ホームベーシック SC-3000 SK-1100 共用

SC-3000シリーズ SK-1100共用

ホームベーシック

テキスト

もくじ

はじめに

セガのホームベーシックをお買い上げいただきましてありがとうございます。このホームベーシックには、ベーシックと、プログラムを作らなくても楽しめる3つのプログラムが用意されています。

ベーシックでプログラムを作るのも良し、基本的な「ゲーム」で遊ぶのも良し。「パターンのへんこう」でキャラクタをつくり、「ゲーム」に登場させるも良し、「計算ボード」で計算するも良し …… 。使い方はあなた次第、自由にこのホームベーシックを使いましょう。

エラーが出ても、おそれずにキーボードに触れてください。きっとパソコンはあなたの友だちになってくれることでしょう。ホームベーシックは、パソコンを始める人が、楽しみながら覚えられるようにと作られたものなのですから。

このホームベーシックは ―

◉ **ひらがなが使えます。**

SC－3000 ベーシックでは「ひらがな」は使えませんでした。でも、ホームベーシックなら、カタカナだけでなく、ひらがなも使えます。

◉ **文字が大きくなりました。**

SC－3000 ベーシックより文字が大きくなり、読みやすくなりました。

◉ **グラフィックスが強化されました。**

描いたものをテープに保存できます。ですから、気に入った絵をいつでも取り出して使うことができます。

◉ **音楽の演奏が簡単になりました。**

ドレミと音階を書くだけで音楽が奏でられます。

◉ **整数型になりました。**

整数型ベーシックとは、小数点を持っていないベーシックのことです。整数型にしたことで、ベーシックの処理がはやくなり、ゲームなど作るのに有利になりました。

◉ **グラフィック記号が少なくなりました。**

SC－3000 や SK－1000 のキーボードに書かれた、いろいろなグラフィック記号のうち、不用なものは省きました。(キーの使い方をご覧ください。)

◉ **FUNC キーの働きがありません。**

SC－3000 や SK－1000 のキーボードに書かれたベーシックの命令文はホームベーシックでは使えません。

これらは SC－ 3000 ベーシックで使います。

2 つのベーシックの使用目的 ―

SC－3000 ベーシック	**本格的プログラミング** **精度の高い計算用**
ホームベーシック	**計算精度は必要としないが、グラフィックスや音楽を簡単に使いたい** **ベーシックの基本を覚えたい**

目的にあわせてお使いください。

第１章　コンピュータの使い方

接続のしかた

SK－1100，SG－1000 をお持ちの方

1. SG－1000 を用意して下さい。
   （電源はまだ入れないで下さい。）
2. SK－1100 本体より出ているフラットケーブルを、SG－1000 の差し込み口にあるエッジコネクターにしっかりと差し込んでください。
   コネクターを差し込むとき、表と裏をまちがえないよう注意してください。
3. 接続が終わったら、HOME BASIC カートリッジを差し込んで、SG－1000 の電源を入れてください。

> SK－1100 と SG－1000
> SG－1000 にはキーボードがありませんので、パソコンとして使えませんでした。
> SK－1100 キーボードと専用 BASIC と共に用いることで、高性能パソコンとして使っていただけるようになりました。

SC－3000 をお持ちの方

1. スイッチボックスを、テレビに接続してください。
   （ビデオ入力のあるテレビの場合は、専用ケーブルで、本体とテレビの入力端子と音声端子に直接つないでください。）

2. スイッチボックスをCOMPUTER（コンピュータ）側にして、テレビのチャンネルを1または2チャンネルのうち空いている方にしてください。
3. 本体のチャンネル切替スイッチを、1または2チャンネルに合わせてください。
4. ホームベーシックカートリッジを正しく差し込んでください。
5. 接続が終わったら、ケーブルの接続を確かめてから、テレビの電源を入れてください。
6. コンピュータの電源も入れてください。

キーボードの使い方

カーソル

画面左上に ▦ が点滅しています。これは、カーソルといって、キーから入力した文字や記号が表示される位置を示すものです。

カーソルには、▦ ■ ＋ ＊ の4種類があり、

▦ ····· 英数モード
■ ····· カナモード
＋ ····· ひらがなモード
＊ ····· グラフモード

であることを示します。

ふつう、カーソルは ▦ の状態です。英力 数ナ キー、或いは GRAPH キーを押すことによって、カーソルを変え、キーから入力するものを、カナ、ひらがな、あるいは記号というように、選択することができます。

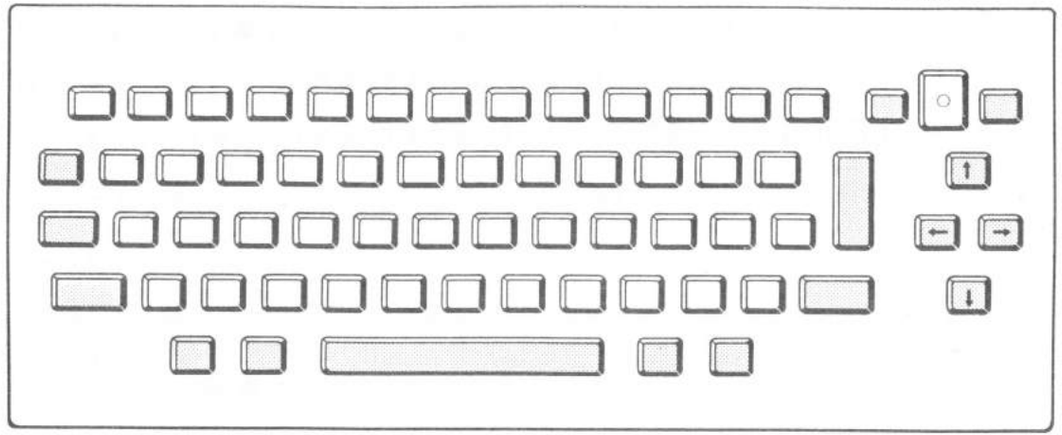
英力 数ナ キー（キーボード左下にあります。）を押してください。カーソルは ■ に変わります。同じキーをもう一度押してください。＋ に変わります。さらにもう一度押すと、もとのカーソル ▦ にもどります。 GRAPH キー（ 英力 数ナ キーのとなりにあります。）を押すと、＊ に変わります。

何か入力するときは、必ず適切なモードを選んでください。

キーのこと

キーボードをながめて下さい。いろいろなキーが並んでいます。

＜キーボードの図＞



これらのキーは文字や記号の書き込まれたキー（たとえば [／ 田 / 1 ヌ] と、それ以外の特殊なキー（ [CR] など）に大別されます。

★★★ 文字と記号のキーのこと ★★★

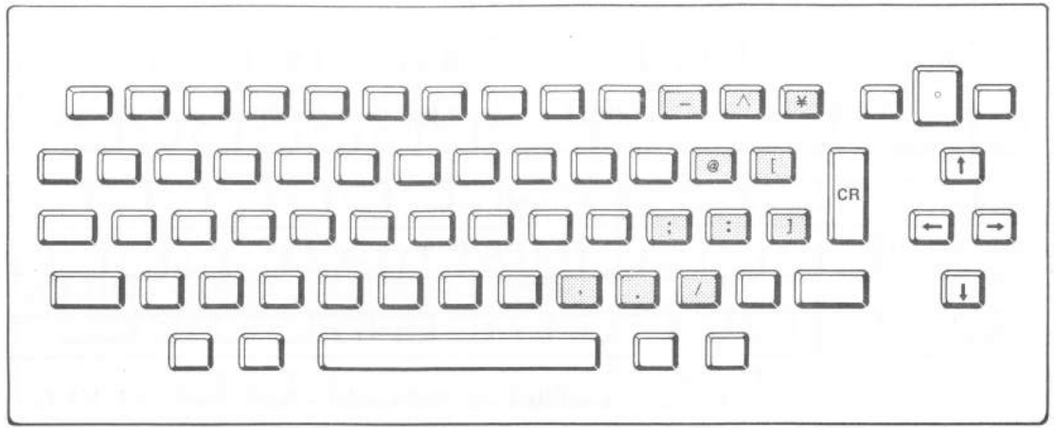
キーボードには、英文字、数字、カナ文字、記号が３～４種類書き込まれたキーがあります。それらを使うと、文字や記号を画面に表示することができます。

まず、[英カ / 数ナ] キー、或いは [GRAPH] キーを使って、適切なモードを選びます。そうして、文字や記号のキーを単独で押す、或いは [SHIFT] キーという特殊キーと共に用いると、お望みの文字や記号が画面に表示されます。

画面に表示したいもの	モード(カーソル)	キーの押しかた
英文字 (大文字) 数字	▦	英文字や数字のキーを打ちます。
英文字 (小文字)	▦	[SHIFT] キーを 押しながら、英文字のキーを打ちます。
ひらがな	＋	カナ文字のキーを打ちます。
ひらがな ・小さい文字(あいうえおやゆよつ) ・を	＋	[SHIFT] キーを押しながら、カナ文字のキーを打ちます。
カタカナ	■	カナ文字のキーを打ちます。
カタカナ ・小さい文字(ァイゥェォャュョッ) ・ヲ	■	[SHIFT] キーを押しながら、カナ文字のキーを打ちます。
グラフィック記号	＊	グラフィック記号のキーを打ちます。
グラフィック記号	＊	[SHIFT] キーを押しながら、グラフィックキーを打ちます。

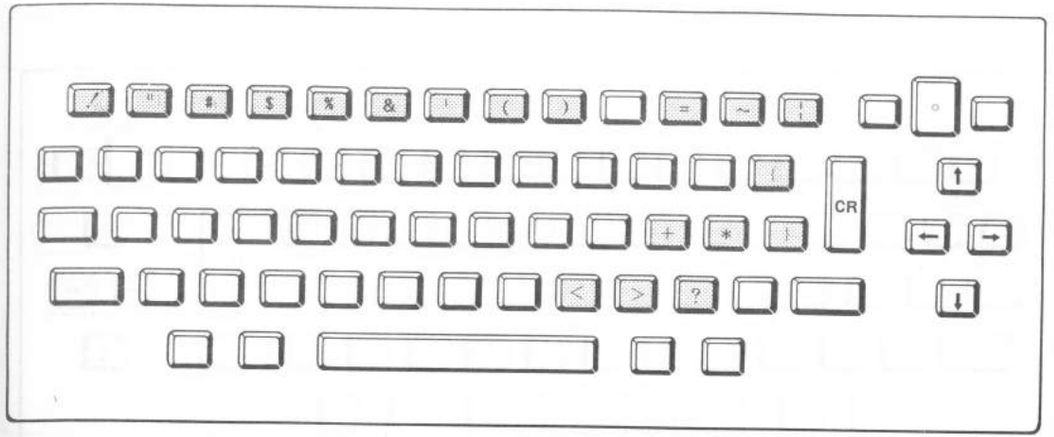
英数モード ▦

図に記した記号、及び英大文字、数字



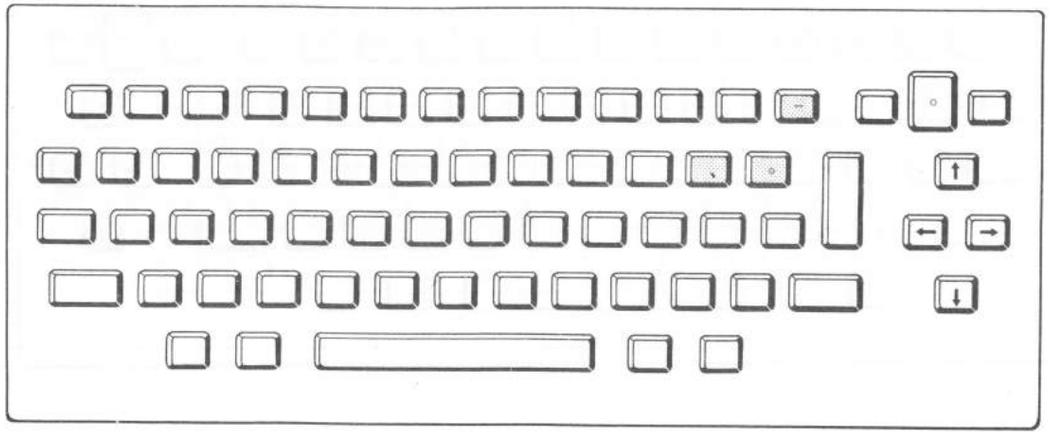
英数モード ▦ で SHIFT キーを押す

図に記した記号、及び英小文字



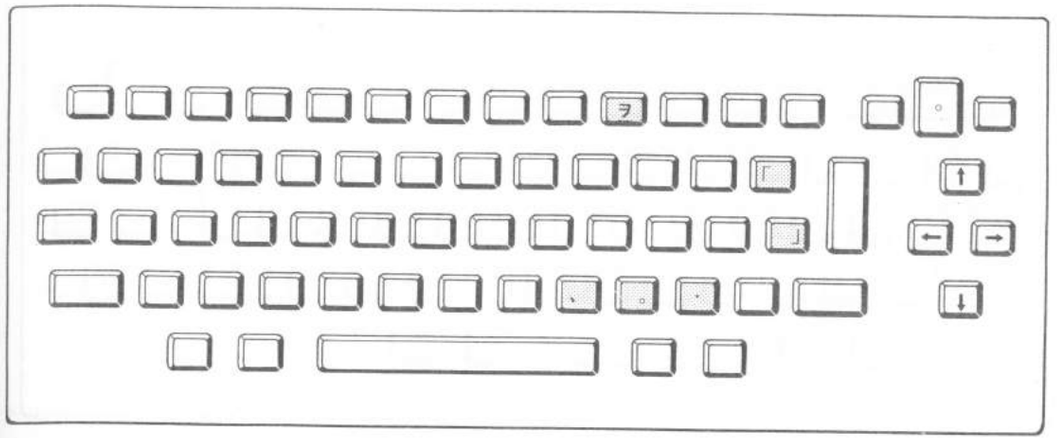
カナモード ■　　ひらがなモード ＋

ひらがな、或いはカタカナの濁点、半濁点、長音符号及びカタカナ、ひらがな。



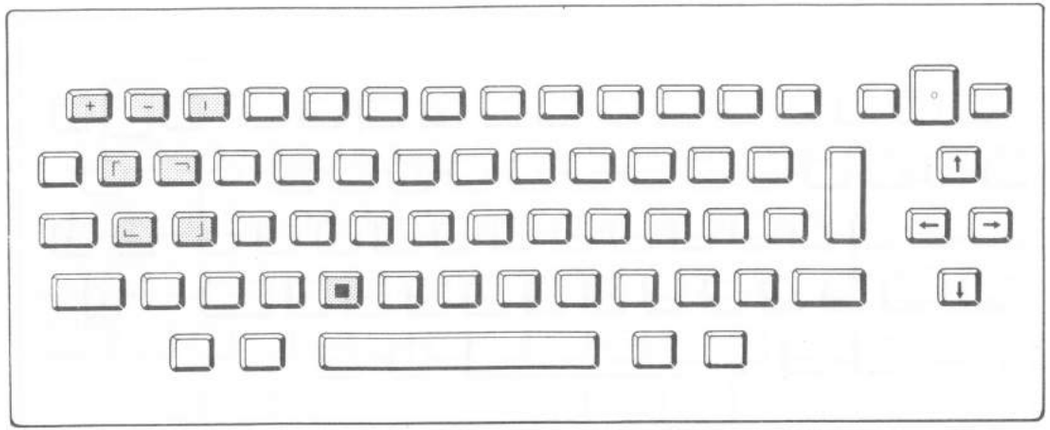
カナモード ■ で SHIFT キーを押す

図に記した記号、及びカタカナの小さい文字



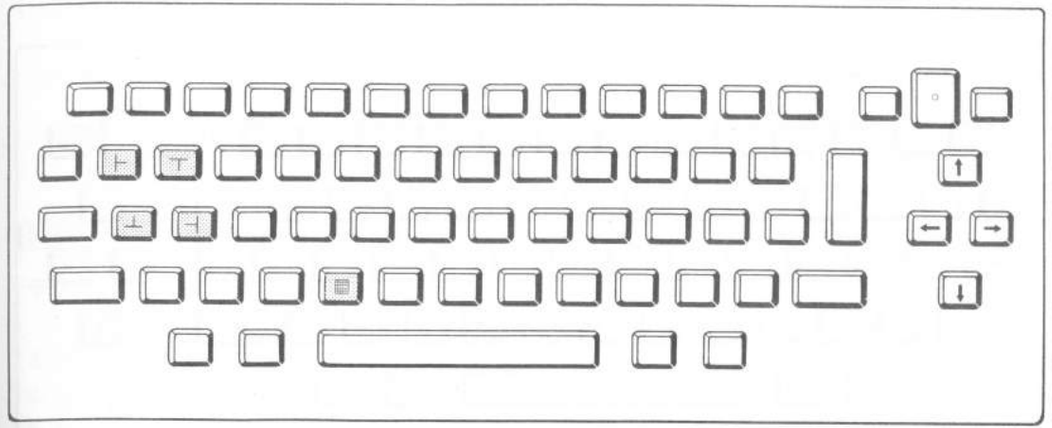
グラフモード ＊

図に記した記号



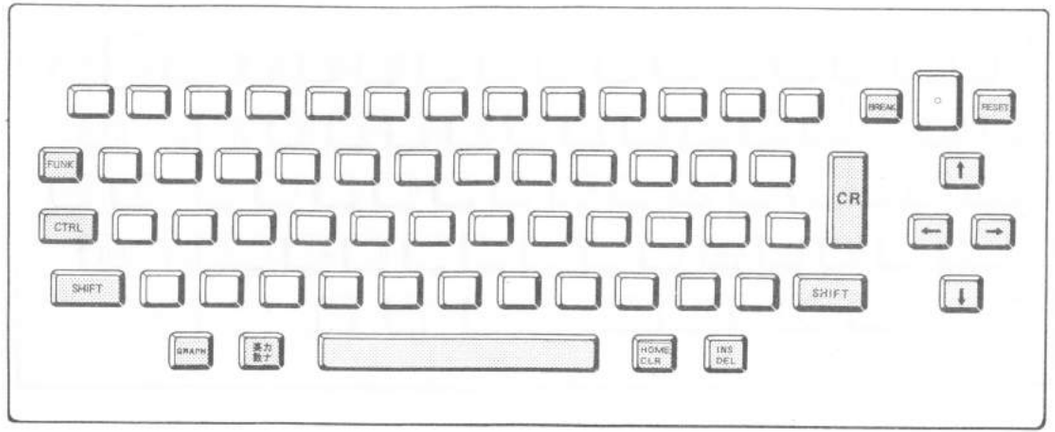
グラフモード ＊ で SHIFT キーを押す

図に記した記号



★★★ 特殊キーのこと ★★★

キーボードには、 SHIFT CTRL のような特殊な働きをするキーもあります。



≪文字を消す・文字の訂正≫

HOME／CLR （ホーム／クリア）

このキーを押すと、画面上の文字が消えて、カーソルは左上のホームポジションに戻ります。画面上の文字、記号を消したいときに使ってください。また、キーを打ち始める前に使ってください。

note ! SHIFT キーを押しながら、このキーを押すと、文字は消えずにカーソルだけがホームポジションに戻ります。

INS／DEL （インサート／デリート）

デリート

このキーは、文字を一文字ずつ消したり追加する場合に使います。たとえば、ＡＢＣＤと入力する所を、ＡＢＣＥと打ってしまった場合、 INS／DEL キーを押すと、カーソルが一文字分、前にもどってＥの文字を消します。そこにＤの文字を入力しますと、ＡＢＣＤというふうに、訂正されます。

Ａ Ｂ Ｃ Ｅ ▦ ← Ｄのかわりに、間違ってＥと打ってしまいました。
Ａ Ｂ Ｃ ▦ ← INS／DEL キーを押しますと、カーソルが左にひとつ移動します。
Ａ Ｂ Ｃ Ｄ ▦ ← Ｄの文字を打ちます。訂正されました。

インサート

このキーと SHIFT キーを合わせて用いると、カーソルの点滅がはやくなり、インサートモードという状態になります。たとえば、ＡＢＣＤのＢとＣの間にＴという文字を入れたい場合、次のようにします。

ＡＢＣＤ▦

ＡＢ▦Ｄ　← カーソルをＣの上に移動させ、SHIFT キーと INS／DEL キーを押します。(インサートモードになりました。)

ＡＢＴ▦Ｄ　← Ｔの文字を入力します。

ＡＢＴＣＤ▦　← 入力した文字が入り、ＣＤが右側にずれました。

インサートモードから抜け出すには

① CR キーを押す。

② カーソルキーを押す。

③ SHIFT ＋ INS／DEL キーを押す。

のいずれかの方法をとってください。

note ! INS (インサート) は文字を追加する意味です。
DEL (デリート) は、文字を消す意味です。

≪入力したものを記憶させる≫

CR (キャリッジ・リターン又はリターン)　キーで画面に命令文を入力しても、まだ命令を実

行しませんし、メモリーに記憶もされません。命令文を実行させたり、メモリーに記憶させるためには、CR キーを押します。プログラムを修正した場合も、CR キーを押します。

≪その他の特殊キー≫

グラフ／BREAK (画面切替／ブレーク)

このキーは二つの働きがあります。グラフ (画面の切り替え) と BREAK (ブレイク) です。

グラフ …… 画面を切替えるときに使います。コンピュータは、テキスト画面 (プログラムを書く画面) と、グラフィック画面 (グラフィックスを表示する画面) を持っています。その画面の切替えに使います。
詳しくはグラフィックスの SCREEN のところを見てください。

BREAK …… プログラムの実行中、プログラムを停止させたい時に使います。

RESET

電源を入れた時の画面 (オープニング画面) と同じ状態に戻すキーです。プログラムの実行中、画面に異常が出た場合、このキーを押すと 1 ～ 2 秒後に、オープニングの画面にもどります。

SPACE (スペースキー)

文字や記号のあいだをあけます。

A␣B␣C

スペースキーを 1 回押すと、一文字分の空間ができます。

↑
← → （カーソルキー）
↓

カーソルを上下左右に移動させます。

GRAPH （グラフィックキー）　　英カ数ナ （英数・カナキー）

どちらも、モードを選択するのに使います。これらのキーを押すと、カーソルの形が変わります。

SHIFT （シフトキー）

他のキーとあわせて使うことで、さまざまな機能を発揮します。キーの押し方の表を見てください。

CTRL （コントロール・キー）

このキーは、他の文字キーとあわせて使います。たとえば、CTRL キーを押したまま H の文字キー を打ちますと、文字を消す働きをします。
付録の「コントロールコード」をみてください。

第２章　メニュー紹介

パソコン本体の接続は正しくできましたか。
電源を入れると、宇宙船の操縦席があらわれて、そこにメニューがうつります。

1. シュートゲーム
2. けいさんボード
3. BASIC
4. パターンのへんこう

このホームベーシックには、BASIC のほかに、3つのプログラムが組み込まれています。
あなたはどれを選びますか？
好きなプログラムの番号キーを押しましょう。

1. シュートゲーム

1のキーを押すと、シュートゲームの画面になります。

<プレイ方法>

ジョイスティックでもキーボードでもプレイできます。(ただし、キーボードでは、片方のプレイヤシップしか動きません。)

	ジョイスティック	キーボード
スタートのしかた	プッシュスイッチを2つ同時に押す。	HOME/CLR キーと INS/DEL キーを同時に押す。
プレイヤシップの操縦	ジョイスティックのレバー	カーソルキー
ミサイルの発射	プッシュスイッチ	HOME/CLR キーもしくは INS/DEL キー

得点は、すべて10点です。
このゲームをもとにして、いろいろと変えることができます。

<へんこうできるもの>

① キャラクタ(プレイヤシップや敵)の形　メニューの4(パターンのへんこう)で説明します。
② キャラクタの色
③ キャラクタの動き　} 「シュートゲームの内容変更」で説明します。
④ ゲームの背景　BASICのグラフィックスで描いた絵は、自動的に、シュートゲームの背景になります。
(ただし、電源を切ると消えてしまいます。)

シュートゲームの内容変更

BASICからシュートゲームの内容をPOKE文を使って変えることができます。

≪プレイヤシップのキャラクタ変更≫

プレイヤシップのキャラクタは、1Pが00～7まで、2Pが8～15まで、計16個用意されています。電源を入れたとき、プレイ中はシップの進む方向によってキャラクタ番号が切り替わっています。切り替えを中止させることもできます。

キャラクタ番号と方向の関係

キャラクタ番号		進む方向
1P	2P	
0	8	↖ ↑ ↗
1	9	↙ ↓ ↘
2	10	
3	11	
4	12	←
5	13	
6	14	→
7	15	

メニューからBASICに入ります。
直接命令かリストで式を実行します。

◎キャラクタ切り替えしない。（　）は10進数

POKE&H8307,&H00　(0)

◎キャラクタ切り替えする。

POKE&H8307,&H01　(1)

≪プレイヤ数の変更≫

		10進数
二人用にする	POKE&H8308,&H00	(0)
一人用にする	POKE&H8308,&H01	(1)

≪連射する弾数の変更≫

一発	POKE&H8309,&H01	(1)
二発	POKE&H8309,&H02	(2)
三発	POKE&H8309,&H03	(3)

≪弾の発射方向の変更≫

発射方向　上方のみ	POKE&H830F,&H00	(0)
進行方向	POKE&H830F,&H01	(1)

≪プレイヤシップの持数≫

持数の設定	POKE&H8313,&H01～&H07 (1～7)

7機以上設定しても7機しか表示されません。

≪フルーツ数の変更≫

		10進数
一画面のフルーツ数	POKE&H830E，&H01～&H08	(1～8)

≪フルーツの場所≫

左側のフルーツを1番として

1番のフルーツ	X座標	&H8320，&H00～&HFF	(0～255)
	Y座標	&H8321，&H00～&HBF	(0～191)
		∫ (アドレスの変化幅は2です。)	
8番のフルーツ	X座標	&H832E，&H00～&HFF	(0～255)
	Y座標	&H832F，&H00～&HBF	(0～191)

≪フルーツの色を変える≫

最初の画面を1面として

1面のフルーツ色番号	&H8330，&H01～&H0F	(0～15)
使用するパターン番号	&H8331，&H20～&H27	(32～39)
	∫ (アドレスの変化幅は2です。)	
8面のフルーツ色番号	&H833E，&H01～&H0F	(0～15)
使用するパターン番号	&H833F，&H20～&H27	(32～39)

パターン番号は10進数で00から59まで使えます。

≪単独で飛ぶ敵の変更≫

◎ 敵の種類を変える

二つのアドレスの値を合計して、それに24を加えた数のパターンから後ろのパターンがよびだされます。

アドレス 1	&H830A，1	(10進数)
アドレス 2	&H830B，3	(10進数)

この場合は、1＋3＋24＝28で、28番から後ろのパターンが現われます。

◎ 敵の種類の数を変える

&H830C，1～15　(10進数)

◎ 敵の色を変える

&H8106，0～15　(10進数)

単独の敵のアドレスは、&H8106から20Hごとです。アドレスの後ろは色番号です。

(10進数)

≪その他のキャラクタの色をかえる≫

◎ プレイヤシップ

1P	&H8006，0～15	(10進数)
2P	&H8026，0～15	(10進数)

◎ 弾の色を変える

&H8046，0～15	(10進数)
≀	
&H80E6，0～15	(10進数)

20H ごとに6種類の弾が書き込まれています。

◎ 初期化（変更した要素をすべて元に戻す。）

&H8301，1

または電源を切ります。

敵編隊の変更

≪編隊の出現順序≫

アドレス	初期値	変化幅
&H8340，0		0～ 2
41，1		0～16
42，2		0～16
43，255		0～16

≪編隊の要素変更≫

	アドレス	初期値(10進数)	変化幅(10進数)
編隊のキャラクタ数	&H8360，	5	1～5
編隊のキャラクタ間隔	61，	0	0～255
飛び方変更	62，	5	0～5
〃	63，	0	0～255
〃	64，	55	0～255
〃	65，	88	0～255
〃	66，	3	0～255
〃	67，	2	0～255
キャラクタ変更	68，	25	0～59
キャラクタの色変更	69，	4	0～15

編隊の飛び方変更	&H836A , 96	0～255
	6B ,240	0～255
	以下未定義	

シュートゲームは、&H8000以降に書かれています。

POKE 文で変更する場合、プログラムにすると便利でしょう。BASIC でプログラムの作り方を覚えて応用しましょう。

変更した部分をもとに戻すには、表の初期値を使うか次の命令を実行してください。

```
POKE&H8301 , 1
```

アドレスは16進数で書いていますが、データは16進数と10進数を併用しています。

10 進数　　数字だけで書きます。

16 進数　　数字の前に &H を付ける。または数字の後ろに H を付けて区別します。

2. けいさんボード

計算専用のプログラムです。

BASIC でも、計算はできますが、整数型ベーシックであるために、小数点のつく数は計算できません。また、32767 以上の数も計算できません。それで、この「けいさんボード」をもちました。

＜式の書き方＞

★★★　式で使う記号　★★★

＋	プラス符号　または　たし算
－	マイナス符号 または ひき算
＊	かけ算
／	わり算
(　　)	かっこ

★★★　優先順位　★★★

(a)　かっこで囲まれた部分　　(　)

(b)　マイナス符号をつける

(c)　かけ算またはわり算　　＊　／

(d)　たし算またはひき算　　＋　－

同じ順位の計算記号を二つ以上書いたときは、左側から計算されます。

note !　コンピュータで計算をする場合、かっこはすべて(　)を使います。
{　} のようなかっこは使えません。

＜けいさんボードの使い方＞

メニューの２を選び、番号キーを押すと、「けいさんは？」と表示されます。カーソルが点滅していますから、数式を入力して CR キーを押してください。「こたえは　○○○○○」と答えが一行下に表示されます。

```
セガ　けいさん　ボード
けいさんは？　２５６＊１９２
　こたえは　　４９１５２
けいさんは？　５０－８＊３＋１５／４
　こたえは　　２９．７５
けいさんは？　▦
```

一つの計算が終わると、つぎの計算の入力待ちになります。

計算式をまちがえたときは、カーソルをまちがえた所にあわせて正しく打ち直し、 CR キーを押します。

＜計算できる数．計算精度＞

整数部分８桁に小数点以下８桁（けた）の計算ができます。８桁を越えると「けいさんできません」と表示がでます。

□□□□□□□□．□□□□□□□□＋○○○○○○○○．○○○○○○○○

わり算で割り切れない場合には、小数点以下９桁目で四捨五入して８桁目に表示します。

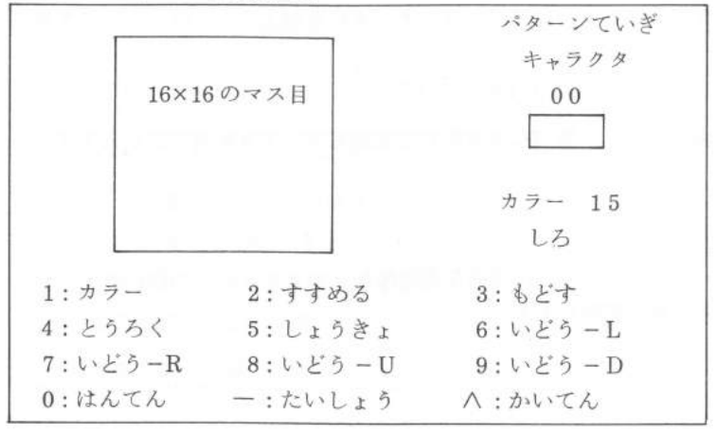
3. BASIC

ベーシックでは、楽しいアニメーションや、実用的な電話帳などのプログラムが作れます。シュートゲームの背景も描けます。第３章で説明します。

4. パターンのへんこう

４のキーを押してください。次の画面があらわれます。これを使って、シュートゲームのキャラクタを変えてみましょう。

パターンの変更画面



ジョイスティックまたはキーボードを使って、画面左にある16×16のマス目を白で埋めてキャラクタを描きます。

機　能	ジョイスティック	キーボード
カーソルを動かす	方向レバー	カーソルキー
点を打つ(白くする)	右のプッシュボタン	INS DEL キー
点を消す(黒くする)	左のプッシュボタン	HOME CLR キー

パターンを描く機能、キー番号の説明

1. **カラー** …… キャラクタの色を決めます。色番号と色の名前がキャラクタの下に表示されます。

   ここで決めた色を参考にして、ベーシックのプログラムの中で使うとよいでしょう。(カラーは登録できません。参考にするだけです。)

2. **すすめる** …… キャラクタ番号を進めます。キーを押し続けるとキャラクタがつぎつぎに変わります。

   番号は、00～59までの60個分あります。(この番号と、ベーシックで使うパターン番号とは関連があります。P. 196 参照)

   キャラクタの登録やコピーなどしたときは、2のキーを押してもすぐには進みません。2のキーを押して"Y"キーを押すとキャラクタが変わります。

3. **もどす** …… キャラクタ番号を戻します。(番号は循環してますから、00の後ろは59,58,57となります。)

4. **とうろく** …… 出来上がったキャラクタを、コンピュータに覚えさせます。

   4のキーを押すと**「これでいいですか (Y／N)？」**と画面から聞いてきます。登録して良ければ"Y"、まだ変更したいときは、"N"のキーを押します。キーを押すと**「これでいいですか (Y／N)？」**の表示が消えて登録が終わります。

   電源を切るまでは、変更したキャラクタはそのままです。

5. **しょうきょ** ……　画面の消去には５つの方法があります。
５のキーを押す前に、あらかじめ消したい場所へカーソルを移動しておきます。

[5] のキーを押すと、“これでいいですか”(Y or N):ALL と表示されます。ALL の文字はカーソルを押すと次のように変わります。キーと意味を覚えましょう。

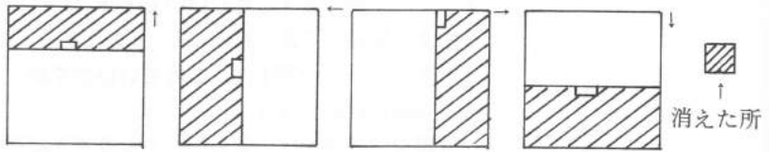
: ALL　全体を消します。

[↑] : UP　カーソルのある位置から上方が消えます。

[←] : LEFT　カーソルのある位置から左側が消えます。

[→] : RIGHT　カーソルのある位置から右側が消えます。

[↓] : DOWN　カーソルのある位置から下方が消えます。

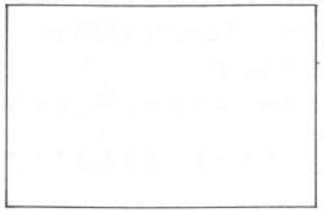


コンピュータからの**「これでいいですか(消去していいですか)」**の問いかけに“Y”キーを押すと**「これでいいですか　(Y／N)?」**の表示が消えてキャラクタも消えます。
“N”キーを押したときはキャラクタは消えません。

消し終わったら [4] のキーで“とうろく”します。

6. **いどう−L** ……　キャラクタが左に移動します。(キャラクタの修正などに使います。)
7. **いどう−R** ……　キャラクタが右に移動します。(キャラクタの修正などに使います。)
8. **いどう−U** ……　キャラクタが上に移動します。(キャラクタの修正などに使います。)
9. **いどう−D** ……　キャラクタが下に移動します。(キャラクタの修正などに使います。)
0. **はんてん** ……　キャラクタの白い点の部分が反対になります。
− **たいしょう** ……　キャラクタを裏返した状態になります。
＾ **かいてん** ……　キャラクタが 90 度回転します。
180 度回転したときは、たいしょうと違い上下の関係が逆さになります。

スペースキーをおすと、画面にある命令が一部変わります。



パターンていぎ
キャラクタ
００

カラー　１５
しろ

1：カラー	2：すすめる	3：もどす
4：とうろく	**5：とりかえし**	**6：コピー**
7：くみあわせ	**－：リスト**	**∧：ごうせい**

(スペースキー)**5　とりかえし**

前ページの5のキーで、うっかりキャラクターを消してしまったときは、これでキャラクタを取り返すことができます。しかし、4のキーで登録した場合は取り返せません。

(スペースキー)**6　コピー**

あるキャラクターを別のキャラクタ番号の所へコピーします。例として、キャラクタ 30 を 41 にコピーします。

① …… まず、コピーしたい 30 番のキャラクタを画面にだしておきます。

② …… 6のキーを押すと
「コピー：FROM　30　TO　30」

「これでいいですか　(Y／N)？」
と画面から聞いてきます。

③ …… そこで、2のキーをおして TO 30 の番号を 41 になるまで押します。
(41 には何も描かれていません。)

④ …… 番号が 41 になったら "Y" キーを押します。

⑤ …… 30 のキャラクタがコピーされてきます。

⑥ …… 4のキーで登録してください。登録を忘れると消えてしまいます。

⑦ …… これでコピーは終わりました。

(スペースキー)**－　リスト**

「パターンのへんこう」画面でつくったキャラクタのデータを与えてくれます。データは次のように、16進数であらわされます。

```
# 160 → 01  01  02  02  05  05  09  09
# 161 → 06  0B  15  20  68  EB  EA  D7
# 162 → 00  00  80  80  40  40  20  20
# 163 → C0  A0  50  A8  2C  AE  AE  D6
```

このデータは、ベーシックの PATTERN 文に入れて使うことができます。

例

PATTERN S # 160 , "  "

↑この部分に上記のデータを入力。

note !　16進数や、PATTERN 文については、第3章 ベーシック の PATTERN文、SPRITE 文の項をご覧ください。

(スペースキー)∧　**ごうせい**

4個までのスプライトを合成したときの状態を確かめられます。スプライトごとに色わけして、それを合わせるとカラフルなキャラクタがつくれます。

① …… 作りたいキャラクタの下絵をグラフ用紙などに16×16のマス目で書いて、それを4個以内に分けます。

② …… 分けた部分を、別々のキャラクタ番号に登録します。
たとえば、40,41,42,43に登録しておきます。
登録した40番のキャラクタを出しておきます。

③ …… 1のキーでカラーをきめます。

④ …… ごうせいのキーを押すと、つぎの表示がでます。
「40 TO 01 」
「これでいいですか （Y／N）?」

⑤ …… 1のキーを押します。(仮りにスプライト番号を1にきめる。)

⑥ …… "Y"キーを押すと、色番号の下にキャラクタ番号とキャラクタが現われます。
これで40番のキャラクタがスプライト#1となりました。

⑦ …… つぎに、"すすめる"で41番のキャラクタをだして、④から同じ手順でスプライト番号を1,2,3,4とします。
(同じスプライト番号を使うと、前のキャラクタが消えてしまいます。)

⑧ …… 必要な回数を繰り返してください。
ここで使ったキャラクタは、参考にするだけでほかには使えません。

合成したキャラクタを消すには、空白を"ごうせい"と同じ手順で消すか、BREAKキーでオープニングに戻します。

(スペースキー)7　**くみあわせ**

2つのキャラクタを組みあわせた形を見たいときに使います。ここで組みあわせて作った形も、参考にするだけで他には使えません。

例として、キャラクタ28と30を組みあわせてみましょう。

① …… まず、組みあわせたいキャラクタ2つのうち片方(28)を画面に出しておきます。

② …… 7のキーを押すと、
「くみあわせ：FROM 28 TO 28」
「これでいいですか （Y／N）?」
と画面から聞いてきます。

③ …… そこで、2のキーを押して TO 28 の番号を、30になるまで押します。

④ …… 番号が41になったら"Y"キーを押します。

⑤ …… 28と30のキャラクタが組みあわされます。

第３章　BASIC

人間の言葉には、日本語、英語、フランス語など、たくさんの言葉があります。

コンピュータも、たくさんの言葉を持っています。ベーシックはその中の一つです。ベーシックを使えば、コンピュータにいろいろな仕事をさせることができるのです。

ベーシックは、英語ですがけっしてむずかしいものではありません。毎日少しずつ使っていると自然に覚えてしまいます。では、やさしいところからはじめましょう。

ダイレクト・モード（直接命令）

命令のしかたには、直接命令（ダイレクト・モード）と間接命令（一般に、行番号がついており、「プログラム」とよばれています。）の二種類があります。

ここでは簡単な方、つまり、直接命令について説明しましょう。(プログラムは、のちほど扱います。)

PRINT（プリント）

ダイレクト・モードで、コンピュータにＰＲＩＮＴという命令を実行させましょう。

PRINT とは、コンピュータが仕事をした結果を画面に出させる命令です。

≪文字や記号を書く≫

BASIC TEST と書いてみましょうか。

次の手順で、コンピュータを操作して下さい。

HOME／CLR キーをおします。

PRINT " BASIC SPACE TEST " と入力してください。

このとき、ダブルクォーテーション " " を忘れずにつけてください。(シフトキーを押したまま 2 のキーを押せば " が出ます。

CR キーを押し、実行させます。

そうすると、次のような画面になります。

```
PRINT " BASIC   TEST "
BASIC   TEST
Ready
```

← BASIC TEST を画面に出せと命令しました。

← 実行しました。

← 次の命令を待っています。

note1 ! ：コンピュータに文字や記号を書かせる場合は、必ず、" "（ダブルクォーテーション）をつけてください。

note2 ! ：PRINT と打ちこむ代わりに、？を使っても、同じ働きをします。

HOME／CLR ? " BASIC TEST " CR

と入力しても、結果は同じです。

note3 ! ：Syntax Error (シンタックス・エラー) について

BASIC 命令を正しく書くためには、シンタックス (構文) という約束を守らなければなりません。BASIC 命令を書き込んで実行したとき、もしも

```
? Syntax Error
Ready
```

と画面に出たら、キーを打ちまちがえているということですから、画面の文字を見直してください。つづりを少し間違えても、コンマの代わりにピリオドを使ってしまった場合にも、エラーになります。正しく入力するよう心がけてください。

≪簡単な計算≫

コンピュータに四則計算をさせましょう。

4＋3　の計算をしましょう。次のような手順をとります。

HOME／CLR キーを押します。

PRINT　4＋3　と入力します。

(または、？4＋3のように、？を使ってもかまいません。)

CR キーを押して実行させます。

そうすると、画面は次のようになります。

```
PRINT  4+3
 7
Ready
```

← 4＋3 の計算結果を出せと命令しました。
← 実行して、答えを出しました。
← 次の入力を待っています。

note1 ! ：計算のときは、ダブルクォーテーション " " は必要ありません。

note2 ! ：コンピュータで計算する場合、計算に使う記号 (演算子) は、一部普通の計算と違います。

たし算 …… ＋(プラス)
ひき算 …… －(マイナス)
かけ算 …… ＊(アスタリスク)
わり算 …… ／(スラッシュ)

note3 ! ：わり切れないわり算について ————

たとえば、10／3の計算をすると、答えは 3 と出ます。小数点以下はきり捨てられるのです。

MOD (モド) を使うと、わり算の余りを出すことができます。

？10 MOD 3　と入力すると、1 と余りが表示されます。

note4 ! ：計算は、−32768 から 32767 の範囲内でのみ、可能です。その範囲をこえる場合は、「けいさんボード」を使ってください。

note5 ! ：四則演算には優先順位があります。それについては、けいさんボードの項をご覧ください。

≪PRINT 命令の応用≫

これまでに、PRINT の二つの使い方を説明しました。
それは文字の表示と、計算の二つでした。それらを組み合わせると、こんなことができます。

```
PRINT " 4+3= " ;4+3
4+3=7
```

← " 4＋3＝ " を文字として表示して、4＋3 の計算結果を表示せよと命令しました。

← 4＋3＝ が表示され、答え (7) が出ました。

note1 ! 文字として扱う 4＋3 には必ず " " ダブルクォーテーションをつけてください。文字として扱う部分と、計算式の部分を区別するために、；(セミコロン) を忘れずにつけて下さい。

note2 ! セミコロンのかわりに、，(コンマ) を使うと、画面の端から 8 桁離れた場所に答が表示されます。

(例) ? " 2＋3＝ " , 2＋3

```
2+3=                    5
```

8 桁

プログラム・モード　その 1

あなたの名前をコンピュータが書きます

あなたの名前は何ですか。ハナコさん？　それともミサコさん？　これから、コンピュータに命令して、あなたの名前を書かせてみましょう。

まずは、ダイレクト・モードを使いましょうか。さきほど、BASIC TEST と書いたのと同じことです。

仮に、アキコ という名前だとします。

HOME／CLR

PRINT　" アキコ "　CR

CR キーを押すと、画面には ア キ コ と表示されますね。

ダイレクト・モードでは、CR キーを押して命令を実行してしまうと、それきりおしまいです。もう一度名前を書かせたいと思ったら、またはじめからコンピュータに打ちこまなければならないのです。何度も同じことをやりたいとき、これでは不便です。どうしましょうか。

次のように入力してください。(各行の終わりに、かならず CR キーを押してください。)

```
NEW
10 PRINT " アキコ "
20 END
```

これが、アキコと書かせるためのプログラムです。これを実行させるには、RUN　と入力し、CR キーを押します。そうすると、

```
アキコ
Ready
■
```

と画面に出ます。命令が実行されたのです。プログラムは、パソコン本体のスイッチを切らない限り、何度でも、RUN と入力すれば同じように実行されます。実際に、何度も RUN と入力してみてください。

```
アキコ
Ready
RUN ―――――( CR キーを押す )
アキコ
Ready
RUN ―――――( CR キーを押す )
アキコ
Ready
RUN ―――――( CR キーを押す )
  :
```

と、RUN するたびにコンピュータは何度でも同じように名前を書いてくれます。

「1度だけ」と「何度でも」。これがダイレクト・モードとプログラムモードとの大きな違いなのです。もうひとつ注目してほしいことがあります。各行の左側につけた番号です。これは行番号と呼ばれ、

プログラムはその番号の順序で実行されます。行番号については、後でもう少し詳しくふれます。

さて、名前もひとつだけではつまらないですね。今度は、たくさん名前を書かせてみましょう。次のプログラムを打ちこんでください。

(プログラムを打ちこむときは、各行の終わりに必ず CR キーを押す習慣をつけましょう。)

```
NEW
10 PRINT "アキコ"
20 GOTO 10
30 END
```

プログラムが完了したら RUN してください。

```
アキコ
アキコ
アキコ
  :
```

と、縦にずっと並びますね。

note ! コンピュータは、いつまでも「アキコ」を書いています。止めるときは BREAK キーを使ってください。

GOTO は、「指定された行番号 (ここでは10です) へとべ」という命令です。コンピュータは忠実にそれを実行し、アキコという名前を、縦に書き続けたわけです。

では、決められた回数だけ、名前を書かせるためにはどうしたらよいでしょうか。

次のプログラムを打ちこんでください。

```
NEW
10 FOR N=1 TO 5
20 PRINT "アキコ"
30 NEXT N
40 END
```

打ちこんだら RUN してください。画面には、

```
アキコ
アキコ
アキコ
アキコ
アキコ
```

と、縦に5つアキコが並びます。これは名前を5つだけ書かせるプログラムなのです。ここで、FOR ～ NEXT という命令が使われていますが、この命令は一定回数、繰り返して仕事をさせる命令です。

さいごに、ちょっと遊んであなたの名前で画面をいっぱいにしてみましょう。次のプログラムを打ちこんでください。


```
NEW
10 PRINT "アキコ" ;
20 GOTO 10
30 END
```

さあ、RUN してください。どうなりましたか？

note ! PRINT 文で；を使うと、文字をすぐ横に並べて表示します。

これまでに、あなたの名前をひとつ書かせたり、たくさん書かせたり、横に並べたり、縦に並べたりしましたが、それらのプログラムの中には、いろいろな命令がでてきましたね。ここで、もう一度見直しておきましょう。

NEW（ニュー）…… プログラムを打ちこむ前に必ずこれを入力してください。前にあったプログラムを消します。

PRINT（プリント）…… 画面に、プログラムの実行結果を表示します。

note 1 ! 文を区切るときは，または；をつかいます。

note 2 ! PRINT CR とすると、改行します。

END（エンド）…… プログラムが終わりであることを示します。

RUN（ラン）…… 打ちこんだプログラムを実行させます。

(例) RUN …… プログラムを先頭から実行する。

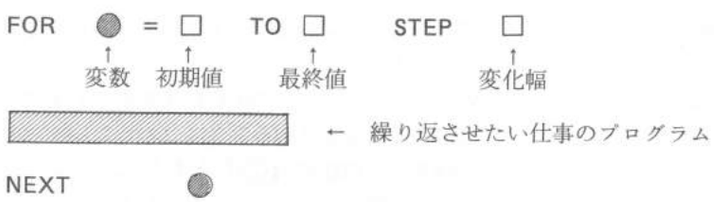
RUN 100 …… 行番号100から実行する。

GOTO（ゴーツー）指定された行番号にとびます。GOTOで指定すれば、どこにでもとぶことができます。

（例）　　行番号

50 GOTO → 10　　前へもどります。
50　　同じところにいます。（グラフィック画面でよく使います。）無限ループをつくるといいます。
100　　先にとびます。

FOR～NEXT（フォー ～ ネクスト）…… 一定回数、仕事を繰り返します。
FOR～NEXT～STEP～（フォー ～ ネクスト ～ ステップ）…… 一般に、



の形であらわされ、ある仕事を、変数が初期値から最終値まで変化する間だけ繰り返します。STEPは変化の幅を指定します。

たとえば、

```
10 FOR N=0 TO 10 STEP 2
20 PRINT N;
30 NEXT N
```

とすると

```
0 2 4 6 8 10
```

のように、（変化幅が2だから）2つおきに数字が並びます。
また、

```
10 FOR N=10 TO 0 STEP -2
20 PRINT N;
30 NEXT N
```

とすると、

```
10 8 6 4 2 0
```

のように、（変化幅が−2だから）2ずつ減って、数字が並びます。

STEP 1は省略可能です。さきほどの例（アキコと書かせるプログラム）では、STEP 1が省略されていたわけです。

```
10 FOR N=1 TO 5        (STEP 1)
20 PRINT "アキコ"
30 NEXT N
```

は、変数Nが1から5まで変化する間（つまり、5回）、PRINT "アキコ" という仕事を繰り返すのです。

FOR～NEXT～STEP文は、二重、三重に重ねることができます。
(8回まで可能です。)

```
10 FOR J=1 TO 2
20 FOR I=1 TO 3
30 PRINT "アキコ";
40 NEXT I
50 PRINT
60 NEXT J
```

このプログラムを実行すると

```
アキコアキコアキコ
アキコアキコアキコ
Ready
```

となります。これを**「入れ子」**といいます。

その他に次のような命令文もあります。

CLS…… 画面上の表示を消す命令です。表示されているものを消しても、プログラムはメモリの中に残っています。その点が **NEW** と違うのです。

note ! プログラムの先頭に入れておけば、画面を消してから実行します。

STOP …… プログラムの実行を途中で止めたいときに使います。

CONT…… STOP や BREAK キーで止めた後、再びプログラムを続けたいとき使います。

STOP と CONT を使った
サンプルプログラムです。

```
10 REM - STOP ト CONT -
20 FOR N=1 TO 100
30 PRINT N;
40 IF N<>50 THEN 80
50 PRINT:PRINT
60 PRINT"CONT ト ｳｯﾃﾈ"
70 STOP
80 NEXT N
90 END
RUN
 1 2 3 4 5 6 7 8 9 10 11 12 1
3 14 15 16 17 18 19 20 21 22
23 24 22 26 27 28 29 30 31 32
 33 34 35 36 37 38 39 40 41 4
2 43 44 45 46 47 48 49 50

CONT ト ｳｯﾃﾈ
Break in 70
CONT
 51 52 53 54 55 56 57 58 59 6
0 61 62 63 64 65 66 67 68 69
70 71 72 73 74 75 76 77 78 79
 80 81 82 83 84 85 86 87 88 8
9 90 91 92 93 94 95 96 97 98
99 100
Ready
```

プログラムは保存できます

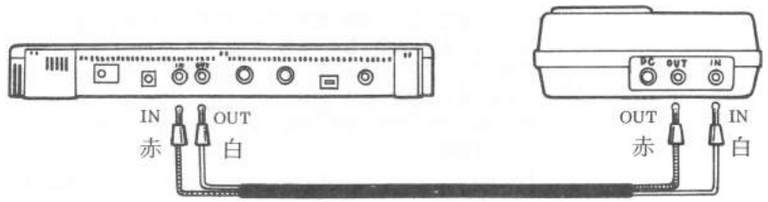
さきほど、ダイレクトモードとプログラムの違いを述べました。プログラムは、同じことを何度でも繰り返し実行できて便利だといいました。けれども、それは、「パソコン本体のスイッチを切らない限り」という条件づきでしたね。せっかく作ったプログラムがスイッチを切ったとたんに消えてしまうなんて…とがっかりしているあなたが目にうかびます。でも、大丈夫。プログラムは保存ができるのです。

SAVE（セーブ）

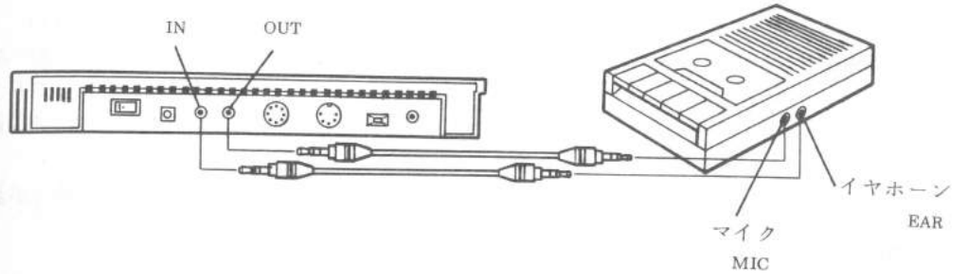
セーブとはカセットテープにプログラムを録音することです。

プログラムはカセットテープに保存しておくことができます。次の手順でやってください。

• セガデータレコーダ SR－1000 を使う場合



• 他社のカセットレコーダを使う場合



カセットレコーダ用のケーブルは図のような、両側がミニプラグになったものを使います。
当社製品のケーブル以外を、お使いになるときは、抵抗の入ってないものをお使いください。

SAVE ″××××″と入力します。(××××には、プログラムの名前を入れます。)
CR キーをおします。
Saving Start の表示がでたら、オーディオカセットの録音ボタンを押します。
約8秒後にピーッと音がしてSAVEが始まります。
Saving End と、記録がおわったことを知らせる表示がでたら、カセットを止めます。
テープによっては、巻き始めの部分に録音できない所があります。少し巻いてからセーブしてください。

note 1! ファイルネーム(プログラムの名前)について
プログラムの内容がわかる名前を16字以内でつけてください。

note 2! カセットデッキはお手持ちのものも使えます。カセットデッキにカウンターがついていましたらカウンターの数字をカセットテープのラベルに記入しておいて下さい。
お手持ちのカセットデッキを使う場合、音のレベルにより、カセットテープに書き込みや、読み取りが出来ない場合があります。
これは、コンピュータの故障ではなく、カセットデッキの性能による場合があります。

VERIFY(ベリファイ)

SAVE(プログラムの保存)が正確に行われたかをチェックするには、次のようにします。

- SAVE を始めた位置までテープを巻きもどします。
- VERIFY と入力して、 CR キーを押します。
- カセットデッキのプレイボタンを押してください。

正しく SAVE されていれば、Verify end と表示されます。
end が出ない場合は、RESET キーで止めて、初めから SAVE しなおしてください。

LOAD(ロード)

カセットテープに SAVE されているプログラムをとり出し、コンピュータに移すことをロードするといいます。

LOADは、次のようにします。

LOAD "××××" (××××はプログラム名)と入力して CR キーを押します。
LOAD CR のようにプログラム名は省略できます。
Loading Start と表示が出たら、カセットデッキのプレイボタンを押してください。
Found "×××"
Loading End と表示が出たら、カセットを止めてください。

note ! **ープリンタを持っている方へー**

プリンタ用紙に印字して、目に見える状態で残すこともできます。

LLIST(エルリスト)

プリンタにプログラムリストを書かせるのは、LLIST 命令です。使い方は次のとおりです。

```
LLIST                          ; プログラム全部を書きます。
LLIST   行番号                 ; 指定した行番号を書きます。
LLIST   行番号 ー 行番号       ; 指定した行番号から行番号までを書きます。
LLIST   ー行番号               ; 先頭から指定した行番号までのリストを書きます。
LLIST   行番号ー               ; 指定した行番号以降のリストを書きます。
```

文字や変数の値をプリンタで印字する命令は LPRINT です。使い方は PRINT と同じです。

```
LPRINT "セガ"
```

と入力すれば、セガとプリントアウトされます。

知っていると便利です

★★★ プログラム内容の確認 ★★★

何かプログラムをつくったとします。それが正しく組まれているか、確めたいときは LIST と入力して CR キーを押して下さい。プログラムが表示されます。これを「リストを出す」といいます。

アキコと書かせるプログラムを例にとりますが、プログラムを END にした後、あるいは、 RUN した後、LIST と入力しますと、

```
10  PRINT  "アキコ"
20  END
```

と、プログラムの全内容が表示されます。表示された内容を見て、訂正したいところは訂正し、つけ加えたいことはつけ加えます。

note1 !

LIST の使い方は、次の方法があります。

LIST		プログラムの内容を全部表示します。
LIST	行番号	1行のみ表示します。
LIST	行番号－行番号	行番号から行番号までを表示します。
LIST	行番号－	行番号から後ろのプログラムの内容を表示します。
LIST	－行番号	プログラムの先頭から行番号までを表示します。

note2 !

LIST したとき、画面いっぱいになるほどプログラムの行数が多いと、リストはせり上がっていきます。

それをとめてリストを見たいときは SPACE キーを押してください。もう1度 SPACE キーを押すと、LIST は再開します。

★★★ プログラムの訂正 その1 ★★★

LIST でプログラム内容を確かめたとき、 何か間違いを発見したら、次のようにして訂正してください。

カーソルを、訂正したい文字（あるいは数字，記号）の上に移動させます。そこに、正しい文字（あるいは数字，記号）を打ち込んでください。

正しく書き替えたら CR キーを押してください。

note1 ! インサート・デリートの項目も参照してください。

note2 ! せりあがっていく長いプログラムリストの中に間違いをみつけたときは、 BREAK キーを押して下さい。その時点で LIST の動きがとまり、カーソルが点滅します。カーソルを動かして、訂正してください。

★★★　プログラムの訂正　その２　★★★

LIST で出したプログラムを見て、一行を消したいときは、行番号だけ入力し、 CR キーを押します。

＜例＞　30 行を消したいとき

30　CR

★★★　行番号のこと　★★★

プログラムの各行に番号をつけるということは、前に述べました。行番号は、たいてい、10　20　30　40　……　というように、10きざみです。どうしてでしょうか。

行番号を、ある程度間隔をあけてつけるのは、あとで新しく行をつけ加える場合、その方が便利だからです。

たとえば

10　××××

20　△△△△

この２行の間に、□□□□という一行を加えたい場合には 15　□□□□ というように、10 と 20 の間にある数字を、行番号にします。もし、仮に１　２　３　４…… というように行番号がついていたなら、新しい一行をつけ加えることができないのです。

note !　行番号の範囲は１から 65535 までです。

★★★　新しい一行をつけ加える法　★★★

```
NEW
10 PRINT "アキコ"
20 END
```

のプログラムの 10 と 20 のあいだに GOTO 10 という一行をつけ加えたいときは、15 GOTO 10 と入力して CR キーを押して下さい。（図①）

（15 の下：行番号）

念のため、LIST してみましょうか。図②のようになりますね。

```
NEW
10 PRINT "アキコ"
20 END
15 GOTO 10
Ready
```

図 ①

```
LIST
10 PRINT "アキコ"
15 GOTO 10
20 END
Ready
```

図 ②

番号順に入力しなくても、ちゃんと小さい番号から並びます。

★★★　自動装置 AUTO（オート）★★★

行番号を打ち込むのがめんどうな方、AUTOを使いましょう。行番号を自動的につけてくれます。

AUTOと入力して CR キーを押してください。自動的に10と表示されましたね。行番号10のところに何かプログラムを入力し CR キーを押すと、今度は20と表示されます。このように、10　20　30　40 …… と、10きざみに自動的に行番号が表示されていきます。

また、次のようにも命令することができます。

AUTO　100　CR　……　行番号が、100番から10きざみで発生

AUTO　10 , 20　CR　…　行番号が、10番から20きざみで発生

(始めの番号,きざみ幅)

AUTOの機能を止めるには、 BREAK キーを押します。

★★★　見出しをつけよう　REM（レム）★★★

プログラムを作成するとき、注釈文(コメント)を入れておくと、後でプログラムリストを見たとき便利です。ちょうど、新聞などの見出しのような役割を果してくれます。

```
10 REM ××× ケイサン ×××
20 PRINT 2+3
 ↓
```

上記のように、REMという命令を使います。REM文の行は実行されません。

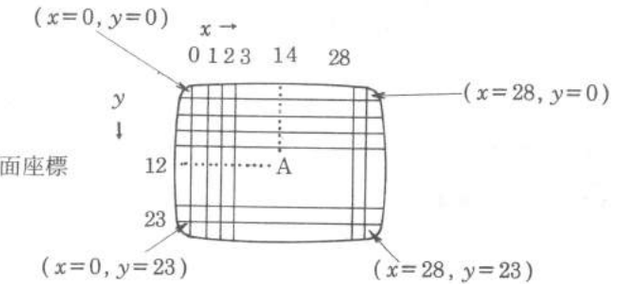
注釈文は、プログラムの内容、製作者の名前、製作した年月など、あなたが後で見易いように工夫してください。　　*note !* REMはリマークとも呼びます。

★★★　画面上のレイアウト !?
CURSOR, SPC, TAB, コンマとセミコロン　★★★

≪CURSOR(カーソル)≫

ふつう、プログラムをRUNすると、画面の左側に表示されますね。でも、表示の位置を指定することもできるのです。CURSORという命令です。

テキスト画面(プログラムを書く画面)は次のように29×24の696のマス目で構成されています。



x(横)とy(縦)の座標指定によって、好きなように表示する場所を指定できます。

CURSOR　x軸, y軸

と入力して下さい。たとえば、上の図の位置にAを表示するには、

CURSOR 14,12:PRINT "A"

と入力して CR キーを押します。

note ! CURSOR 文で座標を指定すると、表示したい文字の先頭位置が決まります。

≪SPC(スペース)と TAB(タビュレーション)≫

文字と文字の間にスペースを設けたい場合は、SPC で指定します。

```
10 PRINT "ABC";SPC(10);"XYZ"
RUN
ABC ␣␣␣␣␣␣␣␣␣␣ XYZ
```

10 文字分の空白(スペース)

TAB は、画面の端から何文字目に出すかを指定します。

```
10 PRINT TAB(5); "ABC"
RUN
␣␣␣␣␣ ABC
```

5 文字分のスペース

note1 ! SPC も TAB も、PRINT 文で使います。

note2 ! SPC で指定した範囲に文字があると消されてしまいますが、TAB の場合は、文字があっても消されません。

≪セミコロンとコンマ≫

セミコロンは、PRINT 文の区切りに使います。セミコロンで区切ると、実行したとき、接近した形で横にならびます。コンマを使うと、間隔を置いた形で横にならびます。(下図参照)

```
PRINT "A";"B";"C";"D";"E";"F"
RUN
ABCDEF

PRINT "A","B","C","D","E","F"
RUN
A   8文字     B          C          D
E              F
```

ここで、HOME BASICで使われる記号について、説明しておきましょう。わかりやすくするために、表にしてみました。

記号名	
セミコロン　(;)	PRINT文の区切りに使います。
コンマ　(,)	PRINT文、INPUT文、DATA文で、区切りとして使います。
コロン　(:)	文の区切り記号。1つの行番号の中に2つ以上の文を書くとき使います。(マルチ・ステートメント)
ダブルクォーテーション　(" ")	この記号でかこまれた文字は、文字のかたまり(文字列)として扱われます。
ドルマーク　($)	文字列変数につけて数値変数と区別します。(PLAY文のフラット記号)
疑問符　(?)	PRINT文の代用です。
負記号またはハイフォン　(−)	負記号、引き算に使います。また、LIST文、DELETE文で行の指定に使います。
スペース (スペースキーを押す)	スペースを設けるときに使います。 (スペースも文字のひとつとして扱われます。)
アンド　(&)	Hとあわせて、&HFのように使用して、16進数をあらわします。
シャープ　(#)	PLAY文で使います。(PLAY文のシャープ記号) SPRITE, PATTERN文でも使います。

記号を(特にセミコロンとコンマ)使いわけて、見易い表示を心がけましょう。

★★★　その他の知っていると便利なこと　★★★

≪残っているメモリは？ FRE(フリー)≫

プログラムを入力すると、メモリがだんだん減っていきます。あとどれくらい残っているか知りたいときは、FRE関数で調べます。

例

```
PRINT FRE [CR]
6538
```

あと6538 BYTE(バイト)のプログラムを入力できることを示しています。
(バイトは単位のひとつです。)

≪カーソルの移動範囲の指定 CONSOLE(コンソール)≫

たとえば、

```
CONSOLE 5, 15
```

と入力すると、5行から19行まで、15段の間をカーソルが移動します。

また、CONSOLEは、クリック音をONにしたり、OFFにすることや、英文字の大小切り替えの指定もできます。

一般に次のような形をとります。

CONSOLE V, L, C, S

V；スクロールの上限(0～22)

L；スクロールの長さ(2～24)

C；クリック音の有無 0;無し 1=有り

S； 英数の大小　0；シフトなし大文字
1；シフトなし小文字

— 解除する場合 —

CONSOLE 0，24，1，0 でもとの状態に戻ります。RESET キーを押しても、もとにもどります。

プログラム・モード　その２

もっといろいろ、計算できます

★★★ LET，変数，INPUT ★★★

ダイレクト・モードのところで、四則計算をやりましたね。4＋3 の計算は、 HOME／CLR PRINT 4＋3 CR でした。

この計算は、プログラム・モードでは

```
10 LET A=4
20 LET B=3
30 LET C=A+B
40 PRINT C
50 END
```

のように入力します。RUN して実行させると、7 と答えが出ます。

LET という新しい命令がでてきましたが、これは「代入する (変数に数値または文字を入れる)」命令です。さきのプログラムの、A＝4 は A という変数に 4 を代入する、B＝3 は B という変数に 3 を代入する、C＝A＋B は C という変数に A＋B を代入する、という意味なのです。

この場合、注意しなければならないことは、必ず、変数を左辺に置くということです。4＝A ではエラーになります。気をつけましょう。

LET は省略することができますから、さきのプログラムは

```
10 A=4          40 PRINT C
20 B=3          50 END
30 C=A+B
```

とすることもできます。

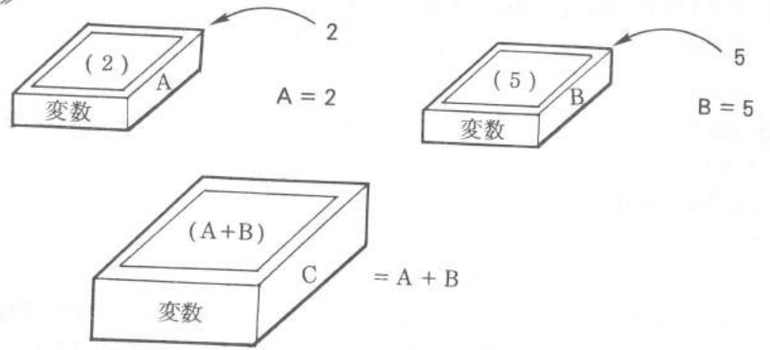
≪ INPUT（インプット）≫

さて、4＋3 の計算だけでなく、1＋2 でも 4＋8 でも、二つの数を加える計算なら何でもできる便利なプログラムにしましょうか。次のように入力してください。

```
10 INPUT A
20 INPUT B
30 C=A+B
40 PRINT C
50 END
```

INPUT は、数値、あるいは文字を入力してくださいという命令です。RUN で実行させると、コンピュータは ? という表示を出して、何か数を入力してくださいと要求してきます。あなたが何か (たとえば 2) を入力し、 CR キーを押すと、また ? が出て、今度は B に数を入力してくださいと要求します。あなたが何か (たとえば 5) 入力し、 CR キーを押すと、コンピュータは A＋B の計算をして答え (この場合 7) を出してくれます。

≪ 変数 ≫



INPUT と ? は組みになっていると考えてください。? が表示されたら、あなたは何かを入力しなくてはなりません。

ところで、INPUT A の A は、さきほどふれた「変数」なのですが、変数にはもう一種類、「**文字変数**」というものがあります。それは一般に A$ のように、アルファベットに $ (ドルマーク) がついた形であらわされます。INPUT A$ と入力して CR キーを押すと、やはり ? マークが表示されます。

文字変数を使って、名前をたずねるプログラムをつくりましょう。

```
NEW
10 INPUT "なまえは"  ;A$
```

```
20 PRINT A$;"さんですね"
30 END
```

これを実行させると

なまえは？　　ときいてきます。そこで、あなたが自分の名前をタイプすると、(ひとしとしましょうか、)

ひとしさんですね

Ready

 　　と画面に出ます。

note1 ! ：INPUT 文は、プログラムの実行中にキーボードから数値や文字を変数に入れるときに使いますが、その際、INPUT 文に続く変数の形に気をつけましょう。

```
INPUT X
```
数字しか入れられません。(×は数値変数)

```
INPUT A$
```
文字や記号を入れます。数字を入れたときは文字として扱われます。　(A$は文字変数)

note2 ! ：2つ以上の文字変数を足すことができます。

```
10 INPUT A$   ← 数字を入れてください。(123を入れてみる。)
20 INPUT B$   ← 数字を入れてください。(456を入れてみる。)
30 C$=A$+B$   ← たし算ではなく数字をつなげます。
```



```
40 PRINT C$   ← 123456とつながります。
50 GOTO 10    ← はじめに戻ります。 文字も入れてみましょう。
```

数字と文字をつなげることもできます。

計算の話にもどります。GOTO を計算プログラムに利用すると、とても便利です。やってみましょう。たし算のプログラムに GOTO を使った一行をつけ加えて、次のように入力して下さい。

```
10 INPUT A
20 INPUT B
30 C=A+B
40 PRINT C
50 GOTO 10
60 END
```

note ! ：INPUT 文は、キーから入力があるまで、? を表示して待っています。

このようにすると、そのつど RUN しなくても、A と B に数値を与えてあげれば、コンピュータは次々に計算して答えを出してくれます。

プログラム・モード　その３

今までは、コンピュータの仕事といっても、ほんの初歩のことでした。コンピュータを使わなくてもできるようなことでした。でも、こんなこともできるんですよ。

合格と不合格にふりわけます

★★★ IF ～ THEN ★★★

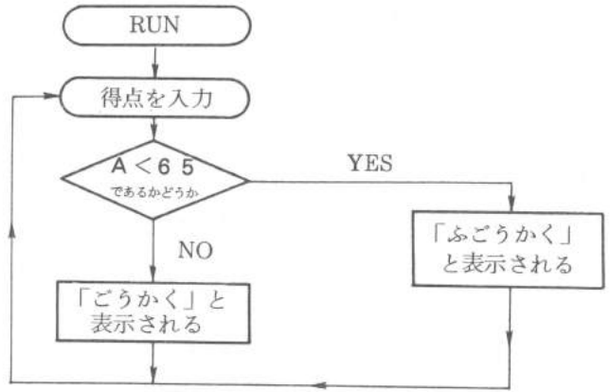
IF ～ THEN は条件判断をします。ある条件に従って、次にすすむべき方向を指定するのです。

具体的に、合格，不合格にふりわけるプログラムをつくってみましょう。

```
10 INPUT A
20 IF A＜65 THEN 100
30 PRINT〝ごうかく〟
40 GOTO 10
100 PRINT 〝ふごうかく〟
110 GOTO 10
```

実行するとコンピュータは、まず、「とくてん？」と聞いてきますから、あなたは得点を入力して下さい。入力した得点が 65 点以上であれば「ごうかく」、65 点に満たないと「ふごうかく」と表示されます。

このような、ある時点で分岐するプログラムは、フローチャート（流れ図）に書くと、よくわかります。



IF ～ THEN は、次のように使うこともできます。

IF～THEN GOSUB 行番号	(指定行番号にとびます)
IF～THEN PRINT 〝×××〟	(画面に表示します)
IF～THEN END	(プログラム終了)
IF～THEN STOP	(プログラム実行中止)
IF～THEN BEEP	(音を出します)

ゴチュウモンは？

★★★ どこにとぶ ON ～ GOTO ★★★

IF ～ THEN と似ているものに、ON ～ GOTO があります。

IF ～ THEN は、

IF [条　件] THEN [実行すべき仕事]

の形をとり、もし [条　件] なら [実行すべき仕事] せよという命令です。

それに対して、ON ～ GOTO は、

ON [変　数] GOTO [行番号] , [行番号] , [行番号] ・・・・

の形をとります。変数が1であれば、GOTO の後に並んでいる1番目の行番号へ、変数が2であれば、2番目の行番号へとんで仕事を実行せよという命令なのです。IF ～ THEN が、ある条件によって行先が決まるとすれば、ON ～ GOTO は、変数の値によって行先が決まるのだといえます。

では、実際に ON ～ GOTO を使ってプログラムをつくりましょう。次のプログラムは、表示されたメニューの中から何か品物を選ぶと、その値段がでてくるというプログラムです。

```
5 CLS
10 PRINT "メニュー"
20 PRINT " 1・・・コーヒー"
30 PRINT " 2・・・ジュース"
40 PRINT " 3・・・ケーキ"
50 PRINT "INPUT 1～3"
```



```
60 INPUT "ゴチュウモンは？"; A
70 ON A GOTO 100, 200, 300
80 GOTO 10
100 PRINT:PRINT "コーヒー・・・￥250"
110 PRINT:GOTO 10
200 PRINT:PRINT "ジュース・・・￥300"
210 PRINT:GOTO 10
300 PRINT:PRINT "ケーキ・・・￥350"
310 PRINT:GOTO 10
```

note ! 100, 110, 200, 210, 300, 310 行のはじめの PRINT は、プログラムに一行分の空白をつくり、見やすくするためのものです。

このプログラムを実行すると、画面は次のようになります。

```
メニュー
1・・・コーヒー
2・・・ジュース
3・・・ケーキ
ゴチュウモンは？
```

あなたが1を入力すると、100行が実行され、

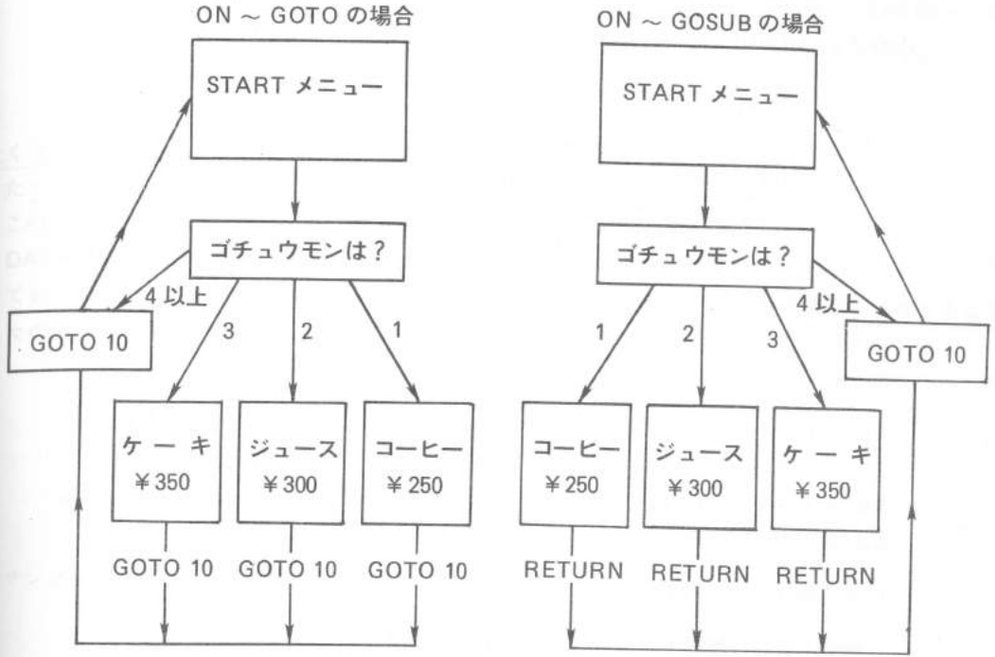
```
コーヒー・・・ ￥250
```

と表示されます。2を入力すると200行、3を入力すると300行が実行されます。

ON ～ GOTO を使った上記のプログラムは、ON ～ GOSUB を使って次のように書き変えることができます。ON ～ GOSUB は、RETURN と組みで使います。

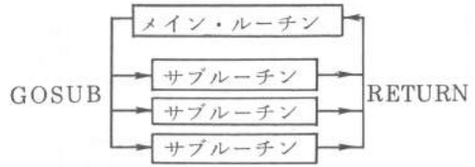
```
10 PRINT "メニュー"
20 PRINT " 1・・・コーヒー"
30 PRINT " 2・・・ジュース"
40 PRINT " 3・・・ケーキ"
50 PRINT " 1～3までのかずをいれてください"
60 INPUT "ゴチュウモンは？"; A
70 ON A GOSUB 100,200,300
80 GOTO 10
100 PRINT:PRINT "コーヒー・・・ ￥250"
110 RETURN
200 PRINT:PRINT "ジュース・・・ ￥300"
210 RETURN
300 PRINT:PRINT "ケーキ・・・ ￥350"
310 RETURN
```


二つのプログラムを図にすると次のようになります。



ON ～ GOTO の場合は、プログラムのはじめ (行番号 10) にもどすには、GOTO を使いますが、ON ～ GOSUB の場合は、RETURN で GOSUB の次の行にもどります。

note ! ：GOSUB で指定されてとぶ先を「サブルーチン」といいます。



デジタル時計に早がわり ?!

コンピュータの内部には、水晶発振の正確なクォーツ・デジタル時計の機能があります。次のプログラムを実行したら、

```
10 TIME$ = " 08:15:00 "  ← 現在の時刻 (" 時：分：秒 ")
20 CURSOR 15,15          ← 表示する場所を指定します。
30 PRINT TIME$           ← 時刻を表示しなさいと命令しました。
40 GOTO 20               ← 時を刻み続けます。
```

コンピュータは時計に変身です。

note1 ! ：解除するには、RESET を押します。

note2 ! ：ダイレクト・モードで

PRINT TIME$

と入力して CR キーを押すと、電源を入れてから、または RESET してからの時間が表示されます。

たくさんのデータも．．．READ , DATA

たくさんのデータの処理も、READ, DATA 文を使えば簡単です。

この二つの命令はいつでも組みで使います。

DATA 文でデータをあらかじめ入力しておき、READ 文でそれを読みます。DATA 文をどこに置いても、READ 文はプログラムにある DATA 文の、始めのデータから読みだします。

note1 ! ：注意すること

DATA の数が少ないと Out of data error になります。

note2 ! ：DATA 文にそのまま使えない記号があります。

, は " , "
: は " : "
" は " " "

のようにします。

サンプルプログラムを参考にして、応用して下さい。

たとえば友だちの電話番号。ギッシリ書きこまれたアドレス帳から目当ての番号をさがすのは大変ですね。名前を打ち込むだけで電話番号がわかる、そんなことができたらいいなと思いませんか？名付けて「コンピュータの電話帳」。案外簡単にできるんです。

```
100 REM ----------------------
110 REM: デンワチョウ
120 INPUT "シメイヲ イレテクダサイ：";IN$
130 READ SHI$, TEL$
140 IF IN$ <> SHI$ THEN GOTO 160
150 PRINT SHI$;"  ";TEL$
160 IF SHI$ <> "**" THEN GOTO 130
170 END
180 DATA SEGA,03-742-3171,名前,番号,名前,番号 ……
999 DATA **,**
```

変数は英字２文字まで有効で、３字目からは無視されます。ここでは、SHI$はSH$、TEL$はTE$として扱われます。

このプログラムをRUNすると、「シメイヲ　イレテクダサイ」と、コンピュータが要求してきます。あなたが、たとえば、SEGAと入力すると、コンピュータは、あらかじめ入れておいたデータを初めから読んでゆき、さがし出し、名前とその電話番号を表示してくれます。

実行したとき、いったいどのようなことが行なわれたのでしょうか。もう少し詳しく、上のプログラムを分析しましょう。


```
100 REM ----------------------
110 REM: デンワチョウ          …… プログラムのタイトルをつけます。
120 INPUT "シメイヲ イレテクダサイ：";IN$……名前の入力を要求します。
130 READ SHI$,TEL$             …… 180行のDATA文のデータをはじめから
                                  読んでゆきます。
140 IF IN$ <> SHI$ THEN GOTO 160 …… インプットした名
                                  前がデータの中にある名前と一致するか
                                  どうか判断しています。
150 PRINT SHI$;"  ";TEL$…… データの中に、インプットした名前と一
                                  致する名前をみつけたら、名前と電話番
                                  号を表示します。
160 IF SHI$ <> "**" THEN GOTO 130 …… 一致する名前が
                                  みつからないと、130行へもどってデータ
170 END                           を読み続けます。
180 DATA SEGA, 03-742-3171.....   あらかじめたくさんの名前
                                  と電話番号をデータとして入れておきます。
999 DATA **,** ……              データの終わりを示しています。READ文
                                  は、はじめから順にデータを読み続け、
                                  終わりのデータ**,**を読んだところで、読むのをやめます。
```

★★★　RESTORE（リストアー）★★★

電話帳のプログラムに、RESTORE文をつけると、＊＊を読んだところで、プログラムの始めにもどり、名前をきいてきます。

次のプログラムは、デンワチョウのプログラムにRESTORE文を付け加えて、繰り返し使えるようにしたものです。

```
100 REM ----------------------
110 REM: デンワ チョウ
120 INPUT "シメイヲ イレテクダサイ：";IN$
130 RESTORE
140 READ SHI$,TEL$
150 IF IN$<>SHI$ THEN GOTO 200
160 PRINT
170 PRINT SHI$;"  ";TEL$
180 PRINT
190 GOTO 120
200 IF SHI$<>"**" THEN 140
210 PRINT
220 GOTO 120
230 END
240 DATA 名前,番号,名前,番号,名前,番号 ……
999 DATA **,**
```


note !　このプログラムを実際に電話帳として応用するには、190行からDATA文を付けたしてください。

```
190   DATA   名前，電話番号   CR
200   DATA   名前，電話番号   CR
 ≀     ≀      ≀      ≀      ≀   の形、或いは

190   DATA   名前，番号，名前，番号 …… と続けて並べる形で入
```

力します。

プログラムの中のREADとDATAに注目してください。この2つの命令文が重要な役割を果しているのです。

READとDATAはいつも組みで使います。DATA文でたくさんのデータを入れておいて、使いたいときにREAD文で読み出します。DATA文が、いくらプログラムの後にあっても、プログラムの流れがREAD文のところにくると、DATAを先に読みます。

気をつけることは、READ文で使う変数の数が、データより多いとOut of data errorになるということです。ないデータを読むということは不可能ですから。

逆に、データの方が多いときは、エラーにならず、余ったデータを無視して、先へ進みます。その先にまたREAD文があれば、データの続きを読みます。

note !　データ文に，、：、＂を使う場合は、＂　＂で囲みます。

★★★ さらに便利に …… ★★★

デンワチョウのプログラム、もうひとつ書いておきます。260行から990行までに、友達の名前と電話番号を付け加えてから使ってみましょう。

該当するデータがないとき、「トウロクサレテイマセン」と画面に表示するプログラム

```
110 REM  デンワ  チョウ
120 INPUT "シメイヲ イレテクダサイ：";IN$
130 RESTORE
140 READ SHI$,TEL$
150 IF IN$<>SHI$ THEN GOTO 200
160 PRINT
170 PRINT SHI$;"  ";TEL$
180 PRINT
190 GOTO 120
200 IF SHI$<>"**" THEN GOTO 140
210 PRINT
220 PRINT "トウロクサレテ イマセン"
230 PRINT
240 GOTO 120
250 END
260 DATA SEGA,03-742-3171 ，名前，番号，名前，番号……
999 DATA **,**
```


配列（もうひとつの変数）

配列は、添字（そえじ）つき変数とも呼ばれます。変数ですから、数値や文字を入れる箱には違いないのですが、今までに学んだ変数よりも便利なのです。配列を使うと、一度にたくさんの箱を用意することができます。

配列を使う場合は、

DIM 配列名（添字の最大値）

この形式で配列を宣言します。たとえばDIM A（4）とすると、

A (0)
A (1)
A (2)
A (3)
A (4)

の5つの箱が用意されます。

と、5つの変数をひとつひとつ用意するよりも楽なわけです。

配列には、一次元配列と二次元配列があります。

一次元配列

添字（カッコの中の数字）がひとつだけです。

例

```
10 DIM D(5),D$(5)
20 FOR N=0 TO 5
30 INPUT "なまえは？";A$
35 INPUT "とくてんは？";A
40 D$(N)=A$
45 D(N)=A
50 NEXT N
60 FOR N=0 TO 5
70 PRINT D$(N);D(N)
80 NEXT N
```

RUNしたら、6人分の名前と得点を順番に入れてください。

二次元配列

添字がふたつです。

例1　ゲームの点数表を作りましょう。3名のゲームの得点を表にしましょう。

ナマエ / コウモク	Aサン	Bサン	Cサン
No.1ゲーム	D(1,1)	D(1,2)	D(1,3)
No.2ゲーム	D(2,1)	D(2,2)	D(2,3)
No.3ゲーム	D(3,1)	D(3,2)	D(3,3)

```
10 DIM D(3,3)
20 FOR M=1 TO 3
25 INPUT "コウモク=";C$(M)
30 FOR N=1 TO 3
40 INPUT "テンスウ=";A
50 D(N,M)=A
60 NEXT N
80 NEXT M
90 FOR M=1 TO 3
95 PRINT C$(M);";";
100 FOR N=1 TO 3
120 PRINT D(N,M);
130 NEXT N
140 PRINT
150 NEXT M
```

DIM D(3,3)の一行で9(3×3)個の箱が用意されます。

例２　九九の表を作りましょう。

```
10 DIM J(9,9)
20 FOR N=1 TO 9
30 FOR M=1 TO 9
40 J(N,M)=N*M
45 IF J(N,M)<10 THEN PRINT"  ";
50 PRINT J(N,M);
60 NEXT M
70 PRINT
80 NEXT N
```

DIM J(9,9)で、81(9×9)個の箱が用意され、

1	2	3	4	5	6	7	8	9
2	4	6	8	10	12	14	16	18
3	6	9	12	15	18	21	24	27
4	8	12	16	20	24	28	32	36
5	10	15	20	25	30	35	40	45
6	12	18	24	30	36	42	48	54
7	14	21	28	35	42	49	56	63
8	16	24	32	40	48	56	64	72
9	18	27	36	45	54	63	72	81

このような表ができあがります。

★★★　配列のとりけし....ERASE（イレース）　★★★

一度配列を宣言したら、同じ配列をプログラム内でもう一度使いたくてもできません。そのようなときは、いったん配列をとりけして、もう一度宣言します。

配列を無効にするにはERASEを使います。次の２つのやり方があります。

```
100 ERASE(プログラム内の配列をすべて無効にします。)
100 ERASE 配列名 , 配列名 ,……(指定した配列のみ無効にします。)
```

note 1 !　配列を宣言すると、配列を入れるメモリだけ先に確保しますから、あまり大きな配列を宣言すると、プログラムする場所が小さくなります。

note 2 !　このベーシックは、添字の最大値10まではすでに用意されているので、DIM (10)までの配列は宣言をしなくてもかまいません。その場合は、自動的に宣言されます。しかし、DIM (11)以上の配列を使うときは、必ずプログラムのはじめに、DIM 配列名(添字) と入力して宣言してください。

文字列関数と関数

変数のことを思い出してください。数値の変数と文字の変数の二種類がありましたね。

プログラムに変数は不可欠で、変数という箱に数値、あるいは文字を放り込んで使うことはひんぱんです。

ここでは、変数の箱に入れる数値や文字を扱う方法を学びましょう。

文字列と数値の入れ換え　VAL と STR$

★★★　VAL（バリュー）★★★

VALは、文字列として扱われている数字を、数値に変えます。

例1

```
PRINT VAL ("123")      — 文字列としての123が……
 123                   — 数値の123として表示されました。
```

note !　数字を画面に表示したとき、先頭に一字分のスペースができます。
これは、+(プラス)記号が入るところですが、それを省略してあるためです。

例2

```
10 A$="234"                — ここでは234は文字扱い。
20 B$="222"                — 222も同様。
30 C$=A$+B$                — 文字列と文字列のたし算。
40 C=VAL(A$)+VAL(B$)       — 数値と数値のたし算。
50 PRINT C$
60 PRINT C
RUN
 23422                     ← C$ (文字列のたし算の結果)
 456                       ← C  (数値のたし算の結果)
```

★★★　STR$（エス・ティー・アールダラー）★★★

数値を文字列にします。

```
10 A=123                   — 数値の123と
20 B=456                     456が……
30 C$=STR$(A)+STR$(B)      — 文字列に変えられ、たし算されました。
40 PRINT C$
RUN
 123 456
```

アスキーコードのこと　ASC と CHR$

コンピュータは文字や記号を理解できません。コンピュータ内部ではすべて、アスキーコードと呼ばれる番号で理解しています。(付録のキャラクタセット・キャラクタコード参照。)

文字や記号と、アスキーコードの変換をするのがＡＳＣ文とＣＨＲ＄です。

★★★　ＡＳＣ（アスキー）★★★

ASC 文は、例のように文字を数字に変換します。

```
?ASC("Z")
 90
?ASC(" ")
 32
?ASC("1")
 49
```

スペース（空白）です。　これも文字のひとつです。

このように文字を数字にしますから、文字の順位をくらべることができます。

ＡＳＣと反対の命令がＣＨＲ＄です。

★★★　CHR$（キャラクタダラー）★★★

数字を文字記号に変えます。

アスキーコードの 0 から 255 のうち、32 から 255 までの数字に、記号と文字が割りあてられています。0 から 31 まではコントロールコードとよばれ、画面に出力されても文字や記号を表示することはありませんが、何らかの特別な働きをします。

例 1（32 から 255 までのアスキーコードを文字、記号に変換）

```
10 FOR N=32 TO 255
20 PRINT CHR$(N);"  ";
30 NEXT N
□ ! " # $ ……
```

CHR$(32) はスペース（空白）です。スペースも文字と同じ扱いを受けます。

例 2（コントロールコード 17 と 16 の働き）

```
10 SCREEN2:CLS
20 PRINT CHR$(17) ← 文字を横 2 倍にする。（SCREEN 2 のときに限ります。）
30 PRINT "ABC"
40 PRINT CHR$(16) ← 文字を標準にする。
50 PRINT "ABC"
```

文字列を扱う　LEN, LEFT$, RIGHT$, MID$

★★★　LEN（レングス）★★★

LEN は、文字列の長さを調べます。

```
10 A$="BASIC"
20 N=LEN(A$)
30 PRINT N
RUN
 5
```

A$ の持っている "BASIC" は何文字であるかを調べました。

★★★ LEFT$, RIGHT$, MID$ ★★★

長い文字列の中から、文字の一部を取り出す関数です。

例 1 LEFT$（レフトダラー）

文字列 A$ の、左から 4 番目までを取り出して、M$ の中に入れ、画面に表示させます。

```
10 A$="コーヒー ココア ミルク"
20 M$=LEFT$(A$, 4)
30 PRINT M$          ←(左から 4 番目までの文字)
RUN
コーヒー
```

例 2 RIGHT$（ライトダラー）

文字列 A$ の、右から 3 番目までの文字を取り出して M$ の中に入れ、画面に表示させます。

```
10 A$="コーヒー ココア ミルク"
20 M$=RIGHT$(A$,3)
30 PRINT M$          ←(右から 3 番目までの文字)
RUN
ミルク
```


例 3 MID$（ミッドダラー）

文字列の左から 6 番の文字を起点として 3 つの文字を取り出して、M$ の中に入れ、画面に表示させます。

```
10 A$="コーヒー ココア ミルク"
20 M$=MID$(A$, 6, 3)
30 PRINT M$          ↑起点   ←取り出す文字数
RUN
ココア
```

数値を扱う SGN, ABS, HEX$

★★★ SGN（サイン） ★★★

ある数の正負符号を与えます。

例 PRINT SGN(−5)
−1

ある数が正（8, 120 など）のとき ⇨ 1
ある数が 0 のとき ⇨ 0
ある数が負（−7, −150 など）のとき ⇨ −1

3 種類の結果しかありません。

★★★　ABS（アブソリュート）★★★

ある数の絶対値を求めます。

絶対値とは、その数字についている符号（プラスやマイナス）を取り去った形をいいます。

例　PRINT ABS(−5)
　　5

PRINT ABS(6＊(−5))
30

PRINT ABS(2＊3)
6

★★★　HEX＄（ヘキサダラー）★★★

10進数を16進数に変換します。コンピュータは、10進数よりも16進数の方が得意です。

PRINT HEX＄(255)　この逆は　PRINT &HFF
FF　　255

＜10進数と16進数＞

10進数　0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16
16進数　0 1 2 3 4 5 6 7 8 9 A B C D E F 10

16進数ではFの次で桁上りして10になります。

PLAY文とSOUND文。

この2つを使うと音が出ます。

音楽を奏でるにはドレミ …… で入力できるPLAY文、効果音や擬音を出すのは周波数で入力するSOUND文がよいでしょう。

まずはPLAY文。

★★★　PLAY（プレイ）★★★

PLAY "CEG" CR

PLAY文と文字列で、音を出してくれます。文字列の中には、音高、オクターブ指定など、いろいろな要素が入ります。いろいろな要素って??　これから順々に説明していくとしましょう。

PLAY文で指定できる要素は

- 音高　　ドレミファ……（もしくはCDEF……）
- オクターブ指定　　Qに数字をつける（1〜5）
- 音符（音の長さ）　　音高に数字をつける（0〜9）
- 休止符　　Rに数字をつける（0〜9）
- 音の大きさ　　Vに数字をつける（0〜15）

- テンポ (速さ)　　T に数字をつける (40 ～ 208)
- 音色　　I に数字をつける (0 ～ 2)
- 和音　　: で文字列をわける

1. 音高 (ドレミファ ……)

英字、カナ、ひらがなのいずれかで入力します。
ドはト、ファはフに省略してもかまいません。

音高

	音高	ド	レ	ミ	ファ	ソ	ラ	シ
入	英字	C	D	E	F	G	A	B
	カナ	ド,ト	レ	ミ	ファ,フ	ソ	ラ	シ
力	ひらがな	ど,と	れ	み	ふぁ,ふ	そ	ら	し

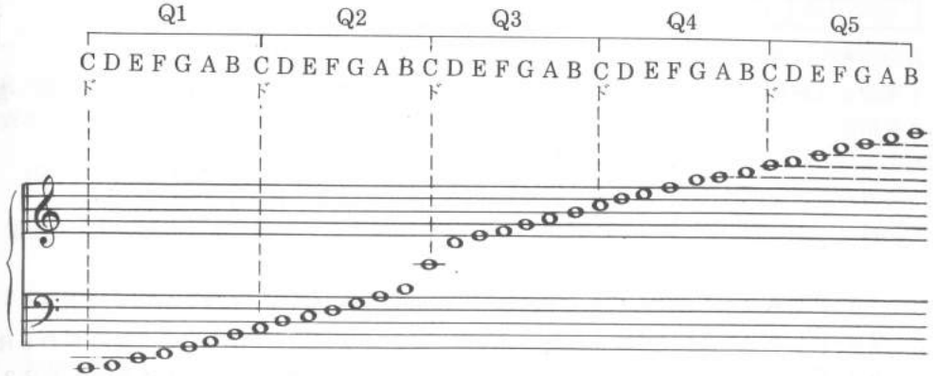
半音は音高の後に # (シャープ) や $ (フラット) を付けます。指定した音高のみ有効です。

(例)

“CEF#” 又は “ドミファ#”

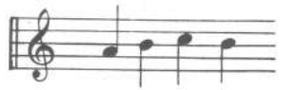
2. オクターブの指定

図を参考に、Q1 ～ Q5 のいずれかに指定します。(電源を入れたときは Q3 になっています。)



一度 Q4 などと指定した場合は、次のオクターブ指定があるまでその指定は有効です。
臨時に一音だけ上または下のオクターブの音を指定する時は、(例)のように、音高符号の直前に +(高)、または −(低) をつけます。この指示はその音のみ有効です。

(例)



“Q3ラ5シQ4 ドQ3シ”
又は
“Q3ラ5シ+ドシ”

3. 音の長さ（音符）

音符	𝅘𝅥𝅯	𝅘𝅥𝅮	𝅘𝅥𝅮.	♪	♪.	♩	♩.	𝅗𝅥	𝅗𝅥.	𝅝
記号	0	1	2	3	4	5	6	7	8	9

表を参考に、音高の後ろに数字をつけてください。
（電源を入れたとき、またはリセットしたとき、音の長さは５になっています。）音高と音の長さを指定して音符にします。
たとえば、C７は「ドの二分音符（𝅗𝅥）」です。
一度音の長さを決めれば、その後に続く音は同じ長さで演奏されます。
(例)

～ ジングルベル ～

```
10 PLAY "C3AGFC5R3
20 PLAY "C1CC3AGFD5R
90 END
```

～ ジングルベル ～

```
10 PLAY "C3AGFC5R3
20 PLAY "C1CC3AGFD5R
50 PLAY "E3+C#BAE5R3
60 PLAY "E1EE3+C#BAG$5
90 END
```

シャープとフラットを使っています。

note ! 楽譜を、あらかじめC3AG……のように書き変えておくと、入力するときラクです。

4. 休止符

休止符	𝄽	𝄾	𝄿	𝅀	𝅁	𝄼	𝄼.	𝄻	𝄺	𝄹
記号	R0	R1	R2	R3	R4	R5	R6	R7	R8	R9

Rは休止符の記号です。休止符は音符と同じように使います。
表を参考に、Rの後ろに数字を付けてください。

(例) PLAY "C5RER3G5" CR

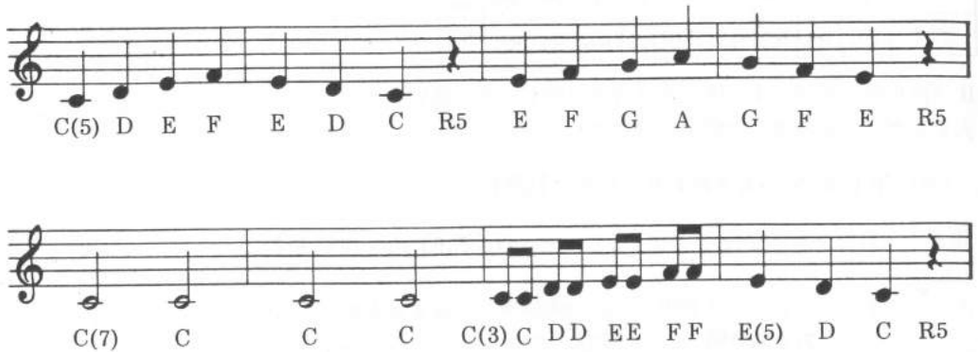
電源を入れたとき、またはリセットしたときは、R５になっています。

さて、今までのテクニックを使って、一曲演奏してみませんか。

カエルの唄 — 輪唱です —

```
200 PLAY "T160:T160"
210 PLAY "Q4CDEFEDCR5:R5RRRRRRR"
220 PLAY "Q4EFGAGFER5:Q1CDEFEDCR5"
230 PLAY "Q4C7CCC:Q1E5FGAGFER5"
240 PLAY "Q4C3CDDEEFF:Q1C7CCC"
250 PLAY "Q4E5DCR5:Q1C3CDDEEFF"
260 PLAY "Q4R5RRRRRRR:Q1E5DCR5"
290 GOTO 200
```

カ エ ル の 唄



note 1 ! 200行の T 160はテンポ指定です。
後で説明します。

note 2 ! 210行 ～ 260行の：は、音を重ねるためのものです。輪唱になっているのは、このためです。(和音の項　参照)

5. 音の大きさ

V 0から V 15まで使って音の大きさを変えます。

(例)

```
10 PLAY "V8 C3AGFC5
20 PLAY "V15 C1CC3AGFD5
90 END
```

V0　OFF（次の音の大きさをきめるまで音がでません。）
〜
V15　最大音量
電源を入れたとき、またはリセットしたときは、V12になっています。

6. 速さ (テンポ)

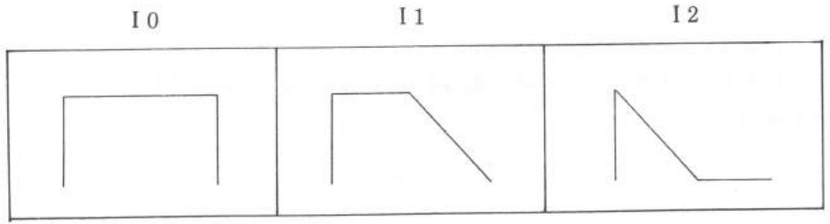
1分間に四分音符をいくつ使うかを数字で指定します。(T 40 ～ T 208 の範囲で指定。)

(例)

```
5 PLAY "T40"
10 PLAY "V8 C3AGFC5
20 PLAY "T160 V15 C1CC3AGFD5
90 END
```

7. 音色

音色の記号はIです。I0からI2まで3つの音を使いわけられます。



この音の波形を参考にして音を組み立てましょう。

note !

タイ  2つ以上の同じ高さの音を一つの音符のように演奏します。

```
"I0C3I1C
```

スラー  2つ以上の違う高さの音を切れ目なく演奏します。

```
"I0C3I1E
```

タイ、スラー、三連音符などは楽器の種類や演奏のテクニックによるものですから、楽譜をそのまま入力してもきれいに聞えない場合があります。耳で確かめながら修正しましょう。

8. 和音（2つ、或いは3つの音を重ねる）

文字列の中を":"で区切ると和音が出せます。

(例1) PLAY "ト:ミ:ソ" CR （ドレミの和音）

note 1 ! 二重和音、三重和音を含む楽譜は、小節ごとに行番号をつけると間違えずに入力できます。

note 2 ! Q, V, T, Iの指定は各和音それぞれ独立して設定できます。

(例2) PLAY "T60Q1C5EG:T120Q4C5EGGEC"

例2では左側は四分音符が3つ、右側は四分音符が6つですね。でも、テンポがT60とT120のように、2倍になっているため、PLAYしたとき、左右同時に終わります。

和音を：で区切った場合、タイミングを合わせたいときは、最初に1つだけ音符や休止符を入れます。

(例3)
```
10 PLAY "R:R"
20 PLAY "CEG:-E-G-E"
```

note ! －をつけると、オクターブ低くなります。

和音を使用した長い曲のタイミングを合わせるには、音符や休符で 1/60 秒の区切に端数が出ないテンポにセットすると良いでしょう。
たとえば、T＝90，T＝120　など

PLAY 文　サンプルプログラム

カチューシャ　　ブランデル作曲

楽譜の小節とプログラムの行番号は 10 行ごとに対応しています。

```
5   PLAY  "R          :R        :R"
10  PLAY  "F6G3       :R7       :R7"
20  PLAY  "A$6F3      :R7       :R7"
30  PLAY  "A$3A$GF    :R7       :R7"
40  PLAY  "G5C3R3     :R7       :R7"
50  PLAY  "G6A$3      :R7       :R7"
60  PLAY  "B$6G3      :R7       :R7"
70  PLAY  "B$3B$A$G   :R7       :R7"
80  PLAY  "F7         :R7       :R7"
90  PLAY  "Q4C5F      :A$5Q4D$  :R7"
```

和音のスタートタイミングを合わせるため

単音の演奏


```
100 PLAY "I0E$5I1F3E$:C5D$3C :R7"
110 PLAY "I0D$3I1D$:Q3B$3B$  :R5"
111 PLAY "I0C3I1−B$3:A$G     :R5"
120 PLAY "C5−F      :A$5F    :R7"
130 PLAY "R3D$5−B$3:R3B$5G3:R7"
140 PLAY "C6−A$3    :A$6F3   :R7"
150 PLAY "Q3G3CA$G:G3CDE:    R7"
160 PLAY "F7        :F7      :R7"
170 PLAY "Q4C5F     :A$5+D$  :R5G"
180 PLAY "I0E$      :+C5     :A$3"
181 PLAY "I1F3      :+D$3    :F5"
182 PLAY "E$        :+C      :A#3"
190 PLAY "I0D$      :B$3     :R3"
191 PLAY "I1D$      :B$      :F3"
192 PLAY "I0CI1−B$  :A$G     :FF"
200 PLAY "I1C5−F    :A$5F    :F5F"
210 PLAY "Q3R3      :R3      :R3"
211 PLAY "+D$5      :B$5     :F5"
212 PLAY "B$3       :G3      :F3"
220 PLAY "+C6A$3    :A$6F3   :F6R3"
```

2 重和音の演奏

3 重和音の演奏

```
230 PLAY "GCA$G      :GCDE   :R7"
240 PLAY "F7         :F7     :R7"
300 GOTO 5
```
2重和音の演奏

110, 180, 190, 210 行は 1 行に納まらないので分割してあります。
100 行の I0, I1 でスラー(♪♪)の感じを出します。
110 行の I0, I1 でタイ(♫♫)の感じを出します。 タイは同じ高さの音符を同一の音符のように演奏するので、♫ を ♩ に置き替えても良いのですが、演奏を聞いて感じの良い方を使います。

note ! PLAY 文の実行は 1 行ずつ実行されるのではなく、文字列を一度バッファ (メモリーの一部) に取り入れてから実行します。ですから、演奏中に BREAK キーを押しても、止まった行番号と、実際出ていた音とは一致しません。

≪演奏おしまい …… PLEN (プレイエンド)≫

プログラムの中で PLAY 文を使うと、 音楽を演奏しながらプログラムを実行します。
しかし、演奏とそれ以外のプログラムを、別々に実行したい場合もありますね。そんなときはこの PLEN 文を使います。

IF PLEN(n)=−1 THEN ××× 又は IF PLEN(n) THEN ×××	n チャンネル (n=0 の時は全チャンネル) の PLAY が終了したら XXX へジャンプ
IF PLEN(n)=0 THEN ××× 又は IF NOT PLEN(n) THEN ×××	n チャンネル (n=0 の時は全チャンネルのどれか 1 つでも) が PLAY 中なら XXX へジャンプ

note ! PLAY 文の文字列の中で " : " で区切られた部分をチャンネルといいます。
左から " 1 チャンネル : 2 チャンネル : 3 チャンネル " です。

(例1)

```
10  PLAY "CDEF:EGEG:+CBAG"
20  PLAY "CDEF:EGEG:+CBAG"
30  PLAY "CDEF:EGED"
40  PLAY "CDEF:EGED"
45  IF PLEN(3)=-1 THEN 50
49  GOTO 45
50  SCREEN 2:CLS
60  FOR R=10 TO 80 STEP 4
70  CIRCLE(128,96),R,15
80  NEXT R
```

(例2)

```
10  PLAY "CDEF:EGEG:+CBAG"
20  PLAY "CDEF:EGEG:+CBAG"
30  PLAY "CDEF:EGED"
40  PLAY "CDEF:EGED"
45  IF PLEN(3) = 0 THEN 45
50  SCREEN 2:CLS
60  FOR R=10 TO 80 STEP 4
70  CIRCLE(128,96),R,15
80  NEXT R
```

どちらのプログラムも、20行の演奏が終ると円を描きます。

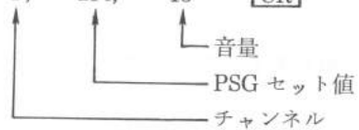
★★★ SOUND（サウンド） ★★★

次は SOUND 文です。

ゲームに使う効果音や擬音が出せます。ドレミ …… と音楽も奏でます。

同じ ドレミ …… を鳴らすにも、PLAY 文と SOUND 文は違います。PLAY 文では、音高、音の長さなど、いろいろな要素で音を指定しましたね。でも、SOUND 文では、**チャンネル、PSGセット値（音の高さ）、音量（音の大きさ）**の3要素から音を出します。

(例) SOUND 1, 214, 15 CR



ドの音がでます。

では、それぞれについて説明しましょう。

チャンネル

チャンネルは、0～5のいずれかに指定します。

0：音を消します。

1：
2：
3：
} 音階表にある音と、それ以外の音を出します。PSG セット値（1～1023までの値）で音が決まります。音階は次表を参照してください。

4：ホワイトノイズ（ザーザー）

5：同期ノイズ（ブー）

ノイズといっても雑音ではないのです。

効果音として使います。

1，2，3の3つのチャンネルは、ひとつのチャンネルからひとつの音がでます。ですから、三重和音まで可能なわけです。

例

```
10 SOUND 1，214，15
20 SOUND 2，170，15
30 SOUND 3，143，15
40 FOR W=0 TO 1000 :NEXT W      ドミソの和音です。
50 SOUND 0
```

PSG セット値（音の高さ）

チャンネル1～3のとき　セット値を入れます。

音階の表

Oct.	音　高	PSGセット値	理論周波数 Hz	実際の周波数
Q1	C	855	130.8	130.8
	C#，D$	807	138.6	138.6
	D	762	146.8	146.8
	D#，E$	719	155.6	155.6

Oct.	音　高	PSGセット値	理論周波数 Hz	実際の周波数
Q1	E	679	164.8	164.7
	F	641	174.6	174.5
	F#，G$	605	185.0	184.9
	G	571	196.0	195.9
	G#，A$	539	207.7	207.5
	A	508	220.0	220.2
	A#，B$	480	233.1	233.0
	B	453	246.9	246.9
Q2	C	428	261.6	261.4
	C#，D$	404	277.2	276.9
	D	381	293.7	293.6
	D#，E$	360	311.1	310.7
	E	339	329.6	330.0
	F	320	349.2	349.6
	F#，G$	302	370.0	370.4
	G	285	392.0	392.5
	G#，A$	269	415.3	415.8
	A	254	440.0	440.4
	A#，B$	240	466.2	466.1
	B	226	493.9	495.0

Oct.	音　高	PSGセット値	理論周波数 Hz	実際の周波数
Q3	C	214	523.3	522.7
	C# , D$	202	554.4	553.8
	D	190	587.3	588.7
	D# , E$	180	622.3	621.4
	E	170	659.3	658.0
	F	160	698.5	699.1
	F# , G$	151	740.0	740.8
	G	143	784.0	782.2
	G# , A$	135	830.6	828.6
	A	127	880.0	880.8
	A# , B$	120	932.3	932.2
	B	113	987.8	989.9
Q4	C	107	1046.5	1045.4
	C# , D$	101	1108.7	1107.5
	D	95	1174.7	1177.5
	D# , E$	90	1244.5	1242.9
	E	85	1318.5	1316.0
	F	80	1396.9	1398.3
	F# , G$	76	1480.0	1471.9
	G	71	1568.0	1575.5

Oct.	音　高	PSGセット値	理論周波数 Hz	実際の周波数
Q4	G# , A$	67	1661.2	1669.6
	A	64	1760.0	1747.8
	A# , B$	60	1864.7	1864.4
	B	57	1975.5	1962.5
Q5	C	53	2093.0	2110.6
	C# , D$	50	2217.5	2237.2
	D	48	2349.3	2330.4
	D# , E$	45	2489.0	2485.8
	E	42	2637.0	2663.4
	F	40	2793.8	2796.5
	F# , G$	38	2960.0	2943.7
	G	36	3136.0	3107.2
	G# , A$	34	3322.4	3290.0
	A	32	3520.0	3495.7
	A# , B$	30	3729.3	3728.7
	B	28	3951.1	3995.0

↓
等分平均律による音高

note 1 ! PLAY文ではドレミ …… やCDE …… で入力しますが、SOUND文ではこの表と照らしながら入力しなくてはなりません。

note 2 ! 音階表にあるPSGセット値以外の数値を入力すると、ドとレの中間の音など、ドレミファ …… 以外の音が出ます。
(PSGセット値は周波数をもとにした値で、1～1023の範囲内の整数です。

```
100 FOR N=1023 TO 1 STEP-1
110 SOUND 1,N,15
120 PRINT N
130 NEXT N
140 SOUND 0
```

1～1023のすべての音が出ます。)

チャンネルが4または5のとき ― 0,1,2,3のいずれかに指定します。

セット値0：
1：
2：　} 三段階の、すでに決定している音の高さになります。　512/N　1024/N　2048/N

セット値3：チャンネル3で指定する音の高さになります。

(例)
```
300 SOUND 3, 193, 1 : SOUND 4,3,15
335 FOR W=0 TO 500:NEXT W
350 SOUND 0
```


音　量

0～15の数字で指定します。

0：音を消す
1：最小の音量
〜
15：最大の音量

note ! SOUND文を使ったら必らずSOUND 0で音を消します。

サンプルプログラムを実行してみましょう。ドレミファソ …… と音が出ますよ。

サンプルプログラム (1)

```
10 FOR N=1 TO 36
20 READ A
30 SOUND 1,A ,15
35 FOR W =0 TO 200 :NEXT W
40 NEXT N
50 SOUND 0
100 DATA 855,762,679,641,571,508,453
110 DATA 428,381,339,320,285,254,226
120 DATA 214,190,170,160,143,127,113
130 DATA 107,95,85,80,71,64,57
140 DATA 53,48,42,40,36,32,28,26
```

次は、雑音…

サンプルプログラム (2)

```
200 FOR N=0 TO 2
210 SOUND 4,N,15
235 FOR W=0TO 500:NEXT W
240 NEXT N
250 SOUND 0
```


★★★ BEEP ★★★

PLAY 文、SOUND 文の他に、BEEP があります。短い音を出します。

BEEP　　ピッと音が鳴る。
BEEP 0　BEEP 音をとめる。
BEEP 1　ピーと鳴り続ける。
BEEP 2　ピポピポと鳴る。

ゲームの効果音などに使いましょう。

(例)

```
10 DIM A$(12)
20 FOR N=0 TO 12
30 READ A$(N)
40 PRINT A$(N);
50 BEEP
60 NEXT N
70 DATA H,O,M,E," ",C,O,M,P,U,T,E,R
RUN
HOME COMPUTER

Ready
```

グラフィックス

むずかしい技術はいりません。グラフィックスに使う命令をいくつか覚えさえすればいいのです。
さあ、トライしましょう。

SCREEN

★★★ 画面のこと ★★★

画面にはテキスト画面とグラフィック画面の二種類あります。オープニングからベーシックに入った時の画面は、テキスト画面といい、プログラムを書くのに使います。(前章まではずっと、テキスト画面を使っていたのです。)

これからグラフィックをやろうという皆さんは、グラフィック画面という専用の画面に切り替えなくてはなりません。というのは、グラフィック専用の命令文は、それ専用の画面にしか使えないからなのです。

では、どのようにして画面を選択し指定すればよいのでしょうか。

★★★ SCREEN ★★★

SCREEN 文は次のように使います。

テキスト画面を指定するとき

SCREEN 1

グラフィック画面を指定するとき

SCREEN 2,1 (画面を切り替えたとき絵がメモリの方に一度移ります。ですからSHIFT + BREAK で画面を何度切替えても絵はもどります。)

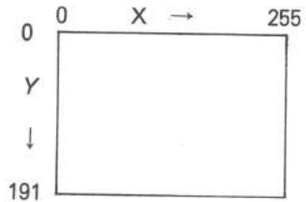
SCREEN 2, 0 (画面を切り替えたとき、絵は消えたままです。)

note1 ! SCREEN 2, 1 の 1 は省略可能です。SCREEN 2 と入力してもかまいません。しかし、SCREEN 2, 0 を指定した後は、SCREEN 2, 1 と、省略せずに入力して下さい。

note2 ! テキスト画面からグラフィック画面へ切り替えるときには [SHIFT] キーと [グラフ/BREAK] キーを、グラフィック画面からテキスト画面へ切り替えるときには [グラフ/BREAK] キーのみ押してください。

LINE

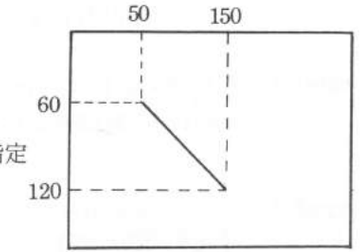
線を引きます。
グラフィック画面は、



このように、横（X方向）に0から255、縦（y方向）に0から191の座標を持っています。線を引くときは、線のはじめとおわりを座標で指定します。

例①

```
10  SCREEN2:CLS
20  LINE(50,60)-(150,120),8
          ↑        ↑       ↑
      線のはじまり  線のおわり  色の指定
      (x1 , y1 )   (x2 , y2 )
30  GOTO 30
```

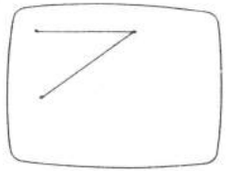


note1 !　色の指定については、COLOR の項を参照してください。

note2 !　30行の GOTO 30 は、画面が消えないようにしているのです。「無限ループをつくる」といいます。

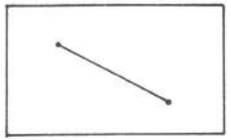
note3 !　書き始めの座標を省略しますと前に描いた座標から線を引きます。(次のプログラムの 30行参照)


```
10  SCREEN2:CLS
20  LINE(50,50)-(150,50),8
30  LINE-(50,150),8
```



線を描き終わると、テキスト画面にもどってしまいます。グラフィック画面にするには、SHIFT キーを押しながらBREAKキーを打ちます。グラフィック画面からテキスト画面にもどるには、BREAK キーを打ちます。2秒ぐらいでテキスト画面にもどります。

もう一度**例**①のプログラムをRUNしてください。
ななめの線が引けましたね。



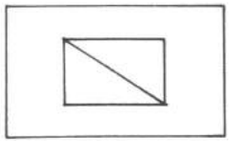
このプログラムに、少し手を加えましょう。**例**②のように、20行を変えてください。

例②

```
10  SCREEN2
20  LINE(50,60)-(150,120),8, B
30  GOTO 30
```

└ , Bを加えました。

実行すると、



こんなふうに、

LINE 文で引いた線を対角線にした BOX が描けました。このように BOX (箱)を描くときには、LINE 文で対角線を指定して、色指定の後に B をつけます。

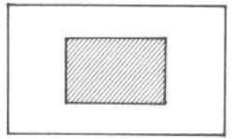
きらに手を加えましょう。**例**③のように、20行を変えてください。

例③

```
10  SCREEN 2
20  LINE(50,60)-(150,120), 8, BF
30  GOTO 30
```

└ 箱の中を塗ります。

実行すると、



こんなふうに

箱に色がつきました。

note1 ! : BF は Box Fill を意味します。
note2 ! : LINE 文を使うときは、必ず、SCREEN でグラフィック画面にしてください。

LINE 文で、変数を使うと面白い使い方ができます。

LINE 文の応用

```
10 SCREEN 2:CLS
20 Y=0:C=2 ..............................  Cは色です。ミドリから始めます。
```

```
30 FOR X=8 TO 240 STEP 16
40 LINE(X,Y)-(X+16,Y+16),C,BF
50 C=C+1 ..... 色を変えます。
60 IF C>15 THEN C=2 ..... 色をミドリにもどします。
70 NEXT X
80 Y=Y+16 ..... 一段下に四角を描きます。(16を24に変えると?)
90 IF Y>160 THEN END ... 一番下で終わりにします。
100 GOTO 30
```

} 四角の大きさです。(16を24に変えると?)

CIRCLE

箱を描いたら、こんどは円です。
次のように入力します。

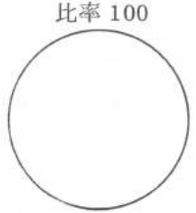
(例)

```
                (1)      (2) (3)  (4)  (5)  (6)  (7)
                座標     半径 色  比率 始点 終点
CIRCLE (125,95),50,5,100,0,100,BF
```

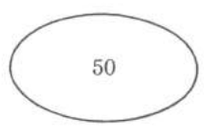
例を詳しくみていきましょう。

(1) 座標 ..... 円の中心点です。
(2) 半径 ..... 文字通り、半径です。
(3) 色 ..... 色の指定。COLOR の項を参照してください。

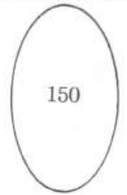
(4) 比率 ..... 真円、よこ長のだ円、たて長のだ円のいずれかに指定します。



真 円

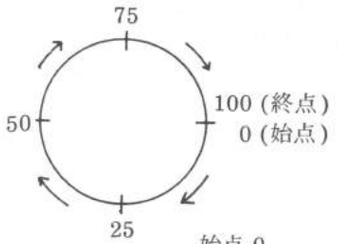


よこ長のだ円

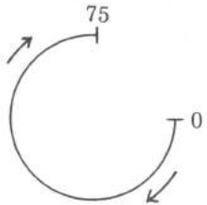


たて長のだ円

(5) 始点 ..... 書き始めの位置を示します。
(6) 終点 ..... 書き終わりの位置を示します。

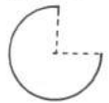


始点 0
終点 100 の場合



始点 0
終点 75 の場合

(7) BF を指定しないとき ..... 円周のみ描きます。



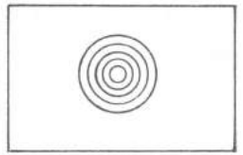
BF を指定すると …… (3)で指定した色でぬりつぶします。

B を指定すると …… 円周だけでなく、始点と中心、終点と中心をむすぶ線も描きます。

CIRCLE 文を使ったプログラムを、ひとつあげておきます。ためしてみてください。

```
10 SCREEN 2,1:CLS
20 FOR R=10 TO 50 STEP 10 (半径が 10 ずつ増えます。)
30 CIRCLE(125,95),R,8,100,0,100
40 NEXT R
50 GOTO 50
```

どうですか。



こんなふうに描けましたか？

note1 ! CIRCLE 文で描いた円を消すときは、下地と同じ色で円を描きます。

note2 ! CIRCLE 文を使うときは、必ずグラフィック画面にしてください。

PSET

点を打ちましょう。

一般に、

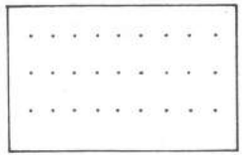
```
          座標     色
PSET(x,y),4
```

この形で入力します。

次のプログラム、ためしてください。

```
10 SCREEN 2,1:CLS
20 FOR X=0 TO 240 STEP 20
30 FOR Y=0 TO 190 STEP 20
40 PSET(X,Y)
50 NEXT Y
60 NEXT X
70 GOTO 70
```

こんなふうに、



テンがいっぱい並びましたか？

note ! ：PSET 文を使うときは、必ずグラフィック画面にしてください。

PRINT

文字を表示します。使い方はテキスト画面のときと同じです。

CURSOR

グラフィック画面で、文字の表示位置を指定するときも、CURSOR 文を使います。
グラフィック画面も座標をもっていますから（LINE の項目の図参照）

```
CURSOR x , y
```

のように、座標で表示したい位置を指定してください。

note ! ：グラフィック画面の座標の数値はテキスト画面の座標の数値と違いますから注意してください。

COLOR と PAINT

コンピュータ・グラフィックスで、

線が引けました。
箱（四角）も円も描けました。
点もいっぱい打てました。

さて、次は色ぬりです。

★★★ COLOR ★★★

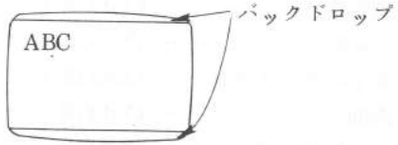
≪テキスト画面≫

テキスト画面で

```
10 COLOR 1,7,15
20 PRINT "ABC"
```

と入力し、実行してください。

ABC という文字は黒、画面は水色、画面の周囲（バックドロップ。画面を切り替えても変わらない部分）は白になったでしょう。



10 行をみてください。

```
        ①  ②  ③
COLOR 1 , 7 , 15
```

① は文字の色、② は画面の色、③ はバックドロップの色の指定です。それぞれの数字が、色をあらわしています。(下の表を参照)

色番号	色	色番号	色	色番号	色
0	透　明	6	こい赤	12	こい緑
1	黒	7	水色(シアン)	13	マゼンダ
2	緑	8	赤	14	灰
3	うすい緑	9	うすい赤	15	白
4	こい青	10	こい黄		
5	うすい青	11	うすい黄		

note1 ! ：0 (透明) に指定すると、バックドロップの色がでます。

note2 ! ：COLOR 文の一部を省略することも可能です。

文字の色指定省略 …… COLOR , 7 , 15
画面　〃　…… COLOR 1, , 15
バックドロップ 〃 …… COLOR 1, 7
文字の色だけ指定 …… COLOR 1
画面　〃　…… COLOR , 7
バックドロップ 〃 …… COLOR , , 15

≪グラフィック画面≫

グラフィック画面で色を変える方法は、テキスト画面と少し違います。テキスト画面では

```
COLOR ①,②,③
```

と入力して、文字の色や地の色を指定しましたが、グラフィック画面では次のようにします。

グラフィック画面の**文字の色**を変えたいときは、PRINT 文の前に COLOR 文で色を指定します。

```
10 SCREEN2:CLS
20 COLOR 6
30 PRINT "ABC"
```

PRINT 文の前に COLOR 文を置くので、ひとつひとつの文字の色を変えることも可能なわけです。たとえば、

```
10 SCREEN2:CLS
20 COLOR 5
30 PRINT "A";
40 COLOR 15
50 PRINT "B";
60 COLOR 2
70 PRINT "C";
```

このようにすると、A は青、B は白、C は緑で、ABC と表示されます。部分的に文字の色を変え

ることは、テキスト画面ではできません。そこが大きな違いです。

地の色を変えるには、CLS の前に、COLOR 文を使います。

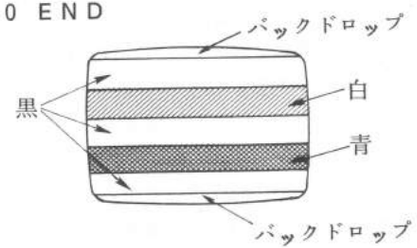
```
10 SCREEN2:COLOR , 3:CLS
```

とすると、COLOR 文で指定した色（3＝緑）で画面をクリアして（CLS 文）、画面は緑色になります。

note ! ：グラフィック画面の地の色を変えるときは CLS を忘れずに。

地の色は、LINE 文の BF でも変わります。
LINE 文を使うと、（LINE 文の項参照）

```
10 SCREEN2:CLS
20 LINE(0 , 48)-(255 , 90) , 15 , BF
30 LINE(0 , 120)-(255 , 160) , 5 , BF
40 END
```



こんなふうに画面の一部の色を変えることができます。

部分的に変えることができるかどうか、それがテキスト画面とグラフィック画面の違いのひとつですね。

次のプログラムをためしてみましょう。

```
10 SCREEN 2:CLS
20 FOR C=2 TO 15
30 COLOR C
40 PRINT " ■ ";
50 NEXT C
```

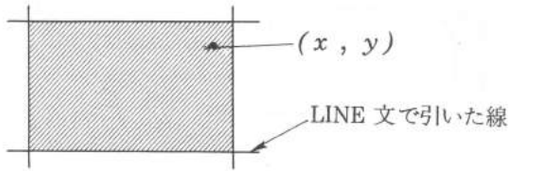
note ! ：GRAPH キーを押したあと V のキーをおすと ■ が表示されます。

■ が 1 個ずつ色が変わっていくでしょう。

こんなことができるのも、グラフィック画面ならではです。

★★★ PAINT ★★★

PAINT は、



図のような、LINE 文や CIRCLE 文などで引いた線にかこまれたところを塗ります。一般に、次のように入力します。

```
PAINT(x, y), 色
```

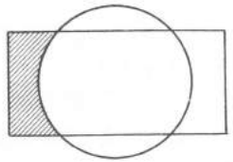
斜線の部分（塗りたい部分）にある点のうち、ひとつの座標を指定。

その際、PAINT 文で指定する色と、LINE 文，CIRCLE 文などで引いた線の色は同じでなくてはなりません。違う色が交差していると、そこから色がもれてしまいます。また、ほんのわずかでも、すき間があれば、そこから色がもれてしまいます。注意しましょう。

それでは、このプログラム、ためしてください。

```
10 SCREEN 2:CLS
20 LINE(36, 48)-(200, 144), 5, B ····· 四角を描いて
30 CIRCLE(128, 96), 60,5 ····· 丸を描きます
40 PAINT(40, 50), 5 ····· 色を塗ると
50 GOTO 50
```

こんなふうに



なったでしょ。

SPRITE，PATTERN，MAG

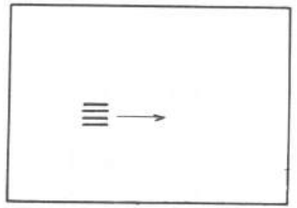
★★★ SPRITE と PATTERN ★★★

自分がつくったキャラクタが、画面の中を動きまわったなら・・・。そんな願いをかなえてくれるのがスプライト機能。グラフィック画面上にキャラクタをつくり、それを動かす働きです。PATTERN 文でキャラクタをつくります。SPRITE 文でキャラクタを動かします。

具体的な説明の前に、まずはプログラムを入力し、実行してください。

```
10 SCREEN 2:CLS
20 X=0 : Y=90
30 PATTERN S#160,"FF00FF00FF00FF00"
40 SPRITE 0 ,(X, Y), 160, 8
50 X=X+1
60 IF X=255 THEN X=0
70 GOTO 40
RUN
```

すると、



こんなふうに

四本の線のキャラクタが画面をよこぎっていきますね。
どうして、こうなるのでしょうか。

≪PATTERN 文 ― キャラクタをつくる ―≫

プログラムの 30 行を見てください。

```
PATTERN S# 160 , "FF00FF00FF00FF00"
           ①   ②          ③
```

① PATTERN 文は、グラフィック画面、テキスト画面のどちらの画面でも使うことができ、前者の場合は S#, 後者の場合は C# と入力します。このプログラムでは S# 。S# と指定すると、PATTERN は SPRITE 文で使うことができます。

② の番号は、スプライト名称といいます。つくったキャラクタにつける番号です。キャラクタは全部で 256 個つくることができます。

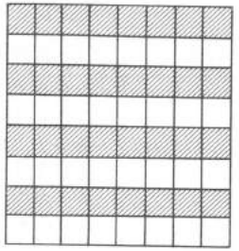
note1 ! ：番号は、16 進数でもかまいません。

例 PATTERN C#&H30

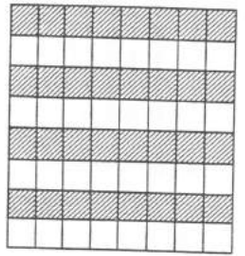
note2 ! ：ベーシックで使うパターン番号と、「パターンのへんこう」のパターンていぎの番号とは関連があります。(P. 196 参照)

③ は何でしょう？ アルファベット文字が混じっていますが、実は、16 進数であらわした、れっきとした数字なのです。

キャラクタを作るときは、「パターンへんこう」の画面で、8×8 のマス目で描きます。このマス目ひとつ分を 1 ドットと呼びます。ドットをぬりつぶして、キャラクタを描いていきます。



それを左右 4 ドットずつに分け、▨の部分を 1、□の部分を 0 として 2 進数の（0 と 1 の 2 種類だけの）数字に直します。



左	右
1 1 1 1	1 1 1 1
0 0 0 0	0 0 0 0
1 1 1 1	1 1 1 1
0 0 0 0	0 0 0 0
1 1 1 1	1 1 1 1
0 0 0 0	0 0 0 0
1 1 1 1	1 1 1 1
0 0 0 0	0 0 0 0

さらに、それを 16 進数に変え、(表を参考にしてください。)

変 換 表

左	右		左	右
1 1 1 1	1 1 1 1		F	F
0 0 0 0	0 0 0 0		0	0
1 1 1 1	1 1 1 1		F	F
0 0 0 0	0 0 0 0	⇨	0	0
1 1 1 1	1 1 1 1		F	F
0 0 0 0	0 0 0 0		0	0
1 1 1 1	1 1 1 1		F	F
0 0 0 0	0 0 0 0		0	0

10進数	2進数	16進数
0	0 0 0 0	0
1	0 0 0 1	1
2	0 0 1 0(桁上り)	2
3	0 0 1 1	3
4	0 1 0 0	4
5	0 1 0 1	5
6	0 1 1 0	6
7	0 1 1 1	7
8	1 0 0 0	8
9	1 0 0 1	9
10(桁上り)	1 0 1 0	A
11	1 0 1 1	B
12	1 1 0 0	C
13	1 1 0 1	D
14	1 1 1 0	E
15	1 1 1 1	F
16	1 0 0 0 0	10(桁上り)

左右左右・・・と横に並べたものが③の "FF00FF00FF00FF00" なのです。「パターンのへんこう」画面で描き、16進数に変えたデータ(FF00・・・)を、PATTERN S# 160, " [redacted] " に入れて実行します。どのようなキャラクタも、このような手順でつくります。その際、16進数であらわした文字列の両側に " "(ダブルクォーテーション)と、文字列の前に ,(コンマ)を必ずつけてください。

≪SPRITE 文 — キャラクタを動かす —≫

先のプログラムの40行を見てください。

```
40 SPRITE 0,(X,Y),160,8
          ①  ②    ③  ④
```

①面番号　②座標　③スプライト名称　④色

③ はスプライト名称、動かしたいキャラクタです。このプログラムの場合、30行の PATTERN 文でつくったキャラクタ(160番)を呼びだしています。

④ で、そのキャラクタの色を決めます。この場合は8(赤)です。

PATTERN でつくったキャラクタを呼び出して、色をつけました。動かすのは ② の座標指定です。指定のしかたによって、キャラクタはあちらこちらへ動きます。

キャラクタを右に動かす	X = X + 1
〃 左に動かす	X = X − 1
〃 下に動かす	Y = Y + 1
〃 上に動かす	Y = Y − 1

このプログラムの場合は、X = 0, Y = 90(20行), X = X + 1(50行)と入力しているので、四本線のキャラクタは左端(X = 0)から右へと動いていくのです。

note ! : スプライトが画面の端まで行ったらどうなるでしょう。右に動いていった場合はエラーにならず反対側から出て来ますが、それ以外の方向の画面の端では

エラーになります。
そのため、スプライトが画面の端で止まるように制限をつけます。「リミットをつける」と言います。
60行のIF文が、右側のリミットです。

左側のリミット

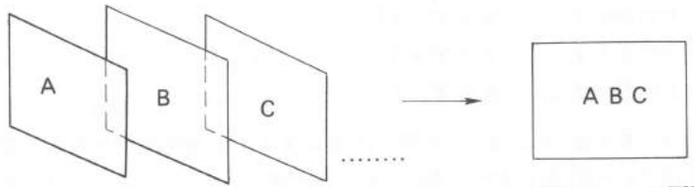
```
IF X=0 THEN X=255      もとにもどります。
```

または、

```
IF X=0 THEN X=X+1      同じ所で止まっています。
```

上や下のリミットも同様です。Xの代わりにYをあてはめて考えてください。

さて、①の面番号について少し説明します。
スプライト面は32枚あり、0,1,2,3,..... 31と番号がついています。

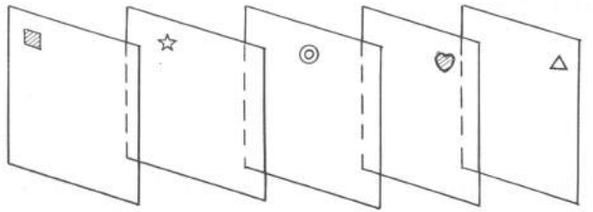


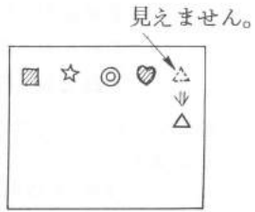
キャラクタを描くときには、スプライト面32枚のうちのどれか1枚を選びます。何番の面に描くか指定するのが①の面番号です。

スプライトには優先順位があり、番号の小さい面に描かれたキャラクタほど、優先順位が高いのです。いくつものキャラクタが画面上を動いていると、キャラクタがかさなりあう場合もでてきますね。そのときに、優先順位の低い方のキャラクタは、優先順位の高い方のキャラクタにかくれてしまうのです。交差した場合にどちらが前にきたらよいか、ということを考えながら、スプライト面番号の指定をすることが大切でしょう。

note ! : スプライトの制限

キャラクタは、水平線上に4つまで並べられます。もし、5つ、水平線上にキャラクタを並べたら、面番号の大きい(優先順位の低い)キャラクタは見えなくなります。





下へずらすと見えるようになります。

≪SCOIN≫

スプライトを使ってゲームを作る場合、スプライトがぶつかったかどうか判定できなければなりません。この判定は次のようにします。

SCOIN ⟹(RUN) −1　衝突した。
　　0　衝突しない。

```
IF SCOIN THEN ～
```

又は、

```
IF SCOIN=−1 THEN ～
```

THEN の後には、プログラムに応じた命令をつけます。

```
THEN BEEP
THEN GOSUB 行番号
THEN PLAY                    など
```

面白いアイデアを考えましょう。

サンプルプログラム

```
10 SCREE2:CLS
20 PATTERN S#160,"FF00FF00FF00FF00"
30 PATTERN S#164,"183C7EFFFF7E3C18"
50 X=0:XX=255:Y=100:E=1
60 SPRITE 10,(X,Y),160,8
70 SPRITE 11,(XX,Y),164,5
80 X=X+E:XX=XX−E
90 IF SCOIN=−1 THEN GOTO 200
100 GOTO 60
200 BEEP:BEEP
210 CLS
220 GOTO 50
```


≪BASIC と「パターンのへんこう」≫

BASIC のパターン番号と「パターンのへんこう」のパターンていぎの番号は関連があるということは、すでに触れました。関連があるとはどういうことなのでしょう。

付録の「BASIC の S# PATTERN とパターンていぎ # との関係」を見てください。

BASIC	パターンていぎ
S# 0	0
1	
2	
3	
4	1
5	

これは、図に書くと、

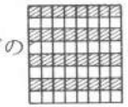
BASIC S# 0	BASIC S# 2
BASIC S# 1	BASIC S# 3

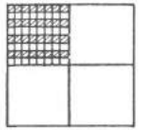
パターンていぎ 0

BASIC S# 4	BASIC S# 6
BASIC S# 5	BASIC S# 7

パターンていぎ 1

のような関係になります。BASIC の PATTERN 文でキャラクタをつくって RUN した場合、BASIC のS#0はパターンていぎ 0 の左上の部分に、BASIC の S#1はパターンていぎ 0の左下の部分に影響を与えます。

たとえば、さきほどの  は、S#160 ですから、RUN した場合、パターンていぎの 40の左上の部分に影響を与えます。ためしに、「パターンへんこう」画面にして、40 を出してご覧なさい。



このように、左上の部分に S#160 がはいっていることがわかるでしょう。

パターンていぎ 0～39 (BASIC のパターン番号 0～159) は、シュートゲームのキャラクタとして使われています。ですから、BASIC では、S#160 からあとの番号を使った方がよいでしょう。(シュートゲームのキャラクタの形がかわってしまいますから。) もし、シュートゲームのキャラクタを変更して保存しておくのでしたら、BASIC で S # の 00 ～ 159 にパターンを描いてテープに取っておきましょう。

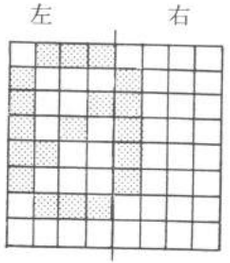
≪テキスト画面の PATTERN 文 (C# 指定の場合)≫

ふつうは、A のキーを押せば A という文字が、!のキーを押せば ! という記号が画面に表示されます。でも、キーを押したとき、別の文字や記号が表示されるようにすることだって、PATTERNを使えばできるのです。次のプログラムを入力してみてください。

```
1000 PATTERN C# 65,"708898A8C8887000"
```

RUN して、A のキーを押すと ..... 。A ではなく 0 が画面に表示されました。

"708898A8C8887000" は、16 進数で、0 の形をあらわしています。



左	右
0 1 1 1	0 0 0 0
1 0 0 0	1 0 0 0
1 0 0 1	1 0 0 0
1 0 1 0	1 0 0 0
1 1 0 0	1 0 0 0
1 0 0 0	1 0 0 0
0 1 1 1	0 0 0 0
0 0 0 0	0 0 0 0

2 進数

左	右
7	0
8	8
9	8
A	8
C	8
8	8
7	0
0	0

16 進数

黒いマス＝1， 白いマス＝0

65は、Aのキャラクターコードで、“指定席”のようなもので「65番はA」と決まっています。ところが、PATTERN文を使って、「65番は0」と変更してしまったのでAのキーを押しても0が表示されるのです。

一般に、テキスト画面のPATTERN文は、

PATTERN C# キャラクタコード,"文字列式"

の形をとります。

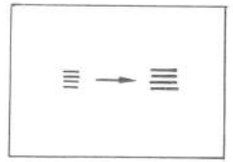
キャラクタコードは、付録を参考にしてください。

★★★ MAG ★★★

MAGはSPRITEで動かすキャラクタの大きさを変える命令です。P. 144のプログラムに、MAG2という1行をつけ加えてみましょう。　40 MAG 2

（例）
```
10 SCREEN 2:CLS
20 X=120: Y=90
30 MAG 2
40 PATTERN S#160,"FF00FF00FF00FF00"
50 SPRITE 8,(X, Y), 160, 3
```


すると、

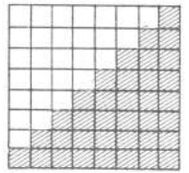


こんなふうに

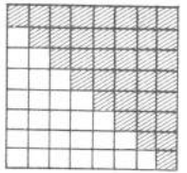
タテとヨコが2倍、面積が4倍の
大きさになります。

MAGには、MAG0, MAG1, MAG2, MAG3がありますが、それぞれは、どのような働きをするのでしょうか。

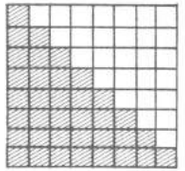
PATTERN文を使って、S#160, S#161, S#162, S#163の4つのパターンをつくったとします。



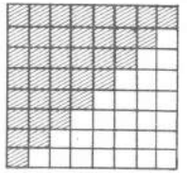
S#160



S#161



S#162



S#163

そして、

たとえば

```
SPRITE 2,(X,Y),160,8
```

というように、SPRITE 命令をするとき、

① MAG0 に指定すると

PATTERN 文でつくった S#160 が、そのまま (8 ドット×8 ドットの大きさ)であらわれます。

② MAG1 に指定すると

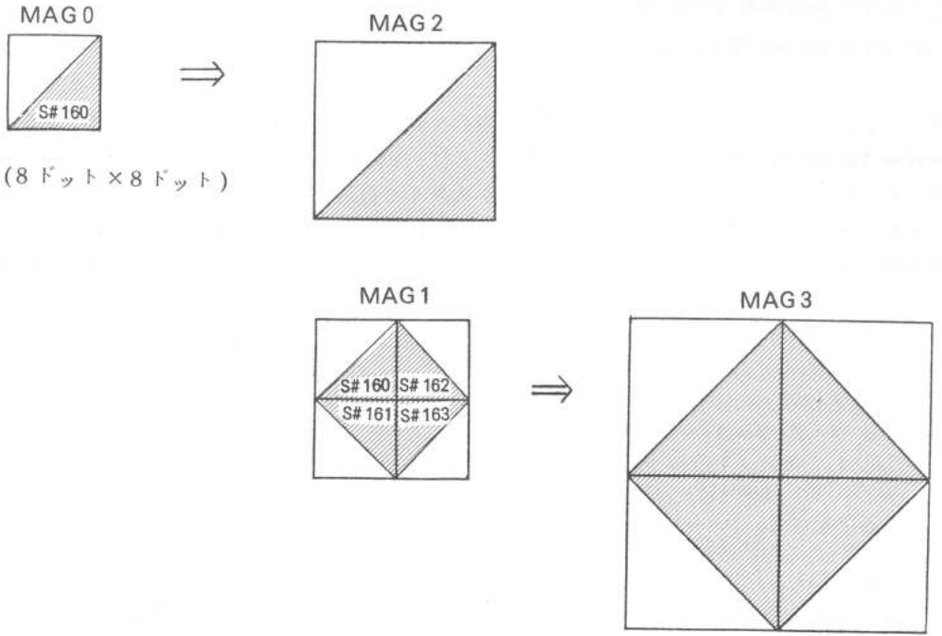
S#160, S#161, S#162, S#163 の 4 つのパターンを組み合わせた形であらわれます。

③ MAG2 に指定すると

S#160 が、MAG0 の 2 倍の大きさであらわれます。

④ MAG3 に指定すると

S#160～163 を組みあわせた形が、MAG1 の 2 倍の大きさであらわれます。



― プログラムを消却するには ―

（オーバーフローエラーになったとき）

グラフィックスの長いプログラムを実行した場合、実行後、またはBREAKキーを押したとき、**オーバーフローエラー**になることがあります。これは、VRAMのデータがメインメモリーに退避しようとしても、メインメモリー内のプログラムが大きくて退避しきれないためです。このような場合は、メインメモリーのプログラムを消却して、VRAMのデータを退避させます。

プログラムが長すぎて、オーバーフローエラーが出た場合は、以下のプログラムの100行以下を、そのプログラムに追加してください。

```
10 LIMIT &HBF00
20 SCREEN 2
30 FOR N=0 TO 5
40 X=0:Y=0:X1=255:Y1=191
50 LINE (X+N*10,Y+N*10)-(X1-N
*10,Y1-N*10),RND(14)+2,B
60 NEXT
100 DATA AF,2A,02,86,77,23,22
,04,86,77,23,22,06,86,77,23
102 DATA 22,08,86,23,22,0A,86
,C3,9F,05
104 C=&HBF00
110 FOR M=1 TO 26
```

グラフィックスのサンプルプログラムです。

プログラムを消却するプログラムです。


```
120 READ A$:A=VAL("&H"+A$)
130 POKE C,A:C=C+1
140 NEXT
150 CALL &HBF00
160 END
```

実行すると、グラフィックスを描いた後、CALL文のある行でBREAK（プログラムが止まること）します。そうしたら、SHIFT＋BREAKでグラフィックに切換えて絵を確認しましょう。（絵はVRAMに転送されます。） 確認した後、BREAKキーでテキスト画面にもどし、**SAVEV**"ファイル名"でカセットテープにグラフィックスをセーブしてください。

100～170行のプログラムは、実行後、メモリ内のプログラムをすべて消してしまいます。ですから、実行する前に、かならずメモリ内のプログラムをテープにセーブしておいてください。

SAVEV , VERIFYV , LOADV

テキスト画面のプログラムを保存することができるように、グラフィック画面に描いたものやスプライトパターンも、やはりカセットテープに保存できます。手順は、SAVE, LOADと同じです。

★★★ SAVEV ★★★

グラフィック面、及びスプライトパターンをカセットテープへ保存します。

ＳＡＶＥＶ　＂ファイル名＂

のように入力します。

★★★ VERIFYV ★★★

カセットテープに、正確に SAVE されたか確かめます。

ＶＥＲＩＦＹＶ　＂ファイル名＂

のように入力します。

★★★ LOADV ★★★

カセットテープに保存されているグラフィック面、及びスプライトパターンを、コンピュータのメモリの中へ取り込みます。

ＬＯＡＤＶ　＂ファイル名＂

のように入力します。

ＬＯＡＤＶ　（ファイル名指定無し）

の場合は、最初のファイルをロードします。

SAVE, VERIFY, LOAD の項 (P. 52) も参照してください。

LOADV で Loding end が画面に出たら、SHIFT + BREAK でグラフィック画面にします。絵がすぐ出てきます。

INKEY $ と RND

ゲーム作りに不可欠な2つの命令を紹介します。

まずは INKEY $。

★★★ INKEY $（インキーダラー）－すばやい判断－★★★

キーボードが押されたかどうかをすぐに判断します。

たとえば

```
10 PRINT INKEY$:GOTO 10
```

を RUN して、キーをどれでもいいから押してください。押したとたんに字が表示されます。これは、INKEY $ を、GOTO 10 を使って無限ループ（グルグル回りしているプログラム）の中に入れたためです。このようにすると、INKEY $ は、いつキーが押されるか、常に見張っています。そして、キーが押されたら、すぐにその文字を教えてくれるのです。ただし、INKEY $ で受けつけるのは、英数モードのノーマルとシフト、それにコントロールキー＋英字です。

次のプログラムを見てください。

```
5 REM キー コントロール
10 CLS
20 X=15:Y=10:E=1
100 CURSOR X,Y:PRINT"A"
105 Z$=INKEY$
110 IF Z$=CHR$(28)THEN E=1:GOSUB 200
120 IF Z$=CHR$(29)THEN E=-1 :GOSUB 200
```



```
130 IF Z$=CHR$(30)THEN E=-1:GOSUB 300
140 IF Z$=CHR$(31)THEN E=1:GOSUB 300
170 GOTO 100
200 CURSORX,Y:PRINT" "
210 X=X+E:IFX<=0 OR X>28 THEN X=X-E
220 RETURN
300 CURSOR X,Y:PRINT" "
310 Y=Y+E:IF Y<=0 OR Y>=22 THEN Y=Y-E
320 RETURN
```

105行に INKEY $ が使われています。CHR $(28), CHR $(29), CHR $(30), CHRS (31) はコントロールコード（「文字関数と関数」の CHR $ の項、或いは付録の表を参照してください。）で、それぞれ [→] [←] [↑] [↓] の、4つのカーソルキーです。

[→] のキーを押すと、INKEY $ はすばやく判断して、200行へとびます。つまりA の文字は右へ動きます。[↑] を押すと、またすぐに判断して 300行へとび、A の文字は下へ動きます。A の文字は、カーソルキーによって、上下左右へ動くわけです。

これを応用すれば、ゲームの中で、キャラクタを、キーコントロールで動かすことができるのです。

ゲームに便利なもうひとつの命令は、RND です。

★★★ RND（ランダム）－ 意外性 － ★★★

ランダムは乱数のサイコロを振ったとき、何が出るかわからないような、規則のない数のことです。

```
10 FOR R=1 TO 20
20 PRINT RND (6);
30 NEXT R
```

このプログラムをRUNすると、0から5までの数字が20個並びます。一般にRNDはRND(n)の形をとり、カッコの中の数字は、出したい乱数の最大値よりもひとつ大きくします。

ところで、サイコロは、1から6までの乱数ですね。ということは、

PRINT [RND (6)] +1

0,1,2,3,4,5の数のどれかが出ます。

にすればよいわけです。

さっそく"さいころ"のプログラムをつくってみましょう。

```
10 PRINT " さいころ "
20 PRINT RND(6)+1;
30 IF INKEY$ <> "␣" THEN 30
40 FOR W=0 TO 200:NEXT W
50 GOTO 20
```

スペースキーを押すたびに、数字がひとつ出ます。

note 1 ! 30行でINKEY$を使っています。無限ループにしました。

note 2 ! 40行は、時間かせぎするために入れた一行です。
INKEY$の判断はとてもはやいので、このようにしないと、一度にたくさんの数が次々に画面に出てしまいます。

RND文の応用はたくさんあります。

- 座標を乱数できめて図形を描く。
  LINE, CIRCLE, PSET等
- キャラクタの動きをデタラメにする。
- サイコロやジャンケンに使う。

} など、ゲームに使うと意外性がでておもしろくなります。

RND文を使ったグラフィックスのサンプルプログラムをあげておきます。参考にしてください。

サンプルプログラム①
星空(きれいですよ)

```
10 SCREEN 2:CLS
20 FOR N=0 TO 200
30 X=RND(255)
40 Y=RND(191)
50 C=RND(16)
60 PSET(X,Y),C
70 NEXT N
```

座標と色を乱数で作ります。

サンプルプログラム②
丸がいっぱい

```
10 SCREEN 2:CLS
20 FOR N=0 TO 30
30 X=RND(230)
40 Y=RND(170)
50 C=RND(16):IF C<2 THEN50
60 R=RND(30)
70 CIRCLE(X,Y),R,C
80 NEXT N
```

座標と色と半経を乱数で作ります。

SYSTEM（システム）

ベーシックからメニューにもどります。

シュートゲームの背景をベーシックで作ったら、SYSTEM 文をプログラムの終わりに付け加えておけば自動的にメニューにもどります。

例

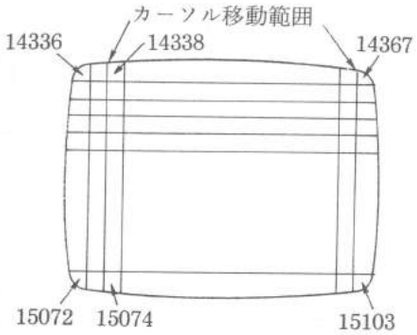
```
100 REM ---- SYSTEM テスト ----
120 SCREEN 2:CLS
130 Y=0:YY=191
140 FOR X=0 TO 255 STEP 8
150 LINE(X,Y)-(X,YY),15
160 NEXT X
200 X=0:XX=255
210 FOR Y=0 TO 191 STEP 8
220 LINE(X,Y)-(XX,Y),15
230 NEXT Y
240 SCREEN 1
300 SYSTEM
```


VPOKE（ブイポーク）、VPEEK（ブイピーク）

★★★ VPOKE ★★★

テキスト画面には座標があることを CURSOR 文の項で説明しました。テキスト画面には座標の他にアドレス（番地）もあります。VPOKE 文ではアドレスを使います。指定したテキスト画面のアドレスに文字や記号を書き込むのです。

テキスト画面のアドレス



アドレスは、カーソルの移動できる範囲より左に 2、右に 1 広くなっています。
ブラウン管の形状によっては文字が見えなくなることがあるので、カーソル移動の範囲をせばめて表示場所を内側に寄せているのです。

(例)

VPOKE　14336, 127
　　これでは、ほとんど見えません。

VPOKE　14337, 127
　　画面の左上に表示されます。

このように、

ＶＰＯＫＥ　アドレス，キャラクタコード

と入力すると、その場所に文字や記号を書き込むことができます。

★★★ VPEEK ★★★

テキスト画面に何が書き込まれているか調べます。

(例)

```
VPOKE 14860, 65
```

を実行すると画面にAが表示されます。

```
PRINT VPEEK (14860)
```

を実行すると65と表示されます。

キャラクターコード65をVPOKEで書き込むとAの文字になり、VPEEKで読むと65とコードで返ってきます。

VPEEKはテキスト画面でゲームを作る場合に、キャラクタを見分けることができるので便利です。

VPOKEとVPEEKのサンプルプログラム

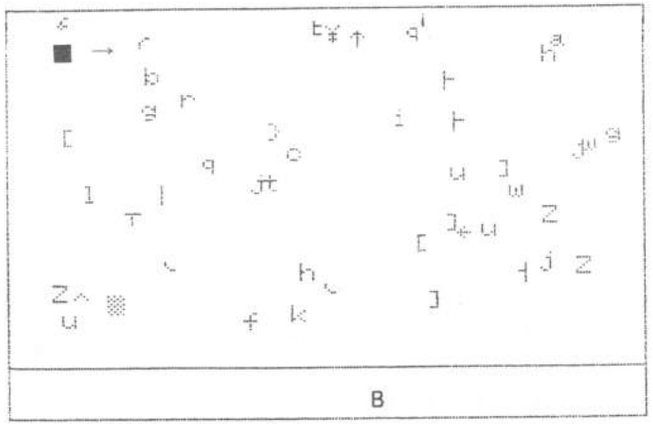
```
10 CLS
20 FOR N=0 TO 50
30 V=RND(27)
40 W=RND(20)
50 C=RND(255)
60 IF C<=32 THEN 50
70 CURSOR V,W:PRINT CHR$(C)
80 NEXT N
```

画面にランダムに文字や記号を散りばめます。


```
90 CURSOR 0,20:PRINT "――――――――――――――――――――"
100 X=14338
110 VPOKE X,127
120 P=VPEEK(X+1)
130 CURSOR 15,21:PRINT CHR$(P)
140 FOR W=0 TO 30:NEXT W
150 VPOKE X,32
160 X=X+1:IF X=14972THEN END
170 IF P< > 32 THEN BEEP:GOTO200
190 GOTO 110
200 FOR W=0 TO 500:NEXT W
210 CURSOR15, 21:PRINT"          "
220 GOTO 110
```

VPOKEで書き込んだ■が移動していきます。
文字や記号に■がぶつかると……

どうなるでしょう。



↑

■ がぶつかった文字や記号を表示します。

ジョイスティックを使って

（ジョイスティックをお持ちの方）

STICK (n) と STRIG (n) はジョイスティックに関係のある命令文です。

★★★ STRIG（スティックトリガ）★★★

ジョイスティックのプッシュスイッチの状態を見ます。

使い方は

IF STRIG (n) = ▨ THEN ～

STRIG (1)　1 Player 用のジョイスティック

STRIG (2)　2 Player 用のジョイスティック

▨ の所に入れる数値

0 の場合	OFF（オフ）	押されていません。
1 〃	左 ON（オン）	左が押されています。
2 〃	右 ON（オン）	右が押されています。
3 〃	左, 右 ON（オン）	左, 右が押されています。

★★★ STICK（スティック）★★★

ジョイスティックの状態を見ます。この命令でキャラクタやスプライトを操縦しましょう。

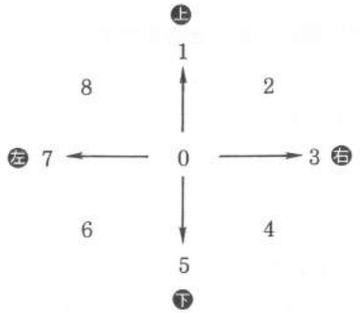
使い方は

IF STICK (n) = ▨ THEN ～

STICK (1)　　1 player 用のジョイスティック

STICK (2)　　2 player 用のジョイスティック

▨の所に入れる数値



▨の 値を図の方向の値にすれば、その方向にレバーを倒したことになります。

たとえば

IF　STICK (1) = 3　THEN ～

(レバーを右に倒したら ……)

IF　STICK (1) = 7　THEN ～

(レバーを左に倒したら ……)

このように使います。

サンプルプログラムを参考にしてください。

サンプルプログラム

```
10 SCREEN2:CLS
20 X=120: Y=90
30 PATTERN  S#160,"183C7EFFFF7E3C18"
40 S1=STICK(1)
50 SPRITE 0,(X,Y),160,5
60 IF S1=1 THEN E=-1:GOTO 200
70 IF S1=3 THEN E=1:GOTO 300
80 IF S1=5 THEN E=1:GOTO 200
90 IF S1=7 THEN E=-1:GOTO 300
100 GOTO 40
200 Y=Y+E:IF Y<0 OR Y>180 THEN Y=Y-E
210 GOTO 100
300 X=X+E:IF X<0 OR X>255 THEN X=X-E
310 GOTO 100
```

これは基本型です。30行と 40行の間や 40行と 50行の間にいろいろの要素を入れます。

```
25 V=100:W=30
35 PATTERN  S#180,"FF00FF00FF00FF00"
55 SPRITE 1,(V,W),180,8
95 V=V+1
```

CALL, POKE, LIMIT, PEEK

機械語という言葉があります。BASICは英語ですが、私たちの日常の言葉に近いのでわかりやすくて初心者向きです。でも機械語は少しやっかいです。16進法ですし、コンピュータの仕組みがわからないと、なかなか使えないからです。

ここでは、機械語プログラムについては、あまり詳しく触れません。ただ、コンピュータ内部のメモリ(情報をたくわえておくところ)を分ければ、BASICプログラムだけでなく機械語プログラムもつくれるのだということを覚えておいてください。

★★★ LIMIT(リミット) ★★★

BASICプログラム領域の終了番地を設定します。

BASICプログラム領域を限定します。LIMIT文で設定した番地以降はBASICインタプリタからの影響を受けずに使用可能になります。

note! 「番地」

メモリには番地がつけられています。これは、ちょうど住所の番地のような働きをします。つまり、メモリ内の特定の場所を指定するのです。

ベーシックの行番号とは違います。

★★★ POKE(ポーク) ★★★

POKE 番地 , データ

の形で、指定したメモリの番地にデータを書き込みます。プログラム領域を十分に把握したうえで、空いている領域に書き込んでください。

シュートゲームの内容を変えるときに使えます。

```
POKE&H830F,&H1 CR
```

ベーシックでこのように打ち込んでから、シュートゲームにもどってゲームをしてください。プレイヤシップが動く方向に弾が出るようになります。

★★★ CALL(コール) ★★★

BASICプログラム領域外の、機械語プログラムを呼び出します。

一般に、

CALL 開始番地

の形で入力します。

★★★ PEEK(ピーク) ★★★

番地で指定したメモリの内容を見せてくれます。

さきほど、POKE文のところで&H830Fの内容を変更しましたから、それをPEEKで確認しましょうか。

```
PRINT PEEK(&H830F) CR
```

CR キーを押すと、1が表示されます。これは、

```
POKE&H830F,&H1
```

の1です。(10進法の1です。PEEK 文では、結果は10進法で表示されるのです。)
PEEK は、上記のように PRINT 文で直接内容を見る他に、プログラムの中で使います。

OUT と INP

データをコンピュタの外部へ出して、理解しやすい形に(プリントアウトする、或いは画面上に表示)することを出力(アウトプット)といいます。逆に、キーボードやデータレコーダからデータを入れることを入力(インプット)といいます。データが出入りする場所をポートといいます。

★★★ OUT(アウトポート) ★★★

出力ポートにデータを出力させます。一般に、

```
OUT 出力ポート番号 , データ
```

の形をとります。 出力ポートには、プリンタ出力は何番、カセット出力は何番 …… というように、あらかじめ指定された番号がついています。(例) データレジスタ &H BE コマンドレジスタ &H BF サウンドジェネレータ &H 7F
出力ポート番号は 0 ～ 255 (&H 00 ～ &H FF) までの整数です。

★★★ INP(インポート) ★★★

I/O ポート番号を指定して、ポート上にあるデータを読み出します。

(例)

```
10 A=INP(&HBE)
20 PRINT A
RUN
 32
```

行番号10で、I/O ポート番号 BE(16進数)にあるデータを、変数Aに読み出します。

note ! I/O ポート番号は、システム中ですでに決定している番号で、0 ～ 255 (&H00 ～ &HFF) までの整数です。

サンプルプログラム

イロのテスト

```
10 REM "イロノ テスト"
20 SCREEN 2:CLS
30 X=24:Y=16:R=12:C=2
40 CIRCLE(X,Y),R,C,100,0,100,
BF
50 X=X+30
55 C=C+1:IF C=16 THEN C=2
60 IF X>=255 THEN X=24:Y=Y+16
70 IF Y>=191 THEN 90
80 GOTO 40
90 GOTO 90
```

イタズラガキ

```
10 REM "イタズ゛ラガ゛キ"
20 SCREEN 2:CLS
30 RX=RND(220):RY=RND(180)
40 C=RND(15)
50 IF C=0 OR C=1 THEN 40
60 LINE-(RX,RY),C
65 FOR N=0 TO 500:NEXT N
70 GOTO 30
```


イロのボール

```
10 REM "イロノ ボ゛ール
20 SCREEN 2:CLS
30 X=24:Y=12:R=12:C=2
40 CIRCLE(X,Y),R,C,100,0,100,
BF
50 X=X+30
55 C=C+1:IF C=16 THEN C=2
60 IF X>=255 THEN X=24:Y=Y+28
70 IF Y>=191 THEN 90
80 GOTO 40
90 GOTO 90
```

エンをカク

```
10 REM エンヲ カク
20 R=0:C=0:H=0:S=0:E=0
30 PRINT " エンヲ カク"
40 INPUT "ハンケイ (R)=1~90  ";R
50 INPUT "イ ロ   (C)=2~15  ";C
60 INPUT "ヒリツ  (H)=1~500 ";H
70 INPUT "ハジ゛メ (S)=0~95  ";S
80 INPUT "オワリ  (E)=5~100 ";E
110 SCREEN 2:CLS
120 X=128:Y=96
130 CIRCLE(X,Y),R,C,H,S,E
140 SCREEN 1
150 GOTO 20
```

モジ　ナラベカエ

```
200 REM ----モジ ナラベカエ----
210 CLS
220 INPUT "モジヲ イクツ ナラベマスカ?
";S
230 PRINT
240 DIM A$(S)
250 FOR N=1 TO S
260 INPUT "モジヲ イレル。 ";A$(N)
270 NEXT N
280 FOR N=1 TO S-1
290 FOR M=N+1 TO S
300 IF A$(N)<=A$(M) THEN GOTO
 340
310 X$=A$(N)
320 A$(N)=A$(M)
330 A$(M)=X$
340 NEXT M
350 NEXT N
360 FOR N=1 TO S
370 PRINT A$(N)
380 NEXT N
390 END
```


ケイサン

```
1 REM ------ケイサン------
10 T=1
20 FOR N=0 TO 5
30 CLS
40 X=5:Y=5
50 CURSOR X,Y:INPUT "A= ";A
60 CURSOR X,Y+1:INPUT "B= ";B
70 C=A+B
80 CURSOR X,Y+3:PRINT A;"+";B;"="
90 CURSOR X+10,Y+3:INPUT D
100 FOR W=0 TO 200:NEXT W
110 IF C<>D THEN 90
120 PRINT:PRINT
130 PRINT "セイカイ ! スペースキーヲ オシテクダサイ。";
T;" カイメ"
140 T=T+1:IF T=6 THEN 170
150 IF INKEY$="" THEN 150
160 NEXT N
170 END
```

スウジ　ナラベカエ

```
1 REM ----ｽｳｼﾞ ﾅﾗﾍﾞｶｴ----
10 CLS
20 INPUT "ｽｳｼﾞｦ ｲｸﾂ ｸﾗﾍﾞﾏｽｶ? ";S
30 PRINT
40 DIM A(S)
50 FOR N=1 TO S
60 INPUT "ｽｳｼﾞｦ ｲﾚﾙﾄ。 ";A(N)
70 NEXT N
80 FOR N=1 TO S-1
90 FOR M=N+1 TO S
100 IF A(N)<=A(M) THEN GOTO 140
110 X=A(N)
120 A(N)=A(M)
130 A(M)=X
140 NEXT M
150 NEXT N
160 FOR N=1 TO S
170 PRINT A(N)
180 NEXT N
190 END
```


CURSOR コントロール

```
5 REM ｷｰ ｺﾝﾄﾛｰﾙ
10 CLS
20 X=15:Y=10:F=1
100 CURSOR X,Y:PRINT "A"
105 Z$=INKEY$
110 IF Z$=CHR$(28) THEN F=1:GOSUB 200
120 IF Z$=CHR$(29) THEN F=-1:GOSUB 200
130 IF Z$=CHR$(30) THEN F=-1:GOSUB 300
140 IF Z$=CHR$(31) THEN F=1:GOSUB 300
160 CURSOR X,Y:PRINT "A"
170 GOTO 100
200 REM
205 CURSOR X,Y:PRINT " "
210 X=X+F:IF X=0 OR X=29 THEN X=X-F
220 RETURN
300 REM
305 CURSOR X,Y:PRINT " "
310 Y=Y+F:IF Y=0 OR Y=23 THEN Y=Y-F
320 RETURN
```

```
5 REM ｶｰ ｺﾝﾄﾛｰﾙ
10 CLS
20 V=14736:A=65:F=1:T=16:S=12
100 VPOKE V,A
105 Z$=INKEY$
110 IF Z$=CHR$(28) THEN E=1:GOSUB 200
120 IF Z$=CHR$(29) THEN E=-1:GOSUB 200
130 IF Z$=CHR$(30) THEN F=-32:GOSUB 30
0
140 IF Z$=CHR$(31) THEN F=32:GOSUB 300
170 GOTO 100
200 REM
205 VPOKE V,32
210 V=V+E:T=T+E:IF T<2 OR T>29 THEN V=
V-E
220 RETURN
300 REM
305 VPOKE V,32
310 V=V+F
320 RETURN
```



```
10 SCREEN 2:CLS
20 X=24:Y=90:XX=240:YY=90
50 PATTERN S#0,"FF00FF00FF00F
F00"
60 PATTERN S#4,"AAAA000000AAA
AAA"
100 SPRITE 0,(X,Y),0,5
110 SPRITE 4,(XX,YY),4,8
120 X=X+1
130 IF SCOIN=-1 THEN GOSUB 20
0
190 GOTO 100
200 REM
210 X=24
230 BEEP:BEEP
290 RETURN
```

スプライト　テスト

```
5 REM スプライト テスト
10 SCREEN 2:CLS
20 X=48:Y=90:XX=130:YY=90
30 M=1
50 PATTERN S#0,"0000000000000000"
51 PATTERN S#1,"0000000000000000"
52 PATTERN S#2,"FFFFFF0707070707"
53 PATTERN S#3,"0707070707FFFFFF"
60 PATTERN S#4,"00000000000000FF"
61 PATTERN S#5,"FFFF0000000000000"
62 PATTERN S#6,"00000000101810FE"
63 PATTERN S#7,"FFFE1C1810000000"
70 PATTERN S#8,"FFFFFFE0E0E0E0E0"
71 PATTERN S#9,"E0E0E0E0E0FFFFFF"
72 PATTERN S#10,"0000000000000000"
73 PATTERN S#11,"0000000000000000"
100 MAG M
110 SPRITE 0,(XX,YY),0,3
120 SPRITE 2,(XX,YY),8,5
130 SPRITE 1,(X,Y),4,8
140 X=X+1:IF X=255 THEN GOSUB 200
190 GOTO 110
200 X=0
210 M=M+2:IF M>3 THEN M=1
220 MAG M
230 RETURN
```


ニゲロニゲロ

```
10 REM ----- オイカケ -----
20 SCREEN 2:CLS
30 GOSUB 1000
40 X=120:Y=90
50 V=16:W=90
60 LINE(16,0)-(248,191),3,B
70 MAG 2
100 REM --- キー ヲ ミル ----
110 Z$=INKEY$
120 IF Z$=CHR$(28)THEN F=2:GOSUB 300
130 IF Z$=CHR$(29)THEN F=-2:GOSUB 300
140 IF Z$=CHR$(30)THEN F=-2:GOSUB 400
150 IF Z$=CHR$(31)THEN F=2:GOSUB 400
200 REM --- オイカケル ---
210 SPRITE 0,(X,Y),208,8
220 SPRITE 16,(V,W),216,5
250 IF V>X THEN E=-1
255 IF V<X THEN E=1
260 IF W<Y THEN D=1
265 IF W>Y THEN D=-1
270 V=V+E:W=W+D
275 REM --- ツカマエタ ---
280 IF SCOIN THEN BEEP:GOTO 600
290 GOTO 100
300 REM --- ヨコニ ニゲロ ---
310 X=X+F:IF X<16 OR X>232 THEN X=X-F
320 RETURN
400 REM --- タテ ニ ニゲロ ---
410 Y=Y+F:IF Y<0 OR Y>175 THEN Y=Y-F
420 RETURN
600 REM --- ツカマッタ シルシ ---
610 FOR R=5 TO 40 STEP 5
620 CIRCLE(X,Y),R,10
630 CIRCLE(X,Y),R,15
640 NEXT R
650 GOTO 20
1000 PATTERNS#208,"383810FE10102844"
1010 PATTERNS#216,"183C5ADBFF244281"
1050 RETURN
```

スペースゲームの背景　BREAK キーで止めてから、シュートゲームに移りましょう。

```
10 REM ------ ｽﾍﾟｰｽ ｹﾞｰﾑ ------
20 REM --- ｼﾞﾒﾝ ｦ ｶｸ ------
30 SCREEN 2:CLS
40 LINE(0,120)-(255,190),7,BF
50 Y=120:U=1
60 FOR TY=0 TO 8
70 LINE(0,Y+U)-(255,Y+U),10
80 U=U+1:Y=Y+U
90 NEXT TY
100 X=120
110 FOR XX=0 TO 255 STEP 31
120 X=X+8
130 LINE(X,120)-(XX,190),15
140 NEXT XX
150 YH=190
160 FOR X=120 TO 88 STEP -6
170 YH=YH-10
180 LINE(X,120)-(0,YH),15
190 NEXT X
200 YH=190
210 FOR X=200 TO 236 STEP 8
220 YH=YH-14
230 LINE(X,120)-(255,YH),15
240 NEXT X
250 REM --- ﾎｼ ｦ ｶｸ -----
260 FOR P=0 TO 100
270 X=RND(255)
280 Y=RND(120)
290 PSET (X,Y),C
300 C=RND(15)
310 NEXT P
320 GOSUB 530
330 REM ------ ｵﾘｷｻﾏ ------
340 CIRCLE(48,48),32,10,,,,BF
350 X=48:Y=48
360 CIRCLE(X-8,Y+4),3,1,150,,
,BF
370 CIRCLE(X+8,Y+4),3,1,150,,
,BF
380 CIRCLE(X-8,Y-10),5,1,140,
50,100
390 CIRCLE(X+8,Y-8),5,1,120,5
0,100
400 CIRCLE(X,Y+18),4,1,150
```



```
410 PLAY"C"
420 CIRCLE(X,Y+18),4,10,150
430 CIRCLE(X,Y+18),4,1,80
440 PLAY"E"
450 CIRCLE(X,Y+18),4,10,80
460 CIRCLE(X,Y+18),4,1,1
470 PLAY"G"
480 CIRCLE(X,Y+18),4,10,1
500 Z$=INKEY$
510 IF Z$=" " THEN SYSTEM
520 GOTO 400
530 REM ------ ﾋﾟﾗﾐｯﾄﾞ ------
540 X=180:Y=122:C=5
550 LINE(X,Y+5)-(X-30,Y),15
560 LINE-(X-5,Y-25),15
570 LINE-(X,Y+5),15
580 PAINT(X-10,Y-10),15
590 LINE(X,Y+5)-(X-5,Y-25),5
600 LINE-(X+10,Y-2)
610 LINE-(X,Y+5)
620 PAINT(X+5,Y-5),5
630 RETURN
```

```
10 REM -- ﾍﾟﾝｷﾞﾝ ﾄ ﾎﾞｰﾙ --
20 SCREEN 2:CLS
30 GOSUB 370
40 LINE(0,0)-(255,48),15,BF
50 LINE(0,104)-(255,144),7,BF
60 LINE(0,170)-(255,191),3,BF
70 M=1:MAG M
80 REM
90 PATTERN S#200,"070F040000C
0603F"
100 PATTERN S#201,"180000000000
000001"
110 PATTERN S#202,"80C0C06163
267EFC"
120 PATTERN S#203,"FC3818181818
000080"
130 PATTERN S#204,"00000B1515
3F1F00"
140 PATTERN S#205,"070F0F6F6F
7F3700"
150 PATTERN S#206,"0000008080
C08000"
160 PATTERN S#207,"80C0E0E2E7
FFEC00"
170 PATTERN S#208,"070F040000
00001F"
180 PATTERN S#209,"3860800000
000001"
190 PATTERN S#210,"80C0C06060
2078FE"
200 PATTERN S#211,"FE3B191918
000000"
210 REM
220 COLOR 1,3
230 PRINTCHR$(17)
240 PRINT:PRINT"    S E G A
"
250 PRINT:PRINT"   HOME BASIC
"
260 X=255:Y=90:P=200:E=-2:F=9
0
270 MAG M
280 SPRITE 10,(X,Y),P,4
290 SPRITE 11,(X,Y),204,15
```



```
300 SPRITE 12,(X+60,F),160,8
310 P=P+8:IF P>208 THEN P=200
320 F=F+E:IF F< 50 THEN E=2
330 IF F>=91 THEN E=-2
340 X=X-2:IF X<=0 THEN X=255:
GOTO 420
350 BEEP
360 GOTO 270
370 PATTERN S#160,"030F3F3F7F
7FFFFF"
380 PATTERN S#161,"FFFF7F7F3F
3F0F03"
390 PATTERN S#162,"C0F01CCCE6
F2FBFB"
400 PATTERN S#163,"FFFFFEFEFC
FCF0C0"
410 RETURN
420 M=M+2:IF M>3 THEN M=1
430 GOTO 270
```

ヒコーキ　ゲキツイ　ゲーム

```
10 REM - ﾋｺｰｷ ｹﾞｷﾂｲ ｹﾞｰﾑ -
20 REM ---- ｼｮｷ ｾｯﾃｲ ----
30 SCREEN 2:COLOR 15,1:CLS
40 MAG 1:SP=188
50 X=40:Y=64:V=64
60 GOSUB 2000
90 X=40:Y=64:V=64
100 REM ---- ｷｰ ｦ ﾖﾐﾙ ----
120 Z$=INKEY$
130 IF Z$=CHR$(30) THEN V=V-4
135 IF V<8 THEN V=8
140 IF Z$=CHR$(31) THEN V=V+4
145 IF V>180 THEN V=180
150 IF Z$=CHR$(8) THEN 6000
160 IF Z$=CHR$(12) THEN 6000
190 IF Z$="" THEN 210
200 REM ---- ﾋｺｰｷ ｦ ﾄﾊﾞｽ ----
210 SPRITE1,(XX,V),184,4
220 SPRITE6,(X,Y),188,8
230 X=X+4:IF X>255 THEN X=0
235 XX=XX-4
240 IF XX<=0 THEN XX=255:GOTO
 120
250 IF SCOIN=-1 THEN 5000
260 K=RND(8)-4
270 Y=Y+K:IF Y<0 THEN Y=16
280 SOUND 4,1,15
290 GOTO 120
2000 REM ---- ﾋｺｰｷ ﾉ ｶﾀﾁ ----
2080 PATTERN S#184,"0000001F8
989BFFF"
2090 PATTERN S#185,"B09F80000
0000000"
2100 PATTERN S#186,"000000800
60F8FFF"
2110 PATTERN S#187,"7CE000000
0000000"
2120 PATTERN S#188,"000000016
0F0F1FF"
2130 PATTERN S#189,"3E0700000
0000000"
2140 PATTERN S#190,"000000F89
191FDFF"
2150 PATTERN S#191,"0DF901000
0000000"
2160 PATTERN S#200,"070300010
101013F"
2161 PATTERN S#201,"233F7F7F7
F030007"
2162 PATTERN S#202,"F0E080C0C
0C0C0FE"
2163 PATTERN S#203,"E2FEFFFFF
FE080F0"
2190 RETURN
5000 REM ---- ｼｮｳﾄﾂ 2ｷ ﾂｲﾗｸ ---
-
5010 S1=200:S2=200
5020 SPRITE 1,(XX+8,V),S1,5
5030 SPRITE 6,(X,Y),S2,8
5040 Y=Y+1:V=V+1:IF Y<>180 TH
EN 5010
5050 SOUND 4,2,15
5060 FOR N=0 TO 1000:NEXT N
5070 XX=255
5080 HY=0:HX=0:S1=188:S2=184
5090 GOTO 10
6000 REM ---- ｷｶﾝｼﾞｭｳ ﾊｯｼｬ ----
6010 VY=V+8:HY=Y:HX=X
6020 IF HX>XX THEN HX=0
6030 LINE(XX,VY)-(HX,VY),15
6040 IF VY<Y+4 OR VY>Y+12 THE
N GOTO 6060
6050 IF X=HX THEN 8000
6060 LINE(XX,VY)-(HX,VY),0:HX
=0
6070 GOTO 120
8000 REM ---- ﾃｯｷ ｹﾞｷﾂｲ ----
8010 SP=200
8020 LINE(XX,VY)-(HX,VY),0:HX
=0
8030 FOR N=20 TO 1000 STEP 5
8040 SPRITE 6,(X,Y),SP,8
8050 SOUND 1,N,15
8060 Y=Y+1:IF Y=180 THEN 8080
8070 NEXT N
8080 HY=0:HX=0:SP=188
8090 SOUND 0:GOTO 10
```

付　　録

変数・配列

数値変数 …… A，B，～Z，AA，AB，～ZZ
　　AO，A1～A9
数値配列 …… 2次元まで　A(15)，B(5，5)，
　　AC(3，3)
文字列変数 …… A$，AB$，A1$
文字列配列 …… 2次元まで　A$(15)，
　　B$(5，5)，AC$(3，3)

- 変数名は、先頭の1文字を英字、以降を英字または数字とする。長さは何文字でも良いが先頭の2文字で区別する。
- 変数と配列の名前が同じでも良い。

<文字列変数、配列の範囲>

文字の長さ　　0～31

定　数

<数値定数>

- 整数形式　　（例）3，－2，9992
- 16進形式　　&H［16進の値］…… 0000～FFFF
　　（例）　&H64 …… 100と同じ
　　&HFFFF …… －1と同じ

整数の範囲　　＋32767～－32768
　　&H8000～&H7FFF

- 負の最大値の表記について
　負の最大値 (−32768) はそのまま表記するとオーバーフローエラーとなる。
　（理由：−32768は定数32768にマイナスの単項演算子を付けたものと解釈されるため。）
　負の最大値は次の様に表記する。
　(−1−32767)　又は　&H8000

<文字列定数>

" で囲む。" は "" を2つ用いて示す。

（例）　"ABC"　→ 文字ABC
　　""　→ 文字NULL（なし）
　　""""　→ 文字 "
　　"A3""64"　→ 文字A3"64

内　　容	制　　限
画面から内部に取り込まれる文字数	255文字
文字列として扱うことの可能な文字数	31文字
FOR ～ NEXT のネスティングのレベル数	8レベル
GOSUB, RETURN のネスティングレベル数	16レベル

CTRL (コントロール)

CTRL キーを押したまま、文字キーを打ちますと、表にある動作が行われます。

コントロールコード

キー操作	PRINT CHR$(値);	機　　能
CTRL + A	PRINT CHR$(1);	NULL 文字なし
C	——	BREAK プログラムの実行中止
E	5	カーソル以降の文字をクリア
G	7	BELL ピッと音を出す
H	8	DEL 文字をデリート
I	9	HT 水平 TAB
J	10	LF ラインフィード
K	11	HM カーソルを左上端に戻す HOME
L	12	CLR 画面のクリア　HOME
M	13	CR キャリッジリターン
N	14	カナ ↔ 英数字切りかえ
O	15	♪ 画面をテキスト ↔ グラフィック切りかえ

キー操作	PRINT CHR$（値）;	機　　能
P	16	標準文字サイズ
Q	17	横２倍文字サイズ (SCREEN 2)
R	18	INS インサート
S	19	キー入力（A～Z）シフトなし大文字
T	20	〃　（a～z）シフトなし小文字
U	21	ラインをクリアしカーソルを左端に戻す。
V	22	ノーマルモードすべて初期の状態にする。
W	23	GRAPH キー入力グラフモード ↔ 英字切換
X	24	クリック音の ON ↔ OFF 切換
—	28	⇨ カーソル移動
—	29	⇦　〃
—	30	⇧　〃
—	31	⇩　〃

プログラム中でコントロールコードを使う場合は、
PRINT CHR$（値）で入力して下さい。

キャラクターコード

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0				0	@	P	┼	p	┐	┬	┬	ー	タ	ミ	た	み
1			!	1	A	Q	a	q	├	あ	。	ア	チ	ム	ち	む
2			"	2	B	R	b	r	┌	い	「	イ	ツ	メ	つ	め
3			#	3	C	S	c	s	└	う	」	ウ	テ	モ	て	も
4			$	4	D	T	d	t	┐	え	、	エ	ト	ヤ	と	や
5			%	5	E	U	e	u	┘	お	・	オ	ナ	ユ	な	ゆ
6			&	6	F	V	f	v	を	か	ヲ	カ	ニ	ヨ	に	よ
7			'	7	G	W	g	w	ぁ	き	ァ	キ	ヌ	ラ	ぬ	ら
8			(	8	H	X	h	x	ぃ	く	ィ	ク	ネ	リ	ね	り
9			)	9	I	Y	i	y	ぅ	け	ゥ	ケ	ノ	ル	の	る
A			*	:	J	Z	j	z	ぇ	こ	ェ	コ	ハ	レ	は	れ
B			+	;	K	[	k	{	ぉ	さ	ォ	サ	ヒ	ロ	ひ	ろ
C			,	<	L	¥	l	|	ゃ	し	ャ	シ	フ	ワ	ふ	わ
D			−	=	M	]	m	}	ゅ	す	ュ	ス	ヘ	ン	へ	ん
E			.	>	N	^	n	~	ょ	せ	ョ	セ	ホ	゛	ほ	—
F			／	?	O	▦	o	■	っ	そ	ッ	ソ	マ	゜	ま	▏

コントロール・コード

キャラクターコード

32	SP	48	0	64	ⓐ	80	P	96		112	p	128		144	
33	!	49	1	65	A	81	Q	97	a	113	q	129		145	あ
34	"	50	2	66	B	82	R	98	b	114	r	130		146	い
35	#	51	3	67	C	83	S	99	c	115	s	131		147	う
36	$	52	4	68	D	84	T	100	d	116	t	132		148	え
37	%	53	5	69	E	85	U	101	e	117	u	133		149	お
38	&	54	6	70	F	86	V	102	f	118	v	134	を	150	か
39	▼	55	7	71	G	87	W	103	g	119	w	135	あ	151	き
40	(	56	8	72	H	88	X	104	h	120	x	136	い	152	く
41	)	57	9	73	I	89	Y	105	i	121	y	137	う	153	け
42	*	58	:	74	J	90	Z	106	j	122	z	138	え	154	こ
43	+	59	;	75	K	91	[	107	k	123	{	139	お	155	さ
44	,	60	<	76	L	92	¥	108	l	124	:	140	や	156	し
45	−	61	=	77	M	93	]	109	m	125	}	141	ゆ	157	す
46	.	62	>	78	N	94	^	110	n	126	~	142	よ	158	せ
47	/	63	?	79	O	95		111	o	127	■	143	つ	159	そ

160		176	ー	192	タ	208	ミ	224	た	240	み
161	。	177	ア	193	チ	209	ム	225	ち	241	む
162	「	178	イ	194	ツ	210	メ	226	つ	242	め
163	」	179	ウ	195	テ	211	モ	227	て	243	も
164	、	180	エ	196	ト	212	ヤ	228	と	244	や
165	・	181	オ	197	ナ	213	ユ	229	な	245	ゆ
166	ヲ	182	カ	198	ニ	214	ヨ	230	に	246	よ
167	ァ	183	キ	199	ヌ	215	ラ	231	ぬ	247	ら
168	ィ	184	ク	200	ネ	216	リ	232	ね	248	り
169	ゥ	185	ケ	201	ノ	217	ル	233	の	249	る
170	ェ	186	コ	202	ハ	218	レ	234	は	250	れ
171	ォ	187	サ	203	ヒ	219	ロ	235	ひ	251	ろ
172	ャ	188	シ	204	フ	220	ワ	236	ふ	252	わ
173	ュ	189	ス	205	ヘ	221	ン	237	へ	253	ん
174	ョ	190	セ	206	ホ	222	゛	238	ほ	254	
175	ッ	191	ソ	207	マ	223	゜	239	ま	255	

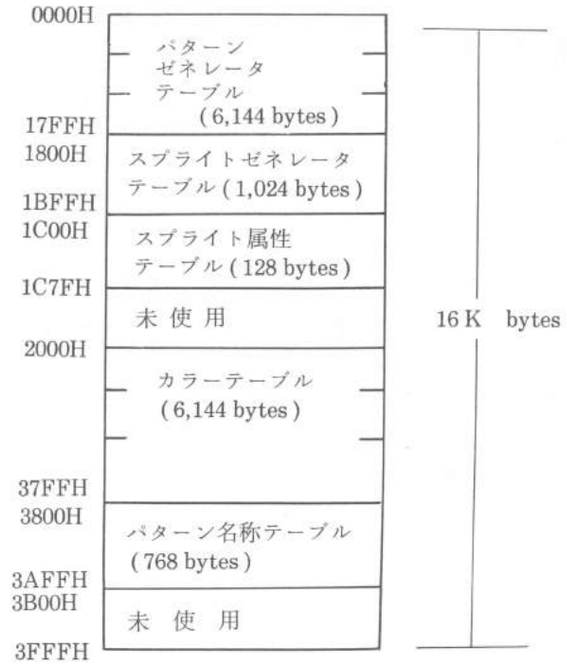
BASIC の S# PATTERN とパターンていぎ # との関係

BASIC	パターン	BASIC	パターン	BASIC	パターン	BASIC	パターン	BASIC	パターン	BASIC	パターン
0	0	22		44	11	66		88	22	110	
1		23		45		67		89		111	
2		24	6	46		68	17	90		112	28
3		25		47		69		91		113	
4	1	26		48	12	70		92	23	114	
5		27		49		71		93		115	
6		28	7	50		72	18	94		116	29
7		29		51		73		95		117	
8	2	30		52	13	74		96	24	118	
9		31		53		75		97		119	
10		32	8	54		76	19	98		120	30
11		33		55		77		99		121	
12	3	34		56	14	78		100	25	122	
13		35		57		79		101		123	
14		36	9	58		80	20	102		124	31
15		37		59		81		103		125	
16	4	38		60	15	82		104	26	126	
17		39		61		83		105		127	
18		40	10	62		84	21	106		128	32
19		41		63		85		107		129	
20	5	42		64	16	86		108	27	130	
21		43		65		87		109		131	

BASIC	パターン	BASIC	パターン	BASIC	パターン	BASIC	パターン	BASIC	パターン	BASIC	パターン
132	33	155		178		201		224	56	247	
133		156	39	179		202		225		248	—
134		157		180	45	203		226		249	
135		158		181		204	51	227		250	
136	34	159		182		205		228	57	251	
137		160	40	183		206		229		252	—
138		161		184	46	207		230		253	
139		162		185		208	52	231		254	
140	35	163		186		209		232	58	255	
141		164	41	187		210		233			
142		165		188	47	211		234			
143		166		189		212	53	235			
144	36	167		190		213		236	59		
145		168	42	191		214		237			
146		169		192	48	215		238			
147		170		193		216	54	239			
148	37	171		194		217		240	—		
149		172	43	195		218		241			
150		173		196	49	219		242			
151		174		197		220	55	243			
152	38	175		198		221		244	—		
153		176	44	199		222		245			
154		177		200	50	223		246			

メインメモリーマップ

VRAM マップ



エラーメッセージ

1　表　示　方　法

(1)　コマンド入力又はステートメントをダイレクトに入力してエラーが発生した場合。

?　メッセージ　error

(2)　テキスト(プログラム)を実行中にエラーが発生した場合。

?　メッセージ　error in　ラインナンバー

(3)　INPUT ステートメントによるキー入力データに誤りがあった場合。

?　メッセージ

エラーメッセージ(アルファベット順)

ARRAY NAME	DIM 文のパラメータが配列でない。
CAN'T CONTINUE	CONT による続行が不可能。
DEVICE NOT READY	プリンタが接続されていない。もしくは、プリンタが故障している。
DIVISION BY ZERO	わり算の分母が0である。
EXTRA IGNORED	INPUT 文の入力データがおかしい。
FOR NESTING	FOR～NEXT の入れ子が8回を越えた。
FOR VARIABLE NAME	FOR 文のあとの変数が数値変数でない。(文字などになっている。)適切な変数に変えよ。
GOSUB NESTING	GOSUB の入れ子が16回を越えた。
ILLEGAL DIRECT	ダイレクト命令実行不可能。
ILLEGAL FUNCTION CALL	関数のパラメタ、命令文のパラメタがおかしい。
ILLEGAL LINE NUMBER	行番号がおかしい。
LINE IMAGE TOO LONG	ラインが長くなりすぎる。
MEMORY WRITING	メモリへの書き込みエラー。
NEXT WITHOUT FOR	NEXT に対応する FOR 文がない。
NO PROGRAM	プログラムがないのに、SAVE しようとした。
NO VRAM−DATA	VRAM データがないのに、SAVE しようとした。

OUT OF DATA	READ 文で読み込むためのデータがない。
OUT OF MEMORY	メモリが足りない。
OVERFLOW	値や演算結果が許容範囲を越えた。
REDIM'D ARRAY	配列を 2 重に定義しようとした。
REDO FROM START	INPUT 文の入力がおかしい。最初から入力しなおせ。
RETURN WITHOUT GOSUB	GOSUB を行わないで RETURN 文が実行された。
STACK OVERFLOW	• カッコの使いすぎ。数値式が複雑すぎる。 • 文字列が複雑すぎる。 • PAINT する図形が複雑すぎる。
STRING TOO LONG	文字列が長すぎる。(255 を越えた。)
SUBSUCRIPT	添字の数 (個数)、或いは添字の値がおかしい。
SYNTAX	文法 (構文) 上のエラー。
SYSTEM	BASIC インタプリタのプログラムの動作エラー。
TYPE MISMATCH	代入する側と代入される側のタイプが合わない。(数値、文字列)
UNDEF'D LINE NUMBER	(GOTO, GOSUB, IF～THEN, RESTORE, RUN などで) 指定された行番号がない。
UNDEFINDED ARRAY	未定義の配列を ERASE しようとした。
UNPRINTABLE	上記以外のエラー。

コマンド・ステートメント・関数　索 引

(アルファベット順)

		ページ
FOR～TO NEXT ～ STEP	指定された条件で仕事をくりかえす。	48
FRE	どれくらいメモリが残っているか示す。	65
GOSUB	サブルーチンへ分岐する。	76
GOTO	指定された行番号へとぶ。	48
HEX $ (X)	Xの16進数文字列を与える。	96
IF～THEN	条件にしたがって仕事をする。(条件分岐)	72
INKEY $	キーが押されたかどうか調べる。	156
INP	入力ポートの入力内容を与える。	171
INPUT	キーから入力するように要求する。	68
LEFT $	文字列の左からX番目の文字列を与える。	94
LEN (S)	文字列の長さを与える。	93
LET	代入する。	67
LIMIT	メモリの上限を決める。	168
LINE	線を引く。	121
LIST	プログラム・リストを表示する。	56
LLIST	プログラム・リストをプリンタ用紙に書く。	55
LOAD	カセット・テープに入ったプログラムをメモリに入れる。(ロードする。)	54
LOADV	グラフィック面をロードする。	154
LPRINT	プリンタ用紙に書きこむ。	55

		ページ
MAG	スプライトで動かすキャラクタの大きさを指定する。	148
MID $ (S, X, Y)	文字列Sの左からX番目から長さYの文字列を与える。	94
NEW	プログラムを消す。(クリアする。)	47
ON～GOTO／ON～GOSUB	分岐する行番号を選択する。	74
OUT	出力ポートに出力する。	170
PAINT	囲まれた範囲をぬりつぶす。	135
PEEK	メモリX番地の内容を与える。	169
PLAY	音楽を演奏する。	97
PLEN	演奏(PLAYの実行)終了をチェックする。	109
POKE	メモリへ書き込みをする。	169
PRINT	画面に表示する。	38
PSET	点を打つ。	129
PATTERN	●キャラクタをつくる。●文字などのパターンを変更する。	138
READ	DATA文のデータを読みとる。	79
REM	コメントをつける。	60
RESTORE	READ文により読み込むDATA文の場所を指定する。	82
RETURN	GOSUBで分岐したサブルーチンからもどる。	76
RIGHT $	文字列の右からX番目までを与える。	94
RND (X)	乱数をつくる。	157
RUN	プログラムを実行する。	47

SEGA ホームベーシック解説書

発行所　株式会社　セガ・エンタープライゼス
H E 事 業 部
☎ 03－743－7435
お問合せは本社消費者サービス課
☎ 03－742－7068
〒144　東京都大田区羽田1丁目2番12号
昭和60年8月15日発行

印刷・製本　㈱ 日 本 現 図 社
〒140　東京都品川区北品川1丁目29番11号

108601

株式会社 セガ・エンタープライゼス