0.1 Linked Lists – Delete entire list

0.1.1 List Deletion

Linked lists can be implemented in several different ways. Implementing lists as classes makes them very easy to use once the code is functioning properly. Creating and / or using list objects requires us to use some features of C++ that may possibly be new to you.

0.1.2 Initial Class Interface

```
1  /* list.h  */
2
3     // Node structure
4  typedef struct node {
5     int    info ;
6     struct node * next ;
7  } ;
8
9  typedef struct node *NodePtr;
10
11 class LinkedList
12 {
13 public:
14    LinkedList();
15
16    void Add( int x );
17
18    void DeleteFirst();
19
20    void Print();
21
22 private:
23    NodePtr head;
24 };
```

0.1.3 Initial Class Implementation

```
1  /* list.cpp  */
2
3  #include <iostream>
4
5  using namespace std;
6
7  #include "list.h"
8
9
10 LinkedList::LinkedList()
11 {
12     head = NULL;
13 }
14
15 void LinkedList::Add( int x )
16 {
17     NodePtr n = new node;
18
19     n->info = x;
20     n->next = NULL;
21
22     if( head == NULL )
23         head = n;
24     else
25     {
26         n->next = head;
27         head = n;
28     }
29 }
```

```
1  void LinkedList::DeleteFirst()
2  {
3      if( head != NULL )
4      {
5          NodePtr del = head;
6
7          head = head->next;
8          del->next = NULL;
9
10         delete del;
11     }
12 }
```

```
1  void LinkedList::Print()
2  {
3      NodePtr p = head;
4
5      while( p != NULL )
6      {
7          cout << p->info << endl;
8          p = p->next;
9      }
10 }
```

0.1.4 Test Program

```
1  * testList.cpp
2   */
3
4  #include <iostream>
5
6  #include "list.h"
7
8  using namespace std;
9
10    // prototypes
11 void DeleteList( LinkedList L );
12
13
14 int main()
15 {
16   LinkedList L;
17
18   L.Add( 1 );     L.Add( 2 );     L.Add( 3 );
19
20   cout << "Initial list:" << endl;
21   L.Print();
22
23   cout << "Deleting list:" << endl;
24   L.DeleteAll();
25   //DeleteList( L );
26   L.Print();
27
28     // Add to list
29   L.Add( 5 );     L.Add( 4 );
30
31   cout << "New list:" << endl;
32   L.Print();
33
34   return 0;
35 }
```

0.1.5 List – DeleteAll1

A few modifications are necessary to add `DeleteAll1()` to the linked list class.

```
1     // Usage -- add to test program
2  L.DeleteAll1();
3
4     // Interface
5  void DeleteAll1();
6
7     // Implementation
8  void LinkedList::DeleteAll1()
9  {
10    while( head != NULL )
11    {
12       NodePtr del = head;
13
14       head = head->next;
15       del->next = NULL;
16
17       delete del;
18    }
19
20    head = NULL;
21 }
```

Another way to solve this problem is to use a node counter.

A few minor changes to the class are necessary to add a counter.

0.1.6 Revised interface

```
1  class LinkedList
2  {
3  public:
4      LinkedList();
5
6      void Add( int x );
7
8      void DeleteFirst();
9      void DeleteAll1();
10     void DeleteAll2();      // new method
11
12     void Print();
13
14     bool IsEmpty();         // auxiliary methods
15     int  Length();
16
17 private:
18     NodePtr head;
19
20     int count;              // nodes in list
21 }
```

0.1.7 Revised methods

```
1  LinkedList::LinkedList()
2  {
3      head = NULL;
4      count = 0;              // initialize counter
5  }
6
7  void LinkedList::Add( int x )
8  {
9      NodePtr n = new node;
10
11     n->info = x;
12     n->next = NULL;
13     count++;                // update counter
14
15     if( head == NULL )
16         head = n;
17     else
18     {
19         n->next = head;
20         head = n;
21     }
22 }
```

```
1  void LinkedList::DeleteFirst()
2  {
3      if( head != NULL )
4      {
5          NodePtr del = head;
6
7          head = head->next;
8          del->next = NULL;
9
10         delete del;
11
12         --count;              // update counter
13     }
14 }
```

0.1.8 Auxiliary methods

```
1 bool LinkedList::IsEmpty()
2 {
3     return head == NULL;
4 }
5
6 int LinkedList::Length()
7 {
8     return count;
9 }
```

Use the additional object methods to implement `DeleteAll()` as a *free function*.

What is a free function? A function that is *not* associated with a class.

Try this:

```
1 void DeleteList( LinkedList L )
2 {
3   while( !L.IsEmpty() )
4       L.DeleteFirst();
5 }
```

0.1.9 Alternative methods

Use *pass by reference* to correct the problem with the first implementation.

```
1 void DeleteList( LinkedList & L )
2 {
3   while( !L.IsEmpty() )
4     L.DeleteFirst();
5 }
```

```
1 void DeleteList( LinkedList & L )
2 {
3     while( L.Length() != 0 )
4         L.DeleteFirst();
5 }
```

Similar methods can be put in the List class. Note the differences between the free function form and these!

```
1 void LinkedList::DeleteAll2()
2 {
3     while( !IsEmpty() )
4         DeleteFirst();
5 }
```

or

```
1 void LinkedList::DeleteAll2()
2 {
3     while( Length() != 0 )
4         DeleteFirst();
5 }
```