

# 《基于领域特定语言的客服机器人的设计与实现》

课程 程序设计与实践

学院 计算机学院

班级 2022211307

姓名 于孟孟

学号 2022211260

[《基于领域特定语言的客服机器人的设计与实现》 1](#_Toc185842074)

[一 题目描述 3](#_Toc185842075)

[1.1 描述 3](#_Toc185842076)

[1.2 基本要求 3](#_Toc185842077)

[二 开发及测试环境 3](#_Toc185842078)

[三 程序介绍 4](#_Toc185842079)

[3.1 文件结构 4](#_Toc185842080)

[3.2 使用方法 4](#_Toc185842081)

[四 脚本文法 5](#_Toc185842082)

[4.1 脚本文件格式 5](#_Toc185842083)

[4.2 注释 5](#_Toc185842084)

[4.3 记录的定义和使用 5](#_Toc185842085)

[4.4 状态的定义 5](#_Toc185842086)

[五、概要设计 8](#_Toc185842087)

[5.1 流程图 8](#_Toc185842088)

[六、详细设计 8](#_Toc185842089)

[6.1 数据结构 8](#_Toc185842090)

[6.2 模块划分 13](#_Toc185842091)

[6.3 接口 18](#_Toc185842092)

[七 测试 21](#_Toc185842093)

[7.1 测试桩 21](#_Toc185842094)

[7.2 自动化测试 21](#_Toc185842095)

[7.2 测试结果 23](#_Toc185842096)

## 一 题目描述

### 描述

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。

### 1.2 基本要求

1. 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。

2. 程序输入输出形式不限，可以简化为纯命令行界面。

3. 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 开发及测试环境

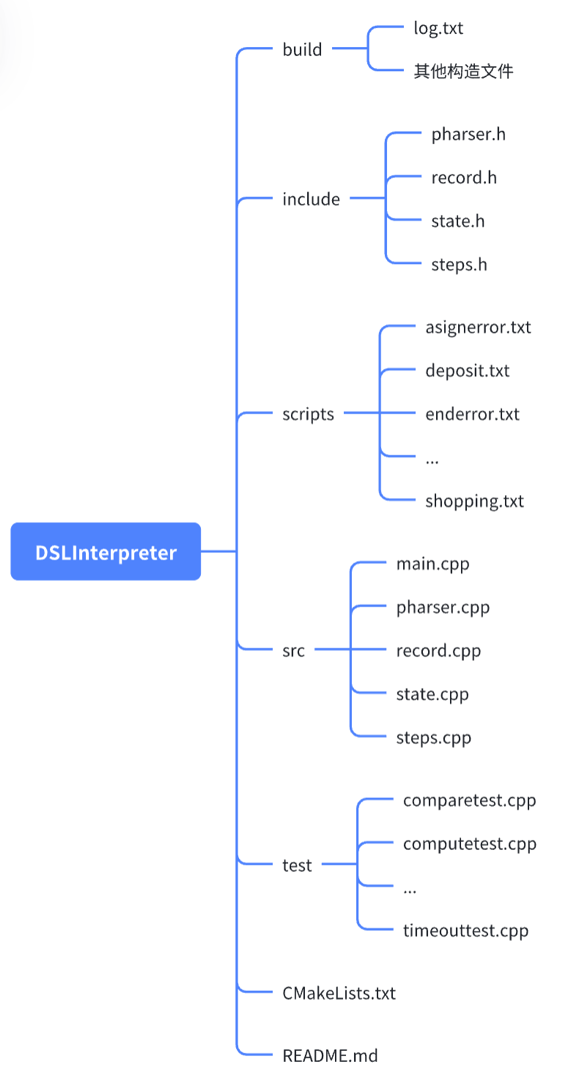
1. 系统 macos，Apple M2芯片，Sonoma 14.6.1
2. 开发语言：C++11
3. IDE：vscode
4. 构建工具：CMake
5. 测试工具：Googletest

图形用户界面, 文本, 应用程序

描述已自动生成

## 三 程序介绍

### 3.1 文件结构



### 3.2 使用方法

1. cd build

2. cmake ..

3. make

编译

4. ctest

运行自动化测试脚本，进行测试

5. ./DSLInterpreter

运行

## 四 脚本文法

### 4.1 脚本文件格式

1. 必须是.txt文件；

2. 所有空行都会被忽略；

3. 每一行语句的最左边不能有空格；

4. 对State的所有操作的定义必须以一个制表符开头，一个制表符之后不能有空格，必须是能匹配的操作；

5. 每条语句将被顺序执行

6. 一定要有State Start作为程序入口

### 4.2 注释

以 ’#’ 开头，不被包含在字符串里的 ‘#’ 及其之后的内容均被忽略

可以置于行首或行尾，但不能出现在一行代码中间

### 4.3 记录的定义和使用

1. 开头：let username = “<用户名>”；

2. 属性(另起一行，以let开头)： let <属性名> = “<字符串>” | <数字>；

3. 要求属性名的定义必须由字母或下划线开头，由字母、数字和下划线组成；

4. 只能用数字或字符串为属性赋值(数字只能是整数，字符串不能跨行)；

5. 对每一个用户必须规定确认username；其他属性应该紧跟在let username这一行的下面，不能被其他语句分开

6. 变量引用时需要 $<varialbename>，定义时则不需要

### 4.4 状态的定义

1. 定义状态：State <状态名>

状态名只能由字母组成(区分大小写)

2. 可以为状态添加操作

(1) 退出程序的指令：Exit

(2) 本状态下的输出：Output “<字符串>”

a. 字符串中可包含变量(以$开头)，但用户需要保证该变量已经被定义过

(3) 需要等待用户输入: Waitinput

一定要有与之对应的Timeout

可以紧跟其后对用户输入进行操作，如Compare, Compute等

(3) 状态转移：Jump <条件> <下一状态名> | Jump <下一状态名>

condition可有可无；

如果有的话，将用最近的用户输入来与condition进行比较(如果在此之前未定义用户输入，将用空串进行比较)；

如果相等，则状态跳转到nextState；

如果没有的话，直接跳转到nextState；

如果有条件跳转的话，一定要有Default；

(4) 默认匹配：Default “字符串(不能正确匹配时的输出信息)” <状态转移>

a. 字符串内部除了”之外都会被识别为字符串的内容，如果要在字符串内写入”可以用\”表示

b. 字符串不能跨行

c. 只能用英文

(5) 最长等待时间：Timeout <等待时间(s)> <状态转移>

(6) 对变量进行操作：

a. 让变量加上或减去用户的输入/用用户的输入给变量赋值: $<变量名> + | - | = (其中=只能用与对字符串的操作，+|-只能用与对数字类型的变量进行操作

b. 将变量与用户输入进行比较，比较成功则跳转状态:

Compare $<变量名> <|>|<=|>=|=|!= <下一状态名>

其中<|>|<=|>=只能对整型变量使用，=|!=可以对所有变量使用，=既可以对数字类型的变量也可以对字符串类型的变量进行操作，相当于赋值运算

3. 对操作的限制

(1) 对条件的定义：必须输入双引号包裹的字符串，用该字符串与用户输入进行匹配，匹配成功时会进行该跳转；

(2) 对Output的要求：输出给用户足够的使用说明，包括对用户输入的要求限制；

(3) 对于每一个状态，除Exit之外一定要在有限时间内离开这个状态，即必须满足一下条件之一；

a. 有退出程序指令Exit

b. 有Waitinput：一定要有Timeout，一定要有条件跳转(在有无条件跳转的时候可以没有)

c. 有条件跳转：一定要有Default，一定要有Waitinput

d. 有无条件跳转

(5) 对变量的操作一定和Input同时存在；

(6) 跳转至的状态一定是有定义的状态；

3. 对脚本的限制

(1) 一定要有Start状态作为程序的入口;

(2) 脚本文件必须是全英文，输入输出信息也应由英文构成，否则可能会无法识别；

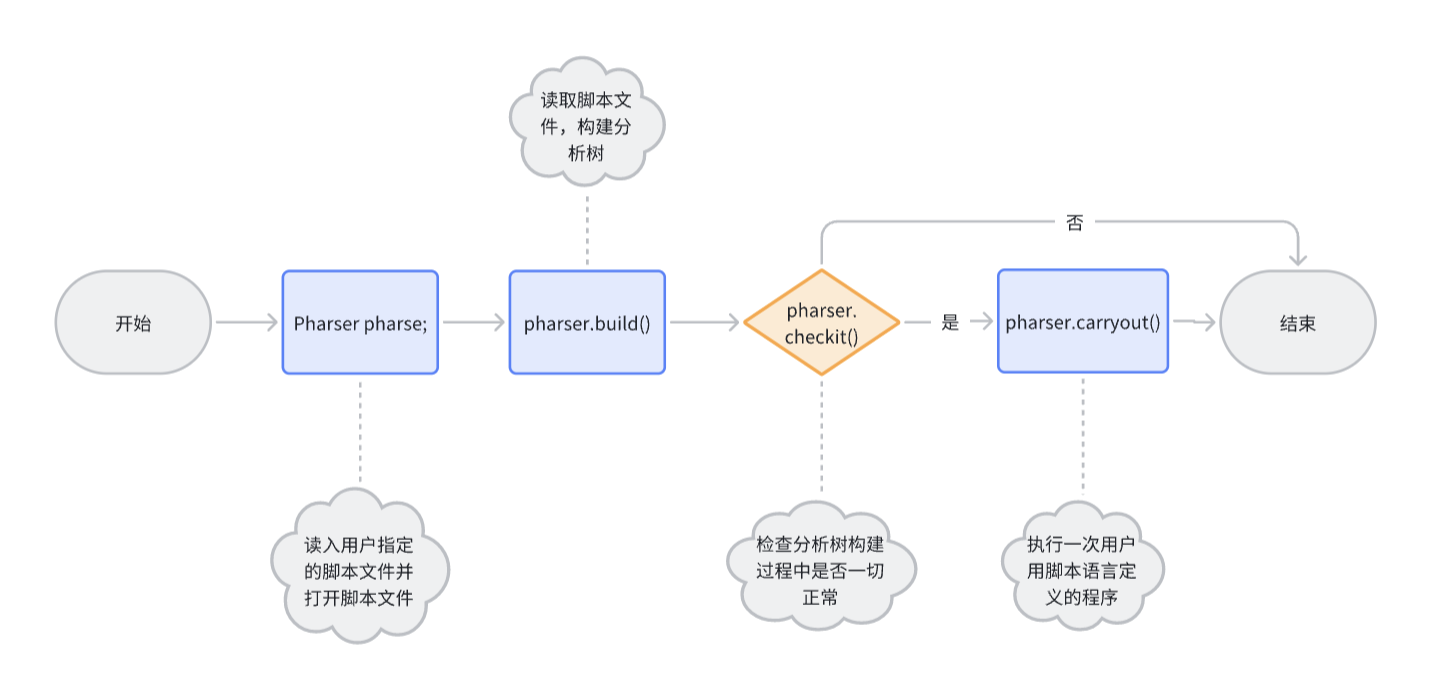
(3) 如果脚本中使用变量，则一定要有位置确定用户的username

4. 注意

(1) 对变量使用之前不需要预先定义，故需要书写脚本时保证书写正确的变量名；没有预先定义的变量将会被默认赋值为0；

## 五、概要设计

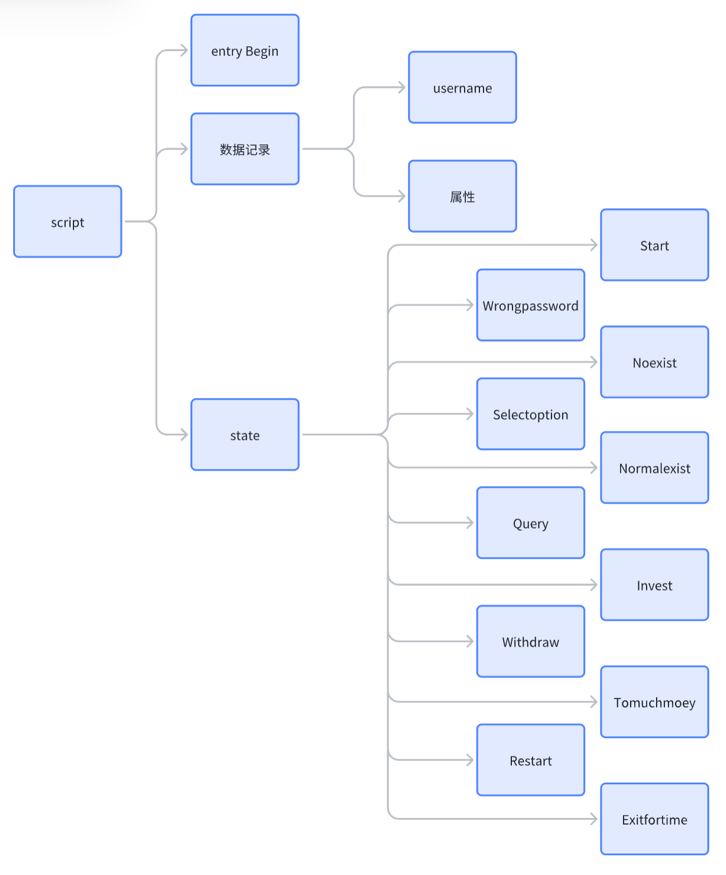
### 5.1 流程图



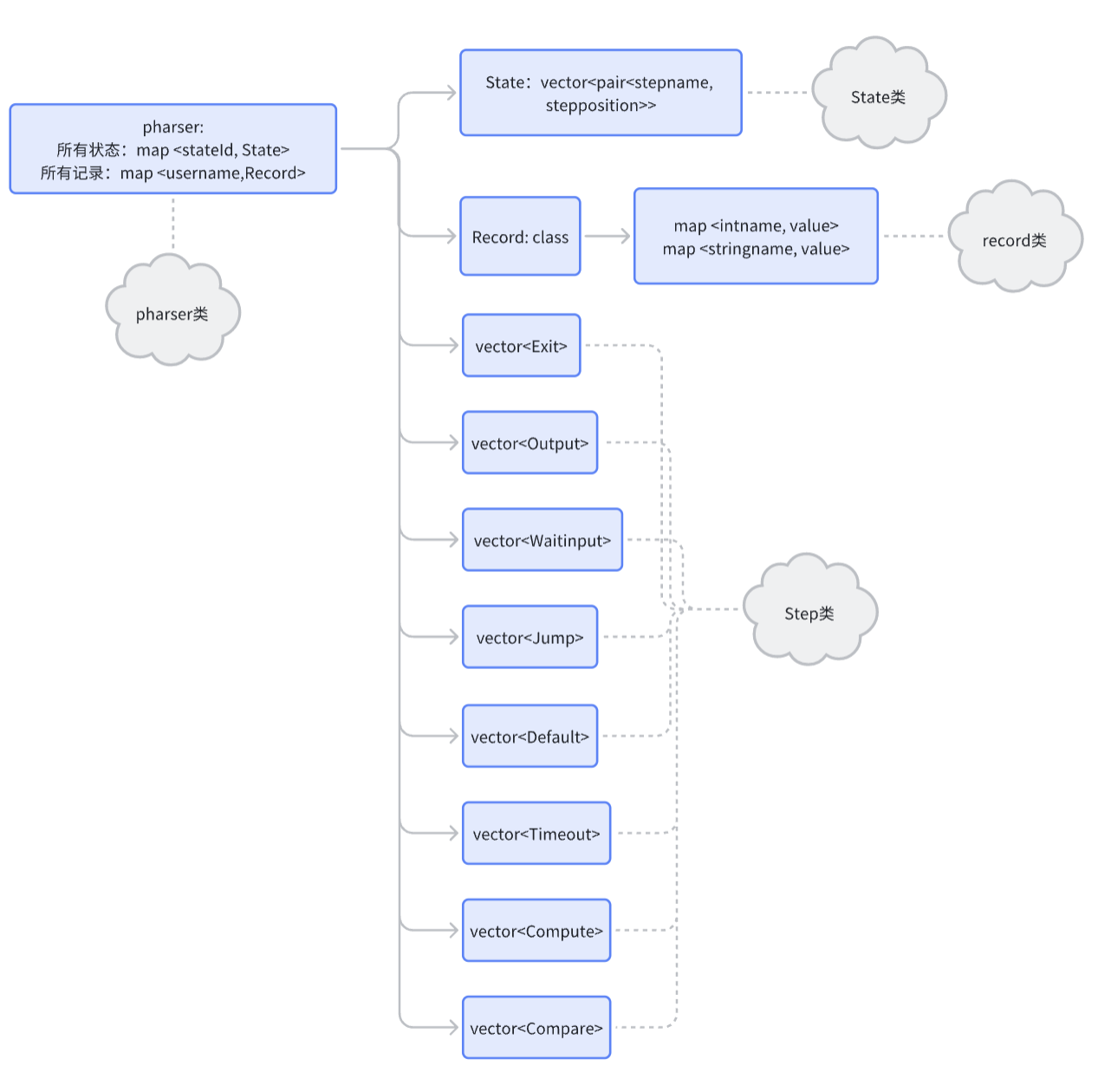
## 六、详细设计

### 6.1 数据结构

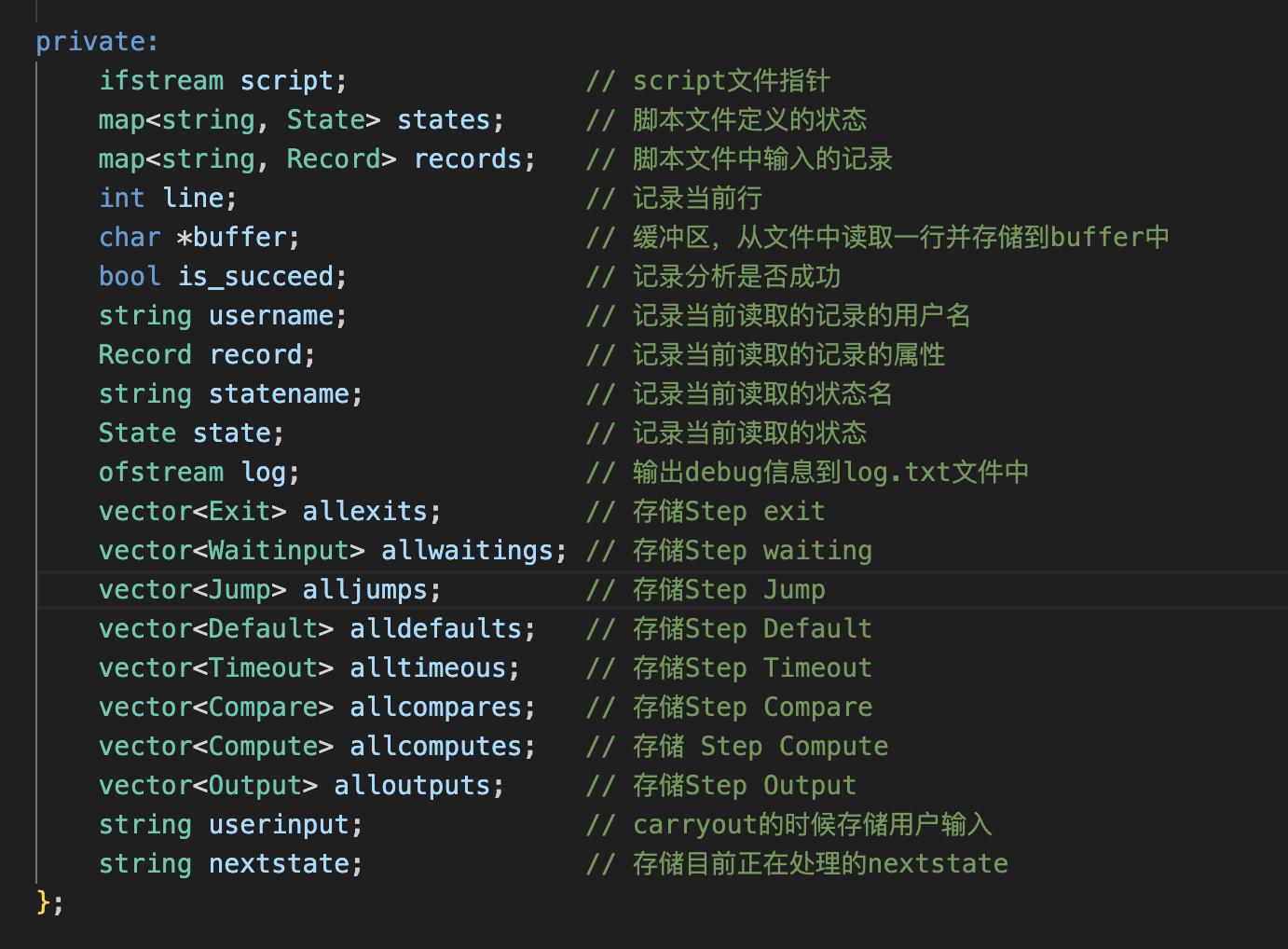
1. 将脚本文件抽象成语法树(以deposit.txt为例)



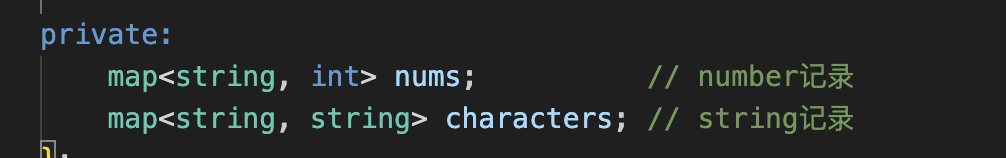
2. 语法树的数据结构



3. script类



4. record类



5. State类



6. Step类

(1) Output

文本

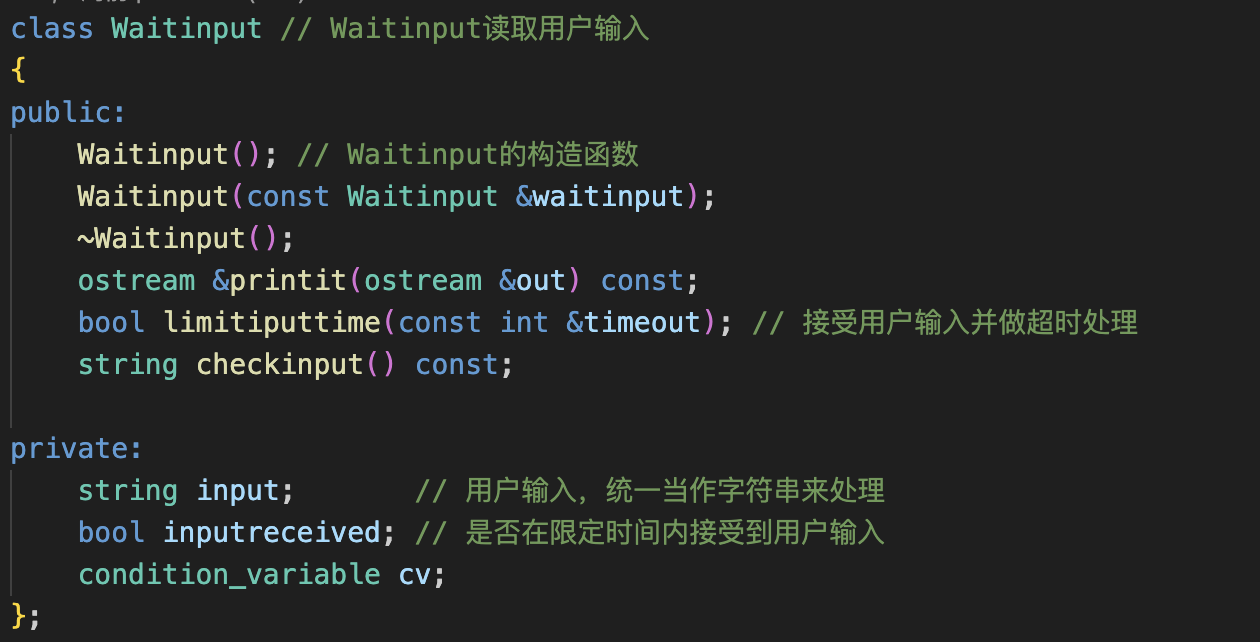
描述已自动生成

(2) Exit

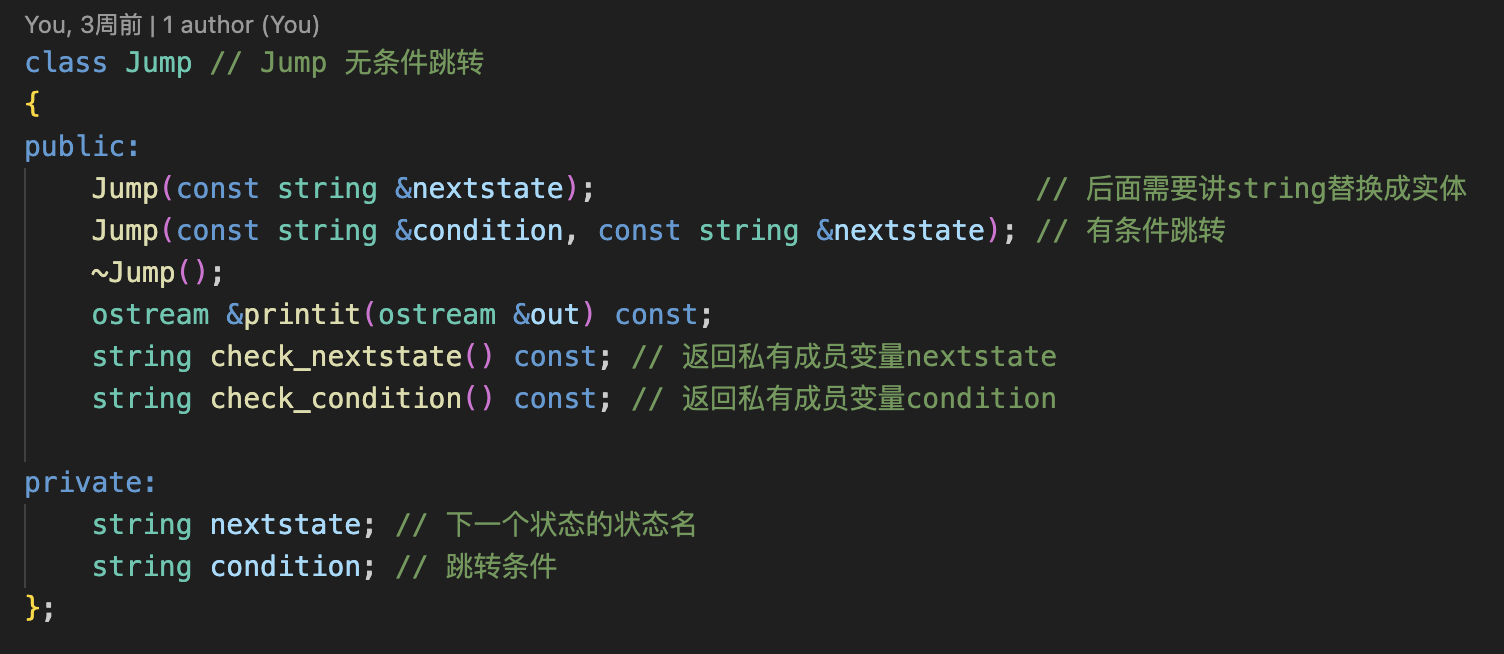
文本

描述已自动生成

(3) Waitinput



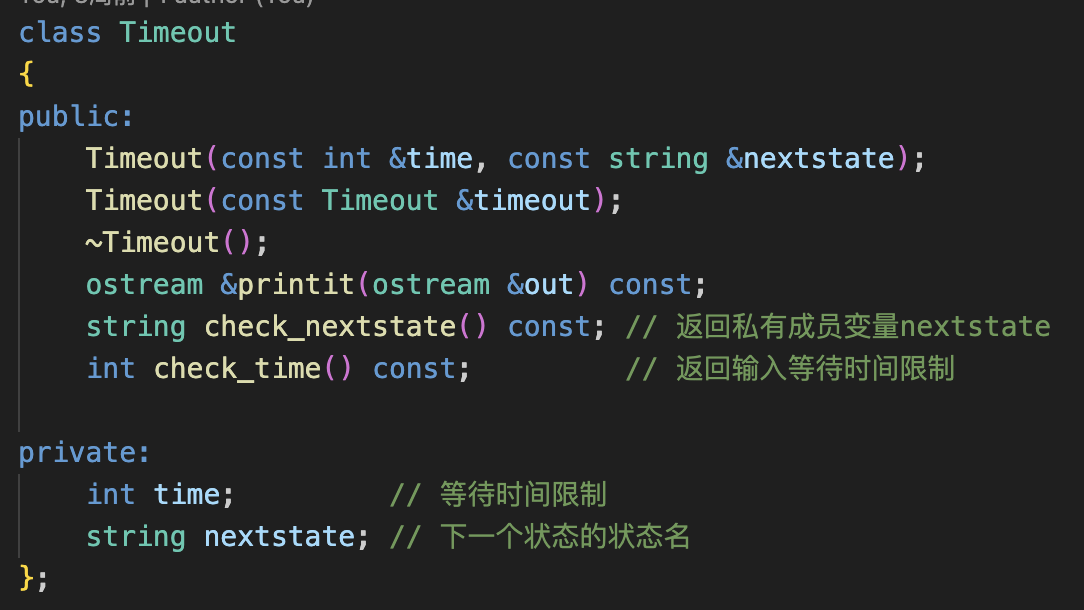
(4) Jump



(5) Default



(6) Timeout



(7) Compute



(8) Compare

文本

描述已自动生成

### 6.2 模块划分

1. Pharser pharser;

(1) 将正常标识位is\_succeed置为0；

(2) 将当前行数置为0；

(3) 为缓冲区申请内存；（在析构函数中释放）

(4) 读取用户输入的文件名，并打开文件，等待读取；(在析构函数中关闭文件)

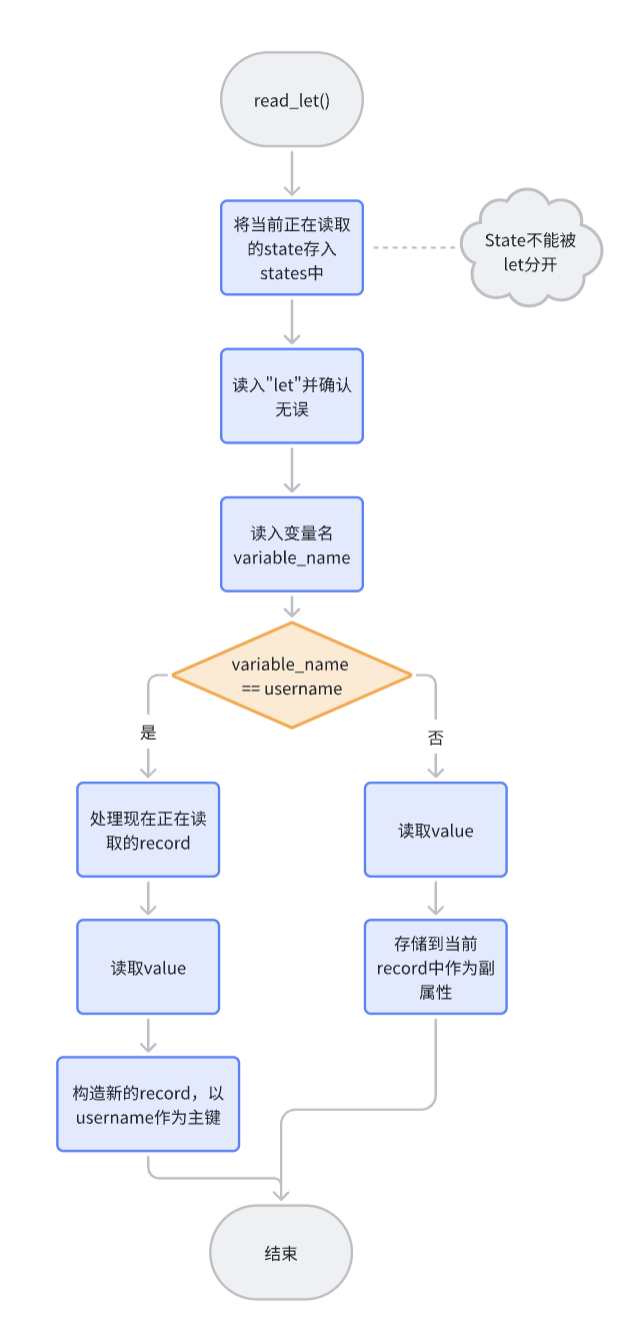
2. pharser.build();

(1) 整体思维导图

图示

描述已自动生成

(2) read\_let()



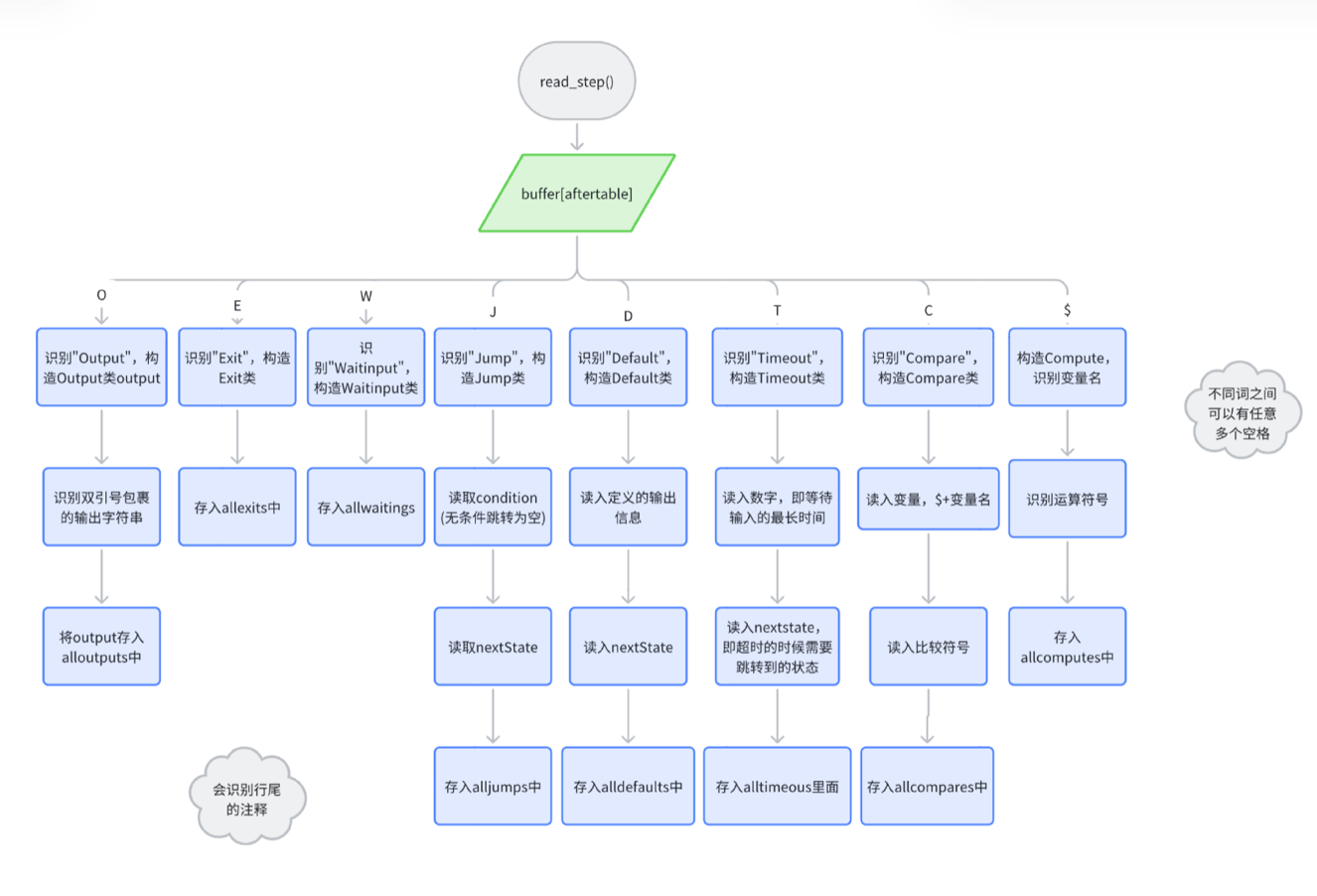
(3) read\_state()

a. 将处理的上一个state存入states中，清空state

b. 读入 “State” 并确认无误

c. 读入statename

3. read\_step()



4. pharser.carryout();

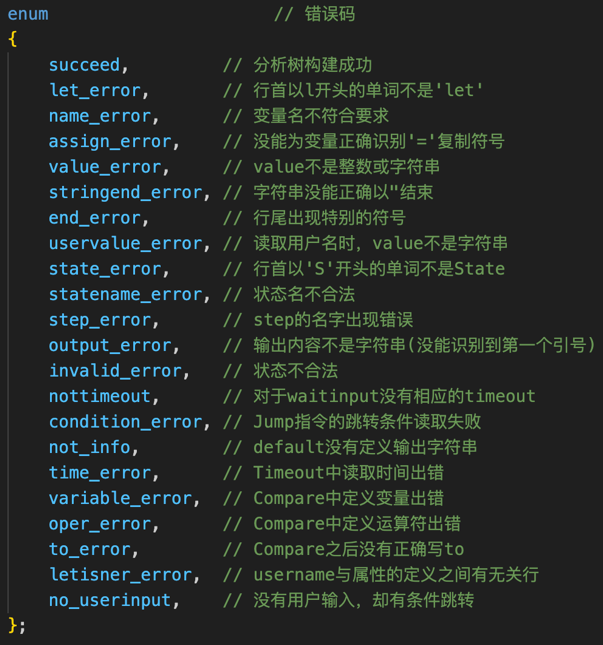
图示

描述已自动生成

5. 错误处理

(1) 分析树构造过程中错误处理

a. 语法错误



b. 逻辑错误

Step Waitintput之后一定要有Step Timeout;

每个State都一定要有Exit或Jump，使程序一定会在有限时间内离开这个状态；

let语句之间不能被State分割开；

一个State内部不能夹杂let语句；

如果有Jump <condition> nextstate，则一定要有Default / Jump nextstate

文本

低可信度描述已自动生成

(2) 分析树构造完成后的检查

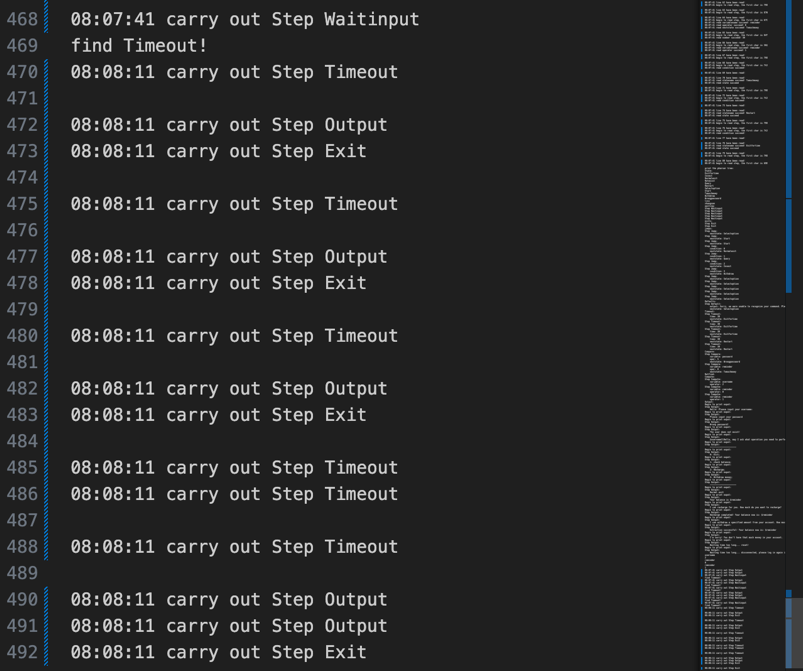
a. 检查所有的nextstate是否有定义；

b. 如果程序中用到变量操作，检查有没有输入username的地方；

c. 检查有没有State Start作为程序的入口；

6. log.txt与debug()

a. log.txt存储程序运行中即时输出的状态信息；每次运行程序都会删除log.txt中的上一次记录并重新写入；

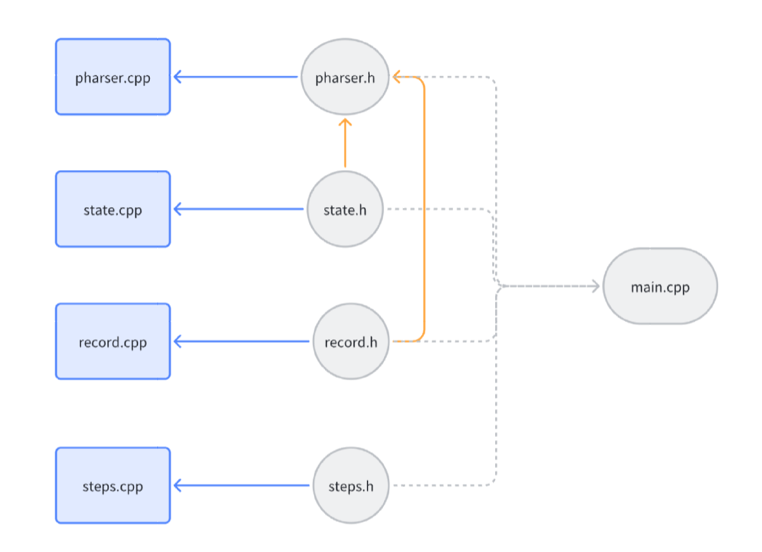


b. debug()在终端输出程序运行中遇到的严重错误

### 6.3 接口

1. 程序间接口

(1) 不同的.cpp文件通过.h文件的引用连接在一起；



如图，通过引用.h头文件，可以定义/使用类的public函数；

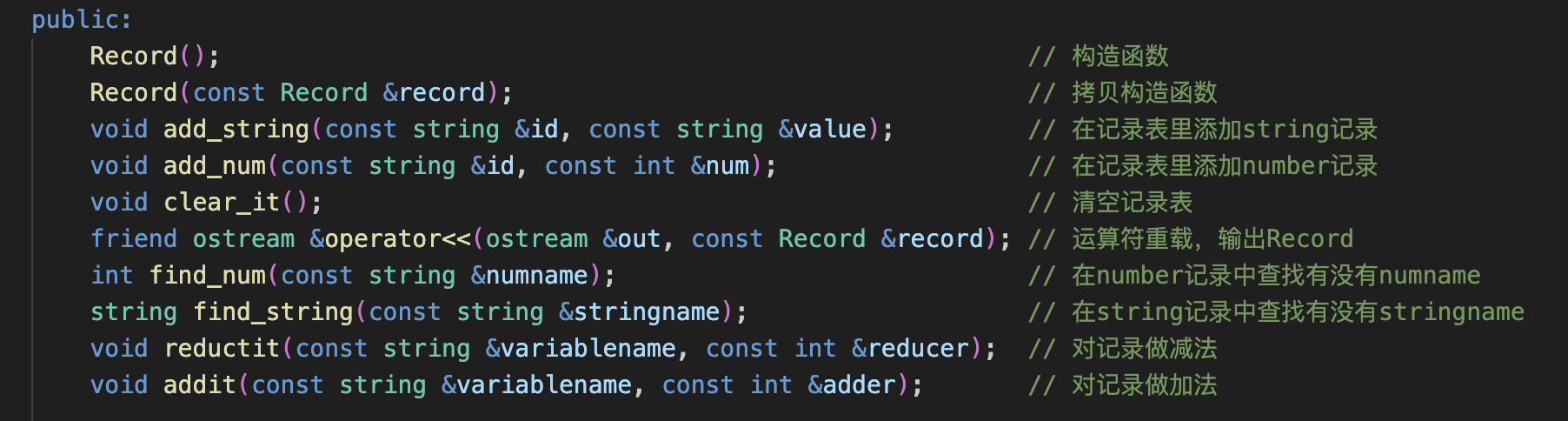
(2) 细节

a. Pharser

文本

描述已自动生成

b. Record



c. State



d. Steps

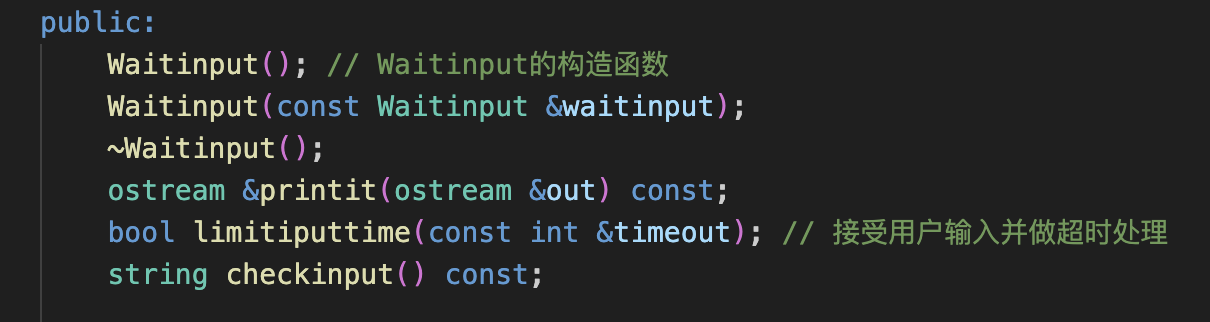
Output

Exit

文本

描述已自动生成

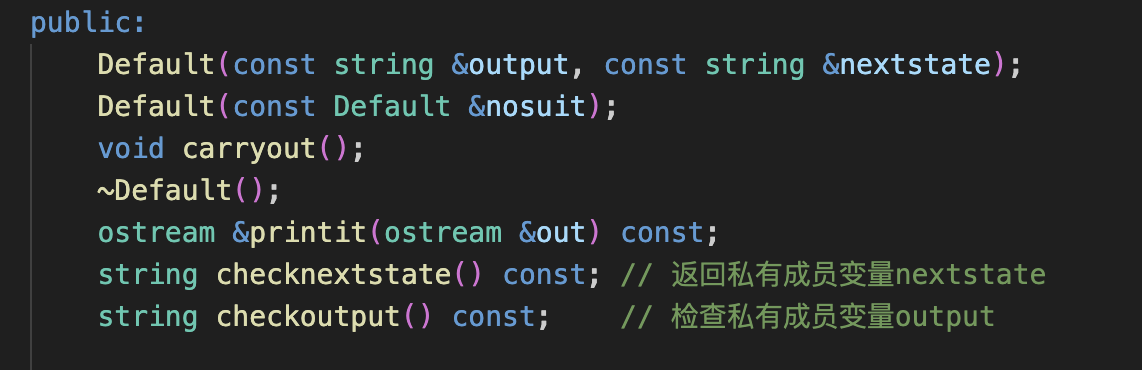
Waitinput



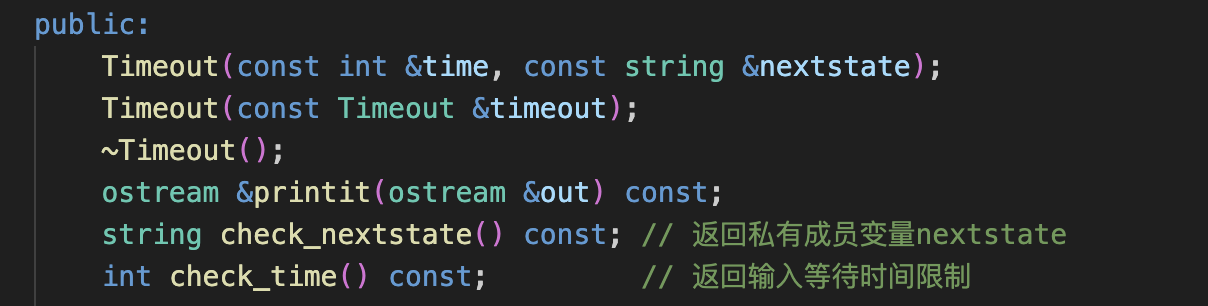
Jump



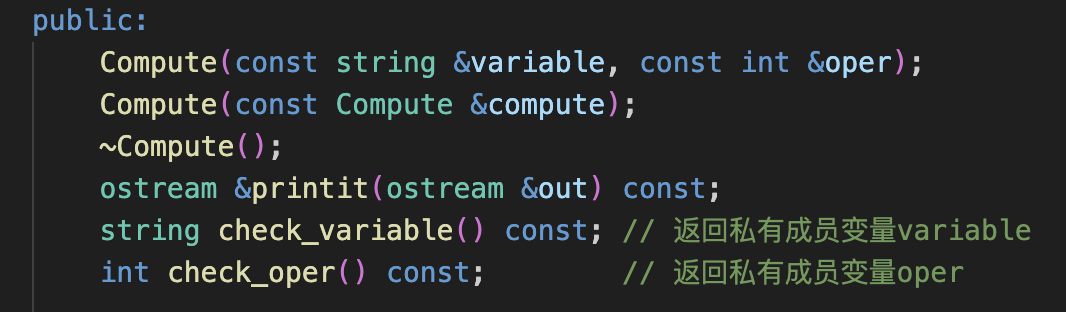
Default



Timeout



Compute



Compare



2. 人机接口

命令行界面输入输出；

脚本文件定义Step Output，作为程序输出给用户的提示；

脚本文件定义Step Waitinput，指示程序读取用户输入；

## 七 测试

### 7.1 测试桩

1. 用户输入与脚本文件定义的字符串匹配作为测试桩，代替自然语言处理程序；

2. 本地维护record，并对record做运算操作，作为测试桩，代替数据库存取操作；

3. 命令行输出，作为测试桩，代替媒体服务器的语音合成与语音播放；

4. 直接调用pharser的carryout()函数，作为测试桩，模拟用户输入；

### 7.2 自动化测试

1. 10个测试点

(1) Recordtest

测试Record中的构造函数，add\_string()，find\_string()，add\_num()，find\_num()，celar\_it()，addit()，reduceit()函数；

(2) Computetest

测试Compute中的构造函数，check\_variable()与check\_oper()函数；

(3) Comparetest

测试Comparetest中的构造函数，check\_variable()，check\_oper()与check\_nextstate()函数；

(4) Defaulttest

测试Default中的构造函数，checkoutput()， checknextstate()函数；

(5) Jumpttest

测试Jump中的两种构造函数，check\_nextstate()，check\_condition()函数；

(6) Outputtest

测试Output中的构造函数，checkoutput()函数；

(7) Timeouttest

测试Timeout中的构造函数，check\_time()函数，check\_nextstate()函数；

(8) Statetest

测试state中的构造函数，check\_valid()函数，putit\_valid()函数，check\_need()

函数，putit\_need()函数，reduceinput\_need()函数，check\_defaultneed()函数，adddefault\_need()函数，reducedefault\_need()函数，check\_conjumpneed()函数，addcon\_need()函数，reducecon\_need()函数，checksteps()函数，addstep()函数；

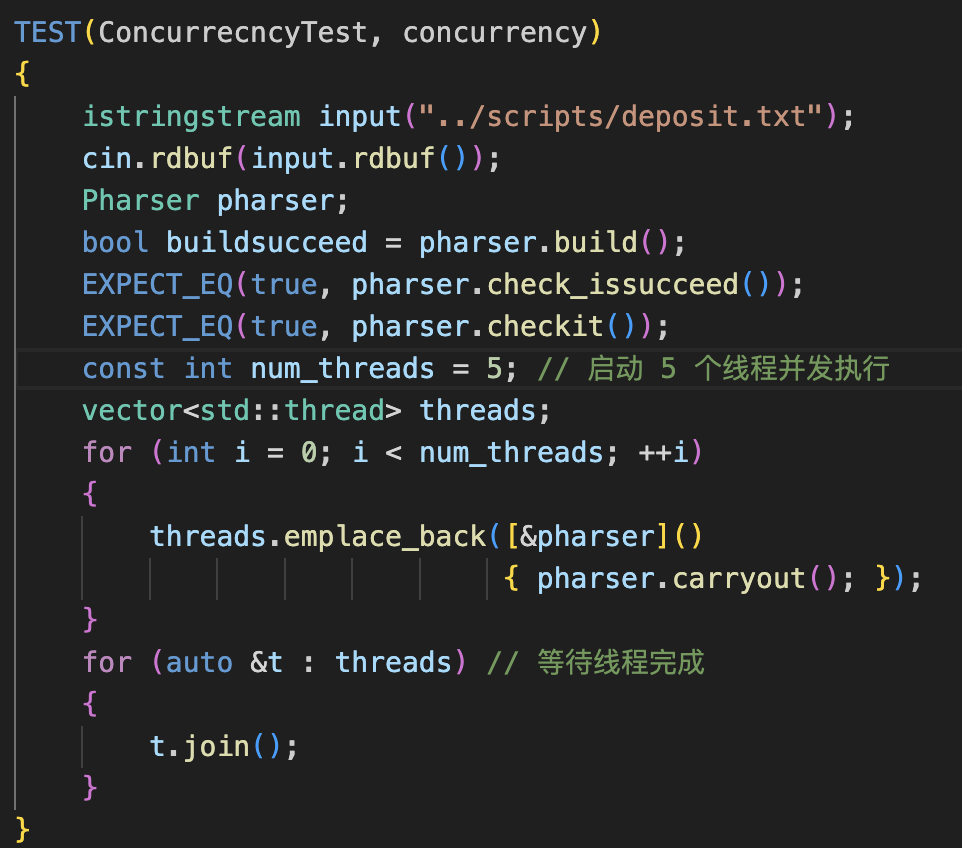
(9) Pharsertest

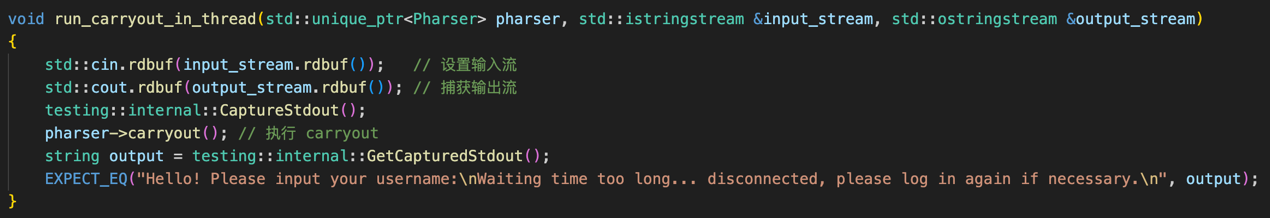
测试Pharser的is\_alpha(), is\_num(), build(), is\_succeed()函数；

在测试build()函数时，用scripts中的脚本文件样例作为输入，测试正确或出现不同错误时对应的不同输出；

(10) Concurrencytest

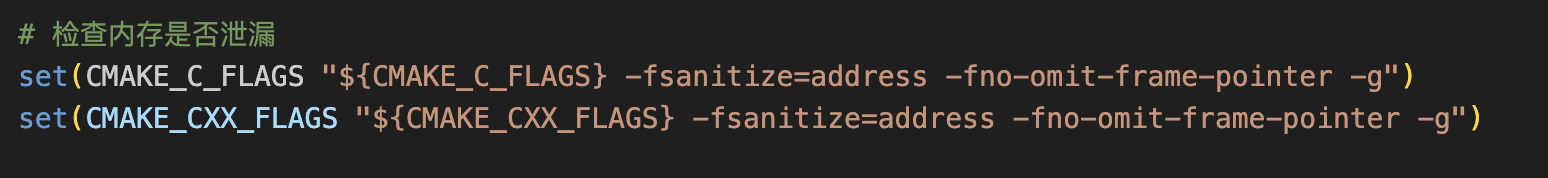
创建5个线程，并发执行carryout()函数，分别验证输出；





2. 内存管理测试

在CMakeLists.txt中添加如下内容，在程序编译和运行过程中会自动检查内存是否泄漏；如果泄漏的话会输出提示信息；



3. 并发测试

见测试点(10)；

### 7.2 测试结果

1. ctest运行结果

表格

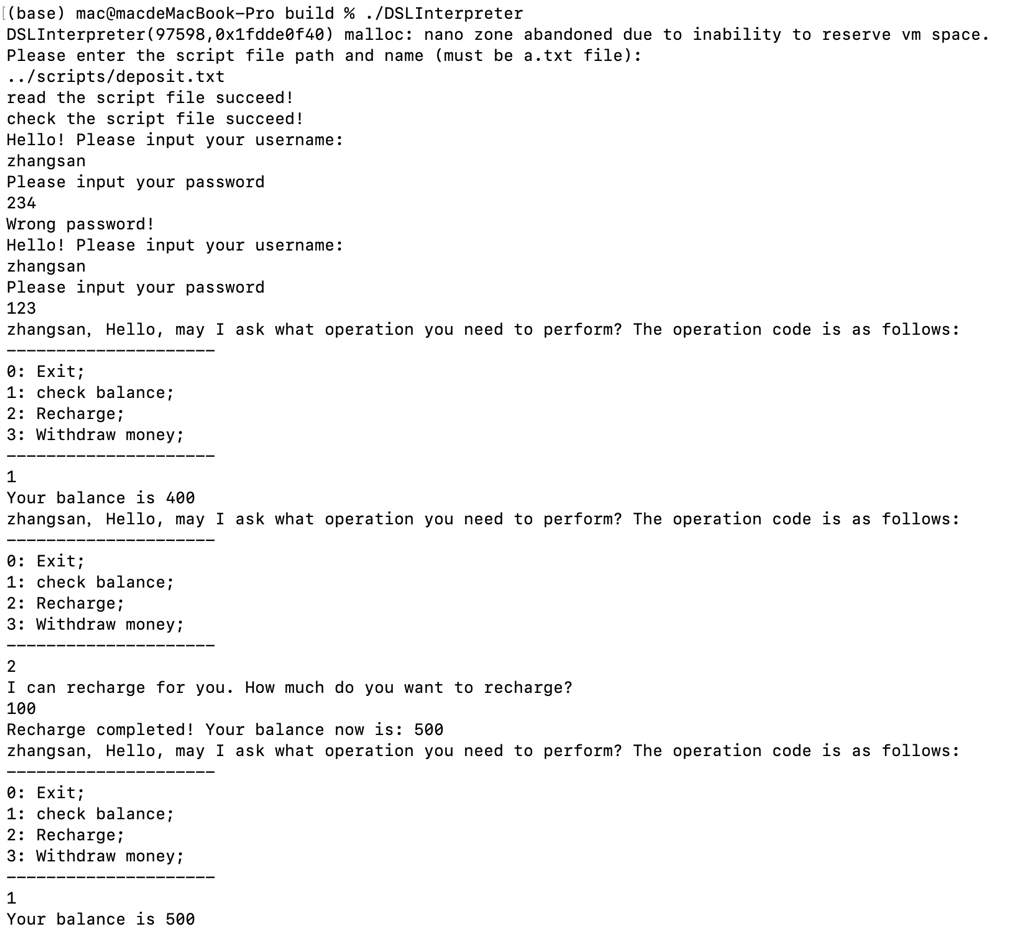
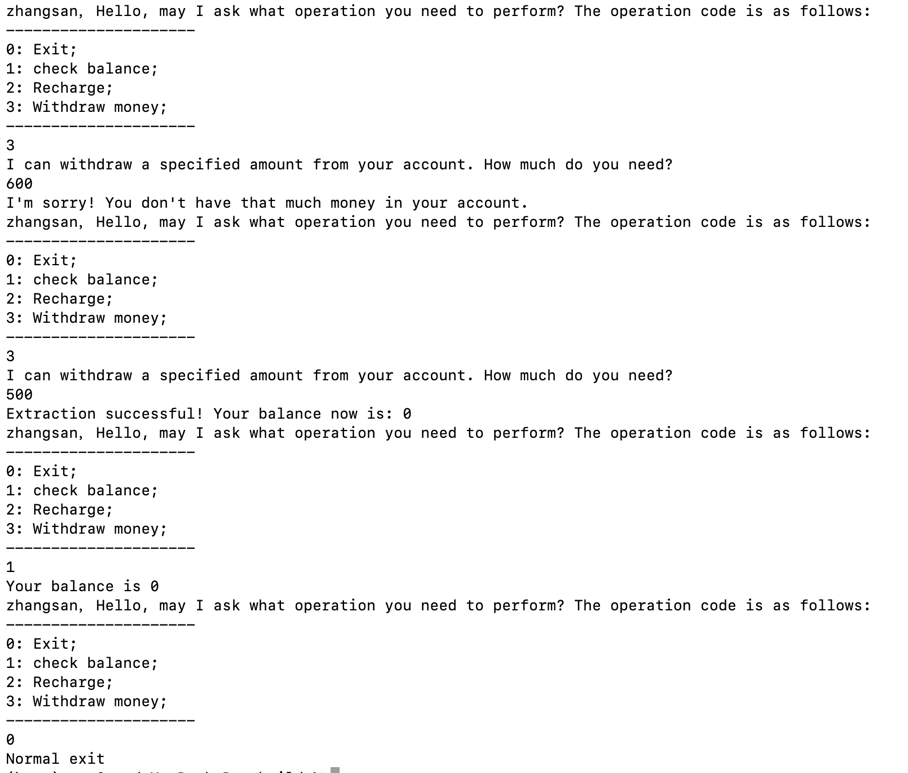
描述已自动生成

2. 实际运行

(1) deposit.txt

完成功能：输出提示信息、读入用户输入、读取已有record、对已有record做简单运算与简单比较；

应用场景：银行存取款机器人，需要用户输入用户名和密码，密码错误不予登入；可完成查余额、存钱、取钱的操作，取钱时会检查余额是否充足，余额不足的时候不能取钱；



(2) shopping.txt

完成功能：输出提示信息、读取用户输入、查看已有record；

应用场景：购物客服机器人，用户可查看自己已购物品的数目、投诉的操作；

