プチコン3号
SMILE BASIC

僕が考えた最強のこんにちはプチコン3号

SmileBasic 入門編

ぼくが考えた最強の「こんにちはプチコン３号」

プチコン３号ってなに？
スマイルブームによって開発されたニンテンドー3DS 用のプログラム言語。
プログラムっていうのはコンピューター（パソコンなど、3DS もコンピューターだよ）を動かすための命令のこと。
人とコンピューターは直接会話をすることができないから、プログラム言語を使ってコンピューターに命令をするんだ。

例えばこんなの...

```
A=10
B=50
C=A+B
PRINT C
```

言葉には「日本語」「英語」「フランス語」「中国語」など数多くあるけど、プログラム言語にもいろいろな種類があるよ。
例えば...「BASCI」「C 言語」「Java」が有名かな？
他にもたくさんあるし、新しいプログラム言語が作られることもあるよ。
プチコン３号は「SMILE BASIC」という「BASIC」の仲間なんだ。

「BASCI」というと、その名の通り基本を勉強するための言語と思われがちだけど．．．
「SMILE BASIC」はゲームが作りやすいように拡張（ゲーム作り専用の命令が増えてる）されてるんだ。
割と簡単に 3DS の機能を自分自身の手で活用することができる、という夢のような言語が「SMILE BASIC」なんだ。
「SMILE BASIC」はスマイル ベーシックって読むんだ。

プログラムって難しいの？
そりゃ、売ってるようなゲームを作ろうと思うとはっきり言って難しいよ。
というかこれを書いている僕も作れません。
だから売っているような、すごく派手なゲームを作りたい人はこんなのを読んでる場合じゃありません。

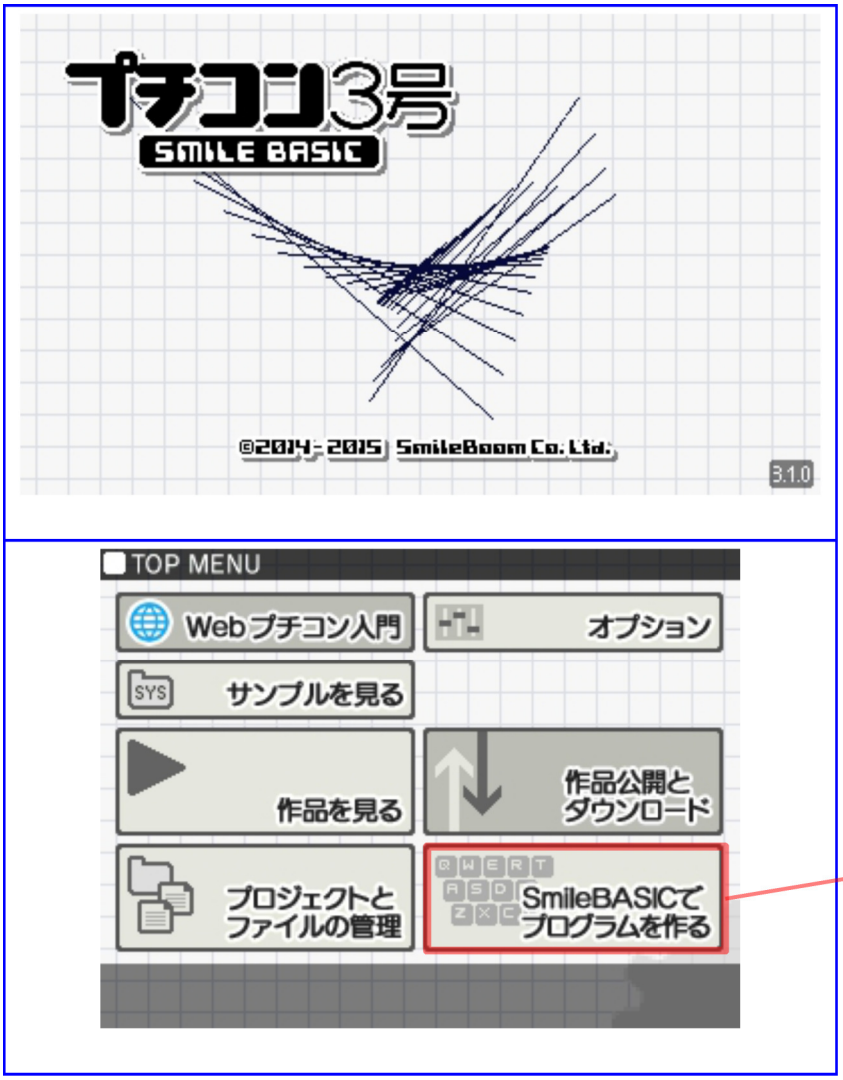
でも、なにごとも一歩から。
これはプチコン３号で初めてプログラミングをする人に向けて書いています。
まったくの初心者でも大丈夫。
プログラムを作るということはそんなに難しいことではありません。
ほんの少し覚えるだけでもプログラムは作れるからね。

さっそくプログラミング、と行きたいところだけとちょっとだけ準備をしよう。
3DS とプチコン３号の用意はいいかな？

さあ、3DS の電源を入れてプチコン３号を起動しよう！

１．さいしょのさいしょ

プチコン３号は起動できたかな？
この画面がでればＯＫだ。



ここをタッチしよう。
さあ、ここがプログラミングの世界の入り口だ！

3DS はもってるけど、プチコン３号をもっていないというキミへ
プチコン３号はダウンロードソフトです。
お店に買いに行っても売っていません。
3DS をインターネットにつなげないと買うことはできません。
価格は 1000 円です。
お父さん、お母さんに相談してみよう。
この本を見せて「プチコン３号って、プログラム作れるんだよ！」って言うと理解してもらいやすかもしれないね。

さあ、プログラムを作る画面に移ったと言いたいところだけど、これは作ったプログラムを動かす画面なんだ。
これを DIRECT モードって言うんだ。

このままではプログラミングができないので、EDIT モードに切り替えるよ。
プログラムは EDIT モードで作るんだからね。
（DIRECT はダイレクト、EDIT はエディットって読むよ。）

ここをタッチして、EDIT モードに切り替えよう！

ついにプログラミングできる画面にたどりついたよ。

DIRECT モードに戻す時もここを使うよ。
ここで DIRECT モードと EDIT モードを切り替えるってことはわかったかな？

ついでにもう少し説明しておこう。
この緑のところで、入力する文字を変えられるよ。
例えば、「ア」のところを押すと...カナ入力になる。

「ひらがな」や🏠♪🐱などの記号もあるので、いろいろためしてみよう。

２．足し算

まずは簡単なところで、足し算をしてみよう。

［例１］ 足し算

```
1  A=10↵
2  B=50↵
3  C=A+B↵
4  PRINT  C↵
```

これは最初の方に例として載せてあったプログラムだけど、実はこれがBASICだったんだ。

↵マークは改行をあらわすんだ。
「ENTER」を押すとこのマークが付いて次の行に移ることができるよ。
（後で出てくるけど、画面上の１行とプログラムの１行がいつも一致するとは限らないから↵マークが必要になるんだ。）

プチコン３号でキミもプログラミングしてみよう！
足し算プログラムを例のとおりに入力してね。

どうかな、こんなふうに入力できたかな？

じゃあ、プログラムを動かしてみよう。
プログラムを動かすのはDIRECTモードだったね。

ここを押してDRECTモードに切り替えるよ。

まずは画面をキレイにしよう。
プログラムが正しく動いたかわかりにくいので画面をクリアするよ。

コマンドは **CLS** だ！

> DIRECT モードではコマンドを入力していろいろな操作をするんだ。
> 画面をクリアするコマンドは CLS だよ。

CLS と入力して「ENTER」を押してみよう！

よけいなものが消えて画面がキレイになったよね？
CLS はよく使うので覚えておいてね。

いよいよプログラムの実行だ。
プログラムの実行コマンドは **RUN** だ！
RUN と書いてランと読むんだよ。

RUN と入力して「ENTER」を押そう！

答え合わせ

どうかな、この画面のようになったかな？
もし、違う画面になっていたら EDIT モードに戻してプログラムを見直してみよう。

プログラムの解説に入る前に、せっかく作ったプログラムなので保存しておこう。
プログラムの保存には **SAVE** コマンドだ！
SAVE と書いてセーブと読むよ。


```
SAVE"TASHIZAN
     ↑
```

これはダブルクォーテーションという記号だ。
ちょんちょんが二つくっついて一文字なんだ。

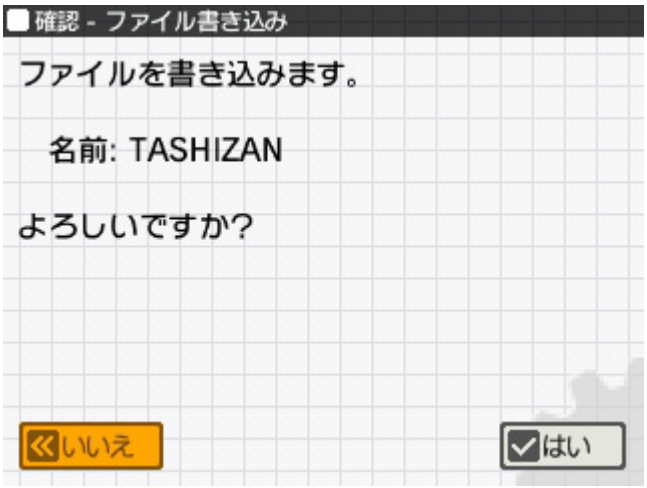
ダブルクォーテーションはこれだよ！

```
SAVE"TASHIZAN
     ↑
```

SAVE コマンドの後には、保存する名前（ファイル名）を入力するんだ。
とくに決まったルールはないので、自分がわかりやすい名前を付けよう！
ただし 14 文字までなので、そこだけは注意が必要だ。
ここでは **TASHIZAN** にしたよ。

それじゃあ、セーブ実行だ！
SAVE コマンドが入力できたら「ENTER」を押して。

こんなふうに、下画面にセーブしてもいいか聞いてくるので「はい」を押してね。



これで、セーブできました。
こんな感じで下画面のメッセージ（ダイアログという）にしたがって、操作する場面はときどきあるからよく読んで「はい」「いいえ」を押してね。

ついでにロード（読み込み）についても説明しておくよ。
セーブしたプログラムを呼び出すには **LOAD** コマンドを使うんだ。


```
LOAD"TASHIZAN
```

LOAD と書いてロードと読む。
さっきの SAVE とよく似てるね。
セーブしたファイル名を使ってプログラムを呼び出すよ。
ロードするときもダイアログが表示されるので、よく読んでから実行してね。

セーブする名前を変えれば、たくさんのプログラムを保存することができるんだ。
名前を間違えるとプログラムが消えちゃったりするから気を付けよう！

さらについでに、ファンクションキーについても説明しておくね。

ここを見てね。
LOAD”、SAVE”、RUN、なんか見たことがある文字がならんでるね。

実はここを押すと...
一回押すだけでコマンドが入力されるんだ。
ためしに押してみて？

ファンクションキーって言うんだけど、とっても楽ちんな機能だね。
基本はこの五つのコマンドが登録されてるけど、自分の好きなコマンドに変更することもできるよ。
よく使うコマンドに変更してもっと便利にしちゃおう！

一番左に **FILES** っていうコマンドがあるので、これも試してみよう！
FILES と表示されたら「ENTER」を押そう。

今までに保存したファイルが一覧表示されるよ。
ファイル名を忘れちゃってもこれで安心だね。

これは僕のプチコン３号のファイルなんだ。
こんなふうに名前を付けてるよ。
ちょっとゴチャゴチャしてるかな？

ドキドキ、じゃあプログラムを１行ずつ確認していこう！

［例１］ 足し算

```
1  A=10
2  B=50
3  C=A+B
4  PRINT  C
```

まずは１行目

```
1  A=10
```

なんとなく意味はわかると思うけど、こんなイメージで考えるとわかりやすいかな？

```
1  A←10
```

やじるしになっているね。
これは A に １０ を入れるってことなんだ。

A は数値を入れるための入れ物ってことになるんだけど、この入れ物を変数（へんすう）と言う。
ちなみに、変数に値を入れることを代入（だいにゅう）って言うんだよ。

言いかえると、１行目のプログラムは変数 A に １０ を代入するってことなんだ。

じゃあ、２行目に行くよ。

```
2  B=50
```

これはさっきとほとんど同じだね。

そう、変数 B に ５０ を代入しているんだ。

調子が出てきたところで、３行目に行くよ。

```
3  C=A+B
```

代入しているのは同じだけど、ちょとややこしい？

じゃあ、これも置きかえてみるよ。

```
3  C←A+B
```

A＋B を C に入れるってことはわかるかな？
A と B は変数なので、中に入っている数値を足して、その答えを C に入れるってことなんだ。

つまり．．．
変数 A に入っている数値と、変数 B に入っている数値を足して、変数 C に代入してるってことだね。

A には １０、B には ５０ が入っているから、１０＋５０ で答えは ６０ だ。
C には ６０ が入ってるってことだね。

いよいよ最後の４行目だね。

```
4  PRINT  C
```

新しい命令が出てきたよ。
この命令はプリントと言って画面に文字を表示するときに使うんだ。
足し算をしても答えを表示しなかったら確認できないからね。

さっきの答え合わせの画面を思い出して．．．

これだね。

答えは、ちゃんと６０って表示されてるね。

この命令がちゃんと変数Ｃの値を表示してくれたってことだ。

```
    4  PRINT  C↵
```

でも、これは何でＣって表示されないんだろうね？
ＰＲＩＮＴ　Ｃって書いてあるんだから答えはＣってなってもよさそうなのに。

これも、やっぱりＣが変数だからなんだ。

ちなみに画面にＣを表示したいときはこんなふうに書くんだよ。

```
    4  PRINT  "C"↵
```

ダブルクォーテーションでかこうと、そのままＣを表示してくれるんだ。

ためしにダブルクォーテーションを付けてみて？
答えの６０のかわりにＣが表示されるよ。
このとき、ダブルクォーテーションは表示されないんだ。
結果は自分で試して確認してね。

これで４行目まで説明が終わった。
はじめてのプログラムなので最後にまとめておくよ。

```
    1  A=10↵          変数Ａに１０を入れる。
    2  B=50↵          変数Ｂに５０を入れる。
    3  C=A+B↵         変数ＡとＢを足して変数Ｃに入れる。
    4  PRINT  C↵      変数Ｃの値を画面に表示する。
```

こんなふうに言葉におきかえるとわかりやすくならないかな？

プログラム言語をおぼえることもプログラムを作ることも時間がかかることだけど、こうやって一つずつ学んでいけばきっと思い通りの作品が作れるようになるよ。

「千里の道も一歩から」だね！

変数についてもう少しだけくわしく見ていくよ。
まずは変数の名前、変数名についてだ。

さっきは
```
A=10
```
だったよね？

これを
```
KAZU1=10
```
っていうようにしてもいいんだ。

ロンよりハーマイオ．．．じゃなかった、論より証拠。
これでも結果は同じになるよ。
```
1 KAZU1=10↵
2 KAZU2=50↵
3 KOTAE=KAZU1+KAZU2↵
4 PRINT KOTAE↵
```

変数名は A のように 1 文字でなければいけないわけではないんだ。
KOTAE のように自分がわかりやすいように付けていいんだ。

英語でバチッとキメてもいいし、日本語をローマ字にしてもＯＫ。
プチコン 3 号はすご～く長い変数名でも大丈夫だけど、あまり長すぎると入力するのが大変なのでほどほどにね。

こんどは文字列についてだよ。

さっきのこれを思い出して？
```
4 PRINT "C"↵
```
これは画面に C を表示するんだったよね。

これをちょっと変えてみよう。
```
4 PRINT "コンニチハ"↵
```
こうするとｺﾝﾆﾁﾊって表示されるようになるんだ。

ダブルクォーテーションでかこったものは文字列として扱われるということ。
ここまでは、いいかな？

じゃあ、この文字列を変数に入れてみるよ。

簡単だって？
```
A="コンニチハ"↵
```

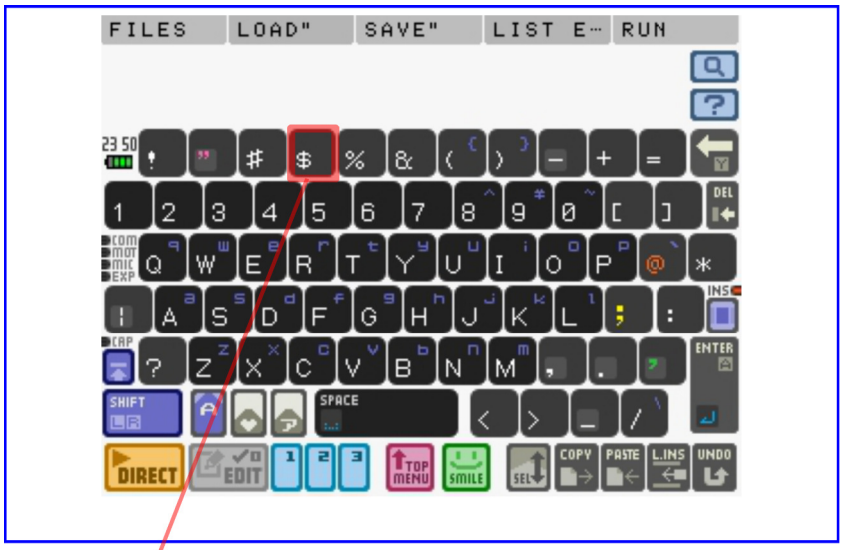
う～ん、おしい。
これじゃダメなんだ。

正しくはこう。

```
A$="コンニチハ"
```

文字列の変数名には、おしりに$を付けるっていうキマリがあるんだ。
これはドルマークって言うんだけど、プログラマーだとダラーって呼ぶ人の方が多いのかな？

ダラーって言って通じなかったらドルマークって言いなおせばいいと思うよ。
どんな言い方をするかじゃなくて、相手にちゃんと伝わるかどうかが大切だからね。



ダラーはこれだよ！

変数名は自由に付けられるのでこれでもいいよね？

```
AISATSU$="コンニチハ　キョウハ　イイテンキデスネ"
```

やっぱり簡単だったね。
$が付くだけだった。

じゃあ、文字列を使ったプログラムを作ってみるね。

```
1  AISATSU$="キョウハ　イイテンキデスネ"
2  ASA$="オハヨウゴザイマス　"
3  HIRU$="コンニチハ　"
4  PRINT　ASA$;AISATSU$
5  PRINT　HIRU$;AISATSU$
```

じっくり見れば、なにも難しくはないよね？

一つだけ．．．
PRINT に注目してね！

```
4  PRINT　ASA$;AISATSU$
```

↑
これはセミコロンっていうんだ。
PRINT するときに、くっつけて表示してくれるようになるんだ。

セミコロンはこれだよ！

RUN してみるよ～。

おまけ
足し算プログラムを文字列の足し算にしちゃおう！

```
1  A$="10"↵      変数Ａ$に文字列”１０”を入れる。
2  B$="50"↵      変数Ｂ$に文字列”５０”を入れる。
3  C$=A$+B$↵     変数ＡとＢの文字列を足して変数Ｃに入れる。
4  PRINT C$↵     変数Ｃの値を画面に表示する。
```

さあ、結果はどうなったかな？

四則演算（加算，減算，乗算，除算）の算術演算子

加算（かさん）は、足し算のことだから．．．
　減算（げんざん）　＝引き算
　乗算（じょうざん）＝掛け算
　除算（じょざん）　＝割り算
だね。
これは大丈夫だね？

足し算は＋だったからキミたちが習う算数と同じだったね。
引き算も－なので算数と同じ。

でも、ここからがちっと違う。

掛け算は＊で、割り算は／になるんだ。

＊はアスタリスク、／はスラッシュって言うんだ。

コンピュータの場合、使える文字が限られているからこんなふうに置きかえられているんだ。
これらの記号をまとめて算術演算子（さんじゅつえんざんし）と言うんだ。

足し算プログラムをこんなふうに書きかえると。

```
1  A=10
2  B=50
3  C=A*B
4  PRINT  C
```

あっという間に掛け算プログラムに変身だ。

こんどは割り算プログラムに変身。

```
1  A=10
2  B=50
3  C=A/B
4  PRINT  C
```

さて、割り算の話になったのでゼロ除算エラー（ぜろじょざんえらー）にもふれておこう。

割り算プログラムをさらにチョイチョイと書きかえると．．．

```
1  A=10
2  B=0
3  C=A/B
4  PRINT  C
```

書きかえるというより 5 を消したといったほうがいいな。
5 0 を0にした。

B には0が入ってるってことだね。

つまり 3 行目は、
　C=10/0
ってことに．．．

算数だとゼロでは割れないって習ったよね。

これで RUN するとどうなるか．．．？

エラーになっちゃった。
この画面は英語で「ゼロ除算になってるよ」って書いてあるんだ。
プログラムでもやっぱり答えは出ないんだ。

この割り算プログラムだとすぐに間違いに気づくけど、複雑なプログラムだと見つけるのが難しいこともある。
バグの原因にもなるので、ゼロで割ることがないように注意しなければいけないよ。

次は、足し算プログラムを少しずつ改造していって、足し算ゲームにしていくよ。

ぷちコラム「エラーいこっちゃ！」

エラーが発生したので、エラーについてもう少し詳しく見て行こう。

これはゼロ除算エラーだったね。

エラーの行だけ抜き出すよ。

```
Divide by zero in 0:3
```

最後の数字（赤くしてある）に注目しよう。
これはエラーが発生したプログラムの行番号をあらわしているんだ。

つまり、3行目でゼロ除算エラーが起きているってことなんだ。

プログラムを確認してみるよ。

```
1  A=10↵
2  B=0↵
3  C=A/B↵      ←やっぱりココだ！
4  PRINT C↵
```

こんなふうに、行番号とメッセージの内容をたよりに間違いを見つけるんだ。
たまに、行番号とは違う行が原因で不具合が起きることもあるから注意ね。
このサンプルでもエラーは3行目だけど、2行目でゼロを入れてることが原因だし。

エラーメッセージは他にもあるんだ。
これはプチコン3号の説明書から一部を抜き出したものなんだけど。
最初のうちはSyntax errorをよく見るかも知れないね？

Syntax error	おかしな文法が含まれている
Out of range	数値が有効範囲を超えた
Out of memory	メモリーに空きがない
Undefined label	GOTO, GOSUB, IF などで指定した分岐先のラベルがない
NEXT without FOR	FORがないのにNEXTがある
Out of DATA	READで読むべきデータがDATA文にない
Illegal function call	命令や関数の呼び方が間違っている
:	:
:	:

参照サイト：http://smileboom.com/special/ptcm3/e-manual/36.php

３．足し算、答えを入力
まずは答えを入力して、答え合わせができるところまでを考えるよ。

［例２］ 足し算、答えを入力

```
 1  A=10
 2  B=50
 3  C=A+B
 4  PRINT  A;"+";B;"=";
 5  INPUT "コタエヲニュウリョク";KOTAE
 6  IF  C==KOTAE  THEN
 7   PRINT  "セイカイ"
 8  ELSE
 9   PRINT  "マチガイ  セイカイハ  ";C
10  ENDIF
```

新しい命令は目立つように赤くしといたよ。

じゃあ、プログラムを見て行くよ。
3 行目までは”**［例１］** 足し算”と同じだ。

4 行目は問題を画面に表示しているんだ。
セミコロンとダブルクォーテーションはもう習ったから、よく見ればわかるよね？

```
 4  PRINT  A;"+";B;"=";
```

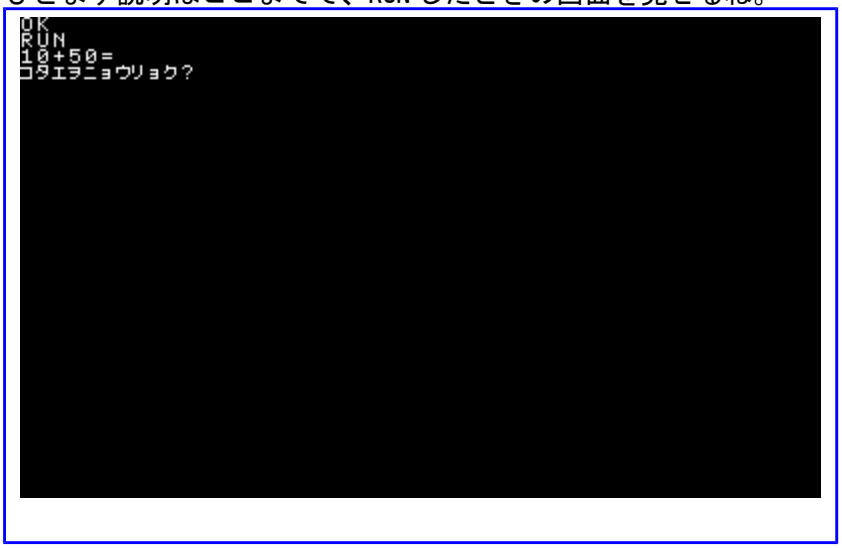
5 行目は新しい命令だ。

```
 5  INPUT "コタエヲニュウリョク";KOTAE
```

ＩＮＰＵＴ はキーボードからの入力を受け付ける命令だよ。
読み方はインプットだよ。

"コタエヲニュウリョク" は文字列で、ＫＯＴＡＥ は変数名なんだ。

ひとまず説明はここまでで、RUN したときの画面を見せるね。



問題が表示されて．．．
何か入力があるまでプログラムはここでストップするんだ。

これは ＩＮＰＵＴ 命令が何か入力されるまで待ってるってことなんだ。
で、入力された内容は ＫＯＴＡＥ に入れられる。

これが ＩＮＰＵＴ 命令の役割だ。

次はＩＦ文だ。
これが出てくるとグッとプログラムらしくなってくるよ。

```
 1  A=10
 2  B=50
 3  C=A+B
 4  PRINT A;"+";B;"=" 
 5  INPUT "コタエヲニュウリョク";KOTAE
 6  IF C==KOTAE THEN
 7   PRINT "セイカイ"
 8  ELSE
 9   PRINT "マチガイ セイカイハ ";C
10  ENDIF
```

ＩＦ文は「もしも、〇〇なら、××する」というように何か判定や判断をしたいときに使うんだ。

ＩＦ文はこの形が基本になるんだ。

```
IF 条件 THEN
  条件に合うときの処理
ELSE
  条件に合わないときの処理
ENDIF
```

他の書き方もできるけど、慣れるまではこの形を守るようにしよう。

読み方だよ。

ＩＦ	イフ
ＴＨＥＮ	ゼン
ＥＬＳＥ	エルス
ＥＮＤＩＦ	エンドイフ

さあＩＦ文を見て行くよ。
赤字のところは大丈夫だね？
問題は、緑のところ。

```
 6  IF C==KOTAE THEN
```

これが条件なんだ。

抜き出してみるよ。

```
C==KOTAE
```

Ｃ とＫＯＴＡＥ は変数だってのはいいね？

となると真ん中の＝＝が何を意味するのか？！

これはイコールが二つ並んでいるんだ。
Ｃ とＫＯＴＡＥ は同じですか？って聞いてるんだよ。

まとめてみるよ。

```
 6  IF C==KOTAE THEN          C と KOTAE は同じですか？
 7   PRINT "セイカイ"          同じ  正解と表示
 8  ELSE
 9   PRINT "マチガイ セイカイハ ";C   違う  間違い，正しい答えを表示
10  ENDIF
```

ＩＦ文で正解と不正解の判定をしているんだね。

もう一度プログラムを見直すよ。

```
 1  A=10
 2  B=50
 3  C=A+B
 4  PRINT  A;"+";B;"="
 5  INPUT  "コタエヲニュウリョク";KOTAE
 6  IF  C==KOTAE  THEN
 7   PRINT  "セイカイ"
 8  ELSE
 9   PRINT  "マチガイ  セイカイハ  ";C
10  ENDIF
```

C にはプログラムが計算した、足し算の結果が入っているのはいいね？

KOTAE には INPUT 命令でプレーヤーが入力した答えが入っている。
これもいいね？

その二つを IF 文で比較して、正解，不正解の判断をしている。
これもいいね？

正解の時の処理と不正解の時の処理があって、最後は ENDIF で閉じて終了だ！

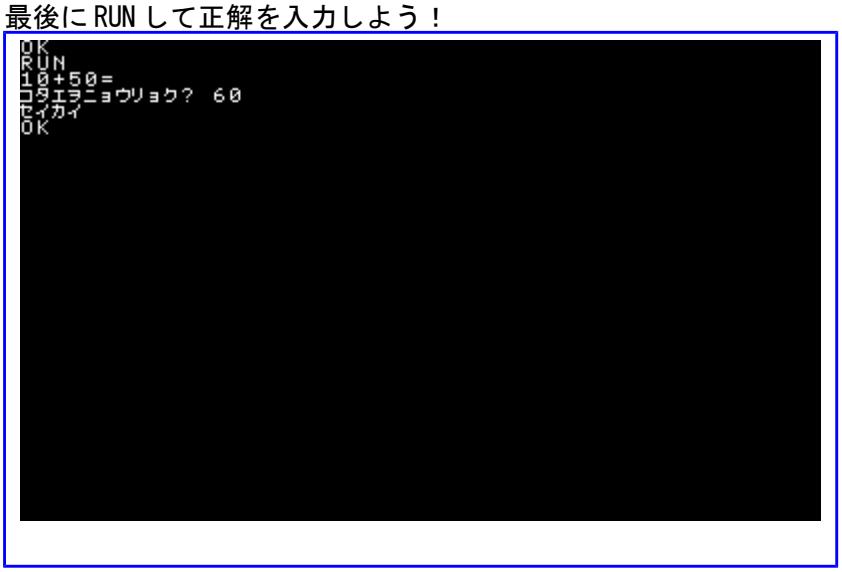
不正解のときの処理が必要なければ ELSE を書かなくてもいいよ。

```
IF  条件  THEN
  条件に合うときの処理
ENDIF
```

＝＝以外の条件の書き方も載せておくよ。
算数，数学と同じようなイメージだよね。

```
IF  A==B  THEN     AとBは等しい
IF  A!=B  THEN     AとBは等しくない
IF  A>B   THEN     AはBより大きい
IF  A<B   THEN     AはBより小さい
IF  A>=B  THEN     AはB以上
IF  A<=B  THEN     AはB以下
```

最後に RUN して正解を入力しよう！



IF 文はとっても重要なので何度も見直してマスターしちゃおう！

ぷちコラム「インデントはいい伝統？」

あれ？
もうＩＦ文の説明は終わったかと思ったのに、またＩＦ文だ。
でも、こんどはＩＦ文自体は関係ないんだ。

```
 6 IF C==KOTAE THEN
 7  PRINT "セイカイ"
 8 ELSE
 9  PRINT "マチガイ セイカイハ ";C
10 ENDIF
```

ピンク色になっているところに注目してみて。
気付いていたかもしれないけど、じつは一文字分スペースを入れてあるんだ。

なんでこんなことをするの？

スペースを入れて一段下げることで、「どこからどこまでがＩＦ文の内容なのか」が分かりやすくなるんだ。
分かりやすくなると思って僕はスペースを入れてる、と言ったほうがいいかな？
ちなみに、こうやって一段下げるのをインデントって言うんだ。

でもルールで決まっているわけではないから、スペースを入れなくてもいいんだ。

```
 6 IF C==KOTAE THEN
 7 PRINT "セイカイ"
 8 ELSE
 9 PRINT "マチガイ セイカイハ ";C
10 ENDIF
```

スペースが入っているのとどっちが分かりやすいかな？

ＩＦ文以外でもインデントをすることもあるよ。

おおぜいでプログラム開発をするときはルールを決めた方がいい場合もあるけど、まずは自分に分かりやすいことが大切。
バグも防げるし、プログラムを改造するときにもね。

これはＣ言語の一部を抜き出したもの。

```
if (c == kotae) {
    printf("セイカイ");
} else {
    printf("マチガイ");
}
```

インデントは他の言語でもよく使われているんだ。

よし、これで答えを入力できるようになった。

でも．．．
「問題が毎回おんなじでつまんないんだけど」！

だよね～

４．足し算、問題を自動で．．．
今度は問題を 3DS が自動で作るように改造しよう。

［例３］ 足し算、問題を自動で．．．

```
 1  A=(RND(9)+1)*10+RND(10)
 2  B=(RND(9)+1)*10+RND(10)
 3  C=A+B
 4  PRINT  A;"+";B;"="
 5  INPUT  "コタエヲニョウリョク";KOTAE
 6  IF  C==KOTAE  THEN
 7    PRINT  "セイカイ"
 8  ELSE
 9    PRINT  "マチガイ  セイカイハ  ";C
10  ENDIF
```

新しい命令は目立つように赤くしといたよ。

あれ？
あまり変わってないな。

1，2 行目だけだね。
しかも内容同じだし．．．

じゃあ、さっそく 1 行目を抜き出してみるよ。

```
 1  A=(RND(9)+1)*10+RND(10)
```

注目は赤字のところなんだけど、同じ命令が 2 回使われているだけだからさらに切り出すよ。

一番最後の命令だけを切り出したよ。

```
RND(10)
```

この命令はランダム命令って言えばいいのかな？
RANDOM を略して RND になってると思うんだけど。

RND は乱数を発生させる機能があるんだ。
乱数っていうのは、コンピューターが勝手に作り出す数値と言えば分ってもらえるかな？

例えば、

```
A=RND(10)
```

こうだったとする。

すると A には 0 から 9 までの 10 個の数値のうちのどれかがが入るんだ。
（1 から 10 ではないから注意してね。）
そして何が入るかはプレーヤーには分からないんだ。

RND を使うとコンピューターに数値を決めてもらうことができるってことだ。

どんな感じでＲＮＤを使うか例を書いてみるよ。
ＲＮＤとＩＦ文を組み合わせたじゃんけんだ。

```
 1  A=RND(3)
 2  IF  A==0  THEN
 3   PRINT  "じゃんけん　グー"
 4  ENDIF
 5  IF  A==1  THEN
 6   PRINT  "じゃんけん　チョキ"
 7  ENDIF
 8  IF  A==2  THEN
 9   PRINT  "じゃんけん　パー"
10  ENDIF
```

ＲＮＤ(３)で０から２までのどれかの数値がＡに入れられるので、ＩＦ文で判定してグー，チョキ，パーを表示しているんだ。

ゲームを作るのにＲＮＤはとても重要な命令なんだ。
だってＲＮＤがなかったら毎回同じ結果になってしまうだろ？
グー，チョキ，パーが毎回同じ順番だったらつまんないよね。

じゃあ話をもとに戻して。

```
 1  A=(RND(9)+1)*10+RND(10)
```

ＲＮＤ(９)は ０から ８までのどれかっていうのはいいね。

前半部分を切り出すよ。
(ＲＮＤ(９)+１)*１０

前半部分の全部のパターンを書いてみるよ。

```
(0+1)*10=10
(1+1)*10=20
(2+1)*10=30
(3+1)*10=40
(4+1)*10=50
(5+1)*10=60
(6+1)*10=70
(7+1)*10=80
(8+1)*10=90
```

ＲＮＤ(９)は０から８なので
この９パターンになるんだ。
赤字のところがＲＮＤね。

つまり(ＲＮＤ(９)+１)*１０は１０から９０までをランダムに出すってことだ。
10の位の数値を作ってるってことだね。

それに+ＲＮＤ(１０)で０から９を足しているってこと。

結果．．．
Ａ=(ＲＮＤ(９)+１)*１０+ＲＮＤ(１０)
は １０から９９までのうちのどれかをＡに入れるってことだね。

問題が簡単になりすぎないように、二けたの数値になるように調整しているんだ。

例えば

```
1  A=RND(100)
```

こんなふうにすると。
A は0から99になってしまうので問題が一桁になっちゃうことがある。

じゃあ、 A が10 から 99 になるように他の方法を考えてみようか？

```
1  A=RND(90)+10
```

こうすると、RND(90)は 0 から 89 のどれかで、それに10 を足すから．．．
ちゃんと A は10 から 99 になるっ！

あれ、さっきより簡単になってる？？
難しく考えてそんした感じ。

それに RND を１回しか使ってないからスピードもちょっとだけ速そう。
なるべく簡単な方法を見つけられるようにアレコレ考えてみよう！

ちょっとおまけで、正解するまで終われないように改造しよう。

［例４］ 足し算、正解するまで終われない。

```
 1  @LOOP
 2  A=(RND(9)+1)*10+RND(10)
 3  B=(RND(9)+1)*10+RND(10)
 4  C=A+B
 5  PRINT A;"+";B;"=";
 6  INPUT "コタエヲニョウリョク";KOTAE
 7  IF C==KOTAE THEN
 8    PRINT "セイカイ"
 9  ELSE
10    PRINT "マチガイ セイカイハ ";C
11    GOTO @LOOP
12  ENDIF
```

新しい命令は目立つように赤くしといたよ。
追加するのは**ラベル**と GOTO だ。

1 行目のこれがラベル。

```
 1  @LOOP
```

目印になるように名前を付けておくだけ。
ラベル自体は何もしないよ。

先頭についてる@はアットマーク。

アットマークはこれだよ！

11 行目のこれが GOTO 文。

```
11    GOTO @LOOP
```

GOTO の読み方はゴートゥーかゴーツーで。

GOTO の後ろにラベル名が書いてあるけど、**「そのラベル名まで飛んで行きなさい」**っていう命令なんだ。
GOTO とラベルはセットで使うんだよ。

間違いのときに先頭にもどるように GOTO を追加してあるので、正解するまでこのプログラムは終わらないってことだね。

５．足し算、10問チャレンジ
もっとゲームぽっくするために問題を10問にして何問正解できるか競うように改造しよう。
ちゃんと得点が出るようにもするよ。

［例５］ 足し算、10問チャレンジ

```
 1  CLS
 2  TOKUTEN=0
 3  FOR I=1 TO 10
 4   A=(RND(9)+1)*10+RND(10)
 5   B=(RND(9)+1)*10+RND(10)
 6   C=A+B
 7   PRINT A;"+";B;"="
 8   INPUT "コタエヲニョウリョク";KOTAE
 9   IF C==KOTAE THEN
10    PRINT "セイカイ"
11    TOKUTEN=TOKUTEN+1
12   ELSE
13    PRINT "マチガイ セイカイハ ";C
14   ENDIF
15  NEXT
16  PRINT "アナタノトクテンハ ";TOKUTEN
```

新しい命令は目立つように赤くしといたよ。
あ、そうそうこのプログラムでもインデントしているから注意して見てね。

さあ、問題を10問出すにはどうしたらいいのか？？

足し算プログラムを10個つなげてしまえば．．．
たしかに、それでも動きそうだ。
コピーして貼り付けるだけだから簡単だし。

でも、あんまりいい方法ではないねぇ。
と言うよりもっと楽な方法があるのでそれでやってみよう。

その方法と言うのはコレだ！

```
FOR I=1 TO 10
    :（繰り返ししたい処理）
NEXT
```

これはＦＯＲ文と言って、繰り返し処理がしたいときに使う命令なんだ。

ＦＯＲ　ＴＯ　ＮＥＸＴがセットなんだ。
フォー　ツー　ネクストって呼ぶよ。
　　（トゥー）

この命令のルール。
ＦＯＲとＮＥＸＴのあいだをぐるぐる回るってこと。

そして繰り返す回数は条件で決まる。

```
 3  FOR I=1 TO 10
```

緑にしてあるところが条件なんだ。

Ｉは変数だよ。
これはＩは１から始まって、１０になるまでの10回繰り返すっていう条件なんだ。

でも、これを見るとおかしいよね？

```
FOR I=1 TO 10
    :（繰り返ししたい処理）
NEXT
```

どこにも I＝I＋1 っていう命令が入っていないのになぜ I は 10 になるのか？
不思議だよね。

実験してみよう。

```
1 FOR I=1 TO 10
2  PRINT I
3 NEXT
```

これを RUN すると

ちゃんと I が 1 ずつ増えているのが分かる。

NEXT まで来ると 1 が足される。
わざわざ 1 を足す命令を書かなくても自動的に増やしてくれるのが FOR 文の便利なところでもあるんだ。

じつは．．．
なくても動くから書いてなかったけど、FOR 文には STEP っていうのもあるんだ。

```
1 FOR I=1 TO 10 STEP 1
2  PRINT I
3 NEXT
```

これでも同じ結果になるんだ。
STEP が書いてなければ勝手に STEP 1 ってことにしてくれてるからね。

STEP はステップと言って刻み（きざみ）のことなんだけど、まあ「いくつずつ足すか」ってこと。

だからこんなふうに STEP を 2 にすると

```
1 FOR I=1 TO 10 STEP 2
2  PRINT I
3 NEXT
```

1 3 5 7 9 って 2 つおきに表示される。
そして次の 11 は 10 を超えちゃうから表示されないんだ。

さらにこんなことも

```
 1 FOR I=10 TO 1 STEP -1
 2  PRINT I
 3 NEXT
```

STEP にはマイナスも指定できるんだ。
これだと 10 から 1 までのカウントダウンになるよ。

さあ、プログラムを見直そう。

```
 1 CLS
 2 TOKUTEN=0
 3 FOR I=1 TO 10
 4  A=(RND(9)+1)*10+RND(10)
 5  B=(RND(9)+1)*10+RND(10)
 6  C=A+B
 7  PRINT A;"+";B;"="
 8  INPUT "コタエヲニョウリョク";KOTAE
 9  IF C==KOTAE THEN
10   PRINT "セイカイ"
11   TOKUTEN=TOKUTEN+1
12  ELSE
13   PRINT "マチガイ セイカイハ ";C
14  ENDIF
15 NEXT
16 PRINT "アナタノトクテンハ ";TOKUTEN
```

もう大丈夫だね。
FOR 文で 10 問出せるようになっていることが分かるね？

得点にかかわる部分は青くしておいたよ。
処理としてはこれだけ。
・最初に変数 TOKUTEN に 0 を入れて初期化する。
・正解のときに 1 点追加する処理を追加。
・10 問終わったら得点を表示する。
簡単だね。

あと毎回面倒なので、先頭に CLS を入れるようにしたよ。

よし、じゃあ最後に RUN して遊んでみよう！

```
62+47=
コタエヲニョウリョク? 99
セイカイ
58+20=
コタエヲニョウリョク? 78
セイカイ
10+80=
コタエヲニョウリョク? 90
セイカイ
85+60=
コタエヲニョウリョク? 145
セイカイ
48+90=
コタエヲニョウリョク?
マチガイ セイカイハ 138
91+26=
コタエヲニョウリョク? 117
セイカイ
27+49=
コタエヲニョウリョク? 76
セイカイ
68+79=
コタエヲニョウリョク? 144
マチガイ セイカイハ 147
62+29=
コタエヲニョウリョク? 91
セイカイ
アナタノトクテンハ 8
OK
```

ぷちコラム「アイは永遠に？」

もう一度ＦＯＲ文を見てみよう。

```
1  FOR I=1 TO 10
2   PRINT I
3  NEXT
```

ほかの言語のFOR文も見てみよう。

C言語

```
int i;
for (i = 1; i <= 3; i++){
  printf("こんにちは¥n");
}
```

JavaScript

```
<script>
for (var i=1 ; i<=3 ; i++){
  document.write("こんにちは<br>");
}
</script>
```

ネットで「FOR文」を検索するといろいろなサンプルプログラムが見つかると思うけど、多くのサンプルで変数にＩ（アイ）が使われていることに気付くだろう。

この本でもＩを使っているけど、べつにＩを使わなきゃいけないってルールがあるわけじゃないんだ。
これはFORTRAN（フォートラン）というプログラム言語のなごりだと言われている。

プログラム言語によっては、変数を使う時に前もって型宣言をしなければならないものがあるんだ。
型宣言っていうのはC言語のサンプルの１行目にあるこれ。

```
int i;
```

これは整数型の変数ｉを定義しているんだよ。
こうやって整数，文字列，浮動小数点型などちゃんと決めておかなければならないんだ。
今勉強している「SMILE BASIC」でも型宣言はできるんだ。

FORTRANでも型宣言はあるんだけど、変数名の先頭がＩ，Ｊ，Ｋ，Ｌ，Ｍ，Ｎで始まるものは宣言しなくても整数型になるという暗黙の型宣言というのがあるんだ。
そこで整数と判断できるＩが一般的に使われるようになって、今にいたるということらしい。

複数のFORループを重ねる場合もFORTRANにならってＩ，Ｊ，Ｋ...と使うことも多い。

```
1  FOR I=1 TO 10
2   FOR J=1 TO 10
3    FOR K=1 TO 10
4     PRINT I,J,K
5    NEXT
6   NEXT
7  NEXT
```

さっきも書いたけど決まりがあるわけじゃない。
分かりやすい変数名を使ってもいいんだ。

プログラミングを教える人がＩを使う。
それを習った人もまたＩを使う。
アイは永遠に語り継がれる．．．かも？

10問チャレンジにしたからちょっとゲームっぽくなったけど、なんか物足りないなぁ．．．
あ、そうか時間制限がないからゆっくりやれば全問正解なんて簡単だもんね。

じゃあ、こんどはもっとゲームらしくできないか考えてみよう。

６．足し算、10秒タイムチャレンジ

やっぱり競うなら時間制限があったほうがいいな。
よし、10秒間で何問正解できるかを競うように改造しよう。

【例６】 足し算、10秒タイムチャレンジ

```
 1  CLS
 2  'カウントダウン
 3  T1$=TIME$
 4  T2$=T1$
 5  FOR I=4 TO 1 STEP -1
 6   CLS
 7   IF I!=4 THEN PRINT I
 8   WHILE T1$==T2$
 9    T2$=TIME$
10   WEND
11   T1$=T2$
12  NEXT
13
14  'ゲームカイシ
15  CLS
16  TOKUTEN=0
17  'ジカンヲビョウニ ヘンカンシテ 10ヲ カサン
18  T1S=VAL(MID$(T1$,0,2))*3600+VAL(MID$
    (T1$,3,2))*60+VAL(MID$(T1$,6,2))+10
19  WHILE T1S>T2S
20   A=RND(9)+1
21   B=RND(9)+1
22   C=A+B
23   CLS
24   MON$=STR$(A)+"+"+STR$(B)+"="
25   INPUT MON$;KOTAE
26   IF C==KOTAE THEN
27    TOKUTEN=TOKUTEN+1
28   ENDIF
29   T2$=TIME$
30   T2S=VAL(MID$(T2$,0,2))*3600+VAL(MID$
    (T2$,3,2))*60+VAL(MID$(T2$,6,2))
31  WEND
32  CLS
33  PRINT "アナタノトクテンハ ";TOKUTEN;" テンデス"
```

新しい命令は目立つように赤くしといたよ。
18と30行は折り返しているから注意してね。
プログラムの1行がどこまでなのか分かるようにするために↵マークが付けてあるっていうのは前に説明したよね。

なんか真っ赤になっちゃったけど次の五つが新しく出てきた命令だよ。
TIME$，WHILE，MID$，VAL，STR$

すごくプログラムが長いので前半と後半に分けるよ。
プログラムの機能が前半と後半では違うので分けても問題ないしね。

前半がゲーム開始前のカウントダウンで、後半が足し算タイムチャレンジになっているんだ。

まずは前半のカウントダウンから。

```
 1  CLS
 2  'カウントダウン
 3  T1$=TIME$
 4  T2$=T1$
 5  FOR I=4 TO 1 STEP -1
 6   CLS
 7   IF I!=4 THEN PRINT I
 8   WHILE T1$==T2$
 9    T2$=TIME$
10   WEND
11   T1$=T2$
12  NEXT
```

RUNすると、３，２，１と順に画面に表示するんだ。
このプログラムはゲーム開始前のカウントダウン。
突然問題が表示されても戸惑うからね。

青色に変えてあるのはコメントといって、メモのようなもので命令ではないんだ。

```
 2  'カウントダウン
```

プログラムを読みやすくするために入れるんだよ。
先頭に'を付けてあとは好きなことを書いていいんだ。
文字種の制限もないから、ひらがな，カタカナ，記号...なんでもＯＫ。
'はシングルクォーテーションかアポストロフィーと呼ぶよ。

シングルクォーテーションとアポストロフィーは厳密には別物らしいんだけど特に区別はないんだ。
どっちの読み方でも通じると思うけど、プログラミングのときはシングルクォーテーションって呼ぶことが多いのかな？
英語ではアポストロフィーって習うよね。
前にも書いたけど、どっちが正しいかより相手にきちんと伝わっているかを考えて使うようにしよう！

シングルクォーテーションはこれだよ！

じゃあ先にすすむよ。

```
    3  T1$=TIME$
```

TIME$は今の時間を調べる命令なんだ。
読み方はタイム（＋ダラー）。

文字列で時間が返されるので、受け取る変数名は文字列型のT1$にしてあるんだ。

形式は”HH:MM:SS”なんだけど、これだけでは意味が分かりにくいよね。

```
HH:MM:SS
↓  ↓  ↓
時:分:秒
```

HHは時、MMは分、SSは秒をあらわしているんだ。

例えば．．．
10時15分23秒　→　”10:15:23”
2時45分32秒　→　”02:45:32”
23時01分11秒　→　”23:01:11”

一桁しかないところは0が入れられるんだ。

次はWHILE文。
これもFOR文と同じように繰り返し処理がしたいときに使うんだ。

```
    8  WHILE T1$==T2$
    9    T2$=TIME$
   10  WEND
```

基本形はこれ。
条件が満たされるときに繰り返ししたい処理が実行されるんだ。

```
WHILE 条件
    :（繰り返ししたい処理）
WEND
```

WHILEとWENDはセットなんだ。
WHILEはホワイル、WENDはホワイルエンドかな？
ダブルエンドでいいような気がするけど。

条件はIF文と同じように書くことができるよ。
IF文ふうに書くと「もしも、〇〇なら、繰り返す」ってところかな？

```
WHILE A==B      AとBは等しい
WHILE A!=B      AとBは等しくない
WHILE A>B       AはBより大きい
WHILE A<B       AはBより小さい
WHILE A>=B      AはB以上
WHILE A<=B      AはB以下
```

こういった繰り返し処理のことをループって言ったりもする。
WHILE文はホワイルループって呼ぶこともあるしFOR文もフォーループって言ったりする。

ＷＨＩＬＥループは8～10行なんだけど、それだけだと意味が分からないのでもう少し抜き出すね。

```
  3  Ｔ１＄＝ＴＩＭＥ＄↵
  4  Ｔ２＄＝Ｔ１＄↵
  8   ＷＨＩＬＥ  Ｔ１＄＝＝Ｔ２＄↵
  9     Ｔ２＄＝ＴＩＭＥ＄↵
 10   ＷＥＮＤ↵
```

これがなにをやってるのか分かるかな。
「Ｔ１＄とＴ２＄が同じ（等しい）あいだだけループする」っていうのは分かるね？

で、Ｔ１＄とＴ２＄には何が入っているのかってことだけど、ＴＩＭＥ＄命令で時間を入れている。

さあ、何のためにループしているのか．．．？

これは時間（秒）が変わる瞬間を待ってるんだ。
一つずつ状況を考えて行こう。

・最初Ｔ１＄とＴ２＄には同じ時間が入っている。
・ＷＨＩＬＥループの中ではＴ２＄だけに現在時刻がセットされている。
・ループの条件はＴ１＄とＴ２＄が同じであること。

逆に考えると
　Ｔ１＄とＴ２＄が同じではなくなったときループは終了する
ということ。

例えばRUNした時の時刻が12時31分15秒だったとする。
で、12時31分16秒になったらこのループは終了する。

15秒と16秒は数値としては1しか違わない。
でも、1秒より小さい時間も存在している。
0.1秒だったり、もっと小さい0.0001秒だったり。

プチコン3号は処理スピードがすごく速いから、このループで12時31分16秒000...になる瞬間を待ってるんだ。

じゃあ、カウントダウンプログラムに戻るよ。

ＦＯＲ ループでさっきの１秒を待つ処理を４回動かしているんだ。
で、１秒たつごとに３，２，１と表示する。

```
 1 CLS                              画面クリア
 2 'カウントダウン
 3 T1$=TIME$                        Ｔ１＄に現在時刻を入れる
 4 T2$=T1$                          Ｔ２＄にＴ１＄を入れる
 5 FOR I=4 TO 1 STEP -1             4 から 1 までの 4 回ループ
 6  CLS                              画面クリア
 7  IF I!=4 THEN PRINT I             4 以外なら画面に Ｉ を表示
 8  WHILE T1$==T2$                   Ｔ１＄とＴ２＄が同じなら繰り返す
 9   T2$=TIME$                        Ｔ２＄に現在時刻を入れる
10  WEND                             ループ(WHILE)
11  T1$=T2$                          Ｔ１＄にＴ２＄を入れる。
12 NEXT                             ループ(FOR)
```

カウントダウンは３から始めたいから、４の時は画面に表示しないようにしているよ。
ＦＯＲ ループを３から始めてしまうと３の時に表示時間が１秒より短くなってしまうことがある。
3,2,1 と１秒ずつカウントダウンしたいからこんなふうになっているんだね。

このカウントダウンは急にゲームが始まらないようにするためのものなんだけど、もう一つ役割があるんだ。

もう一つの役割はプレーヤーの条件を同じにするためなんだ。

ＴＩＭＥ＄命令だと１秒単位でしか時間を得ることができないから、１秒より小さい時間で差がついてしまう恐れがある。
0.00 秒からスタートする人と、0.85 秒からスタートする人がいたら不公平だよね？

このカウントダウンで全てのプレーヤーが、秒が変わった瞬間にゲームを開始できるようになるんだ。

こんどは後半、10秒間で何問答えられるか競うゲーム部分だね。

```
14  'ゲームカイシ
15  CLS
16  TOKUTEN=0
17  'ジカンヲビョウニ ヘンカンシテ 10ヲ カサン
18  T1S=VAL(MID$(T1$,0,2))*3600+VAL(MID$
    (T1$,3,2))*60+VAL(MID$(T1$,6,2))+10
19  WHILE T1S>T2S
20   A=RND(9)+1
21   B=RND(9)+1
22   C=A+B
23   CLS
24   MON$=STR$(A)+"+"+STR$(B)+"="
25   INPUT MON$;KOTAE
26   IF C==KOTAE THEN
27    TOKUTEN=TOKUTEN+1
28   ENDIF
29   T2$=TIME$
30   T2S=VAL(MID$(T2$,0,2))*3600+VAL(MID$
    (T2$,3,2))*60+VAL(MID$(T2$,6,2))
31  WEND
32  CLS
33  PRINT "アナタノトクテンハ ";TOKUTEN;" テンデス"
```

処理のイメージがつかみやすいように、まず全体の流れを説明しておくことにする。

ＷＨＩＬＥループで10秒を越えたらゲームが終了するようにしている。
Ｔ１Ｓはゲーム開始から10秒後の時刻。
Ｔ２Ｓは現在時刻。
Ｔ２ＳがＴ１Ｓより小さい間だけループすれば10秒制限ができる。

ただし、10秒たったらすぐに終了するわけではないよ。
25行にＩＮＰＵＴ命令があるのでなにも入力しなければ10秒を超えてもゲームは終了しない。
10秒間でゲームが終わるのではなくて、10秒間だけ問題が出題されるってことだね。

10秒を計るにはどうするか？
今の時刻より10秒あとの時刻になったときが10秒間

時分秒のままだと10秒後をもとめるのが難しいよね。
だから、ここでは時間を秒に統一しているんだ。
それが18行。

```
18  T1S=VAL(MID$(T1$,0,2))*3600+VAL(MID$
    (T1$,3,2))*60+VAL(MID$(T1$,6,2))+10
```

18行は2行に表示されてるけどプログラムとしては1行なんだ。
マークまでが1行だからね。

12時31分15秒を秒に統一するには．．．
（12時Ｘ3600）＋（31分Ｘ60）＋15秒
でどうだろか？
で、18行では10秒後の時間を作るために最後に10秒を足している。

細かく見て行こう。
色分けするよ。

（12時X 3600）＋（31分X 60）＋15秒＋10秒後の時刻にする

```
 18  T1S=VAL(MID$(T1$,0,2))*3600+VAL(MID$
     (T1$,3,2))*60+VAL(MID$(T1$,6,2))+10
```

まず時のところを理解しよう。
上のプログラムの赤い部分だよ。

これは二つの命令の組み合わせだ。
分かりやすいように色分けしたよ。

```
        VAL(MID$(T1$,0,2))*3600
```

MID$とVAL だ。
MID$はミッドって呼ぶらしい。
VAL はバル、VALUE（バリュー）の略かな？
この辺命令の呼び方はよくわからない．．．

まずMID$から

```
        MID$(T1$,0,2)
```

MID$は文字列から必要な部分だけを切り出す命令なんだ。

T1$は文字列変数、切り出す元となる文字列。
そのあとの数値は切り出す開始位置と切り出す文字数なんだ。

T1$にはゲームを開始した時刻が入っているんだ。
（カウントダウン処理の 11 行でセットされてるから確認してね。）

文字列から必要な部分を切り出すんだけど、ここで必要な部分っていうのは時のこと。

T1$＝12:31:15
　　時:分:秒

目的の時は先頭の 2 文字。
つまりT1$から先頭の 2 文字を切り出したいんだ。

```
        MID$(T1$,0,2)
```

↑
切り出す開始位置
先頭は 0 からなんだ。

```
        MID$(T1$,0,2)
```

↑
切り出す文字数

これで先頭から 2 文字切り出すことができる。
切り出されるのは“12”だ。

で、次にＶＡＬ。
ＶＡＬ は文字を数値に変換してくれるんだ。

```
VAL("12")
```

↑
ＭＩＤ＄で切り出した”12”

さっき切り出したのは文字列だから。
時を秒に変換するためには 3600 を掛けなければならないけど、文字列だと計算できないよね。

そこで ＶＡＬ 命令を使って数値の 12 に型変換（かたへんかん）するんだ。

変数の型で考えると分かりやすいかも。

```
A=VAL(A$)
```

文字列型から数値型へ変換ってこと。

これで、時を 12 X 3600 で無事に秒に変換できた。

じゃあ分のところも見てみるよ。
まったく同じだけどね。

```
VAL(MID$(T1$,3,2))*60
```

↑
分のところが欲しいから切り出す開始位置が変わってる。

Ｔ１＄＝12:31:15
時:分:秒

分の位置は 4 文字目だけど、文字列の先頭は 0 から数えるから切り出す開始位置は 3 だね。
切り出したものを数値に変換して 60 を掛ける。
これで分の秒への変換は完了だ。

秒も全く同じなので説明は省くね。

秒に変換できたものに 10 を足して、ゲーム終了の時間を作って Ｔ１Ｓ に入れておく。
これが 18 行のすべて。

ここまでこればあとは簡単。
10秒経ったらゲームを終了するための判定処理を確認しよう。

```
14  'ゲームカイシ
15  CLS
16  TOKUTEN=0
17  'ジカンヲビョウニ ヘンカンシテ 10ヲ カサン
18  T1S=VAL(MID$(T1$,0,2))*3600+VAL(MID$
    (T1$,3,2))*60+VAL(MID$(T1$,6,2))+10
19  WHILE T1S>T2S
20   A=RND(9)+1
21   B=RND(9)+1
22   C=A+B
23   CLS
24   MON$=STR$(A)+"+"+STR$(B)+"="
25   INPUT MON$;KOTAE
26   IF C==KOTAE THEN
27    TOKUTEN=TOKUTEN+1
28   ENDIF
29   T2$=TIME$
30   T2S=VAL(MID$(T2$,0,2))*3600+VAL(MID$
    (T2$,3,2))*60+VAL(MID$(T2$,6,2))
31  WEND
32  CLS
33  PRINT "アナタノトクテンハ ";TOKUTEN;" テンデス"
```

重要なところだけ色を変えたよ。
まず、赤のところ。

これはさっき説明した内容とやっていることは全く同じだ。
現在時刻を秒に変換して T2S に入れてるだけなんだ。

で、青の WHILE ループ。
ゲーム終了時間の T1S と現在時刻の T2S を比較して、T2S が T1S 以上になったらタイムアップ。

タイムアップしたら最後に得点を表示して終了。

これで終わりなんだけど、あと一つだけ新しい命令があるので確認しよう。

```
24   MON$=STR$(A)+"+"+STR$(B)+"="
25   INPUT MON$;KOTAE
```

STR$だ。
VAL とは反対に、数値型を文字列型に変換するんだ。

24,25行は何をやっているかというと、問題を表示して入力（INPUT）するため。
24行で問題を表示するための準備をしているんだ。

変数の型で見るとこうだね。

```
A$=STR$(A)
```

STR$はSTRING（ストリング）の略だと思う。
読み方はわかんないなぁ、スター？，エスティーアール？

最後に RUN して 10 秒チャレンジだ！

あ、そうそう問題は一桁の計算にしておいたよ。
二桁だと 10 秒だと僕みたいな普通の人には難しすぎるからね。

あぁ、忘れてた．．．
「**[例６]** 足し算、10 秒タイムチャレンジ」には致命的なバグがあるんだった。
ゲームを始める時間によっては永久ループになってしまうというバグが。
永久ループってのは、ループから抜け出せなくなってしまってプログラムが終わらないってこと。

日付が変わるときにそのバグは動き出す．．．
23 時 59 分 55 秒にゲームを開始したとする。
その 10 秒後は 0 時 0 分 5 秒．．．

24 時 0 分 5 秒になれば問題ないけどそれは無理だ。
0 時を過ぎる場合に ＩＦ 文などで判断して動くようにすればいいんだけど今回は仕様とするよ。
仕様（しよう）っていうのは「バグではなくて、分かっていてそういうふうに作った」ってこと。

だって、良い子このキミたちはもう寝ている時間だからね。
お金の計算をしたり病院で使うプログラムだったらバグはゆるされないだろうけど、計算ゲームだからこのくらいはいいよね？

ずいぶん難しくなってきたけど、もう少しだけ続くよ。

10 秒チャレンジしたあとに得点だけじゃなくて答え合わせをしたいよね？

え、したくない？

```
Ａ$＝”ＮＯ”↵
ＷＨＩＬＥ　Ａ$！＝”ＹＥＳ”↵
　ＩＮＰＵＴ　“ｺﾀｴｱﾜｾ　ｼﾀｲﾖﾈ？　ＹＳＥ，ＮＯ”；Ａ$↵
ＷＥＮＤ
```

したいよねぇ、うん。
よしよし。
じゃあ、さきに進めようか。

7．足し算、最後に答え合わせをしよう！
ちょっと難しいから飛ばしてもいいよ。
プログラムをいくつか作ってからでも遅くはないよ。

[例7] 足し算、最後に答え合わせをしよう！

```
 1  CLS
 2  DIM MA[100]
 3  DIM MB[100]
 4  DIM MKOTAE[100]
 5  SUU=0
 6  'カウントダウン
 7  T1$=TIME$
 8  T2$=T1$
 9  FOR I=4 TO 1 STEP -1
10   CLS
11   IF I!=4 THEN PRINT I
12   WHILE T1$==T2$
13    T2$=TIME$
14   WEND
15   T1$=T2$
16  NEXT
17
18  'ゲームカイシ
19  CLS
20  TOKUTEN=0
21  'ジカンヲビョウニ ヘンカンシテ 10ヲ カサン
22  T1S=VAL(MID$(T1$,0,2))*3600+VAL(MID$(T
    1$,3,2))*60+VAL(MID$(T1$,6,2))+10
23  WHILE T1S>T2S
24   A=RND(9)+1
25   B=RND(9)+1
26   C=A+B
27   CLS
28   MON$=STR$(A)+"+"+STR$(B)+"="
29   INPUT MON$;KOTAE
30   IF C==KOTAE THEN
31    TOKUTEN=TOKUTEN+1
32   ENDIF
33   'ハイレツニ ホゾン
34   MA[SUU]=A
35   MB[SUU]=B
36   MKOTAE[SUU]=KOTAE
37   INC SUU
38   '
39   T2$=TIME$
40   T2S=VAL(MID$(T2$,0,2))*3600+VAL(MID
    $(T2$,3,2))*60+VAL(MID$(T2$,6,2))
41  WEND
42  CLS
43  FOR I=0 TO SUU-1
44   IF MA[I]+MB[I]==MKOTAE[I] THEN
45    OX$="○"
46   ELSE
47    OX$="✖"
48   ENDIF
```

```
 49   MON$=STR$(MA[I])+"+"+STR$(MB[I])+"="+
    STR$(MKOTAE[I])
 50   PRINT OX$,MON$
 51 NEXT
 52 PRINT "アナタノトクテンハ ";TOKUTEN;" テンデス"
```

[例６]に追加した部分を赤くしておいたよ。

問題と答えを記憶しておく場所を作ったのと、最後に問題と答えの一覧を表示する処理を追加しただけだよ。

ゲーム終了でこんなふうに表示される。

問題と答えを記憶しておくために配列（はいれつ）を使ったんだけど．．．
このプログラムを追っていくと難しいからもう少し簡単なサンプルで配列の使い方を見て行くよ。

その前にもう一つ新しい命令が出てきているけど、すごく簡単なので先に説明しておくよ。

```
 37   INC SUU
```

ＩＮＣ は INCREMENT（インクリメント）の略で呼び方はインクかな？
1 を加算するだけ。

同じ処理を違う書き方にするとこうなる。

```
      SUU=SUU+1
```

後ろに数値を追加すると

```
      INC SUU,3
```

こうなる。

```
      SUU=SUU+3
```

加算が簡単に書けるというだけ。

ちなみにマイナスもあるよ。

```
      DEC SUU,3
```

ＤＥＣ は DECREMENT（デクリメント）の略で呼び方はデクかな？
減算になっているだけで、使い方は ＩＮＣ と同じ。

じゃあ配列の説明にとりかかろう。
配列を使ったプログラムの例だよ。

```
 1 DIM GAME$[3]
 2 DIM SCORE[3]
 3 GAME$[0]="ゼビウス"
 4 GAME$[1]="パックマン"
 5 GAME$[2]="メトロクロス"
 6 SCORE[0]=23800
 7 SCORE[1]=8540
 8 SCORE[2]=765
 9 FOR I=0 TO 2
10  PRINT GAME$[I],SCORE[I]
11 NEXT
```

RUN するとこうなる。

プログラムは 3 色に色分けしておいた。
・赤は配列の定義
・青は配列にデータのセット
・緑は配列の内容を画面に表示
となっている。

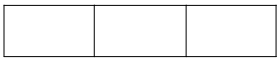
まず赤、配列の定義。

```
 1 DIM GAME$[3]
```

変数は定義をしなくても使えたけど、配列の場合は使う前に定義をしなければならないんだ。
これは GAME$ っていう配列を定義している。
その後ろの[3]が配列の数。
配列の数を要素数（ようそすう）と言う。

変数と配列のちがいは？
・変数はデータを入れる箱が 1 つだけある。
・配列はデータを入れる箱がいくつも並んでいる。
（[3]って書いてあったら箱が 3 つってこと。）

変数 A 

配列 A[3] 

配列も変数の仲間なんだけど、分かりやすいように変数と配列って呼んで区別するようにしているよ。

次は青の配列へのデータセット（代入）だ。

```
 3  GAME$[0]="ゼビウス"
 4  GAME$[1]="パックマン"
 5  GAME$[2]="メトロクロス"
```

配列を定義するときの[3]は要素数って言うんだけど、配列を使う時の[0]は添え字（そえじ）と呼び方が変わるんだ。
それと、添え字は 0 から始まるんだ。
要素数が 3 の場合、添え字は 0,1,2 となる。

この例では三つの箱に順番にデータを代入しているんだ。
変数に添え字が増えただけなので難しくはないよね？

最後は緑の配列の内容を画面に表示する処理だ。

```
 9  FOR I=0 TO 2
10   PRINT GAME$[I],SCORE[I]
11  NEXT
```

GAME$[I]は添え字に I を使っているんだ。
こうするとＦＯＲ文と組み合わせることができるから便利だよね。

でもどんな時に配列を使うといいんだろう？

まず、配列がなかった場合を考えてみようか。

```
 1  GAME1$="ゼビウス"
 2  GAME2$="パックマン"
 3  GAME3$="メトロクロス"
 4  SCORE1=23800
 5  SCORE2=8540
 6  SCORE3=765
 7  PRINT GAME1$,SCORE1
 8  PRINT GAME2$,SCORE2
 9  PRINT GAME3$,SCORE3
```

前半はさっきの例と同じだけど、画面に表示する部分でＦＯＲループが使えなくなってる。
まあ、それでもこれはデータが 3 個だからとくに問題は感じないよね。

でも、これが 10 個．．．いや 100 個だったら？

まずは、「データがたくさんあるとき」に配列が使えないかな？って考えてみよう。
やっぱり少しでも楽にプログラムを作りたいよね。

[例７]は細かく解説はしないけど、ここまでこれたキミならじっくり見れば分かるはずだ。
赤色の部分をよく見てね。
配列に問題と答えをしまっておいて、あとで ＦＯＲ ループで表示しているだけだよ。
あと 33～37 行に注目してね。
配列にデータを代入するときも添え字に変数を使っているんだ。
こうすると、たくさんのデータを扱うのがとても簡単になるよね。

1～3 行の配列の定義で、要素数を 100 にしているけどこんなには必要ないかな？
10 秒間で 100 問以上答えられる人はいないから 100 にしているけど 50 もあれば十分だろう。

```
  1 CLS
  2 DIM MA[100]
  3 DIM MB[100]
  4 DIM MKOTAE[100]
  5 SUU=0
  :    :（カウントダウンは省略）
 18 'ゲームカイシ
 19 CLS
 20 TOKUTEN=0
 21 'ジカンヨビョウニ ヘンカンシテ 10ヨ カサン
 22 T1S=VAL(MID$(T1$,0,2))*3600+VAL(MID$(T
    1$,3,2))*60+VAL(MID$(T1$,6,2))+10
 23 WHILE T1S>T2S
 24  A=RND(9)+1
 25  B=RND(9)+1
 26  C=A+B
 27  CLS
 28  MON$=STR$(A)+"+"+STR$(B)+"="
 29  INPUT MON$;KOTAE
 30  IF C==KOTAE THEN
 31   TOKUTEN=TOKUTEN+1
 32  ENDIF
 33  'ハイレツニ ホゾン
 34  MA[SUU]=A
 35  MB[SUU]=B
 36  MKOTAE[SUU]=KOTAE
 37  INC SUU
 38  '
 39  T2$=TIME$
 40  T2S=VAL(MID$(T2$,0,2))*3600+VAL(MID
    $(T2$,3,2))*60+VAL(MID$(T2$,6,2))
 41 WEND
 42 CLS
 43 FOR I=0 TO SUU-1
 44  IF MA[I]+MB[I]==MKOTAE[I] THEN
 45   OX$="○"
 46  ELSE
 47   OX$="✖"
 48  ENDIF
 49  MON$=STR$(MA[I])+"+"+STR$(MB[I])+"="+
    STR$(MKOTAE[I])
 50  PRINT OX$,MON$
 51 NEXT
 52 PRINT "アナタノトクテンハ ";TOKUTEN;" テンデス"
```

最後までよく頑張ったね。
これでBASICの基本編は終わりだよ。

８．できることから始めよう！
なんか途中から難しくなったなぁと感じたら、できることから始めればいいよ。

例５でＦＯＲ文を習ったよね？
［例５］ 足し算、10問チャレンジ

```
 1 CLS
 2 TOKUTEN=0
 3 FOR I=1 TO 10
 4  A=(RND(9)+1)*10+RND(10)
 5  B=(RND(9)+1)*10+RND(10)
 6  C=A+B
 7  PRINT A;"+";B;"="
 8  INPUT "コタエヲニュウリョク";KOTAE
 9  IF C==KOTAE THEN
10   PRINT "セイカイ"
11   TOKUTEN=TOKUTEN+1
12  ELSE
13   PRINT "マチガイ セイカイハ ";C
14  ENDIF
15 NEXT
16 PRINT "アナタノトクテンハ ";TOKUTEN
```

でもＦＯＲ文なんてなくても同じことはできるんだ。
次のプログラムはＩＦ文とＧＯＴＯ文を組み合わせて例５と同じことをしているんだ。

［例５’］ 足し算、10問チャレンジ

```
 1 CLS
 2 TOKUTEN=0
 3 I=1
 4 @LOOP
 5  A=(RND(9)+1)*10+RND(10)
 6  B=(RND(9)+1)*10+RND(10)
 7  C=A+B
 8  PRINT A;"+";B;"="
 9  INPUT "コタエヲニュウリョク";KOTAE
10  IF C==KOTAE THEN
11   PRINT "セイカイ"
12   TOKUTEN=TOKUTEN+1
13  ELSE
14   PRINT "マチガイ セイカイハ ";C
15  ENDIF
16  I=I+1
17 IF I<=10 THEN GOTO @LOOP
18 PRINT "アナタノトクテンハ ";TOKUTEN
```

ＷＨＩＬＥループだってＩＦ文とＧＯＴＯ文があれば同じことができちゃうんだ。
プログラムの作り方は一つじゃない。
難しいなと思っても、キミにもできる方法があるはずだ。
あきらめずに一歩ずつ前に進もう。

「僕が考えた最強のこんにちはプチコン３号」の第一章「BASIC 入門」にあたる部分はこれで終了です。
この先を作る予定はありませんが、こんなふうになったらいいなと思っています。

第二章「もっと派手なゲームが作りたい...プチコン３号のことをもっと知ろう！」

・ハードウェアについて
　コンソール，グラフィック，スプライトなどの基本的なこと
・グラフィックの使い方
　線や丸などを書いてみる
　プリセットの画像をコピー
・スプライトの使い方
　キャラクターの表示（DEF，SPO...）
　キャラクターの移動、アニメーション
　　FOR ループを使い座標指定して移動
　　もっと簡単に SPANIM
　　飛び出す？（拡大縮小，回転，3D）
　　ボタン入力との組み合わせ
・BG の使い方
　パターンを並べる
　スクロール，回転
・効果音，音楽
　プリセットを鳴らす
　おまけで TALK

この章ではそれぞれの機能を単独で動かすことを基本とする。
サンプルはシンプルに！

第三章「ゲーム作りの基本的な形」

・インベーダーのようなゲーム
　敵の移動、弾を SPANIM で
　当たり判定
　キャラクター，音楽などはプリセットを利用
・メインループ
・インプット
・画面出力
・音楽
・スクロールするアクションゲームのサンプル

巻末

命令表（公式マニュアル程度）
文字コード表
キャラクターコード表

最後に．．．

これは自分の子供に BASIC を教えるために「あったらいいな」を形にしたものです。
私の個人的な見解や認識の違いなどが含まれてる可能性があります。
これを読んだがために「アマグラマー」どころか私のような「ゆるグラマー」に育ってしまう恐れもあります。

BASIC の入門者の助けになるように作ったつもりですがご利用は自己責任でお願いいたします。

2015.05.21
ゆうぱぱ