C言語修了課題（9900上級編）

99　修了課題

実行例のような動作をするプログラムを作成してください。プログラムに必要な関数などは、各自で考えて作成してください。なお、表示例内で、下線付き斜体の文字はキーボードからの入力を表します。

1．　kadai9901.c

次のような、文字がスクロールする簡易電光掲示板を作成してください。

・スクロールは、右から左へとスクロールします。
・スクロールする文字は、半角20文字以内とします。
・表示する文字は、予め設定しておきましょう。
・スクロールは高速なので、次のライブラリを利用して遅延時間を設定してください。
　Sleep(n)関数 ←　nで指定したミリ秒だけ停止する。n=1000の場合、1秒停止する。
　#include <Window.h>　←Sleep()関数を利用するためのヘッダー・ファイル
・同じ行に重ね書きする場合、¥nではなく¥rを利用します。
　printf(”ABCDEF¥r”); printf(”123”);　→　実行結果：123DEF

実行例

```
Enjoy fun C world!!    ←このメッセージが、右から左にスクロールして表示
        ↓
C world!! Enjoy fun
```

参考：Sleep()関数のプログラム例と実行例

```
/* TEST Sleep(n) */
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
#include <Windows.h>
void main(void)
{
  int i = 0;
  for(i = 0; i < 10; i++)
  {
    printf(”i=%d ”, i);
    Sleep(500); //  0.5秒(500/1000秒)だけ待つ
  }
  return;
}
```

```
i=0 i=1 i=2 i=3 i=4 i=5 i=6 i=7 i=8 i=9
```

0.5秒ごとに、i=<数字>が表示されます。

２．　kadai9902.c

int型の２次元配列に格納された値を基にして、全角文字で図形を表示します。
表示は、図形を時計回りに0～270度まで90度づつ回転した、3通りの図形を表示します。

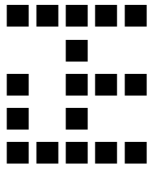
配列内の要素の位置を、処理により入れ替えてください。
入れ替えの結果を、■と全角空白で表示します。
単純にprintf()文でパターンを表示してはいけません。

配列の内容
・配列は、5行5列での２次元配列です。
・配列の要素と表示内容　　1→■を表示、0→全角空白を表示
・このプログラムの図形イメージ

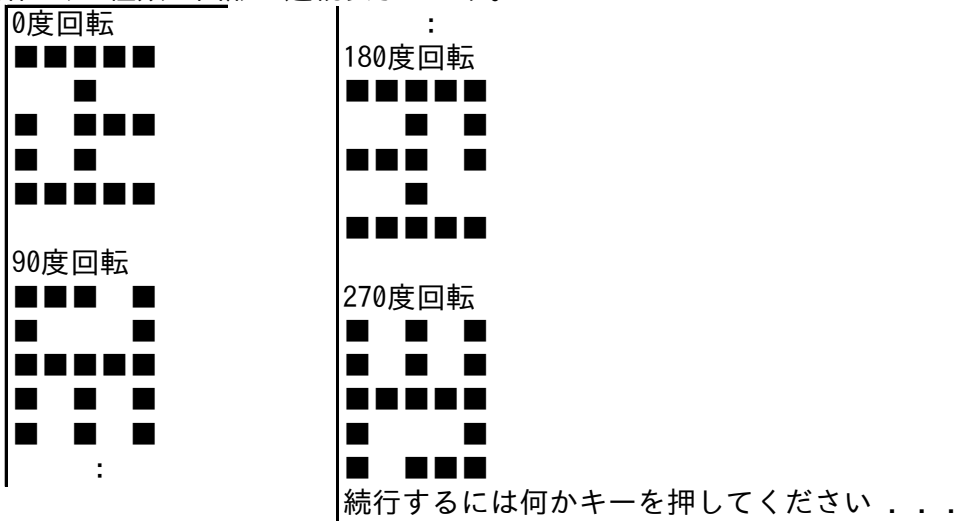
配列内の要素

1	1	1	1	1
0	0	1	0	0
1	0	1	1	1
1	0	1	0	0
1	1	1	1	1

表示時のイメージ



実行例
　1回の動作で、4種類の図形を連続表示します。



３．　kadai9903.c

よく見かけるバーコードとして次のような種類があります。

・標準コード(JANコード)　13桁 4901480192505 など
・短縮コード(JANコード)　 8桁 49123456 など
・UPCコード(米国、カナダ) 12桁 061414112345 など
・ISBNコード(現在の規格) 13桁 978-4-8657-9047-4 など
・ISBNコード(旧規格)　　10桁 4-865-79056-X など

入力されたバーコードが正しいか否かを返す関数chkBarcode()と、確認用のプログラムを作成してください。

引数の文字配列の最後には必ずNULLが入っており、文字は0～1、X以外は無いものとします。

```
int chkBarcode(char bar[])
戻り値：　int型 0:読取正常、1:読取エラー
引　数：　char型 任意の桁数のバーコードの数字や文字が入った文字配列。
```

なお、main()では、バーコードを記憶する配列として、次の配列を宣言してください。
この配列内の各要素に、入力したバーコードの数字や文字を1文字ずつ記憶します。

```
char bar[15] = "";
```

バーコードの最後の1桁は、正常に読み取ったか否かを確認する数字や文字になります。この桁をチェックデジットといいます。チェックデジットの計算は、次の手順で行います。

・標準コード(JANコード)　13桁 4901301262288 など

桁	13	12	11	10	9	8	7	6	5	4	3	2	1
値	4	9	0	1	3	0	1	2	6	2	2	8	8

① 13～3までの、奇数桁の数字を加算します。　4 + 0 + 3 + 1 + 6 + 2 → 16
② 12～2までの、偶数桁の数字を加算します。　9 + 1 + 0 + 2 + 2 + 8 → 22
③ ②の値を3倍します。　22 × 3 → 66
④ ①＋②を計算します。　16 + 66 → 82
⑤ ④の値を、10で割った余りを求めます。　82 ÷ 10 → 8あまり2 → 2
⑥ 10から⑤の値を引きます。　10 - 2 → 8
　この値が、チェックデジットです。　チェックデジットは、８となります。
⑦ ⑥の値とバーコードの1桁目の値が同じなら、正常に読み取られています。

・短縮コード(JANコード)　 8桁 49123456 など

桁	13	12	11	10	9	8	7	6	5	4	3	2	1
値	0	0	0	0	0	4	9	1	2	3	4	5	6

※13桁～9桁までの5桁に0を埋め込みます。計算方法は標準コードと同じです。

① 13～3までの、奇数桁の数字を加算します。　0 + 0 + 0 + 9 + 2 + 4 → 15
② 12～2までの、偶数桁の数字を加算します。　0 + 0 + 4 + 1 + 3 + 5 → 13
③ ②の値を3倍します。　13 × 3 → 39
④ ①＋②を計算します。　15 + 39 → 54
⑤ ④の値を、10で割った余りを求めます。　54 ÷ 10 → 5あまり4 → 4
⑥ 10から⑤の値を引きます。　10 - 4 → 6
　この値が、チェックデジットです。　チェックデジットは、６となります。
⑦ ⑥の値とバーコードの1桁目の値が同じなら、正常に読み取られています。

・UPCコード(米国、カナダ) 12桁 061414112345 など

桁	12	11	10	9	8	7	6	5	4	3	2	1
値	0	6	1	4	1	4	1	1	2	3	4	5

※基本的な計算方法は標準コードと同じです。

① 11～3までの、奇数桁の数字を加算します。　6 + 4 + 4 + 1 + 3 → 18
② 12～2までの、偶数桁の数字を加算します。　0 + 1 + 1 + 1 + 2 + 4 → 9
③ ②の値を3倍します。　9 × 3 → 27
④ ①＋②を計算します。　18 + 27 → 45
⑤ ④の値を、10で割った余りを求めます。　45 ÷ 10 → 4あまり5 → 5
⑥ 10から⑤の値を引きます。　10 - 5 → 5
　この値が、チェックデジットです。　チェックデジットは、５となります。
⑦ ⑥の値とバーコードの1桁目の値が同じなら、正常に読み取られています。

・ISBNコード(現在の規格) 13桁 978-4-8657-9047-4 など

桁	13	12	11	10	9	8	7	6	5	4	3	2	1
値	9	7	8	4	8	6	5	7	9	0	4	7	4

※基本的な計算方法は標準コードと同じです。

① 13～3までの、奇数桁の数字を加算します。　9 + 8 + 8 + 5 + 9 + 4 → 43
② 12～2までの、偶数桁の数字を加算します。　7 + 4 + 6 + 7 + 0 + 7 → 31
③ ②の値を3倍します。　31 × 3 → 93
④ ①＋②を計算します。　43 + 93 → 126
⑤ ④の値を、10で割った余りを求めます。　126 ÷ 10 → 12あまり6 → 6
⑥ 10から⑤の値を引きます。　10 - 6 → 4
　この値が、チェックデジットです。　チェックデジットは、４となります。
⑦ ⑥の値とバーコードの1桁目の値が同じなら、正常に読み取られています。

・ISBNコード(旧規格)　10桁 4-865-79056-X など

桁	10	9	8	7	6	5	4	3	2	1
値	4	8	6	5	7	9	0	5	6	X

① 10～2の各桁について、桁の番号と桁の値を掛け、全桁を集計します。
　10×4 + 9×8 + 8×6 + 7×5 + 6×7 + 5×9 + 4×0 + 3×5 + 2×6 → 309
② ①の値を、11で割った余りを求めます。　309 ÷ 11 → 28あまり1 → 1
③ 11から②の値を引きます。　11 - 1 → 10
④ ③の値に応じて、次のように値を決めます。　1～9の場合：そのまま1～9の値
　10の場合：X
　11の場合：0
　この値が、チェックデジットです。　チェックデジットは、Xとなります。
⑤ ④の値とバーコードの1桁目の値が同じなら、正常に読み取られています。

実行例

```
バーコード=4901480192505
OK!
バーコード=4901480192500
ERROR!
バーコード=49123456
OK!
バーコード=49123450
ERROR!
バーコード=061414112345
OK!
バーコード=061414112340
ERROR!
```

```
バーコード=9784865790474
OK!
バーコード=9784865790470
ERROR!
バーコード=486579056X
OK!
バーコード=4865790560
ERROR!
バーコード=4865790470
OK!
バーコード=4865790471
ERROR!
```

４．　kadai9904.c

次のような、動作をする自動販売機を作成してください。

〇自販機の機能

・商品は、120円のお茶の1種類です。
・投入できるお金は、10,50,100円の硬貨のみとします。
・商品は、販売機内に10本まで在庫できます。
・売切れ、または、釣銭不足の場合は販売できません。
・商品は、必ず購入するものとします。
・起動時は、次のような構成とします。
　・商品数：10缶
　・釣り銭用硬貨の枚数：100円、50円、10円すべて10枚
・現在、ここまでの機能とします。
　余裕があれば、商品点数の追加やコインの種類を増やしてみましょう。

〇購入時の動作

・硬貨の投入前に、「いらっしゃいませ。」と表示します。
・金額の大きい順に、硬貨の枚数を入力します。
・最後の硬貨を入力後、自動的に1本販売します。
・販売後、次のような動作を行います。
　・「ありがとうございました。」と表示。
　・釣り銭を、各硬貨の枚数で表示。
　・商品を1減らす。
　・自販機内の硬貨を、釣り銭に応じて減らす。
・商品数が0個になった場合、次のような動作を行います。
　・「売切れです。」と表示します。
・釣り銭の処理
　・釣り銭が発生しない場合、釣り銭の表示は行いません。
　・釣り銭は、100円から順番に支払います。
　・硬貨が不足する場合は、より金額の小さい金額の硬貨で支払います。
　・最終的に硬貨が不足する場合
　　・「釣り銭切れです。」と表示します。
　　・投入された金額を、釣り銭として返却します。
・自販機内のコインスタッカー
　・自販機内では、硬貨は無制限に保存できるものとします。

・簡単な処理手順

※自→自販機内の枚数、投→投入枚数、(釣)→釣予定枚数、釣→釣り銭枚数

・通常販売時1(100円1枚１,10円2枚を投入)

	自
100円	10
50円	10
10円	10
在庫	10

2800

→

投
1
0
2

→

自
11
10
12
10

→

(釣)
0
0
0

→

釣	自
0	11
0	10
0	12
	9

2800

ありがとうございました。

・通常販売時2(100円2枚を投入)

	自
100円	11
50円	10
10円	12
在庫	9

2800

→

投
2
0
0

→

自
13
10
12
9

→

(釣)
0
1
3

→

釣	自
0	13
1	9
3	9
	8

2800

ありがとうございました。

・通常販売時3(10円20枚を投入)

	自
100円	13
50円	9
10円	9
在庫	8
	2800

→

投
0
0
20

→

自
13
9
29
8

→

(釣)
0
1
3

→

釣	自
0	13
1	8
3	26
	7
	2800

ありがとうございました。

・より小さな硬貨での支払い時(50円4枚を投入)

	自
100円	0
50円	30
10円	70
在庫	5
	2800

→

投
0
4
0

→

自
0
34
70
5

→

(釣)
0
1
3

→

釣	自
0	0
1	33
3	67
	4
	2800

ありがとうございました。

・より小さな硬貨での支払い時(100円2枚を投入)

	自
100円	0
50円	0
10円	244
在庫	3
	2800

→

投
2
0
0

→

自
2
0
244
3

→

(釣)
0
0
8

→

釣	自
0	2
0	0
8	236
	2
	2800

ありがとうございました。

・釣り銭切れ時(100円2枚を投入)

	自
100円	20
50円	11
10円	1
在庫	2
	2800

→

投
2
0
0

→

自
22
11
1
2

→

(釣)
0
1
3 *

→

釣	自
0	20
1	11
3	1
	2
	2800

釣り銭切れです。

・売切れ時(100円2枚を投入)

	自
100円	23
50円	9
10円	5
在庫	0
	2800

→

投
2
0
0

→

自
25
9
5
0 *

→

(釣)
-
-
-

→

釣	自
2	23
0	9
0	5
	0
	2800

売切れです。

実行例

```
いらっしゃいませ。
100円:1
 50円:0
 10円:2
ありがとうございました。
***お釣り***
100円:0枚    50円:0枚    10円:0枚

いらっしゃいませ。
100円:
        :
```

```
売切れです
100円:1
 50円:1
 10円:0
売切れです
***お釣り***
100円:1枚    50円:1枚    10円:0枚

売切れです
100円:
        :
```

５．　kadai9905.c

次のような、移動方向をアニメーションで表示する画面を作成してください。

・起動時は、●のパターンを表示します。
・[↑][↓][←][→]を押すと、キーに応じたアニメーションが流れます。
・[Enter]キーで停止します。
・[Esc]キーで終了します。
・ドットパターンとキー入力を検知する部分は提供します。
・プログラムの雛型と、動作サンプルのexeファイルを提供します。
・スクロールは高速なので、ライブラリを利用して遅延時間を設定してください。
　Sleep(n)関数 ←　nで指定したミリ秒だけ停止する。n=1000の場合1秒停止
　system("cls") ←　画面クリアを行う
　#include <Window.h>　← 上記の関数を利用するためのヘッダー・ファイル

実行例

```
Enterで停止/Escで終了

   *
  ***
 *****
   *
   *
   *
```

６．　kadai9906.c

次のような、デジタルクロックを作成してください。

・時計機能にアクセスするために、次のライブラリを利用してください。

```
#include <time.h>          ←　時刻を利用するためのヘッダー・ファイル
#include <Windows.h>       ←　時間待ち関数Sleep()を利用する場合のヘッダー・ファイル
```

実行例

```
Ctrl+Cで終了
2016/12/26 10:00:00    ←時刻は、同一行にリアルタイムで表示されます
```

7．　kadai9907.c

あるI/Oポートからは、画像スキャンした白黒のパターンが次のような95個の0と1のビットパターンとして送られてきます。ここでは、このひとつひとつのビットパターンをモジュールと呼ぶことにします。　このモジュールの構成を解析して、送られてきたビットパターンが表す一連の数字を表示してください。

なお、ビットパターンの入力は面倒ですので、次のようにプログラム内に文字列のデーターとして記述して制作してください。

例：unsigned char code[96] =
{"10100010110100111001100101111011011100101001110101010111001110010101000011100101010000101000010 1"}

（※本来、このようなビットパターンのデーターはビット列で扱います。ここでは最初は'0'や'1'の1バイトの文字で扱うことにします。）

実行例

```
解析結果 = 4901750406066
```

●ビットパターンの認識方法

次ページからの参考資料を読んで、作成してください。

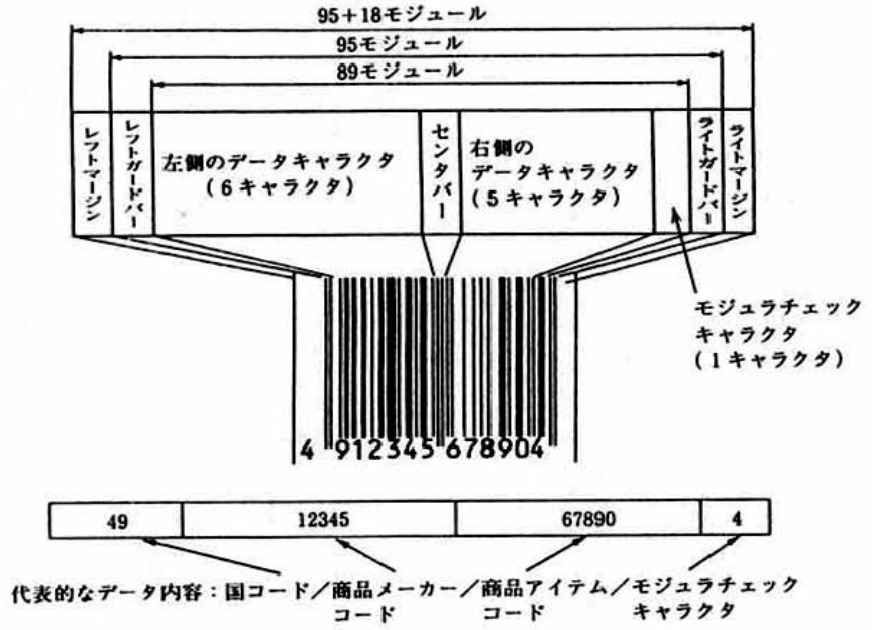
参考資料

●JANコード概要

・ JANコードは13桁または8桁で表される数字のコードです。(EANコードも同じ)

・ ４種類の太さの色付のバー(棒：多くは黒色)とスペース(空白：多くは白色)で構成されます。

・ JANコードにはキャラクタ間のギャップは無く、１キャラクタは７モジュール(２本のバーと２本のスペース)で構成されます。



●１個のキャラクター内のモジュール構成

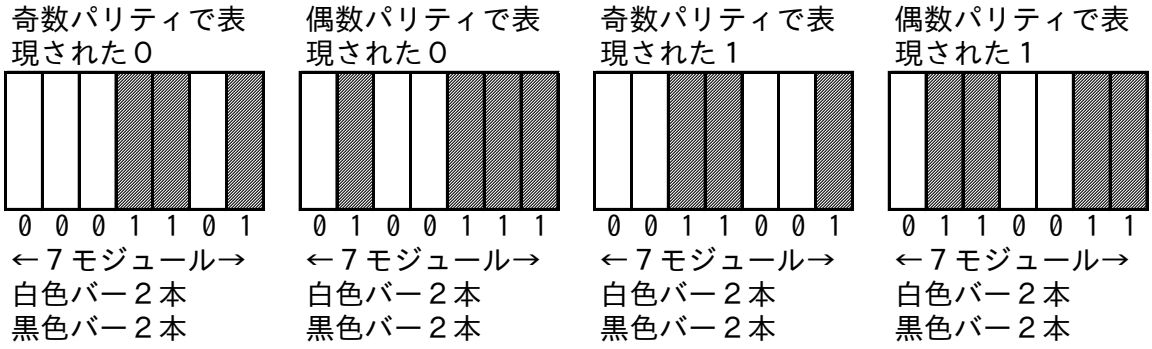
○パリティについて

ある数字の並び(多くは２進数)の合計が偶数または奇数かどうかを調べることで通信の誤りを検出する技術です。偶数をeven parity(偶数パリティ)、奇数をodd parity(奇数パリティ)といいます。そして、その誤りを調べることをパリティ・チェック(parity check)といいます。

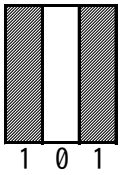
○モジュールの構成例

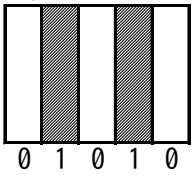
JANコードでは、左右に配置された０～９の数字(データーキャラクター)を表すため次のようなモジュール構成を採用しています。

１個のキャラクタ(０から９までの１文字)は、図のように７モジュールで表現され、２本のバーと２本のスペースで表します。



●ガードバーとセンターバーのモジュール構成
ガードバーは３モジュール、センターバーは５モジュールで構成されています。
各々のモジュール構成は、次のようになっています。
○ガードバー(左右同じ構成です)　　○センターバー





●JANコードの構成
13桁のJANコードは、左側から次のような構成となっています。

レフトマージン	(11モジュール以上)
レフトガイドバー	(３モジュール)
６桁のデーターキャラクタ	(42モジュール)
センターバー	(５モジュール)
５桁のデーターキャラクタ	(35モジュール)
１桁のモジュラーチェックキャラクター(チェックデジット)	(７モジュール)
ライトガイドバー	(３モジュール)
ライトマージン	(７モジュール以上)

□内は、合計95モジュール

８桁のJANコードは、左側から次のような構成となっています。

レフトマージン(７モジュール以上)	
レフトガイドバー	(３モジュール)
４桁のデーターキャラクタ	(28モジュール)
センターバー	(５モジュール)
３桁のデータキャラクター	(21モジュール)
１桁のモジュラーチェックキャラクター(チェックデジット)	(７モジュール)
ライトガイドバー	(３モジュール)
ライトマージン	(７モジュール以上)

□内は、合計67モジュール

●プリフィクスとパリティ
センターバーの左右に出現するキャラクターは、常に同じモジュール構成ではありません。13桁と８桁各々のコードでは、次のようなモジュール構成を採用しています。

○13桁のJANコードの場合
・センターバーより左側のキャラクター
最初の１桁目のキャラクター(プリフィックスキャラクター)の数値に応じて、奇数と偶数のパリティを組み合わせてキャラクターを構成します。

プリフィックスキャラクター	組み合わせ
0	奇 奇 奇 奇 奇 奇
1	奇 奇 偶 奇 偶 偶
2	奇 奇 偶 偶 奇 偶
3	奇 奇 偶 偶 偶 奇
4	奇 偶 奇 奇 偶 偶 ←日本は国コードが49、45のため、この行です。
5	奇 偶 偶 奇 奇 偶
6	奇 偶 偶 偶 奇 奇
7	奇 偶 奇 偶 奇 偶
8	奇 偶 奇 偶 偶 奇
9	奇 偶 偶 奇 偶 奇

奇…奇数パリティ
偶…偶数パリティ

・センターバーより右側のキャラクター
常に、右側専用の奇数パリティで表現されるモジュール構成となります。

○８桁のJANコードの場合
・センターバーより左側のキャラクター
常に、奇数パリティで表現されるモジュール構成となります。
・センターバーより右側のキャラクター
常に、右側専用の奇数パリティで表現されるモジュール構成となります。

○キャラクターとモジュール構成の詳細

10進数	左側のデーターキャラクター 奇数パリティ	左側のデーターキャラクター 偶数パリティ	右側専用のデーターキャラクタ及びモジュラチェックキャラクター
0	0 0 0 1 1 0 1	0 1 0 0 1 1 1	1 1 1 0 0 1 0
1	0 0 1 1 0 0 1	0 1 1 0 0 1 1	1 1 0 0 1 1 0
2	0 0 1 0 0 1 1	0 0 1 1 0 1 1	1 1 0 1 1 0 0
3	0 1 1 1 1 0 1	0 1 0 0 0 0 1	1 0 0 0 0 1 0
4	0 1 0 0 0 1 1	0 0 1 1 1 0 1	1 0 1 1 1 0 0
5	0 1 1 0 0 0 1	0 1 1 1 0 0 1	1 0 0 1 1 1 0
6	0 1 0 1 1 1 1	0 0 0 0 1 0 1	1 0 1 0 0 0 0
7	0 1 1 1 0 1 1	0 0 1 0 0 0 1	1 0 0 0 1 0 0
8	0 1 1 0 1 1 1	0 0 0 1 0 0 1	1 0 0 1 0 0 0
9	0 0 0 1 0 1 1	0 0 1 0 1 1 1	1 1 1 0 1 0 0

※左側の偶数パリティのパターンの逆になっています。

例：4901750406066のビットデーターの場合

１）まず、ガイドバーのビットパターンを読み飛ばします。ガイドバーは、手動でスキャンした場合などスキャン速度を一定に保つことができない場合の速度補正用などに利用されます。これは、センターバーも同じです。

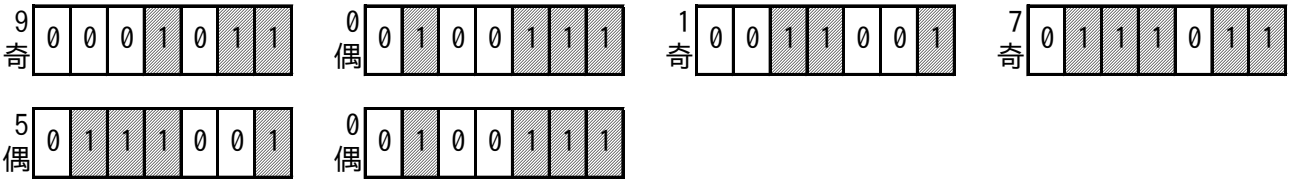
２）13桁のJANコードですが、実際にスキャンされるビットデーターは12桁分です。

つまり、901750406066という先頭の１文字が無い12桁がスキャン結果として得られます。

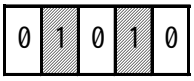
先頭の４は、左側の６文字が持つ奇数偶数のパリティの並びにより決定します。

たとえば、901750という６桁の各々のビットパターンを左用のパターン一覧から検索します。

各々の桁のピットパターンのパリティが次のような順番で並んでいる場合なら、
奇数、偶数、奇数、奇数、偶数、偶数
プリフィクスキャラクターは、４であることが一覧表から判ります。

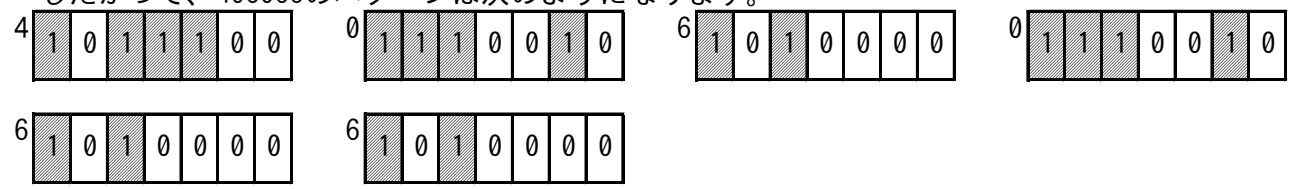


３）コードの中心にはセンターバーが存在します。



（センターバー）

４）右側のビットパターンは、右側専用の統一されたパターンです。
したがって、406066のパターンは次のようになります。



※補足
JANコードは、左右いずれの方向からスキャンしても判断できるように設計されています。
注目点は２箇所あります。まず、左側の偶数パリティのパターンと右側の偶数パリティのパターンは、ちょうど左右逆になっています。そして、プリフィクスキャラクターの構成は必ず奇数パリティから開始しています。
例えば、右からスキャンすると最初の文字は偶数パリティを持つ６という文字であることが判ります。しかし、最初の文字が偶数パリティであるプリフィクスキャラクターは存在しません。
そこで、読み込んだビットのパターンを左右反転して再度、解析を行います。

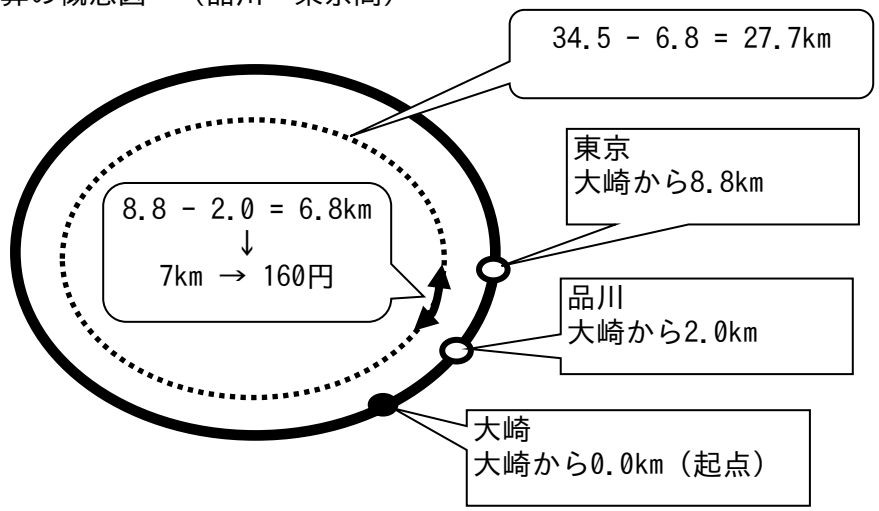
８．　kadai9908.c

手軽に山手線の運賃計算を行うシステムを作ることになりました。
乗車駅と降車駅を入力すると、距離と運賃を表示するプログラムを作成してください。

入力条件
・駅名は漢字で入力します。
・「乗車駅 ==>」を表示して乗車駅を漢字で入力します。
・「降車駅 ==>」を表示して降車駅を漢字で入力します。
・該当する駅が存在しない場合は、再度、入力メッセージを表示して再入力します。
・乗車駅に半角の「END」を入力すると、処理を終了します。

計算方法
１．　入力した各々の駅について、大崎からの距離を検索して求める。
２．　検索した２種類の距離の差を求めて記憶する。（正の数で求めます）
３．　山手線の１周の距離から、２で求めた距離を引いて記憶する。
４．　２と３で求めた距離の短いほうを採用する。
５．　距離は小数点以下の端数を切り上げとする。
６．　５で求めた距離をもとに、運賃換算表から該当する運賃を求める。

運賃計算の概念図　（品川～東京間）



運賃表示
距離（全体の桁数は５桁で小数点１桁）と運賃（５桁）を表示してください。

実行例

```
乗車駅 ==> 品川
降車駅 ==> 東京
距離 =   6.8km
運賃 =   160円
乗車駅 ==> 東京
降車駅 ==> 品川
距離 =   6.8km
運賃 =   160円
乗車駅 ==> 大阪
乗車駅 ==> 品川
降車駅 ==> 難波
降車駅 ==> 東京
乗車駅 ==> END
```

※プロンプト内の漢字入力は、Alt+[漢字]を押します。
終了は、再度、Alt+[漢字]または[漢字]を押します。

■駅間の距離と運賃換算表

○駅名と駅間の距離

No	駅名	駅間 (km)	大崎から (km)
1	大崎	0.0	0.0
2	品川	2.0	2.0
3	田町	2.2	4.2
4	浜松町	1.5	5.7
5	新橋	1.2	6.9
6	有楽町	1.1	8.0
7	東京	0.8	8.8
8	神田	1.3	10.1
9	秋葉原	0.7	10.8
10	御徒町	1.0	11.8
11	上野	0.6	12.4
12	鶯谷	1.1	13.5
13	日暮里	1.1	14.6
14	西日暮里	0.5	15.1
15	田端	0.8	15.9
16	駒込	1.6	17.5
17	巣鴨	0.7	18.2
18	大塚	1.1	19.3
19	池袋	1.8	21.1
20	目白	1.2	22.3
21	高田馬場	0.9	23.2
22	新大久保	1.4	24.6
23	新宿	1.3	25.9
24	代々木	0.7	26.6
25	原宿	1.5	28.1
26	渋谷	1.2	29.3
27	恵比寿	1.6	30.9
28	目黒	1.5	32.4
29	五反田	1.2	33.6
1	大崎	0.9	34.5

○運賃換算表

営業キロ

km	運賃(円)
1～ 3	130
4～ 6	150
7～10	160
11～15	190
16～20	250
21～25	380
26～30	450
31～35	540

※kmの端数は切り上げ

以上