

第6回 C言語演習

関数

東京工科大学 加納 徹

本日の流れ

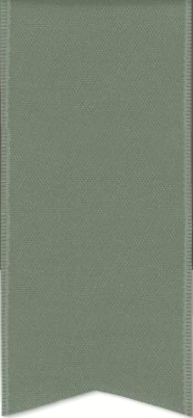
1. 関数って何？

2. 関数の作り方

3. 練習問題

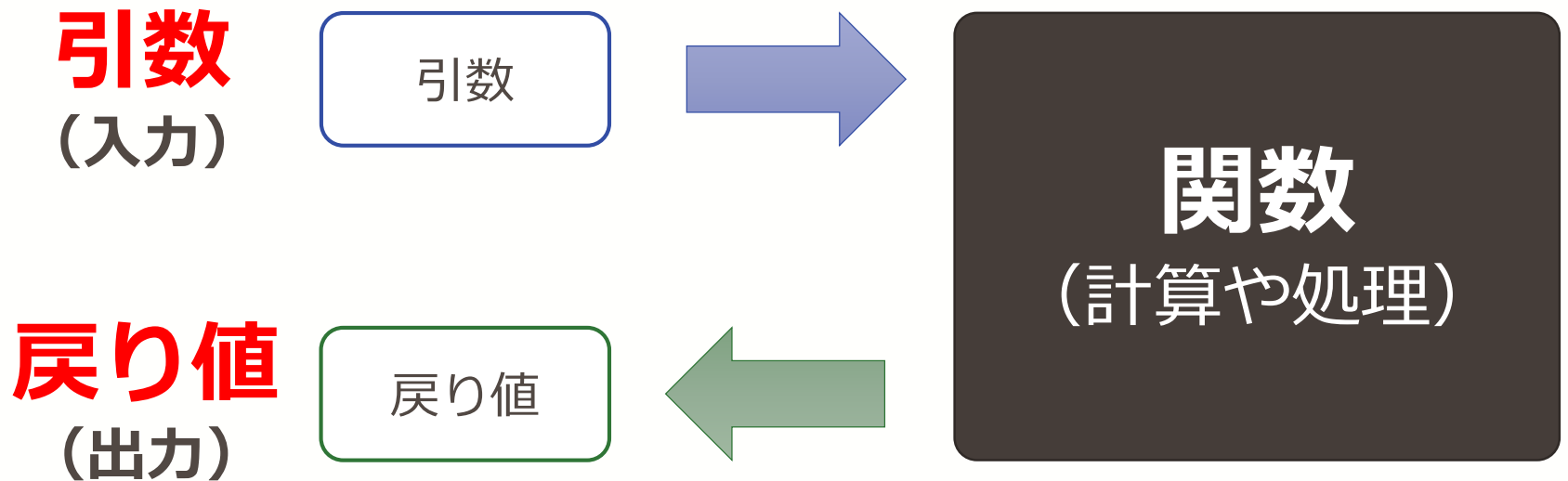
4. まとめ

5. 課題とおまけ問題



1. 関数って何？

関数って何？



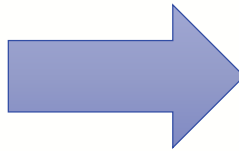
数学の関数は、決められた **計算** をするもの
プログラミングの関数は、決められた **仕事** をするもの



関数って何？

引数
(入力)

$$x = 2$$

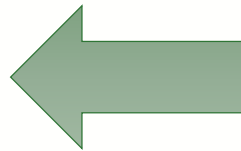


関数

$$f(x) = x^2 + 3x$$

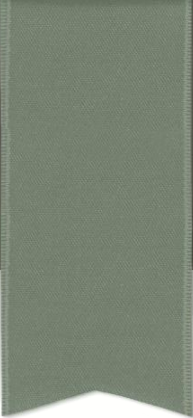
戻り値
(出力)

10



一度関数を作れば何度でも再利用することができる
長いコードを書く場合には必要不可欠





2. 関数の作り方

$f(x) = x^2 + 3x$ の計算

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

```
int main(void){  
    int ans;
```

```
    // f(2) の計算
```

```
    ans = 2 * 2 + 3 * 2;
```

```
    printf("f(2) = %d\n", ans);
```

```
    // f(-1) の計算
```

```
    ans = (-1) * (-1) + 3 * (-1);
```

```
    printf("f(-1) = %d\n", ans);
```

```
    return 0;
```

```
}
```

ここを関数にしたい

関数の作り方①

戻り値の型

関数の名前

引数

```
int calc(int x) {
```

関数の処理内容

(計算や文字の表示等)

```
}
```

戻り値や引数がない場合は、**void** (空っぽの意味) と記入



関数の作り方②

計算結果用の変数

```
int calc(int x) {
```

```
    int res;
```

計算内容の記述

```
    res = x * x + 3 * x;
```

```
    return res;
```

```
}
```

return で呼び出し元に結果を返す

引数として受け取った値は x に格納され、関数内で自由に使える



関数の作り方③

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

```
int calc(int x);
```

プロトタイプ宣言
この関数を使うということを
事前にプログラムに教える

```
int main(void) {
```

```
    int ans;
```

```
    ans = calc(2);
```

```
    printf("f(2) = %d¥n", ans);
```

```
    return 0;
```

```
}
```

関数の呼び出し

```
int calc(int x) {
```

```
    int res;
```

```
    res = x * x + 3 * x;
```

```
    return res;
```

```
}
```

関数の作り方④（まとめ）

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

```
int calc(int x);
```

関数のプロトタイプ宣言

```
int main(void){  
    int ans;
```

```
// f(2) の計算
```

```
ans = calc(2);
```

関数の呼び出し

```
printf("f(2) = %d¥n", ans);
```

```
// f(-1) の計算
```

```
ans = calc(-1);
```

```
printf("f(-1) = %d¥n", ans);
```

```
return 0;
```

```
}
```

関数の内容 ($x^2 + 3x$ を返す関数)

```
int calc(int x) {
```

```
    int res;
```

```
    res = x * x + 3 * x;
```

```
    return res;
```

計算結果を返す

```
}
```

コードを短く美しく

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

```
int calc(int x);
```

```
int main(void){
```

```
    // f(2) の計算
```

```
    printf("f(2) = %d¥n", func(2));
```

```
    // f(-1) の計算
```

```
    printf("f(-1) = %d¥n", func(-1));
```

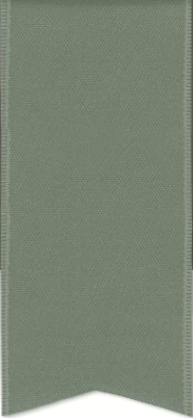
```
    return 0;
```

```
}
```

```
int calc(int x) {  
    return x * x + 3 * x;  
}
```

関数の呼び出し

よりシンプルな記述



3. 演習問題

練習問題

問題 1

int 型の引数 n を受け取って、 $1 \sim n$ までの整数の和を
戻り値として返す関数 `calc_sum1` を作りなさい。
(次のページのコードを使っても良い)

問題 2

int 型の引数 m と n を受け取って、 $m \sim n$ までの整数
の和を戻り値として返す関数 `calc_sum2` を作りなさい。

まずは関数を使わない場合を考えてみよう



関数を使わない場合

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

```
int main(void){
```

```
    int i, sum;
```

```
    // 1 から 10 までの和
```

```
    sum = 0;
```

```
    for (i = 1; i <= 10; i++) {  
        sum += i;  
    }
```

ここを関数にしたい

```
    printf("1 から 10 までの和は、 %d¥n", sum);
```

```
    // 1 から 100 までの和
```

```
    sum = 0;
```

```
    for (i = 1; i <= 100; i++) {  
        sum += i;
```

```
    }
```

```
    printf("1 から 100 までの和は、 %d¥n", sum);
```

```
    return 0;
```

```
}
```

問題 1 解答例

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

```
int calc_sum1(int n);
```

プロトタイプ宣言

```
int main(void){
```

```
    // 1 から 10 までの和
```

```
    printf("1 から 10 までの和は、 %d¥n", calc_sum1(10));
```

```
    // 1 から 100 までの和
```

```
    printf("1 から 100 までの和は、 %d¥n", calc_sum1(100));
```

```
    return 0;
```

```
}
```

関数の呼び出し

```
int calc_sum1(int n) {
```

```
    int i, sum = 0;
```

```
    for (i = 1; i <= n; i++) {  
        sum += i;
```

```
    }
```

```
    return sum;
```

```
}
```

1 から n までの和を計算
して結果を返す関数

問題 2 解答例

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

```
int calc_sum2(int m, int n);
```

プロトタイプ宣言

```
int main(void){
```

```
    // 5 から 10 までの和
```

```
    printf("5 から 10 までの和は、 %d¥n", calc_sum2(5, 10));
```

```
    // 10 から 100 までの和
```

```
    printf("10 から 100 までの和は、 %d¥n", calc_sum2(10, 100));
```

```
    return 0;
```

```
}
```

引数を2つ与えて呼び出し

```
int calc_sum2(int m, int n) {
```

```
    int i, sum = 0;
```

```
    for (i = m; i <= n; i++) {  
        sum += i;
```

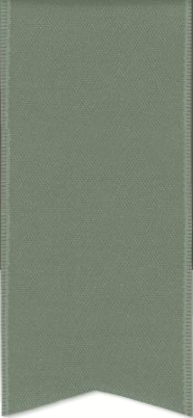
```
    }
```

```
    return sum;
```

```
}
```

引数として m と n の2つを受け取る

m から n までの和を計算
して結果を返す関数



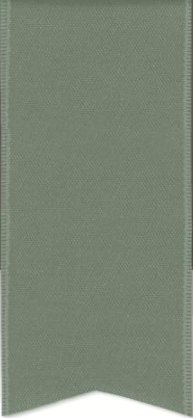
4. まとめ

関数のまとめ

- 関数とは、決められた **仕事** をするもの
- 関数は、**[戻り値の型] [関数名] ([引数])** の形で定義される
- 関数を使う際は、必ず **プロトタイプ宣言** をする
- 戻り値がある場合は、**return 文** で記述する
- 戻り値や引数が無い場合は、定義の際に **void** を記述する

C言語を書くことは、関数を書くことに等しい





5. 課題 + おまけ問題

課題

課題 1

int 型の引数 n を受け取って、その絶対値を返す関数 `calc_abs` を作りなさい。

課題 2 - A

int 型の引数 n を受け取って、 $n!$ を返す関数 `calc_fact` を作りなさい。
また、 n に様々な値を入れて結果を検証しなさい。

課題 2 - B

int 型の引数 n と k を受け取って、 $nCk \left(= \frac{n!}{k!(n-k)!} \right)$ を返す関数 `calc_comb` を作りなさい。ただし、関数 `calc_fact` を利用すること。

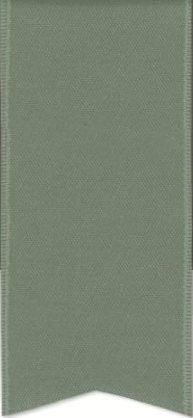
おまけ問題

関数 `calc_fact` を、for ループを使わずに実装しなさい。

ヒント：関数の中から自分自身の関数を呼び出すこともできる（再帰）

できるかな？





終わり