Flex提取C/C++代码中的整数、浮点数

> 17341137 宋震鹏

本项目生成的分析器可以识别C/C++代码文件(.c, .cpp)中的整数以及浮点数，筛选规则简要如下：

1. 不识别函数名中的数字（如foo123不识别123）
2. 不识别变量名中的数字（如var123不识别123）
3. 对字符串内部的数字，一律将`-`号视为负号，而不是减操作符。

   （如123-234=-111，识别为123、-234、-111）
4. 行内注释、跨行注释（行内使用、跨行使用）皆不识别。

Project Sheet

- inputfile.c：测试用的C语言文件
- ints-and-floats.l：.l文件，提供给flex以生成lex.yy.c中间代码
- lex.yy.c：flex生成的中间代码，供gcc编译
- ints-and-floats：解析器程序
- makefile：使用make命令进行调试

Main Function

1. 匹配行内注释：`[/]{2}.*`

   在本行内，出现两个'/'后的部分视为注释，该表达式将匹配之。
2. 匹配跨行注释：`[/][*][^*]*[*]+([^*/][^*]*[*]+)*[/]`

   由于flex不支持贪婪的匹配方案，因此将匹配出现在`'/*'`后的第一个`'*/'`。

   同时，`[^*]`匹配任意个换行符，`[*]+`匹配至少一个`*`星号；

   此后，括号的内容将被匹配任意次：`[^*/]`匹配星号、斜杠以外的任意字符，`[^*]*`匹配任意个星号以外的任意字符，`[*]+`匹配至少一个`*`星号；

   最后，匹配一个斜杠以结束。
3. 匹配浮点数：`-?[1-9][0-9]*\.[0-9]+`

   首先，`-?`保证不遗漏负数，再匹配一个1-9之间的数字保证筛去不合理的前置0的浮点数（如0001.5），再匹配任意个0-9之间的数字、小数点，以及至少一个0-9之间的数字（考虑到有效数字，并不对尾部后置0进行筛去）
4. 匹配函数、变量名中的整数：`[a-zA-Z_]+-?[0-9]+`

   在C/C++规范中，函数、变量名中的整数可能由大小写字母、下划线起头。
5. 匹配整数：`-?[0-9]+`

   首先，`-?`保证不遗漏负数，再匹配任意个0-9之间的数字。
6. 匹配其余部分：
   - `\n`：匹配文件中的换行符。
   - `.`：匹配换行符以外的任意字符。
   - `<<EOF>>`：匹配文件结尾。

Code

```
/**
*  This .l file help generate a scanner with flex.
*  Target file name: inputfile.c
*
* Run in terminal:
*  make
*
* With verbose:
*  = 1: show log of other pattern found.
*  = 0: show log of intergers and float numbers only.
*/

%{
int intergers = 0;
int floats = 0;
int comments = 0;
int var_funcs = 0;
int verbose = 0;
%}

%%
[/]{2}.* {if (verbose) printf(">>> [Comment]: \"%s\" is a comment.\n", yytext); comments++;}
[/][*][^*]*[*]+([^*/][^*]*[*]+)*[/] {if (verbose) printf(">>> [Multiline-Comment]: \"%s\" is a multilined-
comment.\n", yytext); comments++;}

-?[1-9][0-9]*\.[0-9]+ {printf(">>> [Float]: \"%s\" is a float number.\n", yytext); floats++;}
[a-zA-Z_]+-?[0-9]+ {if (verbose) printf(">>> [Var/Func]: \"%s\" is a variable or a function name.\n", yytext);
var_funcs++;}
-?[0-9]+ {printf(">>> [Interger]: \"%s\" is an interger.\n", yytext); intergers++;}

\n {}
. {}
<<EOF>> {printf("File End.\n"); return 0;}
%%

int yywrap() {}

int main() {

  printf("\n### Find Interger(s) and Float number(s) ###\n");
  printf(">>> verbose(1/0):");
  scanf("%d", &verbose);
  getchar();


  freopen("./inputfile.c", "r", stdin);
  yylex();

  printf("\n### Conclusion ###\n");
  printf("Number of Interger(s) in the code - %d\n", intergers);
  printf("Number of Float number(s) in the code - %d\n", floats);
  printf("Number of Comment(s) in the code - %d\n", comments);
  printf("Number of Variable(s)/Function(s) in the code - %d\n", var_funcs);

  return 0;
}
```

Result

> 在根目录下使用make指令即可进行测试。

testcase1

```
#include <stdio.h>
#include <stdlib.h>

void foo123()
{
   int f123;
}

void foo_234()
{
   int f_234;
}

int main()
{

   // Testing bare comments
   /* Testing bare comments */

   /* Inline comment with 123 231.132 */
   /*
      Crossline with
      123,231 321.123
   */
   int vector[105];
   char *s = "String with Numbers:123,-456,12.21,-34.46, 123-234=-111";
   int i1 = 123, i2 = -123;        // comments as appendix with 321
   float j1 = 567.89, j2 = -567.89; // comments as appendix with 321
   return 0;
}
```

verbose == 0

```
### Find Interger(s) and Float number(s) ###
>>> verbose(1/0):0
>>> [Interger]: "105" is an interger.
>>> [Interger]: "123" is an interger.
>>> [Interger]: "-456" is an interger.
>>> [Float]: "12.21" is a float number.
>>> [Float]: "-34.46" is a float number.
>>> [Interger]: "123" is an interger.
>>> [Interger]: "-234" is an interger.
>>> [Interger]: "-111" is an interger.
>>> [Interger]: "123" is an interger.
>>> [Interger]: "-123" is an interger.
>>> [Float]: "567.89" is a float number.
>>> [Float]: "-567.89" is a float number.
File End.

### Conclusion ###
Number of Interger(s) in the code - 8
Number of Float number(s) in the code - 4
Number of Comment(s) in the code - 6
Number of Variable(s)/Function(s) in the code - 8
```

verbose == 1

```
### Find Interger(s) and Float number(s) ###
>>> verbose(1/0):1
>>> [Var/Func]: "foo123" is a variable or a function name.
>>> [Var/Func]: "f123" is a variable or a function name.
>>> [Var/Func]: "foo_234" is a variable or a function name.
>>> [Var/Func]: "f_234" is a variable or a function name.
>>> [Comment]: "// Testing bare comments" is a comment.
>>> [Multiline-Comment]: "/* Testing bare comments */" is a multilined-comment.
>>> [Multiline-Comment]: "/* Inline comment with 123 231.132 */" is a multilined-comment.
>>> [Multiline-Comment]: "/*
    Crossline with
    123,231 321.123
  */" is a multilined-comment.
>>> [Interger]: "105" is an interger.
>>> [Interger]: "123" is an interger.
>>> [Interger]: "-456" is an interger.
>>> [Float]: "12.21" is a float number.
>>> [Float]: "-34.46" is a float number.
>>> [Interger]: "123" is an interger.
>>> [Interger]: "-234" is an interger.
>>> [Interger]: "-111" is an interger.
>>> [Var/Func]: "i1" is a variable or a function name.
>>> [Interger]: "123" is an interger.
>>> [Var/Func]: "i2" is a variable or a function name.
>>> [Interger]: "-123" is an interger.
>>> [Comment]: "// comments as appendix with 321" is a comment.
>>> [Var/Func]: "j1" is a variable or a function name.
>>> [Float]: "567.89" is a float number.
>>> [Var/Func]: "j2" is a variable or a function name.
>>> [Float]: "-567.89" is a float number.
>>> [Comment]: "// comments as appendix with 321" is a comment.
>>> [Interger]: "0" is an interger.
File End.

### Conclusion ###
Number of Interger(s) in the code - 9
Number of Float number(s) in the code - 4
Number of Comment(s) in the code - 6
Number of Variable(s)/Function(s) in the code - 8
```

---

testcase2

```
#include <iostream>
#include <bits/stdc++.h>

using namespace std;

vector<int> v = {1, 2, -3, 7};

void foo123()
{
  int f123;
}

int main()
{

    // Testing bare comments
    /* Testing bare comments */

    /* Inline comment with 123 231.132 */
    /*
        Crossline with
        123,231 321.123
    */

    string s = "String with Numbers:123,-456,12.21,-34.46";
    int i1 = 123, i2 = -123;        // comments as appendix with 321
    float j1 = 567.89, j2 = -567.89; // comments as appendix with 321
    return 0;
}
```

verbose == 0

```
### Find Interger(s) and Float number(s) ###
>>> verbose(1/0):0
>>> [Interger]: "1" is an interger.
>>> [Interger]: "2" is an interger.
>>> [Interger]: "-3" is an interger.
>>> [Interger]: "7" is an interger.
>>> [Interger]: "123" is an interger.
>>> [Interger]: "-456" is an interger.
>>> [Float]: "12.21" is a float number.
>>> [Float]: "-34.46" is a float number.
>>> [Interger]: "123" is an interger.
>>> [Interger]: "-123" is an interger.
>>> [Float]: "567.89" is a float number.
>>> [Float]: "-567.89" is a float number.
>>> [Interger]: "0" is an interger.
File End.

### Conclusion ###
Number of Interger(s) in the code - 9
Number of Float number(s) in the code - 4
Number of Comment(s) in the code - 6
Number of Variable(s)/Function(s) in the code - 6
```

verbose == 1

```
### Find Interger(s) and Float number(s) ###
>>> verbose(1/0):1
>>> [Interger]: "1" is an interger.
>>> [Interger]: "2" is an interger.
>>> [Interger]: "-3" is an interger.
>>> [Interger]: "7" is an interger.
>>> [Var/Func]: "foo123" is a variable or a function name.
>>> [Var/Func]: "f123" is a variable or a function name.
>>> [Comment]: "// Testing bare comments" is a comment.
>>> [Multiline-Comment]: "/* Testing bare comments */" is a multilined-comment.
>>> [Multiline-Comment]: "/* Inline comment with 123 231.132 */" is a multilined-comment.
>>> [Multiline-Comment]: "/*
```

```
      Crossline with
      123,231 321.123
    */" is a multilined-comment.
>>> [Interger]: "123" is an interger.
>>> [Interger]: "-456" is an interger.
>>> [Float]: "12.21" is a float number.
>>> [Float]: "-34.46" is a float number.
>>> [Var/Func]: "i1" is a variable or a function name.
>>> [Interger]: "123" is an interger.
>>> [Var/Func]: "i2" is a variable or a function name.
>>> [Interger]: "-123" is an interger.
>>> [Comment]: "// comments as appendix with 321" is a comment.
>>> [Var/Func]: "j1" is a variable or a function name.
>>> [Float]: "567.89" is a float number.
>>> [Var/Func]: "j2" is a variable or a function name.
>>> [Float]: "-567.89" is a float number.
>>> [Comment]: "// comments as appendix with 321" is a comment.
>>> [Interger]: "0" is an interger.
File End.

### Conclusion ###
Number of Interger(s) in the code - 9
Number of Float number(s) in the code - 4
Number of Comment(s) in the code - 6
Number of Variable(s)/Function(s) in the code - 6
```

2020/5

17341137 宋震鹏