81-作业报告

程序功能介绍

该程序为一个编程类解谜游戏，玩法机制是使用一个图灵完备的字符串替换语言"A = B"来完成不同的任务，在不断地探索语言机制的巧妙之处的同时逐步揭示华茂公司和刹那之间神秘的真相。

游戏的模式分为两种。一种是带有剧情的正常模式；另一种是自定义模式。

在正常模式之中，玩家在右下角的代码编辑区中进行代码的编写，同时可以使用中间的暂停，加减速等功能按钮进行 debug。随着剧情的不断深入，"A = B"语言的更多语法机制也会逐渐揭露，例如"(return)"关键字等，逐步展示"A = B"的精巧的内核。

自定义模式的关卡可以由玩家通过关卡编辑器进行设计和保存，玩家可以相互之间通过关卡设计的方式对语言机制进行更加深层的交流。

在设置界面可以进行关卡的一键解锁和一键锁定（风灵月影既视感）。

项目各模块和类设计细节

Mainwindow 类：qt 项目创建自带的类，用 stackedWidget 连接开始界面、关卡选择界面、模式选择界面、设置界面等，类内有一个 Gamewindow 类的指针，Mainwindow 和 Gamewindow 的转换通过 widget 的 show 和 hide 实现。

Stage 类：包括关卡的名字、标准答案、样例输入输出等信息，类内有存储和读取自定义关卡的函数。Stage 具体的剧情存储在./plot.txt 下，需要时读取。

Gamewindow 类：实现游戏中关卡信息的呈现、提供代码输入的 IDE 界面、输出是否通过和具体错误的 case。用 isEdit 变量判断模式。在编辑模式下，玩家可自行编写题目描述与程序运行的等效代码并通过 upload 按钮上载，以实现关卡自定义；而游玩模式下题目描述锁定，玩家可在代码编辑区输入代码并验证测试。本类包含关卡编辑器，自定义关卡及故事模式中游戏部分的界面。

System 类：包含一个多线程模块来实现对代码输入、编译和执行的处理以及 debug 和输出有关的功能。主体函数是 run，使用 signal-slot 机制实现 system 和 gamewindow 上控件的信息交换。

Code 类：代码块类，包含一个完整代码块的原始字符串信息和代码编译、执行、输出等函数

功能实现：

存档和剧情读取：全部使用 txt 明文存储，使用 QFile 进行读写，保存在相应的 Stafe 类中

"A = B" 语法和代码解释：先对输入字符串进行子串匹配，后总代码块转化为 Code 类对象，其中包含若干 command 类对象，一个 command 类对象即单行代码。然后循环执行 Code，将玩家的 code 和标准 answerCode 对同一随机生成字符串的执行得到的 Output 进行比较，得到一个 case 的结果。

小组成员分工情况

罗君彦（队长）：System 类实现+debug 测试。统筹小组工作，组织交流，分配任务。

朱俊霖：Mainwindow 类和 Stage 类的实现。Mainwindow 类和其他界面的协调。

闫昕：main 实现、Gamewindow 类设计和实现、剧情设计。关卡设计。

项目总结与反思

这个项目是对 Steam 上同名游戏的复刻。这个游戏的内核并不复杂，无非是设计了一种处理字符串的语言。当然，我们因为是用 c++开发的，也并没有涉及到更底层的语言，所以运行速度上也无非就是 c++处理字符串的速度，并不具备更好的性能。但是通过一层简单的包装，我们可以发现编程语言形式上的灵活性，就像自然语言一样，同一种意思，同一个目的，我们当然可以用不同的词组、句式来表达。对于编程与语言这一点特性的兴趣，驱动着我们完成这个项目。另一方面，对于简单的统计字符串中字符个数之类的任务，在 A2B 的设计规则中，要绕很大一圈来实现，也是一种对于编程思维的训练，可能有利于我们更好理解字符串操作的本质。

就开发中提出功能到实现功能的过程来看，因为不熟悉 qt 的语法和功能，我们往往是根据功能需求求助于搜索引擎或者 chatgpt,而不是直接阅读 qt 的使用手册。这种按需学习的方式有利有弊，利在于快速的上手开发实践，弊在于缺少对 qt 这门语言整体设计上的认知，因此对于很多底层设计上的 bug 只知其一不知其二，只能采用迂回的方式避免，比如反思 1 所提到的 QStackedWidget 和在 Mainwindow 里内置其他 widget

指针的区别。通过这一现象，我想指出的是，以应用为目的开发和以学习技术为目的的开发应该是有侧重的。

项目开发过程中一个比较重要的体验是开发者和用户的交互。比如说 gamewindow 中测试界面的呈现，根据用户的实际需要，设计了加速和减速两个按钮。还有各个按钮的分布与布局需要符合大众的使用习惯（通过设置字体大小、加粗、斜体等方式）以及对于按钮的说明介绍等。在这一过程中，我们不像在 openjudge 上面只考虑算法的通过与否，更多地关注使用者的需求，这种身份与视角的转换是很有必要的。

反思：

1. 没有深入理解 c++指针和内存空间对应的关系。一开始我们在 Mainwindow 类储存的 Gamewindow 指针只是普通指针，没有用智能指针或者是 QPointer 之类，导致如果将 gamewindow 放入 stackedWidget 就可能导致多个指针指向同一片内存空间，在 mainwindow 析构时发生类似多次 delete 一片内存空间的报错。
2. 没有调动好团队积极性。在最开始进行项目分工时对于最终成果没有进行很完备的讨论，在代码编写的过程中也缺乏对项目的乐趣性的沟通，这可能是由于最开始的分工对于实现过于乐观，未能及时进行调整导致。在开发过程中没有合理设计分工，基本上是一个人开发完接下一个人继续开发，没有合理地进行任务的合理分割。团队之间缺乏统一的交流时间，导致交流比较混乱，也不好进行进度对接。当时进行的阶段性目标设计过于笼统以至于无法很好地实践，后期也没有进一步细化阶段性目标，这可能也是我（罗君彦）作为队长缺乏决断力导致。反思：在初期项目分工定好一个大的方向后先进行一个基本模块的细致分工，确定一个 checkpoint 进行试验开发，此时重点放在对项目流程和交流方式进行磨合上。checkpoint 达到后，根据已有经验再继续向下调整原有的项目分工。同时在继续进行开发时设立多个 checkpoint，在达到 checkpoint 后再继续进行分工，这样可以及时弥补因为缺乏实际经验导致的乐观估计，也有更多机会来进行团队交流。
3. 缺乏文档和注释。长期项目中由于开发时间长，一个代码不仅需要给其他人进行阅读，还需要给未来的自己进行阅读，所以保持代码的可读性十分重要。平时进行代码编写的时候一般只聚焦于当前的 checkpoint 或者是当下的 bug，而没有考虑整体代码的低耦合和易读性，同时解决了一个 bug 还可能因为变量作用域比较大，导致另外一个地方出现新的 bug。在这方面，我认为我们需要在开发时对于可能被交叉使用的变量和函数提供一个合适的接口说明，同时在作出改动时相互告知，避免可能造成的冲突。
4. 没有很好地发挥 Github 版本控制的优势。项目开发前期对于 Git 的运用比较简单，只是简单的向 Github 上添加文件和下载，有时候会莫名其妙地覆盖了别人的修改或者说某个文件依然停留在以前的版本。这个的主要问题在于我们对于 git 的原理没有系统理解。一个重要的经验是，在开发前应该组织对于开发工具的学习，保证小组内用同一种合理的方式来提交代码。另外，其实通过提交时创建新的 branch 然后 merge 也能避免代码的冲突，但是 merge 时的取舍也需要一个对于代码整体结构有把握的人来决定。不会用 Git 也是反思 2 中

不能并行开发的原因之一。