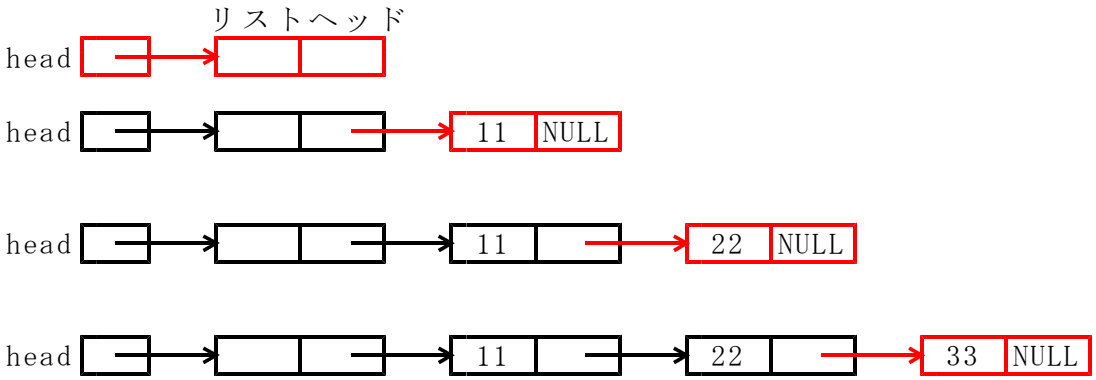
リストの先頭を指すポインタをもつデータ(構造体)を用いると、空リストの処理と空リストでないときの処理を分けて扱う必要がなくなり、処理が簡潔になる。これをリストヘッドと呼ぶ。headは、リストの先頭を指すポインタ変数。



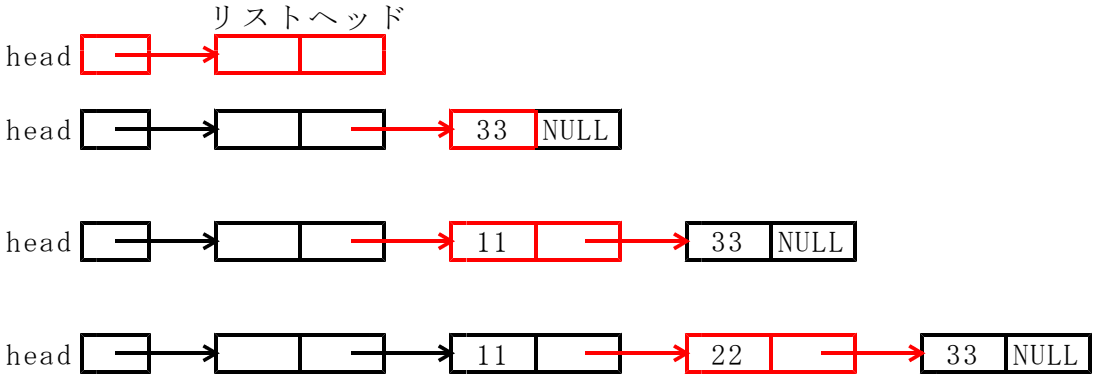
●作成法１：データ(11,22,33)をリストの先頭に追加していく。



●作成法２：データ(11,22,33)をリストの末尾に追加していく。



●作成法３：データ(33,11,22)を昇順を保存しながらリストに追加していく。



１．３．１　リストの先頭に追加する方法

リストの先頭を指すポインタ変数headと新たなデータを指すポインタ変数rが必要である。

一般的な処理は、

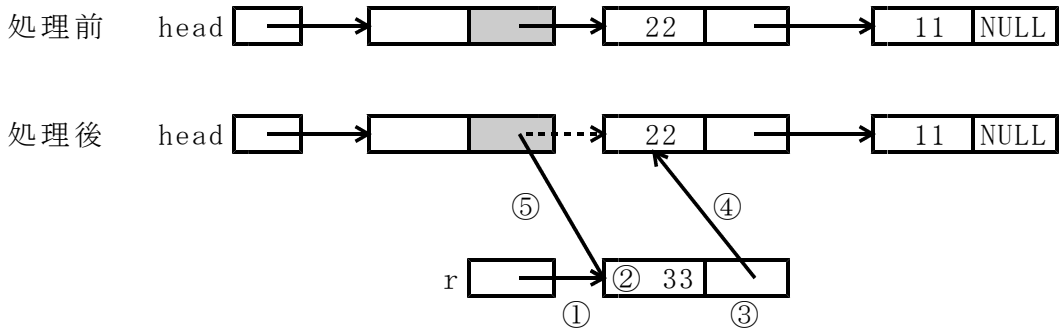
処理１：既存のリストの先頭に、新たなデータを追加する場合への対応。

になる。特別な処理は、

処理２：空リストに、最初のデータを追加する場合への対応。

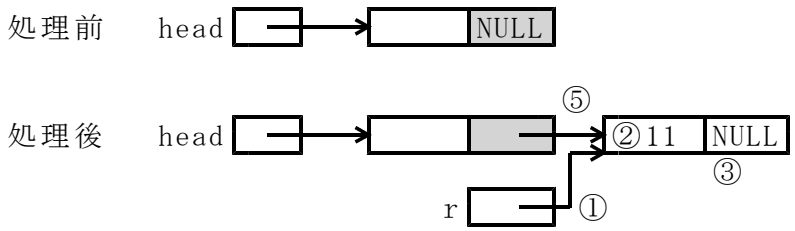
になる。前者と後者を組み合わせると、簡潔なプログラムになる。

●処理１




```
   /* データの読込み。*/
   scanf("%d",&data);
   /* ポインタ変数の設定。*/
①  r = (struct NODE *)malloc(sizeof(struct NODE));
   /* リストへの追加。*/
②  r->info = data;
③  r->next = NULL;
④  r->next = head->next;
⑤  head->next = r;
```

●処理２




```
   scanf("%d",&data);
①  r = (struct NODE *)malloc(sizeof(struct NODE));
②  r->info = data;
③  r->next = NULL;
⑤  head->next = r;
```

処理１で実行される④は、処理２においては、head->next = NULL なので、③と同じになり、実質的意味はなくなる。この結果、次のプログラムを得る。

●プログラム（ls131.c）

```
1  /* << ls131.c >> */
2  /* リストの作成と表示。*/
3  #include <stdio.h>
4  #include <stdlib.h>
5  struct NODE {
6    int info;
7    struct NODE *next; /* nextはポインタ変数。*/
8  };
9
10 int main() {
11   struct NODE *head, /* リストの先頭を指すポインタ変数。*/
12               *p,    /* 作業用ポインタ変数。*/
13               *r;    /* 新たなデータを指すポインタ変数。*/
14   int data;          /* データ。*/
15
16   /* リストの作成。*/
17
18   /* リストヘッドの作成。*/
19   head = (struct NODE *)malloc(sizeof(struct NODE));
20   head->next = NULL;
21
23   while( 1 ) {
24     /* データの読み込み。*/
25     scanf("%d",&data); if( data <= 0 ) { break; }
26
27     /* データの追加。*/
28     r = (struct NODE *)malloc(sizeof(struct NODE));      /*①*/
29     r->info = data;                                      /*②*/
30     r->next = NULL;                                      /*③*/
31
32     r->next = head->next;                                /*④*/
33     head->next = r;                                      /*⑤*/
34   }
35
36   /* リストの表示。*/
37   printf("リスト：");
38   p = head->next;
39   while( p != NULL ) {
40     printf("%d ",p->info);
41     p = p->next;
42   }
43   printf("¥n");
44 }
```

実行結果

```
% cc ls131.c
% ./a.out
0
リスト：

% ./a.out
11 0
リスト：11

% ./a.out
11 22 0
リスト：22 11

% ./a.out
11 22 33 44 55 0
リスト：55 44 33 22 11
```

（考察）データ構造を少し変えるだけでプログラムが簡潔に書ける。
分岐（if文）が無くなっていることに着目。

１．３．２　リストの末尾に追加する方法

リストの先頭を指すポインタ変数headと末尾を指すポインタ変数tと新たなデータを指すポインタ変数rが必要である。

一般的な処理は、

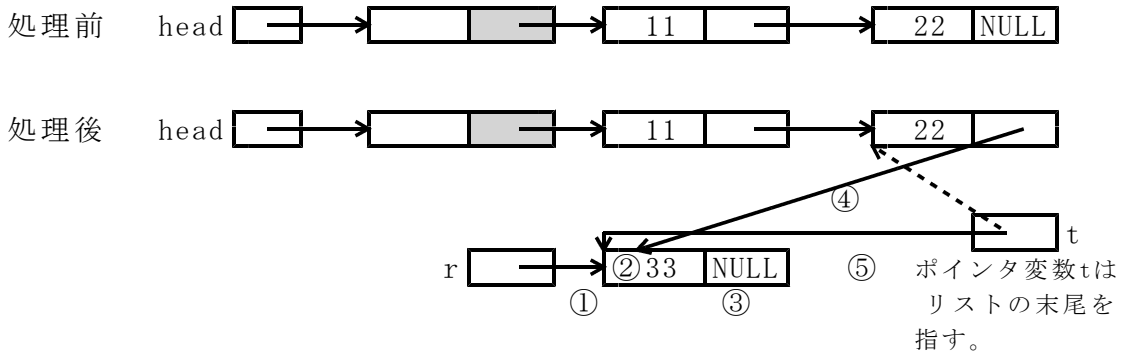
処理１：既存のリストの末尾に、新たなデータを追加する場合への対応。

になる。特別な処理は、

処理２：空リストに、最初のデータを追加する場合への対応。

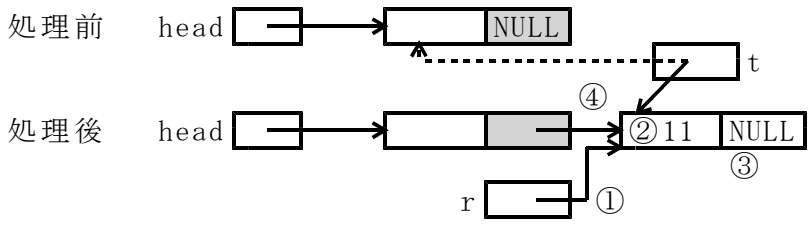
になる。前者と後者を組み合わせると、簡潔なプログラムになる。

●処理１




```
   /* データの読込み。*/
   scanf("%d",&data);
   /* ポインタ変数の設定。*/
①  r = (struct NODE *)malloc(sizeof(struct NODE));
   /* リストへの追加。*/
②  r->info = data;
③  r->next = NULL;
④  t->next = r; /* ポインタ変数tはリストの末尾を指す。*/
⑤  t = r;
```

●処理２




```
   scanf("%d",&data);
①  r = (struct NODE *)malloc(sizeof(struct NODE));
②  r->info = data;
③  r->next = NULL;
④  t->next = r; /* ポインタ変数tはリストの末尾を指す。*/
⑤  t = r;
```

処理１と処理２は全く同じになる。この結果、次のプログラムを得る。

●プログラム（ls132.c）

```
1  /* << ls132.c >> */
2  /* リストの作成と表示。*/
3  #include <stdio.h>
4  #include <stdlib.h>
5  struct NODE {
6    int info;
7    struct NODE *next; /* nextはポインタ変数。*/
8  };
9
10 int main() {
11   struct NODE *head, /* リストの先頭を指すポインタ変数。*/
12               *p,    /* 作業用ポインタ変数。*/
13               *r,    /* 新たなデータを指すポインタ変数。*/
14               *t;    /* リストの末尾を指すポインタ変数。*/
15   int data;          /* データ。*/
16
17   /* リストの作成。*/
18
19   /* リストヘッドの作成。*/
20   head = (struct NODE *)malloc(sizeof(struct NODE));
21   head->next = NULL;
22
23   /* 初期設定。*/
24   t = head;
25
26   while( 1 ) {
27     /* データの読み込み。*/
28     scanf("%d",&data); if( data <= 0 ) { break; }
29
30     /* データの追加。*/
31     r = (struct NODE *)malloc(sizeof(struct NODE));   /*①*/
32     r->info = data;                                   /*②*/
33     r->next = NULL;                                   /*③*/
34
35     t->next = r;                                      /*④*/
36     t = r;                                            /*⑤*/
37   }
38
39   /* リストの表示。*/
40   printf("リスト：");
41   p = head->next;
42   while( p != NULL ) {
43     printf("%d ",p->info);
44     p = p->next;
45   }
46   printf("¥n");
47 }
```

実行結果

```
% cc ls132.c
% ./a.out
0
リスト：

% ./a.out
11 0
リスト：11

% ./a.out
11 22 0
リスト：11 22

% ./a.out
11 22 33 44 55 0
リスト：11 22 33 44 55
```

１．３．３　昇順を保存しながらリストに追加する方法

リストの先頭を指すポインタ変数headと新たなデータを指すポインタ変数rに加えて、挿入位置を探し、挿入位置の直前の要素を指すポインタ変数qと、挿入位置の直後の要素を指すポインタ変数pが必要である。

一般的な処理は、

処理１：既存のリストに、昇順を保存しながら新たなデータを追加する場合への対応。

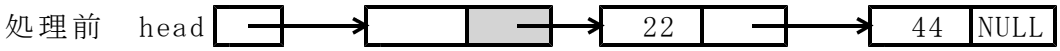
になる。特別な処理は、

処理２：空リストに、最初のデータを追加する場合への対応。

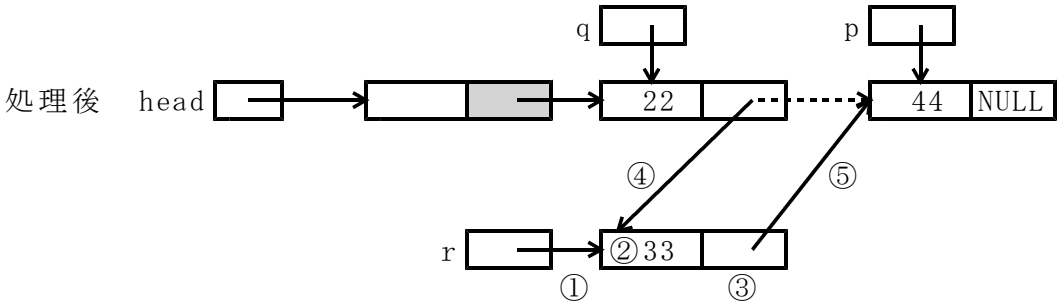
になる。前者と後者を組み合わせると、簡潔なプログラムになる。

●処理１

・リストの途中に追加する場合



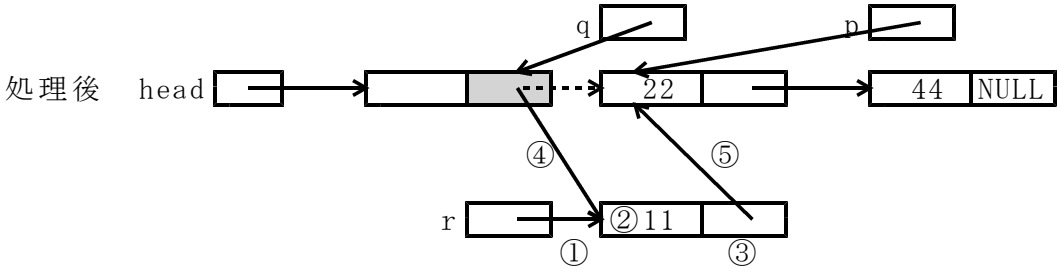
ポインタ変数qは、挿入位置の直前の要素を指す。
ポインタ変数pは、挿入位置の直後の要素を指す。




```
   /* データの読込み。*/
   scanf("%d",&data);
   /* ポインタ変数の設定。*/
①  r = (struct NODE *)malloc(sizeof(struct NODE));
   /* リストへの追加。*/
②  r->info = data;
③  r->next = NULL;
   /* 挿入位置を探す。*/
   /* ポインタ変数pは、挿入位置の直後の要素を指し、
      ポインタ変数qは、直前の要素を指す。*/
④  q->next = r;
⑤  r->next = p;
```

・リストの先頭に追加する場合

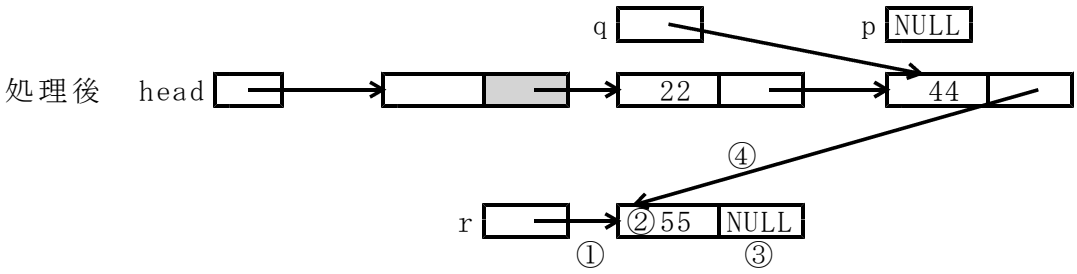
ポインタ変数qは、挿入位置の直前の要素を指す。
ポインタ変数pは、挿入位置の直後の要素を指す。




```
   /* データの読込み。*/
   scanf("%d",&data);
   /* ポインタ変数の設定。*/
①  r = (struct NODE *)malloc(sizeof(struct NODE));
   /* リストへの追加。*/
②  r->info = data;
③  r->next = NULL;
   /* 挿入位置を探す。*/
   /* ポインタ変数pは、挿入位置の直後、
      ポインタ変数qは、直前の位置を指す。*/
④  q->next = r;
⑤  r->next = p;
```

・リストの末尾に追加する場合

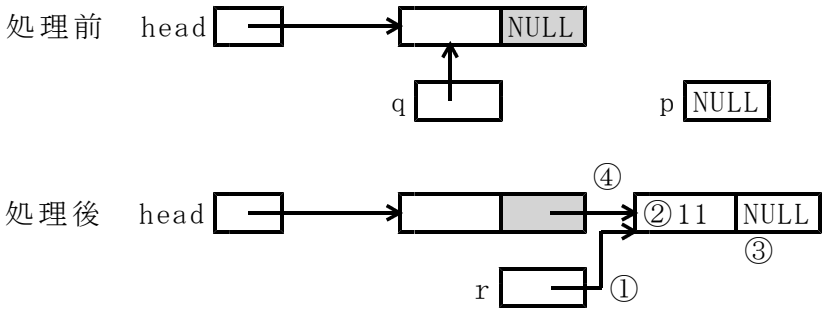
ポインタ変数qは、挿入位置の直前の要素を指す。
ポインタ変数pは、挿入位置の直後の要素を指す。




```
   /* データの読込み。*/
   scanf("%d",&data);
   /* ポインタ変数の設定。*/
①  r = (struct NODE *)malloc(sizeof(struct NODE));
   /* リストへの追加。*/
②  r->info = data;
③  r->next = NULL;
   /* 挿入位置を探す。*/
   /* ポインタ変数pは、挿入位置の直後、
      ポインタ変数qは、直前の位置を指す。*/
④  q->next = r;
```

●処理２

・空リストに追加する場合




```
   scanf("%d",&data);
①  r = (struct NODE *)malloc(sizeof(struct NODE));
②  r->info = data;
③  r->next = NULL;
④  q->next = r;
```

①②③④は共通である。⑤を必要としない処理では、p=NULLとなっており、実質的な意味はない。この結果、次のプログラムを得る。

●プログラム（ls133.c）

```
1  /* << ls133.c >> */
2  /* リストの作成と表示。*/
3  #include <stdio.h>
4  #include <stdlib.h>
5  struct NODE {
6    int info;
7    struct NODE *next; /* nextはポインタ変数。*/
8  };
9
10 int main() {
11   struct NODE *head, /* リストの先頭を指すポインタ変数。*/
12               *p,    /* 挿入位置の直後の要素を指すポインタ変数。*/
13               *q,    /* 挿入位置の直前の要素を指すポインタ変数。*/
14               *r,    /* 新たなデータを指すポインタ変数。*/
15               *s;    /* 作業用ポインタ変数。*/
16   int data;          /* データ。*/
17
18   /* リストの作成。*/
19
20   /* リストヘッドの作成。*/
21   head = (struct NODE *)malloc(sizeof(struct NODE));
22   head->next = NULL;
23
24   while( 1 ) {
25     /* データの読み込み。*/
26     scanf("%d",&data); if( data <= 0 ) { break; }
27
28     /* データの追加。*/
29     r = (struct NODE *)malloc(sizeof(struct NODE));     /*①*/
30     r->info = data;                                     /*②*/
31     r->next = NULL;                                     /*③*/
32
33     /* 挿入する位置を探索。*/
34     p = head->next; q = head;
35     while( p != NULL ) {
36       if( p->info > r->info ) { break; }
37       q = p; p = p->next;
38     }
39     q->next = r;                                        /*④*/
40     r->next = p;                                        /*⑤*/
41   }
42
43   /* リストの表示。*/
44   printf("リスト：");
45   s = head->next;
```

```
46    while( s != NULL ) {
47      printf("%d ",s->info);
48      s = s->next;
49    }
50    printf("¥n");
51  }
```

実行結果

```
% cc ls133.c
% ./a.out
0
リスト：

% ./a.out
11 0
リスト：11

% ./a.out
11 22 0
リスト：11 22

% ./a.out
22 11 0
リスト：11 22

% ./a.out
22 11 44 33 0
リスト：11 22 33 44

% ./a.out
55 11 88 33 66 99 77 22 44 0
リスト：11 22 33 44 55 66 77 88 99
```

リストの作成と表示は、今後よく使うので関数化しておく。

```
/* リストの作成法２。make list */
struct NODE *mlist() {
  int data;
  struct NODE *head,*r,*t;

  /* リストヘッドの作成。*/
  head = (struct NODE *)malloc(sizeof(struct NODE));
  head->next = NULL;

  /* リストの作成。*/
  t = head;
  while( 1 ) {
    scanf("%d",&data); if( data <= 0 ) { break; }
    r = (struct NODE *)malloc(sizeof(struct NODE));
    r->info = data; r->next = NULL;
    t->next = r;
    t = r;
  }
  return head;
}
```

```
/* リストの表示。show list */
void slist(struct NODE *head) {
  struct NODE *p;
  p = head->next;
  while( p != NULL ) {
    printf("%d ",p->info);
    p = p->next;
  }
  printf("¥n");
}
```

１．４　問題

問題１

リスト作成後、リスト中のデータの個数を出力するプログラムを完成せよ。

●プログラム（ls141.c）

```
1  /* << ls141.c >> */
2  #include <stdio.h>
3  #include <stdlib.h>
4  struct NODE {
5    int info;
6    struct NODE *next; /* nextはポインタ変数。*/
7  };
8
9  int main() {
10   struct NODE *head, /* リストの先頭を指すポインタ変数。*/
11               *p,    /* 作業用ポインタ変数。*/
12               *r,    /* 新たなデータを指すポインタ変数。*/
13               *t;    /* リストの末尾を指すポインタ変数。*/
14   int data,           /* データ。*/
15       count;          /* データの個数。*/
16
17   /* リストの作成。*/
18
19   /* リストヘッドの作成。*/
20   head = (struct NODE *)malloc(sizeof(struct NODE));
21   head->next = NULL;
22
23   /* 初期設定。*/
24   t = head;
25
26   while( 1 ) {
27     /* データの読み込み。*/
28     scanf("%d",&data); if( data <= 0 ) { break; }
29
30     r = (struct NODE *)malloc(sizeof(struct NODE));
31     r->info = data;
32     r->next = NULL;
33
34     t->next = ①      ;
35     t = ②      ;
36   }
37
38   /* データの個数を求める。*/
39   count = 0;
40   p = ③           ;
41   while( p != ④      ) {
```

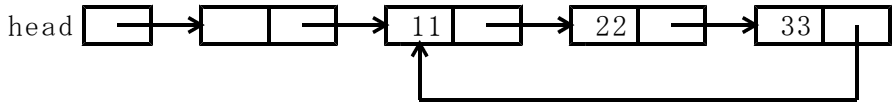
```
42      count++;
43      p = ⑤          ;
44    }
45    printf("%d¥n",count);
46  }
```

実行結果

```
% cc ls141.c
% ./a.out
0
0

% ./a.out
11 0
1

% ./a.out
11 22 0
2

% ./a.out
11 22 33 44 55 0
5
```

問題２

リスト作成後、円状に並ぶリストにして、先頭から7個のデータを出力するプログラムを完成せよ。



上記の場合、11 22 33 11 22 33 11 と出力される。

●プログラム（ls142.c）

```
1  /* << ls142.c >> */
2  #include <stdio.h>
3  #include <stdlib.h>
4  struct NODE {
5    int info;
6    struct NODE *next; /* nextはポインタ変数。*/
7  };
8
9  int main() {
10   struct NODE *head, /* リストの先頭を指すポインタ変数。*/
11               *p,    /* 作業用ポインタ変数。*/
12               *r,    /* 新たなデータを指すポインタ変数。*/
13               *t;    /* リストの末尾を指すポインタ変数。*/
14   int data,           /* データ。*/
15       count;          /* データの個数。*/
16
17   /* リストの作成。*/
18
19   /* リストヘッドの作成。*/
20   head = (struct NODE *)malloc(sizeof(struct NODE));
21   head->next = NULL;
22
23   /* 初期設定。*/
24   t = head;
25
26   while( 1 ) {
27     /* データの読み込み。*/
28     scanf("%d",&data); if( data <= 0 ) { break; }
29
30     r = (struct NODE *)malloc(sizeof(struct NODE));
31     r->info = data;
32     r->next = NULL;
33
34     t->next = ⑥[    ];
35     t = ⑦[    ];
36   }
37
```

```
38    /* 円状にする。*/
39    ⑧                          ;
40
41    /* 出力。*/
42    printf("出力：");
43    count = 0;
44    p = head->next;
45    while( p != NULL ) {
46      printf("%d ",p->info);
47      count++;
48      if( count == 7 ) { break; }
49      p = p->next;
50    }
51    printf("¥n");
52  }
```

実行結果

```
% cc ls142.c
% ./a.out
0
出力：

% ./a.out
11 0
出力：11 11 11 11 11 11 11

% ./a.out
11 22 33 0
出力：11 22 33 11 22 33 11

% ./a.out
11 22 33 44 55 66 77 88 99 0
出力：11 22 33 44 55 66 77
```