Below is a small implementation of the Unix “Info” utility implemented by me.

***Infotree.cxx***

#include <iostream>

#include <vector>

#include <string>

using namespace std;

#include "info\_tree"

using namespace info;

namespace info{

info\_tree::

info\_tree(int addr,string\* n\_n,string\* d\_n,int type)

{

info\_node \*new\_node=new info\_node(addr,\*n\_n,\*d\_n);

root=new\_node;

root->type=type;

}

info\_tree::

~info\_tree(void)

{

delete (root);

for(int i=0;i<sub\_info\_tree.size();i++)

{

sub\_info\_tree.pop\_back();

}

}

info\_tree::

info\_tree(const info\_tree& item)

{

root=new info\_node(item.root);

sub\_info\_tree=item.sub\_info\_tree;

}

info\_tree\* info\_tree::

Get\_Subtree(string\* dis\_n,int type)

{

info\_tree \*temp;

string nn;

if (type ==0)

{

for(int i=0;i<sub\_info\_tree.size();i++)

{

temp=sub\_info\_tree.at(i);

nn=temp->root->dis\_n;

if(nn.find(\*dis\_n,0)!=string::npos && temp->root->type=='m')

return temp;

}

}

else{

for(int i=0;i<sub\_info\_tree.size();i++)

{

temp=sub\_info\_tree.at(i);

if(temp->root->type=='m')

break;

nn=temp->root->dis\_n;

if(nn==\*dis\_n && temp->root->type=='c')

return temp;

}

}

return NULL;

}

info\_tree\* info\_tree::Get\_next()

{

info\_tree \*ret\_tree;

if(root->index!=(root->\_up)->sub\_info\_tree.size() && root->index>-1)

{

ret\_tree= (root->\_up)->sub\_info\_tree.at(root->index+1);

return ret\_tree;

}

return NULL;

}

info\_tree\* info\_tree::Get\_prev()

{

info\_tree \*ret\_tree;

if(root->index!=0 && root->index>-1)

{

ret\_tree=(root->\_up)->sub\_info\_tree.at(root->index-1);

return ret\_tree;

}

return NULL;

}

info\_tree\* info\_tree::Get\_up()

{

info\_tree \*ret\_tree;

if(root->\_up!=NULL)

{

ret\_tree=root->\_up;

return ret\_tree;

}

return NULL;

}

void info\_tree::Set\_Subtree(info\_tree\* item)

{

sub\_info\_tree.push\_back(item);

item->root->\_up=this;

(\*item).root->index=sub\_info\_tree.size();

}

void info\_tree::Set\_Node\_adress(int addr)

{

root->addr=addr;

}

void info\_tree::Set\_Node\_name(string\* name)

{

root->node\_n=\*name;

}

void info\_tree::Set\_node\_disname(string\* disname)

{

root->dis\_n=\*disname;

}

void info\_tree::Set\_node\_docname(string\* docname)

{

root->doc\_n=\*docname;

}

string info\_tree::Get\_disname()

{

return root->dis\_n;

}

string info\_tree::Get\_docname()

{

return root->doc\_n;

}

string info\_tree::Get\_name()

{

return root->node\_n;

}

int info\_tree::Get\_adress()

{

return root->addr;

}

}

***Inforeader.cxx***

#include <fstream>

#include <iostream>

#include <string>

#include <process.h>

#include "info\_tree"

#include "inforeader"

using namespace std;

using namespace info;

namespace info

{

void Adjust\_Nodes(string\* link,info\_tree\* up,int type)

{

int middle=link->find(":",0);

int cont=link->find("]");

string \*d\_n=new string(link->substr(link->find("[")+1,middle-2));

string \*dis\_n=new string(link->substr(cont+1,link->find(")")-cont-1));

if(\*d\_n=="")

\*d\_n=up->Get\_docname();

char typec='c';

if (type==0)

typec='m';

string \*n\_n=new string(link->substr(middle+2,cont-middle-2));

info\_tree \*new\_item=new info\_tree(0,n\_n,d\_n,typec);

new\_item->Set\_node\_disname(dis\_n);

up->Set\_Subtree(new\_item);

readfile(d\_n,new\_item);

}

info\_tree\* readfile(string\* filename,info\_tree\* node\_n)

{

ifstream infile;

infile.open(filename->c\_str());

if (!infile)

{

cout << "Unable to open file: " << &filename << endl;

return NULL;

}

string line;

char cline[80];

infile.getline(cline,80);

line.assign(cline);

int line\_count=0,begin,end,middle;

info\_tree \*start;

while(begin=line.find("[",0)>-1)

{

infile.getline(cline,80);

line.assign(cline);

line\_count++;

}

if(node\_n==NULL)

{

middle=line.find(":",0);

string\* n\_n=new string(line.substr(begin+1,middle-begin-1));

end=line.find("]",0);

string\* dis\_n=new string(line.substr(middle+1,end-middle-1));

string\* fn=new string(\*filename);

start=new info\_tree(infile.tellg(),n\_n,fn,-1);

start->Set\_node\_disname(dis\_n);

}

else{

bool pass=true;

while(pass)

{

int begin=line.find("[",0);

if(begin>-1 && line.find("(",0)==string::npos)

{

int middle=line.find(":",0);

string\* n\_n=new string(line.substr(begin+1,middle-begin-1));

if(\*n\_n==node\_n->Get\_name())

{

start=node\_n;

start->Set\_Node\_adress(infile.tellg());

pass=false;

}

}

infile.getline(cline,80);

line.assign(cline);

}

}

while(line.find("\* Menu",0)==string::npos && line.find("%%",0)==string::npos)

{

if(line.find("([",0)!=string::npos)

{

begin=line.find("(",0);

end=line.find(")",1);

string str=line.substr(begin,end-begin);

Adjust\_Nodes(&str,start,1);

}

infile.getline(cline,80);

line.assign(cline);

}

if(line.find("%%",0)==string::npos)

{

infile.getline(cline,80);

infile.getline(cline,80);

line.assign(cline);

while(line.find("\*",0)!=string::npos)

{

begin=line.find("(",0);

end=line.find(")",1);

string str=line.substr(begin,end-begin+1);

Adjust\_Nodes(&str,start,0);

infile.getline(cline,80);

line.assign(cline);

}

}

infile.close();

return start;

}

char disp(info\_tree\* dis)

{

system("cls");

cout<<"node:"<<dis->Get\_disname()<<", document:"<<dis->Get\_docname()<<endl;

string dis\_n\_p="-",dis\_n\_n="-",dis\_n\_u="-";

if(dis->Get\_docname()!="dir.txt")

{

if(dis->Get\_next()!=NULL)

dis\_n\_n=dis->Get\_next()->Get\_disname();

if(dis->Get\_prev()!=NULL)

dis\_n\_n=dis->Get\_prev()->Get\_disname();

if(dis->Get\_up()!=NULL)

dis\_n\_n=dis->Get\_up()->Get\_disname();

}

cout<<"next:"<<dis\_n\_n<<", prev:"<<dis\_n\_p<<", up:"<<dis\_n\_u<<endl;

ifstream infile;

infile.open(dis->Get\_docname().c\_str());

if (!infile)

{

cout << "Unable to open file: " <<dis->Get\_docname()<< endl;

return 'a';

}

string line;

char cline[80];

int begin,end;

infile.seekg(dis->Get\_adress());

infile.getline(cline,80);

line.assign(cline);

int linecount=1;

while(line.find("%%",0)==string::npos)

{

if(linecount%59!=0)

{

if(line.find("([",0)!=string::npos)

{

begin=line.find("]",0);

end=line.find(")",1);

string str=line.substr(begin+1,end-begin-1);

if(line.find("\*",0)!=string::npos)

str="\n\* "+str;

cout<<str<<line.substr(end+1,line.length()-end)<<endl;

infile.getline(cline,80);

line.assign(cline);

}

else{

cout<<"\n"<<line;

infile.getline(cline,80);

line.assign(cline);

}

linecount++;

}

else{

cout<<"press 'f' to see more "<<endl;

char command=getchar();

getchar();

if (command=='f')

linecount++;

else{

return command;

}

}

}

cout<<"enter your choice"<<endl;

char command=getchar();

getchar();

return command;

}

info\_tree\* command\_m(info\_tree\* subject,string\* searched)

{

info\_tree \*ret=subject->Get\_Subtree(searched,0);

return ret;

}

info\_tree\* command\_n(info\_tree\* subject)

{

info\_tree \*ret=subject->Get\_next();

return ret;

}

info\_tree\* command\_p(info\_tree\* subject)

{

info\_tree \*ret=subject->Get\_prev();

return ret;

}

info\_tree\* command\_g(info\_tree\* subject,string\* searched)

{

info\_tree \*ret=subject->Get\_Subtree(searched,1);

return ret;

}

info\_tree\* command\_u(info\_tree\* subject)

{

info\_tree \*ret=subject->Get\_up();

return ret;

}

}