3.2 if/else 選択構造

- if — 条件が真のときだけ，指定した動作が実行される

```
if ( 条件式 )
    文
```

- if/else — 真と偽の両方の条件に対して，実行される動作を各々指定する
- if/else の疑似コード例:

```
学生の成績が 60 点以上なら
    「合格」と表示（プリント）する
それ以外なら
    「落第」と表示する
```

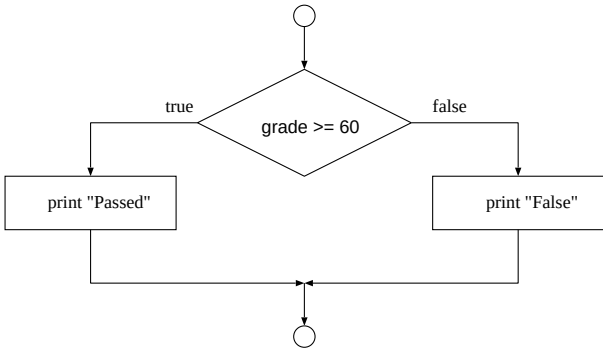
  - 字下げ (indent) に注意
- 擬似コードに対応する C のコード:

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

- if/else の書式

```
if ( 条件式 )
   文 1
else
   文 2
```

- if/else のフローチャート



- 参考: 三項条件演算子 — 3 つのオペランド (被演算子) をとる唯一の演算子
  - 使い方 — 条件 ? 真のときの値 : 偽のときの値
  - 前の if/else のコードと同じ動作をするコード

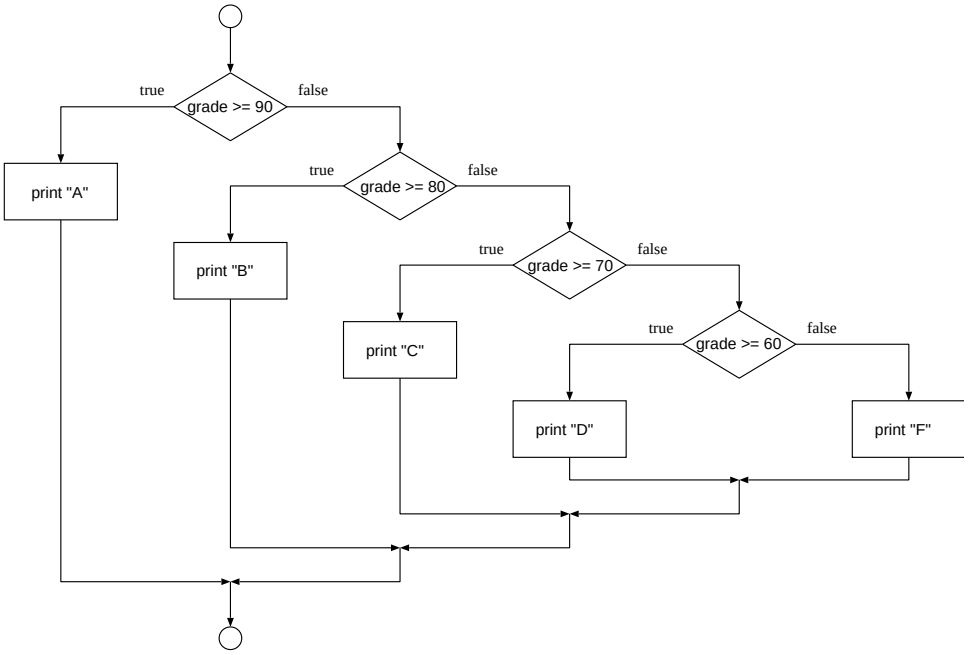
```
printf("%s\n", grade >= 60 ? "合格" : "落第");
```

3.3 if/else の入れ子構造を用いた多重分岐

- if/else 構造の中に if/else 構造を置く (入れ子; nest) ことで，多重の場合分けを実現できる

- 疑似コードとフローチャート

学生の成績が 90 **点以上なら**
　　A **をプリントする**
それ以外で
　　学生の成績が 80 **点以上なら**
　　　　B **をプリントする**
　　それ以外で
　　　　学生の成績が 70 **点以上なら**
　　　　　　C **をプリントする**
　　　　それ以外で
　　　　　　学生の成績が 60 **点以上なら**
　　　　　　　　D **をプリントする**
　　　　　　それ以外なら
　　　　　　　　F **をプリントする**



条件が一つ真になると，残りは実行されない

- 擬似コードに直接対応する C のコード (非推奨)

```
if (grade >= 90)
   printf("A\n");
else
   if (grade >= 80)
      printf("B\n");
   else
      if (grade >= 70)
         printf("C\n");
```

```
        else
            if (grade >= 60)
                printf("D\n");
            else
                printf("F\n");
```

- 字下げ量が多くなると読みにくいので，次のように書く

```
if (grade >= 90)
    printf("A\n");
else if (grade >= 80)
    printf("B\n");
else if (grade >= 70)
    printf("C\n");
else if (grade >= 60)
    printf("D\n");
else
    printf("F\n");
```

3.4 文とエラーの種類

- 制御構造の本体 (body) を複数の文にするには { } (brace) で囲む
  - 例:

```
if (grade >= 60)
    printf("合格\n");
else {
    printf("落第\n");
    printf("この科目を再履修してください\n");
}
```

  - { } が無いと，

```
printf("この科目を再履修してください\n");
```

    は無条件に実行される (非致命的な論理エラー)
- 文 (statement)
  - 文の種類
    - 単文 `---;`
    - 複文 (= ブロック) `{ ---; ---; ---; }`
    - 空文 `;`
  - 文法上は三つ全てが文 => 単文を書ける場所には複文や空文も書ける
  - 文法上，制御構造の本体 (body) は一つの文なので，複数の文を置くには複文にする
- 構文エラーと論理エラー
  - 原因に基づくエラーの二分類

  - 構文 (syntax) エラー: 文法上の誤り
    - コンパイラが検出する (コンパイルエラー)
  - 論理 (logic) エラー: 文法的には正しい
    - 実行時に影響を及ぼす
    - 非致命的 (non-fatal): プログラムは動くが，結果は正しくない
    - 致命的 (fatal): 異常終了する (実行時エラー)

問い：次のコードをコンパイルするとどうなるか。また実行するとどう動くか。

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

3.5 switch 多重選択構造

- switch/case: 変数や式の持つ値に応じて異なる処理を行いたいときに便利
- 書式
  - 一連の ”case” ラベルと，オプションの ”default” ラベルから成り，“変数” 値に対応するラベルの箇所から文の実行が始まる

```
switch ( 変数 ) {
   case 値 1:
      (複数の) 文
      break;
   case 値 2:
      (複数の) 文
      break;
        ⋮
   default:
      (複数の) 文
      break;
}
```

  - switch 構造から抜け出すために break; を置く
    文法的には break; は必要ないが，プログラムの構造を明確にして論理エラーを防止するために，原則として，各 case の文の最後に break; を置くべき

■リスト 3.1 switch/case **を用いた成績変換**

```
/* switch を使った成績変換 */
```

```
#include <stdio.h>

int main()
{
     int grade;

     printf("Enter grade ");
     scanf("%d", &grade);

     switch (grade) {
     case 10:
          printf("A\n");
          break;
     case 9: case 8:
          printf("B\n");
          break;
     case 7: case 6:
          printf("C\n");
          break;
     case 5:
          printf("D\n");
          break;
     case 4: case 3: case 2: case 1: case 0:
          printf("F\n");
          printf("You must take this course again.\n");
          break;
     default:
          printf("Wrong grade value.\n");
          break;
     }

     return 0;
}
```

実行結果 1

```
Enter grade 10
A
```

実行結果 2

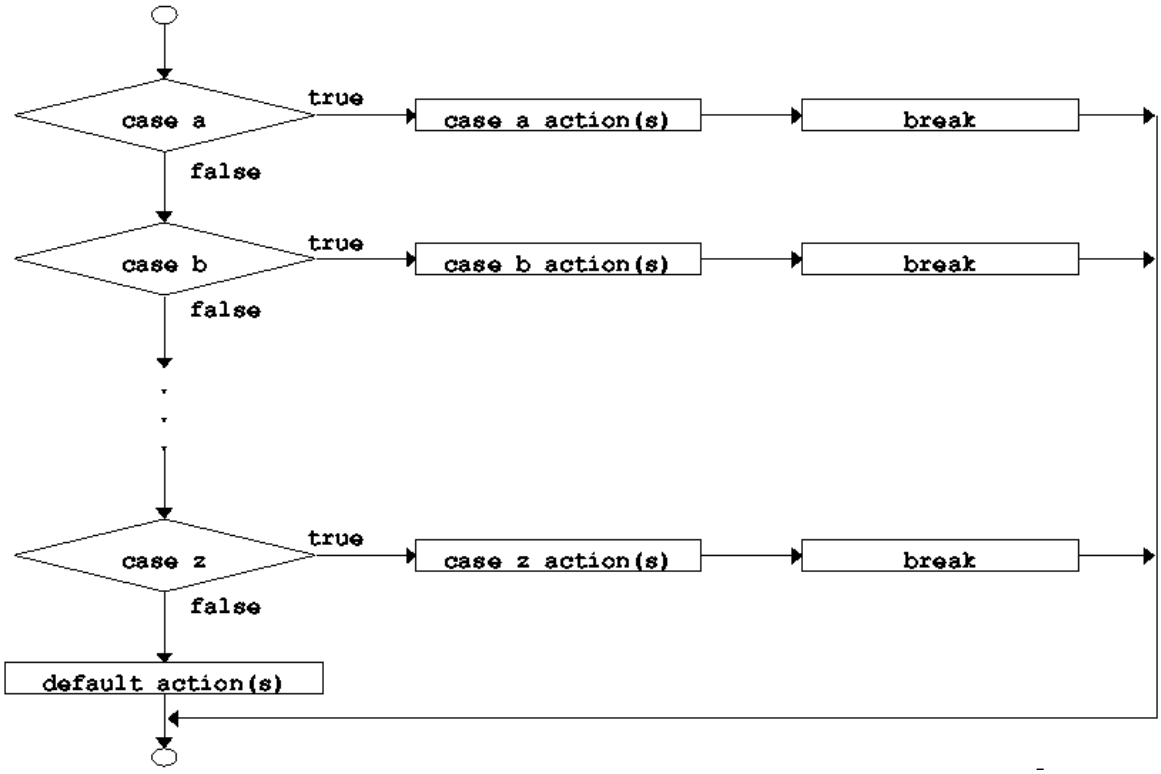
```
Enter grade 4
```

```
F
You must take this course again.
```

実行結果 3

```
Enter grade 11
Wrong grade value.
```

- switch/case 構造のフローチャート



-

- case ラベルに書けるのは整定数式のみ
  - 整数定数 (1 や 2 など)
  - 文字定数 ('a' や 'A' など)
  - 変数は許されない
  - 浮動小数点数 (1.0 など) は許されない