Robot Specific Language Report

一、实验内容

领域特定语言（Domain Specific Language，DSL）可以提供一种相对简单的文法，用于特定领域的业务流程定制。本作业要求定义一个领域特定脚本语言，这个语言能够描述在线客服机器人（机器人客服是目前提升客服效率的重要技术，在银行、通信和商务等领域的复杂信息系统中有广泛的应用）的自动应答逻辑，并设计实现一个解释器解释执行这个脚本，可以根据用户的不同输入，根据脚本的逻辑设计给出相应的应答。

- 脚本语言的语法可以自由定义，只要语义上满足描述客服机器人自动应答逻辑的要求。
- 程序输入输出形式不限，可以简化为纯命令行界面。
- 应该给出几种不同的脚本范例，对不同脚本范例解释器执行之后会有不同的行为表现。

二、 实验环境

- Windows 11
- Visual Studio 2022

三、 程序设计

1. 数据结构设计

1.1 记号类 **Token**

```
class Token {
    int mark;           // 记号的类型
    string attribute;   // 记号的属性
public:
    explicit Token(int mark = -1, string attribute = " ");
    Token(const Token& token);
    ~Token();
    int getMark() const;
    string getAttribute();
    void setMark(int i);
    void setAttribute(string attribute);
    void output();
};
```

1.2 记号流类 TokenStream

```
class TokenStream {
    int size;           // 记号流的最大空间
    int length;         // 记号流的有效长度
    Token* tokens;      // 保存记号流的记号数组
public:
    explicit TokenStream(int size = 50);
    TokenStream(TokenStream& tokenStream);
    ~TokenStream();
    int getLength() const;
    Token getToken(int index);
    void append(int mark, string attribute);
    void output();
};
```

1.3 状态类 State

```
class State {
    StateId id;                 // 状态的标识
    vector<string> write;       // 状态中要输出的字符串
    int read;                   // 状态中要等待输入的时间长度，以秒为单位
    map<string, StateId> rouse; // 状态的调用关系，以关键词调用对应的状态标识
    StateId silence;            // 状态中的沉默处理，即处理用户无输入的情况
```

```
    StateId defaults;           // 状态中的默认处理，即处理用户输入无法解析的情况
    bool exit;                  // 标识是否跳转至结束状态；若为1，则状态结束后跳转至结束状态；否则，正常结束
public:
    State() = default;
    State(StateId id);
    State(const State& state);
    ~State();
    void setWrite(string sentence);
    string getWrite();
    bool isWriteEmpty();
    void setRead(int period);
    int getRead();
    void insertRouse(string key, StateId value);
    StateId getRouse(StateId key);
    vector<string> getRouseKeys();
    void setSilence(StateId silence);
    StateId getSilence();
    void setDefaults(StateId defaults);
    StateId getDefaults();
    void setExit(bool exit);
    bool getExit();
    void output(void);
};
```

1.4 语法分析树类 Parser

```
class Parser {
    int index;                  // 语法分析树中当前处理的记号流中的记号的索引
    TokenStream tokens;         // 语法分析树需要处理的记号流
    StateId entry;              // 语法分析树的入口状态的状态标识
    StateId currentState;       // 语法分析树当前处理的状态
public:
    map<StateId, State> states; // 语法分析树的状态对应表
    vector<string> varname;     // 语法分析树的变量表
    Parser(TokenStream tokens);
    ~Parser();
    int getIndex(void);
    void updateIndex(void);
```

```
    void rollbackIndex(void);
    void setEntry(StateId entry);
    StateId getEntry();
    Token getToken();
    bool dealState();
    bool dealWrite();
    bool dealRead();
    bool dealRouse();
    bool dealSilecne();
    bool dealDefaults();
    void dealExit();
    void output();
};
```

2. 模块划分与模块调用关系图

客服机器人项目主要分为服务器和客户端两个模块，服务器负责利用脚本生成解释执行程序并提供服务器功能，客户端负责处理用户的输入并将服务器的消息输出给用户，服务器和客户端两个模块配合运行便实现在线客服机器人功能。

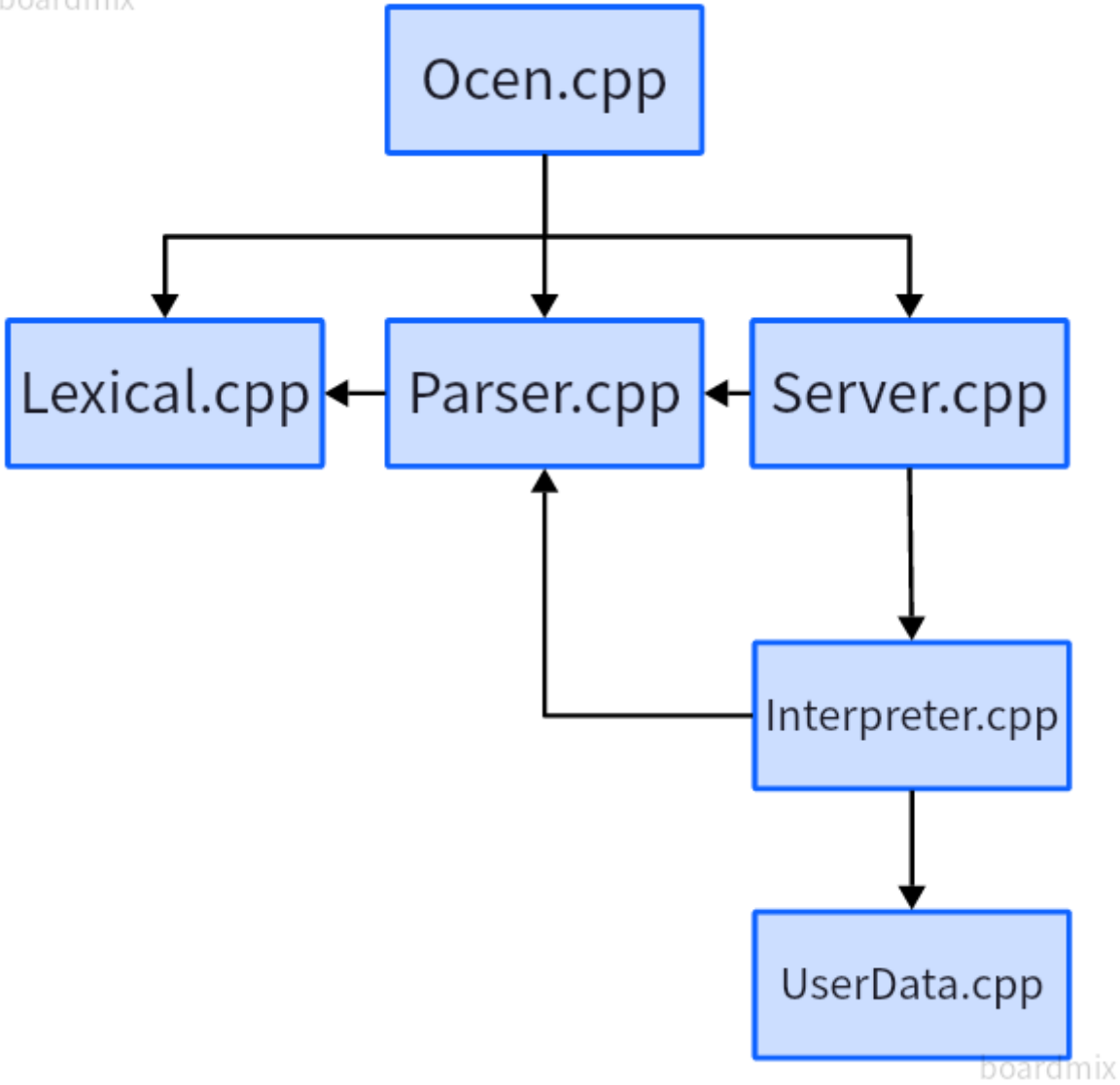
2.1 服务器

模块划分

服务器主要划分为词法分析模块Lexical、语法分析模块Parser、解释执行模块Interpreter、服务器模块Server和用户数据模块UserData。

- 词法分析模块Lexical：负责从脚本中读取单词并进行处理，生成记号流。
- 语法分析模块Parser：负责将词法分析模块生成的记号流处理转化为语法分析树
- 用户数据模块UserData：测试桩模块，负责利用用户名生成对应的用户信息
- 服务器模块Server：负责监听客户端连接请求
- 解释执行模块Interpreter：负责通过与客户端的通信、利用语法分析树和执行环境解释执行脚本

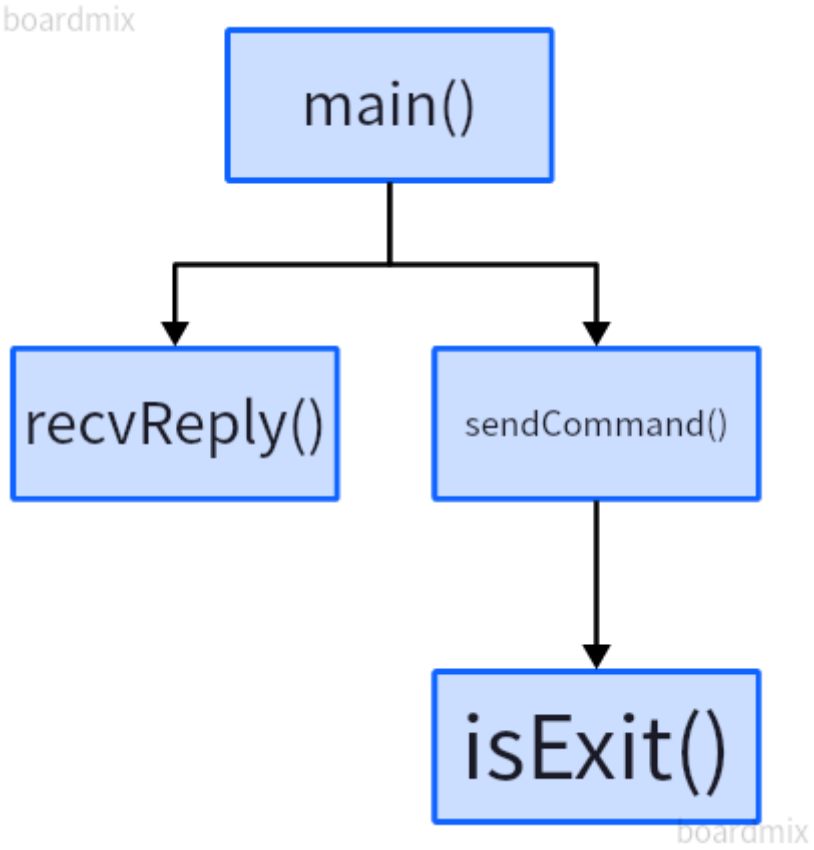
模块调用关系图



2.2 客户端模块划分

客户端功能简单，无需进行模块划分；但客户端功能主要分为三部分：请求与服务器建立连接、获取服务器返回的消息、向服务器发送用户输入的命令，分别由主函数main、recvReply回复接收函数、sendCommand发送命令完成；同时还有识别服务器退出通知的isExit()函数。

函数调用关系图



3. 函数说明

3.1 服务器模块函数说名

见附件1-服务器模块函数说明HTML文档

3.2 客户端模块函数说明

3.2.1 消息接收函数 **recvReply()**

```
/**
 * @brief 通过socket接收服务器的消息
 * @param sock 与服务器通信的socket
 * @param isQuit 保存服务器退出信息的布尔值
*/
void recvReply(shared_ptr<tcp::socket> sock, bool& isQuit)
```

3.2.2 命令发送函数 sendCommand()

```
/**
 * @brief 从标准输入获取用户的命令并发送给服务器
 * @param sock 与服务器通信的socket
*/
void sendCommand(shared_ptr<tcp::socket> sock)
```

3.2.3 退出通知检测函数 isExit()

```
/**
 * @brief 判断服务器的消息是否包含退出通知
 * @param reply 待处理的服务器消息
 * @return 若包含退出通知，则返回true；否则，返回false
*/
bool isExit(string& reply)
```

4. 接口

接口定义了一个抽象规范，描述了模块间的行为，有助于实现代码的模块化和松耦合性，使得程序更易于维护和扩展。从程序接口和人机接口两个方面展开。

4.1 程序接口

客服机器人的程序接口用与规范模块之间的调用行为，每个模块都定义了接入接口

4.1.1 词法分析模块 Lexical 接口——lexical()

```
/**
 * @brief 词法分析函数，处理脚本并转化为记号流
 * @rslPath 保存机器人语言脚本路径的字符串
 * @return 保存脚本对应的记号流
*/
TokenStream* lexical(string rslPath);
```

4.1.1 语法分析模块 Parser 接口——parser()

```
/**
 * @brief 语法分析函数，处理记号流并生成语法分析树
 * @param tokens 待处理的记号流
 * @return  报错记号流对应的语法分析树
*/
Parser* parser(TokenStream* tokens);
```

4.1.1 解释执行模块 Interpreter 接口——interpreter()

```
/**
 * @brief 用语法分析树和执行环境进行解释执行函数
 * @param parser 语法分析树
 * @param sock 与客户端通信的socket
*/
void interpreter(Parser* parser, shared_ptr<tcp::socket> sock);
```

4.1.1 服务器模块 Server 接口——server()

```
/**
 * @brief 服务器函数，通过脚本对应的语法分析树提供多客户端的服务器功能
 * @param parser 对应的语法分析树
*/
void server(Parser* parser);
```

4.1.1 用户数据模块 UserData 接口——initUserData()

```
/**
 * @brief 初始化用户数据
 * @param username 保存用户名的字符串
 * @return 返回初始化后的用户数据
*/
UserData* initUserData(string username);
```

4.2 人机接口

人机接口采用命令行的方式完成，从服务器人机接口和客户端人机接口展开：

4.2.1 服务器人机接口

在命令输入下方命令便可启动服务器：

> *Ocen.exe rslFilePath*

其中：

- Ocen.exe是服务器程序名
- rslFilePath 是要运行的脚本的文件路径

4.2.2 客户端人机接口

在客户端输入下方命令便可启动客户端：

> *Ocen_Client.exe*

其中：

- Ocen_Client.exe是客户端程序名

进入客户端程序后：

- 服务器发送的消息将会以如下方式显示

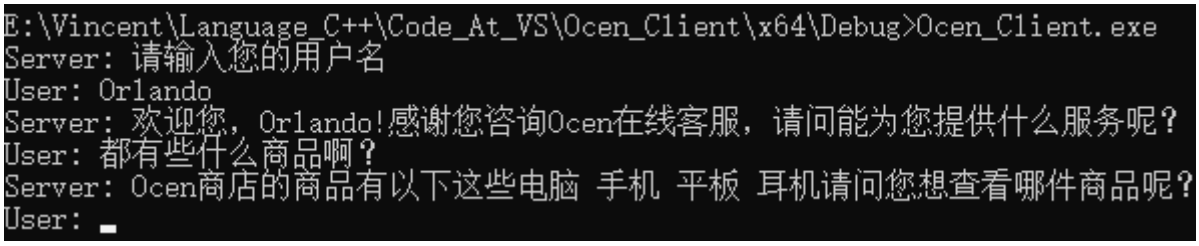
  > *Server:【reply】*

  【reply】代表服务返回的回复
- 用户输入的命令将会以如下方式显示

  > *User:【command】*

  【command】代表用户输入的命令
- 用户可进行模糊化的命令输入，即只需输入的命令中包含脚本调用关键词即可

实际效果如下所示图：



四、测试

1. 测试桩

测试桩为用户数据测试桩，用户模拟从数据库中获取用户数据的过程，位于用户数据UserData模块

1.1 测试桩接口——initUserData()

```
/**
 * @brief 初始化用户数据
 * @param username 保存用户名的字符串
 * @return 返回初始化后的用户数据
*/
UserData* initUserData(string username);
```

如果输入的用户名已注册，则会提示这是一个注册用户；否则，提示这是一个未注册用户

1.2 测试桩示例用法

使用下方代码调用测试桩接口:

```
    initUserData("Orlando");
    initUserData("Violet");
    initUserData("李华");
    initUserData("莉莉");
```

获得测试结果如下：

> *Orlando*是一个注册用户
>
> *Violet*是一个注册用户
>
> 李华是是一个未注册用户
>
> 莉莉是一个未注册用户

2. 自动测试脚本

使用windows下的batch批处理脚本，脚本功能是启动服务器，然后预准备的多个用户输入运行客户端，将运行结果与预测结果进行比对，同时输出比对结果。

2.1 自动测试脚本

```
@echo off
setlocal enabledelayedexpansion

set "failed_tests=0"
set "successful_tests=0"

start "" "..\..\Ocen\bin\Ocen.exe"
../../Ocen/Robot/script/store.txt

for %%i in (test*.txt) do (
    ..\x64\Debug\Ocen_Client.exe < "%%i" > "output%%i"

    fc "output%%i" "answer%%i" > nul
    if errorlevel 1 (
        echo %%i : BAD
        set /a failed_tests+=1
    ) else (
        set /a successful_tests+=1
```

```
    )
)

echo.
echo Tests passed: %successful_tests%
echo Tests failed: %failed_tests%

if %failed_tests% equ 0 (
    echo All tests passed successfully!
) else (
    echo Some tests failed. Check the output for details.
)

endlocal
```

2.2 预准备用户输入示例

用户输入示例**1**

> *Orlando*
> 你们有什么商品？
> 我想看看平板

用户输入示例**5**

> *Orlando*
> 我要投诉你们
> 你们的服务太差劲了

2.3 预测运行结果示例

预测运行结果示例**1**

> *Server*: 请输入您的用户名
> *User*: *Server:* 欢迎您，*Orlando!*感谢您咨询*Ocen*在线客服，请问能为您提供什么服务呢？
> *User*: *Server*: *Ocen*商店的商品有以下这些电脑 手机 平板 耳机请问您想查看哪件商品呢？
> *User*: *Server*: *Ocen*平板的信息如下：价格 *2500* 分辨率 *2K* 内存 *256GB* 操作系统 *OcenOS*
> *Server*: 谢谢您对*Ocen*的支持，期待与您的再次相遇，再见*!*

预测运行结果示例**5**

> *Server*: 请输入您的用户名
> *User*: *Server:* 欢迎您，*Orlando!*感谢您咨询*Ocen*在线客服，请问能为您提供什么服务呢？
> *User*: *Server*: 您的意见是我们改进的动力，请问您还有什么需要补充的吗？
> *User*: *Server:* 感谢您对*Ocen*的支持！期待您的再次惠顾！
> *Server*: 谢谢您对*Ocen*的支持，期待与您的再次相遇，再见*!*

2.4 自动测试脚本运行结果示例

> *Tests passed:*
> *Tests failed: 0*
> *All tests passed successfully*

```
E:\Vincent\Language_C++\Code_At_VS\Ocen_Client\autotest>autotest.bat

Tests passed: 5
Tests failed: 0
All tests passed successfully
```

五、记法 **Ocen**

1. 记法描述

1.1 关键词

Ocen记法的关键字有下方六种：

关键字	描述
State	状态关键字
Write	输出关键字
Read	输入关键字
Rouse	调用关键字
Default	默认调用关键字
Silence	沉默调用关键字
Exit	退出关键字

1.2 变量名

Ocen记法的变量名采用如下规则：变量名必须包裹在括号中

> *{varname}*

变量名样例如下：

> *{username}*
>
> *{product}*
>
> *{bill}*
>
> *{order}*

1.3 状态

Ocen记法的状态规则如下：使用State关键字声明，后跟状态名和冒号: 而且状态名只能由字母组成

> *State stateName:*

状态声明样例如下：

> *State welcome:*
>
> *State billProc:*
>
> *State thanks:*
>
> *State silenceProc:*

1.4 输出

Ocen记法的输出规则如下： 使用Write关键字声明， 后跟输出标识 << 最后加入字符串或变量名的随机排列，以空格分割。其中字符串必须包裹在引号""之中

> *Write << "string" {varname} "string" "string" {varname}*

输出样例如下：

> *Write << "欢迎您，" {username} "!感谢您咨询Ocen在线客服，请问能为您提供什么服务呢？"*
>
> *Write << "Ocen商店的商品有以下这些" {product} "请问您想查看哪件商品呢？"*
>
> *Write << "Ocen电脑的信息如下：" {computer}*

1.5 输入

Ocen记法的输入规则如下：使用Read关键字声明，后跟输入表示>>, 再跟一个整数，表示接收输入的最大时间，单位为秒

> *Read >> INTERGER*

输入样例如下：

> *Read >> 20*
>
> *Read >> 5*
>
> *Read >> 15*

1.6 调用

Ocen记法的调用规则如下：使用Rouse关键字声明，后跟调用关键词字符串，再跟对应的状态名

> *Rouse "keyWord" stateName*

调用样例如下：

> *Rouse "电脑" computerProc*
>
> *Rouse "耳机" podsProc*
>
> *Rouse "商品" productProc*

1.7 沉默调用

Ocen记法的沉默调用规则如下：使用Silence关键字声明，后跟对应的状态名

> *Silence stateName*

沉默调用样例如下：

> *Silence silenceProc*
>
> *Silence thanks*

1.8 默认调用

Ocen记法的默认调用规则如下：使用Default关键字声明，后跟对应的状态名

> *Default stateName*

沉默调用样例如下：

> *Default defaultProc*
>
> *Default thanks*

1.9 退出

Ocen记法的退出规则如下：使用Exit关键字声明即可

> *Exit*

2. 脚本样例

2.1 脚本样例1

```
State welcome:
    Write << "欢迎您，" {username} "!感谢您咨询Ocen在线客服，请问能为您提供什么服务呢？"

    Read >> 20

    Rouse "商品" productProc

    Rouse "账单" billProc

    Rouse "投诉" complainProc

    Silence silenceProc

    Default defaultProc

State productProc:
    Write << "Ocen商店的商品有以下这些" {product} "请问您想查看哪件商品呢？"

    Read >> 20

    Rouse "电脑" computerProc

    Rouse "手机" phoneProc

    Rouse "平板" padProc

```

```
    Rouse "耳机" podsProc

    Silence silenceProc

    Default thanks

State computerProc:
    Write << "Ocen电脑的信息如下：" {computer}

    Exit
State phoneProc:
    Write << "Ocen手机的信息如下：" {phone}

    Exit
State padProc:
    Write << "Ocen平板的信息如下：" {pad}

    Exit
State podsProc:
    Write << "Ocen耳机的信息如下：" {pods}

    Exit
State billProc:
    Write << "您目前的订单信息如下：" {order}

    Exit
State complainProc:
    Write << "您的意见是我们改进的动力，请问您还有什么需要补充的吗？"

    Read >> 20

    Default thanks
State silenceProc:
    Write << "您未输入任何信息，将在20秒后断开连接。"

    Read >> 20

    Rouse "商品" productProc

    Rouse "账单" billProc

    Rouse "投诉" complainProc
```

```
69
70      Silence thanks
71
72      Default defaultProc
73  State defaultProc:
74      Write << "十分抱歉，未能明白您的意思，请您再说详细一些。"
75
76      Read >> 20
77
78      Rouse "商品" productProc
79
80      Rouse "账单" billProc
81
82      Rouse "投诉" complainProc
83
84      Silence silenceProc
85
86      Default defaultProc
87  State thanks:
88      Write << "感谢您对Ocen的支持！期待您的再次惠顾！"
89
90      Exit
```

2.2 脚本样例2

```
1   State welcome:
2       Write << "欢迎您，" {username} "!感谢您咨询Ocen通信在线客服，请问能为您提供什么服务呢？"
3
4       Read >> 20
5
6       Rouse "流量" flowProc
7
8       Rouse "话费" billProc
9
10      Rouse "投诉" complainProc
11
12      Silence silenceProc
```

```
13
14      Default defaultProc
15
16  State flowProc:
17      Write << "您的流量信息如下:" {flow}
18
19      Exit
20  State billProc:
21      Write << "您的话费信息如下：" {bill}
22
23      Exit
24  State complainProc:
25      Write << "您的意见是我们改进的动力，请问您还有什么需要补充的吗？"
26
27      Read >> 20
28
29      Default thanks
30  State silenceProc:
31      Write << "您未输入任何信息，将在20秒后断开连接。"
32
33      Read >> 20
34
35      Rouse "流量" flowProc
36
37      Rouse "话费" billProc
38
39      Rouse "投诉" complainProc
40
41      Silence thanks
42
43      Default defaultProc
44  State defaultProc:
45      Write << "十分抱歉，未能明白您的意思，请您再说详细一些。"
46
47      Read >> 20
48
49      Rouse "流量" flowProc
50
51      Rouse "话费" billProc
52
53      Rouse "投诉" complainProc
54
```

```
    Silence silenceProc

    Default defaultProc
State thanks:
    Write << "感谢您对Ocen通信的支持！"

    Exit
```

2.3 脚本样例3

```
State welcome:
    Write << "欢迎您，" {username} "!感谢您咨询Ocen理发店在线客服，请问能为您提供什么服务呢？"

    Read >> 20

    Rouse "发型" hairProc

    Rouse "会员" vipProc

    Rouse "投诉" complainProc

    Silence silenceProc

    Default defaultProc

State hairProc:
    Write << "Ocen理发店的发型有以下这些" {haircuts} "请问您了解哪个发型呢？"

    Read >> 20

    Rouse "毛寸" buzzcutProc

    Rouse "烫发" permProc

    Rouse "染发" dyeProc

```

```
27      Silence silenceProc
28
29      Default thanks
30
31  State buzzcutProc:
32      Write << "毛寸发型的信息如下：" {buzzcut}
33
34      Exit
35  State permProc:
36      Write << "烫发的信息如下：" {perm}
37
38      Exit
39  State dyeProc:
40      Write << "染发的信息如下：" {dye}
41
42      Exit
43
44  State vipProc:
45      Write << "您的会员信息如下：" {vip}
46
47      Exit
48  State complainProc:
49      Write << "您的意见是我们改进的动力，请问您还有什么需要补充的吗？"
50
51      Read >> 20
52
53      Default thanks
54  State silenceProc:
55      Write << "您未输入任何信息，将在20秒后断开连接。"
56
57      Read >> 20
58
59      Rouse "发型" hairProc
60
61      Rouse "会员" vipProc
62
63      Rouse "投诉" complainProc
64
65      Silence thanks
66
67      Default defaultProc
68  State defaultProc:
```

```
    Write << "十分抱歉，未能明白您的意思，请您再说详细一些。"

    Read >> 20

    Rouse "发型" hairProc

    Rouse "会员" vipProc

    Rouse "投诉" complainProc

    Silence silenceProc

    Default defaultProc
State thanks:
    Write << "感谢您对Ocen理发店的支持！"

    Exit
```