程序设计课程报告

22009200601 汤栋文

2023 年 5 月 23 日

目录

1. 个人信息

姓名：汤栋文
学号：22009200601
日期：2023 年 5 月 23 日

2. 任务重述

在所有题目当中完成 8 道题，其中每一类至少完成一道题目。这里我选择了完成如下题目。
（由于某些题目之间的相似性和关联性，所以有些相近的题目做到了一个 Project 当中）

Project1: FileProcess

1. （1.字符串处理）比较两个文本文件并打印出它们第一个不相同的行。
2. （2.文件处理）将合法 C 源程序每行前加上行号并删除其所有注释。
3. （3.文件处理）统计一个英文文本文件中 26 个英文字母出现次数并按英文字母序输出统计结果，查找并替换此英文文本文件中某字符串。
4. （4.文件处理）统计一个英文文本文件中所有单词出现次数并按英文字母序输出统计结果，查找并替换此英文文本文件中某单词。

Project2: LibraryManager

（此部分完成的和作业要求稍微有一点区别，加入了书籍的模糊查找功能，减少了借阅记录功能）

（5.综合系统）模拟图书馆管理系统。每名读者只能借一本书，读者可借书或还书。管理员可对图书和读者信息进行录入、修改和删除。图书信息至少应包括：编号、书名、数量，读者信息至少应包括：编号、姓名、所借图书。可根据图书名称或编号进行图书信息查询，可查询某本书现在被哪些读者借走。

Project3: RandomNumber

1. （6.其他问题）程序自动生成一个位于 99 内的随机数，要求用户猜这个数。用户输入数字后，程序有三种应答：too big, too small, you win。
2. （7.其他问题）产生一组随机数，要求每个数字不能重复。

Project4: Binary2Decimal

将输入的 2 进制字符串转换为 10 进制数输出。

3. 背景和说明

这里的程序都使用 C++完成，而且都在 Visual Studio 的一个项目中呈现。*（由于某些题目之间的相似性和关联性，所以有些相近的题目做到了一个 Project 当中，不要认为我只做了四个题目，这 4 个 Project 里面包含了 8 个题目）*在学 C++之前我已经学过 Python 和 Java 所以在编程风格和部分思想上，可以很大程度上受到这两种面向对象的编程语言的影响。但总之都不是什么大问题。*（作业文档中只呈现部分核心代码，完整代码由多个文件构成而且很长，具体看每个项目的文件夹。）（截图中部分不规范的写法只是为了截图方便，可能进行了一些不太合理的压行操作）*

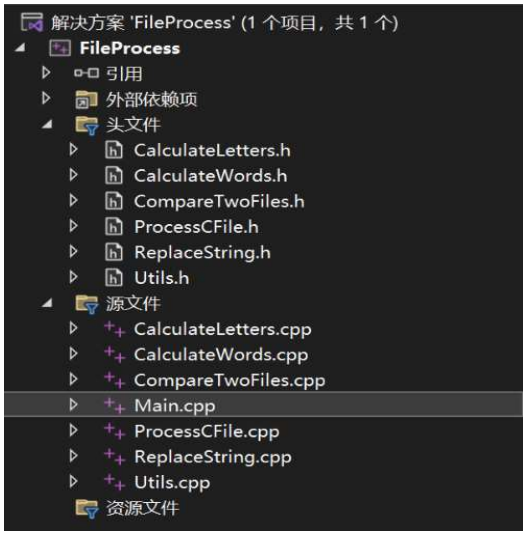
4. 主要程序解释

4.1 Project1: FileProcess

这个程序中实现了四个功能：

1. (1.字符串处理）比较两个文本文件并打印出它们第一个不相同的行。
2. (2.文件处理）将合法 C 源程序每行前加上行号并删除其所有注释。
3. (3.文件处理）统计一个英文文本文件中 26 个英文字母出现次数并按英文字母序输出统计结果，查找并替换此英文文本文件中某字符串。
4. (4.文件处理）统计一个英文文本文件中所有单词出现次数并按英文字母序输出统计结果，查找并替换此英文文本文件中某单词。

下面对这四个功能的实现和整体架构依次分析。



4.1.1 Main Structure

程序的输入是一个文件的路径，可以是相对路径可以是绝对路径。程序的输入有两种选择，一种可以直接使用命令行参数，程序会自动检测命令行参数格式和数量是否正确，并给出对应的反馈。如果没有任何命令行参数的输入，那么程序运行的需开始便要求用户输入一个文件路径（如下图所示），程序会在一切开始之前先把这个要处理的文本文件读入内存。

```
Enter new file name: C:\Users\MTDoven\source\repos\FileProcess\TestCode.c
```

下面便是一个用户操作界面，可以在此选择相应的功能。

```
C:\Users\MTDoven\source\repos\FileProcess\x64\Debug\FileProcess.exe
File Operater [Version 0.1.3]
(c)MTDoven. All rights reserved.

-------------------- File Operater --------------------
0. Close file and exit.
1. Count the number of letters.
2. Search and replace a string.
3. Count the number of words.
4. Search and replace a word.
5. Process C/C++ files.
6. Reload another file.
7. Compare two files.
8. Get help.
Enter a number:
```

4.1.2 Compare Two Files

文件比较部分比较简单，当我们选择文件比较的时候，程序会要求用户输入第二个文件名，使第二个文件与当前文件比较。比较方式很直接，就是从头开始遍历所有字符，遍历结束一直都相同，那就返回两个文件没有区别；否则发现第一个不同的字符就停止遍历，然后从此位置向前向后分别找到一个'\n'字符将两个换行符中间的内容返回。便很容易就得到了两个文件中不同的字符串。

```
Enter a number: 7

Enter new file name: C:\Users\MTDoven\source\repos\FileProcess\TestCode2.c
Difference in the these two lines below:
File1: /* Void neccessary varies */
File2: /* Void neccessary varies TEST */

Press enter to continue...
```

4.1.3 Calculate Letters

字母的统计通过 map 实现，把字符串读取进来以后通过一个 letter_filter 过滤掉所有非字母的字符，然后遍历字符通过 map 统计数目

```
map<char, int> calculate_letters(string str) {
    map<char, int> map;
    for (char letter : str) {
        letter = letter_filter(letter);
        if (map.count(letter) == 0) map[letter] = 1;
        else map[letter]++;
    } // count
    return map;
```

```
Enter a number: 1

Number of Letters:
        A:13    B:1     C:6     D:9     E:32    F:6     G:6     H:10    I:13
        K:2     L:11    M:8     N:11    O:11    P:5     R:8
        S:12    T:14    U:2     V:1     W:1     Y:2

Press enter to continue...
```

4.1.4 Calculate words

单词计数比字母计数稍微复杂一点，其实道理是一样的也是通过 map 实现的，但在开始统计之前，我利用正则表达式把字符串中非字母的所有元素替换成空格，然后再把所有连续空格替换成单一空格，这么一来，整个字符串就变成了一个比较干净的，仅仅由单词和空格组成的字符串。然后以空格切分，就可以获取对应单词，然后同 3.1.3 一样统计数量。

```
string string_filter(string str) {
    transform(str.begin(), str.end(), str.begin(), ::tolower);
    str = regex_replace(str, regex("[^a-zA-Z ]"), " ");
    str = regex_replace(str, regex(" +"), " ");
    return str;
}
```

4.1.5 Replace String

字符串替换，同样全部采用正则表达式处理，这样做的一个好处是，用户不仅可以替换指定的字符串，还可以输入正则表达式替换特定规则的字符串。

```
string replace_string(string str, string to_erase, string to_add) {
    str = regex_replace(str, regex(to_erase), to_add);
    return str;
}
string replace_string_icase(string str, string to_erase, string to_add) {
    str = regex_replace(str, regex(to_erase, regex::icase), to_add);
    return str;
}
string string_tolower(string str) {
    transform(str.begin(), str.end(), str.begin(), ::tolower);
    return str;
}
```

4.1.6 Replace Words

单词替换更加简单，因为我们已经有了字符串替换的函数，而且我们也有了一开始时统计单词数量的一套方法。因此我们可以判断单词是否在这个 map 中，即文章中是否存在这个单词，如果不存在则反馈给用户，如果存在就替换指定字符串。

替换过程我们直接利用已有的字符串替换函数，但是要注意替换的字符串并不仅仅是单词，第一要忽略大小写，第二要保证单词两端是非字母字符（防止出现替换“an”时把“want”变成了“wt”这种低级错误），但这两个问题都可以由正则表达式轻易解决掉。

```
words_map = calculate_words(str);
if (words_map.count(string_tolower(string_to_erase)) == 0) {
    cout << "Cannot find word " << string_to_erase << endl;
    pause(); goto start;
} // connot find inputed word
replaced_string = replace_string_icase(
    str, "\\b"+string_to_erase+"\\b", string_to_add);
string_to_file(replaced_string, arg);
```

4.1.7 Process C/C++ Files

要求删除注释并且加行号。同理，我们仍然使用正则表达式实现，但是这里在注释上，正则表达式的匹配变得有点困难，这里我 debug 了很久但是到最后仍然有极少部分的多行注释会处理错误，并且如果注释符号出现在双引号中也会被误删。这是两个已知的 bug 但是修改起来过于复杂……所以暂时就当它是一个特性吧。

```
string erase_annotation(string str) {
    string regex_string = "(/\\*([^*]|[\\r\\n]|(\\*+([^*/]|[\\r\\n])))*\\*+/)|(//(.*)\\n)";
    str = regex_replace(str, regex(regex_string), "");
    return str;
} // erase annotation
```

4.1.8 Other Explanation

当一个文件处理结束以后用户可以选择重新加载另一个文件。

说实话，这个程序中使用了不少 goto 语句，但是我认为在这个特定的任务中，goto 操作更灵活而且在逻辑上更通顺，可以根据用户的需要在不同的界面之间跳转，如果采用规范的循环操作想要实现此处的一些跳转逻辑，反而会显得有些混乱。

```
    else if (choice == 6) {
reload:
        cout << "\nEnter new file name: ";
        cin >> arg;
        goto load;
    } // To reload
```

4.2 Project2: LibraryManager

(此部分完成的和作业要求稍微有一点区别，加入了书籍的模糊查找功能，减少了借阅记录功能)

我完成的模拟图书馆管理系统，包含下列这些功能：

1. 每名读者可以借三本书，读者可借书或还书。
2. 管理员可对图书和读者信息进行录入、修改和删除。
3. 图书信息包括：编号、书名、数量、出版社。
4. 读者信息包括：编号、姓名、所借图书。
5. 可根据图书编号和学生学号进行信息查询。
6. 可以对图书信息进行模糊查找。

4.2.1 Book class

这里的书本的实现完全由一个 Book 类实现，完全按照面向对象的编程思想实现，里面的每一个函数其实都比较简单，只是把他们集合到一起，就会变得复杂起来。如果细心可以发现，这段代码中有几个函数名很像 java 中的函数名，正是如此，这整个项目其实是从我的 java 课程一次作业中移植过来的。

```
class Book {
private:
    int id;
    int number;
    string name;
    string publisher;
    string writer;
public:
    Book();
    Book(int id, int number, string name, string publisher, string writers);
    int getId();
    int getNumber();
    string getName();
    string getPublisher();
    string getWriters();
    bool equals(Book book);
    string toString();
    double Levenshtein(const string& s1, const string& s2);
    double Similarity(string str);
    bool is_available();
    void borrow();
    void restore();
    void add(int number);
```

4.2.2 Student class

同样，学生也是由一个类实现的。同上面的 Book 类类似，储存基本信息有些简单函数。

```
class Student {
private:
    Long id;
    string name;
    int number;
public:
    Student();
    Student(Long id, string name);
    Student(Long id, int number, string name);
    Long getId();
    string getName();
    int getNumber();
    bool equals(Student stu);
    string toString();
    bool is_borrowable();
    void borrow();
    void restore();
};
```

4.2.3 Database class

在这个类中把以上的 Student 类和 Book 类结合起来，在 Book 类和 Student 类相关函数的基础上，完成了集成度更高的工作。

其中构造函数 Database(string path) 这个函数从指定的文件中加载数据，加载到当前的 database 实例中，而 save 函数把数据储存到对应的文件中。

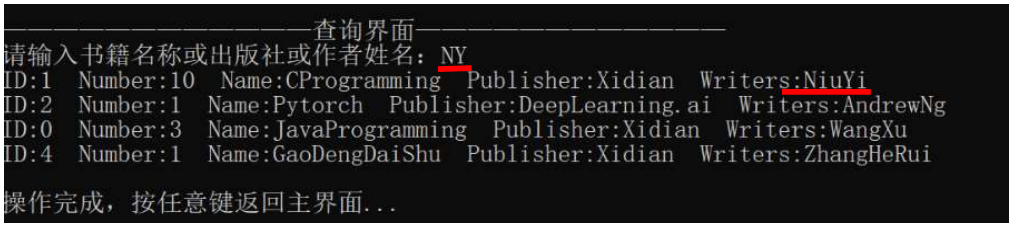
```
class Database {

private:
    int countbook;
    int countstudent;

public:
    map<int, Book> books;
    map<Long, Student> students;
    Database();
    Database(string path);
    int save(string path);

    int addbook(int number, string name, string publisher, string writer);
    void delbook(Book book);
    Long addstudent(Long id, string name);
    void delstudent(Student student);

    string toString();
    map<int, Book> getBooks();
    map<Long, Student> getStudents();
    int getCountbook();
    int getCountstudent();

    list<Book> search_book(string str);
    Book search_book(int id);
    Student search_student(string str);
    Student search_student(Long id);
};
```

4.2.4 User Page

整个程序有非常好的人机交互性能，可以在每一步给出对应的提示，引导用户完成相应的操作。具体的功能过于复杂也没有什么特别的，所以具体功能就不在此展示，具体可以参考代码。

4.2.5 Fuzzy lookup

这里使用了 Levenshtein distance 算法实现了模糊查抄，例如这里想要搜索“牛毅”写的书，那么我们只需要搜索“NY”便可以查找出作者是“NiuYi”的书。(其他书也被列出是因为这里是根据相似度进行排序的，默认列出前 4 个，这里牛毅的搜索相似度最高，所以这本 Cprogramming 排在了第一位)



4.2.6 Lost Log & Future Work

关于借阅记录的部分，我本来是做上去了的，但是因为按照原来的逻辑 Book 类和 Student 里里面各有一个 Log 类型的数组，而 Log 类里面又有一个 Book 和 Student 类型的成员变量。这样的代码在 Java 中没什么问题。但是当我移植过来以后就出现些奇怪的问题，好像是这样相互包含的关系在 C++中会引起些什么冲突，到最后也没调整好。所以索性删掉了整个 Log 记录的部分。*(这里粘贴出原来在 Java 中实现的一个 Log 类)*

```java
public class Log implements Serializable {

    @Serial private static final long serialVersionUID = 1027L;
    private final Student student;
    private final Book book;
    private final String state;
    private final String time;

    public Log(Student student, Book book, String state, String time){…
    }

    public boolean isBorrow(){ return this.state.equals("borrow"); }
    public boolean isRestore(){ return this.state.equals("restore"); }
    public String getState(){ return this.state; }
    public int getBookId(){ return this.book.getId(); }
    public long getStudentId(){ return this.student.getId(); }

    @Override
    public String toString() {…
    } // get information
```

4.3 Project3: RandomNumber

随机数以及相关小游戏，在此部分完成了两个任务：

1. （6.其他问题）程序自动生成一个位于 99 内的随机数，要求用户猜这个数。用户输入数字后，程序有三种应答：too big, too small, you win。
2. （7.其他问题）产生一组随机数，要求每个数字不能重复。

在开始具体的说明之前，我不得不吐槽一下这个 rand 函数，虽然 C 里面提供了这样一个函数。但是他有三个让人无法接受的缺点：

1. 必须用 srand 设置种子才能真的随机工作，否则每次返回同一个数字。
2. 最大值是由宏定义在头文件中写死的，除非修改基本的头文件不然没法修改最大值。
3. 也是最致命的，一个随机函数竟然不随机，我的意思是它的随机性极差极差，看起来简直就像不随机一样，甚至我在排除重复数字的时候，因为它每次给出相同的数字导致了死循环的出现，让程序编写过程困难重重。

综上，我放弃了 rand 函数，自己利用时间、字符串、哈希函数，写了一个伪随机。

4.3.1 Random Function

这个伪随机过程分为四步。第一步初始化随机类，第二步利用当前时间添加“随机成分”，第三步把累计的时间随机成分换算成字符串，第四步根据字符串哈希值获得随机值。

1. 初始化随机类：这一步非常简单不做过多解释。

```
Random::Random() {
    auto t = chrono::system_clock::to_time_t(std::chrono::system_clock::now());
    string_stream << put_time(localtime(&t), "%Y%M%H%m%d%S");
    int random_mode = 1;
} // init
```

2. & 3. 添加“随机成分” & 换算成字符串：把当前时间按照特定格式输出到字符串。这里的字符串是在原来的基础上不断积累的(过长后会重置)，这就导致了每次的字符串不仅内容相差很多而且长度相差也很多，在很大程度上保证了随机性。

```
void Random::randomstamp() {
    string rubbish;
    rubbish = string_stream.str();
    if (rubbish.size() > 1024) {
        string_stream.str("");
        string_stream.clear();
    } // avoid too long
    auto t = chrono::system_clock::to_time_t(std::chrono::system_clock::now());
    if (random_mode == 1) {
        string_stream << put_time(localtime(&t), "%m%Y%d%S%H%M");
        random_mode == 2;
    } else if (random_mode == 2) {
        string_stream << put_time(localtime(&t), "%Y%S%m%M%d%H");
        random_mode == 3;
    } else { // (random_mode == 3)
        string_stream << put_time(localtime(&t), "%Y%M%H%m%d%S");
        random_mode == 1;
    } // put into string_stream
```

4. 根据字符串哈希值获得随机值：利用一个获取哈希值的函数获得了当前字符串的哈希值。把这个无符号整形根据输入的上下限进行转换，取余等操作，最终得到了特定范围内的随机数。

```
int Random::randint(int low, int high) {
    randomstamp();
    unsigned int randint = (unsigned int)hasher(string_stream.str());
    int result = randint % (high - low + 1) + low;
    return result;
} // get randint
```

0. 我做了相关实验，实验证明这种方式得到的随机数在数字给定的随机上下限较小且较接近时，这个随机是一个很标准的平均随机。但是当给定上下限较大的时候（相比较于 unsigned int 的最大表示范围而言，达到最大表示范围的大约 1/8 以上时）就不那么平均了。这其实是由于取模运算导致的问题，具体情况是数学问题不继续分析，但这相比于那个不随机的随机函数 rand 已经好多了。

4.3.2 Guess Number Game

有了随机性很好的随机函数，这个任务就易如反掌了。
逻辑部分代码实在是过于简单，这里直接掠过了。

4.3.3 Random Number Generator

有了随机性很好的随机函数，这个任务也比较简单了。这里唯一要注意的就是，不能重复。不重复的方法很简单，就是如果重复了就再来一次随机，这里要注意避免一个死循环（用户想要 5 个随机数，但是给的随机范围是 1-4，这就会出现死循环）。另外，我发现在 scanf 中其实也可以使用正则表达式，这样就可以让用户输入更具有鲁棒性，我们的程序不容易因为用户的胡乱输入而崩溃。

```
void RandomNumberGenerater(Random& randomer) {
    cout << "\n";
    cout << "How many numbers do you want: ";
    int items; cin >> items;
    if (items < 0) items = -items;
    cout << "Input the lower and higher bound of numbers: ";
    int low, high;
    char rubbish[RUBBISH_LENGTH] = "";
    scanf("%[^0-9]%d%[^0-9]%d", rubbish, &low, rubbish, &high);
    cin.ignore(numeric_limits<streamsize>::max(), '\n'); // clear
    // input and time counting
    if (items > high - low + 1) {
        cout << "Number of items is bigger than bound.\n";
        return;
    } // check number
    int randints[LARGEST_ITEMS] = { 0 };
    for (int i = 0; i < items; i++) {
        int randint = randomer.randint(low, high);
        randints[i] = randint;
        for (int j = 0; j < i; j++)
            if (randint == randints[j]) i--;
    } // get random numebr
    cout << "\nOutput of the random number: \n";
    for (int i = 0; i < items; i++)
        cout << randints[i] << " ";
    cout << "\n";
    // output
}
```

4.4 Project4: Binary2Decimal

这个任务也是比较简单的，毕竟要求每一类题目都要做，所以数据处理部分就做了这一个进制转换。实现原理也是非常简单，就是循环着去乘以 2 的对应次方，我知道这并不是最高效的方法。但是在这里作为一个任务完成我觉得也还算可以的。

4.4.1 Main Algorithm

核心就是这一个循环仅此而已，非常简单。

```
int result = 0;
for (int i = length-1; i >= 0; i--)
    result += str[length-i-1] * pow2(i);
```

4.4.2 User Part

要防止用户给出奇奇怪怪的输入影响程序的执行，所以在用户信息读取方面稍微有一点注意。利用字符串读入并且判断是否全是 0/1 值。

```
int main() {
    cout << "Input a binary number: ";
    char str[MAX_INPUT_LENGTH] = { 0 };
    scanf("%s", str);
    int length = strlen(str);
    for (int i = 0; i < length; i++) {
        int this_number = AsciiToInt(str[i]);
        if (this_number < 0 || this_number>1) {
            cout << "Please input binary number with right format.";
            return 1;
        } // check if between 0,1
        str[i] = this_number;
    } // get input finished
```

4.4.3 Future Work

这里的转换其实对数字的大小有要求，并不能完成超过一定大小的数字，对于超大数字的二进制转换可以成为这个项目接下来的工作，包括运算的效率也是如此。

5. 任务总结

（再次说明一次：由于某些题目之间的相似性和关联性，所以有些相近的题目做到了一个 Project 当中，不要认为我只做了四个题目，这 4 个 Project 里面包含了 8 个题目）

这些代码很大程度上提高了我的 Coding 能力。借着这次作业的机会，我在已经学过 Python 和 Java 的基础上，基本上说是用了很短的时间自学了 C++。

正如老师说的，语言没那么重要，更多的是一种编程的思维。我们不仅要考虑算法的实现过程，怎么才能让用户用着舒服，怎么才能让我们的程序在用户的糟蹋下仍然不会出错，这些也是一个完成的程序中需要考虑的问题。

最后感谢老师一年以来的辛勤教导，也恭喜我自己已经基本掌握三种主流编程语言。