Flex-bison演示

编译原理期末项目

Flex 词法分析器介绍

作用：识别词组，将输入分隔成有意义的单元

Flex代码结构

- .l为文件后缀
- 由三部分组成，%%符号分割
  - 定义(definitions)
  - 规则(rules)
  - 代码(user code)

```
%{
#include <stdio.h>
%}

%%

([1-9]+[0-9]*)|[0]  ECHO;
[0-9]+\.[0-9]+
[a-zA-Z]+
.

%%

int main(int argc, char **argv)
{
  yylex();
  yywrap();
}
int yywrap()
{
    return 1;
}
```

flex代码示例：test.l

flex示例与解析

步骤：

1. 编写test.l文件
2. 编译文件flex test.l （生成C文件lex.yy.c）
3. 使用gcc编译成可执行文件 gcc lex.yy.c （得到可执行文件a.exe)
4. 运行a.exe开始测试。

```
Alice bought 3 apples, 2 pearls, they cost 1.5$
//理应输出3 2
int a=23;
//理应输出23
double b=0.45;
//理应输出空行
```

Bison　语法分析器介绍

作用：分析语法，分析输入单元间的联系

通常与flex联合使用

bison代码结构

- .y为文件后缀，通常与flex一起使用
- 也是由三部分组成，%%符号分割
  - 定义(definitions)
  - 规则(rules)
  - 代码(user code)
- 示例：左边是flex，

  右边是bison。

```
/*test.l*/
%{
#include <stdio.h>
#include "y.tab.h"
void yyerror(char *);
%}
NUM [1-9]+[0-9]*|0
%%

{NUM}                       return NUM;
"+"                         return ADD;
"-"                         return SUB;
"*"                         return MUL;
"/"                         return DIV;
[a-zA-Z_$]+[a-zA-Z_$0-9]*   return VAR;
\n                          return CR;
[ \t]+                     /* ignore whitespace */;
.
%%
```

```
/*test.y*/
%{
#include <stdio.h>
#include <string.h>
int yylex(void);
void yyerror(char *);
%}

%token NUM ADD SUB MUL DIV VAR CR

%%
       line_list: line
                | line_list line
                ;

            line : expression CR  {printf("YES\n");}

      expression: term
                | expression ADD term
                | expression SUB term
                ;

            term: single
                | term MUL single
                | term DIV single
                ;

          single: NUM
                | VAR
                ;
%%
void yyerror(char *str){
    fprintf(stderr,"error:%s\n",str);
}

int yywrap(){
    return 1;
}
int main()
{
    yyparse();
}
```

bison代码示例：test.y

bison示例与解析

步骤:

1. 编写test.l文件与test.y文件
2. 编译文件: flex test.l （生成C文件lex.yy.c）
3. 编译文件: bison –yacc –dv test.y （生成文件y.tab.c）
4. 使用gcc编译成可执行文件 gcc –o teest y.tab.c lex.yy.c (得到可执行文件test.exe，这里的test是指定生成的可执行文件的名字)
5. 运行test.exe开始测试。

```
输入：1+a-b*3/4
输出：YES
输入：2**5-t
输出：报错退出
```

下载与安装

- 下载地址：
  - Flex: https://gnuwin32.sourceforge.net/packages/flex.htm (点setup)
  - Bison: https://gnuwin32.sourceforge.net/packages/bison.htm （点setup）
  - Gcc: 推荐使用mingw（涉及路径配置，推荐找教程对着安装）仅供参考：https://zhuanlan.zhihu.com/p/76613134