ロボット博士養成講座

ロボティクスプロフェッサーコース

不思議アイテムⅢ-2②

第4回

キャラクターを動かそう

講 師 用

目　　次

○　授業開始にあたって

授業のはじめは、着席させ、大きな声であいさつしてから始めます。

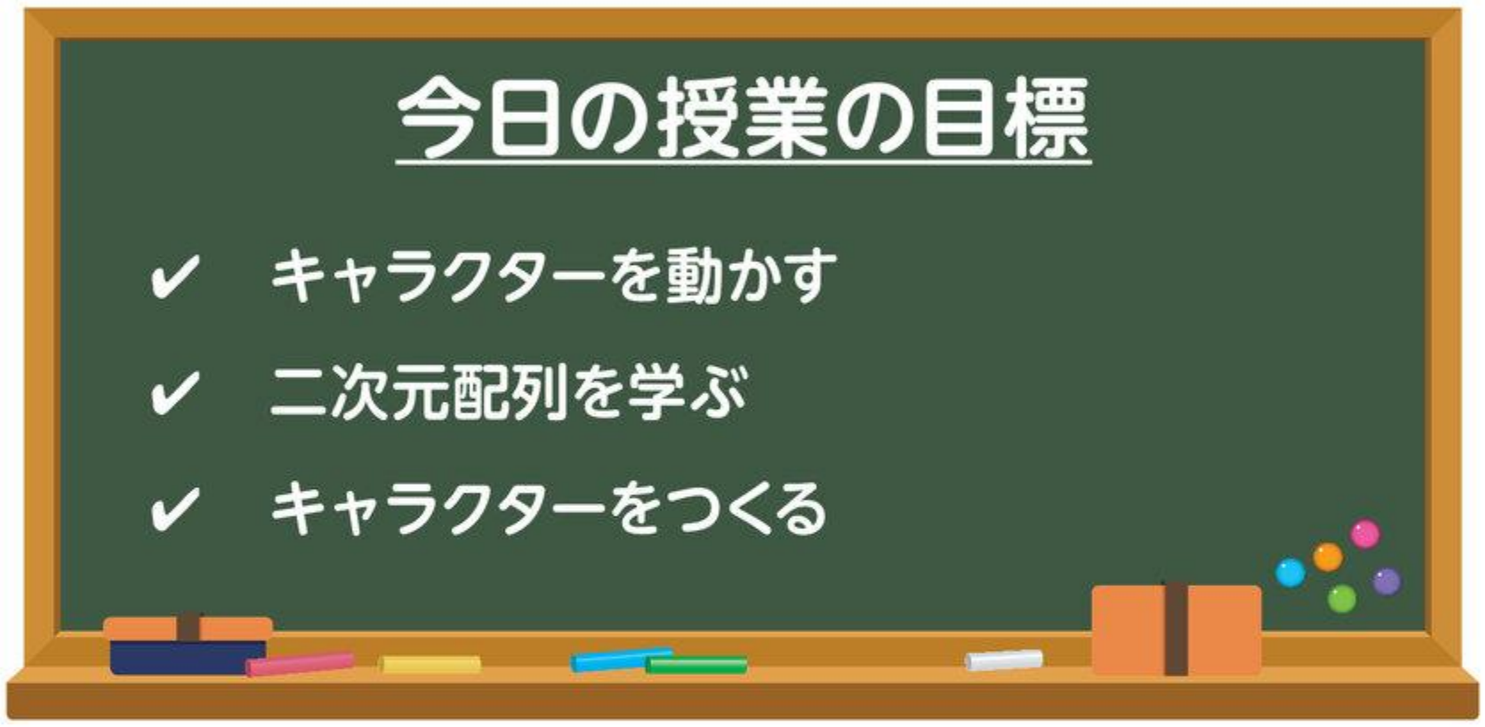
○　今回の目標をパネルで用意するか、黒板に予め書いておきます。

（授業の目標を明確化することは大変重要なことですので、生徒によく理解させます）

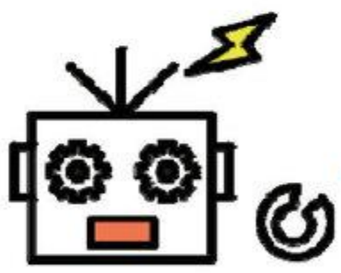
目安時間は授業時間 120 分のうち、休憩 10 分程度取ることを想定しています。
生徒の進捗状況により、休憩時間などを調整して授業を行ってください。

0.　キャラクターを動かそう（目安 5 分）

0.0.　「キャラクターを動かそう」でやること



今回は、前回使ったキャラクターを移動、回転、拡大するなど自在に動かしていきましょう。
また、「配列」「二次元配列」を使ってデータ化したキャラクターを効率的に動かすための
テクニックを実践(じっせん)します。
前回も説明しましたが、マイコンのメモリーはパソコンのメモリーに比べるととても小さく、キャラクターの動きによっていちいちプログラムしていたらメモリーの領域を無駄に使ってしまいます。
メモリーの節約になるように、ひとつのキャラクターデータを有効に活用することはとても大切で、それはプログラマーへの近道になります。
授業の後半では、キャラクターを自分でつくって動かしてみましょう。



キャラクターを自在に操ろう！

0.1.　必要なもの

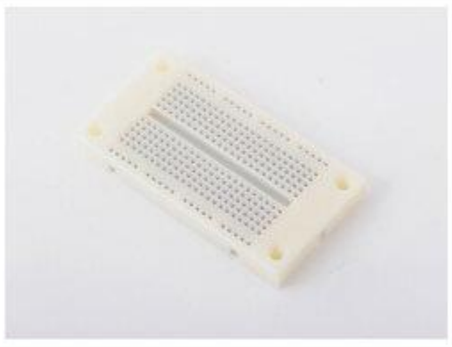
ラジオペンチ	1	USB ケーブル	1	マイコンボード	1	ロボプロシールド	1
							
301 ブレッドボード	1	ジャンパー線	65	抵抗 (100 Ω /1k Ω /10k Ω)	各10	タクトスイッチ	10
							
緑色 LED	10	姿勢検出シールド	1	液晶ディスプレイシールド	1		
							

図 0-1　必要なもの

1．　配列を使ってキャラクターを表示する（目安 100 分）

1.0.　変数と配列の復習

まず、動作確認も兼(か)ねて前回使ったキャラクターを表示してみましょう。
次の２つのプログラムの実行結果では、どちらも同じキャラクターが表示されますが、実はプログラムが異なります。（A）は配列を使ったプログラム、（B）は二次元配列を使ったプログラムです。

プログラムの書き込み

（A）RoboticsProfessorCourse3 > MagicItemLCD4 > Chara1
（B）RoboticsProfessorCourse3 > MagicItemLCD4 > Chara2

そもそも配列とは複数の同じ型の変数を1つにまとめたものです。
つまり、１つの変数名で複数の記憶領域(きおくりょういき)を管理することができるということです。
プログラム「Chara1/2」はドット絵をかくための「点の位置」と「色番号」を使うので、データを自動的に読み出したいときは配列を使うととても便利です。
それでは、配列を使ったプログラム、二次元配列を使ったプログラムの2つの違(ちが)いを見比べてみましょう。

プログラム「Chara1」より抜粋(ばっすい)

```
const char chara[] = {
    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
    0, 0, 0, 0, 0, 0, 0, 6, 6, 0, 0, 0, 0, 0, 0, 0,
    0, 0, 0, 0, 0, 0, 0, 6, 6, 0, 0, 0, 0, 0, 0, 0,
    0, 0, 0, 0, 0, 0, 6, 6, 6, 6, 0, 0, 0, 0, 0, 0,
    0, 0, 0, 0, 0, 0, 6, 6, 6, 6, 0, 0, 0, 0, 0, 0,
    0, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6,
    0, 0, 6, 6, 6, 6, 0, 6, 6, 0, 6, 6, 6, 6, 0, 0,
    0, 0, 0, 6, 6, 6, 0, 6, 6, 0, 6, 6, 6, 0, 0, 0,
    0, 0, 0, 0, 6, 6, 0, 6, 6, 0, 6, 6, 0, 0, 0, 0,
    0, 0, 0, 0, 6, 6, 6, 6, 6, 6, 6, 6, 0, 0, 0, 0,
    0, 0, 0, 6, 6, 6, 6, 2, 2, 6, 6, 6, 6, 0, 0, 0,
    0, 0, 0, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 0, 0, 0,
    0, 0, 6, 6, 6, 6, 6, 0, 0, 6, 6, 6, 6, 6, 0, 0,
    0, 0, 6, 6, 6, 0, 0, 0, 0, 0, 0, 6, 6, 6, 0, 0,
    0, 6, 6, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 6, 6, 0
};
```

プログラム「Chara2」より抜粋

```
const char chara[sizeY][sizeX] = {    // Y、X の座標順になることに注意
      {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0}
    , {0, 0, 0, 0, 0, 0, 0, 6, 6, 0, 0, 0, 0, 0, 0}
    , {0, 0, 0, 0, 0, 0, 0, 6, 6, 0, 0, 0, 0, 0, 0}
    , {0, 0, 0, 0, 0, 0, 6, 6, 6, 6, 0, 0, 0, 0, 0}
    , {0, 0, 0, 0, 0, 0, 6, 6, 6, 6, 0, 0, 0, 0, 0}
    , {0, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6}
    , {0, 0, 6, 6, 6, 6, 0, 6, 6, 0, 6, 6, 6, 6, 0}
    , {0, 0, 0, 6, 6, 6, 0, 6, 6, 0, 6, 6, 6, 0, 0}
    , {0, 0, 0, 0, 6, 6, 0, 6, 6, 0, 6, 6, 0, 0, 0}
    , {0, 0, 0, 6, 6, 6, 6, 6, 6, 6, 6, 6, 0, 0, 0}
    , {0, 0, 0, 6, 6, 6, 6, 2, 2, 6, 6, 6, 6, 0, 0}
    , {0, 0, 0, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 0, 0}
    , {0, 0, 6, 6, 6, 6, 6, 0, 0, 6, 6, 6, 6, 6, 0}
    , {0, 0, 6, 6, 6, 0, 0, 0, 0, 0, 0, 6, 6, 6, 0}
    , {0, 6, 6, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 6, 6}
};
```

配列内の数字の並びはどちらも同じですが、変数chara についている添え字（[ ]内の数）が「Chara1」では1つ、「Chara2」では2つになっていますね。この、添え字を2つ用いた配列のかき方が重要なのです。

プログラム「Chara1」より抜粋

```
TFTscreen.stroke(B[chara[y * sizeX + x]], G[chara[y * sizeX + x]],
R[chara[y * sizeX + x]]); // 色見本データを参照する
```

プログラム「Chara2」より抜粋

```
TFTscreen.stroke(B[chara[y][x]], G[chara[y][x]], R[chara[y][x]]);
// 色見本データを参照する
```

変数chara を使用する部分を見比べると、「Chara2」の方がだいぶシンプルになっています。「Chara1」ではあるマスを塗るとき「ここはx = 6, y = 4なので、左上から数えて4×15＋6＝66番のセルです」と変換していましたが、二次元配列を使うことで「ここはx = 6, y = 4のセルです」と直接言えるようになったためです。

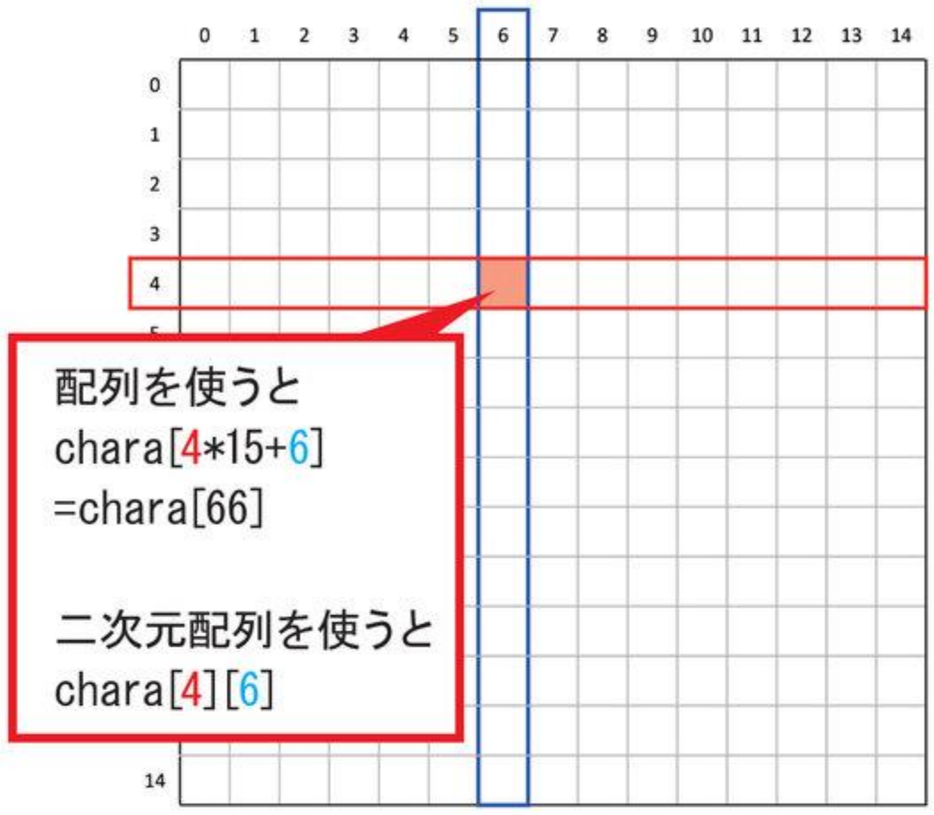


図1-0　配列と二次元配列

1.1.　キャラクターを表示させる

やってみよう！

前回つくった「たこ？」の画像データをかきかえ、二次元配列を用いた形に直してみよう！

ヒント

各色のデータは以下の通りだよ。ケタ数をそろえるため、10進数ではなく16進数でかいてみよう。
プログラムは前回使用した「DotChara3」をベースにしていこう！

色番号（10 進数 / １６進数表記）

色番号	色	色番号	色
0 (0x00)	黒	8 (0x08)	灰色
1 (0x01)	明るい青	9 (0x09)	少し暗い青
2 (0x02)	明るい赤	10 (0x0a)	少し暗い赤
3 (0x03)	明るい紫	11 (0x0b)	少し暗い紫
4 (0x04)	明るい緑	12 (0x0c)	少し暗い緑
5 (0x05)	明るい水色	13 (0x0d)	少し暗い水色
6 (0x06)	明るい黄色	14 (0x0e)	少し暗い黄色
7 (0x07)	白	15 (0x0f)	少し暗い白

講　解答例は以下のプログラムです。
RoboticsProfessorCourse3　＞　MagicItemLCD4　＞　Chara3

チャレンジ課題

オリジナルキャラクターをつくって、液晶ディスプレイに表示してみよう。
まずはデザインを考えて、**表1-0**（21×21）に色データをかき出してみよう。
表が完成したら、プログラム**「Chara2」**のキャラクターデータを変更(へんこう)して、液晶ディスプレイに表示させてみよう。イメージ通りに表示されたら成功だよ。
プログラム内の縦(たて)横のサイズを定義するsizeX、sizeYの値を忘れずに変更(へんこう)してね。

表1-0　データ用方眼

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
0																					
1																					
2																					
3																					
4																					
5																					
6																					
7																					
8																					
9																					
10																					
11																					
12																					
13																					
14																					
15																					
16																					
17																					
18																					
19																					
20																					

1.2.　キャラクターを動かす

次にキャラクターを動かすプログラムをつくります。
だんだん本格的な動作になり難しそうに見えてくる頃かもしれませんが、今までのロボット製作と同様に「必要な動作を切り分けて考えて、1つずつつくったものを最後に合体させる」という手順を守っていれば比較的悩むことなく作業を進められるでしょう。

まずは、最終的にどのような動作をめざすかを見ておきましょう。

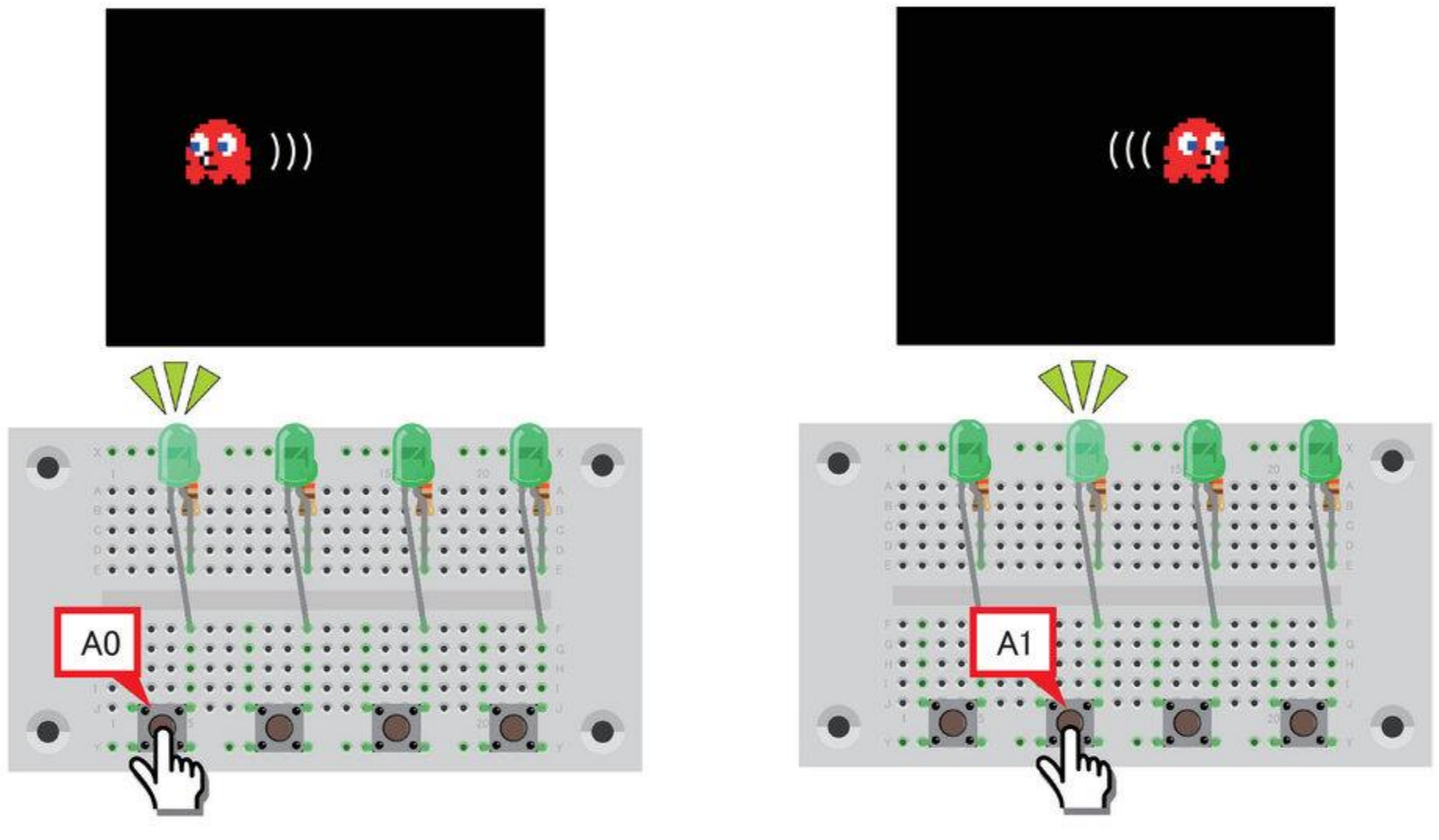


図1-1　キャラクターを左右に動かす

せっかくゲーム風のプログラムを製作するのでコントローラーを使いたいところですが、液晶ディスプレイと無線受信モジュールで使用ピンがかぶってしまうので、かわりにタクトスイッチを接続します。
画面上のキャラクターが、A0のスイッチを押したら左を向いて左に移動、A1のスイッチを押したら右を向いて右に移動するようにしようと思います。

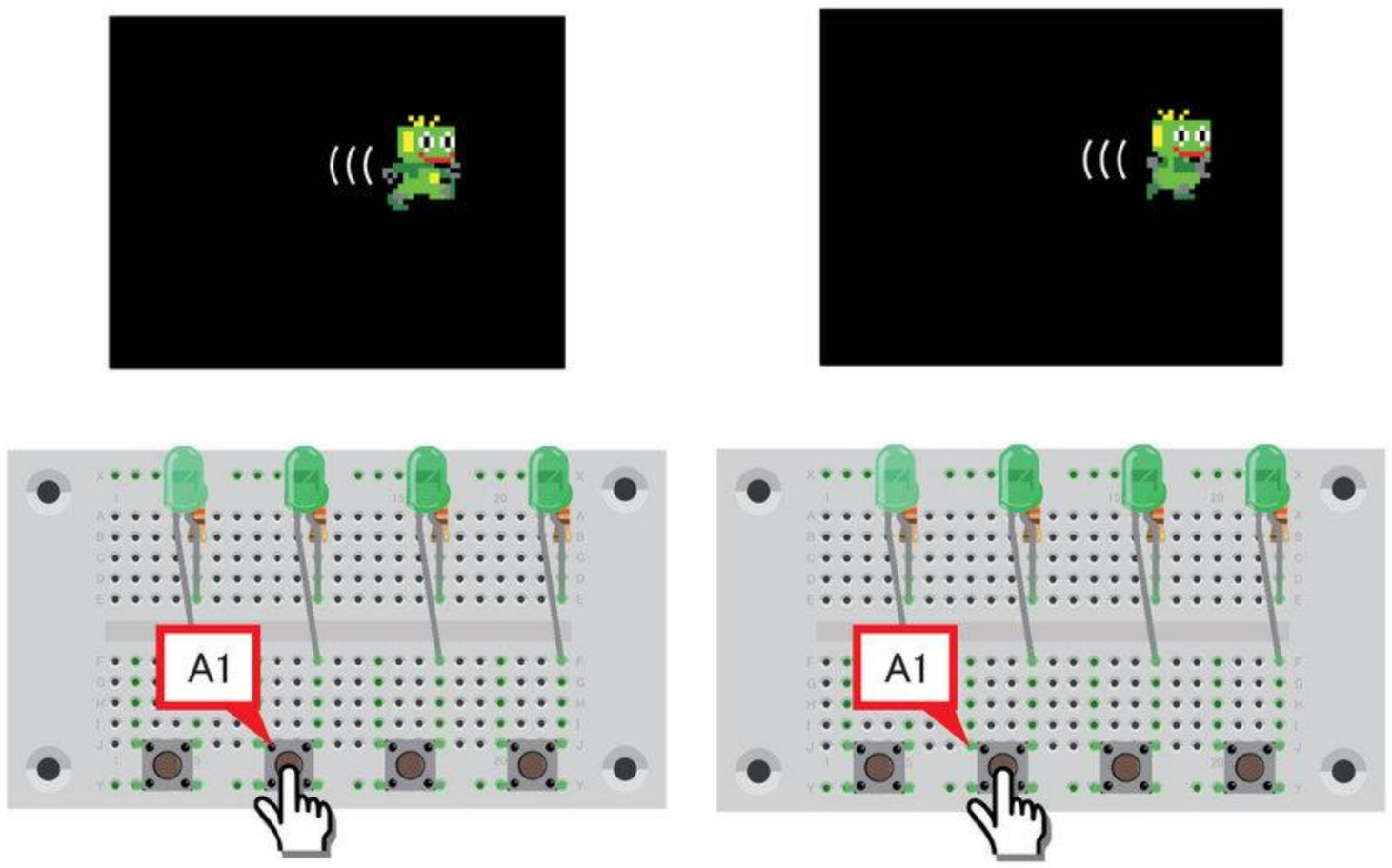


図1-2　キャラクターを左右に動かす（アニメーション入り）

また、それができるようになったら、今度は**図1-2**のように足踏(あしぶ)みのアニメーションを入れながら移動させることにも挑戦してみましょう。
まずは、これらの動きを切り分けて考えていきましょう。

やってみよう！

図1-1、**図1-2**のようなプログラムをつくるとき、前回製作した「液晶ディスプレイにキャラクターを表示する」以外にどのような機能が必要かな？
なるべく具体的にかいてみよう！

切り分けて考えることはできましたか？
大まかにあげると、以下のような処理(しょり)を追加する必要がありますね。

POINT

・キャラクターを左右反転で表示する。
・表示するキャラクターを切り替える。（アニメーションの場合）
・ボタンが押されるごとに、キャラクターの表示位置を変更する。

では、それぞれつくり方を考えていきましょう。

1.3. キャラクターの反転・拡大

まずは、キャラクターを反転して表示する方法を学びます。せっかくなので、左右反転だけでなく上下反転や90度回転、あるいは拡大して表示する方法も一緒に学んでいきましょう。

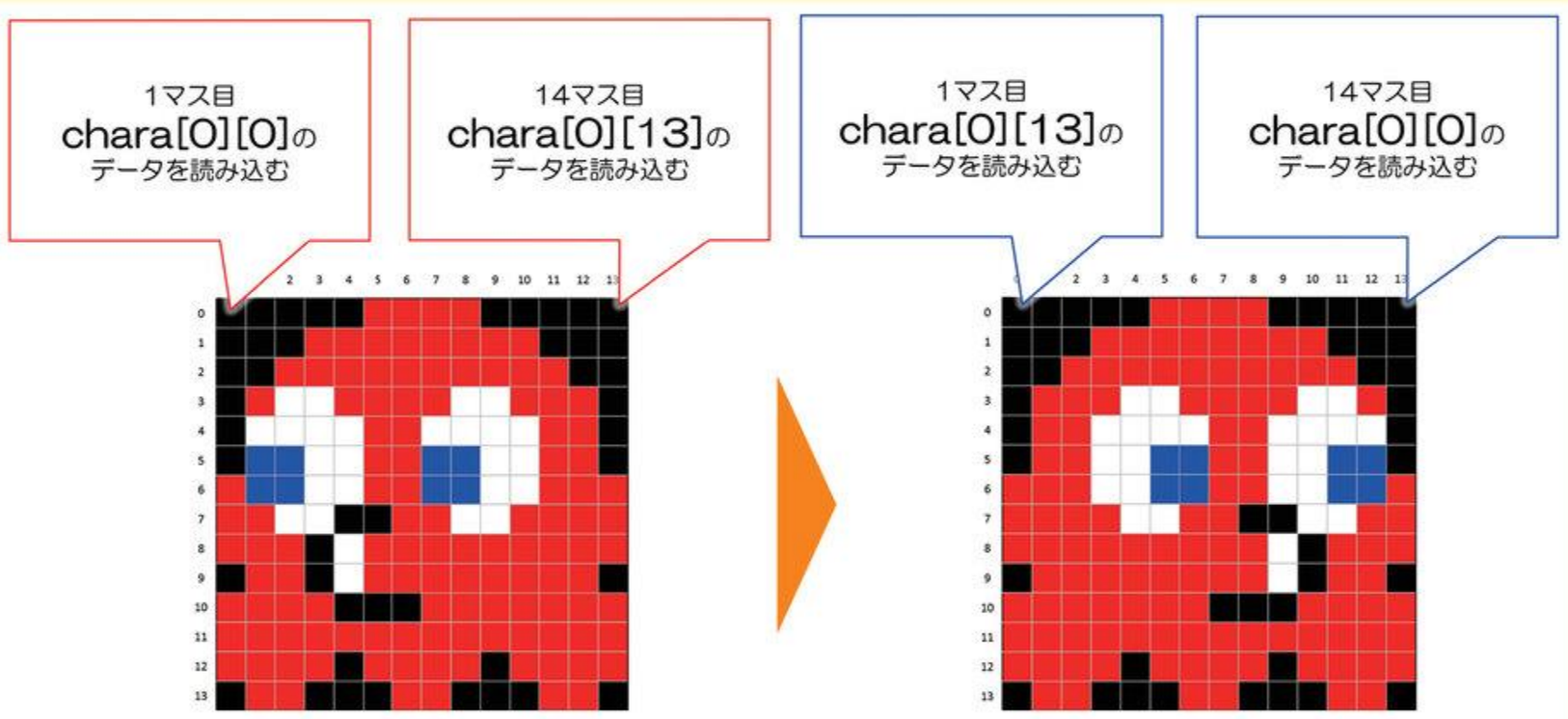
左右反転するからといって、キャラクターデータをかきかえるのは大変ですし、ミスの危険も高くなります。ここでは手を付ける部分を最小限に抑えるため、キャラクターデータ以外の部分を修正していきましょう。

ステップアップ

プログラム「Chara3」のキャラクターデータ以外の部分をかきかえ、左右反転のキャラクターを表示するようにしてみよう。

ヒント

下の図を参考にしてみよう。キャラクターデータの配列はそのままなので、読み込んでいく順番を逆転させればいいんだね。



解答例は以下のプログラムです。

プログラムの書き込み

RoboticsProfessorCourse3 > MagicItemLCD4 > Chara3RL

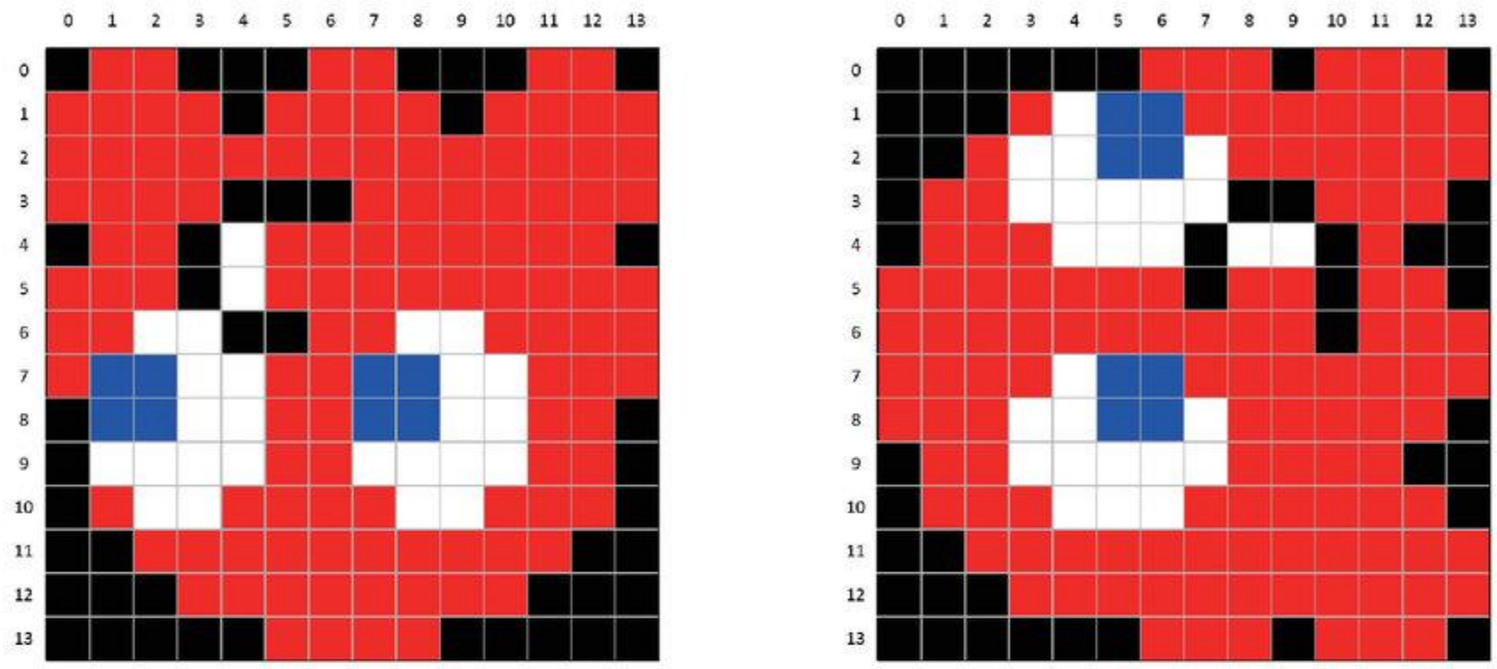
chara[y][sizeX - x - 1]とかきかえることで、変数xが0, 1, 2, …と増加していくのにあわせchara[0][13], chara[1][12], chara[2][11], …と、逆の順番でキャラデータを読み込んでいくようになります。

定数sizeXの値は14ですが、変数xは0 ～ 13の間でしか変化しないため、-1を追加することで調整しています。

講 なお別解として、xのfor文の条件式を13から1ずつ減少していくように書きかえても同じ処理にできますが、その場合は描画位置をpx - xと書きかえる処置も必要です。

チャレンジ課題

プログラム「Chara3RL」を参考にして、キャラクターを下の図のような状態でそれぞれ表示させてみよう！



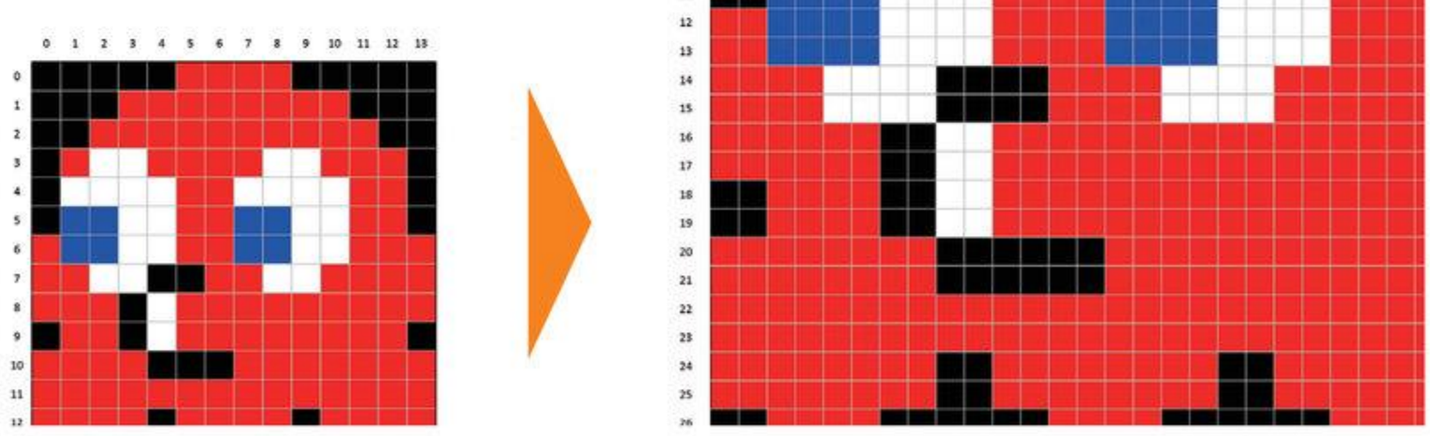
講

解答例はそれぞれ以下のプログラムです。
RoboticsProfessorCourse3 ＞ MagicItemLCD4　＞ Chara3UD
RoboticsProfessorCourse3 ＞ MagicItemLCD4　＞ Chara3ROLL

次は拡大です。ひらめきや柔軟（じゅうなん）な考え方が必要なので難しいですが、ぜひ挑戦してみてください。
次のページにヒントがあるので、必要なら読んでみましょう。

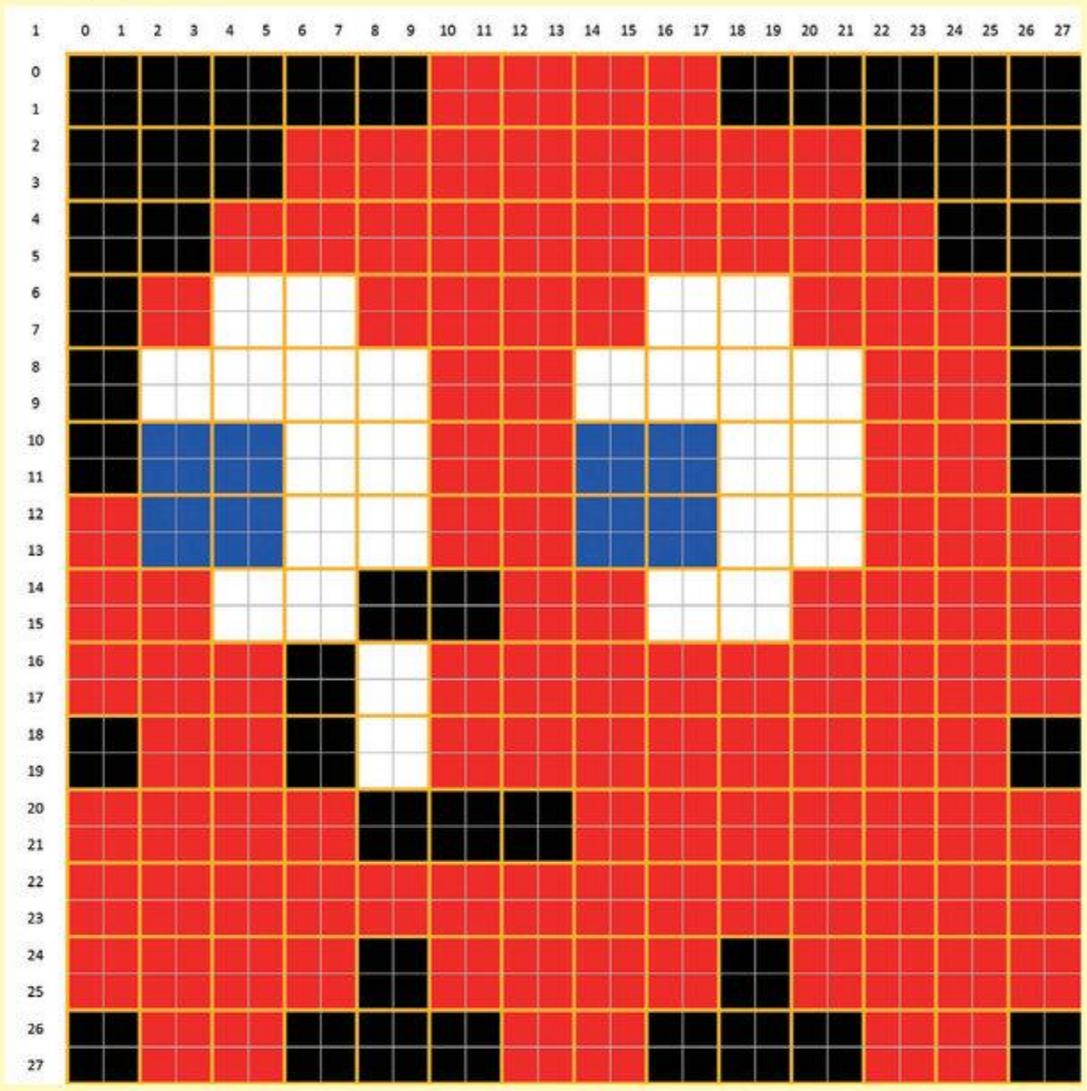
チャレンジ課題

プログラム「Chara3RL」を参考にして、キャラクターを縦（たて）横ともに2倍に拡大して表示させてみよう！



ヒント

縦横(たて)とも2倍ということは、もとは1マスだった部分が縦横(たて)2マスずつに拡大されるということだよね。
つまり、「点を並べていく」というよりは「正方形を並べていく」という考え方になるのではないかな？



講

解答例は以下のプログラムです。
RoboticsProfessorCourse3 ＞ MagicItemLCD4　＞ Chara3EX
正方形を並べるので、point(x, y)命令からrect(x, y, a, b)命令へ変更しています。
詳しくは第1回テキストを参照してください。
1マスずつ描画していた時と違い、x、yが1増加するごとに2マス先に正方形を描画しなければならないため、描画する位置はpx + 2 * xなどとなっています。うまくいかず悩む生徒が多いとは思いますが、試行錯誤してもらうことが目的の課題ですので、時間が許す限り主体的に取り組ませてあげてください。
なお、ディスプレイ範囲の許す限り拡大率は上げられます。「Chara3EX3」が3倍拡大の例です。

1.4.　キャラクターをタクトスイッチで動かしてみよう

まずは、液晶ディスプレイシールドと接続する回路をつくります。

図1-3のようにつくりましょう。つくり終えたら、緑色LEDとタクトスイッチの向きなどを点検しましょう。なお､緑色LEDは画像の上側（100Ω抵抗側）が＋極になります。また、抵抗は100Ω（茶黒茶金）を使用します。各パーツの位置が左右にずれると回路が正しく機能しないので注意しましょう。

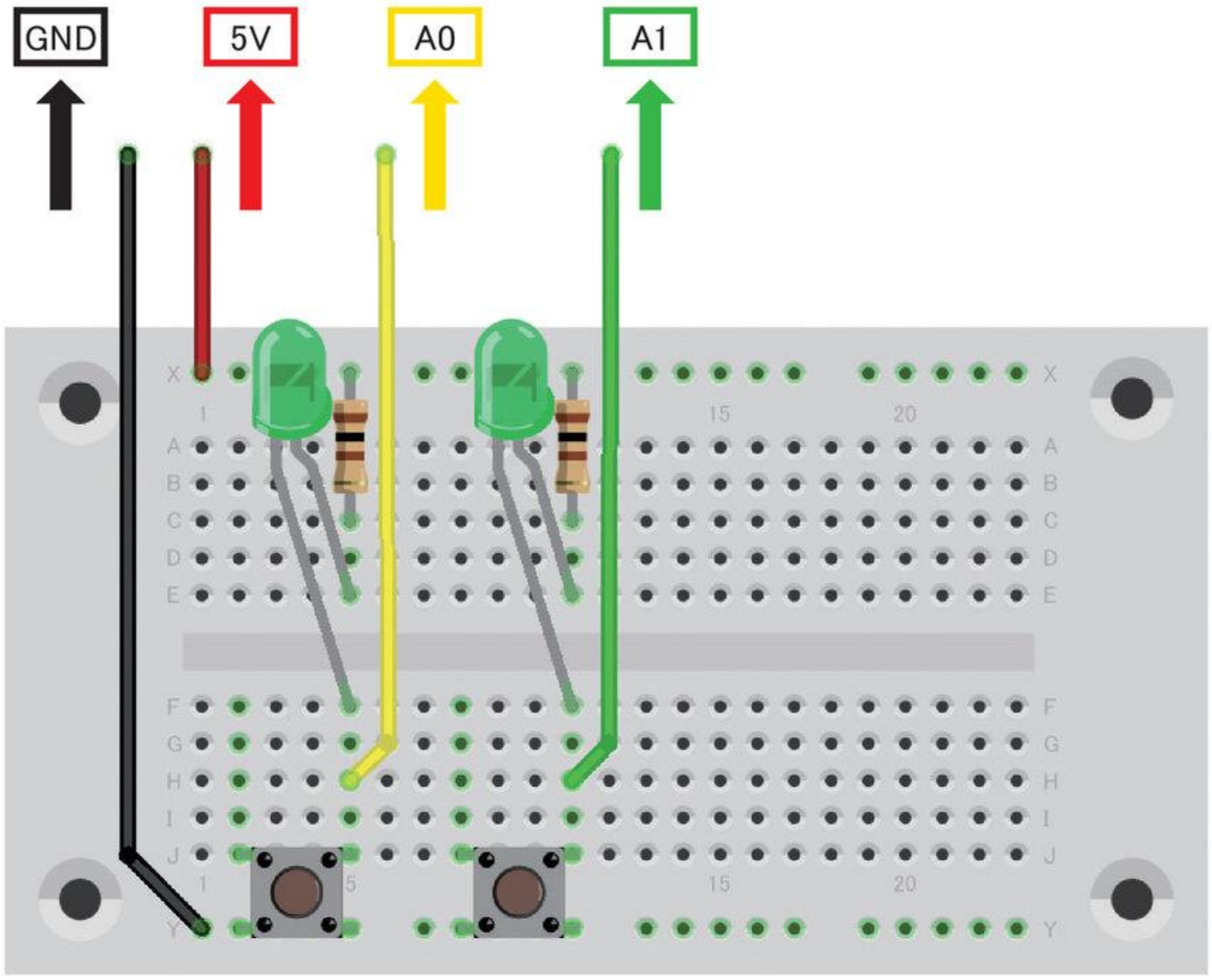


図1-3　液晶ディスプレイシールドと接続する回路

今度は、つくった回路のタクトスイッチを使って、キャラクターを動かしてみましょう。
かきかえていくプログラムとしては「Chara2」を使用しますが、キャラクターはこれまで登場したものの中から好きなものを選んだり、自分でオリジナルキャラをつくったりして差しかえても構いません。
まず、タクトスイッチを使用するため、以下のような宣言(せんげん)が必要です。`setup()`内に追加しておきましょう。

```
pinMode(A0, INPUT_PULLUP);
pinMode(A1, INPUT_PULLUP);
```

`INPUT_PULLUP`モードでタクトスイッチを使うと、スイッチがオンのときを判定するときには`if(digitalRead(A0) == LOW)`などとif文をかけばよいことになります。
では、プログラム製作にチャレンジしましょう。

ステップアップ

プログラム「Chara2」をかきかえ、以下のような動作にしてみよう！
`A0`のスイッチを押したら、左に動く
`A1`のスイッチを押したら、右に動く

ヒント

「左右に動く」ということは、1マスずつ左または右にキャラクターを表示すればいいということだね。どの数値を変更(へんこう)すればいいのかな？

講　移動したときに模様や線が残ってしまうことがありますが、この原因と対策は次のページで解説します。

できましたか。解答例は以下のプログラムです。

プログラムの書き込み

RoboticsProfessorCourse3 > MagicItemLCD4 > Chara5A

ボタンが押されるごとにキャラクターの表示位置px をかえればよい、ということに気付けましたか？

プログラム「Chara5A」より抜粋

```
if (digitalRead(A0) == LOW) px--;
if (digitalRead(A1) == LOW) px++;

for (int y = 0; y < sizeY; y++){          //縦方向のデータを描画する
    for (int x = 0; x < sizeX; x++){      //横方向のデータを描画する
        TFTscreen.stroke(B[chara[y][x]], G[chara[y][x]], R[chara[y][x]]);
                                          //色見本データを参照する
        TFTscreen.point(px + x, py + y); //データを描く
    }
}
```

ただ、ここに気づいてプログラムをかきかえられた人の中には、「キャラクターは動かせたけど、動いた後に模様や線が残ってしまう！」という人もいたのではないでしょうか。これはボタンを押す前にキャラクターが表示されていたマスが、特に何の処理もされずそのまま残されているためです。そのため、次のループに行く前にいったんキャラクター全体を黒で塗りつぶして消す処理が必要です。

プログラム「Chara5A」より抜粋

```
TFTscreen.stroke(0, 0, 0);
TFTscreen.fill(0, 0, 0);
TFTscreen.rect(px, py, sizeX, sizeY);      //データを描く
```

講

1行目ではこれから描画する線の色を、
2行目ではこれから描画する図形の中を塗りつぶす色を
指定しています。今回はどちらも黒（0,0,0）となります。
3行目は、1，2行目で指定された色で四角形を描くプログラムです。
キャラクターが描かれた同じ位置に、キャラクターと同じ大きさで描かれます。
3行のプログラムが実行されることで、キャラクターが黒で塗りつぶされます。

これで、まずは基本的な移動ができるようになりました。

あとはよりゲームらしくするため、はじめに目標にしていた動作を完成させましょう！

ステップアップ

プログラム「Chara5A」をかきかえ、左右に動くとき、キャラクターが進行方向を向くようにするようにしてみよう！

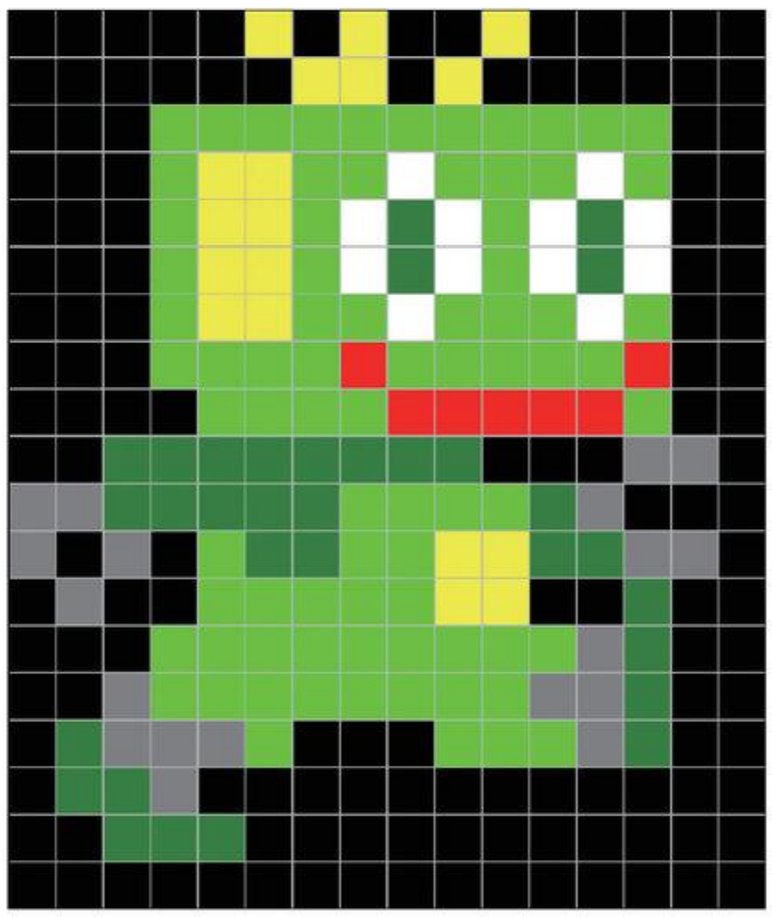
ヒント

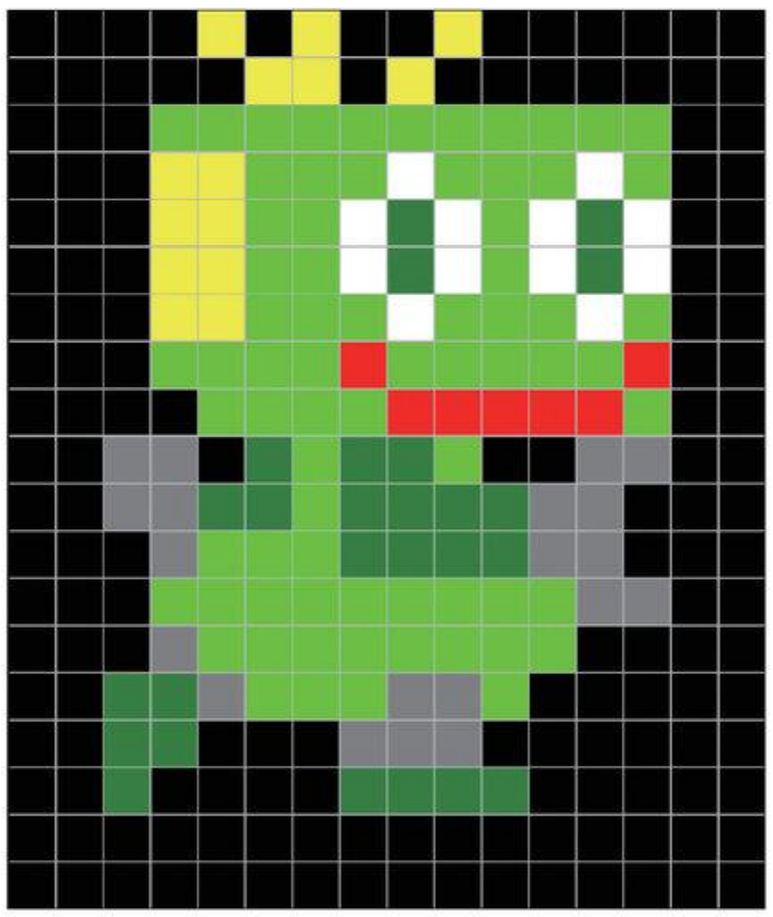
「左を向いたキャラ」と「右を向いたキャラ」の2通りのデータが必要になるね！

講　解答例は以下のプログラムです。
RoboticsProfessorCourse3 ＞ MagicItemLCD4　＞ Chara5B

チャレンジ課題

プログラム「Chara5A」をさらにかきかえ、キャラクターが移動するときに歩くようなアニメーションをするようにしてみよう！

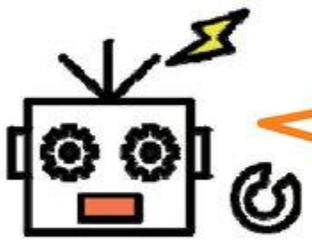




講　解答例は以下のプログラムです。
RoboticsProfessorCourse3 ＞ MagicItemLCD4　＞ Chara5C

２．まとめ（目安５分）

今回は、第1回～ 3回までの復習もかねながら、キャラクターデータのプログラムの編集を通して、データの扱いや動作処理（しょり）の方法を勉強しました。
一度だけでは覚えきれないほどの内容でしたが、くり返すことで、授業で行った以上のクオリティのキャラクターがつくれるかもしれません！
また、今回説明した「二次元配列」の変数を使うと、いろいろとわかりやすいプログラムをつくれることがわかったでしょうか？ これは、キャラクター表示だけではなく、ロボットの動作でも活用できるものです。
第3回でも説明しましたが、メモリーの使い方が本格的なプログラムをかく上で大切な要素になります。配列や関数の使い方次第（しだい）で、良いプログラムにも、悪いプログラムにも、なり得るということです。配列の使い方一つでも非常に効率の良いプログラムがかけるようになります。
これは、メモリーの節約にも非常に役立ちます。
次回は、プログラミングの考え方と、コンピューターグラフィックスについて、勉強します。



《次回必要なもの》

次回は、以下のパーツを持ってきてください。

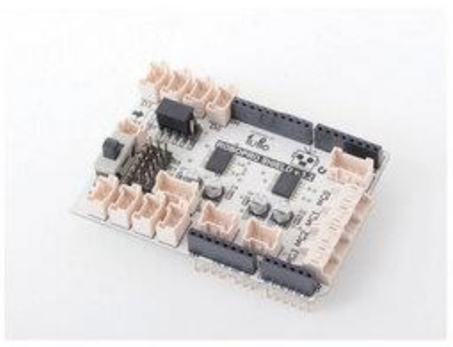
USB ケーブル 1	マイコンボード 1	ロボプロシールド 1	タッチセンサー 1
			
301 ブレッドボード 1	**ジャンパー線 65**	**タクトスイッチ 10**	**姿勢検出シールド 1**
			
液晶ディスプレイシールド 1			
			

図 2-0　次回必要なもの

講

○以下の理解度を確認してください。
・キャラクターを動かす
・二次元配列を学ぶ
・キャラクターをつくる
○次回は、「ゲームプログラムをのぞいてみる」を行います。