一、设计思路

1. 过滤。通过用户输入的文件路径读取文件，过滤文件（删除 cpp 文件里的注释），原文件中的注释、空格以及空行会对相似度结果造成影响，所以要先将这些字符删除；由于汉字没有对应的 ASCII 码并且也会对相似度造成一定影响，所以要将汉字也一一删除。打开文件后，按照字符读取文件，当读到特定字符的时候选择跳过，然后将非特定字符存到字符串中，读到’/’就需要判断下一个字符，当下一个字符为’/’时，继续读取，但不保留，直到读到换行符’\n’;当下一个字符为’*’时，继续读取但不保留，判断下一个字符是否为’*’：若是，再读取一个字符，若为’/’，则结束这次读取；若不是，则继续往下寻找”*/”组合，这样就直接跳过了注释，不将其写到目标字符串中；对于汉字则识别’”’，将两个双引号的之间的字符直接跳过。
2. 计算 hash 值。先将第一步过滤后的整个字符串切成长度为 K 的 N-K+1 个子串（N 为原字符串的长度），通过公式：

$$H(C_1\cdots C_k)=C_1*B^{k-1}+C_2*B^{k-2}+\cdots+C_k$$

来计算每个子串对应的 hash 值(C_k表示子串中的每个字符，B 表示基地，由用户自行设定)。该 hash 值具有抗冲突性，即不同的子串得到的 hash 值不可能相同。
3. 求特征值。由于存在原文件可能会很大的情况，所以从原始文档得到的 hash 值数目也可能会很庞大，而且每个 hash 值出现的次数应该相近，因此没必要保留所有的 hash 值，所以只需保留少数几个 hash 作比较就可以了。在原 hash 集上设置一个长度为 W 的滑窗，在每个窗口中选一个最小的 hash 值，将其保留下来（已选过的不选），这样就大大减小了比较量。
4. 求相似度。先对上面所求的特征值集排序，然后求两个集合具有相同元素的个数之和，即交集长度，并集长度即为两特征值集的长度之和，两个文件的相似度即为 Nc/Nt。（Nc 为交集长度，Nt 为并集长度）

二、伪代码

```
Function DeleteAnnotation(path)//过滤文本,path 是文本路径
    Begin
    Str←”” //存放过滤后的字符串
    Open(path,ios::in)
    If(文件未打开) then return “文件打开异常”
```

```
While get(c1) do//c1 为 char 型，逐字符读取文件
    If(c1=’ /’ )//删除注释
    Then
        Get(c2)//c2 为 char 型
        If(c2=’ /’ )//删除行注释
            While get(c1) do
                If(c1=’ \n’ ) then break
                Else then continue
            end
        Else if(c1==’ *’ )
        Then
            While get(c1) do
                If(c1=’ *’ ) then Get(c1)
                    If(c1=’ /’ ) then get(c1) break
                Continue
            End
    Else if(c1= ‘” ’ )//删除汉字
    Then
        While get(c2) do
            If(c2=’ ” ’ ) then get(c1) break
            Else then continue
        End
    If(c1≠’  ‘ and c1≠’ \n’  and c1≠’    ‘) then str+=c1/*删除空格、
空行*/
End
关闭文件
Return str
End

Function HashValue(str)//求 hash 值,str 为过滤后的字符串
```

```
Begin
While i+K<=N do//i 从 0 开始，K 为每个子串的长度，N 为 str 的长度
    For j i to K+i-1 step 1 do
        S+=str[j]//s 是一个空字符串用来存放子串
    end
    Shingles.push_back(s)//shingles 是一个 string 类型的 vector 数组
    i++
End
While i<N-K+1 do
    For j 0 to K-1 step 1 do
        Value+=asc(shingles[i][j])*pow(base,K-j-1)/*value 为 0 存放每个子串的 hash 值,base 为一个常数,pow 为求 base 的幂的一个函数,asc 为求字符 ASCII 的函数 */
    End
    hashvalue.push_back(value)//hashvalue 是一个 int 型的 vector 数组
    i++
end
return hashvalue
end

Function FlagValue(list)//求特征值，list 是一个 hash 集
Begin
For i 0 to n-1 step 1 do//n 为 hash 集长度
    Listnum[i].value←list[i]
    Listnum[i].used←false/*listnum 为一个结构体数组（Dict 包含 value 和 used）用来标记是否已选为特征值*/
End
While i+W<=n do//W 为滑窗长度,i 从 0 开始
    Mark=i//标记当前特征值下标
    For j i to W+i-1 step 1 do
```

```
            If(listnum[mark].value>=listnum[j].value) then mark=j//选最小值
        End
        If(!listnum[mark].used)//判断是否已用
            Then
                Flag.push_back(listnum[mark].value)/*flag 为一个 int 型的
vector 数组，存放特征值*/
                Listnum[mark].used=true//将最小值加进数组，并标记为已用
        i++
    end
    return flag
    end

Function Simility(list1,list2)//求相似度，list1、list2 均为特征值集
    Begin
    Sort(list1)
    Sort(list2)//对两个特征集进行排序
    unionset←list1.size()+list2.size()//并集长度
    for i 0 to list1.size()-1 step 1 do
        if(i>0)
            then
            if(list1[i]=list1[i-1] and flag)/*flag 表示在 list2 中是否有与之
相等的元素*/
                then intersection++ continue//交集长度加一
            else then flag←false
        for j 0 to list2.size() step 1 do
            if(list2[j]=list1[i])
                then flag=true intersection++
        end
        if(flag) intersection++/*若在第二个集合中找到了 list1[i]，并集长度加
一*/
```

```
    end
    value←intersection/unionset//相似度
    return value
    end
```

main.cpp

```
    str1 = DeleteAnnotation(path1);//第一个文件过滤后的字符串
    str2 = DeleteAnnotation(path2);//第二个文件过滤后的字符串
    hashvalue1 = HashValue(str1);//第一个文件的hash集
    hashvalue2 = HashValue(str2);//第二个文件的hash集
    Flag1 = FlagValue(hashvalue1);//第一个文件的特征值集
    Flag2 = FlagValue(hashvalue2);//第二个文件的特征值集
    result = Simility(Flag1, Flag2);//相似度
    return result
```

三、运行结果

```
C:\Users\_孤_\Desktop>codecheck.exe  C:\Users\_孤_\Desktop\实习\1.软件相似性实验\待测代码\02228.cpp  C:\Users\_孤_\Desktop\实习\1.软件相似性实验\待测代码\02228.cpp
1.0000
```

```
C:\Users\_孤_\Desktop>codecheck.exe 1.txt 2.txt
0.5556
```

```
C:\Users\_孤_\Desktop>codecheck.exe 3.txt 2.txt
0.4483
```

```
C:\Users\_孤_\Desktop>codecheck.exe  C:\Users\_孤_\Desktop\实习\1.软件相似性实验\待测代码\02304.cpp  C:\Users\_孤_\Desktop\实习\1.软件相似性实验\待测代码\03373.cpp
0.6239
```

```
C:\Users\_孤_\Desktop>codecheck.exe  C:\Users\_孤_\Desktop\实习\1.软件相似性实验\待测代码\02241.cpp  C:\Users\_孤_\Desktop\实习\1.软件相似性实验\待测代码\02231.cpp
0.5294
```

```
C:\Users\_孤_\Desktop>codecheck.exe  C:\Users\_孤_\Desktop\实习\1.软件相似性实验\待测代码\02304.cpp  C:\Users\_孤_\Desktop\实习\1.软件相似性实验\待测代码\02638.cpp
0.5208
```

```
C:\Users\_孤_\Desktop>codecheck.exe  C:\Users\_孤_\Desktop\实习\1.软件相似性实验\待测代码\02295.cpp  C:\Users\_孤_\Desktop\实习\1.软件相似性实验\待测代码\02294.cpp
0.2592
```