Final Report

CSE-0302 Summer - 2021

Sadia Tasnim
UG02-50-19-018
Department of Computer Science and Engineering
State University of Bangladesh (SUB)
Dhaka, Bangladesh
sadiatasnim2214190@gmail.com

***Abstract*—Main theme of your assignment or academic projects.**

n

***Index Terms*—The word mostly used in your report.**

I. Introduction

Assignment 4 : Detecting Simple Syntax Errors

Syntax errors are very common in source program.The main purpose of this session is to write programs to detect and report simple syntax errors.

Assignment 5 : Use of CFGs for Parsing

We can think of using CFGs to parse various language constructs in the token streams freed from simple syntactic and semantic errors, as it is easier to describe the constructs with CFGs.But CFGs are hard to apply practically. In this session,we implement a simple recursive descent parser to parse a number of types of statements after exercising with simpler CFGs.We note that a recursive decent parser can be construsted from a CFGs with reduced left recursion and ambiguity.

Assignment 6 : Predictive Parsing

Manual implementation of LL(1) and LR(1) parsing algorithms .

II. Literature Review

Assignment 4 : Detecting Simple Syntax Errors

A frustrating aspect of software development is that compiler error messages often fail to locate the actual cause of a syntax error.Syntax Errors Just Aren't Natural.Jashua Charles (Department of Computing Science),Abram Hindle (department of Computing Science),Jose Nelson Amaral(Department of Computing Science) Improving Error Reporting with Language Models.

Assignment 5 : Use of CFGs for Parsing

Context Free Grammars (CFG) can be classified on the basis of following two properties: 1) Based on number of strings it generates. During Compilation, the parser uses the grammar of the language to make a parse tree(or derivation tree) out of the source code. Vilhjálmur orsteinsson, Hulda Óladóttir,Hrafn Loftsson(Department of Computer Science) .Both present open-source,wide-coverage context-free grammer (CFG) for Icelandic and an accompanying parsing system.

Assignment 6 : Predictive Parsing

A predictive parser is a recursive descent parser with no backtracking or backup. It is a top-down parser that does not require backtracking. At each step, the choice of the rule to be expanded is made upon the next terminal symbol.

III. Proposed Methodology

IV. Conclusion and Future Work

Every Computer Engineer should learn compiler design so that an interpreted scripting language and interpreter.I think thatwhat is useful is how to :Parse an expression tree,Robust error handling,General-purpose text processing technique,Sanitize input,Schedule tasks in the future with cross-platform timers,Creation of virtual machines.

Acknowledgment

I would like to thank my honourable**Khan Md. Hasib Sir** for his time, generosity and critical insights into this project.

References

[1] G. Eason, B. Noble, and I. N. Sneddon, "On certain integrals of Lipschitz-Hankel type involving products of Bessel functions," Phil. Trans. Roy. Soc. London, vol. A247, pp. 529–551, April 1955.

[2] J. Clerk Maxwell, A Treatise on Electricity and Magnetism, 3rd ed., vol. 2. Oxford: Clarendon, 1892, pp.68–73.

[3] I. S. Jacobs and C. P. Bean, "Fine particles, thin films and exchange anisotropy," in Magnetism, vol. III, G. T. Rado and H. Suhl, Eds. New York: Academic, 1963, pp. 271–350.

[4] K. Elissa, "Title of paper if known," unpublished.

[5] R. Nicole, "Title of paper with only first word capitalized," J. Name Stand. Abbrev., in press.

[6] Y. Yorozu, M. Hirano, K. Oka, and Y. Tagawa, "Electron spectroscopy studies on magneto-optical media and plastic substrate interface," IEEE Transl. J. Magn. Japan, vol. 2, pp. 740–741, August 1987 [Digests 9th Annual Conf. Magnetics Japan, p. 301, 1982].

[7] M. Young, The Technical Writer's Handbook. Mill Valley, CA: University Science, 1989.

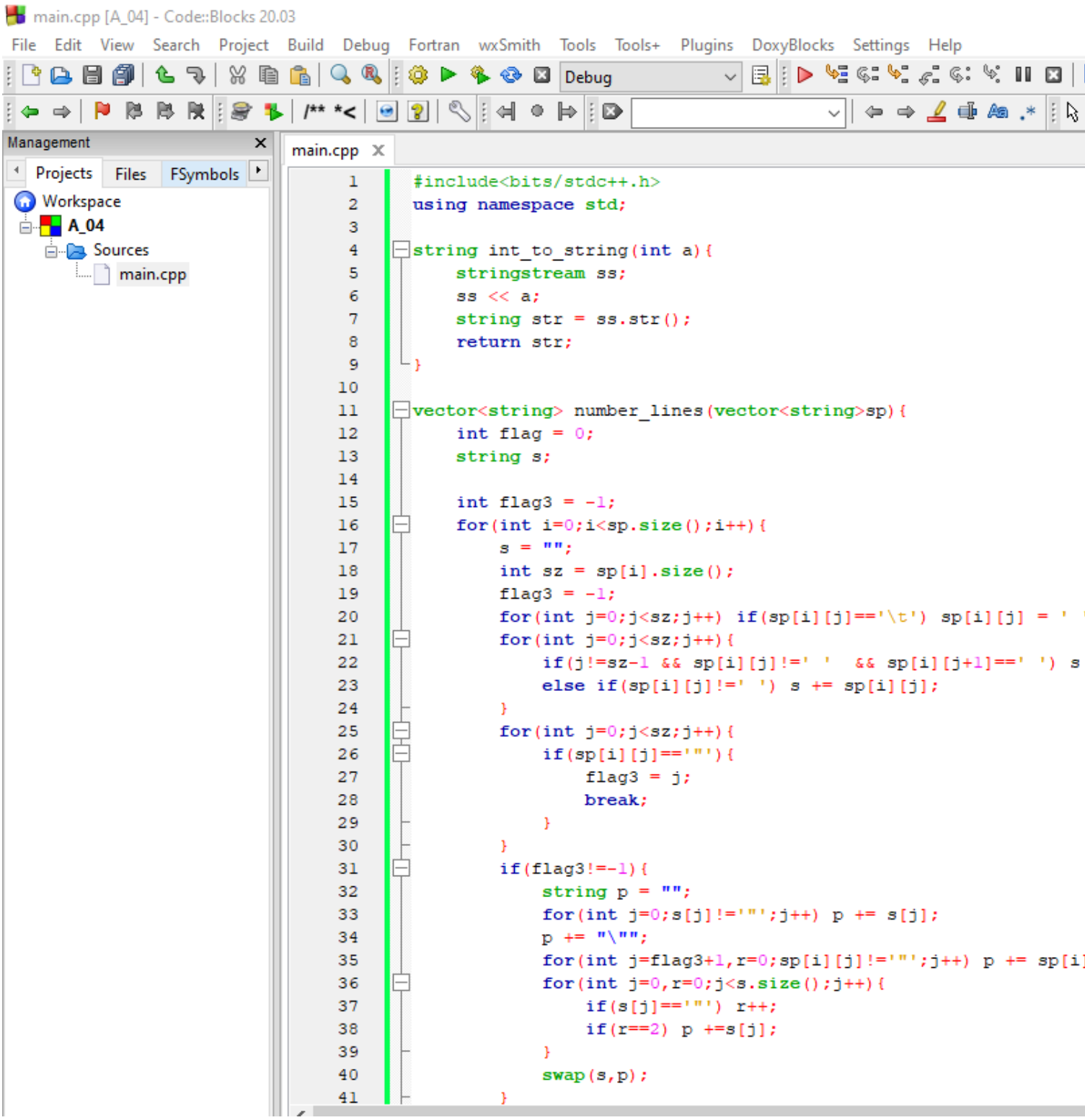

Fig. 1. Proposed Methodology

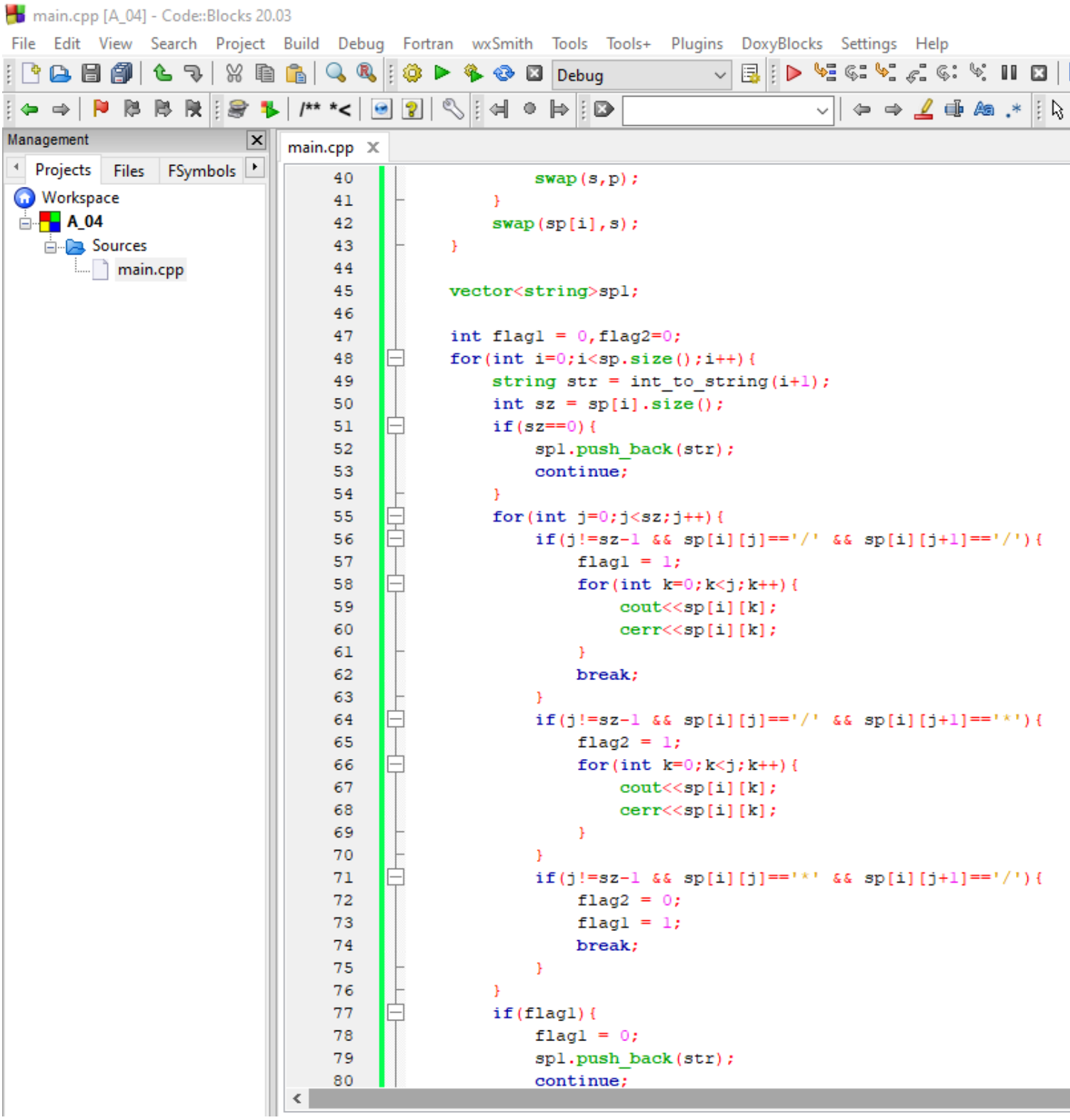

Fig. 2. Proposed Methodology

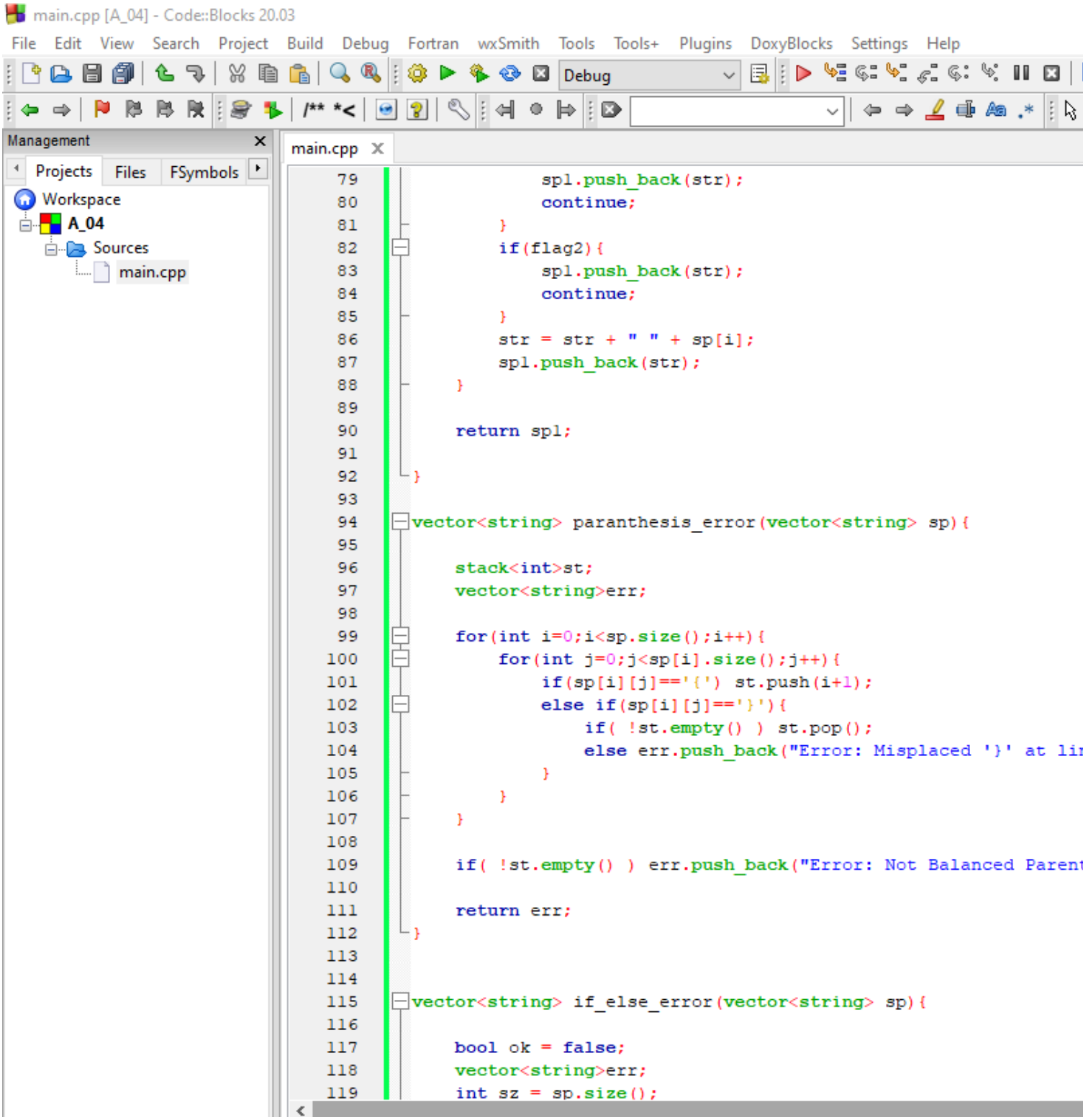

Fig. 3. Proposed Methodology

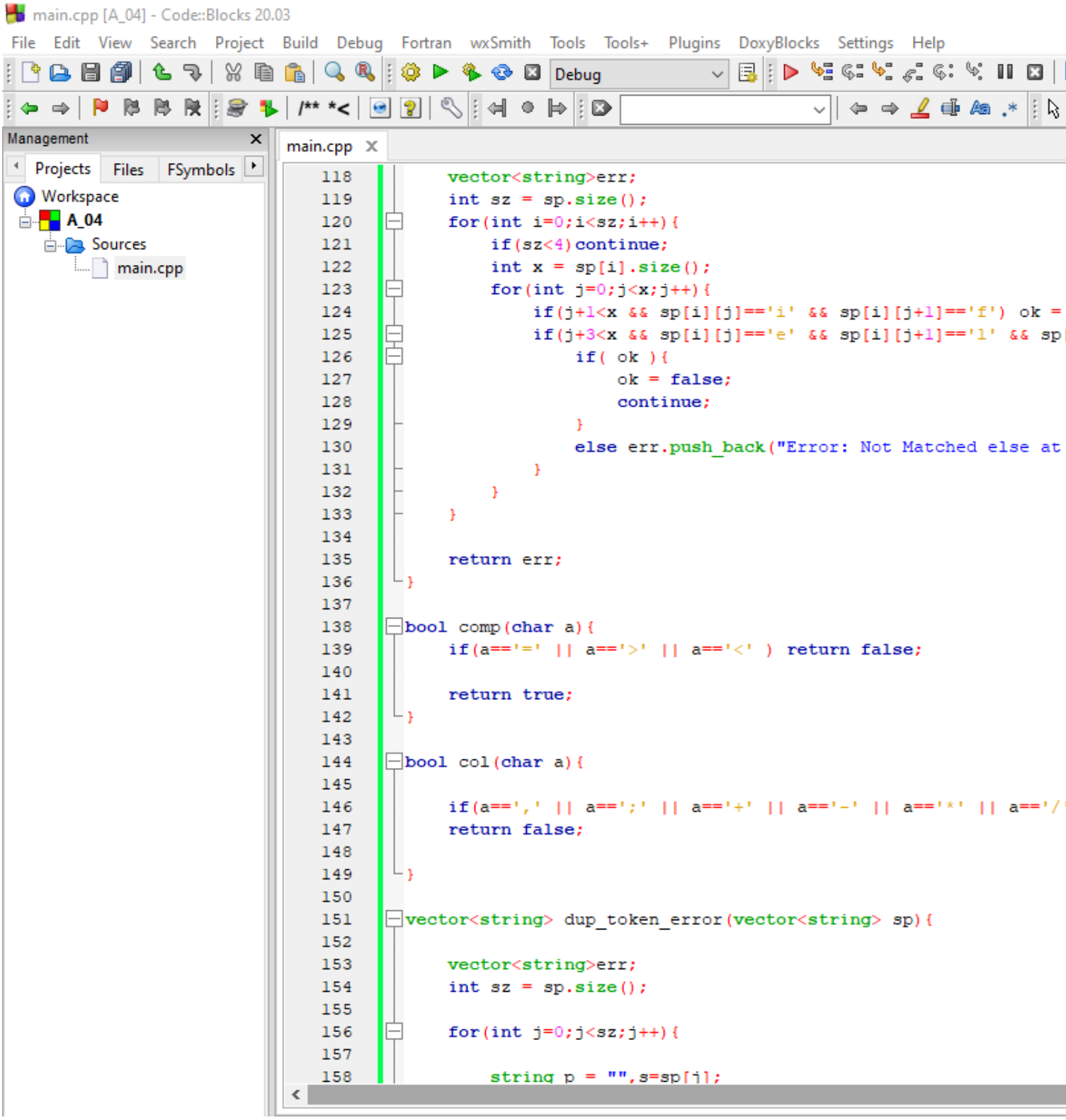

Fig. 4. Proposed Methodology

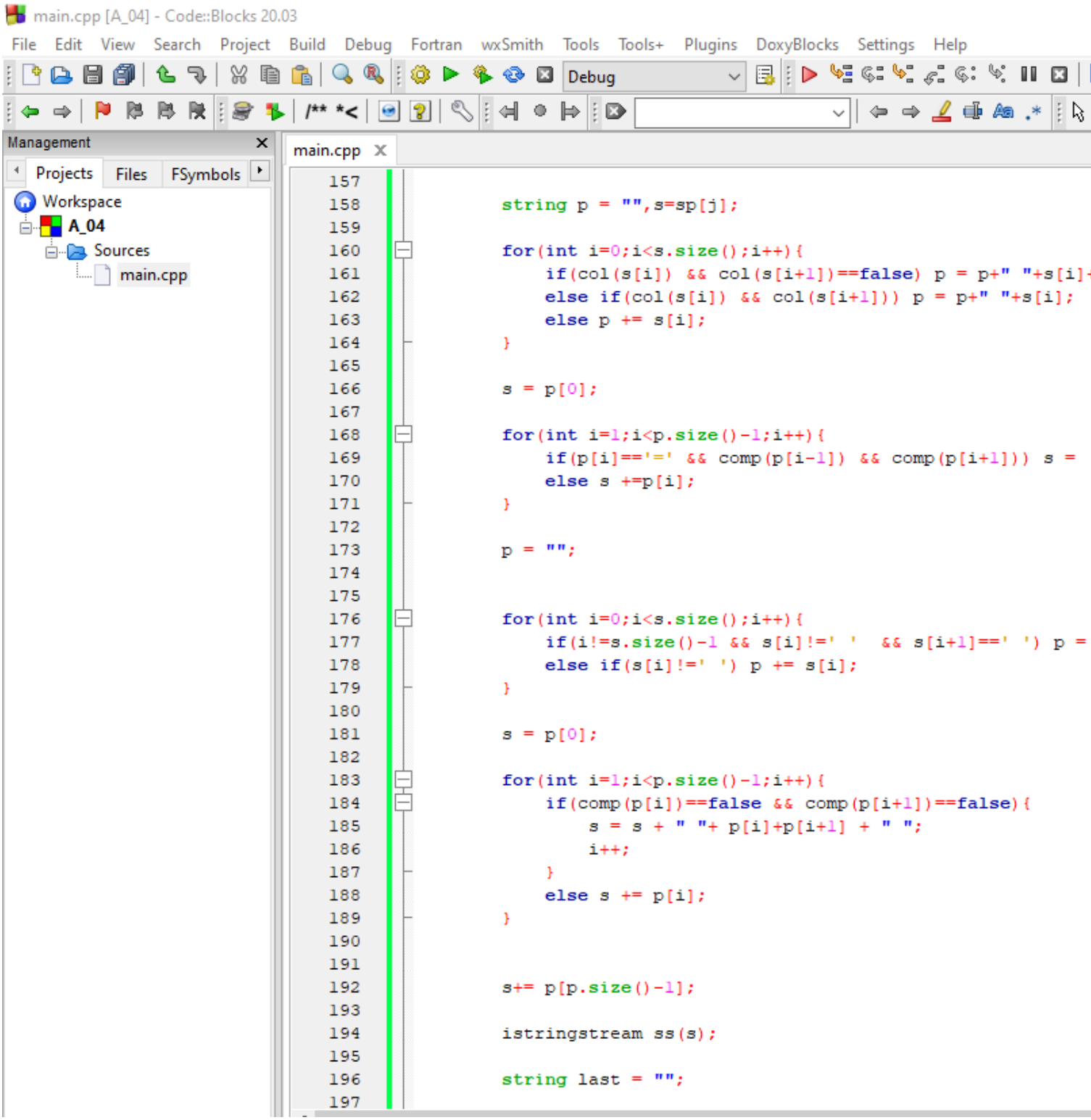

Fig. 5. Proposed Methodology

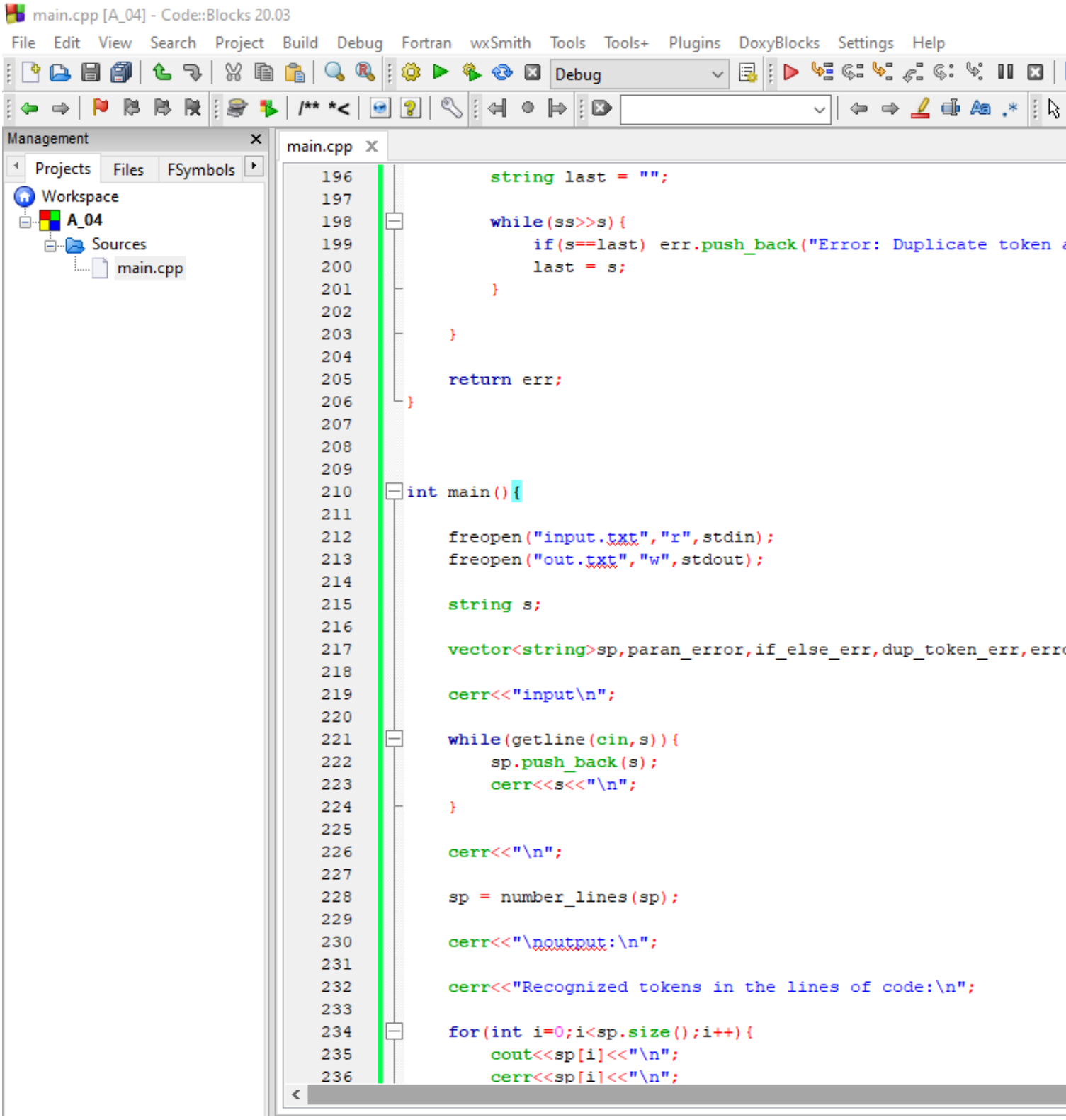

Fig. 6. Proposed Methodology

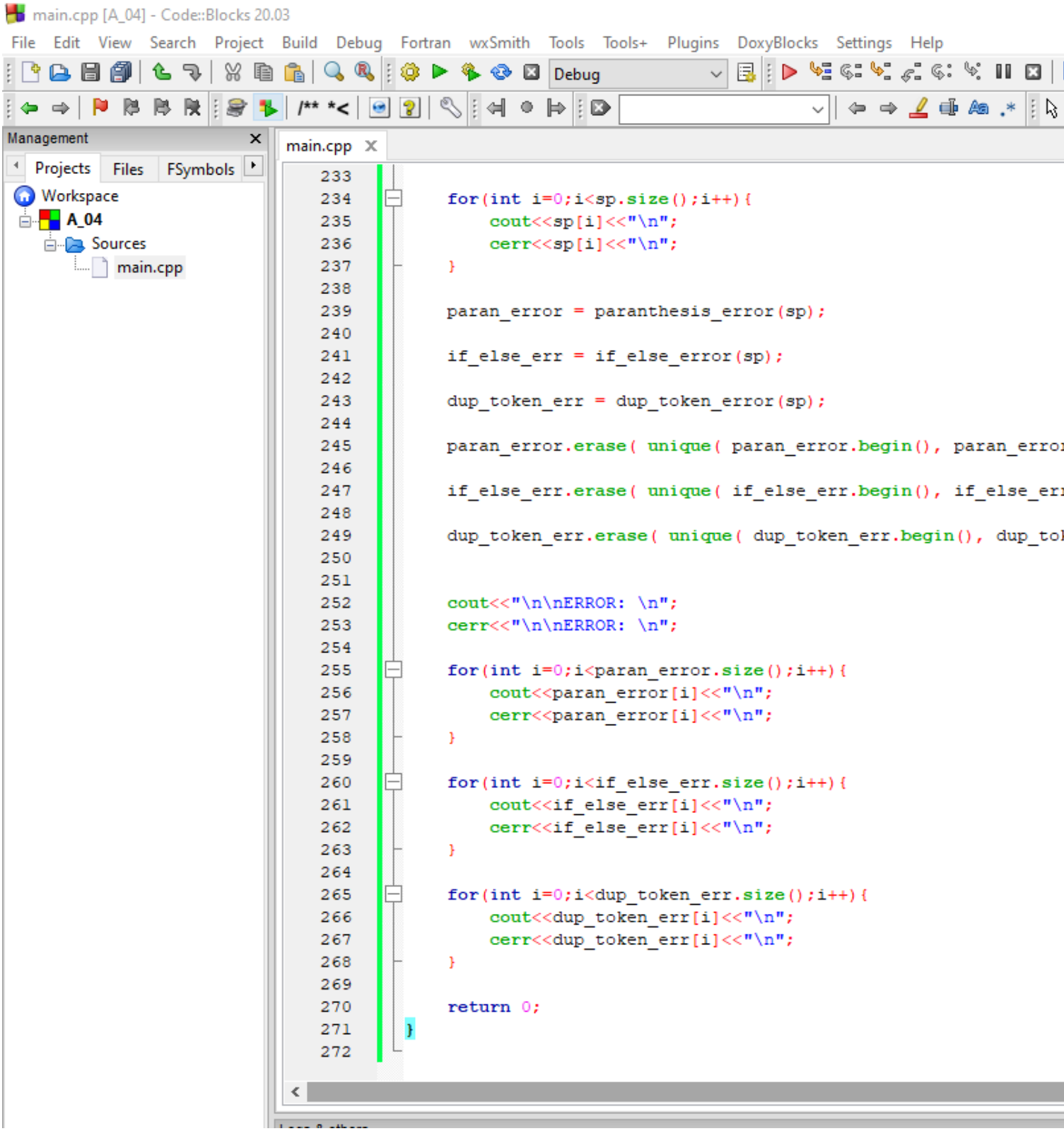

Fig. 7. Proposed Methodology

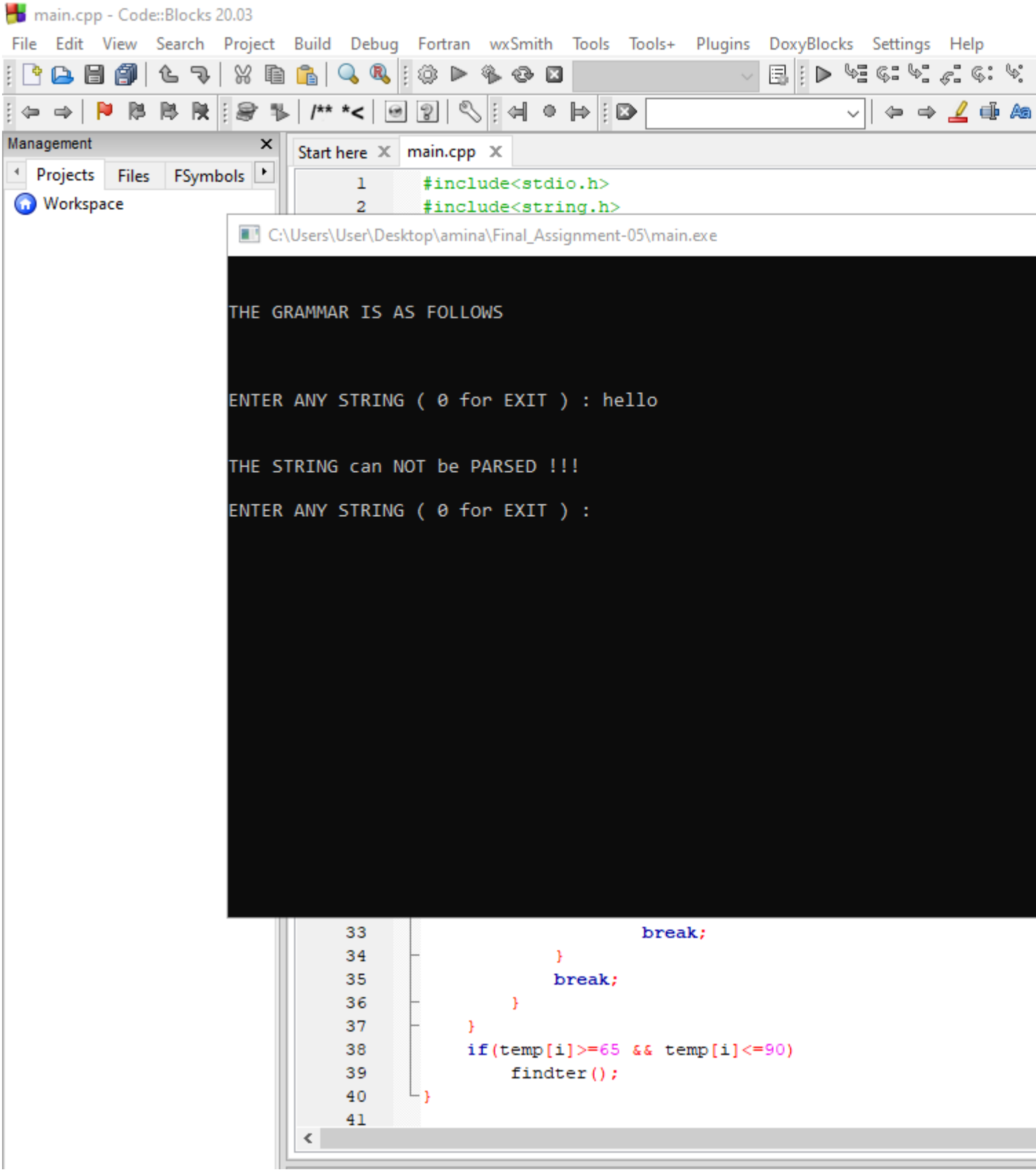

Fig. 8. Proposed Methodology

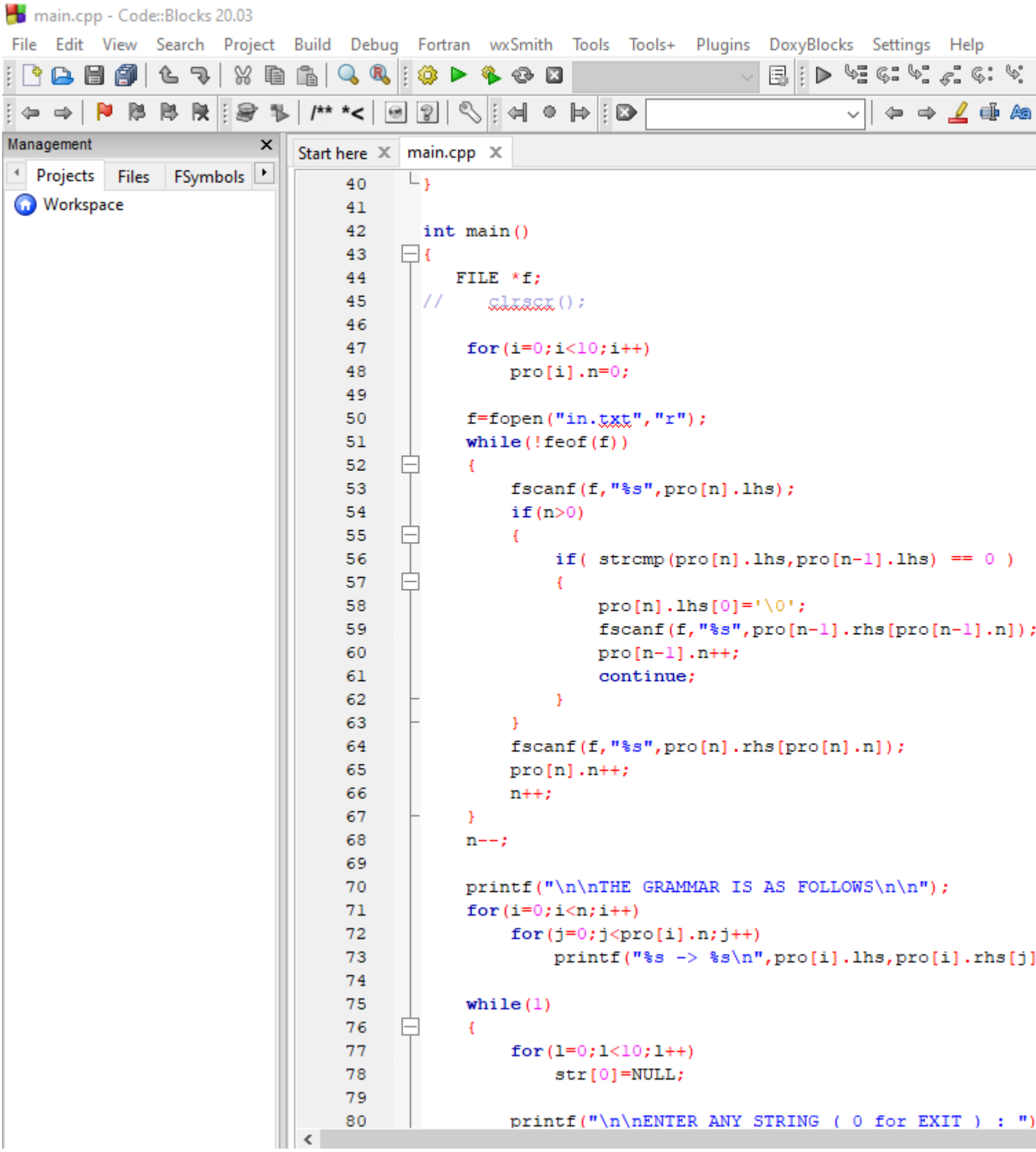

Fig. 9. Proposed Methodology

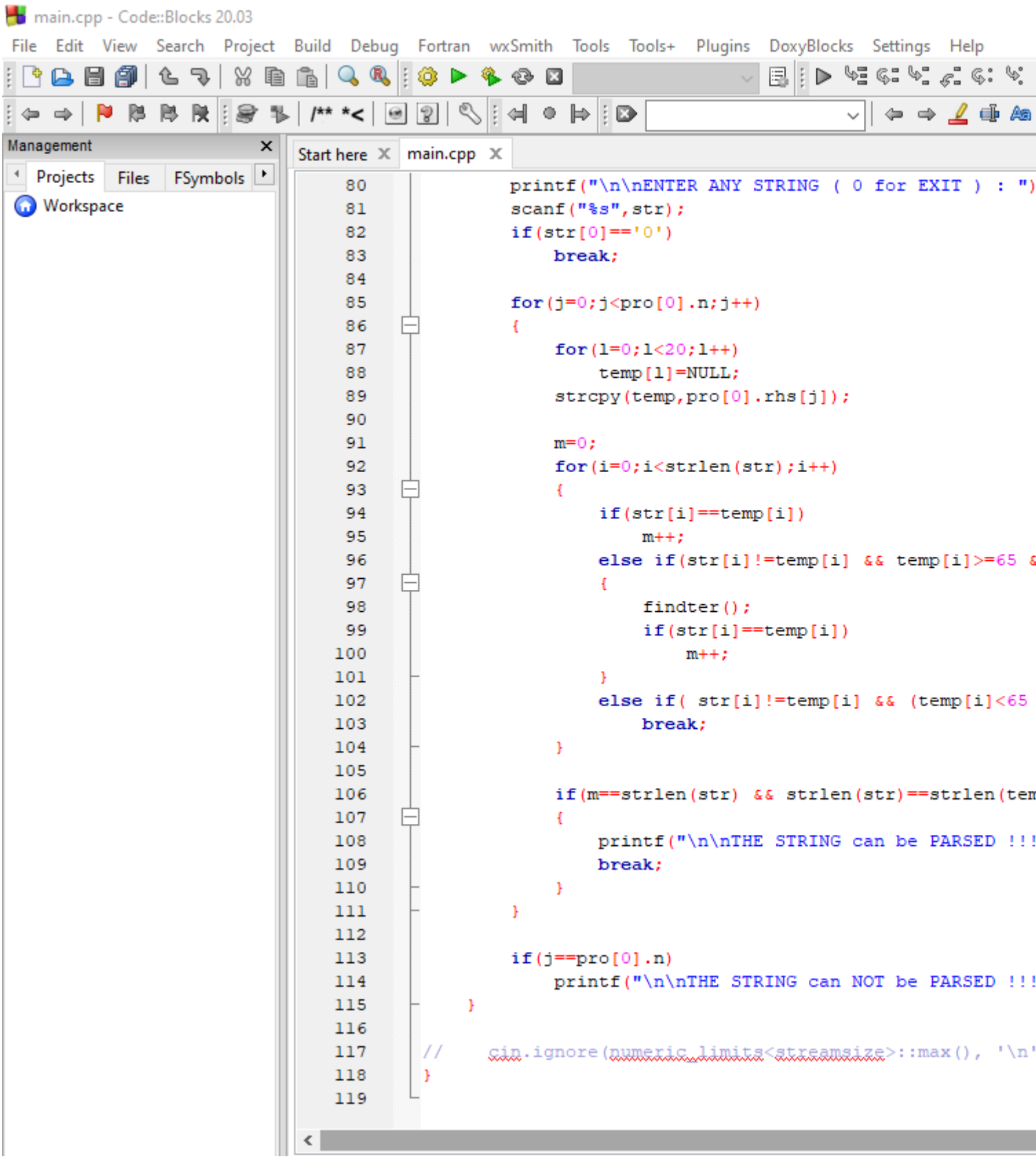

Fig. 10. Proposed Methodology

Given Grammer

$S \rightarrow aXd$

$X \rightarrow YZ$

$Y \rightarrow b \mid \varepsilon$

$Z \rightarrow cX \mid \varepsilon$

(1)

First of the given grammer

	$First$	$Follow$
S	a	$
X	b, c, ε	d
Y	b, ε	c, d
Z	c, ε	d

(2)

Parsing table LL(1)

	a	b	c	d	$
S	$S \rightarrow aXd$				
X		$X \rightarrow YZ$	$X \rightarrow YZ$		
Y		$Y \rightarrow b$	$Y \rightarrow \varepsilon$	$Y \rightarrow \varepsilon$	
Z			$Z \rightarrow cX$	$Z \rightarrow \varepsilon$	

Fig. 11. Proposed Methodology

input **abcd**

$S \rightarrow aXd$

$S \rightarrow aYZd \qquad using\ \ X \rightarrow YZ$

$S \rightarrow abZd \qquad using\ \ Y \rightarrow b$

$S \rightarrow abcXd \qquad using\ \ Z \rightarrow cX$

$S \rightarrow abc\varepsilon d \qquad using\ \ Z \rightarrow \varepsilon$

$S \rightarrow abcd \qquad using\ \ Z \rightarrow \varepsilon$

abcd is accepted by the given grammer.

Fig. 12. Proposed Methodology

(4)

LR(0) grammar

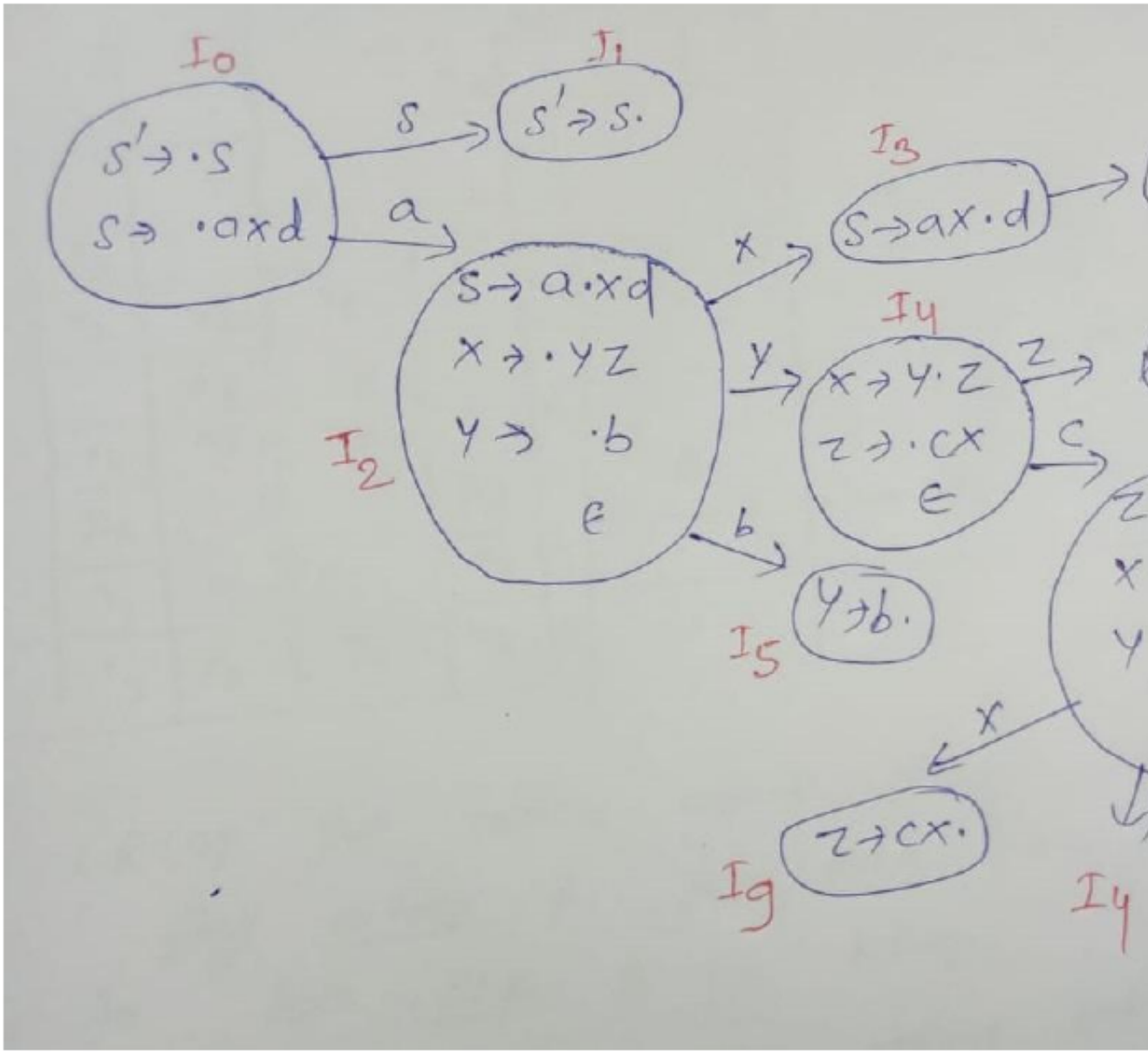

Fig. 13. Proposed Methodology

LR(O) Parsing Tabel

	Action	Action	Action	Action	Action	GOTO	GOTO	GOTO	GOTO
	a	b	c	d	$	S	X	Y	Z
0	S_2					1			
1					accept				
2	r_4	S_5 / r_4	r_4	r_4	r_4				
3				S_6					
4	r_6	r_6	S_8 / r_6	r_6	r_6				
5	r_3	r_3	r_3	r_3	r_3				
6	r_1	r_1	r_1	r_1	r_1				
7	r_2	r_2	r_2	r_2	r_2				
8		S_5					9	4	
9	r_5	r_5	r_5	r_5	r_5				

in the LR(0) parsing table Shift-reduce conflict occurs which can be seen in table.

Fig. 14. Proposed Methodology

(5)

augumented grammar for LR(1) Parsing table

Fig. 15. Proposed Methodology

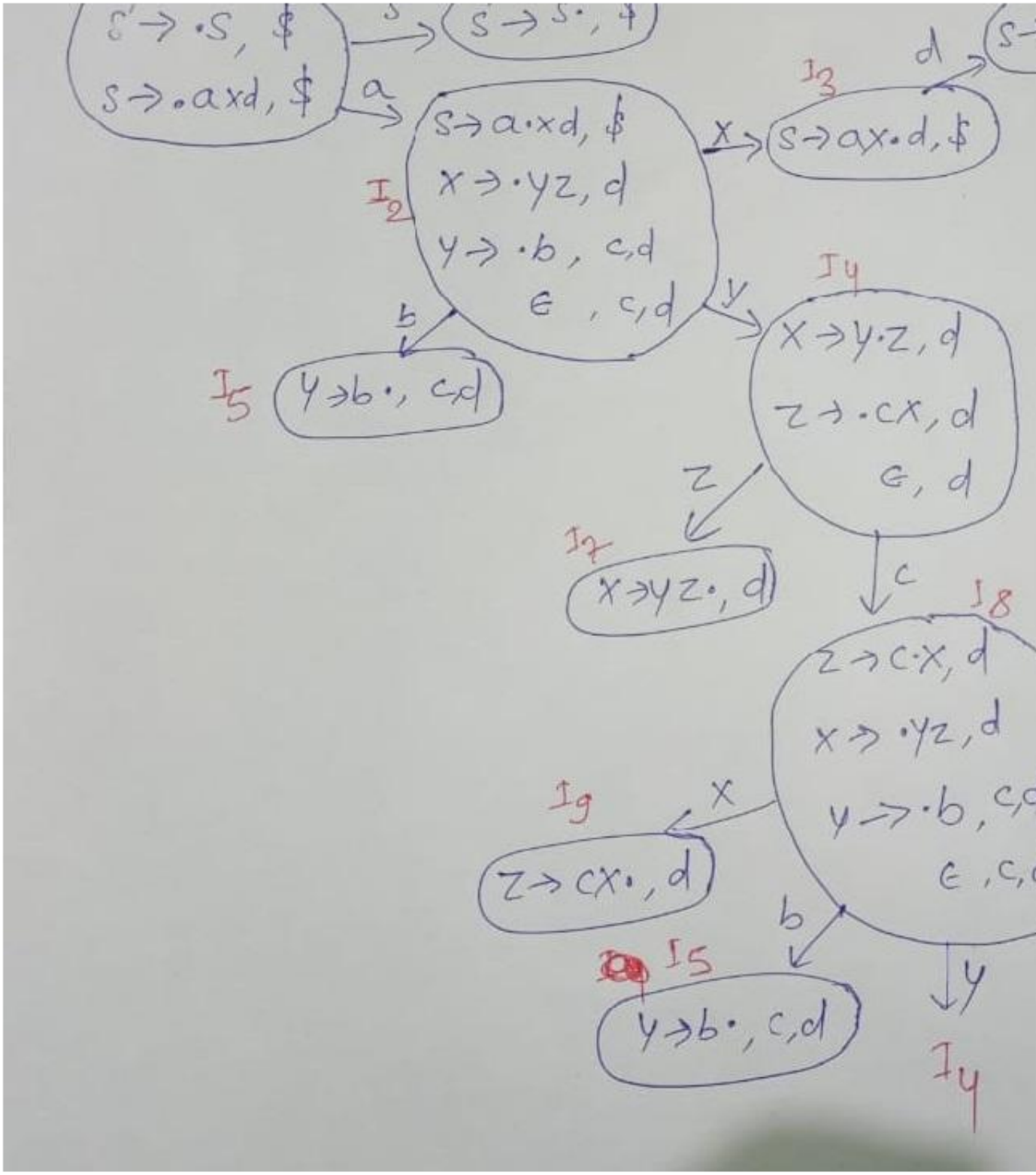

Fig. 16. Proposed Methodology

	Action	Action	Action	Action	Action	GOTO	GOTO	GOTO	GOTO
	a	b	c	d	$	S	X	Y	Z
0	S_2					1			
1					accept				
2		S_5	r_4	r_4			3	4	
3				S_6					
4			S_8	r_6					
5			r_3	r_3					
6					r_1				
7				r_2					
8		S_5	r_4	r_4			9	4	
9				r_5					

Fig. 17. Proposed Methodology

(6) moves of the parser for given input **abcd**

input	current input	stack	production	action		Remarks
abcd$	a	0	[0,a]	S_2		
bcd$	b	0a2				
bcd$	b	0a2	[2,b]	S_5		
bcd$	b	0a2b5				
cd$	c	0a2b5	[5,c]	r_3	$Y \rightarrow b$	two tim
cd$	c	0a2Y	[2,Y]	4		
cd$	c	0a2Y4	[4,c]	S_8		
d$	d	0a2Y4c8	[8,d]	r_4	$Y \rightarrow \varepsilon$	no time
d$	d	0a2Y4c8Y	[8,Y]	4		
d$	d	0a2Y4c8Y4	[4,d]	r_6	$Z \rightarrow \varepsilon$	no time
d$	d	0a2Y4c8Y4Z	[4,Z]	7		
d$	d	0a2Y4c8Y4Z7	[7,d]	r_2	$X \rightarrow YZ$	four tim
d$	d	0a2Y4c8X	[8,X]	9		
d$	d	0a2Y4c8X9	[9,d]	r_5	$Z \rightarrow cX$	four tim
d$	d	0a2Y4Z	[4,Z]	7		
d$	d	0a2Y4Z7	[7,d]	r_2	$X \rightarrow YZ$	four tim

Fig. 18. Proposed Methodology

$	$	0a2X3d6	[6,$]	r_1	$S \rightarrow aXd$	six time pop fr
$	$	0S	[0,S]	1		
$	$	0S1	[1,$]	accept		

Fig. 19. Proposed Methodology

main.cpp - Code::Blocks 20.03

File Edit View Search Project Build Debug Fortran wxSmith Tools Tools+ Plugins DoxyBlocks Settings Help

Management

Projects Files FSymbols

Workspace

Start here main.cpp

```cpp
#include<bits/stdc++.h>
using namespace std;

vector<string>sp,ke,ri;
map<string,string>mp,mpp;
string ans;

bool isTERMINAL(char a){
    if(a>='A' && a<='Z') return true;
    return false;
}

void FIRST(string key){

    string val = mp[key];

    if(isTERMINAL(val[0])){
        string p = "";
        p += val[0];
        FIRST(p);
    }
    else{
        ans += val[0];
        ans += ",";
        int flag = 0;
        for(int i=0;i<val.size();i++){
            if(val[i]=='|'){
                flag = 1;
                continue;
            }
            if(flag){
                ans += val[i];
            }
        }

    }

}

void FOLLOW(string key,int z){

```

Fig. 20. Proposed Methodology

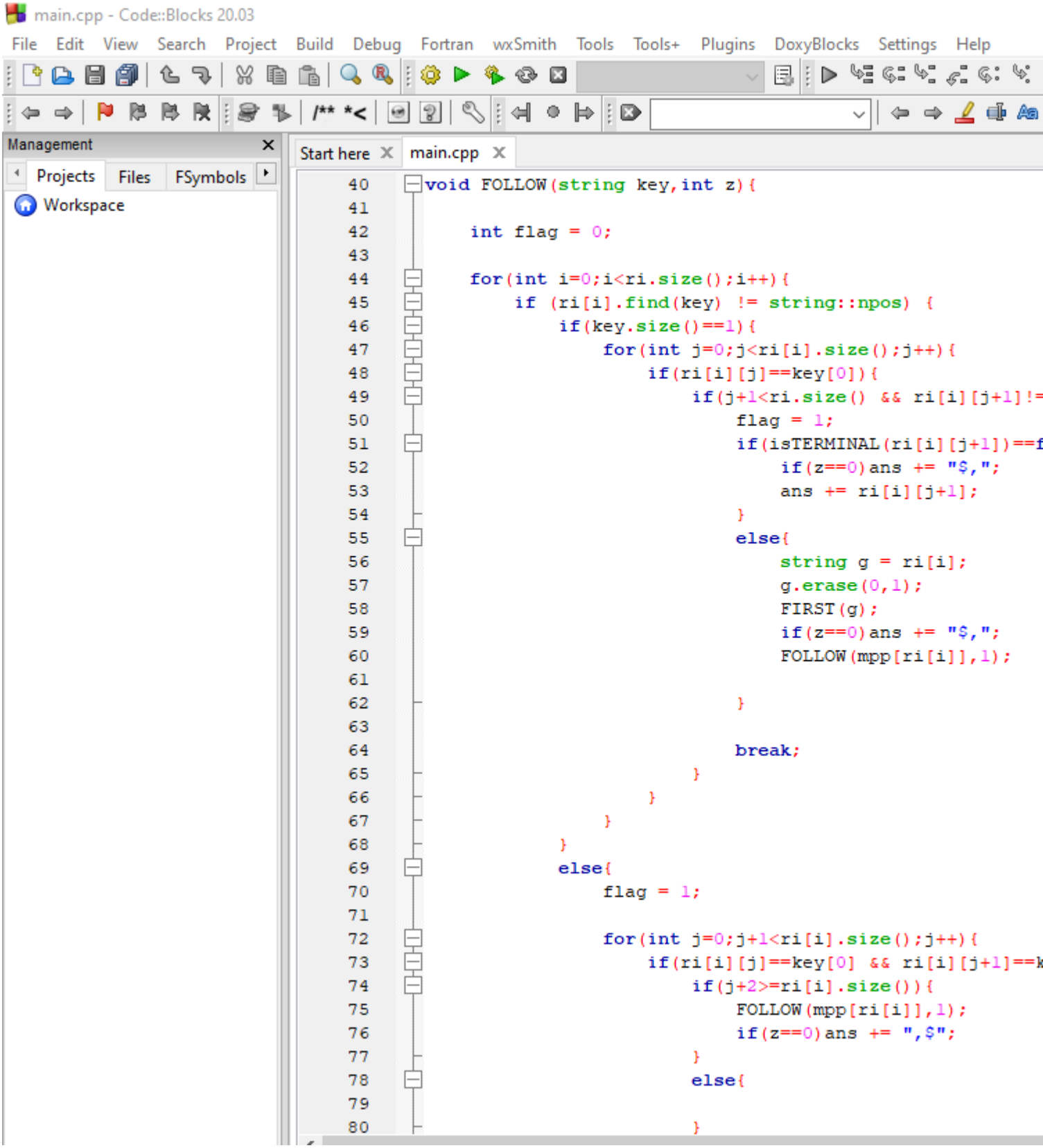

Fig. 21. Proposed Methodology

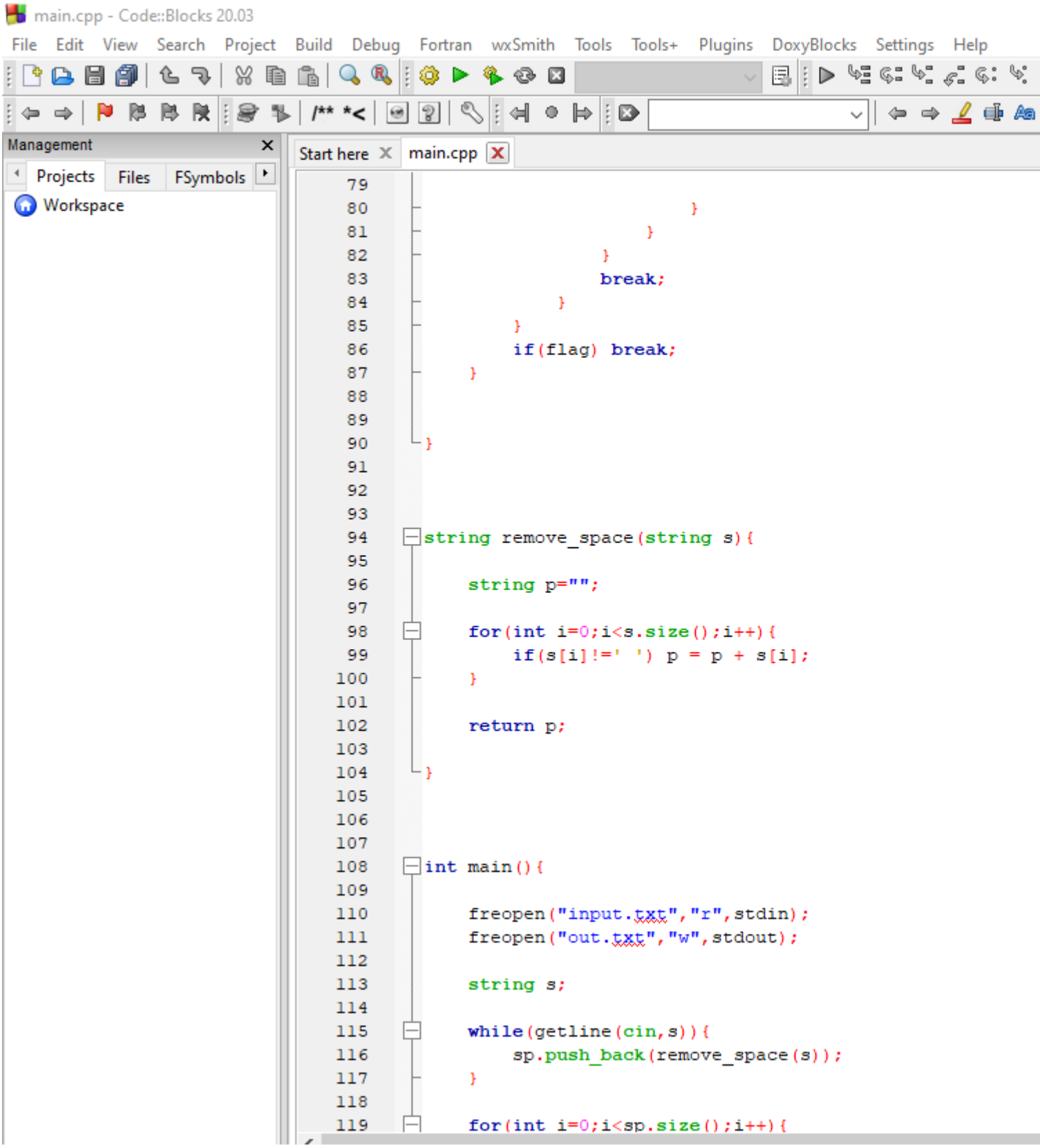

Fig. 22. Proposed Methodology

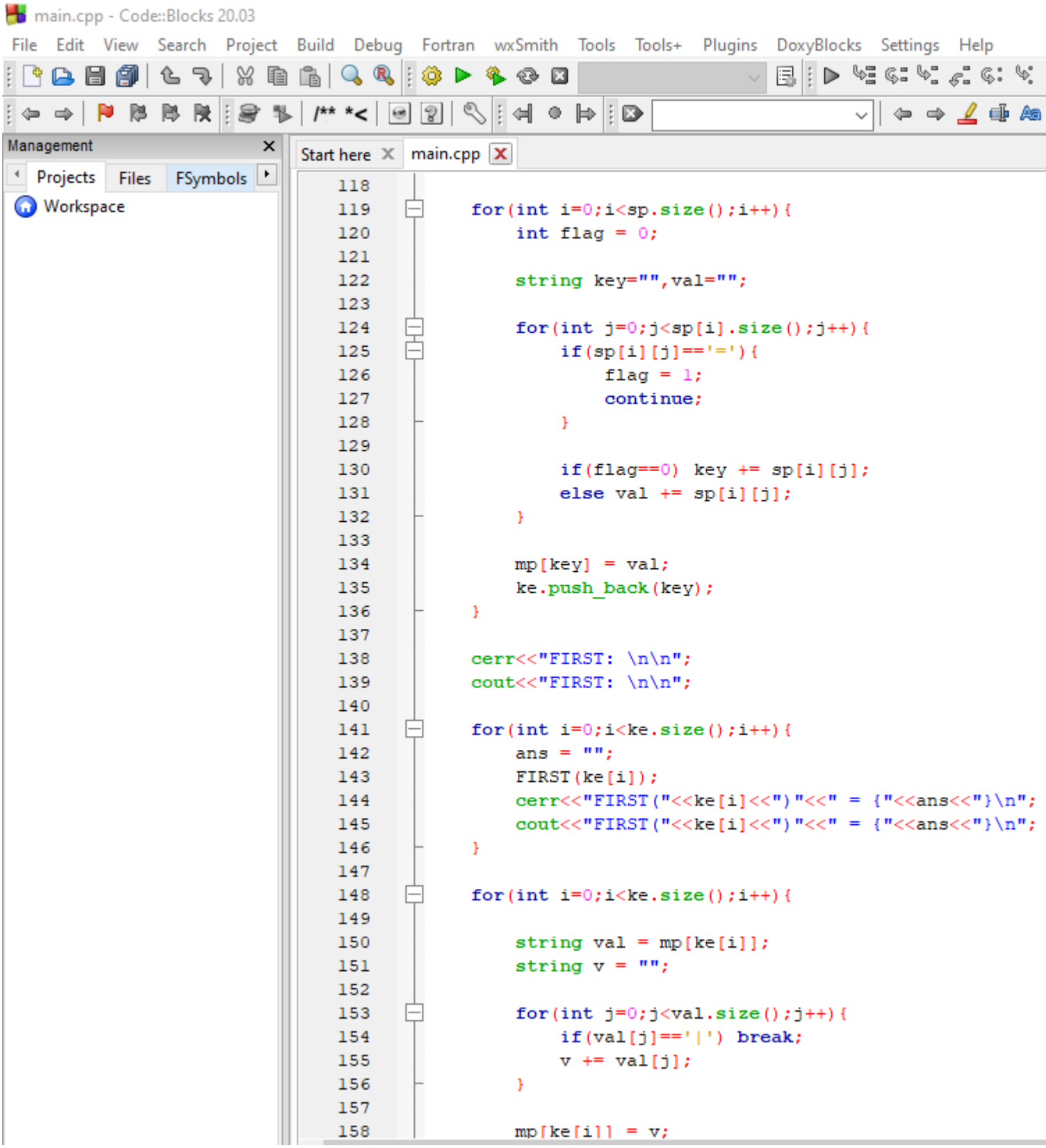

Fig. 23. Proposed Methodology

main.cpp - Code::Blocks 20.03

File Edit View Search Project Build Debug Fortran wxSmith Tools Tools+ Plugins DoxyBlocks Settings Help

Management

Projects Files FSymbols

Workspace

Start here | main.cpp

```cpp
    cerr<<"FIRST: \n\n";
    cout<<"FIRST: \n\n";

    for(int i=0;i<ke.size();i++){
        ans = "";
        FIRST(ke[i]);
        cerr<<"FIRST("<<ke[i]<<")"<<" = {"<<ans<<"}\n";
        cout<<"FIRST("<<ke[i]<<")"<<" = {"<<ans<<"}\n";
    }

    for(int i=0;i<ke.size();i++){

        string val = mp[ke[i]];
        string v = "";

        for(int j=0;j<val.size();j++){
            if(val[j]=='|') break;
            v += val[j];
        }

        mp[ke[i]] = v;
        mpp[v] = ke[i];
        ri.push_back(v);
    }

    cerr<<"\nFOLLOW: \n\n";
    cout<<"\nFOLLOW: \n\n";


    for(int i=0;i<ke.size();i++){
        ans = "";

        FOLLOW(ke[i],0);
        cerr<<"FOLLOW("<<ke[i]<<")"<<" = {"<<ans<<"}\n";
        cout<<"FOLLOW("<<ke[i]<<")"<<" = {"<<ans<<"}\n";
    }


}
```

Fig. 24. Proposed Methodology