COGNITIVE COMPLEXITY

A Rudimentary Implementation

MAY 10, 2018

Contents:-

Problem statement:-

Prepare a tool to calculate Cognitive Complexity of a given program.

Languages used:- C .

Introduction:

Cyclomatic Complexity has long been the standard for measuring the complexity of a method's control flow. **Cognitive** Complexity has been formulated to address modern language structures, and to produce values that are meaningful at the class and application levels.

Basic criteria and methodology-

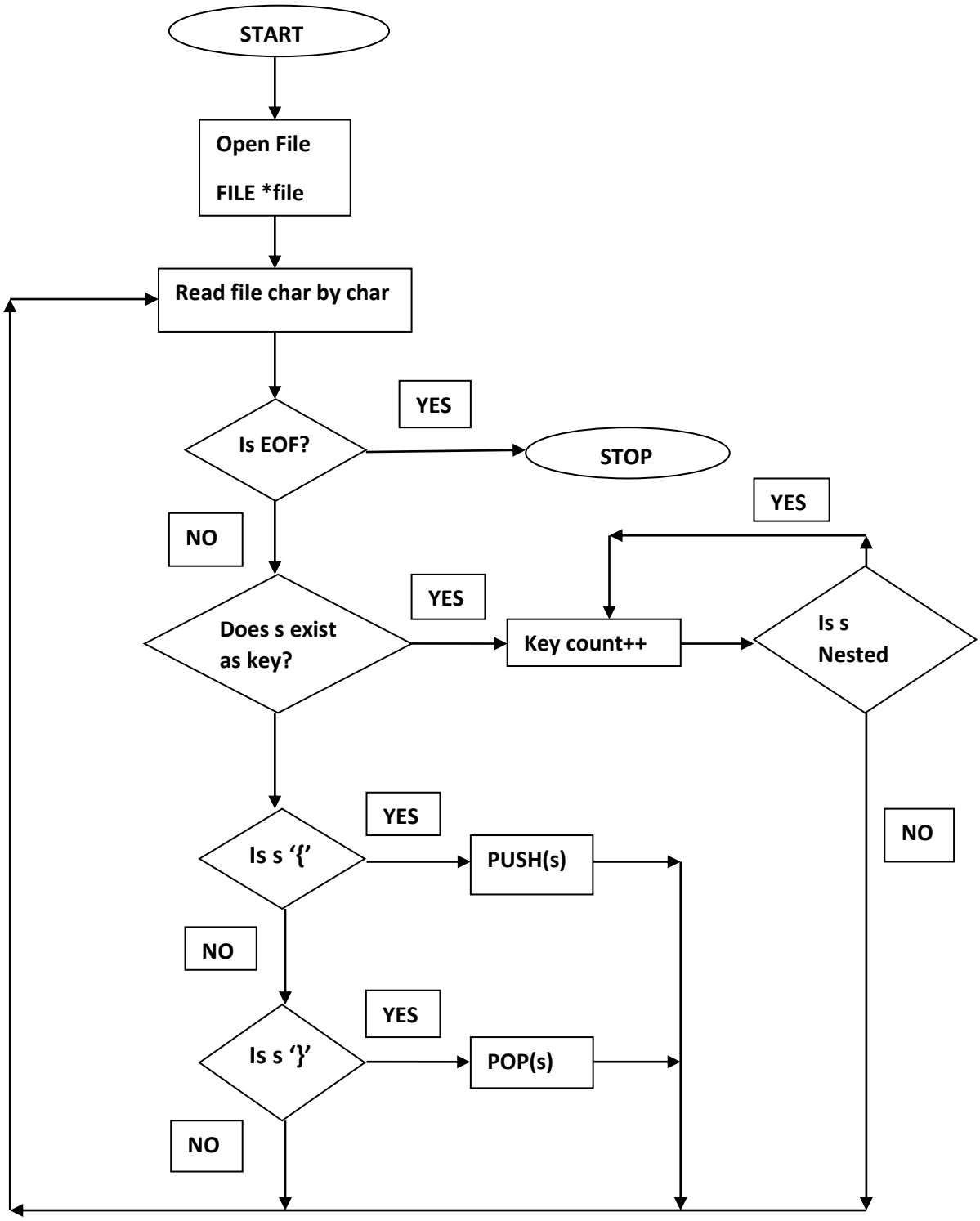
A Cognitive Complexity score is assessed according to three basic rules:

1. Ignore structures that allow multiple statements to be readably shorthanded into one

2. Increment (add one) for each break in the linear flow of the code

3. Increment when flow-breaking structures are nested

Flow Chart:-



's' is used to store strings in it

Results:

We perform various test cases and obtain corresponding results-

Case I:

code1.txt - Notepad

File Edit Format View Help

```
if()
{
}
for()
{
for()
{
}
}
```

Output:

P:\Projects\SourceCode\CognitiveComplexity.exe

```
Enter the filename to open
code1.txt
if:1
for:3
else:0
while: 0
do-while: 0
else if: 0
Total Cognitive Complexity is 4

Process returned 32 (0x20)   execution time : 7.675 s
Press any key to continue.
```

Case II:

code4.txt - Notepad

File Edit Format View Help

```
#include<stdio.h>
int main()
[
int i,j;
int sum=100;
for(i=0;i<sum;i++)
{
if(sum==100)
{
//do something
}
else
{
//do something
}
for(j=i;j>0;j--)
{
//do something
}
}
]
void func1()
[
if()
{
}
for()
{
for()
{
//do  something
}
}
]
```

Output:

P:\Projects\SourceCode\CognitiveComplexity.exe

```
Enter the filename to open
code4.txt
if:3
for:6
else:2
while: 0
do-while: 0
else if: 0
Total Cognitive Complexity is 11

Process returned 33 (0x21)   execution time : 37.511 s
Press any key to continue.
```

Conclusion:

Writing and maintaining code are human processes. This is why mathematical models are inadequate to assess the effort they require.

Cognitive Complexity breaks from the practice of using mathematical models to assess software maintainability. More importantly, it departs from the practice of evaluating code based on mathematical models so that it can yield assessments of control flow that correspond to programmers' intuitions about the mental, or *cognitive* effort required to understand those flows.

Cognitive Complexity thus becomes a tool for measuring the relative understandability of classes and applications.