# プログラミング応用

http://bit.ly/ouyou3d

## 再帰処理（2）

前期 第11週

2018/7/3

# 本日は・・・

## 再帰アルゴリズムと
## 再帰呼び出しについて
## 更に例を使って学びます

- 再帰呼び出しに、更に条件を付けてみる
- 複数の再帰呼び出しからの結果を演算に使う

# 【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のプログラム例

「nは必ず偶数が与えられる」のが前提ならば・・・

```
int sum_even(int n)
{
    if(n > 0) {
        printf("%d + ", n);
        return n + sum_even(n-2);
    } else {
        printf("%d = ", n);
        return n;
    }
}
```

# 課題10-1のプログラム例

## もし、sum_even()のnに奇数が与えられると・・・

[mainでの処理]

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

[実行結果]

```
   7 + 5 + 3 + 1 + -1 = 15
```

ifの条件「n>0」を最初に満たさない整数は -1のため、-1が加算されてしまう。
（nが奇数の場合、nの値は0になることはないため）

# 課題10-1のプログラム例

**では、nが偶数・奇数どちらでも、**
**「1からnまでの偶数の合計を求める」関数を作るには？**

この条件をもったif文を追加する

**1からnまでの整数に対して、偶数だったらその値を加算する**
**（奇数だったらその値の代わりに0を加算する）**

# 課題10-1の改良版

```
 1: #include <stdio.h>
 2: int sum_even2(int n);
 3:
 4: int sum_even2(int n)
 5: {
 6:     if(n > 0) {
 7:         if(n % 2 == 0) {
 8:             printf("%d + ", n);
 9:             return n + sum_even2(n-1);
10:         } else {
11:             return 0 + sum_even2(n-1);
12:         }
13:     } else {
14:         printf("%d = ", n);
15:         return n;
16:     }
17: }
18:
19: int main(void)
20: {
21:     printf("%d\n", sum_even2(7));
22:     printf("%d\n", sum_even2(10));
23:     return 0;
24: }
```

【Point 1】偶数かどうかの条件を追加する

【Point 2】nを加算して再帰呼び出し

【Point 3】0を加算して再帰呼び出し
（つまり奇数は加算しない）

# 【練習11-1】

**「課題10-1の改良版」の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11-1】

再帰呼び出しを利用して、「引数で与えられた配列aの中から引数xと同じ値の要素数を求める」関数count()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 count(int *a, int n, int x);

  /* 再帰呼び出しのif文は課題10-4と同じ */
  /* 配列aのn番目の値がxと等しい場合、1を加算して再帰呼び出しする */
  /* 等しくない場合、0を加算して再帰呼び出しする */
  /* 再帰呼び出しの際に、配列aと整数xはそのまま渡し、nは1減らして渡す */
```

# 【課題11-1】

[mainでの処理]

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

[実行結果]

```
  3
  2
```

# 【課題11-2】

**再帰呼び出しを利用して、「n番目のフィボナッチ数を求める」関数fibo()を作成してください。**

- fibo(0) = 0, fibo(1) = 1とする
- n番目のフィボナッチ数 fibo(n)は、fibo(n) = fibo(n-1) + fibo(n-2)
- つまり、「1つ前のフィボナッチ数」と「2つ前のフィボナッチ数」を加算すると次のフィボナッチ数が求まる

**※詳しく知りたい人はWebで調べてみましょう**

[ この関数のプロトタイプ宣言 ]

```
int fibo(int x);

   /* xの値が0の場合、0を戻す → fibo(0)は0 */
   /* xの値が1の場合、1を戻す → fibo(1)は1 */
   /* 上記以外の場合、x-1の再帰呼び出しとx-2の再帰呼び出しを
        加算した結果を戻す → fibo(n)は fibo(n-1)+fibo(n-2) */
```

# 【課題11-2】

[mainでの処理]

```
  int i;
  for(i=0; i<20; i++) {
      printf("%d ", fibo(i));
  }
  printf("\n");
```

[実行結果]（前にある2つの整数を加算した結果が次の整数になっている）

```
  0 1 1 2 3 5 8 13 21 34 55 89 144 233 377 610 987 1597
2584 4181
```

# 【課題11-3】

再帰呼び出しを利用して、「引数で与えられた文字列strの中から引数chと同じ文字の要素数を求める」関数count_char()を作成してください。

[この関数のプロトタイプ宣言]

```
int count_char(char *str, int n, char ch);

  /* 再帰呼び出しのif文は課題10-5と同じ */
  /* 文字列strのn番目の文字がchと等しい場合、1を加算して再帰呼び出しする */
  /* 等しくない場合、0を加算して再帰呼び出しする */
  /* 再帰呼び出しの際に、配列strと文字chはそのまま渡し、nは1増やして渡す */
```

# 【課題11-3】

[mainでの処理]

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

[実行結果]

```
  2
  3
  4
```

# 【課題11-4】

**2進数の各桁を格納する整数型の配列を考える。**
**（つまり、配列の各要素には0または1のみが入る）**

```
int a3[4] = {1, 1, 0, 1};   //4ビットの2進数値1101(2)を表す
```

**再帰呼び出しを利用して、「引数で与えられた配列aが表している2進数の値から10進数の値を求める」関数todecimal()を作成してください。ただし、引数nは配列aの開始場所、iは2進数の桁を表すとします。**

# 【課題11-4】

[ この関数のプロトタイプ宣言 ]

```
int todecimal(int *a, int n, int i);

  /* 再帰呼び出しに対するifの条件は「nが0より大きい」にする */
  /* 「nが0より大きい」場合、配列aのn番目の値（0または1）に2iを
     かけた結果に、再帰呼び出しの結果を加算した結果を戻す */
/* そうでない場合、配列aのn番目の値（0または1）に2iを
   かけた結果を戻す */
/* 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、
   iは1増やして渡す */
/* べき乗の計算には標準ライブラリ関数powを使う
   （コンパイルには「-lm」を付ける必要がある） */
```

# 【課題11-4】

[mainでの処理]

```
  int a3[4] = {1, 1, 0, 1};
  int a4[8] = {1, 1, 1, 0, 1, 1, 0, 1};
  printf("%d\n", todecimal(a3, 3, 0));
  printf("%d\n", todecimal(a4, 7, 0));
```

[実行結果]

13 ← 1101(2) = 13(10)

237 ← 11101101(2) = 237(10)

まだ余裕のある人は… 【課題11-5】

**再帰呼び出しを利用して、「引数で与えられた整数が素数かどうかを判定する」関数isprime2を作って下さい。（課題9-3で作ったisprimeの再帰処理版）**

[ この関数のプロトタイプ宣言 ]

```
int isprime2(int num, int n);

  /* 再帰呼び出しに対するifの条件は「nがnumと等しくない」にする */
  /* ● 上記のif文の中で、「numがnで割り切れた」場合は0を戻す
       （つまり、numは素数ではなかった） */
  /* ● 「numがnで割り切れない」場合は、n+1に対する再帰呼び出しをする */
  /* 「nがnumと等しくなった」場合は、numを戻す
       （つまり、2～num-1の整数では割りきれなかったため素数である）*/
```

# 【課題11-5】

[mainでの処理]

```
  int i;
  for(i=2; i<=100; i++) {
      printf("%d, ", isprime2(i, 2));
  }
  printf("\n");
```

[実行結果]（100までの素数が出力される）

```
2, 3, 0, 5, 0, 7, 0, 0, 0, 11, 0, 13, 0, 0, 0, 17, 0, 19,
0, 0, 0, 23, 0, 0, 0, 0, 0, 29, 0, 31, 0, 0, 0, 0, 0, 37,
0, 0, 0, 41, 0, 43, 0, 0, 0, 47, 0, 0, 0, 0, 0, 53, 0, 0,
0, 0, 0, 59, 0, 61, 0, 0, 0, 0, 0, 67, 0, 0, 0, 71, 0, 73,
0, 0, 0, 0, 0, 79, 0, 0, 0, 83, 0, 0, 0, 0, 0, 89, 0, 0, 0,
0, 0, 0, 0, 97, 0, 0, 0,
```

# 小テストについて

## 小テストの注意点

- □ 他人の力は借りずに、自分だけでプログラムを作成する。（つまり定期試験と同様）

- □ 小テスト中は、演習室外へのネットワークアクセスは遮断される。

# 小テストについて

## 小テスト中に参照できるもの

□ **教科書, 配付資料**

□ **自分のホームディレクトリ（ホームフォルダ）以下に保存されているファイル**

□ **小テストでは紙媒体のものは参照可能**

□ **上記以外の情報を参照することは不正行為とする**
**例：USBで接続された機器に保存されているファイルの参照**
**ネットワークを介した情報の参照、など**