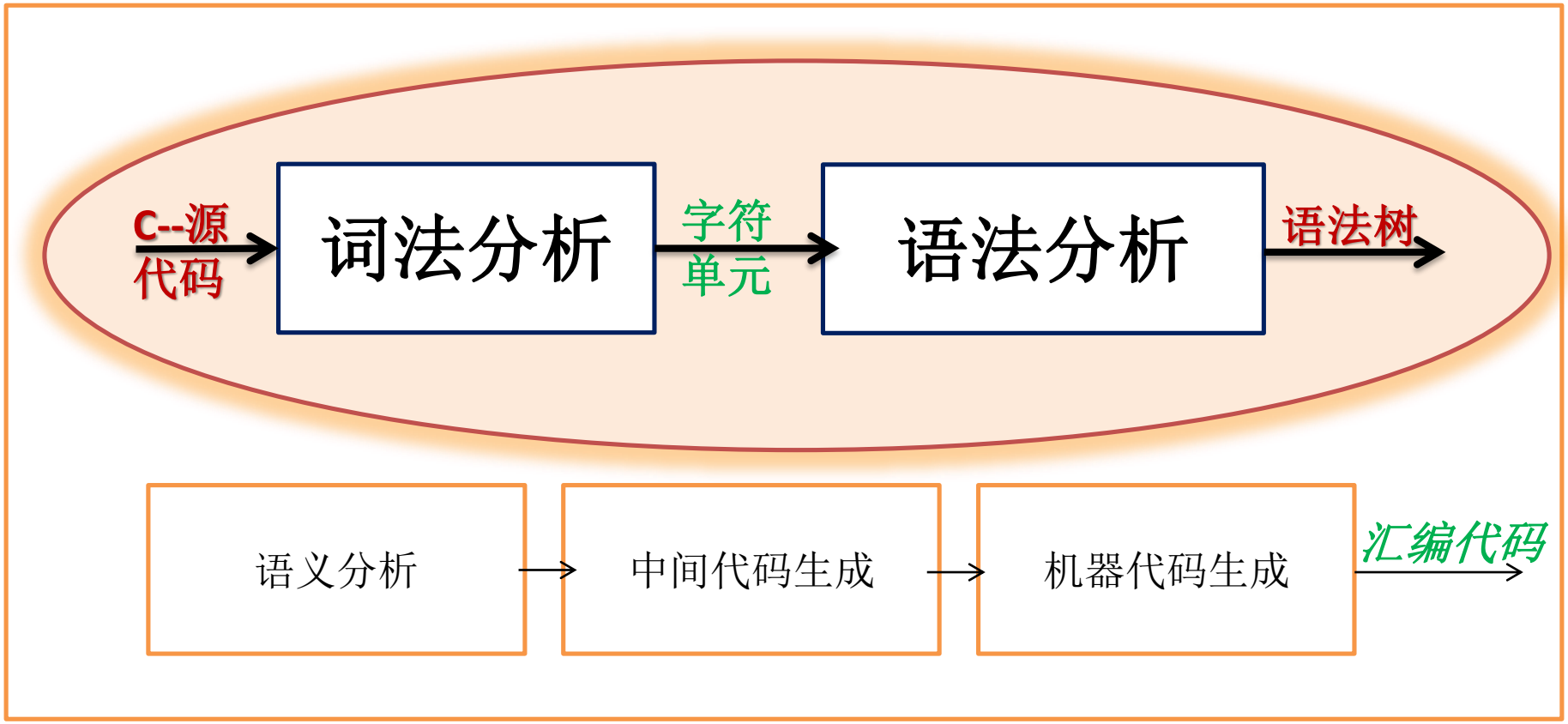
实验一 词法分析和语法分析

任课老师：戴新宇
助教：
尚迪(shangd@nlp.nju.edu.cn)

编译器模块分解图



概要

- 提交说明
- 实验任务（必做 + 选做）
  - 所有任务完成可以得到100%的分数。
  - 只做必做最多只能得到80%的分数。
- 编译环境及过程
  - 词法分析与flex
  - 语法分析与bison
- 实验讲解
  - Flex
  - Bison

提交说明

- **ftp地址：ftp://114.212.190.181:40/**
- **上传账号/密码： upload/upload**
- **格式：学号+实验编号命名的压缩包 (*zip/rar*) 如：121220000_lab1.rar**
- **截止日期：4月3日 晚上12：00之前**
- **内容：**
  - 源程序(*ex1.**l**, ex1.**y**; 额外的.c文件*)
  - 可执行程序(*命名为parser*)
  - 报告PDF(*完成的功能点，编译步骤，实现方法，结点的数据结构表示；不超过3页*)
- **备注：可重新提交**
  - **加后缀：121220000_lab1_1.rar 121220000_lab1_2.rar**

实验任务

- 必做：(./parser test.cmm)
  - 无错误：打印语法树
  - 识别错误类型1: 词法错误
  - 识别错误类型2: 语法错误
- 选做：
  - 选做1：两种风格的注释：//，/* */
  - 选做2：八进制数：012
  - 选做3：十六进制：0xa，0Xa，0xA，0XA
  - 选做4：指数形式的浮点数：1E1，01e1

```
int main()
{
    int I = 1;
    int j = ~I;
}
```

Error type 1 at line 4: Mysterious character '~'

```
int main()
{
    float a[10][2];
    int i;
    a[5,3] = 1.5;
    if (a[1][2]==0) i=1 else i = 0;
}
```

Error type 2 at line 4: syntax error

Error type 2 at line 5: syntax error

```
int inc()
{
    int i;
    i = i+1;
}
```

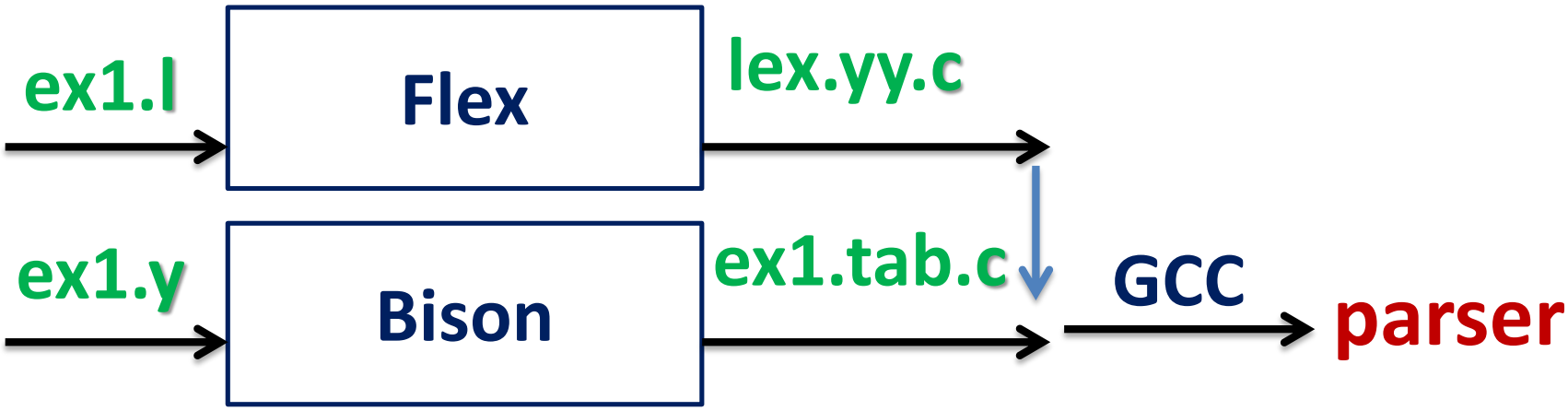
```
Program (1)
  ExtDefList (1)
    ExtDef (1)
      Specifier (1)
        TYPE: int
      FunDec (1)
        ID: inc
        LP
        RP
      CompSt (2)
        LC
        DefList (3)
          Def (3)
            Specifier (3)
              TYPE: int
            DecList (3)
              Dec (3)
                VarDec (3)
                  ID: i
            SEMI
        StmtList (4)
          Stmt (4)
            Exp (4)
              Exp (4)
                ID: i
              ASSIGNOP
              Exp (4)
                Exp (4)
                  ID: i
                PLUS
                Exp (4)
                  INT: 1
              SEMI
        RC
```

编译环境及过程

- GNU Flex, GNU Bison, GCC, Linux Ubuntu
  - sudo apt-get install *flex*
  - sudo apt-get install *bison*
- 源文件{ex1.l, ex1.y} → 可执行程序parser
  - Flex： ex1.l → lex.yy.c
  - Bison: ex1.y → ex1.tab.c
  - GCC： *.c → parser

./parser test.c //测试命令

编译方法



编译命令：

flex ex1.l

bison -d ex1.y

gcc -o parser ex1.tab.c

-ll: lib of lex -lfl: lib of flexx -ly: lib of yacc

Flex & Bison

%{ ***ex1.l***

Declarations

#include "ex1.tab.h"

%}

Definitions (**RegEx**)

%%

Rules

%%

subroutines (***e.g main***)

%{ ***ex1.y***

Declarations

#include "lex.yy.c"

%}

Definitions (**%Token**)

%%

Productions

%%

subroutines

Flex： .l文件格式

```
%{
Declarations
%}
Definitions
%%
Rules        //将保留字置于标识符{id}之前
%%
subroutines
```

Flex：一个简单的flex程序

test.l

```
%{
  /* 此处省略#include部分 */
  int chars = 0;
  int words = 0;
  int lines = 0;
%}
letter  [a-zA-Z]
%%
{letter}+  { words++; chars+= yyleng; }
\n  { chars++; lines++; }
.  { chars++; }
%%
int main(int argc, char** argv) {
  if (argc > 1) {
    if (!(yyin = fopen(argv[1], "r"))) {
      perror(argv[1]);
      return 1;
    }
  }
  yylex();
  printf("%8d%8d%8d\n", lines, words, chars);
  return 0;
}
```

flex test.l

gcc lex.yy.c –lfl –o parser

./parser test.cmm

{comment}
{comment2}
{ws}
Int
Float
struct
return
while
if
else
{int}
{float}
{id}
";“
",“
"=“
"<“
……

Flex：你需要做的内容

- **yytext，yyleng:** 词素字符串
- **yylineno:** *%option yylineno*
- **yylval:** 全局变量，当前词法单元的属性值

```
id       {letter}({letter}|{digit})*
{id}  {
        printf("Line %d:(ID   , %s)\n",yylineno,yytext);
      }
{id}  {  //printf("Line %d:(ID   , %s)\n",yylineno,yytext);
    yylval = (struct treeNode*)malloc(sizeof(struct treeNode));
    yylval->lineno = yylineno;
    yylval->type = 1;
    yylval->tokentype = 26;
    yylval->name = malloc(strlen(yytext)+1);
    strcpy(yylval->name,yytext);
    return ID;
    }
```

Bison: .y文件格式

%{

Declarations

%}

Definitions

//优先级与结合性

```
%right  ASSIGNOP
%left   AND
%left   RELOP
%left   PLUS MINUS
%left   STAR DIV
%right  NOT UMINUS
%left   DOT LB RB LP RP
```

%%

Productions

Exp | MINUS EXP ***%prec*** UMINUS

%%

subroutines

Bison语法树创建与打印

- 多叉树的构建：
  - Exp : ID { $$ = createNode(1,$1); }
  - Exp : MINUS EXP { $$ = createNode(2,$1,$2); }
  - *#include<stdarg.h>*

  struct Node* createNode(int arity, …);
- 递归层次的前序遍历

  void printNode(struct Node* root, int nLayer);

Bison文法符号结点的数据结构

- 保存的信息：*struct Node{ … };*
  - 结点类型： 非终结符，终结符(数, 标识符…)
  - 结点名字： Exp，TYPE，ID
  - 所在行号： *%option yylineno*
  - 字符串属性值：TYPE.int，ID.lexeme
  - 数值属性值：INT，FLOAT
  - 多叉树孩子：*int arity,*

  struct Node children[N]; //vector<Node*>*

  -…(可扩展)

语法解析的错误恢复产生式

- Bison在当前状态对yylex()返回的token没有定义 时即发生了语法错误，调用yyerror:

  yyerror(char* str){ printf(“syntax error\n”); }

- Bison不断丢弃词法单元直至遇到同步 单元 (例如：分号，右括号)

- 机制：错误恢复产生式

  Stmt: error SEMI

Warning！！！

任何形式的代码抄袭都是不能容忍的

一旦发现，抄袭者和被抄袭者均为0分