連結リスト（片方向リスト）を用いたデータの基本操作

(1) 片方向リストを作成するにはまず次のような構造体を定義する：

```
 1 /*   cell(構造体 cell の定義)    */
 2 struct cell
 3 {
 4   int data;
 5   struct cell *next;
 6 } *root;
```

(2) キーボードから次々にデータを読み込みリストを作っていく。

```
 1 /*    make(リストの生成)  */
 2 void make()
 3 {
 4   int i,x;
 5   struct cell *old,*new,*ptr;
 6
 7   root=old=(struct cell*)malloc(sizeof(struct cell));
 8
 9 /*   データをキーボードから読み込む。終るときは Ctrl-d   */
10   while (scanf("%d",&x) == 1)
11     {
12       new=(struct cell*)malloc(sizeof(struct cell));
13       new->data=x;
14       old->next=new;
15       old=new;
16     }
17   new->next=NULL;
18 }
```

(3) リストの印刷

```
 1 /*    print(リストの印刷)   */
 2 void print()
 3 {
 4   struct cell *ptr;
 5   ptr=root;
 6   while (ptr->next != NULL)
 7     {
 8       printf("%d ",ptr->next->data);
 9       ptr=ptr->next;
10     }
11   printf("\n");
12 }
```

(4) x に一致するデータがあるかどうかを探索

```
 1 /*   search(リストの探索)   */
 2 void search()
 3 {
 4   struct cell *p;
 5   int x;
 6
 7   printf("何を探していますか? \n");
 8   scanf("%d",&x);
 9
10   p=root->next;
```

```
11   while ( (p->next != NULL) && (p->data != x) )
12     p=p->next;
13
14   if (p->data = x)
15     printf("ありました。\n");
16   else
17     printf("ありませんでした。\n");
18 }
```

(5) セルの更新（x を y に更新する）

```
 1 /*   renew(セルの更新)   */
 2 void renew()
 3 {
 4   struct cell *p;
 5   int x,y;
 6
 7   printf("どれを更新しますか? \n");
 8   scanf("%d",&x);
 9   printf("何に更新しますか? \n");
10   scanf("%d",&y);
11
12   p=root->next;
13   while ( (p->next != NULL) && (p->data != x) )
14     p=p->next;
15
16   if (p->data == x)
17     p->data=y;
18   else
19     printf("ありませんでした。\n");
20 }
```

(6) セルの削除（x を削除する）

```
 1 /*     delete(セルの削除)   */
 2 void delete()
 3 {
 4   int x;
 5   struct cell *p,*q;
 6 /*  x を削除する   */
 7
 8   printf("x を削除します。\n");
 9   printf("x= ");
10   scanf("%d",&x);
11
12   q=root;
13   p=root->next;
14   while ((p->next != NULL) && (p->data !=x ))
15     {
16       q=p;
17       p=p->next;
18     }
19
20   if ( p->data == x)
21     q->next=q->next->next;
22   else
23     printf("%d はありません\n",x);
24
```

```
25 }
```

(7) セルの挿入

```
 1 /*     insert(セルの挿入)   */
 2 void insert()
 3 {
 4   int x,y;
 5   struct cell *p,*q;
 6
 7 /*    x の後に y を挿入  */
 8   printf("x の後に y を挿入します。\n");
 9   printf("x= ");scanf("%d",&x);
10   printf("y= ");scanf("%d",&y);
11
12   p=root->next;
13   while ((p->next != NULL) && (p->data !=x ))
14     p=p->next;
15
16   if ( p->data == x )
17     {
18       q=(struct cell*) malloc(sizeof(struct cell));
19       q->data=y;
20       q->next=p->next;
21       p->next=q;
22     }
23   else
24     printf("%d はありません\n",x);
25 }
```

実行例

プログラムを作るときは，上記関数 (cell, make, print など）を一つにまとめたファイル (`lib.c`) を作り，include する (5 行目参照) と見やすくなる。

1. リストの作成と印刷

```
 1 /*   main.c(単方向連結リストの処理)   */
 2 /*   データの作成と印刷               */
 3 #include <stdio.h>
 4 #include <stdlib.h>
 5 #include "lib.c"
 6
 7 void main()
 8 {
 9   make();
10   print();
11 }
```

```
実行例
123
256
215
330
151
33
123 256 215 330 151 33
```

2. データの探索

```
 1 /*   main.c(単方向連結リストの処理)   */
 2 /*     データの探索 (search)          */
 3 #include <stdio.h>
 4 #include <stdlib.h>
 5 #include "lib.c"
 6
 7 void main()
 8 {
 9   make();
10   search();
11   print();
12 }
```

実行例
何を探していますか?
18
ありました。
15 34 11 18 22 13

3. データの更新

```
 1 /*   main.c(単方向連結リストの処理)    */
 2 /*    データの更新 (renew)             */
 3 #include <stdio.h>
 4 #include <stdlib.h>
 5 #include "lib.c"
 6
 7 void main()
 8 {
 9   make();
10   print();
11   renew();
12   print();
13 }
```

実行例
24 15 33 22 18 10
どれを更新しますか?
33
何に更新しますか?
19
24 15 19 22 18 10

4. データの削除

```
 1 /*   main.c(単方向連結リストの処理)    */
 2  /*    データの削除 (delete)           */
 3 #include <stdio.h>
 4 #include <stdlib.h>
 5 #include "lib.c"
 6
 7 void main()
 8 {
 9   make();
```

```
10   print();
11   delete();
12   print();
13 }
```

実行例

```
14 23 18 10 15 32
x を削除します。
x= 23
14 18 10 15 32
```

5. データの挿入

```
 1 /*   main.c(単方向連結リストの処理)   */
 2 /*    データの挿入(insert)           */
 3 #include <stdio.h>
 4 #include <stdlib.h>
 5 #include "lib.c"
 6
 7 void main()
 8 {
 9   make();
10   print();
11   insert();
12   print();
13 }
```

実行例

```
14 22 15 32 41 17
x の後に y を挿入します。
x= 22
y= 17
14 22 17 15 32 41 17
```