０．はじめに

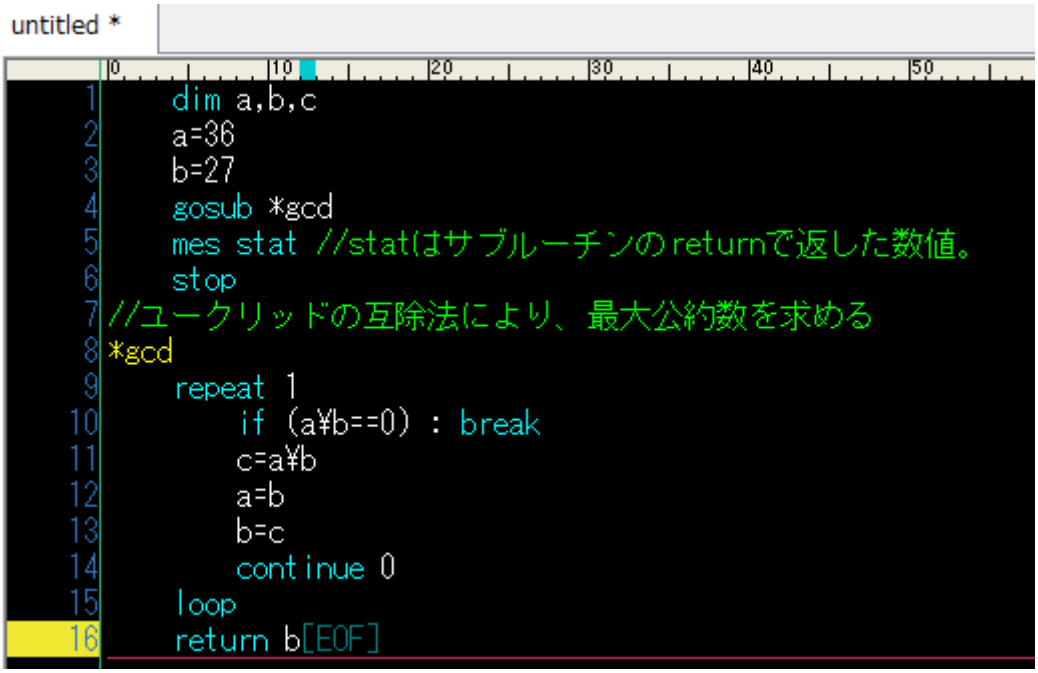
さて、いままで様々な関数や命令を使ってきました。今回はそれらを自分で作る方法や、その活用方法について説明していきます。

１．メインプログラムから分離する処理

同じような処理を繰り返したり、メインループ内を簡略化したりするためにメインプログラムから分離して別の処理をやらせる方法として、関数、命令、サブルーチンの３つがあります。それぞれの特徴は下のようになります

	返り値	呼び出し時の値の引き渡し
関数	あり	あり
命令	なし	あり
サブルーチン	あり	なし

表ではサブルーチンに値の引き渡しがない、と書かれていますが、実際にはサブルーチンに値を渡せないわけではありません。以下のプログラムを見てください。



変数 a や b に値を入れておけば、引き渡しのようなことはできます。上のプログラムで出てきた stat のように、サブルーチンにも値を返させることができます。その場合、返す値の種類によって以下のようなシステム変数に返り値が代入されます。

返り値の種類	代入されるシステム変数
数値(dim)	stat
文字列(sdim)	refstr
実数値(ddim)	refdval

これらのメイン処理から分離させて処理を行うことの利点は，プログラムを見やすくでき

ることです。後から見直すとき，同じような処理を別のプログラムで利用するときに分離している・していないでの差は大きいです。
さて、このように変数経由でサブルーチンに値を渡すことができますが、問題は様々な値で、そのサブルーチンを何度も呼び出す場合です。以下のプログラムを見てみましょう。

untitled *

```
1     dim a,b,c
2     dim x,y,z
3     x=36
4     y=27
5     z=72
6
7     a=x
8     b=y
9     gosub *gcd
10    mes stat //statはサブルーチンのreturnで返した数値。
11    a=x
12    b=z
13    gosub *gcd
14    mes stat
15    stop
16  //ユークリッドの互除法により、最大公約数を求める
17  *gcd
18    repeat 1
19        if (a¥b==0) : break
20        c=a¥b
21        a=b
22        b=c
23        continue 0
24    loop
25    return b[EOF]
```

別々の値で何度も呼び出す場合、値を渡すたびにいちいち変数を代入していては面倒です。しかし、関数や命令ならば、呼び出し時に次のように変数から直接代入できるので便利です。

untitled *

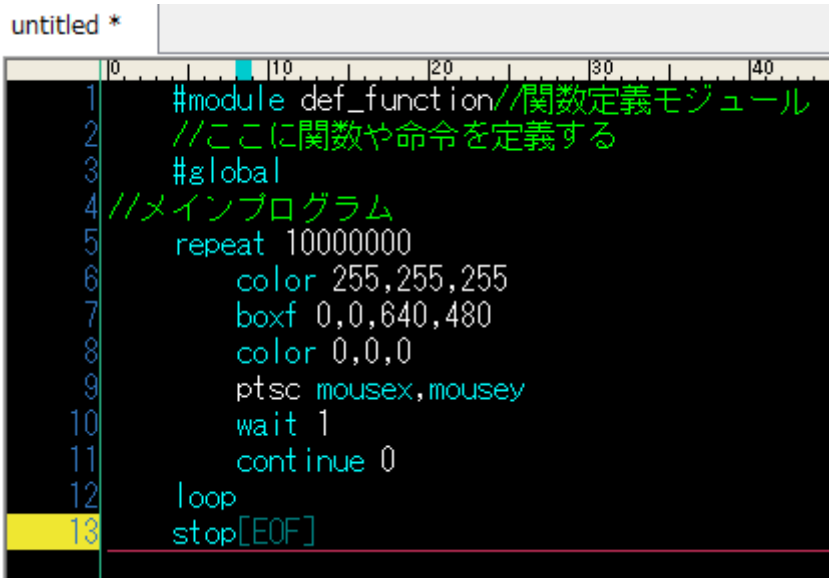
```
1   //事前にユークリッドの互除法により、最大公約数を求める関数を定義しておく
2     dim a,b,c
3     dim x,y,z
4     x=36
5     y=27
6     z=72
7     mes gcd(x,y)//２変数を渡すとその最大公約数を返す関数gcdがあるとする
8     mes gcd(x,z)
9     stop
10  [EOF]
```

よって、以下のように使い分けるといいでしょう。

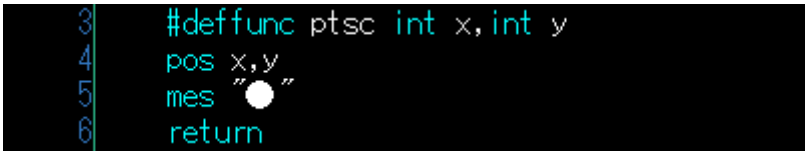
	用途
関数	値を渡す必要があり、値を返す必要もある場合
命令	値を渡す必要があり、値を返さなくてよい場合
サブルーチン	値を渡さなくてよい場合

２．命令のユーザー定義

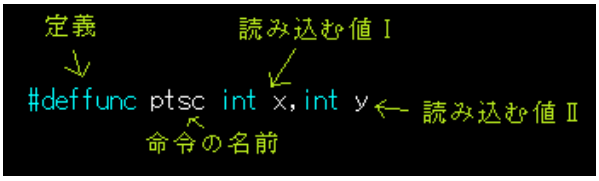
さて、では実際に自分で関数を作っていきましょう。まずそのまえにモジュールについて軽く説明します。モジュールは、プログラム同士をグループ分けするなどに使いますが、今回は#module~#global までの間は処理が飛ばされるものだと思ってくれれば大丈夫です。のちに詳しく説明する予定です。今回は表示座標を与えることで「●」をその座標の場所に表示する関数を作ります。まずは、命令を定義し使用する準備をします。下のようなプログラムを書いてください。



マウス座標である mousex,mousey を呼び出し時に読み込む関数 ptsc を作っていきます。さて、さっそくモジュールが出てきました。１行目から３行目までがモジュール「def_function」になります（「module モジュール名」でモジュールに名前を付けることができます)。関数や命令を定義するときは、とりあいずモジュールの中に書くということにしておきます。(別にモジュールを使わない方法もありますが、今回は今後のモジュールを使います）さて、今回の命令の仕様は、(「ptsc x,y」の形でかくと、x,y をそれぞれ x 座標、y 座標として「●」を表示する）というものです。では、この命令を定義してみましょう。モジュールの中に次のようなプログラムを打ち込んでみてください。



この定義について順を追って説明していきます。まず、deffunc 命令は、「命令を定義する」命令です。deffunc~return で命令を定義します。さて、deffunc 命令は「deffunc 命令の名前 読み込む値」のように定義します。また、読み込む値が今回のように複数ある場合、「,」で区切って複数の読み込みをさせることも可能です。さて、上のプログラムでの定義をもう一度見返してみます。下の図を見てください。



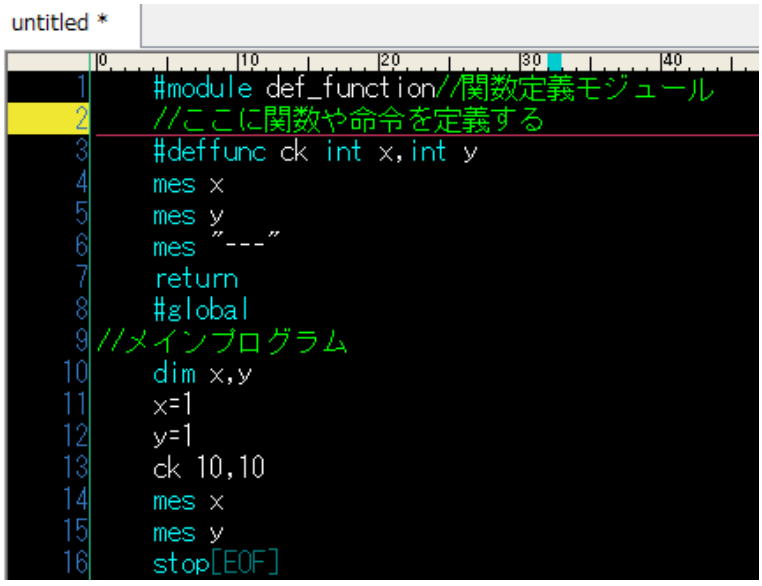
つまり、プログラムとしての動作は以下のようになります。

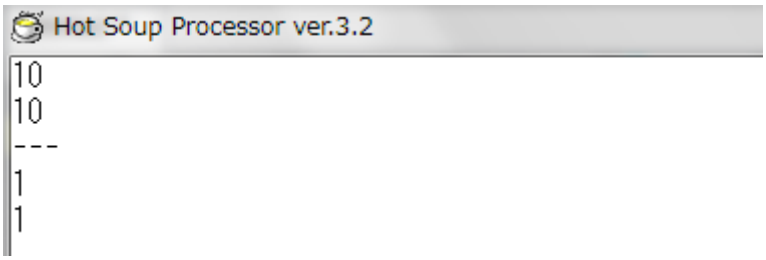
	ptsc	mousex ↓代入	, mousey ↓代入	呼び出し
deffunc	ptsc	int x	, int y	定義

さて、読み込む値として、「int x」のように書いてあります、これらはさらに「変数の型 読み込んだ値を代入する変数」の形に分割できます。「変数の型」は、読み込む値の種類によって以下のように分類されます。

呼び出し値の種類	変数の型
整数値	int
配列でない変数	var
配列変数	array
文字列	str
実数値	double

今回は、整数値を２つ読み込み、それぞれを整数値型変数 x,y に代入するといった意味になります。さて、命令上で変数ｘ、ｙを使っていますが、これらの特徴を見てみましょう。次のプログラムを実行してみてください。

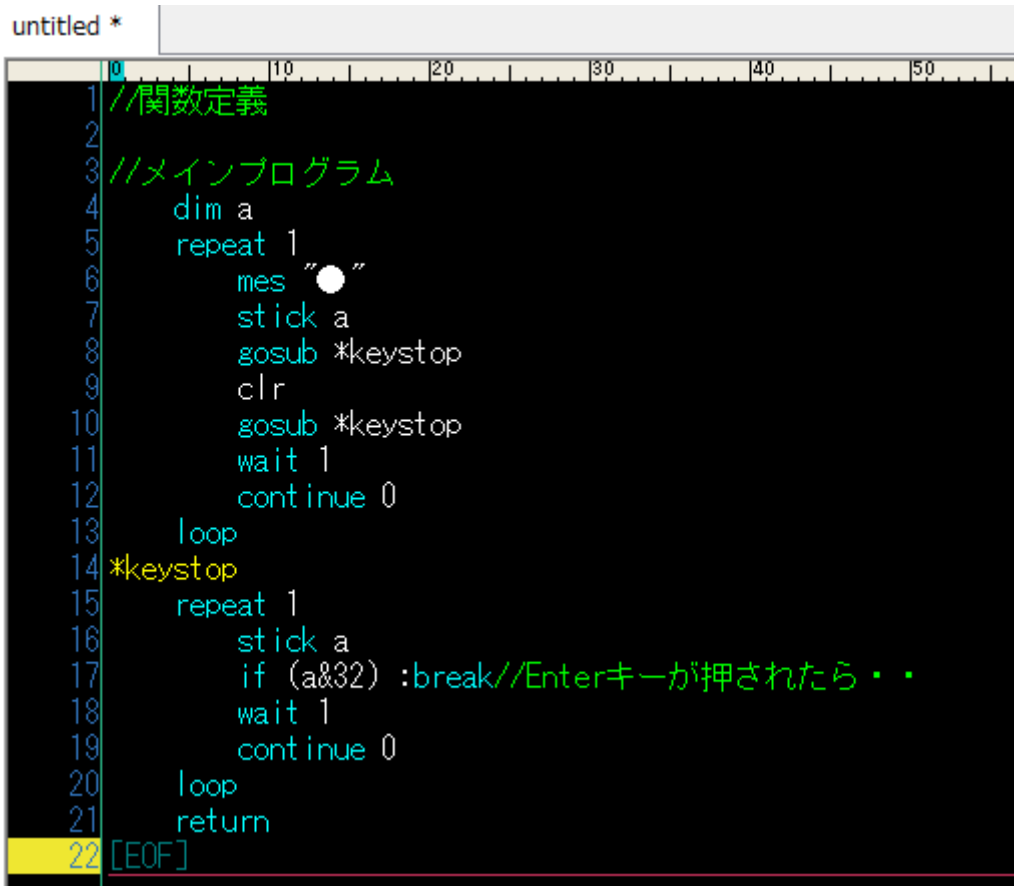




さて、上のプログラムをみるとｘ、ｙを宣言した後、命令定義においてｘ、ｙに１０を代入しているにもかかわらず、メインプログラムに戻ってきた後はｘ、ｙの値はそれぞれ１に戻っています。これは、関数・命令で引数定義した変数は関数・命令内とメインプログラム内では、同じ名前でも違う変数として扱うという性質があるからです。ちなみに、違う名前のモジュールの中とメインプログラムの中でも、同じ名前でも別々の変数として扱います（プログラム内のどこでもその名前は同じ変数として扱う変数として、グローバル変数がありますが、いまのところこれについては省略します)。

課題２．１

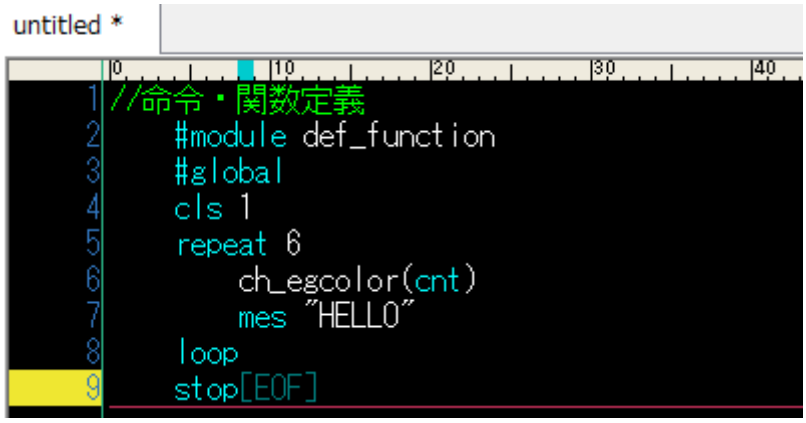
呼び出すと画面を白く塗りつぶす命令 clr を作りなさい。次のプログラムに命令の定義を追加し、動作を確認しなさい。



課題２．２

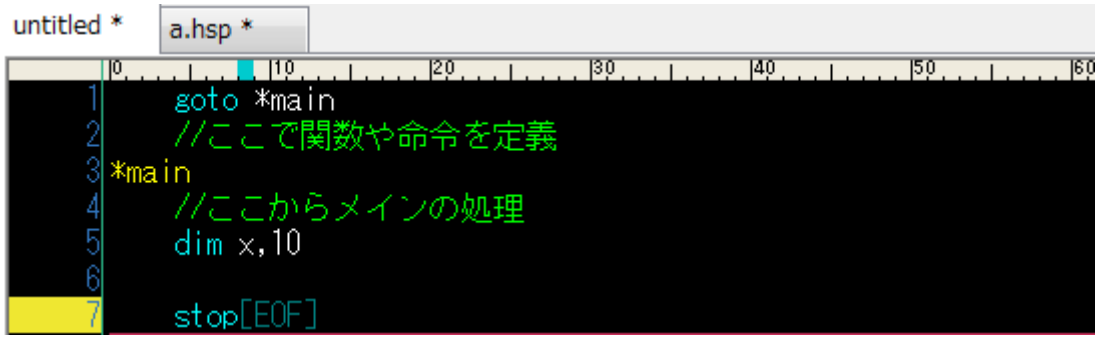
呼び出し時に指定した値によって、文字色を以下のように変更する命令 ch_egcolor()をつくりなさい。なお、動作は下記のプログラムによって行う。

与える値	色
0	白
1	黒
2	赤
3	緑
4	青
5	黄色



課題２．３

メイン関数内で配列 dim x,10 をつくり、呼び出すと配列の要素を逆順に並び替える命令を作りなさい。ただし、今回はメインで定義した配列の要素を変更するために以下のような形で関数定義を行うこと。このように定義すると、メインで定義した配列 x をそのままの名前で、命令内で変更できる。なお、配列に適当な要素を入れて変更後、変更前の状態を表示させて命令が正しく動いていることを確認しなさい。



３．関数のユーザー定義

関数と命令とは、返り値をもつか否かによって異なり，返り値を持つのが関数です。関数も、命令と似たように定義できます。

```
//関数定義
    #module def_function
    //足し算の関数
    #defcfunc add int x,int y
    return x+y
    #global

    dim x
    dim y
    x=10
    y=5
    mes add(x,y)
    stop[EOF]
```

命令が#deffunc 命令で定義されたのに対して，関数は#defcfunc によって定義されます。また，関数の特徴として返り値を return 命令で指定する必要があります。

課題 3.1

引数として 2 つの数値を読み込み，2 つの最小公倍数を返す関数を作成しなさい。なお，a と b の最小公倍数は a×b÷（a と b の最大公約数）で求めることができる。

課題 3.2

引数として数値を読み込むと、２進変換し文字列を返す関数を作成しなさい。
ただし読み込む数値の範囲は０～２５５とし、配列や文字列の足し算を活用しなさい。プログラムの概形として以下のサンプルを利用してもいい。