●はじめてのプログラム●●●

プログラム実行の手順の再確認と簡単なプログラムの説明をします。何事もはじめが肝心ですが、気負わず、コーヒー・紅茶を片手にリラックスしてはじめましょう。また画面に文章を表示するいろいろな方法も覚えましょう。

とにかくプログラムを書いて実行してみたい！

プログラミングというのは、『習うより慣れろ』の部分が多いので、なかなか、本を読んだだけでは使いこなせるようになりません。そこで、多少手間はかかりますが、自分で入力してプログラムの動作を確かめながら進んでください。

もうなでしこはインストールしてありますね？デスクトップやスタートメニューにある「なでしこエディタ」をダブルクリックしてエディタを起動してみましょう。

エディタが起動したら、図1のようにエディタ部分へ以下のプログラムを入力して、画面上部にある実行ボタンを押してみてください。

はじめて.nako

```
01:「こんにちは」と表示。
```

図1　エディタへプログラムを入力

このプログラムを実行すると図2のように「こんにちは」と画面に表示されます。

図2　実行してみたところ

うまく動かなかった場合●●●

プログラムを実行しても画面に何も表示されなかったという人はいませんか？それは『「こんにちは」と表示』のカギカッコ「…」を書き忘れた場合です。なでしこでは、カギカッコや丸カッコ、タブや改行がプログラムとして意味を持ちます。これを書き忘れてないか確認してみてください。

プログラムの構造 ●●●

このプログラムは図3のような構造になっています。『「***」と表示』と書くと、***に指定した文字を表示します。

図3　プログラムの構造

『表示』というのが命令で『「***」と』の部分が命令の説明や動作を表す引数部分です。ここで命令語句と説明語句を区切るのは、「と」や「を」などの助詞です。なでしこでは、助詞により語句の意味が区切られるということを覚えておいてください。

他の文章を画面に表示する方法

ダイアログに表示 ●●●

次に画面ではなく、メッセージ表示用のダイアログを表示して挨拶をさせてみます。前回と同じようにプログラムを入力し、実行させてみてください。図4のように画面にメッセージが表示されれば成功です。

はじめて2.nako

```
01:「こんにちは」と言う。
```

図4『言う』命令でダイアログへ文字を表示したところ

クジラが話す？ ●●●

次にクジラを使って挨拶をさせてみます。上のプログラムと同じようにプログラムを入力して実行してみましょう。これを実行すると図5のようにクジラが表示されて「こんにちは」と挨拶をします。

はじめて3.nako

```
01:クジラが「こんにちは」と言う。
```

図5　クジラがメッセージを話します

適当に書いても動きません。

このように「なでしこ」では、あたかも日本語で誰かに命令するようにプログラムを書いて動かすことができます。しかし、適当に書けば動くというわけではありません。例えば、さっき、クジラと書いていたところを、適当に『エリマキトカゲ』にしてどうなるのか実験してみましょう。

間違い.nako

```
01: エリマキトカゲが「こんにちは」と言う。
```

このプログラムを実行すると、図6のようなダイアログが出て『記述ミスがあります。プログラムを見直してください。』と怒られてしまいます。

なでしこは、プログラミング言語です。日本語をただ適当に書いただけでは動きません。プログラミング言語では、ある一定の規則に従ってプログラムを記述する必要があるのです。それでこのように、知らない単語が出てくると間違いがあることを知らせるメッセージがでるようになっています。

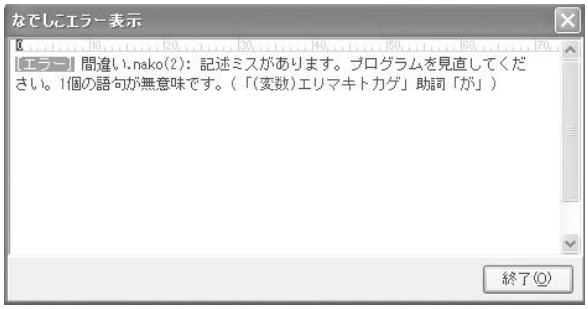


図6　でたらめな語句を入れるとエラーがでます。

好きな言葉を表示させてみよう！

画面に文字を表示するいろいろな方法を見てきました。そこで今度は自分で好きな言葉を画面に表示させるプログラムを作ってみてください。『「★★★」を表示』の★★★のところを書き換えるだけで好きな言葉を表示できます。例えば、以下のプログラムは画面に「明日は明日の風が吹く」という文章を表示します。

好きな言葉.nako

```
01: 「明日は明日の風が吹く」と表示。
```

エディタ部分にプログラムを書きます。

実行すると「明日は明日の風が吹く」という文章を表示します。

➔好きな言葉に書き換えて実行してみよう。

プログラムを保存しよう ●●●

プログラムを保存しておけば、次回使いたい時に、読み込んで再度実行することができます。
プログラムを保存するには、メインメニューから[ファイル - 名前を付けて保存]をクリックします。
するとプログラムの一行目にある行が名前の候補として表示されます。
保存したいフォルダへ移動し、もし適切な名前でなければ修正して[保存]ボタンをクリックします。

ファイルメニューから名前を付けて保存を選択します。

用語の確認

ここで今までに出てきた用語や一般的に使われるプログラミング用語をまとめてみます。

プログラム（英語:program）●●●

一般的に『プログラム』と言えば、運動会や演劇の進行手順を順を追って書いたものですが、コンピューターの世界でも、コンピューターに命令するための進行手順を書いたものがプログラムです。

映画やアニメのSFに出てくる『プログラム』は、でたらめな意味の無い記号の羅列であることが多いので、得体のしれないものという印象がありますが、実際に使われている『プログラム』はできるだけ分かりやすく書くものです。

プログラミング（英語:programming）●●●

プログラムを作成することを『プログラミング』と呼びます。また、プログラムを作成する人を『プログラマー』と呼びます。なでしこ開発時のキャッチフレーズは『なでしこで誰でも簡単プログラマー』です。なでしこで簡単にプログラマーになりましょう！

文法（英語:syntax）●●●

プログラムはコンピュータに動作手順を記したものなので、コンピュータが理解できるように、ある一定の規則で記述しなければなりません。この規則のことを『文法』と呼び、各プログラミング言語によって多少の違いがあります。

エラー（英語:error）●●●

プログラムの間違いをエラーと言います。そしてプログラムにエラーがあることを、バグ(英語:bug)があるとも言います。バグとは虫とかばい菌という意味です。エラーもバグも、どこかで聞いたことのある単語ではないでしょうか。

ソース（英語:source）●●●

日本語で『ソース』と言えば焼きそばなどにかける調味料のことですが、プログラムのことを『ソースコード』や『ソース』と呼ぶこともあります。英語の"source"とは根源とか源の意味です。コンピューターの動く源ということでしょうか。また、プログラムの書かれているファイルのことを『ソースファイル』と呼びます。

●なでしこを電卓として使いたい●●●

コンピューターのことを電子計算機とも言います。パソコンは計算が得意なのです。そこで、なでしこを電卓代わりに使う方法を紹介します。しかし、ただの電卓の代わりでは面白くないので、時間計算機や関数電卓として使う方法も紹介します。

足し算と引き算

まずは、足し算からやってみます。以下のプログラムを入力して実行してみましょう。

足し算.nako

```
01: 3＋4を表示。
```

➔足し算の結果「7」が表示されれば足し算ができていることを確認できます

次に引き算をやってみましょう。引き算も同じように書く事ができます。今度はちょっと大きな桁で試してみます。

引き算.nako

```
01: 3000－1500－1000を表示。
```

➔引き算の結果「500」が表示されれば引き算ができていることを確認できます

この例のように『A-B-Cを表示』や『A+B-C+D+Eを表示』など、計算式はいくつでも並べて書く事ができます。

注意！

このSTEPの画像は、見やすくするために文字を大きく表示しています。文字を大きく表示する方法は、110頁参照してください。

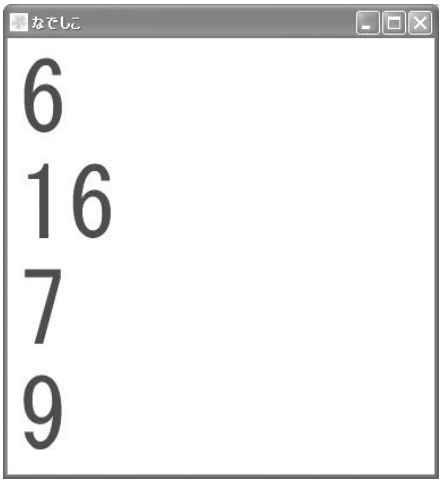
四則演算

かけ算や割り算も、上の足し算と同じように書くことができます。以下のプログラムを実行して結果を確認してみましょう。

四則演算.nako

```
01: 3×2を表示。
02: 4×8÷2を表示。
03: 1＋2×3を表示。
04: (1＋2)×3を表示。
```

計算結果が順に表示されています



実行結果は予想通りでしたか？かけ算や割り算は問題ないですね。ちょっと気になる点だけ補足します。

プログラムの3行目の計算結果を見てください。1＋2×3の答えが7になってます。足し算とかけ算があった場合、足し算よりもかけ算が先に計算されているのを確認してください。

4行目のように、丸カッコ（...）をつけるとカッコの中を優先して計算するようになります。普通の電卓では、電卓で計算出来るようにあらかじめ頭の中で組み立てないといけませんが、なでしこでは、一般的な計算式の順番にそって計算されるのです。

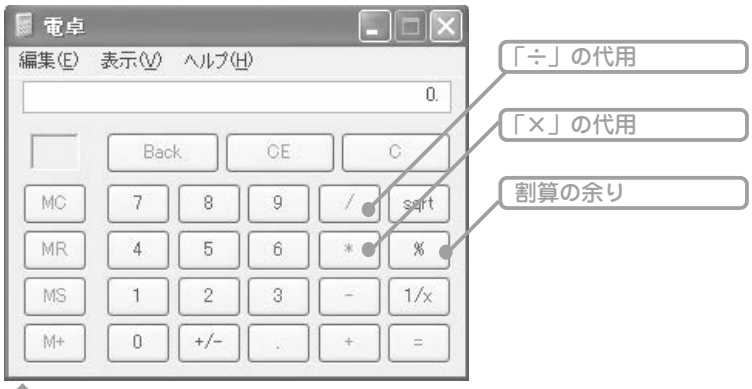
省略記号 ●●●

ところで、「×」という文字を入力するには、漢字変換で「かける」と入力して変換すると出てきます。また「÷」を入力するには「わる」と入力して変換します（Windows標準の漢字変換ソフトIME2003を使った場合）。

これでは、変換がちょっと面倒なので、もっと簡単にできる方法を紹介します。

計算式の中で「×」を「＊」(アスタリスク)で、「÷」は「/」(スラッシュ)で代用できるようになっています。これは、他のプログラミング言語やExcelなどの表計算ソフトでも同じです。

ここで、Windowsに標準でついてくる電卓を起動してみてください（スタートメニューのアクセサリにあります。または、なでしこで『「calc」を起動。』と書いて実行するとすぐ起動します。）。



Windows標準の電卓です

これを見ると分かりますが、Windows標準の「電卓」では、はじめから、「×」を「＊」、「÷」を「/」と表しているのです。私は仕事柄パソコン初心者の方から質問を受けることがあるのですけど「Windowsの電卓って掛け算と割り算できないんですか？」と質問を受けることもしばしばです。

「＊」と「/」の記号でかけ算、割り算ができることを覚えておきましょう。

割り算の余り ●●●

それから、電卓をよく見てみると、右の方に「%」という記号があります。これは、割り算の余りを求めるときに使います。例えば「３％２」と書けば、３を２で割った余りを求めるという意味になります。ではプログラムを実行して確認してみましょう。

割り算の余り.nako

```
01: 3％2を表示。
02: 19％7を表示。
```

これを実行すると１と５が表示されます。

３を２で割ると１あまり１。19を７で割ると２あまり５なので、このように表示されるのです。

累乗（べき乗） ●●●

他によく使う計算として累乗（べき乗）があります。これは「＾」という記号で計算できます。累乗とは、ある数に同じ数を次々と何回か掛けることです。

べき乗.nako

```
01: 2＾3を表示。
02: 8＾5を表示。
```

これを実行すると８と32768が表示されます。

『２＾３』とは『２×２×２』と同じ意味で答えは８です。同じく『８＾５』は『８×８×８×８×８』の意味で答えは32768です。

計算に使える演算子の一覧 ●●●

いろいろ計算記号が出てきましたので、使える記号（演算子）をまとめてみました。

記号	意味	使用例	結果
＋	足す	３＋２を表示	5
－	引く	５－３を表示	2
×または＊	掛ける	３×２を表示	6
÷または／	割る	４÷２を表示	2
％	割り算の余り	４％２を表示	0
＾	累乗	２＾３を表示	8

計算の練習 ●●●

これで一通り四則演算の計算ができるようになったので練習をしてみましょう。なでしこを使って以下の問題を解くプログラムを作ってください。

(1) 1111111 × 1111111 の答えは？
(2) (11 + 22) × 33 ÷ 44 の答えは？
(3) 100円の消しゴムを5つと、180円のサインペンを3つ買ったらいくら？

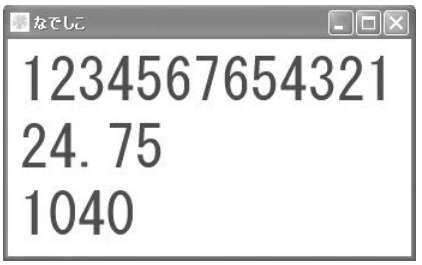
答えを見なくてもプログラムを作ることができたでしょうか？

計算練習.nako

```
#1（１）の答え
１１１１１１１１×１１１１１１１１を表示。

#（２）の答え
（１１＋２２）×３３÷４４を表示。

#（３）の答え
１００×５＋１８０×３を表示。
```



計算練習の実行結果です

コメントについて

ちなみに、上のプログラムで「#」(シャープ)から始まる行がありますが、これはコメントと言って、プログラムとして何も意味も持たない文です。注釈文とも言います。ちょっとしたメモ代わりとして使えます。以下のように文の途中にコメントを記述することもできます。

コメント.nako

```
# コメントのテスト
「ここはコメントではない」と表示。# ここはコメント
```

『表示』命令についての補足

計算をするのに『表示』は必要？ ●●●

ここで『表示』を書かずに、3＋4とだけ書いて実行したらどうなるでしょうか？やってみると分かりますが、何も表示されません。これはどういうことでしょうか？

表示なし.nako

```
3＋4
```

3＋4とだけ書いて実行してみたが何も表示されなかった

『3＋4』と書けば、3＋4の計算はされるのです。しかし答えは画面に表示されません。なぜなら、計算

結果を必ず画面に表示してしまうと、たくさんの計算を行った場合によっては画面が数字だらけになってしまうことがあります。

計算結果を画面ではなくファイルに保存したいこともありますし、ネットワークでつながったコンピューターに転送したいこともあるのです。

それで、計算結果を画面に表示したいときは『ｘｘｘを表示』と書く必要があるのです。

カギカッコは不要なの？ ●●●

『表示』を覚えたとき、文章を表示するときは『「***」と表示』のように、表示する内容をカギカッコで囲う必要がありました。しかし、今回のように計算を行うときには、カギカッコで囲う必要はありません。

確認のために、カギカッコを書いたときと、書かなかったときと、どのようにプログラムの動作が変わるのか比較してみましょう。

カギカッコ比較.nako

```
01:「３＋２」を表示。
02:３＋２を表示。
```

➔一行目がカギカッコをつけた場合、
二行目がカッコを書かなかった場合

実行してみて結果に納得ができたでしょうか。カギカッコを書くと、カギカッコの中に書いた内容がそのまま表示されます。そして、カギカッコを書かない場合で、計算式があれば、それを計算した結果が表示されます。

ついでに、ちょっと気の利いた表示にしてみましょう。

気の利いた表示.nako

```
01:「答えは」を表示。
02:３＋２を表示。
03:「です。」を表示。
```

➔説明メッセージを計算結
果に加えて表示したものです

時間計算機・日付計算機として使う

さて、ここまでは「なでしこ」を普通の電卓の代わりとして使った訳ですが、普通の電卓では付いてない「時間計算機」や「日付計算機」として使う方法を紹介します。

時間の計算 ●●●

なでしこを使えば普通の電卓ではできないような計算も、らくらく計算することができるのです。以下は、8時10分00秒から8時20分30秒の秒差を計算するためのプログラムです。例えば、勉強の開始時間と終了時間

を入れて勉強時間を調べたり、会社の出社時間と退社時間を入れて勤務時間を調べるのに使えますね。

時間を計算する.nako

```
01:「08:10:00」と「08:20:30」の秒差を表示。
```

➔『秒差』命令で、8:10:00と08:20:30の秒差を計算して表示してみたところです。

プログラムを実行すると、図のように、計算結果の630が表示されます。時間や日付を指定するときは「08:10:00」のように時間や日付をカギカッコで囲う必要があります。もしうまく実行できなかった場合は、カッコや語句が合っているかを確認してみてください。

今のは時間の引き算でしたが、次に時間を足してみます。9時32分10秒に1時間を足すには以下のように書きます。

時間を加算する.nako

```
01:「09:32:10」に「1:00:00」を時間加算して表示。
```

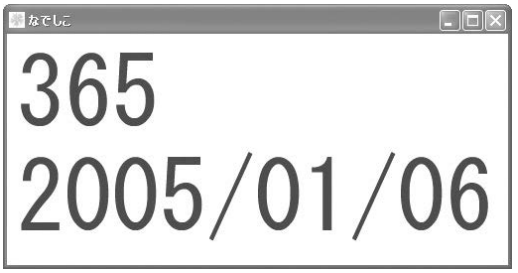
➔『時間加算』命令で09:32:10に1時間を足して表示してみたところです。

日付の計算 ●●●

次は日付についてやってみます。日付も時間と同じように計算ができます。以下のプログラムは、2005年1月1日から2006年1月1日までの日数差を表示します。そして、次いで、2005年1月1日に5日分加算した日付を表示します。

日付計算.nako

```
01:「2005/01/01」と「2006/01/01」の日数差を表示。
02:「2005/01/01」に「0/0/05」を日付加算して表示。
```



➔1行目が『日数差』命令を使って2005年元旦から2006年元旦までの日数差、2行目が『日付加算』命令を使って2005年元旦に5日足した日付です。

日数差は冒頭の紹介で恋人の誕生日まで何日か調べるのにも利用しました。私は、以前、資格試験を受験するときに、テストまであと何日かを調べ、勉強の進み具合を調べるのにこの命令を使いました。時間や日付の計算は日常生活でも便利なものですよね。

関数電卓として使う

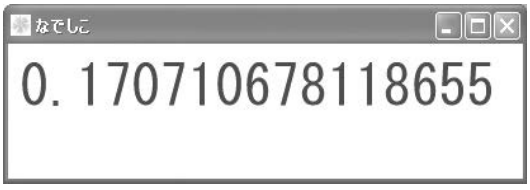
今度はなでしこを関数電卓として使う方法を見ていきます。関数電卓というのは、三角関数や自然対数、絶対値などの計算を行うことができる高度な電卓です。日常生活ではあまり使うことがありませんが、少し高度な計算をしたいときは便利です。

関数電卓によくある関数だけを抜き出してみました。以下が利用できる関数の一部です。

関数名	意味
SQRT(x)	xの平方根（ルート）
SIN(x)	x（ラジアン単位）の角の正弦（サイン）
COS(x)	xの角の余弦（コサイン）
ARCTAN(x)	xの角の逆正接（アークタンジェント）
DEG2RAD(x)	ラジアンxを角度に変換
RAD2DEG(x)	角度xをラジアンに変換
EXP(x)	e（自然対数の底)のx乗
LN(x)	xの自然対数（LN(x) = 1）
LOG2(x)	xの対数（基数2）
LOG10(x)	xの対数（基数10）
ABS(x)	xの絶対値
INT(x)	xの整数部分を返す
FRAC(x)	xの小数部分を返す

関数電卓.nako

```
01: (SIN(PI/4)^3+COS(PI/3))/5を表示。
```



三角関数を使った計算もこの通り

計算機の代わりに「なでしこ」を使うメリット

Windowsの電卓も使いやすいのですが、電卓では数値を入力して「＋」など演算子を入力すると前の数値が消えてしまいます。ちょっと複雑な計算をしようと思った時、どこまで計算したかを忘れてしまって、はじめから計算し直した経験がある人いませんか？

なでしこなら計算式がそのまま残るので、どこまで計算したか一目瞭然です。計算式を入力して間違いがないか確認してから計算結果を得ることができます。

STEP-5

●変数を使いたい●●●

変数とはプログラム中で使う数値や文字などのデータをしまっておくことができる箱のようなものです。変数を使うと数値だらけだった計算式を意味のある単語に置き換えることができるので、ぐっとプログラムが見やすくなります。

計算式を変数で分かりやすく

変数を使うことで、計算式を分かりやすくプログラムにすることができます。例えば、以下のような計算を、なでしこを使ってプログラムにしてみましょう。

気のあった友人が３人集まって食事会へ行きました。そこで、500円のビールを2杯と350円のウーロン茶を1杯、それから、1000円のパスタを3皿、900円のピザを2枚頼みました。お会計は仲良くワリカンに（３等分）することにします。一人分の支払いはいくらになるでしょう？

ビールを飲まなかった人がちょっと損をしますが、気の合う仲間同士なので気にしなかったことにします。単純にプログラムにすると以下のようになります。

計算式.nako

```
01: (500×2＋350×1＋1000×3＋900×2)÷3を表示。
```

➔食事会の計算結果

次に、この計算を変数を使ってプログラムにしてみます。

計算式2.nako

```
01: ビールは、500円
02: ウーロン茶は、350円
03: パスタは、1000円
04: ピザは、900円
05:
06: 人数は、３人
07:
08: 合計金額は、ビール×2＋ウーロン茶×1＋パスタ×3＋ピザ×2
09: 合計金額÷人数を表示。
```

実行結果は同じ2050ですが、プログラムはずいぶん違うものになりました。変数を使っていない数字の羅列と、変数を使って数値を置き換えたものと、どちらが分かりやすいでしょうか。

簡潔に書いてあるのは前者ですが、後者の変数を使った方が、計算式の意味が分かりやすく、間違いがあったときにも、修正が簡単にできそうです。

例えば、もし、以前この店に来たときは、パスタもピザも値段が同じ1000円だったとします。それで、ピザもパスタも1000円で計算したのですが、実際に会計を済ませてみると、パスタの値段が1300円に値上がりしていました。この場合、前者の変数を使わない方法だと、どの1000が何を表しているのか分からないので式を組み立てなおす必要がありますが、後者の変数を使った方法では、パスタの値段を書き換えるだけで良いのです！！

このように、変数を使うと複雑な計算に意味を持たせ、プログラムを分かりやすくすることができるのです。

算数の公式に当てはめてみる

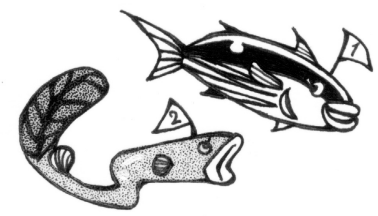
次に、簡単な公式を変数を使ってプログラムにしてみようと思います。以下は速さに関する公式です。この公式はこれだけで既になでしこのプログラムとして動かすことができます。これを使って、小学校の５年生の問題を解いてみましょう。

●速度＝距離÷時間
●時間＝距離÷速度
●距離＝速度×時間

魚のかけっこ問題 ●●●

海の中を泳ぐ魚は、いったいどのぐらいのスピードで泳いでいるのでしょうか。魚の中でいちばん速いのはメカジキです。なんと時速100kmで泳ぎます。つぎはマグロで80kmです。そしてカツオは60kmです。ウナギなどはゆっくりなので時速4kmです。

そこで問題です。カツオとウナギがスタート地点からゴールまで同じ距離を同時に走りました。その時、ウナギはゴールするのに30時間もかかりました。それでは、カツオは何時間かかったのでしょうか？



魚のかけっこ.nako

```
01: #(1) ゴールまでの距離を求める
02: 速度は 4 #ウナギの速さ
03: 時間は30 #ゴールまでかかった時間
04: 距離＝速度×時間
05:
06: #(2) カツオがゴールまでかかる時間を求める
07: 速度は60 #カツオの速さ
08: 時間＝距離÷速度
09:
10: 時間を表示。
```

小学生の問題と言ってもなかなか高度です。まずは、ウナギの泳ぐ速さとゴールまでにかかった時間を使って、ゴールまでの距離を求めます。速さと時間が分かっているので『距離＝速度×時間』の公式で求められま

す。次に、ゴールまでの距離が出れば、『時間＝距離÷速度』でカツオがゴールすまでにかかった時間が求められます。

これを実行すると2が表示され、カツオは2時間でゴールできるということが分かります。

地球から月までの距離 ●●●

次の問題です。地球から月までの距離を光が往復するのに、2.56秒かかるそうです。光の速さは秒速30万kmと言われています。それでは地球から月までの距離は何kmでしょうか？

地球から月までの距離.nako

```
速度は300000 # 秒速30万km
時間は2.56  # 地球から月までかかった時間

距離＝速度×時間
距離を表示。
```

さっきの問題が解ければ、今度のは簡単ですね。ちょっと単位が大きいので入力間違いに注意すれば問題ないでしょう。『距離＝速度×時間』の公式に当てはめるだけです。プログラムの実行結果は、768000と表示されます。

変数の代入方法

既にここまで見てきた中で、変数へいろいろな数値を代入してきました。なでしこでは、右の3つの方法で代入を行うことができます。

	形式	例
(1)	(変数)は(値)	値段は300
(2)	(変数)＝(値)	値段＝300
(3)	(変数)に(値)を代入	値段に300を代入

この3つの方法はどれを使っても同じ意味になります。一般的なプログラム言語では、(2)の方法で代入を書きますが、なでしこでは、好きな方法を選べます。

文字列の代入と変数の埋め込み

文字列の代入 ●●●

また、変数には文字を代入することもできます。プログラムの世界では、語句や文章などのデータのことを、『文字列』と呼びます。文字が列として連なったデータなので文字列です。

プログラムで数値を指定する時は、そのまま、30とか77とか書きますが、文字や文章などの文字列を指定するときには、カギカッコで囲くという規則があります。例えば、「こんにちは」とか「有言実行」のように書きます。いままでも、文章を表示するときは、カギカッコで囲っていましたね。

文字列の代入.nako

```
01: 名前は「クジラ」
02: 趣味は「ギター」
03:
04: 名前を表示。
05: 趣味を表示。
```

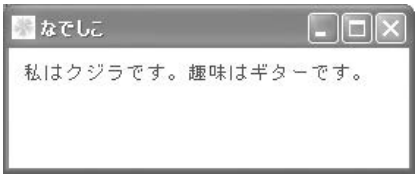
➔文字列は必ずカギカッコで囲います

変数を文字列に埋め込む方法 ●●●

文字列は変数に埋め込むことができます。例えば、上の『文字列の代入』のプログラムにある自己紹介を自然な文章で表示してみます。

変数の埋め込み.nako

```
01: 名前は「クジラ」
02: 趣味は「ギター」
03:
04:「私は{名前}です。趣味は{趣味}です。」と表示。
```

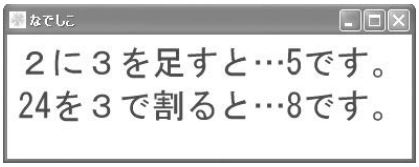


➔文字列に変数を埋め込んで表示したところ

このように、文字列の中で「...{変数}...」と書くことにより、変数を文字列の中に埋め込むことができます。また、変数だけでなく、以下のように計算式を埋め込むことも可能です。

式の埋め込み.nako

```
01:「2に3を足すと…{2+3}です。」を表示。
02:「24を3で割ると…{24/3}です。」を表示。
```



➔文字列に式を埋め込んで表示したところ

STEP-6

●ユーザーからの入力を得たい●●●

さて今度はプログラムの途中でユーザーから何かデータを入力してもらうプログラムを作ってみます。これによりプログラムを書き換えなくても変数の内容を変えて表示結果を変えることができます。

『尋ねる』命令を使う

ユーザーから何か数値や文字列を入力してもらうには『尋ねる』命令を使うのが一番簡単です。尋ねる命令は以下のように使います。

尋ねる.nako

```
01:「年齢は？」と尋ねる。
02:それを表示
```

一行目、『「年齢は？」と尋ねる』と書くと、図1のような入力ダイアログが出ます。ここには、数値や文字列などを入力できます。そして『尋ねる』命令で入力された文字列は、『それ』という変数に自動的に代入されます。そこで、『それを表示』と書くと、ユーザーの入力した値がそのまま画面に表示されます。

図1　『尋ねる』命令を使うと入力ダイアログが出て値を入力できます

18を入力して[決定]ボタンを押すと入力したものが表示されます

インチ→センチ変換プログラム

『尋ねる』の練習として、インチ(Inch)→センチ(cm)の変換プログラムを作ってみようと思います。1インチは2.54cmです。インチと言えば、電気屋さんで、ＴＶの大きさや、パソコンのモニターの大きさを表すのに使われていますよね。以下が変換プログラムです。

インチ→センチ.nako

```
01:「何インチですか？」と尋ねる。          # 1
02:インチはそれ。                          # 2
03:センチはインチ×2.54                     # 3
04:「{インチ}インチは{センチ}cmです。」を表示。  # 4
```

プログラムを解説します。1行目の『尋ねる』文では、ユーザーへ「何インチですか？」と質問します。2行目ではユーザーが入力した値を変数『インチ』へ代入しています。3行目では2.54をかけてインチをセンチへと変換しています。

そして最後の4行目では、変換結果を表示しています。ここでは、前項でやった文字列へ変数を埋め込む技を使って分かりやすくプログラムの結果を表示しています。

インチ→センチの変換プログラムで16を入力してみたところ

健康チェック～BMIで肥満度を確認

さて、いくつか『尋ねる』命令を使ってプログラム作成の練習をしてみようと思います。手始めに、なでしこで、肥満度チェックをやってみます。簡単な健康チェックの方法にBMIというのがあります。BMIを使うと理想体重を調べることができます。そこでこれを使って肥満度を算出してみます。

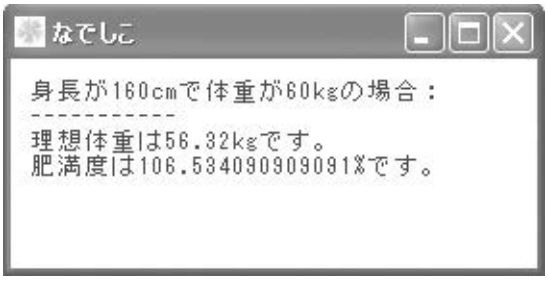
もともとBMIとはこの体重なら病気になる確率がもっとも低いという疫学調査から導き出されたものです。理想体重の求め方は以下のようにします。

```
理想体重＝((身長÷100)^2)×22
```

この理想体重の算出方法を元にプログラムを作ってみます。ユーザーから身長と体重を尋ね、その入力を元に理想体重と肥満度を算出します。

肥満度チェック.nako

```
01:「身長(cm)は？」と尋ねる。
02:身長はそれ
03:「体重(kg)は？」と尋ねる。
04:体重はそれ
05:
06:理想体重＝((身長÷100)^2)×22
07:肥満度は体重÷理想体重×100
08:
09:「身長が{身長}cmで体重が{体重}kgの場合：
10:----------
11:理想体重は{理想体重}kgです。
12:肥満度は{肥満度}%です。」と表示。
```



身長160cm、体重60kgを入力した場合の結果

プログラムの解説をします。これは前の『インチ→センチ変換』プログラムを応用したものです。『尋ねる』命令を使って身長や体重をユーザーから入力してもらい、それを変数『身長』『体重』に代入します。この変数を使って理想体重、肥満度を計算して、最後に計算結果を表示するというものになってます。

最後の『表示』命令ですがカギカッコの合間に改行を挟んでいます。なでしこでは、一度に複数の行を表示したい場合に、このようにカギカッコの中に改行を挟んでも良いことになっています。

ちょっと横道

ちなみに、本来なら男女差や職業による運動量の差も考慮しないといけないので、プログラムの実行結果については参考程度に考えてくださいね。

歩いた歩数と消費カロリーをチェック

肥満度をチェックしたところで、次に歩いた歩数と消費カロリーのチェックをしてみようと思います。

昔から、『一日一万歩歩くと健康になる』と言われていますが、万歩計でも買わないと歩いている歩数なんか分かりません。そこで、体重60kgの人の場合、1分間100歩のペースで歩くと約3.3kcalを消費するというデータから、だいたいの歩いた歩数と消費カロリーを算出してみましょう。

歩数と消費カロリー.nako

```
01:「今日は何分間くらい歩いたの？」と尋ねる。
02:歩いた分数はそれ。
03:歩数は１００×歩いた分数。
04:カロリーは、3.3×歩いた分数
05:「{歩数}歩、歩きました。
06:消費カロリーは{カロリー}kcalです」と言う。
```

40を入力したところ。40分歩いただけでは1万歩には届かない

プログラム的には「肥満度チェック」を応用したものになっています。そこで、ここでプログラムの基本的な流れを確認してみます。どんなプログラムでも基本的にはこの流れに沿ったものになります。

（1）データの入力
（2）入力されたデータを元に計算など処理
（3）プログラムの結果を出力

まず（1）のデータの入力というのは、今回で言えば、何分歩いたのかユーザーからの入力を得る部分です。そして、（2）が歩数やカロリーなどの計算部分、最後の（3）で、画面に計算結果を表示するという部分に当たります。

自分でプログラム作ろうと思ったときは、この流れを意識して作ると考えやすいと思います。

卒業年度を調べるプログラム

今度は卒業年度をチェックするプログラムを作ってみます。就職やアルバイトの履歴書を書くとき、いつも卒業年を調べるのは大変なことです。そこで、プログラムを使って手っ取り早く計算してしまおうというわけです。

プログラム自体はただの足し算ですが、ここでは、早生まれの判定はやってませんので、1～3月生まれの人は生まれた年を1つ減らして入力してみてください。この問題については、『もし～ならば～』構文を使うことで改良できますので、また後で改良することにします。

卒業年度チェック.nako

```
01:「生まれた年度（西暦）は？」と尋ねる。    # (1)
02:生年はそれ。
03:
04:小学校は(生年＋13)を和暦変換              # (2)
05:中学校は(生年＋16)を和暦変換
06:高校は(生年＋19)を和暦変換
07:短大は(生年＋21)を和暦変換
08:大学は(生年＋23)を和暦変換
09:
10:# 結果の表示                              # (3)
11:「小学校卒業：{小学校}
12:中学校卒業：{中学校}
13:高校卒業：{高校}
14:---
15:短大卒業：{短大}
16:大学卒業：{大学}」と言う
```

1980を入力してみたところ。

プログラムの(1)の部分は生まれた年を西暦で質問しています。ユーザーから入力された値を変数『生年』に代入します。(2)では、卒業年度を計算し、それを『和暦変換』命令で和暦表記に変換しています。そして(3)では結果を表示します。

プログラムは保存して使おう！

関連付けで実行 ●●●

なでしこプログラムでファイルの拡張子は「.nako」です。インストールすると、拡張子「.nako」となでしこの実行ファイルが関連付けされます。関連付けされると、エクスプローラー上で「xxx.nako」というファイルをダブルクリックした時に、なでしこのプログラムが実行されるようになります。

例えば、以下のプログラムを入力したら、『文字数チェック.nako』というファイル名で保存してください。（なでしこエディタの[ファイル]メニューから[名前をつけて保存]をクリックすると保存できます。）

文字数チェック.nako

```
01: ファイル選択
02: それを開いて文字数を表示。
```

エクスプローラー上でこの『文字数チェック.nako』をダブルクリックすると、直接プログラムが実行されます。このプログラムは選択したテキストファイルの文字数を数えて表示するという単純なものです。

いままでは、なでしこエディタを起動した後、プログラムを入力して実行という手順を踏んでいましたが、関連付けされていれば、ダブルクリックしただけでプログラムを実行することができるのです。

たった2行のプログラムでも、いちいち入力しなおしたり、エディタから開いて実行するのは面倒です。そこで、これをデスクトップなど実行しやすい場所にコピーしておけば、いつでもダブルクリックでプログラムを実行できるという訳です。

コードを保存

文字数を表示

注意！

実行ファイルの作り方ついては、424ページ参照。