福州大学

软件学院程序设计实践报告

题　目：C 语言程序相似性检测系统

姓　　名：方林升

学　　号：102300311

学　　院：计算机与大数据学院

专　　业：软件工程

年　　级：2023 级

教　　师：王灿辉

2024 年 06 月 16 日

目　录

第 1 章 程序设计报告

一、相关算法概述

此次的任务是完成一个 C 语言代码相似度检测系统。对于一个代码相似度检测系统，需要考虑的方面有很多。首先，对于代码的抄袭情况，不同的学者有不同的划分方法。一种较为普遍认可的划分方法[1]，将代码抄袭的情况分为四种类型，如表 1 所示：

表 1

类型	概述
第一型	除了空白部分、代码注释和布局，其他完全相同。
第二型	除了标识符名称、类型、布局和注释等变化，代码结构和语法完全相同。
第三型	在第二型的基础上，进行添加，删除或者修改语句。
第四型	代码功能相同，但完全采用不同的语法结构实现。

同时，经过多年的研究发展，产生了多种的代码查重技术。概况来说，可以将主要的查重技术分为五个类型[2]：

1. **基于文本匹配的算法。**这一类算法主要是将代码去除其中的空白和注释后，转化为字符串的方式，设置阈值进行比较。这一比较方法主要适用于第一型的代码相似度检测，且不能自动定位相似的代码片段，具有一定的局限性；
2. **基于 token 的相似度检测算法。**这一类算法主要是通过编程语言的词法规则，对代码进行提取 token，将代码转化为 token 序列的表现形式。之后，再进一步对 token 序列进行匹配和比较，这一过程一般是通过字符串搜索算法或者是后缀树算法来实现的。这一算法比较适用于第一型和第二型以及少量的第三型情况；
3. **基于抽象语法树的检测算法。**即对程序通过语法分析，建立抽象语法树（Abstract Synax Tree，即 AST），然后通过树匹配技术进行比较。在这一抽象的过程中，会忽略空格，注释等部分的差异，因此能较好适用于第一型和第二型的检测。另外由于它度量的是语法结构，因此对于第三型的检测也具有较好效果，但是对于第四型的检测并不适用。后人对这一算法有进一步的优化，使其适用于更多更广泛的检测情况；[3]
4. **基于程序依赖图和控制流图的检测算法。**对程序进行分析，建立程序依赖图和控制流图等，来描述代码内的逻辑流程，并做进一步比较。这一过程中同样只保留了代码的逻辑信息，因此可以检测出代码结构存储的相似。但是由于图搜索的耗时较大，计算效能较低；
5. **基于深度学习的检测算法。**深度学习已被广泛的应用于自然语言处理 等多个领域，并且达到了较为理想的效果。代码方面的分析，与自然语言处理的问题非常

相似。因此有许多学者尝试运用深度学习等技术解决代码相似检测的问题并取得了一些成果。[4]

在此次任务中，笔者主要采用基于文本匹配算法中的编辑距离算法和简化的抽象语法树算法进行代码相似度检测，并给出一个 0~100 之间的数值代表程序的相似度。

二、问题的简化与假设

由于笔者的能力有限，为了简化此次的实践任务，我们做出如下假设：

- 对于笔者最终程序的设计目标，主要是为了应对第一型，第二型和一部分第三型的代码相似检测。对于第四型的检测，由于难度较大，暂时不做考虑；
- 为了方便对代码的预处理，假设要检测的代码不存在语法格式上的错误，且不存在一行多个语句的情况；
- 为了方便编辑距离算法的处理，假设代码一行中的字符个数不超过 500 个；
- 为了方便简易语法树的构建，假设对于所有可以省略“{}”的部分，代码中均有加上“{}”。

另外，对于代码的相似度评价，学术界有诸如结合特征矩阵[5]，特征向量，余弦分析等算法。为简化程序编写和数据的处理，笔者此次直接采用相似部分占总代码部分的比值作为相似度进行处理。

三、对代码文件的预处理

C 语言的语法格式自由，代码具有多样性。为了方便对代码的比较检测，在设计相似度检测系统时，一般都需要对源代码文件做一定的预处理。[6]在此次任务中，笔者会对要进行比较的代码文件进行注释，空行和在双引号内的字符串文本内容的去除。程序在与 exe 文件相同的文件目录下创建两个以.tmp 为后缀的文件，用于存储处理好的代码片段。在程序结束或者用户更换要比较的文件时，程序会自行将.tmp 文件进行删除。此后的相似度检测均基于预处理好的两个.tmp 文件中的代码。

四、编辑距离算法

编辑距离算法是一种经典的字符串模糊匹配算法。[7]它描述了这样一个问题：

设 A 和 B 是两个字符串。我们要用最少的字符操作次数，将字符串 A 转换为字符串 B。这里所说的字符操作共有三种：

- 删除一个字符；
- 插入一个字符；
- 将一个字符改为另一个字符。

一种经典的求解方法是使用动态规划算法。假设 $f(i,j)$ 表示的是把字符串 A 中 [1,i] 这部分转换到字符串 B 中 [1,j] 部分所需要的最少操作次数。首先考虑边界情况：

- 对于$i=0$的情况，即字符串 A 为空串，那么对应的$f(0,j)$的值就是 j，即增加 j 个字符，使得字符串 A 转化成 B；
- 对于$j=0$的情况，即字符串 B 为空串，那么对应的$f(i,0)$的值就是 i，即减少 i 个字符，使得字符串 A 转化成 B。

进一步考虑一般情况，假设对于 $f(i,j)$ ，我们此前已经进行了 K 次操作。而此处的“之前”，一共有三种情况。

1. 将 A 的 [1,i] 转化为 B 的 [1, j-1]；
2. 将 A 的 [1,i-1] 转化为 B 的 [1,j]；
3. 将 A 的 [1,i-1] 转化为 B 的 [1,j-1]；

对于第 1 种情况，只需要在最后将 B[j] 加上就可以了，总共需要 K+1 次操作。

对于第 2 种情况，只需要在最后将 A[i] 删除就可以了，总共需要 K+1 次操作。

对于第 3 种情况，只需要在最后将 A[i] 替换为 B[j] 就可以了，总共需要 K+1 次操作。但是如果 A[i] 与 B[j] 相同，则不需要替换，此时总共需要 K 次操作。

取上述三种情况的最小值作为 $f(i,j)$ 的求解值，如是递推，得到最终结果。

采用此种传统的编辑距离算法，对程序进行逐行的求解，得到每一行的编辑距离D_i。同时运行用户输入一个阈值X，统计$D_i \le X$的行数L_X和程序的总行数L_{total}，那么最终两段代码的相似度即为：

$$\frac{L_X}{L_{total}} \times 100\%$$

五、简易语法树算法

要实现完整的 AST 算法是比较困难的，具体要包括 AST 的生成，特征序列的构造，聚类分析和抄袭检测等步骤。[8]因此，笔者决定自行对 AST 算法进行简化。

为了建立简化的 AST，我们考虑对代码中的关键语句进行识别，并利用 hash 算法的思想，给每一个识别出来的语句赋一个两位数的十进制特征值。各个语句与其特征值的映射关系如表 2 所示：

表 2

语句类型	特征值
变量创建语句	十位为 1，个位根据变量创建的类型而不同
函数声明/定义语句	十位为 2，个位根据函数的返回值类型而不同
循环语句	十位为 3，个位根据循环语句的类型而不同
判断语句	十位为 4，个位根据判断语句的类型而不同

编译预处理语句	十位为 5，若为 include 则个位为 1，否则为 0
其他语句	60

根据程序中的“{}”嵌套关系，进行建树。对整个语法树的结构，采用边链表进行存储。对每一行的代码，以其行数作为节点标号，用 STL map + STL vector 建立动态分配空间的边链表，同时存储节点的特征值。对于整个程序代码，建立 0 号节点作为整个语法树的根节点。自此，简化的语法树建立完成。

分别对要比较的两份源代码文件建立语法树，得到两个 AST。对于 AST 的比较，采用深度优先搜索，两个树同时从根节点开始遍历。对于每个节点，先判断它的子节点中相似的节点个数，然后判断当前节点的特征值是否相同。将相同的节点个数累加，作为 AST 的比较结果。

对于 AST 比较的相似度计算，设相似度为S，两份代码文件的语句行数分别为L_A和L_B，两个 AST 上相似的节点共有C对，那么相似度的计算公式为：

$$S = \frac{C \times 2}{L_A + L_B} \times 100\%$$

第 2 章 程序使用手册

一、初始界面

打开代码相似度检测器，可以看到如图 1 所示的界面，即为代码相似度检测器的初始界面。

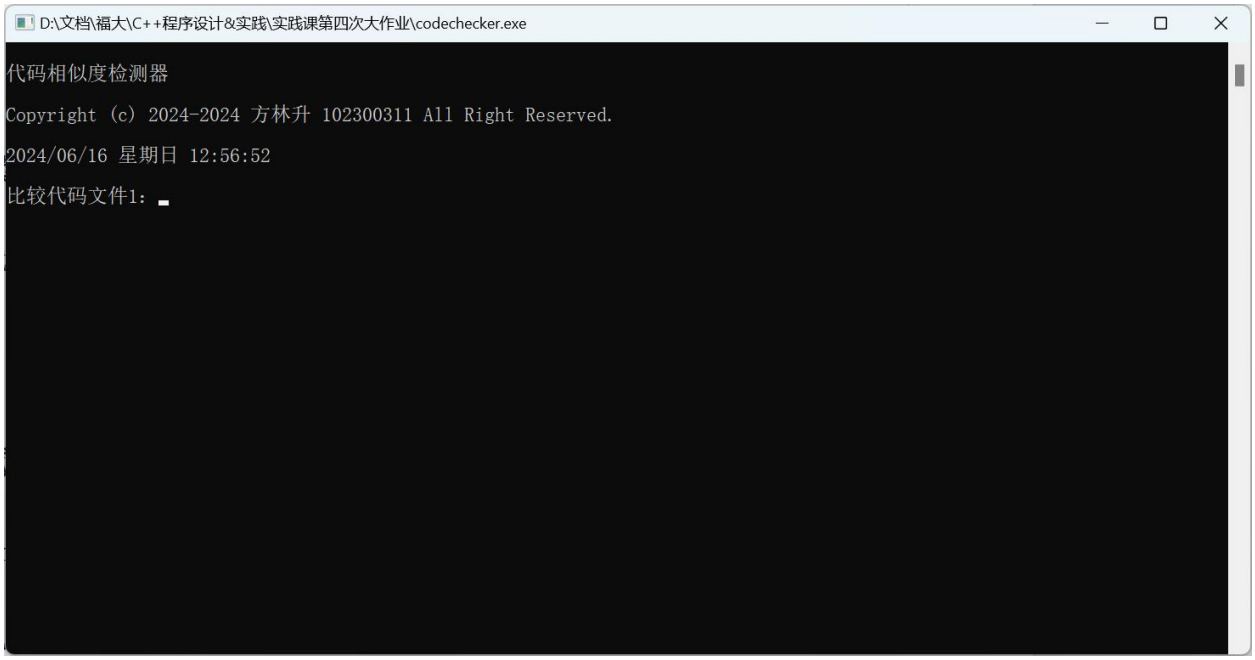

图 1

二、比较文件的输入

在打开程序后，用户需要输入要比较的代码文件来进行比较。输入时需要输入代码文件的后缀名。如图 2-1 所示。

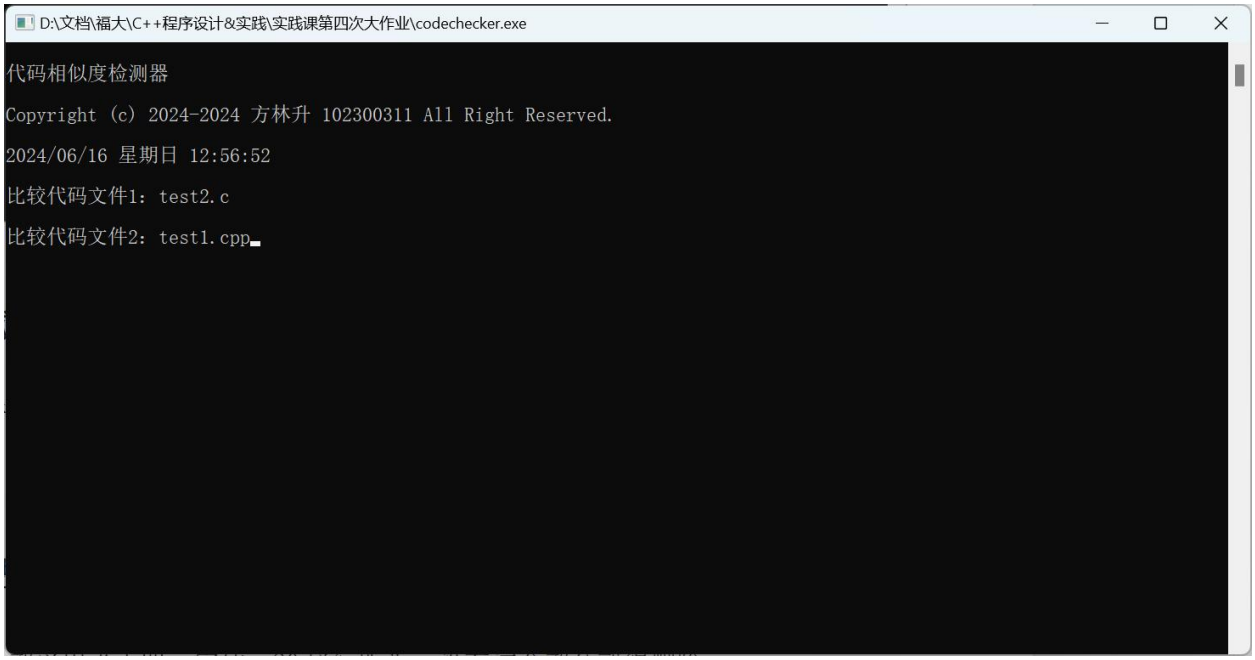

图 2-1

程序会在用户输入正确的代码文件名后，完成对代码文件的预处理，同时在与 exe 文件的同文件目录下，创建.tmp 后缀文件，即为处理好的代码文件。如图 2-2 所示。

图 2-2

用户可以使用记事本打开.tmp 后缀文件，即可看到预处理完成的代码，如图 2-3 所示。

test2.c.tmp

```
1  #include<stdio.h>
2  int main(){
3      char str[]=;
4      char s[]=;
5      printf();
6      return 0;
7  }
```

图 2-3

如果用户输入的文件名错误，或者程序无法打开代码文件，则程序会报错，如图 2-4 所示。

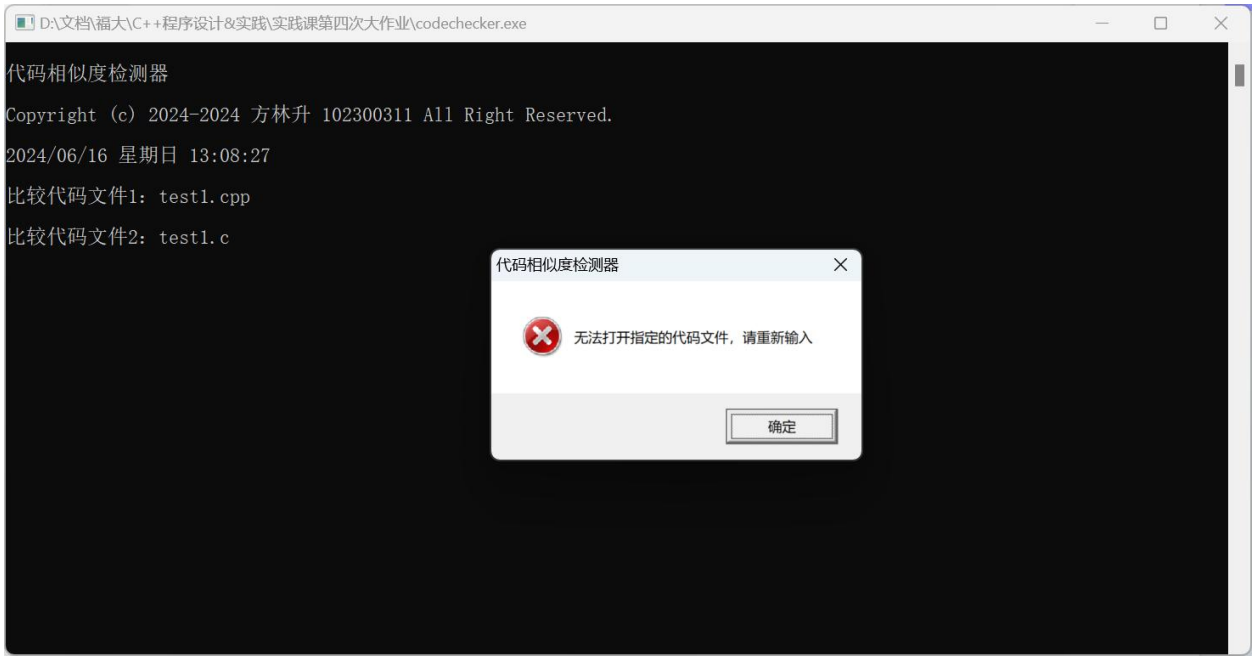

图 2-4

此时用户按下确定键后可以修改刚才输入的文件名，之后再按下 Enter 确认输入来修正错误的代码文件名。

三、主菜单

若用户输入的两个代码文件名均正确且可以打开，则用户会进入如图 3 所示的主菜单界面。

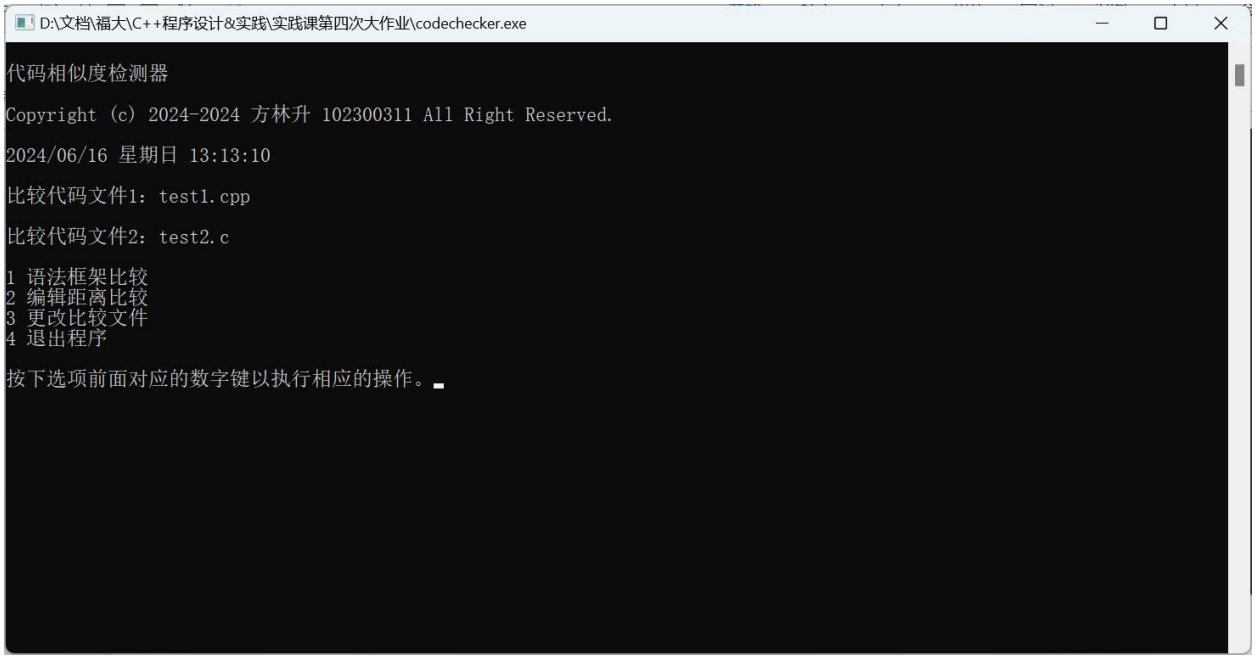

图 3

在主菜单界面，用户可以按下选项前对应的数字键来进行相应的功能操作。

四、语法结构比较

在主菜单界面，按下键盘上的“数字 1”按键，即可进行语法结构比较。

此时，程序会对两个代码文件建立 AST，并根据前文所述的比较过程和公式计算出比较结果，将结果以百分数的形式输出反馈给用户，如图 4 所示。

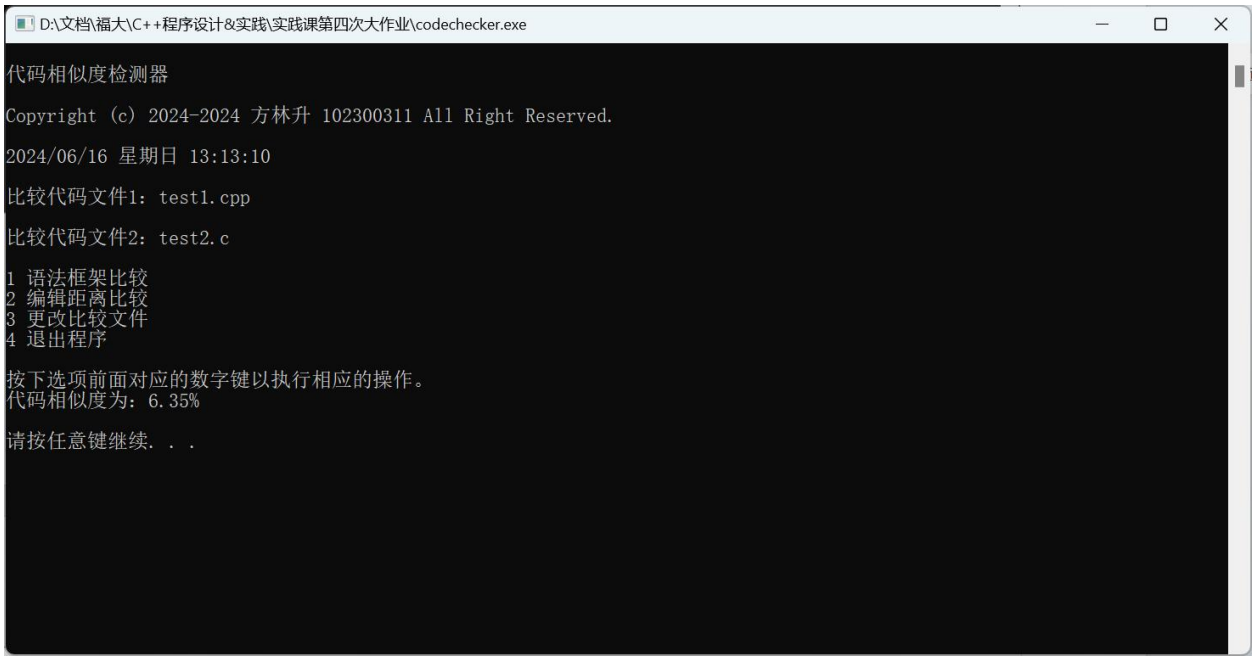

图 4

此时用户按下任意键，即可回到主菜单界面继续操作。

五、编辑距离比较

在主菜单界面，按下键盘上的“数字 2”按键，即可进行编辑距离比较。

此时程序会要求用户输入比较的阈值，如图 5-1 所示。

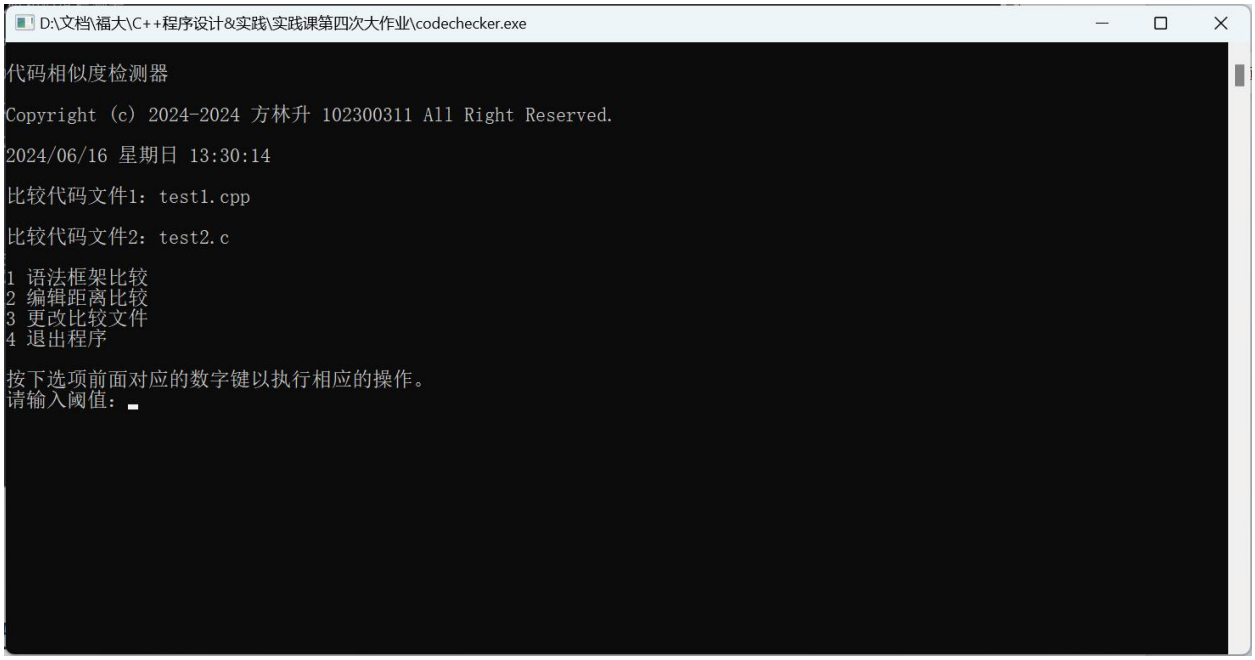

图 5-1

此时用户应当输入一个整数。值得注意的是，用户此时可以输入负数，但根据上文公式，这会导致比较得到的代码相似度为 0%。同理，如果用户输入的整数值过大，则会导致比较得到的代码相似度为 100%。用户需要自行根据情况选取合适的阈值。

若用户输入的不是一个整数，则程序会报告错误，并要求用户重新输入，直到用户输入的内容是一个整数为止，如图 5-2 所示。

图 5-2

若用户输入了合适的阈值，则程序会给出编辑距离比较的结果，如图 5-3 所示。

```
代码相似度检测器

Copyright (c) 2024-2024 方林升 102300311 All Right Reserved.

2024/06/16 星期日 13:37:07

比较代码文件1：test1.cpp

比较代码文件2：test2.c

1 语法框架比较
2 编辑距离比较
3 更改比较文件
4 退出程序

按下选项前面对应的数字键以执行相应的操作。
请输入阈值：fanglingsheng
fanglingsheng不是一个有效的阈值，请重新输入：fls
fls不是一个有效的阈值，请重新输入：10

代码相似度为：83.67%

请按任意键继续. . .
```

图 5-3

此时用户按下任意键即可回到主菜单继续操作。

六、更换比较文件

在主菜单界面，按下“数字 3”按键，即可进行比较文件的更换。

此时程序会重新回到初始界面，让用户重新输入要比较的文件。同时，旧的比较文件创建的.tmp 文件会被程序删除。如图 6 所示。

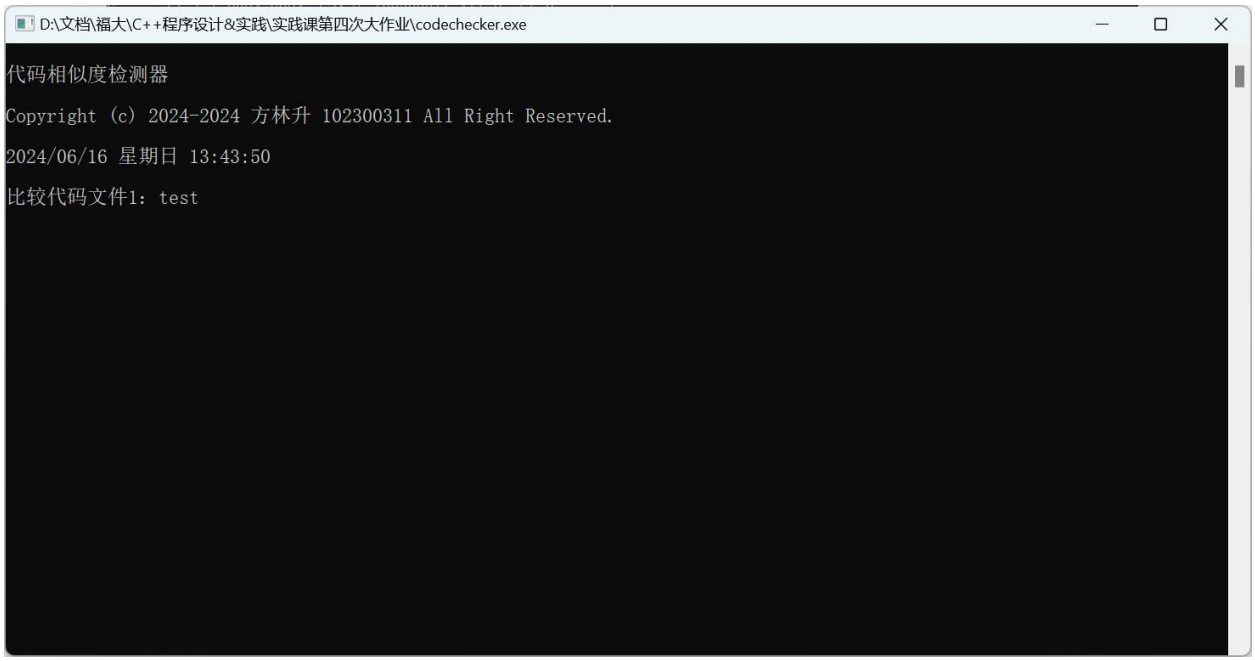

图 6

再次输入新的要比较的文件后，程序会重新回到主菜单界面供用户进行操作。

七、退出程序

在主菜单界面，按下“数字 4”按键，即可退出程序。

此时程序会删除先前创建的.tmp 文件，并关闭程序。

第 3 章 总结与体会

一、收获和体会

写一个代码相似度比较器确实是很困难的。在笔者一步一步实现的过程中，笔者逐渐发现要考虑的细节实在是太多了。就拿去除注释这一个小步骤来说。除了要考虑注释本身，还要考虑在字符串中的注释符号。既然要考虑在字符串中的注释符号，那就要考虑到对双引号的判定。而要判定能当作字符串的双引号，又要进一步判定这个双引号是不是转移字符或者是被单引号括起来的字符双引号。如此一来，一个简单的去除注释，实际上要考虑的情况已经是非常复杂多样的了。而去除注释仅仅只是整个设计过程中简单的一小部分，对于代码语句的判定情况，由于 C 语言对语法格式的高度自由，导致要考虑的情况更加复杂多样。于是不得已，笔者要对代码的风格做出一些假设，来简化自己的编写。

虽然这一次写出来的相似度比较器是很简易的，没有办法应对全部的代码抄袭情况。但是笔者在完成此次任务的过程中，通过文献的查阅，初步了解了代码查重技术的发展状况，对一些主流的算法有了初步的了解。这一过程为笔者未来在大学的进一步算法的学习和设计更复杂更困难的项目，奠定了重要的基础。

二、难点和创新点

难点：

- ✓ 复杂情况的简化；
- ✓ 对代码预处理的设计；
- ✓ 对编辑距离问题运用动态规划算法的求解，并将其修改后运用于代码查重过程中；
- ✓ 对 AST 算法的简化和编写实现。

创新点：

- ✧ 自主设计了代码相似度的简单计算公式；
- ✧ 对 AST 算法的简化和 hash 思想的加入运用。

三、程序概述

程序总行数：563 行

函数总数（含 main 函数）：7 个

四、自评

笔者对自己的程序进行了测试。这个代码能应对大部分第一型和第二型的代码抄袭情况，对一部分第三型的代码抄袭情况也能进行应对。笔者深知，设计一个完整的代码抄袭检测软件，其流程是非常复杂的。[9]由于笔者能力有限，必然存在设计和代码中的不足之处，欢迎批评指正。

自评情况如表 3 所示。

表 3

内容	得分
对源代码文件进行了去除注释，空行和字符串内容的预处理	1
对代码文件运用编辑距离算法进行比较，得出代码相似度	1
对代码文件建立简易语法树进行比较，得出代码相似度	1
其他工作（如查阅文献，设计简易界面等）	0.5
C 语言程序相似性检测系统	**3.5**

附录 参考文献

[1]孙祥杰,魏强,王奕森,等.代码相似性检测技术综述[J].计算机应用,2024,44(04):1248-1258.
[2]董文苑.基于代码风格分类的抄袭检测技术研究与应用[D].北京邮电大学,2021.DOI:10.26969/d.cnki.gbydu.2021.000340.
[3]刘飞翔,龙冬冬,欧幸茹,等.基于抽象语法树的代码抄袭检测方法的改进[J].吉首大学学报(自然科学版),2022,43(06):20-25.DOI:10.13438/j.cnki.jdzk.2022.06.005.
[4]陈恩恩.基于深度学习的代码相似性检测算法研究[D].华中科技大学,2021.DOI:10.27157/d.cnki.ghzku.2021.005346.
[5]孙令成,肖铁军.程序代码集到特征矩阵文本特征提取算法的研究[J].计算机与数字工程,2023,51(10):2363-2368+2378.
[6]张汉东.学生实验源码查重系统的设计与实现[D].华中科技大学,2021.DOI:10.27157/d.cnki.ghzku.2021.005578.
[7]张建雄.基于编辑距离的 C 代码相似度度量算法研究[D].华中科技大学,2017.
[8]朱良梅,洪晓彬.基于 AST 的程序代码抄袭检测方法研究[J].电脑知识与技术,2023,19(16):61-64.DOI:10.14004/j.cnki.ckt.2023.0872.
[9]段若恒.代码查重软件的设计与实现[D].西安电子科技大学,2021.DOI:10.27389/d.cnki.gxadu.2021.002695.