|  |
| --- |
|  |
| **同济大学2016年数学建模竞赛A题** |
| 去库存问题 |
|  |
|  |
|  |

|  |
| --- |
|  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 姓名 | 学号 | 学院 | 专业 | 联系方式 |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

流通库存决策探讨

摘要

货物的流通决策作为物流产业中的重要部分，其库存管理决策方案关系到运输效率、储存成本等各方面问题。在得到实际流通网络后，如何做出适当的流通环节假设，确定相关参数的定义、推导变量间的关系，运用已知网络计算各变量表达式，建立模型以获得最小总流通库存，用实际流通状况数据与构建模型求得的理想结果比较分析验证模型的优越性及最终定量给出货物流通方案成为本问题的关键。

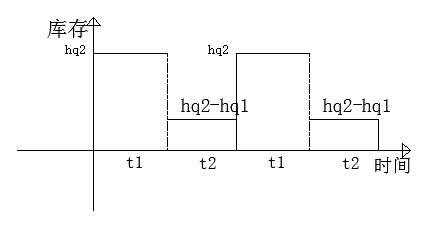
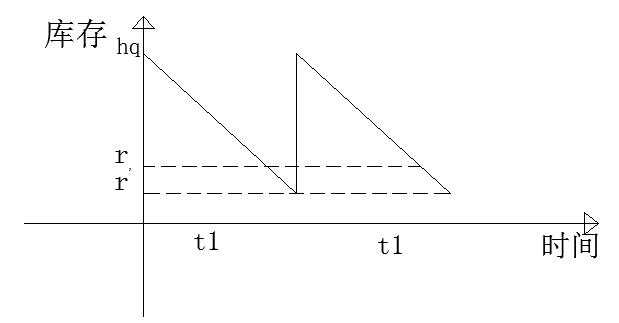
首先，研究小组对某种产品在网络中的流通环节进行适当简化假设，提取出模型中相关参数进行定义与表达。其中我们定义总流通库存为一次订货周期内除末端收货商所有分销商的库存量总和；进而找出生产能力一定时，总流通库存量与到货率的关系。

我们采用从最末端收货商依次根据需求向上级补订货的思想，用递推的方法将网络中各个环节供应商的安全库存r、单次订单补货量q以及流通库存k用末端收货商的和表示出来，通过建立模型给出安全库存与单次补货量的计算方法使表示出的库存量达到最小。

接下来，为了确定最优的末端收货商的安全库存与每次订单的补货量, 在末端收货模型中，研究小组引进了经济订货批量模型EOQ：通过实际情况中成本条件的限制以及调查相关参数的具体值求解出满足实际情况的安全库存与每次订单的补货量，代入整体流通网络中求出各级各公司的r和q,并最终求出总体最小库存。

随后，小组将题目中给出的网络进行分析，用将流通数据网络导入C++程序中，编程求出网络关系，找出各公司在网络中所处的级数和对应的末端收货商数。通过C++编程先根据Excel流转数据表给出的各公司实际单次订单补货量求出实际的流通库存；再采用模型编程分析得到的单次订单补货量q以及流通库存k，在流通网络中计算出最优的流通库存；最后对实际流通库存与改进建议的最优流通库存进行比较，检验模型结果的正确性。

更进一步，模型中考虑到某公司对应的多个子公司向其要求补货的时间间隔是随机的，且当其要求补货的子公司数较多时，呈阶梯状递减的公司库存量可以简化为线性递减即商品持续消耗的库存模型；但在实际情况中，会出现公司只对应一两个下级公司的情况，据此提出改进方案，通过统计各类公司所占比例根据权重确定流通网络的总库存。



关键字：（Q，R）策略、EOQ模型、数据分析、流通库存

一 问题重述

近年来，我国物流业已转型升级至新的阶段，产业规模快速增长、服务能力显著提升、基础设施日趋完善，为进一步提高物流流通环节的效率，减少流通网络中的总库存量已成为当务之急，为此，研究小组着手考虑各流通级做出的库存管理决策方案。

一般来说，库存控制采用的方法是订货点控制策略，基本策略有以下四种：

1. 连续性检查的固定订货量、固定订货点（Q，R）策略；
2. 连续性检查的固定订货点、最大库存（s,S）策略；
3. 周期性检查（T,S）策略；
4. 综合库存策略，（T,s,S）策。

本题中所述的方法为第一种方法。

在某行业的货物流通网络中，货物供应商通过各公司，进而向下级子公司直至零售商发行商品，某个发货商有可能同时在其它订单中也作为收货商。在只考虑某种商品的流通情况时，货物流通网络呈树形结构，在网络中的级数越靠上供应商数越少，为保证下级供应商的需求，相应的其库存量应适当增大；当生产能力一定时，上级供应商要向多个下级供货，下级发来的订单未必能全部得到满足，因此我们将实际供应商的消费需求和库存量作为市场需求预测和库存补货的解决方法。在已知末端收货商的到货率的情况下，根据供应商所处的货物流通网络确定该公司的安全库存r和每次补货的订单量q。即在平日里，各公司都有一个初始库存，假设公司的库存量一旦小于某个r值就会立即向其某个其对应的上级下订单补货，订单量为常数q。为了解决供应与需求在时间、空间上的不协调，人们在供需之间加入了存储这一环节，从而产生了物资存储问题，批量订货点模型从更一般的角度来研究问题，大大提高了存储模型的现实可用。

在上述决策模式中，站在该行业供应商的角度，在生产能力一定的情况下，考虑库存与到货率的关系。当满足相应到货率条件时，确定安全库存r与每次订货量q，使流通链中总库存最小。这里定义总流通库存为一次订货周期内除末端收货商所有分销商的库存量总和。

二 符号说明

Q ——一定时间内商品生产总量

m ——各流通级公司初始库存总和

K ——总流通库存

H ——末端收货商拥有的商品量

q ——每次补货的订单量

h ——到货率

——某供应商库存量

——该公司每次补货的订单量

——该公司的安全库存

第i级供应商（收货商）个数

——某级供应商（收货商）的到货率

——末端收货商的单次订单量

——末端收货商的安全库存

——末端收货商的个数

三 模型假设

3.1对于基于某级公司总体分析建模的假设：

1. 末端收货商的销售速率一定（即库存降低速率一定）且在整个时间段内保持一致；
2. 在同一产品的一条树状生产链中，一级公司数量远小于二级公司数量，二级公司数量远小于三级公司数量，以此类推，进而各级公司库存——时间函数均可视为与末端收货商一样；
3. 处于同一级的公司其需求量、订单量、安全库存货物实到量均相同；
4. 除末端收货商外，各级公司所设置的安全库存为其对应的下级子公司的最大值，即至少能给任意一个下级公司发货；
5. 末端收货商的安全库存采用EOQ模型中的假设：=销售速率\*订单时间；
6. 假设安全库存的设置是每个公司至少能给一个下级公司发货而不至断货。

3.2对基于个体公司的分析建模假设：

1. 末端收货商的销售速率一定（即库存降低速率一定）且在整个时间段内保持一致；
2. 在同一产品的一条树状生产链中，一级公司数量远小于二级公司数量，二级公司数量远小于三级公司数量，以此类推，进而各级公司库存——时间函数均可视为与末端收货商一样；
3. 对于某个公司来说，其需求量、订单量、安全货物库存与其所对应的末端子公司个数有关；
4. 除末端收货商外，各级公司所设置的安全库存为其对应的下级子公司的最大值，即至少能给任意一个下级公司发货；
5. 末端收货商的安全库存采用EOQ模型中的假设：=销售速率\*订单时间；
6. 假设安全库存的设置是每个公司至少能给一个下级公司发货而不至断货。

四 问题分析

首先如何理解题目中所说的网状结构，即在某条销售运输链中，某公司可能是处于供应链中的第三级（将题目给出的第一级FAC视为第一级公司，下一级公司为第二级，再下一级公司为第三级，以此类推），但在另一条链中，改公司可能处于第二级；但是，正如题目中所说的，各级公司之间是有明确的上下级关系的，同时，在对excel流转数据及网状关系进行数据分析后发现，同一产品的情况下，上下级的相对关系不会发生改变，即同一产品的一条从第一级公司开始流通的流通链，可视为树状流通链结构。

题目中给出，平日里，各公司都会有一个初始库存，一旦库存量少于某个r值便向某个上级订货，该订货策略为典型的定量订货策略，其原理为：预先确定一个订货点，在日常管理中连续不断地监控库存水平，当库存水平降低至订货点时，发出订货通知、订货。这种库存控制方法必须连续不断地检查库存物品的库存数量，所以又称为连续库存检查控制法。

我们的主要目的是在尚未提高生产能力之前提高到货率，降低流通库存。这几者之间经过初步的推算，呈现如此关系：

Q+m=K+H

（Q——总生产能力；

M——总初始库存；

K——总流通库存；

H——最末端收货商的到货量）

因此可以看到，当Q不变时，流通库存越少，最后的末端收货商的到货量越大，因此，我们的关键任务即建模目标是：如何用r、q量化分析K、H等，从而获得一个最优的r、q。

题目中给出r和q均为一个常量，从实际情况上考虑，不同级的r和q应该是不同的。因此，题目中要确定一个最优的r和q使流通库存最小，实际上是要确定下来r和q与该公司的级数、下级子公司数目等之间的关系。因此，我们分别从公司个体和各级公司总体两个角度分析建模，得出使得总流通库存达到最优时的r和q与该公司的级数、下级子公司数目等之间的定量关系。

五 模型前的准备

5,1 数据的处理

5.1.1 流通网络可视化

将题目中表格1给出的流通网络通过C++程序实现网络结构的整理与可视化，效果如图1–1所示。具体实现代码见附件一。

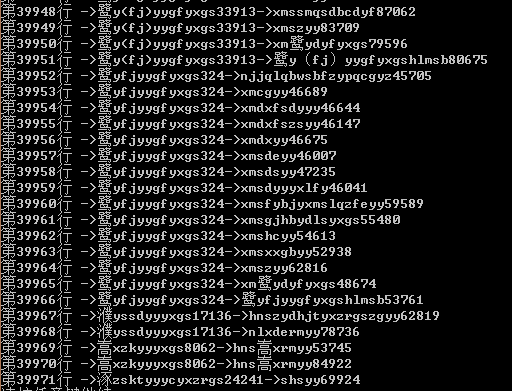


图1-1

5.1.2 网状流通链结构的算法研究

根据题目提供的网状结构，考虑采用链表结构构造该网状结构，以下将给出完整的网状流通链算法，该算法中含有计算各流通环节公司的上下级关系、订单需求量、安全库存的算法。

算法实现功能条件：仅考虑一种商品的全部网状结构销售过程，其他四种产品的处理均与之类似。

算法实现功能需求：根据公司名称、当前读取流通链，进而确定：

1. 当前读取公司的上级公司名称、下级公司名称；
2. 读取公司向其上级公司提交的订单数量、读取公司供给给下级的订单数量和；

为满足以上功能需求，考虑采用c++链表结构实现以上功能。

为表明各公司间的网状供给关系、及各供给关系中详细的订单量等相关属性，构建树状链表结构p1，再通过不同公司流通链树状结构的交叉表达出完整的网状结构。

5.1.2.1链表基本结构的构建

struct pl{

char name[30];

int l;

long q,r,q1;

int pre;

int sub; };

其中：

name——当前公司名称；

l——该公司在货物流通过程中所在级数；

q——对该公司的建议订单量；

r——对该公司的建议安全库存；

q1——该公司当前实际收货量；

pre——该流通链中该公司的上级公司编号；

sub——经流该公司全部链条的末端公司总个数。

则在读入全部数据的过程中，将所有公司按照读取数据赋予其对应编号，第n个公司的全部信息均记录在结构体p[n]中。

5.1.2.2 树状结构的表达

在确定了以上的链表基本结构后，为了保证pre属性能够找到该流通链中该公司上层公司的位置、sub属性能够记录其末端公司数，以确保确定整个网状结构关系，通过引入x数组x[i]记录树状结构：

for(int i=2;i<=15000;i++) //按行读取数据

{

for (int j = 1; j <= 7; j++) //该行从左至右按列读取各个数据

{

line[j] = my\_csv.get\_data(j, i);//将当前扫描行的第j列数据记入line[j]数组中

if(line[j][0]) //统计该行最低级子公司的级数，记为num，此级数即为次链条上物流流通结束之时的末端公司所在级数

{

num=j;

}

}

for (int j = 1; j <= 7; j++)

{

if(line[j][0]!='\0') //若读入的第i行第j列数据不为空，即读入位置存在公司时，则进入判断；

{

p[n].sub=0;

head[j] = line[j];

strcpy(p[n].name,line[j]); //将当前读入公司的名称记入其结构体的name中；

p[n].q=p[n].r=p[n].q1=0;

p[n].l=j; //记录该公司级数为j，记入其结构体级数l中；

x[j]=n; //当前读取的公司编号赋值给x[j];

p[n].pre=x[j-1]; //再上一次循环中获得的该公司上级公司编号x[j-1]记入该公司结构体的pre,即记录其上级公司编号，在此过程中即实现了网状结构的上级搜索功能，可通过程序得知各公司的上级对应的全部公司；

if(j==num) //判断是否为最末级子公司，是则进入执行，以下程序所需实现的功能即为：一旦查询到一个流通链的末端公司，则对该链条上所有该末端公司提交过订单的全部非末端公司的sub值（即其末端公司数量）进行加1处理；

{

int nn=p[n].pre; //引入可变中间变量nn，赋值为其上级公司编号；

while(nn!=0){p[nn].sub++;nn=p[nn].pre;}

//判断当前公司是否为第一级公司，若不是第一级公司，则对其上级公司的sub（即末端子公司数）加1；接着继续循环动作，将当前公司更新为其上级公司继续进行判断，同样的，若不是第一级公司，则对其上级公司的sub加1；直至循环至第一级公司，即不在存在上级公司时，跳出循环。此过程将记录下所有非末级公司的末端公司总数，也是完整算法中逻辑的难点。

}

n++;

}

}

}

通过以上的程序实现，完整的在c++的链表结构中保留了原网状结构中所有的上下级对应关系，并保留了各公司的末端公司数量的属性值。

六 模型的建立与求解

6.1问题一：总库存与末端到货率的关系

6.1.1问题一的主要思想

站在生产商的角度考虑，当生产能力一定时，探究库存与到货率之间的关系。研究过程关心的只是末端收货商的到货率，所以需要整体考虑货物流通网络，确定流通总库存与末端到货率的关系。

6.1.2相关参数的量化

研究小组在考虑整体货物流通网络时，假设一段时间内生厂商生产能力为Q，且生产能力在这段时间内保持为~~某~~这一定值。因为初始库存值的大小是随时间变化的随机值，并且初始库存对末A端收货商的到货率不产生影响，可以设流通链中所有环节公司有各自的初始库存总和为m。调取该产品所在的物流网，统计位于流通链末端的收货商个数以及整个网络中各个经销商（不包括最末端）的库存量之和为K，相应的末端收货商实际拥有的货物量为H。根据题意，规定末端收货商的单次补货定单量为常数q，定义末端收货商到货率为h。将末端到货率h用Q、m、、K、H表示。

6.1.3问题一的求解与分析

研究小组发现在商品的流通环节中，生产商品总数与最初各个公司初始库存之和等于经销商总流通库存量与末端收货商商品数之和，即Q+m=K+H。在这个式子中，Q、m为假设的已知量，K为需要统计计算的参数。当在具体的流通网络中获得末端收货商个数与单次订单量的值时，可以表示出末端的到货率即末端实际拥有商品数量与末端所需的商品数量之和的比值。

公式表示为：

由此可知，当生产能力一定时，流通的库存量越大，末端对应的到货率越小，二者成反比关系。

6.2问题二：基于各级公司的总体层面模型的建立

6.2.1建立该的主要思想：

将各级的需求量、各级的货物实到量作为一个整体去考虑，并不细致考虑各级公司之间的具体差别，而是将各级公司的需求量、货物实到量等“平均化”处理。根据实际需求推导出各级供应商的安全库存量与单次进货数.找出流通库存与二者的关系，在满足货物需求的前提下通过优化定义安全库存与单次进货数，使所有分销商的库存流通量总和最小。

6.2.2相关参数的量化

末端收货商单个的需求： ——为该系统的一个自变量

末端收货商的安全库存： ——为该系统的一个自变量

第L级对应的末端收货商数量：

末端收货商总需求： ——该参数为末端收货商总的需求，其中为末端收货商数量，每一个收货商的需求是，因此末端收货商的这一级的总需求为

末端收货商总的到货量： ——最终总到货量等于其需求乘以该级的缺货率。

各级公司到货量： ——在以各级整体为考虑对象的模型中，各级的到货量都是相同的

第L级的需求量（即订单总数）：()/

——这一级所有公司实到的货物量之和与上一级的缺货率之比为这一级的订单总数

第L级的安全库存： （/）\*

——正如假设中所说,安全库存的设置是为了保证每个公司在向上级供应商要求补货期间至少能给一个下级公司发货而不至断货，第L级的安全库存至少留下能给一个下级收货商供货的量，平均与迭代计算下来就是相当于该级一个公司的下级公司对应了几个末级收货商再乘以末端收货商的平均需求量即单次订单量。

6.2.3模型一的建立与求解

按照上面所说的假设，则各级公司的库存—时间函数均为线性关系，可表示为：

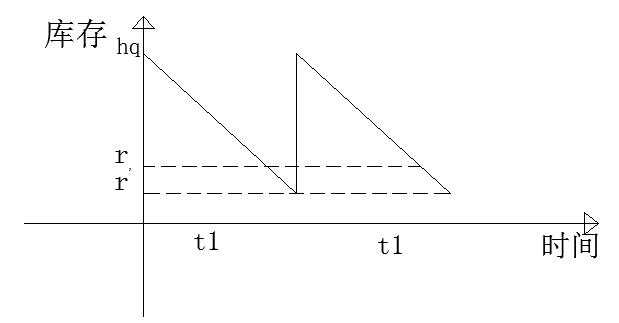


图1-2

定义公司流通库存为其一个周期内的平均库存，即为一个周期内其围城的面积除以该周期时间为：

K=hq/2+r’

在这里，假设库存消耗速率不变，从向上级下订单到收到订单的时间不变，则r-r’可以视为一个常数，表示公司达到安全库存后在向上级要求补货期间，由于商品的持续消耗导致的库存的减少量。这个常数可以在实际情况中调查计算得出，所以上式可以写为：

K=hq/2+r-c

那么，从整体上来考虑，总体的流通库存则为：

-c=-C

这样则建立起了流通库存总量与q和r之间的关系，具体各级之间的流通关系，需求、实到量与安全库存（见表1-1）：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 级数 | 该级公司数量 | 该级总需求 | 该级总实到货物量 | 该级各个公司安全库存 |
| x（末端） |  |  |  |  |
| x-1 |  |  |  |  |
| x-2 |  |  |  |  |
| … |  |  |  |  |
| 1 |  |  |  |  |

表1-1

（注：该表中末端收货商为第X级）

由此建立起立总流通库存K和各级公司总体的q和r的关系，而各级公司总体的q和r又与末端收获商的q和r建立起了关系，进而则可以用末端收货商的和来表示总的流通库存，则可以通过调整、，以期得到一个最优的流通库存量。

通过上表，可以得出各级最优的订单量与安全库存与末端收货商订单量之间的关系，但是并无法直接得到末端收获商的安全库存，与末端收货商订单量，这两个参数借用EOQ模型来求解：

EOQ模型的基本假设为：

1）单位时间内的需求量不变，即需求速率R均匀且为常量

2）订货提前期TK不变

3）每次订货费用相同，与订货量无关

4）没有数量折扣

5）保管费用与库存量量成正比

6）缺货率不会超过题目中所给出的缺货率，每次所定的货物都能一次到达

7）单一品种

在EOQ模型假设的情况下，末端收货商库存消耗速率不变，其订货点r应为：订单时间\*库存消耗速率（订单时间指的是，需求方从下订单开始到需求方收到货物的时间）在该模型中，库存消耗速率与订单时间均可由市场调查统计得到。

同时，在EOQ模型中，末端收获商的最佳订货批量，即本题中的确定主要考虑的是成本问题，如图1 -3、图1-4 所示：

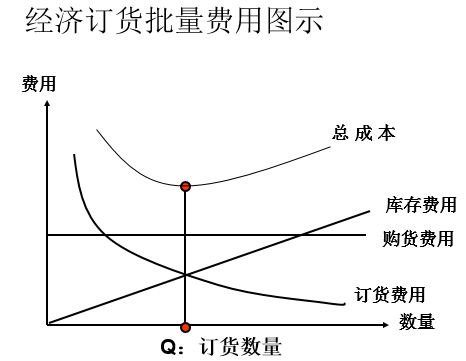
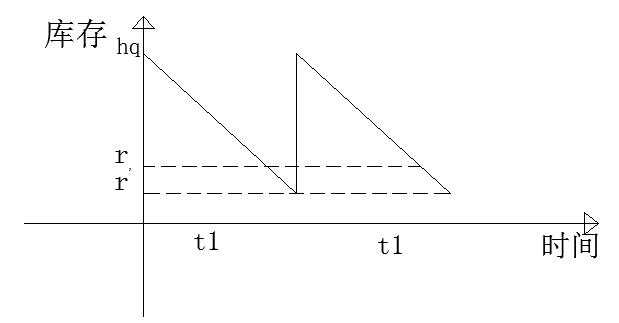


图1-3 图1-4

注：总成本最小处的订货数量Q便为末端收货商最优的订货量

综上由EOQ模型可知，末端收货最优的订货量（其中D为末端收货商的需求量，S为单次的订购成本，H为产品的储存成本，这些参数均可通过市场调查得到），末端收货商最优的安全库存=库存消耗速率\*订单时（订单时间指的是从末端收货商下订单开始到其接到货物的时间，库存消耗速率与订单时间均可通过市场调查得到）。

6.3问题三：基于各个公司的个体层面模型的建立

6.3.1 模型二的主要思想

从个体角度分析，根据每个供应商在流通网络中所处的位置不同，找出其所对应的末端收货商个数；再根据实际需求推导出该供应商的安全库存量与单次进货数.找出流通库存与二者的关系，在满足货物到货率需求的前提下通过优化定义安全库存与单次进货数，使所有分销商的库存总和最小。

6.3.2 相关参数的量化

（1）某公司每次补货的订单量

定义个体供应商（收货商）每次补货的订货量时，考虑到订货量与该公司实际需求密切相关，而公司需求又取决于下一级各公司每次补货的订货量，以此类推订货量最终取决于该公司对应的末级收货商的总订货量数。设该公司对应的末端公司数为,假设各个末端收货商的需求量即单次补货定量为常数，可以得到该公司的订货量定义式为：

.

在图1-5中，第一级公司每次补货的订货量，第二级A公司每次补货的订单量，以此类推。

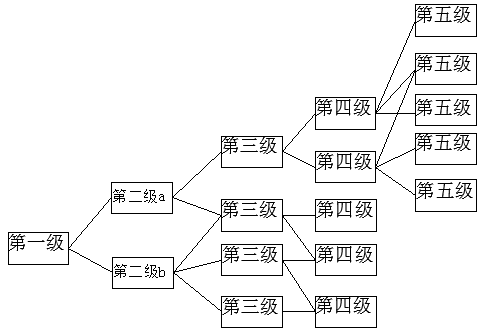


图1-5

（2）某公司要求补货的安全库存

定义个体供应商（收货商）拥有的商品数达到的某一定值即安全库存时，考虑到要保证在其向上级要求补货期间，若有下级公司向该公司提交订单要求补货，该公司能凭借安全库存的商品数满足其补货的要求。因此要找到该公司对应的所有下级公司中每次要求补货的安全库存的最大值，作为该公司安全库存的值。即该公司每次补货的安全库存只与其对应下级中的最大的补货订单量存值有关，可以用公式表示为

（3）各个公司的到货率h

考虑到当多个下级向公司要求补货时，公司将其下级的需求量向上级报告，并希望尽可能多的得到需求的商品数量。但由于流通链顶端的生产能力是一定的，所以整条链上从一开始的供给量即已经无法满足，且实际供给的数量与要求补货的数量（即补货量）在各个流通级或各个不同供应商中是相同的。而到货率与生产能力有关，在生产能力一定时，将各个公司的到货率定义为常量h。

（4） 某公司的流通库存

考虑到某公司对应的多个子公司向其要求补货的时间间隔是随机的，且当其要求补货的子公司数较多时，呈阶梯状递减的公司库存量可以简化为线性递减即商品持续消耗的库存模型，如图1–6所示：

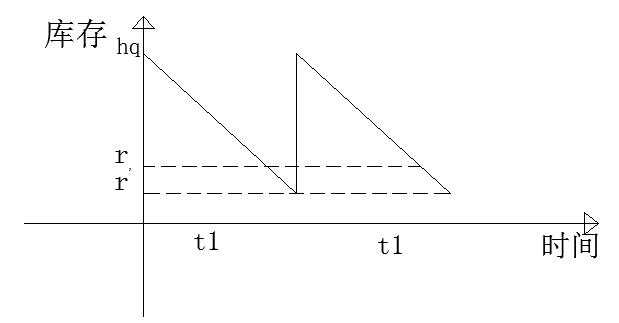


图1-6

在该库存模型中，由线性变化的数学关系可得，平均流通库存等于该公司补货定单量的一半加上安全库存值。考虑到实际情况下当公司达到安全库存要求补货时，一定时间后货物才能运达，而这段时间里公司库存因持续的消耗而不断减少，直至补货货物到达公司时，公司库存达到最小值。因此进一步优化模型，假设商品消耗速度与要求补货商品到达的时间一定，那么安全库存与库存最小值之差是一个大于零的常数，将该常数设为c,即：

那么该公司的流通库存可以用公式表示为：

6.3.3 模型二的建立

变量推导的思想如表1–2所示：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 该公司级数 | 该公司对应末端收货商数量 | 该公司需求 | 该公司实到货物量 | 该公司安全库存 |
| L |  |  |  |  |
| L-1 |  |  |  |  |
| L-2 |  |  |  |  |
| … |  |  |  |  |
| 1 |  |  |  |  |

表1-2

（注：树状流通网从上到下公司级数依次为1、2...L,末端收货商为第L级公司）

由推导过程可以看出：

——该公司对应的末端供货商的数量

且：

因此在该模型中，确定模型对应产品的流通网络后，各公司的订单量、要求补货的安全库存都可以用末端收货商的订单量以及要求补货的安全库存表示，根据公式：

计算出各个公司的流通库存量，进而表示出整个网络的流通总库存：

在生产能力一定，即各公司到货率h一定（等于末端到货率h）的前提下，确定适当的末端收货商的订单量以及要求补货的安全库存的值，以期达到目标要求——使总流通库存达到最小。

6.3.4 模型二的求解

6.3.4.1 单次订货量与安全库存的求解与计算

在数据处理部分中，构造出以一种产品为例的全部网状结构后，利用附件中给出的订单数据，计算出实际流通库存：即除末端公司订单量以外，各个公司向其上级公司发出的订单量总和，即各公司的实际q1值之和；

同时，通过各公司对应的全部末端公司的订单数据，计算出模型中给出的计算方法所求得的各公司的改进q值，为之后带入模型给出的流通库存公式，最后计算各级的流通库存之和做准备。

以上思路在c++程序中进行程序实现：

\_csv2.Read("流转数据.csv");

for(int i=2;i<=\_csv2.get\_row();i++) //按行读取数据

{

if(strcmp(\_csv2.get\_data(1,i),"MTMKF酯JN")==0

//以单一产品“MTMKF酯JN”为例进行计算，首先实现一种产品的网状结构的所需数据计算

{

for(int j=1;j<n;j++) //该行从左至右按列读取各个数据

{

if(strcmp(\_csv2.get\_data(13,i),p[j].name)==0&&strcmp(\_csv2.get\_data(8,i),p[p[j].pre].name)==0) //若读取到的数据为收货公司（receiver company）的名字且该公司的上级公司名字为空，即若读取到了第一级公司；

{

int m=atoi(\_csv2.get\_data(19,i));

//则继续读取该条数据，获得该接货公司实际获得的货物量

int n1=p[j].pre; //读取当前数据公司的上级公司编号，被赋值给n1；

p[j].q1+=m; //获取当前读取公司的实际得货量，赋值给该公司的q1（实际订单量属性值）

if(p[n1].r<m)p[n1].r=m; //若当前读取公司的上级公司的安全库存小于该公司的需求订单量，则将上级公司的安全库存更新为较大的下级订单需求量，通过不断循环最终将所有上级公司的安全库存均更新为了其下级公司中最大的订单需求量，存储在结构体p的r（建议安全库存）属性值中

if(p[j].sub==0)while(n1) //判断当前读取公司是否为末端公司，是则进入以下循环体，当循环至头级公司时跳出循环；

{

p[n1].q+=m/0.9; //将当前末端公司的订单需求量加至其上级公司的模型建议q值中；

n1=p[n1].pre; //将上级公司不断上移，即每一个位于该末端公司流通链的非末端公司的建议q值均加上此末端公司的订单需求量；每次一经查询到末端公司，则在其流通链上的全部非末端公司的建议q值均加上此末端公司的订单量，经过循环可计算出全部非末端公司的建议订单量。

}

break;

}

}

}

}

6.3.4.2 流通库存的求解与计算

（1）在以上的计算过程中，经过计算已经将全部非末端公司的实际收货量q1、建议订单量q2、建议安全库存r都已经计算得出并保留至结构体中，下面根据模型建立给出的流通库存定义，按照赌赢的计算方法计算此产品的网状结构中实际的流通库存和按给出建议改进后的流通库存，计算方法如下：

double k1=0;

for( int i=1;i<n;i++)

{

if(p[i].sub==0)

k1+=p[i].q1;} //将所有末端公司的实际收获量相加，得出末端的非流通库存k1；

cout<<k1<<endl;

system("pause");

在以上计算结果基础上，将该产品的全部订单量由excel求和计算得出，两数相减得到的差值即是此产品的流通网络中实际的流通库存；结果如下：



图1-7

Excel全部订单求和结果为1995123；

即得到此流通网络的实际流通库存值为19955123-561159=1433964。

（2）分级计算按照模型分配各公司q、r值后的改进流通库存，以计算全部第一级公司的流通库存为例：

double sum=0;

for(int i=1;i<n;i++)

{

if(p[i].r!=0&&p[i].l==1) //判断该公司是否为末端公司、是否位于第一级，若为非末端的第一级公司，则通过判断进入以下计算：

{

sum+=(p[i].r+p[i].q\*0.9)/2; //带入模型给出的流通库存计算公式进行计算

cout<< "序号" <<i << "\t订单量" << p[i].q << "\t安全库存" << p[i].r << "\t子公司数" << p[i].sub << '\t' << k << endl;

}

}

cout<<"理论流通库存"<<sum<<endl;

结果如下：

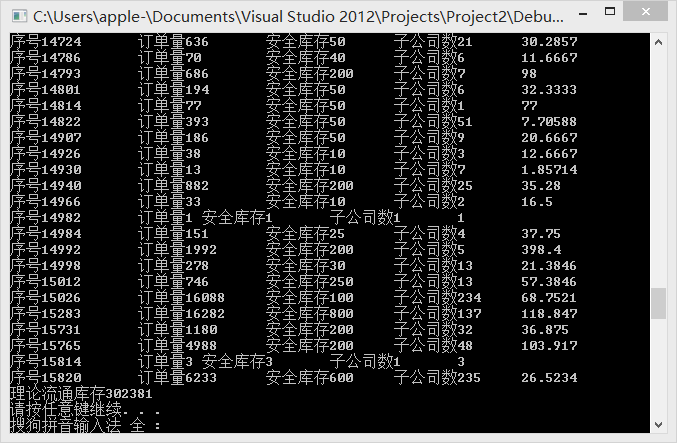


图1-8

即第一级的总流通库存值为302381；按照相同的程序实现方法分别计算非末端的前六级的各级流通库存，结果分别为：

|  |  |
| --- | --- |
| 级数 | 各级改进流通库存 |
| 1 | 302381 |
| 2 | 109460 |
| 3 | 6152.6 |
| 4 | 85.9 |
| 5 | 10922.7 |
| 6 | 29601 |

表1-3

经过以上计算，确定此产品的流通网络中，按照模型给出的建议q、r值进行计算，算得各级流通库存后，求和得出的最终改进流通库存为452450.6；

6.3.4.3 计算结果的对比与分析

对比当前实际流通库存，1433964>452450.6，发现明显降低了网络总流通库存，模型成立。

需要注意的是在这里的末级订单量的处理：

由于已有数据中未提供能够采用EOQ算法计算末端公司建议订单量的数据依据，上述结构链算法中，带入的末端订单量仍然是从已有数据中能够获得的实际订单量，而对于非末端公司，在采用实际末端订单量的基础上，带入论文前述的非末端公司建议订单量、建议安全库存的计算方法进行计算，得到了上述的流通库存结果；即是说，我们如果能够获取各末端公司销售速率、订货成本等相关数据，用EOQ算法精确计算出各末端公司的建议q值，再带入前文提供的非末端公司q、r计算方法，最终得到的改进总流通库存将会得到进一步优化。

以上的计算结果通过附件中给出的程序实现了输出至excel中，并记录所有计算结果的完善功能。

七 问题的分析与求解

7.1总库存与末端到货率的关系

详见第六部分模型的建立与求解6.1

7.2求若要满足目前到货率90%不变，并且使所有分销商的库存量总和最小，和的值应该为多少？

定义库存总和为一次订货周期内除末端收货商所有分销商的库存量总和。

该公司的订货量定义式为，表示流通网络中该公司对应的末级收货商的总订货量数，为对应的末级收货商数，为末端每个公司单次订单量。

该公司的安全库存定义式为：

表示流通网络中该公司对应的所有下级公司中每次要求补货的安全库存的最大值，以此使该公司能凭借安全库存满足其下级公司的补货要求。的计算最终用末端收货商单次订单量表示。

我们用各个公司的单次订货量与安全库存表示流通库存，流通库存定义式为：

=

进而表示出库存总和为：

的计算最终用末端收货商以及安全库存表示。为使所有分销商的库存量总和最小，引进EOQ模型，根据实际销售情况确定最优与，带入式中计算得出最优库存总和。

7.3若生产能力提高，估算能使末端收货商的到货率提高至95%，请重新估算供应商的最优库存。

思路同第二问相同，流通库存表示为：

流通网络中各个公司的安全库存与单次订单量通过其在流通网络中的位置推导迭代求得，最优库存总和为：

八 模型结果的检验、讨论和分析

8.1 模型结果的检验

题目中给出的调配目标是使流通总库存量达到最小，研究小组通过比较表格中给出的实际流通库存与模型求得的最有流通库存的大小来判断优化结果是否有效。以下是具体检验思路：

1求实际库存

研究小组求解实际流通库存=所有订单量之和-末端收货商向上级要求补货的订单量，经计算求得实际流通库存为：1433964。其中在流通网络中求得末端订单量的具体算法参考5.1数据的处理部分和6.3.4模型二的求解部分，详细代码见附件二。

2求最优库存

根据模型二的推导求解出各个公司的最优库存：

进而表示出整个网络的流通总库存：

其中、均可用末端收货商的单次订单量表示。

由EOQ模型可得，末端收货商订单量：

=

D、S、H均可由市场调查得到，但是小组从题目中不能得到数据，所以在检验时直接利用数据中给出的末端收货商的订单量，代入最优库存公式，求得库存总量为：452450.6。具体算法参考5.1数据的处理部分和6.3.4模型二的求解部分，详细代码见附件二。

8.1.3比较分析

从求的结果可以看出（2）中求得的库存量452450.6小于（1）中实际的流通库存量1433964，更进一步，最后模型中基于EOQ模型优化方案所求得的流通库存总量小于（2）中所求得的流通库存总量，所以模型优化的流通库存量小于（1）中实际求得的流通库存总量。综上，模型得到检验。

8.2 模型结果的讨论和分析

模型最优库存：

在该模型检验中，由于和不可直接通过数据计算得到，所以检验计算公式为：

由于该式中的q和r并不是由末端收获商最优的和计算所得而是利用题目中给出订货量迭代求出，所以。通过实际计算得出< K，所以最后可得：< K，模型得到检验。检验思路如图1–7所示。

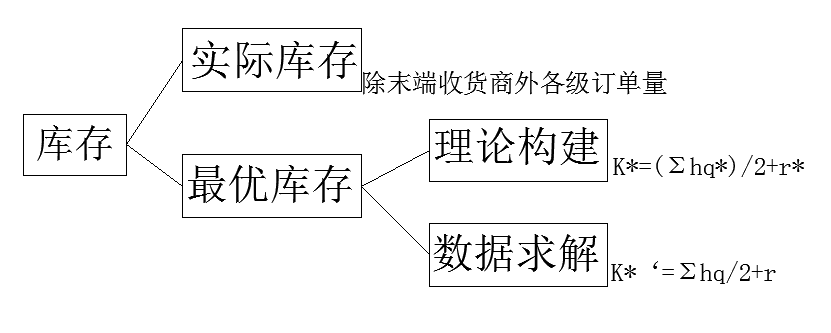


图1-9

九 模型的优缺点与改进

9.1 模型的评价

9.1.1基于某级公司总体的模型的评价

1.对各个公司本身的订单量、安全库存等进行了平均化处理。但对于各个公司的针对性不强，实际的网状结构中处于同一级的公司因其对应末端经销商个数的不同，相应的单次补货定单量与安全库存值会发生变化；

2. 考虑到某公司对应的多个子公司向其要求补货的时间间隔是随机的，且当其要求补货的子公司数较多时，将除末端收货商外各级公司的库存——时间函数合理地简化为了线性关系。

9.1.2.1基于个体公司的模型的优点分析

1. 个体针对性分析：

考虑到了某公司所对应的末端收货商总数的不同导致的同级之间不同公司的差别，根据单个公司在网络中的位置，针对性地计算相应安全库存、补货量与流通库存量；

2. 少量变量表示复杂模型：

采用从最后一级依次向前递推的分析思路，根据单个公司在流通网中的位置，用末端收货商的单次补货定量确定各个的位置公司的单次补货量与安全库存值，减少模型分析的影响因素数量；

3. 合理假设与理论分析：

将看作到货率h是根据生产能力的宏观调控值；并简化每个公司订货时间、订单到达周期以及初始库存等随机分布变量对模型分析的影响；根据库存变化的数量特征，根据库存量——时间图合理定义流通库存的表示方法，确定核心影响变量与表达公式；

4.参考EOQ模型的概念确定末端安全库存的值；

5. 流通网数据的合理运用：

首先将题目给出的规模较大的流通网络简化分析，根据公司名称统计出各个产品每个公司在流通网中对应的末端收货商个数，根据统计值表示出该产品流通总库存。进一步运用题目给出的流通调配方案进行拟合，确定变量取值范围，再用C++编程通过改变末端收货商的安全库存和补货量使流通总库存达到最小值，实现预期目标。

9.1.2.2对基于个体公司的分析模型的缺点

考虑到某公司对应的多个子公司向其要求补货的时间间隔是随机的，且当其要求补货的子公司数较多时，将除末端收货商外各级公司的库存—时间函数简化为线性关系。实际流通网络中，存在有少量公司对应的下级公司只有一两个的特殊情况，其库存量不适用于线性关系的表达，需另外计算。

9.2 模型的改进

基于现实情况的多样化，第二级的公司数量并不是远远小于第一级的公司数量、第三级的公司数量并不是远远小于第二级的公司数量，以此类推。在这种情况下，将除末端收货商外的各级公司的库存——时间函数简单的视为线性函数则与现实相差较远，这时需要具体的考虑，各级子公司向上级的要货时间满足怎样的分布。

考虑本级公司中仅仅对应一个下级公司的比率有多少，本级公司中对应两个子公司的比率有多少，本级公司中对应三个子公司的比率有多少，以此类推，而不同的情况下流通库存的计算方法略有不同，以对应于一个子公司与对应于两个子公司为例：

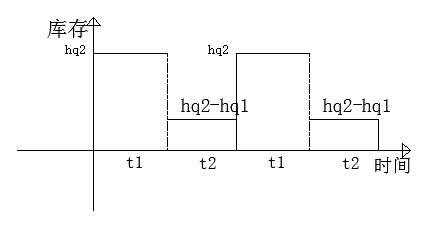


图1-10该公司下级对应一个子公司

该类型中的流通库存为：

K=（hq\*t1+r\*t2）/(t1+t2)

（假设一个子公司取货之后库存量便降到安全库存r以下，hp2为该级实到量，hp1为其下级一个公司的实到量）

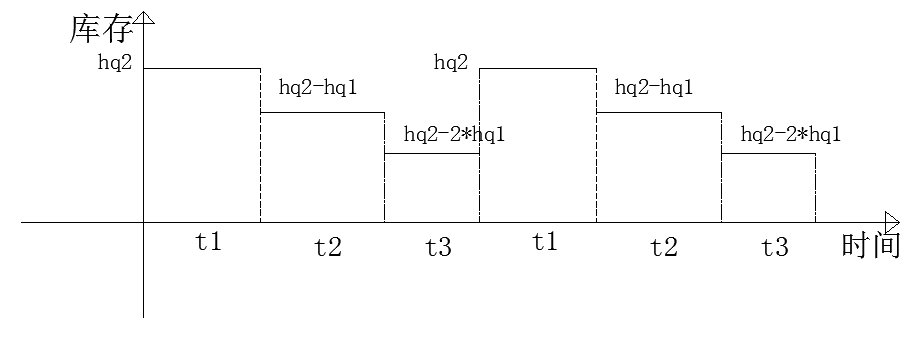


图1-11 该公司下级对应两个子公司

在该类型中的流通库存为：

K=(hq2+hq2-hq1+hq2-2\*hq1)/(t1+t2+t3)

(假设两个子公司都取货之后的库存量降至安全库存r以下hp2为该级实到量，hp1为其下级一个公司的实到量)

以此类推，当对应下级公司数较少时（少于4个）可算出一对一、一对二等类型的流通库存；其他情况当对应多个下级时仍采用库存——时间线性函数计算，统计出在某个流通网络中过每种情况出现的概率（推测呈正态分布），进而各个公司的流通库存乘以各自概率、再求和，便可得出总的流通库存。即计算出各种对应情况下、针对各个公司的总流通库存量。

十 参考文献

[1] 许中蓉，多级供应链条件下库存优化模型及其算法，北京交通大学硕士论文，2009年6月

[2] 李军、孙玉甫，存贮问题中的批量订货点模型，大庆石油学院学报，1995年三月第19卷第1期

[3] 姜启源、谢金星、叶俊，《数学建模》第四版，第三章，简单的优化模型，第一节，存贮模型，高等教育出版社

十一 附件

附件一 C++编程实现流通网络可视化

#include <iostream>

using namespace std;

#include <iostream>

#include <fstream>

#include <cstring>

#include <cstdlib>

//#define LINUX

#define WINDOWS

#define CSV\_DEBUG

#define TRUE 1

#define FALSE 0

#define OK 1

#define ERROR 0

#define INFEASIBLE -1

#define LOVERFLOW -2

typedef int Status;

#define STR\_LENGTH 4096

class CSV

{

protected:

int col;

int row;

char \*\*\*data;

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

函数名称:FreeData

函数功能:若申请过动态空间，则释放并将col和row置为0，data指针置为NULL

调用清单:无

输入参数:无

输出参数:无

函数返回:始终返回TRUE

其它说明:无

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

Status FreeData();

public:

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

函数名称:CSV

函数功能:构造函数，将col和row置为0，data指针置为NULL

调用清单:无

输入参数:无

输出参数:无

函数返回:无

其它说明:无

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

CSV();

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

函数名称:～CSV

函数功能:析构函数，释放已申请的动态空间

调用清单:FreeData

输入参数:无

输出参数:无

函数返回:无

其它说明:无

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

~CSV();

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

函数名称:get\_col

函数功能:获取csv表格列数

调用清单:无

输入参数:无

输出参数:无

函数返回:表格列数

其它说明:无

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

int get\_col() const;

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

函数名称:get\_row

函数功能:获取csv表格行数

调用清单:无

输入参数:无

输出参数:无

函数返回:表格行数

其它说明:无

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

int get\_row() const;

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

函数名称:get\_data

函数功能:获取csv表格指定单元格的字符串指针

调用清单:无

输入参数:无

输出参数:无

函数返回:指定数据字符串的指针

其它说明:无

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

const char\* get\_data(int col, int row);

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

函数名称:Read

函数功能:读取csv文件

调用清单:FreeData

open

getline

exit

clear

seekg

peek

strcpy

close

输入参数:无

输出参数:无

函数返回:指定数据字符串的指针

其它说明:无

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

Status Read(const char \*file\_name);

#ifdef CSV\_DEBUG

friend int main();

#endif // CSV\_DEBUG

};

Status CSV::FreeData()

{

if (this->data != NULL)

{

for (int i = 0; i < this->row; i++)

{

for (int j = 0; j < this->col; j++)

{

delete []this->data[i][j];

this->data[i][j] = NULL;

}

delete []this->data[i];

this->data[i] = NULL;

}

delete []this->data;

this->row = 0;

this->col = 0;

this->data = NULL;

}

return TRUE;

}

CSV::CSV()

{

this->row = 0;

this->col = 0;

this->data = NULL;

}

CSV::~CSV()

{

FreeData();

}

int CSV::get\_row() const

{

return this->row;

}

int CSV::get\_col() const

{

return this->col;

}

const char\* CSV::get\_data(int col, int row)

{

if (col > 0 && col <= this->col && row > 0 && row <= this->row)

return data[row - 1][col - 1];

else

return NULL;

}

Status CSV::Read(const char \*file\_name)

{

std::ifstream infile;

char line[STR\_LENGTH];

char temp[STR\_LENGTH];

int percent\_history = 0;

int percent\_current = 0;

FreeData();

infile.open(file\_name, std::ios::in);

if (!infile)

{

std::cout << "打开文件 '" << file\_name << "'失败" << std::endl;

return FALSE;

}

//获取列数

infile.getline(line, STR\_LENGTH);

for (int i = 0; line[i] != '\0'; i++)

{

if (line[i] == ',')

this->col++;

}

this->col++;

//获取行数

for (this->row = 0; infile.peek() != EOF; this->row++)

{

infile.getline(line, STR\_LENGTH);

}

this->row++;

#ifdef CSV\_DEBUG

std::cout << "文件:" << file\_name << " - 列数:" << this->col << " - 行数:" << this->row << std::endl;

#endif // CSV\_DEBUG

//申请空间

this->data = new char\*\*[this->row];

if (!this->data)

exit(LOVERFLOW);

for (int i = 0; i < this->row; i++)

{

this->data[i] = new char \*[this->col];

if (!this->data[i])

exit(LOVERFLOW);

for (int j = 0; j < this->col; j++)

this->data[i][j] = NULL;

}

infile.clear();

infile.seekg(0);

for (int row\_pos = 0; row\_pos < this->row; row\_pos++)

{

#ifdef LINUX

percent\_current = (row\_pos/static\_cast<double>(this->row)) \* 100;

if (percent\_current > percent\_history)

{

percent\_history = percent\_current;

std::cout << "正在载入:" << percent\_history << "%" << std::endl;

std::cout << "\033[1A";

}

#endif // LINUX

infile.getline(line, STR\_LENGTH);

for (int j = 0, k = 0, col\_pos = 0; col\_pos < this->col; j++)

{

if (line[j] != ',' && line[j] != '\0' && line[j] != '\r' && line[j] != '\n')

{

temp[k] = line[j];

k++;

}

else

{

temp[k] = '\0';

this->data[row\_pos][col\_pos] = new char[k + 1];

if (!this->data[row\_pos][col\_pos])

exit(LOVERFLOW);

strcpy(data[row\_pos][col\_pos], temp);

col\_pos++;

k = 0;

}

}

}

infile.close();

#ifdef LINUX

std::cout << std::endl;

#endif // LINUX

return TRUE;

}

int main()

{

const char \*head[7];

const char \*line[7];

int num;

CSV my\_csv;

my\_csv.Read("产品流转网状关系参考 - 20160414.csv");

for (int i = 2; i <= my\_csv.get\_row(); i++)

{

cout << "第" << i << "行 ";

for (int j = 1; j <= my\_csv.get\_col(); j++)

{

line[j] = my\_csv.get\_data(j, i);

if (strcmp(line[j], "") != 0)

num = j;

}

for (int j = 1; j <= num; j++)

{

if (strcmp(line[j], "") != 0)

{

head[j] = line[j];

cout << "->" << line[j];

}

else

{

cout << "->" << head[j];

}

}

cout << endl;

}

return 0;

附件二 网状结构的算法实现

#include <iostream>

#include "libxl.h"

using namespace std;

using namespace libxl;

#include <iostream>

#include <fstream>

#include <cstring>

#include <cstdlib>

//#define LINUX

#define WINDOWS

#define CSV\_DEBUG

#define TRUE 1

#define FALSE 0

#define OK 1

#define ERROR 0

#define INFEASIBLE -1

#define LOVERFLOW -2

typedef int Status;

#define STR\_LENGTH 4096

class CSV

{

protected:

int col;

int row;

char \*\*\*data;

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

函数名称:FreeData

函数功能:若申请过动态空间，则释放并将col和row置为0，data指针置为NULL

调用清单:无

输入参数:无

输出参数:无

函数返回:始终返回TRUE

其它说明:无

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

Status FreeData();

public:

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

函数名称:CSV

函数功能:构造函数，将col和row置为0，data指针置为NULL

调用清单:无

输入参数:无

输出参数:无

函数返回:无

其它说明:无

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

CSV();

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

函数名称:～CSV

函数功能:析构函数，释放已申请的动态空间

调用清单:FreeData

输入参数:无

输出参数:无

函数返回:无

其它说明:无

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

~CSV();

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

函数名称:get\_col

函数功能:获取csv表格列数

调用清单:无

输入参数:无

输出参数:无

函数返回:表格列数

其它说明:无

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

int get\_col() const;

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

函数名称:get\_row

函数功能:获取csv表格行数

调用清单:无

输入参数:无

输出参数:无

函数返回:表格行数

其它说明:无

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

int get\_row() const;

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

函数名称:get\_data

函数功能:获取csv表格指定单元格的字符串指针

调用清单:无

输入参数:无

输出参数:无

函数返回:指定数据字符串的指针

其它说明:无

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

const char\* get\_data(int col, int row);

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

函数名称:Read

函数功能:读取csv文件

调用清单:FreeData

open

getline

exit

clear

seekg

peek

strcpy

close

输入参数:无

输出参数:无

函数返回:指定数据字符串的指针

其它说明:无

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

Status Read(const char \*file\_name);

#ifdef CSV\_DEBUG

friend int main();

#endif // CSV\_DEBUG

};

Status CSV::FreeData()

{

if (this->data != NULL)

{

for (int i = 0; i < this->row; i++)

{

for (int j = 0; j < this->col; j++)

{

delete []this->data[i][j];

this->data[i][j] = NULL;

}

delete []this->data[i];

this->data[i] = NULL;

}

delete []this->data;

this->row = 0;

this->col = 0;

this->data = NULL;

}

return TRUE;

}

CSV::CSV()

{

this->row = 0;

this->col = 0;

this->data = NULL;

}

CSV::~CSV()

{

FreeData();

}

int CSV::get\_row() const

{

return this->row;

}

int CSV::get\_col() const

{

return this->col;

}

const char\* CSV::get\_data(int col, int row)

{

if (col > 0 && col <= this->col && row > 0 && row <= this->row)

return data[row - 1][col - 1];

else

return NULL;

}

Status CSV::Read(const char \*file\_name)

{

std::ifstream infile;

char line[STR\_LENGTH];

char temp[STR\_LENGTH];

int percent\_history = 0;

int percent\_current = 0;

FreeData();

infile.open(file\_name, std::ios::in);

if (!infile)

{

std::cout << "打开文件 '" << file\_name << "'失败" << std::endl;

return FALSE;

}

//获取列数

infile.getline(line, STR\_LENGTH);

for (int i = 0; line[i] != '\0'; i++)

{

if (line[i] == ',')

this->col++;

}

this->col++;

//获取行数

for (this->row = 0; infile.peek() != EOF; this->row++)

{

infile.getline(line, STR\_LENGTH);

}

this->row++;

#ifdef CSV\_DEBUG

std::cout << "文件:" << file\_name << " - 列数:" << this->col << " - 行数:" << this->row << std::endl;

#endif // CSV\_DEBUG

//申请空间

this->data = new char\*\*[this->row];

if (!this->data)

exit(LOVERFLOW);

for (int i = 0; i < this->row; i++)

{

this->data[i] = new char \*[this->col];

if (!this->data[i])

exit(LOVERFLOW);

for (int j = 0; j < this->col; j++)

this->data[i][j] = NULL;

}

infile.clear();

infile.seekg(0);

for (int row\_pos = 0; row\_pos < this->row; row\_pos++)

{

#ifdef LINUX

percent\_current = (row\_pos/static\_cast<double>(this->row)) \* 100;

if (percent\_current > percent\_history)

{

percent\_history = percent\_current;

std::cout << "正在载入:" << percent\_history << "%" << std::endl;

std::cout << "\033[1A";

}

#endif // LINUX

infile.getline(line, STR\_LENGTH);

for (int j = 0, k = 0, col\_pos = 0; col\_pos < this->col; j++)

{

if (line[j] != ',' && line[j] != '\0' && line[j] != '\r' && line[j] != '\n')

{

temp[k] = line[j];

k++;

}

else

{

temp[k] = '\0';

this->data[row\_pos][col\_pos] = new char[k + 1];

if (!this->data[row\_pos][col\_pos])

exit(LOVERFLOW);

strcpy(data[row\_pos][col\_pos], temp);

col\_pos++;

k = 0;

}

}

}

infile.close();

#ifdef LINUX

std::cout << std::endl;

#endif // LINUX

return TRUE;

}

int main()

{

int x[8];x[0]=0;

const char \*head[8];

const char \*line[8];

struct pl{

char name[30];

int l;

long q,r,q1;

int pre;

int sub;

};

pl p[17000];

p->name[0]=0;

p->l=0;

p->pre=0;

p->sub=0;

int n=1;

int num;

CSV my\_csv;

my\_csv.Read("流转网状关系参考.csv");

for(int i=2;i<=15000;i++)

{

for (int j = 1; j <= 7; j++)

{

line[j] = my\_csv.get\_data(j, i);

if(line[j][0])

{

num=j;

}

}

for (int j = 1; j <= 7; j++)

{

if(line[j][0]!='\0')

{

p[n].sub=0;

head[j] = line[j];

strcpy(p[n].name,line[j]);

p[n].q=p[n].r=p[n].q1=0;

p[n].l=j;

x[j]=n;

p[n].pre=x[j-1];

if(j==num)

{

int nn=p[n].pre;

while(nn!=0){p[nn].sub++;nn=p[nn].pre;}

}

n++;

}

}

}

CSV \_csv2;

\_csv2.Read("流转数据.csv");

for(int i=2;i<=\_csv2.get\_row();i++)

{

if(strcmp(\_csv2.get\_data(1,i),"MTMKF酯JN")==0)

{

for(int j=1;j<n;j++)

{

if(strcmp(\_csv2.get\_data(13,i),p[j].name)==0&&strcmp(\_csv2.get\_data(8,i),p[p[j].pre].name)==0)

{

double m=atoi(\_csv2.get\_data(19,i));

int n1=p[j].pre;

p[j].q1+=m;

if(p[n1].r<m)p[n1].r=m;

if(p[j].sub==0)while(n1)

{

p[n1].q+=m/0.9;

n1=p[n1].pre;

}

break;

}

}

}

}

double k1=0;

for( int i=1;i<n;i++)

{

if(p[i].sub==0)

k1+=p[i].q1;}

cout<<k1<<endl;

system("pause");

double sum=0;

for(int i=1;i<n;i++)

{

if(p[i].r!=0&&p[i].l==1)

{

sum+=(p[i].r+p[i].q\*0.9)/2;

double k=(p[i].q\*1.0/p[i].sub);

cout<< "序号" <<i << "\t订单量" << p[i].q << "\t安全库存" << p[i].r << "\t子公司数" << p[i].sub << '\t' << k << endl;

}

}

cout<<"理论流通库存"<<sum<<endl;

Book\* book = xlCreateBook();

if(book)

{ system("pause");

if(book->load("C:\\Users\\apple-\\Documents\\Visual Studio 2012\\Projects\\Project2\\Project2\\example.xls"))

{

Sheet\* sheet = book->getSheet(0);

if(sheet)

{

int k=2;

for(int i=1;i<n;i++)

{

sheet->writeStr(1, 01, "sub");

sheet->writeStr(1, 02, "q1");

sheet->writeStr(1, 03, "q");

sheet->writeStr(1, 04, "r");

if(p[i].r!=0&&p[i].l==1)

{

sheet->writeNum(k, 01, p[i].sub);

sheet->writeNum(k, 02, p[i].q1);

sheet->writeNum(k, 03, p[i].q);

sheet->writeNum(k, 04, p[i].r);

k++;

}

}

}else cout << "false" << endl;

book->save("example.xls");

}else cout << "false1" << endl;

system("pause");

book->release();

}

return 0;

}

附件三 excel数据导入导出的语言包准备

（1）、 1.h

#include <iostream>

using namespace std;

#include <iostream>

#include <fstream>

#include <cstring>

#include <cstdlib>

//#define LINUX

#define WINDOWS

#define CSV\_DEBUG

#define TRUE 1

#define FALSE 0

#define OK 1

#define ERROR 0

#define INFEASIBLE -1

#define LOVERFLOW -2

typedef int Status;

#define STR\_LENGTH 4096

class CSV

{

protected:

int col;

int row;

char \*\*\*data;

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

函数名称:FreeData

函数功能:若申请过动态空间，则释放并将col和row置为0，data指针置为NULL

调用清单:无

输入参数:无

输出参数:无

函数返回:始终返回TRUE

其它说明:无

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

Status FreeData();

public:

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

函数名称:CSV

函数功能:构造函数，将col和row置为0，data指针置为NULL

调用清单:无

输入参数:无

输出参数:无

函数返回:无

其它说明:无

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

CSV();

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

函数名称:～CSV

函数功能:析构函数，释放已申请的动态空间

调用清单:FreeData

输入参数:无

输出参数:无

函数返回:无

其它说明:无

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

~CSV();

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

函数名称:get\_col

函数功能:获取csv表格列数

调用清单:无

输入参数:无

输出参数:无

函数返回:表格列数

其它说明:无

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

int get\_col() const;

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

函数名称:get\_row

函数功能:获取csv表格行数

调用清单:无

输入参数:无

输出参数:无

函数返回:表格行数

其它说明:无

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

int get\_row() const;

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

函数名称:get\_data

函数功能:获取csv表格指定单元格的字符串指针

调用清单:无

输入参数:无

输出参数:无

函数返回:指定数据字符串的指针

其它说明:无

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

const char\* get\_data(int col, int row);

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

函数名称:Read

函数功能:读取csv文件

调用清单:FreeData

open

getline

exit

clear

seekg

peek

strcpy

close

输入参数:无

输出参数:无

函数返回:指定数据字符串的指针

其它说明:无

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

Status Read(const char \*file\_name);

#ifdef CSV\_DEBUG

friend int main();

#endif // CSV\_DEBUG

};

Status CSV::FreeData()

{

if (this->data != NULL)

{

for (int i = 0; i < this->row; i++)

{

for (int j = 0; j < this->col; j++)

{

delete []this->data[i][j];

this->data[i][j] = NULL;

}

delete []this->data[i];

this->data[i] = NULL;

}

delete []this->data;

this->row = 0;

this->col = 0;

this->data = NULL;

}

return TRUE;

}

CSV::CSV()

{

this->row = 0;

this->col = 0;

this->data = NULL;

}

CSV::~CSV()

{

FreeData();

}

int CSV::get\_row() const

{

return this->row;

}

int CSV::get\_col() const

{

return this->col;

}

const char\* CSV::get\_data(int col, int row)

{

if (col > 0 && col <= this->col && row > 0 && row <= this->row)

return data[row - 1][col - 1];

else

return NULL;

}

Status CSV::Read(const char \*file\_name)

{

std::ifstream infile;

char line[STR\_LENGTH];

char temp[STR\_LENGTH];

int percent\_history = 0;

int percent\_current = 0;

FreeData();

infile.open(file\_name, std::ios::in);

if (!infile)

{

std::cout << "打开文件 '" << file\_name << "'失败" << std::endl;

return FALSE;

}

//获取列数

infile.getline(line, STR\_LENGTH);

for (int i = 0; line[i] != '\0'; i++)

{

if (line[i] == ',')

this->col++;

}

this->col++;

//获取行数

for (this->row = 0; infile.peek() != EOF; this->row++)

{

infile.getline(line, STR\_LENGTH);

}

this->row++;

#ifdef CSV\_DEBUG

std::cout << "文件:" << file\_name << " - 列数:" << this->col << " - 行数:" << this->row << std::endl;

#endif // CSV\_DEBUG

//申请空间

this->data = new char\*\*[this->row];

if (!this->data)

exit(LOVERFLOW);

for (int i = 0; i < this->row; i++)

{

this->data[i] = new char \*[this->col];

if (!this->data[i])

exit(LOVERFLOW);

for (int j = 0; j < this->col; j++)

this->data[i][j] = NULL;

}

infile.clear();

infile.seekg(0);

for (int row\_pos = 0; row\_pos < this->row; row\_pos++)

{

#ifdef LINUX

percent\_current = (row\_pos/static\_cast<double>(this->row)) \* 100;

if (percent\_current > percent\_history)

{

percent\_history = percent\_current;

std::cout << "正在载入:" << percent\_history << "%" << std::endl;

std::cout << "\033[1A";

}

#endif // LINUX

infile.getline(line, STR\_LENGTH);

for (int j = 0, k = 0, col\_pos = 0; col\_pos < this->col; j++)

{

if (line[j] != ',' && line[j] != '\0' && line[j] != '\r' && line[j] != '\n')

{

temp[k] = line[j];

k++;

}

else

{

temp[k] = '\0';

this->data[row\_pos][col\_pos] = new char[k + 1];

if (!this->data[row\_pos][col\_pos])

exit(LOVERFLOW);

strcpy(data[row\_pos][col\_pos], temp);

col\_pos++;

k = 0;

}

}

}

infile.close();

#ifdef LINUX

std::cout << std::endl;

#endif // LINUX

return TRUE;

}

（2）、enum.h

#ifndef LIBXL\_ENUM\_CPP\_H

#define LIBXL\_ENUM\_CPP\_H

namespace libxl {

enum Color {COLOR\_BLACK = 8, COLOR\_WHITE, COLOR\_RED, COLOR\_BRIGHTGREEN, COLOR\_BLUE, COLOR\_YELLOW, COLOR\_PINK, COLOR\_TURQUOISE, COLOR\_DARKRED,

COLOR\_GREEN, COLOR\_DARKBLUE, COLOR\_DARKYELLOW, COLOR\_VIOLET, COLOR\_TEAL, COLOR\_GRAY25, COLOR\_GRAY50, COLOR\_PERIWINKLE\_CF,

COLOR\_PLUM\_CF, COLOR\_IVORY\_CF, COLOR\_LIGHTTURQUOISE\_CF, COLOR\_DARKPURPLE\_CF, COLOR\_CORAL\_CF, COLOR\_OCEANBLUE\_CF, COLOR\_ICEBLUE\_CF,

COLOR\_DARKBLUE\_CL, COLOR\_PINK\_CL, COLOR\_YELLOW\_CL, COLOR\_TURQUOISE\_CL, COLOR\_VIOLET\_CL, COLOR\_DARKRED\_CL, COLOR\_TEAL\_CL,

COLOR\_BLUE\_CL, COLOR\_SKYBLUE, COLOR\_LIGHTTURQUOISE, COLOR\_LIGHTGREEN, COLOR\_LIGHTYELLOW, COLOR\_PALEBLUE, COLOR\_ROSE, COLOR\_LAVENDER,

COLOR\_TAN, COLOR\_LIGHTBLUE, COLOR\_AQUA, COLOR\_LIME, COLOR\_GOLD, COLOR\_LIGHTORANGE, COLOR\_ORANGE, COLOR\_BLUEGRAY, COLOR\_GRAY40,

COLOR\_DARKTEAL, COLOR\_SEAGREEN, COLOR\_DARKGREEN, COLOR\_OLIVEGREEN, COLOR\_BROWN, COLOR\_PLUM, COLOR\_INDIGO, COLOR\_GRAY80,

COLOR\_DEFAULT\_FOREGROUND = 0x0040, COLOR\_DEFAULT\_BACKGROUND = 0x0041, COLOR\_TOOLTIP = 0x0051, COLOR\_AUTO = 0x7FFF};

enum NumFormat {NUMFORMAT\_GENERAL, NUMFORMAT\_NUMBER, NUMFORMAT\_NUMBER\_D2, NUMFORMAT\_NUMBER\_SEP, NUMFORMAT\_NUMBER\_SEP\_D2,

NUMFORMAT\_CURRENCY\_NEGBRA, NUMFORMAT\_CURRENCY\_NEGBRARED, NUMFORMAT\_CURRENCY\_D2\_NEGBRA, NUMFORMAT\_CURRENCY\_D2\_NEGBRARED,

NUMFORMAT\_PERCENT, NUMFORMAT\_PERCENT\_D2, NUMFORMAT\_SCIENTIFIC\_D2, NUMFORMAT\_FRACTION\_ONEDIG, NUMFORMAT\_FRACTION\_TWODIG,

NUMFORMAT\_DATE, NUMFORMAT\_CUSTOM\_D\_MON\_YY, NUMFORMAT\_CUSTOM\_D\_MON, NUMFORMAT\_CUSTOM\_MON\_YY,

NUMFORMAT\_CUSTOM\_HMM\_AM, NUMFORMAT\_CUSTOM\_HMMSS\_AM, NUMFORMAT\_CUSTOM\_HMM, NUMFORMAT\_CUSTOM\_HMMSS,

NUMFORMAT\_CUSTOM\_MDYYYY\_HMM,

NUMFORMAT\_NUMBER\_SEP\_NEGBRA = 37, NUMFORMAT\_NUMBER\_SEP\_NEGBRARED,

NUMFORMAT\_NUMBER\_D2\_SEP\_NEGBRA, NUMFORMAT\_NUMBER\_D2\_SEP\_NEGBRARED, NUMFORMAT\_ACCOUNT, NUMFORMAT\_ACCOUNTCUR,

NUMFORMAT\_ACCOUNT\_D2, NUMFORMAT\_ACCOUNT\_D2\_CUR, NUMFORMAT\_CUSTOM\_MMSS, NUMFORMAT\_CUSTOM\_H0MMSS,

NUMFORMAT\_CUSTOM\_MMSS0, NUMFORMAT\_CUSTOM\_000P0E\_PLUS0, NUMFORMAT\_TEXT};

enum AlignH {ALIGNH\_GENERAL, ALIGNH\_LEFT, ALIGNH\_CENTER, ALIGNH\_RIGHT, ALIGNH\_FILL, ALIGNH\_JUSTIFY, ALIGNH\_MERGE, ALIGNH\_DISTRIBUTED};

enum AlignV {ALIGNV\_TOP, ALIGNV\_CENTER, ALIGNV\_BOTTOM, ALIGNV\_JUSTIFY, ALIGNV\_DISTRIBUTED};

enum BorderStyle {BORDERSTYLE\_NONE, BORDERSTYLE\_THIN, BORDERSTYLE\_MEDIUM, BORDERSTYLE\_DASHED, BORDERSTYLE\_DOTTED, BORDERSTYLE\_THICK,

BORDERSTYLE\_DOUBLE, BORDERSTYLE\_HAIR, BORDERSTYLE\_MEDIUMDASHED, BORDERSTYLE\_DASHDOT, BORDERSTYLE\_MEDIUMDASHDOT,

BORDERSTYLE\_DASHDOTDOT, BORDERSTYLE\_MEDIUMDASHDOTDOT, BORDERSTYLE\_SLANTDASHDOT};

enum BorderDiagonal {BORDERDIAGONAL\_NONE, BORDERDIAGONAL\_DOWN, BORDERDIAGONAL\_UP, BORDERDIAGONAL\_BOTH};

enum FillPattern {FILLPATTERN\_NONE, FILLPATTERN\_SOLID, FILLPATTERN\_GRAY50, FILLPATTERN\_GRAY75, FILLPATTERN\_GRAY25,

FILLPATTERN\_HORSTRIPE, FILLPATTERN\_VERSTRIPE, FILLPATTERN\_REVDIAGSTRIPE, FILLPATTERN\_DIAGSTRIPE,

FILLPATTERN\_DIAGCROSSHATCH, FILLPATTERN\_THICKDIAGCROSSHATCH, FILLPATTERN\_THINHORSTRIPE, FILLPATTERN\_THINVERSTRIPE,

FILLPATTERN\_THINREVDIAGSTRIPE, FILLPATTERN\_THINDIAGSTRIPE, FILLPATTERN\_THINHORCROSSHATCH, FILLPATTERN\_THINDIAGCROSSHATCH,

FILLPATTERN\_GRAY12P5, FILLPATTERN\_GRAY6P25};

enum Script {SCRIPT\_NORMAL, SCRIPT\_SUPER, SCRIPT\_SUB};

enum Underline {UNDERLINE\_NONE, UNDERLINE\_SINGLE, UNDERLINE\_DOUBLE, UNDERLINE\_SINGLEACC = 0x21, UNDERLINE\_DOUBLEACC = 0x22};

enum Paper {PAPER\_DEFAULT, PAPER\_LETTER, PAPER\_LETTERSMALL, PAPER\_TABLOID, PAPER\_LEDGER, PAPER\_LEGAL, PAPER\_STATEMENT, PAPER\_EXECUTIVE, PAPER\_A3,

PAPER\_A4, PAPER\_A4SMALL, PAPER\_A5, PAPER\_B4, PAPER\_B5, PAPER\_FOLIO, PAPER\_QUATRO, PAPER\_10x14, PAPER\_10x17, PAPER\_NOTE, PAPER\_ENVELOPE\_9,

PAPER\_ENVELOPE\_10, PAPER\_ENVELOPE\_11, PAPER\_ENVELOPE\_12, PAPER\_ENVELOPE\_14, PAPER\_C\_SIZE, PAPER\_D\_SIZE, PAPER\_E\_SIZE, PAPER\_ENVELOPE\_DL,

PAPER\_ENVELOPE\_C5, PAPER\_ENVELOPE\_C3, PAPER\_ENVELOPE\_C4, PAPER\_ENVELOPE\_C6, PAPER\_ENVELOPE\_C65, PAPER\_ENVELOPE\_B4, PAPER\_ENVELOPE\_B5,

PAPER\_ENVELOPE\_B6, PAPER\_ENVELOPE, PAPER\_ENVELOPE\_MONARCH, PAPER\_US\_ENVELOPE, PAPER\_FANFOLD, PAPER\_GERMAN\_STD\_FANFOLD,

PAPER\_GERMAN\_LEGAL\_FANFOLD, PAPER\_B4\_ISO, PAPER\_JAPANESE\_POSTCARD, PAPER\_9x11, PAPER\_10x11, PAPER\_15x11, PAPER\_ENVELOPE\_INVITE,

PAPER\_US\_LETTER\_EXTRA = 50, PAPER\_US\_LEGAL\_EXTRA, PAPER\_US\_TABLOID\_EXTRA, PAPER\_A4\_EXTRA, PAPER\_LETTER\_TRANSVERSE, PAPER\_A4\_TRANSVERSE,

PAPER\_LETTER\_EXTRA\_TRANSVERSE, PAPER\_SUPERA, PAPER\_SUPERB, PAPER\_US\_LETTER\_PLUS, PAPER\_A4\_PLUS, PAPER\_A5\_TRANSVERSE, PAPER\_B5\_TRANSVERSE,

PAPER\_A3\_EXTRA, PAPER\_A5\_EXTRA, PAPER\_B5\_EXTRA, PAPER\_A2, PAPER\_A3\_TRANSVERSE, PAPER\_A3\_EXTRA\_TRANSVERSE, PAPER\_JAPANESE\_DOUBLE\_POSTCARD,

PAPER\_A6, PAPER\_JAPANESE\_ENVELOPE\_KAKU2, PAPER\_JAPANESE\_ENVELOPE\_KAKU3, PAPER\_JAPANESE\_ENVELOPE\_CHOU3, PAPER\_JAPANESE\_ENVELOPE\_CHOU4,

PAPER\_LETTER\_ROTATED, PAPER\_A3\_ROTATED, PAPER\_A4\_ROTATED, PAPER\_A5\_ROTATED, PAPER\_B4\_ROTATED, PAPER\_B5\_ROTATED,

PAPER\_JAPANESE\_POSTCARD\_ROTATED, PAPER\_DOUBLE\_JAPANESE\_POSTCARD\_ROTATED, PAPER\_A6\_ROTATED, PAPER\_JAPANESE\_ENVELOPE\_KAKU2\_ROTATED,

PAPER\_JAPANESE\_ENVELOPE\_KAKU3\_ROTATED, PAPER\_JAPANESE\_ENVELOPE\_CHOU3\_ROTATED, PAPER\_JAPANESE\_ENVELOPE\_CHOU4\_ROTATED, PAPER\_B6,

PAPER\_B6\_ROTATED, PAPER\_12x11, PAPER\_JAPANESE\_ENVELOPE\_YOU4, PAPER\_JAPANESE\_ENVELOPE\_YOU4\_ROTATED, PAPER\_PRC16K, PAPER\_PRC32K,

PAPER\_PRC32K\_BIG, PAPER\_PRC\_ENVELOPE1, PAPER\_PRC\_ENVELOPE2, PAPER\_PRC\_ENVELOPE3, PAPER\_PRC\_ENVELOPE4, PAPER\_PRC\_ENVELOPE5,

PAPER\_PRC\_ENVELOPE6, PAPER\_PRC\_ENVELOPE7, PAPER\_PRC\_ENVELOPE8, PAPER\_PRC\_ENVELOPE9, PAPER\_PRC\_ENVELOPE10, PAPER\_PRC16K\_ROTATED,

PAPER\_PRC32K\_ROTATED, PAPER\_PRC32KBIG\_ROTATED, PAPER\_PRC\_ENVELOPE1\_ROTATED, PAPER\_PRC\_ENVELOPE2\_ROTATED, PAPER\_PRC\_ENVELOPE3\_ROTATED,

PAPER\_PRC\_ENVELOPE4\_ROTATED, PAPER\_PRC\_ENVELOPE5\_ROTATED, PAPER\_PRC\_ENVELOPE6\_ROTATED, PAPER\_PRC\_ENVELOPE7\_ROTATED,

PAPER\_PRC\_ENVELOPE8\_ROTATED, PAPER\_PRC\_ENVELOPE9\_ROTATED, PAPER\_PRC\_ENVELOPE10\_ROTATED};

enum SheetType { SHEETTYPE\_SHEET, SHEETTYPE\_CHART, SHEETTYPE\_UNKNOWN };

enum CellType {CELLTYPE\_EMPTY, CELLTYPE\_NUMBER, CELLTYPE\_STRING, CELLTYPE\_BOOLEAN, CELLTYPE\_BLANK, CELLTYPE\_ERROR};

enum ErrorType {ERRORTYPE\_NULL = 0x00, ERRORTYPE\_DIV\_0 = 0x07, ERRORTYPE\_VALUE = 0x0F, ERRORTYPE\_REF = 0x17, ERRORTYPE\_NAME = 0x1D,

ERRORTYPE\_NUM = 0x24, ERRORTYPE\_NA = 0x2A, ERRORTYPE\_NOERROR = 0xFF};

enum PictureType {PICTURETYPE\_PNG, PICTURETYPE\_JPEG, PICTURETYPE\_GIF, PICTURETYPE\_WMF, PICTURETYPE\_DIB, PICTURETYPE\_EMF,

PICTURETYPE\_PICT, PICTURETYPE\_TIFF, PICTURETYPE\_ERROR = 0xFF};

enum SheetState {SHEETSTATE\_VISIBLE, SHEETSTATE\_HIDDEN, SHEETSTATE\_VERYHIDDEN};

enum Scope {SCOPE\_UNDEFINED = -2, SCOPE\_WORKBOOK = -1};

enum Position {POSITION\_MOVE\_AND\_SIZE, POSITION\_ONLY\_MOVE, POSITION\_ABSOLUTE};

}

#endif

（3）、IBooKT.h

#ifndef LIBXL\_IBOOKT\_H

#define LIBXL\_IBOOKT\_H

#include <stddef.h>

#include "setup.h"

#include "enum.h"

namespace libxl {

template<class TCHAR> struct ISheetT;

template<class TCHAR> struct IFormatT;

template<class TCHAR> struct IFontT;

template<class TCHAR>

struct IBookT

{

virtual bool XLAPIENTRY load(const TCHAR\* filename) = 0;

virtual bool XLAPIENTRY save(const TCHAR\* filename) = 0;

virtual bool XLAPIENTRY loadRaw(const char\* data, unsigned size) = 0;

virtual bool XLAPIENTRY saveRaw(const char\*\* data, unsigned\* size) = 0;

virtual ISheetT<TCHAR>\* XLAPIENTRY addSheet(const TCHAR\* name, ISheetT<TCHAR>\* initSheet = 0) = 0;

virtual ISheetT<TCHAR>\* XLAPIENTRY insertSheet(int index, const TCHAR\* name, ISheetT<TCHAR>\* initSheet = 0) = 0;

virtual ISheetT<TCHAR>\* XLAPIENTRY getSheet(int index) const = 0;

virtual SheetType XLAPIENTRY sheetType(int index) const = 0;

virtual bool XLAPIENTRY delSheet(int index) = 0;

virtual int XLAPIENTRY sheetCount() const = 0;

virtual IFormatT<TCHAR>\* XLAPIENTRY addFormat(IFormatT<TCHAR>\* initFormat = 0) = 0;

virtual IFontT<TCHAR>\* XLAPIENTRY addFont(IFontT<TCHAR>\* initFont = 0) = 0;

virtual int XLAPIENTRY addCustomNumFormat(const TCHAR\* customNumFormat) = 0;

virtual const TCHAR\* XLAPIENTRY customNumFormat(int fmt) = 0;

virtual IFormatT<TCHAR>\* XLAPIENTRY format(int index) = 0;

virtual int XLAPIENTRY formatSize() = 0;

virtual IFontT<TCHAR>\* XLAPIENTRY font(int index) = 0;

virtual int XLAPIENTRY fontSize() = 0;

virtual double XLAPIENTRY datePack(int year, int month, int day, int hour = 0, int min = 0, int sec = 0, int msec = 0) = 0;

virtual bool XLAPIENTRY dateUnpack(double value, int\* year, int\* month, int\* day, int\* hour = 0, int\* min = 0, int\* sec = 0, int\* msec = 0) = 0;

virtual Color XLAPIENTRY colorPack(int red, int green, int blue) = 0;

virtual void XLAPIENTRY colorUnpack(Color color, int\* red, int\* green, int\* blue) = 0;

virtual int XLAPIENTRY activeSheet() const = 0;

virtual void XLAPIENTRY setActiveSheet(int index) = 0;

virtual int XLAPIENTRY pictureSize() const = 0;

virtual PictureType XLAPIENTRY getPicture(int index, const char\*\* data, unsigned\* size) const = 0;

virtual int XLAPIENTRY addPicture(const TCHAR\* filename) = 0;

virtual int XLAPIENTRY addPicture2(const char\* data, unsigned size) = 0;

virtual const TCHAR\* XLAPIENTRY defaultFont(int\* fontSize) = 0;

virtual void XLAPIENTRY setDefaultFont(const TCHAR\* fontName, int fontSize) = 0;

virtual bool XLAPIENTRY refR1C1() const = 0;

virtual void XLAPIENTRY setRefR1C1(bool refR1C1 = true) = 0;

virtual void XLAPIENTRY setKey(const TCHAR\* name, const TCHAR\* key) = 0;

virtual bool XLAPIENTRY rgbMode() = 0;

virtual void XLAPIENTRY setRgbMode(bool rgbMode = true) = 0;

virtual int XLAPIENTRY version() const = 0;

virtual int XLAPIENTRY biffVersion() const = 0;

virtual bool XLAPIENTRY isDate1904() const = 0;

virtual void XLAPIENTRY setDate1904(bool date1904 = true) = 0;

virtual bool XLAPIENTRY isTemplate() const = 0;

virtual void XLAPIENTRY setTemplate(bool tmpl = true) = 0;

virtual bool XLAPIENTRY setLocale(const char\* locale) = 0;

virtual const char\* XLAPIENTRY errorMessage() const = 0;

virtual void XLAPIENTRY release() = 0;

virtual ~IBookT() {}

};

}

extern "C" XLAPI libxl::IBookT<char>\* XLAPIENTRY xlCreateBookA();

extern "C" XLAPI libxl::IBookT<wchar\_t>\* XLAPIENTRY xlCreateBookW();

extern "C" XLAPI libxl::IBookT<char>\* XLAPIENTRY xlCreateXMLBookA();

extern "C" XLAPI libxl::IBookT<wchar\_t>\* XLAPIENTRY xlCreateXMLBookW();

#endif

（4）IFonT.h

#ifndef LIBXL\_IFONTT\_H

#define LIBXL\_IFONTT\_H

#include "setup.h"

#include "enum.h"

namespace libxl {

template<class TCHAR>

struct IFontT

{

virtual int XLAPIENTRY size() const = 0;

virtual void XLAPIENTRY setSize(int size) = 0;

virtual bool XLAPIENTRY italic() const = 0;

virtual void XLAPIENTRY setItalic(bool italic = true) = 0;

virtual bool XLAPIENTRY strikeOut() const = 0;

virtual void XLAPIENTRY setStrikeOut(bool strikeOut = true) = 0;

virtual Color XLAPIENTRY color() const = 0;

virtual void XLAPIENTRY setColor(Color color) = 0;

virtual bool XLAPIENTRY bold() const = 0;

virtual void XLAPIENTRY setBold(bool bold = true) = 0;

virtual Script XLAPIENTRY script() const = 0;

virtual void XLAPIENTRY setScript(Script script) = 0;

virtual Underline XLAPIENTRY underline() const = 0;

virtual void XLAPIENTRY setUnderline(Underline underline) = 0;

virtual const TCHAR\* XLAPIENTRY name() const = 0;

virtual bool XLAPIENTRY setName(const TCHAR\* name) = 0;

virtual ~IFontT() {}

};

}

#endif

（5）、IFormatT.h

#ifndef LIBXL\_IFORMATT\_H

#define LIBXL\_IFORMATT\_H

#include "setup.h"

#include "enum.h"

namespace libxl {

template<class TCHAR> struct IFontT;

template<class TCHAR>

struct IFormatT

{

virtual IFontT<TCHAR>\* XLAPIENTRY font() const = 0;

virtual bool XLAPIENTRY setFont(IFontT<TCHAR>\* font) = 0;

virtual int XLAPIENTRY numFormat() const = 0;

virtual void XLAPIENTRY setNumFormat(int numFormat) = 0;

virtual AlignH XLAPIENTRY alignH() const = 0;

virtual void XLAPIENTRY setAlignH(AlignH align) = 0;

virtual AlignV XLAPIENTRY alignV() const = 0;

virtual void XLAPIENTRY setAlignV(AlignV align) = 0;

virtual bool XLAPIENTRY wrap() const = 0;

virtual void XLAPIENTRY setWrap(bool wrap = true) = 0;

virtual int XLAPIENTRY rotation() const = 0;

virtual bool XLAPIENTRY setRotation(int rotation) = 0;

virtual int XLAPIENTRY indent() const = 0;

virtual void XLAPIENTRY setIndent(int indent) = 0;

virtual bool XLAPIENTRY shrinkToFit() const = 0;

virtual void XLAPIENTRY setShrinkToFit(bool shrinkToFit = true) = 0;

virtual void XLAPIENTRY setBorder(BorderStyle style = BORDERSTYLE\_THIN) = 0;

virtual void XLAPIENTRY setBorderColor(Color color) = 0;

virtual BorderStyle XLAPIENTRY borderLeft() const = 0;

virtual void XLAPIENTRY setBorderLeft(BorderStyle style = BORDERSTYLE\_THIN) = 0;

virtual BorderStyle XLAPIENTRY borderRight() const = 0;

virtual void XLAPIENTRY setBorderRight(BorderStyle style = BORDERSTYLE\_THIN) = 0;

virtual BorderStyle XLAPIENTRY borderTop() const = 0;

virtual void XLAPIENTRY setBorderTop(BorderStyle style = BORDERSTYLE\_THIN) = 0;

virtual BorderStyle XLAPIENTRY borderBottom() const = 0;

virtual void XLAPIENTRY setBorderBottom(BorderStyle style = BORDERSTYLE\_THIN) = 0;

virtual Color XLAPIENTRY borderLeftColor() const = 0;

virtual void XLAPIENTRY setBorderLeftColor(Color color) = 0;

virtual Color XLAPIENTRY borderRightColor() const = 0;

virtual void XLAPIENTRY setBorderRightColor(Color color) = 0;

virtual Color XLAPIENTRY borderTopColor() const = 0;

virtual void XLAPIENTRY setBorderTopColor(Color color) = 0;

virtual Color XLAPIENTRY borderBottomColor() const = 0;

virtual void XLAPIENTRY setBorderBottomColor(Color color) = 0;

virtual BorderDiagonal XLAPIENTRY borderDiagonal() const = 0;

virtual void XLAPIENTRY setBorderDiagonal(BorderDiagonal border) = 0;

virtual BorderStyle XLAPIENTRY borderDiagonalStyle() const = 0;

virtual void XLAPIENTRY setBorderDiagonalStyle(BorderStyle style) = 0;

virtual Color XLAPIENTRY borderDiagonalColor() const = 0;

virtual void XLAPIENTRY setBorderDiagonalColor(Color color) = 0;

virtual FillPattern XLAPIENTRY fillPattern() const = 0;

virtual void XLAPIENTRY setFillPattern(FillPattern pattern) = 0;

virtual Color XLAPIENTRY patternForegroundColor() const = 0;

virtual void XLAPIENTRY setPatternForegroundColor(Color color) = 0;

virtual Color XLAPIENTRY patternBackgroundColor() const = 0;

virtual void XLAPIENTRY setPatternBackgroundColor(Color color) = 0;

virtual bool XLAPIENTRY locked() const = 0;

virtual void XLAPIENTRY setLocked(bool locked = true) = 0;

virtual bool XLAPIENTRY hidden() const = 0;

virtual void XLAPIENTRY setHidden(bool hidden = true) = 0;

virtual ~IFormatT() {}

};

}

#endif

(6)ISheetT.h

#ifndef LIBXL\_ISHEETT\_H

#define LIBXL\_ISHEETT\_H

#include "setup.h"

#include "enum.h"

namespace libxl

{

template<class TCHAR> struct IFormatT;

template<class TCHAR>

struct ISheetT

{

virtual CellType XLAPIENTRY cellType(int row, int col) const = 0;

virtual bool XLAPIENTRY isFormula(int row, int col) const = 0;

virtual IFormatT<TCHAR>\* XLAPIENTRY cellFormat(int row, int col) const = 0;

virtual void XLAPIENTRY setCellFormat(int row, int col, IFormatT<TCHAR>\* format) = 0;

virtual const TCHAR\* XLAPIENTRY readStr(int row, int col, IFormatT<TCHAR>\*\* format = 0) = 0;

virtual bool XLAPIENTRY writeStr(int row, int col, const TCHAR\* value, IFormatT<TCHAR>\* format = 0) = 0;

virtual double XLAPIENTRY readNum(int row, int col, IFormatT<TCHAR>\*\* format = 0) const = 0;

virtual bool XLAPIENTRY writeNum(int row, int col, double value, IFormatT<TCHAR>\* format = 0) = 0;

virtual bool XLAPIENTRY readBool(int row, int col, IFormatT<TCHAR>\*\* format = 0) const = 0;

virtual bool XLAPIENTRY writeBool(int row, int col, bool value, IFormatT<TCHAR>\* format = 0) = 0;

virtual bool XLAPIENTRY readBlank(int row, int col, IFormatT<TCHAR>\*\* format) const = 0;

virtual bool XLAPIENTRY writeBlank(int row, int col, IFormatT<TCHAR>\* format) = 0;

virtual const TCHAR\* XLAPIENTRY readFormula(int row, int col, IFormatT<TCHAR>\*\* format = 0) = 0;

virtual bool XLAPIENTRY writeFormula(int row, int col, const TCHAR\* value, IFormatT<TCHAR>\* format = 0) = 0;

virtual const TCHAR\* XLAPIENTRY readComment(int row, int col) const = 0;

virtual void XLAPIENTRY writeComment(int row, int col, const TCHAR\* value, const TCHAR\* author = 0, int width = 129, int height = 75) = 0;

virtual bool XLAPIENTRY isDate(int row, int col) const = 0;

virtual ErrorType XLAPIENTRY readError(int row, int col) const = 0;

virtual double XLAPIENTRY colWidth(int col) const = 0;

virtual double XLAPIENTRY rowHeight(int row) const = 0;

virtual bool XLAPIENTRY setCol(int colFirst, int colLast, double width, IFormatT<TCHAR>\* format = 0, bool hidden = false) = 0;

virtual bool XLAPIENTRY setRow(int row, double height, IFormatT<TCHAR>\* format = 0, bool hidden = false) = 0;

virtual bool XLAPIENTRY rowHidden(int row) const = 0;

virtual bool XLAPIENTRY setRowHidden(int row, bool hidden) = 0;

virtual bool XLAPIENTRY colHidden(int col) const = 0;

virtual bool XLAPIENTRY setColHidden(int col, bool hidden) = 0;

virtual bool XLAPIENTRY getMerge(int row, int col, int\* rowFirst, int\* rowLast, int\* colFirst, int\* colLast) = 0;

virtual bool XLAPIENTRY setMerge(int rowFirst, int rowLast, int colFirst, int colLast) = 0;

virtual bool XLAPIENTRY delMerge(int row, int col) = 0;

virtual int XLAPIENTRY mergeSize() const = 0;

virtual bool XLAPIENTRY merge(int index, int\* rowFirst, int\* rowLast, int\* colFirst, int\* colLast) = 0;

virtual bool XLAPIENTRY delMergeByIndex(int index) = 0;

virtual int XLAPIENTRY pictureSize() const = 0;

virtual int XLAPIENTRY getPicture(int index, int\* rowTop = 0, int\* colLeft = 0, int\* rowBottom = 0, int\* colRight = 0,

int\* width = 0, int\* height = 0, int\* offset\_x = 0, int\* offset\_y = 0) const = 0;

virtual void XLAPIENTRY setPicture(int row, int col, int pictureId, double scale = 1.0, int offset\_x = 0, int offset\_y = 0, Position pos = POSITION\_MOVE\_AND\_SIZE) = 0;

virtual void XLAPIENTRY setPicture2(int row, int col, int pictureId, int width = -1, int height = -1, int offset\_x = 0, int offset\_y = 0, Position pos = POSITION\_MOVE\_AND\_SIZE) = 0;

virtual int XLAPIENTRY getHorPageBreak(int index) const = 0;

virtual int XLAPIENTRY getHorPageBreakSize() const = 0;

virtual int XLAPIENTRY getVerPageBreak(int index) const = 0;

virtual int XLAPIENTRY getVerPageBreakSize() const = 0;

virtual bool XLAPIENTRY setHorPageBreak(int row, bool pageBreak = true) = 0;

virtual bool XLAPIENTRY setVerPageBreak(int col, bool pageBreak = true) = 0;

virtual void XLAPIENTRY split(int row, int col) = 0;

virtual bool XLAPIENTRY splitInfo(int\* row, int\* col) const = 0;

virtual bool XLAPIENTRY groupRows(int rowFirst, int rowLast, bool collapsed = true) = 0;

virtual bool XLAPIENTRY groupCols(int colFirst, int colLast, bool collapsed = true) = 0;

virtual bool XLAPIENTRY groupSummaryBelow() const = 0;

virtual void XLAPIENTRY setGroupSummaryBelow(bool below) = 0;

virtual bool XLAPIENTRY groupSummaryRight() const = 0;

virtual void XLAPIENTRY setGroupSummaryRight(bool right) = 0;

virtual bool XLAPIENTRY clear(int rowFirst = 0, int rowLast = 1048575, int colFirst = 0, int colLast = 16383) = 0;

virtual bool XLAPIENTRY insertCol(int colFirst, int colLast) = 0;

virtual bool XLAPIENTRY insertRow(int rowFirst, int rowLast) = 0;

virtual bool XLAPIENTRY removeCol(int colFirst, int colLast) = 0;

virtual bool XLAPIENTRY removeRow(int rowFirst, int rowLast) = 0;

virtual bool XLAPIENTRY copyCell(int rowSrc, int colSrc, int rowDst, int colDst) = 0;

virtual int XLAPIENTRY firstRow() const = 0;

virtual int XLAPIENTRY lastRow() const = 0;

virtual int XLAPIENTRY firstCol() const = 0;

virtual int XLAPIENTRY lastCol() const = 0;

virtual bool XLAPIENTRY displayGridlines() const = 0;

virtual void XLAPIENTRY setDisplayGridlines(bool show = true) = 0;

virtual bool XLAPIENTRY printGridlines() const = 0;

virtual void XLAPIENTRY setPrintGridlines(bool print = true) = 0;

virtual int XLAPIENTRY zoom() const = 0;

virtual void XLAPIENTRY setZoom(int zoom) = 0;

virtual int XLAPIENTRY printZoom() const = 0;

virtual void XLAPIENTRY setPrintZoom(int zoom) = 0;

virtual bool XLAPIENTRY getPrintFit(int\* wPages, int\* hPages) const = 0;

virtual void XLAPIENTRY setPrintFit(int wPages = 1, int hPages = 1) = 0;

virtual bool XLAPIENTRY landscape() const = 0;

virtual void XLAPIENTRY setLandscape(bool landscape = true) = 0;

virtual Paper XLAPIENTRY paper() const = 0;

virtual void XLAPIENTRY setPaper(Paper paper = PAPER\_DEFAULT) = 0;

virtual const TCHAR\* XLAPIENTRY header() const = 0;

virtual bool XLAPIENTRY setHeader(const TCHAR\* header, double margin = 0.5) = 0;

virtual double XLAPIENTRY headerMargin() const = 0;

virtual const TCHAR\* XLAPIENTRY footer() const = 0;

virtual bool XLAPIENTRY setFooter(const TCHAR\* footer, double margin = 0.5) = 0;

virtual double XLAPIENTRY footerMargin() const = 0;

virtual bool XLAPIENTRY hCenter() const = 0;

virtual void XLAPIENTRY setHCenter(bool hCenter = true) = 0;

virtual bool XLAPIENTRY vCenter() const = 0;

virtual void XLAPIENTRY setVCenter(bool vCenter = true) = 0;

virtual double XLAPIENTRY marginLeft() const = 0;

virtual void XLAPIENTRY setMarginLeft(double margin) = 0;

virtual double XLAPIENTRY marginRight() const = 0;

virtual void XLAPIENTRY setMarginRight(double margin) = 0;

virtual double XLAPIENTRY marginTop() const = 0;

virtual void XLAPIENTRY setMarginTop(double margin) = 0;

virtual double XLAPIENTRY marginBottom() const = 0;

virtual void XLAPIENTRY setMarginBottom(double margin) = 0;

virtual bool XLAPIENTRY printRowCol() const = 0;

virtual void XLAPIENTRY setPrintRowCol(bool print = true) = 0;

virtual bool XLAPIENTRY printRepeatRows(int\* rowFirst, int\* rowLast) = 0;

virtual void XLAPIENTRY setPrintRepeatRows(int rowFirst, int rowLast) = 0;

virtual bool XLAPIENTRY printRepeatCols(int\* colFirst, int\* colLast) = 0;

virtual void XLAPIENTRY setPrintRepeatCols(int colFirst, int colLast) = 0;

virtual bool XLAPIENTRY printArea(int\* rowFirst, int\* rowLast, int\* colFirst, int\* colLast) = 0;

virtual void XLAPIENTRY setPrintArea(int rowFirst, int rowLast, int colFirst, int colLast) = 0;

virtual void XLAPIENTRY clearPrintRepeats() = 0;

virtual void XLAPIENTRY clearPrintArea() = 0;

virtual bool XLAPIENTRY getNamedRange(const TCHAR\* name, int\* rowFirst, int\* rowLast, int\* colFirst, int\* colLast, int scopeId = SCOPE\_UNDEFINED, bool\* hidden = 0) = 0;

virtual bool XLAPIENTRY setNamedRange(const TCHAR\* name, int rowFirst, int rowLast, int colFirst, int colLast, int scopeId = SCOPE\_UNDEFINED) = 0;

virtual bool XLAPIENTRY delNamedRange(const TCHAR\* name, int scopeId = SCOPE\_UNDEFINED) = 0;

virtual int XLAPIENTRY namedRangeSize() const = 0;

virtual const TCHAR\* XLAPIENTRY namedRange(int index, int\* rowFirst, int\* rowLast, int\* colFirst, int\* colLast, int\* scopeId = 0, bool\* hidden = 0) = 0;

virtual int XLAPIENTRY hyperlinkSize() const = 0;

virtual const TCHAR\* XLAPIENTRY hyperlink(int index, int\* rowFirst, int\* rowLast, int\* colFirst, int\* colLast) = 0;

virtual bool XLAPIENTRY delHyperlink(int index) = 0;

virtual void XLAPIENTRY addHyperlink(const TCHAR\* hyperlink, int rowFirst, int rowLast, int colFirst, int colLast) = 0;

virtual const TCHAR\* XLAPIENTRY name() const = 0;

virtual void XLAPIENTRY setName(const TCHAR\* name) = 0;

virtual bool XLAPIENTRY protect() const = 0;

virtual void XLAPIENTRY setProtect(bool protect = true) = 0;

virtual SheetState XLAPIENTRY hidden() const = 0;

virtual bool XLAPIENTRY setHidden(SheetState state = SHEETSTATE\_HIDDEN) = 0;

virtual void XLAPIENTRY getTopLeftView(int\* row, int\* col) const = 0;

virtual void XLAPIENTRY setTopLeftView(int row, int col) = 0;

virtual bool XLAPIENTRY rightToLeft() const = 0;

virtual void XLAPIENTRY setRightToLeft(bool rightToLeft = true) = 0;

virtual void XLAPIENTRY setAutoFitArea(int rowFirst = 0, int colFirst = 0, int rowLast = -1, int colLast = -1) = 0;

virtual void XLAPIENTRY addrToRowCol(const TCHAR\* addr, int\* row, int\* col, bool\* rowRelative = 0, bool\* colRelative = 0) = 0;

virtual const TCHAR\* XLAPIENTRY rowColToAddr(int row, int col, bool rowRelative = true, bool colRelative = true) = 0;

virtual ~ISheetT() {}

};

}

#endif

(8)、libxl.h

#ifndef LIBXL\_CPP\_H

#define LIBXL\_CPP\_H

#define LIBXL\_VERSION 0x03060500

#include "IBookT.h"

#include "ISheetT.h"

#include "IFormatT.h"

#include "IFontT.h"

namespace libxl {

#ifdef \_UNICODE

typedef IBookT<wchar\_t> Book;

typedef ISheetT<wchar\_t> Sheet;

typedef IFormatT<wchar\_t> Format;

typedef IFontT<wchar\_t> Font;

#define xlCreateBook xlCreateBookW

#define xlCreateXMLBook xlCreateXMLBookW

#else

typedef IBookT<char> Book;

typedef ISheetT<char> Sheet;

typedef IFormatT<char> Format;

typedef IFontT<char> Font;

#define xlCreateBook xlCreateBookA

#define xlCreateXMLBook xlCreateXMLBookA

#endif

}

#endif

（9）、setup.h

#ifndef LIBXL\_SETUP\_CPP\_H

#define LIBXL\_SETUP\_CPP\_H

#ifdef LIBXL\_STDCALL

#define LIBXL\_CALLING \_\_stdcall

#else

#define LIBXL\_CALLING \_\_cdecl

#endif

#if !defined(LIBXL\_STATIC) && (defined(\_MSC\_VER) || defined(\_\_WATCOMC\_\_))

#ifdef libxl\_EXPORTS

#define XLAPI \_\_declspec(dllexport)

#else

#define XLAPI \_\_declspec(dllimport)

#endif

#define XLAPIENTRY LIBXL\_CALLING

#else

#ifdef libxl\_EXPORTS

#define XLAPI \_\_attribute\_\_ ((visibility ("default")))

#else

#define XLAPI

#endif

#if defined(\_\_MINGW32\_\_)

#define XLAPIENTRY LIBXL\_CALLING

#else

#define XLAPIENTRY

#endif

#endif

#endif