

【A-1】IchigoJam でスキーゲームを作る

●今回の目標

IchigoJam で動くスキーゲームを作ります。

プレイヤーを左右に動かして、下から出てくるポールの間を通ると、ポイントが入ります。



●スキーゲームを作る手順

以下の順番で、スキーゲームを作っていきます。

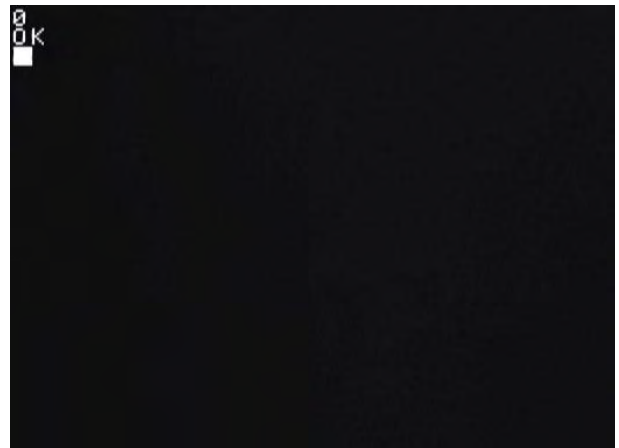
項目	内容	ページ
ゲーム画面を表示	点数を画面に表示	2
自分のキャラクターを表示	自分のキャラクターを画面上に表示	4
キャラクターを左右に動かす	矢印キーで自分のキャラクターを左右に動かす	5
はみ出しチェック	自分のキャラが画面からはみ出さないようにする	8
ポールを登場させる	画面下からポールを登場させる	9
点数、ミス	自分がポールを通過したら点数が入る、ポールに当たったらミスになる	11
ゲームのレベルアップ	点数が上がるとポールがだんだんせまくなる	14

●ゲームの画面を表示する

まず、プログラムの初期設定をした後、画面にスコア(点数)を表示します。

10 / *SKI*	コメントで、プログラムのタイトルを入れる
20 CLS:CLV	画面をクリア、変数をクリア
30 LOCATE 0,0	カーソルを画面左上へ移動
40 PRINT S	スコアを表示

入力できたら、「RUN」で実行してみましょう。
画面がクリアされて、左上に「0」が表示されます。



プログラムの内容を説明します。

10行:コメントで、プログラムのタイトルを入れています。

先頭に「/」（アポストロフィ、キーボードでは Shift キーを押しながら「7」を押す）を付けると、その行はコメントとなり、何も実行されません。ですから、「/」のあとは好きな文字を書くことができます。

プログラムを後で見た時にわかりやすくするために、プログラムにいろいろコメントを入れるといいでしょう。

20行:CLS 命令で、画面をクリアします。

その後の CLV(シーエルブイ)命令は、**変数**(へんすう)をクリアする命令です。

変数とは、数字を入れる入れ物(箱)と思ってください。

小学校の算数でやる「□」(四角)、中学校の数学でやる「x」と考えればいいです。

今回は、ゲームの最初に、全ての変数をクリアしています。

この行では、2つの命令を「:」(**コロン**)で続けて書いています。「:」を使うと、複数の命令を続けて書くことができます。

30 行:LOCATE(ローケート)命令は、画面に文字を表示する位置(カーソル位置)を設定する命令です。

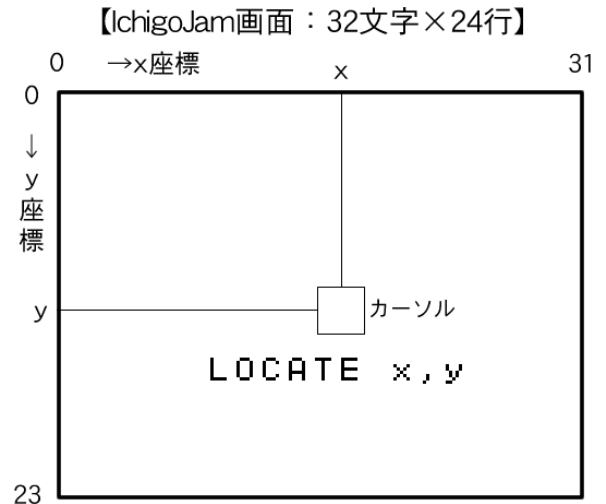
LOCATE **0** , **0**
 x 座標 y 座標

x 座標	画面の x 座標 (0~31)。
y 座標	画面の y 座標 (0~23)。

IchigoJam の画面サイズは、横 32 文字×縦 24 行になっています。

横(x 方向)の座標は 0~31、縦(y 方向)の座標は 0~23 になっています。

今回は「LOCATE 0,0」として、画面左上の座標を指定しています。



40 行:「PRINT S」で、画面にスコア変数 S の値を表示しています。

最初に CLV 命令で変数をクリアしていて、S の値も 0 なので、「0」と表示されます。

「PRINT “S”」と S をダブルクォーテーションで囲むと、文字「S」を表示しますが、「PRINT S」だと、「変数 S の値を表示する」という意味になります。

LOCATE 命令の座標の数字を変えると、スコア「0」の表示位置が変わります。

いろいろ変えて試してみましょう。

●自分のキャラクターを表示する

スコアと同じように、自分のキャラクターを表示してみましょう。
以下のプログラムを追加します。

```

10  / *SKI*
20  CLS : CLV
30  LOCATE 0,0
40  PRINT S
50  X=15 : Y=3
60  LOCATE X,Y
70  PRINT CHR$(249)

```

自機の x 座標を 15、y 座標を 3 にする
カーソルを座標(X,Y)へ移動
自分のキャラクターを表示

プログラムを実行してみましょう。
画面中央上に、人間のキャラクターが
表示されます。



50 行: 自分のキャラクターの横座標の変数 X、縦座標の変数 Y に値を入れています。「X=15」は、「X と 15 が等しい」という意味ではなく、「X に 15 を入れる」(代入する)という意味です。

60 行: LOCATE 命令で、カーソルを(X,Y)の座標へ移動します。
このように、座標を変数で指定することもできます。

70 行: PRINT 命令で、自分のキャラクターを表示しています。
ここでは「PRINT CHR\$(249)」として、CHR\$(シーエッチアールドル、キャラクター)関数を使って、文字コードでキャラクターを指定しています。

文字コードは右
の表を見てくだ
さい。

50 行の変数 X,Y の値
を変えれば、自分のキ
ャクターの表示位置
が変わります。
いろいろ変えて試して
みましょう。



文字コード 249

●キャラクターを左右に動かす

自分のキャラクターを、キー操作で左右に動かしてみましょう。
以下のように、プログラムを追加します。

```

10  ^*SKI*
20  CLS:CLV
30  LOCATE 0,0
40  PRINT S
50  X=15:Y=3
60  LOCATE X,Y
70  PRINT CHR$(249)
80  ^---GAMELOOP---
90  LOCATE X,Y
100 PRINT " "
110 IF BTN(LEFT)=1 THEN X=X-1
120 IF BTN(RIGHT)=1 THEN X=X+1
130 LOCATE X,Y
140 PRINT CHR$(249)
150 GOTO 80

```

キャラクターを消す

左矢印キーが押されていたら
X座標を1減らす

右矢印キーが押されていたら
X座標を1増やす

キャラクター
を再度表示

80行へ戻ってループ

プログラムを実行してみましょう。

矢印キーの左「←」・右「→」を押すと、キャラクターが左右に動きます。

矢印キーが押されているかどうかを調べるには、**BTN**(ボタン)関数を使います。
BTN 関数の文法は以下のとおりです。

BTN(LEFT)
キー指定

キー指定	LEFT…左矢印キー(←) RIGHT…右矢印キー(→) UP…上矢印キー(↑) DOWN…下矢印キー(↓) SPACE…スペースキー
返回值	キーが押されている=1 キーが押されていない=0

指定したキーが押されているとBTN 関数の値が 1、押されていないと 0 になります。

その BTN 関数の値を、IF (イフ) 命令で判断します。

```
IF  BTN(LEFT)=1  THEN  X=X-1  ELSE  ~
      条件式                条件が成り立つ      条件が成り
                          時に実行                立たない
                                                                時に実行
```

条件式	条件を判断する式。
THEN (ゼン) の後	条件が成り立つ時に実行するプログラム。
ELSE (エルス) の後	条件が成り立たない時に実行するプログラム。 ELSE 以下は省略可能。

110 行は、「もし BTN(LEFT) が 1 だったら (= 左矢印キーが押されていたら)、変数 X を 1 減らす」という意味になります。

なお「X=X-1」は、「X から 1 を引いて、それを X に入れる」という意味です。「X と X-1 が等しい」という意味ではないので注意してください。

同様に 120 行では、「もし右矢印キーが押されていたら、X を 1 増やす」処理をしています。

最後に 150 行では、続けてキャラクターを動かすため、GOTO (ゴートゥー) 命令で前へ戻しています。

```
GOTO  80
      行番号
```

GOTO 命令は、指定した行番号へ実行を移します。

今のプログラムのままだと、キャラクターがちらついて見づらいので、時間待ちを入れて、動きを遅くします。

```
10  '*SKI*
20  CLS:CLV
30  LOCATE 0,0
40  PRINT S
50  X=15:Y=3
60  LOCATE X,Y
70  PRINT CHR$(249)
80  '---GAMELOOP---
90  LOCATE X,Y
100 PRINT " "
110 IF BTN(LEFT)=1 THEN X=X-1
120 IF BTN(RIGHT)=1 THEN X=X+1
130 LOCATE X,Y
140 PRINT CHR$(249)
149 WAIT 5
150 GOTO 80
```

5/60秒待つ

プログラムを実行してみましょう。だいぶキャラクターの動きが見やすくなります。

プログラムの途中で時間待ちをするには、WAIT(ウェイト)命令を使います。

WAIT 5
待ち時間

待ち時間	60分の1秒単位で指定。60=1秒。
------	--------------------

待ち時間の値を変えると、キャラクターが動く速度が変わります。
いろいろ変えて試してみましょう。

●キャラクターのはみ出しチェック

このプログラムだと、以下の問題があります。

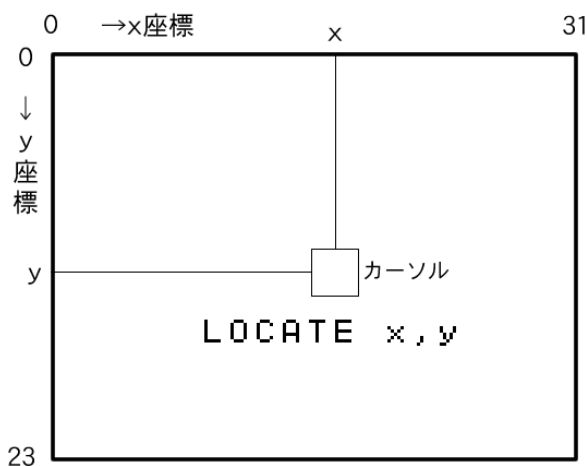
- キャラが画面左はじへ行っても左矢印キーを押し続けると、キャラは動かないが、次に右へ動かそうとすると、左矢印キーを押していたのと同じ時間だけ右矢印キーを押し続けないと、右へ動かない。
- 右はじでも同様。

これは、キャラの横座標 X の範囲を考えずに増やしたり減らしたりしてしまい、 X がマイナスの値になってしまったりするからです。

画面サイズを考えると、キャラを動かす範囲は、 x 座標が $0 \sim 31$ の間にしないといけません。

キャラが画面からはみ出さないように、プログラムを改造します。

【IchigoJam画面：32文字×24行】



…(前略)…

```

80  /---GAMELOOP---
90  LOCATE X,Y
100 PRINT " "
110 IF BTN(LEFT)=1 AND X>0 THEN X=X-1
120 IF BTN(RIGHT)=1 AND X<31 THEN X=X+1
130 LOCATE X,Y
140 PRINT CHR$(249)
149 WAIT 5
150 GOTO 80

```

はみ出さないように条件を追加

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

```
110 IF BTN(LEFT)=1 AND X>0 THEN X=X-1
```

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

両方の条件が成り立った時だけ、「THEN」以下の命令が実行されます。

画面の左端に来ると、 X が0になるので、条件が成り立たなくなり、それ以上左へ行きません。

同じように、画面の右はじでもはみ出さないように、AND 条件式を入れます。

```
120 IF BTN(RIGHT)=1 AND X<31 THEN X=X+1
```


●ポールを登場させる

自分のキャラクターの動きができたので、下からポールを登場させてみましょう。
ポールを表示する部分をサブルーチンにして呼び出すようにします。

```

10  ^*SKI*
20  CLS:CLV
30  LOCATE 0,0
40  PRINT S
50  X=15:Y=3
60  LOCATE X,Y
70  PRINT CHR$(249)
75  P=12
80  ^---GAMELOOP---
90  LOCATE X,Y
100 PRINT " "
110 IF BTN(LEFT)=1 AND X>0 THEN X=X-1
120 IF BTN(RIGHT)=1 AND X<31 THEN X=X+1
125 GOSUB 200
130 LOCATE X,Y
140 PRINT CHR$(249)
142 LOCATE 0,0
144 PRINT S
149 WAIT 5
150 GOTO 80
200 ^---POLE---
210 C=C+1
220 IF C<10 THEN LOCATE 0,23:PRINT:RET
URN
230 P=P+RND(3)-1
240 IF P<0 THEN P=0
250 IF P>24 THEN P=24
260 LOCATE P,23
270 PRINT "|_ _ _ _ _|"
280 C=0
290 RETURN

```

ポールの最初の位置を設定

ポール表示のサブルーチンを呼ぶ

スコアを再度表示

ポール表示のサブルーチン

ポールを10回に1回表示する。表示しない時は改行だけして戻る

ポールの位置を、左右に1ずつの範囲で移動

ポールの位置が画面をはみ出していたら戻す

ポールを表示。これで画面が1行上へスクロール

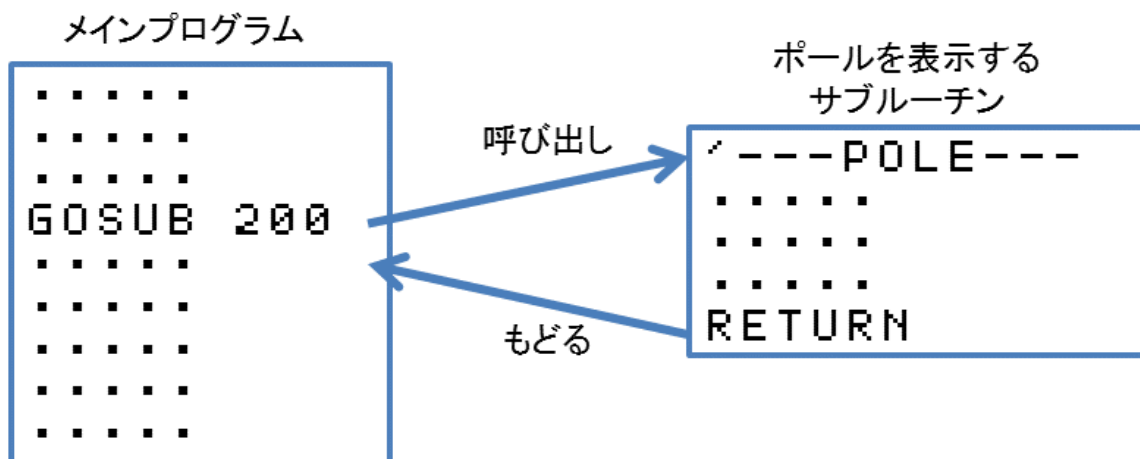
改行のカウントを0に戻す

メインループへ戻る

プログラムを実行してみましょう。
ポールが下から出てきます。



125 行では、GOSUB (ゴースブ) 命令でポールを表示するサブルーチンを呼び出します。



メインプログラムからは GOSUB 命令でサブルーチンへジャンプし、サブルーチンからは RETURN (リターン) 命令でもどります。もどった後は、GOSUB 命令の続きへプログラムの処理が移ります。

サブルーチンに分けると、プログラムがすっきりしてわかりやすくなります。また、何度も同じサブルーチンを呼び出して使うことができます。

```

GOSUB 200
      行番号

```

行番号	サブルーチンの行番号。その行番号へジャンプする。 RETURN 命令でもどってきた時は、GOSUB の次の命令へ移る。
-----	--

```

RETURN

```

呼び出した GOSUB の次の命令へもどる。

● 点数、ミス

ポールを通ったら点数が入るようにしましょう。

ポールを通ったかどうかは、プレイヤーの位置に上がって来た文字を読み取って、ポールの間にある「_」(アンダースコア)かどうかで判断します。

```

10  ' *SKI*
...(中略)...
80  ' ---GAMELOOP---
90  LOCATE X,Y
100 PRINT " "
110 IF BTN(LEFT)=1 AND X>0 THEN X=X-1
120 IF BTN(RIGHT)=1 AND X<31 THEN X=X+1
125 GOSUB 200
127 D=SCR(X,Y)
130 LOCATE X,Y
140 PRINT CHR$(249)
141 IF D=95 THEN BEEP:S=S+1
142 LOCATE 0,0
144 PRINT S
149 WAIT 5
150 GOTO 80
...(後略)...

```

キャラクターの位置に上がってきた文字を読み取る

もしポールを通過したら
BEEP音を出して
スコアを1増やす

プログラムを実行してみましょう。
ポールを通過すると、点数が増えます。



画面に表示されている文字を読み取るには、**SCR**(スクリーン)関数を使います。

```
SCR( X , Y )
      x座標  y座標
```

x座標	読み取りたい場所の x 座標。
y座標	読み取りたい場所の y 座標。
返り値	その場所に表示されている文字の文字コード。 その場所に何も表示されていない時や、座標が画面をはみ出している時は「0」が返る。

127 行で、自分のキャラクターの位置に上がってきた文字を読み取って、文字コードを変数 D に入れています。

ポールの間の「_」は文字コードが 95 なので、141 行の IF 命令で D が 95 かどうかを判断して、もしそうだったら **BEEP**(ビーブ)命令で音を出して、スコアを加算しています。

```
BEEP 30 , 30
      音の高さ 音の長さ
```

音の高さ	値が小さいほど高く、大きいほど低くなる。省略すると標準の高さになる。
音の長さ	1/60 秒単位で指定。60=1 秒。省略すると標準の長さになる。

逆に、両端のポールに当たるとミスになるようにしてみましょう。

```

…(前略)…
141 IF D=95 THEN BEEP:S=S+1
142 LOCATE 0,0
144 PRINT S
146 IF D=124 THEN GOTO 160
149 WAIT 5
150 GOTO 80
160 '---GAMEOVER---
170 LOCATE X,Y
180 PRINT "X"
190 BEEP 30,30
195 END
200 '---POLE---
…(後略)…

```

もしポールに当たったら、ミスの処理へジャンプ

自分のキャラクターの位置に「X」を表示

ミスした音を出す

プログラムを終了する

プログラムを実行してみましょう。
両端のポールに当たると、ミスになってゲームが終了します。



両端のポール「|」は文字コードが 124 なので、146 行で画面から読み取った文字コード(変数 D)が 124 かどうかを判断して、もしそうなら 160 行のミスの処理へジャンプします。160 行からのミスの処理では、自分のキャラクターの位置に「X」を表示して、ミスした音を出し、END(エンド)命令でプログラムを終了しています。

END プログラムを終了します。

●ゲームのレベルアップ

ずっと同じ状態でゲームが続くと面白くありません。

点数が上がっていくと、だんだんポールがせまくなるようにしましょう。

```
10  /*SKI*
20  CLS:CLY
30  LOCATE 0,0
40  PRINT S
50  X=15:Y=3
60  LOCATE X,Y
70  PRINT CHR$(249)
75  P=12
77  W=5
80  /---GAMELOOP---
90  LOCATE X,Y
100 PRINT " "
110 IF BTN(LEFT)=1 AND X>0 THEN X=X-1
120 IF BTN(RIGHT)=1 AND X<31 THEN X=X+1
125 GOSUB 200
127 D=SCR(X,Y)
130 LOCATE X,Y
140 PRINT CHR$(249)
141 IF D=95 THEN BEEP:S=S+1
142 LOCATE 0,0
144 PRINT S
146 IF D=124 THEN GOTO 160
147 W=5-S/10
148 IF W<1 THEN W=1
149 WAIT 5
150 GOTO 80
160 /---GAMEOVER---
170 LOCATE X,Y
180 PRINT "X"
190 BEEP 30,30
195 END
```

…(↓次ページへ続く↓)…

最初にポール間を5に設定

スコア10点ごとにポール間Wを1減らす。
1より小さくなると通過できないので、1にもどす。

…(↓前ページから続く↓)…

```

200 ' ---POLE---
210 C=C+1
220 IF C<10 THEN LOCATE 0,23:PRINT:RET
URN
230 P=P+RND(3)-1
240 IF P<0 THEN P=0
250 IF P+W>28 THEN P=28-W
260 LOCATE P,23
270 PRINT "I";
272 FOR I=1 TO W
274 PRINT "-";
276 NEXT
278 PRINT "I"
280 C=0
290 RETURN

```

ポール幅 W を使ったはみ出しチェックに変更

ポール幅 W を使ったポール表示に変更

プログラムを実行してみましょう。
スコアが上がると、だんだんポールがせまくなります。



このプログラムでは、ポールの幅を変数 W で設定し、W の値は 147~148 行で変更しています。

147 行「 $W = 5 - S / 10$ 」では、

- スコア S が 0~9 点…IchigoJam は小数の計算ができないので、 $S/10$ は 0 になる。
よって、 $W = 5 - 0 = 5$ となる。
- S が 10~19 点……… $S/10$ は 1 になるので、 $W = 5 - 1 = 4$ となる。
- S が 20~29 点……… $S/10$ は 2 になるので、 $W = 5 - 2 = 3$ となる。

…と、スコア 10 点ごとに W は減っていきます。

200 行以降のポール表示サブルーチンでは、W を使った表示に変更しています。

PRINT 命令の最後に「;」(セミコロン)を付けると、その表示のあとに改行せず、次の PRINT 命令の文字は後ろに続けて表示されます。

ポールがせまくなる条件を変えたりして、いろいろ改造してみましょう。

ほかにアイテムを追加して、通過するとボーナス点が入るようにしてもおもしろいでしょう。