オブジェクト指向哲学
～入門編～

第2回　配列を使ったプログラム

～データをまとめて扱う方法～

学習目標

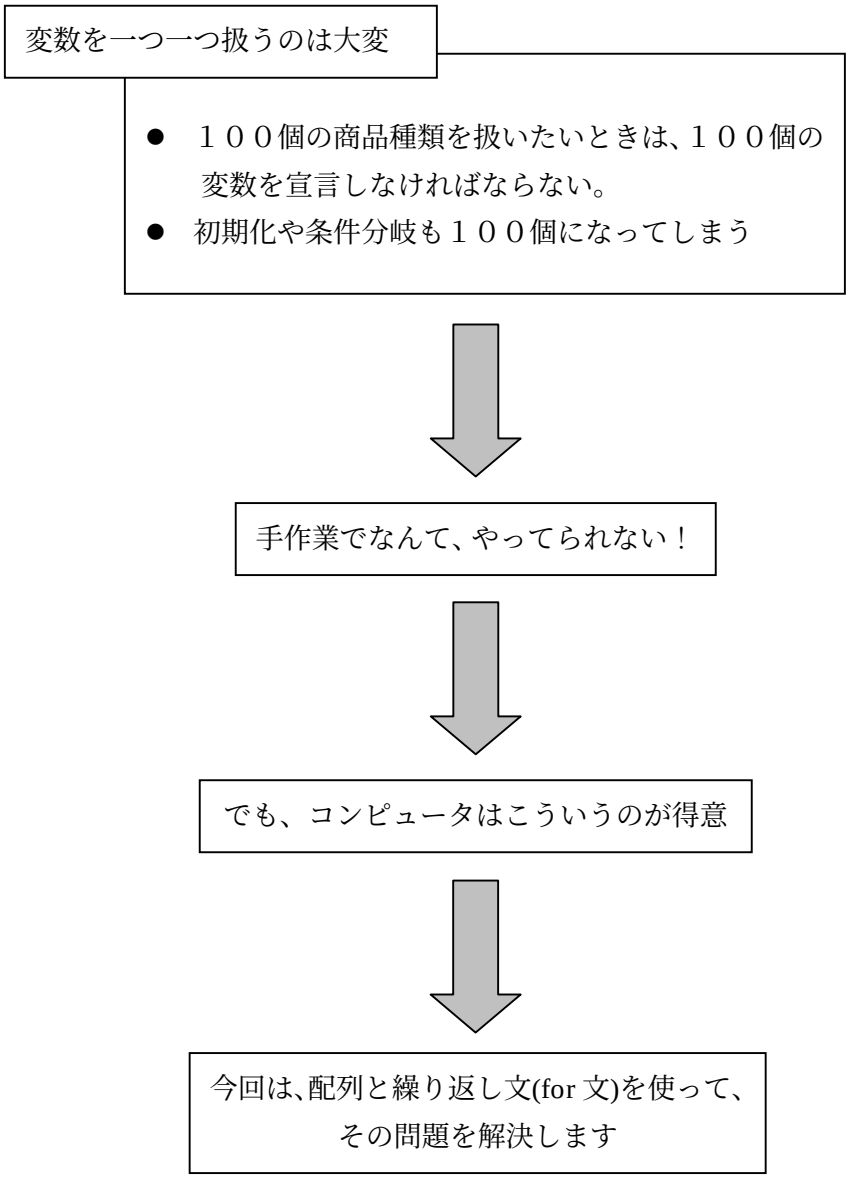
- 配列を使ったプログラムが書ける
  - 配列を使う利点を説明できる
  - Java での配列の使い方を説明できる
  - 配列と繰り返し文を使ったプログラムが書ける
- 情報を管理する簡単なアルゴリズムを考え、実装できる
  - 情報の追加アルゴリズムを考え、実装できる
  - 情報の削除アルゴリズムを考え、実装できる
  - 情報の検索アルゴリズムを考え、実装できる

2.1.　配列と for 文

2.1.1.　コンピュータに得意な仕事をさせる

①.前回のプログラムの問題点

前回のプログラム（例題 1-5）は一番初歩的なプログラムだったため、同じことが何度も書かれていました。



Java Tips　－for 文－

同じ作業を繰り返したい場合には for 文が非常に便利です。例えば、１００円のおつりとして１０円玉を十回出力したい場合（５０円玉、１００円玉がつり銭切れの場合）は、これまでの方法では以下のように書く必要がありました。

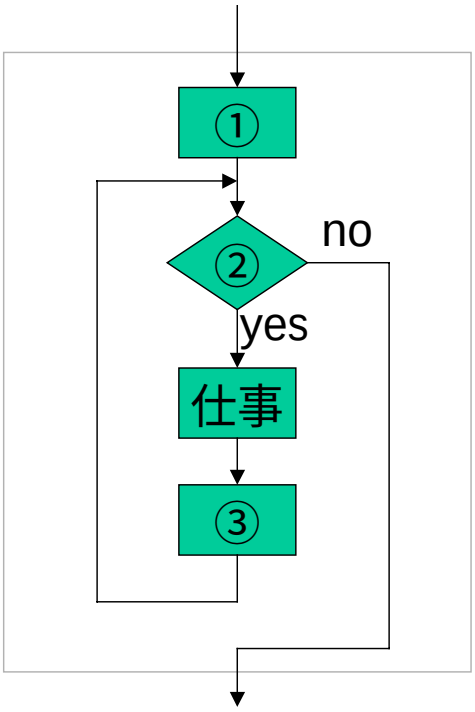
```
System.out.println("１０円が出ました");
System.out.println("１０円が出ました");
System.out.println("１０円が出ました");
System.out.println("１０円が出ました");
System.out.println("１０円が出ました");
System.out.println("１０円が出ました");
System.out.println("１０円が出ました");
System.out.println("１０円が出ました");
System.out.println("１０円が出ました");
System.out.println("１０円が出ました");
```

ところが for 文を使うと、同じ「何回くりかえす」という命令を、以下のように書くことができます。

```
for( int i=0 ; i<10 ; i++ ){
  System.out.println("１０円が出ました");
}
```

for 構文は、以下のような仕組みになります。

```
for( ①  ;  ②  ;  ③  ){
        仕事;
}
```



① 初期処理

上の例の場合、int i = 0;

初期値を設定する。適用は最初だけ。

この場合、for 文開始時に i が 0 に設定される。

②.継続条件

上の例の場合、i < 10;

この条件を満たす限り、処理を継続する。

この場合、i が１０未満の間は処理が続く。

③.繰り返し処理

上の例の場合、i++;

一回の処理が終わるごとに、この処理を実行。

よく使う++は、「＋１」の意。

すなわち、一回毎に i の値が１づつ増えていく。

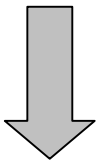
2.1.2.　配列を使う

①.for 文だけでは問題解決しない

しかし、問題だった前日のソースコード(例題 1-5)を for 文を使って初期化することはできるでしょうか？

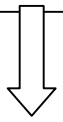
```
//商品種類を保存するための変数を定義する
int itemType01;
int itemType02;
int itemType03;

//商品種類を保存するための変数を初期化する
//何も入っていないことを-1 で表す。
itemType01 = -1;
itemType02 = -1;
itemType03 = -1;
```



```
//商品種類を保存するための変数を定義する
int itemType01;
int itemType02;
int itemType03;

//商品種類を保存するための変数を初期化する
//何も入っていないことを-1 で表す。
for( int i=0 ; i<3 ; i++ ){
        [redacted]  = -1;
}
```



ここに何を書くべきでしょうか？

itemType0 + i ？

②.データをまとめて扱う

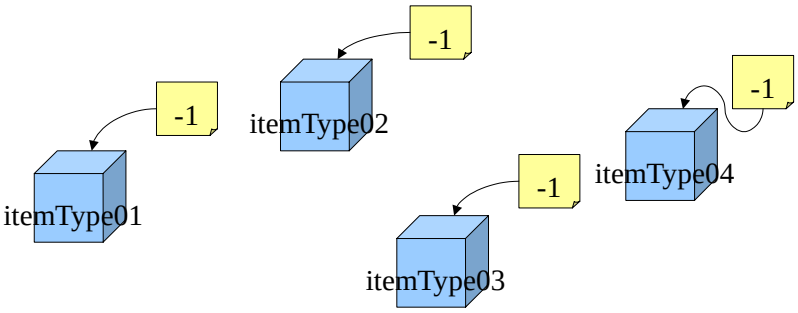
(1)今までは

今までは、itemType01,itemType02 のように「個別の名前」で変数を作っていました。

これだと、01 や 02 のようにほぼ同じ目的に使用する似た変数まで「全部の変数をいちいち宣言しなくてはならない」「その都度、初期化や条件分岐の宣言をしなければならない」という面倒くささがあります。

特に、データが大量になって、変数がたくさん要るときは大変です。

100 個や 200 個、さらには何千何万のデータを処理する場合を考えてみてください。



(2)配列を利用すると

配列を使うと、こういう「似た目的に使う変数」を、ほとんど一括して処理できるようになりとても便利です。

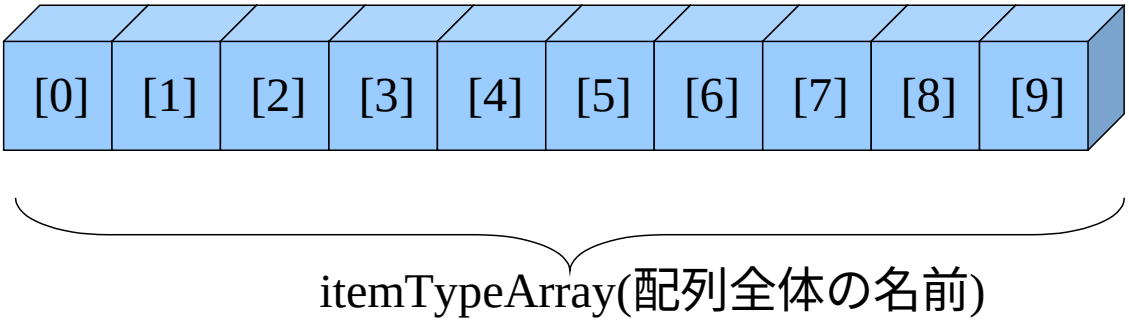
たとえば宣言は　int[] itemTypeArray = new int[10]　と書くだけで、

１０個の配列が一気に作成できます。

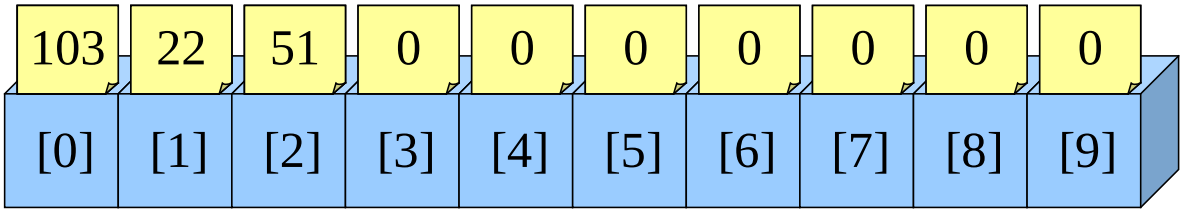
初期化も簡単です。配列は[]内の処理を数字のように扱えるので、

```
for (int i =0; i < 10; i++){
  itemTypeArray [i] = -1;
}
```

と書くだけで、itemTypeArray[0]~itemTypeArray [9]まで、１０個分の初期化が全て完了します。これは配列が何百・何千とあっても同様に処理できます。



③.Java で配列を使う



具体的に、java で配列を作る様子を見てみましょう。

同じ配列でも、C 言語など他の言語とは形式が多少異なります。

(1)配列の作成

大きさが 10 の配列を作る場合、

```
int[] itemTypeArray = new int[10];
```

と書きます。

このように、Java で配列を作るには

```
データ型[] 配列名 = new データ型[個];
```

という形式で宣言して作ります。

たとえば、「 double[] account = new double[2000] 」と書けば、account[0]から account[1999]まで、計２０００個の double 型の配列が生成されます。

なお、C 言語その他と異なり、Java の変数は作成時点で 0 で初期化されています。ラッキーです。それでもあえて上記プログラム群がさらに「-1」で初期化しているのは、「何も入っていないときは－1 で表現する」と今回は決めているからです。

(2)配列へのアクセス

配列の各要素には(ブラケット)“ [] ”でアクセスします。

- 「配列名[数値]」で、それぞれの要素にアクセス（読み書き）できます。

```
itemTypeArray[0] = 103;
itemTypeArray[1] = 22;
itemTypeArray[2] = 51;
```

- さらに、数値の存在を利用して、for 文を使って一括処理もできます。

配列はこうして[]内部を数字として処理できるため、for 文と組み合わせることで繰り返し処理に強みを発揮します。

④.配列を利用したプログラム

こうして for 文と配列を使うと、同じ仕事をするプログラムがちょっと賢くなりました。コードがだいぶ減りました。これなら要素がいくつ増えてもよさそうですね。

例題 2-1：配列を使う(Example2_1.java)

```
1:    /**
2:    * オブジェクト指向哲学　入門編
3:    * 例題 2-1：配列を使う
4:    * 商品種類を追加して表示するプログラム
5:    *
6:    * メインクラス
7:    */
8:    public class Example2_1 {
9:
10:     /**
11:     * メイン
12:     * 商品種類を追加して表示するプログラム
13:     */
14:     public static void main(String[] args) {
15:
16:       //自動販売機プログラムの開始を知らせる
17:       System.out.println("自動販売機が開始しました。");
18:
19:       //商品種類を保存するための配列を定義する
20:       int[] itemTypeArray = new int[10];
21:
22:       //商品種類を保存するための変数を初期化する
23:       for(int i=0;i<10;i++){
24:         itemTypeArray[i] = -1;//何も入っていないことを-1 として扱う
25:       }
26:
27:       //商品種類を追加する
28:       itemTypeArray[0] = 1001;//コーラ
29:       itemTypeArray[1] = 1002;//ソーダ
30:       itemTypeArray[2] = 1003;//お茶
31:
32:       //保存されている商品種類を表示する
33:       for(int i=0;i<10;i++){
34:         if(itemTypeArray[i] != -1){//商品種類が入っている
35:           System.out.println(itemTypeArray[i]+"は販売中です");
36:         }
37:       }
38:     }
39:   }
```

⑤.配列を使う利点

＜議論しよう！＞配列を使う利点

●同じことを何度も書かないという視点から

●他人に分かりやすいプログラムという視点から

2.2.　商品種類の管理

2.2.1.　　商品種類を管理するとは

　今までは商品種類リストといっても表示するだけでしたが、自動販売機には商品種類が変更されたりすることがありますね。
　配列を応用して「管理」に挑戦してみましょう。

(1)商品種類を管理するとは

- 取り扱う商品種類が増えたら、商品種類リストに商品種類を「追加」する。
- その商品種類を取りあつかっているかチェックするために、商品種類リストから、商品種類の商品番号を「探す」。
- その商品種類を取り扱うのをやめたら、商品種類リストから、商品種類を「削除」する。

(2)今回プログラムすることー目的の階層構造を考えてみよう！

-
-
-

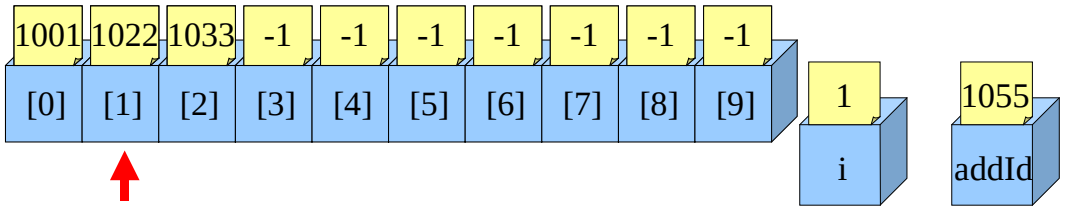
2.2.2.　商品種類の管理プログラム

①.商品種類の追加

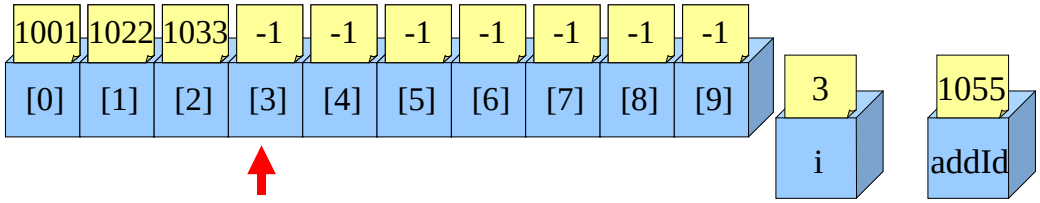
(1)追加の手順を考える

追加するとはプログラムでどういう手順で実現できるでしょう？

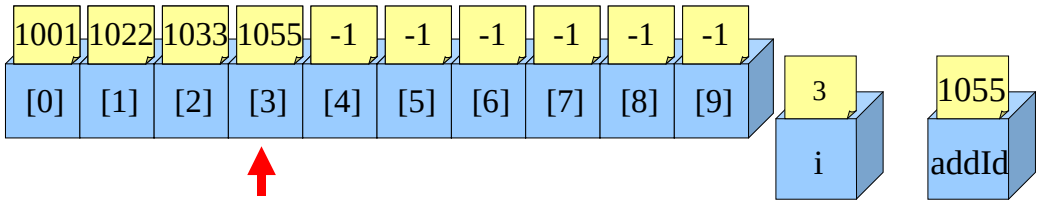
(2)まだ商品番号の入っていない箱を探す。(-1 が入っている箱を探す。)

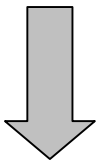


(3)見つけたら、そこに商品番号を書き込む。



(4)これで追加が完了。





手順のことを「アルゴリズム」といいます

(2)追加プログラム

例題 2-2：商品種類を追加する(Example2_2.java)

```
1:      /**
2:      * オブジェクト指向哲学　入門編
3:      * 例題 2-2：商品種類を追加する
4:      * 商品種類を追加するプログラム
5:      *
6:      * メインクラス
7:      */
8:      public class Example2_2 {
9:
10:       /**
11:       * メイン
12:       * 商品種類を追加するプログラム
13:       */
14:       public static void main(String[] args) {
15:
16:         //自動販売機プログラムの開始を知らせる
17:         System.out.println("自動販売機が開始しました。");
18:
19:         //商品種類を保存するための配列を定義する
20:         int[] itemTypeArray = new int[10];
21:
22:         //商品種類を保存するための変数を初期化する
23:         for(int i=0;i<10;i++){
24:           itemTypeArray[i] = -1;//何も入っていないことを-1 として扱う
25:         }
26:
27:         //商品種類を追加する
28:         int addId = 1001;//コーラを追加する
29:         for(int i=0;i<10;i++){//商品種類が入っていない箱を探す
30:           if(itemTypeArray[i] == -1){//入っていない
31:             itemTypeArray[i] = addId;//書き込む
32:             break;
33:           }
34:         }
35:       }
36:     }
```

②.商品種類の検索

(1)検索のアルゴリズム

＜考えよう！＞検索のアルゴリズムを考えてみよう

今回の検索は、見つかったときには「見つかりました」と表示し、見つからなかったときに「見つかりませんでした」と表示することとします。

日本語で記述してみよう！

(2)検索プログラム

例題 2-3：商品種類を検索する(Example2_3.java)

```
1:      /**
2:      * オブジェクト指向哲学　入門編
3:      * 例題 2-3：商品種類を検索する
4:      * 商品種類を検索するプログラム
5:      *
6:      * メインクラス
7:      */
8:      public class Example2_3 {
9:
10:       /**
11:       * メイン
12:       * 商品種類を検索するプログラム
13:       */
14:       public static void main(String[] args) {
15:
16:         //自動販売機プログラムの開始を知らせる
17:         System.out.println("自動販売機が開始しました。");
18:
19:         //商品種類を保存するための配列を定義する
20:         int[] itemTypeArray = new int[10];
21:
22:         //商品種類を保存するための変数を初期化する
23:         for(int i=0;i<10;i++){
24:           itemTypeArray[i] = -1;//何も入っていないことを-1 として扱う
25:         }
26:
27:         //検索の準備として商品種類を追加する
28:         int addId = 1001;//コーラを追加する
29:         for(int i=0;i<10;i++){//商品種類が入っていない箱を探す
30:           if(itemTypeArray[i] == -1){//入っていない
31:             itemTypeArray[i] = addId;//書き込む
32:             break;
33:           }
34:         }
35:         addId = 1002;//ソーダを追加する
36:         for(int i=0;i<10;i++){//商品種類が入っていない箱を探す
37:           if(itemTypeArray[i] == -1){//入っていない
38:             itemTypeArray[i] = addId;//書き込む
39:             break;
40:           }
41:         }
42:
43:         //商品種類を検索する
44:         int searchId = 1002;//ソーダを検索する
45:         int i;//配列を辿った回数を保存する
46:         for(i=0;i<10;i++){
```

```
47:           if(itemTypeArray[i] == searchId){//見つかった
48:             break;
49:           }
50:         }
51:         if(i == 10){//最後まで辿ったが見つからなかった
52:           System.out.println("見つかりませんでした");
53:         }else{
54:           System.out.println("見つかりました");
55:         }
56:
57:         searchId = 1004;//DD レモンを検索する
58:         for(i=0;i<10;i++){
59:           if(itemTypeArray[i] == searchId){//見つかった
60:             break;
61:           }
62:         }
63:         if(i == 10){//最後まで辿ったが見つからなかった
64:           System.out.println("見つかりませんでした");
65:         }else{
66:           System.out.println("見つかりました");
67:         }
68:       }
69:     }
```

(3)別のアルゴリズムでも検索できる

例題 2-3 とは異なるアルゴリズムで書かれた例題 2-4 も紹介します。

例題 2-4：異なる方法の検索アルゴリズム(Example2_4.java)

```
1:      /**
2:      * オブジェクト指向哲学　入門編
3:      * 例題 2-4：異なる方法の検索アルゴリズム
4:      * 商品種類を検索するプログラム
5:      *
6:      * メインクラス
7:      */
8:      public class Example2_4 {
9:
10:       /**
11:       * メイン
12:       * 商品種類を検索するプログラム
13:       */
14:       public static void main(String[] args) {
15:
16:         //自動販売機プログラムの開始を知らせる
```

```
17:         System.out.println("自動販売機が開始しました。");
18:
19:         //商品種類を保存するための配列を定義する
20:         int[] itemTypeArray = new int[10];
21:
22:         //商品種類を保存するための変数を初期化する
23:         for(int i=0;i<10;i++){
24:           itemTypeArray[i] = -1;//何も入っていないことを-1 として扱う
25:         }
26:
27:         //検索の準備として商品種類を追加する
28:         int addId = 1001;//コーラを追加する
29:         for(int i=0;i<10;i++){//商品種類が入っていない箱を探す
30:           if(itemTypeArray[i] == -1){//入っていない
31:             itemTypeArray[i] = addId;//書き込む
32:             break;
33:           }
34:         }
35:         addId = 1002;//ソーダを追加する
36:         for(int i=0;i<10;i++){//商品種類が入っていない箱を探す
37:           if(itemTypeArray[i] == -1){//入っていない
38:             itemTypeArray[i] = addId;//書き込む
39:             break;
40:           }
41:         }
42:
43:         //商品種類を検索する
44:         int searchId = 1002;//ソーダを検索する
45:         for(int i=0;i<10;i++){
46:           if(itemTypeArray[i] == searchId){//見つかった
47:             System.out.println("見つかりました");
48:             break;
49:           }
50:           if(i == 9){//最後まで辿ったが見つからなかった
51:             System.out.println("見つかりませんでした");
52:           }
53:         }
54:
55:         searchId = 1004;//DD レモンを検索する
56:         for(int i=0;i<10;i++){
57:           if(itemTypeArray[i] == searchId){//見つかった
58:             System.out.println("見つかりました");
59:             break;
60:           }
61:           if(i == 9){//最後まで辿ったが見つからなかった
62:             System.out.println("見つかりませんでした");
63:           }
64:         }
65:       }
66:     }
```

③.商品種類の削除

(1)削除のアルゴリズム

＜考えよう！＞削除のアルゴリズムを考えてみよう

日本語で記述してみよう！

(2)削除プログラム

例題 2-5：商品種類を削除する(Example2_5.java)

```
1:      /**
2:      * オブジェクト指向哲学　入門編
3:      * 例題 2-5：商品種類を削除する
4:      * 商品種類を削除するプログラム
5:      *
6:      * メインクラス
7:      */
8:      public class Example2_5 {
9:
10:       /**
11:       * メイン
12:       * 商品種類を削除するプログラム
13:       */
14:       public static void main(String[] args) {
15:
16:         //自動販売機プログラムの開始を知らせる
17:         System.out.println("自動販売機が開始しました。");
18:
19:         //商品種類を保存するための配列を定義する
20:         int[] itemTypeArray = new int[10];
21:
22:         //商品種類を保存するための変数を初期化する
```

```
23:         for(int i=0;i<10;i++){
24:           itemTypeArray[i] = -1;//何も入っていないことを-1 として扱う
25:         }
26:
27:         //削除の準備として商品種類を追加する
28:         int addId = 1001;//コーラを追加する
29:         for(int i=0;i<10;i++){//商品種類が入っていない箱を探す
30:           if(itemTypeArray[i] == -1){//入っていない
31:             itemTypeArray[i] = addId;//書き込む
32:             break;
33:           }
34:         }
35:         addId = 1002;//ソーダを追加する
36:         for(int i=0;i<10;i++){//商品種類が入っていない箱を探す
37:           if(itemTypeArray[i] == -1){//入っていない
38:             itemTypeArray[i] = addId;//書き込む
39:             break;
40:           }
41:         }
42:
43:         //商品種類を削除する
44:         int deleteId = 1002;//ソーダを削除する
45:         int i=0;//ループの回数を保存する
46:         for(i=0;i<10;i++){
47:           if(itemTypeArray[i] == deleteId){//見つかった
48:             itemTypeArray[i] = -1;//見つかったら、削除する（実は不要）
49:             break;
50:           }
51:         }
52:         //残りの要素をシフトする
53:         for(;i<9;i++){
54:           itemTypeArray[i] = itemTypeArray[i+1];
55:         }
56:       }
57:     }
```

2.2.3.　今回のプログラムの問題点

(1)追加のとき、配列がいっぱいだったらどうするか？

例示したプログラムでは、配列がいっぱいだったとき、それ以上追加できません。

つまりせっかく追加するつもりだったデータが追加されないままに終わるかもしれません。また、ユーザーはその事実を知ることができないかもしれません。

解決方法を考えてみましょう。

(2)削除のとき、みつからなかったらどうするか？

データを削除するときに、データが配列に見つからなかったとき、今回のプログラムはどのような動作になりますか？

解決方法と共に考えてみましょう。

(3)データの重複をどうする？

挿入した、あるいは挿入しようとするデータが一部同一のもので重複する場合、無駄に配列のスペースを消費してしまいますし、意図した動作をしない可能性があります。

問題点と解決方法を考えてみましょう。

これらは今回の演習プログラムでは考えていませんが、実際にはきちんと考えなければならないことです。

練習問題

＜記述問題＞

☆記述問題 2-1

配列を使う利点を自分の言葉で説明せよ。また、配列が使えない時はどのような時か考えよ。

＜プログラム問題＞

☆プログラム問題 2-1

商品種類を配列で管理するプログラムを書け。ただし、プログラム仕様と出力仕様を次のようなものとする

- プログラム仕様（仕事の手順）

1. コーラ、ソーダ、お茶の順番に登録する
2. ソーダを検索する
3. ソーダを削除する
4. ソーダを検索する
5. 商品種類リストを表示する