

プログラミング第一同演習

慶應義塾大学 理工学部 情報工学科

講義担当：河野 健二

演習担当：杉浦 裕太

- これまでに習った制御構造
 - if 文
 - for 文

- C には他にも多くの制御構造がある.
本日のお題は以下の制御構造
 - switch 文
 - while 文
 - do-while 文
 - break
 - continue

- 変数の値で場合分けをするときに便利
 - if 文でやってもよい
 - switch 文の方が見やすくなる時がある

- 例: char 型の変数 c の値で場合分け

- char c;

- ...

```
switch (c) {    // 変数 c の値で場合分け
    case 'a':
        // 変数 c の値が 'a' に等しい時に実行
        break;
    case 'b':
        // 変数 c の値が 'b' に等しい時に実行
        break;
}
```

switch 文:一般形

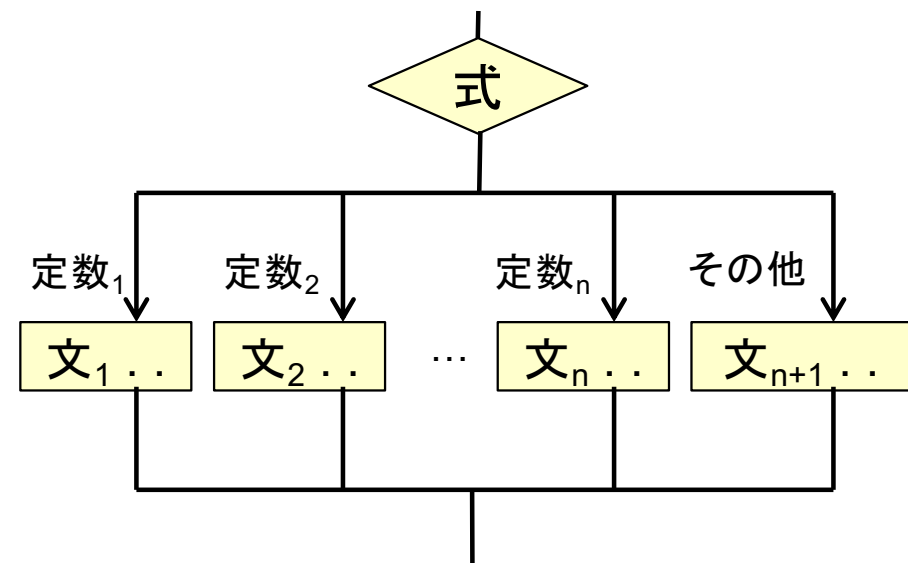
```
switch (式) {
case 定数1 :
    文1 . . .
    break;
case 定数2 :
    文2 . . .
    break;
. . .
case 定数n :
    文n . . .
    break;
default :
    文n+1 . . .
    break;
}
```

整数型 (int) または文字型 (char)

定数のみ

どの定数にも一致しなかったときに実行する.
default の部分は無くてもよい

1. 式の値を計算する
式の結果は char 型を含む int などの整数の型
2. 計算した結果の値を定数₁から順に比べる
3. 値が一致したところの文を実行する
4. 計算した結果がどの定数にも一致しなかったら default のところを実行する



例題: 摂氏・華氏の変換

- 摂氏あるいは華氏で表される温度を読み込み, 摂氏であれば華氏に, 華氏であれば摂氏に変換し, 変換前後の値を小数第1位まで表示しなさい

温度はその数値と単位記号 (摂氏は 'C', 華氏は 'F') で与えられるものとする

なお読み込まれた単位が摂氏でも華氏でもなければ "wrong unit symbol" と表示しなさい

$$c = \frac{5}{9}(f - 32) \quad f = \frac{9}{5}c + 32$$

摂氏・華氏の変換: 入力まで

```
#include <stdio.h>
#include <stdlib.h>
```

```
int main()
{
```

```
    float deg, fahr, cels;
    char unit;
```

- ・ 入力温度(deg), 華氏(fahr), 摂氏(cels) は float 型
- ・ 単位('C' or 'F') (unit) は char 型

```
// “%f %c” と指定することで同時にふたつの入力が可能
//     例: 25.4C と入力すると deg に 25.4 が入り, unit に 'C' が入る
//     33.3F と入力すると deg に 33.3 が入り, unit に 'F' が入る
if (scanf("%f %c", &deg, &unit) != 2) {
    printf("incorrect input\n");
}



- ・ 温度(deg)は float なので、変換指定は %f
- ・ 単位(unit)は char なので、変換指定は %c

```

摂氏・華氏の変換: 単位の判断と計算

```
    . . .
    switch (unit) {
    case 'C':
        fahr = 1.8 * deg + 32.0;
        printf("%4.1f C = %4.1f F\n", deg, fahr);
        break;

    case 'F':
        cels = (deg - 32.0) / 1.8;
        printf("%4.1f F = %4.1f C\n", deg, cels);
        break;

    default:
        printf("wrong unit symbol\n");
        break;

    }

    return 0;
}
```

摂氏・華氏の変換: if 文でやると？

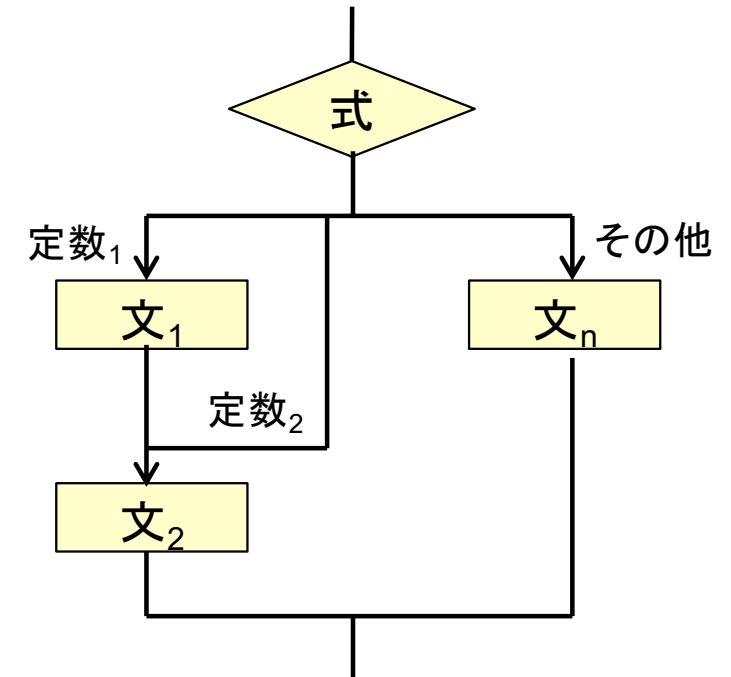
■ if 文でも書ける

```
...  
if (unit == 'C') {  
    fahr = 1.8 * deg + 32.0;  
    printf("%4.1f C = %4.1f F\n", deg, fahr);  
} else if (unit == 'F') {  
    cels = (deg - 32.0) / 1.8;  
    printf("%4.1f F = %4.1f C\n", deg, cels);  
} else {  
    printf("wrong unit symbol\n");  
}  
  
return 0;  
}
```

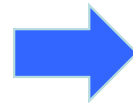

switch 文の補足: break 文の無い形

```
switch (式) {
case 定数1 :
    文1 . . .
case 定数2 :
    文2 . . .
    break;
default:
    文n . . .
    break;
}
```

← “break;” がない.
式の値が定数₁ に一致したときは、
文₁ を実行した後、
文₂ も実行する



- break がないと...
次の case 文をそのまま実行する
(わざとやる場合もある)

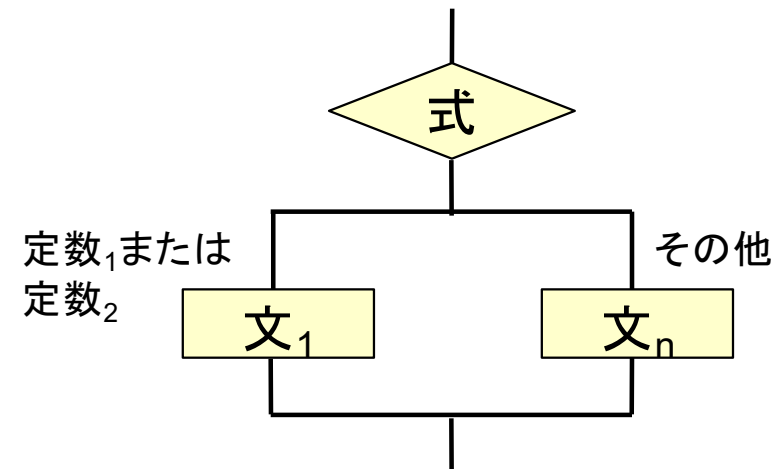


```
switch (式) {
case 定数1 :
    文1 . . .
    // pass through
case 定数2 :
    文2 . . .
    break;
}
```

意図的であることを示すとよい

switch 文: 複数の case ラベルの形

```
switch (式) {  
case 定数1 :   文1に2つの case ラベルが  
case 定数2 :   文1についている.  
    文1 . . .  
    break;     式の値が定数1または定数2に  
               一致するとき文1を実行する  
    . . .  
default :  
    文n . . .  
    break;  
}
```



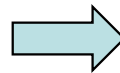
- ある条件が満たされるまで処理を繰り返す
 - 繰り返しのことをループ (loop) という

- 例題:
 - 100 より大きい最小の平方数を求めたい
 - ◆ 次のように考える
 - 変数 i を 1 から順に大きくしていき,
 $i * i$ が 100 を超えたところで, $i * i$ の値を表示する
 - ◆ 言い換えると,
 - $i * i$ が 100 以下の間, i の値をひとつずつ増やす
 - $i * i$ が 100 を超えたら, `printf("%d¥n", i * i)` をする

Quiz: for 文の復習

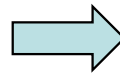
■ 何が表示されますか？

```
int i;  
for (i = 0; i < 3; i++) {  
    printf("i = %d\n", i);  
}
```



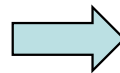
```
i = 0  
i = 1  
i = 2
```

```
int i;  
for (i = 1; i <= 3; i++) {  
    printf("i = %d\n", i);  
}
```



```
i = 1  
i = 2  
i = 3
```

```
int i;  
for (i = 0; i <= 4; i += 2) {  
    printf("i = %d\n", i);  
}
```

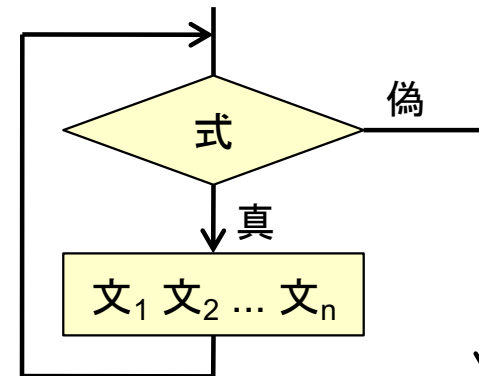


```
i = 0  
i = 2  
i = 4
```

while 文による繰り返し

```
while (条件) {  
    文1  
    文2  
    . . .  
    文n  
}
```

ループ本体



- 条件が真である間, 文₁から文_nを実行する
- より正確には・・・
 1. 条件を評価する
 2. 条件が真なら文₁から文_nを順に実行し, 1. に戻る
 3. 条件が偽なら先へ進む (文₁ から文_n は実行しない)

さきほどの例題

- 100 より大きい最小の平方数を求めたい
 - $i * i$ が 100 以下の間, i の値をひとつずつ増やす
 - $i * i$ が 100 を超えたら, `printf("%d¥n", i * i)` をする

```
#include <stdio.h>

int main()
{
    int i = 1;        // 変数を宣言するのと同時に値を初期化

    while (i * i <= 100) {
        i++;
    }

    printf("%d¥n", i * i);

    return 0;
}
```

例題: 平均点

- 複数個のテストの点を読み込み, それらの合計点と平均点を求め, 読み込んだデータの個数とともに表示しなさい
 - 実行イメージ

```
input scores
78 83 89 100 65 87 <
59 80 67 <
^d <
number of items = 9
total = 708
average = 78.67
```

Enter を入力しても入力終了にはならない

Enterのあとに
Ctrl-d を入力すると
入力完了となる

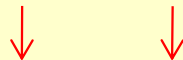
- Ctrl-d の入力については補足説明を参照のこと

例題: 変数の宣言

```
#include <stdio.h>
```

```
int main()  
{
```

変数の初期化



```
    int data, count = 0, total = 0;  
    float average;
```

- 入力データ (data), 入力個数 (count), 合計点 (total) は int 型
- 変数を宣言するときに値を初期化する
 count = 0, total = 0;
- 平均点 (average) は float 型

scanf() の使い方

- scanf() を実行すると値が返ってくる
 - scanf() もただの関数. あらかじめ定義しておいてあるだけ
- 入力データがある場合:
 - 変数への代入が行われたデータの個数が返される
- 入力データがない場合 (Ctrl+d を押したときなど):
 - EOF という特別な値が返ってくる (実際には負の値)
- scanf("%d", &data) を実行すると・・・
 - 入力データがあれば, それを data に代入し 1 を返す
 - 入力データがなければ, EOF という値が返ってくる

例題：平均点

```
#include <stdio.h>

int main()
{
    int data, count = 0, total = 0;
    float average;

    printf("input scores¥n");

    // Ctrl+d が入力されるまで, scanf を使って点数を読み込む
    // scanf() が EOF を返すまでループの本体を繰り返す
    while (scanf("%d", &data) != EOF) {
        total += data;
        count++;
    }
```

例題: 平均点

```
#include <stdio.h>

int main()
{
    int data, count = 0, total = 0;
    float average;

    while (scanf("%d", &data) != EOF) {
        total += data;
        count++;
    }
    if (count > 0) {
        average = (float)total / count;
        printf("number of items = %d¥n", count);
        printf("total = %d¥n", total);
        printf("average = %5.2f¥n", average);
    } else {
        printf("no data item¥n");
    }
    return 0;
}
```

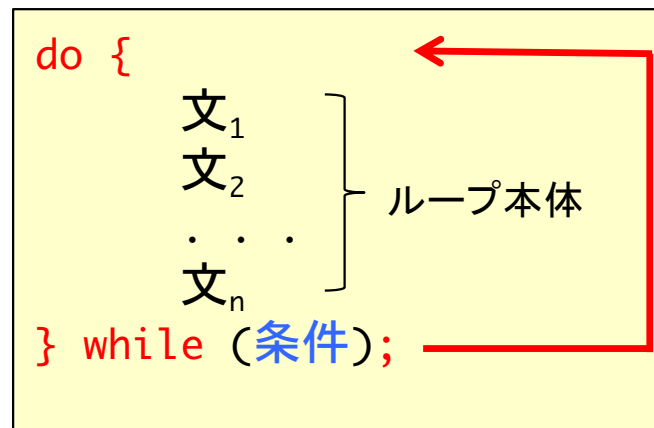
・ 入力があった場合

・ 入力がなかった場合

do ~ while 文

■ while 文と似ている

- **まず**, ループ本体 (文₁ ~ 文_n) を実行する
- **それから**, 条件が成り立っているかどうか調べる
 - ◆ 条件が成り立っていたら, 文₁ から処理を繰り返す



while 文との違い（その1）

■ while 文の場合:

- **まず, 条件**が成り立つかどうか調べる
 - ◆ 条件が成り立っていたらループの本体(文₁から文_n)を実行する
 - ◆ その後, 再び条件が成り立つかどうかを調べる

```
while (条件) {  
    文1  
    文2  
    . . .  
    文n  
}  
}
```

ループ本体

while 文との違い (その2)

- while 文では, まず, $i > 0$ かどうか調べる. そのため実行結果は:

- done

```
int i = 0;
while (i > 0) {
    printf("i = %d\n", i);
}
printf("done\n");
```

- do-while 文では, ループ本体を実行後, $i > 0$ かどうか調べる. そのため実行結果は:

- $i = 0$
done

```
int i = 0;
do {
    printf("i = %d\n", i);
} while (i > 0);
printf("done\n");
```

例題: 最大公約数

- キーボードから 2 つの正整数 a, b ($a \geq b$) を読み込み, その最大公約数 (G.C.D.) を求めなさい
- ユークリッドの互除法を使う
 - google で「ユークリッド 互除法」で検索
 - 2 つの正整数 m, n ($m \geq n$) の最大公約数は・・・
 1. m を n で割った余りを r とする
 2. m に n を代入する
 3. n に r を代入する
 4. もし, r が 0 なら m が最大公約数
 5. そうでなければ, 1. に戻る

ユークリッドの互除法: 具体例

- 20 と 8 の最大公約数を求める
 - $m = 20, n = 8$ とする
 - r に $20 / 8$ の余りを入れる. すなわち, $r = 4$
 - m に n の値を入れる. すなわち, $m = 8$
 - n に r の値を入れる. すなわち, $n = 4$
 - r の値は 0 ではないので繰り返し
 - r に $8 / 4$ の余りを入れる. すなわち, $r = 0$
 - m に n の値を入れる. すなわち, $m = 4$
 - n に r の値を入れる. すなわち, $n = 0$
 - r の値が 0 なので, m が最大公約数となる
 - よって, 20 と 8 の最大公約数は 4

1. m を n で割った余りを r とする
2. m に n を代入する
3. n に r を代入する
4. もし, r が 0 なら m が最大公約数
5. そうでなければ, 1. に戻る

最大公約数: 変数宣言と入力

```
#include <stdio.h>
```

```
int main()
```

```
{
```

```
    int a, b, m, n, r;
```

• a, b, m, n, r はすべて整数

```
    scanf("%d %d", &a, &b);
```

```
    return 0;
```

```
}
```

最大公約数: m, n の初期化

```
#include <stdio.h>
```

```
int main()  
{
```

```
    int a, b, m, n, r;
```

```
    scanf("%d %d", &a, &b);
```

```
    m = a;
```

```
    n = b;
```

• m, n に a, b を代入. 簡単のため $a \geq b$ は仮定してよい

```
    return 0;
```

```
}
```

最大公約数: ループの本体

```
#include <stdio.h>

int main()
{
    int a, b, m, n, r;

    scanf("%d %d", &a, &b);

    m = a;
    n = b;
    do {
        r = m % n;
        m = n;
        n = r;
    } while (r != 0);

    return 0;
}
```

1. m を n で割った余りを r とする
2. m に n を代入する
3. n に r を代入する
4. もし, r が 0 なら m が最大公約数
5. そうでなければ, 1. に戻る

- r に m を n で割った剰余を代入
- m に n を代入し, n に r を代入

最大公約数: do-while 文の利用

```
#include <stdio.h>

int main()
{
    int a, b, m, n, r;

    scanf("%d %d", &a, &b);

    m = a;
    n = b;

    do {
        r = m % n;
        m = n;
        n = r;
    } while (r != 0);

    printf("G.C.D of %d and %d is %d\n", a, b, m);
    return 0;
}
```

1. m を n で割った余りを r とする
2. m に n を代入する
3. n に r を代入する
4. もし, r が 0 なら m が最大公約数
5. そうでなければ, 1. に戻る

do-while文

- 剰余(r)が0でない場合はステップ1に戻る
- 剰余(r)が0の場合はdo-while文を終了

最大公約数: 実行例



■ プログラムを入力して実行してみよう！

プログラムを実行

231 110

G.C.D of 231 and 110 is 11

プログラムを実行

689 403

G.C.D of 689 and 403 is 13

break 文

■ ループを途中で抜けるためのもの

- for 文や while 文のループ本体からループの外に抜ける
 - ◆ 例: ループの本体実行中に, ある条件が成立したらループから抜ける

```
for (i = 0; i < n; i++) {  
    ...  
    if (条件) {  
        break;  
    }  
    ...  
}
```

for 文を抜けて,
for 文の次の文
から実行を続ける

```
while (i < n) {  
    ...  
    if (条件) {  
        break;  
    }  
    ...  
}
```

do-while 文でも使える

while 文を抜けて,
while 文の次の文
から実行を続ける

例題：平均点の計算

- 複数個のテストの点を読み込み、それらの合計点と平均点を求め、読み込んだデータの個数とともに表示しなさい。ただし、負の数を入力したところで点数の読み込みを終えるものとする

```
input scores
```

```
78 83 89 100 65 87 <
```

```
59 80 67 <
```

```
-1 <
```

```
number of items = 9
```

```
total = 708
```

```
average = 78.67
```

Enter を入力しても入力終了にはならない

負の数を入力すると
入力完了となる

例題: 平均点の計算

```
#include <stdio.h>
int main()
{
    int data, count = 0, total = 0;
    float average;

    while (scanf("%d", &data) != EOF) {
        if (data < 0) { break; }
        total += data;
        count++;
    }
    if (count > 0) {
        average = (float)total / count;
        printf("number of items = %d¥n", count);
        printf("total = %d¥n", total);
        printf("average = %5.2f¥n", average);
    } else {
        printf("no data item¥n");
    }
    return 0;
}
```



負の数が入力されたら while から抜ける

例題：101 が素数かどうかの判定

- 101 を 2, 3, 4, 5, ..., 100 で順に割ってみる
 - どれかで割り切れたら素数ではない
 - どれでも割り切れなかったら素数

```
#include <stdio.h>
int main()
{
    int i;
    for (i = 2; i < 101; i++) {
        if (101 % i == 0) {           // 割り切れる数が見つかったら for 文を抜ける
            break;
        }
    }
    if (i >= 101) {                  // for ループを最後まで実行したのなら
        printf("prime number\n");
    } else {
        printf("not a prime number\n");
    }
    return 0;
}
```

例題：素数かどうかの判定

- 正整数 n をキーボードから読み込み, その整数が素数かどうか判定するプログラムを作りなさい
 - 考え方: n を $2, 3, \dots, n-1$ で順に割ってみればよい
 - 少し工夫しよう
 - $2 \leq i \leq \sqrt{n}$ を満たす任意の整数 i で割り切れなければ n は素数
- 
- $4 \leq i * i \leq n$ を満たす任意の整数 i で割り切れなければ n は素数
- 
- i の値を $2, 3, 4$ と増やしながら, n / i が割り切れるかどうか調べる
ただし, $i * i > n$ となったらやめる

例題：素数かどうかの判定

```
#include <stdio.h>
int main()
{
    int i, n;
    scanf("%d", &n);

    for (i = 2; i * i <= n; i++) {          // i * i <= n, つまり  $i \leq \sqrt{n}$  まで i を増やす
        if (n % i == 0) {                  // 割り切れるかどうか調べる
            break;
        }
    }

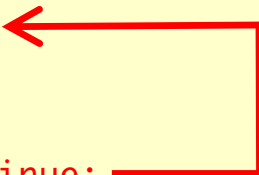
    if (i * i > n) { // for 文を最後まで回ったのかどうか調べる
        printf("prime number\n");
    } else {
        printf("not a prime number\n");
    }

    return 0;
}
```

continue 文

- ループ本体の処理を(一部)飛ばして、繰り返しを続ける
 - for 文, while 文, do-while 文で使える

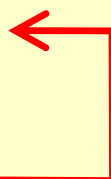
```
while (i < n) {  
    ...  
    if (条件) {  
        continue;  
    }  
    ...  
}
```



条件を満たした場合：

- ・ $i < n$ が成り立つかどうか調べる
($i < n$ が成り立つならループ本体を実行)
($i < n$ が成り立たないなら while 文を抜ける)

```
for (i = 0; i < n; i++) {  
    ...  
    if (条件) {  
        continue;  
    }  
    ...  
}
```



条件を満たした場合：

- ・ $i++$ を実行する
- ・ $i < n$ かどうか調べる
($i < n$ が成り立つならループ本体を実行)
($i < n$ が成り立たないなら for 文を抜ける)

continue 文の例

■ i の値が 2 で割り切れるときは、次の行はスキップ

◆ `printf("i = %d¥n", i);`

```
#include <stdio.h>
main()
{
    int i;
    for (i = 1; i < 5; i++) {
        if (i % 2 == 0) {
            continue;
        }
        printf("i = %d¥n", i);
    }
    return 0;
}
```

実行結果

```
i = 1
i = 3
```

- i に 1 を代入する
- i の値は 2 で割り切れない
- `printf("i = %d¥n", i)` を実行
- i = 1 が表示される
- (for 文なので) i++ を実行し、i の値は 2 になる
- i < 5 が成立するため、ループ本体を実行する
- i は 2 で割り切れる
- continue を実行する
(`printf("i = %d¥n", i)` はスキップ)
- (for 文なので) i++ を実行し、i の値は 3 になる
- i < 5 が成立するため、ループ本体を実行する

例題: 偶数の和

- 複数個の整数を読み込み, それらの合計を求めなさい. ただし, 偶数の場合のみ合計に含めるようにし, 負の数を入力したところで整数の読み込みを終えるものとする

input numbers

4 8 3 2

-1

sum of even numbers = 14

負の数を入力すると
入力完了となる

入力された値のうち, 偶数の合計を求める.
つまり, 入力された 4, 8, 3, 2 のうち,
偶数である 4, 8, 2 の合計になる

例題：偶数の和

```
#include <stdio.h>
int main()
{
    int sum = 0;

    printf("input numbers¥n");
    while (scanf("%d", &data) != EOF) {
        if (data < 0) {          // 負の数が入力されたら終了
            break;
        }
        if (data % 2 != 0) { // 奇数は合計に含めない
            continue;
        }
        sum += data;
    }
    printf("sum of even numbers = %d¥n", sum);
    return 0;
}
```

- 制御構造について学んだ
 - switch 文
 - while 文
 - do-while 文
- 前半で学んだ if 文, for 文でも事は足りる
 - 「わかりやすさ」に応じて適切なものを選ぶようにする
- 制御構造を制御する仕組み
 - break 文
 - continue 文
 - 多用するとぐちゃぐちゃになるので注意