マイクロワールドEX
MicroWorlds EX™

利用の手引

3 章

重要な概念とテクニック

1. プロセスの管理

・例示された命令は、コマンドセンターに入力して試します。Enter キーを押すことで命令が実行されます。(*コマンドセンターで命令を実行しないで次の行に移りたい場合は、Ctrl を押しながら Enter キーを押します。)
・例示された手順は、手順タブエリアに入力します。コマンドセンターに手順名入力し、Enter キーを押すことで手順が実行されます。

マイクロワールド EX では、複数のオブジェクトの制御や、様々な命令の実行が同時に行えます。例えば、ボタンをクリックしてあることを行いながら、カメをクリックして、また別のことを行うことができます。鳥を飛ばしながら、曲を奏でるというように。コンピュータの世界では、これを並列処理と呼んでいます。

(1)命令の実行と並列処理

ロゴは1つの命令を“読み取り”、その命令を実行してから、次の命令の読み取りに進みます。コマンドセンターに命令を入力して、その命令を実行するには、最後に Enter キーを押します。Enter キーを押すと、命令行の最後に大きな黒の点が現れ、マイクロワールド EX がビジー状態になって、命令を実行していることを示します。ごく特殊な場合を除いて、現在の命令の実行が終わるまで、コマンドセンターが別の命令を受け付けることはできません。

コマンドセンターに次の命令を入力してみてください。(ページ上に、あらかじめ2つのカメを作ってから実行します)。

かめ1、　くりかえす　100　「前へ　1」　かめ2、　くりかえす 100　「前へ　1」

最後に Enter キーを押します。

2つの「**繰り返す**命令」が順に実行されます。
これらのシーケンシャルな流れを変更する基本用語は、**始める**、**無限に**、**次の時に**の3つです。

始めるは、後の2つの基本用語の土台となっているともいえます。実際、この後すぐに説明するように、**無限に**と**次の時に**は、始めるを含む手順と考えることもできます。

始めるを使うということは、独立したプロセスを起動するということです。これはコマンドセンターで確認できます。**始める**を使った命令を入力すると、行の最後の黒い点がすぐに消えて、別の命令をただちに実行することができます。

始めるは、インプットに命令リストを取ります。
一見、ごく普通の命令リストのようですが、命令の1つ1つを**独立したプロセス**として実行します。マイクロワールド EX は、それらの命令の実行が終了するのを待たず、次の命令に進みます。

始めるをコマンドセンターで試してみましょう。

```
かめ1、 始める 「くりかえす　5000 「前へ　1」」
かめ2、 始める 「くりかえす　5000 「前へ　1」」
```

２行を入力し終わったら、それぞれの行の最後で**Enterキー**を押します。

1行目を読み取ると、黒い点はすぐに消え、命令の実行が終わらなくても、次の命令を読み取ります。
つまり、**始める**は、完了するまでに長い時間がかかる命令を起動する場合に使います。いったん命令が起動されると、別のことを行うことができます。

もう１つの例を紹介しましょう。

```
かめ1、ペンを下ろす
くりかえす　360 「前へ　1」 くりかえす　360 「右へ　1」
```

命令は一度に１つずつ読み取られます。カメは360歩前進し、完全に１回転します。同じ命令を**始める**を使って試してみましょう。

```
始める 「くりかえす 360 「前へ　1]] 始める 「くりかえす　360 「右へ　1」」
```

カメによって円が描かれます。

幾何学に関心があるのなら、このプロセスをさらに詳しく見てみましょう。描かれた円は、次のコマンドで描かれた円と同じでしょうか。

```
くりかえす　360 「前へ　1　右へ　1」
```

上の**始める**の命令では、２つの独立したプロセスを起動しました。

無限には、実際には**始める**から派生した基本用語です。次の2つの命令は同じ意味になります。

```
無限に 「前へ　1」
始める 「くりかえす　99999 「前へ　1」」
```

無限には、命令を繰り返し実行します。その間、マイクロワールドEXは別の命令を実行することができます。**無限に**はプロセスを１つだけ起動します。

無限にの実行は、次のいずれかの方法で止めることができます。
・ツールバーの［すべてを止める］をクリックする。
・［編集］メニューから［すべてを止める］を選択する。
・コマンドセンターから**取り消す　「無限に」**を入力する。

命令が並列に実行されていることを確認するための実験をしましょう。
この実験を通じて、ロゴにおいてプロセスがどのようなものであるかも分かります。

コマンドセンターで、次の3行の**無限に**命令を入力してください。それぞれの行の最後でEnterキーを押します。

```
無限に 「前へ　1」
無限に 「右へ　1」
無限に 「左へ　1」
```

カメがまっすぐに前進します。

ロゴでは、MIMD（Multiple Instruction Multiple Data）パラダイムが採用されています。上記の例では、各プロセスのコマンドが同時に実行され、命令の同期が取られます。

独立したプロセスを起動する残る最後の基本用語は、**次の時に**です。**次の時に**も、**始める**コマンドを使って説明することができます。次の2つの行は同じ働きをします。

```
次の時に 「縦の位置　>　100」「後ろへ　100」
始める 「くりかえす　99999 「待機する 「縦の位置　>　100」 後ろへ　100 」」
```

次の時にのインプットは、2つの命令リストです。
最初のリストは、**真**または**偽**を報告する条件文です。この条件が真の場合には、2つ目の命令リストが実行されます。(2つ目の命令リストは、1回だけ実行されることに注意してください。繰り返し実行されることはありません。)

始めるをもとに**次の時に**というコマンドを解剖すると、条件文が繰り返し実行されるということがわかります。独立したプロセスとして起動されるわけですから、同時に他の命令を実行することができます。次の命令を試すことによって、**次の時に**の働きを見てみましょう。

```
次の時に 「縦の位置　>　100」「後ろへ　100」
くりかえす　1000 「前へ　1」
```

最初の行は、現在のカメのy座標値が100より大きいかどうかを調べるプロセスを起動します。この条件が真になると、カメを100歩後退させます。2行目は、カメを前進させる命令を繰り返し実行します。

次の時には、1つのプロセスしか起動しないことに注意してください。**次の時に**のプロセスを止めるには、次のいずれかの方法を使います。

- ツールバーの［すべてを止める］をクリックする。
- コマンドセンターから**取り消す　「縦の位置　>　100」**を入力する。
  （**取り消す**のインプットに**次の時に**の1つ目のインプットを入力します）。

別の命令が実行されていない場合、視覚的に確認できるものがないため、**次の時に**命令を使う場合には注意が必要です。
実行を取り消さなかったために、後でこの命令が残っていて、思わぬ結果をまねくことになります。

(2) ボタンとプロセス

プロセスの一団は、ファミリーと考えることができます。命令を実行するボタンをクリックすると、1つの**プロセスファミリー**が起動します。ファミリーのプロセスは、そのボタンが再度、押されない限り動作し続けます。
このときでも、別のボタンをクリックしたり、コマンドセンターに命令を入力したりすることができます。命令を設定したカメをクリックすると、カメがそのプロセスファミリーを起動します。
ボタンおよびカメのダイアログボックスでは、命令を［一回］または［無限に］実行するように設定することができます。［無限に］は、コマンドの**無限に**と同様、単に命令を繰り返し実行するだけです。［一回］は、コマンドの**始める**と似た働きをします。

ボタン

ボタンについて見てみましょう。2つのボタンを使い、コマンドセンターで起動したのと同じプロセスを起動することができます。次のことを試してください。

ボタンを2つ作ります。
1つ目のボタンには、命令として**くりかえす　360 「前へ　1」**を設定し、回数には［一回］を選択します。
2つ目のボタンには、命令として**くりかえす　360 「右へ　1」**を設定し、回数には［一回］を選択します。
(これらのボタンは横長なボタンにして、命令を見やすいようにサイズを調整しておきます)。

コマンドセンターから

```
かめ1、 ペンを下ろす
```

と入力して、カメ1をアクティブなカメにし、ペンを下げた状態にします。
ここで、2つの［くりかえす］ボタンを同時にクリックします。

・2つのボタンを同時にクリックするには、両方のボタンを選択して、一方をクリックします。
これで、両方のボタンを同時に押すことができます。

コマンドセンターを使った前の例で描いたのと、まったく同じ円が描かれることが分かります。命令の実行中、ボタンは押されているように見えます。命令の実行が終わると、ボタンは元の状態に戻ります。

これは、コマンドセンターで次の命令を入力したのと同じことです。

```
始める 「くりかえす　360 「前へ　1」」 始める 「くりかえす　360 「右へ　1」」
```

次に、さらにボタンを2つ作ります。
1つ目のボタンには、**前へ　1**を設定して、回数には［無限に］を選択します。
2つ目のボタンには、**右へ　1**を設定して、回数には［無限に］を選択します。

2つのボタンを同時にクリックして、プロセスを起動してみましょう。次のいずれかを行わない限り、ボタンはいつまでも押された状態に留まります。
・再びボタンをクリックする。
・ツールバーから［すべてを止める］をクリックする。
・コマンドセンターから**取り消す**を使う。

プロセスという点では、これらの2つのボタン（**前へ　1**と**右へ　1**）は、先に作った2つの［**くりかえす**］ボタンと同じです。

コマンドセンターに命令を入力することによって起動できるのは、1つのプロセスファミリーだけです。これに対し、ボタンは1つ1つが異なるプロセスファミリーを起動します。このため、同時に複数のプロセスファミリーを実行することができます。（ただし、同時に実行できるプロセス数は最大で40個です。）

(3) カメとクリックオン

カメも、ボタンと同じようにプロセスを起動します。

ただし、ボタンは、そこから起動したプロセスによって、ボタン自体の状態が変わることはありませんが、カメは、自身が起動した命令によってしばしばカメの状態（位置、向きなど）に影響します。

次の例では、2つのカメが花を描きます。それぞれのカメが独自のプロセスを起動します。
手順タブエリアで次の手順を作ってください。

```
手順は　弧　：a　：d
くりかえす　：a　/　2「前へ　：d　右へ　2」
終わり

手順は　花びら　：n
くりかえす　2「弧　90　：ｎ　右へ　90」
終わり

手順は　花1
左へ　45
くりかえす　5「花びら　1　右へ　360　/　5」
右へ　45
終わり

手順は　花2
左へ　45
くりかえす 6「花びら 1　右へ　360　/　6」
右へ　45
終わり
```

続いて、2つのカメを作ります。それぞれのカメのペンは下げた状態（**ペンを下ろす**）にしてください。
・一方のカメには、命令として**花 1** を設定し、実行モードとして［一回］を選択します。
・もう一方のカメには、命令として**花 2** を設定し、実行モードとして［一回］を選択します。

最後にカメをクリックして、それらのカメのプロセスを起動します。

クリックオン

カメでプロセスを起動するもう一つの方法として、**クリックオン**というコマンドを利用する方法があります。クリックオンを使うには、カメのダイアログボックスに命令が設定されている必要があります。

クリックオンを使えば、コマンドセンターや手順からプロセスを起動することができます。コマンドセンターに次の命令を入力して、試してみましょう。

```
全部 「クリックオン」
```

両方のカメが花を描き始めます。**全部 「クリックオン」**のパワーは、カメがそれぞれ異なる仕事をするようにプログラミングされるアニメーションの場面でもっと明らかになります。

ヒント　：2つのカメのペンを下げた状態にするには、
・それぞれのカメについて、コマンドセンターから**ペンを下ろす**を実行する。
・**全部 「ペンを下ろす」**で一度に2つのカメの状態を変更する。

という方法があります。
カメ2については、コマンドセンターから　**色は　"あか**として色を変えてあります。

(4) プロセスを止める

ボタンやカメから起動したプロセスは、次のいずれかの操作を行うことによって止めることができます。

・カメまたはボタンを、もう一度クリックする（命令が実行中の場合)。
・ツールバーの［すべてを止める］をクリックする。
・［編集］メニューから［すべてを止める］を選択する。

また、コマンドセンターや手順からプロセスを止める基本用語は、次の4つあります。

クリックオフ
取り消す
これを止める
設定する

クリックオフ

クリックオフは､カメをクリックして止めるのとまったく同じ働きをします｡**クリックオフ**は、カメから起動したプロセスしか止めません。

カメとクリックオンの花を描くプログラムでは､次の命令を入力することによって､カメのプロセスを止めることができます。

```
全部 「クリックオフ」
```

取り消す

取り消すは､もっと微妙な働きをします｡最初に､**始める**､**無限に**､**次の時に**のいずれかを使った命令で起動したプロセスを見てみましょう。
取り消すのインプットは､起動したコマンドのインプットになっている命令と同じでなければなりません（**次の時に**の場合は1つ目のインプット)｡簡単な例で､**取り消す**の働きを見てみましょう。

＜**無限に** で起動したプロセスを取り消す＞

カメを作ります。
コマンドセンターに次の命令を入力します。

```
無限に 「前へ　1」
```

命令を実行し、しばらくしたら次の命令を入力します。

```
取り消す 「前へ　1」
```

カメが止まります。

＜**次の時に**で起動したプロセスを取り消す＞

次の命令を入力してください。

```
無限に　「前へ　1」
次の時に 「縦の位置　>　100」「後ろへ　100」
```

次の時にで起動したプロセスをキャンセルするには、次の命令を入力します。

```
取り消す 「縦の位置　>　100」
```

実行モードが［一回］または［無限に］のどちらであるかに関係なく、ボタンやカメから起動されたプロセスも、**取り消す**を使って止めることができます。例えば、次のような具合です。

・ボタンを作り、命令として**前へ 1**を入力します。
・実行モードとして「無限に」を選択します。
・ボタンをクリックします。カメが前進します。
・コマンドセンターに次の命令を入力します。

```
取り消す　「前へ　1」
```

上記の**取り消す**の例では、そのインプットがプロセスの名前ではなくプロセスの命令であることに注意してください。

これを止める

これを止めるは、内部からプロセスを止めます。例えば、次のような具合です。

```
無限に　「前へ　1　もし （距離　　”かめ２）　>　100　「これを止める」」
```

設定する

ボタンやカメは、**設定する**というコマンドを使って止めることもできます。これは、命令実行中のボタンを再度クリックして止める操作と同じ働きをします。

```
設定する　”ボタン１　”はじまったか　”うそ
```

次の命令は、ボタン１をクリックして、そのプロセスを起動するのと同じです。

```
設定する　”ボタン１　”はじまったか　”ほんとう
```

設定するは、次の例に示すように、ボタンの命令の設定に使うこともできます。

```
設定する　”ボタン１　”めいれい　「始める 「花1」」
```

(5) タイミングと同期

プロセスを個々に取り消しすることによって、プロセスのタイミングと順序を制御することができます。

無限にと取り消す

2つのプロセスを同期させる方法としては、最初の命令として**無限に**を実行してから、2つ目の命令を起動し、2つ目の命令が終了したときに1つ目の命令を取り消すという方法があります。例えば、次のような具合です。
(＊これらの手順を試すには、［ファイル］メニューの［インポート］を使ってChopinというMIDIファイルをページに読み込んでおかなければなりません。)

```
手順は　トリック
無限に　「点滅」　　"点滅"という子手順を繰り返し実行します。
canon　　"点滅"を実行中にcanonというMIDIファイルの音楽を再生します。
取り消す　「点滅」　"点滅"を止めます。
終わり
```

```
手順は　点滅
出てくる　待つ　1　隠れる　待つ　1
終わり
```

点滅は、カメを表示したり隠したり、交互に切り替えます。**トリック**という手順の中の**無限に**は、**点滅**を繰り返し実行します。**canon**は、音楽を再生します。再生が終わると、次の命令である**取り消す**を実行します。
取り消すに対するインプットは、必ず**無限に**のインプットと同じにします。

クリックオンとクリックオフ

アニメーションと音楽の同期は、**クリックオン**と**クリックオフ**を使って行うこともできます。例えば、かめ1というカメを作って、ダイアログボックスに**点滅**を設定し、回数には［無限に］を選択します。音楽を再生して、アニメーションと同時に終了するようにするには、コマンドセンターに次の命令を入力し実行ます。

```
かめ1、 クリックオン　canon　クリックオフ
```

別の例を紹介しましょう。
ここでは、**待機する**および**終わったか**を使って、命令を実行するタイミングを制御します。**終わったか**は、**始める**または**無限に**で起動したプロセスが完了した場合に**真**を報告します。**待機する**は、インプットの命令から**真**が報告されると待機を解きます。

ページに新しいカメを作ります。名前は**かめ2**とします。このカメのダイアログボックスには、命令として**点滅**を設定し、回数には［無限に］を選択します。続いて、**円**と**合図**の手順を作ります。

(* これらの手順を試すには、［ファイル］メニューの［インポート］を使って Chopin という MIDI ファイルをページに読み込んでおかなければなりません。)

```
手順は　点滅
出てくる　待つ　1　隠れる　待つ　1
終わり

手順は　円
前へ　5　右へ　5
終わり

手順は　合図
クリックオン
始める　「Chopin」
無限に　「円」
待機する 「終わったか　「Chopin」」
クリックオフ
取り消す 「円」
終わり
```

では、コマンドセンターから次の命令を入力し実行ます。

```
かめ2、 合図
```

(6) グローバルなカメ

作品やページに複数のカメがある場合、命令の対象となるカメを確認したり、指定する必要があります。

グローバルなカメ

ページには、必ず"**グローバルなカメ**"(コマンドセンターから実行される命令を受け取るアクティブなカメ) が1つ存在します。
"グローバルなカメ"になるのは、次のカメです。

- 最後に作られたカメ
- **かめは** を使い、コマンドセンターから最後に指定されたカメ
- ページ上でクリックされるか、または**クリックオン** を使って最後に指定されたカメ
- **カメに聞く**で最後に指定されたカメ

ボタンやカメからプロセスを起動しても、**グローバルなカメ**が変わることはありません。起動されたプロセスは、コマンドセンターからグローバルなカメを受け継ぎます。プロセスは、自身のグローバルなカメを変更することはできますが、コマンドセンターの“グローバルなカメ”を変更することはできません。

次のようにして確認してみましょう。
新しいページに**かめ1**と**かめ2**という2つのカメを作ります。コマンドセンターに次の命令を入力します。

```
下へ書く　今のカメ

かめ 2
```

かめ2を後で作成したので、グローバルなカメとして**かめ2**が報告されます。
次にボタンを作ります。ダイアログボックスに、命令として**かめ1**を設定し、回数として「一回」を選択します。
ボタンをクリックしたあとで、コマンドセンターから再び`下へ書く　今のカメ`を実行します。
答えはやはり**かめ2**です。
ボタンによって、**グローバルなカメ**が変更されることはありません。
(複数のプロセスを実行しているときでも、グローバルなカメを記憶しておくことができるのは、大変重要なことです)。
以下に、プロセスにおけるグローバルなカメに関する“ルール”をまとめておきます。

ルール

- コマンドセンターからの命令を受け取る"グローバルなカメ"が存在します。
- 各プロセスには、それぞれにアクティブなカメが存在します。
- 各プロセスが別のプロセスのアクティブなカメを変更することはできません。変更できるのは、自身のアクティブなカメだけです。
- プロセスは、ページごとに自身のアクティブなカメを保持します。

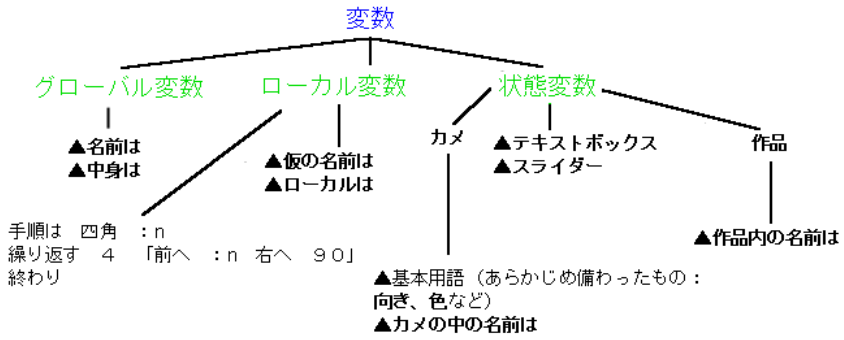
一般に、プロセスを起動するときにグローバルなカメを変更することはありません。しかし、ごくわずかな例で、プロセスの中でグローバルなカメを変更する必要が出てくることがあるのです。この時には、**カメに聞く**を使います。
カメに聞くは、**グローバルなカメ**を設定します。
つまり、このコマンドを使えばボタンから起動したプロセスの中でグローバルなカメを変更することができます。

2. 変数のテクニック

■この節の内容はオンラインヘルプからも見ることができます。
EX　　→メニューバーの［ヘルプ］をクリック→［リファレンス］
EX（ジュニア）　→メニューバーの［ヘルプ］をクリック→［リファレンス］

・例示された命令は、コマンドセンターに入力して試します。Enterキーを押すことで命令が実行されます。(*コマンドセンターで命令を実行しないで次の行に移りたい場合は、Ctrl を押しながら Enter キーを押します。)
・例示された手順は、手順タブエリアに入力します。コマンドセンターに手順名入力し、Enter キーを押すことで手順が実行されます

データに名前を付けたり、データを記憶したりする手段として変数が利用されます。マイクロワールド EX の変数には、グローバル変数、ローカル変数、状態変数の 3 種類があります。
グローバル変数は、データに名前を付けるときに使われます。名前は、データの入れものに貼られたラベルと見なすことができます。ローカル変数は、手順に対するインプットに使われ、親と子の手順の間の情報をやりとりする手段になります。
状態変数は情報を提供します。情報変数を使い、カメやテキストボックス、スライダー、作品などのいろいろなオブジェクトの状態を変更することができます。オブジェクトには、新しい状態変数を追加できるものがあります。



(1) グローバル変数

グローバル変数は、**名前は** または**中身は**のいずれかを使って作成します。
例えば、1 つのページに 6 つのカメがあり、これらのカメのうちの 3 つを A チーム、残る 3 つを B チーム　と分けて名付けたいとします。このためには、次の命令を使います。

名前は　”A チーム　「かめ 1　かめ 2　かめ 3」
名前は　”B チーム　「かめ 4　かめ 5　かめ 6」

命令の中では、変数を次のように使います。
名前のすぐ前にコロン（：）を付けると、その名前によって示されるもの（値）が必要であるという意味になります。

```
下へ書く　:Aチーム
かめ1　かめ2　かめ3
```

これで、Aチームと Bチームを別々に扱うことができます。

```
かめは　：Aチーム
前へ　100
かめは　：Bチーム
右へ　90
```

グローバル変数の値は、マイクロワールドEXを終了するか、コンピュータの電源を切るまでコンピュータのメモリに留まります。
このことは、別の作品に切り替えても、Aチームと Bチームがその値を保持していることを意味します。

もう1つの例を紹介しましょう。いくつかの音声に名前を付けて、実行するたびにランダムに音声が選ばれるようにしたいとします。
(作品に音声を読み込む必要があります)。

```
名前は　”サウンド　「cheers5　happy　flute　shout」
```

これで、サウンドという変数を**どれか**(レポータ)のインプットとして利用することができます。

```
下へ書く　どれか　:サウンド
```

flute

やるというコマンドを使って、ランダムに音声を再生してみてください。

```
やる　どれか　：サウンド
```

この例に見られるように、グローバル変数は、コマンドやレポータのインプットにすることができます。上記の変数の値はリストですが、他の種類のデータでもかまいません。

数字の例

```
名前は　"x　3
下へ書く　:x
```

3

```
下へ書く 平方根　:x
```

1.732051

単語の例

```
名前は　”あいさつ　”おはよう
下へ書く：あいさつ
おはよう
```

ロングワードの例

```
名前は　”文章　"|では　皆さん　お元気で|
下へ書く　”文章
```

では　皆さん　お元気で

ロゴのレポータを使って、変数の値を知ることができます。
(**質問**は質問ダイアログボックスを表示する特殊なコマンドです)。

```
質問 「お元気ですか」
名前は　”返事　答え
下へ書く:返事
```

とっても

名前はでは、それ自身が他の変数の値である変数を作ることができます。

```
名前は　”動物　”猫
```

名前はの最初のインプットは文字通りの単語ですから、通常、引用符を付けます。

```
下へ書く　:動物
```

猫

動物の値は**猫**です。
続いて、動物の値（猫）を別の名前にしてみましょう。

```
名前は　:動物　”チビ
```

上記の命令は、次の命令と同じことです。

```
名前は　”猫　”チビ
```

確認するには、次の命令を入力します。

```
下へ書く　:動物
```

猫

```
下へ書く　:猫
```

チビ

次の例に示すように、レポータを使って、変数名を作ることもできます。

```
名前は　ワード　”住所　1「609　西町」
下へ書く　:住所1
609　西町
```

上記の命令は、次の命令と同じことです。

```
名前は　”住所1「609　西町」
```

(2) ローカル変数

ローカル変数は、手順のタイトル行に**インプット**として付けられています。ローカル変数は、その変数が定義されている手順が実行されている間だけ値を保持します。このため、ローカル変数は一時的な結果を保存する優れた手段になります。手順タブエリアに次のように入力します。

```
手順は　歓迎　:a
下へ書く　文「お会いできてうれしいです」:a
終わり
```

コマンドセンターから、次のように入力し、Enterキーを押して実行します。

```
歓迎　洋子
```

以下のように表示されます。

お会いできてうれしいです　洋子

インプットの名前はそれぞれの手順に固有です。インプットの名前を使い、親手順は子手順に情報を渡すことができます。この好例は、子手順が独自のインプット値を持つ、次の例のような再帰手順で見られます。

```
手順は　数える　:n
もし　:n　>　5「止まる」
下へ書く　:n
数える　:n　+　1
下へ書く　:n
終わり
```

コマンドセンターから、次のように入力し、Enterキーを押して実行します。

```
数える　1
```

ロゴには、ローカル変数を作るための基本用語が2つあります。**ローカルは**、**仮の名前は**です。**ローカルは**を使ってローカル変数を作るには、**名前は**と組み合わせます。

```
手順は　紹介
ローカルは　”人名
質問 「あなたの苗字は？」
名前は　”人名　答え
歓迎　：人名
終わり
```

紹介 は、**歓迎** を呼び出します。**歓迎**のインプットは、**名前は**によって設定されます。**ローカルは** は変数名がローカルであることを指示しますから、手順が終了すると、**：人名** の値は失われます。

コマンドセンターから、次のように入力し、Enter キーを押してみましょう。

```
下へ書く　：人名
```

以下のように表示されます。

人名 という名前の変数はありません。

仮の名前は は、ローカル変数を作るもう1つの基本用語です。**仮の名前は** は、**ローカルは** と **名前は** を1つにまとめたものを考えることができます。**ローカルは** を使った上の例は、次のように書き直すことができます。

```
手順は　紹介
質問 「あなたのお名前は？」
仮の名前は 「人名　答え」
歓迎　：人名
終わり
```

1つの**仮の名前は** で、一度に複数のローカル変数を定義することができます。

```
手順は　動き
仮の名前は 「方向　90　距離　100　スピード　1」
向きは　：方向
すべる　：距離　：スピード
終わり
```

上記の例では、方向、距離、スピードの3つのローカル変数を設定しています。

仮の名前は を使うと、簡潔に値を設定することができますが、構造が少し複雑です。**仮の名前は** はプログラミングに少し慣れてから使うほうが良いかもしれません。

(3) ローカル変数とグローバル変数の操作

高度なプログラムになると、手順の中で**ローカル変数**と**グローバル変数**が両方が使われます。1つのサンプルプログラムを分解して、変数がどのような目的に使われているかを詳しく見てみましょう。

ここで紹介するプログラムでは、一群の単語を別の言語に翻訳します。

```
手順は　辞書　：単語　：翻訳
名前は　：単語　：翻訳
終わり

手順は　辞書-保管　：リスト
もし　空か　：リスト　「止まる」
辞書　幾つめ　1　：リスト
幾つめ　2　：リスト
辞書-保管　最初以外　最初以外　：リスト
終わり

手順は　翻訳-保管　：リスト
仮の名前は　「結果　「」」
それぞれをやる　「単語　：リスト」「名前は　”結果　文　：結果　翻訳　：単語」
表示　：結果
終わり

手順は　翻訳　：単語
表示　中身　：単語
終わり
```

この**辞書**という手順では、**名前は**が使われています。ここでグローバル変数が必要なのは、各単語の訳語をプログラムに記憶させ、参照できるようにする必要があるためです。

```
辞書　’hello　’bonjour
下へ書く　：hello

bonjour

下へ書く 翻訳　’hello

bonjour
```

辞書-保管のインプットはリストです。再帰しますから、**辞書-保管**を呼び出すたびに、独自のインプットが保持され、**辞書**に渡されます。**：リスト**は一時的なローカル変数であり、再帰呼び出しのたびに値が変化します。

```
辞書-保管　「the le sky ciel is est blue bleu」
```

全部の名前というレポータは、作成されているグローバル変数を報告します。

```
下へ書く　全部の名前

名前は　"the "le
名前は　"sky "ciel
名前は　"is "est
名前は　"blue "bleu
名前は　"hello "bonjour
```

翻訳-保管 には、インプットが1つと**仮の名前は**式が1つ、それに**名前は**式をインプットとする**それぞれをやる**が含まれています。次を試してください。

```
下へ書く　翻訳 - 保管 「the sky is blue」

le ciel est bleu
```

手順を１行ずつ調べてみましょう。

```
仮の名前は 「結果 「」」
```

この命令は、インプットに空のリストを持つ、結果というローカル変数を作ります。

```
それぞれをやる 「単語　：リスト」「名前は　"結果　文　：結果　翻訳　：単語」
```

ここには、**それぞれをやる**というコマンドがあります。このコマンドによって、命令がループします。このループ構造で、それぞれのローカル変数に値を割り当てることができます。

それぞれをやるは、リスト内の要素ごとに命令セットを実行します。1つ目のインプットは、一時的な変数に対して１つの値範囲を設定します。この例の**単語**は、インプットで与えられたリスト（すなわち、[the sky is blue]）をその値とするローカル変数です。
2つ目のインプットは、命令、すなわち、この例では、**名前は　"結果　文　：結果　翻訳　：単語** を実行します。

それぞれをやるは、リストの[the sky is blue]の要素ごとに名前はを実行します。**結果**には、最後に追加された**単語**の訳語が保持されます。

```
：結果　翻訳　：単語
```

翻訳はインプットを持つレポータであり、**表示　中身　：単語** を実行します。**中身**がそのインプットの値を出力することを忘れないでください。

```
下へ書く　翻訳　"sky

ciel
```

それぞれをやるが実行され、**：結果** の値が [le]になると、**名前は** の値は[le ciel]になります。

このようにして、**表示　：結果** は、**：結果** の最後の値、すなわち、**[le ciel est blue]**を出力します。

:結果 は**仮の名前** で定義しましたから、**翻訳-保管** を実行した後もローカル変数のままです。**それぞれをやる**命令で**名前は**が使われているとしても、**:結果**がローカル変数であることを確認することができます。

```
下へ書く　:結果
```

結果という名前の変数はありません。

(4) 状態変数

マイクロワールドEXの状態変数は、オブジェクトに関する変数です。
カメ、スライダー、テキストボックス、作品という4つのオブジェクトについて状態変数があります。状態変数の構文は、手順や基本用語と同じです。変数であることを示すためのコロン(:)は必要ありません。

a. カメの状態変数

あらかじめ用意された状態変数

カメには、**位置は、位置、向きは、向き、色は、色** などの状態変数があらかじめ用意されています。

```
向きは　90
下へ書く　向き
```

90

カメの状態変数は、**カメの中の名前は**を使って新たに定義することができます。**カメの中の名前は** で、現在の作品のカメの1つ1つに新しい状態変数を作ります。よく作られるカメの状態変数は、**スピード**です。

```
カメの中の名前は　"スピード
```

カメの中の名前は のインプットは、すべてのカメに割り当てられる変数になります。変数を設定した状態では、この変数には、値はありません。カメの変数には、**設定する** と変数名を使って、値を割り当てることができます。

```
かめ1、スピードは　2　　…かめ1というカメの速度を2に設定します。
かめ2、スピードは　10　　…かめ2というカメの速度を10に設定します。
```

この後、これらのカメに**前へ　スピード**という命令を実行させた場合、**かめ2**の動きは**かめ1**より速くなります。

```
頼む「かめ1　かめ2」「くりかえす　200「前へ　スピード」」
```

もう1つ実用的な例を紹介しましょう。**カメの中の名前は** を使って、それぞれのカメの"ホームポジション"を記憶しさせます（ページに多数のカメがある場合に便利）。

```
カメの中の名前は　”ホーム
全部 「ホームは　位置」
```

現在の位置が、それぞれのカメに、"ホームポジション"として記憶されます。
カメを動かした後ですべてのカメをホーム位置に戻すには、次の命令を使います。

```
全部 「位置は　ホーム」
```

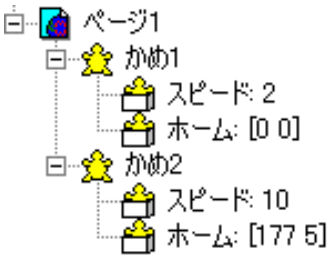
状態変数名を単独で使った場合は、アクティブなカメのその変数値が報告されます。

```
かめ1、 下へ書く　スピード

2
```

カメの中の名前は という変数のカメの値は、語尾に"**の**"を付けた短縮形を使って確認することもできます。

```
下へ書く　かめ2の　”スピード

10
```

作品タブエリアのカメのアイコンをクリックすることによって、特定のカメの変数と値のリストを表示することができます。



無くす　スピード
この命令は、ページ上のすべてのカメから**スピード**というカメの変数を削除します。

b．スライダーとテキストボックスの状態変数

スライダーの状態変数とは、そのスライダーによって報告される値です。
ページにスライダーを作成した状態では、スライダーの初期設定値は 50 になっています。

```
下へ書く　スライダー1

5 0
```

設定するというコマンドとスライダー名を使うことによって、コマンドセンターや手順からスライダーの値を変更することができます。

```
スライダー1は　5
```

テキストボックスの状態変数とは、テキストボックスの内容です。

ページ上にテキストボックスを作成し、テキストを何も入力していない場合、例えば**下へ書く　テキスト1**では、空の単語が報告されます。

設定するとテキストボックス名を使い、コマンドセンターや手順からテキストボックスの内容を編集することができます。
例えば次の命令を入力すると、テキストボックス内に絶えず変化する値が表示されます。

```
くりかえす　20 「テキスト1は　タイマー　待つ　2」
```

c．作品の状態変数

作品には手順と変数が含まれているので、コンピュータ用語の定義では**作品**もオブジェクトです。
作品の状態変数を作ることによって、作品の状態を新たに定義することができます。

ローカル変数は、別の作品に切り替えると失われますが、**作品の状態変数**はその作品と一緒に保存されます。
次回同じ作品を開くと、作品の状態変数には前回保存したときの値が保持されています。

グローバル変数は作品を切り替えた後もその値を保持していますが、マイクロワールドを終了すると、失われます。

作品の状態変数は、**作品内の名前は**を使って作成します。

作品内の名前はで**ポイント**という状態変数を作成してみましょう。

```
作品内の名前は　"ポイント
```

```
ポイントは 10                …ポイントを 10 に設定します。
下へ書く　ポイント           …現在のポイント値を表示します。
```

10

```
ポイントは　ポイント　＋　5　… ポイント値に 5 を加えます。
下へ書く　ポイント                …現在のポイント値を表示します。
```

15

作品内の名前 は、作品に含まれるすべての作品の変数名を報告します。

```
下へ書く　作品内の名前

ポイント                    …現在の作品の作品の変数名
```

3. 作品の保存と共有

(1) 作品保存の注意

音楽、音声、ビデオを使った作品

作品を作る際に音楽、音声、ビデオなどのメディアファイルをインポート（取り込み）した場合、保存の際に注意が必要です。
作った作品を同じコンピュータで見る場合には問題が起きなくても、フロッピーディスクやMO、CD-ROMなどの移動媒体に入れた作品を別のコンピュータで見ようとすると、コマンドセンターから次のようなエラーメッセージが表示されることがあります。単に、その音楽や音声、ビデオの再生が行なえない場合もありますが、プログラムの内容によっては、作品自体を再生できないことがあります。

エラーメッセージの例

startup の中で *"C:Documents and Settings\6-3\"* にあった *"myvoice.wav"* が見つかりません。

(斜体の部分は、作品のプログラムや使っているコンピュータ環境などによって違います。)
これは、インポート（取り込み）したメディアファイルのデータ自体を移動媒体にコピーしなかったために発生します。音楽や音声(録音)、ビデオなどのファイルはインポート（取り込み）すると作品の一部のように見えますが、実際には、ハードディスク上に別々に格納されています。
＊メディアファイルの保存のしくみについては、「メディアオブジェクトの保存」(p.163)を参照してください。

このような問題が起きないように、**作品を保存する際には、フォルダを新しく作り、そこに作品全体を収納すること**を習慣としてください。
そのフォルダの中に、使ったメディアファイルをすべてコピーしておきます。

このようにすれば、必要なときにフォルダごと移動媒体にコピーして使うことができます。

(**別なコンピュータ**とは、「マイクロワールドＥＸがインストールされている別のコンピュータ」を指します。マイクロワールドＥＸがインストールされていないコンピュータで作品を再生する場合は、**(2)インターネットブラウザでの作品利用**　を参照してください。)

(2) インターネットブラウザでの作品利用

マイクロワールドＥＸの作品は、マイクロワールドEXがインストールされていないコンピュータでも、**ウェブプレイヤー（Web Player）**をインストールすることで、作品を見ることができます。ただし、そのためには作品自体もウェブプレイヤー用に「**HTML保存**」を行なって**ウェブ用作品**にしておく必要があります。

ウェブプレイヤー（Web Player）について

・ウェブプレイヤーは、インターネットブラウザの機能を拡張するプラグインです。マイクロワールドＥＸをインストールしたコンピュータには、ウェブプレイヤーも自動的にインストールされます。

・ウェブプレイヤーはマイクロワールドのホームページからダウンロードして、無償で自由にインストールすることができます。また、マイクロワールドＥＸがインストールされていないコンピュータの場合、まずウェブプレイヤーの、インストールを済ませてから、ウェブ用作品を開きます。

マイクロワールドのホームページ　http://www.microworlds.jp/

＊ウェブプレイヤーは『マイクロワールド EX』CD-ROM の中に「PLUGIN」フォルダとして収録されています。

a. ウェブ用作品を作るときの注意

ウェブ用作品を作るときに最も注意すべきことは、音楽や音声などのメディアファイルを含めた作品全体のサイズです。作品全体のサイズを1MB以内にすることをおすすめします。他にも、以下の点に注意してください。

○必ず半角英小文字でファイル名を付けてください。
すでに日本語（カタカナ、ひらがな、漢字）のファイル名を付けている場合は、改めて半角英小文字のファイル名を付けなおしてください。

○ビデオファイルは含めないでください。

○作品の表示サイズはマイクロワールドスモール（530 × 384）以下をおすすめします。作品をこのサイズにするには、新しい作品を作るときに［ファイル］メニューの［作品のサイズ］から［マイクロワールドスモール］を選択します。

［Jr.］ジュニア版では、［ファイル］メニューの［作品の大きさ］から［小さい］を選択します。

○ウェブ用作品を作る際の、もっともおすすめできるサイズは**［Web Player］サイズ**です（400 × 300 ピクセル）。［Jr.］ジュニア版では**［ホームページ］**です。
短時間で確実に表示されることを考えると、ブラウザでの作品表示サイズは小さいほど有利です。

○長時間の音声ファイルは含めないでください。音については、できるだけMIDIファイルを使用するようにしてください。

○ウェブプレイヤーは、ディスクアクセス関係の基本用語と、次の基本用語には対応していません。これらの用語は使わないように注意して下さい。

命令を消す	背景の保存	表を開く
マージ	形を読み込む	表を閉じる
作品のサイズは	形の保存	セルの値は
作品の保存	絵の場所	セルの値
フッターは	字の保存	
下へ書く	字のインポート	
絵を読み込む	字のエクスポート	

ウェブ用作品モードでは、「下へ書く」が自動的に「メッセージを出す」になります。

プログラムの実行や作品のページの切り替えは、ボタンで行えるように設定してください（Web Player ではコマンドセンターがありません）。

インターネットでは、いろいろな構成のシステムが使われています。いろいろな OS やハードウェアプラットフォームに対応できるよう、ウェブ用作品を作るときは次のことにも注意してください。

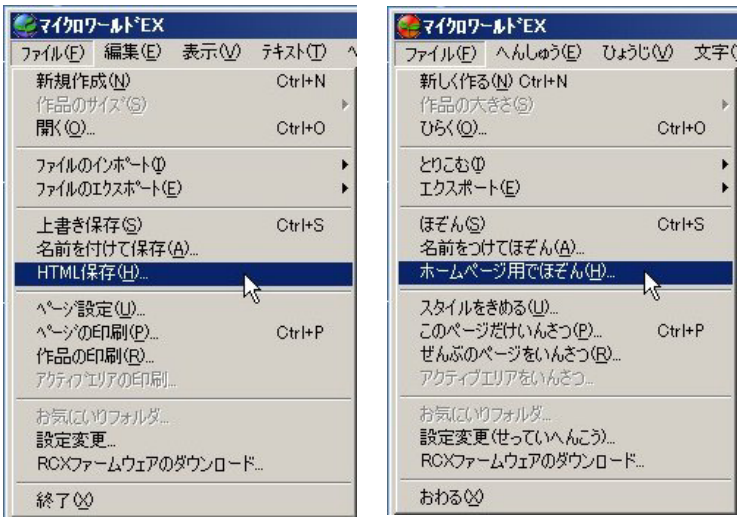
○作品に透明なテキストボックスが含まれている場合は、スタンプを使って、テキストを背景画像にし、テキストボックスは削除してください。このようにすれば、システム環境が変わっても、使われているフォントや色、サイズがそのまま残ります。
○スペイン語やポルトガル語を使って、作品が表示されることもあるかもしれません。このため、数字の区切り記号に小数点やカンマを使わないでください。
代わりに、除算記号（例えば、2.1 ではなく、21 / 10）を使ってください。

b.HTML 形式での保存

・先にマイクロワールドＥＸの作品としてファイル名を付けて保存しないと、この操作は行なえません。
・マイクロワールドＥＸの作品名が、そのままＨＴＭＬ形式のファイル名として使われます（前項目に書いたようにファイル名は半角英小文字で付けてください）。

マイクロワールドEXでは､｢HTML保存｣を行なうと､保存場所として決めた任意のフォルダの中に､作品ファイルと作品に使用したすべてのメディアファイルがコピーされ､そこにHTML形式のウェブ用作品ファイルも格納されます。

(1)［ファイル］メニューから［HTML保存］をクリックします。［Jr.］ジュニア版では［ホームページ用ほぞん］をクリックします。



(2) 保存先のフォルダを選ぶウィンドウが開きます。

例として、マイドキュメントの中の「61yamada」というフォルダに保存する場合を示します。

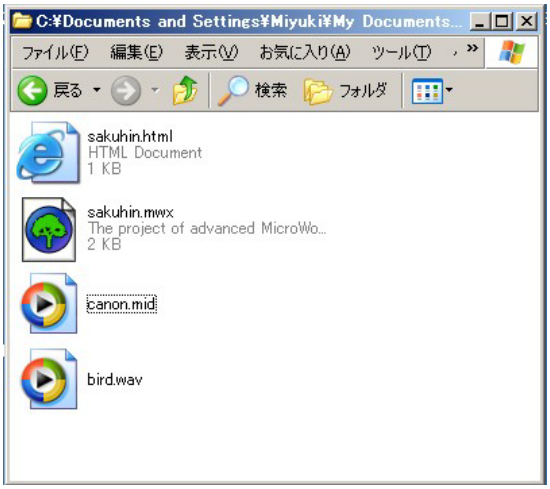
(3) マイドキュメントをクリックします。

(4)マイドキュメントの中にあるフォルダが表示されます。

(5)「61yamada」フォルダをクリックします。

(6)［OK］をクリックします。

これで「61yamada」フォルダに、マイクロワールドEXの作品ファイル（.mwx）と使ったメディアファイルがコピーされ、さらにHTMLファイルが作られました。HTMLファイルをダブルクリックすれば（この例では、sakuhin.htm)、ブラウザで作品を見ることができます。



(3) 作品の結合

共同で作品を作っていて、他の人の作品のページを自分の作品に追加したい場合は、マージというコマンドを使います。

ただし、マージを使うときは、先にいったん作品を保存してください。これは、大量の画像やメディアファイルを含むページを読み込もうとすると、メモリ不足になることがあるためです。

マージでは、別の作品のページや手順を読み込むことができます。マージは2つのインプットを取ります。1つ目のインプットは読み込もうとする作品の名前です。読み込む作品は、現在使っているディレクトリにある必要があります。2つ目のインプットには、読み込むものを指定します。

コマンドセンターから次のように使います。

マージ　"サンプル　"ページ

「サンプル」という作品からすべてのページを読み込みます。読み込み先の作品に同じ名前のページがある場合は、名前を変えて読み込まれます。

マージ　"サンプル　"手順

「サンプル」という作品から手順（手順タブエリアの手順）と作品の変数を読み込みます。読み込んだ手順は、読み込み先の作品に追加されます。

(4)画面サイズの変更

例えば1024x768 のサイズで作られた作品を 800x600 の画面サイズで開いた場合、画面には作品の一部しか表示されません。作品全体を表示するには、マージを使って、大きな作品を変換して、小さくする必要があります。このために、次の手順に従ってください。

・［ファイル］メニューの新規作成で作品を新しく開きます。
・元の作品と新しい作品の間でページ名の重複が起きないよう、ページ1に"削除用"（あとで分かりやすい名前）という名前を付けます。
・コマンドセンターに次の命令を入力します。

マージ　"旧作品名　"ページ	新しい作品にページを読み込みます。
マージ　"旧作品名　"手順	新しい作品に手順と作品の変数を読み込みます。

・必要な部分の読み込みが終わったら、作品タブエリアで"削除用"ページを右クリックして、［削除］を選択します。

マージについては［ヘルプ］メニューの［基本用語の検索］も参照してください。