

Nov 09, 14 17:16

stdin

Page 1/25

```
##### # # # ##### # # #####  
##### # # # # # # # #  
##### # # ##### # # #####  
##### # # # # # # # #  
##### # # ##### # # #####
```



Nov 09, 14 17:16

stdin

Page 2/25

Nov 9 11:52 2014 by932/binarytree.cpp Page 1

```

1  /*****
2  *Filename:binarytree.cpp      *
3  *Login:by932                 *
4  *AssignmentNo:ass5           *
5  *DateLastModified:2/11/2014  *
6  *****/
7  #include <iostream>
8  #include "binarytree.h"
9  using namespace std;
10
11 int TreeDataCmp(const linked_list& first, const linked_list& second)
12 {
13     if (first->line_num < second->line_num)
14         return -1;
15     if (first->line_num == second->line_num)
16         return 0;
17     if (first->line_num > second->line_num)
18         return 1;
19     return 0;
20 }
21
22 BinaryTree::BinaryTree()
23 {
24     root = NULL;
25 }
26 BinaryTree::~BinaryTree()
27 {
28     postorderdelete(root);
29 }
30 void BinaryTree::postorderdelete(TreeNode* curr)
31 {
32     if (curr != NULL)
33     {
34         postorderdelete(curr->left);
35         postorderdelete(curr->right);
36
37         delete curr;
38     }
39 }
40 void BinaryTree::SetIterator()
41 {
42     curr = root;
43     if (curr != NULL)
44         while (curr->left != NULL)
45             curr = curr->left;
46 }
47
48 linked_list BinaryTree::Next()
49 {
50     linked_list rvalue = curr->data;
51
52     if (curr->right != NULL)      // into the right subtree
53     {
54         curr = curr->right;
55         while (curr->left != NULL)
56             curr = curr->left;

```

Nov 09, 14 17:16

stdin

Page 3/25

Nov 9 11:52 2014 by932/binarytree.cpp Page 2

```

57     }
58     else if (curr->parent == NULL) // no parent
59         curr = NULL; // we are finished
60     else // find ancestor
61     {
62         while (curr->parent != NULL
63             && (curr->parent)->left != curr)
64             curr = curr->parent;
65         curr = curr->parent;
66     }
67     return rvalue;
68 }
69
70
71 bool BinaryTree::More()
72 {
73     if (curr != NULL)
74         return true;
75     else
76         return false;
77 }
78
79
80 void BinaryTree::Insert(const linked_list& data)
81 {
82     InsertNode(data, root);
83 }
84
85 bool BinaryTree::Locate(const linked_list& data, linked_list& founddata)
86 {
87     TreeNode* foundNode = FindNode(data, root);
88     if (foundNode != NULL)
89     {
90         founddata = foundNode->data;
91         return true;
92     }
93     else
94         return false;
95 }
96
97
98 bool BinaryTree::Delete(const linked_list& data, linked_list& deldata)
99 {
100     TreeNode *parent, *prev, *curr = FindNode(data, root);
101     if (curr == NULL)
102         return false;
103     deldata = curr->data;
104     parent = curr->parent;
105     if (curr->left == NULL) // no left subtree
106     {
107         if (curr->right == NULL) // no children
108         {
109             if (parent == NULL) // curr is root

```

Nov 09, 14 17:16

stdin

Page 4/25

Nov 9 11:52 2014 by932/binarytree.cpp Page 3

```

113         root = NULL;
114     else if (parent->left == curr)
115         parent->left = NULL;
116     else
117         parent->right = NULL;
118     }
119     else // only right subtree
120     {
121         if (parent == NULL)
122         {
123             root = curr->right;
124             root->parent = NULL;
125         }
126         else if (parent->left == curr)
127         {
128             parent->left = curr->right;
129             curr->right->parent = parent;
130         }
131         else
132         {
133             parent->right = curr->right;
134             curr->right->parent = parent;
135         }
136     }
137 }
138 else if (curr->right == NULL) // only left subtree
139 {
140     if (parent == NULL)
141     {
142         root = curr->left;
143         root->parent = NULL;
144     }
145     else if (parent->left == curr)
146     {
147         parent->left = curr->left;
148         curr->left->parent = parent;
149     }
150     else
151     {
152         parent->right = curr->left;
153         curr->left->parent = parent;
154     }
155 }
156 else // has both subtrees
157 {
158     prev = curr->left;
159     while (prev->right != NULL)
160         prev = prev->right;
161     prev->right = curr->right;
162     curr->right->parent = prev;
163     if (parent == NULL)
164     {
165         root = curr->left;
166         root->parent = NULL;
167     }
168     else

```

Nov 09, 14 17:16

stdin

Page 5/25

Nov 9 11:52 2014 by932/binarytree.cpp Page 4

```

169         {
170             parent->right = curr->left;
171             curr->left->parent = parent;
172         }
173     }
174     delete curr;
175     return true;
176 }
177
178
179 BinaryTree::TreeNode* BinaryTree::FindNode(const linked_list& data, BinaryTree::TreeNode* tree_root)
180 {
181     if (tree_root == NULL)
182     {
183         return 0;
184     }
185     int Result = TreeDataCmp(data, tree_root->data);
186
187     if (Result == 0)
188         return tree_root;
189     if (Result < 0 && tree_root->left != NULL)
190         return FindNode(data, tree_root->left);
191     else if (Result > 0 && tree_root->right != NULL)
192         return FindNode(data, tree_root->right);
193     return 0;
194 }
195
196
197 void BinaryTree::InsertNode(const linked_list& data, BinaryTree::TreeNode*& tree_root)
198 {
199     //insert root node
200     if (tree_root == NULL)
201     {
202         tree_root = new TreeNode;           //set up a new node
203         tree_root->data = data;
204         tree_root->left = NULL;
205         tree_root->right = NULL;
206         tree_root->parent = NULL;
207     }
208     else if (TreeDataCmp(data, tree_root->data) <= 0)
209     {
210         if (tree_root->left == NULL)
211         {
212             tree_root->left = new TreeNode;
213             tree_root->left->data = data;
214             tree_root->left->left = NULL;
215             tree_root->left->right = NULL;
216             tree_root->left->parent = tree_root;
217         }
218         else
219             InsertNode(data, tree_root->left);
220     }
221     else
222     {
223         if (tree_root->right == NULL)
224         {

```

Nov 09, 14 17:16

stdin

Page 6/25

Nov 9 11:52 2014 by932/binarytree.cpp Page 5

```
225             tree_root->right = new TreeNode;
226             tree_root->right->data = data;
227             tree_root->right->left = NULL;
228             tree_root->right->right = NULL;
229             tree_root->right->parent = tree_root;
230         }
231     else
232         InsertNode(data, tree_root->right);
233     }
234 }
235
```

Nov 09, 14 17:16

stdin

Page 7/25

Nov 9 11:52 2014 by932/linkedlist.cpp Page 1

```

1  /*****
2  *Filename:linkedlist.cpp      *
3  *Login:by932                 *
4  *AssignmentNo:ass5           *
5  *DateLastModified:2/11/2014  *
6  *****/
7  #include <iostream>
8  #include "linkedlist.h"
9  using namespace std;
10 int listdatacmp(const T& a, const T& b)          // user defined compare function
11 {
12     // return < 0 if a < b
13     // return == 0 if a == b
14     // return >0 if a > b;
15
16     return 0;
17 }
18
19 // iterator methods
20
21 void linkedlist::setiterator()
22 {
23     iterator_current = head;
24 }
25
26 // is there more in the iterator
27
28 bool linkedlist::more()
29 {
30     if (iterator_current != NULL)
31         return true;
32     else
33         return false;
34 }
35
36 // get next piece of data out of iterator
37
38 T linkedlist::next()
39 {
40     T tmp = iterator_current->data;
41     iterator_current = iterator_current->next;
42     return tmp;
43 }
44
45 void linkedlist::insertbeforecurrent(const T& newdata)          // insertion method
46 {
47     nodeptr tmp;
48     tmp = new node;
49     tmp->data = newdata;
50     tmp->next = NULL;
51
52     nodeptr curr;
53     nodeptr prev;
54     curr = head;
55     prev = NULL;
56

```

Nov 09, 14 17:16

stdin

Page 8/25

Nov 9 11:52 2014 by932/linkedlist.cpp Page 2

```

57         while (curr != iterator_current)
58         {
59             prev = curr;
60             curr = curr->next;
61         }
62
63         if (prev == NULL)
64             head = tmp;
65         else
66             prev->next = tmp;
67         tmp->next = curr;
68     }
69
70     bool linkedlist::deletecurrent(T& retdata)
71     {
72         nodeptr curr;
73         nodeptr prev;
74         curr = head;
75         prev = NULL;
76
77         if (iterator_current == NULL)
78             return false;
79
80         while (curr != iterator_current)           // find it
81         {
82             prev = curr;
83             curr = curr->next;
84         }
85
86         if (curr == NULL)
87         {
88             return false;
89         }
90
91         if (prev)
92         {
93             prev->next = curr->next;
94             retdata = curr->data;
95             delete curr;
96             return true;
97         }
98         else
99         {
100             head = curr->next;
101             retdata = curr->data;
102             delete curr;
103             return true;
104         }
105     }
106
107     // constructor
108     linkedlist::linkedlist()
109     {
110         head = NULL;
111     }
112     //destructor

```



Nov 09, 14 17:16

stdin

Page 9/25

Nov 9 11:52 2014 by932/linkedlist.cpp Page 3

```

113     linkedlist::~~linkedlist()
114     {
115         node* temp = head;
116         while(temp && temp->next)
117         {
118             temp=temp->next;
119             delete temp;    //delet linked-list
120         }
121     }
122     // add to tail method
123     void linkedlist::addtotail(char token[], int line_no)
124     {
125         nodeptr tmp, curr;
126         tmp = new node;
127         tmp->data.content = token;
128         tmp->data.line_num = line_no;
129         tmp->next = NULL;
130
131         if (head != NULL)
132         {
133             curr = head;
134             while (curr->next)
135                 curr = curr->next;
136             curr->next = tmp;
137         }
138         else
139             head = tmp;
140     }
141
142     // check if list is empty method
143     bool linkedlist::isempty()
144     {
145         if (head == NULL)
146             return true;
147         else
148             return false;
149     }
150
151     // remove from head method
152     T linkedlist::removefromhead()
153     {
154         nodeptr tmp;
155         T data;
156
157         data = head->data;
158         tmp = head;
159         head = head->next;
160         delete tmp;
161         return data;
162     }
163
164     void linkedlist::insert(const T& newdata)                // insertion method
165     {
166         nodeptr tmp;
167         tmp = new node;
168         tmp->data = newdata;

```

Nov 09, 14 17:16

stdin

Page 10/25

Nov 9 11:52 2014 by932/linkedlist.cpp Page 4

```

169         tmp->next = NULL;
170
171         nodeptr curr;
172         nodeptr prev;
173         curr = head;
174         prev = NULL;
175
176         while (curr && listdatacmp(newdata, curr->data) >= 0)
177         {
178             prev = curr;
179             curr = curr->next;
180         }
181
182         if (prev == NULL)
183             head = tmp;
184         else
185             prev->next = tmp;
186         tmp->next = curr;
187
188     }
189 }
190
191 bool linkedlist::locate(const T& keydata, T& retresult)           // locate method
192 {
193     nodeptr curr = head;
194
195     while (curr && listdatacmp(keydata, curr->data) != 0)
196     {
197         curr = curr->next;
198     }
199
200     if (curr == NULL)
201         return false;
202     else
203     {
204         retresult = curr->data;
205         return true;
206     }
207 }
208
209 bool linkedlist::delete_node(const T& keydata, T& retdata)       // delete node method
210 {
211     nodeptr curr;
212     nodeptr prev;
213     curr = head;
214     prev = NULL;
215
216     while (curr != NULL && listdatacmp(keydata, curr->data) != 0)
217     {
218         prev = curr;
219         curr = curr->next;
220     }
221
222     if (curr == NULL)
223     {
224         return false;

```

Nov 09, 14 17:16

stdin

Page 11/25

Nov 9 11:52 2014 by932/linkedlist.cpp Page 5

```
225         }
226
227         if (prev)
228         {
229             prev->next = curr->next;
230             retdata = curr->data;
231             delete curr;
232             return true;
233         }
234         else
235         {
236             head = curr->next;
237             retdata = curr->data;
238             delete curr;
239             return true;
240         }
241     }
242     bool linkedlist::print(ostream&)
243     {
244         if (head == NULL)
245         {
246             return false;
247         }
248         nodeptr temp = head;                // do not change the head pointer
249         while (temp != NULL)
250         {
251             cout << "token = " << temp->data.content << '\t' << "line_num = " << temp->data.line_num;
252             cout << endl;
253             temp = temp->next;
254         }
255         return true;
256     }
257 }
```

Nov 09, 14 17:16

stdin

Page 12/25

Nov 9 11:52 2014 by932/main.cpp Page 1

```
1  /*****
2  *Filename:main.cpp          *
3  *Login:by932               *
4  *AssignmentNo:ass5         *
5  *DateLastModified:2/11/2014 *
6  *****/
7  #include <iostream>
8  #include "program.h"
9  using namespace std;
10
11  int main()
12  {
13      program test_program_class;
14      LIST test_list_class;
15
16      test_program_class.getline(test_list_class);
17      test_list_class.print(cout);
18      return 0;
19  }
```

Nov 09, 14 17:16

stdin

Page 13/25

Nov 9 11:52 2014 by932/program-list.cpp Page 1

```

1  /*****
2  *Filename:program-list.cpp      *
3  *Login:by932                   *
4  *AssignmentNo:ass5             *
5  *DateLastModified:2/11/2014    *
6  *****/
7  #include <iostream>
8  #include "program-list.h"
9  using namespace std;
10 // constructor
11 LIST::LIST()
12 {
13     head = NULL;
14 }
15
16 //destructor
17 LIST::~LIST()
18 {
19     node* temp = head;
20     while(temp && temp->next)
21     {
22         temp=temp->next;
23         delete temp;    //delet linked-list
24     }
25 }
26
27 bool LIST::load(char token[], int line)
28 {
29     if (head == NULL)        // linked list is empty - so this will be the head node
30     {
31         NPtr newnode = new node;
32
33         if (newnode == NULL)    // could not allocate memory
34         {
35             cout << "Allocation error occured" << endl;
36             return false;
37         }
38         strcpy(newnode->content, token);
39         newnode->line = line;
40         newnode->next = NULL;
41         head = newnode;
42     }
43     else
44     {
45         // if not here then the linked list exists
46         NPtr check = head;
47         while (check != NULL)
48         {
49             if (check->next == NULL)
50             {
51                 NPtr newnode = new node;
52                 if (newnode == NULL)
53                 {
54

```

Nov 09, 14 17:16

stdin

Page 14/25

Nov 9 11:52 2014 by932/program-list.cpp Page 2

```
57             cout << "Allocation error occurred" << endl;
58             return false;
59         }
60         strcpy(newnode->content, token);           //store token
61         newnode->line = line;                       //give the line number
62         newnode->next = NULL;                      //new node next node = null
63         check->next = newnode;
64         break;
65     }
66     check = check->next;
67 }
68 }
69 }
70 return true;
71 }
72 }
73
74 bool LIST::print(ostream&)
75 {
76     cout << "Start output Part 1" << endl;
77     if (head == NULL)
78     {
79         return false;
80     }
81     NPtr temp = head;                             // do not change the head pointer
82     while (temp != NULL)
83     {
84         //cout << "token = " << temp->content << '\t' << "line_num = " << temp->line;
85         //cout << endl;
86         temp = temp->next;
87     }
88     cout << "End output Part 1" << endl;
89     return true;
90 }
91 }
```

Nov 09, 14 17:16

stdin

Page 15/25

Nov 9 11:52 2014 by932/program.cpp Page 1

```

1  /*****
2  *Filename:program.cpp          *
3  *Login:by932                  *
4  *AssignmentNo:ass5            *
5  *DateLastModified:2/11/2014   *
6  *****/
7  #include <iostream>
8  #include <fstream>
9  #include <cstring>
10 #include "program.h"
11 #include "binarytree.h"
12 using namespace std;
13 //Global Data
14 BinaryTree WordTree;
15
16 bool bonus(char []);//delete 62 reserved words
17 program::program()
18 {
19     line_no = 1;    //init line_no
20
21 }
22 program::~program()
23 {
24     fin.close();    //close the file
25 }
26
27 bool program::open(char source_file[])
28 {
29
30     fin.open(source_file); //open file
31
32     if(!fin)
33     {
34         cout << "Cann't find file"<< endl;
35         return false;    // open file false
36     }
37     return true;    //open file good
38
39 }
40
41 bool program::getline(LIST& obj)
42 {
43     char fname[100], token[40], ch;
44     int line_pos, carryon, i;
45
46     cout << "Enter a filename: ";
47     cin >> fname;
48
49     bool good_or_bad = true;    //open file status good or bad, true is good
50     good_or_bad = open(fname);
51
52     if (good_or_bad == true)
53     {
54         line_pos = 0;
55         token[0] = 0;    // initialise token
56

```

Nov 09, 14 17:16

stdin

Page 16/25

Nov 9 11:52 2014 by932/program.cpp Page 2

```

57         ch = fin.get();
58         while (!fin.eof())
59         {
60             //Eating all comments
61             if (ch == '/')
62             {
63                 ch = fin.get();
64                 line_pos++;
65                 /*"/*"type comment - eat both characters and all to newline
66                 if (ch == '/')
67                 {
68                     do
69                     {
70                         ch = fin.get();
71                     } while (ch != '\n');
72                     line_no++;
73                     line_pos = 0;
74                     ch = fin.get(); // now positioned at start of next line
75                 }
76                 //old type comment/* */ - eat both characters and all to end
77                 else if (ch == '*')
78                 {
79                     carryon = 1;
80                     ch = fin.get();
81                     while (carryon)
82                     {
83                         if (ch == '\n')
84                         {
85                             line_no++;
86                             line_pos = 0;
87                             ch = fin.get();
88                         }
89                         else if (ch == '*')
90                         {
91                             ch = fin.get();
92                             if (ch == '/')
93                                 carryon = 0;
94                         }
95                         else
96                             ch = fin.get();
97                     }
98                     ch = fin.get();
99                     line_pos = 1;
100                 }
101             }
102             else if (ch == '"')
103             {
104                 //character string constant - eat all characters to other end
105                 //avoiding \anything
106                 ch = fin.get();
107                 while (ch != '"')
108                 {
109                     if (ch == '\\')
110                         ch = fin.get();
111                     ch = fin.get();
112                 }

```



Nov 09, 14 17:16

stdin

Page 17/25

Nov 9 11:52 2014 by932/program.cpp Page 3

```

113         ch = fin.get();
114     }
115     else if (ch == '\\')
116     {
117         //character constant - as for character string
118         ch = fin.get();
119         while (ch != '\\')
120         {
121             if (ch == '\\\\')
122                 ch = fin.get();
123             ch = fin.get();
124         }
125         ch = fin.get();
126     }
127     else if (line_pos == 0 && ch == '#')
128     {
129         // # pre-processor line - eat # and all to newline
130         do
131         {
132             ch = fin.get();
133         } while (ch != '\n');
134         ch = fin.get();
135         line_no++;
136         line_pos = 0;
137     }
138     else if (isalpha(ch)) // start of token
139     {
140         i = 0;
141         do
142         {
143             token[i++] = ch;
144             ch = fin.get();
145         } while (isalnum(ch) || ch == '_');
146         token[i] = 0; //token end
147         bool jud = true;
148         jud = bonus(token); //delete 62 reserved words
149         if (jud == false)
150             continue;
151         //This is part 1
152         obj.load(token, line_no); //store token and line number to the list
153
154         //This is part 2
155         bnode *temp = new bnode;
156         temp->content = token;
157         temp->line_num = line_no;
158         temp->data.addtotail(token, line_no);
159         linked_list foundData;
160
161         if (WordTree.Locate(temp, foundData))
162         {
163             //This part is store same line token to the linked-list
164             foundData->line_num = line_no;
165             foundData->data.addtotail(token, line_no);
166             delete temp;
167         }
168     }

```

Nov 09, 14 17:16

stdin

Page 18/25

Nov 9 11:52 2014 by932/program.cpp Page 4

```

169         else
170         {
171             //This is set up a new tree node
172             WordTree.Insert(temp);
173         }
174
175
176         token[0] = 0;    // initialise token
177     }
178     else if (ch == '\n')    // just a newline
179     {
180         line_no++;
181         line_pos = 0;
182         ch = fin.get();
183     }
184     else
185         ch = fin.get();
186     }
187 }
188 return true;
189
190 }
191 //this is part 3
192 bool bonus(char token[])
193 {
194     if (strcmp(token, "asm") == 0 ||
195         strcmp(token, "casecatch") == 0 ||
196         strcmp(token, "const_cast") == 0 ||
197         strcmp(token, "do") == 0 ||
198         strcmp(token, "enum") == 0 ||
199         strcmp(token, "float") == 0 ||
200         strcmp(token, "if") == 0 ||
201         strcmp(token, "mutable") == 0 ||
202         strcmp(token, "private") == 0 ||
203         strcmp(token, "reinterpret_cast") == 0 ||
204         strcmp(token, "sizeof") == 0 ||
205         strcmp(token, "switch") == 0 ||
206         strcmp(token, "true") == 0 ||
207         strcmp(token, "typename") == 0 ||
208         strcmp(token, "virtual") == 0 ||
209         strcmp(token, "while") == 0 ||
210         strcmp(token, "auto") == 0 ||
211         strcmp(token, "char") == 0 ||
212         strcmp(token, "continue") == 0 ||
213         strcmp(token, "double") == 0 ||
214         strcmp(token, "explicit") == 0 ||
215         strcmp(token, "for") == 0 ||
216         strcmp(token, "inline") == 0 ||
217         strcmp(token, "namespace") == 0 ||
218         strcmp(token, "protected") == 0 ||
219         strcmp(token, "return") == 0 ||
220         strcmp(token, "static") == 0 ||
221         strcmp(token, "template") == 0 ||
222         strcmp(token, "try") == 0 ||
223         strcmp(token, "union") == 0 ||
224         strcmp(token, "void") == 0 ||

```

Nov 09, 14 17:16

stdin

Page 19/25

Nov 9 11:52 2014 by932/program.cpp Page 5

```
225     strcmp(token, "bool") == 0 ||
226     strcmp(token, "class") == 0 ||
227     strcmp(token, "default") == 0 ||
228     strcmp(token, "dynamic_cast") == 0 ||
229     strcmp(token, "extern") == 0 ||
230     strcmp(token, "friend") == 0 ||
231     strcmp(token, "int") == 0 ||
232     strcmp(token, "new") == 0 ||
233     strcmp(token, "public") == 0 ||
234     strcmp(token, "short") == 0 ||
235     strcmp(token, "static_cast") == 0 ||
236     strcmp(token, "this") == 0 ||
237     strcmp(token, "typedef") == 0 ||
238     strcmp(token, "unsigned") == 0 ||
239     strcmp(token, "volatile") == 0 ||
240     strcmp(token, "break") == 0 ||
241     strcmp(token, "const") == 0 ||
242     strcmp(token, "delete") == 0 ||
243     strcmp(token, "else") == 0 ||
244     strcmp(token, "false") == 0 ||
245     strcmp(token, "goto") == 0 ||
246     strcmp(token, "long") == 0 ||
247     strcmp(token, "operator") == 0 ||
248     strcmp(token, "register") == 0 ||
249     strcmp(token, "signed") == 0 ||
250     strcmp(token, "struct") == 0 ||
251     strcmp(token, "throw") == 0 ||
252     strcmp(token, "typeid") == 0 ||
253     strcmp(token, "using") == 0 ||
254     strcmp(token, "wchar_t") == 0)
255     return false;
256
257     return true;
258
259 }
```

Nov 09, 14 17:16

stdin

Page 20/25

Nov 9 11:52 2014 by932/binarytree.h Page 1

```

1  /*****
2  *Filename:binarytree.h      *
3  *Login:by932                *
4  *AssignmentNo:ass5          *
5  *DateLastModified:2/11/2014 *
6  *****/
7  #include <iostream>
8  #include <cstring>
9  #include "linkedlist.h"
10 using namespace std;
11 // Definition of data contained in BinaryTree
12 struct bnode
13 {
14     linkedlist data;          //store token
15     int line_num;             //story line number
16     char *content;            //token content[40];
17 };
18
19 typedef bnode * linked_listPtr;
20
21 typedef linked_listPtr linked_list;
22
23
24 class BinaryTree
25 {
26     public:
27         BinaryTree();//constructor
28         ~BinaryTree();// destructor
29         void Insert(const linked_list&);
30         bool Locate(const linked_list&, linked_list&);
31         bool Delete(const linked_list&, linked_list&);
32
33         // iterator methods
34         void SetIterator();          // find left most node
35         linked_list Next();          // return next data item
36         bool More();                 // are there any more nodes?
37
38     private:
39         struct TreeNode
40         {
41             linked_list data;
42             TreeNode *left, *right, *parent;
43         };
44
45         TreeNode* root;
46         TreeNode* curr;              // used by iterator
47
48         TreeNode* FindNode(const linked_list&, TreeNode*);
49         void InsertNode(const linked_list&, TreeNode*&);
50         //this function is given the root node and will do a post order deletion of the nodes
51         void postorderdelete(TreeNode*);
52     };
53
54

```

Nov 09, 14 17:16

stdin

Page 21/25

Nov 9 11:52 2014 by932/linkedlist.h Page 1

```

1  /*****
2  *Filename:linkedlist.h      *
3  *Login:by932               *
4  *AssignmentNo:ass5         *
5  *DateLastModified:2/11/2014 *
6  *****/
7  #include <iostream>
8  using namespace std;
9
10 struct linkedlist_node
11 {
12     char *content;
13     int line_num;    //story line number
14 };
15
16 typedef linkedlist_node T;
17
18
19
20 class linkedlist
21 {
22     public: // public methods
23         linkedlist();
24         ~linkedlist();
25         // the destructor has been removed - when you need to delete you must do by hand
26         void addtotail(char [], int);
27         bool print(ostream&);    //Print out the linked-list
28
29
30         bool isempty();
31         T removefromhead();
32         void insert(const T&);
33         bool locate(const T&, T&);
34         bool delete_node(const T&, T&);
35
36         bool deletecurrent(T&);
37         void insertbeforecurrent(const T&);
38         void setiterator();
39         T next();
40         bool more();
41     private: // node data and declaration - hidden in class
42         struct node;
43         typedef node* nodeptr;
44         nodeptr iterator_current;
45         struct node
46         {
47             T data;
48             nodeptr next;
49         };
50         nodeptr head;
51     };
52

```

Nov 09, 14 17:16

stdin

Page 22/25

Nov 9 11:52 2014 by932/program-list.h Page 1

```
1  /*****
2  *Filename:program-list.h      *
3  *Login:by932                  *
4  *AssignmentNo:ass5           *
5  *DateLastModified:2/11/2014  *
6  *****/
7  #include <iostream>
8  #include <cstring>
9  using namespace std;
10 struct node
11 {
12     char content[40];
13     int line;
14     node *next;
15 };
16 typedef node* NPtr;
17 class LIST
18 {
19     public:
20         LIST();
21         ~LIST();
22         bool load(char[], int);
23         bool print(ostream&);
24     private:
25         NPtr head;    // pointer to list of chars
26         NPtr next;
27 };
```

Nov 09, 14 17:16

stdin

Page 23/25

Nov 9 11:52 2014 by932/program.h Page 1

```
1  /*****
2  *Filename:program.h          *
3  *Login:by932                *
4  *AssignmentNo:ass5          *
5  *DateLastModified:2/11/2014 *
6  *****/
7  #include <iostream>
8  #include <fstream>
9  #include <cstring>
10 #include "program-list.h"
11 using namespace std;
12
13 class program
14 {
15     public:
16
17         program();
18         ~program();
19         bool open(char[]);          // open a file return state i.e. success
20         bool getline(LIST&);
21
22     private:
23         ifstream fin;
24         int line_no;
25 };
26
27
```

Nov 09, 14 17:16

**stdin**

Page 24/25



Nov 09, 14 17:16

stdin

Page 25/25

```
Enter a filename: Start output Part 1  
End output Part 1
```