

プチコンでロールプレイングゲームを作る (1)

プチコンは、NINTENDO DSi/3DS で BASIC (ベーシック) プログラムを作るソフトです。

●今回の目標

プチコンでゲームのプログラムを作ります。

ゲームは、プログラムの作り方から見て、「非リアルタイム系」「リアルタイム系」の大きく2種類に分かれます。

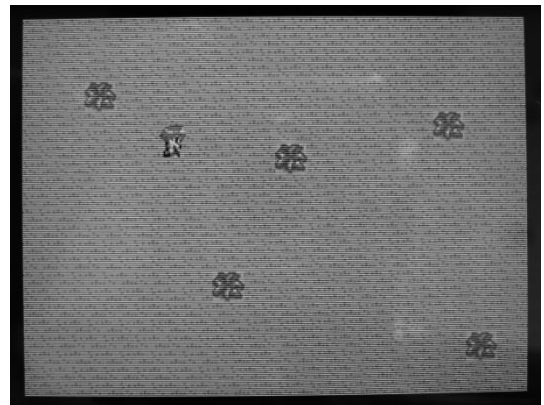
| | |
|--|---|
| <p>① 非リアルタイム系 プレイヤーが何か入力すると先に進む。自動では進行しない。</p> <p style="text-align: center;">↑↓</p> | <p>クイズ、ボードゲーム アドベンチャー、ロールプレイング シミュレーション</p> |
| <p>② リアルタイム系 自動的にどんどん先へ進む。</p> | <p>アクション、シューティング</p> |

はっきりどちらか2種類に分かれている訳ではなく、両者の中間に当たるゲームもたくさんあります。

プログラムを作るのは、「非リアルタイム系」の方が簡単です。「リアルタイム系」はいろいろな物を同時に動かさないといけないので難しいです。

今回は非リアルタイム系の代表として、ロールプレイングゲーム(RPG)を作ってみましょう。

自分のキャラクターがフィールドを歩いて行って、敵に出会ったら戦う、というゲームです。



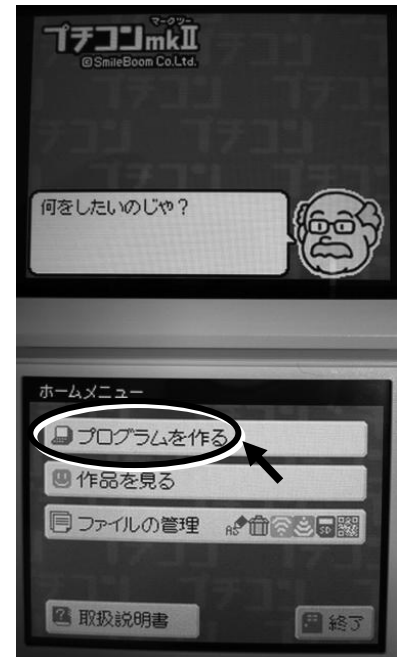
●プチコンの操作

3DS のメニューから、ソフト「プチコン mkII」(マークツォー)を動かします。



ソフトが動くと、最初のメニュー画面が表示されます。

「何をしたいのじゃ?」と聞かれるので、下画面のメニューから「プログラムを作る」を選びます。

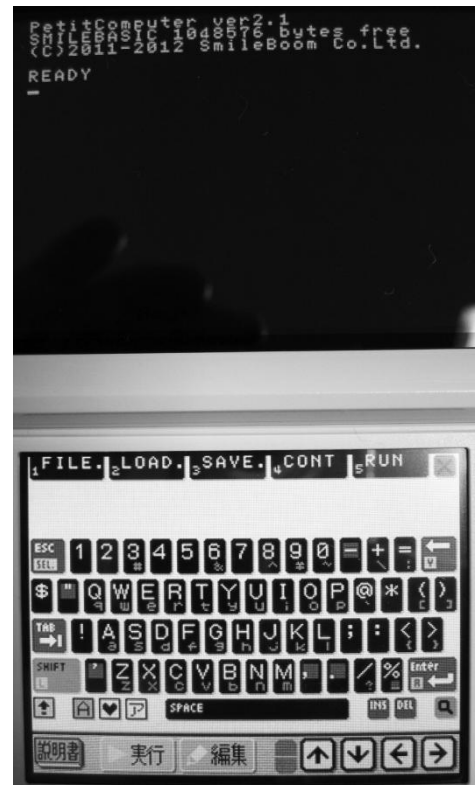


プログラムを作る画面が表示されます。


上画面が実行・表示画面、下画面がキーボードです。下画面でキーをタッチすると、上画面にその文字が入力されます。

キーボードで入力する文字の種類を変えるには、以下のようにします。

- 普通に打つ…大文字アルファベット「A B C」
- SHIFT キーを押してから打つ…小文字アルファベット「a b c」
- 左下の文字きりかえキーで「♥」を押す…グラフィック文字が打てる。SHIFT キーでさらにきりかえ。
- 左下の文字きりかえキーで「ア」を押す…カタカナ文字が打てる。SHIFT キーでさらにきりかえ。



試しにいろいろな文字を打ってみましょう。

打ってある文字を消す時は、バックスペースキー  またはデリートキー  を使います。

- バックスペースキー…カーソルの前の文字が消える

▲_ → _

- デリートキー…カーソルの後ろの文字が消える

_▲ → _

●実行モードでプログラムを動かす

まずは実行モード画面で、かんたんなプログラムを打って、動かしてみましよう。

```
PRINT "コンニチハ"
```

「PRINT」の後ろは、空白(スペース)を1文字空けます。

「コンニチハ」の両側は「"」(ダブルクォーテーション)で囲みます。

上のプログラムを打って、Enter キーを押すと、次の行に「コンニチハ」と表示されます。

```
PRINT "コンニチハ"  
コンニチハ
```

「PRINT」(プリント)命令は、画面に文字を表示する命令です。

文字をいろいろ変えて、表示してみましよう。

PRINT 命令では、計算もできます。

「PRINT」に続いて(1文字空白を入れて)計算式を書いて Enter キーを押すと、次の行に計算の答えが表示されます。

```
PRINT 1+1  
2
```

いろいろな計算をしてみましよう。計算記号は以下のように書きます。

- 足し算…「+」
- 引き算…「-」
- かけ算…「*」
- わり算…「/」

画面に文字がいっぱいになったら、「ACLS」(エーシーエルエス)命令を使うと、画面がクリアされます。

```
ACLS
```

●編集モードで長いプログラムを作る

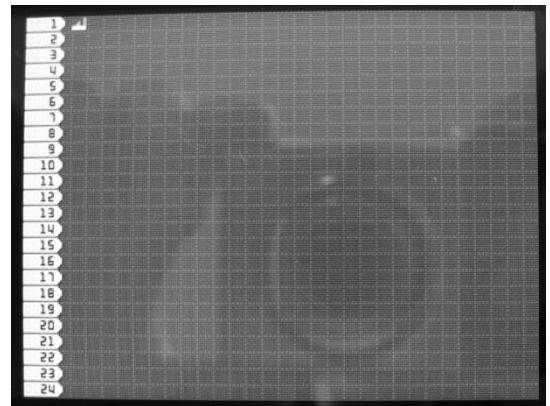
実行モードで直接プログラムを書くと、1行しか書けません。
長いプログラムを作るには、以下の手順で行います。

- ① 編集モードでプログラムを書く
- ② 実行モードでプログラムを実行する

キーボード画面の「編集」ボタンをタッチして、編集モードに移ります。

上の画面が、編集モードの画面になります。

画面左側に「1」「2」…と行番号が表示されます。



少し長いプログラムを書いてみましょう。

```
1 PRINT "コンニチハ"
2 PRINT "コンバンハ"
```

1行目で「コンニチハ」を表示、2行目で「コンバンハ」を表示するプログラムです。

まず、この2行を編集モード画面で打ちます。

次に、キーボード画面左下にある「実行」ボタンをタッチして、実行モードにします。

実行モード画面で「RUN」(ラン)と打って Enter キーを押すと、プログラムが実行されて、2行の文字が表示されます。

```
コンニチハ
コンバンハ
```

プログラムは、基本的に上から下へ順番に実行されます。

何行かプログラムを打って、実行してみましょう。

※プログラムを実行する命令「RUN」は、キーボード画面の上にあるファンクションキーの5番に登録されています。これをタップすると一度に入力できます。



●キャラクターを表示する

プログラムの作り方の基本がわかった所で、いよいよゲームのプログラムを作っていきます。まず、キャラクターを画面に表示してみましょう。

★キャラクターを表示

プチコンには、「スプライト」というキャラクターを表示する機能があります。スプライトを使って、キャラクターを表示してみましょう。編集モードの画面で、以下のプログラムを打ちます。

```
1 SPSET 0, 64, 2, 0, 0, 2
```

実行すると、画面左上にキャラクターが表示されます。



この1行だけだと、その前に表示していた文字などと重なってしまいます。キャラクターを表示する前に、「ACLS」命令で画面をクリアしておいた方がいいでしょう。

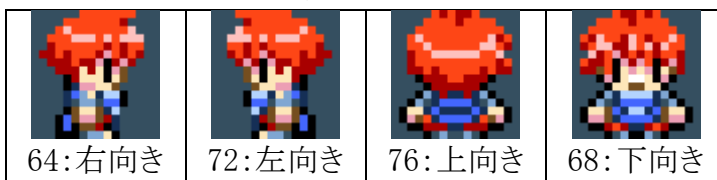
```
1 ACLS
2 SPSET 0, 64, 2, 0, 0, 2
```

SPSET (エスピーセット) 命令の文法は、以下のようになっています。

```
SPSET      0      , 64      , 2      , 0      , 0      , 2
           管理番号  キャラ番号  パレット番号  横反転   縦反転   優先順位
```

| | |
|--------|--|
| 管理番号 | スプライトの管理番号。0～99 の範囲で自由に決められます。 |
| キャラ番号 | スプライトのキャラ番号。0～511(上画面用)、0～117(下画面用)。 |
| パレット番号 | 色のパレット番号を指定します。0～15。 |
| 横反転 | 0=なし、1=横反転(左右逆に表示) |
| 縦反転 | 0=なし、1=縦反転(上下逆に表示) |
| 優先順位 | スプライトを表示する優先順位 0=コンソール(文字表示)の前 1=手前の BG 面(背景表示面)より前 2=2 枚の BG 面の間 3=奥の BG 面の後ろ |

キャラ番号を変えると、表示されるキャラクターが変わります。以下のように変えてみましょう。



★キャラクターの位置を変える

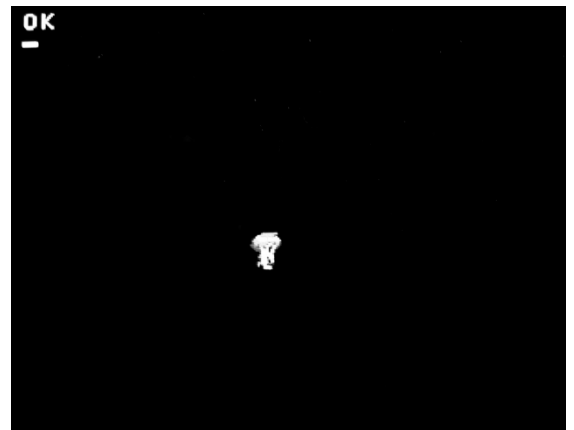
今はキャラクターが画面左上に表示されていますが、位置を変えてみましょう。
スプライトの位置を変えるには、「SPOFS」(エスピーオフセット) 命令を使います。

```

1 ACLS
2 SPSET 0, 64, 2, 0, 0, 2
3 SPOFS 0, 100, 100

```

こうすると、画面のまん中あたりにキャラクターが表示されます。



SPOFS 命令の文法は以下のようになっています。

```

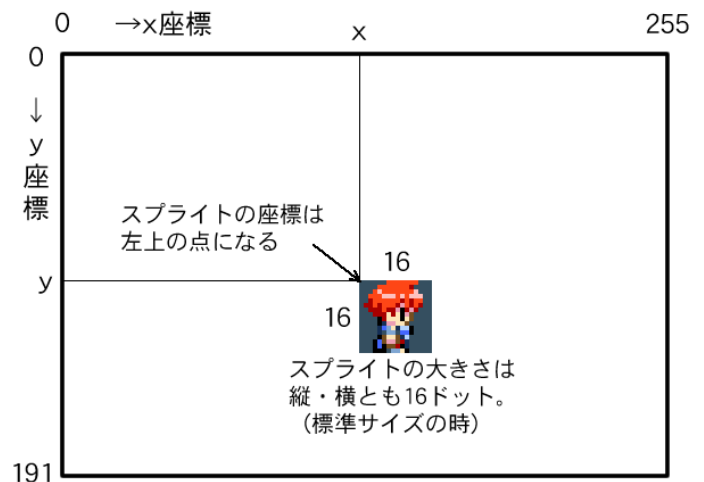
SPOFS      0      ,100      ,100      ,0
           管理番号  x座標   y座標   補間時間

```

| | |
|----------------|------------------------------------|
| 管理番号 | スプライトの管理番号。0～99。 |
| x座標 | スプライトのx座標。-1024～1024。(画面の外でも受け付ける) |
| y座標 | スプライトのy座標。-1024～1024。(画面の外でも受け付ける) |
| 補間時間 (省略可能) | 前の状態から現在の状態へ移動する間にかかる時間。1=60分の1秒。 |

プチコンの画面の座標は、右のようになっています。

座標の数字を変えて、キャラクターをいろいろな位置に表示してみましょう。



●キャラクターを動かす

さっきはキャラクターの座標の数字を変えて、表示する位置を変えました。
実際のゲームでは、いちいち数字を入力して動かしていたら、ゲームになりません。
プログラムで、自動的にキャラクターを動かしてみましょう。

★変数を使う

キャラクターを動かすために、位置(座標)を変数(へんすう)で指定します。
変数は、算数で使う「□」(四角)と同じで、いろいろな数字を入れる箱のようなものです。
(中学の数学なら「x」や「a」などの文字と同じです)

キャラクターのx座標を変数「X」、y座標を変数「Y」で指定することにします。
それぞれ変数を設定して、キャラクターの位置を指定します。

```
1 ACLS
2 SPSET 0, 64, 2, 0, 0, 2
3 X=100
4 Y=100
5 SPOFS 0, X, Y
```

「X=100」は、「変数 X の値を 100 にする」という意味です。「X と 100 が等しい」という意味ではないので、注意してください。

同じように「Y=100」で、変数 Y の値を 100 にしています。

次の SPOFS 命令で、X と Y で指定した座標にキャラクターを表示します。

「X=100」「Y=100」のそれぞれの値をいろいろ変えて、試してみてください。

★キャラクターをくり返して動かす

くり返しをするプログラムで、自動的にキャラクターを動かしてみよう。

まずはX座標を変えて、左右方向に動かしてみます。

くり返し命令「FOR」(フォー)、「NEXT」(ネクスト)を使います。

```

1 ACLS
2 SPSET 0, 64, 2, 0, 0, 2
3 Y=100
4 FOR X=0 TO 239
5 SPOFS 0, X, Y
6 NEXT X

```

…どうですか？

キャラクターが画面左から右へ動くのですが、動きがあまりに速すぎて、一瞬で終わってしまいます。

こんな時は、時間待ち命令「WAIT」(ウェイト)をくり返しの中に入れて、動きを遅くします。

```

1 ACLS
2 SPSET 0, 64, 2, 0, 0, 2
3 Y=100
4 FOR X=0 TO 239
5 SPOFS 0, X, Y
6 WAIT 1
7 NEXT X

```

キャラクターが左から右へ動くのがわかりましたか？

それでは、文法を説明します。まずはくり返しの「FOR」「NEXT」命令から。

| | | | | | | |
|-----|----|------|----|------|------|-----|
| FOR | X | =0 | TO | 239 | STEP | 2 |
| | 変数 | 最初の値 | | 最後の値 | | 変化量 |

指定した変数を、最初の値～最後の値まで変化させて、対応するNEXTまでのプログラムをくり返します。STEPは変化させる量で、プラスの値を指定すると増え、マイナスの値だと減っていきます。STEPを省略した場合は1ずつ増やします。

```

NEXT X
  変数

```

くり返しの最後を表します。FOR命令に対応した変数を指定します。

時間待ちをする WAIT 命令の文法は、以下のとおりです。

```
WAIT      1
          待ち時間
```

待ち時間…60 分の 1 秒単位で指定します。60=1 秒です。

今は左から右に移動しましたが、その後に右から左へ戻して、往復させてみましょう。
もう1つ FOR~NEXT のくり返しを追加します。

```

1 ACLS
2 SPSET 0, 64, 2, 0, 0, 2
3 Y=100
4 FOR X=0 TO 239
5 SPOFS 0, X, Y
6 WAIT 1
7 NEXT X
8 FOR X=239 TO 0 STEP -1
9 SPOFS 0, X, Y
10 WAIT 1
11 NEXT X

```

右から左へ戻すくり返しは、x 座標を 1 ずつ減らさないといけないので、「STEP -1」と変化量をマイナスで指定します。

★行のコピー・ペースト

今回のように、同じようなプログラムの行を打つ時は、前に打ってある行をコピーすると楽です。行をコピーするには、キーボード右下の「コピー」「ペースト」キーを使います。
(表示されていない時は▲▼ボタンをタッチしてください)



- ① 十字キーでカーソルを移動して、コピーしたい行にカーソルが点滅した状態にする。

```
FOR X=0 TO 239
```

- ② 「コピー」キーをタッチする。
- ③ 十字キーでカーソルを下へ移動して、新しい行の場所へ持っていく。
- ④ 「ペースト」キーをタッチすると、行がコピーされて貼りつきます。

```
FOR X=0 TO 239
```

- ⑤ 行の内容を書き換えます。このようにコピーを使うと、最小限の書き換えですみます。

★キャラクターをアニメーションさせる

今はキャラクターが横にすべる感じで動いていますが、アニメーションさせると歩いている感じで動かします。

スプライトをアニメーションさせるには、「SPANIM」(エスピーアニメ) 命令を使います。

```

1 ACLS
2 SPSET 0, 64, 2, 0, 0, 2
3 SPANIM 0, 4, 10
4 Y=100

```

SPSET 命令の行の下に、SPANIM 命令の行を新しく作ります。操作は以下のとおりです。

- ① 「Y=100」の行の先頭にカーソルを持っていきます。

```

2 SPSET 0, 64, 2, 0, 0, 2
3 Y=100

```

- ② Enter キーを押して改行を入れます。

```

2 SPSET 0, 64, 2, 0, 0, 2
3
4 Y=100

```

- ③ 十字キーでカーソルを上移動します。

```

2 SPSET 0, 64, 2, 0, 0, 2
3 _
4 Y=100

```

- ④ SPANIM 命令を打ちます。

```

2 SPSET 0, 64, 2, 0, 0, 2
3 SPANIM 0, 4, 10_
4 Y=100

```

実行してみましょう。男の子のキャラがアニメーションして、歩いているように見えます。



SPANIM 命令の文法は以下のとおりです。

```
SPANIM      0      , 4      , 10      , 0
            スプライト コマ数 コマ表示 くり返し
            番号      時間   回数
```

| | |
|---------|---|
| スプライト番号 | アニメーションさせたいスプライトの番号。0～99。 |
| コマ数 | アニメのコマ数。SPSET 命令で指定した番号のキャラクター絵から、そのコマ数分だけアニメを表示する。 |
| コマ表示時間 | 1コマを表示する時間。60分の1秒単位で指定。 |
| くり返し回数 | アニメをくり返す回数を指定。0=無限にくり返し。省略すると無限くり返し。 |

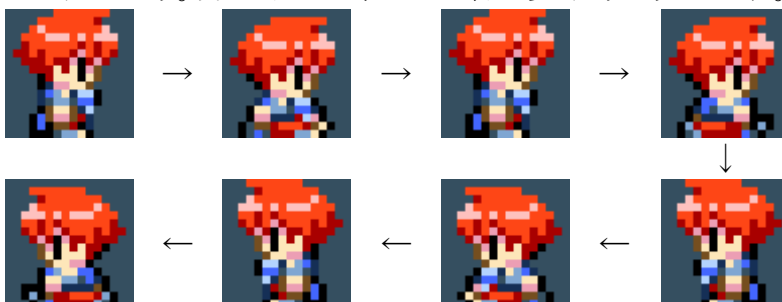
今のままだと、左から右へ歩く時はいいですが、右から左へ戻る時に後ろ向きに歩いているように見えてしまいます。

戻る時の(2回目の)FOR～NEXTのくり返しの前に、キャラクターを左向きのスプライト(72番)で指定し直すといいでしょう。

```

1 ACLS
2 SPSET 0, 64, 2, 0, 0, 2
3 SPANIM 0, 4, 10
4 Y=100
5 FOR X=0 TO 239
6 SPOFS 0, X, Y
7 WAIT 1
8 NEXT X
9 SPSET 0, 72, 2, 0, 0, 2
10 SPANIM 0, 4, 10
11 FOR X=239 TO 0 STEP -1
12 SPOFS 0, X, Y
13 WAIT 1
14 NEXT X
```

実行してみましょう。男の子のキャラが左右に歩くように見えます。



★他の行へジャンプする

これまでのプログラムだと、キャラクターが左右に一往復して終わりますが、ずっとくり返して往復するようにしてみましょう。

プログラムの実行を他の行へジャンプさせる「GOTO」(ゴートゥー)命令を使います。

```
1  ACLS
2  @ARUKU
3  SPSET  0, 64, 2, 0, 0, 2
4  SPANIM 0, 4, 10
5  Y=100
6  FOR X=0 TO 239
7  SPOFS  0, X, Y
8  WAIT  1
9  NEXT X
10 SPSET  0, 72, 2, 0, 0, 2
11 SPANIM 0, 4, 10
12 FOR X=239 TO 0 STEP -1
13 SPOFS  0, X, Y
14 WAIT  1
15 NEXT X
16 GOTO  @ARUKU
```

GOTO 命令の文法は以下のとおりです。

```
GOTO      @ARUKU
          ジャンプ先ラベル
```

ジャンプする先は「ラベル」で指定します。

ラベルは、「@」(アットマーク)に続けてアルファベットや数字で指定します。

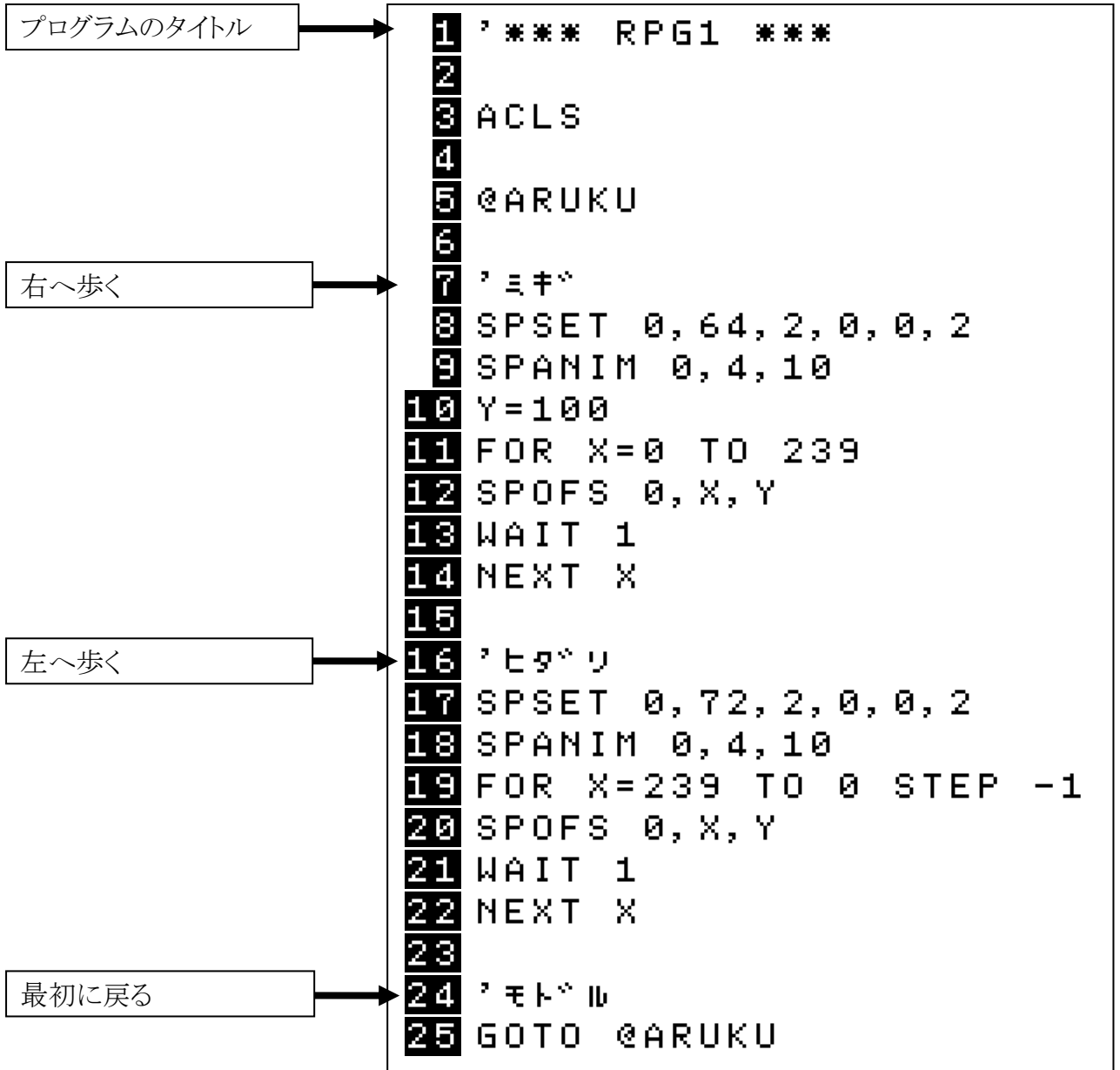
今回の場合は、キャラクターが左へ動いて戻った後に「GOTO @ARUKU」を入れて、プログラムの最初の方に入れた「@ARUKU」ラベルへ実行をジャンプさせます。

プログラムを実行すると、キャラクターが左右に往復し続けます。

いつまでも止まらないので、キーボードの「停止」ボタンをタップして止めてください。

●コメントを入れる

プログラムが長くなってくると、自分でもどこで何をしているのかよくわからなくなります。後で見てわかるように、プログラムの中にコメント(注釈、説明)を入れることができます。



行の先頭に「'」(アポストロフィー)をつけて、そこに続けてコメントを書きます。コメントはアルファベット、数字、カタカナなど、自由に書けます。

また、プログラムのまとまりごとに改行を入れて 1 行空けると、まとまりが区別できて、全体が見やすくなります。

●プログラムを保存する

作ったプログラムは、このままだとプチコンを終了すると消えてしまいます。

消えてしまうと困るので、保存しましょう。

プログラムを保存するには、実行モードの画面に移って、「SAVE」(セーブ)命令を使います。

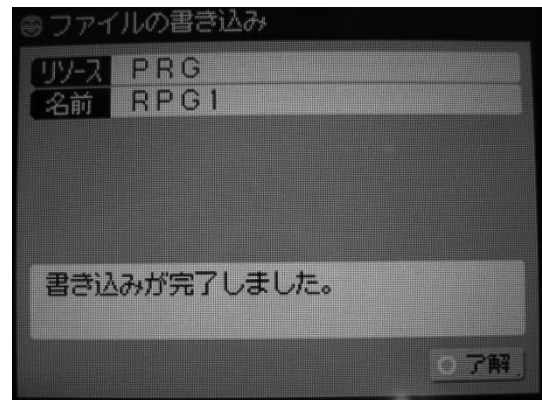
SAVE 命令の文法は以下のとおりです。

```
SAVE  "RPG1"
      ファイル名
```

SAVE に続けて、「"」(ダブルクォーテーション)で囲んで保存するファイル名を指定します。

Enter キーを押すと、下画面に確認が表示されません。「了解」ボタンで元に戻ります。

なお、キーボード上のファンクションキーの3番に「SAVE」が設定されているので、これをタップすると一度に入力できます。



ファイルがちゃんと保存されているか、確認しましょう。

保存されたファイルを一覧で見る「FILES」(ファイルズ)命令を使います。

```
FILES
PRG:RPG1
OK
```

「FILES」と打って Enter キーを押すと、保存されているファイルが一覧で表示されます。

プログラムファイルは「PRG:」に続いてファイル名が表示されます。

なお、ファンクションキーの1番に「FILES」が設定されているので、これをタップすると一度に入力できます。

★保存したプログラムを読み込む

保存してあるプログラムを読み込むには、「LOAD」(ロード)命令を使います。

```
LOAD  "RPG1"
      ファイル名
```

ファンクションキーの2番に「LOAD」が設定されているので、使ってください。

●キャラクターを十字キーで操作する

これまでは、キャラクターをプログラムで自動的に動かしていました。
十字キーで上下左右に操作できるようにしてみましょう。

★キー入力の読み取り

キーやボタンの入力を読み取るには、「BUTTON」(ボタン)関数を使います。
新しいプログラムを作ります。実行モードで「NEW」(ニュー)命令を実行して、プログラムをまっさらにしします。

編集モードで、以下の3行のテストプログラムを入力してください。

```

1 @TEST
2 PRINT BUTTON( )
3 GOTO @TEST

```

PRINT 命令で BUTTON 関数の値を表示し、それをずっとくり返すプログラムです。
実行すると、画面の上から下へダートと「0」が表示されます。
この状態で、十字キーや A ボタンなどを押してみてください。表示される数字がいろいろ変化するのがわかります。

BUTTON 関数の値は、押されたキーやボタンによって、以下の値になります。

| 押されたボタン | 値 |
|-----------|------|
| 何も押さない | 0 |
| 十字キーの上 | 1 |
| 十字キーの下 | 2 |
| 十字キーの左 | 4 |
| 十字キーの右 | 8 |
| A ボタン | 16 |
| B ボタン | 32 |
| X ボタン | 64 |
| Y ボタン | 128 |
| L ボタン | 256 |
| R ボタン | 512 |
| START ボタン | 1024 |

なお、2つのボタンを同時に押すと、それぞれの値を足し算した値になります。
例えば「十字キーの上」を押しながら「A ボタン」を押すと、 $1+16=17$ になります。
ボタンをいろいろ押して、表の値になることを確認してください。

確認できたら、テストプログラムの3行は削除してください。

★キャラクターを最初の位置に表示

まず、最初の位置
(画面左上)に
キャラクターを
表示します。

| | | |
|--------------------------|----|--------------------|
| タイトル | 1 | '*** RPG2 *** |
| | 2 | |
| 画面クリア | 3 | ACLS |
| | 4 | |
| キャラクターの最初の X座標、Y座標を設定 | 5 | 'サイショ |
| | 6 | X=0 |
| | 7 | Y=0 |
| スプライト0に右向きの子を設定 | 8 | SPSET 0,64,2,0,0,2 |
| 4コマのアニメーションを設定 | 9 | SPANIM 0,4,10 |
| 座標(X,Y)に表示 | 10 | SPOFS 0,X,Y |

★十字キーのチェック

続いて、BUTTON 関数で十字キーを読み取ります。

何度もチェックしないといけないので、変数 B に値を入れます。

```
11 B=BUTTON()
```

BUTTON は関数なので、「=」で変数に値を入れる事ができます。

次に、「十字キーの上」が押されているかどうかを判断します。

判断するには「IF」(イフ)～「THEN」(ゼン)～「ELSE」(エルス)命令を使います。

```
12 IF B==1 THEN Y=Y-1
```

IF 命令の文法は以下のとおりです。

| | | | | | |
|----|------|------|---------------------|------|-----------------------|
| IF | B==1 | THEN | Y=Y-1 | ELSE | ～ |
| | 条件式 | | 条件が成り立つ時に 実行する命令 | | 条件が成り立たない時に 実行する命令 |

| | |
|--------|---|
| 条件式 | 判断する条件の式を書く。式の例は以下のとおり。 「A==0」…A が 0 に等しい 「A<0」…A が 0 より小さい(0 はふくまれない) 「A>0」…A が 0 より大きい(0 はふくまれない) 「A<=0」…A が 0 以下(0 もふくまれる) 「A>=0」…A が 0 以上(0 もふくまれる) 「A!=0」…A が 0 に等しくない |
| THEN ～ | 条件が成り立った時に実行する命令を書く。 |
| ELSE ～ | 条件が成り立たない時に実行する命令を書く。 ELSE 以下は省略可能。 |

あらためてプログラムに戻りますが、

```
12 IF B==1 THEN Y=Y-1
```

このIF命令は、「もし十字キーの上が押されていたら、y座標を1減らす」になります。

プチコンの画面では、y座標を1減らすと、上へ1ドット分移動することになります。

※プログラムで「Y=Y-1」は、「YとY-1が等しい」ではなく、「Yから1を引いた値をYに入れる」、つまり「Yを1減らす」という意味になります。

同じように、十字キーの下、左、右が押された時のIF命令を書きます。

(似たような行なので、コピー・ペーストするといいでしょ)

```
12 IF B==1 THEN Y=Y-1
13 IF B==2 THEN Y=Y+1
14 IF B==4 THEN X=X-1
15 IF B==8 THEN X=X+1
```

これで座標が変えられたので、もう一度画面にキャラクターを表示します。

```
16 SPOFS 0, X, Y
```

最後に、GOTO 命令で最初に戻します。

また「@ARUKU」ラベルを作って、そこへ戻すようにします。

これで実行してみましょう。

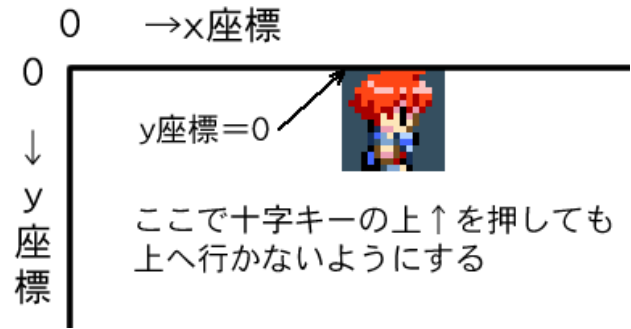
十字キーを押すとキャラクターが動きますが、あまりにスピードが速くて、画面の外へ飛んで行ってしまいます。

キャラクターを表示する SPOFS 命令の次に、WAIT 命令を入れて、動きを遅くしましょう。

```
1 '*** RPG2 ***
2
3 ACLS
4
5 'サイショ
6 X=0
7 Y=0
8 SPSET 0, 64, 2, 0, 0, 2
9 SPANIM 0, 4, 10
10 SPOFS 0, X, Y
11
12 ' --- アルク ---
13 @ARUKU
14
15 ' シュウシキ キー
16 B=BUTTON( )
17 IF B==1 THEN Y=Y-1
18 IF B==2 THEN Y=Y+1
19 IF B==4 THEN X=X-1
20 IF B==8 THEN X=X+1
21
22 SPOFS 0, X, Y
23 WAIT 1
24 GOTO @ARUKU
```

★複数の条件で判断する

今のままではキャラクターが画面の外へ出てしまい、操作できなくなってしまう。これを防ぐには、例えばキャラクターが画面の一番上にいる時は、「十字キーの上」を押してもそれ以上動かないようにして、キャラクターが外へ出ないようにします。



IF 命令の条件式に、「AND」(アンド)でつないで、2つの条件を入れます。

```
17 IF B==1 AND Y>0 THEN Y=Y-1
```

この条件式は、「B が 1 と等しい、かつ、Y が 0 より大きい」という意味になります。

両方の条件が成り立った時だけ、「THEN」以下の命令が実行されます。

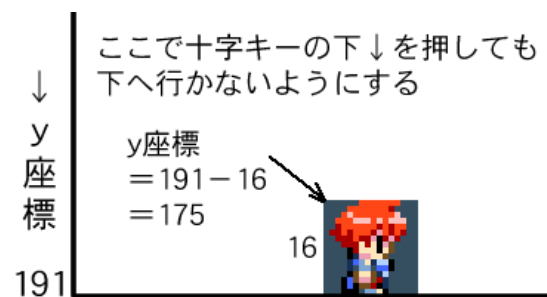
(※「～かつ～」は、中学の数学の「集合」で習います。数式では「 $A \cap B$ 」などと書きます)

他のIF命令も、同様に AND 条件式を入れます。

```
17 IF B==1 AND Y>0 THEN Y=Y-1
18 IF B==2 AND Y<175 THEN Y=Y+1
19 IF B==4 AND X>0 THEN X=X-1
20 IF B==8 AND X<239 THEN X=X+1
```

上限の「175」や「239」は、プチコンの画面レイアウトと、スプライトの大きさ(16×16ドット)から決まります。

これで、キャラクターが画面の外へ出なくなります。



2つの条件をつなぐ条件式は、もう一つ「OR」(オア)があります。

```
IF B==1 OR Y>0 THEN Y=Y-1
```

この条件式は、「B が 1 と等しい、または、Y が 0 より大きい」という意味になります。

2つの条件のどちらかが成り立つと、「THEN」以下の命令が実行されます。

(※「～または～」も、中学の数学の「集合」で習います。数式では「 $A \cup B$ 」などと書きます)

★マルチステートメント

今のプログラムでは、キャラクターがいつも右向きのまま歩いていて、他の方向へ移動する時は不自然です。最初に作ったプログラム「RPG1」のように、動く方向によって表示するスプライトを変えないといけません。

例えば、十字キーの上が押された時のIF命令は、

```
IF B==1 AND Y>0 THEN Y=Y-1
```

となっていて、条件式が成り立った時は「THEN」以降で y 座標を 1 減らしています。

キャラクターを上向きに変えて移動するには、

```
Y=Y-1
SPSET 0,76,2,0,0,2
SPANIM 0,4,10
```

※76番は上向き(後ろ向き)
のスプライト

の3つの命令を「THEN」の後ろに書かなくてははいけません。

「THEN」の後ろには1行分しか書けないので、3つの命令をつなげて書く必要があります。

プチコンで2つ以上の命令をつなげて書くには、「:」(コロン)で区切って書きます。

```
17 IF B==1 AND Y>0 THEN Y=Y-1:SPSET 0,76,2,0,0,2:SPANIM 0,4,10
```

これで、条件が成り立った場合に、「Y=Y-1:SPSET 0,76,2,0,0,2:SPANIM 0,4,10」の命令が全て実行されます。

同じように、十字キーの下、左、右のそれぞれの処理をします。

```
17 IF B==1 AND Y>0 THEN Y=Y-1:SPSET 0,76,2,0,0,2:SPANIM 0,4,10
18 IF B==2 AND Y<175 THEN Y=Y+1:SPSET 0,68,2,0,0,2:SPANIM 0,4,10
19 IF B==4 AND X>0 THEN X=X-1:SPSET 0,72,2,0,0,2:SPANIM 0,4,10
20 IF B==8 AND X<239 THEN X=X+1:SPSET 0,64,2,0,0,2:SPANIM 0,4,10
```

これで、移動する時にそれぞれの方向に向いたキャラクターが表示されます。

このプログラムでは、キーが押された時に、毎回キャラクターとアニメーションを再設定してしまうので、キーを押し続ける間はアニメーションが 1 コマ目で止まってしまいます。キャラクターがすべて動いているように見えてしまいます。

これを防ぐには、1 回前のキー入力を記録しておいて、「1 回前と今回のキー入力が違う時だけ(=向きが変わった時だけ)、スプライトを再設定する」プログラムにするといいでしょう。

(※他の解決方法もありますので、考えてみてください)

```

1 '*** RPG3 ***
2
3 ACLS
4
5 'サイショ
6 X=0
7 Y=0
8 SPSET 0,64,2,0,0,2
9 SPANIM 0,4,10
10 SPOFS 0,X,Y
11
12 '---- アルク ----
13 @ARUKU
14
15 'シ ユウシ ヌ キー
16 B=BUTTON( )
17 IF B==1 AND Y>0 THEN Y=Y-1:S=76
18 IF B==2 AND Y<175 THEN Y=Y+1:S=68
19 IF B==4 AND X>0 THEN X=X-1:S=72
20 IF B==8 AND X<239 THEN X=X+1:S=64
21
22 'ムキカ ヌ カワッタラ スプライト サイセツタイ
23 IF B!=B0 THEN SPSET 0,S,2,0,0,2:SP
ANIM 0,4,10
24
25 SPOFS 0,X,Y
26 WAIT 1
27
28 'コンカイノ キーヲ ホソバン
29 B0=B
30
31 GOTO @ARUKU

```

十字キーが押された時、X座標・Y座標を変えると共に、それぞれの向きのキャラクター番号を変数Sに設定

今回のキー入力Bと、1個前のキー入力B0が違う(=向きが変わった)時は、スプライトをキャラクターSに変更して、アニメを再設定

今回のキー入力Bを、変数B0に保存して、次のループで使う