实验 8 Follow 集合

实验难度：★★☆☆☆
建议学时：2 学时

一、 实验目的

- 了解在上下文无关文法中的 First 集合和 Follow 集合的定义。
- 掌握计算 First 集合和 Follow 集合的方法。

二、 预备知识

- 对 First 集合和 Follow 集合的定义有初步的理解。读者可以参考配套的《编译原理》教材，预习这一部分内容。

三、 实验内容

3.1 阅读实验源代码

Follow.h 文件

主要定义了与文法和集合相关的数据结构。

有关文法的数据结构定义了文法的单链表存储形式。其中 Rule 结构体用于定义文法的名称和文法链表，RuleSymbol 结构体用于定义文法产生式中的终结符和非终结符（**注意：**此处的RuleSymbol结构体与之前在消除左递归和提取左因子中的RuleSymbol结构体有一些差异，原因是在计算 First 集合和 Follow 集合时，将文法中的每个选择都单独写成了一个产生式，所以 RuleSymbol 结构体就进行了相应的简化）。

有关集合的数据结构定义了集合的线性表存储形式。其中 Set 结构体用于定义集合的名称和终结符数组，SetList 结构体用于定义一个以 Set 为元素的线性表。具体内容可参见下面的表格。

Rule 的域	说明
RuleName	文法的名称
pFirstSymbol	指向文法的第一个 Symbol
pNextRule	指向下一条文法

RuleSymbol 的域	说明
pNextSymbol	指向下一个 Symbol
isToken	是否为终结符。1 表示终结符，0 表示非终结符
SymbolName	终结符和非终结符的名称

Set 的域	说明
Nane	集合的名称
Terminal	终结符数组
nTerminalCount	终结符数组元素个数。与前一项构成一个线性表

SetList 的域	说明
Sets	集合数组
nSetCount	集合数组元素个数。与前一项构成一个线性表

下面是一个简单文法，并使用图表说明了该文法的存储结构：

```
A -> Aa
A -> aB
B -> bB
```

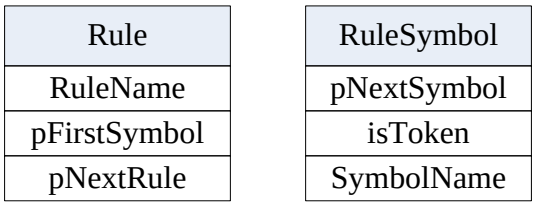

图 8-1：Rule 和 RuleSymbol 结构体图例

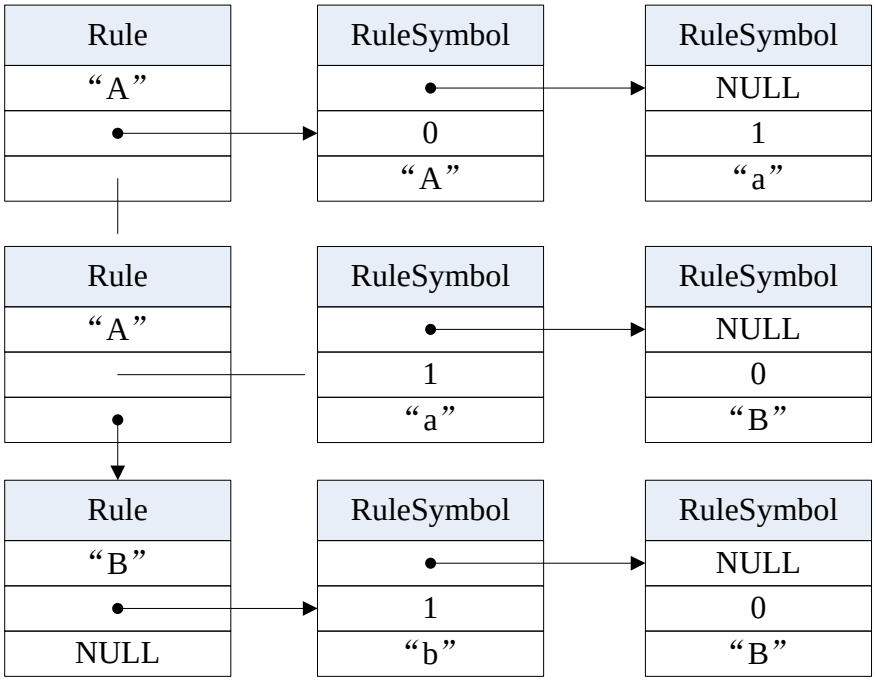

图 8-2：简单文法的存储结构

main.c 文件

定义了 main 函数。在 main 函数中首先调用 InitRules 函数初始化了文法，然后调用了 PrintRule 函数打印文法，接着初始化了保存 First 集合和 Follow 集合的线性表，最后调用 Follow 函数求文法的 First 集合和 Follow 集合。

在 main 函数的后面，定义了一系列函数，有关这些函数的具体内容参见下面的表格。关于这些函数的参数和返回值，可以参见其注释。

函数名	功能说明
AddOneSet	添加一个 Set 到 SetList 中。 **提示：** ● 在这个函数中可以首先调用一次 GetSet 函数，用来判断该 SetList 是否已经存在同名的 Set，从而忽略重复的名称。

	此函数的函数体还不完整，留给读者完成。
GetSet	根据名称在 SetList 中查找 Set。此函数的函数体还不完整，留给读者完成。
AddTerminalToSet	添加一个终结符到 Set。 **提示：** ● 需要忽略重复的终结符。 此函数的函数体还不完整，留给读者完成。
AddSetToSet	将源 Set 中的所有终结符添加到目标 Set 中。 **提示：** ● 需要忽略源 Set 中的 ε 。 ● 可以通过调用 AddTerminalToSet 函数将终结符加入到目标 Set 中。 此函数的函数体还不完整，留给读者完成。
SetHasVoid	判断 Set 的终结符中是否含有 ε 。此函数的函数体还不完整，留给读者完成。
First	求文法的 First 集合。 **提示：** ● 由于对一个文法符号（一个终结符或非终结符）求 First 集合可能会依赖于其他的文法符号的 First 集合，所以设置了一个标识 isChange，并初始化为 0，每当集合列表中的某个集合发生变化后就将该标识赋值为 1，并将此标识作为循环条件，直到没有变化发生时，才能结束循环。 ● 在循环末尾，如果当前文法产生式的每一个符号的 First 集合都包含 ε，就将 ε 终结符添加到当前文法名称的 First 集合中。 此函数的函数体还不完整，留给读者完成。
Follow	求文法的 Follow 集合。 **提示：** ● 如果 A 是开始符号（第一个文法产生式的左边）那么就将终结符“$”加入 Follow（A）。 ● 给出一个非终结符 A，那么集合 Follow（A）则是由终结符组成，此外可能还有“$ ” ● 在函数的开始位置首先要调用 First 函数，求文法的 First 集合。 ● 由于对一个文法符号（非终结符）求 Follow 集合可能会依赖于其他的文法符号的 First 集合或者 Follow 集合，所以 设置了一个标识 isChange，并初始化为 0，每当 Follow 集合列表中的某个集合发生变化后就将该标识赋值为 1，并将此标识作为循环条件，直到没有变化发生时，才能结束循环。 此函数的函数体还不完整，留给读者完成。
InitRules	使用给定的数据初始化文法链表。
CreateRule	创建一个新的 Rule。
CreateSymbol	创建一个新的 Symbol。
PrintRule	输出文法。此函数的函数体还不完整，留给读者完成。

3.2 为函数 InitRules 添加注释

在 InitRules 函数之前定义了两个结构体，这两个结构体用来定义初始化文法数据的存储形式。具体内容可参见下面两个表格。

SYMBOL 的域	说明
isToken	是否为终结符。1 表示终结符，0 表示非终结符。
SymbolName	终结符和非终结符的名称。

RULE_ENTRY 的域	说明
RuleName	文法的名称。
Symbols	SYMBOL 结构体数组，其中每个元素表示一个终结符或非终结符。

为 InitRules 函数添加注释。（注意，在本程序中使用了指针的指针，体会其在单链表的插入和删除操作中的作用）。

3.3 为 PrintRule 函数编写源代码

为 PrintRule 函数编写源代码，同时理解在本程序中文法的链式存储结构，编写完源代码后，选择“Run”菜单中的“Run Without Debugging”，会在“TERMINAL”窗口输出文法的产生式。

3.4 编写源代码并通过验证

按照下面的步骤继续实验：

1. 为 Follow 函数和其他未完成的函数编写源代码。注意尽量使用已定义的局部变量。
2. 按 Ctrl+Shift+B,在弹出的下拉列表中选择“生成项目”。如果生成失败，根据“TERMINAL”窗口中的提示信息修改源代码中的语法错误。
3. 按 Ctrl+Shift+B，然后在下拉列表中选择“测试”，启动验证。在“TERMINAL”窗口中会显示验证的结果。如果验证失败，可以在“EXPLORER”窗口中，右击“result_comparation.html”文件，在弹出的菜单中选择“Open Preview”，可以查看用于答案结果文件与读者编写程序产生的结果文件的不同之处，从而准确定位导致验证失败的原因。

四、思考与练习

1. 编写一个 FreeRule 函数，当在 main 函数的最后调用此函数时，可以将整个文法的内存释放掉，从而避免内存泄露。
2. 使用自己编写的代码对下面的例子进行验证，确保程序可以为所有形式的文法求 Follow 集合，并验证通过（文法中的终结符用粗体表示）。

   A -> B
   A -> **ε**
   B -> C
   B -> **ε**
   C -> AB
   C -> ***
3. 编写一个程序，此程序可以根据 First 集合和 Follow 集合判断文法是否为 LL(1)文法。
4. 编写一个程序，此程序可以根据 LL(1)文法的 First 集合和 Follow 集合构造 LL(1)分析表。
5. 编写一个程序，此程序可以使用 LL(1)分析表自动分析输入的字符串，并输出 LL(1)的分析动作和错误信息。