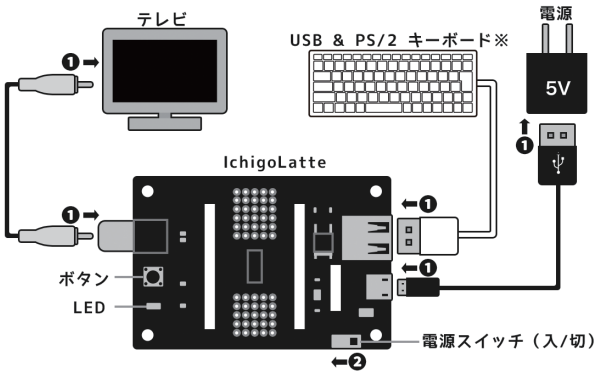
IchigoLatte - The first cup

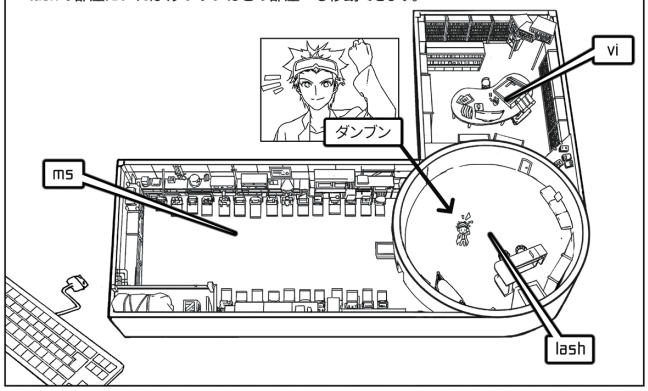


※「USB」のみのキーボードでは動作しません。
※ PS/2→USB信号変換アダプタ(2分岐タイプ)は動作しません。

IchigoLatteの3つの部屋

1

IchigoLatteは小さなコンピュータです。
コンピュータの中には「ダンブン」が住んでいます。
コンピュータの中は「lash」「ms」「vi」という3つの部屋に分かれています。
ダンブンははじめ、lashの部屋で命令を待っています。
部屋によってダンブンのできることが変わります。
lashの部屋にいれば、ダンブンはどの部屋へも移動できます。

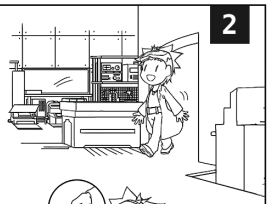


ダンブンに命令してみよう

2

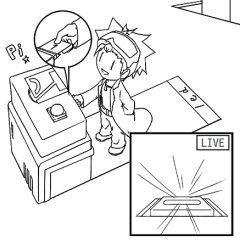
下の命令をキーボードで打ち、Enterキーを押して、
msの部屋に行ってみましょう。

```
lash>ms
```



msは、ダンブンのできることがいちばん多い部屋です。
ためしに、次のようにダンブンに命令してみましょう。
これは「LEDを光らせる」命令です。

```
ms>led(1);
```



ダンブンがLEDを光らせてくれたら成功です。
続いて「LEDを消す」ように命令してみましょう。

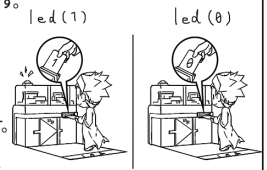
```
ms>led(0);
```

複数の命令をしてみよう

3

msの部屋では、複数の命令をきいてもらうこともできます。
「;」が命令の区切りです。

```
ms>led(1);led(0);
```



このように命令すると、
ダンブンはLEDを光らせた直後にLEDを消してくれます。
しかしダンブンの動きはとても速いので、
上のような命令では、LEDはほんの一瞬しか光りません。

そこで、LEDを光らせた後に
「待ってもらう」という命令をします。

```
ms>led(1);sleep(2000);led(0);
```

「sleep(2000)」は「2000ミリ秒待ってもらう」という命令です(1000ミリ秒=1秒)。
LEDが光って、それから2秒後に消えたら、成功です。

msの部屋からlashの部屋に戻るときは、Escキーを押してください。

プログラムを作ろう

4

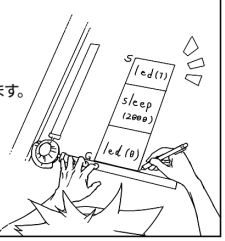
プログラムは命令の書かれた道です。道を作るには、viの部屋に行きます。
次の命令を打って、lashの部屋からviの部屋に移動しましょう。

```
lash>vi
```

viの部屋で、ダンブンに道を作ってもらいます。
ためしに、次の文字を打ってみましょう。
Enterキーを押すと、次の行に移動できます。
命令の最後には「;」を付けます。

```
vi>led(1);
  sleep(2000);
  led(0);
```

Escキーを押すと、ダンブンは道を保存し、lashの部屋に戻ります。
作った道は、IchigoLatteの電源を切ると消えてしまうので、
きちんと保存しましょう。



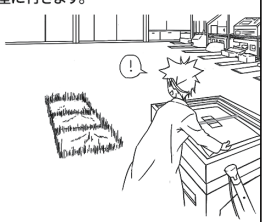
※保存できる道の数はひとつだけです。
　道をたくさん保存しておくには、EEPROMが必要です。
※保存せずに部屋を出たいときは Ctrl + D です。

プログラムを実行しよう

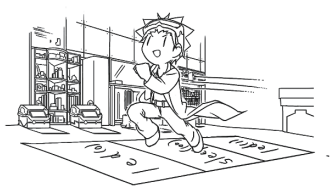
5

作った道の上をダンブンに走ってもらうために、msの部屋に行きます。
msの部屋に道を持っていくには、次のように命令します。

```
lash>ms .
```



「.」は作った道のことです。
msの部屋に道を持っていくと、
ダンブンはすぐに道の上を走ってくれます。



道を走り終わると、ダンブンはlashの部屋に戻ってきます。
道を見たり編集したいときは、またviの部屋に行きましょう。

※道を全て消したいときは、lashの部屋で「echo > .」と打ってEnterキーを押します。

IchigoLatte http://ichigolatte.shizentai.jp/

6 数字や文字を入れる箱1

msの部屋では、ダンブンは数字や文字の入る箱を作り出せます。
箱を作るには、viの部屋で以下のような命令を打ってください。

```
vi> var cargo=1;
```

「var」が箱を作るという命令です。
この命令のすぐ後ろにあるのが、作る箱の名前です。
箱の名前は、アルファベットと数字の組み合わせで
7文字以下なら、自由に決められます。
箱を作るのと同時に数字や文字を入れてもいいですし、
作った後で数字や文字を入れてもかまいません。

箱の名前を「,(コンマ)」で区切ると、
一度に複数の箱を作れます。
以下の命令では、
「cargo2」と「cargo3」という名前の箱を作っています。

```
vi> var cargo2,cargo3;
cargo2=2;
cargo3=3;
```

7 数字や文字を入れる箱2

箱の中身をみるときは、以下のような命令を使います。

```
vi> var cargo=1;
log(cargo);
```

2行目にある「log」が、
「()の中に置かれたものを画面に表示する」
という命令です。
()の中に箱を置くと、箱の中身が画面に表示されます。

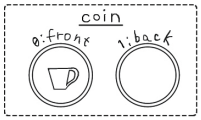
箱の中身をつかって計算することもできます。

```
vi> var cargo=5;
var bag=2;
log(cargo+bag);
```

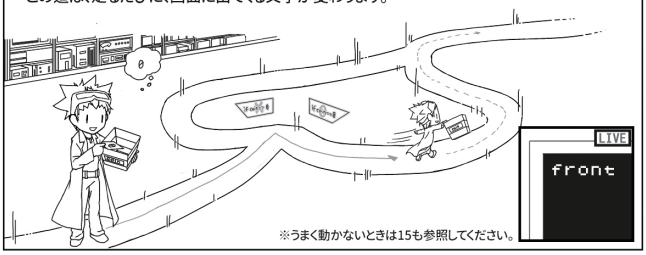
8 もし～がxxだったら

正しいときには、if()の後にある{}の中を走ります。
違うときにはelseの後にある{}の中を走ります。
「{」の後と「}」の後には、「;」を付けなくても大丈夫です。

```
vi> var coin=rnd(2);
if(coin==0){
  log("front");
}else{
  log("back");
}
```



この道は、走るたびに、画面に出てくる文字が変わります。

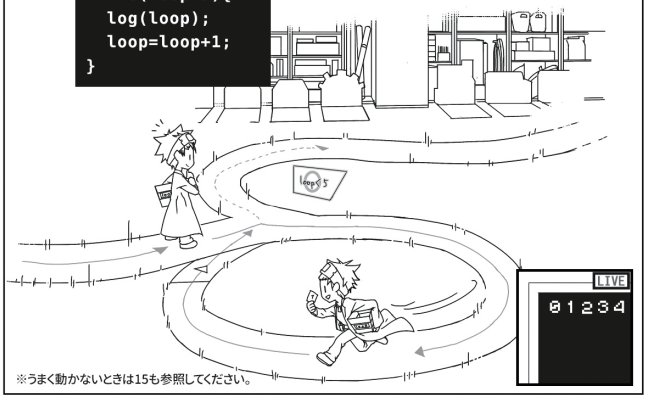


※うまく動かないときは15も参照してください。

9 くり返し

while()の中に書かれたことが正しい間は、{}の中の道をくり返し走ります。
while()の中が最初から間違っていると、{}の中の道を走りません。
while()の中がいつまでも正しいままだと、くり返しが終わらなくなります。

```
vi> var loop=0;
while(loop<5){
  log(loop);
  loop=loop+1;
}
```

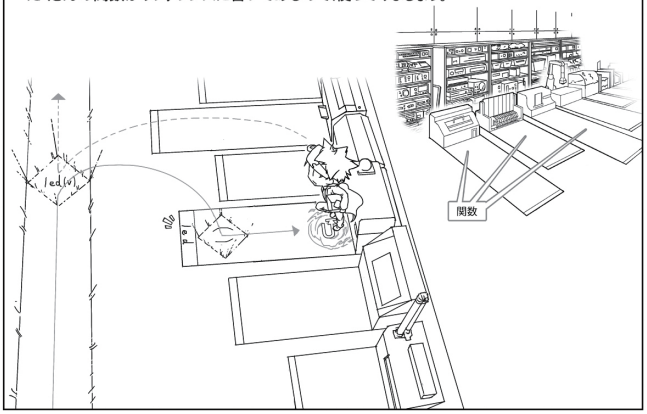


※うまく動かないときは15も参照してください。

10 関数1

関数とは、すでに用意されている道のことです。
ダンブンに関数の中を走ってもらいたいときは、関数の名前の後に「()」を付けます。
数字や文字を持っていける関数では「()」の中に数字を書きます。

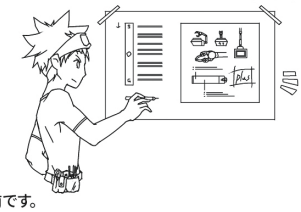
たくさんの関数がリファレンスに書いてあるので、使ってみましょう。



11 関数2

オリジナルの関数も作れます。

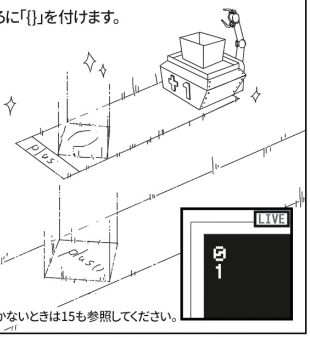
```
vi> var box=0;
function plus(){
  box=box+1;
}
```



「function」が関数を作る命令です。
この命令のすぐ後ろにあるのが、作る関数の名前です。
関数の名前の後ろに「()」をつけて、さらにその後ろに「{}」を付けます。
「{}」の中に、道を書きます。

「function」で関数を作っただけでは、
ダンブンは関数の中を走ってくれません。
作った関数の名前の後に()を付けたものが、
「この関数の中を走る」という命令になります。

```
vi> log(box,"\n");
plus();
log(box,"\n");
```



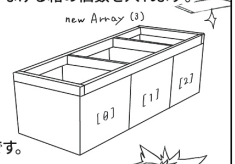
※うまく動かないときは15も参照してください。

 IchigoLatte http://ichigolatte.shizentai.jp/

配列

12

「new Array()」という命令で、つながった箱を作れます。
箱がいくつもつながったものを配列といいます。()の中につなげる箱の個数を入れます。
最大で32個までつなげられます（※ver1.0.0の場合）。




```
var arr=new Array(3);
```

配列の名前「arr」の後に[番号]をつけると、
その番号の箱に文字や数字を入れたり、
箱の中身をみたりできます。3個の配列の場合、番号は0～2です。

```
arr[0]=8;
arr[1]=13;
arr[2]=24;

var i=0;
while(i<3){
  log(arr[i],"\n");
  i=i+1;
}
```

```
8
13
24
```

配列の名前「arr」の後に「.length」をつけると、
箱がいくつつながっているかがみえます。

```
log(arr.length);
```

```
3
```

※うまく動かないときは15も参照してください。

イベント

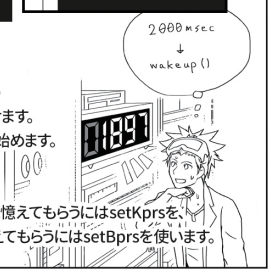
13

何かが起きたときに合わせて、ダンプンに関数の中を走らせることができます。
「何かが起きたとき」とは、「～ミリ秒たったとき」や
「キーボードが押されたとき」や「ボタンが押されたとき」です。
「起きた何か」のことをイベントといいます。
setToutを使うと、～ミリ秒たったときに、ダンプンに関数の中を走らせることができます。

```
function wakeup(){
  led(1);
  log("Good morning!","\n");
}

setTout(wakeup,2000);
led(0);
log("Good night...","\n");
```

```
Good night...
Good morning!
```



ダンプンがsetToutの道を通ると、
ダンプンは「今から2000ミリ秒たったら、wakeupという
関数を走る」ということを憶えて、タイマーをスタートさせます。
全ての道を走り終わると、ダンプンはタイマーを見張り始めます。
そしてタイマーが2000ミリ秒たっていたら、
wakeup関数の道を走り始めます。
「キーボードが押されたとき」に走りたい道をダンプンに憶えてもらうにはsetKprsを、
「ボタンが押されたとき」に走りたい道をダンプンに憶えてもらうにはsetBprsを使います。
※うまく動かないときは15も参照してください。

サンプルプログラム

14

```
var sX=15,sY=22;
var nX=15,nY=22;
var nV=5,sV=99,score=0;

function main(){
  if(sV!=99){
    sY=sY+sV;
    sV=sV+1;
  }

  if(sY>22){
    sY=22;
    sV=99;
    score=score+1;
  }

  nX=nX+nV;

  if(15<nX) nV=nV-1;
  if(nX<15) nV=nV+1;

  cls();

  lc(sX,sY);log("@");
  lc(nX,nY);log("_");
  lc(0,0);log("SCORE:",score);

  if((sX==nX)*(sY==nY)) exit(0);

  setTout(main,100);
}

function kf(key){
  if(key==32) sV=-3;
}

setKprs(kf);
main();
```

「縄跳びさっちゃんゲーム」
すべて打ち込み実行すると、
ゲームとしてあそべます。
（スペースキーをおして、
なわを跳んでください）

※うまく動かないときは15も参照してください。

動かないときは？

15

道の作り方を間違えるとダンプンはうまく走れなくなります。
うまく走れない部分のことを、バグといいます。
ダンプンは「再起動」や「エラーメッセージ」でバグがあることを教えてくれます。

●バグがあると、ダンプンはLEDを点滅させて、コンピュータを再起動します。
プログラムはちゃんと保存されたままなので、バグを探しだして、直しましょう。
たとえばこのプログラムを実行すると、ダンプンはコンピュータを再起動します。

```
log("hello);
```

➡

```
log("hello);
```

ふたつ目の「"」がないので、ダンプンはうまく走れません。「"」を足して、バグを直しましょう。
他にも、「)」「(」「;」なども忘れやすいので、気をつけましょう。

●ダンプンはエラーメッセージを出してくれるときもあります。たとえば、

```
log(coin);
```

```
!not found.('coin');
```

「!not found.('coin')」がエラーメッセージです。これは「coinという名前の箱がないよ」
という意味です。varの命令で、coinという名前の箱を作りましょう。
関数を作り忘れているときも、同じエラーメッセージが出ます。

●この他、再起動もエラーメッセージもないけれど、ダンプンがうまく走れない場合もあります。

```
log(1+++1);
```

```
1204571
```

「+」の記号を3つ続けて打ってしまっているところがバグです。「+」の記号を1つにしましょう。

リファレンス　light1

16

lashのコマンド

lash	e.g.	description
echo	echo ABC	「ABC」を画面に表示する。
	echo ABC > .	「ABC」を.ファイルに書き出す。
cat	cat .	.ファイルを画面に表示する。
	cat .>@0	.ファイルをEEPROMのスロット0に書き出す。
	cat @3 > .	EEPROMのスロット3を.ファイルに書き出す。
	cat . > uart	.ファイルをUART(TX)に書き出す。
	cat uart > @6	UART(RX)をEEPROMのスロット6に書き出す。
ls	ls	EEPROMのスロットリストを表示する。
vi	vi	.ファイルを編集する。
ms	ms	MINIScriptを対話モードで起動する。
	ms .	.ファイルをMINIScriptで実行する。

lashのファイル

.	本体内のファイル（1つ）
@N	EEPROM内のスロットNのファイル
uart	UARTシリアル通信（TX,RX）

リファレンス　light2

17

msの関数

led(数)	数が1ならLEDが光り、0なら消える。
btn()	ボタンが押されていれば1、そうでないとき0を返す。
sleep(数)	数のミリ秒分待つ。1000で1秒。
while(数){次}	数が0でない限り、次をくり返す。
if(数){次}else{次2}	数が0以外のとき次を実行し、0であれば次2を実行する（else以降は省略可能）。
log(数や文字列,数や文字列,...)	文字を画面に表示する（文字列は"で囲む。","で連結できる）。
lc(数,数)	次に文字を書く位置を横、縦の順に指定する(-1で非表示)。
cls()	画面を全部消す
input()	キーボードからの入力を返す。
inkey()	キーボードから1文字入力する。
chr(数)	文字コードに対応する文字を返す。
scr(数,数)	画面上の指定した位置に置かれた文字コードを取得する。
tick()	時間を返す。
setTout(関数,数)	数ミリ秒後に、関数を実行する。
数 == 数	比較して等しい時に1を返す。
数 != 数	比較して等しくない時に1を返す。
数 <= 数	比較して以下の時に1を返す。
数 < 数	比較して未満の時に1を返す。
+ - * / %	足し算、ひき算、かけ算、割り算、割り算した余りを返す。
new Array(数)	長さが数の配列を作る。
var 名前	変数を作る。

function 名前(名前2,名前3,...){
　処理
}
	関数を作る。