課題 5

アルゴリズムとデータ構造B

第13回

課題 5

- リストの基本操作（InsertFront, Remove, InsertbyIndex）の勉強を兼ね，それぞれの処理の途中・終了後の状況がわかるように追加で情報を表示

```
     0: 0x6000013f4090    100 0x6000013f4080
     1: 0x6000013f4080     90 0x6000013f4070
     2: 0x6000013f4070     80 0x6000013f4060
     3: 0x6000013f4060     70 0x6000013f4050
     4: 0x6000013f4050     60 0x6000013f4040
     5: 0x6000013f4040     50 0x6000013f4030
     6: 0x6000013f4030     40 0x6000013f4020
     7: 0x6000013f4020     30 0x6000013f4010
     8: 0x6000013f4010     20 0x6000013f4000
     9: 0x6000013f4000     10 0x0
```

処理前の線形リストの状態

```
list process ?InsertFront(1), InsertbyIndex(2), Search(3) or
挿入するノードのデータを入力してください：110
先頭にノードを挿入した後のリスト
     0: 0x6000013f40a0    110 0x6000013f4090 <- edited node
     1: 0x6000013f4090    100 0x6000013f4080
     2: 0x6000013f4080     90 0x6000013f4070
     3: 0x6000013f4070     80 0x6000013f4060
     4: 0x6000013f4060     70 0x6000013f4050
     5: 0x6000013f4050     60 0x6000013f4040
     6: 0x6000013f4040     50 0x6000013f4030
     7: 0x6000013f4030     40 0x6000013f4020
     8: 0x6000013f4020     30 0x6000013f4010
     9: 0x6000013f4010     20 0x6000013f4000
    10: 0x6000013f4000     10 0x0
```

先頭に追加されたノードを明確に表示

課題 5

- リストの基本操作（InsertFront, Remove, InsertbyIndex）の勉強を兼ね，それぞれの処理の途中・終了後の状況がわかるように追加で情報を表示

```
削除するノードのデータを入力してください：90
i = 0
     0: 0x6000013f40a0    110 0x6000013f4090 <- q, p
     1: 0x6000013f4090    100 0x6000013f4080
     2: 0x6000013f4080     90 0x6000013f4070
     3: 0x6000013f4070     80 0x6000013f4060
     4: 0x6000013f4060     70 0x6000013f4050
     5: 0x6000013f4050     60 0x6000013f4040
     6: 0x6000013f4040     50 0x6000013f4030
     7: 0x6000013f4030     40 0x6000013f4020
     8: 0x6000013f4020     30 0x6000013f4010
     9: 0x6000013f4010     20 0x6000013f4000
    10: 0x6000013f4000     10 0x0

i = 1
     0: 0x6000013f40a0    110 0x6000013f4090 <- q
     1: 0x6000013f4090    100 0x6000013f4080 <- p
     2: 0x6000013f4080     90 0x6000013f4070
     3: 0x6000013f4070     80 0x6000013f4060
     4: 0x6000013f4060     70 0x6000013f4050
     5: 0x6000013f4050     60 0x6000013f4040
     6: 0x6000013f4040     50 0x6000013f4030
     7: 0x6000013f4030     40 0x6000013f4020
     8: 0x6000013f4020     30 0x6000013f4010
     9: 0x6000013f4010     20 0x6000013f4000
    10: 0x6000013f4000     10 0x0
```

```
i = 2
     0: 0x6000013f40a0    110 0x6000013f4090
     1: 0x6000013f4090    100 0x6000013f4080 <- q
     2: 0x6000013f4080     90 0x6000013f4070 <- p
     3: 0x6000013f4070     80 0x6000013f4060
     4: 0x6000013f4060     70 0x6000013f4050
     5: 0x6000013f4050     60 0x6000013f4040
     6: 0x6000013f4040     50 0x6000013f4030
     7: 0x6000013f4030     40 0x6000013f4020
     8: 0x6000013f4020     30 0x6000013f4010
     9: 0x6000013f4010     20 0x6000013f4000
    10: 0x6000013f4000     10
```

削除対象を発見する過程の状態

```
ノードを削除した後のリスト
     0: 0x6000013f40a0    110 0x6000013f4090
     1: 0x6000013f4090    100 0x6000013f4070 <- edited node
     2: 0x6000013f4070     80 0x6000013f4060
     3: 0x60
     4: 0x60
     5: 0x6000013f4040     50 0x6000013f4030
     6: 0x6000013f4030     40 0x6000013f4020
     7: 0x6000013f4020     30 0x6000013f4010
     8: 0x6000013f4010     20 0x6000013f4000
     9: 0x6000013f4000     10 0x0
```

削除されたノードの一つ前のノードを明確に表示

InsertFront関数


```
void InsertFront(int x) {
    struct Element *p;
    p = malloc(sizeof(struct Element));
    p->data = x;
    p->next = head;
    head = p;
}
```

問題 1　Remove関数


```
int Remove(int x) {
    int i;
    struct Element *p, *q;
    for (i=0, p=q=head; p!=NULL; i++, q=p, p=p->next) {
        if (p->data == x) {
            if (p == head) {
                head = p->next;
                free(p);
            } else {
                q->next = p->next;
                free(p);
            }
            return i;
        }
    }
    return -1;
}
```

ここでポインタ p と q を用いてリストを辿っているので，
リスト全体の表示，p と q が指しているノードの表示を行う

問題１　InsertbyIndex関数


```
void InsertbyIndex(int index, int x) {
    if (head == NULL || index == 0) InsertFront(data);
    else {
        int i; struct Element *p, *q, *r;
        for (i=0, p=q=head; i<index && p->next!=NULL; i++, q=p, p=p->next) {
```

ここでポインタ p と q を用いてリストを辿っているので，
リスト全体の表示，p と q が指しているノードの表示を行う．
前回は ; として for文内部で何も処理していなかったので注意

```
        }
```

ここでも上記の表示を行う
（ここで表示しないと for文を抜けた後の状態が表示されない）

```
        r = malloc(sizeof(struct Element));
        r->data = x;
        if (i == index) { // 間に挿入
            // 以下，略…
```

問題２　Display関数の実装例


```
void Display(struct Element *q) {
    int i;
    struct Element *p;
    for (i=0, p=head; p!=NULL; i++, p=p->next){
        printf("%6d: %p %6d %p", i, p, p->data, p->next);
```

ここで挿入したノード，削除したノードの一つ前のノードがどのノードなのかを明確に表示する

```
    }
}
```

// InsertFront, IndexbyIndex, Remove 関数内では，挿入したノード，削除したノードの一つ前のノードのアドレスを引数に Display 関数を呼び出す