

Содержимое файла для синтаксического анализа приведено в таблице 3.

Таблица 3. Содержимое файла для синтаксического анализа

file.c
<pre>// Single line comment /* Multi line comment block */ #include &lt;stdio.h&gt; #include &lt;stdlib.h&gt; #include &lt;wchar.h&gt; static enum Enum { VAR_1 = 0, VAR_2, VAR_3 = 0, VAR_4, VAR_5 = 0, VAR_6 } some_enum1, some_enum2; const volatile int global_var = 1 + 1 &lt;&lt; sizeof(char), another_var = 2 * global_var; static struct Animal {     int animal_int : 32;     const void * volatile animal_reference; } some_animal1, some_animal2; int main(int _, ...) {     int * const r = (int*) malloc(sizeof(int) * 5), k = (2 &gt;&gt; 1) + (1 &lt;&lt; 1) - 1 * 1.0e-5/1e+20;     wprintf(L"%d", ((0 &lt;= 0 &lt; 2 &gt; 0 &gt;= 0) == 1 != 0));     (((k += 1) -= 1) *= 1) /= 1 % 2) %= 2;     ++k; k++; --k; k--;     ((k &amp;= 1    1   1 &amp; 5 &amp;&amp; 5)  = ~k ^ k) ^= !k;     do {         for (k = 0; k &lt; 5; ++k) {             wprintf(L"\nHello, world\n");         }         while (++k &lt; 10) continue;     }     while (0);     fputc(L'\n', stdout);     free(r);     return 0; }</pre>

Вывод программы состоит из 7087 строк. Кратко приведен в таблице 4.

Таблица 4. Вывод программы (кратко)

Вывод программы (кратко)
<pre>===== ===== Lexical analysis ===== &lt;single-line-comment&gt;  '// Single line comment'  &lt;whitespace&gt;  '\n'  &lt;multi-line-comment&gt;  '/*\nMulti line comment block\n*/'  &lt;whitespace&gt;  '\n'  &lt;preprocessor-directive&gt;  '#include &lt;stdio.h&gt;'  &lt;whitespace&gt;  '\n'  &lt;preprocessor-directive&gt;  '#include &lt;stdlib.h&gt;'  &lt;whitespace&gt;  '\n'  &lt;preprocessor-directive&gt;  '#include &lt;wchar.h&gt;'  &lt;whitespace&gt;  '\n'  static  'static'  &lt;whitespace&gt;  ' '  enum  'enum'  &lt;whitespace&gt;  ' '  &lt;identifier&gt;  'Enum'  ..... return  'return'  &lt;whitespace&gt;  ' '  &lt;integer-constant&gt;  '0'  &lt;semicolon&gt;  ';'  &lt;whitespace&gt;  '\n'  &lt;close-curly-bracket&gt;  '}'  ===== Success ===== =====</pre>

```

=====
===== Syntax analysis =====
0. Ambiguous syntax (60 ambiguous rules / 6663 rules)
1. state=0, token='<single-line-comment>', skip
2. state=0, token='<whitespace>', skip
3. state=0, token='<multi-line-comment>', skip
4. state=0, token='<whitespace>', skip
5. state=0, token='<preprocessor-directive>', skip
6. state=0, token='<whitespace>', skip
7. state=0, token='<preprocessor-directive>', skip
8. state=0, token='<whitespace>', skip
9. state=0, token='<preprocessor-directive>', skip
10. state=0, token='<whitespace>', skip
11. state=0, token='static'
    shift 'static' 13
    0 'static' 13
12. state=13, token='<whitespace>', skip
13. state=13, token='enum'
    reduce 'static' 13 -> '<storage-class-specifier>'
    push 21 by (0, '<storage-class-specifier>')
    0 '<storage-class-specifier>' 21
14. state=21, token='enum'
    reduce '<storage-class-specifier>' 21 -> '<declaration-specifier>'
    push 22 by (0, '<declaration-specifier>')
    0 '<declaration-specifier>' 22
15. state=22, token='enum'
    reduce '<declaration-specifier>' 22 -> '<declaration-specifier>+'
    push 4 by (0, '<declaration-specifier>+')
    0 '<declaration-specifier>+' 4
16. state=4, token='enum'
    shift 'enum' 17
    0 '<declaration-specifier>+' 4 'enum' 17
17. state=17, token='<whitespace>', skip
18. state=17, token='<identifier>'
    shift '<identifier>' 50
    0 '<declaration-specifier>+' 4 'enum' 17 '<identifier>' 50
19. state=50, token='<whitespace>', skip
20. state=50, token='{'
    reduce '<identifier>' 50 -> '<identifier>?'
    push 49 by (17, '<identifier>?')
    0 '<declaration-specifier>+' 4 'enum' 17 '<identifier>?' 49
21. state=49, token='{'
    shift '{' 60
    0 '<declaration-specifier>+' 4 'enum' 17 '<identifier>?' 49 '{' 60
22. state=60, token='<whitespace>', skip
23. state=60, token='<identifier>'
    shift '<identifier>' 112
    0 '<declaration-specifier>+' 4 'enum' 17 '<identifier>?' 49 '{' 60 '<identifier>' 112
24. state=112, token='<whitespace>', skip
25. state=112, token='='
    shift '=' 184
    0 '<declaration-specifier>+' 4 'enum' 17 '<identifier>?' 49 '{' 60 '<identifier>' 112 '=' 184
.....
1628. state=35, token='$'
    reduce '<function-definition>' 35 -> '<external-declaration>'
    push 44 by (7, '<external-declaration>')
    0 '<external-declaration>+' 7 '<external-declaration>' 44
1629. state=44, token='$'
    reduce '<external-declaration>+' 7 '<external-declaration>' 44 -> '<external-declaration>+'
    push 7 by (0, '<external-declaration>+')
    0 '<external-declaration>+' 7
1630. state=7, token='$'
    reduce '<external-declaration>+' 7 -> '<translation-unit>'
    push 14 by (0, '<translation-unit>')
    0 '<translation-unit>' 14
===== Success =====
=====

```

```

10 static struct Animal {
11     int animal_int : 32;
12     const void * volatile animal_reference;
13 } some_animal1, some_animal2;
14 int main(int , ...) {
15     int * const r = (int*) malloc(sizeof(int) * 5);
16     wprintf(L"%d", ((0 <= 0 < 2 > 0 >= 0) == 1 !=
17         (((k <= 1) -> 1) * 1) /> 1 % 2) %> 2;
18     ++k; k++; -> k; k--;
19     ((k &= 1 || 1 | 1 & 5 && 5) |= -k ^ k) ^= !k;
20     do {
21         for (k = 0; k < 5; ++k) {
22             wprintf(L"%nHello, world\n")
23         }
24         while (++k < 10) continue;
25     }
26     while (0);
27     fputc(L'\n', stdout);
28     free(r);
29     return 0;
30 }

```

Рисунок 1. Отсутствие точки с запятой (позиция 23:8)

```

10 static struct Animal {
11     int animal_int : 32;
12     const void * volatile animal_reference;
13 } some_animal1, some_animal2;
14 int main(int, ...) {
15     int * const r = (int*) malloc(sizeof(int) * 5);
16     wprintf(L"%d",, (0 <= 0 < 2 > 0 == 0) == 1 !=
17         (((k += 1) -= 1) == 1) / 1 % 2) % 2;
18     ++k; k++; --k; k--;
19     ((k &= 1 || 1 | 1 & 5 & 5) |= -k ^ k) ^= !k;
20     do {
21         for (k = 0; k < 5; ++k) {
22             wprintf(L"\nHello, world\n");
23         }
24         while (++k < 10) // continue;
25     }
26     while (0);
27     fputc(L'\n', stdout);
28     free(r);
29     return 0;
30 }

```

Рисунок 2. Отсутствие тела цикла while (позиция 25:4)

```

9  const volatile int global_var = 1 + 1 << sizeof(char);
10 static struct Animal {
11     int animal_int : 32;
12     const void * volatile animal_reference;
13 } some_animal1, some_animal2;
14 int main(int __, ...) {
15     int * const r = (int*) malloc(sizeof(int) * 5);
16     wprintf(L"%d", ((0 <= 0 < 2 > 0) == 1) !=
17         (((((k += 1) - 1) * 1) / 1 % 2) % 5) /*2*/);
18     ++k; k++; --k; k--;
19     ((k &= 1 || 1 | 1 & 5 && 5) |= -k ^ k) ^= !k;
20     do {
21         for (k = 0; k < 5; ++k) {
22             wprintf(L"\nHello, world\n");
23         }
24         while (++k < 10) continue;
25     }
26     while (0);
27     fputc(L'\n', stdout);
28     free(r);
29     return 0;

```

Рисунок 3. Отсутствие операнда справа от знака %= (позиция 17:46)

18

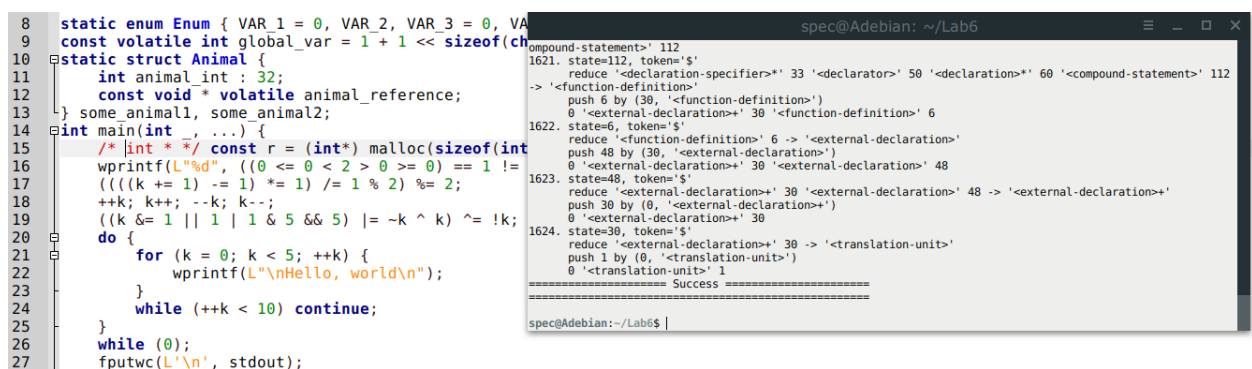
В данном случае генерируется GLALR(1)-парсер. Его преимущество по сравнению с GLR(1)-парсером в том, что число состояний, затрат по времени и памяти используется меньше, однако распознавательная способность ниже. С помощью GLR(1)-парсера корректно предсказать ожидаемые токены проще и его распознавательная способность выше, однако в этом случае число состояний, затрат по времени и памяти будет использоваться больше.

Выбрать генерацию GLR(1)-парсера в коде: аргумент `lalr` функции `syntax_analyzer` → `lalr=False`.

Выбрать генерацию GLALR(1)-парсера в коде: аргумент `lalr` функции `syntax_analyzer` → `lalr=True`.

Результат синтаксического анализа полностью зависит от описания синтаксиса (в коде за синтаксис языка C90 отвечает метод `get_syntax_ebnf()`).

Приведенное описание синтаксиса языка C в приложении A13 в книге «Kernighan, Brian; Ritchie, Dennis M. (March 1988). The C Programming Language (2nd ed.). Englewood Cliffs, NJ: Prentice Hall. ISBN 0-13-110362-8.» некорректно (рис. 4).



```
8 static enum Enum { VAR_1 = 0, VAR_2, VAR_3 = 0, VA
9 const volatile int global_var = 1 + 1 << sizeof(ch
10 static struct Animal {
11     int animal_int : 32;
12     const void * volatile animal_reference;
13 } some_animal1, some_animal2;
14 int main(int _, ...) {
15     /* int * */ const r = (int*) malloc(sizeof(int)
16     wprintf(L"%d", ((0 <= 0 < 2 > 0 >= 0) == 1 !=
17     (((k += 1) -= 1) * 1) /= 1 % 2) % 2;
18     ++k; k++; --k; k--;
19     ((k &= 1 || 1 | 1 & 5 && 5) |= -k ^ k) ^= !k;
20     do {
21         for (k = 0; k < 5; ++k) {
22             wprintf(L"%nHello, world\n");
23         }
24         while (++k < 10) continue;
25     }
26     while (0);
27     fputc(L'\n', stdout);
```

```
spec@Adebian: ~/Lab6
compound-statement>' 112
1621. state=112, token='$'
    reduce '<declaration-specifier>*' 33 '<declarator>' 50 '<declaration>*' 60 '<compound-statement>' 112
    -> '<function-definition>'
    push 6 by (30, '<function-definition>')
    0 '<external-declaration>+' 30 '<function-definition>' 6
1622. state=6, token='$'
    reduce '<function-definition>' 6 -> '<external-declaration>'
    push 48 by (30, '<external-declaration>')
    0 '<external-declaration>+' 30 '<external-declaration>' 48
1623. state=48, token='$'
    reduce '<external-declaration>+' 30 '<external-declaration>' 48 -> '<external-declaration>+'
    push 30 by (0, '<external-declaration>+')
    0 '<external-declaration>+' 30
1624. state=30, token='$'
    reduce '<external-declaration>+' 30 -> '<translation-unit>'
    push 1 by (0, '<translation-unit>')
    0 '<translation-unit>' 1
===== Success =====
spec@Adebian: ~/Lab6$
```

Рисунок 4. Тип не указан, но указан квалификатор `const`, синтаксический анализ успешен, хотя не должен быть

Все дело в некорректном описании нетерминалов `declaration` и `declaration-specifiers` (стр. 234).

## ЗАКЛЮЧЕНИЕ

В результате выполнения лабораторной работы мы создали упрощенный вариант синтаксического анализатора кода на языке программирования C90.