HSP 講座　1

０、はじめに

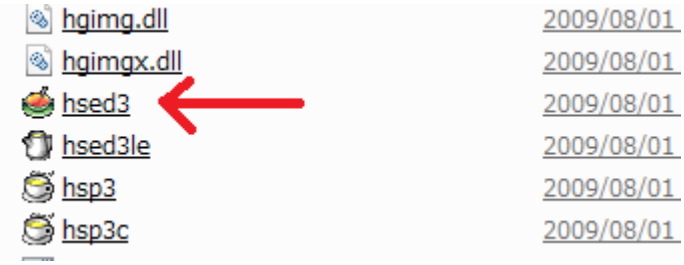
この講座では、HSP を用いた基本的なプログラミングの入門と、ゲーム作りに関する基本的、または初歩的なことについて説明します。

HSP（Hot Soup Processer）とは、プログラミングのうちスクリプト言語と呼ばれるもので、「簡易プログラミング言語」とも呼ばれ、その名の通り他のプログラムに比べて簡単に習得でき、小規模なプログラムをすばやく作成できます。C 言語や C++の様に大規模なプログラムを作ったり、汎用性があったりとはいかないものの、簡単なゲームを作成するにはなかなか有用でないかと思われます。

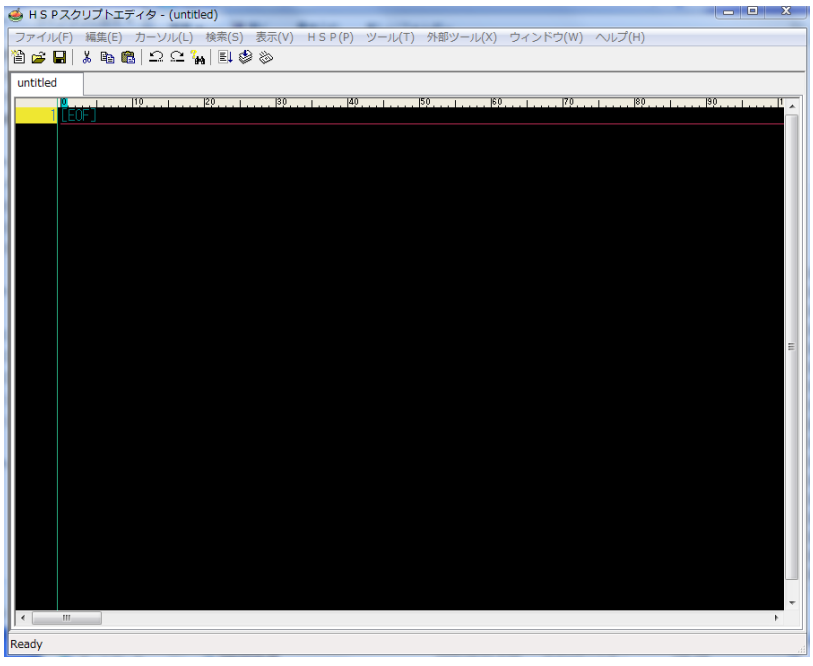
HSP は画像や音声の処理が非常に楽にでき、かつ比較的習得しやすい（と思う）ので、これを使ってプログラミングについて基本的な部分を説明していきます。

１、プログラミングの第一歩

さっそく、HSP のプログラミングをやっていきます。まず、渡した GCC フォルダ内の HSP というフォルダを開いてください。その中には、hsp32 というフォルダ、およびアプリケーションファイル（インストーラ）が入っていると思います。家のパソコンで使いたい場合、このアプリケーションファイルを実行してパソコンにインストールした方が動作的にはいいと思います。今回は、学校など HSP をインストールできない場所での実効の仕方を説明します。Hsp32 というフォルダの方を開いてください。まず、実際に HSP のプログラムファイルを作ってみましょう。



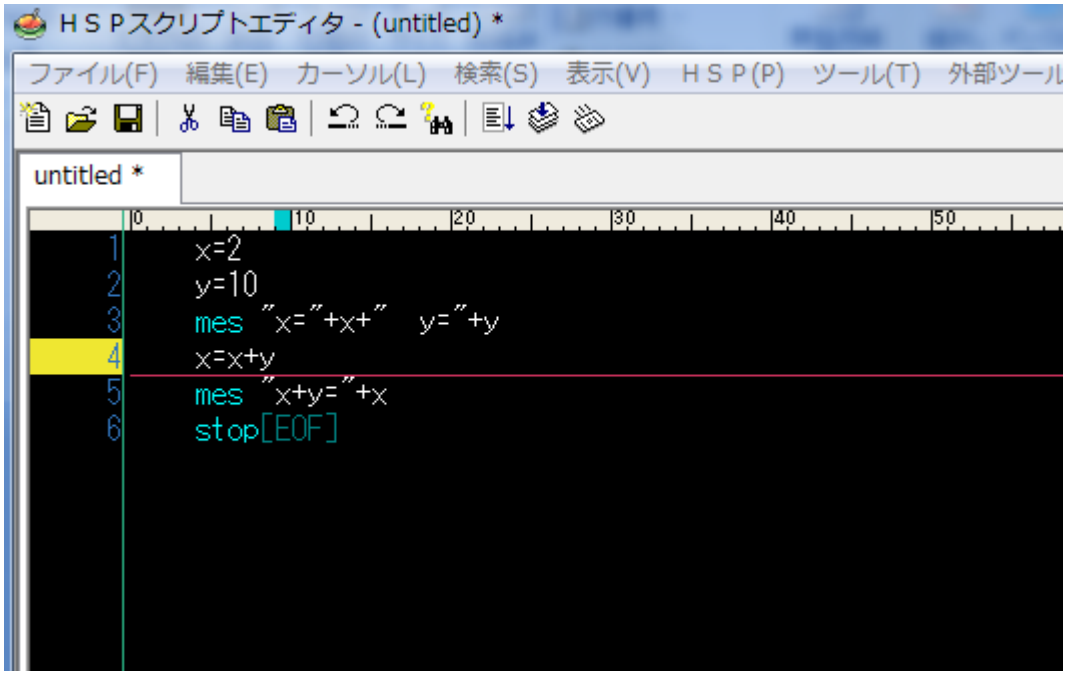
↑のようなファイルたちがたくさん入っていると思いますが、今回は矢印で示した「hsed3」というファイルを開いてください。



こんな感じの画面が出てくると思います。

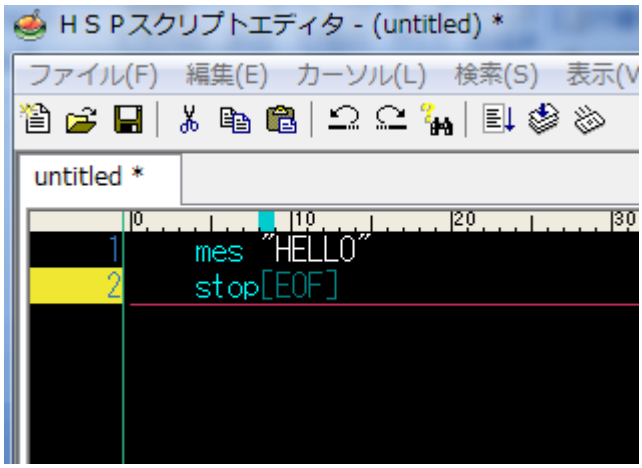
これが、HSP のプログラムをかく画面です。ここにプログラムを書いていきます。

では、まずこれの基本的な使い方を説明します。

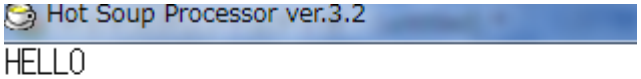


上の図で、黒い部分に白と水色で文字が書いてあります。これらがプログラムです。（たしかデフォルトだと他に緑色の文字とか黄色の文字もあります。一応ツール→オプションで変更できますが、関数とかマクロとかが分かってからじゃないとへたに色を変えるとわかりづらくなる可能性はあります）黒い画面の少し上にゲージがあり、これはカーソルが横軸でどの位置にあるかを水色で表わしていて、今は横軸で９文字目にカーソルがあると示しています。同じように、黒画面左側の数字が、縦軸で何行目かを示していて、今は４行目にカーソルがあるということを示しています。

　では、実際にプログラムを動かしてみます。黒画面に下のように入力してみてください。そのまま入力すれば、mes という部分と stop という部分は水色に変わると思います。



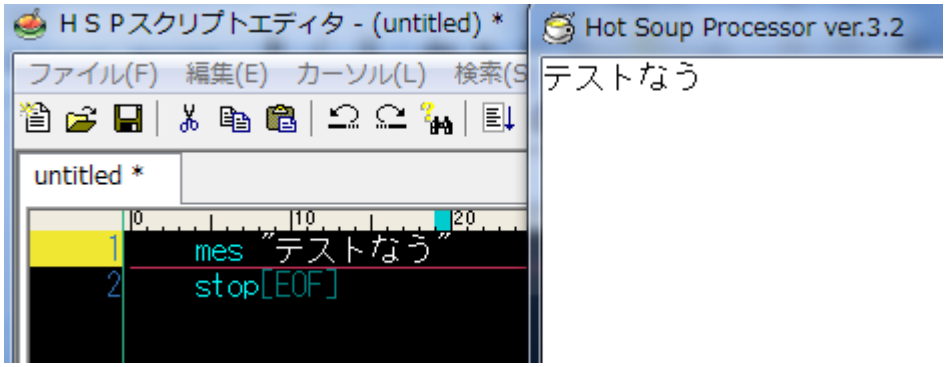
入力したら、今度は画面の上のほうのツールバーにある「HSP」から「コンパイル＋実行(C)」という部分をクリックしてみてください（ショートカットキーが F5 に登録されているので F5 キーを押すことでこれと全く同じ動作をします。）すると、新しい画面が出てきて、次の様に表示されると思います。



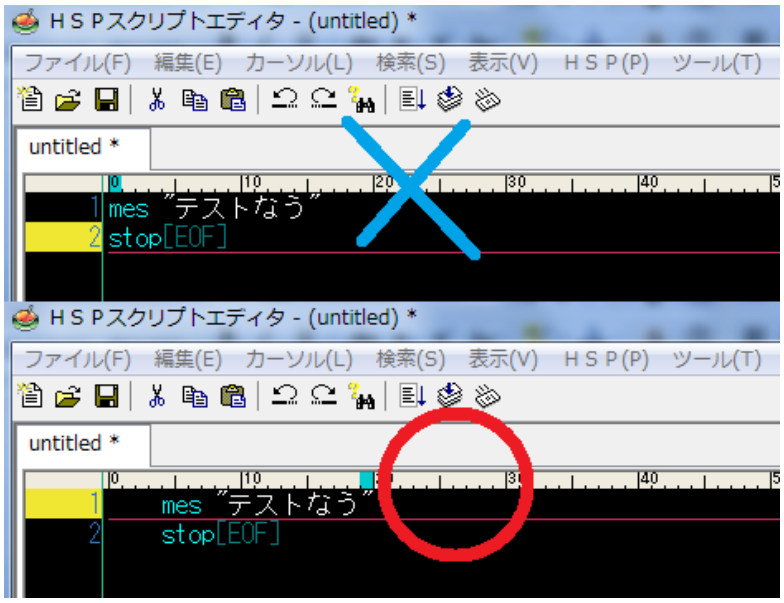
表示されているのが確認できたら、新しく出てきた画面右上の×ボタンを押すか Alt キーを押しながら F4 キー

（Alt+F4）で元の画面に戻ってください。この、「コンパイル+実行」（F5 キー）が、プログラムを動かす方法です。今後、プログラムを動かす、実行すると言う時にはこの動作をする、といった意味で使います。

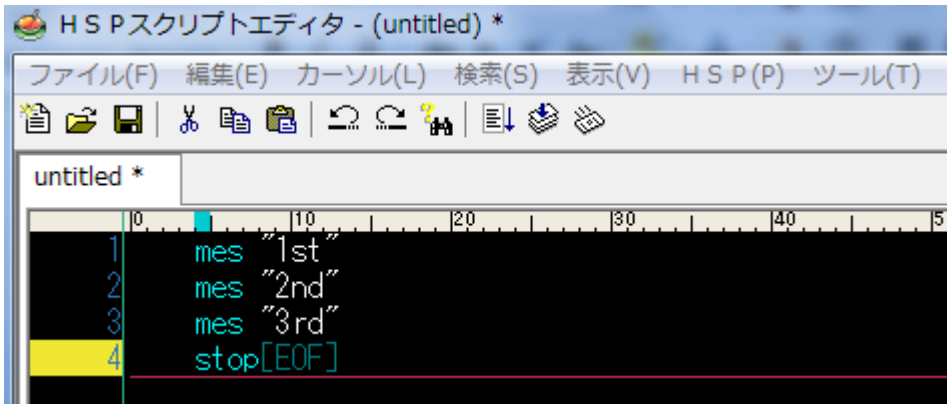
さて、さっきのプログラムを実行したところ、HELLO と表示されました。プログラムは上から順番に処理されることを頭に入れて、プログラムをもう一度見返すと、1 行目に「mes “HELLO”」とありますね。これが、一番はじめのプログラム命令「mes」です（message の略）。“” で囲んだ部分に書かれている文章を表示します。表示する場所を指定することもできますが、指定しない場合はウィンドウの左上がニュートラルポジション（初期位置）になります。では、実際に“HELLO”とある部分の“”の中を、自分の好きな文章や単語に変えて実行してみてください。たぶん、入力した文章が表示されたのではないでしょうか



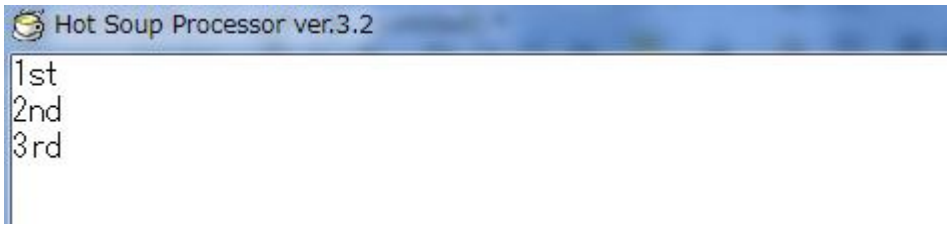
HSP のプログラムを書く時、基本的に左側をタブ 1 つ分空けておいてください（スペースキーのように Tab キーで空白を作る。）後で触れる「ラベル」との兼ね合いや、単純にプログラムが長くなったときの見やすさにつながるので、癖にするようにしてください。



さて、「mes」命令は基本的に文字を表示するときに使う命令ですが、もうひとつ特徴があります、今度は下のようなプログラムを組んでみてください。



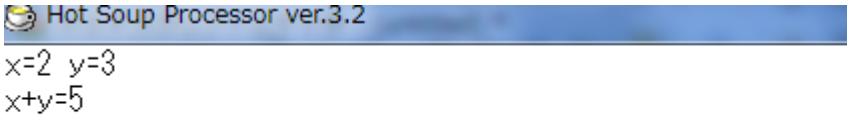
これを実行すると、次の様に表示されます



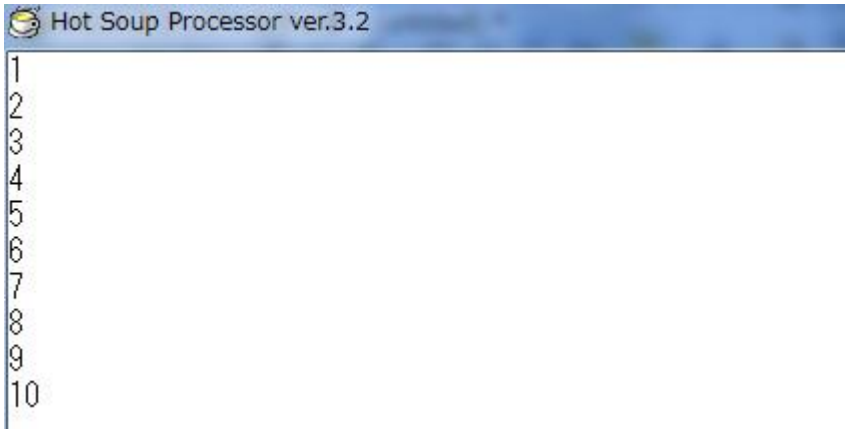
実は、「mes」命令は呼びだすたびに前の mes 命令から改行した位置に文字が表示されるようになります。

さて、今まで放置してきた mes の次に処理される 2 行目の「stop」命令ですが、これはプログラムをこの時点で一時停止するという命令です。基本的にはこの段階ではなくても大丈夫で、本来デバッグ（バグやミスの発見）に使ったりする命令ですが、念のためプログラムを停止させることで万が一の暴走に備えています。実際、先ほど動いたプログラムの「stop」を抜いても問題なく動くのは確認できると思います。が、念のためしばらくの間はプログラムの最後には stop を入れるようにしておいてください。

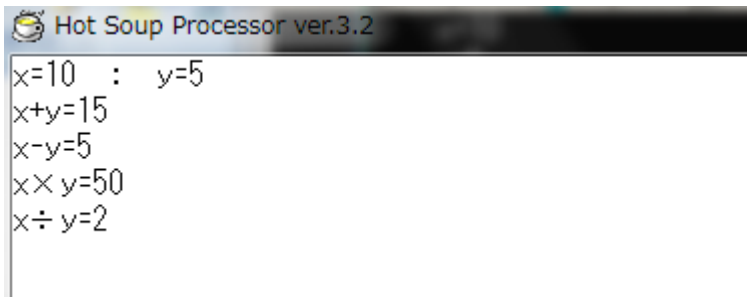
課題１．１　次の様な実行例を表示するプログラムを作りなさい。



課題１．２　次の様な実行例を表示するプログラムを作りなさい。



さて、次に、足し算・引き算などを行い、結果を表示するプログラムを作りましょう。
次のような表示を目標に、説明していきます



x,y を指定して、加減乗除の計算をするプログラムです。
使う命令は、今までの mes を使った表示だけです。これまでの話からこれを表示させようとするとこうなります。

```
mes "x=10  :  y=5"
mes "x+y=15"
mes "x-y=5"
mes "x×y=50"
mes "x÷y=2"
stop[EOF]
```

しかし、ここでｘをたとえば２０に変えたいと思った時、自分で計算して、その都度別の値を入れ替えてプログラムを実行しないといけませんし、プログラムの中で、または実行したときのパソコンの時間でｘやｙの値が変わる場合、この方法は使えません。

ここで登場するのが、「変数」という概念です。変数とは、数字や文字といったものを入れる入れ物のようなものです。たとえば、次のように使います

```
dim number
sdim struct
// 数字を入れる
number = 10
mes number
// 文章（文字列）を入れる
struct = "HELLO"
mes struct[EOF]
```

```
Hot Soup Processor ver.3.2
10
HELLO
```

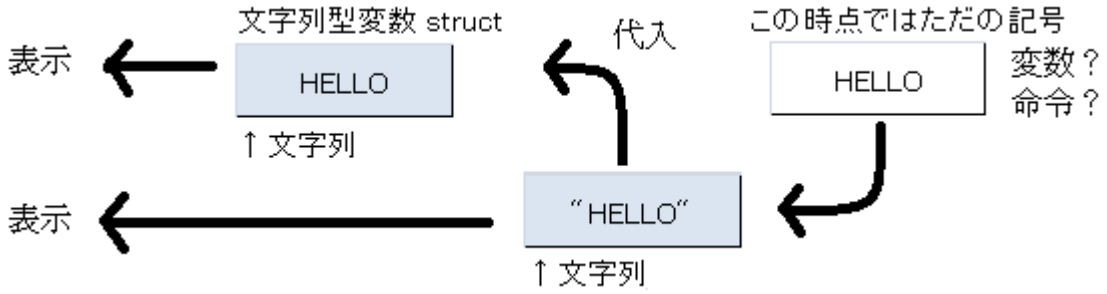
（作った後で stop を入れ忘れたことに気付いたけど気にしない…）はい、いくつか新しいことがでてきました。上のプログラムを見ると、文字が緑色になっている部分がありますが、これは「コメントアウト」といって、プログラムには直接関係ない、メモのような部分になります。「//」の後に続く１行は（デフォルトでは）緑色になり、プログラムに影響しないようになります。なので、上のプログラムと下のプログラムは動作としては全く一緒のものになります。

```
dim number
sdim struct
number = 10
mes number
struct = "HELLO"
mes struct
stop[EOF]
```

さて、では変数について説明します。変数は「この名前は変数として使うよ」という意味を込めて、「dim ~」、「sdim ~」という形式で定義（宣言とも言う）します。「dim」で定義した変数は数値、「sdim」で定義した変数は文字列を入れることができます。（他に実数値（小数など）を入れることができる ddim 等もありますが、説明は先送りにします。）上の例だと、数値を入れることができる変数「number」と文字列を入れることができる

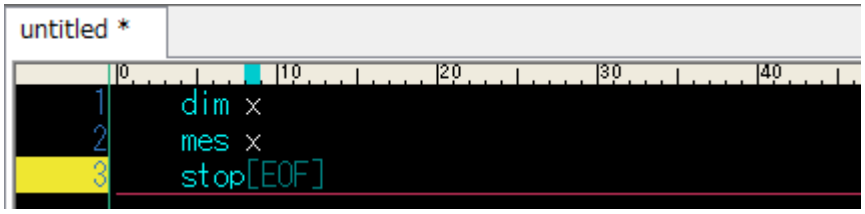
「struct」の 2 つを定義しました。変数の名前は、既に指定されている予約語（関数や命令などのこと。if や mes,stop など）でなければ自由につけることができます。自分であとから見てわかりやすい名前にしておくのがいいと思います。ちなみに、変数の定義は省略することもできなくはないですが、後のために今は定義をしておきます。一応、上のプログラムの上から 2 行を消したり、コメントアウトしたりしても同じように動くことをチェックしてみてください。ＨＳＰ側で要素に合った型の変数を定義してくれます。しばらくは、変数の型について考えたり、どの変数を使っているかをすぐに確認できるように「dim」、「sdim」で定義するようにしてください。

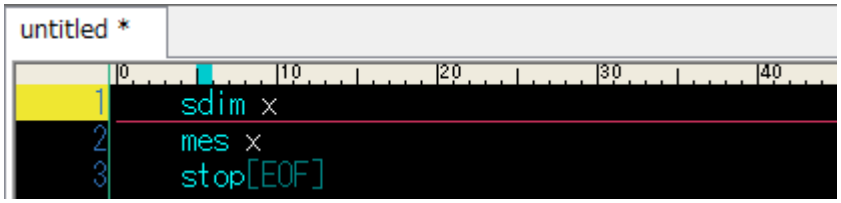
今回は変数「number」、「struct」を作り、それに情報を入れて、表示させました。「変数名 = 入れたい情報」という書式で書くことで、変数に情報を入れることができます。これを「代入」と言います。今後、ちょくちょく使うかもしれません。また「struct」に代入しているのは「"HELLO”」なのに、表示は「HELLO」です。プログラムでは「"~”」のように“で囲まれたものは「文字列」として扱われます。先ほどの mes 表示で文字を”で囲んだのもそのためです。つまり、“” で囲んだものや sdim で定義された変数に入っている文字の列は「文字列」として扱われます。



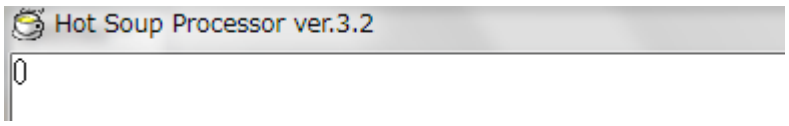
変数に要素が入っているとき、「mes 変数」の形で変数の中身を表示できます。今回は、「mes number」で変数 number の中身を、「mes struct」で struct の中身を表示しています。

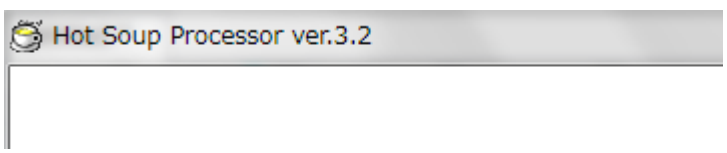
さて、今回は表示の前に変数に値や文字列などの「要素」を代入しましたが、要素を代入しないで、変数を定義してそのまま表示するとどうなるでしょうか





dim 型、sdim 型の変数 x をそれぞれ定義して、表示させてみます

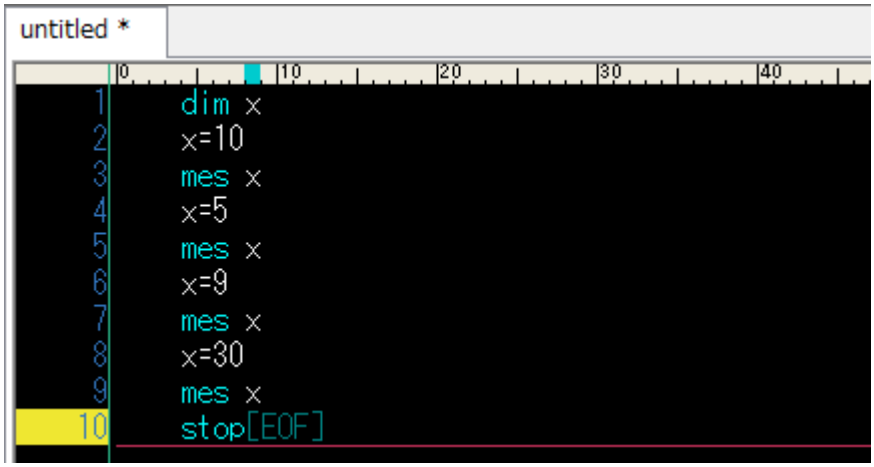


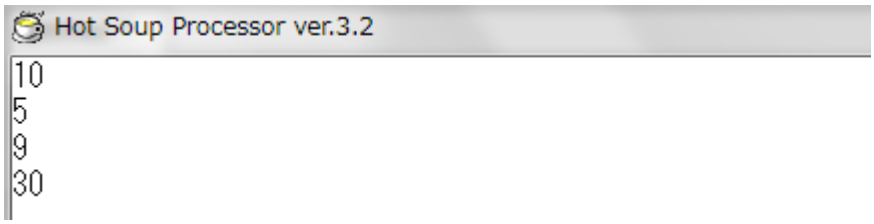


dim で定義した変数には 0 が初期値として入っていて、sdim で定義した変数にはなにも入っていないのがわか

ると思います。初期値としてこれらが変数に入っているというのは覚えておくと使えるかもしれません。

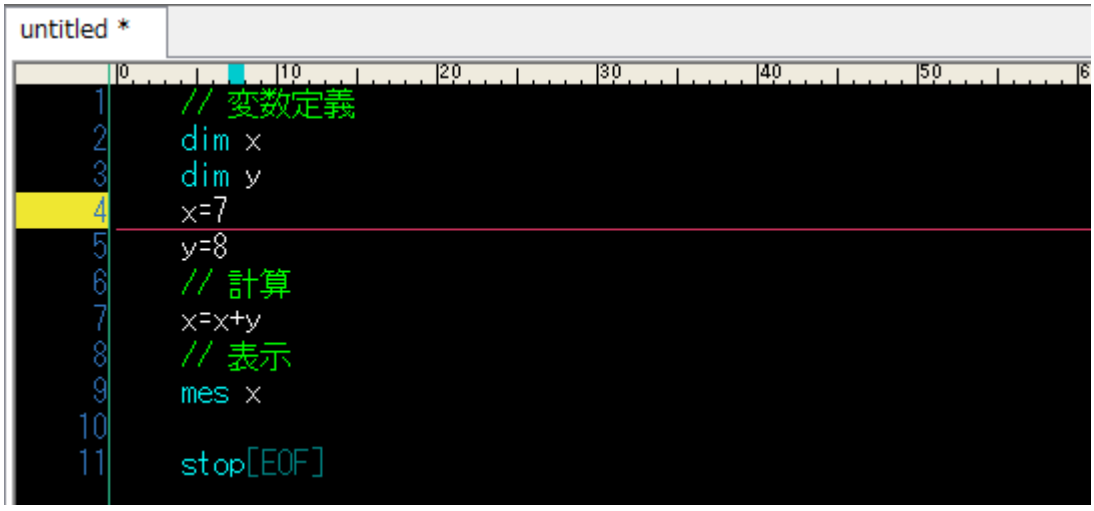
また、変数は何度でも上書きできます。



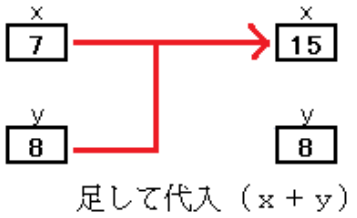


変数の中身は、最後に代入した要素になっているのがわかると思います。

さて、ここからは dim 型変数を使って、話を進めていきます。実は、変数は変数同士で計算したり、といったこともできます。次のプログラムを見てみてください。



変数 x,y の値を足し算して表示するプログラムです。うえから５行で変数 x,y をそれぞれ定義・数値を代入しています。肝心なのは７行目です。「ｘ＝～」は x に～を代入するという意味でした。つまり、今回は、「ｘに x とｙを足したものを代入する」といった意味になります。



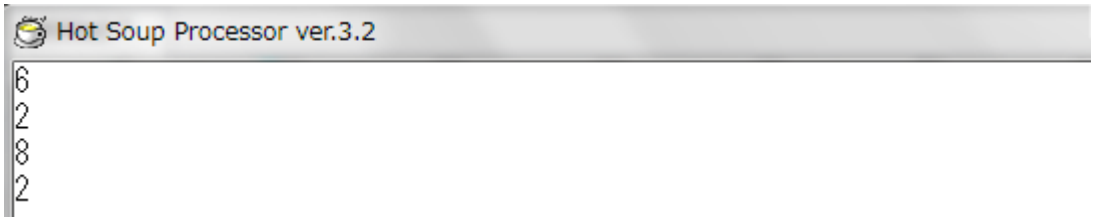
同じように、引き算や掛け算なども計算できます。計算の順序は数学と基本は同じで×÷が＋－より優先です。

数学記号	プログラム上の記号	例	
＋	+	a=x+y	aにx＋yを代入
－	-	a=x-y	aにx－yを代入
×	*	a=x*y	aにx×yを代入
÷	/	a=x/y	aにx÷yを代入

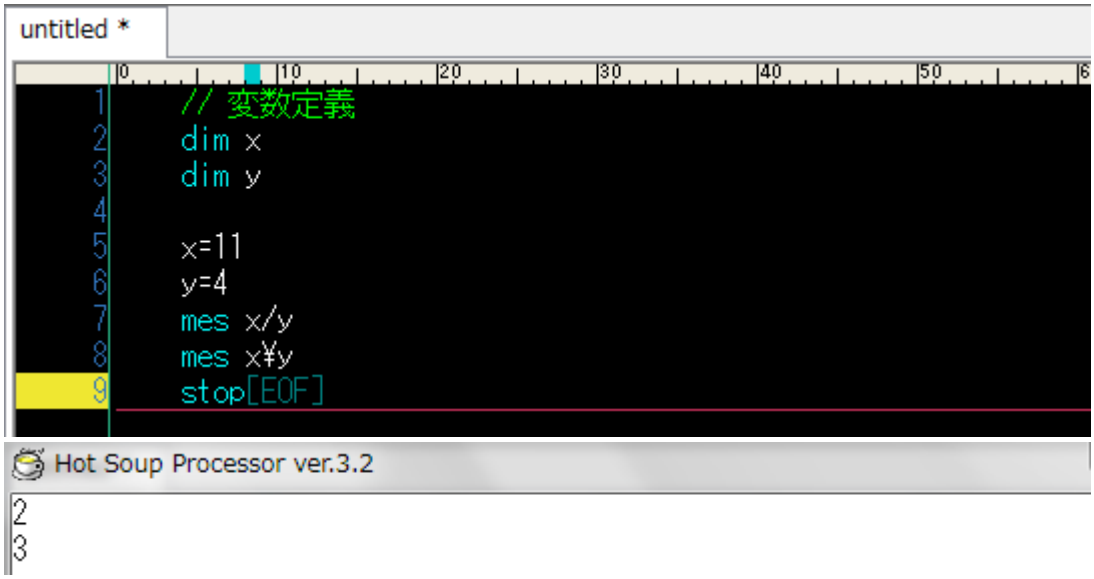
課題 1.3

整数型変数ｘ、ｙをプログラムの中で指定し、それらの足し算、引き算、掛け算、割り算した結果を表示するようにしなさい。ｘ、ｙを次のような値にした時、結果が以下のようになることを確かめなさい。使っていい変数は３つまでとする。なお、先ほどのように「ｘに計算結果を代入して表示」させると、ｘに最初に入っていた「４という数字」が上書きされてしまうので、ｘ＋ｙの表示以降がおかしくなってしまうので注意する。

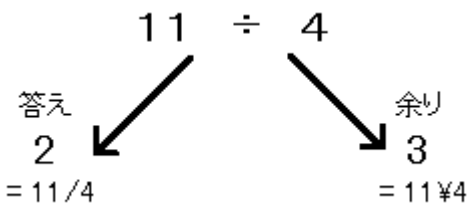
：ｘ＝４、ｙ＝２のとき



また、dim で定義した変数には少し変わった特徴があるので紹介しておきます。割り算のときに発生するので確認しておきます。



このように、計算式の結果を直接表示させることもできます。さて、１１を４で結果は２．７５ですが、表示は２になっています。これは、dim 型の変数は整数型といって、小数点以下は切り捨てる性質を持っているため、２．７５が２になっています。あたらしく「¥」記号が出てきました。これは、整数型（dim 型）変数や整数値どうしで計算する場合に使える、「余りを求める計算記号」です。



dim で宣言した整数型変数は、小数点以下は切り捨てられるというのは、覚えておいてください。小数点以下

を考える場合には実数値型変数（ddim）を使います。ただ、定義の仕方が若干異なるので、あとで「配列」についての説明のときに触れるので、今は飛ばします。

課題 1.4

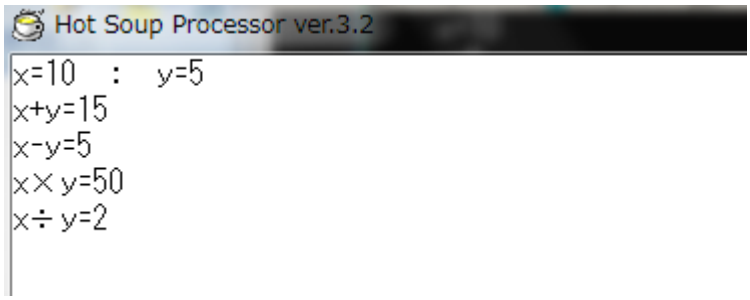
プログラム内で整数型変数ｘを指定し、その２倍、１大きい数、および２乗の３つを表示するプログラムを作りなさい。

たとえば、ｘ＝５とした時、以下のように表示されるようにする。また、変数は１つ（ｘのみ）しか使ってはならないとする。

課題 1.5

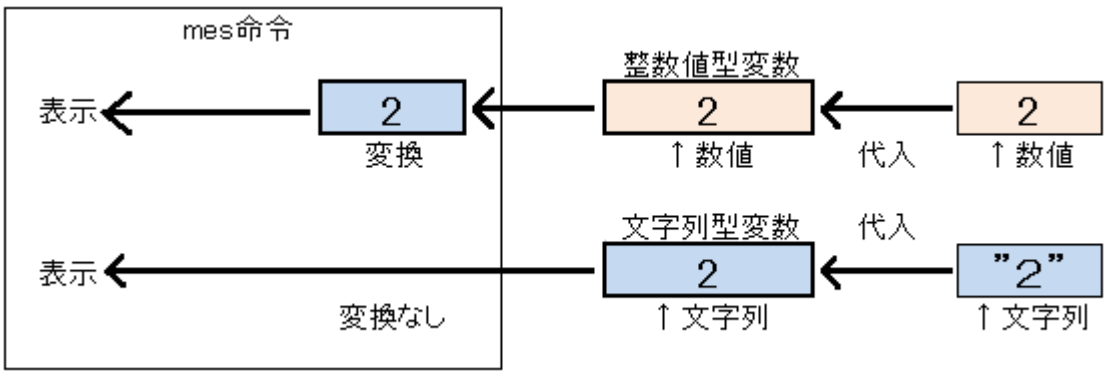
整数型変数ｘをプログラム内で指定し、それが２の倍数（２，４，６…）ならば０を表示し、そうでなければ１を表示するプログラムを作りなさい。使っていい変数は１つまでとする。ヒント…２の倍数かどうかは、２で割ってその剰余で判断する

さて、ここで、最終目標をもういちど見てみましょう

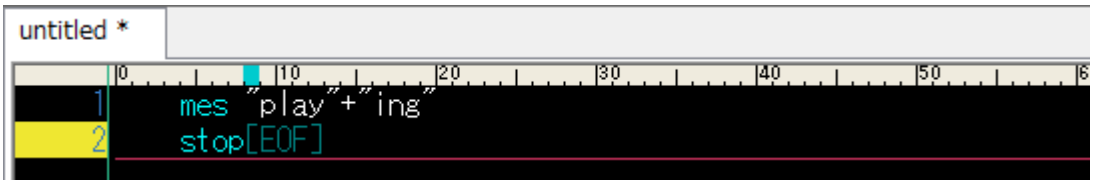


このような表示をするプログラムを作りたいですが、１行の中に変数の数値以外に文章が入っています。これを実装するのは今までの知識に加えて、また１つ新しいテクニックが必要になります。これには、２つほど方法が考えられますが、今回は１つだけを紹介します。（もう１つは次回の内容を使うことで実装できるようになります。）実は、またも mes 命令をつかうのですが、ある工夫によって文字列につなげて変数の値を表示させることができます。

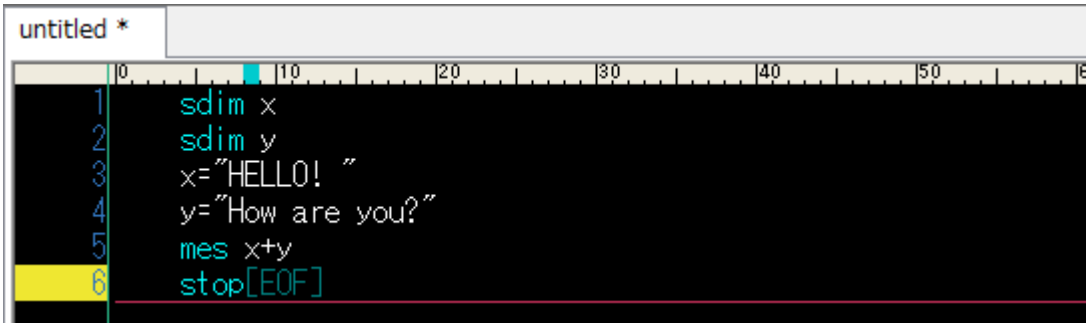
話が少し戻りますが、mes 命令は文字列を表示する命令です。ですが、整数値型変数もその中身を表示できました。実は mes 命令は「表示するときに数値の変数も文字列に変換して表示」をしてくれます



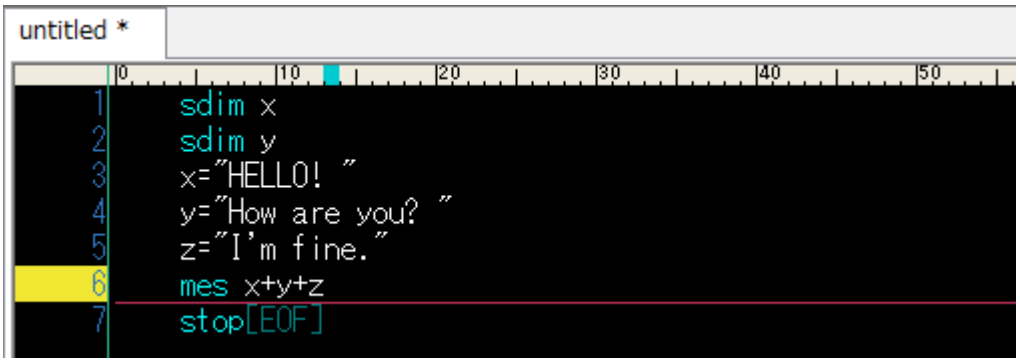
さっき、数値の変数同士で足し算をしました。実は＋にはもう一つ使い方があります。

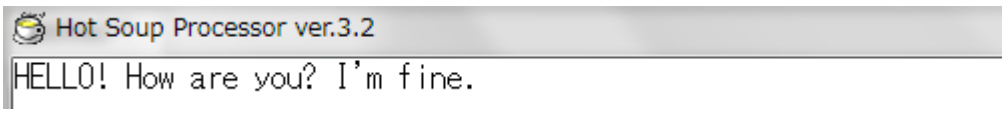


実は、文字列同士でも足し算ができて、文字列同士をつなげることができます。文字列型変数の場合でも試してみましょう。



変数同士でも、文字列の足し算ができていることが確認できると思います。もちろん、３つ以上の文字列の足し算もできます。

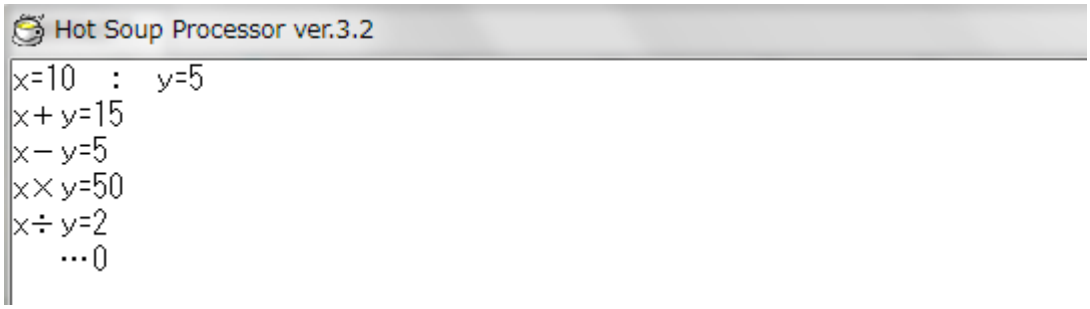




さて、文字列同士の足し算ができ、整数型変数も文字列として扱えることがわかりました。足し算などの計算式に、文字列が最初にでてきている場合には文字列としての足し算、数値が最初に出てきている場合は数値としての足し算になります。ここまでくれば、やり方がわかるのではないでしょうか。

課題 1.6

プログラム中で整数型変数ｘ、ｙの数値を指定し、それらを足し算、引き算、掛け算、割り算した結果を表示するプログラムを作りなさい。使える変数は３つまでとし、実行結果は以下のようになるようにする。また、実行結果はｘ＝１０、ｙ＝５のときのものであるが、ｘ、ｙが別の数値であっても正しく動作することを確認する。また、割り算の結果については、商、余りのどちらも表示させること。



さて、今回は変数操作についてやりました。次回はラベルの操作と、よく使いそうな「命令」や「関数」について説明します。