文献阅读笔记

目录

1. **选题及选题理由…………………3**
2. **阅读的相关文献、阅读目的、阅读后的收获……………………4**

**(3)文献阅读后的感想……………10**

**(1)选题及选题理由**

我选择的题目是**家庭财务管理**DOS版，我选择该题目的理由是：

1. 我刚学习C++不久，对C++的知识掌握的不太全面，所以我尽量去选择在我能力范围之内的题目去完成。最后我选择了家庭财务管理系统。
2. 由于我目前不会MFC，但对Windows消息机制有所了解，考虑到一点都不会MFC，而从0学习一个陌生的东西，在不熟悉的情况下立即用其制作比较大的系统可能会在制作的过程中会带来很大的问题，也可能因为对MFC掌握的不够而产生很多未知的潜在bug。而一个学期下来我都在和DOS接触，对DOS比较熟悉，所以选择DOS版。
3. 题目中有很明确的对象指定，结合上个学期学到的计算机导论，我很快就能知道该制作哪一些类，比较好上手，同时能进一步加深巩固这个学习所学习到的面向对象的知识点。能很好的体现个人面向对象编程的思想和能力，也是对大一所学知识的综合运用。
4. 由于项目是一个人完成，通过这个题目我能学习到很多一个人完成项目的经验，以及遇到问题如何一步步解决的经验。
5. 实现该系统后，我可以将其作为我自己个人的财务管理系统，方便我个人的财务管理，能实现财务信息的自动化，需要时还能随时添加功能，比较和我胃口，符合我的需求。既能将其作为课程的项目来提交，也能作为我自己的一个实用小工具，一举两得，乐而为之。

**(2)阅读的相关文献、阅读目的、阅读后的收获**

阅读的相关文献：

1. C++ STL vector容器元素的查找和搜索 find() find\_if()函数的使用

文献网址: <https://blog.csdn.net/Ibelievesunshine/article/details/80214158>

阅读目的:了解如何在vector中根据某个变量去查找某一个元素。

阅读后的收获:掌握了find()和find\_if()两个函数的用法，并最终在项目中选择使用find\_if()函数作为对象查找函数。

1. vector中find的用法

文献网址:

<https://blog.csdn.net/upcer_Yangyifu/article/details/78943400?utm_medium=distribute.pc_relevant_t0.none-task-blog-BlogCommendFromMachineLearnPai2-1.nonecase&depth_1-utm_source=distribute.pc_relevant_t0.none-task-blog-BlogCommendFromMachineLearnPai2-1.nonecase>

阅读目的:进一步掌握find()的更多用法。以用来比较是find()符合需求还是find\_if()符合需求。

阅读后的收获:在阅读完文献后，我认为find\_if()函数更加适合我的项目，所以我选择使用find\_if()而不使用find()函数，但是阅读完这篇文献后我进一步掌握了find()函数的用法。

1. 为什么静态成员函数不能是const

文献网址: <https://www.cnblogs.com/MyBlog-Richard/articles/5995040.html>

阅读目的:使用静态函数时不能将其标记为const，所以去了解原因。

阅读后的收获:了解到了这是C++的规则，const修饰符用于表示函数不能修改成员变量的值，该函数必须是含有this指针的类成员函数，函数调用方式为thiscall，而类中的static函数本质上是全局函数，调用规约是\_\_cdecl或\_\_stdcall,不能用const来修饰它。一个静态成员函数访问的值是其参数、静态数据成员和全局变量，而这些数据都不是对象状态的一部分。而对成员函数中使用关键字const是表明：函数不会修改该函数访问的目标对象的数据成员。既然一个静态成员函数根本不访问非静态数据成员，那么就没必要使用const了。

1. C++ 有 super() 方法没?

文献网址: <https://blog.csdn.net/xclyfe/article/details/37810143>

阅读目的:做项目时突发奇想，好奇心上来了先了解一下，因为JAVA中有super指针，没听过C++有，就想了解C++有无super指针。

阅读后的收获:C++中没有super指针，因为C++支持多重继承，用super没办法准确指定一个父类，所以就没有super。

1. C中int8\_t、int16\_t、int32\_t、int64\_t、uint8\_t、size\_t、ssize\_t区别

文献网址: <https://blog.csdn.net/yz930618/article/details/84785970>

阅读目的:在看cppreference时发现个别地方使用了uint(x)\_t这种用法，去查看了一下这个是什么类型。

阅读后的收获: int\_t 为一个结构的标注，可以理解为type/typedef的缩写，表示它是通过typedef定义的，而不是一种新的数据类型。因为跨平台，不同的平台会有不同的字长，所以利用预编译和typedef可以最有效的维护代码。

1. std::basic\_ostream<CharT,Traits>::write

文献网址: <https://zh.cppreference.com/w/cpp/io/basic_ostream/write>

阅读目的:了解并掌握write的用法，掌握二进制写文件的方法。

阅读后的收获:掌握了write的用法，刚开始想通过这种方法一次性将整个vector存入文件中，后来发现并不行，原因是vector中有动态分配的内存，用二进制写只能写入指针数据，不能把堆中动态分配的内容存入文件中。后来就放弃了这个方法。

1. std::basic\_istream<CharT,Traits>::read

文献网址: <https://zh.cppreference.com/w/cpp/io/basic_istream/read>

阅读目的: 了解并掌握read的用法，掌握二进制读文件的方法。

阅读后的收获:掌握了read的用法，刚开始想通过这种方法一次性将整个vector存入文件中，并再下一次启动程序时用read一次性读入，但是后来发现并不行，原因是vector中有动态分配的内存，用二进制写只能写入指针数据，不能把堆中动态分配的内容存入文件中。后来就放弃了这个方法。

1. C++ 判断文件是否存在的几种方法

文献网址: <https://blog.csdn.net/guotianqing/article/details/100766120>

阅读目的:了解C++判断文件是否存在的方法，在我的项目中使用。

阅读后的收获:学到了4个方法来判断文件是否存在，使用ifstream打开文件流，成功则存在，失败则不存在；以fopen读方式打开文件，成功则存在，否则不存在；使用access函数获取文件状态，成功则存在，否则不存在；使用stat函数获取文件状态，成功则存在，否则不存在。

1. fstream 中判断是否成功打开文件

文献网址: <https://www.cnblogs.com/LonelyEnvoy/p/5940444.html>

阅读目的:学习用fstream如何判断文件是否打开成功。

阅读后的收获:学习到了if(!fin)这种方法，能够将流对象转化为bool类型进行判断，从而得知文件是否打开成功。

10、C++ ofstream和ifstream详细用法

文献网址: <https://www.cnblogs.com/hjj-fighting/p/10429178.html>

阅读目的:深入学习ofstream和ifstream的详细用法，熟悉文件读写的常用操作。以及判断是否读写成功的方法。

阅读后的收获:掌握了很多的方法，并将其运用到了我的项目当中。同时深入学习了seekp、tellp、seekg和tellg四个函数，熟悉了如何在文件中移动文件读取/写入指针。

11、C语言如何在控制台窗口设置光标到指定位置

文献网址: <https://blog.csdn.net/edc370/article/details/79944251>

阅读目的:掌握如何移动控制台的输出光标，从而达到任意位置输出内容的目的。

阅读后的收获: 掌握了控制台任意位置写的方法，但是后来发现不能覆盖部分已经输出的字体，没有达到我想象的效果，所以选择弃用这个方法，而选择去使用比较简单粗暴的system(“cls”)来达到效果，最终在项目中也使用了system(“cls”)来达到我想要的效果。

12、C++ 中“空引用”与“空指针”的区别

文献网址:

<https://blog.csdn.net/luansxx/article/details/10134139?utm_medium=distribute.pc_aggpage_search_result.none-task-blog-2~all~first_rank_v2~rank_v25-4-10134139.nonecase>

阅读目的:在设计过程中有一个地方想返回空引用，故来查阅是否可以返回空引用。

阅读后的收获:C++不能返回空引用，为了达到目的，我还是选择去改用返回指针的方式，这样就可以返回空指针了。

13、C++对and、or等运算符的支持

文献网址: <https://blog.csdn.net/maray/article/details/41644235>

阅读目的:查看C++中是否已经支持and or 等关键字，这样就不再使用||和&&作为或与的条件判断了。

阅读后的收获:ISO646已经支持了and和or关键字，所以项目中的与和或改用and和or关键字进行表示。

14、c++11 之emplace\_back 与 push\_back的区别

文献网址: <https://blog.csdn.net/p942005405/article/details/84764104>

阅读目的:查看emplace\_back和push\_back两个函数的区别，哪个函数相对来说比较高效。

阅读后的收获:emplace\_back更加高效，因为是原地构造，而push\_back还需要一次复制构造或者移动构造，比较耗费空间和时间，所以根据项目的需要选择使用emplace\_back。

15、如何正确读取和写入std :: vector到一个文件？

文献网址: <https://cloud.tencent.com/developer/ask/44195>

阅读目的:查看如何将一整个vector保存到文件，实现项目“数据保存到文件”的需求。

阅读后的收获:vector内含有动态分配的内存，直接二进制读写到文件不方便，所以我最终在项目中选择将里面成员的数据一个一个的保存到文件中，而不是一整个保存。

16、C++将vector写入文件并重新读取

文献网址: <http://www.veryhuo.com/a/view/87874.html>

阅读目的:学习如何将vector里面成员的数据一个一个的保存到文件中，并且学习将文件的内容重新读入到程序的操作过程。

阅读后的收获:学会了如何将vector中的成员数据保存到文件中，并且经过测试后成功写入和读入，成功将其运用在项目的制作当中。

17、set底层实现方式？

文献网址: <https://www.cnblogs.com/zhuyeshen/p/12675150.html>

阅读目的:刚开始想用set而查看set的底层实现，看是否在查找，添加，删除效率上比较好。

阅读后的收获:了解了set的底层机制为红黑树，和multimap一样，但是set只能存一个元素比较不符合项目要求，所以后来改用了multimap，可以存键值对。

18、C++ STL set emplace()和emplace\_hint()方法详解

文献网址: <http://m.biancheng.net/view/7197.html>

阅读目的:了解set的emplace()和emplace\_hint()的用法，掌握在set中添加元素的方法。

阅读后的收获:掌握了这两个函数的用法，只不过最后决定在项目中使用multimap而不是set，但是multimap的这两个函数的用法是一样的，所以这里也算学习了multimap添加元素的方法。

19、STL之关联容器---set, mutilset, map, mutilmap

文献网址: <https://www.cnblogs.com/chenzhefan/p/5708620.html>

阅读目的:决定在我的项目中使用哪一种容器。

阅读后的收获:掌握了各种容器的区别，我决定在项目中使用multimap。

20、C++ multiset用法详解（附带完整示例）

文献网址: <http://c.biancheng.net/view/545.html>

阅读目的:因为决定在项目中使用multimap，所以要查看相关的用法。

阅读后的收获:知道了multimap能够存储相同键值对多个，所以正好用来记录收支信息，键是日期，值是相关的信息，这样只需要提供日期就能返回相关的值信息，符合项目设计的需求。

21、std::multimap<Key,T,Compare,Allocator>::emplace

文献网址: <https://zh.cppreference.com/w/cpp/container/multimap/emplace>

阅读目的:需要知道emplace函数的返回值，了解其返回值是什么，怎么使用。

阅读后的收获:了解了emplace函数的返回值是插入对象的引用，可以用它来做很多事情，最终也在项目当中体现了这一点，用返回的引用来设置数据。

22、C++11 强类型枚举 enum class

文献网址: <https://www.cnblogs.com/moodlxs/p/10174533.html>

阅读目的:了解c++11的强枚举类型，并在项目中使用。

阅读后的收获:掌握了C++强枚举类型的相关用法，并在项目中用它来表示性别、权限等枚举类型。

23、C++核心准则Enum.5: 不要使用全部大写的枚举值

文献网址: <https://jishuin.proginn.com/p/763bfbd23a26>

阅读目的:因为使用了强枚举类型，所以要了解相关的命名规则，要有良好的代码风格。

阅读后的收获:了解到了因为全大写的枚举类型可能会和宏定义相冲突，所以只有大写首字母就可以了，不用全部大写。我在项目中也遵守了这个规则。

24、C++如何跳出多层循环

文献网址: <https://www.cnblogs.com/dapeng-bupt/p/8622787.html>

阅读目的:在做项目时遇到了这个难题，自己想不到比较好的解决办法，所以来查阅相关的解决方案。

阅读后的收获:了解到了比较常用的三种方法，用bool标志判断，用goto跳转，用return返回，最终在我项目的函数中根据需要使用了goto和bool标志两种方法，但同时我了解到应该少用goto，所以我在项目中尽量用bool标志去判断，少用goto。

25、关于lower\_bound( )和upper\_bound( )的常见用法

文献网址: <https://blog.csdn.net/qq_40160605/article/details/80150252>

阅读目的:学会在multimap中根据一键多值原则查找元素。

阅读后的收获:学会了lower\_bound和upper\_bound的区别，upper\_bound类似于>，lower\_bound类似于>=，在项目中广泛使用到了这两个函数来根据日期获取对应的收支信息。

1. **文献阅读后的感想**

在阅读了众多文献之后，我慢慢掌握了一些阅读的方法。

首先，在阅读文献之前要明确自己阅读这个文献的目的，要带目阅读。在明确目的之后，要广泛的去阅读相关的文献，因为要确保文献的准确性和全面性。对于文献，可以是百度的结果，谷歌的结果，或者其他搜索引擎的结果，也可以是知网的文献，个人认为比较好的是cppreference.com的内容。

其次，在文献阅读过程中，要全面理解写文献的人想表达的意思，掌握相关函数等东西的用法，最好是一边看着文献，一边动手跟着文献作者的思路写一下代码，并且检查每一步和作者得到的结果是否一致。在文献阅读完成之后，需要自己动手做一下，打开一个空项目去试着写一下，练一下手，使用一下刚刚阅读文献所看到的全部内容，得到结果后再回来看看结果是否与文献所说的保持一致，检验自己是否完全掌握文献作者所说的意思，检验自己是否已经完全掌握了相关函数的用法写法。并且如果自己想做一些相关拓展的话可不可行。

最后，在自己写空项目实验过后，完全掌握相关用法之后，再在自己的项目中尝试去使用相关的函数，如果最后测试通过，达到了想要的效果，那就证明了这次的文献阅读是成功的，自己收获了相关函数的用法和写法。如果测试没有通过，说明自己对相关内容的理解还不够，还需要多次反复的阅读相关文献，或者扩大搜索范围，查看是否是文献讲的不够全面。

另外，如果觉得这个文献有精彩的地方或者怕自己学过后忘记了这个做法，可以讲文献的重点内容记录下来，做为以后写代码的备用资料。