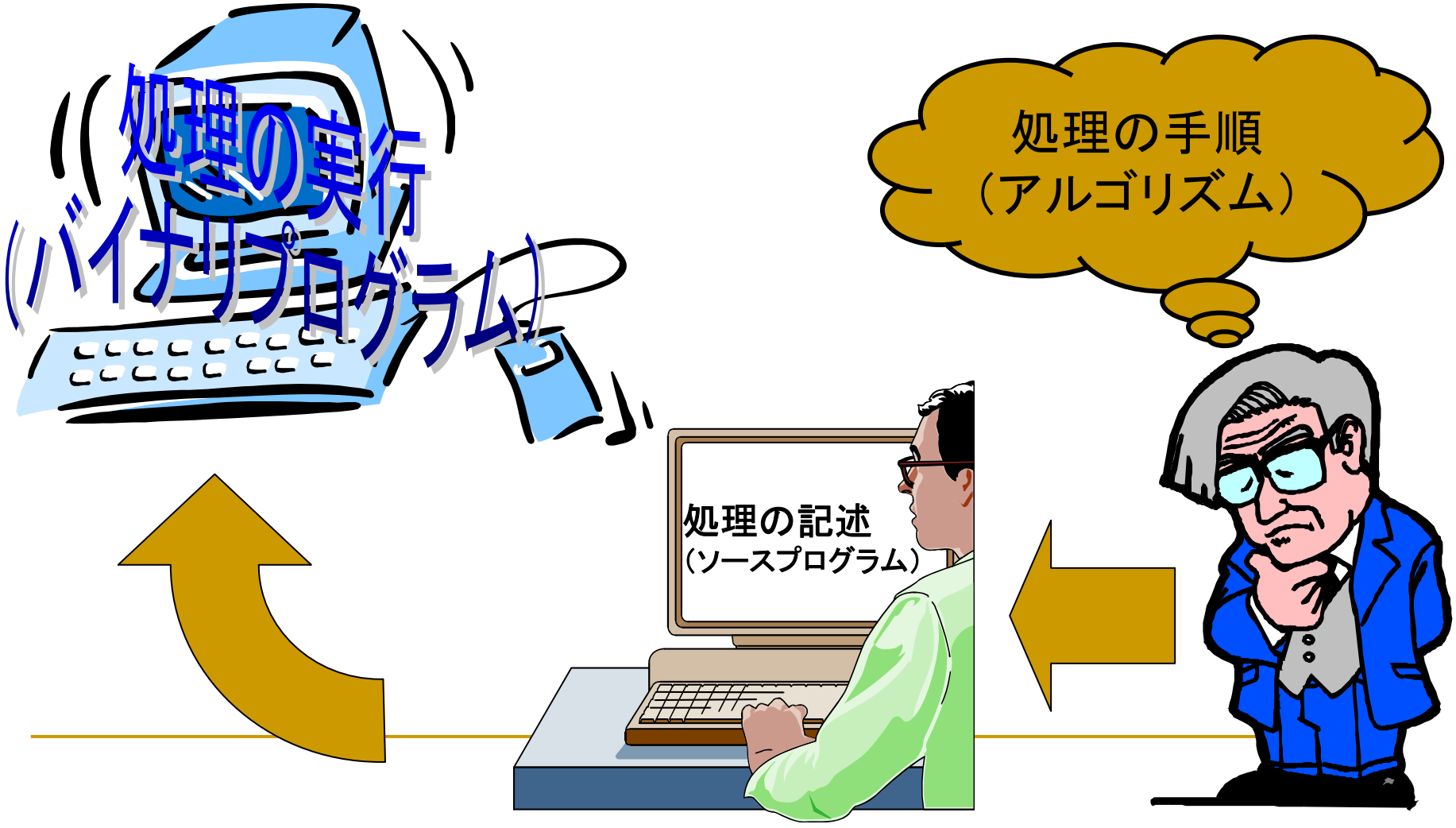
# コンピュータ概論　第9回

繰り返し処理、条件判断

# プログラムとアルゴリズム（おさらい）

- プログラム
  - コンピュータを動作させる命令
  - 動作手順も規定（program＝計画表）
  - 処理装置は命令を逐次実行
- アルゴリズム
  - ある目的を達成するための手続きおよびその手順
  - プログラムの論理的な内容

# プログラムとアルゴリズム



# アルゴリズムの表現方法（おさらい）

- 文字（文章）による表現
  - 自然言語での表現
    - 文章による表現
    - 箇条書きによる表現
  - 人工言語による表現
    - プログラミング言語による記述
    - 仕様記述言語による記述
- 図による表現
  - フローチャート（流れ図）による記述
  - その他の図表（NSチャートなど）による記述

# より高度なアルゴリズム

- 制御構造

  場合分け（条件判断）により異なる動作

  （本来は、順次、分岐、繰り返しの3つをまとめて指す）

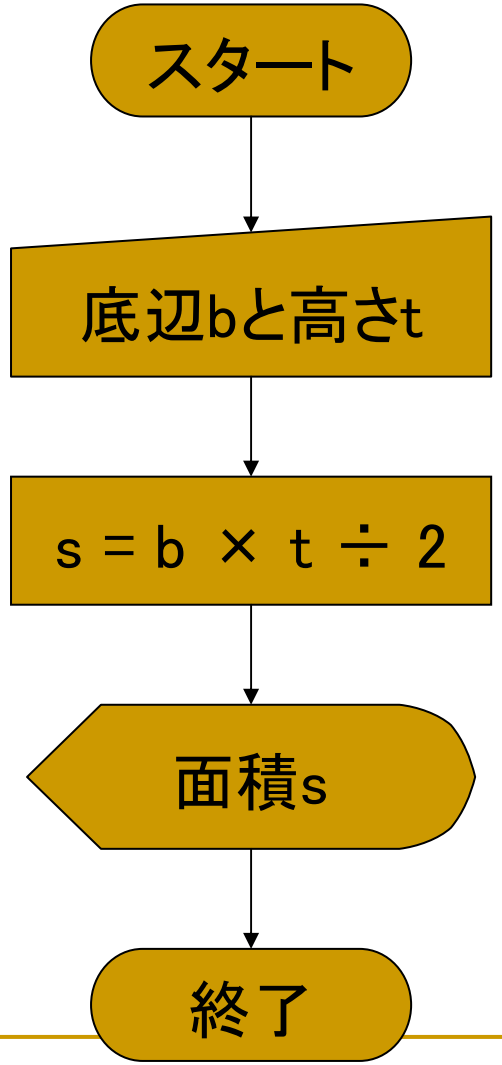
  - 分岐構造
    - 条件分岐（2者択一）
    - ケース分岐（n者択一）：この授業では扱わない
  - 繰返構造
- サブルーチン

  アルゴリズムの一部をまとめて1つの処理と見なす

  - 関数、手続き（12回で説明）

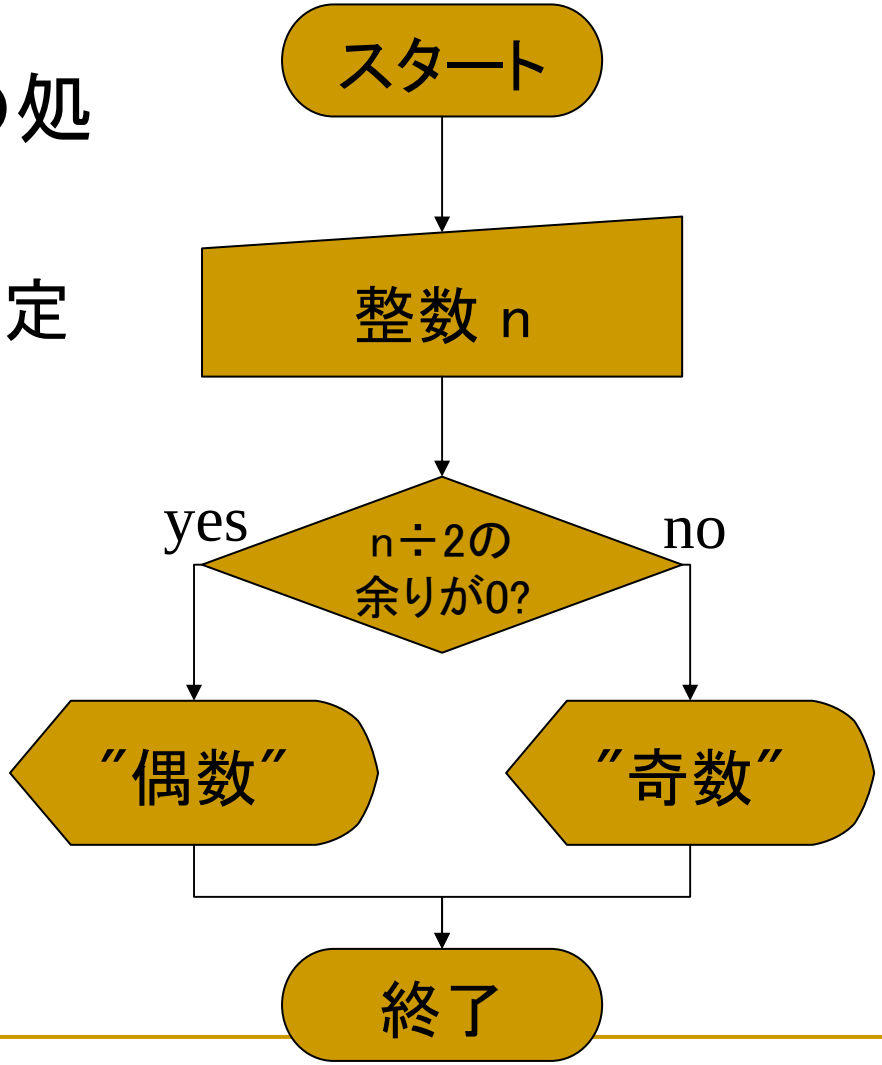
# 順次構造のアルゴリズム

- 初めから終わりまで順次実行される
  - 例）三角形の面積



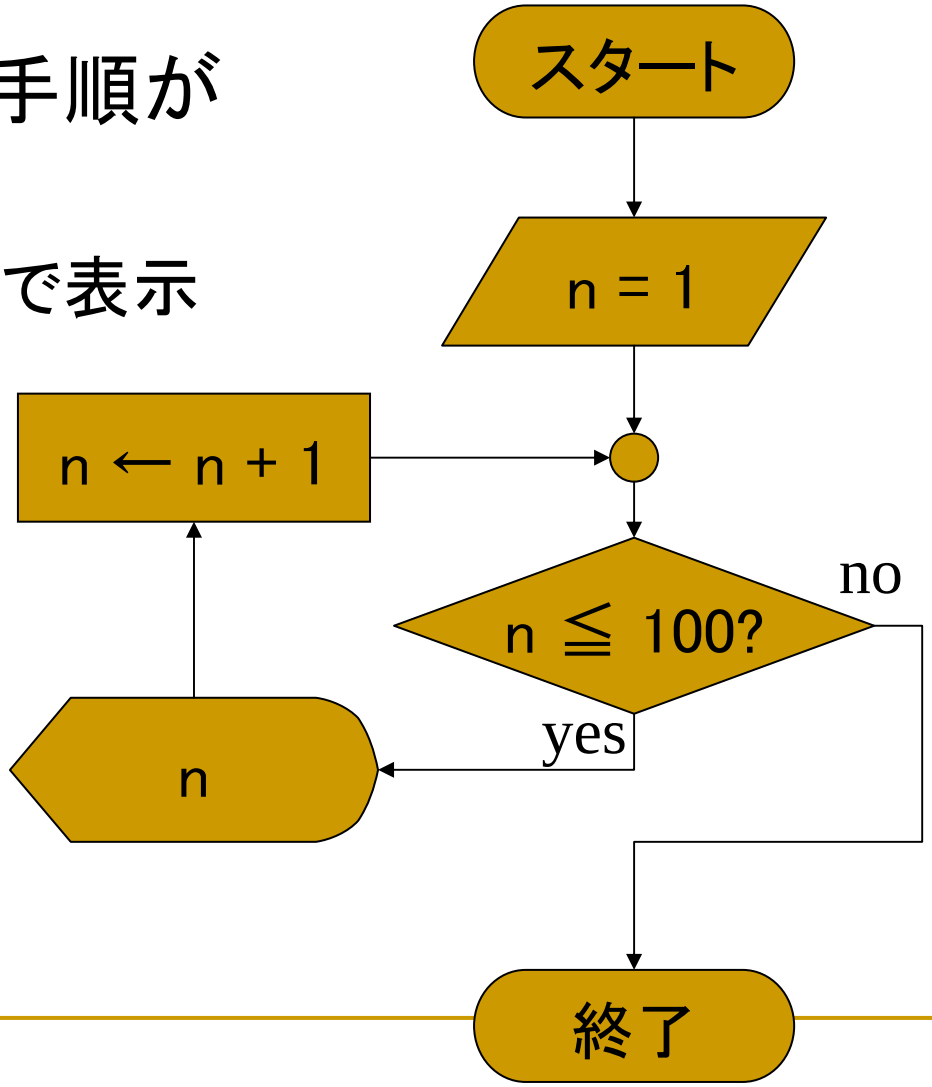
# 分岐構造のアルゴリズム

- 条件によって別の処理が実行される
  - 例）偶数奇数の判定



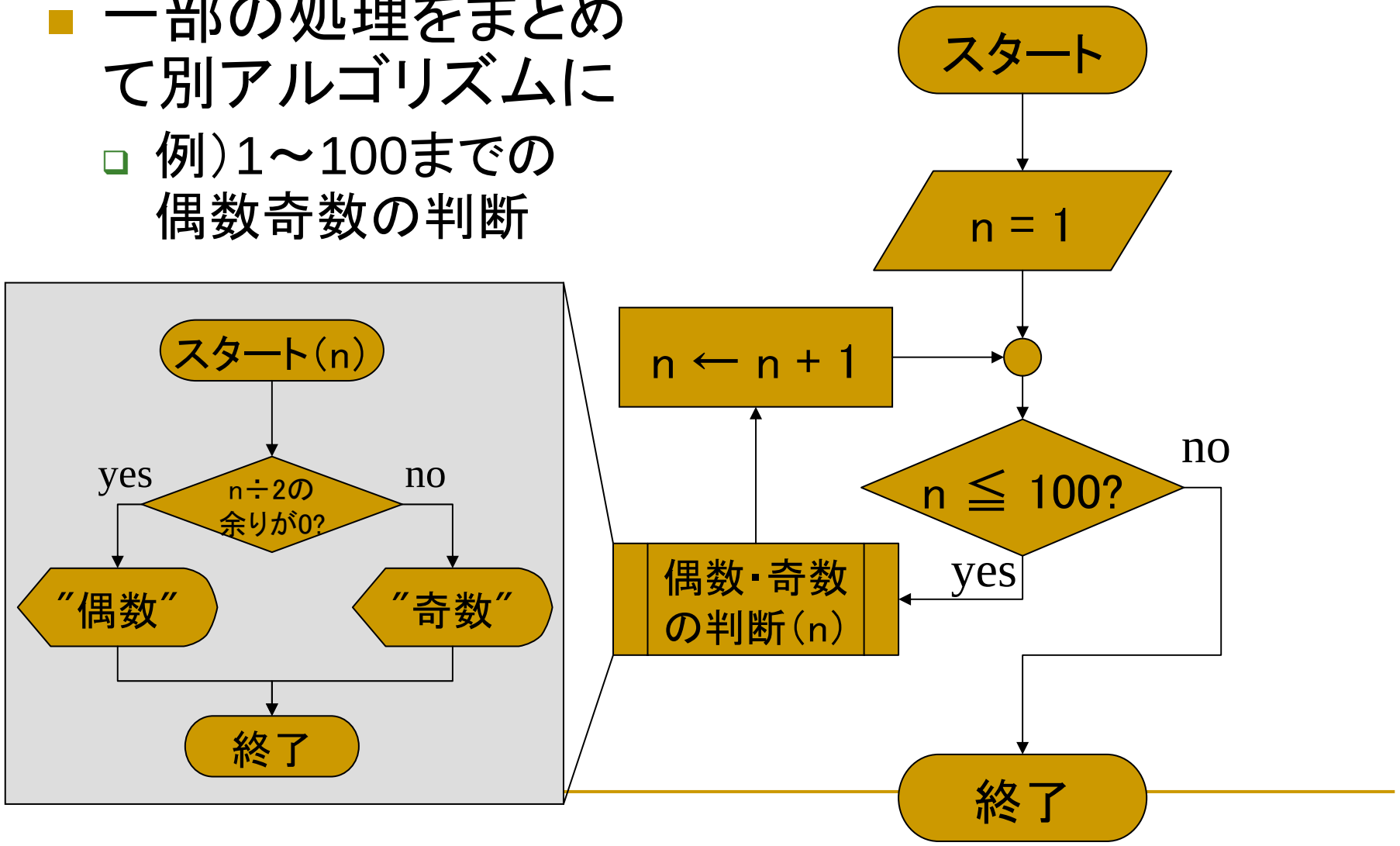
# 繰り返し構造のアルゴリズム

- 条件によって手順が前に戻る
  - 例）1～100まで表示



# サブルーチン構造のアルゴリズム

- 一部の処理をまとめて別アルゴリズムに
  - 例）1～100までの偶数奇数の判断



# Cにおける制御構造の記述

- 順次
  - 順番どおりに命令文を記述
- 分岐
  - パターン1

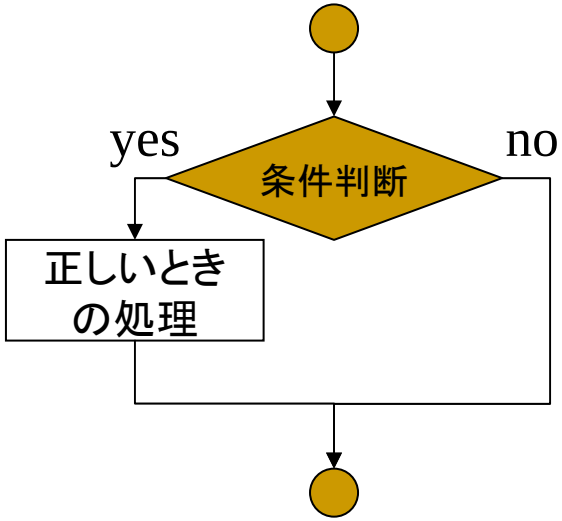
    if （*条件判断*） {
        *条件が正しいときの処理*（複数可能）
    }
  - パターン2

    if （*条件判断*） *条件が正しいときの処理*（１つだけ）
  - パターン3

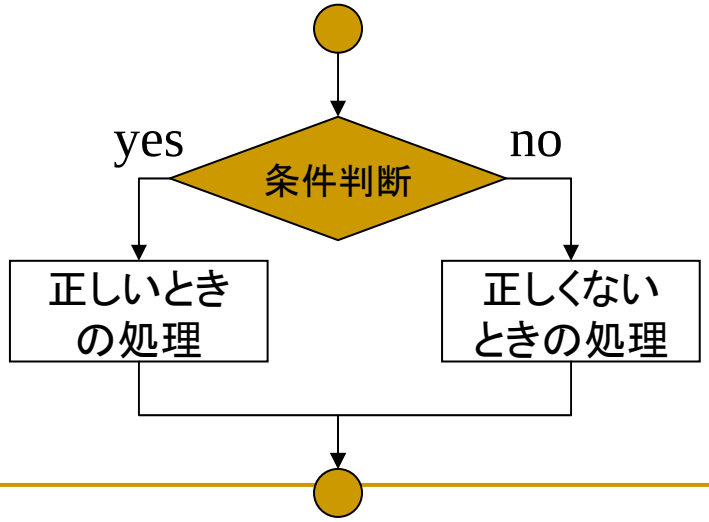
    if （*条件判断*） {
        *条件が正しいときの処理*（複数可能）
    } else {
        *条件が正しくないときの処理*（複数可能）
    }

# フローチャートとの関係

- パターン1
- パターン2




- パターン3



# Cにおける制御構造の記述

- 繰返
  - パターン1(回数が決まっている場合向き)

    for (*初期値*;*条件判断*;*繰返処理*) {

        *条件が正しいときの処理*（複数可能）

    }
  - パターン2(一度も処理されない場合もある)

    while(*条件判断*) {

        *条件が正しいときの処理*（複数可能）

    }
  - パターン3(必ず1度は処理が実行される)

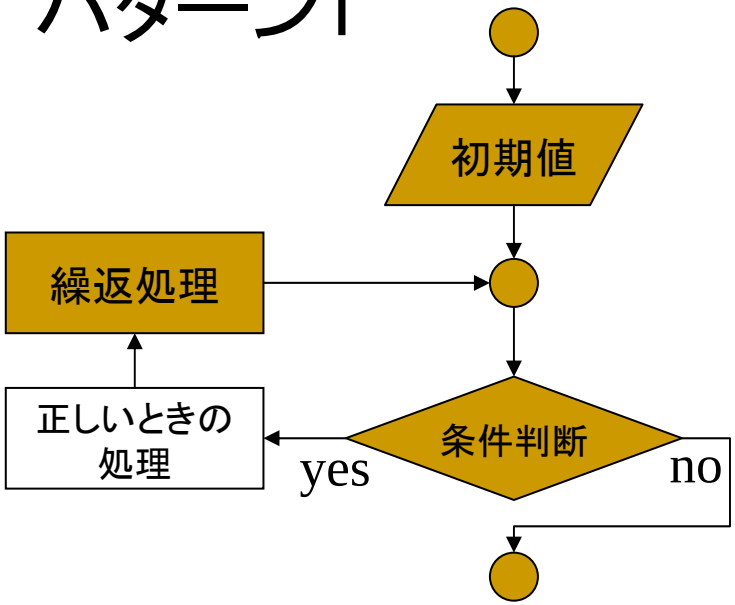
    do {

        *条件が正しいときの処理*（複数可能）

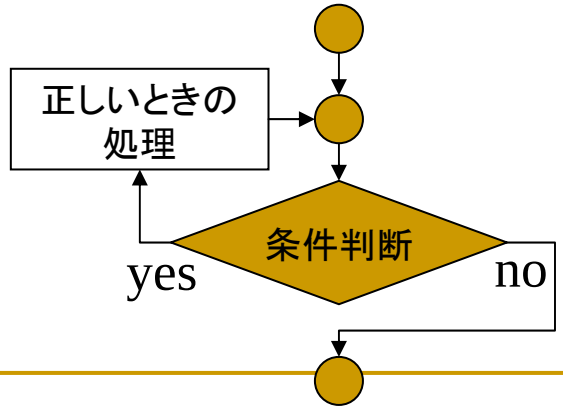
    } while(*条件判断*)

# フローチャートとの関係

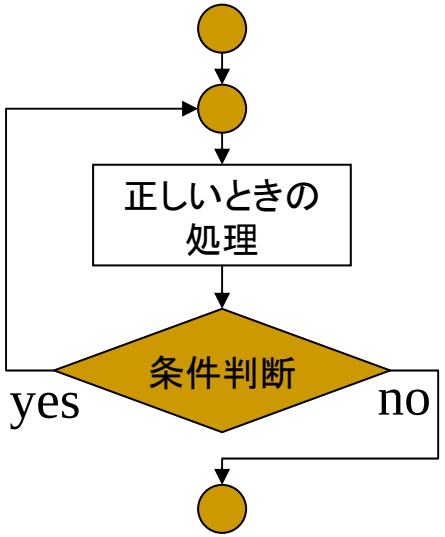
- パターン1




- パターン2




- パターン3



# Cにおける制御構造の記述

- 繰り返しの中断
  - 繰り返しを強制的に中断したい場合

    break;
- 繰り返しの続行
  - 残りの処理を行なわずに次の繰り返しへ進みたい場合

    continue;

- どちらも分岐構造(if文)と共に用いる
  - ある条件が成立した時だけに実行する特別な処理

# 条件判断

- 正しいか正しくないかを判断
- 条件式（数式）により実現
  - 条件演算を含む数式
  - 条件演算

| 演算子 | 意味（左辺は右辺…） |
|---|---|
| > | より大きい |
| < | より小さい |
| >= | 以上 |
| <= | 以下 |
| == | と等しい |
| != | と等しくない |

  - 例）xが偶数かどうか？（%は剰余演算子）
    条件式：　x % 2 == 0

# 特殊な条件式

- 条件演算を含まない数式
  - 値が0のときは正しくない
  - 値が0以外の時は正しい

  - 例）xが奇数かどうか？（%は剰余演算子）

    式： x % 2
    - xが奇数の場合は剰余は1なので「正しい」
    - xが偶数の場合は剰余が0なので「正しくない」
      - x % 2 == 1 と書くのと同じ

# 数学的でない数式

- 等号が成立しない式

  例） n = n + 1

  - n+1の結果をnに代入する
- 項を持たない演算

  例） n += 1

  - n = n + 1の意味
- 等号の無い代入

  例） n++

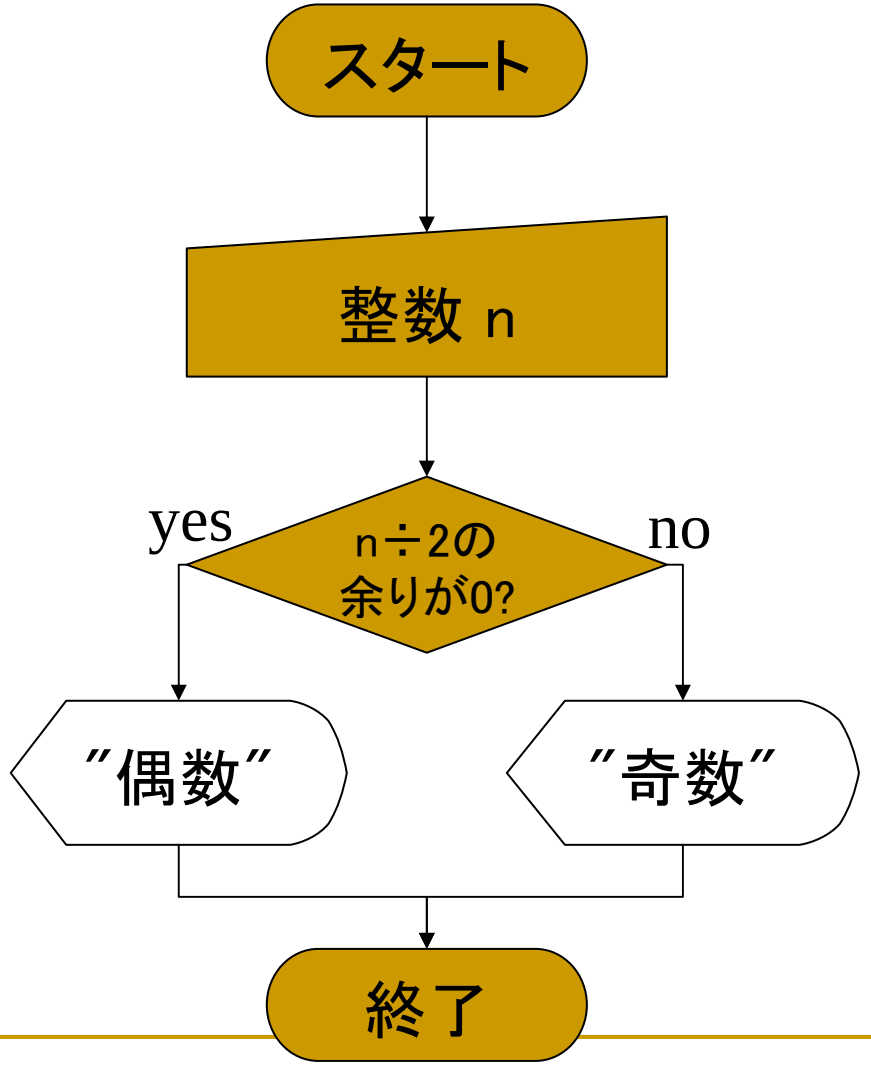
  - nを1増やす（n = n + 1）の意味
  - 1減らすならn--も使える

# 分岐構造のプログラム例

例）偶数奇数の判定

- 分岐パターン3

```
#include<stdio.h>
main(void){
    int n;
    printf("n=");
    scanf("%d",&n);
    if(n % 2 == 0){
        printf("偶数¥n");
    } else {
        printf("奇数¥n");
    }
}
```



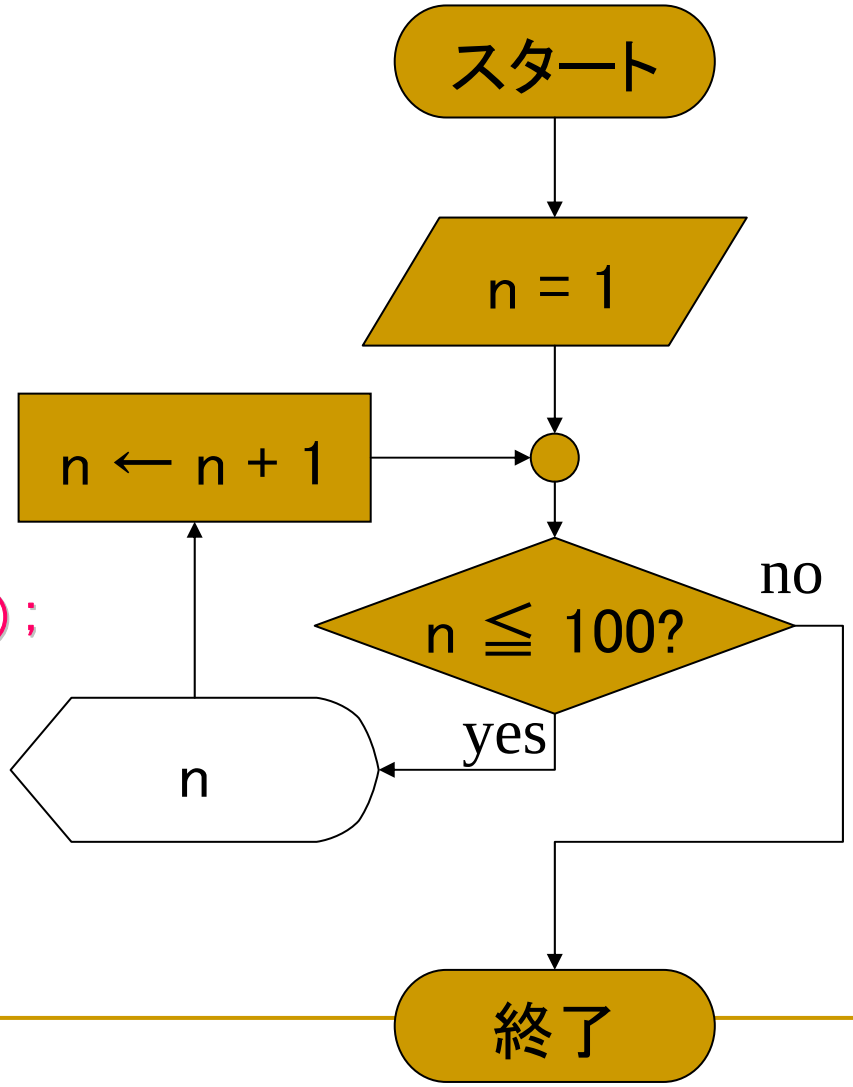
# 繰り返し構造のアルゴリズム

例）1～100の表示

- 繰り返しパターン1

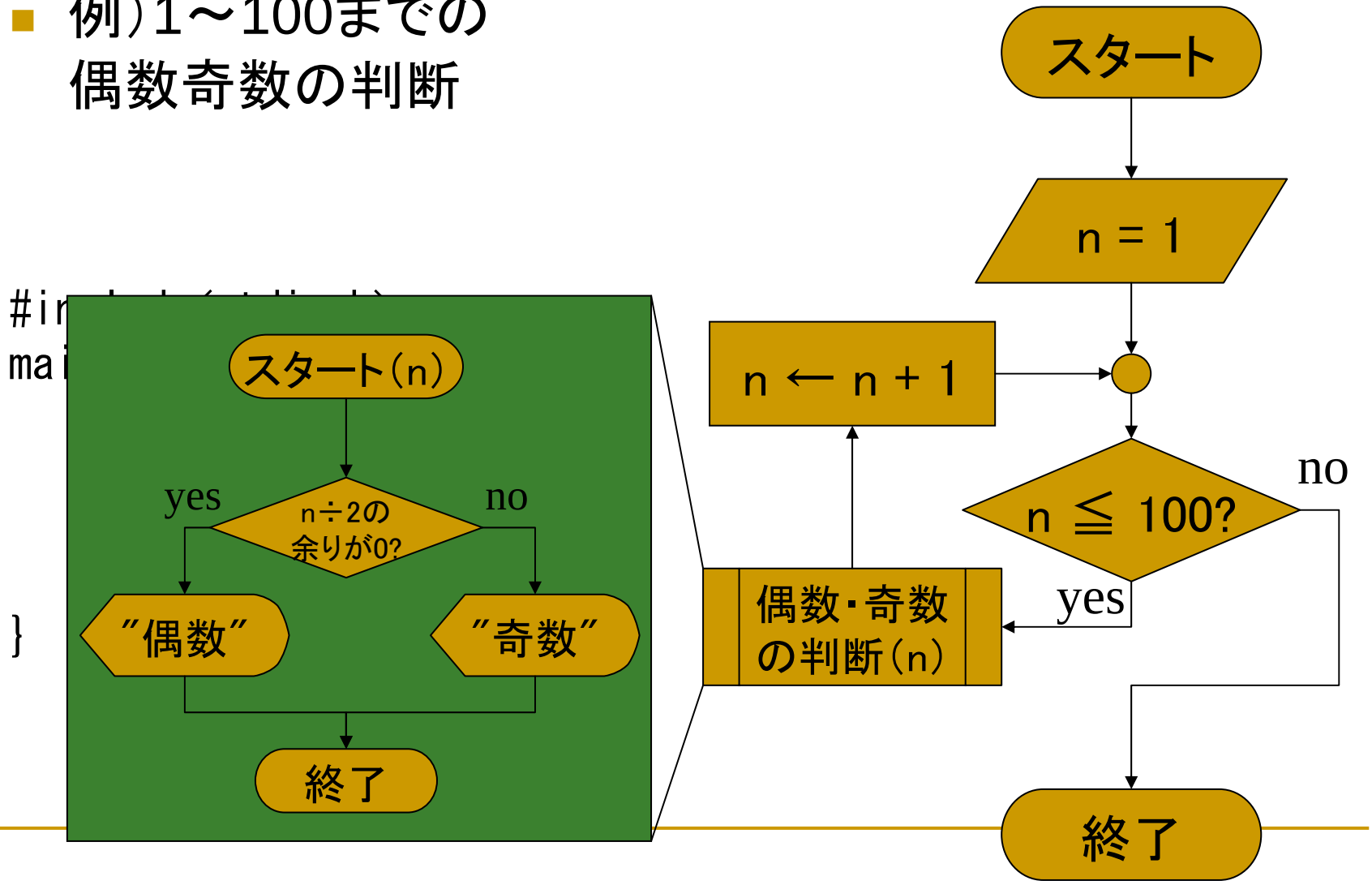
  パターン2や3でも
  実現可能

```
#include<stdio.h>
main(void){
    int n;
    for(n=1; n<=100; n++){
        printf("n=%d¥n",n);
    }
}
```



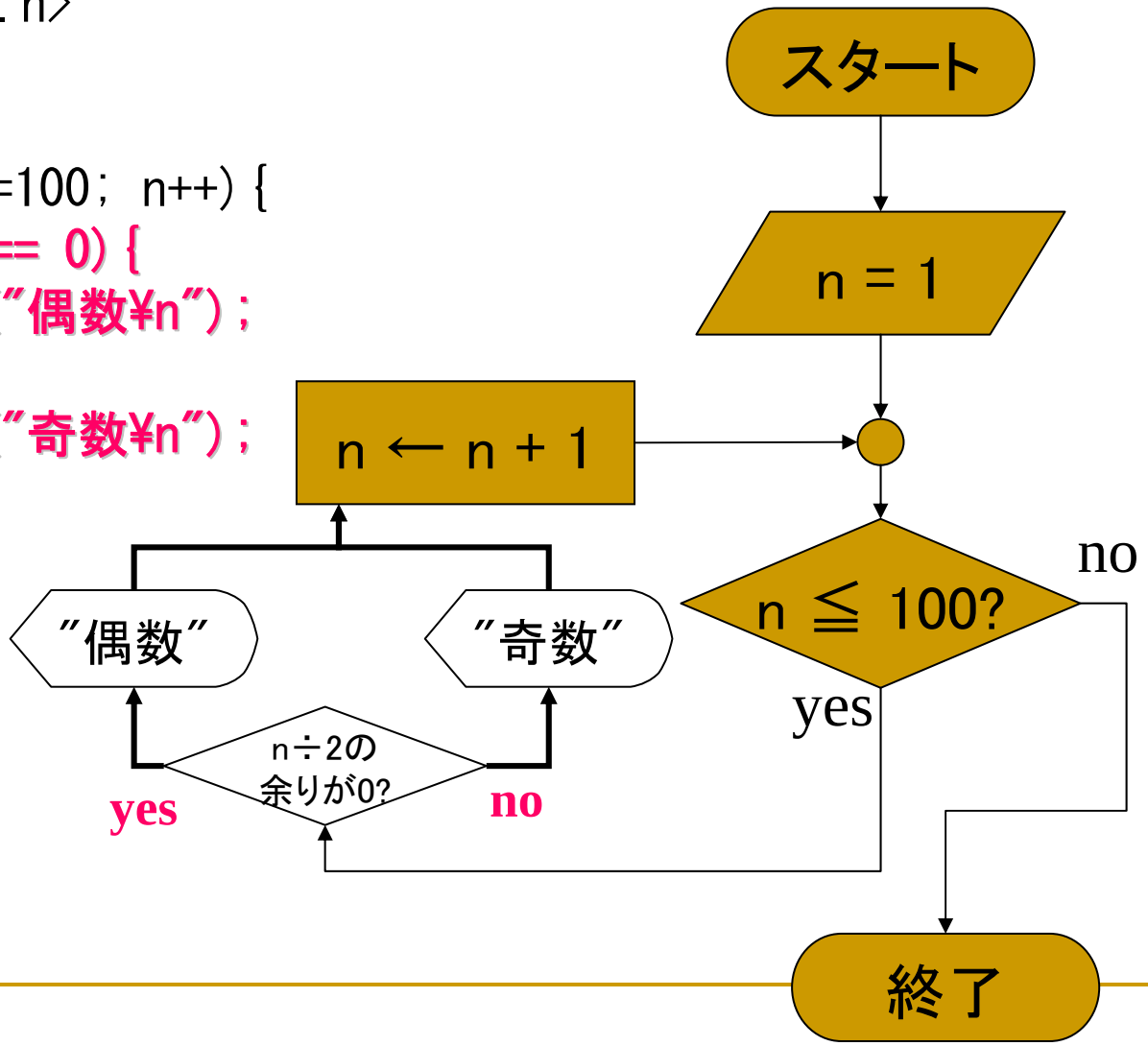
# 制御構造の組み合わせ

- 例）1～100までの
  偶数奇数の判断



# 例）1～100までの偶数奇数の判断

```
#include<stdio.h>
main(void){
    int n;
    for(n=1; n<=100; n++){
      if(n % 2 == 0){
        printf("偶数¥n");
      } else {
        printf("奇数¥n");
      }
    }
}
```



# おさらい

- Cによるアルゴリズムの実現
  - 制御構造
- フローチャートとC言語の記述の関係
  - if文
  - for文
  - while文
  - do while文
  - break文
  - continue文
  - 条件演算子
  - 数学的でない特殊な数式

# 実習　繰り返し構造

- 丸め誤差の確認
  - 0.01を10000回足す
  - floatとdoubleで結果を比較せよ

```
#include<stdio.h>
main(void){
 double s = 0.0, a = 0.01;
   int i;
   for( i=0; i < 10000; i++) s = s + a;
   printf("0.01 * 10000 = %f¥n", s);
}
```

# プログラム課題

- 入力された整数が偶数か奇数かを
  繰り返し判定するプログラムを作成せよ
  - 偶数か奇数かは、剰余演算で判定する
  - 1以下の数は判定対象外
    - 1以下が入力されるまでは、繰り返し入力を
      受け付けて、判定を行なう
    - 1以下が入力されたら、繰り返しを即中断
  - 入力を促すメッセージを表示する
- メールで提出せよ
  - プログラム名はeoro.c