# Contents



# 1 Classes

## 1.1 tree.hpp

```cpp
#ifndef TREE_HPP
#define TREE_HPP

#include <vector>
#include <stdio.h>
#include <iostream>
#include <string>
#include <memory>
#include <stdlib.h>
#include <stack>

using namespace std;

typedef struct node
{
    string value;
    vector<node*> children;
} node;

void node_connect(node* father,node* child)
{
    father->children.push_back(child);
}

string tree_dfs(node* head)
{
    stack <node*> bucket;
    string line;
    bucket.push(head);
    while(!bucket.empty())
    {
        node* current_node = new node;
        current_node = bucket.top();
        bucket.pop();
        line+=current_node->value;
        if(current_node->children.size())
            line+="+";
        else
            line+="-";
        for(int i=0; i<current_node->children.size(); i++)
            bucket.push(current_node->children[i]);
    }
    return line;
}
#endif // TREE_HPP
```

## 1.2 cfg.hpp

```cpp
#ifndef CFG_HPP
#define CFG_HPP

#include <vector>
#include <stdio.h>
#include <iostream>
#include <string>
#include <memory>
#include <stdlib.h>
#include "global.hpp"

using namespace std;

class cfg
{
public:
    vector< vector<string> > rules;
    cfg();
    void debug();
};

cfg::cfg()
{
    FILE* file = fopen("cfg.txt","r");
    if(file==NULL)
    {
        printf("Cannot open cfg.txt\n");
        return;
    }

    char buffer;
    string word;
    vector <string> rule;

    while(!feof(file))
    {
        fread(&buffer,sizeof(char),1,file);
        if(buffer!=' '&&buffer!='\n')
            word+=buffer;
        if(buffer==' '||buffer=='\n')
        {
            rule.push_back(word);
            word.clear();
        }
        if(buffer=='\n')
        {
            rules.push_back(rule);
            rule.clear();
        }
    }
    fclose(file);
}

void cfg::debug()
{
    for(int i=0; i<rules.size()-1; i++)
    {
        cout<<"p: ";
        for(int j=0; j<rules[i].size(); j++)
            cout<<rules[i][j]<<" ";
        if(i<rules.size()-2)
            cout<<endl;
    }
}

#endif // CFG_HPP
```