# Q 関数

ある1つの値を入力すると
ただ1つの値が出力される
計算を関数という.

✓ 数学

$f(x) = 2x + 3$

$f(3)\ (= 2 \cdot 3 + 3) = 9$

3: 入力　9: 出力

✓ 日常生活

入金 → [お釣り関数] → おつり

入金: 入力　おつり: 出力

〜→ C言語でも同じ事が言える.

Q ちなみに.

```
int main(void){
    return 0;
}
```

return 0; の 0 は int 型

例えば"

- return 1; なら.
  int 関数名(引数){ }
- return 'a'; なら
  char 関数名(引数){ }
- return 3.14; なら
  float 関数名(引数){ }

Q 構文.

```
戻り値の型　関数名(引数){
    処理の内容;
    return 値;
}
```

(インデント)

(例)

```
#include <stdio.h>

int Sum(int a, int b){
    int c = a+b;
    return c;
}

int main(void){
    int x = 10;
    int y = 5;
    int z;
    z = Sum(x,y);
    printf("%d \n", z);
    return 0;
}
```

printf("%d \n", z); → 15 が表示

Sum(x,y); が実行されると

Sum(int a, int b); (x,y: 代入される)

つまり a = 10; b = 5;

Sum関数内では.

int c = a+b; (= 10+5)

が実行され.

それが return c; される.

z = Sum(x,y); (Sum(); の処理結果)

(つまり. z = c; )

# 関数の注意点

## ✓ 引数が複数の場合

```
int Sum(int a, int b){…}
```

↑ 引数が複数の場合カンマで区切る.

## ✓ 呼び出す際

```
int Sum(int a, int b){…}

z = sum(x, y);
```

関数を呼び出す際は
定義された引数の型と渡す引数の型をそろえる.

## ✓ 実行時の優先度

```
int Sum(int a, int b){
    :
}

int main(void){
    :
}
```

実行時はmain関数が最初に実行

## ✓ 関数の定義位置.

```
#include <stdio.h>

使うよりも前に定義する
int main(void){
    z = Sum(x, y);
}
int Sum(int a, int b){…}
```

見に行く

## ✓ プロトタイプ宣言

```
int Sum(int a, int b);

int main(void){
    z = sum(x, y);
}

(int a, int b){
    :
}
```

## ✓ 変数の有効範囲(スコープ)

```
#include <stdio.h>

int a;
int my_function(int x){
    int b;
    return x;
    bとxの有効範囲
}
int main(void){
    int x;
    int y = my_function(x);
    return 0;
    xとyの有効範囲
}
```

別物

aの有効範囲

aをグローバル変数
b, x, x, yをローカル変数

## ✓ ローカル変数の寿命

```
void f(void){
   int a;   ← 宣言により、リソースを確保
   a = 1;   ← 変数の利用
   return;  ← 関数終了と同時に破棄
}
```

変数aを

## ◎ static 変数

<構文>

```
static 型名 変数名;
```

✓ 関数の中でしか利用しない
✓ 関数の処理が終わっても破棄しない

```
#include <stdio.h>
void f(void){          ← 最初の1回のみ実行
   static int num = 0;
   num++;
   printf("%d回呼ばれました\n", num);
   return;
}
int main(void){
   f(); ▶ 1回呼ばれました.
   f(); ▶ 2回呼ばれました.
   f(); ▶ 3回呼ばれました.
   return 0;
}
```