11-11 修改本章的综合实例，不再在文件中保存用户输入的命令，而是在每次程序结束前使用 boost 的 Serialization 程序库将当前状态保存到文件中，程序启动后再从文件中将状态恢复。

解：修改 Account 类如下：

```
//包含以简单文本格式实现存档的头文件
#include<boost/archive/text_oarchive.hpp>
#include<boost/archive/text_iarchive.hpp>
#include<boost/serialization/base_object.hpp>
class Account {                                          //账户类
    //…
    //Account 类中为以下函数增加了一个参数，其他成员与例 11-13 完全相同
    friend class boost::serialization::access;
```

```
    template<class Archive>
    void serialize(Archive & ar, const unsigned int version)
    {
        ar & id;
        ar & balance;
        ar & total;
    }
};
```

创建新类 CurrentState，并将原先所有文件中的 accounts 使用 cur.accounts 替换。

```
//包含以简单文本格式实现存档的头文件
#include<boost/archive/text_oarchive.hpp>
#include<boost/archive/text_iarchive.hpp>
#include<boost/serialization/base_object.hpp>
#include<boost/serialization/vector.hpp>

class CurrentState
{
friend class boost::serialization::access;
    template<class Archive>
    void serialize(Archive & ar, const unsigned int version)
    {
       //序列化基类信息
```

```
        ar & accounts;
}
public:
vector<Account * >accounts;
}
```

修改 main 函数如下：

```
CurrentState cur;
int main() {
    Date date(2008, 11, 1);                          //起始日期
    Controller controller(date);
    string cmdLine;
std::ifstream fin("accounts.txt");
    boost::archive::text_iarchive ia(fin);    //文本的输入归档类
    ia>>cur;

    while (!controller.isEnd()) {                    //从标准输入读入命令并执行,直到退出
        cout<<controller.getDate()<<"\tTotal: "<<Account::getTotal()<<"\tcommand>";
        string cmdLine;
        getline(cin, cmdLine);
        if (controller.runCommand(cmdLine))
            fileOut<<cmdLine<<endl;                  //将命令写入文件
```

```
    }

    std::ofstream fout("accounts.txt");          //把对象写到文件中
      boost::archive::text_oarchive oa(fout);
                                          //文本的输出归档类,使用一个 ostream 来构造
    oa<<cur;
    return 0;
}
```