第 8 回 プログラミング応用レポート

15302114 番　山下尚人

提出日：2017 年 12 月 29 日

1　リストの登録 (逆順)

- ソースコード

```
1  #include <stdio.h>
2  #include <stdlib.h>
3
4  struct CELL {
5    struct CELL *next;
6    int value;
7  };
8
9  void set_cell(struct CELL *data, int val){
10   struct CELL *temp;
11   temp = malloc(sizeof(struct CELL));
12
13   if(temp == NULL){
14     printf("Not␣enough␣memory\n");
15     exit(EXIT_FAILURE);
16   }
17
18   temp->next = data->next;
19   temp->value = val;
20   data->next = temp;
21 }
22
23 void print_cell(struct CELL *data){
24   struct CELL *p;
25
26   for(p=data->next; p!=NULL; p=p->next){
27     printf("value=%d\n", p->value);
28   }
29 }
30
31 int main (void){
32   struct CELL data;
33   data.next = NULL;
34   data.value = 0;
35
36   set_cell(&data,1);
37   set_cell(&data,2);
38   set_cell(&data,5);
```

```
39      set_cell(&data,10);
40      set_cell(&data,20);
41      set_cell(&data,100);
42
43      print_cell(&data);
44
45      return EXIT_SUCCESS;
46  }
```

- 実行結果

```
1   value=100
2   value=20
3   value=10
4   value=5
5   value=2
6   value=1
```

- 考察

  9〜21 行目は set_cell 関数を定義している。この関数は引数で指定したセルの一つ後ろに新しいセルを挿入する。引数は構造体 CELL 型のポイント型変数 data と、整数型の変数 val をとる。data は新しく後ろに挿入するセルの一つ前のポインタであり、val は新たに挿入するセルの値となる。
  11 行目は構造体 CELL 型変数 temp(ポインタ型) を、malloc 関数で動的にメモリ領域を確保している。確保するメモリの大きさは構造体の CELL 型の大きさとしている。13〜16 行目はメモリが確保できなかった場合（temp が NULL だった場合）について条件分岐され、プログラムは異常終了する。
  18〜20 行目は data の後ろに新しいセルの temp を挿入している。
  18 行目は temp のメンバ変数 next に、data のメンバ変数 next を代入している。19 行目は temp のメンバ変数 value に、val を代入している。20 行目は data のメンバ変数 next に、temp のポインタ代入している。

  23〜29 行目は print_cell 関数を定義している。この関数は連結リストのセルの値を順番に標準出力に表示する。引数は構造体 CELL 型のポイント型変数 data をとる。data は表示するリストの先頭のセルとなる。

  31〜46 行目は main 関数で、set_cell 関数で連結リストを作成し print_cell 関数で作成したリストを標準出力に表示している。
  32〜34 行目は連結リストの先頭のダミーデータである、構造体 CELL 型の変数 data を定義している。
  36〜41 行目は set_cell 関数により、連結リストを変数 data の後ろにセルを挿入している。43 行目は print_cell 関数により、36〜41 行目で作成したリストの値を標準出力に表示している。

実行結果で 36〜41 行目の値と逆順に表示されるのは、print_cell 関数が原因だと考えた。
36〜41 行目では第一引数にずっと作成したリストの先頭のセルである、data を指定している。
print_cell 関数は第一引数で指定したセルの一つ後ろに新たなセルを挿入する。
よって、新たなセルは常に（ダミーデータを除いた）先頭に作成されるので、逆順にデータは並び、表示される。

2 リストの登録 (順番)

- ソースコード

```
1  #include <stdio.h>
2  #include <stdlib.h>
3
4  struct CELL {
5    struct CELL *next;
6    int value;
7  };
8
9  void set_cell(struct CELL *data, int val){
10   struct CELL *temp;
11   temp = malloc(sizeof(struct CELL));
12
13   if(temp == NULL){
14     printf("Not␣enough␣memory\n");
15     exit(EXIT_FAILURE);
16   }
17
18   while(data->next!=NULL){
19     data = data->next;
20   }
21
22   temp->next = data->next;
23   temp->value = val;
24   data->next = temp;
25 }
26
27 void print_cell(struct CELL *data){
28   struct CELL *p;
29
30   for(p=data->next; p!=NULL; p=p->next){
31     printf("value=%d\n", p->value);
32   }
33 }
34
35 int main (void){
36   struct CELL data;
37   data.next = NULL;
38   data.value = 0;
39
40   set_cell(&data,1);
41   set_cell(&data,2);
42   set_cell(&data,5);
43   set_cell(&data,10);
```

```
44      set_cell(&data,20);
45      set_cell(&data,100);
46
47      print_cell(&data);
48
49      return EXIT_SUCCESS;
50  }
```

- 実行結果

```
1   value=1
2   value=2
3   value=5
4   value=10
5   value=20
6   value=100
```

- 考察

  このプログラムは逆順のプログラムに 18〜20 行目の 3 行を追記したものとなる。

  これにより set_cell 関数は、第一引数に指定したセルの一番最後に新しいデータを作成することになった。

  18 行目は data のメンバ変数 next に関数が NULL となる（リストの一番最後のセルとなる）まで while 文でループする。19 行目は data に、data のメンバ変数の next を代入している。

  これにより 22〜24 行目での data は、第一引数で指定したセルの一番最後のセルになっており、新たなセルは一番最後に作成される。