2 制御構造

2.1 if を使ったプログラム

■リスト 2.1

```
/* if 文と等値・関係演算子の使い方 */
#include <stdio.h>

int main()
{
    int num1, num2;

    printf("整数を 2 つ入力してください。\n");
    printf("それらが満たす関係をお答えします。\n");
    scanf("%d%d", &num1, &num2);   /* 整数を 2 つ読み込む */

    if (num1 == num2)   /* 同値演算子 == :「num1 と num2 は等しい」 */
        printf("%d は %d と等しい\n", num1, num2);

    if (num1 != num2)   /* 同値演算子 != :「num1 と num2 は等しくない」 */
        printf("%d は %d と等しくない\n", num1, num2);

    if (num1 < num2)    /* 関係演算子 < : 「num1 は num2 より小さい」 */
        printf("%d は %d より小さい\n", num1, num2);

    if (num1 > num2)    /* 関係演算子 > : 「num1 は num2 より大きい」 */
        printf("%d は %d より大きい\n", num1, num2);

    if (num1 <= num2)   /* 関係演算子 <= :「num1 は num2 以下である」 */
        printf("%d は %d 以下である\n", num1, num2);

    if (num1 >= num2)   /* 関係演算子 >= :「num1 は num2 以上である」 */
        printf("%d は %d 以上である\n", num1, num2);

    return 0;  /* プログラムが正常終了したことを意味する */
}
```

実行結果例 1

```
整数を 2 つ入力してください。
```

それらが満たす関係をお答えします。
10 20
10 **は** 20 **と等しくない**
10 **は** 20 **より小さい**
10 **は** 20 **以下である**

実行結果例 2

整数を 2 **つ入力してください。**
それらが満たす関係をお答えします。
20
10
20 **は** 10 **と等しくない**
20 **は** 10 **より大きい**
20 **は** 10 **以上である**

実行結果例 3

整数を 2 **つ入力してください。**
それらが満たす関係をお答えします。
15 15
15 **は** 15 **と等しい**
15 **は** 15 **以下である**
15 **は** 15 **以上である**

- if 文
  - 書式

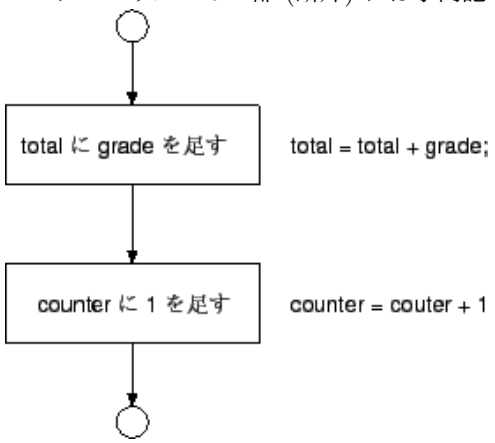
    `if (` *condition* `)`
    body
  - もし条件 (condition) が真ならば，if 文の本体 (body) が実行される
  - 本体 (body) には文を記述する
  - 条件の真偽にかかわらず，if 文の実行を終えると，その次の文に実行が移る
- 字下げ (indent)
  - プログラムを読みやすくするために文を右に寄せること
  - main の本体は全て一段（一定幅）字下げする
  - if の本体はさらに一段字下げする

2.2 構造化プログラミング

- コンピュータを使って問題を解くには，一連の動作を，特定の順序で実行させる必要がある
- アルゴリズム (algorithm) :
  1. 実行すべき (複数の) 動作
  2. 各動作の実行順序

  を記述したもの
- 制御構造 (control structures)

  - 順次構造 (sequence structure): C における既定 (default) の実行順
    プログラムに書かれた順で文が逐次実行される。
  - 選択構造 (selection structure): C には if, if/else, switch の 3 つがある
  - 反復構造 (repetition structure): C には while, do/while, for の 3 つがある
  - すべてのプログラムは，この 3 つの制御構造のみで書ける[*6] -> 構造化プログラミング
- フローチャート (流れ図; flowchart)
  - アルゴリズムの図式表現
  - 専用の記号を流れ線 (flowline) で結んで描く
  - 矩形記号
    * あらゆる動作を表すために使う
  - 楕円記号
    * アルゴリズムの開始 / 終了を表すために使う
      アルゴリズムの一部 (断片) には小円記号を使う



  - 単一入口 / 単一出口 (single-entry/single-exit) の制御構造
    * 一つの制御構造の出口を，次の制御構造の入口に連結 (制御構造スタック) できる
    * プログラムの作成を容易にする
- 疑似コード (pseudo-code)
  - アルゴリズムの開発を助ける人工的で非公式な言語
  - 日常語に類似
  - コンピュータで実行することはできない
  - プログラムを書く前に「じっくり考える」ことを助けるもの
    * C プログラムの各文に対応するように書く
    * 実行文のみで十分

[*6] Böhm (ベーム) と Jacopini (ヤコピーニ) の構造化定理

3 選択構造

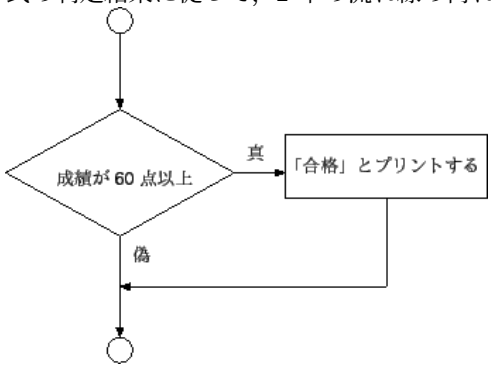
3.1 if 選択構造

- 選択構造:
  - 幾つかの動作の中から一つを選択するために使う
  - 疑似コード

    もしも　学生の成績が `60` **点以上　ならば**
    「合格」と表示する
- 条件が 真 (true) のとき
  -「合格」と表示してから，それに続く文が実行される
  - 偽 (false) ならば，”「合格」と表示する” は無視される
- 疑似コードを C のコードで記述すると

```
if (grade >= 60)
   printf("合格\n");
```

  - 疑似コードは C のコードとよく対応するように書く
  - 字下げ (indent) はプログラムを読みやすくする
    - C では空白文字 (スペース，タブ，改行記号) は無視される
    - if の条件により実行の有無が決まる文 (if の本体) は字下げする
  - C では，if 構造の条件部分にどんな式でも書くことができる
    - 式の値が 0: 偽
    - 式の値が 0 でない: 真
    - 例:
      - `3` は真
      - `i` の値が 1 のとき `i - 1` は偽
- フローチャートの菱形記号 — 選択構造に使える
  - 条件判断が行われることを表す
  - 真または偽となるものを，記号の中に書く
  - 式の判定結果に従って，2 本の流れ線の何れかに進む



- if 構造は単一入口/単一出口構造