配列処理Ｉ

０．目次

１．配列要素の読み込み

１．１　読み込む要素数が未知の場合

正整数xを読み込む。最後のデータは0以下とする。

●プログラム(ar111.c)

```
1  /* << ar111.c >> */
2  #include <stdio.h>
3  #define N 100 /* データの最大個数。*/
4
5  int main() {
6    int a[N+1], /* データを保存する配列。*/
7        i,
8        n,       /* データの個数。*/
9        x;       /* データ。*/
10
11   /* 初期設定。*/
12   n = 0;
13
14   /* 配列要素の読み込み処理。*/
15   while( 1 ) {
16     /* データxの読み込み。*/
17     scanf("%d",&x);
18     if( x <= 0 ) { break; }
19     if( n > N ) {
20       printf("オーバーフロー¥n");
21       break;
22     }
23     n++; a[n] = x;
24   }
25
26   /* データの表示。*/
27   for( i=1; i<=n; i++ ) {
28     printf("a[%d] = %d¥n",i,a[i]);
29   }
30 }
```

実行結果

```
% cc ar111.c
% ./a.out
11 33 55 22 44 0
a[1] = 11
a[2] = 33
a[3] = 55
a[4] = 22
a[5] = 44
```

１．２　読み込む要素数が既知の場合

●プログラム(ar112.c)

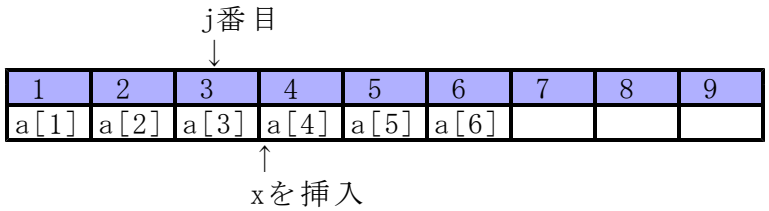
```
1  /* << ar112.c >> */
2  #include <stdio.h>
3  #include <stdlib.h>
4  #define N 100 /* データの最大個数。*/
5
6  int main() {
7    int a[N+1], /* データを保存する配列。*/
8        i,
9        n;       /* データの個数。*/
10
11   /* データの読み込み。*/
12   scanf("%d",&n);
13   if( (n <= 0)||(n > N) ) { exit(0); }
14
15   /* 配列要素の読み込み処理。*/
16   for( i=1; i<=n; i++ ) {
17     scanf("%d",&a[i]);
18   }
19
20   /* 配列の表示。*/
21   for( i=1; i<=n; i++ ) {
22     printf("a[%d] = %d¥n",i,a[i]);
23   }
24 }
```

実行結果

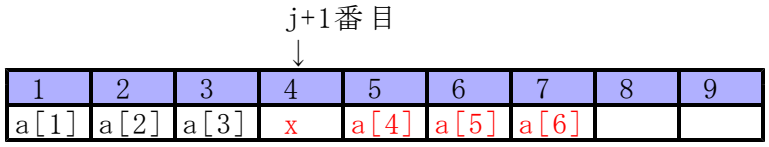
```
% cc ar112.c
% ./a.out
5
11 22 33 44 55
a[1] = 11
a[2] = 22
a[3] = 33
a[4] = 44
a[5] = 55
```

２．配列要素の挿入

j番目の要素の直後にデータｘを挿入。



方法：j+1番目以降のデータをひとつずつ後ろにずらし、j+1番目にデータｘを保存する。



●プログラム(ar211.c)

```
1  /* << ar211.c >> */
2  #include <stdio.h>
3  #include <stdlib.h>
4  #define N 100 /* データの最大個数。*/
5
6  int main() {
7    int a[N+1], /* データを保存する配列。*/
8        i,
9        j,      /* j番目。*/
10       n,      /* データの個数。*/
11       x;      /* 挿入データ。*/
12
13   /* データの読み込み。*/
14   scanf("%d",&n);
15   if( (n <= 0)||(n > N) ) { exit(0); }
16   for( i=1; i<=n; i++ ) { scanf("%d",&a[i]); }
17
18   /* 挿入データxの読み込み。*/
19   scanf("%d%d",&j,&x);
20
21   /* 挿入処理。*/
22   for( i=n; i>j; i-- ) {
23     a[i+1] = a[i];
24   }
25   a[j+1] = x;
26   n++;
27
28   /* 配列の表示。*/
29   for( i=1; i<=n; i++ ) {
30     printf("a[%d] = %d¥n",i,a[i]);
31   }
32 }
```

実行結果

```
% cc ar211.c
% ./a.out
6
11 22 33 44 55 66
3 77
a[1] = 11
a[2] = 22
a[3] = 33
a[4] = 77
a[5] = 44
a[6] = 55
a[7] = 66
```

３．配列要素の削除

データｘを削除（複数存在する場合はすべて）。
a[2]=a[4]=xとする。

1	2	3	4	5	6	7	8	9
a[1]	**x**	a[3]	**x**	a[5]	a[6]			

方法：配列の先頭から末尾まで調べ、データｘを見つけるごとに削除し、
以降のデータを前に詰める。

1	2	3	4	5	6	7	8	9
a[1]	a[3]	a[5]	a[6]					

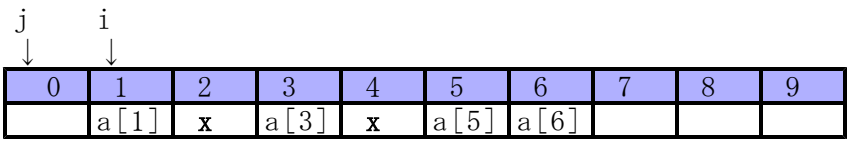
●プログラム(ar311.c)

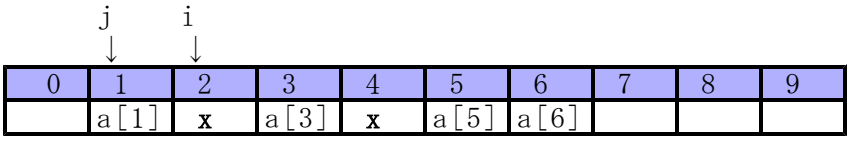
```
1  /* << ar311.c >> */
2  #include <stdio.h>
3  #include <stdlib.h>
4  #define N 100 /* データの最大個数。*/
5
6  int main() {
7    int a[N+1], /* データを保存する配列。*/
8        i,
9        j,      /* 挿入する位置。*/
10       n,      /* データの個数。*/
11       x;      /* 削除データ。*/
12
13   /* データの読み込み。*/
14   scanf("%d",&n);
15   if( (n <= 0)||(n > N) ) { exit(0); }
16   for( i=1; i<=n; i++ ) { scanf("%d",&a[i]); }
17
18   /* 削除データxの読み込み。*/
19   scanf("%d",&x);
20
21   /* 削除処理。*/
22   j = 0;
23   for( i=1; i<=n ; i++ ) {
24     if( a[i] != x ) {
25       j = j+1;
26       a[j] = a[i];
27     }
28   }
29   n = j;
30
31   /* 配列の表示。*/
32   for( i=1; i<=n; i++ ) {
33     printf("a[%d] = %d¥n",i,a[i]);
34   }
35 }
```

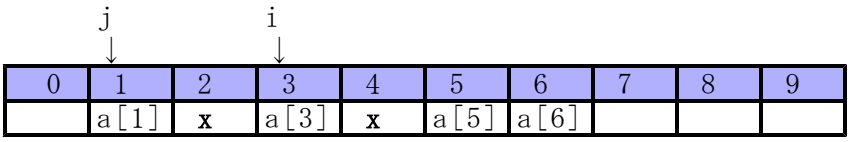
実行結果

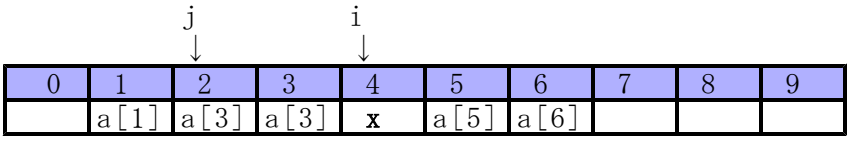
```
% cc ar311.c
% ./a.out
6
11 22 33 44 55 66
33
a[1] = 11
a[2] = 22
a[3] = 44
a[4] = 55
a[5] = 66
```

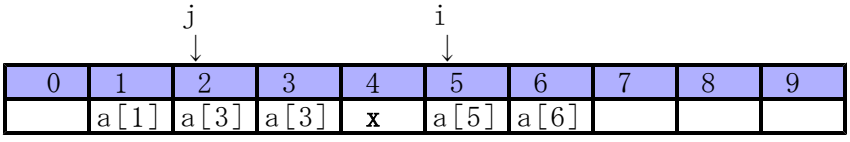
●処理状況

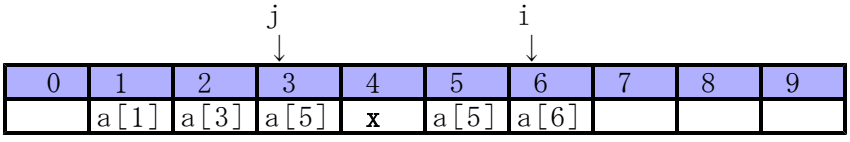


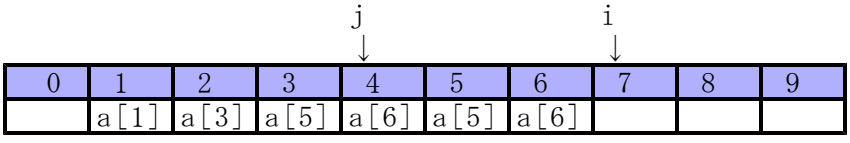


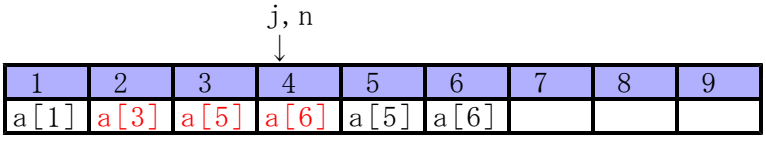








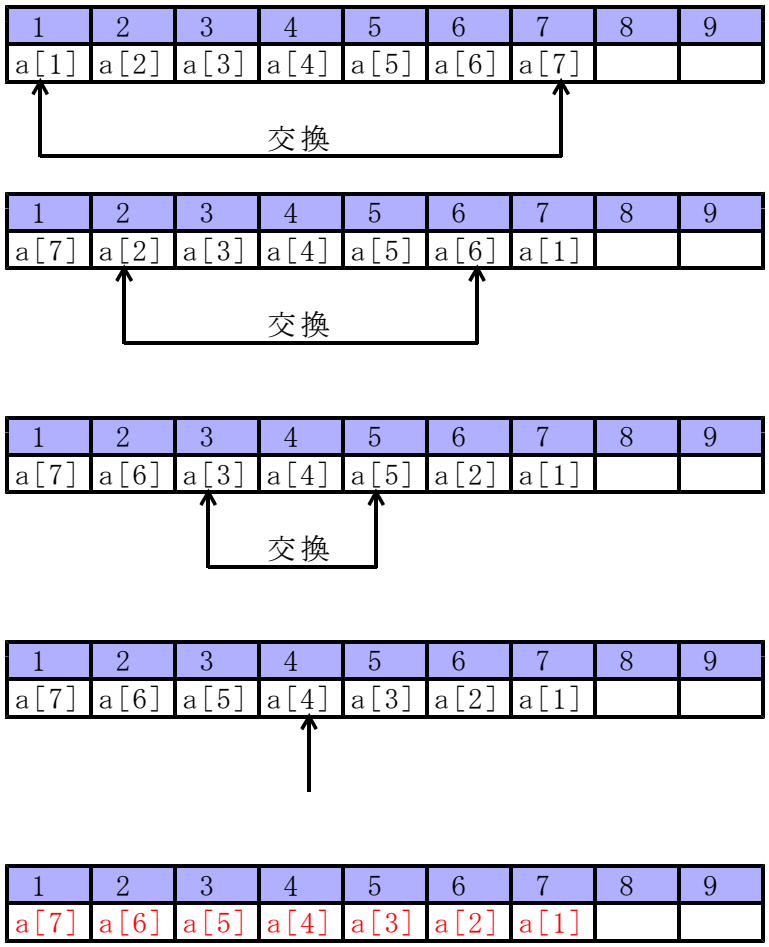




４．配列の反転

配列の要素を反転。

方法：配列の左端と右端から要素を１つずつ選び交換する操作を左端は右方向、右端は左方向に要素がなくなるまで行う。



●プログラム(ar411.c)

```
1  /* << ar411.c >> */
2  #include <stdio.h>
3  #include <stdlib.h>
4  #define N 100 /* データの最大個数。*/
5
6  int main() {
7    int a[N+1], /* データを保存する配列。*/
8        i,      /* 左端。*/
9        j,      /* 右端。*/
10       n,      /* データの個数。*/
11       x;
12
13   /* データの読み込み。*/
14   scanf("%d",&n);
15   if( (n <= 0)||(n > N) ) { exit(0); }
16   for( i=1; i<=n; i++ ) { scanf("%d",&a[i]); }
17
18   /* 反転処理。*/
19   i = 1; j = n;
20   while( i < j ) {
21     /* 要素の交換。*/
22     x = a[i] ; a[i] = a[j]; a[j] = x;
23     i = i+1;
24     j = j-1;
25   }
26
27   /* 配列の表示。*/
28   for( i=1; i<=n; i++ ) {
29     printf("a[%d] = %d¥n",i,a[i]);
30   }
31 }
```

実行結果

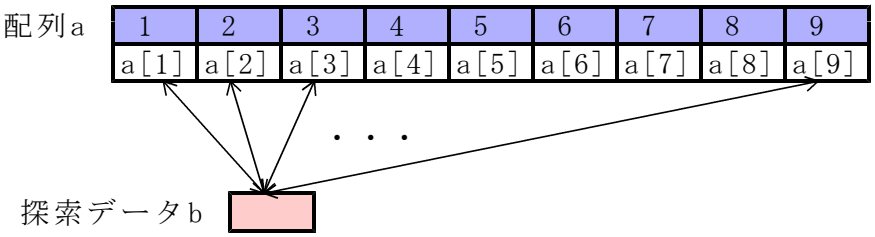
```
% cc ar411.c
% ./a.out
5
11 22 33 44 55
a[1] = 55
a[2] = 44
a[3] = 33
a[4] = 22
a[5] = 11
```

５．探索

５．１　素朴な方法

ランダムなデータ（データは正整数と仮定）から探索。

方法：配列の左端から右端の方向へ１つずつ調べていく。



・探索データと配列データの比較に加えて、配列の右端を越えないように、右端に到達したかどうか調べる必要がある。

●プログラム(ar511.c)

```
1  /* << ar511.c >> */
2  #include <stdio.h>
3  #include <stdlib.h>
4  #define N 1000 /* データの最大個数。*/
5
6  int main() {
7    int a[N+1], /* データを保存する配列。*/
8        b,      /* 探索データ。*/
9        i,
10       k,      /* 探索データbの位置。*/
11       n;      /* データの個数。*/
12
13   /* データの読み込み。*/
14   scanf("%d",&n);
15   if( (n <= 0)||(n > N) ) { exit(0); }
16   for( i=1; i<=n; i++ ) { scanf("%d",&a[i]); }
17
18   /* 配列の表示。*/
19   printf("配列：");
20   for( i=1; i<=n; i++ ) { printf("%4d",a[i]); }
21   printf("¥n");
22
23   /* 探索処理。*/
24   while( 1 ) {
25     scanf("%d",&b);
26     if( b <= 0 ) { break; }
27     printf("探索データ：%4d は、",b);
```

```
28      k = 1; /* 探索データbが見つかった位置。*/
29      while( k <= n ) {
30        if( a[k] == b ) { break; }
31        k++;
32      }
33      if( k <= n ) { /* 見つかった場合。*/
34        printf("%d番目にあります。¥n",k);
35      } else { /* 見つからなかった場合。*/
36        printf("ありません。¥n");
37      }
38    }
39  }
```

実行結果

```
% cc ar511.c
% ./a.out
9
55 88 22 66 11 44 99 33 88
配列：  55  88  22  66  11  44  99  33  88
66
探索データ：  66 は、4番目にあります。
47
探索データ：  47 は、ありません。
0
```

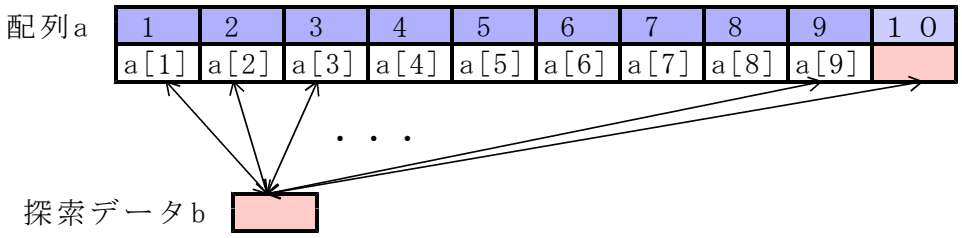
●データa[i]=10*i (1≦i≦500)に対して、探索データbを指定したとき、データ間の比較回数

b	データ間比較回数
10	2
11	1001
500	100
501	1001
5000	1000
5001	1001

５．２　番兵法

ランダムなデータ（データは正整数と仮定）から探索。

番兵法：最後の配列要素の後に探索するデータを代入しておき、
配列の左端から右端の方向へ１つずつ調べていく。



・探索データと配列データの比較のみでよい。
データの位置と値の両方を比較しなくて済むので、素朴な方法と
比べて、探索のための比較回数がほぼ半分になっている。

●プログラム(ar521.c)

```
1  /* << ar521.c >> */
2  #include<stdio.h>
3  #include <stdlib.h>
4  #define N 1000 /* データの最大個数。*/
5
6  int main() {
7    int a[N+1], /* データを保存する配列。*/
8        b,      /* 探索データ。*/
9        i,
10       k,      /* 探索データbの位置。*/
11       n;      /* データの個数。*/
12
13   /* データの読み込み。*/
14   scanf("%d",&n);
15   if( (n <= 0)||(n > N) ) { break; }
16   for( i=1; i<=n; i++ ) { scanf("%d",&a[i]); }
17
18   /* 配列の表示。*/
19   printf("配列：");
20   for( i=1; i<=n; i++ ) { printf("%4d",a[i]); }
21   printf("¥n");
22
23   /* 探索処理。*/
24   while( 1 ) {
25     /* 探索データbの読み込み。*/
26     scanf("%d",&b);
27     if( b <= 0 ) { break; }
```

```
28      printf("探索データ：%4d は、",b);
29      k = 1;        /* bが見つかった位置。*/
30      a[n+1] = b; /* 番兵となるデータ。*/
31      while( a[k] != b ) {
32        k = k+1;
33      }
34      if( k <= n ) {
35        printf("%d番目にあります。¥n",k);
36      } else {
37        printf("ありません。¥n");
38      }
39    }
40  }
```

実行結果

```
1  % cc ar521.c
2  % ./a.out
3  9
4  55 88 22 66 11 44 99 33 88
5  配列：  55  88  22  66  11  44  99  33  88
6  66
7  探索データ：  66 は、4番目にあります。
8  47
9  探索データ：  47 は、ありません。
10 0
```

●データa[i]=10*i (1≦i≦500)に対して、探索データbを指定したとき、データ間の比較回数

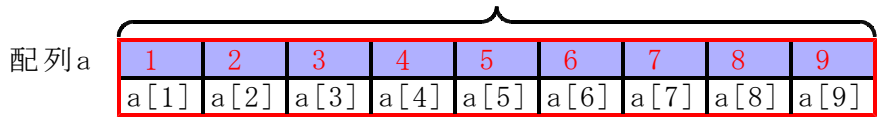
b	データ間比較回数
10	1
11	501
500	50
501	501
5000	500
5001	501

５．３　二分探索法

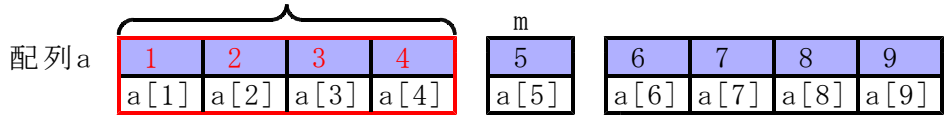
昇順に並べられたn個のデータ(a[1]≦a[2]≦…≦a[n])から探索。

方法：昇順に並べられたn個のデータ(a[1]≦a[2]≦…≦a[n])を中央の位置mで前半部分(a[1]からa[m-1]まで)と後半部分(a[m+1]からa[n]まで)に分ける。
探索データｘがa[m]と一致すれば見つかったとして終了。
探索データｘがa[m]より小さいときは前半部分の探索を続ける。
探索データｘがa[m]より大きいときは後半部分の探索を続ける。

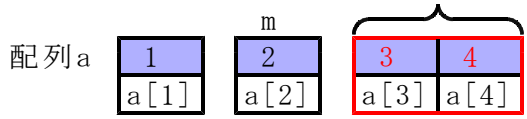
x=a[4]の場合。



m=(1+9)/2=5



m=(1+4)/2=2



m=(3+4)/2=3

m=(4+4)/2=4

●プログラム(ar531.c)

```
1  /* << ar531.c >> */
2  #include <stdio.h>
3  #include <stdlib.h>
4  #define N 1000 /* データの最大個数。*/
5
6  int main() {
7    int a[N+1], /* データを保存する配列。*/
8        b,      /* 探索データ。*/
9        i,j,m,
10       n;      /* データの個数。*/
11
12   /* データの読み込み。*/
13   scanf("%d",&n);
14   if( (n <= 0)||(n > N) ) { exit(0); }
15   for( i=1; i<=n; i++ ) { scanf("%d",&a[i]); }
16
17   /* 配列の表示。*/
18   printf("配列：");
19   for( i=1; i<=n; i++ ) { printf("%4d",a[i]); }
20   printf("¥n");
21
22   /* 探索処理。*/
23   while( 1 ) {
24     /* 探索データbの読み込み。*/
25     scanf("%d",&b);
26     if( b <= 0 ) { break; }
27     printf("探索データ：%4d は、",b);
28     i = 1; j = n; /* a[i]からa[j]までが探索範囲。*/
29     while( i <= j ) {
30       m = (i+j)/2;
31       if( b == a[m] ) { break; }
32       if( b < a[m] ) {
33         j = m-1;
34       } else {
35         i = m+1;
36       }
37     }
38     if( i <= j ) {
39       printf("%d番目にあります。¥n",m);
40     } else {
41       printf("ありません。¥n");
42     }
43   }
44 }
```

実行結果

```
% cc ar531.c
% ./a.out
9
11 22 33 44 55 66 77 88 99
配列：  11  22  33  44  55  66  77  88  99
33
探索データ：  33 は、3番目にあります。
47
探索データ：  47 は、ありません。
0
```

●データa[i]=10*i (1≦i≦500)に対して、探索データbを指定した場合。

b	データ間比較回数
10	24
11	29
500	21
501	29
5000	27
5001	29

５．４　ハッシュ法

入力データ(正整数)：123，666，845，335，287，851

●ハッシュ表の作成

ハッシュ表を配列で実現する。入力データをハッシュ表の大きさ(10)で割ったあまりの位置にデータを格納する。すでに他のデータが格納されている場合は、位置を１ずらしながら空いている位置を見つけて格納する。ハッシュ表の末尾に達したときは、ハッシュ表の先頭に移動する。空いている位置が見つからなかった場合は、ハッシュ表の大きさを変更しデータの再配置を行う。

初期状態

0	0
1	0
2	0
3	0
4	0
5	0
6	0
7	0
8	0
9	0

123登録

0	0
1	0
2	0
→ 3	123
4	0
5	0
6	0
7	0
8	0
9	0

666登録

0	0
1	0
2	0
3	123
4	0
5	0
→ 6	666
7	0
8	0
9	0

845登録

0	0
1	0
2	0
3	123
4	0
→ 5	845
6	666
7	0
8	0
9	0

335登録

0	0
1	0
2	0
3	123
4	0
→ 5	845
→ 6	666
→ 7	335
8	0
9	0

287登録

0	0
1	0
2	0
3	123
4	0
5	845
6	666
→ 7	335
→ 8	287
9	0

851登録

0	0
→ 1	851
2	0
3	123
4	0
5	845
6	666
7	335
8	287
9	0

●探索

x=845の探索
found

0	0
1	851
2	0
3	123
4	0
→ 5	845
6	666
7	335
8	287
9	0

x=335の探索
found

0	0
1	851
2	0
3	123
4	0
→ 5	845
→ 6	666
→ 7	335
8	287
9	0

x=333の探索
not found

0	0
1	851
2	0
→ 3	123
→ 4	0
5	845
6	666
7	335
8	287
9	0

x=679の探索
not found

0	0
1	851
2	0
3	123
4	0
5	845
6	666
7	335
8	287
→ 9	0

●ハッシュ表の例

入力データ：　456, 723, 12, 546, 797, 349, 260, 585, 123

（１）TMAX=13の場合

k	t[k]
0	546
1	456
2	260
3	585
4	797
5	0
6	123
7	0
8	723
9	0
10	0
11	349
12	12

（２）TMAX=15の場合

k	t[k]
0	585
1	0
2	797
3	723
4	349
5	260
6	456
7	546
8	123
9	0
10	0
11	0
12	12
13	0
14	0

●プログラム(ar541.c)

```
1  /* << ar541.c >> */
2  #include <stdio.h>
3  #include <stdlib.h>
4  #define N 1000  /* データの最大個数。*/
5  #define TMAX 31 /* ハッシュ表のサイズ。*/
6
7  int main() {
8    int a[N+1],   /* データを保存する配列。*/
9        i,
10       k,
11       n,        /* データの個数。ただし、n≦TMAX。*/
12       t[TMAX], /* ハッシュ表。*/
13       x;        /* 探索データ。*/
14
15   /* データの読み込み。*/
16   scanf("%d",&n);
17   if( (n <= 0)||(n > N) ) { exit(0); }
18   for( i=1; i<=n; i++ ) { scanf("%d",&a[i]); }
19
20   /* データの表示。*/
21   printf("データ ¥n");
22   for( i=1; i<=n; i++ ) {
23     printf("%4d",a[i]);
24     if( i%10 == 0 ) { printf("¥n"); }
25   }
```

```
26    printf("¥n");
27
28    /* ハッシュ表tの作成。*/
29    for( k=0; k<TMAX; k++ ) { t[k] = 0; }
30    for( i=1; i<=n; i++ ) {
31      k = a[i]%TMAX;
32      while( t[k] != 0 ) {
33        k++;
34        if( k >= TMAX ) { k = 0; }
35      }
36      t[k] = a[i];
37    }
38
39    /* 探索処理。*/
40    while( 1 ) {
41      /* 探索データの読み込み。*/
42      scanf("%d",&x);
43      if( x <= 0 ) { break; }
44      k = x%TMAX;
45      while( (t[k] != x)&&(t[k] != 0) ) {
46        k++;
47        if( k >= TMAX ) { k = 0; }
48      }
49      if( t[k] == x ) { /* 見つかった。*/
50        printf("found %d ¥n",x);
51      } else { /* 見つからなかった。*/
52        printf("not found %d ¥n",x);
53      }
54    }
55  }
```

実行結果

```
1   % cc ar541.c
2   % ./a.out
3   6
4   123 666 845 335 287 851
5   データ
6    123 666 845 335 287 851
7   845
8   found 845
9   335
10  found 335
11  333
12  not found 333
13  679
14  not found 679
15  0
```

●ハッシュ表を完成せよ。

入力データ：　456,723,12,546,797,349,260,585,123

・TMAX=11の場合

k	t[k]
0	
1	
2	
3	
4	
5	
6	
7	
8	
9	
10	