7.总结报告

在上个周里，我们小组三人都为完成程设大作业而努力奋斗。下面就是我们的总结报告：

**7.1任务分工**

|  |  |  |
| --- | --- | --- |
| 姓名 | 分工 | 完成情况 |
| 闫嘉丽 | 绘制类的关系图  撰写部分总结报告 | 全部完成 |
| 刘若冰 | “保险类”是设计和实现  撰写部分需求分析  撰写部分设计说明书  撰写类的数据成员和函数成员说明 | 全部完成 |
| 杨雨函  （组长） | 调试第十章综合实例  将原有vector模板改成链表类模板  将“保险”类嵌入主函数中，并修改调试相应程序  撰写部分需求分析  撰写部分设计说明书  撰写部分总结报告 | 全部完成 |

**7.2优势与特色**

**①**采用链表类模板来保存动态的数据，更有利于在线性的数据结构中在特定的位置增加和删除数据。

②在这个改进后的链表类模板中通过初始化游标的方式来确定游标的位置，因此也不像一般链表类一样需要按顺序挨个访问，大大节省了时间。

③增添“保险”类这一典型理财功能，丰富了原程序的功能选项。在银行系统中抽象缴纳保险这个功能会为用户带来极大的便利。

④组员对保险项目比较了解，知道的细节较多，花了比较多的功夫进行了调研，非常贴近现实生活。

⑤延展性、重用性好，可供改进的空间很大。将基类insurance设置成了抽象类，并将其成员函数设置成虚函数，实现了不同派生类对函数调用时的动态绑定，实现多态性。所以虽然目前“保险”类只有一个派生类project1，但后续继续增加派生类仍然是一件十分方便的事。

⑥我们利用了vector模板创建了基类指针类型的动态数组，这样就能实现动态数据结构的管理，并利用模板内的函数实现得到数组大小、增添数据等功能。

⑦人性化设计。增添提示性指令，让用户知道自己应该输入些什么数据。

**7.3不足与改进**

①由于时间和能力有限，“保险类”只设计了一款产品。可以更加丰富产品选项，改善用户体验。

②程序中类与类之间还存在一些重复的操作，可能以后能够对更多的细节进行重新设计与改进，使程序更加简洁。比如链表类模板中就有许多操作其实是可以通过其他操作来代替的，可以说这既是一种使用类时的简便又是一种编代码时的繁琐。

③对于自己设计的保险类缺少对不合法数据的检验，用户一旦输入错误数据类型，程序就会陷入死循环或者异常中止。可以在以后改进时增添对输入数据的检验，一旦不合法就自动弹出程序或者提醒用户重新输入。

④保险类产品的扣钱和存钱功能主要是在主函数里实现的，没有提取出一个类似于结算函数的成员函数，这意味着对数据的操作仍然有一部分是在类外，类的封装性还不够好，日后修改程序时会比较麻烦。可以借鉴account类的结算函数，再向保险类增添结算函数，在类内部实现对数据的操作。

**7.4收获与体会**

杨雨函：我记得老师在学第一章时就说过，让我们把生成代码和调试过程中遇到的错误类型给记下来，我现在终于深深意识到这是多么有意义是一件事啊！无数次我看到那些错误都“似曾相识”，却怎么也想不起来该怎么改正代码。后来我终于开始学会记载错误类型，很多时候甚至不用调试就能知道错误出在什么地方，大大提高了测试代码的效率。另外，两个周高密度的学习其实很难把老师讲授的所有内容都能很好地消化吸收，在平时就可以多和同学讨论交流，你往往会发现困扰你大半天的bug，别人一眼就能看出来是怎么回事。最后，c++的这种封装、继承、多态的特点，不仅在编程语言里独树一帜，而且其实也是一种新的思考问题解决问题的方法，实实在在地打开了我的眼界和思维模式。

刘若冰：最大的收获肯定是从无到有地在c++上浅入门了，学会了看待问题的另外的角度，之前的时候思考问题始终是面向过程的方法思考，解决问题也是尽量找规律窍门解题，但是现在学完编程之后思考问题多了面向对象的方法，有时候用c++处理问题也可以不用大费周章地考虑如何巧解，直接用简单的方法让电脑解问题就可以了。还有就是我自身的编码实力太弱，调试纠错的效率很低，有的时候做类出来并没有体现类的优势，而是借用了类的壳子把问题弄的越来越复杂。未来的改进，一个是这段时间的编程学习太过仓促，有很多内容还没有消化就开学了，之后要把书里的知识好好看一遍，把各个例题自己写一遍，把课后习题看一遍，才能稍微对编程掌握的更熟悉一点，还有些内容要参考一下其他的教材，尤其是与类相关的应用，多见一点不同的案例，才会多一点理解。

闫嘉丽：任务涉及一些需要自学的东西，完成的时候感觉比较艰巨，因此也为此付出了很多心思。但在队友的支持与帮助下也顺利进行，无论是课程知识的学习还是团队配合的能力都得到了锻炼。