设计文档

程序设计背景

OOP 程序设计大作业

产品需求

以动态链接库提供算法核心 API

模块可复用，算法与实现分离

动态设定单词级别

多种记忆策略

单词测试功能

允许为单词添加例句

查询单词并记录查询历史

程序可以跨平台编译链接

设计思路

我们实现了 GUI 和控制台两种版本的背单词软件，我们大致的设计思路是先设计出各个模式 Pattern 类，对于指令进行处理，再设计出 Word 类和 DataBase 类提供核心算法执行指令。控制台版本中的 Processor 类负责流程的控制，Shell 类负责与用户的交互，而这一点在 GUI 中是通过将各个设计良好的模块嵌入 QT 的管理器中实现的。

功能实现

本词典（ILoveWords）主要实现了以下几个类：

Shell 类：流程控制类，用于和用户的交互和指令的读取。

Processor 类: 指令处理类，策略方法联结 DataBase 和 Pattern，实现模式与数据库之间的交互。切换模式，切换数据库。实现 DataBase 和 Pattern 的组合。

SingleWord 类：单词类，具有释义、拼写、难度、例句集等成员，作为 DataBase 的成员，由 DataBase 提供算法。

DataBase 类：词库类，提供针对单词的核心算法接口。可以保存历史记录。

Pattern 类及其子类 MainPattern、Quiz、Query、Recite 类：模式类，以继承方式实现了不同的模式，提供了各自独特的功能。

Test 类及其子类：用于单词测试。

Log 类：用于 Query 类信息记录

Func 类：一些基础函数。

主要接口及私有函数：

LoadCmd()：

Shell 类实现，用于读入命令

RunCmd()：

流程控制接口，Shell、Processor、Pattern 及其子类中实现，对于读入的命令进行处理，调用不同的模式，不同的算法实现。

其中 LoadCmd()和 RunCmd()相互调用实现流程的可持续性

Shell 类：

LogIn() / SignUp(): 用户登录

LogOut(): 用户切换

本类处理退出和切换用户指令，其余指令传给 Processor 类

Processor 类：

GotoPattern()：切换模式

ChangeDict(): 切换词典

PrevPattern(): 返回上一模式

ShowCmd(): 显示指令集

IncreaseSentence(): 增加例句

本类处理模式和数据库的选择指令，其余指令传给 Pattern 类

Pattern 类及其子类:

Start(): 输出开始提示信息

Quit(): 退出该模式所需操作

Set 等接口：修改 dictname_、username_等成员

MainPattern 类:

RunCmd():额外支持输入单个数字选择模式

Quiz 类：

Input(): 获取测试数量

Create(): 调用 Test 类生成测试

Info(): 输出测试结果

所用的 Test 类：

子类 Test1（给出英文选择题问中文）、Test2（给出中文选择题问英文）、Test3（给出中文问单词拼写）

ShowAns(): 输出题目答案

CreateProb(): 生成问题

Print(): 输出问题

CheckResponse()：判断正误

Query 类:

RunFileQuery(): 判断生词统计指令，调用私有函数 CalcAndNote()处理该文件

CalcAndNote(): 算法，统计生词,

RunWordQuery(): 判断查单词指令，调用 DataBase 中的算法给出单词查询结果

SetHIstoryNum(): 控制历史记录的每次输出条数

RunDelCmd(): 删除历史记录

ShowHistoryByStep(): 逐条显示历史记录

ShowLog(): 显示历史记录

QuitLog():退出历史记录

所用的 Log 类：提供 Query 类所需的与记录有关的算法

Recite 类：

Init(): 背单词过程初始化

Step0(): 开始背某个单词的操作

StepYes(): 用户输入认识该单词后操作

StepNo(): 用户输入不认识该单词后操作

Next(): 进入下一个单词

Undo(): 用户撤销背过标记后的操作

DataBase 类:

提供核心算法接口，由 Processor 调用供 Pattern 类解决问题

使用了 C libaray 中的 struct tm 来获取系统时间，使得操作具有时效性

SameDate(): 判断系统时间是否是同一天

cmp(): 比较两个单词应该出现的时间次序，用于排序

GetSimilar()：核心算法，判断两个单词的相似程度，用于单词测试中给出近似选项，在单词查询中支持模糊查询。

ReadHistory(): 读取历史记录，并将其有效数据保存到 DataBase 的数据成员中，用于处理

ReadDict(): 读取词典，获取当前数据库

SetNewWordPerDay(): 控制每天背单词的数量

CheckFinish(): 判断背单词任务是否完成

CheckTodayFinish(): 判断今天的背单词任务是否完成

GetReciteWord(): 生成一个要背的单词

SetWordPeriod(): 记忆策略，控制每个单词需要背的次数及出现的时间周期

AddWord(): 允许添加新的单词

Undo(): 背完一个单词后维护数据结构

ChangeFeature(): 改变单词难度

CheckWord(): 判断单词是否正确

GetWord(): 从数据库中找出这个单词

GetReciteNum(): 获取已背过单词数量

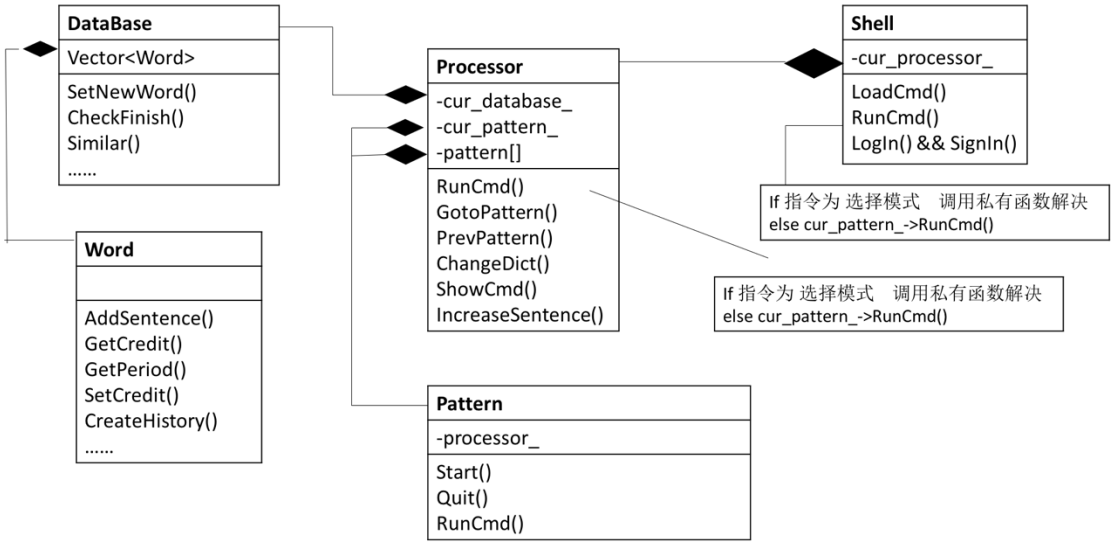
RandWord(): 随机选取数据库中的一个单词

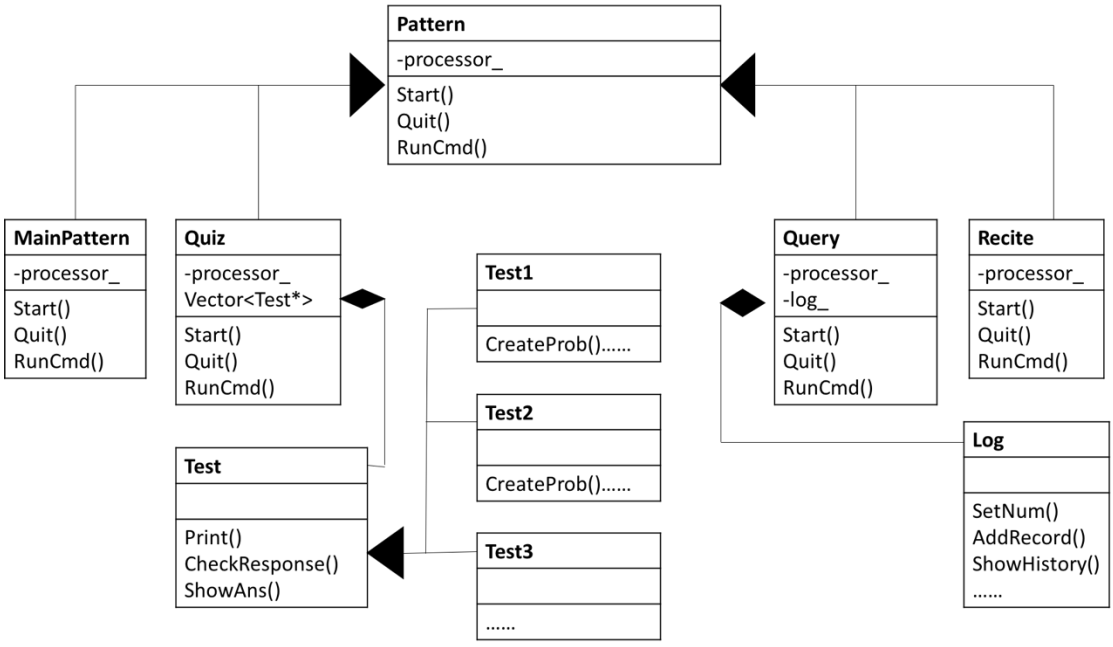
RandReciteWord(): 在要背诵的单词中随机选取一个

Similar(): 返回和该单词最相似的 4 个单词

Check()：输出数据库内容

附：设计模式的 UML 图


DataBase
Vector<Word>
SetNewWord()
CheckFinish()
Similar()
.....
Word
AddSentence()
GetCredit()
GetPeriod()
SetCredit()
CreateHistory()
.....
Processor
-cur_database_
-cur_pattern_
-pattern[]
RunCmd()
GotoPattern()
PrevPattern()
ChangeDict()
ShowCmd()
IncreaseSentence()
Shell
-cur_processor_
LoadCmd()
RunCmd()
LogIn() && SignIn()
If 指令为 选择模式 调用私有函数解决
else cur_pattern_->RunCmd()
If 指令为 选择模式 调用私有函数解决
else cur_pattern_->RunCmd()
Pattern
-processor_
Start()
Quit()
RunCmd()


Pattern
-processor_
Start()
Quit()
RunCmd()
MainPattern
-processor_
Start()
Quit()
RunCmd()
Quiz
-processor_
Vector<Test*>
Start()
Quit()
RunCmd()
Test
Print()
CheckResponse()
ShowAns()
Test1
CreateProb().....
Test2
CreateProb().....
Test3
.....
Query
-processor_
-log_
Start()
Quit()
RunCmd()
Recite
-processor_
Start()
Quit()
RunCmd()
Log
SetNum()
AddRecord()
ShowHistory()
.....