3.6 等値演算子 (==) と代入演算子 (=) の混同に注意

- 通常は構文エラーにならない
- どの制御構造の条件部も式だから（値が決まればいい）
- 条件部の非 0 は真を，0 は偽を意味する
- == を使う例:

```
if (payCode == 4)
   printf("ボーナスを支給する\n");
```

  - payCode が 4 ならばボーナスを支給する
- 誤って == を = と書いた場合:

```
if (payCode = 4)
   printf("ボーナスを支給する\n");
```

  - payCode = 4 は代入 演算子 を含む 式
    - payCode に 4 を代入するとともに，式 payCode = 4 の値は 4 になる。
  - 4 は非 0 なので条件は常に真。
  - よって，payCode に関係なくボーナスを支給する。
  - これは論理エラーであって，構文エラーではない (コンパイラはエラーとして検出しない)

3.7 論理演算子

- 例（擬似コード）:

```
性別が 1（女性）で，かつ年齢が 65 才以上であれば
       Senior femail と表示する
```

- 例（C コード）:

```
if (gender == 1 && age >= 65)
       printf("Senior femail\n");
```

- 条件式の値
  -「条件が偽」のとき式の値は 0
  -「条件が真」のとき式の値は 0 以外
- && (「かつ」，論理積)
  - 二項演算子；両方の条件が真のときのみ，全体の条件が真
  - 真理値表

式 1	式 2	式 1 && 式 2
0 以外	0 以外	1
0 以外	0	0
0	0 以外	0
0	0	0

- || (「または」，論理和)
  - 二項演算子；どちらか (または両方) の条件が真のとき，全体の条件が真

– 真理値表

式 1	式 2	式 1 || 式 2
0 以外	0 以外	1
0 以外	0	1
0	0 以外	1
0	0	0

- ! (「でない」, 否定)
  – 単項演算子: 被演算子は一つ
  – 条件の真偽を反転させる
  – 真理値表

式	! 式
0 以外	0
0	1

  – 例

```
if (!(num1 == num2))
    printf("%d と %d は異なる\n", num1, num2);
```

  と

```
if (num1 != num2)
    printf("%d と %d は異なる\n", num1, num2);
```

  は同じ動作をする

- これまでに登場した演算子の優先順位（上ほど高い）と結合性

演算子	結合性	グループ
()	左から右	カッコ
+ - !	右から左	単項
* / %	左から右	乗法
+ -	左から右	加法
< <= > >=	左から右	関係
== !=	左から右	等値
&&	左から右	論理積
||	左から右	論理和
?:	右から左	条件
=	右から左	代入

- 次のコードは grade の値が何であっても B と表示する (論理エラー)

```
if (grade == 8 || 9)
        printf("B\n");
```

  正しくは

```
if (grade == 8 || grade == 9)
        printf("B\n");
```

4 反復構造

- 反復構造 (繰り返し; loop)
  - 条件が真の間，特定の動作を繰り返す
  - 例:

    買い物リストに品物があるあいだ

    次の品物を買って，（その品物を）リストから消去する

4.1 while 反復構造

- if 選択構造 — 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** `）`

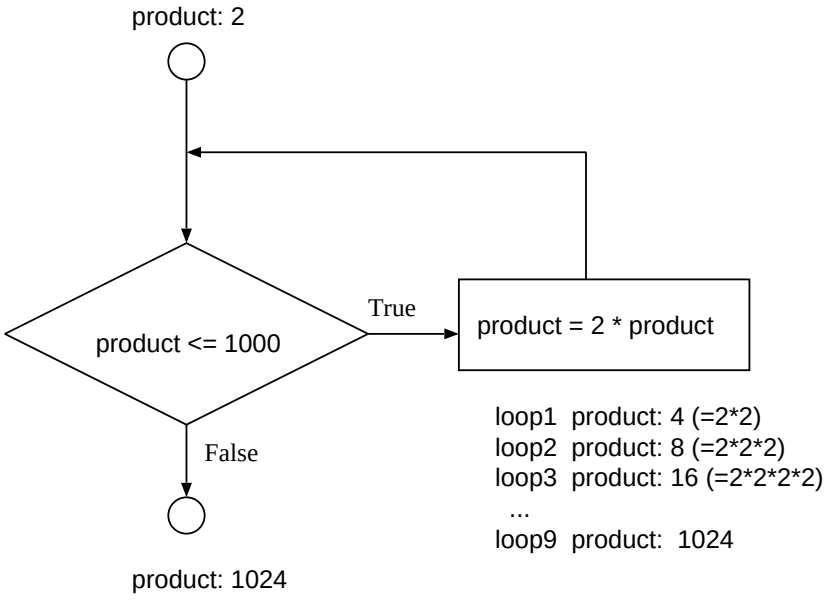
  文
- while 反復構造 — 条件が真の間，指定した動作が繰り返し実行される

  `while（` **条件式** `）`

  文

  条件式が真であり続けると，文の繰り返しが終わらない — 無限ループ; Ctrl-C で強制終了
- 例: 1000 を越える最初の 2 の累乗（1000 以上の 2 の累乗の中の最小値）を求める

```
product = 2;
while (product <= 1000)
   product = 2 * product;
```



4.2 カウンタ制御による while 反復処理

- カウンタ (繰り返し回数を格納する変数) が特定の値になるまで繰り返しを行う
- 予め繰り返し回数を特定できる問題に利用可能
- 例:

  10 人の学生がいるクラスで試験が行われ，その成績 (0 から 100 までの整数) が手元にある。この試験のクラス平均を求めよ
- 疑似コード:

  合計を 0 **にする**
  成績カウンタを 1 **にする**

  成績カウンタが 10 **以下であるあいだ**
  　次の成績を入力する
  　その成績を合計に加える
  　成績カウンタに 1 **を加える**

  クラス平均に，合計を 10 **で割った値を代入する**
  クラス平均をプリントする

■リスト 4.1 **クラス平均プログラム**

```
/* カウンタ制御反復を使ったクラス平均プログラム */
#include <stdio.h>

int main()
{
  int counter, grade, total, average;

  /* 初期化フェーズ */
  total = 0;
  counter = 1;

  /* 処理フェーズ */
  while (counter <= 10) {
    printf("Enter grade: ");
    scanf("%d", &grade);
    total = total + grade;
    counter = counter + 1;
  }
```

```
    /* 終了フェーズ */
    average = total / 10;
    printf("Class average is %d\n", average);

    return 0;
}
```

実行結果例

```
Enter grade: 98
Enter grade: 75
Enter grade: 60
Enter grade: 82
Enter grade: 95
Enter grade: 68
Enter grade: 72
Enter grade: 55
Enter grade: 89
Enter grade: 95
Class average is 78
```

- カウンタ制御による反復に必要なもの
  1. カウンタ（制御変数）の変数宣言
  2. ループ前のカウンタの初期値設定
  3. ループ毎に行うカウンタの増加
  4. カウンタが最終値か否か（反復条件）の確認