リストＩ

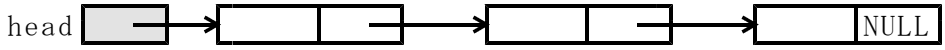
０．目次

１．再帰的なデータ構造によるリストの表現

リストは、データの一部に次のデータの記憶場所を示す情報（ポインタという）を持つ構造をいう。データの挿入、削除等ができる。

構造体を用いた再帰的なデータ構造（構造体のメンバに他の構造体を指すポインタを含む）でリストを実現する。

headは、リストの先頭を指すポインタ変数、
矢印はポインタ、
NULLはデータの最後を意味する。ヌルポインタと呼ばれる。



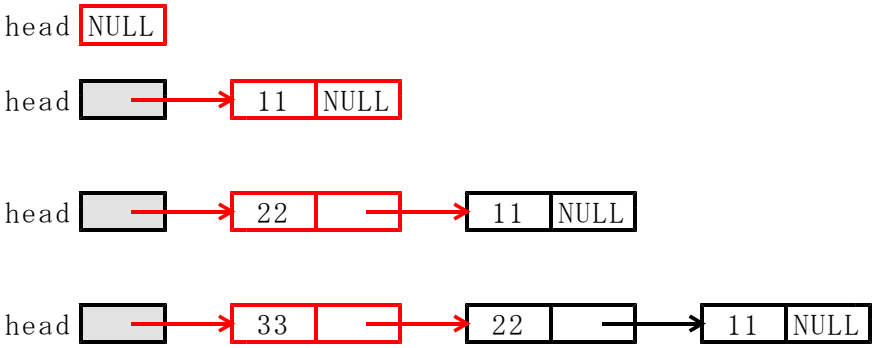
空リストは、headにNULLを代入して表す。

ポインタ変数を使って、リストを処理するプログラムを作成する場合、一般的な処理と特別な処理に分類して書き、両者をまとめると簡潔なプログラムになる。

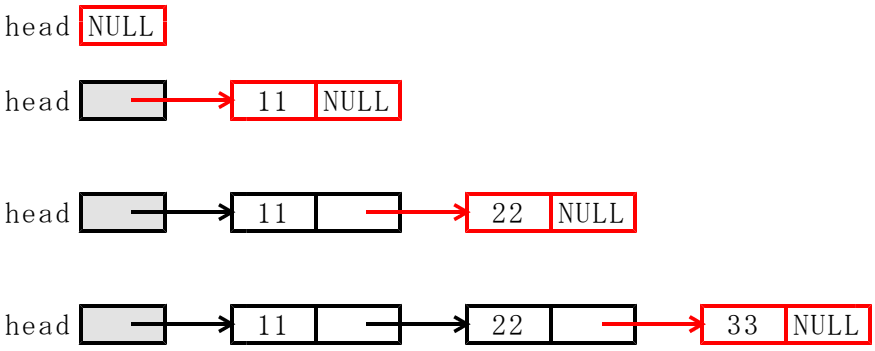
１．１　リストの作成と表示

リスト作成法を３つ示す。

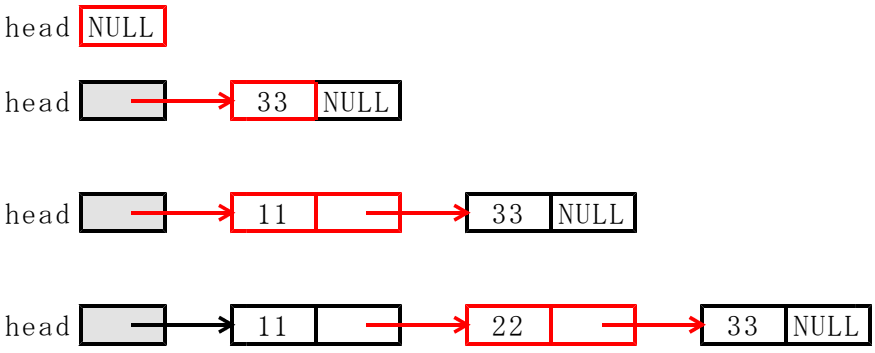
・作成法１：データ(11,22,33)をリストの先頭に追加していく。



・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く。



１．１．１　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、

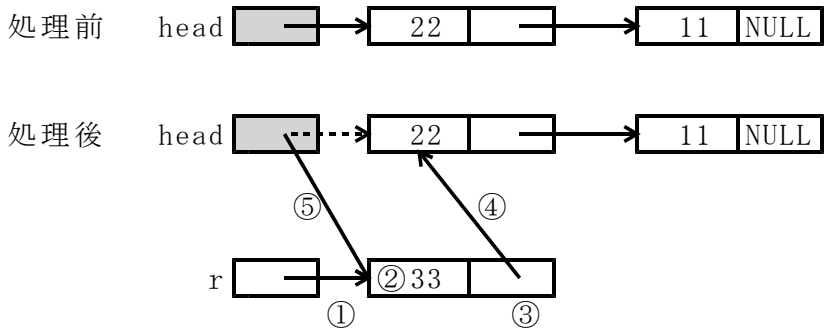
処理１：既存のリストの先頭に、新たなデータを追加する場合への対応。

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。

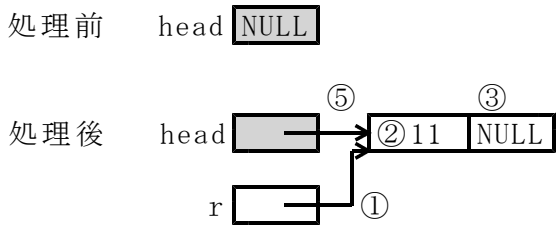
になる。前者と後者を組み合わせると、簡潔なプログラムになる。

●処理１



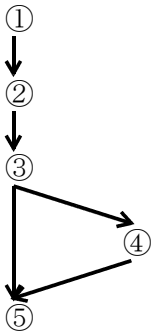

```
   /* データの読込み。*/
   scanf("%d",&data);
   /* ポインタ変数の設定。*/
①  r = (struct NODE *)malloc(sizeof(struct NODE));
   /* リストへの追加。*/
②  r->info = data;
③  r->next = NULL;
④  r->next = head;
⑤  head = r;
```

●処理２




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

処理１と処理２をまとめると、つぎのようになる。



処理１と処理２を比較すると、①②③⑤が共通で、④が異なる。したがって、処理１のときにのみ、④を実行するようにif文で記述すればよい。
この結果、次のプログラムを得る。

●プログラム（ls111.c）

```
1  /* << ls111.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。*/
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   /* 初期設定。*/
19   head = NULL;
20
21   while( 1 ) {
22     /* データの読み込み。*/
23     scanf("%d",&data); if( data <= 0 ) { break; }
24
25     /* データの追加。*/
26     r = (struct NODE *)malloc(sizeof(struct NODE)); /*①*/
27     r->info = data;                                 /*②*/
28     r->next = NULL;                                 /*③*/
29
30     /* 処理１*/
31     if( head != NULL ) {
```

```
32        r->next = head;                                    /*④*/
33      }
34      head = r;                                            /*⑤*/
35    }
36
37    /* リストの表示。*/
38    printf("リスト：");
39    p = head;
40    while( p != NULL ) {
41      printf("%d ",p->info);
42      p = p->next;
43    }
44    printf("¥n");
45  }
```

実行結果

```
% cc ls111.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
```

１．１．２　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、

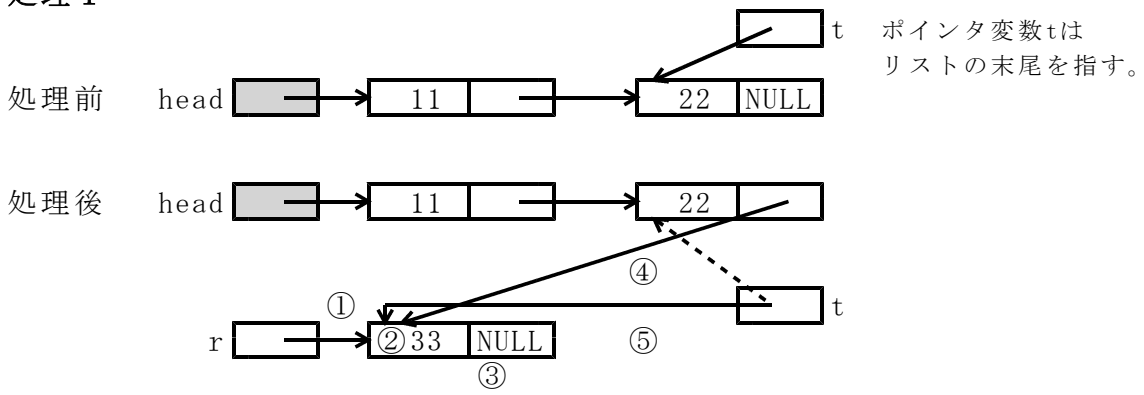
処理１：既存のリストの末尾に、新たなデータを追加する場合への対応。

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。

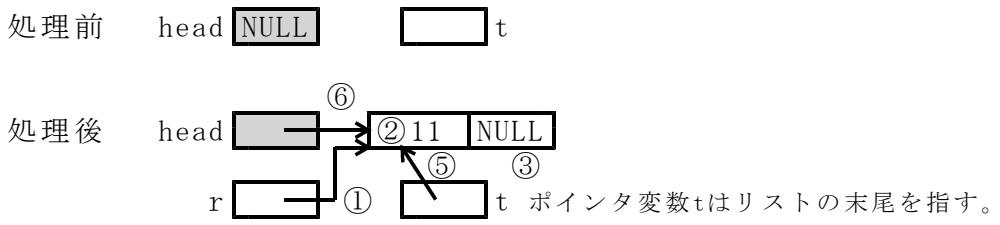
になる。前者と後者を組み合わせると、簡潔なプログラムになる。

●処理１



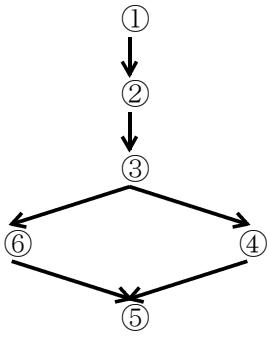

```
   /* データの読込み。*/
   scanf("%d",&data);
   /* ポインタ変数の設定。*/
①  r = (struct NODE *)malloc(sizeof(struct NODE));
   /* リストへの追加。*/
②  r->info = data;
③  r->next = NULL;
④  t->next = r; /* ポインタ変数tはリストの末尾を指す。*/
⑤  t = r;
```

●処理２




```
   scanf("%d",&data);
①  r = (struct NODE *)malloc(sizeof(struct NODE));
②  r->info = data;
③  r->next = NULL;
⑥  head = r;
⑤  t = r;
```

処理１と処理２をまとめると、つぎのようになる。



処理１と処理２を比較すると、①②③⑤が共通で、④⑥が異なる。したがって、処理１のときに④を実行し、処理２のときに⑥を実行するようにif～else文で記述すればよい。この結果、次のプログラムを得る。

●プログラム（ls112.c）

```
1  /* << ls112.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。*/
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   /* 初期設定。*/
20   head = NULL;
21
22   while( 1 ) {
23     /* データの読み込み。*/
24     scanf("%d",&data); if( data <= 0 ) { break; }
25
26     /* データの追加。*/
27     r = (struct NODE *)malloc(sizeof(struct NODE));      /*①*/
28     r->info = data;                                      /*②*/
29     r->next = NULL;                                      /*③*/
30
31     if( head == NULL ) {
32       /* 処理２*/
```

```
33        head = r;                                          /*⑥*/
34      } else {
35        /* 処理１*/
36        t->next = r;                                       /*④*/
37      }
38      t = r;                                               /*⑤*/
39    }
40
41    /* リストの表示。*/
42    printf("リスト：");
43    p = head;
44    while( p != NULL ) {
45      printf("%d ",p->info);
46      p = p->next;
47    }
48    printf("¥n");
49  }
```

実行結果

```
% cc ls121.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
```

１．１．３　昇順を保存して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。

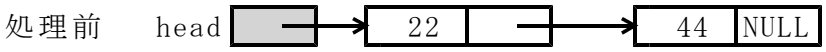
になる。特別な処理は、

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。

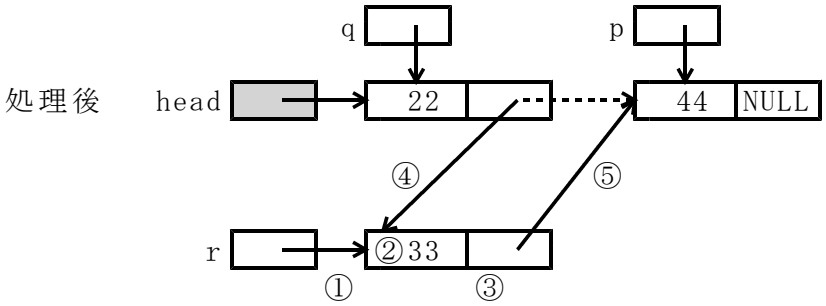
になる。前者と後者を組み合わせると、簡潔なプログラムになる。

●処理１

・リストの途中に追加する場合

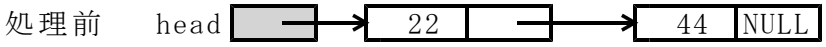


ポインタ変数qは、挿入位置の直前の要素を指す。
ポインタ変数pは、挿入位置の直後の要素を指す。

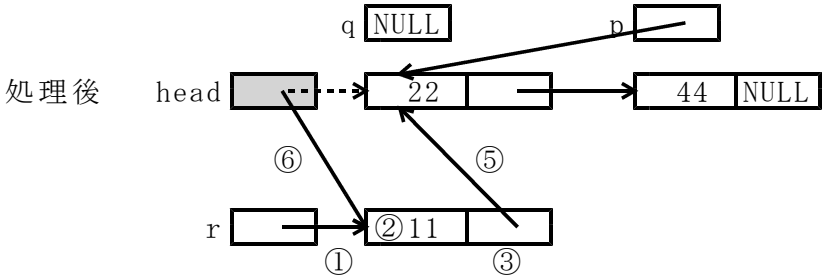



```
   /* データの読込み。*/
   scanf("%d",&data);
   /* ポインタ変数の設定。*/
①  r = (struct NODE *)malloc(sizeof(struct NODE));
   /* リストへの追加。*/
②  r->info = data;
③  r->next = NULL;
   /* 挿入位置を探す。ポインタ変数pは、挿入位置の直後の要素、
   ポインタ変数qは、直前の要素を指す。*/
④  q->next = r;
⑤  r->next = p;
```

・リストの先頭に追加する場合

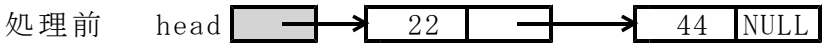


ポインタ変数qは、挿入位置の直前の要素を指す。
ポインタ変数pは、挿入位置の直後の要素を指す。

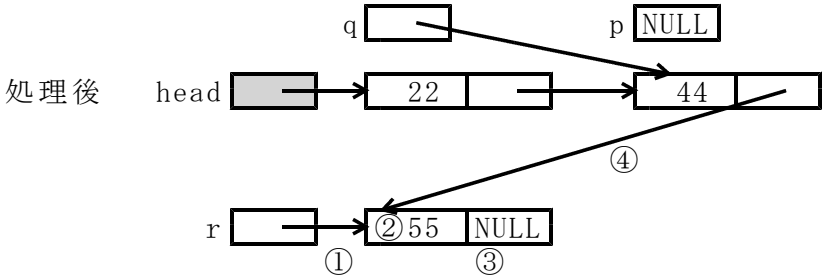



```
   /* データの読込み。*/
   scanf("%d",&data);
   /* ポインタ変数の設定。*/
①  r = (struct NODE *)malloc(sizeof(struct NODE));
   /* リストへの追加。*/
②  r->info = data;
③  r->next = NULL;
   /* 挿入位置を探す。ポインタ変数pは、挿入位置の直後の要素、
   ポインタ変数qは、直前の要素を指す。*/
⑥  head = r;
⑤  r->next = p;
```

・リストの末尾に追加する場合



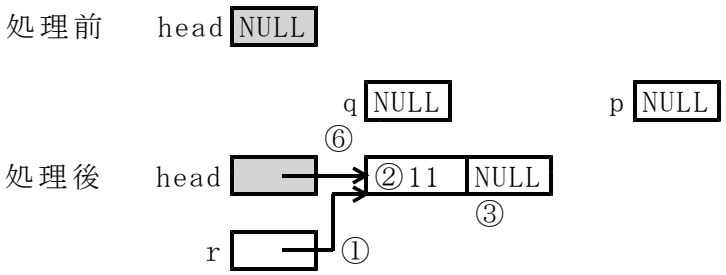
ポインタ変数qは、挿入位置の直前の要素を指す。
ポインタ変数pは、挿入位置の直後の要素を指す。




```
   /* 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、挿入位置の直後の要素、
   ポインタ変数qは、直前の位置を指す。*/
④  q->next = r;
```

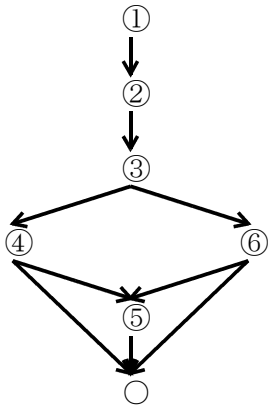
●処理２

・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;
⑥  head = r;
```

処理１と処理２をまとめると、つぎのようになる。



処理１と処理２を比較すると、①②③が共通である。

つぎに、空リストに追加する場合と空でないリストの先頭に追加する場合は、ポインタ変数pとポインタ変数headが一致し、リストの途中と末尾に追加する場合は、ポインタ変数pとポインタ変数headが一致しないので、④と⑥の処理をif～else文で記述する。

最後に、リストの途中に追加する場合とリストの先頭に追加する場合は、ポインタ変数pがNULLでないので、処理⑤を記述する。

この結果、次のプログラムを得る。

●プログラム（ls113.c）

```
1  /* << ls113.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。*/
16   int data;          /* データ。*/
17
18   /* リストの作成。*/
19
20   /* 初期設定。*/
21   head = NULL;
22
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     /* 追加する位置を見つける。
33        ポインタ変数pは、挿入位置の直後の要素、
34        ポインタ変数qは、挿入位置の直前の位置を指す。*/
35     p = head;
36     q = NULL;
37     while( p != NULL ) {
38       if( p->info > data ) { break; }
39       q = p;
40       p = p->next;
41     }
42
43     /* 追加位置を見つけた。*/
44     if( p == head ) {
45       /* 空リストに追加する場合。
46          リストの先頭に追加する場合。*/
```

```
47        head = r;                                          /*⑥*/
48      } else {
49        /* リストの末尾に追加する場合。
50           リストの途中に追加する場合。*/
51        q->next = r;                                       /*④*/
52      }
53      if( p != NULL ) {
54        /* リストの途中に追加する場合。
55           リストの先頭に追加する場合。*/
56        r->next = p;                                       /*⑤*/
57      }
58    }
59
60    /* リストの表示。*/
61    printf("リスト：");
62    s = head;
63    while( s != NULL ) {
64      printf("%d ",s->info);
65      s = s->next;
66    }
67    printf("¥n");
68  }
```

実行結果

```
% cc ls131.c
% ./a.out
0
リスト：
% ./a.out
11 0
リスト：11
% ./a.out
22 44 33 11 0
リスト：11 22 33 44
% ./a.out
55 88 22 11 44 99 66 33 77 0
リスト：11 22 33 44 55 66 77 88 99
```

１．２　問題

問題１

リスト作成(作成法1による)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（ls121.c）

```
1  /* << ls121.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,
11               *p,
12               *r;
13   int count, /* データの個数を保存する変数。*/
14       data;
15
16   /* リストの作成。*/
17   head = NULL;
18
19   while( 1 ) {
20     /* データの読み込み。*/
21     scanf("%d",&data); if( data <= 0 ) { break; }
22
23     r = (struct NODE *)malloc(sizeof(struct NODE));
24     r->info = data;
25     r->next = NULL;
26
27     if( head != NULL ) {
28       r->next = ①          ;
29     }
30     head = ②     ;
31   }
32
33   count = 0;
34   /* リストをたどりデータの個数を求める。*/
35   p = ③          ;
36   while( p != NULL ) {
37     count++; p = ④               ;
38   }
```

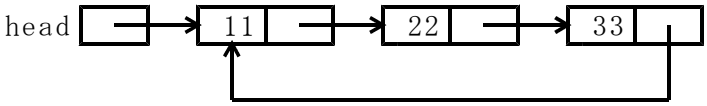
```
39    printf("%d¥n",count);
40  }
```

実行結果

```
% cc ls121.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(作成法２による)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（ls122.c）

```
1  /* << ls122.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,
12               *p,
13               *r,
14               *t;
15   int count,
16       data;
17
18   /* リストの作成。*/
19
20   /* 初期設定。*/
21   head = NULL;
22
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));
29     r->info = data;
30     r->next = NULL;
31
32     if( head == NULL ) {
33       /* 処理２*/
34       head = r;
35     } else {
36       /* 処理１*/
37       t->next = r;
```

```
38      }
39      t = r;
40    }
41
42    /* 円上に並ぶようにする。*/
43    ⑤                    ;
44
45    /* 出力。*/
46    printf("出力：");
47    p = head;
48    count = 0;
49    while( p != NULL ) {
50      printf("%d ",p->info);
51      count++;
52      if( count == 7 ) { break; }
53      p = p->next;
54    }
55    printf("¥n");
56  }
```

実行結果

```
% cc ls122.c
% ./a.out
0
出力：
% ./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
```