小型收银台系统模拟实现

【本节目标】

- 什么是收银台
- 传统收银台缺陷
- 核心功能
- 数据库设计
- 实现
- 扩展

1. 什么是收银台

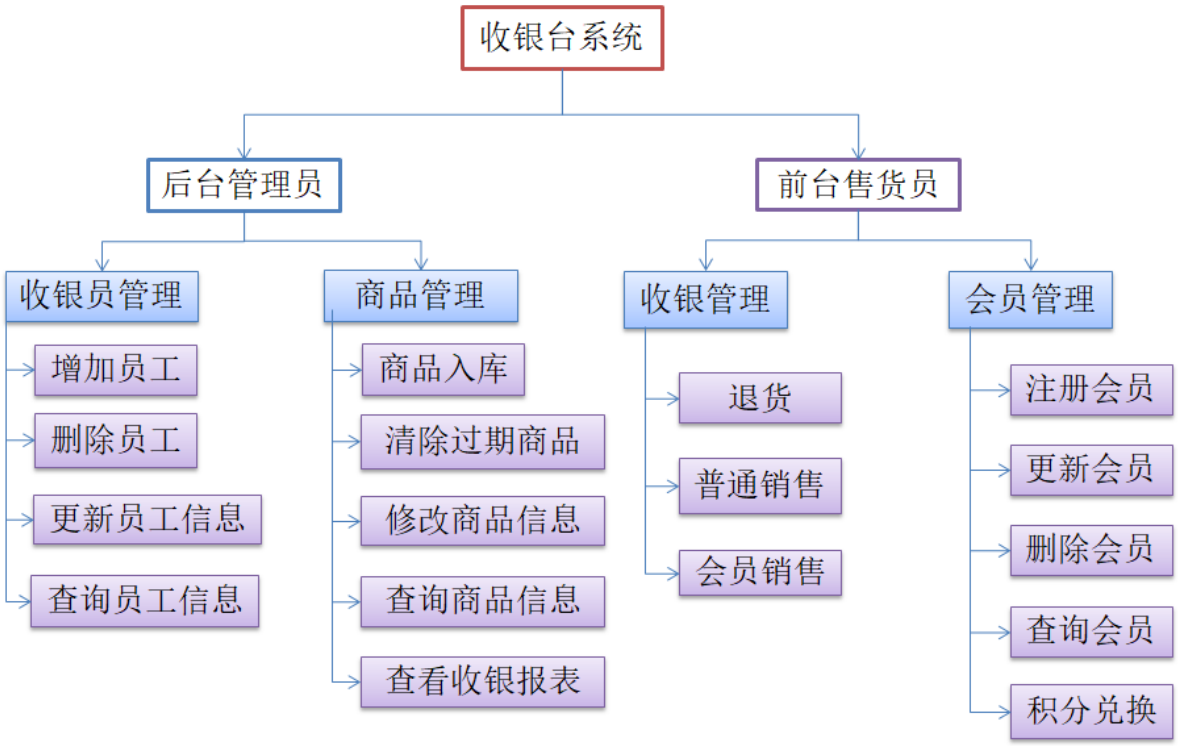
> 收银台作为商场超市所必不可少的配套设施，越来越多的被客户所关注。**收银台俗称付款处，是顾客付款交易的地方，也是顾客在商店最后停留的地方。**
>
> 收银台除了收银这一主要用途外，将在吸引顾客视线的同时发挥出特殊功效。事实上，**收银作业不只是单纯地为顾客提供结账服务而已，收银员收款工作完成后也并不代表卖场的销售行为就此结束，这其中还包括了对顾客的礼仪态度。**

2. 传统收银台缺陷

> 
>
> 1. 收款结算速度慢，容易出现营业差错，
> 2. 不宜进行商品调价，盘点效率底
> 3. 用户体验不好
>
> 收银台的优点：快捷方便，节省大量人力成本，不容易出错，能够快速反馈出商品的详细信息。
>
> 因此：开发这个系统可以方便快捷地查出顾客结帐情况，商品信息情况，每天的售货情况，方便了对超市商品管理、人员管理，大大提高了超市的售货速度。进而加速了社会的发展速度，提高了人民的生活水平。

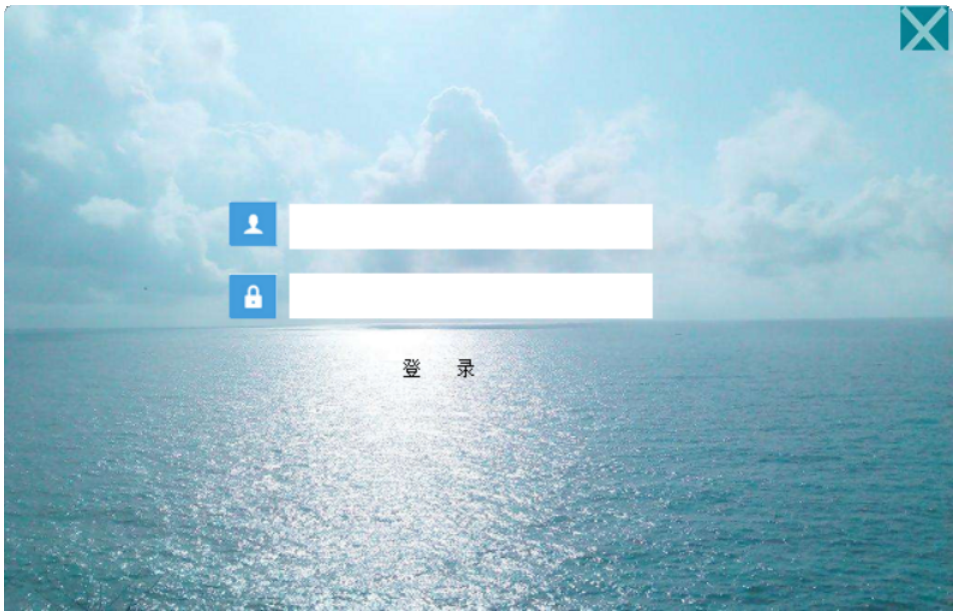
3. 核心功能

> 整体框架：



1. 登录模块

   后台管理员和前台售货员需要根据自己的用户名以及密码进行登录。



   用户输入名户名以及密码后，根据不同身份，显示不同界面，用户进行其相应操作。

   管理员界面：

   售货员界面：

2. 后台管理员
   - 员工操作
     - 查询员工基本信息
     - 添加新员工
     - 员工离职后，删除员工信息

        - 员工信息变更时，更新员工信息，比如：更新员工薪资

    - 商品操作
        - 按照条件查询商品的信息
        - 商品入库
        - 过期商品的删除
        - 商品信息更新，比如：价格发生变动
        - 按照日期查询商品销售情况

3. 售货员模块
    - 售货
        - 录入商品信息
        - 出售：会员出售和普通用户出售
        - 如果客户不满意，退货
    - 会员管理
        - 增加会员
        - 删除会员
        - 查询会员
        - 更新会员信息

4. 数据库设计

为简单起见，本项目设计到以下几种表格：

1. 职工表

```
create table Employee(
id int,                         -- 员工编号
name varchar(20),               -- 员工名字
gender varchar(3),              -- 员工性别
birthday Date,                  -- 生日
password varchar(20),           -- 员工密码
position varchar(10),           -- 员工职位
telphone varchar(11),           -- 联系方式
salary double(9,2)              -- 联系方式
);
```

2. 商品表

```
create table Goods(
GoodsID int,                    -- 商品编号
GoodsName varchar(20),          -- 商品名称
GoodsType varchar(20),          -- 商品类别：水果、烟酒、日常用品、副食等
ProductDate DATE,               -- 商品生产日期
DeadDate DATE,                  -- 商品过期日期
Price double(9,2),              -- 商品价格
Unit  varchar(3),               -- 计量单位
Inventory int,                  -- 库存量：商品剩余数量
AlarmValye int                  -- 报警值：低于该值时，应提醒管理员进货
);
```

3. 售货记录表

```
create table SellRecord(
GoodsName varchar(20),    -- 商品名称
GoodsPrice double(9, 2),  -- 商品价格
Amount int,               -- 售出数量
Unit varchar(3),          -- 计量单位
SellTime Date,            -- 售出时间
Operator varchar(20);    -- 售货员
```

4. 会员表

```
create table Member(
    Name varchar(20),    -- 会员名字
    Telphone varchar(20), -- 会员电话
    Level, int            -- 会员级别
    Score, int            -- 会员积分
    Time,  Date           -- 办理会员日期
);
```

5. 接口

1. 界面

   使用duilib界面库。

   - 管理员操作界面

- 售货员操作界面

2. 数据库操作类封装

```
class MySQL
```

```
{
public:
    MySQL();
    bool ConnectMySql(const char* host,        // 主机名称
                      const char* user,        // 用户名
                      const char* passward,    // 密码
                      const char* dbName,      // 数据库名
                      int port=3306);          // 端口号：默认为3306
    ~MySQL();
    bool Insert(const string& strSQL);
    bool Delete(const string& strSQL);
    bool Update(const string& strSQL);
    size_t GetCount(const string& strSQL);
    vector<vector<string>> Select(const string& strSQL);

    // 切换数据库
    bool SelectDB(const string& daName);

private:
    MYSQL* _mySql;   // mysql连接的实例对象
    std::string _dbName;
    vector<string> _tables;
};


#pragma comment(lib, "ws2_32.lib")
#pragma comment(lib, "libmysql.lib")


MySQL::MySQL()
{
    // 初始化mySql
    _mySql = mysql_init(nullptr);
}

bool MySQL::ConnectMySql(const char* host, const char* user, const char* passward,
                         const char* dbName, int port)
{
    // 连接mySql数据库
    if (!mysql_real_connect(_mySql, host, user, passward, dbName, port,NULL,0))
    {
        return false;
    }

    /*
    c++连接mysql时，比如查询语句中含有中文，或者得到结果中含有中文，经常出现编译出错或乱码问题。
    VS编译器默认使用gbk编码。
    如果将mysql设置为utf-8编码，则需要先将c++中的各种中文字符串转为utf-8编码输入mysql，得到的结果为utf-8编码，需要转为gbk才能正常显示。转来转去很麻烦。
    */
    mysql_query(_mySql, "set names 'gbk'");

    return true;
```

```
}

bool MySQL::SelectDB(const string& dbName)
{
    if (mysql_select_db(_mySql, dbName.c_str()))
    {
        return false;
    }

    return true;
}

bool MySQL::Insert(const string& strSql)
{
    // 执行sql语句
    if (mysql_query(_mySql, strSql.c_str()))
    {
        return false;
    }

    return true;
}

bool MySQL::Update(const string& strSQL)
{
    // 执行sql语句
    if (mysql_query(_mySql, strSQL.c_str()))
    {
        return false;
    }

    return true;
}

vector<vector<string>> MySQL::Select(const string& sql)
{
    vector<vector<string>> vRet;
    // 指定指定SQL语句
    if (mysql_query(_mySql, sql.c_str()))
    {
        string vsRet(mysql_error(_mySql));
        return vRet;
    }

    // 检索完整的数据集到客户端
    MYSQL_RES *res = mysql_store_result(_mySql);
    if (res == NULL)
    {
        return vRet;
    }

    // 用来保存结果集中行的信息

    MYSQL_ROW rows;
```

```
    // 结果集中总共有多少行数据
    int num_fields = mysql_num_fields(res);
    while (rows = mysql_fetch_row(res))
    {
        int i = 0;
        vector<string> vItem;
        vItem.resize(num_fields);
        for (i = 0; i < num_fields; i++)
        {
            vItem[i] = rows[i];
        }
        vRet.push_back(vItem);
    }

    const char* str = mysql_error(_mySql);
    mysql_free_result(res);

    return vRet;
}

size_t MySQL::GetCount(const string& strSQL)
{
    // 指定指定SQL语句
    if (mysql_query(_mySql, strSQL.c_str()))
    {
        return 0;
    }

    // 检索完整的数据集到客户端
    MYSQL_RES *res = mysql_store_result(_mySql);
    if (res == NULL)
    {
        return 0;
    }

    return mysql_num_fields(res);
}


bool MySQL::Delete(const string& strSQL)
{
    // 执行sql语句
    if (mysql_query(_mySql, strSQL.c_str()))
    {
        return false;
    }

    return true;
}

MySQL::~MySQL()

{
```

```
    mysql_close(_mySql);
}
```

本项目没有什么难度，就是主要就是数据库操作。

6. 扩展

> 1. 完成商品的入口操作
> 2. 完成会员管理
> 3. 多人同时进行操作时，如何保证数据安全问题