プログラミング応用

http://bit.ly/ouyou3d

再帰処理

前期 第10週

2018/6/26

本日は・・・

再帰アルゴリズムと

再帰呼び出しについて学びます

「再帰」とは・・・

【再帰的（recursive）】
ある事象が自分自身を含んでいたり、それを用いて定義されている状態

再帰アルゴリズム（recursive algorithm）

- 再帰的な記述により、簡潔に書かれたアルゴリズム
- 再帰を用いない場合よりもしばしば効率的になる

再帰アルゴリズムの3つのポイント

1からnまでの合計を求めるプログラム

```
1: #include <stdio.h>
2:
3: int sum(int n)
4: {
5:     if(n > 0) {
6:         printf("%d + ", n);
7:         return n + sum(n-1);
8:     } else {
9:         printf("%d = ", n);
10:        return n;
11:    }
12: }
13:
14: int main(void)
15: {
16:    printf("%d\n", sum(5));
17:
18:    int result = sum(10);
19:    printf("%d\n", result);
20:
21:    return 0;
22: }
```

【Point 1】再帰呼出しをカウントできる引数を用意する

【Point 2】再帰呼び出しに対する条件を付ける

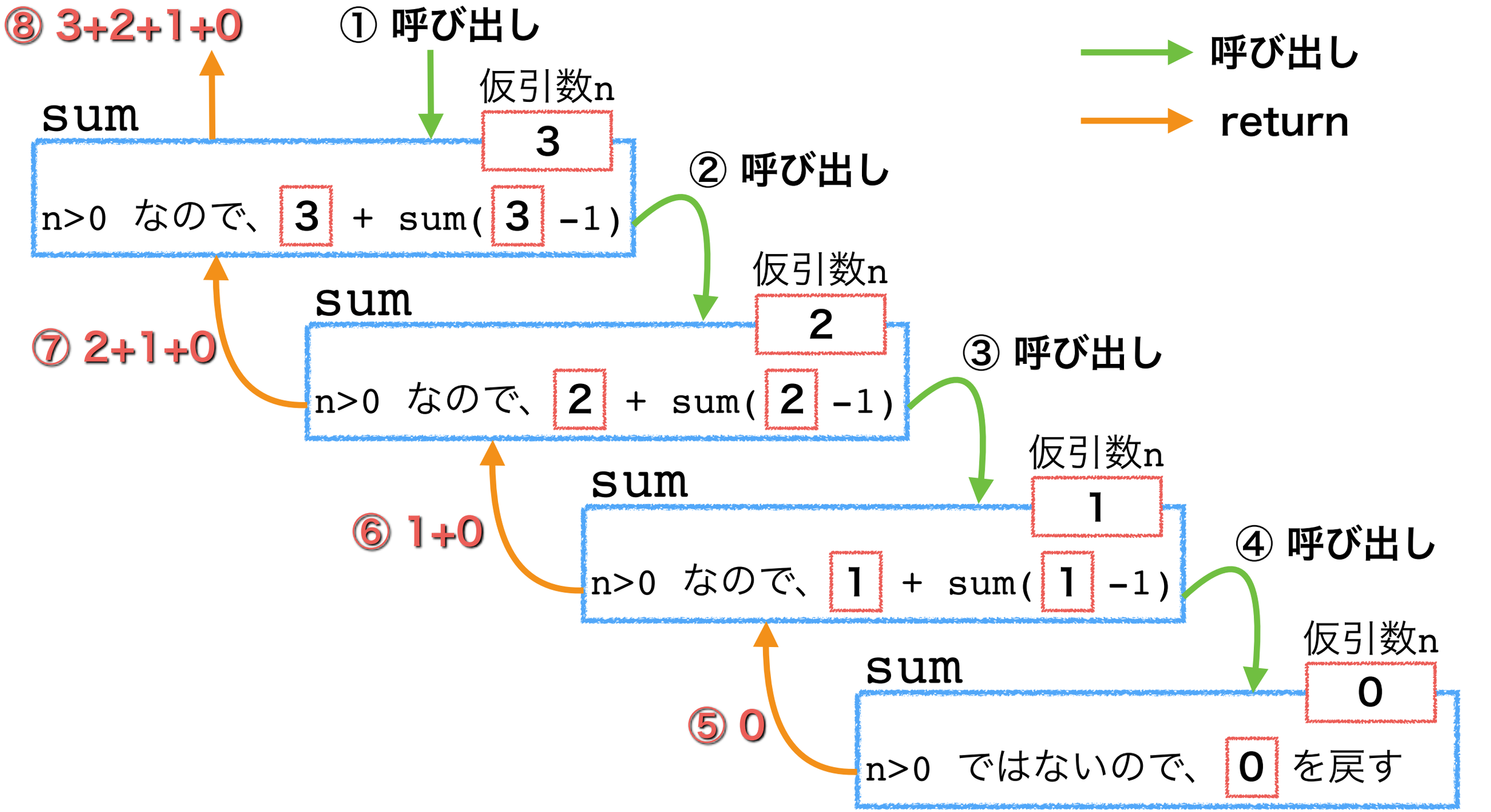
【Point 3】カウントする引数の値を変えて再帰呼び出しをする

再帰呼び出しを開始する（1〜5の合計を求める）

再帰呼び出しの戻り値を変数に入れてから表示してもよい（1〜10の合計を求める）

再帰の動き

```
printf("%d\n", sum(3));
```

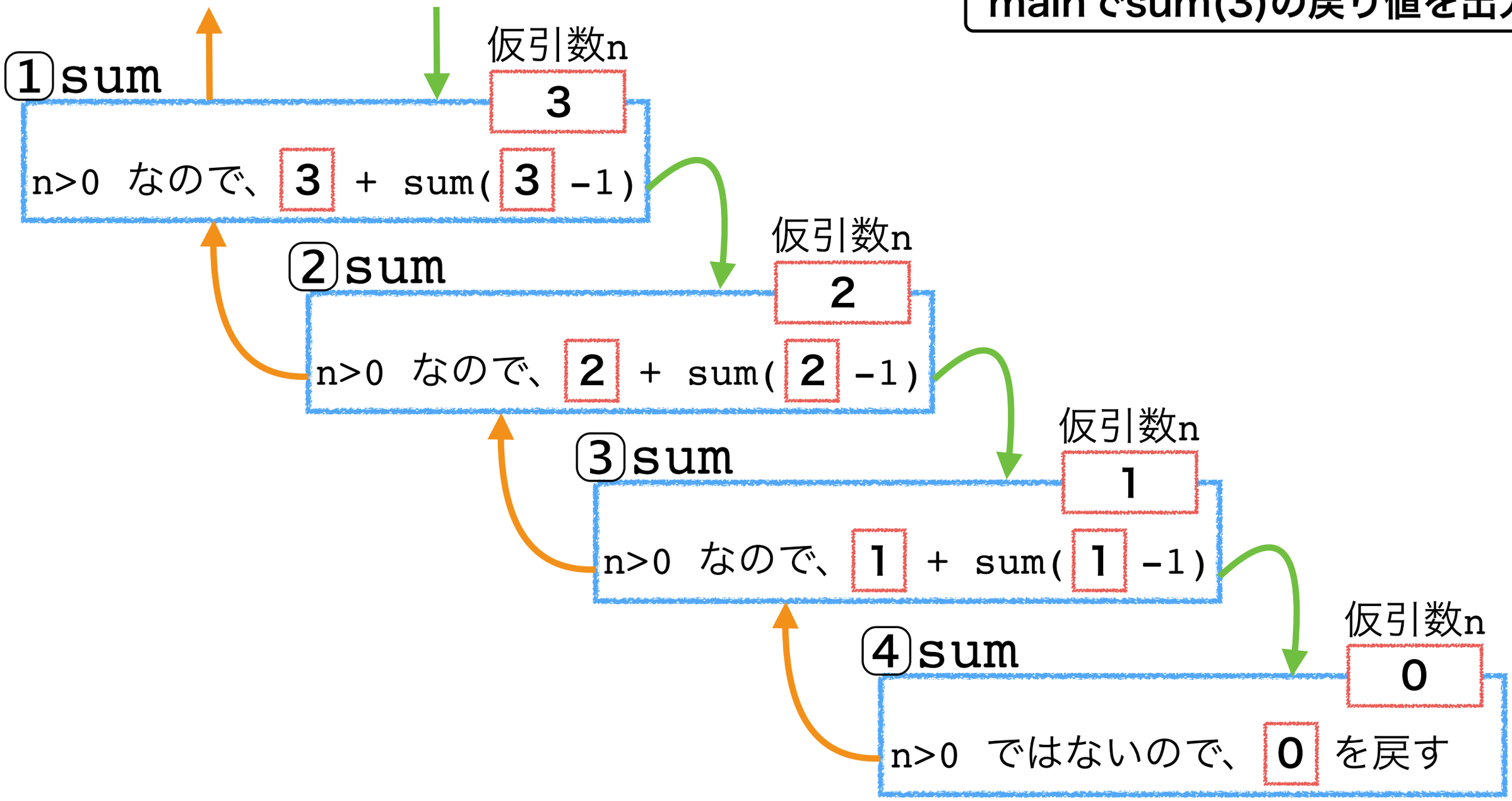


再帰の動き（出力）

3 + 2 + 1 + 0 = 6

① ② ③ ④

mainでsum(3)の戻り値を出力



【練習10-1】

「1からnまでの合計を求める」サンプルプログラム
をコンパイルして、実行結果を確認しましょう。

【課題10-1】

練習10-1のsum()を参考にして、再帰呼び出しを利用して「1からnまでの偶数の合計を求める」関数sum_even()を作成してください。ただし、nは必ず偶数が与えられるものとします。

[この関数のプロトタイプ宣言]

```
int sum_even(int n);

   /* 練習10-1の関数sum()を参考に作れる */
   /* 再帰呼び出しする際に、nを2ずつ減らすようにする */
```

【課題10-1】

[mainでの処理]

```
  printf("%d\n", sum_even(6));
  printf("%d\n", sum_even(10));
```

[実行結果]

```
  6 + 4 + 2 + 0 = 12
  10 + 8 + 6 + 4 + 2 + 0 = 30
```

【課題10-2】

課題10-1のsum_even()を参考にして、再帰呼び出しを利用して「1からnまでの奇数の合計を求める」関数sum_odd()を作成してください。ただし、nは必ず奇数が与えられるものとします。

[この関数のプロトタイプ宣言]

```
int sum_odd(int n);

  /* 課題10-1の関数sum_even()を参考に作れる */
  /* nが1になるまで再帰呼び出しするように条件を変える */
```

【課題10-2】

[mainでの処理]

```
  printf("%d\n", sum_odd(7));
  printf("%d\n", sum_odd(15));
```

[実行結果]

```
  7 + 5 + 3 + 1 = 16
  15 + 13 + 11 + 9 + 7 + 5 + 3 + 1 = 64
```

【課題10-3】

再帰呼び出しを利用して「nの階乗（n!）を求める」関数kaijo()を作成してください。

[この関数のプロトタイプ宣言]

```
int kaijo(int n);

  /* nの階乗は、1からnまでの値をかければ求まる */
  /* 再帰呼び出しの際の演算を「かけ算」にする */
  /* nが1になるまで再帰呼び出しするようにifの条件を作る */
```

【課題10-3】

[mainでの処理]

```
  printf("%d\n", kaijo(5));
  printf("%d\n", kaijo(10));
```

[実行結果]

```
  5 * 4 * 3 * 2 * 1 = 120
  10 * 9 * 8 * 7 * 6 * 5 * 4 * 3 * 2 * 1 = 3628800
```

【課題10-4】

再帰呼び出しを利用して、「引数で与えられた配列aの全要素の値の合計を求める」関数sum_array()を作成してください。ただし、引数nは配列aの開始場所（n番目の要素から開始）を表すとします。

[この関数のプロトタイプ宣言]

```
int sum_array(int *a, int n);

   /* 再帰呼び出しのif文は練習10-1と同じ */
   /* 配列aのn番目の値を加算するように作る */
   /* 再帰呼び出しの際に、配列aはそのまま渡し、nは1減らして渡す */
```

【課題10-4】

[mainでの処理]

```
    int a1[6] = {8, 3, 6, 7, 1, 4};
    int a2[3] = {5, 2, 9};
    printf("%d\n", sum_array(a1, 5));
    printf("%d\n", sum_array(a2, 2));
```

[実行結果]

```
  4 + 1 + 7 + 6 + 3 + 8 = 29
  9 + 2 + 5 = 16
```

【課題10-5】

再帰呼び出しを利用して、「引数で与えられた配列strの文字列の長さを求める」関数length()を作成してください。

[この関数のプロトタイプ宣言]

```
int length(char *str, int n);

  /* if文の条件は「strのn番目が終端文字ではない」にする */
  /* 再帰呼び出しするたびに、1を加算するように作る */
  /* 再帰呼び出しの際に、配列strはそのまま渡し、nは1増やして渡す */
  /* 関数内のprintfの出力はしなくてもよい */
```

【課題10-5】

[mainでの処理]

```
  char str1[] = "Hello!";
  char str2[] = "Good Job!";
  printf("%d\n", length(str1, 0));
  printf("%d\n", length(str2, 0));
```

[実行結果]

```
  6
  9
```

まだ余裕のある人は… 【課題10-6】

課題10-4の関数を、double型の配列に対応した関数sum_array2()を作成してください。

[この関数のプロトタイプ宣言]

```
double sum_array2(double *a, int n);

  /* sum_array()の配列aをdouble型に対応させる */
```

[mainでの処理]

```
  double d1[5] = {8.3, 3.4, 6.2, 7.5, 1.1};
  printf("%lf\n", sum_array2(d1, 4));
```

[実行結果]

```
  1.100000 + 7.500000 + 6.200000 + 3.400000 + 8.300000 =
26.500000
```