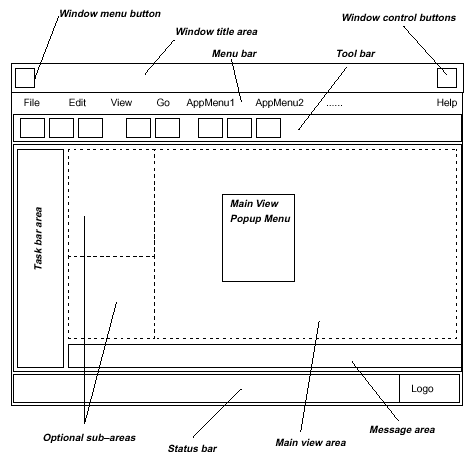
模块算法设计

测试用例确定每一个模块使用的数据结构

用户界面设计

* **程序描述**
* **功能**
* **性能**
* **输入项**
* **输出项**
* **算法**
* **流程逻辑**
* **接口**
* **存储分配**

界面：



一：流程图

二：数据库表图

三：表格（模块内方法、接口、局部数据------抽象定义）

方法：

实现功能

在该模块起到的作用

输入:<要求：给出本函数的输入参数的详细说明, 包括取值范围、含义，及参数间的控制、顺序、独立或依赖等关系。

属性

属性的详细信息：含义 数据类型 访问类型

算法与处理流程：

流程图

返回 :<要求：说明返回值的含义。

四：界面

五:数据库模块？

六：测试数据

* 单入单出的控制结构：是指每个模块内部均用顺序、选择、循环结构来描述。

结构体：

|  |  |
| --- | --- |
| 结构体名称： | 转账单 |
| 所含参数： | 1.userId  2.userName  3.TranType  4.RevId  5.RevName  6.RevCardId  7.TranCount  8.usercountCheck  9.info |
| 定义： | struct TranBill  {  string userId;  string username;  int tranType  string revId;  string revName;  string revCardId;  double tranCount;  double userCount;  }; |

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 描述方面 | 具体描述 | | | | | | | | | | | |
| 模块名称 | 选择转账方式 | | | | | | | | | | | |
| 功能 | 在选择转账方式之前的时候自动在转账单上录入转账方的ID和转账方的昵称形成转账单（1），转账方式里面有三种选择，从好友列表中选择朋友账户，输入收方账号和输入银行卡号的方式。 | | | | | | | | | | | |
| 性能 | 1. 时间特性要求：要求等待时间<0.5s； | | | | | | | | | | | |
| 输入项目 | 名称 | | | 类型 | |  | | | 数据来源 | | | 附注 |
| 账单 | | | 结构体 | | 参数传入 | | |  |
| 输出项目 | 输出项 | | | | 数据类型 | | | 输出形式 | | | 输出内容 | |
| 账单（携带转账方式） | | | | 结构体 | | | 参数返回 | | | 01 02 03 | |
| 参数定义 | | 参数 | 类型 | | | | 含义 | | | 所属 | | |
| TranType | Int | | | | 转账方式 | | | TranChoiceUI | | |
| userBill | struct TranBill | | | | 账单 | | | 传入参数 | | |
| 方法 | | OpenGetRecInfoUi(TranBill userBill,int TranType)  {  如果TranType==01  打开TranChoiceUi\_01界面  如果TranType==02  打开TranChoiceUi\_02界面  如果TranType==03  打开TranChoiceUi\_03界面  } | | | | | | | | | | |
| 算法 | | |  | | | | | | | | | |
| 接口 | | | 该模块调用其下属功能模块完成相应任务。 | | | | | | | | | |
| 存储分配 | | | 无 | | | | | | | | | |
| 测试要点 | | | 测试主要包括以下内容： | | | | | | | | | |

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 描述方面 | 具体描述 | | | | | | | | |
| 模块名称 | 收账方信息录入 | | | | | | | | |
| 功能 | 将转账单（2）作为参数传入，然后按照对应的转账方式来录入收方的信息。如果选择朋友账户，则从好友列表中选取选定的对象，然后将此好友关联的支付宝账户或银行卡账户填入转账单中生成转账单（3）。若选择收方账号或者银行卡支付，则录入收方账号或收方的银行卡信息，生成转账单（3） | | | | | | | | |
| 性能 | 时间特性要求：要求等待时间<0.5s； | | | | | | | | |
| 输入项目 | 名称 | 类型 | | | 数据来源 | | | 附注 | |
| 账单 | 结构体 | | | 参数传入 | | |  | |
| 转账方式 | 整数 | | | 参数传入 | | | 01 02 03 | |
| 输出项目 | 输出项 | | | 数据类型 | | | 输出内容 | | |
| 参数定义 | 参数名称 | | 类型 | | | 含义 | | | 所属 |
| userBill | | struct TranBill | | | 账单 | | | 参数传入 |
| friendName | | List<string> | | | 好友姓名list | | | GetFriend()方法 |
| friendId | | List<string> | | | 好友ID list | | | GetFriend()方法 |
| Userid | | string | | | 用户账号 | | | GetFriend()方法 |
| cardid | | string | | | 银行卡号 | | | getCardinfo() |
| recvname | | string | | | 银行卡姓名 | | | getCardinfo() |
| revid | | string | | | 用户账号 | | | getCardinfo() |
| 方法 | 1. 用户选择了转向好友，那么就涉及一些从数据库获取好友，   以及向好友显示在列表中的方法  list<string> GetFriend(string userid)  {   1. 以userid为索引，查询数据库表“UserFriends” 2. 生成查询userid好友的sql语句 3. 获取好友信息从数据库中获取好友信息   放入list中，姓名放入friendName  Id 放入friendId  }  Void showFriends(list<string> friendName,List<string> friendId)  {  将friendName 信息 与 friended 信息分别显示在  控件的list上  }   1. 用户选择了转向某人银行卡，就涉及了输入银行卡号以及姓名，以及相关的信息格式检查   Void getCardinfo(UserBill userbill)  {  获取recvname,cardid;    If 不为空、银行卡号格式符合  赋值：userbill. RevCardId = cardid;    If 携带信息不为空  赋值: userbill.RevName = recvname;  }   1. 如果选择了向指定帐号转账，则输入一个符合格式的ID:   Void getUserID();  数据库操作：暂缺 | | | | | | | | |
| 附加 | C++ list 学习：  [http://my.oschina.net/Tsybius2014/blog/292778?p={{currentPage+1}}](http://my.oschina.net/Tsybius2014/blog/292778?p=%7b%7bcurrentPage+1%7d%7d)  VC & mysql:  <http://zhidao.baidu.com/link?url>=  j8B5s2tPGUbR1\_8aDZ0WkkvxcFzPoCnBE5BxU9c0Xq5w3e\_TgJCQpb8DsXf-hu\_bmjUczWkzeTbTTudDz-43a | | | | | | | | |
| 算法 |  | | | | | | | | |
| 接口 | 该模块调用其下属功能模块完成相应任务。 | | | | | | | | |
| 存储分配 | 从数据库读取朋友信息 | | | | | | | | |
| 测试要点 | 测试主要包括以下内容： | | | | | | | | |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 描述方面 | 具体描述 | | | | | | | |
| 模块名称 | **输入转账金额及携带信息** | | | | | | | |
| 功能 | 将转账单（3）作为参数，然后在里面加上需要转的金额和一些附加的消息，生成转账单（4） | | | | | | | |
| 性能 | 1. 时间特性要求：要求等待时间<0.5s； 2. 精度要求精确到整型的个位。 | | | | | | | |
| 参数 | 参数名称 | | 类型 | | 含义 | | 所属 | |
| userbill | | struct TranBill | | 账单 | | 参数传入 | |
| Info | | string | | 携带信息 | | GetInfo() | |
| 方法： | Void GetInfo()  {   1. 判断转账金额格式是否正确 2. 将转账金额赋值结构体中的参数 3. 判断携带信息：   If（info!=null）  {  Userinfo.info = info;  }  } | | | | | | | |
| 输入项目 | 名称 | 数据类型 | | 输入方式 | | 数据来源 | | 附注 |
| 转账单 | 结构体 | |  | | 参参数传入数 | |  |
| 输出项目 | 输出项 | | 数据类型 | | 输出形式 | | 输出内容 | |
| 转账金额 | | 浮点数 | | 存储与结构体 | | 转账金额 | |
| 账单携带信息 | | 字符串型 | | 存储与结构体 | | 账单携带信息 | |
| 算法 |  | | | | | | | |
| 接口 | 该模块为上层登录模块提供服务，并调用其下属功能模块完成相应任务。 | | | | | | | |
| 存储分配 | 无 | | | | | | | |
| 测试要点 | 测试主要包括以下内容：  1.功能测试：数据库测试（即数据库内容增删改查等操作过程是否正确）  2.性能测试：连接速度测试（即时间特性要求）、压力测试（单机使用）  3.可用性测试：图形测试、内容测试、整体界面测试（即图形用户界面的显示是否人性化，是否易于操作员操作，以及容错机制）  4.兼容性测试：平台测试（可跨平台） | | | | | | | |

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 描述方面 | 具体描述 | | | | | | | | |
| 模块名称 | **确认余额** | | | | | | | | |
| 功能 | 接收到转账单（4）之后，读取账号信息。用转账单上的转账金额和账号余额作比较，若大于或等于账号余额则返回true，并且生成包括余额信息的转账单（5），若小于账户余额则返回false，并且生成显示余额不足的账单。 | | | | | | | | |
| 输入项目 | 名称 | 数据类型 | | 输入方式 | | 数据来源 | | | 附注 |
| 转账方昵称 | 字符串型 | |  | | 系统自带数据库中的表 | | |  |
| 转账方姓名 | 字符串型 | | 转账方昵称或者用户姓名 | |  | | |  |
| 用户ID | 字符串型 | | 上一个模块传入 | |  | | | 用于查询余额 |
| 输出项目 | 输入项 | | 数据类型 | | 输出形式 | | | 输出内容 | |
| 转账金额 | | 浮点数 | |  | | |  | |
| 账单携带信息 | | 字符串型 | |  | | |  | |
| 参数 | 参数名称 | 类型 | | | 含义 | | 所属 | | |
|  | userBill | struct TranBill | | | 账单 | |  | | |
|  | userid | String | | | 用户ID | | 结构体tranBill/ checkCount() | | |
|  | tranCount | double | | | 转账金额 | | 结构体tranBill/ checkCount() | | |
|  | sql | string | | | 数据库操作语句 | | getUserCount() | | |
|  | leftCount | double | | | 账户真实余额 | | checkCount() | | |
|  |  | | | | | | | | |
|  |  | | | | | | | | |
| 方法 | Double getUserCount(string id)  {   1. 生成sql语句----用于检索用户userid的信息 2. 打开数据库 3. 执行sql语句 4. 获取结果，并从结果中获得用户余额 5. 返回用户余额   }  Bool checkCount(string userid,double tranCount)  {  double leftCount=0.0;  leftCount = getUserCount(userid);  if(tranCount<=eftCount)  {  return false;  }  else  {  return true;  }  } | | | | | | | | |
| 算法 |  | | | | | | | | |
| 接口 | 该模块为上层登录模块提供服务，并调用其下属功能模块完成相应任务。 | | | | | | | | |
| 存储分配 | 获取该用户的可用余额 | | | | | | | | |
| 测试要点 | 测试主要包括以下内容：  1.功能测试：数据库测试（即数据库内容增删改查等操作过程是否正确）  2.性能测试：连接速度测试（即时间特性要求）、压力测试（单机使用）  3.可用性测试：图形测试、内容测试、整体界面测试（即图形用户界面的显示是否人性化，是否易于操作员操作，以及容错机制）  4.兼容性测试：平台测试（可跨平台） | | | | | | | | |

////////////////////////////////////////////////

丢了一个转账支付确认

//////////////////////////////////////////////////

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 描述方面 | 具体描述 | | | | | | | | | | |
| 模块名称 | 支付密码确认、更改余额 | | | | | | | | | | |
| 功能 | 收到账单（5）之后进行密码确认，若密码正确则读取账号信息然后更改余额然后生成正式的转账单，否则生成无效账单。 | | | | | | | | | | |
| 参数 | 参数名称 | | | 类型 | | | 含义 | | | | 所属 |
| **userid** | | | **string** | | | 用户ID | | | | **changeUserCount()** |
| **tranCount** | | | **double** | | | 转账金额 | | | | **changeUserCount()** |
| **sql** | | | **string** | | | 数据库语句 | | | | **changeUserCount()** |
| **paypsw** | | | **string** | | | 支付密码 | | | | **ChenckPayPsw()** |
| **checkTimes** | | | **Static int** | | | 支付密码错误次数 | | | | **ChenckPayPsw()** |
| **userpaypsw** | | | **string** | | | 数据库读取的用户支付密码 | | | | **ChenckPayPsw()** |
| 方法 | Bool changeUserCount(string userid,double tranCount)  {  try  {   1. 生成用于更改余额的数据库语句sql 2. 打开数据库 3. 执行SQL语句   }catch(Exception e)  {  错误用异常处理  return fasle;  }  return true;  }  Bool checkPayPsw(string userid,string paypsw)  {  userpaypsw = getPayPsw(string userid);  If userpaypsw== paypsw;  Return true;  If userpaypsw!= paypsw  If checkTimes = 3;  Return false;  Return false;  }  String getPayPsw(string userid)  {  数据库获取userid的支付密码  这里就不详细说了  }  Void showChecKResult(bool)  {  显示支付结果  成功或者失败  暂时先不具体写了  } | | | | | | | | | | |
| 输入项目 | 名称 | 数据类型 | | | 输入方式 | | | 数据来源 | | 附注 | |
| 转账金额 | | 浮点数 | | | 上一模块传入 | | |  | | |
| 输出项目 | 输入项 | | 数据类型 | | | 输出形式 | | | 输出内容 | | |
| 更改余额结果 | | True|false | | |  | | |  | | |
|  | |  | | |  | | |  | | |
| 算法 |  | | | | | | | | | | |
| 程序逻辑 |  | | | | | | | | | | |
| 接口 | 该模块为上层登录模块提供服务，并调用其下属功能模块完成相应任务。 | | | | | | | | | | |
| 存储分配 | 。获取该用户的可用余额 | | | | | | | | | | |
| 限制条件 | 无 | | | | | | | | | | |
| 测试要点 | 测试主要包括以下内容：  1.功能测试：数据库测试（即数据库内容增删改查等操作过程是否正确）  2.性能测试：连接速度测试（即时间特性要求）、压力测试（单机使用）  3.可用性测试：图形测试、内容测试、整体界面测试（即图形用户界面的显示是否人性化，是否易于操作员操作，以及容错机制）  4.兼容性测试：平台测试（可跨平台） | | | | | | | | | | |

**模块方法分层：**





数据库表：

数据库名：trandatabase

User:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段名 | 类型 | 可否为空 | 长度 | 含义 |
| uid | Int | 否 | 15 | 主键 |
| userid | Varchar | 否 | 20 | 用户账户 |
| password | Varchar | 否 | 20 | 用户密码 |
| username | Varchar | 否 | 20 | 用户昵称 |
| usercount | double | 否 | 15 | 用户余额 |
| portraitid | Int | 否 | 15 | 外键，头像id |

Head portrait：

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 字段名 | 类型 | 可否为空 | 长度 | 键引用 | 含义 |
| portraitid | Int | 否 | 15 | 主键 |  |
| userid | Varchar | 否 | 20 | 外键 | User 表中的userid |
| portrait | mediumBlob | 否 |  | 外键 | 主键  用户头像、朋友图片 |

UserFriends:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 字段名 | 类型 | 可否为空 | 长度 | 键引用 | 含义 |
| Ufid | Int | 否 | 15 | 主键 |  |
| friendId | varchar | 否 | 20 |  | 好友id、账号 |
| Username | Varchar | 否 | 20 |  | 好友昵称 |
| portraitid | Int | 否 | 15 | 外键 | 头像 |

TranBill:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 字段名 | 类型 | 可否为空 | 长度 | 键引用 | 含义 |
| Billid | Int | 否 | 15 |  | 转账单号 |
| Uid | Int | 否 | 15 | 外键 | User表的主键id |
| Ufid | Int | 否 | 15 | 外键 | UserFrirend表的主键id |
| Trantypeid | Int | 否 | 15 | 外键 | 转账方式 |
| Trancount | Double | 否 | 10 |  | 转账金额 |

TranType:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段名 | 类型 | 可否为空 | 长度 | 含义 |
| trantypeid | Int | 否 | 15 | 主键 |
| Trantype | Varchar | 否 | 10 | 转账方式 |

界面：

TranChoice:

跳转Button

使用者用户ID，显示

应用名称

关闭应用Button

03.向银行卡转账

02.向指定账号转账

01.向朋友转账

转账方式选择：

User: xxxxxxxxxx

TranApp

User: xxxxxxxxxx

转账方式选择：

anApp

转账方式01，界面

TranChoiceUi\_01



转账方式02，界面：



转账方式03，界面：



转账余额、携带信息输入：



余额不足提示窗口：



转账确认界面：

