

第10回：関数

文法：

関数の宣言：

main()の前に,

float calc(float x);

等と書く.

関数の名前(自分で勝手につけて良い)



float calc(float x);



関数の
戻り値の型



引数

数学でいうところの関数とは？

$f(x)=x^3$ とか,

$g(x, y)=x^2+3xy+2y^2+5$ とか.

この場合は,

- x とか, x と y が引数
- f とか g が関数の名前

プログラムを作る場合は, 変数がどのような「型」を持っているかも大事.

関数の本体

```
float calc(float x){
```

ここに処理を書く; ← yに返すべき値を入れておく

```
return (y);  
}
```

例えば, 入力されたものを2乗する関数の場合は,

```
float calc(float x){  
    float y;  
    y=x*x;  
    return (y);  
}
```

注意:

1. 中で使う変数は, 定義をしないといけない
(float y)
2. 引数で定義するれば, それは有効 (float x)
3. 返せるものは, 一つだけ.

関数の呼び出し方

`z=calc(w);`

などとする. 今の例では, w の2乗の値が z に入る.

ここの部分はこれまで良く使ってきた
代入と同じ.

注意:

変数はそれぞれの変数の
中で整合性が取れていればよい。

全体のプログラム

```
#include <stdio.h>
```

```
float calc(float x);  
int main(void);
```

関数の宣言

```
float calc(float x)  
{  
    float y;  
    y=x*x;  
    return (y);  
}
```

関数calcの定義

```
int main()  
{  
    char buf[10];  
    float w, z;  
    gets(buf);  
    w=atoi(buf);  
    z=calc(w);  
    printf("%f¥n", z);  
    return (0);  
}
```

関数mainの定義

関数calcの呼び出し

「関数」を使うことの利点

小さいプログラムでは、大してメリットはない。しかし...

1. プログラムがわかりやすくなる.
2. 関数にしておけば、使い回しが簡単.

一回関数を使ってしまえば,
いろいろなプログラムに使い回しが可能.

実際に、(誰かが作ってくれた)多くの関数を使っている.

gets, atoi, printf, sqrt, ...

使うときに気にするのは,

使い方(入力は何?何を計算するのか?)のみ.

中身は知らなくて良い.