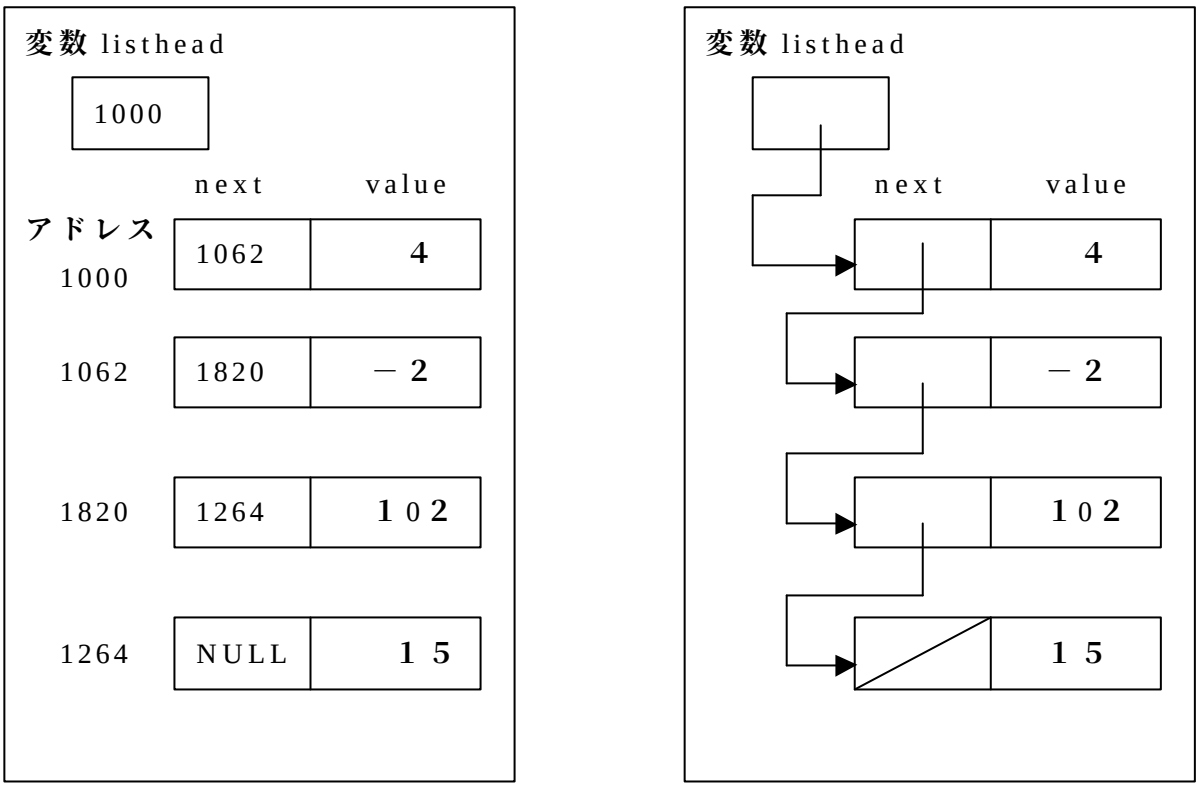
４．　リスト，木構造（データ構造）について(1)

連結リスト(linked list)

リストに含まれる各要素をポインタによってつなぎ合わせて表現する．ポインタによって連結(link)されていて，各要素には次の要素へのポインタを付加する．たとえば，要素４，－２，１０２，１５がこの順で並んだ連結リストを構造体で表現すると，各要素は，2 つのメンバ next と value をもつ構造体となる．図１に示すように，value は要素の値，next は次の要素へのポインタである．連結リストで次の要素がないことを示すのに NULL(ヌル[nʌl])ポインタが使われる．NULL の値として通常は 0 が用いられる．変数 listhead はこの連結リストを指すポインタである．図１(a)では，説明のために具体的なアドレスを記しているが，実際は具体的なアドレスの値は必要とされない．(b)は図による表記である．NULL ポインタは斜線で表現している．



(a)メモリ内の表現　　(b)ポインタを矢印で示す図表現

図１　連結リストの表現

```
struct CELL{
  struct CELL *next;    /*次のセル（cell：リストの箱）へのポインタ*/
  int   value;          /*値*/
}*listhead;
```

図２　構造体定義例

リストの図が箱のイメージであるから構造体を CELL と命名している．変数 listhead はセルへのポインタ，stract　CELL*型として定義されていて，このリストの先頭を指す．変数 listhead からポインタをたどって各要素を順番にアクセスする．

連結リストの基本的な性質

連結リストを配列と“メモリ確保”,“要素のアクセス法”と“要素の参照の計算量”で比較すると，図表を得る：

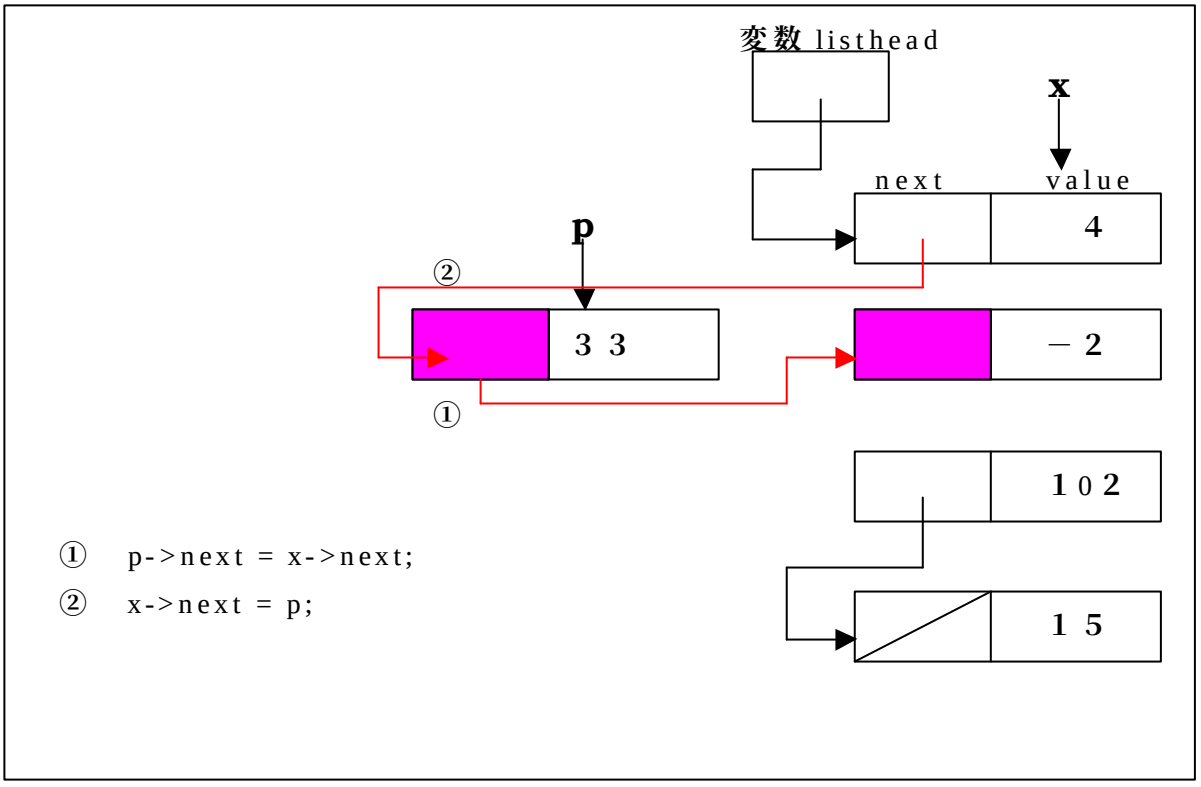
	配列	連結リスト
メモリ確保	領域をまとめて確保	ランダムに確保可能
アクセス（参照）法（注１）	ランダムアクセス	シーケンシャルアクセス
要素参照の平均計算量(注 2)	O(1)	O(n)

（注１）配列ではたとえば 3 番目の要素を直接 a[2]と指定することが可能であるが連結リストは順にポインタをたどっていかなければならない．

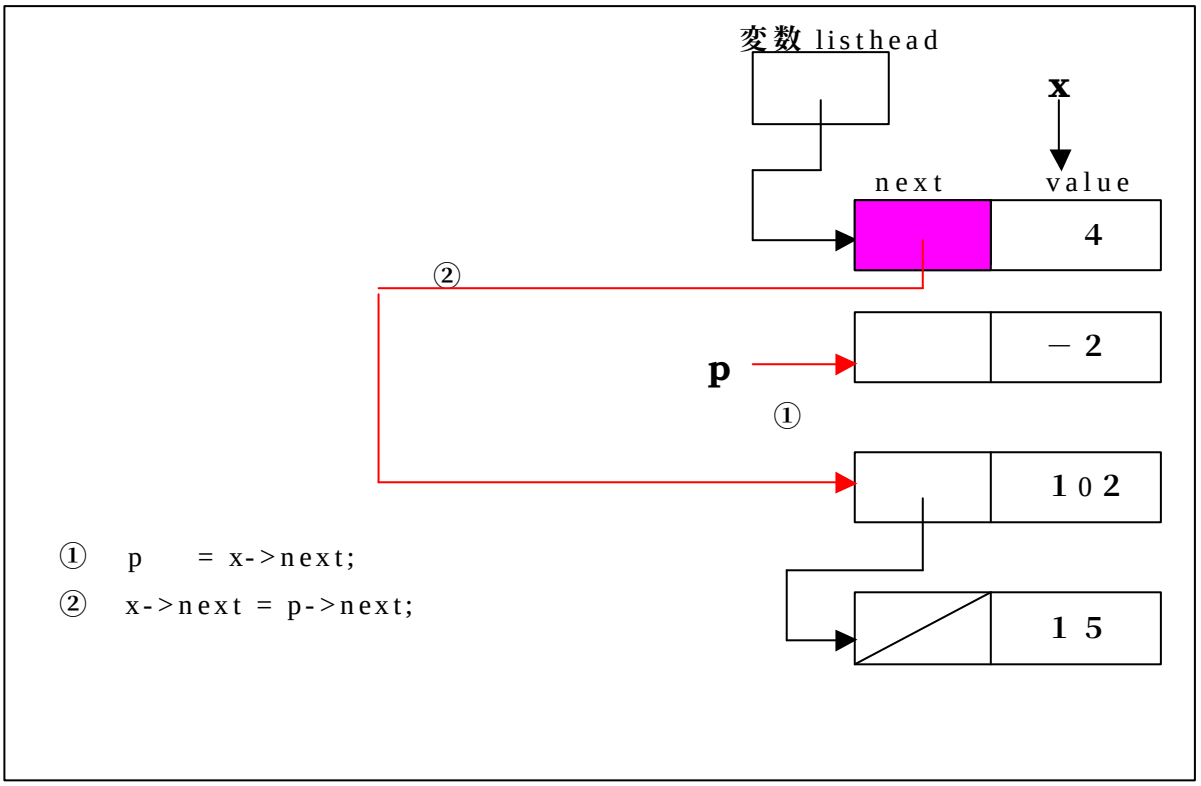
（注２）リストの要素数が n 個であるとき任意の要素を参照するには平均 n/2 回ポインタをたぐる必要がある．

この表ではリストにはあまりメリットがないように思えるが，要素の挿入や削除がより簡単に行うことができる点はメリットである．

連結リストへの挿入



連結リストからの削除



リストのメモリ管理

連結リストを使う場合，要素を入れるセルは必要になるたびに割り当てるのでプログラム上明示的なメモリ管理が必要となる．ちなみに配列の場合は必要な領域をあらかじめ割り当てておく必要があることをすでに学習している．C 言語では標準ライブラリにメモリを割り当てる関数 **malloc()**，メモリを解放する関数 **free()**が用意されている．一方，このような明示的な解放ではなくて使用しなくなったメモリを自動的にシステムが回収する技法をガベージコレクション（garbage collection：ゴミ集め）という．

＜＜関数 **malloc** の書式＞＞

```
malloc（サイズ）;
```

指定したサイズの領域を確保し，先頭アドレスを void 型のポインタ変数として返す．ヘッダファイル stdlib.h に含まれる

＜＜関数 **free** の書式＞＞

```
free（アドレス）;
```

関数 malloc()によって確保した領域を使用しなくなった場合，その領域を解放する．関数 free()により指定したアドレスの記憶領域を解放する．

プログラム例（listc2-s06.c）

（手順）　　malloc（sizeof(struct i_list)）;

これにより確保した領域の先頭アドレスが得られる．構造体 i_list の大きさは sizeof 演算子を用いて求めている．

```
top_pt      = (struct i_list *) malloc(sizeof(struct i_list));
```

リスト構造の先頭を表すポインタ top_pt に先頭アドレスを格納する．このとき戻り値は viod 型のポインタ変数となっているので構造体 i_list 型のポインタ変数にキャスト演算（データ変換の明示的指定）をおこなう．この構造体へのポインタ変数を使用して 1 件目のデータを格納する．

2 件目の入力データを格納するために記憶領域を確保してその領域の先頭アドレスを 1 件目のデータの次ポインタ i_pt に格納する．このようにポインタでつなぐ処理を入力データが終了するまで繰り返す．そして，最後の次ポインタ i_pt にリストの終了を示す NULL を格納する．

【引用文献１】石畑清著『アルゴリズムとデータ構造』，岩波書店，1996．
【引用文献２】近藤嘉雪著『定本 C プログラマのためのアルゴリズムとデータ構造』，ソフトバンクパブリッシング株式会社，2000．