phylotoy

0

Generated by Doxygen 1.8.13

Contents

1 Class Index **1**

1.1 Class List 1

2 File Index **3**

2.1 File List 3

3 Class Documentation **5**

3.1 Controller Class Reference 5

3.1.1 Detailed Description 5

3.1.2 Constructor & Destructor Documentation 5

3.1.2.1 Controller() 5

3.1.3 Member Function Documentation 6

3.1.3.1 CheckAlignmentFilePath() 6

3.1.3.2 CheckChainName() 6

3.1.3.3 CheckCLIOptions() 7

3.1.3.4 CheckRandomSeed() 7

3.1.3.5 GetAlignmentFilePath() 7

3.1.3.6 GetChainName() 8

3.1.3.7 GetRandomSeed() 8

3.1.3.8 Run() 8

3.1.3.9 SetAlignmentFilePath() 9

3.1.3.10 SetChainName() 9

3.1.3.11 SetRandomSeed() 10

3.2 InputReader Class Reference 10

3.2.1 Detailed Description . . . 10
3.2.2 Constructor & Destructor Documentation . . . 10
3.2.2.1 InputReader() [1/2] . . . 10
3.2.2.2 InputReader() [2/2] . . . 11
3.2.3 Member Function Documentation . . . 11
3.2.3.1 GetPath() . . . 11
3.2.3.2 ReadInputFile() . . . 11
3.2.3.3 SetPath() . . . 12
3.3 Node Class Reference . . . 12
3.3.1 Detailed Description . . . 13
3.3.2 Constructor & Destructor Documentation . . . 13
3.3.2.1 Node() . . . 13
3.3.3 Member Function Documentation . . . 13
3.3.3.1 AddNodeToChildVector() . . . 13
3.3.3.2 CreateBifurcatingNode() . . . 13
3.3.3.3 GetChildVector() . . . 14
3.3.3.4 GetIndex() . . . 15
3.3.3.5 GetIsTip() . . . 15
3.3.3.6 GetLengthSubtendingBranch() . . . 15
3.3.3.7 GetNodeInfo() . . . 16
3.3.3.8 GetNodePointer() . . . 16
3.3.3.9 GetParentNode() . . . 17
3.3.3.10 GetSequence() . . . 17
3.3.3.11 GetSpeciesName() . . . 17
3.3.3.12 SetChildVector() . . . 18
3.3.3.13 SetIndex() . . . 18
3.3.3.14 SetIsTip() . . . 18
3.3.3.15 SetLengthSubtendingBranch() . . . 18
3.3.3.16 SetParentNode() . . . 19
3.3.3.17 SetSequence() . . . 19

3.3.3.18 SetSpeciesName() . . . 19
3.4 OutputPrinter Class Reference . . . 19
3.4.1 Detailed Description . . . 20
3.4.2 Constructor & Destructor Documentation . . . 20
3.4.2.1 OutputPrinter() . . . 20
3.4.3 Member Function Documentation . . . 20
3.4.3.1 PrintMessage2Out() . . . 20
3.5 Tree Class Reference . . . 20
3.5.1 Detailed Description . . . 21
3.5.2 Constructor & Destructor Documentation . . . 21
3.5.2.1 Tree() . . . 21
3.5.3 Member Function Documentation . . . 21
3.5.3.1 AddToNodeVector() . . . 22
3.5.3.2 CollectTreeNodesInfoIteratively() . . . 22
3.5.3.3 CollectTreeNodesInfoRecursively() . . . 23
3.5.3.4 CreateBifurcatingTree() . . . 23
3.5.3.5 CreateStarTree() . . . 24
3.5.3.6 GetLength() . . . 24
3.5.3.7 GetRoot() . . . 25
3.5.3.8 GetTreeNodes() . . . 25
3.5.3.9 GetTreeNodeVector() . . . 25
3.5.3.10 SetLength() . . . 25
3.5.3.11 SetRoot() . . . 26
3.5.3.12 SetTreeNodes() . . . 26

4 File Documentation **27**

4.1 /home/sergio/Repos/phylotoy/src/Controller.cpp File Reference . . . 27
4.2 /home/sergio/Repos/phylotoy/src/Controller.h File Reference . . . 27
4.3 /home/sergio/Repos/phylotoy/src/InputReader.cpp File Reference . . . 27
4.4 /home/sergio/Repos/phylotoy/src/InputReader.h File Reference . . . 28
4.5 /home/sergio/Repos/phylotoy/src/Node.cpp File Reference . . . 28
4.6 /home/sergio/Repos/phylotoy/src/Node.h File Reference . . . 28
4.7 /home/sergio/Repos/phylotoy/src/OutputPrinter.cpp File Reference . . . 28
4.8 /home/sergio/Repos/phylotoy/src/OutputPrinter.h File Reference . . . 28
4.9 /home/sergio/Repos/phylotoy/src/Phylotoy.cpp File Reference . . . 29
4.9.1 Function Documentation . . . 29
4.9.1.1 main() . . . 29
4.10 /home/sergio/Repos/phylotoy/src/Tree.cpp File Reference . . . 30
4.11 /home/sergio/Repos/phylotoy/src/Tree.h File Reference . . . 30

Index **31**

Chapter 1

Class Index

1.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Controller 5
InputReader 10
Node 12
OutputPrinter 19
Tree 20

Chapter 2

File Index

2.1 File List

Here is a list of all files with brief descriptions:

/home/sergio/Repos/phylotoy/src/Controller.cpp 27
/home/sergio/Repos/phylotoy/src/Controller.h 27
/home/sergio/Repos/phylotoy/src/InputReader.cpp 27
/home/sergio/Repos/phylotoy/src/InputReader.h 28
/home/sergio/Repos/phylotoy/src/Node.cpp 28
/home/sergio/Repos/phylotoy/src/Node.h 28
/home/sergio/Repos/phylotoy/src/OutputPrinter.cpp 28
/home/sergio/Repos/phylotoy/src/OutputPrinter.h 28
/home/sergio/Repos/phylotoy/src/Phylotoy.cpp 29
/home/sergio/Repos/phylotoy/src/Tree.cpp 30
/home/sergio/Repos/phylotoy/src/Tree.h 30

Chapter 3

Class Documentation

3.1 Controller Class Reference

`#include <Controller.h>`

Public Member Functions

- Controller ()
- void SetRandomSeed (int seed)
- int GetRandomSeed ()
- int CheckRandomSeed ()
- void SetAlignmentFilePath (std::string path)
- std::string GetAlignmentFilePath ()
- std::string CheckAlignmentFilePath ()
- void SetChainName (std::string name)
- std::string GetChainName ()
- std::string CheckChainName ()
- void CheckCLIOptions ()
- void Run ()

3.1.1 Detailed Description

Definition at line 14 of file Controller.h.

3.1.2 Constructor & Destructor Documentation

3.1.2.1 Controller()

`Controller::Controller ( )`

Definition at line 14 of file Controller.cpp.

```
14                      {
15
16
17
18 }
```

3.1.3 Member Function Documentation

3.1.3.1 CheckAlignmentFilePath()

```
std::string Controller::CheckAlignmentFilePath ( )
```

Definition at line 59 of file Controller.cpp.

```
59                                              {
60
61   //if the user forgot to set the ali path this char* will be null and we need to throw an exception
62   if (!alignment_file_path.length()){
63
64     throw "Alignment file path (option -i) was not set but is required";
65
66   }else{
67
68     return alignment_file_path;
69   }
70
71 }
```

3.1.3.2 CheckChainName()

```
std::string Controller::CheckChainName ( )
```

Definition at line 86 of file Controller.cpp.

```
86                                          {
87
88   //if the user forgot to set the chain name this char* will be null and we need to throw an exception
89   if (!chain_name.length()){
90
91     throw "Chain name (option -c) was not set but is required";
92
93   }else{
94
95     return chain_name;
96
97   }
98
99 }
```

3.1.3.3 CheckCLIOptions()

```
void Controller::CheckCLIOptions ( )
```

Definition at line 101 of file Controller.cpp.

```
101                                {
102
103   /*We need to check whether the necessary options were set*/
104
105   try {
106
107     this->CheckAlignmentFilePath();
108     this->CheckChainName();
109     this->CheckRandomSeed();
110
111   } catch (const char* exception) {
112
113     std::string what_exception(exception);
114     std::string error = "Error: " + what_exception + "\n";
115     output_printer.PrintMessage2Out(error);
116
117     exit(1);
118
119   }
120
121 }
```

3.1.3.4 CheckRandomSeed()

```
int Controller::CheckRandomSeed ( )
```

Definition at line 33 of file Controller.cpp.

```
33                                {
34
35   //if the user forgot to set the random seed this int will be null and we need to throw an exception
36   if (random_seed <= 0){
37
38     throw "Random seed (option -r) was not set but is required";
39
40   }else{
41
42     return random_seed;
43   }
44
45 }
```

3.1.3.5 GetAlignmentFilePath()

```
std::string Controller::GetAlignmentFilePath ( )
```

Definition at line 53 of file Controller.cpp.

```
53                                          {
54
55     return alignment_file_path;
56
57 }
```

3.1.3.6 GetChainName()

```
std::string Controller::GetChainName ( )
```

Definition at line 80 of file Controller.cpp.

```
80                                    {
81
82     return chain_name;
83
84 }
```

3.1.3.7 GetRandomSeed()

```
int Controller::GetRandomSeed ( )
```

Definition at line 26 of file Controller.cpp.

```
26                                {
27
28   return random_seed;
29
30
31 }
```

3.1.3.8 Run()

```
void Controller::Run ( )
```

Definition at line 124 of file Controller.cpp.

```
124                       {
125
126   this->CheckCLIOptions();
127
128   //if all option are set, we tell the user how the program was invoked.
129   std::string program_call = "phylotoy was invoked with the following options:\n\n\tRandom seed = " +
      std::to_string(random_seed) + "\n\tAlingment path = " + alignment_file_path + "\n\tChain name = " + chain_name +
      "\n\n";
130
131   output_printer.PrintMessage2Out(program_call);
132
133   //tmp vector of strings
134   std::vector<std::string>* alignment;
135
136   output_printer.PrintMessage2Out("reading alignment\n");
137   //now we need to open and store the Alignment in a string vector
138   alignment = input_reader.ReadInputFile(alignment_file_path);
139
140
141   output_printer.PrintMessage2Out("initializing tree\n");
142
143   phylo_tree.CreateBifurcatingTree(alignment);
144
145   //phylo_tree.CreateStarTree(alignment);
146
147   output_printer.PrintMessage2Out("collecting node info recursively\n");
148   std::vector<std::string>* nodes_info_recursively = phylo_tree.
      CollectTreeNodesInfoRecursively();
149
```

```
150   while(!nodes_info_recursively->empty()) {
151
152     output_printer.PrintMessage2Out("in while\n");
153     output_printer.PrintMessage2Out(nodes_info_recursively->back());
154     output_printer.PrintMessage2Out("\n");
155
156     nodes_info_recursively->pop_back();
157   }
158
159   output_printer.PrintMessage2Out("collecting node info iteratively\n");
160   std::vector<std::string>* nodes_info_iteratively = phylo_tree.
      CollectTreeNodesInfoIteratively();
161
162   while(!nodes_info_iteratively->empty()) {
163
164     output_printer.PrintMessage2Out("in while\n");
165     output_printer.PrintMessage2Out(nodes_info_iteratively->back());
166     output_printer.PrintMessage2Out("\n");
167
168     nodes_info_iteratively->pop_back();
169   }
170
171 }
```

3.1.3.9 SetAlignmentFilePath()

```
void Controller::SetAlignmentFilePath (
              std::string path )
```

Definition at line 47 of file Controller.cpp.

```
47                                                    {
48
49   alignment_file_path = path;
50
51 }
```

3.1.3.10 SetChainName()

```
void Controller::SetChainName (
              std::string name )
```

Definition at line 74 of file Controller.cpp.

```
74                                              {
75
76   chain_name = name;
77
78 }
```

3.1.3.11 SetRandomSeed()

```
void Controller::SetRandomSeed (
            int seed )
```

Definition at line 20 of file Controller.cpp.

```
20                                      {
21
22   random_seed = seed;
23
24 }
```

The documentation for this class was generated from the following files:

- /home/sergio/Repos/phylotoy/src/Controller.h
- /home/sergio/Repos/phylotoy/src/Controller.cpp

3.2 InputReader Class Reference

```
#include <InputReader.h>
```

Public Member Functions

- InputReader ()
- InputReader (std::string path)
- void SetPath (std::string path)
- std::string GetPath ()
- std::vector< std::string > * ReadInputFile (std::string path)

3.2.1 Detailed Description

Definition at line 11 of file InputReader.h.

3.2.2 Constructor & Destructor Documentation

3.2.2.1 InputReader() [1/2]

```
InputReader::InputReader ( )
```

Definition at line 13 of file InputReader.cpp.

```
13 {}
```

3.2.2.2 InputReader() [2/2]

```
InputReader::InputReader (
            std::string path )
```

Definition at line 15 of file InputReader.cpp.

```
15                                      {
16
17   input_file_path = path;
18
19 }
```

3.2.3 Member Function Documentation

3.2.3.1 GetPath()

```
std::string InputReader::GetPath ( )
```

Definition at line 27 of file InputReader.cpp.

```
27                                {
28
29   return input_file_path;
30
31 }
```

3.2.3.2 ReadInputFile()

```
std::vector< std::string > * InputReader::ReadInputFile (
            std::string path )
```

Definition at line 33 of file InputReader.cpp.

```
33                                                              {
34
35   /* create and open the input stream. This will always be done, we need to control later if the stream
36    * is good or not...
37   */
38
39   std::ifstream input_stream(path);
40
41   //create a vector of strings to store the data
42   std::vector<std::string>* lines = new std::vector<std::string>;
43
44   if(input_stream.good()) {
45
46     //dummy string to store the first line
47     std::string first_line;
48
49     //read the first line
50     std::getline(input_stream, first_line);
51
52     //string to store the lines.
53     std::string input_line;
54
55     //now read the file line by line and push the line into the vector
```

```
56     while(std::getline(input_stream, input_line)) {
57
58       //add the lines to the string vector
59       lines->push_back(input_line);
60
61     }
62
63     //input_stream.close();
64
65   }else {
66
67     std::cerr << "Something went wrong reading the alignment file: " << path << "\n";
68     exit(1);
69
70   }
71
72
73   return lines;
74
75 }
```

3.2.3.3 SetPath()

```
void InputReader::SetPath (
              std::string path )
```

Definition at line 21 of file InputReader.cpp.

```
21                                        {
22
23   input_file_path = path;
24
25 }
```

The documentation for this class was generated from the following files:

- /home/sergio/Repos/phylotoy/src/InputReader.h
- /home/sergio/Repos/phylotoy/src/InputReader.cpp

3.3 Node Class Reference

`#include <Node.h>`

Public Member Functions

- Node ()
- void SetSequence (std::string species_sequence)
- std::string GetSequence ()
- void SetSpeciesName (std::string name)
- std::string GetSpeciesName ()
- void SetIndex (int node_index)
- int GetIndex ()
- void SetIsTip (bool tip)
- bool GetIsTip ()
- void SetParentNode (Node *parent)
- Node * GetParentNode ()
- void SetLengthSubtendingBranch (float branch_length)
- float GetLengthSubtendingBranch ()
- void AddNodeToChildVector (Node *child)
- void SetChildVector (std::vector< Node *> childs)
- std::vector< Node * > GetChildVector ()
- void CreateBifurcatingNode (std::vector< Node *>, int *calls, std::vector< Node *> *tree_nodes)
- void GetNodePointer (std::vector< Node *> *tree_nodes)
- std::vector< std::string > * GetNodeInfo (std::vector< std::string > *collected_node_info)

3.3.1 Detailed Description

Definition at line 11 of file Node.h.

3.3.2 Constructor & Destructor Documentation

3.3.2.1 Node()

```
Node::Node (
            void  )
```

Definition at line 14 of file Node.cpp.

```
14 {}
```

3.3.3 Member Function Documentation

3.3.3.1 AddNodeToChildVector()

```
void Node::AddNodeToChildVector (
            Node * child )
```

Definition at line 92 of file Node.cpp.

```
92                                          {
93
94   child_nodes.push_back(child);
95
96 }
```

3.3.3.2 CreateBifurcatingNode()

```
void Node::CreateBifurcatingNode (
            std::vector< Node *> tip_nodes,
            int * current_node_index,
            std::vector< Node *> * tree_nodes )
```

We create the tree as follows: the current node has no children, i.e. the size of the vector child_nodes is 0. We insert the tip node as a child of this node, sending a reference of it to the child node for it to have a pointer to its parent node. Once this is done, we create a new internode, set this node as its parent and add it to the child_nodes vector of the current node. Then we recursively call this method on the new internode but only if we still have more than 1 tip nodes left. If not, we add the last tip node to the current internode.

After this happens the vector of tip nodes should be empty and the method returns.

Parameters

tip_nodes	a std::vector of Node pointers containing the tip nodes
current_node_index	an int pointer to be used to allocate each node with an index.
tree_nodes	a pointer to a std:vector of Node pointers that contains all nodes in a tree.

if only one last tip node is left. We can add it to the current internode. This internode should only have 1 descendent tip.

Definition at line 128 of file Node.cpp.

```
128
          {
129
130   std::cerr << "creating bifurcating node " << tip_nodes.size() << "\n";
131
132   if(!tip_nodes.empty()){
133
134     Node* tip_node_to_insert = tip_nodes.back();
135     tip_nodes.pop_back();//this deletes the last element of the array
136
137     std::cerr << "inserting tip: " << tip_node_to_insert->GetSpeciesName() << "\n";
138     tip_node_to_insert->SetParentNode(this);
139     this->AddNodeToChildVector(tip_node_to_insert);
140     tree_nodes->push_back(tip_node_to_insert);
141
142     if(tip_nodes.size() == 1){
143
150       tip_node_to_insert = tip_nodes.back();
151       tip_nodes.pop_back();//this deletes the last element of the array
152
153       std::cerr << "inserting last tip: " << tip_node_to_insert->GetSpeciesName() << "\n";
154
155       tip_node_to_insert->SetParentNode(this);
156       this->AddNodeToChildVector(tip_node_to_insert);
157       tree_nodes->push_back(tip_node_to_insert);
158
159     }else{
160
161       std::cerr << "inserting new internode " << *current_node_index << "\n";
162
163       Node* internode = new Node();
164       internode->SetIsTip(false);
165       internode->SetParentNode(this);
166       internode->SetIndex(*current_node_index);
167       this->AddNodeToChildVector(internode);
168       tree_nodes->push_back(internode);
169
170       *current_node_index = *current_node_index + 1;//increase the index counter
171       internode->CreateBifurcatingNode(tip_nodes, current_node_index, tree_nodes);
172
173     }
174   }
175 }
```

3.3.3.3 GetChildVector()

```
std::vector< Node * > Node::GetChildVector ( )
```

Definition at line 104 of file Node.cpp.

```
104                                    {
105
106   return child_nodes;
107
108 }
```

3.3.3.4 GetIndex()

```
int Node::GetIndex ( )
```

Definition at line 60 of file Node.cpp.

```
60                  {
61
62   return index;
63
64 }
```

3.3.3.5 GetIsTip()

```
bool Node::GetIsTip ( )
```

Definition at line 48 of file Node.cpp.

```
48                    {
49
50   return is_tip;
51
52 }
```

3.3.3.6 GetLengthSubtendingBranch()

```
float Node::GetLengthSubtendingBranch ( )
```

Definition at line 73 of file Node.cpp.

```
73                                      {
74
75   return length_of_subtending_branch;
76
77 }
```

3.3.3.7 GetNodeInfo()

```
std::vector< std::string > * Node::GetNodeInfo (
            std::vector< std::string > * collected_node_info )
```

Definition at line 210 of file Node.cpp.

```
210                                                                                 {
211
212   if(child_nodes.empty()) {
213
214     std::cerr << "empty child node vector\n";
215
216     //add the tip node index, species name and sequence to the vector collecting the node information
217     std::string node_info = std::to_string(index) + ' ' + species_name + ' ' + sequence;
218
219     std::cerr << "adding " << node_info << " to info vector\n";
220
221     collected_node_info->push_back(node_info);
222
223
224   }else {
225
226     std::cerr << "recursively calling nodes " << child_nodes.size() << "\n";
227     //for each child node, call this function.
228     for(auto child : child_nodes) {
229
230       std::cerr << "in for\n";
231       child->GetNodeInfo(collected_node_info);
232
233     }
234
235     std::cerr << "back at internode \n";
236
237     //add internode index to the vector collecting the node information
238     std::string node_info = std::to_string(index);
239
240     std::cerr << "adding " << node_info << " to info vector\n";
241
242     collected_node_info->push_back(node_info);
243
244
245   }
246
247   return collected_node_info;
248
249 }
```

3.3.3.8 GetNodePointer()

```
void Node::GetNodePointer (
            std::vector< Node *> * tree_nodes )
```

Definition at line 178 of file Node.cpp.

```
178                                                 {
179
180   if(child_nodes.empty()) {
181
182     std::cerr << "empty child node vector\n";
183
184     std::cerr << "adding node " << index << " to node ref vector\n";
185
186     tree_nodes->push_back(this);
187
188   }else {
189
190     std::cerr << "recursively calling nodes" << child_nodes.size() << "\n";
191     //for each child node, call this function.
192     for(auto child : child_nodes) {
193
```

```
194       std::cerr << "in for\n";
195       child->GetNodePointer(tree_nodes);
196 
197     }
198 
199     std::cerr << "back at internode \n";
200 
201     std::cerr << "adding node " << index << " to node ref vector\n";
202 
203     tree_nodes->push_back(this);
204 
205   }
206 
207 }
```

3.3.3.9 GetParentNode()

```
Node * Node::GetParentNode ( )
```

Definition at line 85 of file Node.cpp.

```
85                          {
86 
87   return parent_node;
88 
89 }
```

3.3.3.10 GetSequence()

```
std::string Node::GetSequence ( )
```

Definition at line 23 of file Node.cpp.

```
23                            {
24 
25   return sequence;
26 
27 }
```

3.3.3.11 GetSpeciesName()

```
std::string Node::GetSpeciesName ( )
```

Definition at line 36 of file Node.cpp.

```
36                               {
37 
38   return species_name;
39 
40 }
```

3.3.3.12 SetChildVector()

```
void Node::SetChildVector (
            std::vector< Node *> childs )
```

Definition at line 98 of file Node.cpp.

```
98                                                  {
99
100   child_nodes = childs;
101
102 }
```

3.3.3.13 SetIndex()

```
void Node::SetIndex (
            int node_index )
```

Definition at line 54 of file Node.cpp.

```
54                                  {
55
56   index = node_index;
57
58 }
```

3.3.3.14 SetIsTip()

```
void Node::SetIsTip (
            bool tip )
```

Definition at line 42 of file Node.cpp.

```
42                             {
43
44   is_tip = tip;
45
46 }
```

3.3.3.15 SetLengthSubtendingBranch()

```
void Node::SetLengthSubtendingBranch (
            float branch_length )
```

Definition at line 67 of file Node.cpp.

```
67                                                      {
68
69   length_of_subtending_branch = branch_length;
70
71 }
```

3.3.3.16 SetParentNode()

```
void Node::SetParentNode (
            Node * parent )
```

Definition at line 79 of file Node.cpp.

```
79                                  {
80
81   parent_node = parent;
82
83 }
```

3.3.3.17 SetSequence()

```
void Node::SetSequence (
            std::string species_sequence )
```

Definition at line 16 of file Node.cpp.

```
16                                              {
17
18   std::cerr << "setting sequence\n";
19   sequence = species_sequence;
20
21 }
```

3.3.3.18 SetSpeciesName()

```
void Node::SetSpeciesName (
            std::string name )
```

Definition at line 29 of file Node.cpp.

```
29                                      {
30
31   std::cerr << "setting species name " << name <<"\n";
32   species_name = name;
33
34 }
```

The documentation for this class was generated from the following files:

- /home/sergio/Repos/phylotoy/src/Node.h
- /home/sergio/Repos/phylotoy/src/Node.cpp

3.4 OutputPrinter Class Reference

```
#include <OutputPrinter.h>
```

Public Member Functions

- OutputPrinter ()
- void PrintMessage2Out (std::string text)

3.4.1 Detailed Description

Definition at line 10 of file OutputPrinter.h.

3.4.2 Constructor & Destructor Documentation

3.4.2.1 OutputPrinter()

```
OutputPrinter::OutputPrinter ( )
```

Definition at line 11 of file OutputPrinter.cpp.

```
11 {}
```

3.4.3 Member Function Documentation

3.4.3.1 PrintMessage2Out()

```
void OutputPrinter::PrintMessage2Out (
            std::string text )
```

Definition at line 13 of file OutputPrinter.cpp.

```
13                                                  {
14
15   std::cout << text;
16
17 }
```

The documentation for this class was generated from the following files:

- /home/sergio/Repos/phylotoy/src/OutputPrinter.h
- /home/sergio/Repos/phylotoy/src/OutputPrinter.cpp

3.5 Tree Class Reference

```
#include <Tree.h>
```

Public Member Functions

- Tree ()
- void SetLength (float length)
- float GetLength ()
- void SetRoot (Node *root_node)
- Node * GetRoot ()
- void SetTreeNodes (std::vector< Node *> nodes)
- std::vector< Node * > GetTreeNodes ()
- void AddToNodeVector (Node *node)
- void GetTreeNodeVector ()
- void CreateStarTree (std::vector< std::string > *alignment)
- void CreateBifurcatingTree (std::vector< std::string > *alignment)
- std::vector< std::string > * CollectTreeNodesInfoRecursively ()
- std::vector< std::string > * CollectTreeNodesInfoIteratively ()

3.5.1 Detailed Description

Objects of the class Tree represent phylogenetic trees. These trees are (1) unrooted (the root node used in the code is just an internode of the tree, (2) can be initialized as start trees, (3) can contain polytomies, or (4) can be bifurcating.

This class provide a number of methods to manipulate the tree. For instance, re-rooting the tree. It also provides methods to alter the tree topology using Nearest-neighbor interchange (NNI) or subtree pruning and regraphting (SPR).

A number of attributes of the tree provide short-cuts to make modifying the tree topology or the length of the tree's branches easy. For instance, Tree objects store pointers to all of their nodes and to the current root. This makes easy to rearrange the tree during MCMC.

The Tree Class is also in charge of proposing all modifications to tree topology and branch length.

Definition at line 20 of file Tree.h.

3.5.2 Constructor & Destructor Documentation

3.5.2.1 Tree()

```
Tree::Tree ( )
```

Definition at line 12 of file Tree.cpp.

```
12 {}
```

3.5.3 Member Function Documentation

3.5.3.1 AddToNodeVector()

```
void Tree::AddToNodeVector (
            Node * node )
```

Definition at line 52 of file Tree.cpp.

```
52                                      {
53
54   std::cerr << "here\n";
55   tree_nodes.push_back(node);
56
57 }
```

3.5.3.2 CollectTreeNodesInfoIteratively()

```
std::vector< std::string > * Tree::CollectTreeNodesInfoIteratively ( )
```

This method returns information on the nodes currently included in the tree. It used the tree_nodes vector of Node pointer to iteratively retrieve the Node information

Returns

a vector of strings

Definition at line 144 of file Tree.cpp.

```
144                                                             {
145
146   std::vector<std::string>* tree_nodes_info = new std::vector<std::string>;
147
148   std::cerr << "iterating over " << tree_nodes.size() << " tree nodes to get their information\n";
149
150   for (auto node : tree_nodes){
151
152     if(node->GetIsTip()){
153
154       //node is a tip, get the species name and sequence
155       std::string node_info = std::to_string(node->GetIndex()) + ' ' + node->GetSpeciesName() + ' ' + node
      ->GetSequence();
156       tree_nodes_info->push_back(node_info);
157       std::cerr << "adding tip: " << node_info << " to information vector\n";
158
159     }else{
160       //node is an internode get the index only
161       std::string node_info = std::to_string(node->GetIndex());
162       tree_nodes_info->push_back(node_info);
163       std::cerr << "adding internode: " << node_info << " to information vector\n";
164
165     }
166
167   }
168
169   return tree_nodes_info;
170 }
```

3.5.3.3 CollectTreeNodesInfoRecursively()

```
std::vector< std::string > * Tree::CollectTreeNodesInfoRecursively ( )
```

This method returns information on the nodes currently included in the tree. It recursively traverses the tree starting from the root node and asks each node for its info as a string that is stored on a vector of strings.

Returns

a vector of strings

Definition at line 128 of file Tree.cpp.

```
128                                                          {
129
130   std::vector<std::string>* tree_nodes_info = new std::vector<std::string>;
131
132   std::cerr << "at tree root\n";
133   tree_nodes_info = current_root->GetNodeInfo(tree_nodes_info);
134
135   return tree_nodes_info;
136 }
```

3.5.3.4 CreateBifurcatingTree()

```
void Tree::CreateBifurcatingTree (
            std::vector< std::string > * alignment )
```

This method creates a bifurcating tree. This method creates as many internodes as required to yield a bifurcating tree.

Parameters

alignment	is a sequence alignment stored on a vector of strings containing species names and sequencesseparated by an empty space.

Definition at line 99 of file Tree.cpp.

```
99                                                           {
100
101   int current_node_index = alignment->size();
102
103   //we need to initialize the root node
104   Node* current_root = new Node();
105   current_root->SetIsTip(false);
106   current_root->SetIndex(current_node_index);
107
108   //once the root node has been initialized we add it as the tree root and to the list of tree nodes
109   this->SetRoot(current_root);
110   this->AddToNodeVector(current_root);
111
112   std::cerr << "Creating bifurcating tree with " << current_node_index << " species\n";
113
114   current_node_index = current_node_index + 1;
115
116   //we initialize the tip nodes using the alignment provided by the user
117   current_root->CreateBifurcatingNode(this->InitializeTipNodes(alignment), &
      current_node_index, &tree_nodes);
118
119 }
```

3.5.3.5 CreateStarTree()

```
void Tree::CreateStarTree (
            std::vector< std::string > * alignment )
```

This method creates a star tree. This tree adds all the tip nodes in one alignment to the root node.

Parameters

alignment	is a sequence alignment with species names and sequences separated by an empty space.

Definition at line 64 of file Tree.cpp.

```
64                                                          {
65
66   std::cerr << "Creating star tree\n";
67
68   //initialize the root node
69   Node* current_root = new Node();
70   current_root->SetIsTip(false);
71   current_root->SetIndex(alignment->size());
72
73   //send the alignment to our private tip node initializer
74   current_root->SetChildVector(this->InitializeTipNodes(alignment));
75
76   //once all tips have been added as childs to the root, the root sends a pointer to himself to each of the
       child nodes.
77   for(auto child : current_root->GetChildVector()){
78
79     child->SetParentNode(current_root);
80     //we also need to set the length of the subtending branch leading to the parent
81
82     //finally we need to add this child to the tree list of nodes
83     this->AddToNodeVector(child);
84   }
85
86   //set current_root node as the root of the tree object
87   this->SetRoot(current_root);
88
89   //add root to the node vector
90   this->AddToNodeVector(current_root);
91
92 }
```

3.5.3.6 GetLength()

```
float Tree::GetLength ( )
```

Definition at line 20 of file Tree.cpp.

```
20                      {
21
22   return length;
23
24 }
```

3.5.3.7 GetRoot()

```
Node * Tree::GetRoot ( )
```

Definition at line 33 of file Tree.cpp.

```
33                    {
34
35   return current_root;
36
37 }
```

3.5.3.8 GetTreeNodes()

```
std::vector< Node * > Tree::GetTreeNodes ( )
```

Definition at line 45 of file Tree.cpp.

```
45                                    {
46
47   return tree_nodes;
48
49 }
```

3.5.3.9 GetTreeNodeVector()

```
void Tree::GetTreeNodeVector ( )
```

In case we need to update the list of Nodes in the tree_nodes list, this method should provide a way to get pointers to all the nodes in the tree recursively

Definition at line 178 of file Tree.cpp.

```
178                              {
179
180   std::cerr << "at tree root\n";
181   current_root->GetNodePointer(&tree_nodes);
182
183 }
```

3.5.3.10 SetLength()

```
void Tree::SetLength (
              float length )
```

Definition at line 14 of file Tree.cpp.

```
14                                {
15
16     length = length;
17
18 }
```

3.5.3.11 SetRoot()

```
void Tree::SetRoot (
            Node * root_node )
```

Definition at line 26 of file Tree.cpp.

```
26                                  {
27
28   current_root = root_node;
29
30 }
```

3.5.3.12 SetTreeNodes()

```
void Tree::SetTreeNodes (
            std::vector< Node *> nodes )
```

Definition at line 39 of file Tree.cpp.

```
39                                              {
40
41   tree_nodes = nodes;
42
43 }
```

The documentation for this class was generated from the following files:

- /home/sergio/Repos/phylotoy/src/Tree.h
- /home/sergio/Repos/phylotoy/src/Tree.cpp

Chapter 4

File Documentation

4.1 /home/sergio/Repos/phylotoy/src/Controller.cpp File Reference

```
#include "Controller.h"
#include <string>
#include <exception>
#include <assert.h>
#include <vector>
```

4.2 /home/sergio/Repos/phylotoy/src/Controller.h File Reference

```
#include "InputReader.h"
#include "OutputPrinter.h"
#include "Tree.h"
#include <string>
#include <vector>
```

Classes

- class Controller

4.3 /home/sergio/Repos/phylotoy/src/InputReader.cpp File Reference

```
#include "InputReader.h"
#include <fstream>
#include <vector>
#include <iostream>
```

4.4 /home/sergio/Repos/phylotoy/src/InputReader.h File Reference

```
#include <vector>
#include <string>
```

Classes

- class InputReader

4.5 /home/sergio/Repos/phylotoy/src/Node.cpp File Reference

```
#include <vector>
#include <string>
#include "Node.h"
#include <iostream>
```

4.6 /home/sergio/Repos/phylotoy/src/Node.h File Reference

```
#include <vector>
#include <string>
```

Classes

- class Node

4.7 /home/sergio/Repos/phylotoy/src/OutputPrinter.cpp File Reference

```
#include <iostream>
#include "OutputPrinter.h"
```

4.8 /home/sergio/Repos/phylotoy/src/OutputPrinter.h File Reference

```
#include <string>
```

Classes

- class OutputPrinter

4.9 /home/sergio/Repos/phylotoy/src/Phylotoy.cpp File Reference

```
#include <unistd.h>
#include <stdlib.h>
#include "Controller.h"
#include <string>
```

Functions

- int main (int argc, char ∗argv[ ])

4.9.1 Function Documentation

4.9.1.1 main()

```
int main (
            int argc,
            char * argv[ ] )
```

Definition at line 13 of file Phylotoy.cpp.

```
13                                {
14
15   //Initialize a controller object
16
17   Controller phylotoy_controller;
18
19   /* Read the CLI options set the appropriate variables in the controller
20    * Options are passsed as follows:
21    * -r random seed
22    * -i alignment path
23    * -c chain name
24    */
25
26   int option;
27
28   while ((option = getopt(argc, argv, "r:i:c:")) != -1) {
29
30     switch (option){
31
32       case 'r':
33       {
34         phylotoy_controller.SetRandomSeed(atoi(optarg));
35         break;
36       }
37       case 'i':
38       {
39         std::string input (optarg);
40         phylotoy_controller.SetAlignmentFilePath(input);
41         break;
42       }
43       case 'c':
44       {
45         std::string name (optarg);
46         phylotoy_controller.SetChainName(name);
47         break;
48       }
49       default:
50         abort();
51     }
52   }
53
54   phylotoy_controller.Run();
55
56   return 0;
57 }
```

4.10 /home/sergio/Repos/phylotoy/src/Tree.cpp File Reference

```
#include <vector>
#include <string>
#include "Tree.h"
#include <sstream>
#include <iostream>
```

4.11 /home/sergio/Repos/phylotoy/src/Tree.h File Reference

```
#include <vector>
#include <string>
#include "Node.h"
```

Classes

- class Tree

Index

/home/sergio/Repos/phylotoy/src/Controller.cpp, 27
/home/sergio/Repos/phylotoy/src/Controller.h, 27
/home/sergio/Repos/phylotoy/src/InputReader.cpp, 27
/home/sergio/Repos/phylotoy/src/InputReader.h, 28
/home/sergio/Repos/phylotoy/src/Node.cpp, 28
/home/sergio/Repos/phylotoy/src/Node.h, 28
/home/sergio/Repos/phylotoy/src/OutputPrinter.cpp, 28
/home/sergio/Repos/phylotoy/src/OutputPrinter.h, 28
/home/sergio/Repos/phylotoy/src/Phylotoy.cpp, 29
/home/sergio/Repos/phylotoy/src/Tree.cpp, 30
/home/sergio/Repos/phylotoy/src/Tree.h, 30

AddNodeToChildVector
    Node, 13
AddToNodeVector
    Tree, 21

CheckAlignmentFilePath
    Controller, 6
CheckCLIOptions
    Controller, 6
CheckChainName
    Controller, 6
CheckRandomSeed
    Controller, 7
CollectTreeNodesInfoIteratively
    Tree, 22
CollectTreeNodesInfoRecursively
    Tree, 22
Controller, 5
    CheckAlignmentFilePath, 6
    CheckCLIOptions, 6
    CheckChainName, 6
    CheckRandomSeed, 7
    Controller, 5
    GetAlignmentFilePath, 7
    GetChainName, 7
    GetRandomSeed, 8
    Run, 8
    SetAlignmentFilePath, 9
    SetChainName, 9
    SetRandomSeed, 9
CreateBifurcatingNode
    Node, 13
CreateBifurcatingTree
    Tree, 23
CreateStarTree
    Tree, 24

GetAlignmentFilePath
    Controller, 7
GetChainName
    Controller, 7
GetChildVector
    Node, 14
GetIndex
    Node, 14
GetIsTip
    Node, 15
GetLength
    Tree, 24
GetLengthSubtendingBranch
    Node, 15
GetNodeInfo
    Node, 15
GetNodePointer
    Node, 16
GetParentNode
    Node, 17
GetPath
    InputReader, 11
GetRandomSeed
    Controller, 8
GetRoot
    Tree, 24
GetSequence
    Node, 17
GetSpeciesName
    Node, 17
GetTreeNodeVector
    Tree, 25
GetTreeNodes
    Tree, 25

InputReader, 10
    GetPath, 11
    InputReader, 10
    ReadInputFile, 11
    SetPath, 12

main
    Phylotoy.cpp, 29

Node, 12
    AddNodeToChildVector, 13
    CreateBifurcatingNode, 13
    GetChildVector, 14
    GetIndex, 14
    GetIsTip, 15
    GetLengthSubtendingBranch, 15

GetNodeInfo, 15
GetNodePointer, 16
GetParentNode, 17
GetSequence, 17
GetSpeciesName, 17
Node, 13
SetChildVector, 17
SetIndex, 18
SetIsTip, 18
SetLengthSubtendingBranch, 18
SetParentNode, 18
SetSequence, 19
SetSpeciesName, 19

OutputPrinter, 19
OutputPrinter, 20
PrintMessage2Out, 20

Phylotoy.cpp
main, 29
PrintMessage2Out
OutputPrinter, 20

ReadInputFile
InputReader, 11
Run
Controller, 8

SetAlignmentFilePath
Controller, 9
SetChainName
Controller, 9
SetChildVector
Node, 17
SetIndex
Node, 18
SetIsTip
Node, 18
SetLength
Tree, 25
SetLengthSubtendingBranch
Node, 18
SetParentNode
Node, 18
SetPath
InputReader, 12
SetRandomSeed
Controller, 9
SetRoot
Tree, 25
SetSequence
Node, 19
SetSpeciesName
Node, 19
SetTreeNodes
Tree, 26

Tree, 20
AddToNodeVector, 21
CollectTreeNodesInfoIteratively, 22
CollectTreeNodesInfoRecursively, 22
CreateBifurcatingTree, 23
CreateStarTree, 24
GetLength, 24
GetRoot, 24
GetTreeNodeVector, 25
GetTreeNodes, 25
SetLength, 25
SetRoot, 25
SetTreeNodes, 26
Tree, 21