

# BÁO CÁO THỰC HÀNH XÂY DỰNG CHƯƠNG TRÌNH DỊCH – IT3323

## BÀI 1: PHÂN TÍCH TỪ VỰNG

Họ và tên: Trịnh Hữu An

MSSV: 20225593

Mã lớp: 161629

### Mục lục:

1. Nội dung đã sửa .....	2
2. Kết quả thực hiện với example1, example2, example3 .....	7
3. Kết quả thực hiện hiển thị lỗi .....	13

### Hình ảnh:

Figure 1: File "scanner.c" đã chỉnh sửa .....	7
Figure 2: Kết quả thực hiện với example1.kpl .....	8
Figure 3: Kết quả thực hiện với example2.kpl .....	9
Figure 4: Kết quả thực hiện với example3.kpl .....	13
Figure 5: Thêm file "error.kpl" .....	13
Figure 6: Lỗi "Value of integer number exceeds the range!" .....	14
Figure 7: Lỗi "Invalid const char!" và lỗi "Invalid symbol!" .....	15
Figure 8: Lỗi "Identification too long!" và lỗi "End of comment expected!" .....	16

## 1. Nội dung đã sửa

*File scanner.c đã sửa:*

```
26  Token* getToken(void)
27  {
28      Token *token;
29      switch(state)
30      {
31          case 0:
32              if (currentChar == EOF) state =1;
33              else
34                  switch (charCodes[currentChar])
35                  {
36                      case CHAR_SPACE:
37                          state =2;break;
38                      case CHAR_LETTER:
39                          ln=lineNo;
40                          cn=colNo;
41                          state =3;
42                          break;
43                      case CHAR_DIGIT:
44                          state =7;
45                          break;
46                      case CHAR_PLUS:
47                          state =9;
48                          break;
49                      case CHAR_MINUS:
50                          state =10;
51                          break;
52                      case CHAR_TIMES:
```

```
53     state =11;
54     break;
55     case CHAR_SLASH:
56         state =12;
57         break;
58     case CHAR_LT:
59         state =13;
60         break;
61     case CHAR_GT:
62         state= 16;
63         break;
64     case CHAR_EQ:
65         state =19;
66         break;
67     case CHAR_EXCLAMATION:
68         state = 20;
69         break;
70     case CHAR_COMMA:
71         state =23;
72         break;
73     case CHAR_PERIOD:
74         state =24;
75         break;
```

```
76     case CHAR_SEMICOLON:
77         state=27;
78         break;
79     case CHAR_COLON:
80         state =28;
81         break;
82     case CHAR_SINGLEQUOTE:
83         state =31;
84         break;
85     case CHAR_LPAR:
86         state = 35;
87         break;
88     case CHAR_RPAR:
89         state= 42;
90         break;
91     default:
92         state=43;
93     }
94     return getToken();
95     case 1:
96         return makeToken(TK_EOF, lineNo, colNo);
97     case 2:
98         readChar();
99         state = 0;
100        return getToken();
```

```

101 case 3:
102     ln = lineNo;
103     cn = colNo;
104     int count = 1;
105     str[0] = (char)currentChar;
106     readChar();
107     while ((currentChar != EOF) && ((charCodes[currentChar] == CHAR_LETTER) || (charCodes[currentChar] == CHAR_DIGIT))) {
108         if (count < MAX_IDENT_LEN) str[count++] = (char)currentChar;
109         else {
110             while ((currentChar != EOF) && ((charCodes[currentChar] == CHAR_LETTER) || (charCodes[currentChar] == CHAR_DIGIT))) {
111                 readChar();
112             }
113             error(ERR_IDENTTOOLONG, ln, cn);
114             token = makeToken(TK_NONE, ln, cn);
115             return token;
116         }
117         readChar();
118     }
119     str[count] = '\0';
120     state = 4;
121     return getToken();
122 case 4:
123     int checkKeyW = checkKeyword(str);
124     if (checkKeyW == TK_NONE) state = 5; else state = 6;
125     return getToken();
126 case 5:
127     token = makeToken(TK_IDENT, ln, cn);
128     strcpy(token->string, str);
129     return token;
130 case 6:
131     token = makeToken(checkKeyword(str), ln, cn);
132     return token;

```

```

133 case 7:
134     ln = lineNo;
135     cn = colNo;
136     int idx = 0;
137     char string[11];
138     while (currentChar == '0') readChar();
139     while (currentChar != EOF && charCodes[currentChar] == CHAR_DIGIT) {
140         if (idx >= 10) {
141             while (currentChar != EOF && charCodes[currentChar] == CHAR_DIGIT) readChar();
142             error(ERR_NUMBERTOOLONG, ln, cn);
143             return makeToken(TK_NONE, ln, cn);
144         }
145         string[idx++] = currentChar;
146         readChar();
147     }
148     string[idx] = '\0';
149     token = makeToken(TK_NUMBER, ln, cn);
150     if (idx == 0) strcpy(token->string, "0");
151     else strcpy(token->string, string);
152     return token;
153 case 9:
154     readChar();
155     return makeToken(SB_PLUS, lineNo, colNo-1);
156 case 10:
157     token = makeToken(SB_MINUS, lineNo, colNo);
158     readChar();

```

```

159     return token;
160 case 11:
161     token = makeToken(SB_TIMES, lineNo, colNo);
162     readChar();
163     return token;
164 case 12:
165     token = makeToken(SB_SLASH, lineNo, colNo);
166     readChar();
167     return token;
168 case 13:
169     readChar();
170     if (charCodes[currentChar] == CHAR_EQ) state = 14; else state = 15;
171 return getToken();
172 case 14:
173     readChar();
174     return makeToken(SB_LE, lineNo, colNo-1);
175 case 15:
176     return makeToken(SB_LT, lineNo, colNo-1);
177 case 16:
178     readChar();
179     if ((currentChar != EOF) && (charCodes[currentChar] == CHAR_EQ)) state = 17; else state = 18;
180     return getToken();
181 case 17:
182     token = makeToken(SB_GE, lineNo, colNo);
183     readChar();
184     return token;

```

```

185 case 18:
186     token = makeToken(SB_GT, lineNo, colNo);
187     readChar();
188     return token;
189 case 19:
190     token = makeToken(SB_EQ, lineNo, colNo);
191     readChar();
192     return token;
193 case 20:
194     ln = lineNo;
195     cn = colNo;
196     readChar();
197     if ((currentChar != EOF) && (charCodes[currentChar] == CHAR_EQ)) state = 21; else state = 22;
198     return getToken();
199 case 21:
200     readChar();
201     return makeToken(SB_NEQ, ln, cn);
202 case 22:
203     token = makeToken(TK_NONE, lineNo, colNo-1);
204     error(ERR_INVALIDSYMBOL, token->lineNo, token->colNo);
205     return token;
206 case 23:
207     token = makeToken(SB_COMMA, lineNo, colNo);
208     readChar();
209     return token;
210 case 24:

```

```

211     readChar();
212     if ((currentChar != EOF) && (charCodes[currentChar] == CHAR_RPAR)) state = 25; else state = 26;
213     return getToken();
214 case 25:
215     token = makeToken(SB_RSEL, lineNo, colNo - 1);
216     readChar();
217     return token;
218 case 26:
219     token = makeToken(SB_PERIOD, lineNo, colNo - 1);
220     readChar();
221     return token;
222 case 27:
223     token = makeToken(SB_SEMICOLON, lineNo, colNo);
224     readChar();
225     return token;
226 case 28:
227     readChar();
228     if ((currentChar != EOF) && (charCodes[currentChar] == CHAR_EQ)) state = 29; else state = 30;
229     return getToken();
230 case 29:
231     token = makeToken(SB_ASSIGN, lineNo, colNo - 1);
232     readChar();
233     return token;
234 case 30:
235     return makeToken(SB_COLON, lineNo, colNo - 1);
236 case 31:
237     ln = lineNo;

```

```

238     cn = colNo;
239     readChar();
240     if (currentChar == EOF)
241         state=34;
242     else
243         if(isprint(currentChar))
244             state =32;
245         else state =34;
246     return getToken();
247 case 32:
248     c= currentChar;
249     readChar();
250     if (charCodes[currentChar] == CHAR_SINGLEQUOTE)
251         state=33;
252     else
253         state =34;
254     return getToken();
255 case 33:
256     token = makeToken(TK_CHAR, ln, cn);
257     token->string[0] =c;
258     token->string[1] ='\0';
259     readChar();
260     return token;
261 case 34:
262     error(ERR_INVALIDCHARCONSTANT, lineNo, colNo-2);
263     return makeToken(TK_NONE, lineNo, colNo-2);

```

```

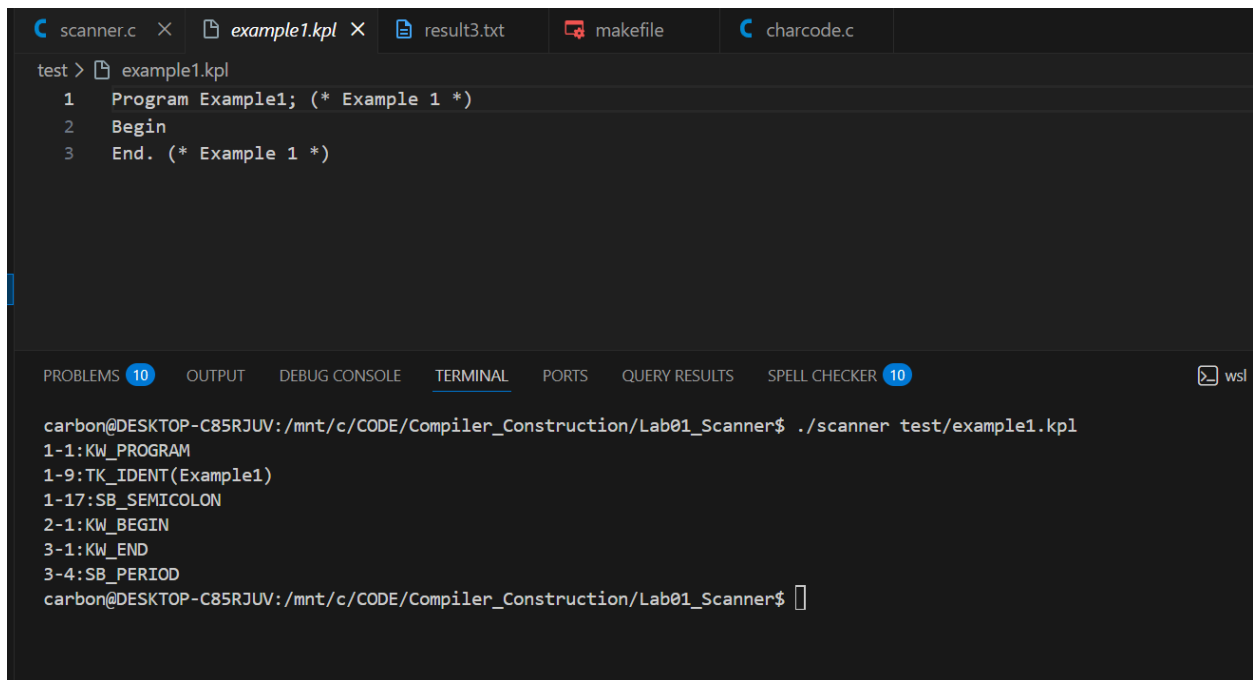
264     case 35: // tokens begin with lpar, skip comments
265         ln = lineNo;
266         cn = colNo;
267         readChar();
268         if (currentChar == EOF)
269             state=41;
270         else
271             switch (charCodes[currentChar])
272             {
273                 case CHAR_PERIOD:
274                     state =36;
275                     break;
276                 case CHAR_TIMES:
277                     state =38;
278                     break;
279                 default:state =41;
280             }
281         return getToken();
282
283     case 36:
284         token = makeToken(SB_LSEL, lineNo, colNo - 1);
285         readChar();
286         return token;
287     case 37:
288         while ((currentChar != EOF) && (charCodes[currentChar] != CHAR_TIMES)) readChar();
289         if (currentChar == EOF) state = 40; else state = 38;
290
291     case 38:
292         while ((currentChar != EOF) && (charCodes[currentChar] == CHAR_TIMES)) readChar();
293         if (currentChar == EOF) state = 40; else if (charCodes[currentChar] == CHAR_RPAR) state = 39; else state = 37;
294         return getToken();
295     case 39:
296         state = 0;
297         readChar();
298         return getToken();
299     case 40:
300         error(ERR_ENDOFCOMMENT, lineNo, colNo);
301     case 41: return makeToken(SB_LPAR, ln, cn);
302     case 42:
303         token = makeToken(SB_RPAR, lineNo, colNo);
304         readChar();
305         return token;
306     case 43:
307         token = makeToken(TK_NONE, lineNo, colNo);
308         error(ERR_INVALIDSYMBOL, lineNo, colNo);
309         readChar();
310         return token;
311 }
312 }

```

Figure 1: File "scanner.c" đã chỉnh sửa

## 2. Kết quả thực hiện với example1, example2, example3

- Kết quả thực hiện với example1.kpl:



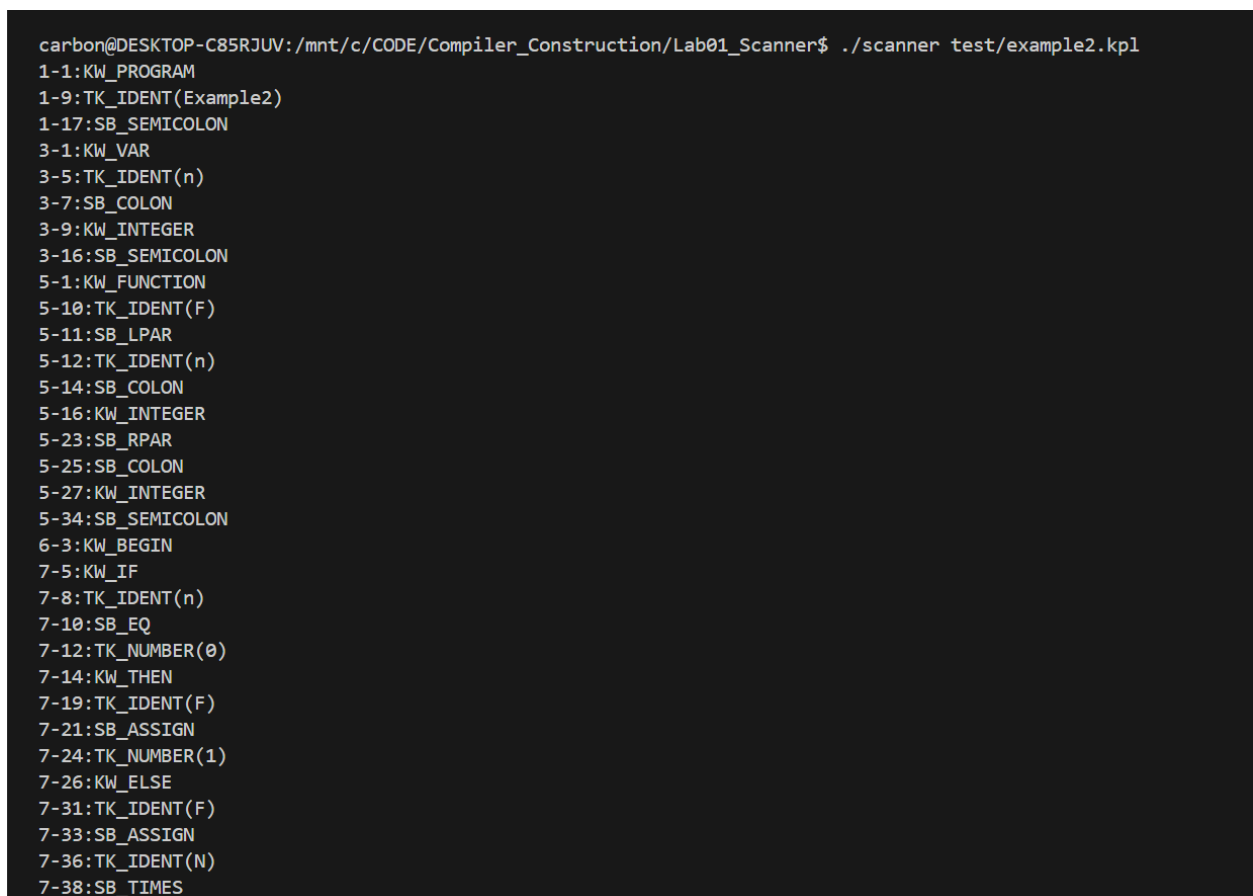
```
scanner.c x example1.kpl x result3.txt makefile charcode.c
test > example1.kpl
1 Program Example1; (* Example 1 *)
2 Begin
3 End. (* Example 1 *)

PROBLEMS 10 OUTPUT DEBUG CONSOLE TERMINAL PORTS QUERY RESULTS SPELL CHECKER 10 wsl

carbon@DESKTOP-C85RJUV:/mnt/c/CODE/Compiler_Construction/Lab01_Scanner$ ./scanner test/example1.kpl
1-1:KW_PROGRAM
1-9:TK_IDENT(Example1)
1-17:SB_SEMICOLON
2-1:KW_BEGIN
3-1:KW_END
3-4:SB_PERIOD
carbon@DESKTOP-C85RJUV:/mnt/c/CODE/Compiler_Construction/Lab01_Scanner$
```

Figure 2: Kết quả thực hiện với example1.kpl

- Kết quả thực hiện với example2.kpl:



```
carbon@DESKTOP-C85RJUV:/mnt/c/CODE/Compiler_Construction/Lab01_Scanner$ ./scanner test/example2.kpl
1-1:KW_PROGRAM
1-9:TK_IDENT(Example2)
1-17:SB_SEMICOLON
3-1:KW_VAR
3-5:TK_IDENT(n)
3-7:SB_COLON
3-9:KW_INTEGER
3-16:SB_SEMICOLON
5-1:KW_FUNCTION
5-10:TK_IDENT(F)
5-11:SB_LPAR
5-12:TK_IDENT(n)
5-14:SB_COLON
5-16:KW_INTEGER
5-23:SB_RPAR
5-25:SB_COLON
5-27:KW_INTEGER
5-34:SB_SEMICOLON
6-3:KW_BEGIN
7-5:KW_IF
7-8:TK_IDENT(n)
7-10:SB_EQ
7-12:TK_NUMBER(0)
7-14:KW_THEN
7-19:TK_IDENT(F)
7-21:SB_ASSIGN
7-24:TK_NUMBER(1)
7-26:KW_ELSE
7-31:TK_IDENT(F)
7-33:SB_ASSIGN
7-36:TK_IDENT(N)
7-38:SB_TIMES
```



```
7-40:TK_IDENT(F)
7-42:SB_LPAR
7-43:TK_IDENT(N)
7-45:SB_MINUS
7-47:TK_NUMBER(1)
7-48:SB_RPAR
7-49:SB_SEMICOLON
8-3:KW_END
8-6:SB_SEMICOLON
10-1:KW_BEGIN
11-3:KW_FOR
11-7:TK_IDENT(n)
11-9:SB_ASSIGN
11-12:TK_NUMBER(1)
11-14:KW_TO
11-17:TK_NUMBER(7)
11-19:KW_DO
12-5:KW_BEGIN
13-7:KW_CALL
13-12:TK_IDENT(WriteLn)
13-19:SB_SEMICOLON
14-7:KW_CALL
14-12:TK_IDENT(WriteI)
14-18:SB_LPAR
14-20:TK_IDENT(F)
14-21:SB_LPAR
14-22:TK_IDENT(i)
14-23:SB_RPAR
14-24:SB_RPAR
14-25:SB_SEMICOLON
15-5:KW_END
15-8:SB_SEMICOLON
```

```
16-1:KW_END
```

```
16-4:SB_PERIOD
```

```
carbon@DESKTOP-C85RJUV:/mnt/c/CODE/Compiler_Construction/Lab01_Scanner$
```

Figure 3: Kết quả thực hiện với example2.kpl

- Kết quả thực hiện với example3.kpl:

```
carbon@DESKTOP-C85RJUV:/mnt/c/CODE/Compiler_Construction/Lab01_Scanner$ ./scanner test/example3.kpl
1-1:KW_PROGRAM
1-10:TK_IDENT(EXAMPLE3)
1-18:SB_SEMICOLON
2-1:KW_VAR
2-6:TK_IDENT(I)
2-7:SB_COLON
2-8:KW_INTEGER
2-15:SB_SEMICOLON
3-6:TK_IDENT(N)
3-7:SB_COLON
3-8:KW_INTEGER
3-15:SB_SEMICOLON
4-6:TK_IDENT(P)
4-7:SB_COLON
4-8:KW_INTEGER
4-15:SB_SEMICOLON
5-6:TK_IDENT(Q)
5-7:SB_COLON
5-8:KW_INTEGER
5-15:SB_SEMICOLON
6-6:TK_IDENT(C)
6-7:SB_COLON
6-8:KW_CHAR
6-12:SB_SEMICOLON
8-1:KW_PROCEDURE
8-12:TK_IDENT(HANOI)
8-17:SB_LPAR
8-18:TK_IDENT(N)
8-19:SB_COLON
8-20:KW_INTEGER
8-27:SB_SEMICOLON
8-30:TK_IDENT(S)
```

```
8-31:SB_COLON
8-32:KW_INTEGER
8-39:SB_SEMICOLON
8-42:TK_IDENT(Z)
8-43:SB_COLON
8-44:KW_INTEGER
8-51:SB_RPAR
8-52:SB_SEMICOLON
9-1:KW_BEGIN
10-3:KW_IF
10-7:TK_IDENT(N)
10-9:SB_NEQ
10-12:TK_NUMBER(0)
10-15:KW_THEN
11-5:KW_BEGIN
12-7:KW_CALL
12-13:TK_IDENT(HANOI)
12-18:SB_LPAR
12-19:TK_IDENT(N)
12-20:SB_MINUS
12-21:TK_NUMBER(1)
12-22:SB_COMMA
12-23:TK_IDENT(S)
12-24:SB_COMMA
12-25:TK_NUMBER(6)
12-26:SB_MINUS
12-27:TK_IDENT(S)
12-28:SB_MINUS
12-29:TK_IDENT(Z)
12-30:SB_RPAR
12-31:SB_SEMICOLON
13-7:TK_IDENT(I)
```

```
13-8:SB_ASSIGN
13-10:TK_IDENT(I)
13-11:SB_PLUS
13-12:TK_NUMBER(1)
13-13:SB_SEMICOLON
14-7:KW_CALL
14-13:TK_IDENT(WRITELN)
14-20:SB_SEMICOLON
15-7:KW_CALL
15-13:TK_IDENT(WRITEI)
15-19:SB_LPAR
15-20:TK_IDENT(I)
15-21:SB_RPAR
15-22:SB_SEMICOLON
16-7:KW_CALL
16-13:TK_IDENT(WRITEI)
16-19:SB_LPAR
16-20:TK_IDENT(N)
16-21:SB_RPAR
16-22:SB_SEMICOLON
17-7:KW_CALL
17-13:TK_IDENT(WRITEI)
17-19:SB_LPAR
17-20:TK_IDENT(S)
17-21:SB_RPAR
17-22:SB_SEMICOLON
18-7:KW_CALL
18-13:TK_IDENT(WRITEI)
18-19:SB_LPAR
18-20:TK_IDENT(Z)
18-21:SB_RPAR
```

```
18-22:SB_SEMICOLON
19-7:KW_CALL
19-13:TK_IDENT(HANOI)
19-18:SB_LPAR
19-19:TK_IDENT(N)
19-20:SB_MINUS
19-21:TK_NUMBER(1)
19-22:SB_COMMA
19-23:TK_NUMBER(6)
19-24:SB_MINUS
19-25:TK_IDENT(S)
19-26:SB_MINUS
19-27:TK_IDENT(Z)
19-28:SB_COMMA
19-29:TK_IDENT(Z)
19-30:SB_RPAR
20-5:KW_END
21-1:KW_END
21-4:SB_SEMICOLON
23-1:KW_BEGIN
24-3:KW_FOR
24-8:TK_IDENT(N)
24-10:SB_ASSIGN
24-13:TK_NUMBER(1)
24-16:KW_TO
24-20:TK_NUMBER(4)
24-23:KW_DO
25-5:KW_BEGIN
26-7:KW_FOR
26-12:TK_IDENT(I)
26-13:SB_ASSIGN
26-15:TK_NUMBER(1)
```

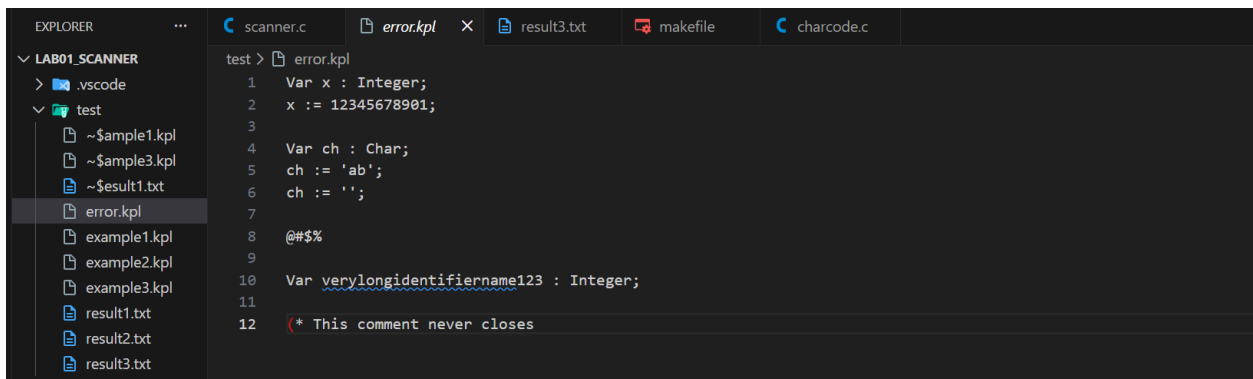
```
26-18:KW_TO
26-22:TK_NUMBER(4)
26-25:KW_DO
27-9:KW_CALL
27-15:TK_IDENT(WRITEC)
27-21:SB_LPAR
27-22:TK_CHAR(' ')
27-25:SB_RPAR
27-26:SB_SEMICOLON
28-7:KW_CALL
28-13:TK_IDENT(READC)
28-18:SB_LPAR
28-19:TK_IDENT(C)
28-20:SB_RPAR
28-21:SB_SEMICOLON
29-7:KW_CALL
29-13:TK_IDENT(WRITEC)
29-19:SB_LPAR
29-20:TK_IDENT(C)
29-21:SB_RPAR
30-5:KW_END
30-8:SB_SEMICOLON
31-3:TK_IDENT(P)
31-4:SB_ASSIGN
31-6:TK_NUMBER(1)
31-7:SB_SEMICOLON
32-3:TK_IDENT(Q)
32-4:SB_ASSIGN
32-6:TK_NUMBER(2)
32-7:SB_SEMICOLON
33-3:KW_FOR
33-8:TK_IDENT(N)
```

```
33-9:SB_ASSIGN
33-11:TK_NUMBER(2)
33-14:KW_TO
33-18:TK_NUMBER(4)
33-21:KW_DO
34-5:KW_BEGIN
35-7:TK_IDENT(I)
35-8:SB_ASSIGN
35-10:TK_NUMBER(0)
35-11:SB_SEMICOLON
36-7:KW_CALL
36-13:TK_IDENT(HANOI)
36-18:SB_LPAR
36-19:TK_IDENT(N)
36-20:SB_COMMA
36-21:TK_IDENT(P)
36-22:SB_COMMA
36-23:TK_IDENT(Q)
36-24:SB_RPAR
36-25:SB_SEMICOLON
37-7:KW_CALL
37-13:TK_IDENT(WRITELN)
38-5:KW_END
39-1:KW_END
39-4:SB_PERIOD
carbon@DESKTOP-C85RJUV:/mnt/c/CODE/Compiler_Construction/Lab01_Scanner$
```

Figure 4: Kết quả thực hiện với example3.kpl

### 3. Kết quả thực hiện hiển thị lỗi

- Thêm vào folder /test file “error.kpl”:



```
test > error.kpl
1  Var x : Integer;
2  x := 12345678901;
3
4  Var ch : Char;
5  ch := 'ab';
6  ch := '';
7
8  @#$$%
9
10 Var verylongidentifiernam123 : Integer;
11
12 (* This comment never closes
```

Figure 5: Thêm file “error.kpl”

- Sau khi thực hiện chạy kết quả với file “error.kpl”, thu được kết quả như màn hình sau với 5 lỗi được ghi nhận:

```
- Value of integer number exceeds the range!
- Invalid const char!
- Invalid symbol!
- Identification too long!
- End of comment expected!
```

The image shows a code editor with several tabs: `scanner.c`, `error.kpl`, `result3.txt`, `makefile`, and `charcode.c`. The `error.kpl` file contains the following code:

```
1  Var x : Integer;
2  x := 12345678901;
3
4  Var ch : Char;
5  ch := 'ab';
6  ch := '';
7
8  @#$$%
9
10 Var verylongidentifiernam123 : Integer;
11
12 (* This comment never closes
```

The terminal window at the bottom shows the command `./scanner test/error.kpl` being executed. The output lists the tokens and symbols found in the file, along with an error message:

```
carbon@DESKTOP-C85RJUV:/mnt/c/CODE/Compiler_Construction/Lab01_Scanner$ ./scanner test/error.kpl
1-1:KW_VAR
1-5:TK_IDENT(x)
1-7:SB_COLON
1-9:KW_INTEGER
1-16:SB_SEMICOLON
2-1:TK_IDENT(x)
2-3:SB_ASSIGN
2-6:Value of integer number exceeds the range!
2-6:TK_NONE
2-17:SB_SEMICOLON
4-1:KW_VAR
4-5:TK_IDENT(ch)
4-8:SB_COLON
4-10:KW_CHAR
4-14:SB_SEMICOLON
```

Figure 6: Lỗi "Value of integer number exceeds the range!"

The image shows a code editor with a file named `error.kpl` open. The code contains several syntax errors. Below the code, the terminal window displays the error messages generated by the compiler.

```
test > error.kpl
1  Var x : Integer;
2  x := 12345678901;
3
4  Var ch : Char;
5  ch := 'ab';
6  ch := '';
7
8  @#$$%
9
10 Var verylongidentifiernam123 : Integer;
11
12 (* This comment never closes
```

PROBLEMS 11 OUTPUT DEBUG CONSOLE TERMINAL PORTS QUERY RESULTS SPELL CHECKER 11 wsl

```
4-10:KW_CHAR
4-14:SB_SEMICOLON
5-1:TK_IDENT(ch)
5-4:SB_ASSIGN
5-7:Invalid const char!
5-7:TK_NONE
5-9:TK_IDENT(b)
5-10:Invalid const char!
5-10:TK_NONE
6-1:TK_IDENT(ch)
6-4:SB_ASSIGN
6-7:Invalid const char!
6-7:TK_NONE
6-9:SB_SEMICOLON
8-1:Invalid symbol!
8-1:TK_NONE
```

Figure 7: Lỗi "Invalid const char!" và lỗi "Invalid symbol!"

```
test > error.kpl
1  Var x : Integer;
2  x := 12345678901;
3
4  Var ch : Char;
5  ch := 'ab';
6  ch := '';
7
8  @#$$%
9
10 Var verylongidentifiernam123 : Integer;
11
12 (* This comment never closes
```

PROBLEMS 11 OUTPUT DEBUG CONSOLE TERMINAL PORTS QUERY RESULTS SPELL CHECKER 11 wsl

8-1:TK\_NONE  
8-2:Invalid symbol!  
8-2:TK\_NONE  
8-3:Invalid symbol!  
8-3:TK\_NONE  
8-4:Invalid symbol!  
8-4:TK\_NONE  
10-1:KW\_VAR  
10-5:Identification too long!  
10-5:TK\_NONE  
10-31:SB\_COLON  
10-33:KW\_INTEGER  
10-40:SB\_SEMICOLON  
12-29:End of comment expected!  
12-1:SB\_LPAR  
carbon@DESKTOP-C85RJUV:/mnt/c/CODE/Compiler\_Construction/Lab01\_Scanner\$

Figure 8: Lỗi "Identification too long!" và lỗi "End of comment expected!"