リスト　（list）

一定の型の要素を，０個以上，一列に，順序付けて並べたデータ構造.

線形リスト（linear list），列，並び（sequence）

リストの例：

学籍番号の並び

5090046　　5090023　　5090004　　5090032

リストの操作（リストの抽象データ型）：

初期化，要素の追加，参照（ｎ番目），削除，

探索（指定された値を探す），

前の要素，後の要素，先頭要素，最終要素.

リスト結合・分割・複製・要素数

１次元配列によるリストの実現

添字で参照

リストの大きさがｎのとき：

参照は計算量Ｏ（１）．

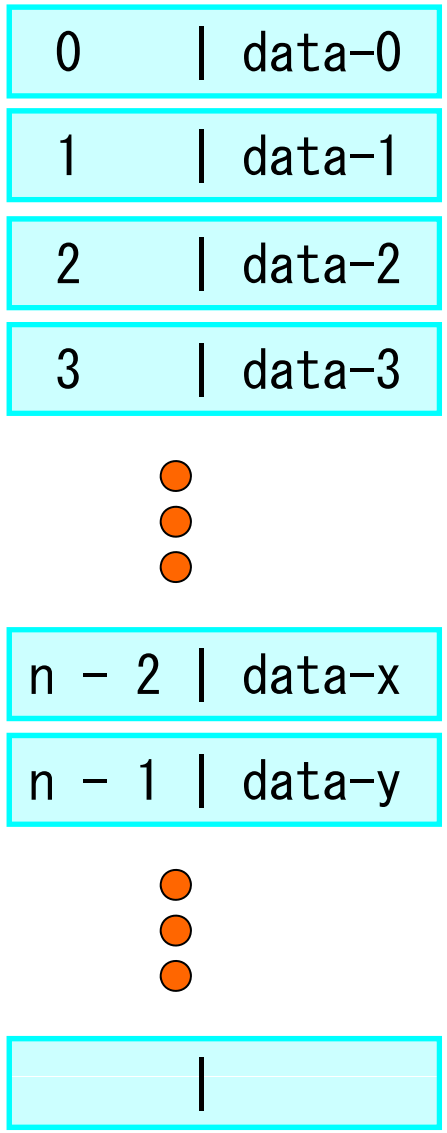
特定されている位置への追加，削除は，

ｎ／２ ～ Ｏ（ｎ）．

追加，削除要素に続く要素を，

移動させる．

探索は，ｎ／２ ～ Ｏ（ｎ）．



連結リスト （linked list） によるリストの実現

セル： リスト要素のデータと，次のセルの所在場所を収めるデータ構造.

所在場所を指定することで，次のセルをつなぎ，リストとする.

構造体　(structure)
複数のデータをまとめ，１つの変数のように扱う．

```
struct  personal  {
  int  id;                                  /* メンバ：学籍番号 */
  int  lang;                                /* メンバ：国語の成績 */
  int  math;                                /* メンバ：数学の成績 */
};

int  main( ) {
  struct  personal  ken;                    /* 構造体変数の宣言 */

  ken.id = 5101;  ken.lang = 90;  ken.math = 95;
                                            /* メンバへの代入 */
  printf (" %d, %d ¥n",
          ken.id,                           /* 学籍番号の表示 */
          ken.lang + ken.math);             /* 国語，数学の計 */
}
```

ポインタ　(pointer) :
メモリ上にあるデータのアドレスとデータの大きさを持つ.

整数型変数とポインタの例

```
int  main ( ) {
  int  k;       /* int 変数 k */
  int  *p;      /* ポインタ変数
                   p */
  k = 3;
  p = &k;       /* アドレス演算 */

  printf("%d, %d, %d ¥n",
    k,
    *p,         /* 間接参照 */
    *p + 2);
}
```

構造体変数とポインタの例

```
int  main ( ) {
  struct  personal  ken;
  struct  personal  *q;

  ken.id = 5101;
  ken.lang = 90;
  ken.math = 95;
  q = &ken;

  printf ("%d, %d ¥n",
        ken.id,
        ken.lang + ken.math);
  printf ("%d, %d ¥n",
        q->id,
        q->lang + q->math);
}
```

構造体によるセルの実現１

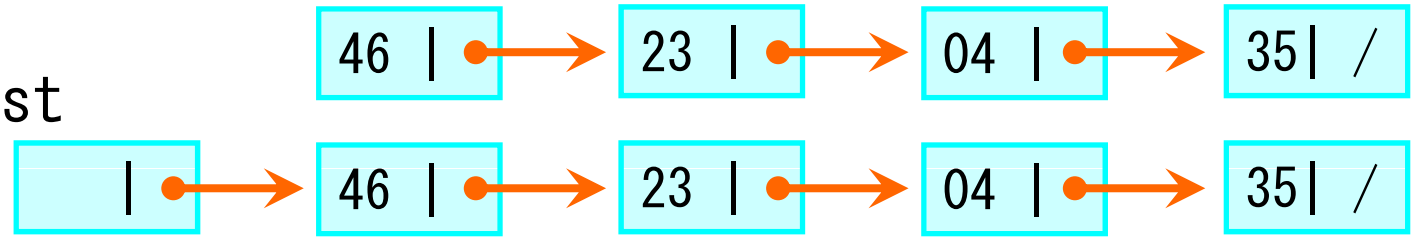
```
struct  list_elm0  {
  int      id;                    /* 要素データ */
  struct  list_elm0  *next;       /* 次のセルの場所 */
};
```

リスト：　　46　　23　　04　　35

構造体変数：　　ａ　　ｂ　　ｃ　　ｄ

ａ（46, ｂへのポインタ）
→　ｂ（23, ｃへのポインタ）
→　ｃ（04, ｄへのポインタ）
→　ｄ（35, end ）

先頭セル lst



```c
#include <stdio.h>

struct list_elm0 {
    int     id;
    struct  list_elm0  *next;
};

int main(void) {
  struct list_elm0 lst;
  struct list_elm0 a, b, c, d;
  struct list_elm0 x;

  lst.next = &a;
  a.id = 46;     a.next = &b;
  b.id = 23;     b.next = &c;
  c.id = 04;     c.next = &d;
  d.id = 32;     d.next = NULL;

    x = lst;
    while (1) {
      if (x.next == NULL)
          break;
      x = *x.next;
      printf ("%d ¥n", x.id);
    }
}
```

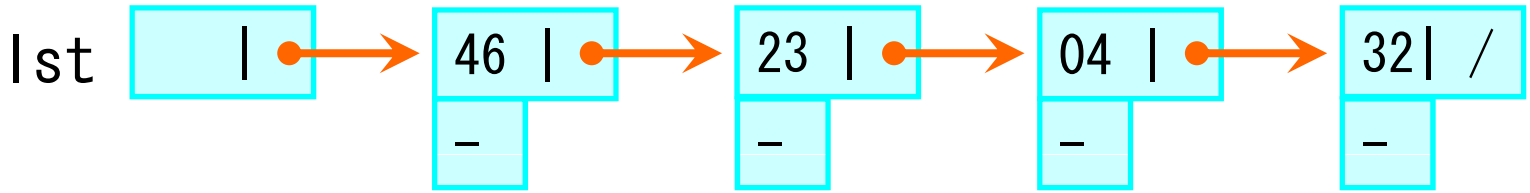
```
46
23
4
32
```

構造体によるセルの実現２（混在した型のデータ）

学籍番号：　46　　23　　04　　32　　・・・
名前：　　Beth　John　Mick　Kate

```
struct  list_elm1  {
        int      id;
        char     name[21];
        struct   list_elm1  *next;
};
```

キーと，キーに対応つけたデータの組を要素とした
リスト　→　連想リスト　（association list）
学籍簿：　id をキーとして，対応する名前を探索.



構造体単純変数での実現

```
#include <stdio.h>
#include <string.h>

struct list_elm1 {
  int     id;
  char    name[21];
  struct  list_elm1  *next;
};

int main(void)
{
  struct list_elm1
    lst, a, b, c, d, x;

  lst.next = &a;

  a.id = 46;
  strcpy(a.name, "Beth");
  a.next = &b;

  b.id = 23;
  strcpy(b.name, "John");
  b.next = &c;

  c.id = 04;
  strcpy(c.name, "Mick");
  c.next = &d;

  d.id = 32;
  strcpy(d.name, "Kate");
  d.next = NULL;

  x = lst;
  while (1) {
    if (x.next == NULL)
      break;
    x = *x.next;
    printf ("%d %s ¥n",
            x.id, x.name);
  }
}
```

```
46 Beth
23 John
4 Mick
32 Kate
```

セルによる連結リスト操作の計算量

追加，削除：　特定されている位置の後ろへの追加，
削除は，Ｏ（１）．
ポインタの書き換えですむ．

参照，探索：　ｎ／２　～　Ｏ（ｎ）
←　位置の特定するためには，
参照，探索が必要．

リスト結合，リスト分割：
長さ，位置のポインタが特定されていれば，
Ｏ（１）．

リスト複製，リスト要素数：　Ｏ（ｎ）．

循環リスト　(circular list)

最終要素の指す先が，先頭要素，または，リストの頭.

双方向リスト（doubly-linked list：　双方向連結リスト）

次の要素へのポインタとともに，直前の要素への
ポインタを持つ.

```
struct  list_elm2  {
        int     id;
        char    name[21];
        struct  list_elm2  * previous;
        struct  list_elm2  * next;
        }
```

特定位置の，前後の，追加，削除が，Ｏ（１）で可能.