JAIST Repository
https://dspace.jaist.ac.jp/

Title	CafeOBJを用いた在庫管理問題の記述と評価
Author(s)	坂本，淳誌
Citation	
Issue Date	2011-03
Type	Thesis or Dissertation
Text version	author
URL	http://hdl.handle.net/10119/9746
Rights	
Description	Supervisor:緒方和博，情報科学研究科，修士

JAIST
JAPAN
ADVANCED INSTITUTE OF
SCIENCE AND TECHNOLOGY

Japan Advanced Institute of Science and Technology

修 士 論 文

CafeOBJ を用いた
在庫管理問題の記述と評価

北陸先端科学技術大学院大学
情報科学研究科情報科学専攻

坂本 淳誌

2011 年 3 月

修 士 論 文

CafeOBJ を用いた
在庫管理問題の記述と評価

指導教官　緒方 和博 准教授

審査委員主査　緒方和博 准教授
審査委員　二木厚吉 教授
審査委員　平石邦彦 教授

北陸先端科学技術大学院大学
情報科学研究科情報科学専攻

s0810027 坂本 淳誌

提出年月: 2011 年 2 月

Copyright © 2011 by Sakamoto Atsushi

概 要

ソフトウェアの開発現場において、仕様に関するトラブルが多く発生している。これらの対応として、仕様を数学的に記述・分析することで品質の高いシステムを効率よく開発する手法であるフォーマルメソッドに関心が向けられている。このフォーマルメソッドの導入事例として、FeliCa の IC チップのファームウェアが挙げられる。FeliCa は使用されている分野が非常に多岐にわたっており、不具合が存在することはあってはならない。この FeliCa を携帯電話に不具合なく組み込むため、仕様記述言語である VDM++での仕様の記述及びテストを行い、FeliCa IC チップのファームウェアの作成を行った。

ファームウェア作成後の調査において、仕様の理解不足と見落としを除き、仕様に関しての不具合原因が非常に小さいことが報告されている。CafeOBJ は VDM++同様に形式仕様言語であり、代数に基づく実行可能な形式手法言語である。しかし、現在までに CafeOBJ で記述された仕様からプログラムが実装された例はない。そこで本論文では、形式手法言語 CafeOBJ で記述された仕様を仕様書としてプログラムの実装が行い、そのときに得られた効果や影響を評価・考察する。

本論文では、対象とする問題を在庫管理問題とした。在庫管理問題は情報処理学会で出題された、プログラムの設計技法に関する問題である。この問題に関して、数多くの設計技法の研究がされているが、これらの研究では仕様の記述を行うか、仕様記述言語でプログラムを作成している事例のみである。つまり、実際に仕様を元にプログラムが作成されていない。本研究では要求に従って仕様書を作成する段階からプログラムを実際に作成することで、CafeOBJ が開発において与える効果などをより実際の開発に近い形で評価・考察することができる。また、実際の開発に近づけるため、実装を行うプログラム言語は採用している分野が幅広い C 言語とした。

CafeOBJ で仕様を記述する前に、対象とする在庫管理問題について問題の要求を深く理解をする必要がある。本論文では問題理解のために在庫管理問題のデータ構造とその手順について、UML 図からクラス図とアクティビティ図を採用した。この 2 つの UML 図を作成することで、手順と求める要求について理解に努めた。

次にこの CafeOBJ の仕様を仕様書とし、C 言語で実装を行う。この実装において、可能な限り仕様書に近い形の実装を行いたいと考え、実装の前に変換規則を提案する。提案した変換規則は、線形リストの使用、モジュールと関数の取り扱い、データ構文の 3 つである。この変換規則を用いて C 言語でプログラムの実装を行い、CafeOBJ がプログラムの実装においてどのような影響を与えるか調査・考察する。

目 次

第 1 章　はじめに **1**

第 2 章　対象問題 **4**

第 3 章　UML での記述 **6**
3.1 クラス図 6
3.1.1 クラス図の概要 6
3.1.2 クラス図による記述 7
3.2 アクティビティ図 9
3.2.1 アクティビティ図の概要 9
3.2.2 アクティビティ図による記述 9

第 4 章　CafeOBJ での記述 **12**
4.1 CafeOBJ による記述 12
4.1.1 データ構造の記述 12
4.1.2 実行部の記述 14
4.2 UML へのフィードバック 15

第 5 章　C 言語での実装 **18**
5.1 変換規則 18
5.2 データ構造の実装 20
5.3 動作部の実装 22

第 6 章　評価・考察 **25**

第 7 章　まとめ **30**

第1章　はじめに

近年のソフトウェア開発において、曖昧な仕様が起因となるトラブルが課題となっている。現在のソフトウェア開発は、クライアントからの依頼を担当者がシステムの分析を行い仕様書を作成、その仕様書を基に開発者が実装を行う。しかし、この仕様が明確でない場合、開発者の不注意や思い込みで生じた誤りを修正する際に担当者は何が間違いであるか、拠り所がわからなくなる。

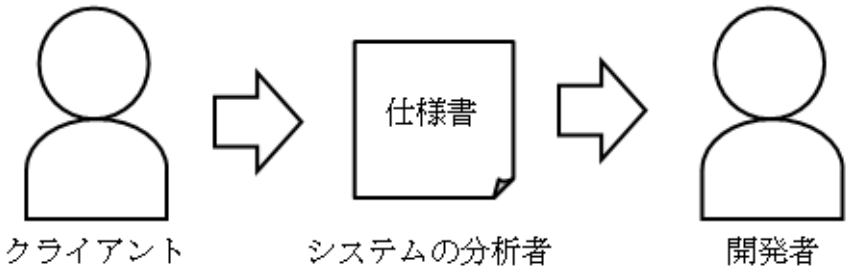

図 1.1: 現在のソフトウェア開発

仕様の記述は最終的なソフトウェアの成否を決定する重要な活動であり、何を作るのか、何を作ったのかを厳密に記述、メンテナンスすることで、開発だけでなく運用や保守も正確かつ精密に行うことができ、またプロジェクトマネージメントやコミュニケーションも建設的に収束させることができるようになる。

仕様の記述に関し、現在、フォーマルメソッド (formal methods、形式手法) に関心が向けられている [1]。フォーマルメソッドは開発対象を数学的に記述し、品質の高いシステムを効率よく開発する手法である。

フォーマルメソッドは現在までに多種多様な理論や手法、ツールが提供されている。現在までに、Jonathan Bowen の web ページには 100 近いフォーマルメソッドやツールが列挙されている。このフォーマルメソッドは大きく分けて、モデル指向型フォーマルメソッドと性質指向フォーマルメソッドに分けられる。

モデル指向型フォーマルメソッドは開発対象となるシステムを抽象的なモデルとして、論理学と集合論を基にした数学的実態を用いて表現することで機能や性質を記述分析する。この抽象的なモデルで、システムが何をするかに注目し実現の詳細に依存しないシステムの本質的性質を抽出して厳密に記述することを目的としている。この抽象的な記述

がシステムの機能に関する仕様を表しており、形式仕様 (formal specification) と呼ばれている。このモデル指向型フォーマルメソッドの代表的なものとして、Z や VDM が挙げられる。

対して、性質指向フォーマルメソッドはシステムが有する性質を公理として記述する。このフォーマルメソッドには代位数仕様に基づく方法も含まれており、代表的なものとして Larch や OBJ が挙げられる。

これらフォーマルメソッドは多種多様であるが、その利用に関して、近年以下の 3 段階が示されている。

1. 形式仕様記述

   数学的な記法を用いて厳密な仕様を記述する。この記述を基にしてプログラムを開発する。証明や分析までは行わない。

2. 形式的開発および検証

   プログラムの性質の証明や、詳細化により仕様からプログラムを開発する。安全性やセキュリティにかかわる高信頼性システムの開発に用いられることが多い。

3. 機械支援による証明

   定理証明器や証明支援器を用いてプログラムの性質を証明する。システムの不具合が致命的な影響を及ぼす絶対的な信頼性が要求に対して、特に核心部分について適応される。

フォーマルメソッドの最も簡単な利用方法である形式仕様記述においても、フォーマルメソッドの効果が大きいを言われている。対象のシステムの機能や構成を厳密に記述することにより、問題の理解が深まると言われている。

これらフォーマルメソッドを導入した事例として、携帯電話組み込みモバイル Felica IC チップのファームウェアが挙げられる [2]。Felica は電子マネーや公共機関の乗車・定期券、クレジットカード、ドアの鍵、身分証明証等非常に多岐にわたり使用されている。そのため、誤動作やバグが存在すれば経済に非常に大きな影響を与えかねないため、プログラムの仕様や記述ミスが許されない。そこで技術者たちは形式手法言語の 1 つである VDM++ を導入して、ファームウェアの形式仕様記述を行い、C/C++、アセンブラ言語を用いてプログラムの作成を行った。

このプロジェクトの期間は 3 年 3ヶ月、平均年齢が約 30 歳の 50〜60 名からなるメンバーで行われた。導入当初、形式仕様記述手法に関する知識・経験のあるメンバーはいなかった。この状態からスタートしたプロジェクトではあったが、ツールベンダが開催する講習会への参加やプロジェクト独自の研究・自習によりスムーズに導入ができた。

そして最終的に作成されたファームウェアは、仕様の理解不足や見落としなどの人為的なミスを除いた仕様自体の不具合は 3%と非常に小さいものであった。また、仕様記述のフレームワーク完成後、仕様策定技術者が仕様策定業務を行った時間は、単純比較こそで

きないものの、このプロジェクトの実装やテスト工程における開発効率と同等であり、大きな混乱も見られなかった。

このように、フォーマルメソッドは開発の現場に置いて非常に有効的であるといえる。

このフォーマルメソッドの研究として、形式手法言語 CafeOBJ がある。CafeOBJ は代数に基づく実行可能な形式仕様言語であり、検証やテストなどをパソコン上で実行することが可能である。しかし、CafeOBJ で記述された仕様を仕様書として、実際にプログラムが作成された事例は報告されていない。

そこで本研究では、実際の問題を事例とし、CafeOBJ でラピッドプロトタイピング (rapid prototyping) を行い仕様書を記述、プログラムを作成し、このときに得られる効果や影響の調査を行う。ラピッドプロトタイピングは問題をプログラムで試作する手法である。開発段階で要求や詳細、デザインを明確にすることが可能であるため、仕様について早期に議論し、完成に近い仕様を作成することができる。[1]

本研究の特色は、実際に問題を挙げ CafeOBJ で仕様を記述することである。要求に従って仕様書を作成する段階からプログラムを実際に作成することで、CafeOBJ が開発において与える効果などをより実際の開発に近い形で評価・考察することができる。

なお、本研究ではプログラムを C 言語で作成するものとした。これは仕様人口が多い、使用できるプラットフォームが多様である、採用している分野が幅広いという C 言語の特徴から採用した。

第2章　対象問題

本研究では実際の問題を例としてプログラムの作成を行う。本研究で対象とする問題は、1984 年の情報処理学会主催の「プログラムの設計技法の実用化と発展」と題したシンポジウムにて出題された在庫管理問題である [4]。この問題は多くのアプローチ法で研究、仕様の記述が行われている。

在庫管理問題は、元々依頼の受理と最適な倉庫管理を行う受付係の仕事を自動化する問題であった。しかし、翌年の同学会で問題定義をまとめ出題された在庫管理問題 [5] では、原問題に含まれていた、「倉庫内のコンテナの数は出来る限り最小にしたい」という条件が削除された。この条件において、最小化の方式自身が難しい数学的問題であり、プログラム設計技法の比較のためには不適当であると判断されたためである。

以下に概要を記す。

> 在庫管理問題はある酒類販売会社の受付係の仕事を自動する問題である。毎日数個の、ビン詰めの酒が積載されたコンテナが搬入される。この酒の取扱銘柄は約 200 種ある。倉庫係は、コンテナを受け取りそのまま倉庫に保管、積荷票を受付係へ手渡す。また、受付係からの指示により内蔵品を出庫する。なお、内蔵品は別コンテナに詰め替えたり、他の場所へ保管しないものとする。
>
> さて、受付係は依頼者から依頼を受け、倉庫係へ依頼された商品の出庫指示を行うことになっている。1 件の依頼では 1 銘柄のみに限られており、在庫がないか数量が不足している場合はその旨を依頼者へ連絡し、在庫不足リストに記入を行う。そして当該品の積荷が必要数あった時点で不足品の出庫指示をする。

この問題に関して、現在までに多くの研究がされている。雨宮真人らはデータフロー概念を用いたプログラム設計法を用いてプログラムの仕様記述を行っている [6]。久世和資はデータの流れ (ストリーム) をプログラムに取り入れたプログラム言語 stella を用いて仕様記述を行っている [7]。

しかしながらこれらの研究は仕様を形式的に記述するか、仕様記述言語を用いてラピットプロトタイピングを行っているものが大部分である。この問題の目的は、新しいプログラム技法を用いて共通のプログラム作成問題を解くことでその相違点や効果等を明確にすることである。そのため、プログラムの設計のみではなく、実際のプログラムを作成する段階においての効果も比較しなければならないのではないかと考える。そこで、本研究で実際にプログラムの作成を行い、開発に与える効果や影響の調査・考察を行う。

なお、プログラムの作成のおいて、入力データのエラー処理などは簡略に扱ってもよいので、本研究では、誤字等の入力データに関してはないものとして扱う。また、出庫依頼と在庫不足リストに関して、出庫依頼の順序を入れ替えたもののリストが在庫不足リスト

となっているため、本研究では在庫不足リストは出庫依頼のリストと同様のものであるとする。

第3章　UMLでの記述

CafeOBJ での記述を行う前に、問題の全体像や構造を深く理解する必要がある。そこで本研究では、まず UML(Unified Modeling Language) を用いて記述を行う。

UML はグラフィカルな要素をいくつも組み合わせてダイアグラム (diagram) を表現する [8]。ダイアグラムはあるシステムを複数の視点で表現した図のことである。この視点 (ビュー) を総じてモデル (model) と呼ぶ。例えば、システムの動作をユーザーの視点で示したものをユースケース (use case) と呼び、このユースケースをダイアグラムで視覚化したものが UML ダイアグラムの 1 つであるユースケース図 (use case diagram) である。他にも、オブジェクトの状態の変化を表現するステートチャート図 (state diagram) や、複数のオブジェクトが互いに与える作用を時系列に沿って並べたシーケンス図 (sequence diagram) などがある。

開発者らはこれら UML ダイアグラムを組み合わせ、また拡張することで様々なモデルを表現する。ただし、これらすべてのダイアグラムを作成する必要はない。むしろほとんどの UML モデルはこれらのダイアグラムの一部のみを使用して表現する。

単一のシステムを複数の視点で表現する理由は、システム開発に複数の関係者が関与し、それぞれ異なった関心を持っているからである。例えば、あるシステムを開発しようと考えた時、ハードウェアを設計する人とソフトウェアを設計する人、クライアントとユーザーの視点はそれぞれ異なる。そこで複数の UML ダイアグラムを使用することで、あらゆる視点からニーズを組み込み、あらゆる関係者から支持されるシステムが設計できる。もちろん、UML モデルが表現するのはシステムの動作であり、実装方法ではない。

本研究では UML ダイアグラムのうち、あらゆる事物を共通の属性をもち一定の振る舞いをする種類に分割したクラス図 (class diagram) とユースケースやオブジェクトの振る舞いを複数の作業単位でまとめたアクティビティ図 (activity diagram) を用いて表記を行うことで問題の全体像や構造について理解を深める。

3.1　クラス図

3.1.1　クラス図の概要

まず、データ構造を理解するため、クラス図の記述を行う。

クラス (class) は共通の属性 (attribute) を持ち、同じように振る舞う操作 (operation) を持つ事物の集合である。クラスは水平線で区分された 3 つの区画を持つ長方形のアイコン

で表現される。一番の上の区画にはクラス名、中間の区画には属性、下部の区画には操作を記述する。このクラスアイコンを線で結び、その関連を示した図がクラス図となる。

また、クラスアイコン同士を結ぶ概念の繋がりを関連 (association) と呼び、関連に参加するクラスのオブジェクトの数を多重度 (multiplicity) と呼ぶ。例えば、野球のチームとゲームに参加する選手は関連があり、１つのチームに対して９人の選手がゲームに参加するため、この例のクラスは１対９の多重度で関連があるといえる。

これらを用いて、作成するシステムのデータ構造を記述する。なお、本研究ではデータ構造を理解するためにクラス図を用いるため、操作に関しての記述は省略する。

3.1.2 クラス図による記述

データ構造に関して、問題より積荷票・出庫指示書・在庫不足リストの３つが考えられる。それぞれのデータ構造に関して以下で述べる。

1. 積荷票のデータ構造

   まず、積荷票のデータ構造をクラス図を用いて記述する。

   積荷票をクラスとすると、属性はコンテナ番号、搬入年月、日時、内蔵品名と数量の繰り返しとなる。しかし、内蔵品名と数量の繰り返しはクラスの属性の数を変動させてしまうため、新たに内蔵品と品数を属性とするリストを新しいクラス作成し、そのリストを内蔵品名と数量の繰り返しの代わりとする。

   また、積荷票のリストを作成すると在庫情報を示すリストとなるので、本研究では在庫情報を積荷票のリストとする。

2. 出庫指示書のデータ構造

   続いて、出庫指示書のデータ構造をクラス図を用いて記述する。

   出庫指示書は注文番号、送り先名、コンテナ番号・品名・数量・空コンテナ搬出マークの繰り返しから構成される。積荷票のときと同様、繰り返しはクラスの属性の数を変動させてしまうため、コンテナ番号・品名・数量・空コンテナ搬出マークを属性とする新しいリストをクラスとして作成し、そのリストを代わりとする。

3. 在庫不足リストのデータ構造

   在庫不足リストのデータ構造をクラス図を用いて記述する。

   在庫不足リストは出庫依頼のリスト[1]であるので、出庫依頼のクラスを作成し、そのクラスを在庫不足リストのクラスとする。出庫依頼の属性は品名、数量、送り先名で構成される。

これらのデータ構造をクラス図を用いて記述した図 3.1 を示す。

[1]第２章の対象問題を参照

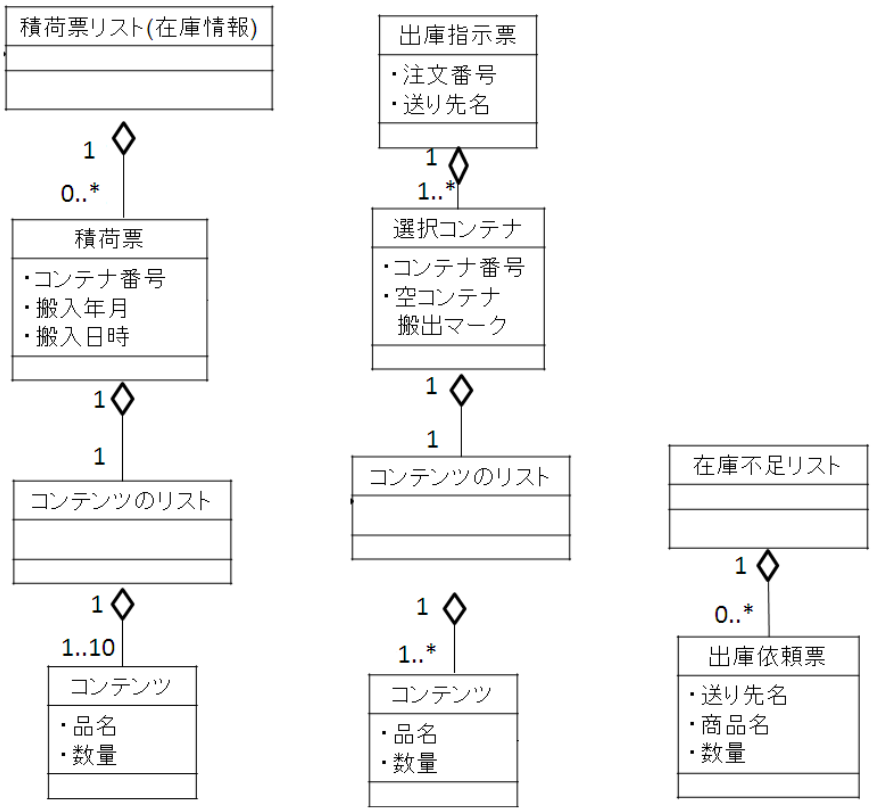

図 3.1: クラス図を用いたデータ構造

3.2　アクティビティ図

3.2.1　アクティビティ図の概要

続いて、システムの流れを理解するため，アクティビティ図の記述を行う.

手順の流れや条件判断を記述するフローチャート図と同様に、操作やプロセスの途中で起こっていることを簡素化した図がアクティビティ図である。

アクティビティ図では開始状態を内側を塗りつぶした円状のアイコンで表し、終了状態を二重円の内側を塗りつぶしたアイコンで表すことができる。また、手順をアクティビティと呼び、丸みのある細長い長方形のアイコンで表す。アクティビティの処理は完了とともに自動的に次のアクティビティへ渡され、この流れはアイコンをつなぐ矢印で表す。

状況により次に行うアクティビティを変更したいなど、条件判断も表記することができる。条件判断を表す方法は２つあり、１つはアクティビティアイコンから複数のアクティビティアイコンに向けて直接矢印を引く方法であり、もう１つは、小さなひし形のアイコンを使用し、ここから複数のアクティビティアイコンに向けて矢印を引く方法である。本研究では、分岐であることを明確にするために、小さなひし形のアイコンから矢印を引く後者を採用する。なお、条件判断によって処理経路が分岐した場合は各経路は相互に排他的な関係となる。

3.2.2　アクティビティ図による記述

受付係の仕事に関して、依頼者から出庫依頼を受理した場合と倉庫係から積荷票を受理した場合の２つが考えられる。それぞれのアクティビティに関して以下で述べる。

1. 出庫依頼を受理した場合

   出庫依頼を受理、つまり依頼者から注文を受けた場合を考える。出庫依頼を受理した時、受付係は依頼者から品名とその数量、送り先の情報を得る。

   依頼された品名の在庫状況を確認する前に、扱い銘柄は約 200 種類と限定されていることから、取扱っている銘柄か確認を行うことができる。取り扱っていない銘柄であった場合、特に動作の指定はされていないため本研究では、取り扱っている銘柄でない旨を通知し出庫依頼を受理できなかったものとして扱う。

   出庫依頼を受理した場合、出荷できるか在庫状況を調べる。本研究では在庫情報を積荷票のリストで構成しているため、依頼された品名があるコンテナのリストを作成した後に在庫の本数を合計、依頼本数を比較して充足していれば出荷、不足しているならば依頼者に電話連絡と在庫不足リストへの追記を行う。本研究では、在庫不足リストへの追記は依頼内容をそのまま追記することとする。なお、商品の在庫数を記したデータを別途用意する方法も考えられるが、データと実際の本数の齟齬

が生じることがあること、どのコンテナから出荷するかを決定する際にも在庫情報である積荷票のリストを調べる必要が考えられるため、本研究では採用しなかった。

在庫が充足していた場合、在庫状況を調べるアクティビティ時に出庫依頼の品名が積載されている積荷票のリストが作成されているので、このリストから出庫するコンテナを選択する。本研究では、選択するコンテナは入荷順に選択していくものとする。リストは入荷されたコンテナ順に並んでいるため、リストの先頭から依頼された本数を満たすまで選択する。

選択後、空となるコンテナであるかどうか調べ、空となるコンテナに印をつけ、選択したコンテナのリストから出庫指示票を作成、依頼受付を完了する。

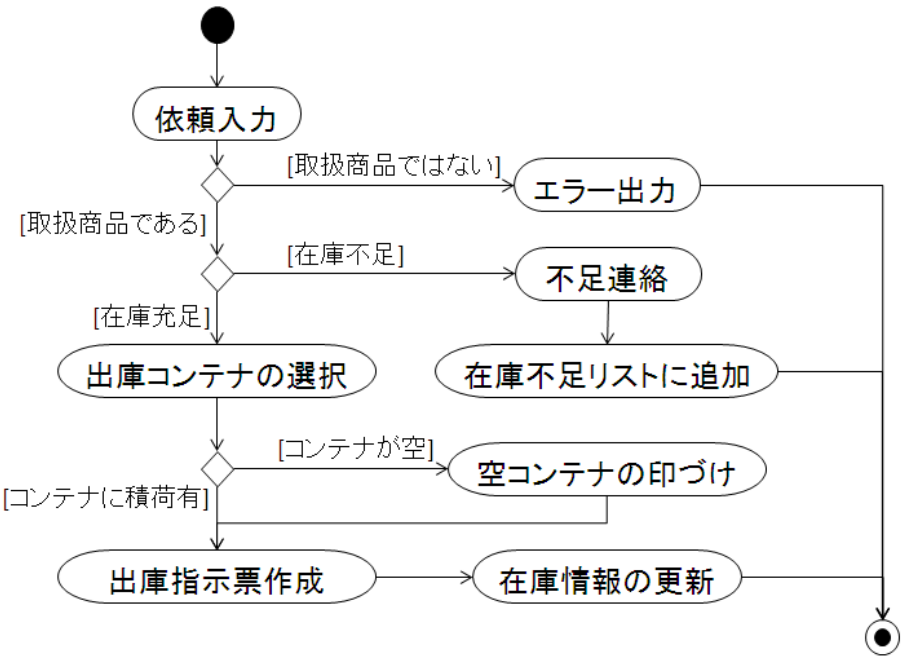

図 3.2: 出庫依頼票受理時のアクティビティ図

2. 積荷票を受理した場合

積荷票を受理、つまり商品が入荷した場合を考える。積荷票を受理した時、積荷票の内容からコンテナ番号、搬入年月、日時、内蔵品名と数量の繰り返しという情報を得る。

問題において、後にその品目に対する積荷が十分にあった場合、依頼者に対して出庫する旨が記述されている。よって、届いた積荷票内に在庫不足リストに記載されている品名の有無を確認し、さらに有った場合に積荷と在庫の数量の合計が在庫不足リストに記載されている数量と比べて不足なくあるか確認を行う。

積荷と在庫の数量の合計が充足している場合、出荷するコンテナを選択、出荷後にコンテナが空となるコンテナを調べ、空となるコンテナをに印をつける。その後出庫指示票を作成し、在庫不足リストから出荷する依頼票を削除を行う。

この一連の動作を繰り返し、積荷票から出荷する商品がなくなった後に積荷票を在庫のリストに追加を行う。

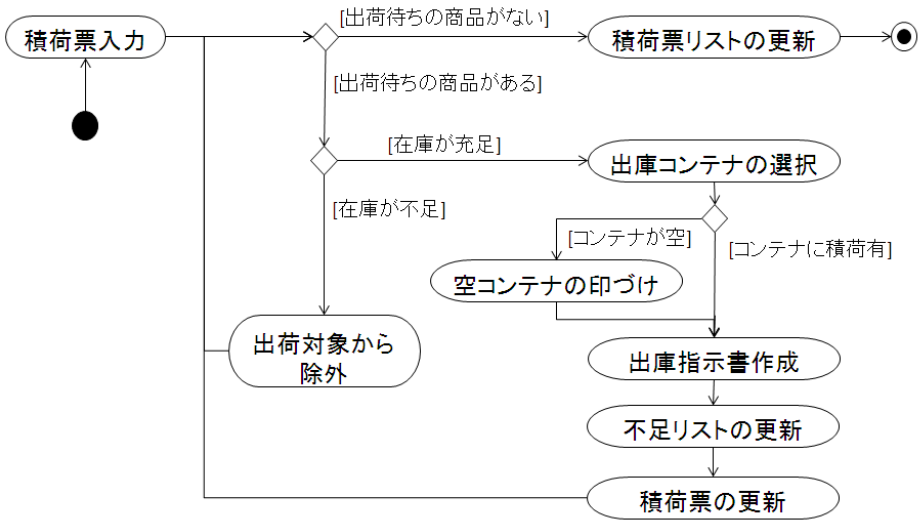

図 3.3: 積荷票受理時のアクティビティ図

第4章　CafeOBJでの記述

前章で作成したデータ構造が記述されたクラス図及びプログラムの手順が記述されたアクティビティ図を参考に、CafeOBJ での記述を行う。なお、本研究で作成した全ソースコードは付録に収録するものとし、本論文では一部を抜粋して記述する。

4.1　CafeOBJ による記述

4.1.1　データ構造の記述

はじめに、積荷票のデータ構造についての記述を行う。積荷票はコンテナ番号、搬入年月、日時、内蔵品のリストの 4 つの属性からなる。また、内蔵品は内蔵品名と数量の 2 つの属性からなる。よって、2 つと 4 つの属性をもつクラスの記述をそれぞれ行う。

まず、2 つの属性をもつクラスのモジュール名を PAIR とし、新しいソートとして Pair を作成する。この PAIR モジュールは汎用性を考え、PAIR モジュールを定義する際に 2 つの型を引数とするものとする。このように型を限定しないことで、PAIR モジュールが内蔵品以外でも使用できるようにした。ソート Pair はモジュール PAIR を使用する際に定義される型をもつソートであり、[ ] で囲まれ, で区切られたソートとする。

同様に、4 つの属性をもつクラスのモジュール名を QUADRUPLE とし、新しいソート Quadruple を作成する。また、Quadruple は< >で囲まれるものとする。

これらを CafeOBJ で記述したものが図 4.1 となる。

```
mod! PAIR(X :: TRIV, Y :: TRIV) {
  [Pair]
  op [_,_] : Elt.X Elt.Y -> Pair
}

mod! QUADRUPLE(V :: TRIV,X :: TRIV,Y :: TRIV,Z :: TRIV) {
  [Quadruple]
  op <_,_,_,_> : Elt.V Elt.X Elt.Y Elt.Z -> Quadruple
}
```

図 4.1: CafeOBJ のソースコード (上:PAIR, 下:QUADRUPLE)

次にリストについての記述を行う。リストの記述を行うとき、PIAR のモジュール時と同様に引数の型を使用時に定義するモジュールにすれば、他のリストを作成する際にも使用できるため、リストのモジュールの構造だけの記述を行う。

リストのモジュール名を FORLIST とし、新しくソート List を作成する。FOTLIST モジュールは型のない引数をとり、::をこの引数と List で挟んだものを List とすると、リスト構造が作成できる.

ただし、このままではリストの末尾の判断やリストが空である場合などの判断が汎用化されていない。そこで空を意味する empty という関数を作成し、これをリストの末尾、または空のリストとする。

次にリスト同士の結合を考える。リストの結合子を@とする。epmty と List で結合した場合は empty を削除し List を返す、先頭に使用時に定義される引数の型の要素をもつリストと他のリストの順でリストが存在した場合は、要素をリストの先頭にしてリスト同士を結合する関数を作成する。この 2 つの関数でリストの結合を行うことができる。

これらを CafeOBJ で記述したものが図 4.2 となる。

```
mod! FORLIST(X :: TRIV) {
  [List]

  op empty : -> List
  op _::_ : Elt.X List -> List
  op _@_ : List List -> List

  var E : Elt.X
  vars L L1 L2 : List
  var N : Nat

  eq empty @ L2 = L2 .
  eq (E :: L1) @ L2 = E :: (L1 @ L2) .
}
```

図 4.2: CafeOBJ のソースコード (FORLIST)

以上の CafeOBJ で記述されたモジュールを用いて、内蔵品、内蔵品のリスト、積荷票、積荷票のリストの順に記述を行う。

内蔵品は品名と数量の 2 つの属性を持っているため、PIAR モジュールを用いて記述を行う。品名は文字列なので STRING、数量は自然数なので NAT を用いる。また、区別化を行うため Pair というソート名を Content に変更をする。

次に内蔵品のリストを記述する。内蔵品のリストのモジュール名を CONTENTLIST とし、リストは FORLIST モジュールを用いて記述を行う。同様に区別化を行うため、List というソート名を ContentList に変更する。

次に積荷票の記述する。積荷票はコンテナ番号、搬入年月、搬入日時、内蔵品のリストの 4 つの属性を持っているため、QUADRUPLE モジュールを用いる。コンテナ番号、搬入年

月、搬入日時は自然数で表現できるのでNAT、内蔵品のリストは作成したCONTENTLISTで表現する。また、区別化のためソートQuadrupleをStockSheetに変更する。

目的であった積荷票のリストを記述する。積荷票のリストのモジュール名をSTOCKSHEETLISTとし、内蔵品のリスト時と同様にFORLISTモジュールを用い、ListをStockSheetListに変更する。

積荷票のリストを、CafeOBJを用いて記述を行ったものが図4.3となる。同様に在庫不足リストや出庫指示票のリストの記述も行った。なお、実際のソースコードにおいては内蔵品のリスト同士を結合する関数などが記述されているが、クラス図を参考に記述を行っているのでここでは記述しない。

```
-- 内蔵品
-- (STRING::品名, NAT::数量の集合)
mod! CONTENT principal-sort Content {
  pr(PAIR(STRING, NAT) * {sort Pair -> Content})
}

-- 内蔵品のリスト
-- (CONTENT::内蔵品)
mod! CONTENTLIST principal-sort ContentList {
  pr(FORLIST(CONTENT) * {sort List -> ContentList}) .
}

-- 積荷票
-- (NAT::コンテナ番号, NAT::搬入年月,
--  NAT::搬入日時, CONTENTLIST::内蔵品の集合)
mod! STOCKSHEET principal-sort StockSheet {
   pr(QUADRUPLE(NAT, NAT, NAT, CONTENTLIST)
     * {sort Quadruple -> StockSheet})
}

-- 積荷票の集合
-- (STOCKSHEET::積荷票)
mod! STOCKSHEETLIST principal-sort StockSheetList {
  pr(FORLIST(STOCKSHEET) * {sort List -> StockSheetList})
}
```

図 4.3: CafeOBJ のソースコード (STOCKSHEETLIST)

4.1.2 実行部の記述

次に実行部の記述を行う。前章で作成したアクティビティ図から、実行部は出庫依頼または積荷票を受理した場合の2つの場合が考えられる。これらの場合についての記述を行っていく。

出庫依頼があった場合、まず依頼品が取扱商品であるかの確認を行う。取扱商品の名前は既知であるため、リストを作成するFORLISTを用いて取扱商品の名前を列挙した文字列のリストを作成する。このリストの文字列と依頼品名を比較し、リスト内に依頼品名が存在するか確認する関数を作成する。

文字列のリストを作成するモジュール名を PRODUCTSAKELIST とする。FORLIST モジュールの引数に文字列を意味する STRING ををわたす。また、区別化のためソート List を SakeNameList に変更する。

さらにこのモジュールを呼び出して取扱商品のリストを作成する PRODUCTSAKELIST モジュールを作成する。このモジュールで引数を取らずに文字列のリスト SakeNameList を返す関数 pronamelist を作成する。この関数は商品名を::で列挙している関数であり、PRODUCTSAKELIST モジュールを宣言しているモジュールにおいて取扱商品のリストを呼び出すことができるようになる。

これまでのモジュールにより、依頼品が取扱商品であるか確認を行うことができるようになったので、確認を行うモジュール及び関数の記述を行う。

取扱商品か確認を行うモジュール名を CHECKPRODUCTSAKE とし、関数を checkproduct とした。出庫依頼があった場合に得られる情報は出庫依頼票に記載される情報のみであるので、確認を行う関数の引数を出庫依頼票とし、品名の有無を調べる関数であるので、その返却値は true か false を返す Bool 型とする。

出庫依頼票を受理した関数 checkproduct は依頼票に記載される品名だけを使用するために送り先・品名・数量から品名だけを抜き出し、品名を引数とする同一名関数 checkproduct を呼び出す。呼び出された関数は、さらに取扱商品のリストと順次比較していくため品名と取扱商品のリスト pronamelist を引数とする同一名関数 checkproduct の呼び出す。品名と取扱商品のリストを受け取った checkproduct は、取扱商品のリストが空であった場合は取扱外商品であるとして false を返す。取扱商品のリストの先頭の要素が品名と同じであった場合は取扱商品と見なし true を返す。先頭の要素と品名が異なる場合、先頭の要素を取り除いたリストを引数として自身を呼び出し、この該当商品が見つかるかリストが空になるまで再帰的に呼び出しを行う。

ここまでのソースを図 4.4 に示す。

このように出庫依頼及び積荷票の受理の場合において CafeOBJ の記述を行った。

4.2 UML へのフィードバック

前節において作成した UML のアクティビティ図において変更点が CafeOBJ で記述を行ったときに発見された。そこで CafeOBJ の記述を元にアクティビティ図の変更を行う。

変更が特に顕著であったのは出庫指示票を作成するステップである。変更前の図では出庫するコンテナを選択した後にコンテナの内蔵品を確認して空となるか調査・印付けを行い、出庫指示票の作成を行っていた。

CafeOBJ 記述時に新たに提案した手順は、出庫するコンテナを選択した後に出庫指示票用にデータをコンテナ毎に変換、このときに空コンテナの印付けという手順をデータ変換が終了するまで続け、出庫指示票の残りのデータを追記して作成を行う、というものである。

```
-- 取扱い商品のリスト
mod! PRODUCTSAKELIST {
  pr(FORLIST(STRING) * {sort List -> SakeNameList}) .
}

mod! PRODUCTNAMELIST {
  pr(PRODUCTSAKELIST) .

  op pronamelist : -> SakeNameList .
  eq pronamelist = "正宗" :: "菊姫" :: empty .
}

-- 取扱商品か確認を行い、Boolを返す
mod! CHECKPRODUCTSAKE {
  pr(PRODUCTNAMELIST) .
  pr(REQUESTSHEET) .

  var  CN : Nat
  vars ST SN : String .
  var  SNL : SakeNameList .
  vars SNA SNB : String . -- 酒の名前

  op checkproduct : RequestSheet -> Bool .
  op checkproduct : String -> Bool .
  op checkproduct : String SakeNameList -> Bool .

  eq checkproduct(< ST, SN, CN >) = checkproduct(SN) .
  eq checkproduct(SNA) = checkproduct(SNA, pronamelist) .
  eq checkproduct(SNA, empty) = false .
  eq checkproduct(SNA, SNA :: SNL) = true .
  eq checkproduct(SNA, SNB :: SNL) =
    if(string=(SNA, SNB)) then true else checkproduct(SNA, SNL) fi .
}
```

図 4.4: CafeOBJ のソースコード (取扱商品か確認を行う)

これは出庫指示票を作成と空コンテナの印付けを別々に動作させることよりも、変換時に並行して処理を行うことでよりスムーズな処理が可能であり、プログラムもより簡単なものにすることができる。

この仕様の変更点を UML のアクティビティ図にフィードバックを行った図が図 4.5 である。

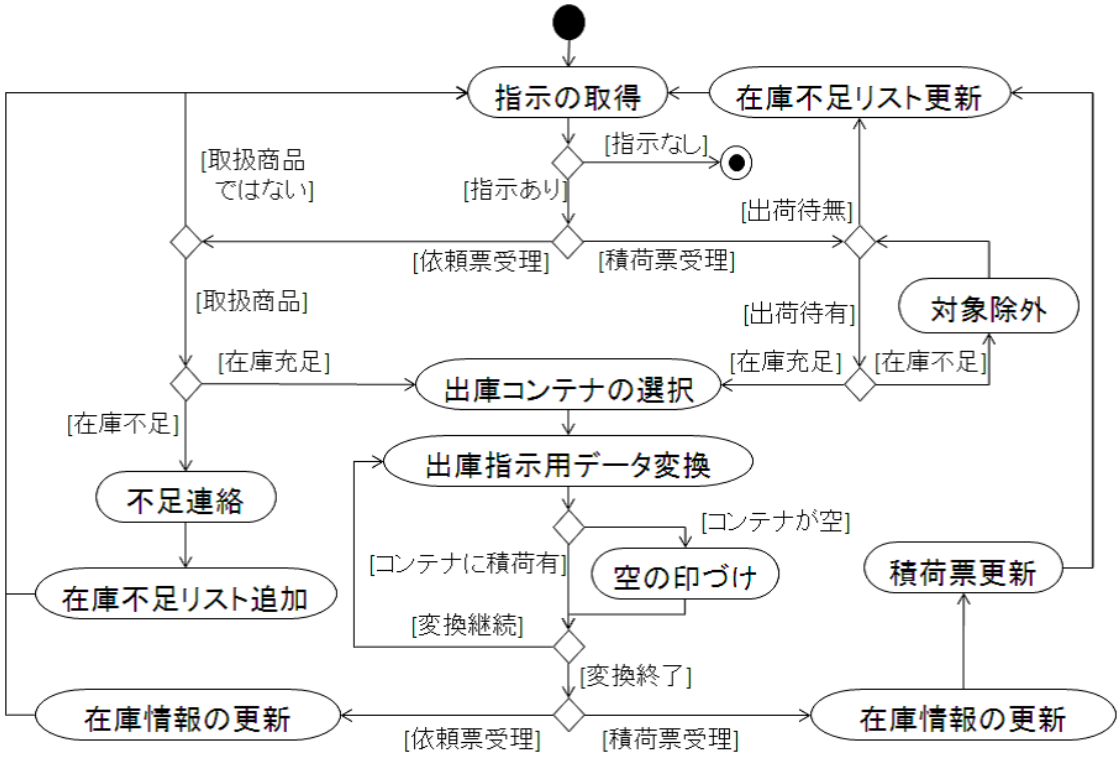

図 4.5: CafeOBJ からフィードバックを行ったアクティビティ図

第5章　C言語での実装

本章では、前章においてCafeOBJで記述を行ったソースコードを仕様としてC言語での実装を行う。

5.1　変換規則

本研究ではCafeOBJでの仕様に出来る限り近づけるため、実装前に考えた規則に従って実装を行った。

1. 線形リストの使用

   本研究ではCafeOBJでのリスト構造に出来るため近づけるため線形リストを用いる。線形リストは構造体にデータとポインタが存在するノードからなり、ポインタが次のデータを参照している。リスト構造はC言語の機能である配列を用いても実装することができるが、配列は番号が振られているなどCafeOBJの仕様から遠ざかるため線形リストを採用した。

2. モジュールや関数の取扱

   モジュールは特定の機能を持つ複数の関数からなる。しかし、C言語にはクラスの概念がないため、モジュールは無視するものとした。

   一方、関数はC言語でも関数として扱うことができる。しかし、C言語では同一名関数を許可していないため、同一名関数を用いている関数はある程度変換して実装しなければならない。本仕様では以下の3つのケースがあり、そのときの変換方法は以下のようにした。

   (a) 再帰関数

   同一名関数として、再帰的動作を行う関数がある。自身を呼び出すだけの再帰関数では変更をする必要はないが、本仕様ではリストの情報を一時的に保持するために引数を増やし、再帰的に呼び出す関数が存在しているため、独立した関数としてそれぞれ分けて考えなければならない。

   この再帰関数は使用する際に呼び出される関数と再帰を行う関数に分けて考えることができる。モジュール単位で開発を行うことを考えると、呼び出される関数が統一されているほうが混乱が生じにくい。そこで本研究では、呼び出さ

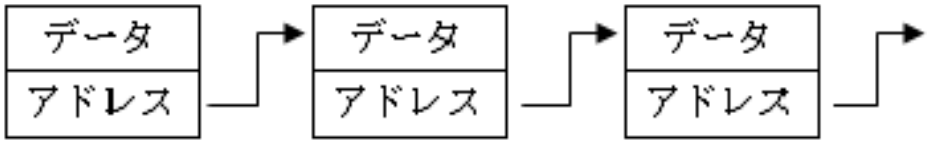

図 5.1: 線形リスト

配列番号	0	1	2	3	4	…	n
データ							

図 5.2: 配列

れる関数名はそのままにし、再帰を行う関数は関数名の末尾に繰り返しを意味する Loop を追記することとした。

(b) 同機能の関数

同じ機能を持つ関数ではあるが、引数の数や型が異なる関数がある。例を挙げると、リストを結合する関数が考えられる。2 つのリストを結合させる場合、リストの先頭にデータを結合させる場合、リストの間にデータを結合させる場合などがある。

これらの関数について、優先順位は存在していないため、再帰関数のようにどの関数名をそのままに、または変更するなど分けることができない。そこで本研究では、引数の頭文字を関数名の末尾に付けることとした。積荷票のリストを例とすると、積荷票のリストの先頭にデータをつける場合は積荷票の頭文字の S とリストの頭文字の L を末尾につけて addStockSheetListSL とし、積荷票のリスト同士を結合する場合はリストの頭文字の L を末尾につけて addStockSheetListLL とした。

しかし、異なる型で同一名関数において情報を削り、情報が少ない同一名関数を使用する関数も存在する。その場合は情報が少ない関数の優先順位をあげ、関数名をそのままとする。一方、情報が多い関数は区別化のため、引数の末尾に引数の型の頭文字をつけるものとする。

(c) 同じ引数の数と型を持つ関数

CafeOBJ の関数定義である op では分けて書かれていないため厳密には同一名

関数ではないが、特定の値のみで機能する関数がある。例を挙げると、引数として取るリストが空であった場合などである。

これは関数として分けるのではなく、場合分けとして考えることができるので、本研究ではC言語のif構文を用いて1つの関数として扱う。

3. データ構文

データ構造を表現する方法としてPIARモジュール等を用いて記述を行った。しかし、C言語ではPIARモジュールのように引数の型を定めないようなデータ構造は作成できないため、PIAR等のモジュールは使用せずに作成を行う必要がある。本研究ではPIARモジュールを廃止する代わりに構造体を使用して実装するものとした。

PAIRモジュールなどは使用できないが、その1つ上のモジュールでは内蔵品CONTENTなどのデータ構造がある。これらのデータ構造では引数の型が決まっているため、CONTENTなどのモジュールを構造体を用いて実装する。

これらの変換規則を用いて次の小節から実装を行っていく。

5.2 データ構造の実装

前章までと同様に積荷票のリストから実装を行う。

まず、内蔵品のモジュールを作成する。先述のとおり、PIARモジュールは使用できないため、規則に従って構造体を使用して表現する。内蔵品は文字列と自然数であるので、charのポインタとintの構造体で記述する。この構造体の名前をモジュール時と同様にCONTENTとする。

次に内蔵品のリストを先述の通り線形リストを用いて実装する。内蔵品のリストのデータはCONTENTである。しかし、CONTENT自体が空である場合があるので、CONTENTはポインタで作成する。また次のデータを表すポインタの型は作成している内蔵品のリストとなる。この構造体の名前をモジュール名と同様にCONTENTLISTとする。なお、内蔵品のリストではデータをdata、ポインタをnextとして作成した。

次に積荷票を作成する。積荷票はコンテナ番号、搬入年月、搬入日時、内蔵品のリストを属性として持つ。内蔵品のリストを除きすべて自然数で表現でき、またリスト自身は空である場合があるのでポインタを用いて表現した。この構造体の名前をSTOCKSHEETとした。

最後に積荷票のリストを作成する。積荷票のリストも内蔵品のリストと同様に線形リストで作成する。リストの構造体の名前をSTOCKSHEETLISTとし、データを積荷票STOCKSHEETのポインタを用いて表現し、dataとする。また次のノードを表すポインタをSTOCKSHEETLISTのポインタを用いて表現し、nextとする。

これにより積荷票のリストが表現できた。しかし、このままではメモリの割り当てが行われていないため文字列をコピーする際などでエラーが生じてしまう。そこで使用時にメ

```c
// 内蔵品
typedef struct {
    char*   name;   // 商品名
    int     cnt;    // 本数
} CONTENT;

// 内蔵品のリスト
typedef struct _CONTENTLIST {
    CONTENT* data;
    struct _CONTENTLIST*    next;
} CONTENTLIST;

// 積荷票
typedef struct {
    int     num;      // コンテナ番号
    int     year;     // 搬入年月
    int     date;     // 搬入日時
    CONTENTLIST* content; // 内蔵品
} STOCKSHEET;

// 積荷票のリスト
typedef struct _STOCKSHEETLIST {
    STOCKSHEET*             data;
    struct _STOCKSHEETLIST* next;
} STOCKSHEETLIST;
```

図 5.3: C のソースコード (データ構造)

モリの割り当てを行うような関数の作成も行う必要がある。逆にメモリを解放しなければメモリの割り当てが残ってしまい、不具合を引き起こす原因となる。

そこで、メモリ操作を行う関数を独自に作成する必要がある。本研究の仕様でメモリの割り当てや解放が必要な場合は、リストのノードの挿入や削除を行う場合である。よって、内蔵品のリストを例に作成を行う。

これまでのデータ構造において、文字列を扱うものは char のポインタで記述を行った。内蔵品の品名も同様に char ポインタで記述が行われている。このメモリを割り当てるため malloc 関数を用いてメモリを割り当てる。メモリを割り当てるサイズは固定とし、define で定義した MAXNAMESIZE とする。

続いて内蔵品のメモリの割り当てを行う。メモリの割り当ては順次行うもので、内蔵品 CONTENT の 1 つ分の割り当てを行えばよい。そのため calloc 関数を用いて CONTENT の 1 つ分の割り当てを行えばよい。

これらの関数を用いて内蔵品のリストにメンバ値の設定を行う関数を作成する。この設定する関数を順次行うことで、リストの読み込みやノードの挿入などを行う。

設定を行う関数を setContentList とし、引数に、先頭の要素に設定を行うリスト、設定するデータ、設定後のリストの次のノードを与える。プログラムはまず、設定を行うリストの先頭の要素に、内蔵品、内蔵品の名前の順にメモリを割り当てる。割り当てられたメモリ領域にデータを設定していき、最後にリストの次のポインタに次のノードを与えればよい。

この設定を行う関数を用いて、先頭にノードを挿入する関数を作成する場合を考える。まず、リスト list のポインタをポインタ ptr にコピーを行う。次に list のポインタのメモリ

を割り当てることで空のリストが作成される。この空のリストにデータを設定し、コピーを行った ptr を次のノードにすれば先頭のノードに挿入することができるようになる。

同様に積荷票や在庫不足リストのケースを作成することで、メモリの割り当てを行う関数ができる。続いて、メモリの解放を行う関数の作成を行う。

メモリの解放も文字列の解放から行っていく。内蔵品の品名を引数に取る removeName は引数のポインタを解放するだけの関数である。内蔵品のポインタを解放する関数 removeContent は関数 removeName を用いて品名を削除後に内蔵品の解放を行う。

内蔵品のリストを解放する場合はリストのデータを順次解放していく必要がある。まずリストが空でないか確認し、空である場合は解放する必要がないのでそのまま終了する。空でない場合は先頭の要素であるデータのポインタを解放し、次の要素の有無を調べ、存在するならば空になるまで解放を行う。

これも同様に、積荷票や在庫不足リストのケースを作成することで、メモリの解放を行う関数ができる。ここまでのソースを示したものが図 5.4、5.5 である。

```
char* allocName(void) {
    return (char*)malloc(MAXNAMESIZE);
}

CONTENT* allocContent(void) {
    return (calloc(1, sizeof(CONTENT)));
}

CONTENTLIST* allocContentList(void) {
    return (calloc(1, sizeof(CONTENTLIST)));
}

void setContentList(CONTENTLIST* list, CONTENT data, CONTENTLIST* next) {
    list->data       = allocContent();
    list->data->name = allocName();
    strcpy(list->data->name, data.name);
    list->data->cnt = data.cnt;
    list->next = next;
}
```

図 5.4: 線形リストのメモリ領域の確保

5.3 動作部の実装

データ構造の実装に続き動作部を、実装前に提案した変換規則に従って実装する。なお、ここでは例として取扱商品用関数の実装を行う。

取扱商品か調査を行う関数は仕様のオペレータより 3 種類あると考えられる。なお、等式では 5 種類存在しているが、残りの 2 種類はいずれかの関数に属しているものである。

3 種類の引数はそれぞれ、出庫依頼票、品名、品名と取扱商品のリストからなる。この中で出庫依頼票の情報を削ったものが品名であり、品名と取扱商品のリストは、品名を引

```
void removeName(char* name) {
    free(name);
    return;
}

void removeContent(CONTENT* cont) {
    removeName(cont->name);
    free(cont);
    return;
}

void removeContentTop(CONTENTLIST** list) {
    if(*list != NULL) {
        CONTENTLIST* clist = *list;
        CONTENTLIST* cont  = clist->next;

        removeContent(clist->data);
        free(*list);

        if(cont == NULL) *list = NULL;
        else             *list = cont;
    }
    return;
}

void removeContentList(CONTENTLIST** list) {
    while (*list != NULL) removeContentTop(list);
    return;
}
```

図 5.5: 線形リストのメモリ領域の解放

数とする関数から呼び出される再帰関数である。よって、規則に従ってそれぞれの関数名を checkProductR、checkProduct、checkProductLoop とする。

checkProductR は出庫依頼票を引数に取っているので、依頼票に記載されている情報を削減して品名のみとし、checkProduct を呼び出す。

checkProduct は品名を引数にとり、取扱商品のリストを読み込む productNameList を用いて再帰関数 checkProductLoop を呼び出す。

checkProductLoop は checkProduct から受け取ったリストを調べ、リストが空となっているならば false を返す。空でないならば品名を比較して一致しているならば true を返す。一致しないときは再帰関数 checkProductLoop の引数に品名とリストの次のノードをわたす。

これで取扱商品の確認を行う関数が実装できた。ここまでのソースを示したものが図 5.6 である。同様に規則に従って C 言語での実装を行った。

```
BOOLEAN checkProductLoop(char *str, PRODUCTSAKELIST **list) {
    PRODUCTSAKELIST *ptr = *list;

    if(ptr == NULL) return FALSE;

    return ((strcmp(str, ptr->name)) == 0) ?
              (TRUE) : (checkProductLoop(str, &(ptr->next)));
}

BOOLEAN checkProduct(char *str) {
    PRODUCTSAKELIST *list = NULL;
    productNameList(&list);

    return checkProductLoop(str, &list);
}

BOOLEAN checkProductR(REQUESTSHEET request) {
    return checkProduct(request.name);
}
```

図 5.6: C 言語の実装部 (取扱商品の確認)

第6章　評価・考察

本研究ではCafeOBJ仕様に基づきプログラムを実装し、このプログラムは対象問題の仕様通りに動くことを確認した。本節ではそのときに得られた効果や影響、生じた問題点についてを評価・考察を行う。

仕様理解の促進

プログラムは仕様通りに作成を行う必要があるが、その仕様が曖昧であると望むプログラムが実装されないことは第1章でも述べた。今回の研究で作成したUML図でのそれが例と言える。本研究では、UML図からCafeOBJで記述を行った際にアクティビティ図に変更があった。特に大きな変更があったのは、出庫をするコンテナを選択した後から出庫指示票を作成するまでのステップである。

この変更はプログラムの合理性を考えて行った。つまり、これは記述を行うまでに理解が不足していたために起こった変更である。

このような仕様の理解に関して、Hallが提唱したフォーマルメソッドの七つの言明[1]の通りであることがわかった。これはフォーマルメソッドを用いることで、問題対象の理解が深まり、見通しが良くわかりやすいシステム記述が得られるということである。仕様記述の曖昧さや不十分さに起因する問題を早期に発見できるため、システムの品質と開発効率を向上させることができる。本研究と照らし合わせても、これがいえるだろう。

UML図のみで実装していたならば、おそらく本研究で実装したプログラムのようにはならなかった。CafeOBJ記述で理解をより深め、仕様をより具体的にすることで合理的にプログラムを実装できるのではないかと考える。

モジュールの汎用性

CafeOBJではリストのようにモジュールが引数に変数の型をもつことで、汎用性の高い関数やモジュールを作成することができる。この性質を利用したものが、PAIRなどの複数のデータの型をまとめるモジュールや、リスト構造を表現するリストのモジュールである。

これらの汎用性が高いモジュールは、引数にデータの型をとり、モジュール内ではこのデータの型を利用してデータ構造を表現を行う。また関数はこのデータの型を利用して演算を行う。モジュールを使用する場合は、宣言時にデータの型を指定するだけでそのデー

タ構造や関数に引数のデータの型が自動的に引き継がれるため、他に記述を行うことなく、関数やデータ構造を利用することができる。そのため、これらのモジュールは非常に高い汎用性がある。

しかし、C 言語ではモジュールの概念がなく、関数の引数の型は基本的に固定しなければならいため CafeOBJ のように汎用性のある関数は作成できない。

例外としては、stdarg.h というライブラリがある。これは C 言語の標準規格で定められた型やマクロ、関数の集まりである標準 C ライブラリの一つであり、printf 関数のような可変個の実引数の操作に関する型とマクロの定義がある。なお、他の標準 C ライブラリには、stdio.h や string.h などがある。しかし、このライブラリを使用しても第一引数だけは引数の数 (符号付き整数型：int) と、固定しなければならないため、CafeOBJ とまったく同一の関数を作成することはできない。

そのため、実装時には汎用性の低いモジュールや関数であることが望ましい。つまり、CafeOBJ での仕様記述においては、汎用性を極力低くする必要があるのではないかと考える。汎用性が低くなれば、それだけ仕様が作成するプログラムの構造を制限することができる。

もちろん、絶対に汎用性が低いモジュールや関数にしなければならないというわけではない。C++のような汎用性のある関数が作成可能なプログラム言語もある。C++では汎用関数 (テンプレート関数) と呼ばれる、コンパイル時にそれぞれの型に応じた関数が自動生成される機能がある。この汎用関数は内部処理は同じだが、データ型が異なる関数が頻繁にある場合に使用される。つまり、今回の例で挙がっていたリスト構造に使用することができる。そのため、汎用性のあるプログラム言語ではその限りではない。

モジュールの分割

本研究での実装に置いて、モジュールは C 言語では特に使用しなかった。これはモジュールを細分化しすぎた、というのが原因の 1 つにある。

CafeOBJ のモジュールは特定の機能を持った複数の関数の集まりである。しかし、本研究で作成した CafeOBJ の仕様書では、その特定の機能を細分化し過ぎたために、同一名関数を含まない関数の数とほぼ同数のモジュールを作成してしまった。

細分化した原因は、C 言語での作成を念頭に入れてしまっていたからである。CafeOBJ を記述する際に、CafeOBJ のモジュールを C 言語の関数として変換しようと考え作成していた。しかし、C 言語でのプログラム作成において選択、変換を行ったのは CafeOBJ の関数から C 言語の関数への変換である。CafeOBJ のモジュールから C 言語の関数への変換では仕様との対応を取るのが難しいと考えたためである。

そのため、関数とほぼ同数存在するモジュールは仕様が困難となってしまった。仮に細分化されていないモジュールであったならば、特定の機能を持つ複数の関数と考えるとライブラリとしてモジュール分割して考えることができる。モジュール分割を行えば、関数などの管理も非常に容易となる。

しかし、C++やJavaといったプログラム言語ではデータやメソッドの集まりであるクラスが存在し、それをモジュールに当てはめられるのではないかと考える。これらの言語においては別途実装を行って考察を行う必要がある。

なお、第1章で話に挙げた、携帯電話組み込みモバイルFeliCa ICチップの事例でも同様の傾向があった。不具合原因の調査の段階において、「仕様の見落とし」と「仕様理解不足」が不具合原因の16.3%あった。この原因のひとつに、「動作する仕様のプログラムに気を取られ、読ませる仕様の記述ができなかったため」と考察している。

だが、前の項で挙げた「モジュールの汎用性」においても軽く触れてはいるが、使用するプログラム言語によっても仕様が変化してしまう可能性がある。つまり、動作する仕様と読ませる仕様を両立していかなければならない。そのため、実装を行うプログラム言語において最適な記述方法を検討していく必要がある。

変数名の意味付け

本研究の等式において、NAやNBといった変数(variable)が見られた。これは自然数(Natural number)のAやBという、変数に番号をつけたx_1やx_2のような使い方で使用していた。

しかし、実際のプログラムの実装においては、それぞれの単語に意味を持たせて実装するケースのほうが一般的であり、本研究での実装時も、NAやNBといった名称で作成せずにある程度意味をもつ英単語や略語を使用して作成した。

変数名はプログラムの読みやすさに直結するものである。変数の名前によって関数の中でその変数がどのような意味をもつか、どのような振る舞いをするのかを記述することで、変数の役割を明確にすることが可能となる。対して、本研究のCafeOBJの仕様のような、変数の付け方では、可読性は低いものとなってしまう。座標や数式の変数を表すx_1といった使い方を除き、これらの明確な意味を持たない変数は可能な限り避けるべきである。

同様に変数の名称だけでなく、関数の名前もいえる。関数は実際に計算などを行う個所でもあり、その挙動や振る舞いは可読性に大きく関係する。特に意味のない関数名であった場合、使用する関数の挙動を想像することは難しく、また記述する関数の挙動もわかりづらいものとなる。そのため、名前付けには注意をしなければならない。

以上のように、CafeOBJでの記述時の変数名は一時的なものではなく意味を持たせた変数名を使用し、プログラムにも変数名を制約した場合のほうが仕様書として的確であると考える。

プログラム言語の流儀

プログラムを作成するときは、アルファベットの大文字や小文字で異なる動作や演算を行う場合がある。また、企業や学校、研究室、開発チームといった集団では、大文字と小

文字の使い方に一定の規則をもたせる場合も考えられる。

本研究でも関数などでその差がみられる。CafeOBJ 記述の仕様書では関数の名前は、関数の動作の意味を表す英語を用いてすべて小文字で記述したが、実装した C 言語では関数の動作の意味を表す英単語の頭文字を大文字で、それ以外を小文字で実装した。

仕様に基づき実装を行うならば、大文字や小文字の規則も基づくほうがより厳密になる。統一ができるのであれば、先述のとおり大文字小文字に関しての差異の有無があるため、プログラムの規則や流儀を変えるのではなく、CafeOBJ の仕様記述時にその規則に従う記述をすればよい。

記述外のソースコード

本研究では CafeOBJ の仕様を元に C 言語の実装を行っている。しかし、CafeOBJ の記述において必要のない関数を C 言語で別途作成する必要がいくつか存在する。

例として、演算結果の表示がある。CafeOBJ では自動的に演算結果がコンソールに表示されるが、C 言語においては演算のみを行い、結果はコンソールに表示しない。他にもファイル操作といったものも考えられる。

これはプログラム言語の立ち位置がことなるためである。CafeOBJ は形式手法言語であるため、仕様の記述や検証が目的である。そのため、CafeOBJ は実行結果が重要となってくる。

対して、C 言語はコンパイルをすることでコンピュータが直接実行可能な機械語のプログラムに変換し、実行するコンパイラ言語であり、その目的はコンピュータがプログラムを実行することである。そのため、C 言語は実行すること自体が重要となってくる。

これらの CafeOBJ と C 言語の立ち位置の違いが、プログラムにも違いが如実に表れてしまっている。これらの違いを埋めるための記述は難しいため、CafeOBJ の仕様書より前の、要求に沿ったものを別途用意する必要がある。そのため、この記述の違いが差を生む可能性も考えられるが、システムの根幹をなす仕様は記述できているため特に問題はないと考える。

総評

CafeOBJ は問題の理解やプログラムの構造などの理解に役立つことが確認できた。そのため、実際にプログラムを作成する前の段階での、早期の開発段階での議論が可能になると考える。実際にプログラムを作成してからの議論よりも早くかつ具体的な内容で議論できるはずである。

しかしながら、本研究では指針がないまま仕様記述を行ったために問題が生じていることも否めない結果となった。今後の課題として、CafeOBJ の仕様書の記述方法も検討する必要がある。なお、実装に使用するプログラム言語によって制約条件も異なってくるため、実装するプログラム言語毎に検討する必要がある。

これらの問題点を検討、修正を行い、複数回 CafeOBJ で記述と C 言語でよりよい仕様書とプログラムができるのはないかと考える。

第7章　まとめ

本研究では、CafeOBJ の仕様書が開発に与える効果や影響を評価・考察を行うため、在庫管理問題の仕様と UML を参考に CafeOBJ で仕様を形式的に記述、それを仕様書として C 言語で実装を行った。

本方法では作成した CafeOBJ のソースコードからの実装前に規則を作成し、それに従って C 言語で実装した。その結果、CafeOBJ 記述法に関して以下の 6 つの項目について利点や問題点、方針を立てる必要性を感じた。これらは検討を行う必要は見られるものの、プログラムのデータ構造や流れなど仕様の理解に関して有効的であることがわかった。

1. 仕様記述を行うことで問題の理解が深る
2. 汎用性の高い関数やモジュールは実装において仕様との差異が生じる可能性がある
3. モジュールを細分化するべきではない
4. 変数名には意味づけを行った名称にすべきではない
5. プログラムの記述法の規則に従うこと
6. 記述外の関数を作成する必要がある

今後の課題としては、対象とするプログラム言語の効果的な記述法とそのプログラムの自動生成法を検討する必要があると考える。

参考文献

[1] 荒木啓二郎, フォーマルメソッドの過去・現在・未来—適用の実践に向けて—, 情報処理, Vol. 49, No. 5, pp493–496, 2008.

[2] 栗田太郎, 携帯電話組込み用モバイル FeliCa IC チップ開発における形式仕様記述手法の適用, 情報処理, Vol. 49, No. 5, pp506–510, 2008.

[3] Joseph Goguen, Jose Meseguer, RAPID PROTOTYPING IN THE OBJ EXECUTABLE SPECIFICATION LANGUAGE, ACM SIGSOFT SOFTWARE ENGINEERING NOTES, Vol. 7, No. 5, pp75–84, 1982.

[4] 山崎利治, 共通問題によるプログラム設計技法解説, 情報処理, Vol. 25, No. 9, p934, 1984

[5] 二村良彦，雨宮真人，山崎利治，淵一博，新しいプログラム・パラダイムによる共通問題の設計, 情報処理, Vol. 26, No. 5, pp458–459, 1985

[6] 雨宮真人，丸山充，関数型言語 Valid による在庫管理システムの記述, 情報処理, Vol. 26, No. 5, pp506–514, 1985

[7] 久世和資，ストリームを扱う言語 Stella による在庫管理システムの記述, 情報処理, Vol. 28, No. 5, pp497–505, 1985

[8] Joseph Schmuller，(訳) 多摩ソフトウェア有限会社独習 UML, 株式会社 翔泳社, 9/30, 2000

付録 A

CafeOBJ 仕様での仕様書

C:\Program Files\cafeOBJ\MainTheme\StockControl-11\SetAndList.mod

-- 集合の定義

```
mod! PAIR(X :: TRIV, Y :: TRIV) {
  [Pair]
  op [_,_] : Elt.X Elt.Y -> Pair
  op fst : Pair -> Elt.X
  op snd : Pair -> Elt.Y
  eq fst([ A:Elt.X,B:Elt.Y ]) = A .
  eq snd([ A:Elt.X,B:Elt.Y ]) = B .
}

mod! TRIPLE(X :: TRIV,Y :: TRIV,Z :: TRIV) {
  [Triple]
  op <_,_,_> : Elt.X Elt.Y Elt.Z -> Triple
  op fst : Triple -> Elt.X
  op snd : Triple -> Elt.Y
  op trd : Triple -> Elt.Z
  eq fst(< A:Elt.X,B:Elt.Y,C:Elt.Z >) = A .
  eq snd(< A:Elt.X,B:Elt.Y,C:Elt.Z >) = B .
  eq trd(< A:Elt.X,B:Elt.Y,C:Elt.Z >) = C .
}

mod! QUADRUPLE(V :: TRIV,X :: TRIV,Y :: TRIV,Z :: TRIV) {
  [Quadruple]
  op <_,_,_,_> : Elt.V Elt.X Elt.Y Elt.Z -> Quadruple
  op fst : Quadruple -> Elt.V
  op snd : Quadruple -> Elt.X
  op trd : Quadruple -> Elt.Y
  op fth : Quadruple -> Elt.Z
  eq fst(< A:Elt.V,B:Elt.X,C:Elt.Y,D:Elt.Z >) = A .
  eq snd(< A:Elt.V,B:Elt.X,C:Elt.Y,D:Elt.Z >) = B .
  eq trd(< A:Elt.V,B:Elt.X,C:Elt.Y,D:Elt.Z >) = C .
  eq fth(< A:Elt.V,B:Elt.X,C:Elt.Y,D:Elt.Z >) = D .
}

mod! OPTION(M :: TRIV) {
  [None Some < Option]
  op none : -> None
  op some : Elt.M -> Some
  op valOf : Some -> Elt.M
  var E : Elt.M
  eq valOf(some(E)) = E .
}

mod! FORLIST(X :: TRIV) {
  pr(OPTION(X) * {sort Option -> OptionLE,
                  sort None -> NoneLE,
                  sort Some -> SomeLE,
                  op none -> noneLE})
  pr(NAT)
  [List]

  op empty : -> List
  op _::_ : Elt.X List -> List
  --
  op hd : List -> OptionLE
  op tl : List -> List
  op nth : List Nat -> OptionLE
  op _@_ : List List -> List
  op len : List -> Nat
  op rev : List -> List
  --
  var E : Elt.X
  vars L L1 L2 : List
  var N : Nat
  --
  eq hd(empty) = noneLE .
  eq hd(E :: L) = some(E) .
  eq tl(empty) = empty .
  eq tl(E :: L) = L .
  eq nth(empty,N) = noneLE .
  eq nth(E :: L, 0) = some(E) .
  ceq nth(E :: L, N) = nth(L,p N) if N > 0 .
  eq empty @ L2 = L2 .
  eq (E :: L1) @ L2 = E :: (L1 @ L2) .
  eq len(empty) = 0 .
  eq len(E :: L) = 1 + len(L) .
  eq rev(empty) = empty .
  eq rev(E :: L) = rev(L) @ (E :: empty) .
}
```

C:\Program Files\cafeOBJ\MainTheme\StockControl-11\DefineData.mod

```
-- ----------------------------------------
-- 定義
-- ----------------------------------------

-- 内蔵品
-- (STRING::品名, NAT::数量の集合)
mod! CONTENT principal-sort Content {
  pr(PAIR(STRING, NAT) * {sort Pair -> Content})
}


-- 選択コンテナ(出庫用データ)
-- (NAT::コンテナ番号, CONTENT::内蔵品, BOOL::搬出マーク)
mod! SELECTCONTAINERDATA principal-sort SelectContainerData {
  pr(TRIPLE(NAT, CONTENT, BOOL)
    * {sort Triple -> SelectContainerData})
}

-- 選択コンテナ(出庫用データ)の集合
mod! SELECTCONTAINERLIST principal-sort SelectContainerList {
  pr(FORLIST(SELECTCONTAINERDATA) * {sort List -> SelectContainerList}) .

  op addcontainer : SelectContainerData SelectContainerList -> SelectContainerList .
  op addcontainer : SelectContainerList SelectContainerData -> SelectContainerList .
  op addcontainer : SelectContainerList SelectContainerList -> SelectContainerList .

  op addcontainer : SelectContainerData SelectContainerList SelectContainerList -> SelectContainerList .
  op addcontainer : SelectContainerList SelectContainerData SelectContainerList -> SelectContainerList .

  var SC  : SelectContainerData .
  var SCL : SelectContainerList .
  vars SCLA SCLB : SelectContainerList .

  eq addcontainer(SC, SCL) = SC :: SCL .
  eq addcontainer(SCL, SC) = SCL @ (SC :: empty) .
  eq addcontainer(SCLA, SCLB) = SCLA @ SCLB .

  eq addcontainer(SC, SCLA, SCLB) = addcontainer(SC, SCLA @ SCLB) .
  eq addcontainer(SCLA, SC, SCLB) = SCLA @ addcontainer(SC, SCLB) .
}


-- 内蔵品のリスト
-- (CONTENT::内蔵品)
mod! CONTENTLIST principal-sort ContentList {
  pr(FORLIST(CONTENT) * {sort List -> ContentList}) .

  op addcontent : Content ContentList -> ContentList .
  op addcontent : ContentList Content -> ContentList .
  op addcontent : ContentList ContentList -> ContentList .
  op addcontent : Content ContentList ContentList -> ContentList .
  op addcontent : ContentList Content ContentList -> ContentList .

  var  CO  : Content .
  var  COL : ContentList .
  vars COLA COLB : ContentList .

  eq addcontent(CO, COL) = CO :: COL .
  eq addcontent(COL, CO) = COL @ (CO :: empty) .
  eq addcontent(COLA, COLB) = COLA @ COLB .
  eq addcontent(CO, COLA, COLB) = addcontent(CO, COLA @ COLB) .
  eq addcontent(COLA, CO, COLB) = COLA @ addcontent(CO, COLB) .
}


-- 積荷票
-- (NAT::コンテナ番号, NAT::搬入年月,
--  NAT::搬入日時, CONTENTLIST::内蔵品の集合)
mod! STOCKSHEET principal-sort StockSheet {
   pr(QUADRUPLE(NAT, NAT, NAT, CONTENTLIST)
     * {sort Quadruple -> StockSheet})
}


-- 積荷票の集合
-- (STOCKSHEET::積荷票)
mod! STOCKSHEETLIST principal-sort StockSheetList {
  pr(FORLIST(STOCKSHEET) * {sort List -> StockSheetList})

  op dummystock : -> StockSheet .

  op addstock : StockSheet StockSheetList -> StockSheetList .
  op addstock : StockSheetList StockSheet -> StockSheetList .
  op addstock : StockSheetList StockSheet StockSheetList
                                        -> StockSheetList .

  op remstock : StockSheet StockSheetList -> StockSheetList .
  op remstock : Nat StockSheetList -> StockSheetList .
  op remstock : Nat StockSheetList StockSheetList -> StockSheetList .

  var SS  : StockSheet .
  var SSL : StockSheetList .
  var COL : ContentList .
  var CN  : Nat .

  vars SSLA SSLB : StockSheetList .
  vars NA NB NC  : Nat .

  eq dummystock = < 10000, 0, 0, empty > .
```

C:\Program Files\cafeOBJ\MainTheme\StockControl-11\DefineData.mod

```
  eq addstock(SS, SSL) = SS :: SSL .
  eq addstock(SSL, SS) = SSL @ (SS :: empty) .
  eq addstock(SSLA, SS, SSLB) = SSLA @ addstock(SS, SSLB) .

  eq addstock(SSL, < NA, NB, NC, empty >) = SSL .


  eq remstock(SS, empty) = empty .
  eq remstock(< NA, NB, NC, COL >, SSL) = remstock(NA, SSL) .

  eq remstock(CN, empty) = empty .
  eq remstock(CN, SSL) = remstock(CN, empty, SSL) .

  eq remstock(CN, SSL, empty) = SSL .
  eq remstock(CN, SSLA, < NA, NB, NC, COL > :: SSLB) =
    if(CN == NA) then SSLA @ SSLB
    else remstock(CN, addstock(SSLA, < NA, NB, NC, COL >), SSLB) fi .
}


-- 依頼票
-- (STRING::送り先名, STRING::品名, NAT::数量)
mod! REQUESTSHEET principal-sort RequestSheet {
  pr(TRIPLE(STRING, STRING, NAT) * {sort Triple -> RequestSheet})

  op dammysheet : -> RequestSheet .
  eq dammysheet = < "ERRER", "ERRER", 0 > .
}

-- 在庫不足リスト
mod! REQUESTSHEETLIST principal-sort RequestSheetList {
  pr(FORLIST(REQUESTSHEET) * {sort List -> RequestSheetList}) .

  op addrequest : RequestSheet RequestSheetList -> RequestSheetList .
  op addrequest : RequestSheetList RequestSheet -> RequestSheetList .
  op addrequest : RequestSheetList RequestSheetList -> RequestSheetList .
  op addrequest : RequestSheetList RequestSheet RequestSheetList -> RequestSheetList .

  var  RS : RequestSheet .
  vars RSL RSLA RSLB : RequestSheetList .

  eq addrequest(RS, RSL) = RS :: RSL .
  eq addrequest(RSL, RS) = RSL @ (RS :: empty) .
  eq addrequest(RSLA, RSLB) = RSLA @ RSLB .
  eq addrequest(RSLA, RS, RSLB) = RSLA @ addrequest(RS, RSLB) .

}

-- 出庫指示票
mod! INSTRUCTIONSHEET principal-sort InstructionSheet {
   pr(TRIPLE(NAT, STRING, SELECTCONTAINERLIST)
     * {sort Triple -> InstructionSheet})
}


-- 取扱い商品のリスト
mod! PRODUCTSAKELIST {
  pr(FORLIST(STRING) * {sort List -> SakeNameList}) .
}


-- 出庫指示票のリスト
mod! INSTRUCTIONSHEETLIST principal-sort InstructionSheetList {
  pr(FORLIST(INSTRUCTIONSHEET) * {sort List -> InstructionSheetList}) .

  op getlastinstnumber : InstructionSheetList -> Nat .
  op getinstnumber : InstructionSheet -> Nat .

  var NA : Nat .
  var ST : String .
  var SCL : SelectContainerList .
  var IS  : InstructionSheet .
  var ISL : InstructionSheetList .

  eq getinstnumber(< NA, ST, SCL >) = NA .

  eq getlastinstnumber(empty) = 0 .

  eq getlastinstnumber(IS :: ISL) =
    if (ISL == empty) then
      if getinstnumber(IS) == 99999 then 9999 else getinstnumber(IS) fi
    else getlastinstnumber(ISL) fi .


  var IS : InstructionSheet .
  vars ISL ISLA ISLB : InstructionSheetList .

  op addinst : InstructionSheet InstructionSheetList -> InstructionSheetList .
  op addinst : InstructionSheetList InstructionSheet -> InstructionSheetList .

  eq addinst(IS, ISL) = IS :: ISL .
  eq addinst(ISL, IS) = ISL @ (IS :: empty) .
}


-- 実行時の成功・エラー表示用
mod! EXECUTION {
   pr(TRIPLE(STOCKSHEETLIST, REQUESTSHEETLIST, INSTRUCTIONSHEETLIST)
     * {sort Triple -> ExecutionData})
```

```
C:\Program Files\cafeOBJ\MainTheme\StockControl-11\DefineData.mod
  [Success ErrerA ErrerB < ExecutionData]

  op ErrA : -> ErrerA .
  op ErrB : -> ErrerB .

  op getssl : ExecutionData -> StockSheetList .
  op getrsl : ExecutionData -> RequestSheetList .
  op getisl : ExecutionData -> InstructionSheetList .

  var SSL : StockSheetList .
  var RSL : RequestSheetList .
  var ISL : InstructionSheetList .

  eq getssl(< SSL, RSL, ISL >) = SSL .
  eq getrsl(< SSL, RSL, ISL >) = RSL .
  eq getisl(< SSL, RSL, ISL >) = ISL .
}


mod! ORDER (X :: TRIV, Y :: TRIV) {
  [SSheet RSheet < Order]

  op s _ : Elt.X -> SSheet .
  op r _ : Elt.Y -> RSheet .
}

mod! ORDERDATA principal-sort OrderData {
  pr(ORDER(STOCKSHEET, REQUESTSHEET) * {sort Order -> OrderData}) .
}

mod! ORDERDATALIST {
  [OrderDataList]
  pr(ORDERDATA) .

  op over : -> OrderDataList .
  op _ | _ : OrderData OrderDataList -> OrderDataList
}
```

C:\Program Files\cafeOBJ\MainTheme\StockControl-11\SendExe.mod

```
-- ----------------------------------------------------
-- 依頼があった場合
-- ----------------------------------------------------

-- 取扱商品か確認を行い、Boolを返す
mod! CHECKPRODUCTSAKE {
  pr(PRODUCTNAMELIST) .
  pr(REQUESTSHEET) .

  var  CN : Nat
  vars ST SN : String .

  var  SNL : SakeNameList .
  vars SNA SNB : String . -- 酒の名前

  op checkproduct : RequestSheet -> Bool .
  op checkproduct : String -> Bool .
  op checkproduct : String SakeNameList -> Bool .

  eq checkproduct(< ST, SN, CN >) = checkproduct(SN) .
  eq checkproduct(SNA) = checkproduct(SNA, pronamelist) .
  eq checkproduct(SNA, empty) = false .
  eq checkproduct(SNA, SNA :: SNL) = true .
  eq checkproduct(SNA, SNB :: SNL) =
    if(string=(SNA, SNB)) then true else checkproduct(SNA, SNL) fi .
}


-- 在庫の検索
-- 在庫がある積荷票を返す
mod! SREACHSTOCKSAKE {
  pr(STOCKSHEETLIST) .
  pr(STOCKSHEETLISTDATA) .
  pr(REQUESTSHEET) .

  var  COL : ContentList .
  vars ST SN SNA SNB : String .
  vars NA NB NC ND CN : Nat .
  vars SSLA SSLB : StockSheetList .

  op searchstock : String StockSheetList -> StockSheetList .
  op searchstock : RequestSheet StockSheetList -> StockSheetList .
  op searchstock : String StockSheetList StockSheetList
                                          -> StockSheetList .

  op searchcon : String ContentList -> Bool .


  eq searchstock(SNA, SSLA) = searchstock(SNA, SSLA, empty) .
  eq searchstock(< ST, SN, CN >, SSLA) = searchstock(SN, SSLA, empty) .

  eq searchstock(SNA, empty, SSLB) = SSLB .
  eq searchstock(SNA, < NA, NB, NC, COL > :: SSLA, SSLB) =
    if (searchcon(SNA, COL))
      then searchstock(SNA, SSLA, addstock(SSLB, < NA, NB, NC, COL >))
      else searchstock(SNA, SSLA, SSLB) fi .

  eq searchcon(SNA, empty) = false .
  eq searchcon(SNA, [ SNA, NA ] :: COL) = true .
  eq searchcon(SNA, [ SNB, NA ] :: COL) =
    if(string=(SNA, SNB)) then true else searchcon(SNA, COL) fi .
}


-- 引数：商品名・内蔵品を受け取り，そのリスト中の数を返す
mod! GETSTOCKCNT {
  pr(CONTENTLIST) .

  var  NA : Nat .
  var  COL : ContentList .
  vars SNA SNB : String .

  op getstockcnt : String ContentList -> Nat .

  eq getstockcnt(SNA, empty) = 0 .
  eq getstockcnt(SNA, [ SNA, NA ] :: COL) = NA .
  eq getstockcnt(SNA, [ SNB, NA ] :: COL) =
    if(string=(SNA, SNB)) then NA else getstockcnt(SNA, COL) fi .
}


-- 在庫数の計算
mod! SUMOFSTOCK {
  pr(STOCKSHEETLIST) .
  pr(GETSTOCKCNT) .

  var  COL : ContentList .
  var  SSL : StockSheetList .
  vars ST SN SNA SNB  : String .
  vars NA NB NC ND CN : Nat .

  op sumofstock  : String StockSheetList -> Nat .
  op sumofstock  : String StockSheetList Nat -> Nat .

  eq sumofstock(SNA, SSL) = sumofstock(SNA, SSL, 0) .
  eq sumofstock(SNA, empty, ND) = ND .
  eq sumofstock(SNA, < NA, NB, NC, COL > :: SSL, ND) =
    sumofstock(SNA, SSL, getstockcnt(SNA, COL) + ND) .
}
```

C:\Program Files\cafeOBJ\MainTheme\StockControl-11\SendExe.mod

```
mod! COMPARESTOCK {
  pr(STOCKSHEETLIST) .
  pr(REQUESTSHEET) .
  pr(SUMOFSTOCK) .

  var  CN : Nat .
  var  SSL : StockSheetList .
  vars ST SN : String .

  op comparestock : RequestSheet StockSheetList -> Bool .

  eq comparestock(< ST, SN, CN >, SSL) =
    if (CN <= sumofstock(SN, SSL)) then true else false fi .
}


-- コンテナの選択
-- コンテナは後入れ先出形式
-- 何本出すのかを記述した積荷票を返す
mod! SELECTCONTAINER {
  pr(STOCKSHEETLIST) .
  pr(REQUESTSHEET) .
  pr(GETSTOCKCNT) .

  vars ST SN SNA SNB : String .
  vars COL COLA COLB : ContentList .
  vars SSLA SSLB : StockSheetList .
  vars NA NB NC ND NE : Nat .

  -- 酒の名前，本数，酒の入っているコンテナ
  op selectcontainer : String Nat StockSheetList -> StockSheetList .
  op selectcontainer : String Nat StockSheetList StockSheetList -> StockSheetList .
  op selectcon : String Nat ContentList ContentList -> ContentList .

  op selectcontainer : RequestSheet StockSheetList -> StockSheetList .


  -- 在庫不足はないことは既知
  -- ※ eq selectcontainer(SN, NA, empty, SSLA) : はありえない.
  eq selectcontainer(SN, NA, SSLA) = selectcontainer(SN, NA, SSLA, empty) .
  eq selectcontainer(< ST, SN, NA >, SSLA) = selectcontainer(SN, NA, SSLA, empty) .

  eq selectcontainer(SN, NA, < NB, NC, ND, COL > :: SSLA, SSLB) =
    if (NA <= getstockcnt(SN, COL))
      then addstock(SSLB, < NB, NC, ND, selectcon(SN, NA, COL, empty) >)
    else
      selectcontainer(SN, sd(NA, getstockcnt(SN, COL)), SSLA,
                                  addstock(SSLB, < NB, NC, ND, COL >))
    fi .

  eq selectcon(SNA, NA, [ SNB, NB ] :: COLA, COLB) =
    if(string=(SNA, SNB)) then addcontent([ SNB, NA ], COLA, COLB)
    else selectcon(SNA, NA, COLA, addcontent(COLB, [ SNB, NB ]))
    fi .
}


-- 積荷票のリスト内にある積荷票の要素，内蔵品のリストの先頭の要素を
-- 引数にとった品名の商品に並べ替える.
mod! REPOSITION {
  pr(STOCKSHEETLIST) .

  op reposition : String StockSheetList -> StockSheetList
  op reposition : String StockSheetList StockSheetList -> StockSheetList

  op reposition : String ContentList -> ContentList
  op reposition : String ContentList ContentList -> ContentList

  vars NA NB NC ND : Nat .
  vars SN SNA SNB  : String .
  vars COL COLA COLB : ContentList .
  vars SSL SSLA SSLB : StockSheetList .

  eq reposition(SN, SSL) = reposition(SN, SSL, empty) .
  eq reposition(SN, empty, SSLB) = SSLB .
  eq reposition(SN, < NA, NB, NC, COL > :: SSLA, SSLB) =
    reposition(SN, SSLA, addstock(SSLB, < NA, NB, NC, reposition(SN, COL) >)) .

  eq reposition(SNA, COL) = reposition(SNA, COL, empty) .
  eq reposition(SNA, empty, COLB) = COLB .
  eq reposition(SNA, [ SNA, NA ] :: COLA, COLB) = addcontent([ SNB, NA ], COLA, COLB) .
  eq reposition(SNA, [ SNB, NA ] :: COLA, COLB) =
    if (string=(SNA, SNB)) then addcontent([ SNB, NA ], COLA, COLB)
    else reposition(SNA, COLA, addcontent(COLB, [ SNB, NA ])) fi .
}


mod! CHECKSTOCKLIST {
  pr(STOCKSHEETLIST) .
  pr(STOCKSHEETLISTDATA) .
  pr(SELECTCONTAINERLIST) .
  pr(REPOSITION) .

  var  SSL : StockSheetList .
  var  SCL : SelectContainerList .
  var  CO  : Content .
  vars SSLA SSLB : StockSheetList .
  vars COL COLA COLB : ContentList .
  vars SN SNA SNB : String .
```

C:\Program Files\cafeOBJ\MainTheme\StockControl-11\SendExe.mod

```
  vars NA NB NC ND NE NF : Nat .

  op checkstock : Nat Content StockSheetList -> Bool .
  op checkstockcnt : Nat Content StockSheetList -> StockSheetList .
  op checkstockcnt : Nat Content StockSheetList StockSheetList -> StockSheetList .

  -- コンテナ番号を比較 -> 商品を検索 -> 取ったら空になるかBoolで返す
  eq checkstock(NA, [ SNA, NB ], < NC, ND, NE, [ SNB, NF ] :: COL > :: SSL) =
    if (NA == NC) then
      if string=(SNA, SNB) then
        if (NB < NF) then false
          else if(COL == empty) then true else false fi
        fi
      else
        checkstock(NA, [ SNA, NB ], addstock(< NC, ND, NE, COL >, SSL))
      fi
    else
      checkstock(NA, [ SNA, NB ], SSL)
    fi .

  -- コンテナ番号を比較 -> 商品を検索 -> 在庫票のリストを返す
  eq checkstockcnt(NA, CO, SSL) = checkstockcnt(NA, CO, empty, SSL) .
  eq checkstockcnt(NA, CO, SSLA, empty) = SSLA .
  eq checkstockcnt(NA, [ SNA, NB ], SSLA, < NC, ND, NE, [ SNB, NF ] :: COL > :: SSLB) =
    if (NA == NC) then
      if string=(SNA, SNB) then
        if (NB < NF) then
          addstock(SSLA, < NC, ND, NE, [ SNB, sd(NB, NF) ] :: COL >, SSLB)
        else
          if (COL == empty) then SSLA @ SSLB
          else addstock(SSLA, < NC, ND, NE, COL >, SSLB)
          fi
        fi
      else
        checkstockcnt(NA, [ SNA, NB ], SSLA,
                      addstock(< NC, ND, NE, addcontent(COL, [ SNB, NF ]) >, SSLB))
      fi
    else
      checkstockcnt(NA, [ SNA, NB ],
                      addstock(SSLA, < NC, ND, NE, addcontent([ SNB, NF ], COL) >) , SSLB)
    fi .
}


-- 選択コンテナを出庫指示票用のデータに書き換える
mod! MAKESELECTDATA {
  pr(STOCKSHEETLIST) .
  pr(STOCKSHEETLISTDATA) .
  pr(SELECTCONTAINERLIST) .
  pr(CHECKSTOCKLIST)

  var  SSL : StockSheetList .
  var  SCL : SelectContainerList .
  var  CO  : Content .
  vars SSLA SSLB : StockSheetList .
  vars COL COLA COLB : ContentList .
  vars SN SNA SNB : String .
  vars NA NB NC ND NE NF : Nat .


  op makedata : String StockSheetList StockSheetList -> SelectContainerList .
  op makedata : String StockSheetList SelectContainerList
                                      StockSheetList -> SelectContainerList .

  -- 選択コンテナ用データの作成
  -- 空コンテナか印字も行う
  eq makedata(SN, SSLA, SSLB) = makedata(SN, reposition(SN, SSLA), empty, SSLB) .
  eq makedata(SN, empty, SCL, SSLB) = SCL .

  eq makedata(SN, < NA, NB, NC, CO :: COL > :: SSLA, SCL, SSLB) =
    if(COL == empty) then
      if checkstock(NA, CO, SSLB) then
        makedata(SN, SSLA, addcontainer(SCL, < NA, CO, true >), SSLB)
      else
        makedata(SN, SSLA, addcontainer(SCL, < NA, CO, false >), SSLB)
      fi
    else
      makedata(SN, SSLA, addcontainer(SCL, < NA, CO, false >), SSLB)
    fi .
}


-- 積荷票リストの更新情報の作成
mod! REMAKESTOCKLIST {
  pr(STOCKSHEETLIST) .
  pr(STOCKSHEETLISTDATA) .
  pr(CHECKSTOCKLIST) .

  var SN : String .
  var CO : Content .
  var COL : ContentList .

  vars NA NB NC : Nat
  vars SSL SSLA SSLB : StockSheetList .

  op remakestockset : String StockSheetList StockSheetList -> StockSheetList .
  op remakestockset : StockSheetList StockSheetList -> StockSheetList .

  -- 出庫用リストと在庫リストの内蔵品の先頭に出庫する酒のデータに変更
  eq remakestockset(SN, SSLA, SSLB)
```

```
C:\Program Files\cafeOBJ\MainTheme\StockControl-11\SendExe.mod
    = remakestockset(reposition(SN, SSLA), reposition(SN, SSLB)) .

  eq remakestockset(empty, SSLB) = SSLB .
  eq remakestockset(SSLA, empty) = empty . -- エラー用.ありえない

  eq remakestockset(< NA, NB, NC, CO :: COL > :: SSLA, SSLB) =
    remakestockset(SSLA, checkstockcnt(NA, CO, SSLB)) .
}


-- 出庫指示票の作成
mod! MAKEINSTSHEET {
  pr(REQUESTSHEET) .
  pr(INSTRUCTIONSHEET) .
  pr(MAKESELECTDATA) .
  pr(REMAKESTOCKLIST) .

  var RN : Nat .    -- RequestNumber(注文番号)
  var ST : String . -- SendTo(送り先)
  var SN : String . -- SakeName(商品名)
  var CN : Nat . -- Count(依頼数)

  vars SSLA SSLB : StockSheetList .

  op makeinstsheet : Nat RequestSheet StockSheetList StockSheetList
                                                -> InstructionSheet .

  eq makeinstsheet(RN, < ST, SN, CN >, SSLA, SSLB) =
    < RN, ST, makedata(SN, SSLA, SSLB) > .
}

-- 在庫情報の更新、および出庫指示票の作成
mod! EXESENDSTEP {
  pr(MAKEINSTSHEET) .
  pr(SREACHSTOCKSAKE) .
  pr(SELECTCONTAINER) .
  pr(EXECUTION) .

  vars RN RC : Nat .
  vars ST SN : String .
  vars SSLA SSLB : StockSheetList .
  var ISL : InstructionSheetList .
  var RSL : RequestSheetList .

  op exesendstep : Nat StockSheetList RequestSheet ExecutionData -> ExecutionData .

  -- selectcontainer実行済み(SSLAには出庫用の積荷票リスト)
  eq exesendstep(RN, SSLA, < ST, SN, RC >, < SSLB, RSL, ISL >)
    = < remakestockset(SN, SSLA, SSLB), RSL,
        addinst(ISL, makeinstsheet(RN, < ST, SN, RC >, SSLA, SSLB)) > .
}


-- 在庫不足を連絡した(動作した、という意味なので必ず成功)
mod! CONTACTCUSTOMER {
  pr(STRING) .
  op contact : String -> Bool .
  eq contact(ST:String) = true .
}


mod! EXERESERVESTEP {
  pr(REQUESTSHEETLIST) .
  pr(CONTACTCUSTOMER) .

  var ST : String .
  var SN : String .
  var CN : Nat .
  var RSL : RequestSheetList .

  op exereservestep : RequestSheet RequestSheetList -> RequestSheetList .

  eq exereservestep(< ST, SN, CN >, RSL) =
    if (contact(ST)) then addrequest(RSL, < ST, SN, CN >) else empty fi .
}
```

C:\Program Files\cafeOBJ\MainTheme\StockControl-11\RequestExe.mod

```
-- ------------------------------------------------
-- 積荷票受理部
-- ------------------------------------------------

-- 在庫不足リストに商品があるか確認
mod! CHECKREQUEST {

  pr(STOCKSHEETLIST) .
  pr(REQUESTSHEETLIST) .

  var CO  : Content .
  var COL : ContentList .
  vars CN NA NB NC : Nat
  vars SN ST SNA SNB : String .

  op checkrequest : String StockSheet -> Bool .
  op checkrequest : StockSheet RequestSheet -> Bool .
  op checkrequest : String ContentList -> Bool .

  eq checkrequest(SN, < NA, NB, NC, COL > ) = checkrequest(SN, COL) .
  eq checkrequest(< NA, NB, NC, COL >, < ST, SN, CN >) = checkrequest(SN, COL) .

  eq checkrequest(SNA, empty) = false .
  eq checkrequest(SNA, [ SNB, CN ] :: COL) =
    if string=(SNA, SNB) then true else checkrequest(SNA, COL) fi .
}

mod! SEARCHNUMBER {
  pr(STOCKSHEETLIST) .

  op searchnumber : StockSheet StockSheetList -> StockSheet .

  vars  COLA COLB : ContentList .
  var  SS : StockSheet .
  var  SSL : StockSheetList .
  vars NA NB NC ND NE NF NM : Nat .

  eq searchnumber(< NA, NB, NC, COLA >, empty) = < NA, NB, NC, empty > .
  eq searchnumber(< NA, NB, NC, COLA >, < ND, NE, NF, COLB > :: SSL) =
    if (NA == ND) then < ND, NE, NF, COLB > else searchnumber(< NA, NB, NC, COLA >, SSL) fi .
}
```

C:\Program Files\cafeOBJ\MainTheme\StockControl-11\Execution.mod

```
-- ------------------------------------------------
-- 実行部
-- ------------------------------------------------

mod! PRSET {
  pr(STOCKSHEETLISTDATA) .
  pr(REQUESTSHEETLISTDATA) .

  pr(EXECUTION) .
  pr(ORDER) .

  pr(CHECKPRODUCTSAKE) .
  pr(SREACHSTOCKSAKE) .
  pr(COMPARESTOCK) .
  pr(SELECTCONTAINER) .
  pr(MAKESELECTDATA) .
  pr(REMAKESTOCKLIST) .
  pr(MAKEINSTSHEET) .
  pr(EXESENDSTEP) .
  pr(EXERESERVESTEP) .

  pr(CHECKREQUEST) .
}


mod! EXESEND {
  pr(PRSET) .

  var ST : String . -- 送り先
  var SN : String . -- 酒の名前
  var CN : Nat .    -- 数量

  var RS : RequestSheet .

  var RSL : RequestSheetList .
  var SSL : StockSheetList .
  var ISL : InstructionSheetList .

  var ED : ExecutionData .

  var SS : StockSheet .

  vars SSLA SSLB : StockSheetList .
  vars RSLA RSLB : RequestSheetList .


  -- ------------------------------------------------
  -- 出庫依頼動作部
  -- ------------------------------------------------

  op runsend : RequestSheet ExecutionData -> ExecutionData .

  op sendfirststep   : RequestSheet ExecutionData -> ExecutionData .
  op sendsecondstep  : RequestSheet StockSheetList ExecutionData -> ExecutionData .
  op reservestep : RequestSheet ExecutionData -> ExecutionData .
  op sendstep    : RequestSheet StockSheetList ExecutionData -> ExecutionData .

  eq runsend(RS, ED) = sendfirststep(RS, ED) .

  -- 在庫情報を調べ、取扱商品ならば依頼票と該当商品のある在庫票のリストを渡す.
  eq sendfirststep(RS, < SSL, RSL, ISL >) = if checkproduct(RS)
    then sendsecondstep(RS, searchstock(RS, SSL), < SSL, RSL, ISL >) else < SSL, RSL, ISL > fi .


  -- 在庫数を調べ，出庫指示か在庫不足かに分岐.
  eq sendsecondstep(RS, SSL, ED)
    = if comparestock(RS, SSL)
    then sendstep(RS, SSL, ED) else reservestep(RS, ED) fi .


  -- 在庫不足に追加.
  eq reservestep(RS, < SSL, RSL, ISL >) = < SSL, exereservestep(RS, RSL), ISL > .


  -- 出庫指示
  eq sendstep(RS, SSLA, < SSLB, RSL, ISL >) =
    if (ISL == empty) then
      exesendstep(10000, selectcontainer(RS, SSLA), RS, < SSLB, RSL, ISL >)
    else
      exesendstep(getlastinstnumber(ISL) + 1,
                  selectcontainer(RS, SSLA), RS, < SSLB, RSL, ISL >)
    fi .
}



-- ------------------------------------------------
-- 積荷票受理部
-- ------------------------------------------------

mod! EXEARRI {
  pr(EXESEND) .
  pr(SEARCHNUMBER) .

  var ST : String . -- 送り先
  var SN : String . -- 酒の名前
  var CN : Nat .    -- 数量

  var RS : RequestSheet .
```

C:\Program Files\cafeOBJ\MainTheme\StockControl-11\Execution.mod

```
  var RSL : RequestSheetList .
  var SSL : StockSheetList .
  var ISL : InstructionSheetList .

  var ED : ExecutionData .

  var SS : StockSheet .

  vars NA NB NC : Nat .
  vars SSLA SSLB : StockSheetList .
  vars RSLA RSLB : RequestSheetList .

  -- 積荷票と"在庫情報，在庫不足，出庫依頼票"のセットデータ
  op runarrive : StockSheet ExecutionData -> ExecutionData .

  op arrivefirststep  : StockSheet ExecutionData RequestSheetList -> ExecutionData .
  op arrivesecondstep : StockSheet ExecutionData RequestSheetList
                                                 StockSheetList   -> ExecutionData .
  op arrivethirdstep  : StockSheet ExecutionData RequestSheetList
                                                 StockSheetList   -> ExecutionData .
  op arrivefourthstep : StockSheet ExecutionData RequestSheetList -> ExecutionData .

  eq runarrive(< NA, NB, NC, empty >, ED) = ED .
  eq runarrive(SS, ED) = arrivefirststep(SS, ED, empty) .

  eq arrivefirststep(< NA, NB, NC, empty >, < SSL, RSLA, ISL >, RSLB) = < SSL, addrequest(RSLB, RSLA), ISL > .
  eq arrivefirststep(SS, < SSL, empty, ISL >, RSLB) = < addstock(SSL, SS), RSLB, ISL > .

  eq arrivefirststep(SS, < SSL, RS :: RSLA, ISL >, RSLB) =
    if (checkrequest(SS, RS)) then
      arrivesecondstep(SS, < SSL, RS :: RSLA, ISL >, RSLB, searchstock(RS, addstock(SSL, SS)))
    else
      arrivefirststep(SS, < SSL, RSLA, ISL >, addrequest(RSLB, RS))
    fi .


  -- trueならば出庫. falseならばfirststepへ.
  eq arrivesecondstep(SS, < SSLA, RS :: RSLA, ISL >, RSLB, SSLB) =
    if comparestock(RS, SSLB) then
      arrivethirdstep(SS, < SSLA, RS :: RSLA, ISL >, RSLB, SSLB)
    else
      arrivefirststep(SS, < SSLA, RSLA, ISL >, addrequest(RSLB, RS))
    fi .


  eq arrivethirdstep(SS, < SSLA, RS :: RSLA, ISL >, RSLB, SSLB) =
    arrivefourthstep(SS, sendstep(RS, SSLB, < addstock(SSLA, SS), RSLA, ISL >), RSLB) .

  eq arrivefourthstep(SS, < SSL, RSLA, ISL >, RSLB) =
    arrivefirststep(searchnumber(SS, SSL), < remstock(SS, SSL), RSLA, ISL >, RSLB) .
}


-- ------------------------------------------------------
--
-- ------------------------------------------------------
mod! COMPLEXEXE {
  pr(EXESEND) .
  pr(EXEARRI) .
  pr(ORDERDATALIST) .
}



-- ---------------------------------------------------------------------
--
-- ---------------------------------------------------------------------

mod! COMPLEXEXECUTION {
  pr(COMPLEXEXE) .

  var ST : String . -- 送り先
  var SN : String . -- 酒の名前
  var CN : Nat .    -- 数量

  var SS : StockSheet .
  var RS : RequestSheet .

  var RSL : RequestSheetList .
  var SSL : StockSheetList .
  var ISL : InstructionSheetList .

  var ED : ExecutionData .

  vars SSLA SSLB : StockSheetList .
  vars RSLA RSLB : RequestSheetList .

  var OD  : OrderData .
  var ODL : OrderDataList .

  -- ------------------------------------------------
  -- 実動作部
  -- ------------------------------------------------

  op inputdata : -> ExecutionData .

  eq inputdata = < stocklist, requestlist, empty > .

  op stockcontrol : OrderDataList -> ExecutionData .
  op stockcontrol : OrderDataList ExecutionData -> ExecutionData .
```

C:\Program Files\cafeOBJ\MainTheme\StockControl-11\Execution.mod

```
  op stockcontrol : OrderData OrderDataList ExecutionData -> ExecutionData .

  op stockcontrol : RequestSheet OrderDataList ExecutionData -> ExecutionData .
  op stockcontrol : StockSheet   OrderDataList ExecutionData -> ExecutionData .


  eq stockcontrol(ODL) = stockcontrol(ODL, inputdata) .
  eq stockcontrol(over, ED) = ED .

  eq stockcontrol(s(SS) | ODL, ED) = stockcontrol(SS, ODL, ED) .
  eq stockcontrol(r(RS) | ODL, ED) = stockcontrol(RS, ODL, ED) .

  eq stockcontrol(RS, ODL, ED) = stockcontrol(ODL, runsend(RS, ED)) .
  eq stockcontrol(SS, ODL, ED) = stockcontrol(ODL, runarrive(SS, ED)) .
}




-- ------------------------------------------------------------------------------
-- ------------------------------------------------------------------------------
-- 以下テスト部
-- ------------------------------------------------------------------------------
-- ------------------------------------------------------------------------------


--  red dec(r(< "住所A", "如水",  5 >)) .

--  red stockcontrol(over) .

--  red runsend((< "住所A", "菊姫",  5 >), < empty, empty, empty >) .

--  red stockcontrol(r(< "住所A", "菊姫",  5 >) | over , < empty, empty, empty >) .
--  red stockcontrol(s(< 19936, 201003, 222132, [ "如水", 5 ] :: empty >) | over) .



  -- ------------------------------------------------
  -- テスト部分
  -- ------------------------------------------------

  -- red checkproduct("正宗") .

  -- red stocklist .
  -- red searchstock("正宗") .

  -- red requestset .
  -- red addrequest(< "住所A", "菊姫", 10 >) .

  -- pr(STOCKSHEETLIST) .
  -- red sumofstock("正宗", stocklist) .
  -- red sumofstock("菊姫", stocklist) .
  -- red sumofstock("正宗", searchstock("正宗")) .

  -- red selectcontainer( "正宗", 11, searchstock("正宗")) .
  -- red selectcontainer( "菊姫", 3, searchstock("菊姫")) .
  -- red selectcontainer( "菊姫", 4, searchstock("菊姫")) .
  -- red selectcontainer( "菊姫", 5, searchstock("菊姫")) .

  -- red checkstock(19932, [ "菊姫", 9 ], < 19932, 201003, 222132, [ "正宗", 10 ] :: [ "菊姫", 10 ] :: empty > :: empty) .
  -- red reposition("菊姫", [ "正宗", 10 ] :: [ "菊姫", 3 ] :: empty , empty) .
  -- red reposition("菊姫", < 19932, 201003, 222132, [ "正宗", 10 ] :: [ "菊姫", 3 ] :: empty >
  --                     :: < 19933, 201003, 222132, [ "正宗", 10 ] :: empty >
  --                     :: < 19934, 201003, 222132, [ "菊姫", 3 ] :: empty > :: empty , empty) .

  -- red makedata("正宗", selectcontainer("正宗", 11, searchstock("正宗"))) .
  -- red makedata("菊姫", selectcontainer("菊姫", 5, searchstock("菊姫"))) .

  -- red remakestockset("正宗", selectcontainer("正宗", 5, searchstock("正宗"))) .
  -- red remakestockset("正宗", selectcontainer("正宗", 10, searchstock("正宗"))) .
  -- red remakestockset("正宗", selectcontainer("正宗", 11, searchstock("正宗"))) .
  -- red remakestockset("菊姫", selectcontainer("菊姫", 3, searchstock("菊姫"))) .
  -- red remakestockset("菊姫", selectcontainer("菊姫", 4, searchstock("菊姫"))) .
  -- red remakestockset("菊姫", selectcontainer("菊姫", 5, searchstock("菊姫"))) .

  -- red makeinstsheet(10000, < "住所A", "正宗", 11 >, selectcontainer("正宗", 11, searchstock("正宗"))) .

  -- op test : -> Execution .
  -- eq test = exe(makeinstsheet(10000, < "住所A", "正宗", 11 >, selectcontainer("正宗", 11, searchstock("正宗")))) .
  --  red test .

--  red runsend(< "住所A", "正宗", 11 >, inputdata) .
--  red runsend(< "住所B", "菊姫", 11 >) .
--  red inexe(runsend(< "住所", "正宗", 10 >)) .

-- ------------------------------------------------------------------------------------------
-- 複合用
-- ------------------------------------------------------------------------------------------


--  red checkproduct("正宗") .
--  red checkproduct("常きげん") .

--  red searchstock("正宗", stocklist) .
--  red searchstock("菊姫", stocklist) .

--  red selectcontainer("正宗", 11, searchstock("正宗", stocklist)) .
```

```
C:\Program Files\cafeOBJ\MainTheme\StockControl-11\Execution.mod
--  red runsend(< "住所Ａ", "正宗", 11 >, inputdata) .
--  red runsend(< "住所Ａ", "正宗", 13 >, inputdata) .
--  red runsend(< "住所Ｂ", "菊姫", 5 >, inputdata) .
--  red runsend(< "住所Ｂ", "菊姫", 11 >, inputdata) .

--  red checkrequest(< 19932, 201003, 222132, [ "正宗", 10 ] :: [ "菊姫", 10 ] :: empty >, < "住所Ａ", "正宗", 11 >) .


--  red selectcontainer(< "住所Ｃ", "菊姫", 5 >, searchstock("菊姫", stocklist)) .


--  red checkrequest(< 19932, 201003, 222132, [ "如水", 5 ] :: [ "手取川", 5 ] :: empty >, < "住所Ａ", "如水", 5 >) .
--  red checkrequest(< 19932, 201003, 222132, [ "如水", 5 ] :: [ "手取川", 5 ] :: empty >, < "住所Ａ", "手取川", 5 >) .
--  red checkrequest(< 19932, 201003, 222132, [ "如水", 5 ] :: [ "手取川", 5 ] :: empty >, < "住所Ｂ", "菊姫", 11 >) .




--  red searchstock(< "住所Ａ", "如水", 11 >, addstock(stocklist, < 19936, 201003, 222132, [ "如水", 5 ] :: empty >)) .

--  red comparestock(< "住所Ａ", "如水", 11 >, searchstock(< "住所Ａ", "如水", 11 >, addstock(stocklist, < 19936, 201003, 222132, [ "如水"
, 5 ] :: empty >))) .
--  red comparestock(< "住所Ａ", "如水",  7 >, searchstock(< "住所Ａ", "如水",  7 >, addstock(stocklist, < 19936, 201003, 222132, [ "如水"
, 5 ] :: empty >))) .

--  red comparestock(< "住所Ａ", "如水",  5 >, searchstock(< "住所Ａ", "如水",  5 >, < 19936, 201003, 222132, [ "如水", 5 ] :: empty > ::
empty)) .
--  red searchstock(< "住所Ａ", "如水",  5 >, < 19936, 201003, 222132, [ "如水", 5 ] :: empty > :: empty) .


-- red addcontainer(< 19936, [ "如水", 5 ] , false >, empty) .
-- red addcontainer(empty, < 19936, [ "如水", 5 ] , false >) .


--  red runarrive(< 19936, 201003, 222132, empty >, inputdata) .
--  red runarrive(< 19936, 201003, 222132, [ "天狗舞", 5 ] :: empty >, inputdata) .

--  red runarrive(< 19936, 201003, 222132, [ "如水", 3 ] :: [ "手取川", 3 ] :: empty >, < empty, empty, empty >) .
--  red runarrive(< 19936, 201003, 222132, [ "如水", 1 ] :: empty >, inputdata) .
--  red runarrive(< 19936, 201003, 222132, [ "如水", 3 ] :: [ "手取川", 3 ] :: empty >, inputdata) .


--  red runarrive(< 19936, 201003, 222132, [ "如水", 5 ] :: [ "手取川", 5 ] :: empty >, < empty, empty, empty >) .
--  red runarrive(< 19936, 201003, 222132, [ "如水", 5 ] :: empty >, < empty, < "住所Ａ", "如水", 5 > :: empty, empty >) .
--  red runarrive(< 19936, 201003, 222132, [ "如水", 5 ] :: empty >, inputdata) .

--  red runarrive(< 19936, 201003, 222132, [ "如水", 3 ] :: [ "手取川", 5 ] ::  empty >, inputdata) .
--  red runarrive(< 19936, 201003, 222132, [ "如水", 5 ] :: [ "手取川", 7 ] :: empty >, inputdata) .


--  red (< 19936, 201003, 222132, [ "如水", 5 ] :: empty >) | over .

--  red s(< 19936, 201003, 222132, [ "如水", 5 ] :: empty >) .
--  red dec(s(< 19936, 201003, 222132, [ "如水", 5 ] :: empty >)) .
--  red dec(r(< "住所Ａ", "如水",  5 >)) .

--  red r(< 19936, 201003, 222132, [ "如水", 5 ] :: empty >) | over .
--  red s(< "住所Ａ", "如水",  5 >) | over .

--  red stockcontrol(s(< 19936, 201003, 222132, [ "如水", 5 ] :: empty >) | r(< "住所Ａ", "菊姫",  5 >) | over) .
```

付録 B

C 言語のソースコード

C:\Documents and Settings\Sakamoto\My Documents\univercity\MainTheme\Program\StockControl-05\DefineData.h

```
#ifndef __DEFINE_DATA_H
#define __DEFINE_DATA_H


#include <stdio.h>
#include <stdlib.h>
#include <string.h>


/*************************************************************/
/* 定数定義                                                  */
/*************************************************************/

// Boolの定義
#ifndef BOOLEAN
#define BOOLEAN int
#endif

#ifndef TRUE
#define TRUE 1
#endif

#ifndef FALSE
#define FALSE 0
#endif

#ifndef MAXCONTENT
#define MAXCONTENT 10
#endif

// 一時メモリ
#ifndef MAXBUFSIZE
#define MAXBUFSIZE 256
#endif


#ifndef MAXNAMESIZE
#define MAXNAMESIZE 128
#endif


#ifndef MAXSENDSIZE
#define MAXSENDSIZE 128
#endif


#ifndef MAXFLAGSIZE
#define MAXFLAGSIZE 8
#endif

#ifndef MAXCONTAINER
#define MAXCONTAINER 99999
#endif

#ifndef MINCONTAINER
#define MINCONTAINER 9999
#endif


/*************************************************************/
/* 型定義                                                    */
/*************************************************************/

// 内蔵品
typedef struct {
    char*   name;   // 商品名
    int     cnt;    // 本数
} CONTENT;


// 内蔵品のリスト
typedef struct _CONTENTLIST {
    CONTENT* data;
    struct  _CONTENTLIST*    next;
} CONTENTLIST;


// 出庫コンテナデータ（出庫用データ）
typedef struct {
    int      num;       // コンテナ番号
    CONTENT  content;   // 内蔵品
    BOOLEAN  flag;      // コンテナ搬出マーク
} SELECTDATA;


typedef struct _SELECTDATALIST {
    SELECTDATA*      data;
    struct  _SELECTDATALIST* next;
} SELECTDATALIST;


// 指示票
typedef struct {
    int     num;            // 注文番号
    char*   sendto;         // 送り先名
    SELECTDATALIST* select; //
} INSTRUCTIONSHEET;


typedef struct _INSTRUCTIONSHEETLIST {
```

```
    INSTRUCTIONSHEET*               data;
    struct _INSTRUCTIONSHEETLIST*   next;
} INSTRUCTIONSHEETLIST;



// 積荷票
typedef struct {
    int     num;      // コンテナ番号
    int     year;     // 搬入年月
    int     date;     // 搬入日時
    CONTENTLIST* content; // 内蔵品
} STOCKSHEET;


typedef struct _STOCKSHEETLIST {
    STOCKSHEET*             data;
    struct _STOCKSHEETLIST* next;
} STOCKSHEETLIST;


// 依頼票
typedef struct {
    char*   sendto;  // 送り先名
    char*   name;    // 品名
    int     cnt;
} REQUESTSHEET;

typedef struct _REQUESTSHEETLIST {
    REQUESTSHEET*               data;
    struct _REQUESTSHEETLIST*   next;
} REQUESTSHEETLIST;

typedef struct {
    STOCKSHEETLIST         *stock;      // 在庫情報
    REQUESTSHEETLIST       *request;    // 在庫不足リスト
    INSTRUCTIONSHEETLIST   *inst;       // 出庫指示票リスト
} EXECUTIONDATA;

typedef struct _PROCUCTSAKELIST {
    char*                   name;
    struct _PROCUCTSAKELIST* next;
} PRODUCTSAKELIST;


/*************************************************************/
/* プロトタイプ関数                                          */
/*************************************************************/

char*                   allocName                   (void);
char*                   allocSendto                 (void);
CONTENTLIST*            allocContentList            (void);
SELECTDATALIST*         allocSelectDataList         (void);
STOCKSHEETLIST*         allocStockSheetList         (void);
REQUESTSHEETLIST*       allocRequestSheetList       (void);
INSTRUCTIONSHEETLIST*   allocInstructionSheetList   (void);
PRODUCTSAKELIST*        allocProductSakeList        (void);


void setContentList             (CONTENTLIST* list, CONTENT data, CONTENTLIST* next);
void setSelectDataList          (SELECTDATALIST* list, SELECTDATA data, SELECTDATALIST* next);
void setStockSheetList          (STOCKSHEETLIST* list, STOCKSHEET data,
                                                    STOCKSHEETLIST* next);
void setRequestSheetList        (REQUESTSHEETLIST* list, REQUESTSHEET data,
                                                      REQUESTSHEETLIST* next);
void setInstructionSheetList    (INSTRUCTIONSHEETLIST* list, INSTRUCTIONSHEET data,
                                                         INSTRUCTIONSHEETLIST* next);


void addContentListCL   (CONTENT cont, CONTENTLIST** list);
void addContentListLC   (CONTENTLIST** list, CONTENT cont);
void addContentListLL   (CONTENTLIST** top, CONTENTLIST** bottom);
void addContentListCLL  (CONTENT data, CONTENTLIST** middle, CONTENTLIST** bottom);

void addSelectDataListSL    (SELECTDATA data, SELECTDATALIST** list);
void addSelectDataListLS    (SELECTDATALIST** list, SELECTDATA data);
void addSelectDataListLL    (SELECTDATALIST** top, SELECTDATALIST** bottom);

void addStockSheetListSL    (STOCKSHEET data, STOCKSHEETLIST** list);
void addStockSheetListLS    (STOCKSHEETLIST** list, STOCKSHEET data);
void addStockSheetListLL    (STOCKSHEETLIST** top, STOCKSHEETLIST** bottom);
void addStockSheetListLSL   (STOCKSHEETLIST** top, STOCKSHEET data, STOCKSHEETLIST** bottom);
void remStockSheetListIL    (int num, STOCKSHEETLIST** list);
void remStockSheetListSL    (STOCKSHEET data, STOCKSHEETLIST** list);

void addRequestSheetListRL  (REQUESTSHEET data, REQUESTSHEETLIST** list);
void addRequestSheetListLR  (REQUESTSHEETLIST** list, REQUESTSHEET data);
void addRequestSheetListLL  (REQUESTSHEETLIST** top, REQUESTSHEETLIST** bottom);

int  getInstructionNumber       (INSTRUCTIONSHEET data);
int  getLastInstructionNumber   (INSTRUCTIONSHEETLIST** list);
void addInstructionSheetListIL  (INSTRUCTIONSHEET data, INSTRUCTIONSHEETLIST** list);
void addInstructionSheetListLI  (INSTRUCTIONSHEETLIST** list, INSTRUCTIONSHEET data);



/*************************************************************/
/* メモリ容量・ノードの確保                                  */
/*************************************************************/
```

C:\Documents and Settings\Sakamoto\My Documents\univercity\MainTheme\Program\StockControl-05\DefineData.h

```
char* allocName(void) {
    return (char*)malloc(MAXNAMESIZE);
}

char* allocSendto(void) {
    return (char*)malloc(MAXSENDSIZE);
}

CONTENT* allocContent(void) {
    return (calloc(1, sizeof(CONTENT)));
}

CONTENTLIST* allocContentList(void) {
    return (calloc(1, sizeof(CONTENTLIST)));
}

SELECTDATA* allocSelectData(void) {
    return (calloc(1, sizeof(SELECTDATA)));
}

SELECTDATALIST* allocSelectDataList(void) {
    return (calloc(1, sizeof(SELECTDATALIST)));
}

STOCKSHEET* allocStockSheet(void) {
    return (calloc(1, sizeof(STOCKSHEET)));
}

STOCKSHEETLIST* allocStockSheetList(void) {
    return (calloc(1, sizeof(STOCKSHEETLIST)));
}

REQUESTSHEET* allocRequestSheet(void) {
    return (calloc(1, sizeof(REQUESTSHEET)));
}

REQUESTSHEETLIST* allocRequestSheetList(void) {
    return (calloc(1, sizeof(REQUESTSHEETLIST)));
}

INSTRUCTIONSHEET* allocInstructionSheet(void) {
    return (calloc(1, sizeof(INSTRUCTIONSHEET)));
}

INSTRUCTIONSHEETLIST* allocInstructionSheetList(void) {
    return (calloc(1, sizeof(INSTRUCTIONSHEETLIST)));
}

PRODUCTSAKELIST* allocProductSakeList(void) {
    return (calloc(1, sizeof(PRODUCTSAKELIST)));
}


/************************************************************/
/* 構造体の操作                                             */
/************************************************************/

void removeName(char* name) {
    free(name);
    return;
}

void removeSendto(char* sendto) {
    free(sendto);
    return;
}

void removeContent(CONTENT* cont) {
    removeName(cont->name);
    free(cont);
    return;
}

void removeContentTop(CONTENTLIST** list) {
    if(*list != NULL) {
        CONTENTLIST* clist = *list;
        CONTENTLIST* cont  = clist->next;

        removeContent(clist->data);
        free(*list);

        if(cont == NULL) *list = NULL;
        else             *list = cont;
    }
    return;
}

void removeContentList(CONTENTLIST** list) {
    while (*list != NULL) removeContentTop(list);
    return;
}

void removeSelectDataTop(SELECTDATALIST** list) {
    if(*list != NULL) {
        SELECTDATALIST* buf    = *list;
        SELECTDATALIST* select = buf->next;

        removeName(buf->data->content.name);
        free(buf->data);
        free(*list);
```

```
        if(select == NULL) *list = NULL;
        else               *list = select;
    }

    return;
}

void removeSelectDataList(SELECTDATALIST** list) {
    while (*list != NULL) removeSelectDataTop(list);
    return;
}


void removeStockSheetTop(STOCKSHEETLIST** list) {
    if(*list != NULL) {
        STOCKSHEETLIST* buf   = *list;
        STOCKSHEETLIST* stock = buf->next;

        removeContentList(&buf->data->content);
        free(buf->data);
        free(*list);

        if(*list == NULL) *list = NULL;
        else              *list = stock;
    }

    return;
}


void removeStockSheetList(STOCKSHEETLIST** list) {
    while (*list != NULL) removeStockSheetTop(list);
    return;
}


void removeRequestSheetTop(REQUESTSHEETLIST** list) {
    if(*list != NULL) {
        REQUESTSHEETLIST* buf     = *list;
        REQUESTSHEETLIST* request = buf->next;

        removeName(buf->data->name);
        removeSendto(buf->data->sendto);
        free(buf->data);
        free(*list);

        if(*list == NULL)  *list = NULL;
        else               *list = request;
    }

    return;
}

void removeRequestSheetList(REQUESTSHEETLIST** list) {
    while (*list != NULL) removeRequestSheetTop(list);
    return;
}

void removeInstructionSheet(INSTRUCTIONSHEET** inst) {
    free(*inst);
    return;
}

void removeInstructionSheetTop(INSTRUCTIONSHEETLIST** list) {
    if(*list != NULL) {
        INSTRUCTIONSHEETLIST* buf  = *list;
        INSTRUCTIONSHEETLIST* inst = buf->next;

        removeSendto(buf->data->sendto);
        removeSelectDataList(&buf->data->select);
        free(buf->data);
        free(*list);

        if(*list == NULL)  *list = NULL;
        else               *list = inst;
    }

    return;
}

void removeInstructionSheetList(INSTRUCTIONSHEETLIST** list) {
    while (*list != NULL) removeInstructionSheetTop(list);
    return;
}



/************************************************************/
/* 構造体の操作                                             */
/************************************************************/

// メンバ値の設定
void setContentList(CONTENTLIST* list, CONTENT data, CONTENTLIST* next) {
    list->data       = allocContent();
    list->data->name = allocName();
    strcpy(list->data->name, data.name);
    list->data->cnt = data.cnt;
    list->next = next;
}
```

```
void setSelectDataList(SELECTDATALIST* list, SELECTDATA data, SELECTDATALIST* next) {

    list->data              = allocSelectData();
    list->data->num         = data.num;
    list->data->content.cnt  = data.content.cnt;
    list->data->flag        = data.flag;
    list->next              = next;
    list->data->content.name = allocName();
    strcpy(list->data->content.name, data.content.name);
}


void setStockSheetList(STOCKSHEETLIST* list, STOCKSHEET data,
                                             STOCKSHEETLIST* next) {
    list->data         = allocStockSheet();
    list->data->content = allocContentList();
    list->data->content->next = NULL;

    list->data->num  = data.num;
    list->data->year = data.year;
    list->data->date = data.date;
    list->next       = next;

    setContentList(list->data->content, *(data.content->data), data.content->next);
}

void setRequestSheetList(REQUESTSHEETLIST* list, REQUESTSHEET data,
                                                 REQUESTSHEETLIST* next) {

    list->data          = allocRequestSheet();
    list->data->sendto   = allocSendto();
    list->data->name     = allocName();
    list->data->cnt      = data.cnt;
    list->next          = next;

    strcpy(list->data->sendto, data.sendto);
    strcpy(list->data->name, data.name);
}

void setInstructionSheetList(INSTRUCTIONSHEETLIST* list, INSTRUCTIONSHEET data,
                                                         INSTRUCTIONSHEETLIST* next) {
    list->data        = allocInstructionSheet();
    list->data->sendto = allocSendto();
    list->data->select = allocSelectDataList();

    list->data->num    = data.num;
    list->next = next;

    strcpy(list->data->sendto, data.sendto);
    setSelectDataList(list->data->select, *(data.select->data), data.select->next);
}


/************************************************************/
/* ContentList関数                                          */
/************************************************************/

// 先頭にノードの挿入
void addContentListCL(CONTENT cont, CONTENTLIST** list) {
    CONTENTLIST* ptr = *list;
    *list = allocContentList();
    setContentList(*list, cont, ptr);
}

// 末尾にノード挿入
void addContentListLC(CONTENTLIST** list, CONTENT cont) {
    if(*list == NULL) addContentListCL(cont, list);
    else {
        CONTENTLIST* ptr = *list;
        while(ptr->next != NULL) ptr = ptr->next;
        ptr->next = allocContentList();
        setContentList(ptr->next, cont, NULL);
    }
}

void addContentListLL(CONTENTLIST** top, CONTENTLIST** bottom) {
    if(*top == NULL) *top = *bottom;
    else {
        CONTENTLIST* ptr = *top;
        while(ptr->next != NULL) ptr = ptr->next;
        ptr->next = *bottom;
    }
}

void addContentListCLL(CONTENT data, CONTENTLIST** middle, CONTENTLIST** bottom) {
    addContentListCL(data, middle);
    addContentListLL(middle, bottom);
}


/************************************************************/
/* SelectDataList関数                                       */
/************************************************************/

void addSelectDataListSL(SELECTDATA data, SELECTDATALIST** list) {
    SELECTDATALIST* ptr = *list;
    *list = allocSelectDataList();
    setSelectDataList(*list, data, ptr);
}
```

```
void addSelectDataListLS(SELECTDATALIST** list, SELECTDATA data) {
    if(*list == NULL) addSelectDataListSL(data, list);
    else {
        SELECTDATALIST* ptr = *list;
        while(ptr->next != NULL) ptr = ptr->next;
        ptr->next = allocSelectDataList();
        setSelectDataList(ptr->next, data, NULL);
    }
}

void addSelectDataListLL(SELECTDATALIST** top, SELECTDATALIST** bottom) {
    if(*top == NULL) *top = *bottom;
    else {
        SELECTDATALIST* ptr = *top;
        while(ptr->next != NULL) ptr = ptr->next;
        ptr->next = *bottom;
    }
}


/************************************************************/
/* StockDataList関数                                        */
/************************************************************/

void addStockSheetListSL(STOCKSHEET data, STOCKSHEETLIST** list) {
    STOCKSHEETLIST* ptr = *list;
    *list = allocStockSheetList();
    setStockSheetList(*list, data, ptr);
}

void addStockSheetListLS(STOCKSHEETLIST** list, STOCKSHEET data) {
    if(*list == NULL) addStockSheetListSL(data, list);
    else {
        STOCKSHEETLIST* ptr = *list;
        while(ptr->next != NULL) ptr = ptr->next;
        ptr->next = allocStockSheetList();
        setStockSheetList(ptr->next, data, NULL);
    }
}

void addStockSheetListLL(STOCKSHEETLIST** top, STOCKSHEETLIST** bottom) {
    if(*top == NULL) *top = *bottom;
    else {
        STOCKSHEETLIST* ptr = *top;
        while(ptr->next != NULL) ptr = ptr->next;
        ptr->next = *bottom;
    }
}

void addStockSheetListLSL(STOCKSHEETLIST** top, STOCKSHEET data,
                                                STOCKSHEETLIST** bottom) {
    if(*top == NULL) {
        addStockSheetListSL(data, bottom);
        *top = *bottom;
    }
    else {
        addStockSheetListLS(top, data);
        addStockSheetListLL(top, bottom);
    }
}


void remStockSheetListIL(int num, STOCKSHEETLIST** list) {

    if (*list == NULL) return;
    else {
        STOCKSHEETLIST* stock = *list;
        STOCKSHEETLIST* next  = stock->next;

        if (num == stock->data->num) {
            removeStockSheetTop(list);
            return;
        }
        else {
            while(next != NULL) {
                if (num == next->data->num) {
                    removeStockSheetTop(&next);
                    stock->next = next;
                    break;
                }
                else {
                    stock = next;
                    next  = next->next;
                }
            }
        }
    }
    return;
}

void remStockSheetListSL(STOCKSHEET data, STOCKSHEETLIST** list) {
    if(*list == NULL) return;
    remStockSheetListIL(data.num, list);
}


/************************************************************/
/* RequestDataList関数                                      */
/************************************************************/
```

```
void addRequestSheetListRL(REQUESTSHEET data, REQUESTSHEETLIST** list) {
    REQUESTSHEETLIST* ptr = *list;
    *list = allocRequestSheetList();
    setRequestSheetList(*list, data, ptr);
}

void addRequestSheetListLR(REQUESTSHEETLIST** list, REQUESTSHEET data) {
    if(*list == NULL) addRequestSheetListRL(data, list);
    else {
        REQUESTSHEETLIST* ptr = *list;
        while(ptr->next != NULL) ptr = ptr->next;
        ptr->next = allocRequestSheetList();
        setRequestSheetList(ptr->next, data, NULL);
    }
}

void addRequestSheetListLL(REQUESTSHEETLIST** top, REQUESTSHEETLIST** bottom) {
    if(*top == NULL) *top = *bottom;
    else {
        REQUESTSHEETLIST* ptr = *top;
        while(ptr->next != NULL) ptr = ptr->next;
        ptr->next = *bottom;
    }
}


/************************************************************/
/* InstructionDataList関数                                  */
/************************************************************/

int getInstructionNumber(INSTRUCTIONSHEET data) {
    return data.num;
}

int getLastInstructionNumber(INSTRUCTIONSHEETLIST** list) {
    int buf = 0;
    INSTRUCTIONSHEETLIST* ptr = *list;

    if(ptr == NULL) ;  // do nothing
    else {
        if(getInstructionNumber(*(ptr->data)) == MAXCONTAINER)
            buf = MINCONTAINER;
        else {
            if (ptr->next == NULL)
                buf = getInstructionNumber(*(ptr->data));
            else
                buf = getLastInstructionNumber(&(ptr->next));
        }
    }

    return buf;
}


void addInstructionSheetListIL(INSTRUCTIONSHEET data,
                               INSTRUCTIONSHEETLIST** list) {
    INSTRUCTIONSHEETLIST* ptr = *list;
    *list = allocInstructionSheetList();
    setInstructionSheetList(*list, data, ptr);
}

void addInstructionSheetListLI(INSTRUCTIONSHEETLIST** list, INSTRUCTIONSHEET data) {
    if(*list == NULL) {
        addInstructionSheetListIL(data, list);
    }
    else {
        INSTRUCTIONSHEETLIST* ptr = *list;
        while(ptr->next != NULL) ptr = ptr->next;
        ptr->next = allocInstructionSheetList();
        setInstructionSheetList(ptr->next, data, NULL);
    }
}




/************************************************************/
/* ProductName用関数                                        */
/************************************************************/

void setProductSakeList(PRODUCTSAKELIST* list, char *name, PRODUCTSAKELIST* next) {
    list->name = allocName();
    strcpy(list->name, name);
    list->next = next;
}


void addProductSakeListNL(char* name, PRODUCTSAKELIST** list) {
    PRODUCTSAKELIST* ptr = *list;
    *list = allocProductSakeList();
    setProductSakeList(*list, name, ptr);
}

void addProductSakeList(PRODUCTSAKELIST** list, char* name) {
    if(*list == NULL) addProductSakeListNL(name, list);
    else {
        PRODUCTSAKELIST* ptr = *list;
        while(ptr->next != NULL) ptr = ptr->next;
        ptr->next = allocProductSakeList();
```

```
        setProductSakeList(ptr->next, name, NULL);
    }
}

#endif
```

C:\Documents and Settings\Sakamoto\My Documents\univercity\MainTheme\Program\StockControl-05\StockForC.h

```
/************************************************************/
/* 仕様記述外の関数                                         */
/************************************************************/


#ifndef __STOCK_FOR_C_H
#define __STOCK_FOR_C_H

#include <stdio.h>
#include <string.h>

#include "DefineData.h"


enum {
    READ, WRITE, ADDWRITE, ADDPLUS
} FILECONTROL;


/************************************************************/
/* プロトタイプ宣言                                         */
/************************************************************/

// プリント関数
void printBoolean               (int flag);
void printContentList           (CONTENTLIST** list);
void printSelectDataList        (SELECTDATALIST** list);
void printStockSheetList        (STOCKSHEETLIST** list);
void printRequestSheetList      (REQUESTSHEETLIST** list);
void printInstructionSheetList  (INSTRUCTIONSHEETLIST** list);


FILE*   fileOpen            (char* fname, int num);
void    readStrings         (char *read, char *str);
void    openStockData       (STOCKSHEETLIST** list);
void    openRequestData     (REQUESTSHEETLIST** list);
void    openInstructionData (INSTRUCTIONSHEETLIST** list);

void    productNameList(PRODUCTSAKELIST **list);

CONTENTLIST*    copyContentListLoop (CONTENTLIST **list, CONTENTLIST *ptr);
CONTENTLIST*    copyContentList (CONTENTLIST *list);
STOCKSHEETLIST* copyStockSheetListLoop (STOCKSHEETLIST **list, STOCKSHEETLIST *ptr);
STOCKSHEETLIST* copyStockSheetList (STOCKSHEETLIST **list);

/************************************************************/
/* プリント関数                                             */
/************************************************************/

void printBoolean(int flag) {
    if (flag == TRUE)   printf("TRUE");
    else                printf("FALSE");

    return;
}

void printContentList(CONTENTLIST** list) {
    CONTENTLIST *ptr = *list;

    printf("--- print content list ---\n");
    while(ptr != NULL) {
        printf("%s\t%3d\n", ptr->data->name, ptr->data->cnt);
        ptr = ptr->next;
    }
    printf("--- over ---\n");

    return;
}


void printSelectDataList(SELECTDATALIST** list) {
    SELECTDATALIST *ptr = *list;

    printf("--- print select list ---\n");
    while(ptr != NULL) {
        printf("%d\t%s\t%d\t", ptr->data->num, ptr->data->content.name,
                                               ptr->data->content.cnt);
        printBoolean(ptr->data->flag);
        printf("\n");
        ptr = ptr->next;
    }

    printf("--- over ---\n");

    return;
}


void printStockSheetList(STOCKSHEETLIST** list) {
    STOCKSHEETLIST *ptr = *list;

    printf("--- print stock list ---\n");
    while(ptr != NULL) {
        printf("%d\t%d\t%d\t", ptr->data->num, ptr->data->year, ptr->data->date);

        CONTENTLIST *buf = ptr->data->content;
        while(buf != NULL) {
            printf("%s\t%d\t", buf->data->name, buf->data->cnt);
            buf = buf->next;
        }
```

```
        printf("\n");
        ptr = ptr->next;
    }
    printf("--- over ---\n");

    return;
}

void printRequestSheetList(REQUESTSHEETLIST** list) {
    REQUESTSHEETLIST *ptr = *list;

    printf("--- print request list ---\n");
    while(ptr != NULL) {
        printf("%s\t%s\t%d\n", ptr->data->sendto, ptr->data->name,
                                                  ptr->data->cnt);
        ptr = ptr->next;
    }
    printf("--- over ---\n");

    return;
}

void printInstructionSheetList(INSTRUCTIONSHEETLIST** list) {
    INSTRUCTIONSHEETLIST *ptr = *list;

    printf("--- print instruction list ---\n");
    while(ptr != NULL) {
        printf("%d\t%s\t", ptr->data->num, ptr->data->sendto);

        printf("\n");

        SELECTDATALIST *buf = ptr->data->select;

        while(buf != NULL) {
            printf("%d\t%s\t%d\t", buf->data->num,
                                   buf->data->content.name,
                                   buf->data->content.cnt);
            printBoolean(buf->data->flag);
            printf("\n");
            buf = buf->next;
        }
        ptr = ptr->next;
    }
    printf("--- over ---\n");

    return;
}


/************************************************************/
/* ファイル操作関数                                         */
/************************************************************/

// ファイルオープン
FILE* fileOpen(char* fname, int num) {
    FILE* fp;
    FILECONTROL = num;

    switch(FILECONTROL) {
      case READ     : fp = fopen(fname, "r");   break;
      case WRITE    : fp = fopen(fname, "w");   break;
      case ADDWRITE : fp = fopen(fname, "a");   break;
      case ADDPLUS  : fp = fopen(fname, "a+");  break;
      default : fp = NULL;
    }

    if(fp == NULL) {
        printf("File Errer\n");
        exit(1);
    }

    return fp;
}

void readStrings(char *read, char *str) {
    int i = 0, j = 0;

    while(1) {
        // 空白文字をスキップ
        if      (str[j] == 32)    {j++; continue;}
        else if (str[j] == '\t')  {j++; continue;}
        else if (str[j] == '\0')   break;

        read[i] = str[j];
        i++;    j++;
    }

    read[i] = '\0';

    return;
}


void openStockData (STOCKSHEETLIST** list) {

    char *str, *str2, *str3;
    FILE *fp;
    STOCKSHEET stock;

    str  = (char *)malloc(sizeof(char) * MAXBUFSIZE);
```

```
    str3 = (char *)malloc(sizeof(char) * MAXBUFSIZE);
    if ((str  == NULL) || (str3 == NULL))  exit(1);

    fp = fileOpen("StockData.dat", READ);

    stock.content->data->name = allocName();


    while(fgets(str, MAXBUFSIZE, fp) != NULL) {

        str2 = strtok(str, ",");
        sscanf(str2, "%d", &stock.num);

        str2 = strtok(NULL, ",");
        sscanf(str2, "%d", &stock.year);

        str2 = strtok(NULL, ",");
        sscanf(str2, "%d", &stock.date);

        str2 = strtok(NULL, ",");
        readStrings(str3, str2);
        strcpy(stock.content->data->name, str3);

        str2 = strtok(NULL, ",");
        sscanf(str2, "%d", &stock.content->data->cnt);

        stock.content->next = NULL;

        while((str2 = strtok(NULL, ",")) != NULL) {
            CONTENT ptr;

            ptr.name = allocName();
            readStrings(str3, str2);
            strcpy(ptr.name, str3);

            str2 = strtok(NULL, ",");
            sscanf(str2, "%d", &ptr.cnt);

            addContentListLC(&(stock.content->next), ptr);
        }

        addStockSheetListLS(list, stock);
    }

    fclose(fp);
    free(str);
    free(str3);
}

void openRequestData(REQUESTSHEETLIST** list) {
    char *str, *str2, *str3;
    FILE *fp;
    REQUESTSHEET request;

    str  = (char *)malloc(sizeof(char) * MAXBUFSIZE);
    str3 = (char *)malloc(sizeof(char) * MAXBUFSIZE);
    if ((str  == NULL) || (str3 == NULL))  exit(1);

    fp = fileOpen("RequestData.dat", READ);

    request.sendto = allocSendto();
    request.name   = allocName();

    while(fgets(str, MAXBUFSIZE, fp) != NULL) {

        str2 = strtok(str, ",");
        readStrings(str3, str2);
        strcpy(request.sendto, str3);

        str2 = strtok(NULL, ",");
        readStrings(str3, str2);
        strcpy(request.name, str3);

        str2 = strtok(NULL, ",");
        sscanf(str2, "%d", &request.cnt);

        addRequestSheetListLR(list, request);
    }

    fclose(fp);
    free(str);
    free(str3);
}


void openInstructionData (INSTRUCTIONSHEETLIST** list) {
    char *str, *str2, *str3;
    FILE *fp;
    INSTRUCTIONSHEET inst;

    str  = (char *)malloc(sizeof(char) * MAXBUFSIZE);
    str3 = (char *)malloc(sizeof(char) * MAXBUFSIZE);
    if ((str  == NULL) || (str3 == NULL))  exit(1);

    fp = fileOpen("InstructionData.dat", READ);

    inst.sendto = allocSendto();
    inst.select = allocSelectDataList();
    inst.select->data = allocSelectData();
    inst.select->data->content.name = allocName();
```

```
    while(fgets(str, MAXBUFSIZE, fp) != NULL) {

        str2 = strtok(str, ",");
        sscanf(str2, "%d", &inst.num);

        str2 = strtok(NULL, ",");
        readStrings(str3, str2);
        strcpy(inst.sendto, str3);

        str2 = strtok(NULL, ",");
        sscanf(str2, "%d", &inst.select->data->num);

        str2 = strtok(NULL, ",");
        readStrings(str3, str2);
        strcpy(inst.select->data->content.name, str3);

        str2 = strtok(NULL, ",");
        sscanf(str2, "%d", &inst.select->data->content.cnt);

        str2 = strtok(NULL, ",");
        readStrings(str3, str2);
        if ((strcmp("FALSE", str3)) != 0) inst.select->data->flag = TRUE;
        else                              inst.select->data->flag = FALSE;

        inst.select->next = NULL;

        while((str2 = strtok(NULL, ",")) != NULL) {
            SELECTDATA ptr;

            ptr.content.name = allocName();

            sscanf(str2, "%d", &ptr.num);

            str2 = strtok(NULL, ",");
            readStrings(str3, str2);
            strcpy(ptr.content.name, str3);

            str2 = strtok(NULL, ",");
            sscanf(str2, "%d", &ptr.content.cnt);

            str2 = strtok(NULL, ",");
            readStrings(str3, str2);
            if ((strcmp("FALSE", str3)) != 0) ptr.flag = TRUE;
            else                              ptr.flag = FALSE;


            addSelectDataListLS(&(inst.select->next), ptr);
        }

        addInstructionSheetListLI(list, inst);
    }

    fclose(fp);
    free(str);
    free(str3);
}



/************************************************************/
/* 取扱商品リスト読込関数                                   */
/************************************************************/

void productNameList(PRODUCTSAKELIST **list) {

    char* str = allocName();
    char* buf = allocName();
    if ((str == NULL) || (buf == NULL)) exit(1);

    FILE* fp = fileOpen("ProductName.dat", READ);

    while(fgets(str, MAXBUFSIZE, fp) != NULL) {
        sscanf(str, "%s", buf);
        addProductSakeList(list, buf);
    }

    fclose(fp);
    free(str);
}



/************************************************************/
/*  リストコピー関数                                        */
/************************************************************/

CONTENTLIST* copyContentListLoop (CONTENTLIST **list, CONTENTLIST *ptr) {
    CONTENTLIST *cont = *list;

    if(cont == NULL) return ptr;
    else {
        CONTENT buf;

        buf.cnt  = cont->data->cnt;
        buf.name = allocName();

        strcpy(buf.name, cont->data->name);
        addContentListLC(&ptr, buf);
    }
```

```
    return copyContentListLoop(&cont->next, ptr);
}

CONTENTLIST* copyContentList (CONTENTLIST *list) {

    CONTENTLIST *ptr  = NULL;

    if(list == NULL) {
        ptr->data = NULL;
        ptr->next = NULL;
        return ptr;
    }

    return copyContentListLoop(&list, ptr);
}


STOCKSHEETLIST* copyStockSheetListLoop (STOCKSHEETLIST **list, STOCKSHEETLIST *ptr) {
    STOCKSHEETLIST *stck = *list;

    if(stck == NULL) return ptr;
    else {
        STOCKSHEET buf;

        buf.num  = stck->data->num;
        buf.year = stck->data->year;
        buf.date = stck->data->date;

        buf.content = copyContentList(stck->data->content);

        addStockSheetListLS(&ptr, buf);
    }

    return copyStockSheetListLoop(&stck->next, ptr);
}

STOCKSHEETLIST* copyStockSheetList (STOCKSHEETLIST **list) {
    STOCKSHEETLIST *ptr  = NULL;
    return copyStockSheetListLoop(list, ptr);
}

#endif
```

C:\Documents and Settings\Sakamoto\My Documents\univercity\MainTheme\Program\StockControl-05\SendExe.h

```
/************************************************************/
/* 依頼票受理用関数                                         */
/************************************************************/


#ifndef __SEND_EXE_H
#define __SEND_EXE_H


#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#include "DefineData.h"
#include "StockForC.h"


/************************************************************/
/* プロトタイプ関数                                         */
/************************************************************/

BOOLEAN checkProductLoop (char *str, PRODUCTSAKELIST **list);
BOOLEAN checkProduct     (char *str);
BOOLEAN checkProductR    (REQUESTSHEET request);


BOOLEAN searchContainer (char *str, CONTENTLIST **list);
STOCKSHEETLIST** searchStockLoop  (char *str, STOCKSHEETLIST **list, STOCKSHEETLIST **ptr);
STOCKSHEETLIST** searchStock      (char *str, STOCKSHEETLIST **list);
STOCKSHEETLIST** searchStockRL    (REQUESTSHEET request, STOCKSHEETLIST **list);

int getStockCnt (char *str, CONTENTLIST **list);

int sumOfStockLoop  (char *str, STOCKSHEETLIST **list, int cnt);
int sumOfStock      (char *str, STOCKSHEETLIST **list);




/************************************************************/
/* 取扱商品用関数                                           */
/************************************************************/

BOOLEAN checkProductLoop(char *str, PRODUCTSAKELIST **list) {
    PRODUCTSAKELIST *ptr = *list;

    if(ptr == NULL) return FALSE;

    return ((strcmp(str, ptr->name)) == 0) ? (TRUE) : (checkProductLoop(str, &(ptr->next)));
}

BOOLEAN checkProduct(char *str) {
    PRODUCTSAKELIST *list = NULL;
    productNameList(&list);

    return checkProductLoop(str, &list);
}

BOOLEAN checkProductR(REQUESTSHEET request) {
    return checkProduct(request.name);
}


/************************************************************/
/* 在庫検索用関数                                           */
/************************************************************/

BOOLEAN searchContainer(char *str, CONTENTLIST **list) {
    CONTENTLIST *ptr = *list;

    if(ptr == NULL) return FALSE;

    return ((strcmp(str, ptr->data->name)) == 0) ? (TRUE) : (searchContainer(str, &(ptr->next)));
}


STOCKSHEETLIST** searchStockLoop(char *str, STOCKSHEETLIST **list, STOCKSHEETLIST **ptr) {

    if (*list == NULL) return ptr;

    STOCKSHEETLIST *stck = *list;
    CONTENTLIST    *cont = stck->data->content;

    if (searchContainer(str, &(cont))) {
        addStockSheetListLS(ptr, *stck->data);
    }

    return searchStockLoop(str, &(stck->next), ptr);
}

STOCKSHEETLIST** searchStock(char *str, STOCKSHEETLIST **list) {
    STOCKSHEETLIST *ptr = NULL;
    return searchStockLoop(str, list, &ptr);
}

STOCKSHEETLIST** searchStockRL(REQUESTSHEET request, STOCKSHEETLIST **list) {
    STOCKSHEETLIST *ptr = NULL;
    return searchStockLoop(request.name, list, &ptr);
}
```

C:\Documents and Settings\Sakamoto\My Documents\univercity\MainTheme\Program\StockControl-05\SendExe.h

```
/*************************************************************/
/* 内臓品集計用関数                                          */
/*************************************************************/

int getStockCnt(char *str, CONTENTLIST **list) {
    CONTENTLIST *ptr = *list;
    if(ptr == NULL) return 0;

    return ((strcmp(str, ptr->data->name)) == 0) ?
        ptr->data->cnt : getStockCnt(str, &(ptr->next));
}


/*************************************************************/
/* 在庫計算用関数                                            */
/*************************************************************/

int sumOfStockLoop(char *str, STOCKSHEETLIST **list, int cnt) {
    STOCKSHEETLIST *ptr = *list;

    if (ptr != NULL) {
        CONTENTLIST *buf = ptr->data->content;
        return sumOfStockLoop(str, &(ptr->next), (cnt + getStockCnt(str, &(buf))));
    }
    else return cnt;
}

int sumOfStock(char *str, STOCKSHEETLIST **list) {
    return sumOfStockLoop(str, list, 0);
}


#endif
```

C:\Documents and Settings\Sakamoto\My Documents\univercity\MainTheme\Program\StockControl-05\RequestExe.h

```
/*************************************************************/
/* 積荷票受付用関数                                          */
/*************************************************************/


#ifndef __REQUEST_EXE_H_
#define __REQUEST_EXE_H_

#include <stdio.h>
#include <string.h>

#include "DefineData.h"
#include "StockForC.h"



/*************************************************************/
/* プロトタイプ関数                                          */
/*************************************************************/

BOOLEAN checkRequestLoop(char *str, CONTENTLIST **list);
BOOLEAN checkRequestCS(char *str, STOCKSHEET data);
BOOLEAN checkRequestSR(STOCKSHEET stock, REQUESTSHEET request);

STOCKSHEET searchNumber(STOCKSHEET data, STOCKSHEETLIST **list);




/*************************************************************/
/* 在庫比較用関数                                            */
/*************************************************************/

BOOLEAN checkRequestLoop(char *str, CONTENTLIST **list) {
    CONTENTLIST *ptr = *list;
    if(ptr == NULL) return FALSE;

    return ((strcmp(str, ptr->data->name)) == 0) ?
        TRUE : checkRequestLoop(str, &(ptr->next));
}

BOOLEAN checkRequestCS(char *str, STOCKSHEET data) {
    CONTENTLIST *cont = data.content;
    return checkRequestLoop(str, &(cont));
}

BOOLEAN checkRequestSR(STOCKSHEET stock, REQUESTSHEET request) {
    CONTENTLIST *cont = stock.content;
    return checkRequestLoop(request.name, &(cont));
}


/*************************************************************/
/* 積荷票検索用関数                                          */
/*************************************************************/

STOCKSHEET searchNumber(STOCKSHEET data, STOCKSHEETLIST **list) {
    STOCKSHEETLIST *ptr = *list;


    if (ptr == NULL) {
        STOCKSHEET buf = {data.num, data.year, data.date};
        buf.content       = NULL;
        buf.content->data = NULL;
        buf.content->next = NULL;

        return buf;
    }

    if (data.num == ptr->data->num) {
        STOCKSHEET buf = {data.num, data.year, data.date};
        buf.content = copyContentList(ptr->data->content);

        return buf;
    }
    else {
        return searchNumber(data, &(ptr->next));
    }
}

#endif
```

C:\Documents and Settings\Sakamoto\My Documents\univercity\MainTheme\Program\StockControl-05\CommonExe.h

```
#ifndef __COMMON_EXE_H_
#define __COMMON_EXE_H_

#include <stdio.h>
#include <string.h>

#include "DefineData.h"
#include "SendExe.h"


/*************************************************************/
/* プロトタイプ宣言                                          */
/*************************************************************/

CONTENTLIST* selectConrainerList(char *str, int cnt, CONTENTLIST **list, CONTENTLIST *ptr);

STOCKSHEETLIST* selectContainerLoop(char *str, int cnt, STOCKSHEETLIST **list, STOCKSHEETLIST *ptr);
STOCKSHEETLIST* selectContainerCIL (char *str, int cnt, STOCKSHEETLIST **list);
STOCKSHEETLIST* selectContainerRL  (REQUESTSHEET request, STOCKSHEETLIST **list);

CONTENTLIST* rePositionContentCL(char *str, CONTENTLIST **list, CONTENTLIST *cont);
CONTENTLIST* rePositionContent(char *str, CONTENTLIST **list);
STOCKSHEETLIST* rePositionCLL(char *str, STOCKSHEETLIST **list, STOCKSHEETLIST *stock);
STOCKSHEETLIST* rePosition(char *str, STOCKSHEETLIST **list);

BOOLEAN checkStockSheetList(int num, CONTENT cont, STOCKSHEETLIST **list);
STOCKSHEETLIST* checkStockSheetListCntLoop(int num, CONTENT cont,
                                           STOCKSHEETLIST *ptr, STOCKSHEETLIST **list);

SELECTDATALIST* makeSelectDataLoop(char *str, STOCKSHEETLIST **list,
                                   SELECTDATALIST* select, STOCKSHEETLIST *ptr);
SELECTDATALIST* makeSelectData(char *str, STOCKSHEETLIST **list, STOCKSHEETLIST *ptr);

STOCKSHEETLIST* remarkStockSheetListSet(STOCKSHEETLIST** list, STOCKSHEETLIST** ptr);
STOCKSHEETLIST* remarkStockSheetList(char *str, STOCKSHEETLIST** list, STOCKSHEETLIST** ptr);

INSTRUCTIONSHEET makeInstructionSheet(int num, REQUESTSHEET request,
                                      STOCKSHEETLIST** list, STOCKSHEETLIST** ptr);




/*************************************************************/
/* 出庫コンテン用関数(内蔵品用)                              */
/*************************************************************/

// コンテナの内蔵品のリストの先頭を出庫品にし，
// その本数を上書きする．
CONTENTLIST* selectConrainerList(char *str, int cnt, CONTENTLIST **list, CONTENTLIST *ptr) {
    CONTENTLIST *buf  = *list;

    if ((strcmp(str, buf->data->name)) == 0) {
        buf->data->cnt = cnt;
        addContentListLL(&buf, &ptr);

        return buf;
    }
    else {
        addContentListCL(*(buf->data), &ptr);
        return selectConrainerList(str, cnt, &(buf->next), ptr);
    }
}


/*************************************************************/
/* 出庫コンテン用関数                                        */
/*************************************************************/

STOCKSHEETLIST* selectContainerLoop(char *str, int cnt,
                                    STOCKSHEETLIST **list, STOCKSHEETLIST *ptr) {
    STOCKSHEETLIST *stck = *list;
    CONTENTLIST    *cont = stck->data->content;

    int count = getStockCnt(str, &cont);

    if(cnt <= count) {
        STOCKSHEET buf;

        buf.num     = stck->data->num;
        buf.year    = stck->data->year;
        buf.date    = stck->data->date;
        buf.content = selectConrainerList(str, cnt, &(cont), NULL);

        addStockSheetListLS(&ptr, buf);

        return ptr;
    }
    else {
        addStockSheetListLS(&ptr, *(stck->data));
        return selectContainerLoop(str, (cnt - count), &(stck->next), ptr);
    }
}


STOCKSHEETLIST* selectContainerCIL(char *str, int cnt, STOCKSHEETLIST **list) {
    STOCKSHEETLIST *ptr = NULL;
    return selectContainerLoop(str, cnt, list, ptr);
}

STOCKSHEETLIST* selectContainerRL(REQUESTSHEET request, STOCKSHEETLIST **list) {
```

```
    STOCKSHEETLIST *ptr = NULL;
    return selectContainerLoop(request.name, request.cnt, list, ptr);
}


/************************************************************/
/* コンテナ再配置用関数                                     */
/************************************************************/

CONTENTLIST* rePositionContentCL(char *str, CONTENTLIST **list, CONTENTLIST *cont) {
    CONTENTLIST *ptr = *list;
    if(ptr == NULL) return cont;

    if((strcmp(str, ptr->data->name)) == 0) {
        addContentListLL(&ptr, &cont);
        return ptr;
    }
    else {
        addContentListLC(&cont, *(ptr->data));
        return rePositionContentCL(str, &(ptr->next), cont);
    }
}

CONTENTLIST* rePositionContent(char *str, CONTENTLIST **list) {
    CONTENTLIST *cont = NULL;
    return rePositionContentCL(str, list, cont);
}


STOCKSHEETLIST* rePositionCLL(char *str, STOCKSHEETLIST **list, STOCKSHEETLIST *stock) {
    STOCKSHEETLIST* ptr  = *list;
    CONTENTLIST*    buf  = NULL;

    if(ptr == NULL) return stock;

    CONTENTLIST* cont = ptr->data->content;
    ptr->data->content = rePositionContent(str, &cont);
    addStockSheetListLS(&stock, *(ptr->data));

    return rePositionCLL(str, &(ptr->next), stock);
}


STOCKSHEETLIST* rePosition(char *str, STOCKSHEETLIST **list) {
    STOCKSHEETLIST* ptr = NULL;
    return rePositionCLL(str, list, ptr);
}


/************************************************************/
/* 積荷票が空か調査する関数                                 */
/************************************************************/

BOOLEAN checkStockSheetList(int num, CONTENT cont, STOCKSHEETLIST **list) {
    STOCKSHEETLIST *ptr = *list;

    if(num == ptr->data->num) {
        if((strcmp(cont.name, ptr->data->content->data->name)) == 0) {
            if(cont.cnt < ptr->data->content->data->cnt) {
                return FALSE;
            }
            else {
                if(ptr->data->content->next == NULL) return TRUE;
                else                                 return FALSE;
            }
        }
        else {
            STOCKSHEETLIST* buf = allocStockSheetList();
            CONTENTLIST*   conl = ptr->data->content->next;

            buf->data->num  = ptr->data->num;
            buf->data->year = ptr->data->year;
            buf->data->date = ptr->data->date;
            buf->next       = ptr->next;

            buf->data->content->data->cnt  = conl->data->cnt;
            buf->data->content->data->name = allocName();
            strcpy(buf->data->content->data->name, conl->data->name);
            buf->data->content->next = conl->next;

            return checkStockSheetList(num, cont, &buf);
        }
    }
    else {
        return checkStockSheetList(num, cont, &ptr->next);
    }
}


STOCKSHEETLIST* checkStockSheetListCntLoop(int num, CONTENT cont,
                                           STOCKSHEETLIST *ptr, STOCKSHEETLIST **list) {
    STOCKSHEETLIST *stck = *list;
    if(stck == NULL)    return ptr;

    CONTENT      cdat = *(stck->data->content->data);
    CONTENTLIST* cntl = stck->data->content->next;

    STOCKSHEET buf;

    buf.num  = stck->data->num;
```

```
    buf.year = stck->data->year;
    buf.date = stck->data->date;

    buf.content = allocContentList();
    buf.content->data = allocContent();
    buf.content->next = NULL;

    if(num == stck->data->num) {
        if((strcmp(cont.name, stck->data->content->data->name)) == 0) {
            if(cont.cnt < stck->data->content->data->cnt) {
                buf.content->data->cnt  = cdat.cnt - cont.cnt;
                buf.content->data->name = allocName();
                strcpy(buf.content->data->name, cdat.name);
                buf.content->next = cntl;

                addStockSheetListLS(&ptr, buf);
                addStockSheetListLL(&ptr, &stck->next);

                return ptr;
            }
            else {
                if(cntl == NULL) {
                    addStockSheetListLL(&ptr, &stck->next);
                    return ptr;
                }
                else {
                    buf.content->data = cntl->data;
                    buf.content->next = cntl->next;

                    addStockSheetListLS(&ptr, buf);
                    addStockSheetListLL(&ptr, &stck->next);

                    return ptr;
                }
            }
        }
        else {
            buf.content->data->cnt  = cntl->data->cnt;
            buf.content->data->name = allocName();
            strcpy(buf.content->data->name, cntl->data->name);
            buf.content->next = cntl->next;

            addContentListLC(&buf.content->next, cdat);
            addStockSheetListSL(buf, &stck);

            return checkStockSheetListCntLoop(num, cont, ptr, &stck);
        }
    }
    else {
        buf.content->data->cnt  = cdat.cnt;
        buf.content->data->name = allocName();
        strcpy(buf.content->data->name, cdat.name);
        buf.content->next = cntl;

        addStockSheetListLS(&ptr, buf);
        return checkStockSheetListCntLoop(num, cont, ptr, &stck->next);
    }
}


STOCKSHEETLIST* checkStockSheetListCnt(int num, CONTENT cont, STOCKSHEETLIST **list) {
    STOCKSHEETLIST *ptr = NULL;
    return checkStockSheetListCntLoop(num, cont, ptr, list);
}



/*************************************************************/
/* 選択するコンテナのデータ生成関数                          */
/*************************************************************/

SELECTDATALIST* makeSelectDataLoop(char *str, STOCKSHEETLIST **list,
                                   SELECTDATALIST* select, STOCKSHEETLIST *ptr){
    STOCKSHEETLIST *stck = *list;
    if(stck == NULL) return select;

    SELECTDATA slct;
    slct.num = stck->data->num;
    slct.content.name = allocName();
    slct.content.cnt  = stck->data->content->data->cnt;
    strcpy(slct.content.name, stck->data->content->data->name);

    if(stck->data->content->next == NULL) {
        if(checkStockSheetList(stck->data->num, *(stck->data->content->data), &ptr)) {
            slct.flag = TRUE;
            addSelectDataListLS(&select, slct);

            return makeSelectDataLoop(str, &stck->next, select, ptr);
        }
        else {
            slct.flag = FALSE;
            addSelectDataListLS(&select, slct);

            return makeSelectDataLoop(str, &stck->next, select, ptr);
        }
    }
    else {
        slct.flag = FALSE;
        addSelectDataListLS(&select, slct);
```

```
        return makeSelectDataLoop(str, &stck->next, select, ptr);
    }
}

SELECTDATALIST* makeSelectData(char *str, STOCKSHEETLIST **list, STOCKSHEETLIST *ptr) {
    STOCKSHEETLIST* buf = rePosition(str, list);
    return makeSelectDataLoop(str, &buf, NULL, ptr);
}


/************************************************************/
/* 在庫情報を変更する関数                                   */
/************************************************************/

STOCKSHEETLIST* remarkStockSheetListSet(STOCKSHEETLIST** list, STOCKSHEETLIST** ptr) {
    STOCKSHEETLIST* resetList = *list;
    STOCKSHEETLIST* resetPtr  = *ptr;

    if (resetList == NULL)  return resetPtr;
    if (resetPtr  == NULL)  return NULL;

    STOCKSHEETLIST* buf = checkStockSheetListCnt(resetList->data->num,
                                        *(resetList->data->content->data), &resetPtr);

    return remarkStockSheetListSet(&resetList->next, &buf);
}


STOCKSHEETLIST* remarkStockSheetList(char *str, STOCKSHEETLIST** list, STOCKSHEETLIST** ptr) {
    STOCKSHEETLIST* resetList = rePosition(str, list);
    STOCKSHEETLIST* resetPtr  = copyStockSheetList(ptr);

    resetPtr = rePosition(str, &resetPtr);

    return remarkStockSheetListSet(&resetList, &resetPtr);
}


/************************************************************/
/* 出庫指示票作成関数                                       */
/************************************************************/

INSTRUCTIONSHEET makeInstructionSheet(int num, REQUESTSHEET request,
                                    STOCKSHEETLIST** list, STOCKSHEETLIST** ptr) {
    INSTRUCTIONSHEET inst;

    inst.num = num;
    inst.sendto = allocSendto();
    strcpy(inst.sendto, request.sendto);

    STOCKSHEETLIST* stck = *ptr;
    inst.select = makeSelectData(request.name, list, stck);

    return inst;
}



#endif
```

C:\Documents and Settings\Sakamoto\My Documents\univercity\MainTheme\Program\StockControl-05\Execution.h

```
/************************************************************/
/* 実行用関数                                               */
/************************************************************/


#ifndef __EXECUTION_H_
#define __EXECUTION_H_

#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#include "CommonExe.h"
#include "DefineData.h"
#include "RequestExe.h"


/************************************************************/
/* プロトタイプ関数                                         */
/************************************************************/

EXECUTIONDATA runSend           (REQUESTSHEET request, EXECUTIONDATA exedata);
EXECUTIONDATA sendFirstStep     (REQUESTSHEET request, EXECUTIONDATA exedata);
EXECUTIONDATA sendSecondStep    (REQUESTSHEET request, STOCKSHEETLIST** stock,
                                                       EXECUTIONDATA exedata);
EXECUTIONDATA reseveStep        (REQUESTSHEET request, EXECUTIONDATA exedata);
EXECUTIONDATA sendStep          (REQUESTSHEET request, STOCKSHEETLIST** stock,
                                                       EXECUTIONDATA exedata);

EXECUTIONDATA exeSendStep(int num, STOCKSHEETLIST** stock,
                          REQUESTSHEET request, EXECUTIONDATA exedata);

REQUESTSHEETLIST* exeReserveStep(REQUESTSHEET request, REQUESTSHEETLIST **list);
BOOLEAN exeContactCustomer(char *str);

EXECUTIONDATA runArrive(STOCKSHEET stock, EXECUTIONDATA exedata);
EXECUTIONDATA arriveFirstStep(STOCKSHEET stock, EXECUTIONDATA exedata,
                                                REQUESTSHEETLIST *rlist);
EXECUTIONDATA arriveSecondStep(STOCKSHEET stock, EXECUTIONDATA exedata,
                               REQUESTSHEETLIST *rlist, STOCKSHEETLIST **slist);
EXECUTIONDATA arriveThirdStep(STOCKSHEET stock, EXECUTIONDATA exedata,
                              REQUESTSHEETLIST *rlist, STOCKSHEETLIST **slist);
EXECUTIONDATA arriveFourthStep(STOCKSHEET stock, EXECUTIONDATA exedata,
                                                 REQUESTSHEETLIST *rlist);



/************************************************************/
/* 出庫依頼受理関数                                         */
/************************************************************/

EXECUTIONDATA runSend(REQUESTSHEET request, EXECUTIONDATA exedata) {
    return sendFirstStep(request, exedata);
}

EXECUTIONDATA sendFirstStep(REQUESTSHEET request, EXECUTIONDATA exedata) {

    STOCKSHEETLIST *slist = copyStockSheetList(&exedata.stock);

    //printf("send first step\n");
    if (checkProduct(request.name)) {
        //printf("go to second step\n");
        return sendSecondStep(request, searchStockRL(request, &slist), exedata);
    }
    else return exedata;
}


EXECUTIONDATA sendSecondStep(REQUESTSHEET request, STOCKSHEETLIST** stock,
                                                   EXECUTIONDATA exedata) {

    //printf("send second step\n");

    if(request.cnt <= sumOfStock(request.name, stock)) {
        //printf("go to sendStep\n");
        return sendStep(request, stock, exedata);
    }
    else {
        //printf("go to reserveStep\n");
        return reseveStep(request, exedata);
    }
}


EXECUTIONDATA reseveStep(REQUESTSHEET request, EXECUTIONDATA exedata) {

    REQUESTSHEETLIST* rlist = exedata.request;

    exedata.request = exeReserveStep(request, &rlist);

    return exedata;
}



EXECUTIONDATA sendStep(REQUESTSHEET request, STOCKSHEETLIST** stock,
                                             EXECUTIONDATA exedata) {
    //printf("sendStep\n");

    if (exedata.inst == NULL) {
```

```
        //printf("format sendStep\n");
        STOCKSHEETLIST* slist = selectContainerRL(request, stock);
        return exeSendStep(10000, &slist, request, exedata);
    }
    else {
        //printf("used sendStep\n");
        int num = getLastInstructionNumber(&exedata.inst);
        STOCKSHEETLIST* slist = selectContainerRL(request, stock);
        return exeSendStep(num, &slist, request, exedata);
    }
}

EXECUTIONDATA exeSendStep(int num, STOCKSHEETLIST** stock,
                          REQUESTSHEET request, EXECUTIONDATA exedata) {

    STOCKSHEETLIST *stck  = *stock;
    STOCKSHEETLIST *slist = exedata.stock;

    INSTRUCTIONSHEET idat;

    EXECUTIONDATA exe = {NULL, NULL, NULL};

//    printf("exe SendStep\n");

    exe.stock   = remarkStockSheetList(request.name, &stck, &slist);
    exe.request = exedata.request;

    idat = makeInstructionSheet(num, request, &stck, &slist);
    addInstructionSheetListLI(&exe.inst, idat);

    return exe;
}

REQUESTSHEETLIST* exeReserveStep(REQUESTSHEET request, REQUESTSHEETLIST **list) {

    REQUESTSHEETLIST *rlist = *list;

    if(exeContactCustomer(request.sendto)) {
        addRequestSheetListLR(&rlist, request);
        return rlist;
    }
    else return NULL;
}


BOOLEAN exeContactCustomer(char *str) {
    return TRUE;
}


/*************************************************************/
/* 積荷票受理関数                                            */
/*************************************************************/

EXECUTIONDATA runArrive(STOCKSHEET stock, EXECUTIONDATA exedata) {

    //printf("run Arrive\n");

    if(stock.content->data == NULL) {
        //printf("Do nothing\n");
        return exedata;
    }
    else {
        //printf("Go to First Step\n");
        return arriveFirstStep(stock, exedata, NULL);
    }
}


EXECUTIONDATA arriveFirstStep(STOCKSHEET stock, EXECUTIONDATA exedata,
                                                REQUESTSHEETLIST *rlist) {
    if(stock.content->data == NULL) {
        addRequestSheetListLL(&rlist, &exedata.request);
        exedata.request = rlist;
        return exedata;
    }

    if(exedata.request == NULL) {
        addStockSheetListLS(&exedata.stock, stock);
        addRequestSheetListLL(&exedata.request, &rlist);
        return exedata;
    }

    REQUESTSHEET request = *(exedata.request->data);

    if (checkRequestSR(stock, request)) {
        STOCKSHEETLIST *slist = copyStockSheetList(&exedata.stock);
        addStockSheetListLS(&slist, stock);
        return arriveSecondStep(stock, exedata, rlist, searchStockRL(request, &slist));
    }
    else {
        //printf("Go to First Step from First Step\n");
        addRequestSheetListLR(&rlist, request);
        exedata.request = exedata.request->next;

        return arriveFirstStep(stock, exedata, rlist);
    }
}
```

C:\Documents and Settings\Sakamoto\My Documents\univercity\MainTheme\Program\StockControl-05\Execution.h

```
EXECUTIONDATA arriveSecondStep(STOCKSHEET stock, EXECUTIONDATA exedata,
                               REQUESTSHEETLIST *rlist, STOCKSHEETLIST **slist) {

    REQUESTSHEET request = *(exedata.request->data);

    if(request.cnt <= sumOfStock(request.name, slist)) {
        // printf("Go to Thied Step from Second Step\n");
        return arriveThirdStep(stock, exedata, rlist, slist);
    }
    else {
        //printf("Go to First Step from Second Step\n");
        addRequestSheetListLR(&rlist, request);
        exedata.request = exedata.request->next;
        return arriveFirstStep(stock, exedata, rlist);
    }
}


EXECUTIONDATA arriveThirdStep(STOCKSHEET stock, EXECUTIONDATA exedata,
                              REQUESTSHEETLIST *rlist, STOCKSHEETLIST **slist) {

    REQUESTSHEET request = *(exedata.request->data);

    addStockSheetListLS(&exedata.stock, stock);
    exedata.request = exedata.request->next;


    EXECUTIONDATA exe = sendStep(request, slist, exedata);

    //printf("Go to Fourth Step\n");
    return arriveFourthStep(stock, exe, rlist);
}


EXECUTIONDATA arriveFourthStep(STOCKSHEET stock, EXECUTIONDATA exedata,
                                                 REQUESTSHEETLIST *rlist) {
    STOCKSHEET stockdata;

    stockdata = searchNumber(stock, &exedata.stock);

    //printf("Four\n");
    //printf("%d\n", stockdata.num);
    //printContentList(&stockdata.content);

    remStockSheetListSL(stock, &exedata.stock);

    return arriveFirstStep(stockdata, exedata, rlist);
}


#endif
```