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I－1②

## 第4回

## CG（コンピューターグラフィックス）で遊ぶ

**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。

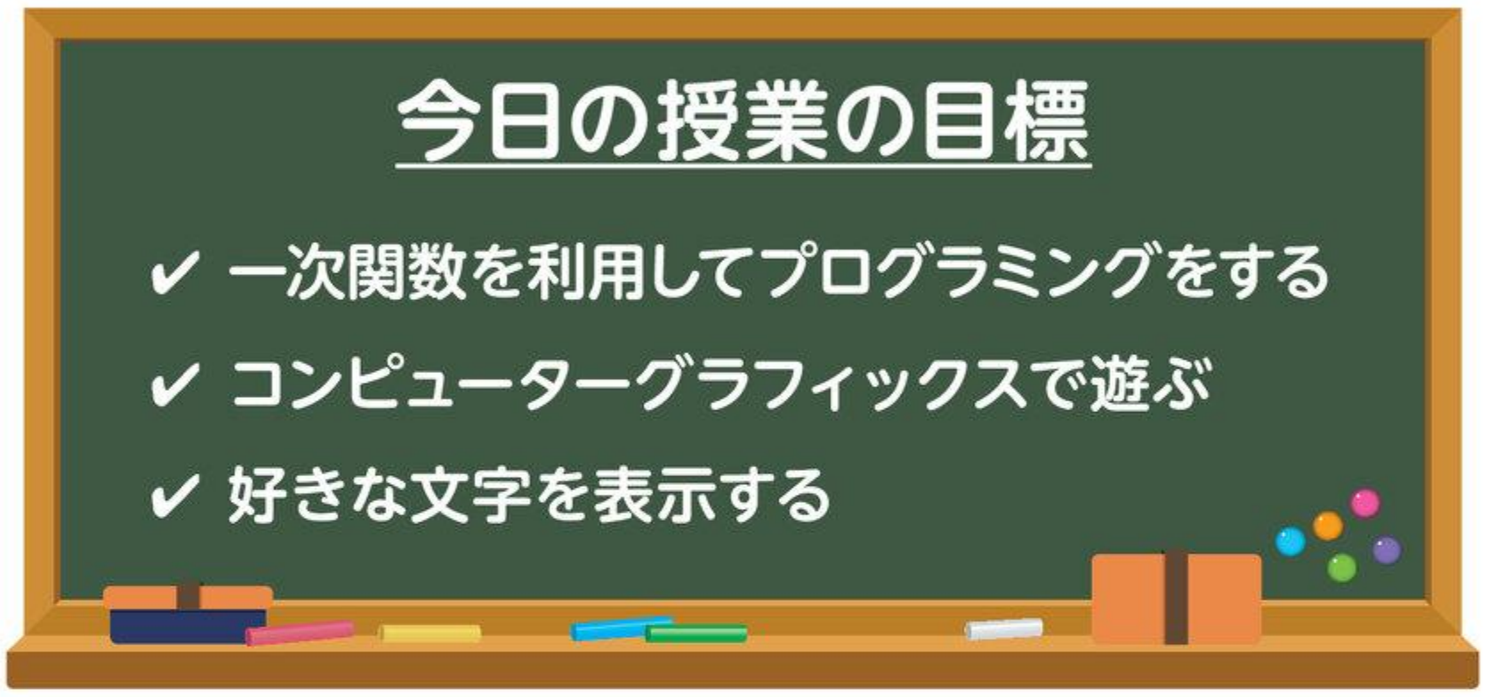
○　今回の目標をパネルで用意するか、黒板に予め書いておきます。

（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）

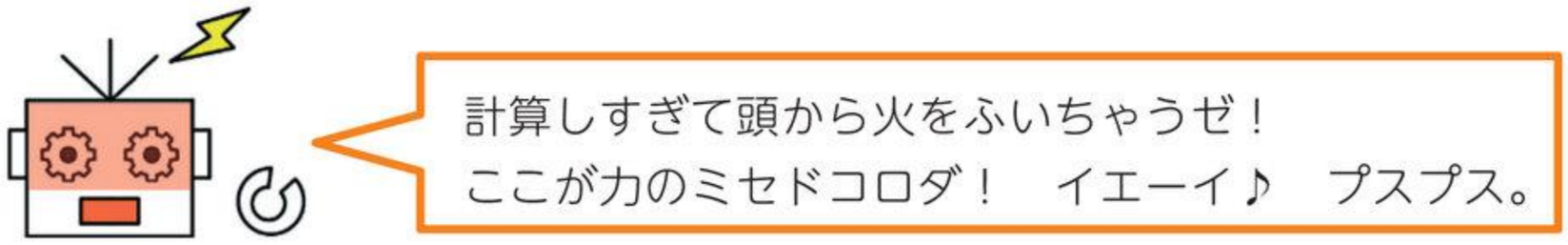
目安時間は授業時間 120 分のうち、休憩 10 分程度を取ることを想定しています。
生徒の進捗状況により、休憩時間などを調整して授業を行ってください。

# 0．CG（コンピューターグラフィックス）で遊ぶ（目安５分）

## 0.0. 「CG（コンピューターグラフィックス）で遊ぶ」でやること




今回は、CG の基礎（きそ）を勉強していきます！ ＣＧとは、コンピューターグラフィックスのことで、一般的（いっぱんてき）には、コンピューターに計算させて絵をかくことをいいます。
今までは、ドットを１つずつ打って、数字に変換（へんかん）して、プログラムに書き込（こ）んで、と面倒（めんどう）な作業をして、やっと１枚の絵ができていました。さらに、アニメーションさせようとすると、何枚も絵をかかねばいけませんでした。いわば、テレビでやっているアニメーションと同じ手法で１枚１枚手でかいて、パラパラ漫画（まんが）にするのと同じです。
今回は CG を使って、簡単にアニメーションをつくる方法を学びましょう！ 数学には「関数」という非常に便利なものがありますので、それをマイコンに計算してもらい、表示してしまうのです。
また、「文字コード」というものを使って、簡単にキャラクターや文字などをたくさん表示して遊んでみましょう！



## 0.1. 必要なもの

前回使用した、マイコンボード、ロボプロシールド、マトリクス LED シールド、マトリクス LED の順番に接続したマイコンユニットを、今回も使います。
今回は、タッチセンサーも、スピーカーも、コントローラーも使いませんので、必要のないものは外しておきましょう。

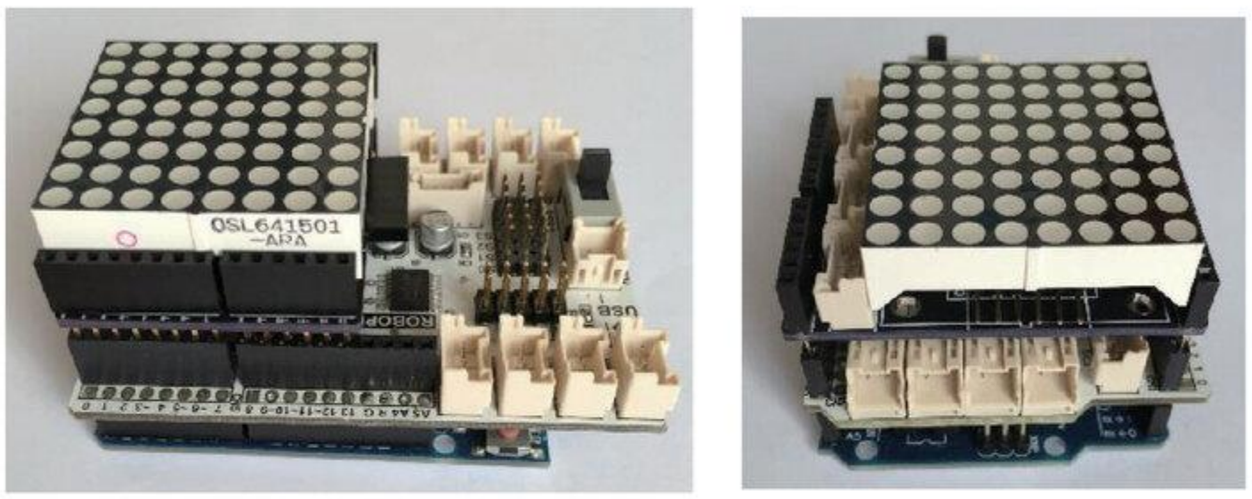


図 0-0　前回までに組み立てたもの

| USB ケーブル 1 | マイコンボード 1 | ロボプロシールド 1 | マトリクス LED シールド 1 |
|---|---|---|---|
|  |  |  |  |

| マトリクス LED 1 |
|---|
|  |

図 0-1　必要なもの

# 1．一次関数を使おう（目安 50 分）

## 1.0.　ラインを書いてみよう

このタームの第 1 回でも学習した、座標を使ってマトリクス LED に表示をさせるプログラムを実行していきます。まず最初のプログラムです。

講　ポート番号（COM）を必ず確認させます。

**プログラムの書き込み**

**RoboticsProfessorCourse1 > MagicItemA4 > MatrixGraph1**

実行結果：**図 1-0** のように、ななめのラインが表示される。

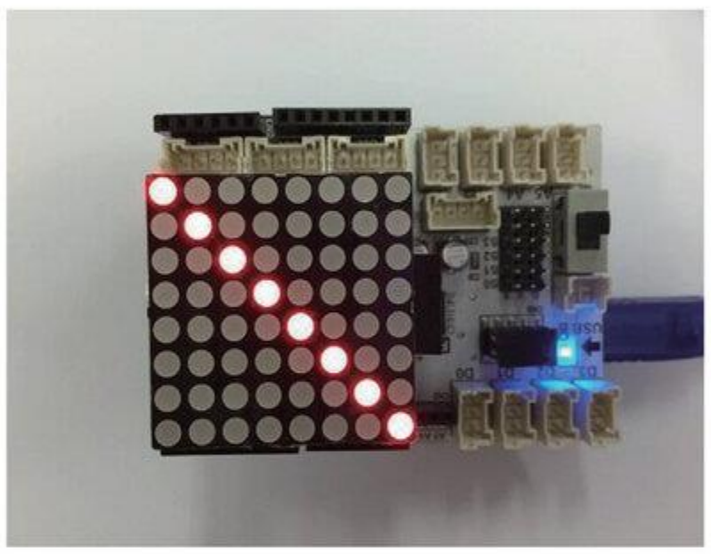


図 1-0　プログラム「MatrixGraph1」の実行結果

では次のプログラムも実行しましょう。

**プログラムの書き込み**

**RoboticsProfessorCourse1 > MagicItemA4 > MatrixGraph2**

**プログラムの書き込み**

**RoboticsProfessorCourse1 > MagicItemA4 > MatrixGraph3**

実行結果：どちらも「MatrixGraph1」と同じ結果になる。

今実行した 3 つのプログラムはどれも同じ実行結果になりましたが、プログラムは違います。プログラムの中身を確認すると、「MatrixGraph1 ⇒ 2 ⇒ 3」、という順にプログラムが簡単になっていることに気づきます。

「MatrixGraph1」「MatrixGraph2」では、第１回で登場した命令を再利用しています。おさらいしましょう。

まず、「MatrixGraph1」では、「特定の LED だけをつける／消す」という命令が使われていますね。

**命　　令　「myMatrix.write」**
実行結果：好きな場所の LED をつける、消す
使 い 方：myMatrix.write(0, 2, HIGH);
　　// x 座標が 0、y 座標が 2 の LED を点灯する

また、「MatrixGraph2」では、「2 進数」を利用してつける LED を指定しています。

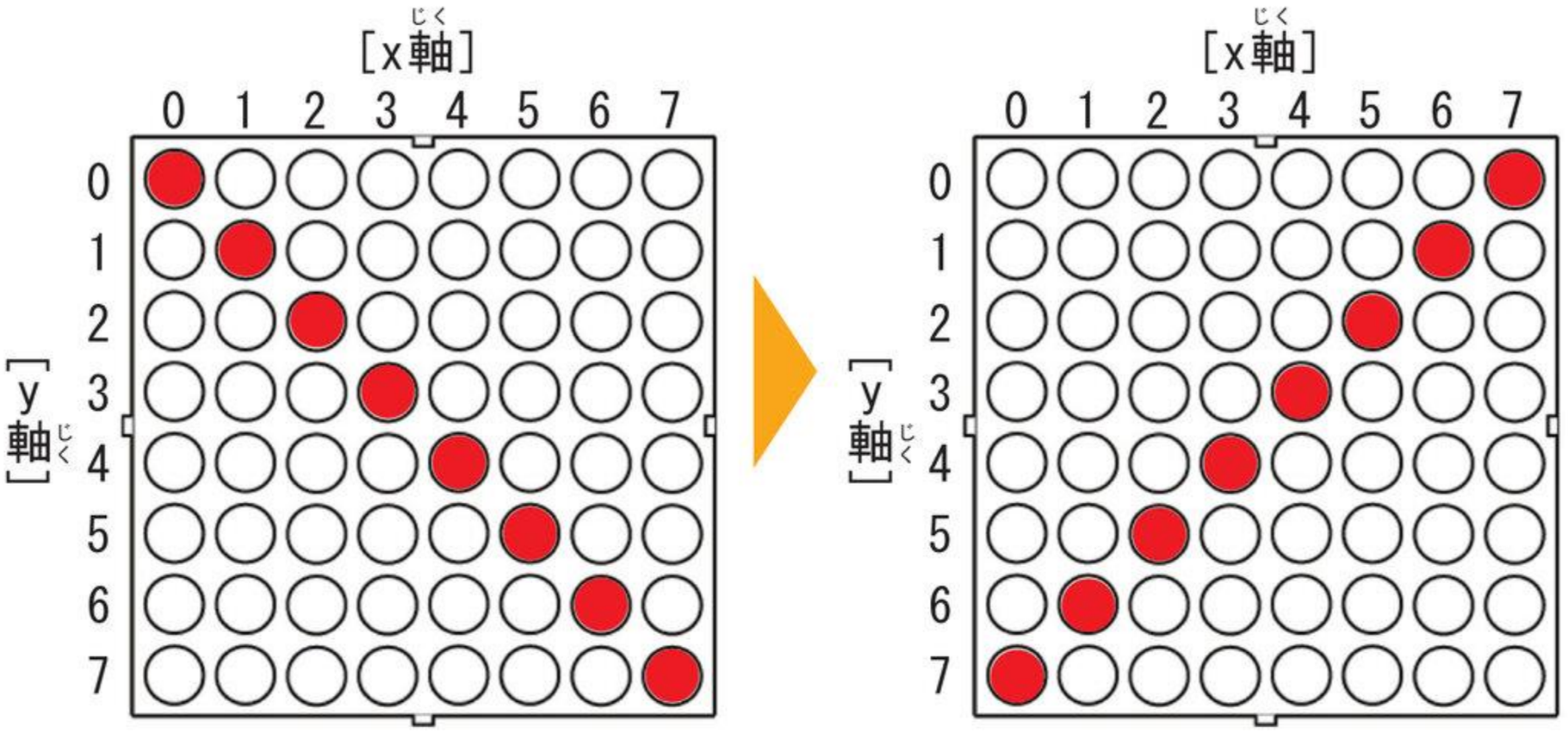
**プログラム「MatrixGraph2」より抜粋（ばっすい）**

```
Sprite pattern1 = Sprite(
    8, 8,
    B10000000,
    B01000000,
    B00100000,
    B00010000,
    B00001000,
    B00000100,
    B00000010,
    B00000001
    );
```

この 2 つはわかりやすい命令ではありますが、別の形に書きかえるのはちょっと大変です。ためしにやってみましょう。

## やってみよう！

プログラム「MatrixGraph1」/「MatrixGraph2」の内容をそれぞれ変更して、マトリクス LED のラインを右図のように変えてみよう。2 つのプログラム、両方でやってみてね。




講　解答例は巻末に記載します。

では「MatrixGraph3」に移りましょう。
よく見てみると、使われているのはこれまでにも登場した「for 文」や、「MatrixGraph1」と同じ myMatrix.write です。しかし、これまでのプログラムよりずっとシンプルにまとまっていますね。

### プログラム「MatrixGraph3」より抜粋

```
void loop(){
    for(x = 0; x < 8; x++){
        y = x;                          // これが一次関数!!
        myMatrix.write(x, y, HIGH);     // LEDを点灯
    }
```

つける LED の座標を、x と y という 2 つの変数で指定していますね。
for 文を見れば、変数 x は 0、1、2、…、6、7 と変化していくことがわかります。
しかし、変数 y の値を決めているのはどこでしょうか。これが、今回の重要ポイントになります。

## プログラム「MatrixGraph3」より抜粋（ばっすい）

```
y = x;                                    // これが一次関数!!
```

変数 y の値を決めているのは、この部分です。 x と y が等号（イコール）で結ばれている、つまり「変数 x と変数 y は同じ値になる」ということです。
たとえば、x の値が 0 のときは y の値も 0 になるので、(0, 0) の LED が点灯します。
x の値が 1 になれば、それにあわせて y の値も 1 になりますね。よって (1, 1) も点灯します。
for 文を使ってくり返せば、(0, 0) から (7, 7) まで、ななめ一直線の LED を点灯させることができますね。

この式をいろいろ変形してみると、y の値の決まり方が変化していきますね。ためしてみましょう。

## ステップアップ

プログラム「MatrixGraph3」を書きかえ、前のページの「やってみよう」のように、表示される線の向きを反転させてみよう！

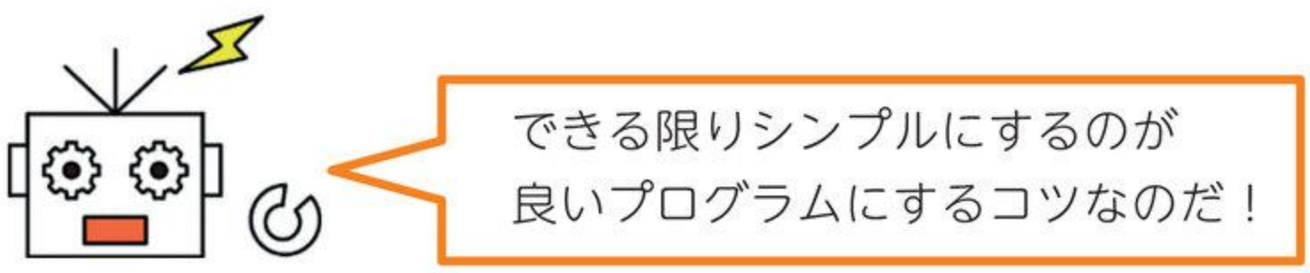
### ヒント

x が 0 のときは y が 7、x が 1 のときは y が 6、…となればいいんだね。
y = xをふくむ式　という形で書くとして、どのような式をつくればいいか考えてみよう！

できましたか？
y = x; を y = 7 - x; に変更（へんこう）すればよいですね！ y = -x + 7; でも問題ありません。

ここで大事なのは、y = x も y = 7 - x も、「x の値を決めたとき、y の値も自動的に決まるようにしたい！」という目的で書かれていることです。
このように、「2 つの変数のうち、片方の値が決まればもう片方も決まる」という関係があるとき、「関数」ということばを使います。
y の値が x の値をもとに決まるなら、「y は x の関数である」などと言えるわけです。

特に、今回扱う y = x や y = 7 - x のような関係を「一次関数」といいます。



# 1.1. 一次関数をマスターしよう

## 1）一次関数をかいてみよう

まずは、一次関数を実際に手でかいてみましょう。今回は一次関数 y=x+4 を考えてみます。
はじめに、x 軸の値を何点か適当に取って、関数に代入します。代入とは、式の中の文字に数字を置きかえて計算することです。今回の場合は x に数字を置きかえて計算してみます。
たとえば、y=x+4 に x=1 を代入すると、y=1+4 となり y=5 となります。
つまり、**表 1-0** のように x が 1 のときは y が 5 であることがわかりました。

表 1-0 「y=x+4」の表

| x | 0 | 1 | 2 | 3 | 4 | … |
|---|---|---|---|---|---|---|
| y | 4 | 5 | 6 | 7 | 8 | … |

**やってみよう！**

**表 1-0** に、残りの x の値を代入したときの y の値を書き込もう。

では次に、上の表の結果を**図 1-1** にプロット（点を打っていくこと）します。
たとえば、先ほどの x=1 、y=5 の結果をプロットすると、**図 1-1** のようになります。

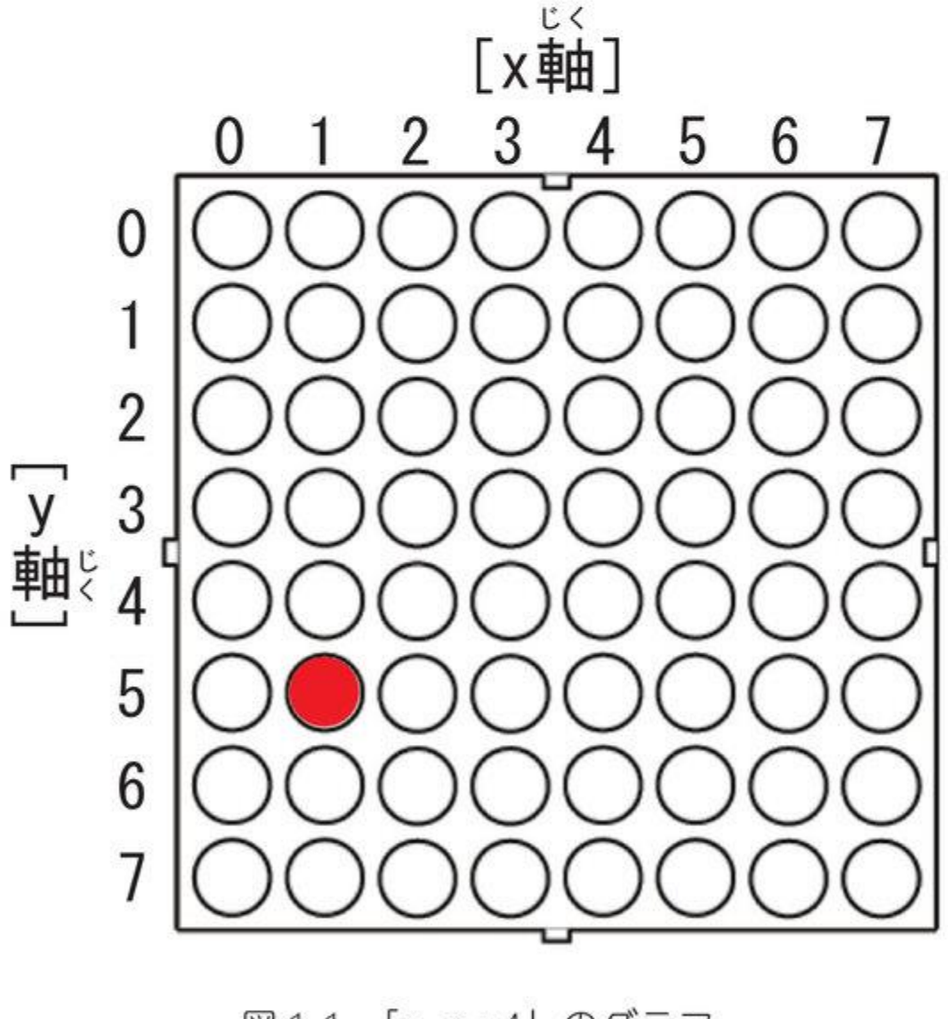



図 1-1 「y=x+4」のグラフ

**やってみよう！**

**図 1-1** の残りの部分もプロットしてみよう。

講　解答例は巻末に記載します。

うまくできたでしょうか？ 今つくったものが、「y=x+4」の関数のグラフです。
このように関数はグラフで表すこともできるのです！

## やってみよう！

プログラム「MatrixGraph3」の中にある y=x; を y=x+4; に変更（へんこう）してみよう。

### プログラム「MatrixGraph3」より抜粋（ばっすい）

```
void loop(){
    for(x = 0; x < 8; x++){
        y = x ;                          // これが一次関数！！
        myMatrix.write(x, y, HIGH);  // LEDを点灯
    }
    delay(100);
    myMatrix.clear();
}
```

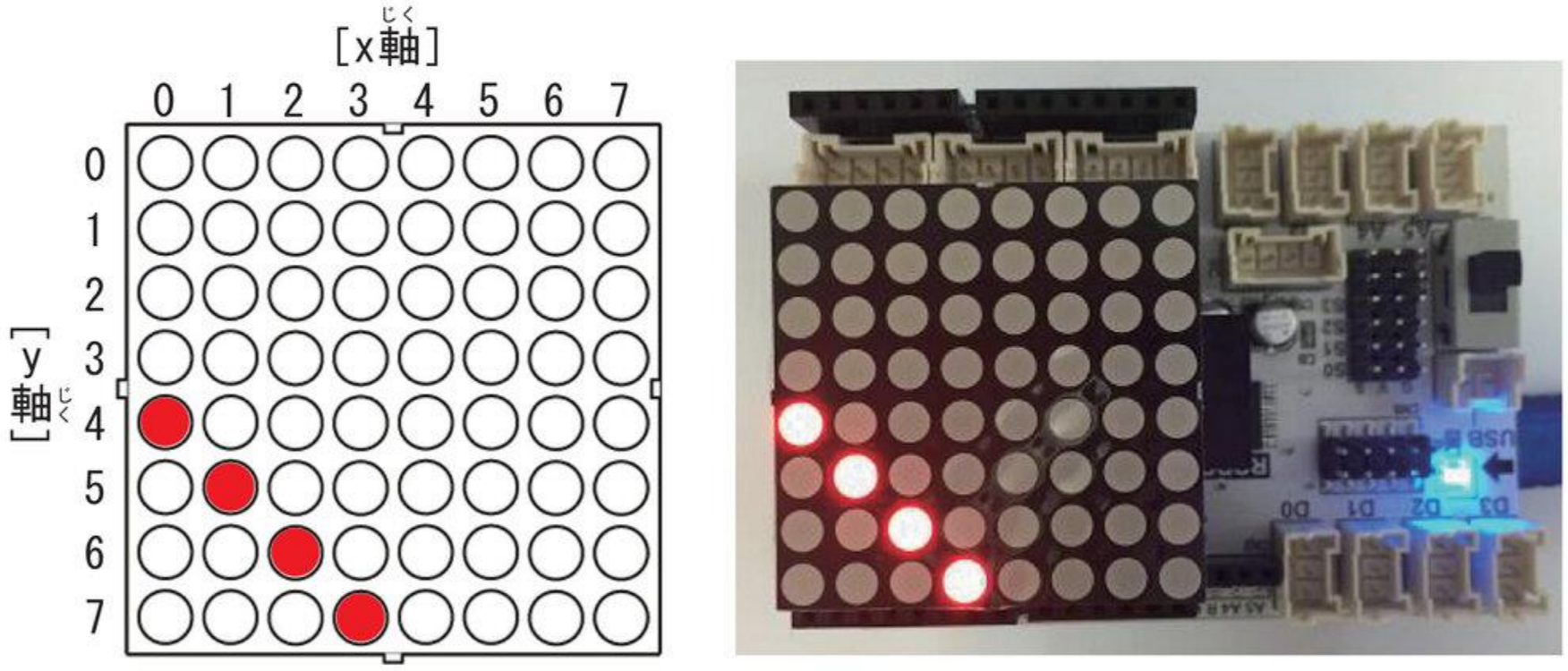


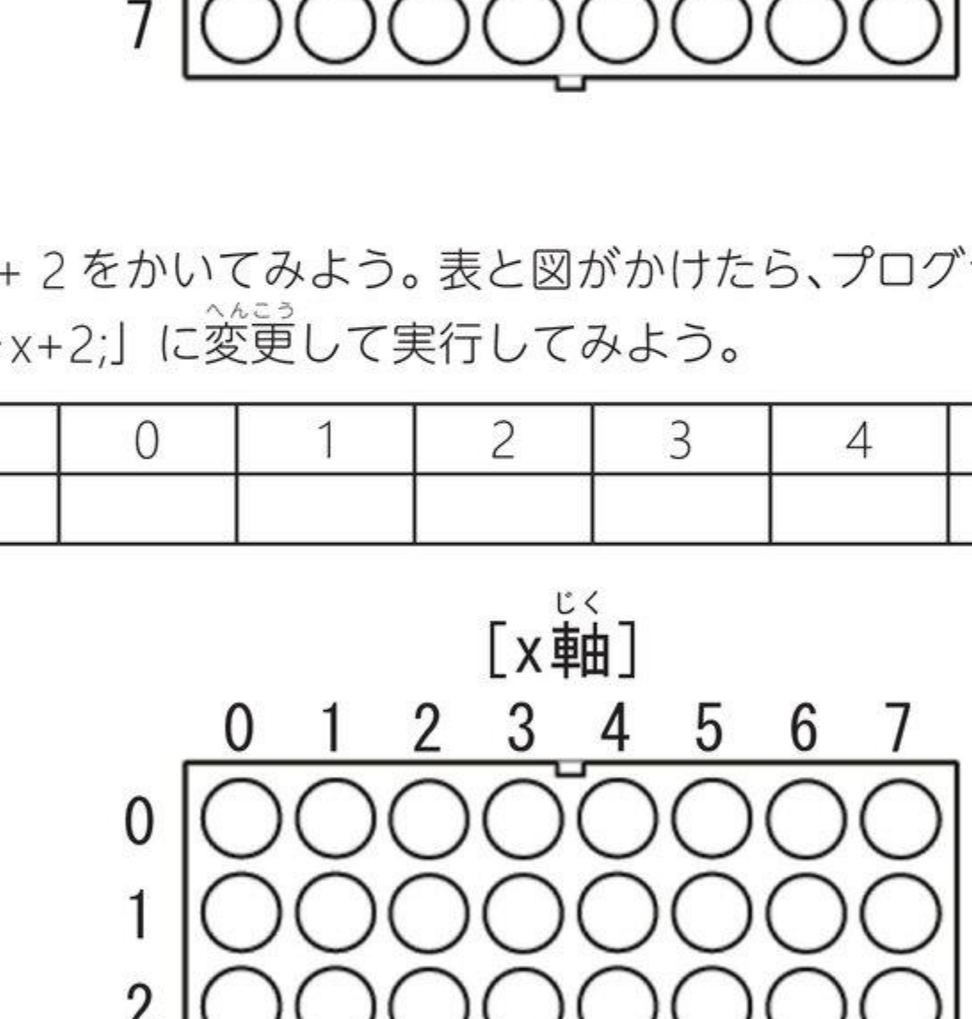

図 1-2 「y=x+4」のときのマトリクス LED の表示

さて、**図 1-2** と同じような形になりましたか？
今回は一次関数という一番簡単な関数なので、形も単純ですが、もっと複雑な形を持つものもあります。では、もう少し関数に慣れてみましょう！

## やってみよう！

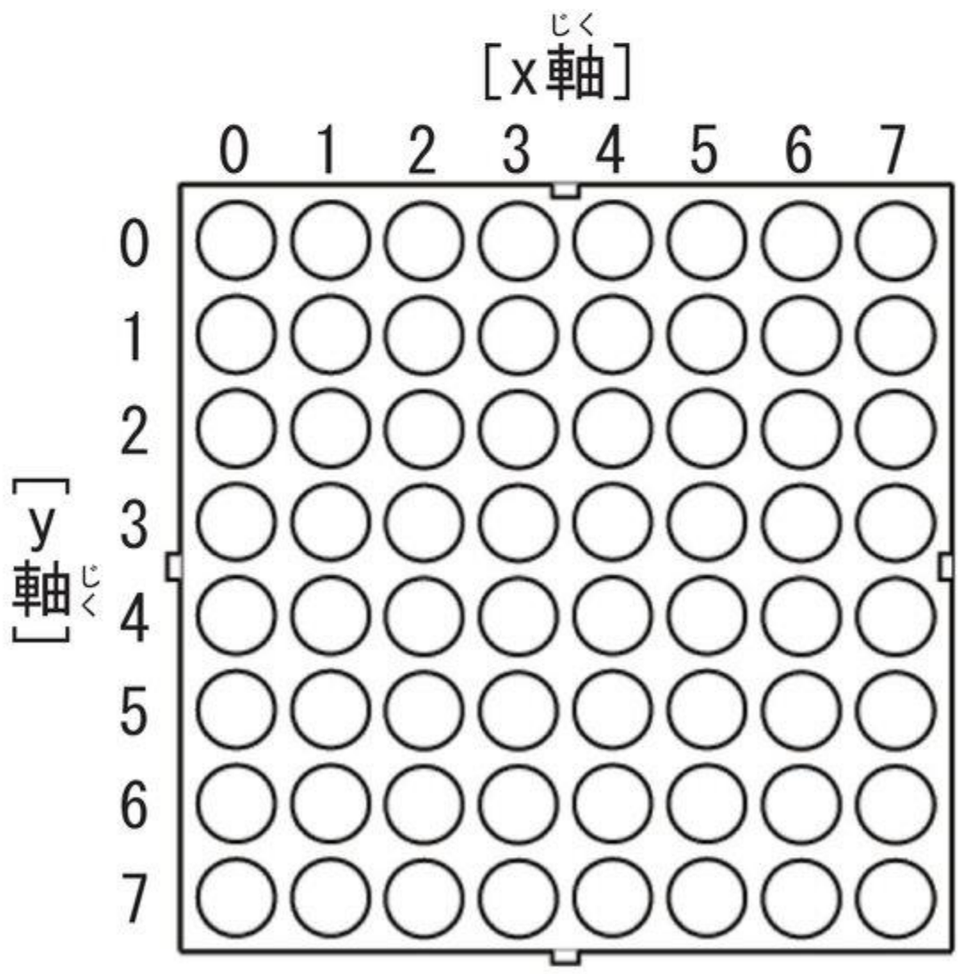
① 関数 y = 2 × x をかいてみよう。表と図がかけたら、プログラム「MatrixGraph3」の関数を「y=2*x;」に変更(へんこう)して実行してみよう。

| x | 0 | 1 | 2 | 3 | 4 | … |
|---|---|---|---|---|---|---|
| y | | | | | | … |




② 関数 y = 3 × x + 2 をかいてみよう。表と図がかけたら、プログラム「MatrixGraph3」の関数を「y=3*x+2;」に変更(へんこう)して実行してみよう。

| x | 0 | 1 | 2 | 3 | 4 | … |
|---|---|---|---|---|---|---|
| y | | | | | | … |




講　解答例は巻末に記載します。

## コラム　一次関数、二次関数について

1つ100円の商品を買うときに、「買う個数」を決めれば「代金」も計算できるようになりますよね。このとき、「代金」は「買う個数」の関数である、と言えます（代金から買う個数を求めることもできるので、「買う個数」は「代金」の関数である、とも言えます）。
「たかし君の身長は150㎝です。体重は何kgでしょうか？」ときかれても計算できません。これは、たかし君の身長と体重が関数ではないからです。

今回登場した関数は、どれもyが一定のペースで変化しています。xを1ずつふやしていくとき、「y = x」や「y = x + 4」であればyはつねに1ずつ、「y = 2 * x」であれば2ずつ、「y = 3 * x + 2」であれば3ずつふえていきました。関数の中でも、このように変数が一定のペースで変化していくものを「一次関数」とよぶのです。
マトリクスLEDに表示されたラインを見るとわかりますが、一次関数はグラフが直線になる、という性質があります。また、その式はかならず「y = ●× x + ■」という形になります（y = xだけちがう形に見えますが、数字を省略せずに書けば「y = 1× x + 0」という式なので、れっきとした一次関数です）。ちなみに、●に入る値を「変化の割合」や「傾き」、■に入る値を「切片」とよびます。

なお、たとえば正方形の「一辺の長さ」と「面積」はたがいに計算できるので関数ですが、一辺の長さを1cm、2cm、3cm、4cm、…とふやしていくと面積は1㎠、4㎠、9㎠、16㎠、…となるので、変化のペースは一定ではないですよね。これは「一次関数」ではなく、「二次関数」という種類の関数だからです。このようにyの変化のしかたによって関数にも種類があります。みなさんも、数学を学んでいく中でたくさんの関数と出会っていくことでしょう。

## 2）一次関数で遊ぼう

では、いよいよ関数を使って遊んでみましょう！ 次のプログラムを実行してください。

### プログラムの書き込み

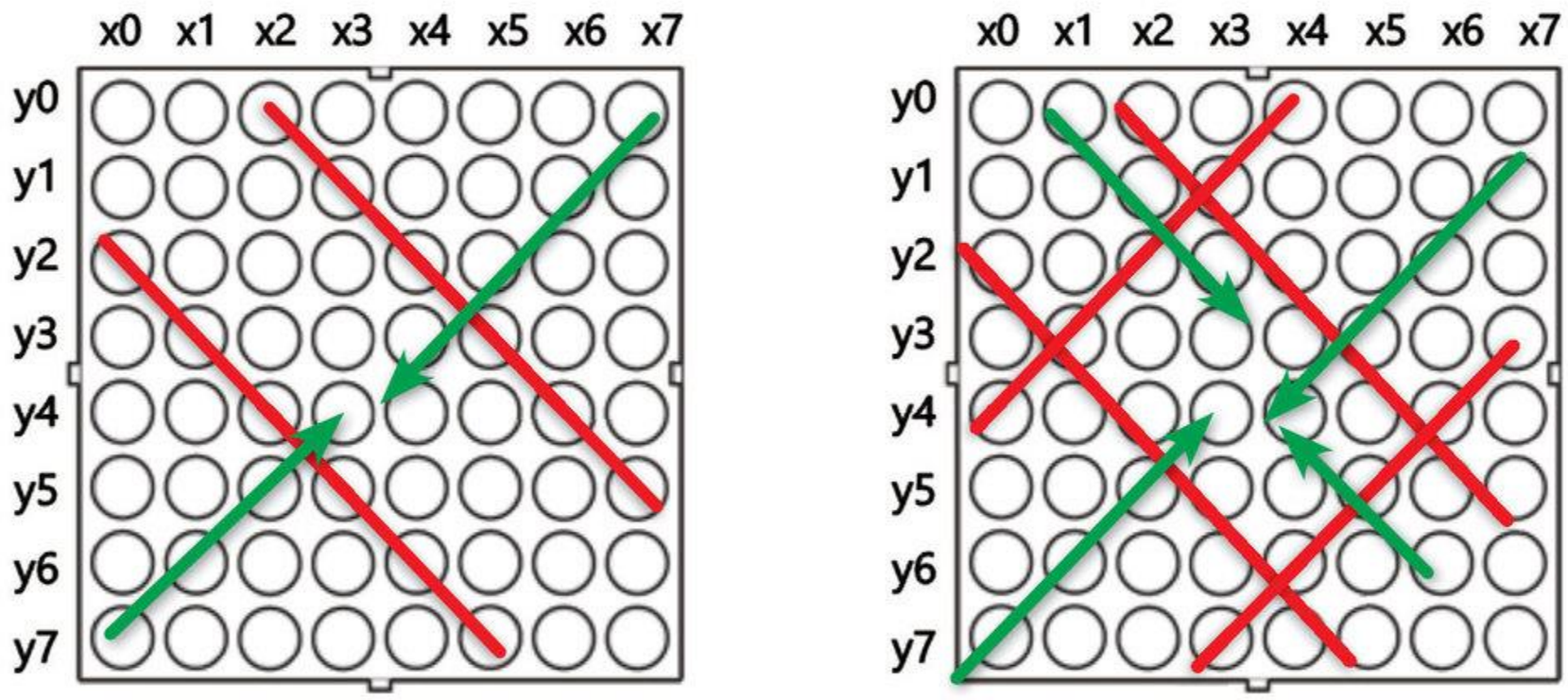
**RoboticsProfessorCourse1 ＞ MagicItemA4 ＞ MatrixGraph4**

実行結果：ななめのラインがマトリクスＬＥＤの片方のすみからもう片方のすみまで移動する。

これは一次関数の切片（せっぺん）（「y ＝ a × x ＋ b」のｂの値）を連続的に変化させて、そのたびにグラフをかくことで、一次関数のグラフがあたかも動いているかのように見せているのです。

### チャレンジ課題

プログラム「MatrixGraph4」を変更（へんこう）して、同じように波を２つにしたもの、４つにしたものをつくってみよう。左図は１行、右図は３行足せばできるよ。




講

解答プログラムはそれぞれ以下となります。
RoboticsProfessorCourse1 ＞ MagicItemA4 ＞ MatrixGraphChallenge1
RoboticsProfessorCourse1 ＞ MagicItemA4 ＞ MatrixGraphChallenge2
解答例は巻末に記載します。

# 2．CG（コンピューターグラフィックス）を使ってみる（目安 35 分）

## 2.0.　line 命令を使おう

今度は次のプログラムを実行してください。

**プログラムの書き込み**
**RoboticsProfessorCourse1 > MagicItemA4 > MatrixGraph5**

実行結果：ななめのラインが表示される。
ラインが表示されるという結果だけを見るとこれまでと同じですが、プログラムの中身が異なります。

**プログラム「MatrixGraph5」より抜粋（ばっすい）**

```
myMatrix.line(x1, y1, x2, y2);  // 座標(x1,y1)から座標(x2,y2)へ線を引く
```

ここで新しく出てきた命令「line」について説明します。

**命　　令「line」**
実行結果：(x1,y1) から (x2,y2) へ線を引く
使 い 方：myMatrix.line(1, 0, 6, 7);  // (1, 0) と (6, 7) をつなぐよう直線を引く

**やってみよう！**

次の 2 つのプログラムを実行してみよう。プログラムの実行結果も書いてみてね。

**プログラムの書き込み**
**RoboticsProfessorCourse1 > MagicItemA4 > MatrixLine1**

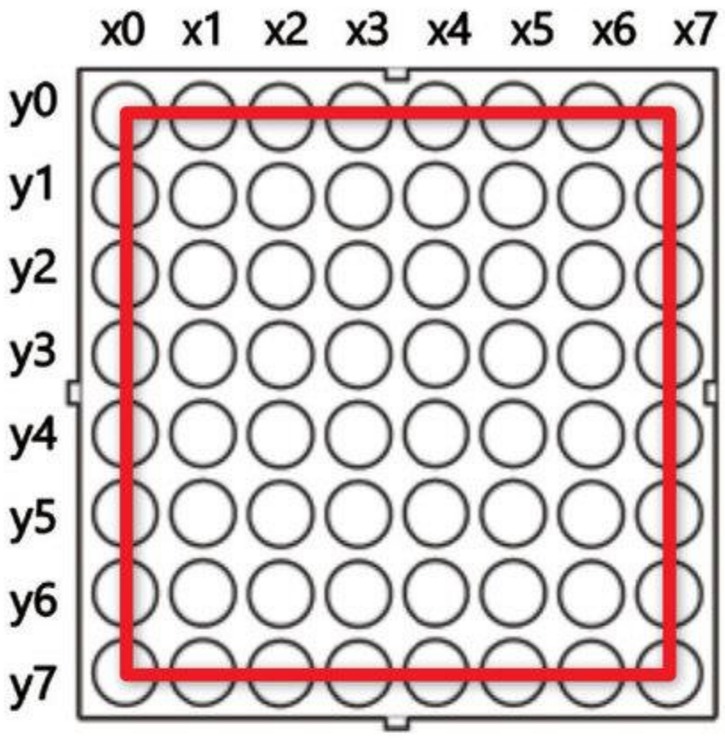
実行結果：十字が表示された。

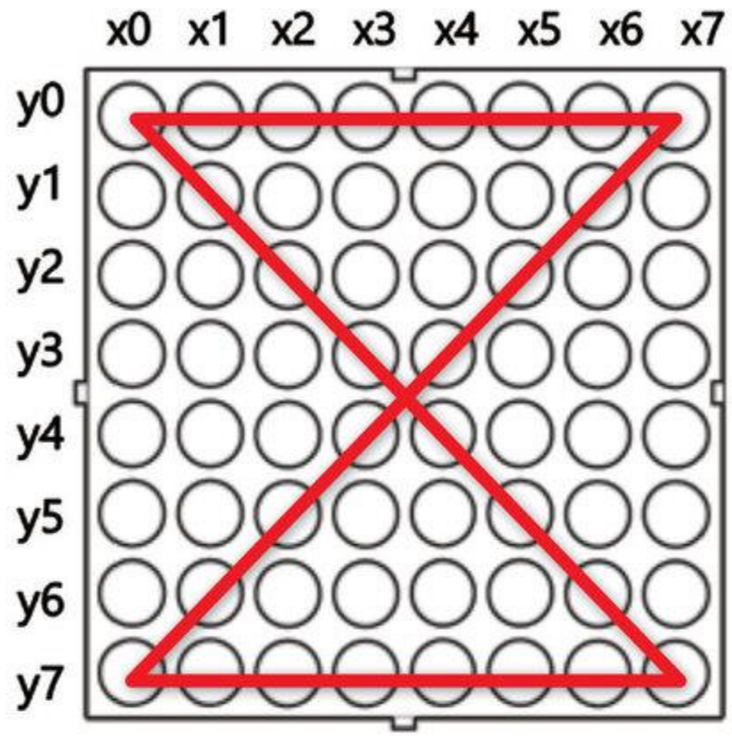
**プログラムの書き込み**
**RoboticsProfessorCourse1 > MagicItemA4 > MatrixLine2**

実行結果：三角形が表示された。

## ステップアップ

プログラム「MatrixLine2」を変更（へんこう）して、四角と砂時計をかいてみよう。




### ヒント

line 命令は 4 つ。いずれも使用する座標は (0,0) (0,7) (7,0) (7,7) の 4 種類だよ。

講

解答プログラムはそれぞれ以下となります。
RoboticsProfessorCourse1 ＞ MagicItemA4 ＞ MatrixLineChallenge1
RoboticsProfessorCourse1 ＞ MagicItemA4 ＞ MatrixLineChallenge2

## 2.1. ラインを自由に移動させてみよう

表示させているだけではつまらないので、ここでは、プログラムを使って自由にラインを動かしてみましょう。まずは、以下のプログラムを実行してください。

### プログラムの書き込み

**RoboticsProfessorCourse1 > MagicItemA4 > MatrixLineMove1**

実行結果：ラインが上から下へ動く。
[y1 座標 ] と [y2 座標 ] の値を変数にすることによって、ラインを動かしているわけです。

### チャレンジ課題

プログラム「MatrixLineMove1」を変更（へんこう）して、１本目のラインが上から下に、２本目のラインが左から右に、と同時に動くプログラムをつくってみよう。

講

解答プログラムは以下となります。
RoboticsProfessorCourse1 > MagicItemA4 > MatrixLineMoveChallenge1

次に、プログラム「MatrixLineMove1」の黄色の部分を以下のように変更してみましょう。

### プログラム「MatrixLineMove1」より抜粋

```
for(int y = 0; y <= 7; y++){
    y1 = y;
    y2 = y;
    myMatrix.line(x1,y1,x2,y2);   // 座標(x1,y1)から座標(x2,y2)へ線を引く
    delay(100);
    myMatrix.clear();             // マトリクスLEDの表示を消す
}
```

### プログラム「MatrixLineMove1」の変更

```
for(int y = 0; y <= 7; y++)
{
    y1 = y;
    y2 = 7 - y;
    myMatrix.line(x1,y1,x2,y2);   // 座標(x1,y1)から座標(x2,y2)へ線を引く
    delay(100);
    myMatrix.clear();             // マトリクスLEDの表示を消す
}
```

実行結果：ラインがマトリクスＬＥＤの画面の中心を軸に反時計まわりに半回転する。
y2 座標は変数「y」の値の変化にあわせて変化しますが、数値の変わり方が、変更前とは異なることがわかります。for 文がくり返し実行されるたびに座標が変化するので、ラインが動くわけです。

### チャレンジ課題

プログラム「MatrixLineMove1」を拡張して、ラインを一周回転できるように変えよう。
さらに、ラインを逆回転させてみよう。

**ヒント**
for 文をコピー＆ペーストして、条件式を変更しよう。

ラインをアニメーションにすることができたでしょうか？
ラインを組み合わせて、もっと複雑な絵をアニメーションにすることもできます！

講 解答プログラムはそれぞれ以下となります。
RoboticsProfessorCourse1 ＞ MagicItemA4 ＞ MatrixLineTurn
RoboticsProfessorCourse1 ＞ MagicItemA4 ＞ MatrixLineTurnR
解答例は巻末に記載します。

# 3．好きな文字を表示する（目安 15 分）

## 3.0.　マトリクス LED に文字を表示しよう

### 1）アルファベットの表示

次に、ロボットに意思表示のための言葉を使う能力をあたえましょう。ここでは、文字をマトリクス LED に表示します！ 以下のプログラムを実行してください。

**プログラムの書き込み**

**RoboticsProfessorCourse1 > MagicItemA4 > textScroll**

実行結果：「Hello! World!!(^o^)/~」などの文字がスクロールする。

**やってみよう！**

1．プログラム「text Scroll」の中を以下のように書きかえてみよう。

```
const char text[] = " Hello! World!!(^o^)/~ "; // 表示するテキスト内容
```

▼

```
const char text[] = " HUMAN ROBOPRO "; // 表示するテキスト内容
```

2．1と同じように黄色の部分に好きな文字を入れてみよう。

## 2）ひらがなの表示

さて、アルファベットは表示することができました。では、ひらがなはどのように表示するのでしょうか？ プログラムを見てみましょう。

**プログラム「textScroll」より抜粋（ばっすい）**

```
const char texth[] = {0x93,0xd3,0xab,0xa1,0xaf,0x2c,0xcd,0xbc,0xb7,0xcd,
'!','!',NULL};  // 表示するテキスト内容（ひらがな）
```

気づいたでしょうか？ 上記の黄色の部分が、ひらがなを表示するデータになっています。この英数字の並びが「文字コード」といわれ、マイコンが理解できる 16 進数です。

表 3-0　文字コード表

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | a | b | c | d | e | f |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0x00 | | | | | | | | | | | | | | | | |
| 0x10 | | | | | | | | | | | | | | | | |
| 0x20 | | ! | " | # | $ | % | & | | ( | ) | * | + | , | - | . | / |
| 0x30 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | : | ; | < | = | > | ? |
| 0x40 | @ | A | B | C | D | E | F | G | H | I | J | K | L | M | N | O |
| 0x50 | P | Q | R | S | T | U | V | W | X | Y | Z | [ | \ | ] | ^ | _ |
| 0x60 | ` | a | b | c | d | e | f | g | h | i | j | K | l | m | n | o |
| 0x70 | p | q | r | s | t | u | v | w | x | y | z | { | \| | } | ~ | |
| 0x80 | | ぁ | あ | ぃ | い | ぅ | う | ぇ | え | ぉ | お | か | が | き | ぎ | く |
| 0x90 | ぐ | け | げ | こ | ご | さ | ざ | し | じ | す | ず | せ | ぜ | そ | ぞ | た |
| 0xa0 | だ | ち | ぢ | っ | つ | づ | て | で | と | ど | な | に | ぬ | ね | の | は |
| 0xb0 | ば | ぱ | ひ | び | ぴ | ふ | ぶ | ぷ | へ | べ | ぺ | ほ | ぼ | ぽ | ま | み |
| 0xc0 | む | め | も | ゃ | や | ゅ | ゆ | ょ | よ | ら | り | る | れ | ろ | ゎ | わ |
| 0xd0 | ゐ | ゑ | を | ん | ゔ | ゜ | | | | | | | | | | |
| 0xe0 | | | | | | | | | | | | | | | | |
| 0xf0 | | | | | | | | | | | | | | | | |

この表の見方を説明します。たとえば、「こんにちは」の「こ」を見てみると、表の左に「0x90」とあります。そして上を見ると「3」となっています。ですから「こ」の文字コードは「0x93」となるわけです。同じように、「ん」の文字コードも見ると、「0xd3」ということがわかりますね。マイコンは文字を読めません。そこで、マイコンは文字に対して、一番得意な数字をつけて、わかるようにしているのです。それが「文字コード」とよばれるものです。これはキャラクターのデータでも、音楽のデータでも同じです。実際にこの表でもグレーの部分には、絵のデータが入っています。

では、ここで再びプログラムの続きを見てみましょう。アルファベットに関しては、「'!'」と同じように「'A'」という表現も使えます（シングルクォーテーションではさむ）。

最後の NULL はおまじないと思って入れておいてください。行の最後を示すものになります。

## 3.1. プログラム例

次のプログラムを実行してください。文字コードを全て入れたプログラムです。

### プログラムの書き込み

**RoboticsProfessorCourse1 ＞ MagicItemA4 ＞ textTest**

実行結果：文字コード登録されている文字と絵柄（えがら）がすべて表示される。
いくつか、隠（かく）しキャラが登場しますが、どうやったら表示できるか、考えてみましょう。

### プログラム「textTest」より抜粋（ばっすい）

```
void loop(){
    for(uint8_t tcode = 0x00; tcode < 0xff; tcode++){ // 文字コード0x00から0xffまで
        myMatrix.putch(0, 0, tcode);
                                                      // フォントのコードを入れて
                                                      //   一文字ずつ表示
        delay(200);
    }
}
```

for 文を使って、0x00 ～ 0xff までを 0.2 秒おきに表示しています。
これは、**表 3-0** を端から端まで表示することになります。
では、表の空欄のところでは何が表示されるでしょうか？

講　白い空欄に何が表示されるかは、第５回の冒頭で確認します。

# 4．まとめ（目安５分）

かけ足でＣＧの基礎を勉強しましたが、理解できましたか？ マイコンは計算するために生まれてきたので、計算がたくさん出てきました。マイコンの一つの得意分野なので、ぜひとも数学をマスターして、もっと使いこなせるようになりましょう。マイコンには、まだまだ潜在能力がありますよ！


**≪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。

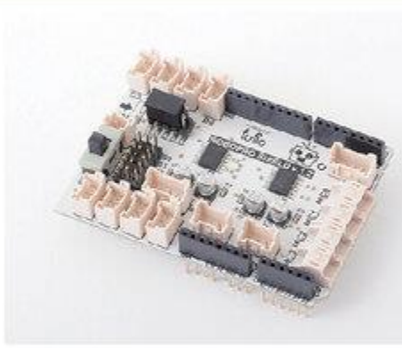
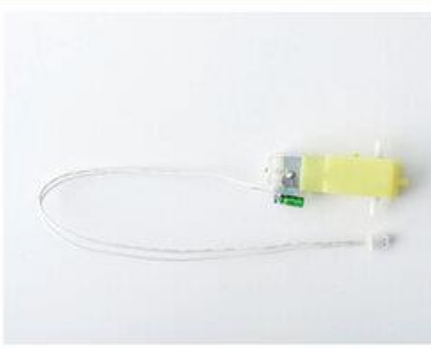
| | | | |
|---|---|---|---|
| USB ケーブル 1 | マイコンボード 1 | ロボプロシールド 1 | ギアドモーター 2 |
|  |  |  |  |
| タッチセンサー 2 | マトリクス LED シールド 1 | マトリクス LED 1 | スピーカー 1 |
|  |  |  |  |

図 4-0　次回必要なもの

# 不思議アイテムⅠ−1②　第４回 CG（コンピューターグラフィックス）で遊ぶ　講師用

講

○　以下の理解度を確認します。
　・一次関数を利用してプログラミングをする
　・コンピュータグラフィックスで遊ぶ
　・好きな文字を表示する
○　次回のテーマは「ラーメンタイマーをつくる」であることを告知します。

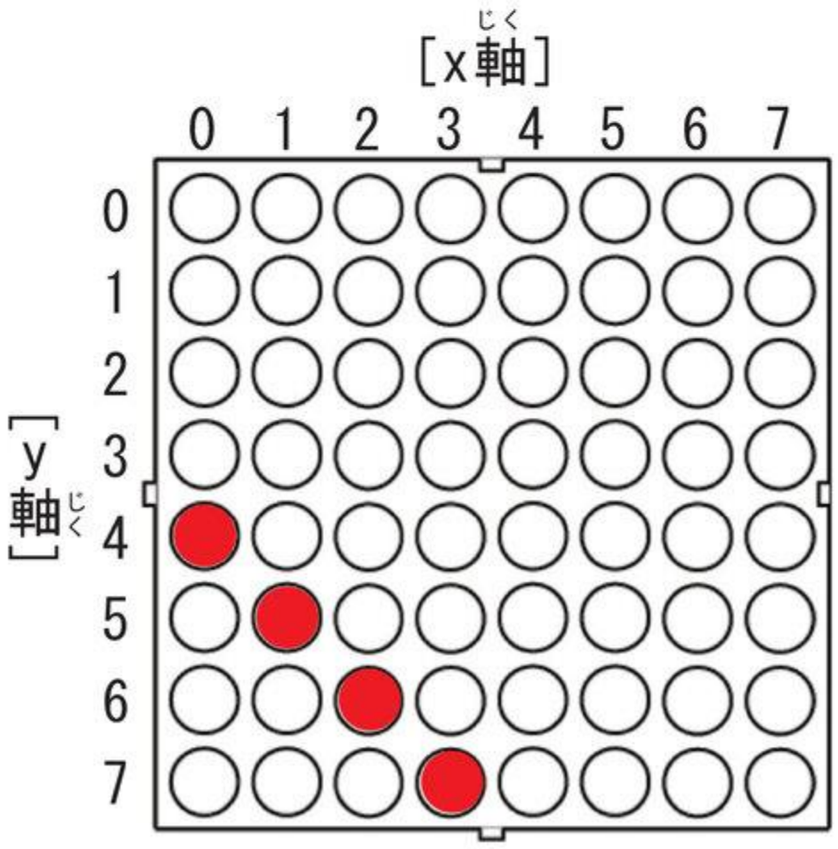
## P.5　やってみよう！　MatrixGraph1　解答例（voidloop以下を抜粋）

```
void loop()
{
 // マトリクスLEDのLEDを指定して点灯する
  myMatrix.write(0, 7, HIGH);
  myMatrix.write(1, 6, HIGH);
  myMatrix.write(2, 5, HIGH);
  myMatrix.write(3, 4, HIGH);
  myMatrix.write(4, 3, HIGH);
  myMatrix.write(5, 2, HIGH);
  myMatrix.write(6, 1, HIGH);
  myMatrix.write(7, 0, HIGH);
}
```

## P.5　やってみよう！　MatrixGraph2　解答例（Sprite以下を抜粋）

```
Sprite pattern1 = Sprite(
    8, 8,
    B00000001,
    B00000010,
    B00000100,
    B00001000,
    B00010000,
    B00100000,
    B01000000,
    B10000000
);
```

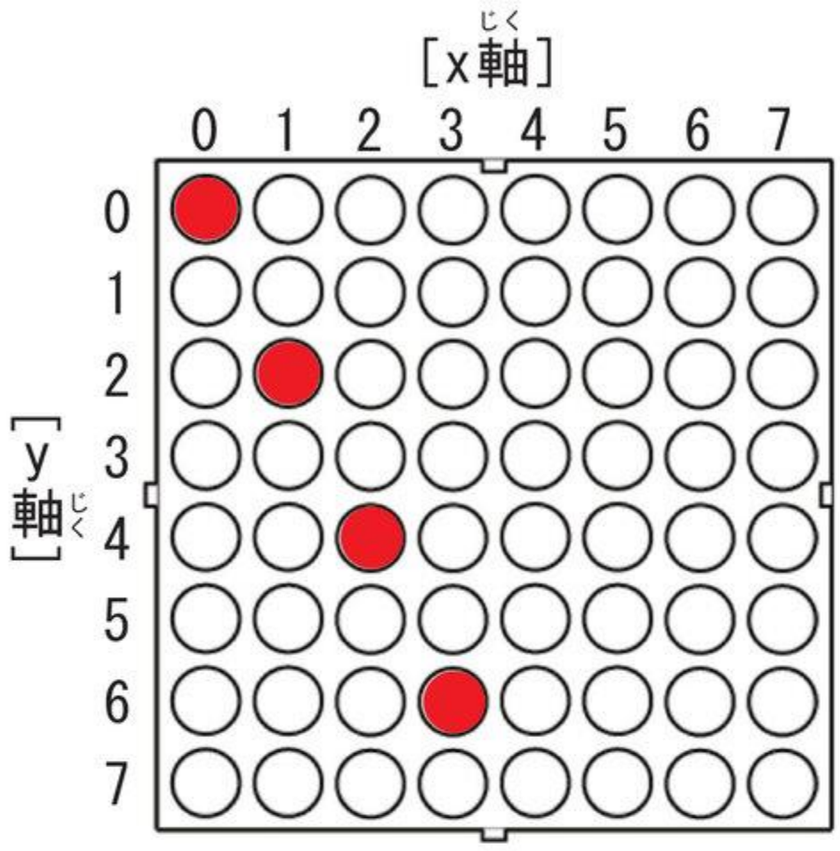
## P.7　やってみよう！　解答例



## P.9　やってみよう！　解答例

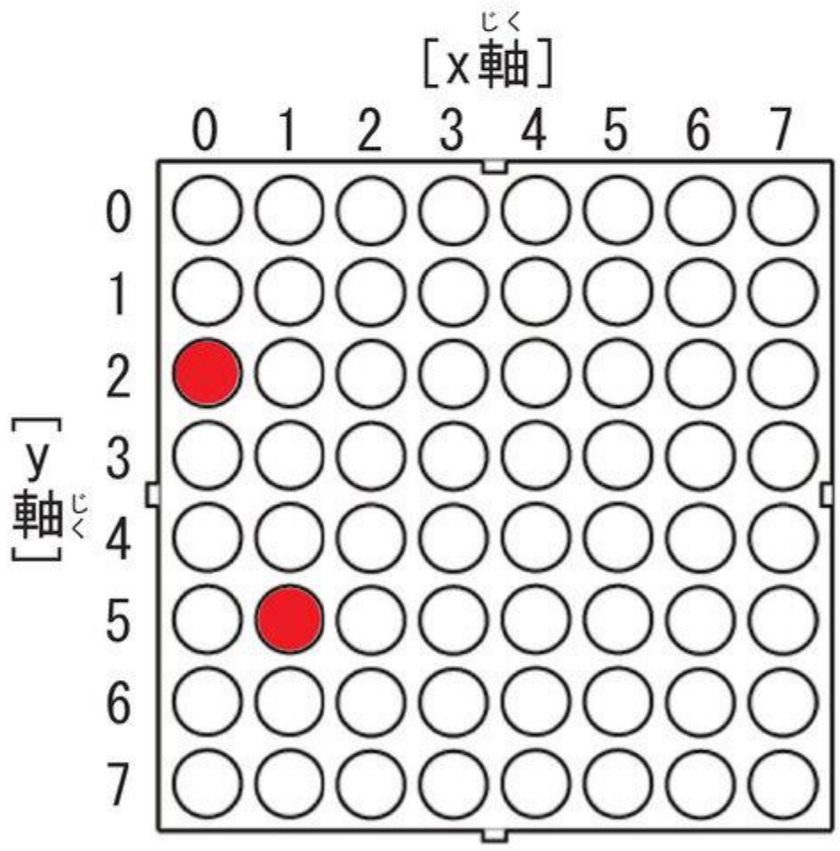
① 関数 y ＝ 2 x の場合

| x | 0 | 1 | 2 | 3 | 4 | … |
|---|---|---|---|---|---|---|
| y | 0 | 2 | 4 | 6 | 8 | … |




② 関数 y ＝ 3 x ＋ 2 の場合

| x | 0 | 1 | 2 | 3 | 4 | … |
|---|---|---|---|---|---|---|
| y | 2 | 5 | 8 | 11 | 14 | … |



## P.11　チャレンジ課題　解答例（MatrixGraphChallenge1）

```
void loop(){
    for(i=-7;i<7;i++){                          // 切片iを-7から6まで動かす
        for(x=0;x<8;x++){                       // xを0から7の範囲にする
            y = x + i;                          // 一次関数(切片i)
            myMatrix.write(x, y, HIGH); // LEDを点灯
            myMatrix.write(y, x, HIGH); // LEDを点灯
        }
        delay(100);
        myMatrix.clear();                       // マトリクスLEDの表示を消す
    }
}
```

## P.11　チャレンジ課題　解答例（MatrixGraphChallenge2）

```
void loop(){
    for(i=-7;i<7;i++){                          // 切片iを-7から6まで動かす
        for(x=0;x<8;x++){                       // xを0から7の範囲にする
            y = x + i;                          // 一次関数(切片i)
                // 組み合わせ次第でさまざまな場所にグラフをかく
            myMatrix.write(x, y, HIGH);     // LEDを点灯
            myMatrix.write(y, x, HIGH);     // LEDを点灯
            myMatrix.write(y, 7-x, HIGH);   // LEDを点灯
            myMatrix.write(7-y, x, HIGH);   // LEDを点灯
        }
        delay(100);
        myMatrix.clear();                       // マトリクスLEDの表示を消す
    }
}
```

## P.15　チャレンジ課題　解答例（MatrixLineTurn）

```
Matrix myMatrix = Matrix(11, 13, 1);  // マトリクスLEDを使うときのオマジナイ

void setup(){
    myMatrix.clear();                 // マトリクスLEDの表示を消す
}

int x1,x2,y1,y2;                      // 始点と終点に使うx・y座標を用意

void loop(){
    // y軸に沿ってラインが動く
    x1 = 0;
    x2 = 7;
    for(int y = 0; y <= 7; y++){
        y1 = y;
        y2 = 7 - y;
        myMatrix.line(x1,y1,x2,y2);   // 座標(x1,y1)から座標(x2,y2)へ線を引く
        delay(100);
        myMatrix.clear();             // マトリクスLEDの表示を消す
    }

    // x軸に沿ってラインが動く
    y1 = 7;
    y2 = 0;

    for(int x = 1; x <= 6; x++){
        x1 = x;
        x2 = 7 - x;
        myMatrix.line(x1,y1,x2,y2);   // 座標(x1,y1)から座標(x2,y2)へ線を引く
        delay(100);
        myMatrix.clear();             // マトリクスLEDの表示を消す
    }
}
```

## P.15　チャレンジ課題　解答例（MatrixLineTurnR）

```
Matrix myMatrix = Matrix(11, 13, 1);  // マトリクスLEDを使うときのオマジナイ

void setup(){
    myMatrix.clear();                 // マトリクスLEDの表示を消す
}

int x1,x2,y1,y2;                      // 始点と終点に使うx・y座標を用意

void loop(){
    x1 = 0;
    x2 = 7;

    for(int y = 7; y >= 0; y--){
        y1 = y;
        y2 = 7 - y;
        myMatrix.line(x1,y1,x2,y2);   // 座標(x1,y1)から座標(x2,y2)へ線を引く
        delay(100);
        myMatrix.clear();
    }

    y1 = 7;
    y2 = 0;

    for(int x = 6; x >= 1; x--){
        x1 = x;
        x2 = 7 - x;
        myMatrix.line(x1,y1,x2,y2);   // 座標(x1,y1)から座標(x2,y2)へ線を引く
        delay(100);
        myMatrix.clear();
```