

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 descent parser can be constructed 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. Joshua 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 Ólafóttir, Hrafn Loftsson (Department of Computer Science) . Both present open-source, wide-coverage context-free grammar (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 that what 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. Magnetism Japan, p. 301, 1982].
- [7] M. Young, The Technical Writer's Handbook. Mill Valley, CA: University Science, 1989.

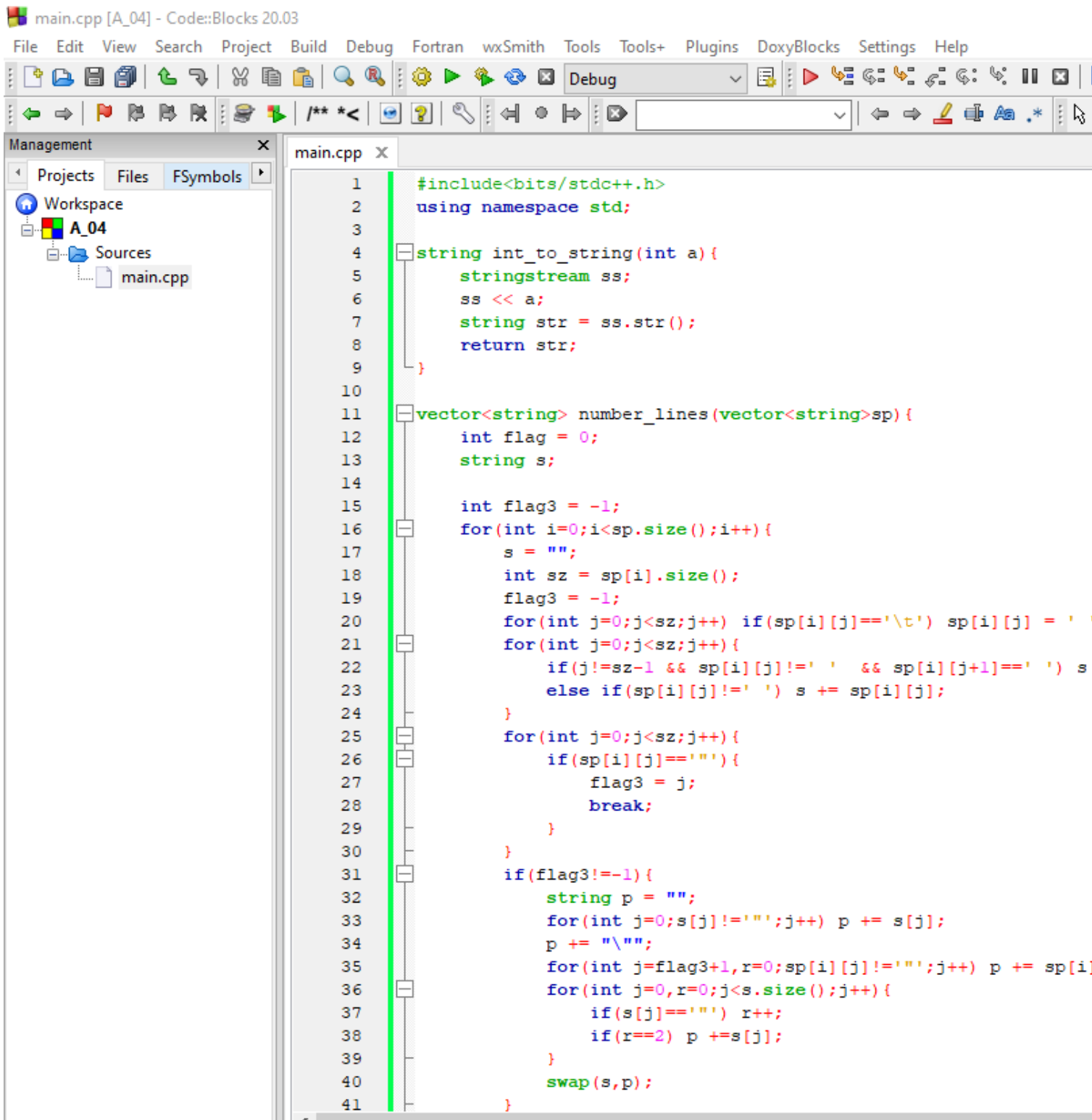


Fig. 1. Proposed Methodology

```

40         swap(s,p);
41     }
42     swap(sp[i],s);
43 }
44
45 vector<string> spl;
46
47 int flag1 = 0, flag2 = 0;
48 for(int i=0; i<sp.size(); i++){
49     string str = int_to_string(i+1);
50     int sz = sp[i].size();
51     if(sz==0){
52         spl.push_back(str);
53         continue;
54     }
55     for(int j=0; j<sz; j++){
56         if(j!=sz-1 && sp[i][j]=='/' && sp[i][j+1]=='/'){
57             flag1 = 1;
58             for(int k=0; k<j; k++){
59                 cout<<sp[i][k];
60                 cerr<<sp[i][k];
61             }
62             break;
63         }
64         if(j!=sz-1 && sp[i][j]=='/' && sp[i][j+1]=='*'){
65             flag2 = 1;
66             for(int k=0; k<j; k++){
67                 cout<<sp[i][k];
68                 cerr<<sp[i][k];
69             }
70         }
71         if(j!=sz-1 && sp[i][j]=='*' && sp[i][j+1]=='/'){
72             flag2 = 0;
73             flag1 = 1;
74             break;
75         }
76     }
77     if(flag1){
78         flag1 = 0;
79         spl.push_back(str);
80         continue;

```

Fig. 2. Proposed Methodology

```
main.cpp [A_04] - Code::Blocks 20.03
File Edit View Search Project Build Debug Fortran wxSmith Tools Tools+ Plugins DoxyBlocks Settings Help

Management
  Projects Files FSymbols
    Workspace
      A_04
        Sources
          main.cpp

main.cpp x
79         spl.push_back(str);
80         continue;
81     }
82     if(flag2){
83         spl.push_back(str);
84         continue;
85     }
86     str = str + " " + sp[i];
87     spl.push_back(str);
88 }
89
90 return spl;
91
92 }
93
94 vector<string> paranthesis_error(vector<string> sp){
95
96     stack<int>st;
97     vector<string>err;
98
99     for(int i=0;i<sp.size();i++){
100         for(int j=0;j<sp[i].size();j++){
101             if(sp[i][j]=='(') st.push(i+1);
102             else if(sp[i][j]==')'){
103                 if( !st.empty() ) st.pop();
104                 else err.push_back("Error: Misplaced ')' at line");
105             }
106         }
107     }
108
109     if( !st.empty() ) err.push_back("Error: Not Balanced Parenthesis");
110
111     return err;
112 }
113
114
115 vector<string> if_else_error(vector<string> sp){
116
117     bool ok = false;
118     vector<string>err;
119     int sz = sp.size();
```

Fig. 3. Proposed Methodology

```

118     vector<string>err;
119     int sz = sp.size();
120     for(int i=0;i<sz;i++){
121         if(sz<4)continue;
122         int x = sp[i].size();
123         for(int j=0;j<x;j++){
124             if(j+1<x && sp[i][j]=='i' && sp[i][j+1]=='f') ok =
125             if(j+3<x && sp[i][j]=='e' && sp[i][j+1]=='l' && sp
126                 if( ok ){
127                     ok = false;
128                     continue;
129                 }
130                 else err.push_back("Error: Not Matched else at
131             }
132         }
133     }
134     return err;
135 }
136
137
138 bool comp(char a){
139     if(a=='=' || a=='>' || a=='<' ) return false;
140
141     return true;
142 }
143
144 bool col(char a){
145
146     if(a==',' || a==';' || a=='+' || a=='-' || a=='*' || a=='/'
147     return false;
148 }
149
150
151 vector<string> dup_token_error(vector<string> sp){
152
153     vector<string>err;
154     int sz = sp.size();
155
156     for(int j=0;j<sz;j++){
157
158         string p = "",s=sp[j];

```

Fig. 4. Proposed Methodology

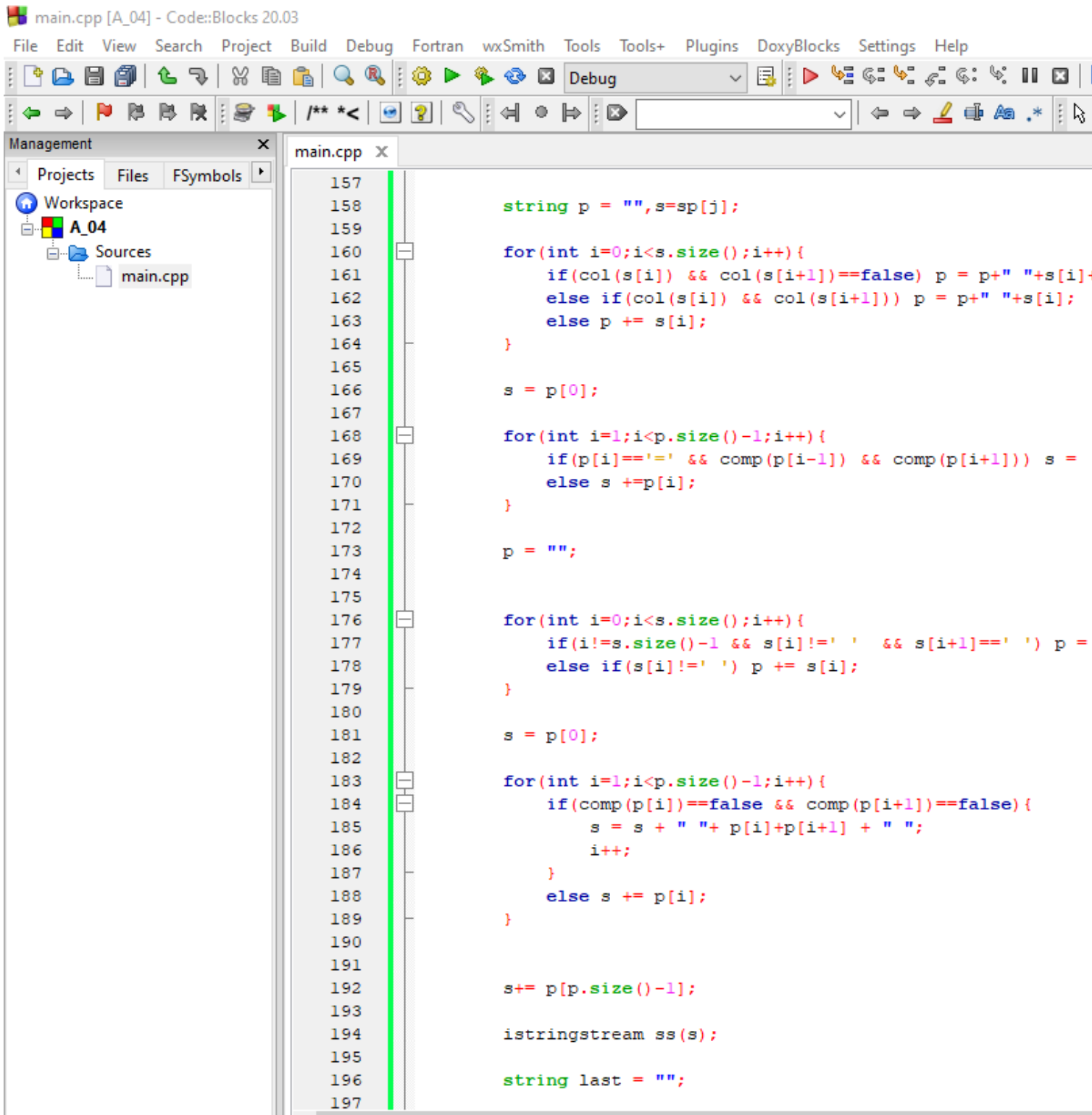


Fig. 5. Proposed Methodology

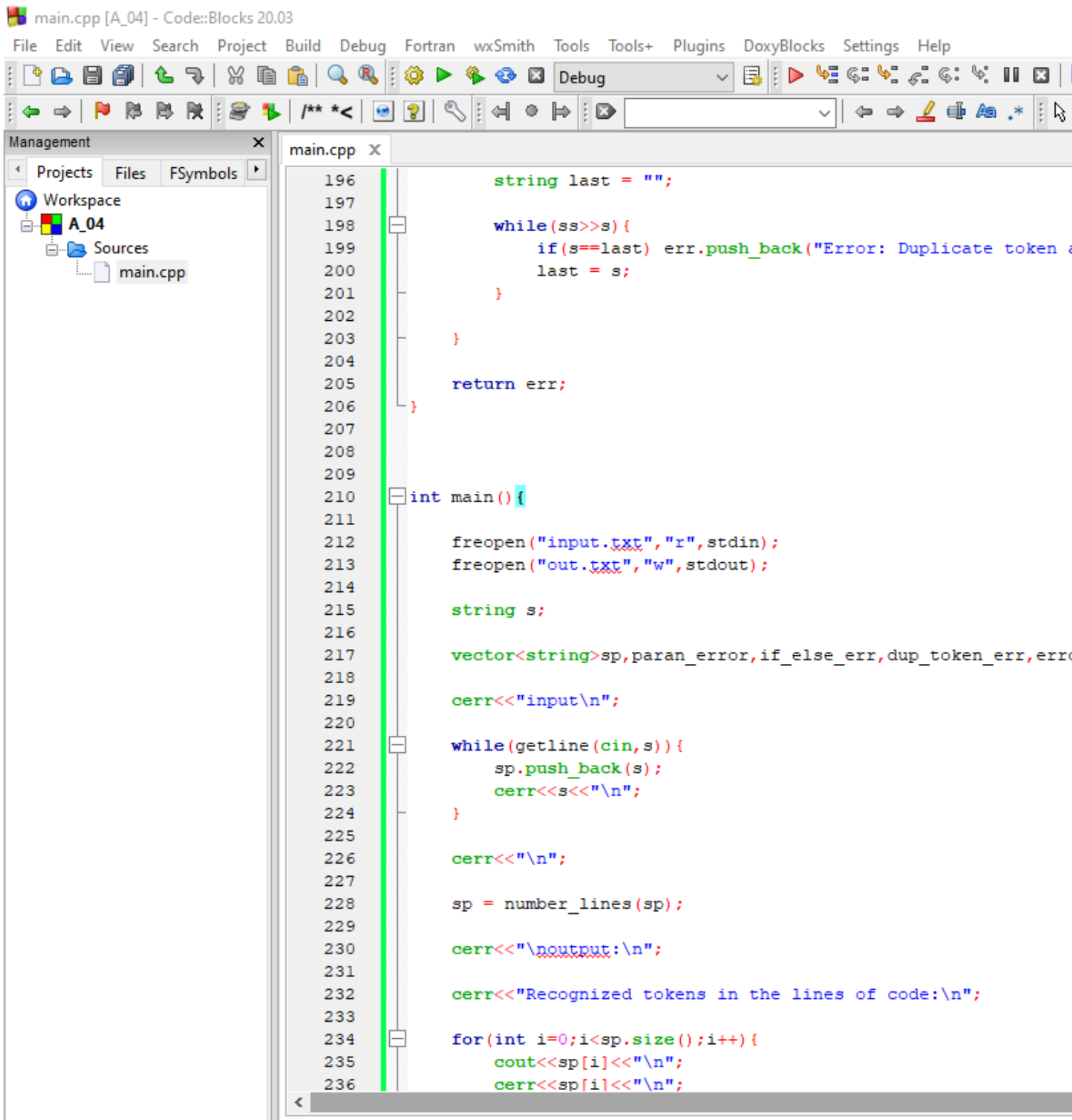


Fig. 6. Proposed Methodology

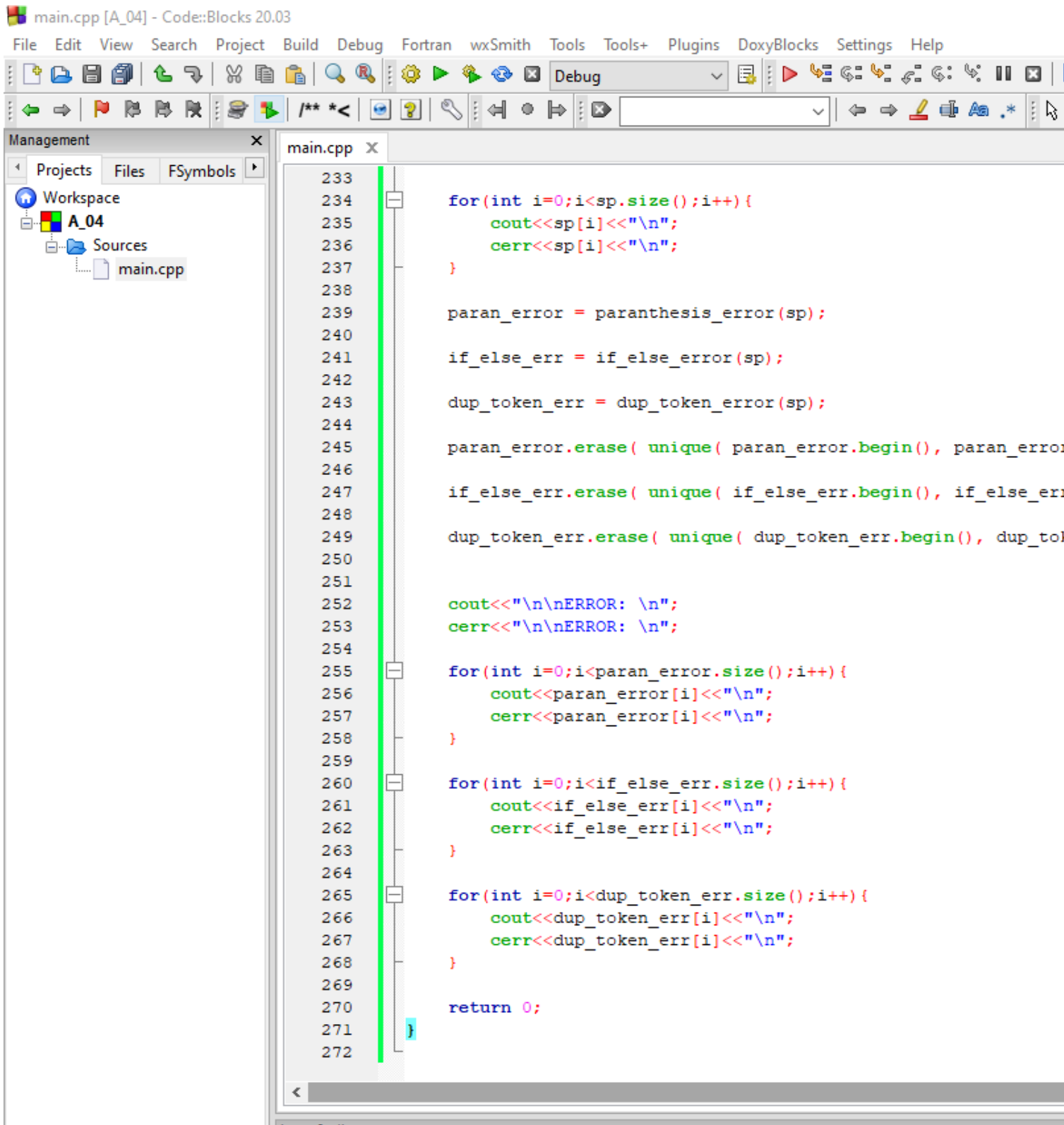


Fig. 7. Proposed Methodology

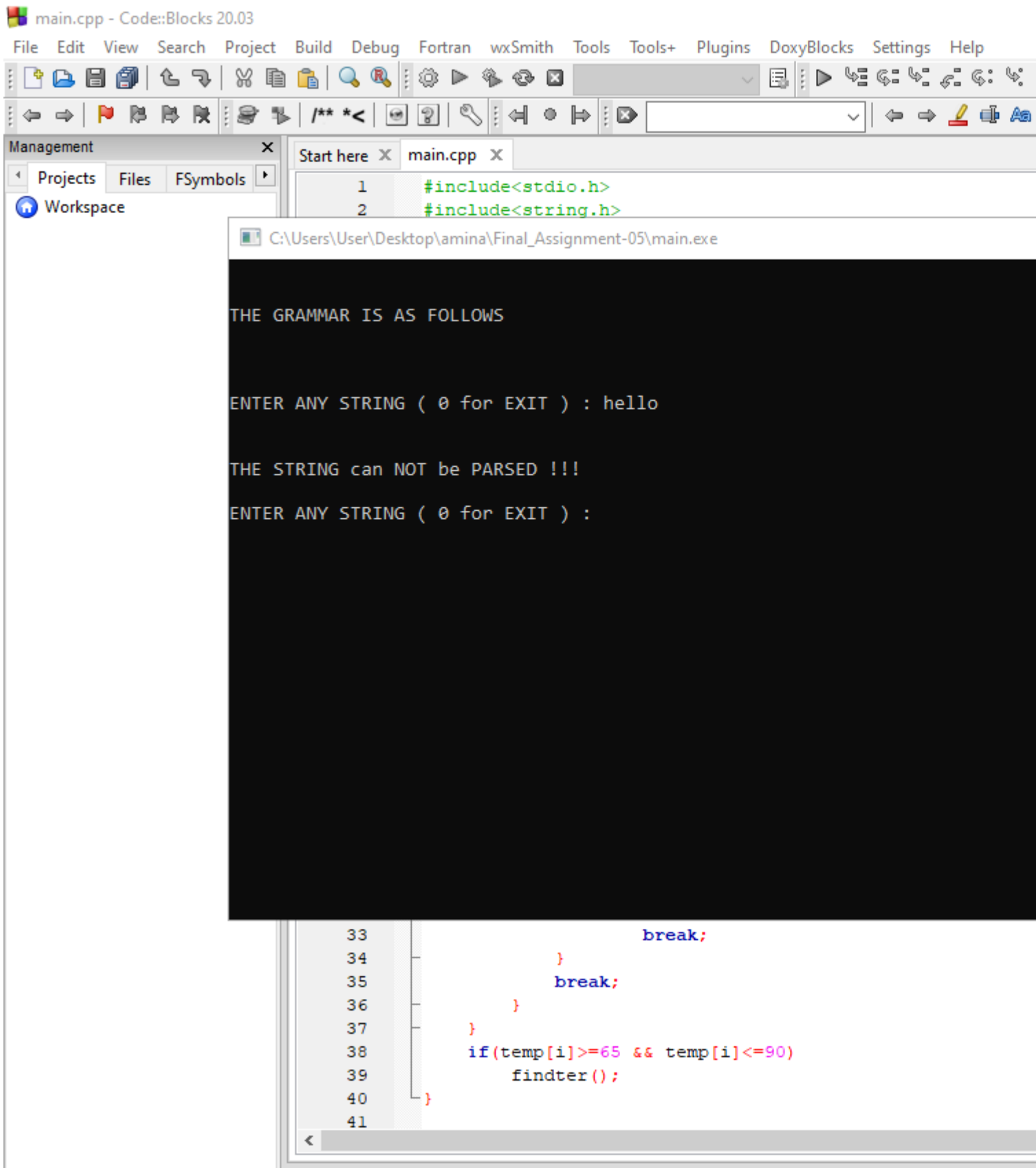
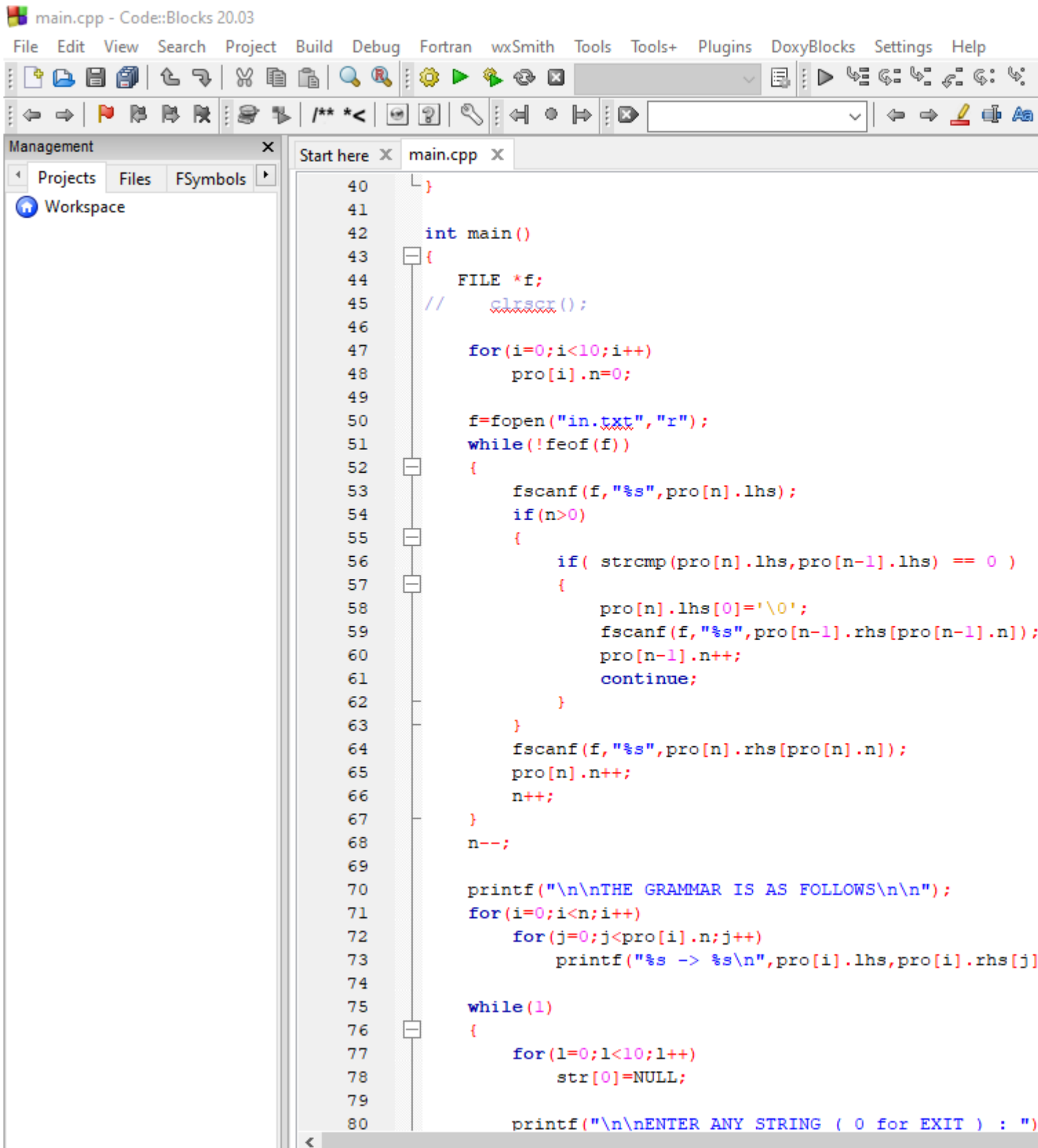


Fig. 8. Proposed Methodology



```
40 }
41
42 int main()
43 {
44     FILE *f;
45     // clrscr();
46
47     for(i=0;i<10;i++)
48         pro[i].n=0;
49
50     f=fopen("in.txt","r");
51     while(!feof(f))
52     {
53         fscanf(f,"%s",pro[n].lhs);
54         if(n>0)
55         {
56             if( strcmp(pro[n].lhs,pro[n-1].lhs) == 0 )
57             {
58                 pro[n].lhs[0]='\0';
59                 fscanf(f,"%s",pro[n-1].rhs[pro[n-1].n]);
60                 pro[n-1].n++;
61                 continue;
62             }
63         }
64         fscanf(f,"%s",pro[n].rhs[pro[n].n]);
65         pro[n].n++;
66         n++;
67     }
68     n--;
69
70     printf("\n\nTHE GRAMMAR IS AS FOLLOWS\n\n");
71     for(i=0;i<n;i++)
72         for(j=0;j<pro[i].n;j++)
73             printf("%s -> %s\n",pro[i].lhs,pro[i].rhs[j]);
74
75     while(1)
76     {
77         for(l=0;l<10;l++)
78             str[l]=NULL;
79
80         printf("\n\nENTER ANY STRING ( 0 for EXIT ) : ")
```

Fig. 9. Proposed Methodology

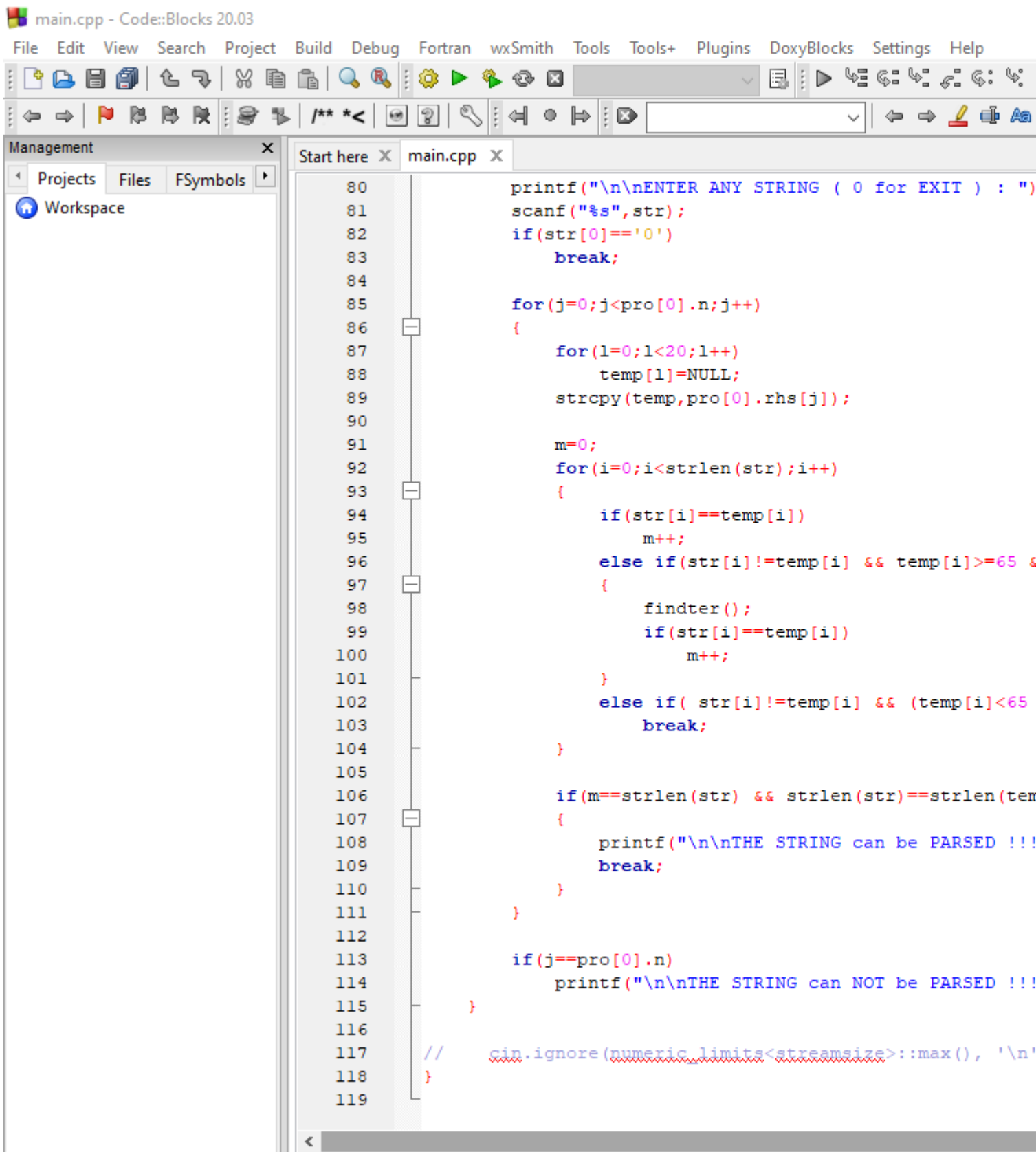


Fig. 10. Proposed Methodology

Given Grammer

$$S \rightarrow aXd$$

$$X \rightarrow YZ$$

$$Y \rightarrow b \mid \varepsilon$$

$$Z \rightarrow cX \mid \varepsilon$$

(1)

First of the given grammer

| | <i>First</i> | <i>Follow</i> |
|----------|----------------|---------------|
| <i>S</i> | <i>a</i> | <i>S</i> |
| <i>X</i> | <i>b, c, ε</i> | <i>d</i> |
| <i>Y</i> | <i>b, ε</i> | <i>c, d</i> |
| <i>Z</i> | <i>c, ε</i> | <i>d</i> |

(2)

Parsing table LL(1)

| | a | b | c | d | S |
|---|---------------------|--------------------|-----------------------------|-----------------------------|---|
| 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**

$$S \rightarrow aXd$$

$$S \rightarrow aYZd \quad \text{using } X \rightarrow YZ$$

$$S \rightarrow abZd \quad \text{using } Y \rightarrow b$$

$$S \rightarrow abcXd \quad \text{using } Z \rightarrow cX$$

$$S \rightarrow abc\varepsilon d \quad \text{using } Z \rightarrow \varepsilon$$

$$S \rightarrow abcd \quad \text{using } Z \rightarrow \varepsilon$$

abcd is accepted by the given grammer.

Fig. 12. Proposed Methodology

(4)

LR(0) grammar

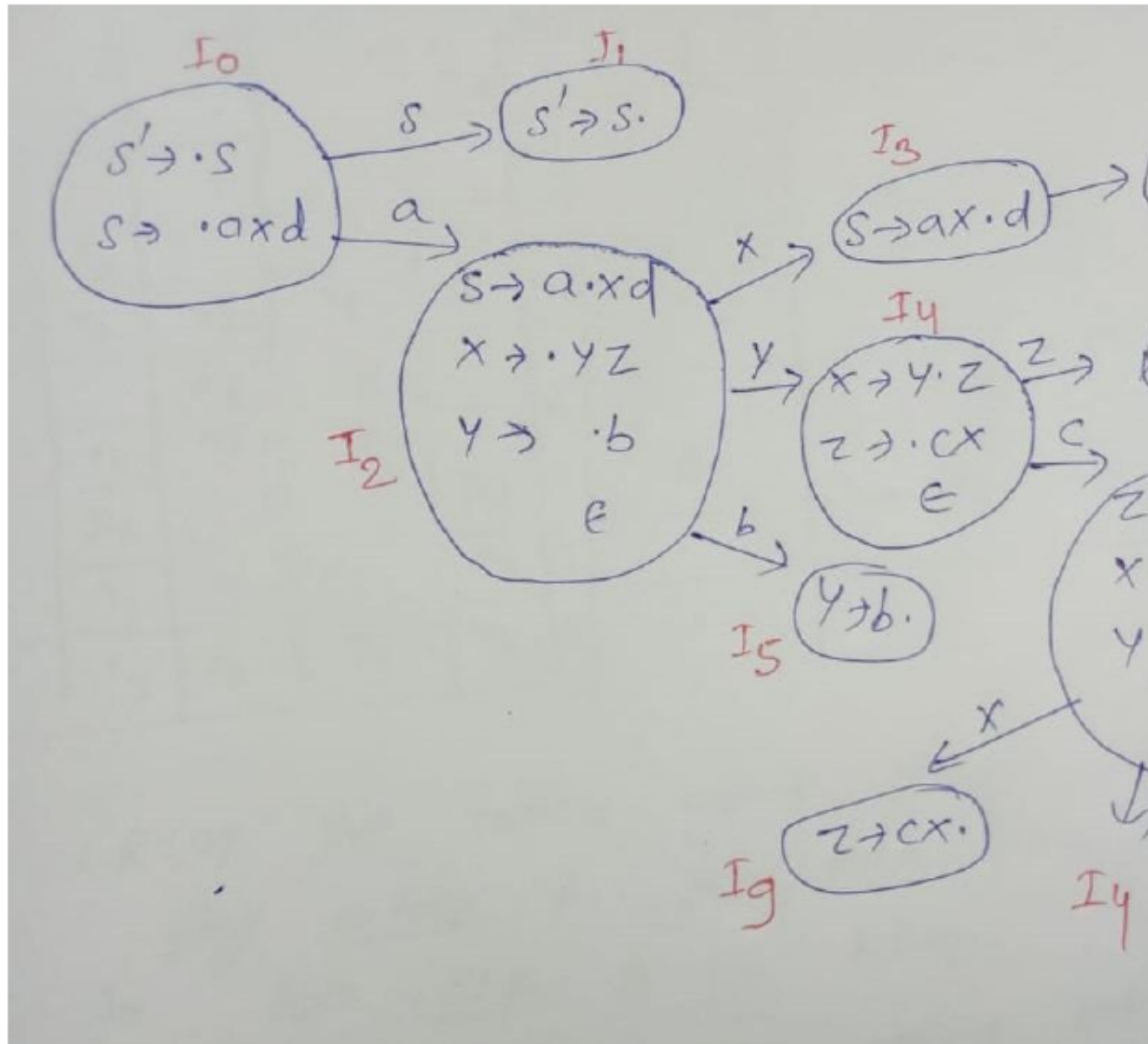


Fig. 13. Proposed Methodology

LR(O) Parsing Tabel

| | Action | Action | Action | Action | Action | GOTO | GOTO | GOTO | GOTO |
|---|--------|-------------|-------------|--------|--------|------|------|------|------|
| | a | b | c | d | S | 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

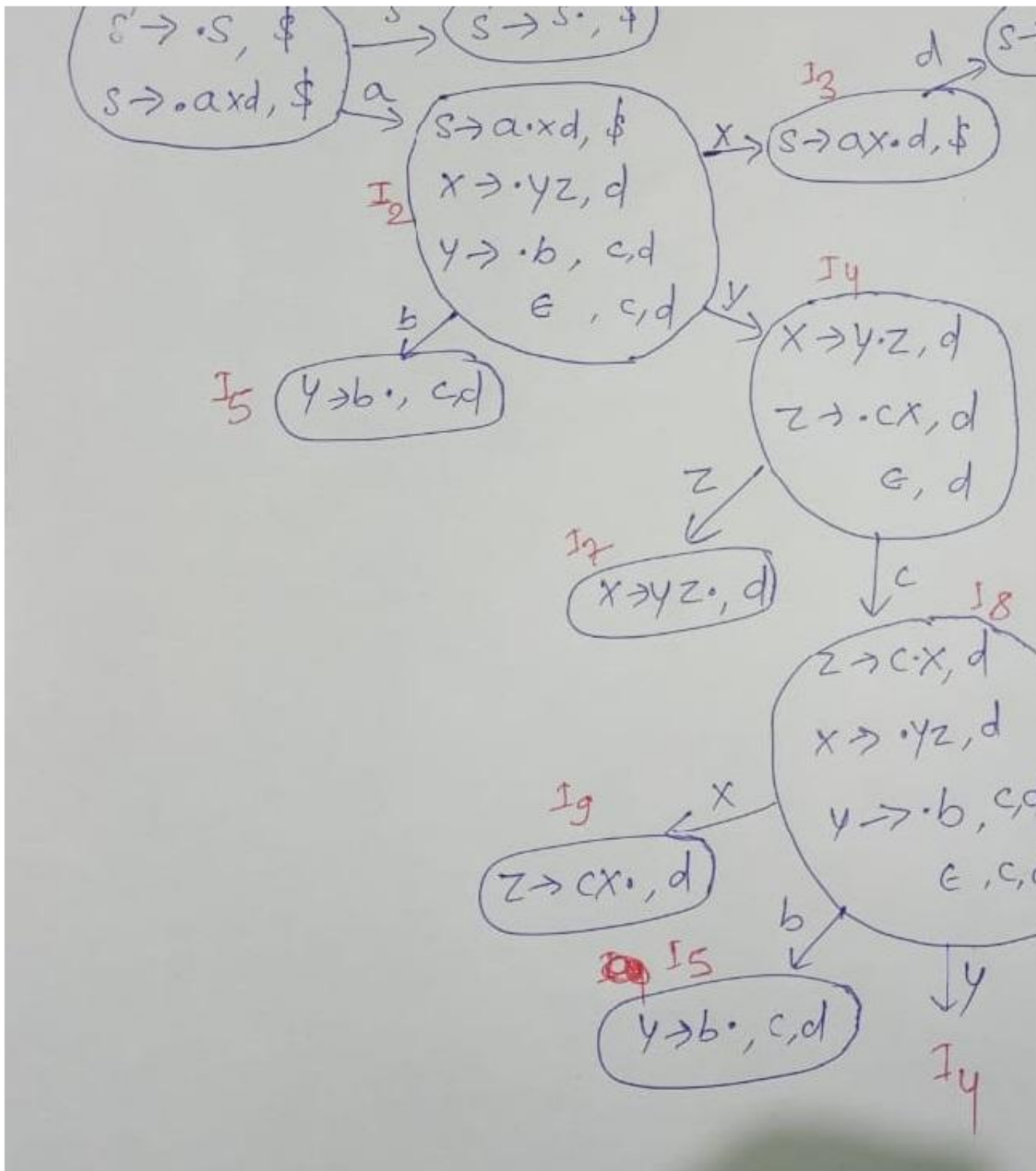


Fig. 16. Proposed Methodology

| | 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 time |
| 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 f |
| \$ | \$ | 0S | [0,\$] | 1 | | |
| \$ | \$ | 0S1 | [1,\$] | accept | | |

Fig. 19. Proposed Methodology

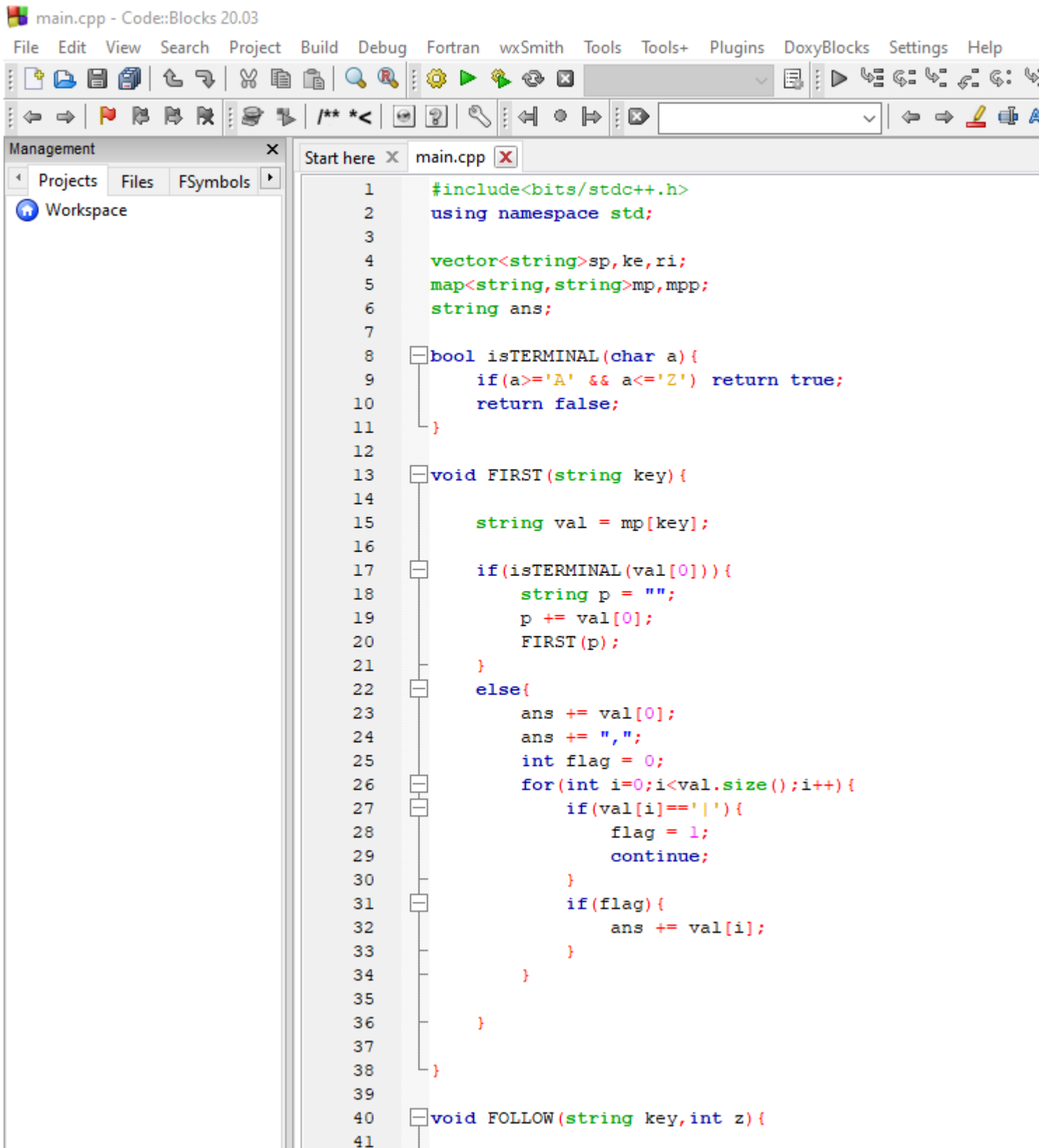


Fig. 20. Proposed Methodology

main.cpp - Code::Blocks 20.03

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

Management x Start here x main.cpp x

Projects Files FSymbols Workspace

```
40 void FOLLOW(string key, int z) {
41
42     int flag = 0;
43
44     for(int i=0;i<ri.size();i++){
45         if (ri[i].find(key) != string::npos) {
46             if(key.size()==1){
47                 for(int j=0;j<ri[i].size();j++){
48                     if(ri[i][j]==key[0]){
49                         if(j+1<ri.size() && ri[i][j+1]!=
50                             flag = 1;
51                             if(isTERMINAL(ri[i][j+1])==f
52                                 if(z==0) ans += "$,";
53                                 ans += ri[i][j+1];
54                             }
55                             else{
56                                 string g = ri[i];
57                                 g.erase(0,1);
58                                 FIRST(g);
59                                 if(z==0) ans += "$,";
60                                 FOLLOW(mpp[ri[i]],1);
61                             }
62                         }
63                     }
64                     break;
65                 }
66             }
67         }
68     }
69     else{
70         flag = 1;
71
72         for(int j=0;j+1<ri[i].size();j++){
73             if(ri[i][j]==key[0] && ri[i][j+1]==k
74                 if(j+2>=ri[i].size()){
75                     FOLLOW(mpp[ri[i]],1);
76                     if(z==0) ans += ",$";
77                 }
78                 else{
79
80
```

Fig. 21. Proposed Methodology

main.cpp - Code::Blocks 20.03

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

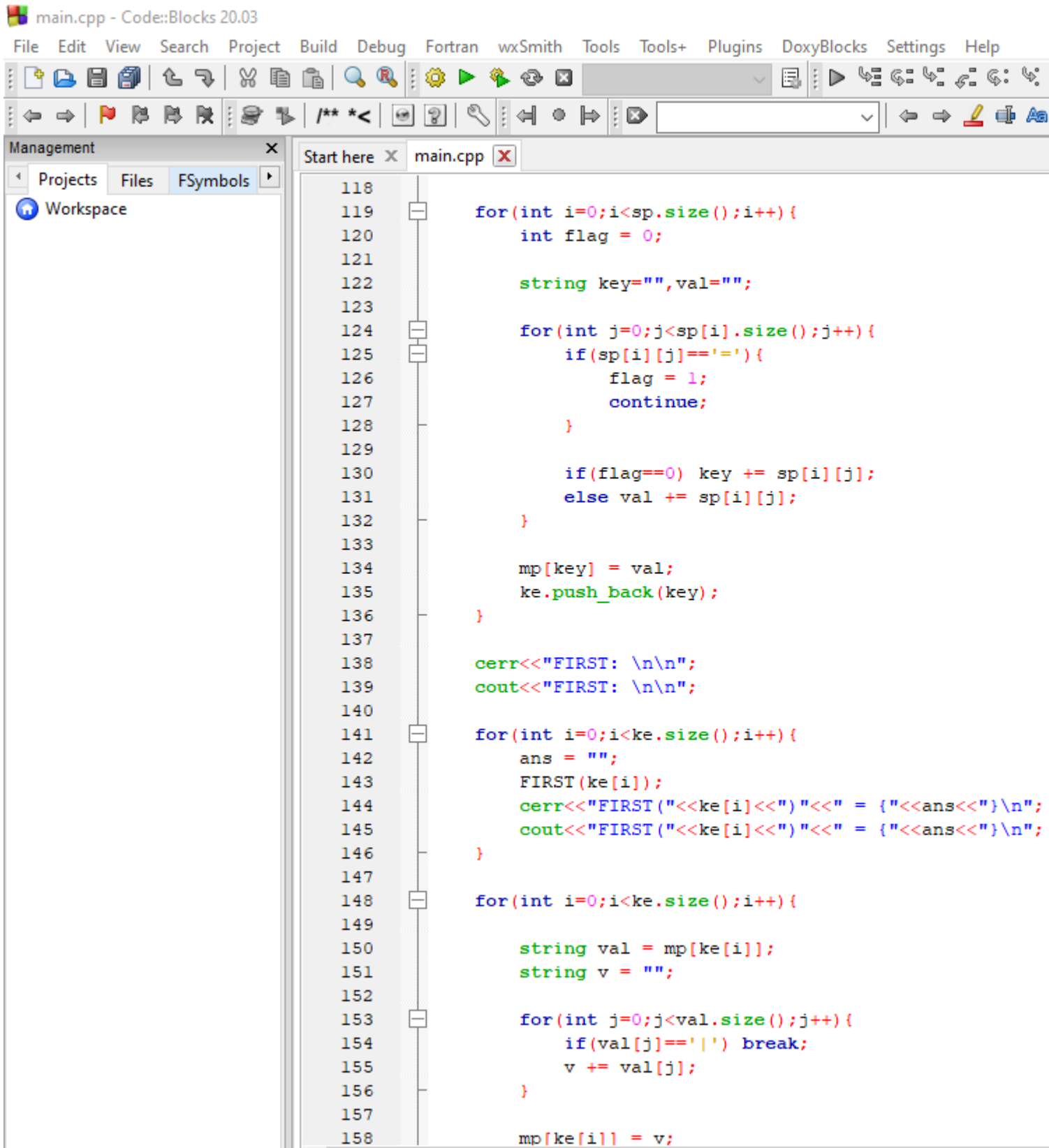
Management x Start here x main.cpp x

Projects Files FSymbols

Workspace

```
79
80
81
82
83     break;
84 }
85 }
86     if(flag) break;
87 }
88
89
90 }
91
92
93
94 string remove_space(string s){
95
96     string p="";
97
98     for(int i=0;i<s.size();i++){
99         if(s[i]!=' ') p = p + s[i];
100     }
101
102     return p;
103 }
104
105
106
107
108 int main(){
109
110     freopen("input.txt","r",stdin);
111     freopen("out.txt","w",stdout);
112
113     string s;
114
115     while(getline(cin,s)){
116         sp.push_back(remove_space(s));
117     }
118
119     for(int i=0;i<sp.size();i++){
```

Fig. 22. Proposed Methodology



```

118
119     for(int i=0;i<sp.size();i++){
120         int flag = 0;
121
122         string key="",val="";
123
124         for(int j=0;j<sp[i].size();j++){
125             if(sp[i][j]==' '){
126                 flag = 1;
127                 continue;
128             }
129
130             if(flag==0) key += sp[i][j];
131             else val += sp[i][j];
132         }
133
134         mp[key] = val;
135         ke.push_back(key);
136     }
137
138     cerr<<"FIRST: \n\n";
139     cout<<"FIRST: \n\n";
140
141     for(int i=0;i<ke.size();i++){
142         ans = "";
143         FIRST(ke[i]);
144         cerr<<"FIRST("<<ke[i]<<")"<<" = {"<<ans<<"}\n";
145         cout<<"FIRST("<<ke[i]<<")"<<" = {"<<ans<<"}\n";
146     }
147
148     for(int i=0;i<ke.size();i++){
149
150         string val = mp[ke[i]];
151         string v = "";
152
153         for(int j=0;j<val.size();j++){
154             if(val[j]=='|') break;
155             v += val[j];
156         }
157
158         mp[ke[i]] = v;

```

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 x

Projects Files FSymbols

Workspace

Start here x main.cpp x

```
138 cerr<<"FIRST: \n\n";
139 cout<<"FIRST: \n\n";
140
141 for(int i=0;i<ke.size();i++){
142     ans = "";
143     FIRST(ke[i]);
144     cerr<<"FIRST("<<ke[i]<<") "<<" = {"<<ans<<"}\n";
145     cout<<"FIRST("<<ke[i]<<") "<<" = {"<<ans<<"}\n";
146 }
147
148 for(int i=0;i<ke.size();i++){
149
150     string val = mp[ke[i]];
151     string v = "";
152
153     for(int j=0;j<val.size();j++){
154         if(val[j]=='|') break;
155         v += val[j];
156     }
157
158     mp[ke[i]] = v;
159     mpp[v] = ke[i];
160     ri.push_back(v);
161 }
162
163 cerr<<"\nFOLLOW: \n\n";
164 cout<<"\nFOLLOW: \n\n";
165
166
167 for(int i=0;i<ke.size();i++){
168     ans = "";
169
170     FOLLOW(ke[i],0);
171     cerr<<"FOLLOW("<<ke[i]<<") "<<" = {"<<ans<<"}\n";
172     cout<<"FOLLOW("<<ke[i]<<") "<<" = {"<<ans<<"}\n";
173 }
174
175
176 }
177
```

Fig. 24. Proposed Methodology