酒屋在庫管理システム

グループ名：A チーム

メンバー　：大田和樹・徳富秀輔

目次

1. 酒屋システムソースコード一式

liquorSystem.h

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

#define ticketDataFile "liquorTicketData.txt"
#define stockDataFile "liquorStockData.txt"

int orderNumberBase;

typedef struct{
   char *liquorName;
   int    numberOfLiquor;
}liquor;

typedef struct{
   char *consumerName;
   char *liquorName;
   int    numberOfLiquor;
}shipmentOrder;

typedef struct{
   char *consumerName;
   char *liquorName;
   int    numberOfLiquor;
   int    orderNumber;
}ticket;
```

```
typedef struct{
   char *liquorName;
   int    numberOfLiquor;
}arriveOrder;
```

main.c

```
#include "liquorSystem.h"

void liquorSystem();

int main(void){

   liquorSystem();

   return 0;

}
```

liquorSystem.c

```
#include "liquorSystem.h"

int getNumberOfStock();
int getNumberOfTicket();
void readStock(liquor *stock);
void readTicketList(ticket *ticketList);
void order(liquor *stock,ticket *ticketList,int numberOfTicket);
void writeStock(liquor *stock);
void writeTicketList(ticket *ticketList);
//void testStockAndTicketList(liquor *stock,ticket *ticketList);

void liquorSystem(){
   orderNumberBase = 0;
   liquor *stock;
   ticket *ticketList;
   int numberOfTicket = 1;
```

```
    int numberOfStock    = 1;

    numberOfStock = getNumberOfStock();
    numberOfTicket = getNumberOfTicket();

    //各データ数を元に領域を確保
    stock = (liquor *)malloc((numberOfStock+2) * sizeof(liquor));
    ticketList = (ticket *)malloc((numberOfTicket+2) * sizeof(ticket));

    readStock(stock);

    readTicketList(ticketList);

    //仮のデータを用意する
    //testStockAndTicketList(stock,ticketList);

    //orderNumberBase を作成。いいやり方が思いつかない。
    orderNumberBase += numberOfTicket;

    order(stock,ticketList,numberOfTicket);

    writeStock(stock);

    writeTicketList(ticketList);

    free(stock);
    free(ticketList);

}

/*void testStockAndTicketList(liquor *stock,ticket *ticketList){
    liquor testLiquor;
    liquor nullLiquor = {NULL,0};
    ticket testTicket;
    ticket nullTicket = {NULL,NULL,0,0};
    char testName[] = "test";
```

```
  int testNum     = 5;
  int i;

  testLiquor.liquorName = (char *)malloc(4 * sizeof(char));
  testLiquor.numberOfLiquor = testNum;
  for(i = 0;i < 4;i++){
    testLiquor.liquorName[i] = testName[i];
  }

  stock[0].liquorName = testLiquor.liquorName;
  stock[0].numberOfLiquor = testLiquor.numberOfLiquor;
  stock[1] = nullLiquor;

  testTicket.consumerName = (char *)malloc(4 * sizeof(char));
  testTicket.liquorName = (char *)malloc(4 * sizeof(char));
  testTicket.numberOfLiquor = 3;
  testTicket.orderNumber      = 441;

  for(i = 0;i < 4;i++){
    testTicket.consumerName[i] = testName[i];
    testTicket.liquorName[i] = testName[i];
  }

  ticketList[0] = testTicket;
  ticketList[1] = nullTicket;

}*/

/*int main(void){
  liquor *stock;
  ticket *ticketList;



  testStockAndTicketList(stock,ticketList);

  return 0;
```

```
}*/
```

order.c

```
#include "liquorSystem.h"

int input_order();
int check_order(int i);
void finish_message();
ticket shipment(liquor *stock);
void arrive(liquor *stock);
void stockdisp(liquor *stock);
void recorddisp(ticket *ticketList);

void order(liquor *stock, ticket *ticketList, int numberOfTicket){
  // 入力画面
  int inputOrder = input_order();

  if(inputOrder == 1){
    // 出荷依頼処理
    ticket ticket;
    do{
      ticket = shipment(stock);
    } while(ticket.consumerName == NULL);
    ticketList[numberOfTicket] = ticket;
    finish_message();
    return;
  }
  else if(inputOrder == 2){
    // 入荷依頼処理
    arrive(stock);
    finish_message();
    return;
  }

  else if(inputOrder == 3){
```

```
    // 在庫表示処理
    stockdisp(stock);
    finish_message();
    return;
  }
  else if(inputOrder == 4){
    // 出荷実績表示処理
    recorddisp(ticketList);
    finish_message();
    return;
  }
}

int input_order(void){
  int i;
  do{
    printf("プラズマ酒店¥n");
    printf("以下から行いたい項目を選び、1~4 のいずれかを入力してください。¥n");
    printf("1： 出荷依頼¥n");
    printf("2： 入荷依頼¥n");
    printf("3： 在庫表示¥n");
    printf("4： 出荷実績表示¥n");
    printf("[入力してください：]");
    scanf("%d", &i);
  } while(check_order(i));
  return i;
}

int check_order(int i){
  if(i >= 1 & i <= 4){
    return 0;
  }
  else{
    return 1;
  }
}
```

```
void finish_message(){
   printf("システムを終了します。¥n");
}
```

shipment.c

```
#include "liquorSystem.h"

//入力を受け付ける
void inputShipmentOrder(shipmentOrder *order);
//注文番号を取得
int              getOrderNumber();
//在庫を確認し出荷を行う 返り値は 1or0 条件判定に用いる 1 = error
int              doShipment(liquor *stock,shipmentOrder order);
//出荷伝票を発行する
ticket          printTicket(char *cName,char *lName,int nOfL,int oN);

ticket shipment(liquor *stock){
   shipmentOrder *thisOrder;
   int                thisOrderNumber;
   ticket             thisTicket;

   thisOrder = (shipmentOrder *)malloc(sizeof(shipmentOrder));
   inputShipmentOrder(thisOrder);

   thisOrderNumber    = getOrderNumber();

   if(doShipment(stock,*thisOrder)){
      //出荷処理に失敗した場合は空の出荷伝票を作成する
      thisTicket = printTicket(NULL,NULL,0,0);
      return thisTicket;
   }

   thisTicket                =                printTicket(thisOrder->consumerName,thisOrder-
>liquorName,thisOrder->numberOfLiquor,thisOrderNumber);
```

```
    return thisTicket;

}

void inputShipmentOrder(shipmentOrder *order){
    //本数
    int number;
    //index
    int i,j;
    //入力のための文字配列
    char str[1000];

    printf("・出荷依頼を行います.お客様の名前、欲しいお酒の銘柄、本数を入力してください.¥n");

    //お客様の名前の入力
    printf("[名前:]");
    scanf("%s",str);

    //入力の文字数を調べる
    i = 0;
    while(str[i] != '¥0'){
        i++;
    }

    order->consumerName = (char *)malloc(i * sizeof(char));

    //値の代入
    for(j = 0;j <= i;j++){
        order->consumerName[j] = str[j];
    }

    //銘柄の入力
    printf("[お酒の銘柄:]");
    scanf("%s",str);
```

```
    //入力の文字数を調べる
    i = 0;
    while(str[i] != '¥0'){
        i++;
    }

    order->liquorName = (char *)malloc(i * sizeof(char));

    //値の代入
    for(j = 0;j <= i;j++){
        order->liquorName[j] = str[j];
    }

    printf("[本数:]");
    scanf("%d",&number);

    order->numberOfLiquor = number;

    return;

}

int getOrderNumber(){
    //注文番号
    orderNumberBase++;

    return orderNumberBase;
}

int doShipment(liquor *stock,shipmentOrder order){
    int i = 0;
    char *name;

    while(stock[i].liquorName != NULL){
        //printf("%s¥n%s¥n",stock[i].liquorName,order.liquorName);
        if(strcmp(stock[i].liquorName,order.liquorName) == 0){
```

```
        //名前がある場合
        if(stock[i].numberOfLiquor >= order.numberOfLiquor){
          //在庫が十分な場合
          printf("在庫が確認できました。出荷を行います。¥n");
          stock[i].numberOfLiquor -= order.numberOfLiquor;
                return 0;
        }
      }
      i++;
  }

  //在庫が確認できなかった場合は、一旦 NULL の出荷伝票を作成して order に返す
  printf("在庫が確認できませんでした.お手数ですがもう一度初めからやり直してください。¥n");
  return 1;
}

ticket printTicket(char *cName,char *lName,int nOfL,int oN){
  ticket thisTicket = {cName,lName,nOfL,oN};

  if(nOfL > 0){
    printf("--出荷伝票--¥n");
    printf("[注文番号]%d¥n",oN);
    printf("[名前]%s¥n",cName);
    printf("[お酒の銘柄]%s¥n",lName);
    printf("[本数]%d 本¥n",nOfL);
    printf("------¥n");
    printf("出荷が完了しました。¥n");
  }

  return thisTicket;

}
```

arrive.c

```
#include "liquorSystem.h"
```

```
void inputArriveOrder(arriveOrder *order);
void addOrderToStock(liquor *stock, arriveOrder arriveOrder);

void arrive(liquor *stock){
   arriveOrder *thisOrder;
   int thisOrderNumber;

   thisOrder = (arriveOrder *)malloc(sizeof(arriveOrder));
   inputArriveOrder(thisOrder);
   addOrderToStock(stock, *thisOrder);
   printf("入荷が完了しました¥n");
}

void inputArriveOrder(arriveOrder *order){
   //  本数
   int number;
   // index
   int i,j;
   //  入力用文字配列
   char str[1000];

   printf("入荷を受け付けます。お酒の銘柄、本数を入力してください。¥n");

   //  酒銘柄・本数入力部
   printf("[お酒の銘柄:]");
   scanf("%s", str);

   //入力の文字数を調べる
   i = 0;
   while(str[i] != '¥0'){
      i++;
   }

   order->liquorName = (char *)malloc(i * sizeof(char));
```

```
    //値の代入
    for(j = 0;j <= i;j++){
        order->liquorName[j] = str[j];
    }

    printf("[お酒の本数:]");
    scanf("%d", &number);

    order->numberOfLiquor = number;

    return;
}

void addOrderToStock(liquor *stock, arriveOrder order){
    int i=0;
    int n=0;
    while(stock[i].liquorName!=NULL){
        if(strcmp(stock[i].liquorName, order.liquorName) == 0){
            stock[i].numberOfLiquor += order.numberOfLiquor;
            n++;
        }
        i++;
    }
    if(n==0){
        stock[i].liquorName = order.liquorName;
        stock[i].numberOfLiquor = order.numberOfLiquor;
    }
    return;
}
```

stockdisp.c

```
#include "liquorSystem.h"

void stockdisp(liquor *stock){
    printf("在庫の表示を行います。¥n");
    printf("--------¥n");
```

```
    int i = 0;
    while(stock[i].liquorName != NULL){
        printf("・%s    ", stock[i].liquorName);
        printf("%d 本¥n", stock[i].numberOfLiquor);
        i++;
    }
    printf("--------¥n");
    return;
}
```

recorddisp.c

```
#include "liquorSystem.h"

void recorddisp(ticket *ticketList){
    char inputName[1000];
    printf("出荷実績を表示します。");
    printf("顧客名を入力してください。¥n");
    scanf("%s", inputName);

    // ticketList 内確認用
    int i = 0;
    // 存在しない場合の確認用
    int n = 0;

    do{
        if(strcmp(ticketList[i].consumerName, inputName) == 0){
            printf("--出荷伝票--¥n");
            printf("[注文番号]%d¥n", ticketList[i].orderNumber);
            printf("[名前]%s¥n", ticketList[i].consumerName);
            printf("[お酒の銘柄]%s¥n", ticketList[i].liquorName);
            printf("[本数]%d 本¥n", ticketList[i].numberOfLiquor);
            printf("--------¥n");
            n++;
        }
        i++;
    } while(ticketList[i].consumerName!=NULL);
```

```
  if(n==0){
    printf("!!この名前は存在しません。もう一度入力し直してください。!!¥n¥n");
    recorddisp(ticketList);
  }
  else{
    return;
  }
}
```

2. テストレポート

1　単体テスト

C1 網羅の条件でホワイトボックステストを行った。各モジュールでの結果を以下に示す。

1.1　[ORDER]テスト

1.1.1 input_order

ユーザに 4 つの処理を選択させる出力が行われることを確認した。
また、1-4 以外の入力が行われた時、もう一度処理選択画面が表示されることを確認した。

1.1.2 check_order

1-4 の数字以外が入力された時、1 を返すこと、正常な入力の時 0 を返すことをコマンドライン上で出力を行い確認した。

1.2　[SHIPMENT]テスト

1.2.1 inputShipmentOrder

入力を行い、正常に機能しているかをコマンドラインの出力により確認する。
以下に例を示す。

```
------------------------------
・出荷依頼を行います.お客様の名前、欲しいお酒の銘柄、本数を入力してください.
[名前:]大田和樹
[お酒の銘柄:]鬼殺し
[本数:]100
```

大田和樹,鬼殺し,100

1.2.2 getOrderNumber

orderNumberBase と返り値が正常に動作しているかを確かめた。

1.2.3 doShipment

３通りについて正常に動作することを確認した。

- 指定の名前の酒が在庫にあり、かつ個数も十分な場合
- 指定の名前の酒が在庫にあるが、個数が十分でない場合
- 指定の名前の酒が在庫にない場合

1.2.4 printTicket

2 通りの場合において確認した

- パラメータを正確に指定した場合 例：("oota","test",3,7)
  正常に動作した。出力の様子をいかに示す。

  --出荷伝票—
  [注文番号]7
  [名前]oota
  [お酒の銘柄]test
  [本数]3 本

  出荷が完了しました。

- パラメータを(NULL,NULL,0,0)とした場合
  正常に動作した。出力は行われなかった。

1.3 [ARRIVE]テスト

1.3.1 inputArriveOrder

入荷依頼時の入力部。コマンドラインにて正常な出力が行われかつ正常に入力が可能かどうかを確認した。

入荷を受け付けます。お酒の銘柄、本数を入力してください。
[お酒の銘柄:]角
[お酒の本数:]5

1.3.2 addOrderToStock

入荷依頼時の酒在庫更新部。在庫が存在するときに入力された数だけ在庫が増えること、存在しない時新たに追加されることを確認した。どちらの場合も試行し、正常であると判断した。

1.4 [STOCKDISP]テスト

1.4.1 stockdisp

存在する全ての在庫が問題なく出力されることを確認した。

1.5 [RECORDDISP]テスト

1.5.1 recorddisp

ユーザからの顧客名の入力を受け出荷実績に

- ある時
  該当する顧客名の全ての出荷伝票の表示
- ない時
  再度の入力を促す

以上の真偽 2 パターンについてコマンドライン上で正常に動作することを確認した。

1.6 [RW]テスト

1.6.1 テストに用いたデータ

- liquorStockData.txt
  test 68
  ayo 51
  dorya 2
  tori 2
- liquorTicketData.txt
  ota sak 2 3562
  tokutomi yei 200 5543

1.6.2 readStock, readTicketList

正常に読み込まれていることをコマンドラインにて確認した。

1.6.3 writeStock, writeTicketList

正常に書き込まれていることをファイルを開いて確認した。

2　結合テスト

1 単体テストでそれぞれのモジュールが正常に動作することを確認した。その後、結合テストとして、小規模なソフトウェアであることを踏まえビックバン方式を採用し、ブラックボックステストを実施した。
プログラムの出力が外部設計での想定される出力と一致していることを確認した。
以下にテストの実施内容を示す。

2.1　全体の動作のテスト

プログラム全体の動作をテストする。
なお、このプログラムにおいては、一度の処理ごとにプログラムが終了する。

テストはコマンドラインの表示による確認を行う。
以下の通り出荷処理が正常に行われていることが確認できた。

テストに用いるデータ

---

liquorStockData.txt

test 68
ayo 49
dorya 2
tori 2

---

liquorTicketData.txt

ota sake 2 3562
tokutomi yei 200 5543

・在庫を表示する

プラズマ酒店

以下から行いたい項目を選び、1~4 のいずれかを入力してください。

1 : 出荷依頼

2 : 入荷依頼

3 : 在庫表示

4 : 出荷実績表示

[入力してください：]3

在庫の表示を行います。

¥--------

・test　68 本

・ayo　49 本

・dorya　2 本

・tori　2 本

¥--------

・出荷実績を表示する

プラズマ酒店

以下から行いたい項目を選び、1~4 のいずれかを入力してください。

1 : 出荷依頼

2 : 入荷依頼

3 : 在庫表示

4 : 出荷実績表示

[入力してください：]4

出荷実績を表示します。顧客名を入力してください。

ota

--出荷伝票--

[注文番号]3562

[名前]ota

[お酒の銘柄]sake

[本数]2 本

¥--------

システムを終了します。

・入荷を行う

----

oodawajunoMacBook-Air:program kazukiota$ ./test

プラズマ酒店

以下から行いたい項目を選び、1~4 のいずれかを入力してください。

1 : 出荷依頼

2 : 入荷依頼

3 : 在庫表示

4 : 出荷実績表示

[入力してください：]3

在庫の表示を行います。

¥--------

・test　76 本

・ayo　50 本

・wei　2 本

・tori　2 本

¥--------

システムを終了します。

oodawajunoMacBook-Air:program kazukiota$ ./test

プラズマ酒店

以下から行いたい項目を選び、1~4 のいずれかを入力してください。

1 : 出荷依頼

2 : 入荷依頼

3 : 在庫表示

4 : 出荷実績表示

[入力してください：]2

入荷を受け付けます。お酒の銘柄、本数を入力してください。

[お酒の銘柄:]ike

[お酒の本数:]4

入荷が完了しました

システムを終了します。

oodawajunoMacBook-Air:program kazukiota$ ./test

プラズマ酒店

以下から行いたい項目を選び、1~4 のいずれかを入力してください。

1 : 出荷依頼

2 : 入荷依頼

3 : 在庫表示

4 : 出荷実績表示

[入力してください：]3

在庫の表示を行います。

¥--------

・test　76 本

・ayo　50 本

・wei　2 本

・tori　2 本

・ike　4 本

¥--------

システムを終了します。

入荷が正常に行われていることが確認できる。

・出荷を行う

----

oodawajunoMacBook-Air:program kazukiota$ ./test

プラズマ酒店

以下から行いたい項目を選び、1~4 のいずれかを入力してください。

1 : 出荷依頼

2 : 入荷依頼

3 : 在庫表示

4 : 出荷実績表示

[入力してください：]3

在庫の表示を行います。

¥--------

・test　47 本

・ayo　50 本

・wei　2 本

・tori　2 本

・ike　4 本

¥--------

システムを終了します。

oodawajunoMacBook-Air:program kazukiota$ ./test

プラズマ酒店

以下から行いたい項目を選び、1~4 のいずれかを入力してください。

1 : 出荷依頼

2 : 入荷依頼

3 : 在庫表示

4 : 出荷実績表示

[入力してください：]4

出荷実績を表示します。顧客名を入力してください。

ota

--出荷伝票--

[注文番号]3562

[名前]ota

[お酒の銘柄]sake

[本数]2 本

¥--------

システムを終了します。

oodawajunoMacBook-Air:program kazukiota$ ./test

プラズマ酒店

以下から行いたい項目を選び、1~4 のいずれかを入力してください。

1 : 出荷依頼

2 : 入荷依頼

3 : 在庫表示

4 : 出荷実績表示

[入力してください：]1

・出荷依頼を行います.お客様の名前、欲しいお酒の銘柄、本数を入力してください.

[名前:]ota

[お酒の銘柄:]test

[本数:]10

在庫が確認できました。出荷を行います。

--出荷伝票--

[注文番号]3

[名前]ota

[お酒の銘柄]test

[本数]10 本

¥------

出荷が完了しました。

システムを終了します。

oodawajunoMacBook-Air:program kazukiota$ ./test

プラズマ酒店

以下から行いたい項目を選び、1~4 のいずれかを入力してください。

1 : 出荷依頼

2 : 入荷依頼

3 : 在庫表示

4 : 出荷実績表示

[入力してください：]3

在庫の表示を行います。

¥--------

・test　37 本

・ayo　50 本

・wei　2 本

・tori　2 本

・ike　4 本

¥--------

システムを終了します。

oodawajunoMacBook-Air:program kazukiota$ ./test

プラズマ酒店

以下から行いたい項目を選び、1~4 のいずれかを入力してください。

1 : 出荷依頼

2 : 入荷依頼

3 : 在庫表示

4 : 出荷実績表示

[入力してください：]4

出荷実績を表示します。顧客名を入力してください。

ota

--出荷伝票--

[注文番号]3562

[名前]ota

[お酒の銘柄]sake

[本数]2 本

¥--------

--出荷伝票--

[注文番号]3

[名前]ota

[お酒の銘柄]test

[本数]10 本

¥--------

システムを終了します。

-----

3. その他コンパイル実行に必要なドキュメントなど

liquorStockData.txt
　倉庫の在庫状況の保持
liquorTicketData.txt
　出荷伝票控えの保持

4. 上流工程評価アンケート

このアンケートはソフトウェア工学特論 I のミニ酒屋問題における**内部設計と実装演習**の後に記述して提出してください．このアンケートは**グループで一枚**必ず提出してください．また，このグループ対象アンケートは，皆さんの**評価に影響を与えます**ので，正しく記述してください．

グループ名：A チーム　　グループリーダ名：徳富秀輔
要求定義書・外部設計書のグループ名：プラズマ

内部設計および実装では，顧客から与えられた要求にもとづいて作成された要求定義書・外部設計書に忠実であることが第一に求められます．本演習においても，内部設計・実装が**そ**

れまでの工程で作成された要求定義書・外部設計書の内容に準拠しているかどうかが評価の対象となります.

そこで，今回の演習で利用した要求定義書・外部設計書を評価し，内部設計・実装を行う際に問題がなかったかどうかを記述してください．ここで問題がある等の記述が無ければ，要求定義書・外部設計書と内部設計・実装の間に**ずれ**が存在した場合に，それは**実装の際のグループに問題があったもの**と判断します.

(a)要求定義書・外部設計書の内容は理解しやすかったですか？

全く分からない　一部しか分からない　半分程度　　ほぼ理解できた　完全に理解できた

5　　4　　3　　②　　1

(a’) 選択肢が 1 以外である場合，その部位を指摘し，不明だった点をどのように解釈したのかを併せて明記しておくこと．この件に関しては，要求定義書・外部設計書の修正および修正箇所を指摘する資料を用意すること(アンケートとは別にメール等で提出してもよい).

後述 **5. 上流工程評価アンケート資料**を参照

今回の要求定義書は……

(b)正しかったですか？　間違った要件が含まれていませんでしたか？

1. 正しい○

2. 正しくない（もしくは恐らく正しくないと思われる項目が含まれている）

(b’) 選択肢が 1 以外である場合，その部位を指摘し，正しくなかった点をどのように解釈したのかを併せて明記しておくこと．この件に関しては，要求定義書・外部設計書の修正および修正箇所を指摘する資料を用意すること(アンケートとは別にメール等で提出してもよい).

(c)一貫性がありましたか？　複数の要件の間に矛盾はありませんでしたか？

1. 一貫性があった○

2. 一貫性がなかった（もしくはない部分があった）

(c’) 選択肢が 1 以外である場合，その部位を指摘し，一貫性がなかった点をどのように解釈したのかを併せて明記しておくこと．この件に関しては，要求定義書・外部設計書の修正および修正箇所を指摘する資料を用意すること(アンケートとは別にメール等で提出してもよい).

(d)実現可能な事項のみが書かれていますか？

1. すべて実現可能である○

2. 実現不可能である（もしくは実現可能だがコスト的，技術的に非常に困難である要件が含まれている）

(d’) 選択肢が 1 以外である場合，その部位を指摘し，実現不可能な点をどのように解釈したのかを併せて明記しておくこと．この件に関しては，要求定義書・外部設計書の修正および修正箇所を指摘する資料を

用意すること(アンケートとは別にメール等で提出してもよい).

(e)外部設計書の内容は要求定義書と対応が取れていましたか？

対応が全く取れていない　漏れが非常に多い　　一部漏れが残ったまま　完全に過不足なく要求定義書と対応している

4　　3　　2　　①

(e’) 選択肢が 1 以外である場合，その部位を指摘し，対応の取れていなかった点をどのように解釈したのかを併せて明記しておくこと．この件に関しては，要求定義書・外部設計書の修正および修正箇所を指摘する資料を用意すること(アンケートとは別にメール等で提出してもよい).

(f)遷移情報と入出力情報を含めた画面設計は十分にできていますか？

1. 画面遷移も入出力情報も完全である○
2. 画面遷移は記述したが，入出力情報に漏れがある（もしくはその逆）
3. 画面遷移も入出力情報も不十分

(f’) 選択肢が 1 以外である場合，その部位を指摘し，不十分であった点をどのように解釈したのかを併せて明記しておくこと．この件に関しては，要求定義書・外部設計書の修正および修正箇所を指摘する資料を用意すること(アンケートとは別にメール等で提出してもよい).

(g)システム内で利用されるデータ構造について，論理データ設計やコード設計は十分でしたか？

1. 全てのデータ項目の定義とデータ間の関係も含めて十分に設計ができていた
2. データ項目の洗い出しはできていたがデータ間の関係が定義できていなかった
3. データ項目の洗い出しが不十分だった○

(g’) 選択肢が 1 以外である場合，その部位を指摘し，内部設計・実装の際に修正した点を併せて明記しておくこと．この件に関しては，要求定義書・外部設計書の修正および修正箇所を指摘する資料を用意すること(アンケートとは別にメール等で提出してもよい).

後述 **5. 上流工程評価アンケート資料**を参照

5. 上流工程評価アンケート資料

[SHIMPENT3.2]はプログラムのフローが不明。内部設計では一旦関数を終了させ、再度実行することで再入力を実現した。

データ項目、構造体、配列という書き方でなく、データ型を指定する。もしくは、細かいデータにおいては型を指定しない、いずれかにしてほしい。特に、在庫は「配列」と書い

てあるが、「銘柄、本数」と 2 つのパラメータを持ち、単純な配列でない。内部設計では、「liquor」という構造体を作り、「銘柄、本数」をメンバとして持たせることで対応した。

DataBase は、特に指定されていなかったので、2 つのファイルで行うことにした。

データ型に String 型が指定されているものがあるが、c 言語には String はないため、char 型のポインタを用いた。