

初級プログラミング I

第5回 繰り返し

中谷 祐介

第5回の構成

◆ 第1講 for 文による繰り返し

- Java における繰り返しの概要, および for 文による繰り返しの基本事項を理解する.

◆ 第2講 while 文による繰り返し

- while 文による繰り返しについて, 基本事項を理解する.

◆ 第3講 繰り返しによるプログラムの作成

- 繰り返しを用いた簡単なプログラムが作成できるようになる.

◆ 第4講 多重ループ

- より複雑な繰り返しを行う二重ループについて理解する.

第5回 繰り返し

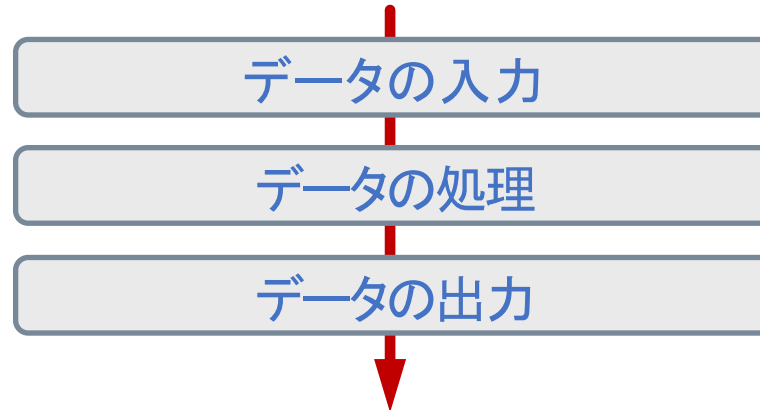
④ 第1講 `for` 文による繰り返し

第1講の学習目標

- ◆ Java における繰り返しについて、基本事項を理解する:
 - 繰り返し
 - for 文
 - 複合代入演算子
 - インクリメント・デクリメント演算子

プログラムの流れ

- ◆ プログラムは、原則として上から下に向かって、記述した順に処理を行う。



プログラムの流れの例

- プログラムとして実現できることが限定される。
- ◆ プログラムの流れを制御する方法を学ぶことで、実現できる範囲が広がる。
 - 条件分岐 — ある条件に従って、その後の処理を変える。 ➡ **第3回**
 - 繰り返し — 同様のことを繰り返し処理する。

例題5-1

例題5-1

1から1000までの整数の総和を計算し、計算結果を標準出力に出力するプログラムを作成してください。

◆ ここまでの知識でプログラムを作成すると、例えば…

```
int sum = 0;

sum = sum + 1;
sum = sum + 2;
sum = sum + 3;
    ⋮
sum = sum + 999;
sum = sum + 1000;
System.out.println("sum : " + sum);
```

- 上のプログラムでも正しく動作する。
- 同じ作業である「`sum = sum + i;`」を1000回記述しなければならない。

繰り返し

- ◆ **繰り返し**(ループ)は、同様の処理を繰り返す制御構造のことである。
- ◆ 繰り返しを実現する方法として、以下を習得する:

- for 文

```
for (初期設定式; 条件式; 繰り返し式) {  
    // 繰り返したいこと  
}
```

- while 文

```
while (条件式) {  
    // 繰り返したいこと  
}
```

- ◆ 前のスライドにおいて...

```
int sum = 0;  
  
sum = sum + 1;  
sum = sum + 2;  
⋮  
sum = sum + 999;  
sum = sum + 1000;  
System.out.println("sum : " + sum);
```

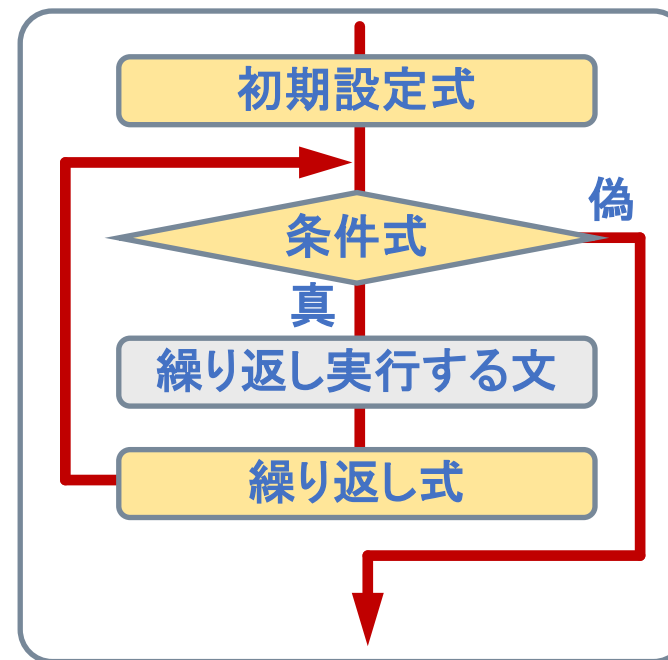
➡ 「**sum = sum + i;**」の繰り返し

for文

- ◆ for文は、同じ処理を繰り返し実行することを実現する文である。

```
for (初期設定式①; 条件式②; 繰り返し式④) {  
    繰り返し実行する文③  
}
```

- ◆ for文は、() の中に、2つの「;」で区切り3つの式を記述する。
- ◆ 以下の手順で処理を行う:
 - 最初に初期設定式①を実行する。
 - 条件式②の判定を行い、「偽」ならfor文を終了する。
 - { }内の繰り返し実行する文③を実行する。
 - 繰り返し式④を実行する。
 - 条件式②の判定に戻る。



for文の処理の流れ

[Sample_05_01.java]

◆ 3回の繰り返しの場合:

```
for (int i=1; i<=3; i=i+1) {  
    System.out.print(i + " ");  
}
```

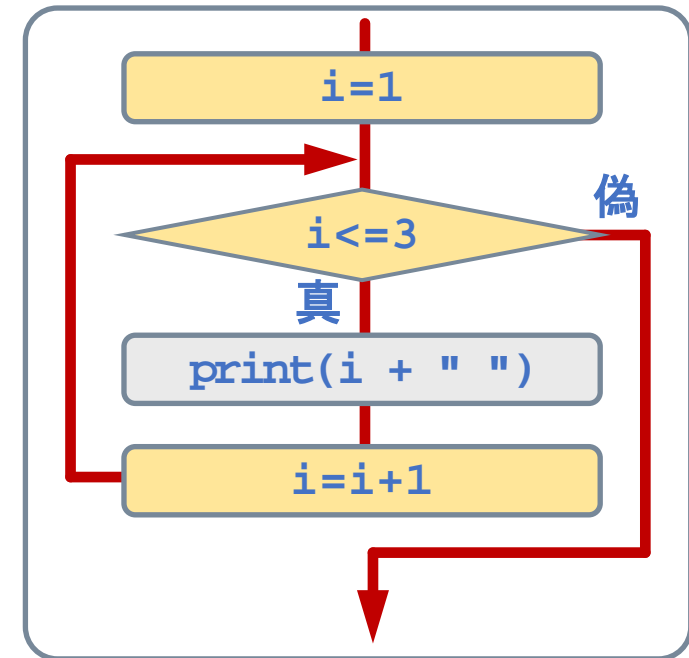
- 繰り返しのための変数 i を宣言し「1」で初期化.
- 変数 i の値が「3」以下の間, 繰り返す.
- 対象の処理が終了したら, 変数 i を「1」増やす.

変数 i

4

標準出力

1 2 3



1から3までの3回の繰り返しの流れ

for 文の使用例 (1)

[Sample_05_02.java]

◆ for文の使用例(5回の繰り返し):

```
for (int i=1; i<=5; i=i+1) {  
    System.out.print(i + " ");  
}
```

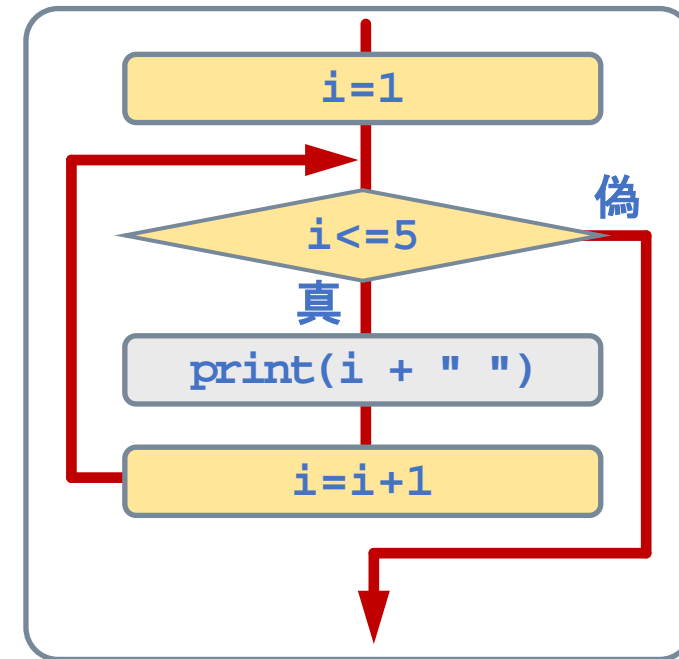
1 2 3 4 5

- iが「1」から「5」までの5回の繰り返し.

```
for (int i=0; i<5; i=i+1) {  
    System.out.print(i + " ");  
}
```

0 1 2 3 4

- iが「0」から「4」までの5回の繰り返し.



1から5までの5回の繰り返しの流れ

◆ 繰り返しの変数 `i` の有効範囲:

```
for (int i=1; i<=5; i=i+1) {  
    System.out.print(i + " ");  
}
```

```
System.out.println("¥n" + i);
```

← エラー

- 変数 `i` は, for 文の中だけで有効.

```
int i;  
for (i=1; i<=5; i=i+1) {  
    System.out.print(i + " ");  
}  
System.out.println("¥n" + i);
```

```
1 2 3 4 5  
6
```

- 変数 `i` は, 宣言した位置より後であれば, for 文の外でも有効.

「例題5-1」の場合の for 文 (1)

◆ 「例題5-1」の場合…

```
int sum = 0;  
  
sum = sum + 1;  
sum = sum + 2;  
⋮  
sum = sum + 1000;
```

- 「sum = sum + **i**」を **i** の値を「1」から「1000」まで「1」ずつ変化させながら1000回繰り返せばよい。



```
int sum = 0;  
for (int i=1; i<=1000; i=i+1) {  
    sum = sum + i;  
}
```

- 複合代入演算子, インクリメント演算子を使用することで, より簡潔に記述することができる。

- ◆ 複合代入演算子は、演算と代入を同時に行う演算子である。
- ◆ 「+=」、「-=」、「*=」、「/=」、「%=」などがある:

```
a += 10;  
x *= 2;
```



```
a = a + 10;  
x = x * 2;
```

- ◆ 複合代入演算子の使用例

```
int a = 5;  
a += 10;  
System.out.println(a);
```

```
double x = 12.3;  
x *= 2;  
System.out.println(x);
```

```
15  
24.6
```

インクリメント・デクリメント演算子

[Sample_05_05.java]

- ◆ インクリメント演算子「++」は、変数の値を1つ増やす演算子である。
- ◆ デクリメント演算子「--」は、変数の値を1つ減らす演算子である。

```
i++;  
j--;
```



```
i = i+1;  
j = j-1;
```

- ◆ インクリメント・デクリメント演算子の使用例

```
int i = 1;  
i++;  
System.out.println(i);  
i++;  
System.out.println(i);  
i++;  
System.out.println(i);
```

```
2  
3  
4
```

```
int j = 10;  
j--;  
System.out.println(j);  
j--;  
System.out.println(j);  
j--;  
System.out.println(j);
```

```
9  
8  
7
```

「例題5-1」の場合の for 文 (2)

◆ 「例題5-1」の場合…

```
int sum = 0;
for (int i=1; i<=1000; i=i+1) {
    sum = sum + i;
}
```



```
int sum = 0;
for (int i=1; i<=1000; i++) {
    sum += i;
}
```

- インクリメント演算子, 複合代入演算子を使用することで, 簡潔に記述できる.

「例題5-1」のプログラム例

例題5-1(再掲)

1から1000までの整数の総和を計算し、計算結果を標準出力に出力するプログラムを作成してください。

```
1 public class Example_05_01 {
2     public static void main(String[] args) {
3         int sum = 0;
4         for (int i=1; i<=1000; i++) {
5             sum += i;
6         }
7         System.out.println("sum : " + sum);
8     }
9 }
```

Example_05_01.java

◆ 実行結果
標準出力

sum : 500500

第1講のまとめ

- ◆ Java における繰り返しについて、基本事項を理解した：
 - 繰り返し
 - for 文
 - 複合代入演算子
 - インクリメント・デクリメント演算子

第5回 繰り返し

 第1講 `for` 文による繰り返し

終わり

第5回 繰り返し

④ 第2講 while 文による繰り返し

第2講の学習目標

- ◆ while 文による繰り返しについて, 基本事項を理解する:
 - 繰り返し
 - while 文
 - do - while 文

繰り返し

◆ **繰り返し**(ループ)は、同様の処理を繰り返す制御構造のことである。

◆ **for 文** ⇒ 第1講で習得

```
for (初期設定式; 条件式; 繰り返し式) {  
    // 繰り返したいこと  
}
```

◆ **for 文の例**

```
for (int i=1; i<=1000; i++) {  
    // 繰り返したいこと  
}
```

- i が「1」から「1000」までの1000回の繰り返し

◆ **while 文** ⇒ ここで習得

```
while (条件式) {  
    // 繰り返したいこと  
}
```

while 文

◆ while文は、同じ処理を繰り返し実行することを実現する文.

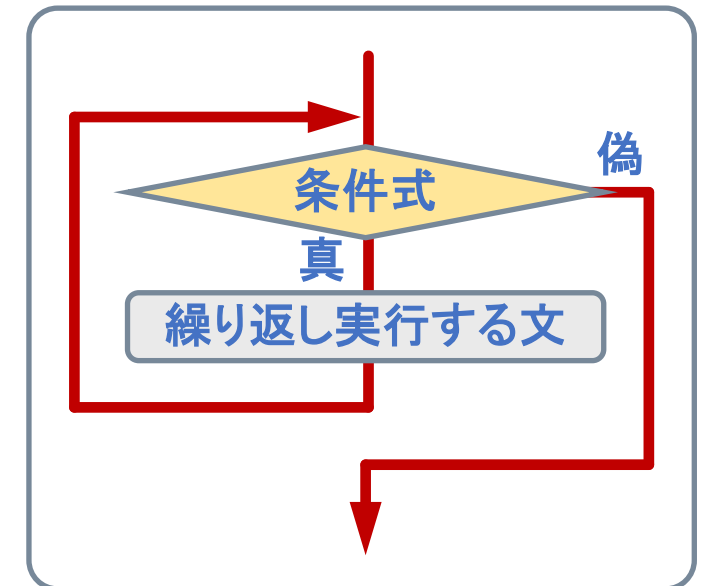
```
while (条件式①) {  
    繰り返し実行する文②  
}
```

◆ ある条件が成立している間、繰り返す.

◆ 以下の手順で処理を行う:

- 条件式①の判定を行い、「偽」ならwhile文を終了する.
- {}内の繰り返し実行する文②を実行する.
- 条件式①の判定に戻る.

◆ 「繰り返し実行する文」を実行しない可能性がある.



while文の処理の流れ

while 文の処理の流れ

[Sample_05_06.java]

- ◆ i が「3」以下という条件の繰り返しの場合:

```
int i = 1;
while (i<=3) {
    System.out.print(i + " ");
    i++;
}
```

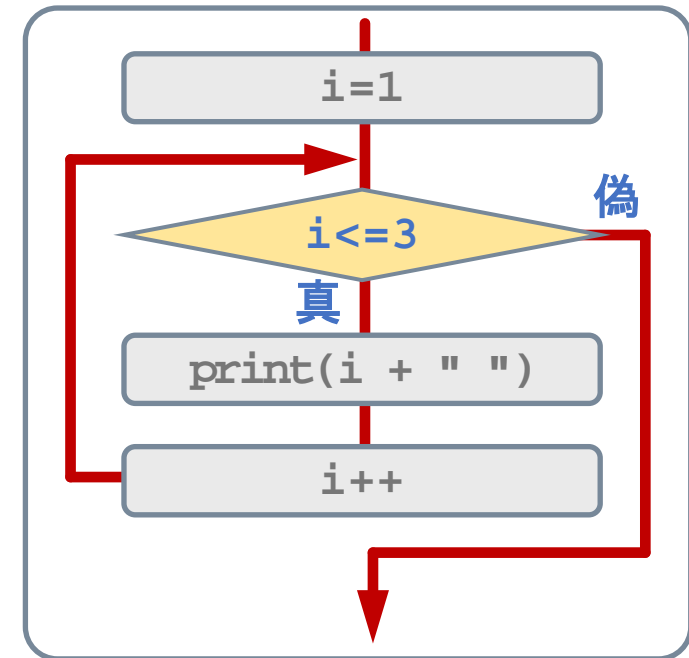
- 変数 i を宣言し「1」で初期化.
- 条件式「 $i \leq 3$ 」が成立する間, 繰り返す.
- 変数 i の内容を出力し, i の値を「1」増加させる.

変数 i

4

標準出力

1 2 3



i が「3」以下という条件の繰り返しの流れ

◆ 10から1までのカウントダウン

```
int i = 10;
while (i>0) {
    System.out.print(i + " ");
    i--;
}
```

10 9 8 7 6 5 4 3 2 1

- i が「0」より大きい間の繰り返し.

◆ 100以下の 2^i ($i = 1, 2, 3, \dots$) の出力

```
int n = 2;
while (n<=100) {
    System.out.print(n + " ");
    n *= 2;
}
```

2 4 8 16 32 64

- n が「100」以下の間の繰り返し. 繰り返すたびに, n に「2」を掛ける.

$$2^i = \underbrace{2 \times 2 \times \dots \times 2}_{i \text{ 個}}$$

「例題5-1」の場合の while 文

◆ 1から1000までの整数の総和の計算

```
int sum = 0;
for (int i=1; i<=1000; i++) {
    sum += i;
}
```



```
int sum = 0;
int i = 1;
while (i<=1000) {
    sum += i;
    i++;
}
```

- for 文の内容を, while 文を使用して記述することもできる.

「例題5-1」のプログラム例 (while 文使用版)

例題5-1(再掲)

1から1000までの整数の総和を計算し、計算結果を標準出力に出力するプログラムを作成してください。

```
1 public class Example_05_01_2 {
2     public static void main(String[] args) {
3         int sum = 0;
4         int i = 1;
5         while (i<=1000) {
6             sum += i;
7             i++;
8         }
9         System.out.println("sum : " + sum);
10    }
11 }
```

Example_05_01_2.java

◆ 実行結果
標準出力

sum : 500500

例題5-2 (1)

例題5-2

標準入力に正の整数値 n を入力すると, 2^i が n を超える最小の i および 2^i を求め, それらを標準出力に出力するプログラムを作成してください.

◆ 整数値 n が「100」のとき...

- 2^i が「100」を超える最小の i および 2^i を求める.

◆ 考え方

- 1に「2」を掛け続ける.
- 「2」を掛けるごとに, i の値を「1」増やす.
- 「100」を超えたら計算をやめる(「100」以下の間, 繰り返す).

$$\begin{array}{rcl} 2^i & & \\ \downarrow & & \\ 2^0 = 1 & & \\ 2^1 = 2 & \times 2 & \\ 2^2 = 4 & \times 2 & \\ 2^3 = 8 & \times 2 & \\ 2^4 = 16 & \times 2 & \\ 2^5 = 32 & \times 2 & \\ 2^6 = 64 & \times 2 & \\ 2^7 = 128 & \times 2 & \\ & & n \\ & & \downarrow \\ & & > 100 \end{array}$$

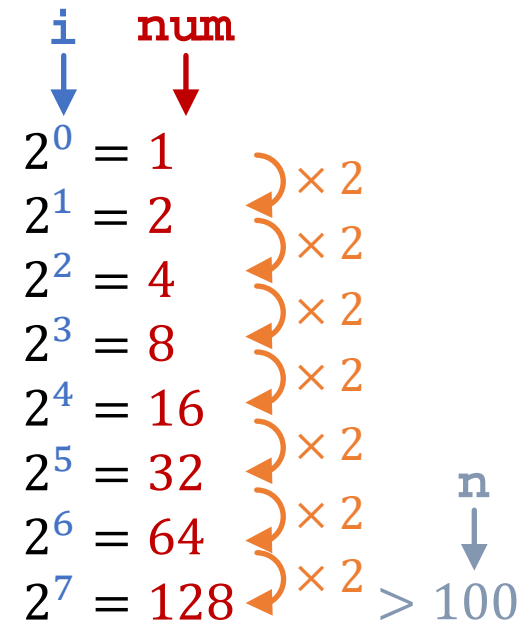
例題5-2 (2)

◆ プログラムの考え方

- int 型の変数 i, num を宣言し, i を「0」で, num を「1」で初期化.
- num が n 以下の間, 以下を繰り返す:
 - num に「2」を掛ける.
 - i を「1」増やす.
- i, num がそれぞれ求める i および 2^i .



```
int num = 1;
int i = 0;
while (num <= n) {
    num *= 2;
    i++;
}
```



「例題5-2」のプログラム例

例題5-2（再掲）

標準入力に正の整数値 n を入力すると, 2^i が n を超える最小の i および 2^i を求め, それらを標準出力に出力するプログラムを作成してください.

```
1 import java.util.Scanner;
2
3 public class Example_05_02 {
4     public static void main(String[] args) {
5         Scanner scan = new Scanner(System.in);
6
7         int n = scan.nextInt();
8         int num = 1;
9         int i = 0;
10        while (num <= n) {
11            num *= 2;
12            i++;
13        }
14        System.out.println("2^" + i + " : " + num);
15    }
16 }
```

Example_05_02.java

◆ 実行例

標準入力

100

標準出力

2^7 : 128↵

標準入力

10000

標準出力

2^14 : 16384↵

do - while 文

◆ while文に類似した反復として, do - while 文がある.

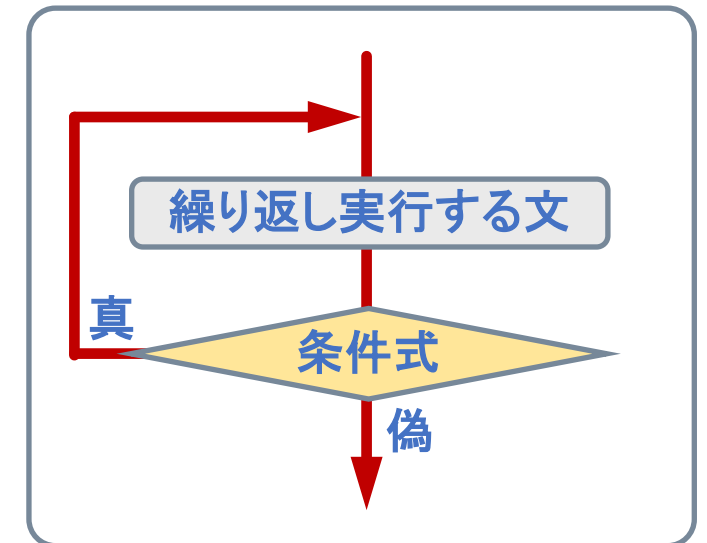
```
do {  
    繰り返し実行する文①  
} while (条件式②);
```

◆ 以下の手順で処理を行う:

- {}内の繰り返し実行する文①を実行する.
- 条件式②の判定を行い, 「偽」なら反復を終了する.
- {}内に戻る.

◆ 「繰り返し実行する文」を必ず1回実行する.

◆ 「while ()」の後に「;」が必要であることに注意.



do - while文の処理の流れ

◆ do - while 文の例

```
int i = 0;
do {
    System.out.print(i + " ");
    i++;
} while (i<5);
```

0 1 2 3 4

◆ while 文と do - while 文の比較

• while 文

```
i = 5;
while (i<5) {
    System.out.print(i + " ");
    i++;
}
```

(何も表示されない)

• do - while 文

```
i = 5;
do {
    System.out.print(i + " ");
    i++;
} while (i<5);
```

5

第2講のまとめ

- ◆ while 文による繰り返しについて, 基本事項を理解した:
 - 繰り返し
 - while 文
 - do - while 文

第5回 繰り返し

 第2講 `while` 文による繰り返し

終わり

第5回 繰り返し

第3講 繰り返しによるプログラムの作成

第3講の学習目標

- ◆ 繰り返しを用いた簡単なプログラムが作成できるようになる:
 - 例題を通してのプログラム作成
 - キャスト演算子

例題5-3 (1)

例題5-3

標準入力に摂氏温度($^{\circ}\text{C}$)を表す2つの整数値 c_1, c_2 ($c_1 \leq c_2$) をこの順に入力すると、以下のことを行うプログラムを作成してください:

- c_1 度から c_2 度まで, 1度ごとに華氏温度($^{\circ}\text{F}$)に変換し, 小数点以下1桁まで出力する.
- $c_1 > c_2$ の場合は, 何も出力しない.
- 摂氏温度 C から華氏温度 F への変換は次のとおり : $F = 1.8C + 32.0$

◆ 実行例

標準入力

```
20
25
```

標準出力

```
20 : 68.0↵
21 : 69.8↵
22 : 71.6↵
23 : 73.4↵
24 : 75.2↵
25 : 77.0↵
```

「例題5-3」(2)

◆ プログラムの内容

1. 摂氏温度($^{\circ}\text{C}$)を表す2つの整数値 c_1, c_2 ($c_1 \leq c_2$) を読み込む.
2. $c_1 > c_2$ の場合は, 何も出力しない.
3. c_1 度から c_2 度まで, 1度ごとに華氏温度($^{\circ}\text{F}$)に変換し, 小数点以下1桁まで出力する.

◆ 摂氏温度($^{\circ}\text{C}$)を表す2つの整数値 c_1, c_2 ($c_1 \leq c_2$) を読み込む.

```
Scanner scan = new Scanner(System.in);
```

```
int c1 = scan.nextInt();
```

```
int c2 = scan.nextInt();
```

- 2つの `int` 型の変数 `c1, c2` に整数値を読み込む.

「例題5-3」(3)

- ◆ $c_1 > c_2$ の場合は, 何も出力しない. $\Rightarrow c_1 \leq c_2$ の場合のみ, 出力する.

```
if (c1 <= c2) {  
    // c1からc2まで, 1度ごとに出力  
}
```

- 条件分岐 (if 文) により, $c_1 \leq c_2$ の場合のみ, 出力

- ◆ c_1 度から c_2 度まで, 1度ごとに華氏温度($^{\circ}\text{F}$)に変換し, 小数点以下1桁まで出力する.

```
for (int i=c1; i<=c2; i++) {  
    double f = 1.8*i + 32.0;  
    System.out.printf("%d : %.1f¥n", i, f);  
}
```

- for 文により, c_1 から c_2 まで, 1ずつ増やしながら繰り返す.

「例題5-3」のプログラム例

Example_05_03.java

```
1 import java.util.Scanner;
2
3 public class Example_05_03 {
4     public static void main(String[] args) {
5         Scanner scan = new Scanner(System.in);
6
7         int c1 = scan.nextInt();
8         int c2 = scan.nextInt();
9
10        if (c1 <= c2) {
11            for (int i=c1; i<=c2; i++) {
12                double f = 1.8*i + 32.0;
13                System.out.printf("%d : %.1f\n", i, f);
14            }
15        }
16    }
17 }
```

◆ 実行例

標準入力

標準出力

15
20

15 : 59.0←
16 : 60.8←
17 : 62.6←
18 : 64.4←
19 : 66.2←
20 : 68.0←

例題5-4 (1)

例題5-4

ある科目のテストの点数データがあるとして、標準入力に、データの個数と点数データを表す整数値を入力すると、点数の合計と平均を計算し、データの個数、総和、平均を標準出力に出力するプログラムを作成してください。平均は、小数点以下2桁まで出力するとして。

◆ 実行例

標準入力

データの個数 ← 4
点数データ ← { 88
76
91
80

標準出力

個数 : 4↵
合計 : 335↵
平均 : 83.75↵

「例題5-4」(2)

◆ プログラムの内容

1. 点数データの個数を表す整数値を読み込む。
2. データの個数分だけ、点数データを表す整数値を読み込みながら、合計を計算する。
3. 点数データの合計と個数から平均を計算する。
4. データの個数、総和、平均(小数点以下2桁まで)を出力する。

◆ 点数データの個数を表す整数値を読み込む。

```
Scanner scan = new Scanner(System.in);
```

```
int n = scan.nextInt();
```

- int 型の変数 n に整数値を読み込む。

「例題5-4」(3)

- ◆ データの個数分だけ、点数データを表す整数値を読み込みながら、合計を計算する。

```
int total = 0;
for (int i=1; i<=n; i++) {
    int data = scan.nextInt();
    total += data;
}
```

- for 文により、n 回(データの個数分)繰り返す。
- int 型の変数 data に点数データを読み込む。
- int 型の変数 total に合計を格納する。最初は「0」で初期化。
- total に data の値を加算し続けることで、最終的に total に合計の結果が格納される。

「例題5-4」(4)

◆ 点数データの合計と個数から平均を計算する.

- 平均は「合計/データの個数」で計算できる:

```
double ave = total / n;
```

- total, n とともに int 型のため, int 型同士の計算結果は小数点以下が切り捨てられ整数値として計算される.
- total, n のどちらかを double 型にすればよい.

- **キャスト演算子**(型変換)「(型)式」により, int 型を double 型に変換し計算する:

```
double ave = (double)total / n;
```

- 「(double)total」により, この部分は double 型として認識される.
- その結果, 「double 型 / int 型」の計算となり, 計算結果は double 型.

◆ データの個数, 総和, 平均(小数点以下2桁まで)を出力する.

- printf メソッドを使用して出力する.

「例題5-4」のプログラム例

Example_05_04.java

```
1 import java.util.Scanner;
2
3 public class Example_05_04 {
4     public static void main(String[] args) {
5         Scanner scan = new Scanner(System.in);
6
7         int n = scan.nextInt();
8
9         int total = 0;
10        for (int i=1; i<=n; i++) {
11            int data = scan.nextInt();
12            total += data;
13        }
14        double ave = (double)total / n;
15
16        System.out.printf("個数 : %d¥n", n);
17        System.out.printf("合計 : %d¥n", total);
18        System.out.printf("平均 : %.2f¥n", ave);
19    }
20 }
```

◆ 実行例

標準入力

6
91
78
72
83
95
81

標準出力

個数 : 6↵
合計 : 500↵
平均 : 83.33↵

例題5-5

例題5-5

標準入力に正の整数値を入力すると、その整数値の桁数を標準出力に出力するプログラムを作成してください。

◆ どう考える？ 入力値：1357 ⇒ 出力値：4

- 整数値同士の割り算は、小数点以下が切り捨てられる。
- 整数値を「10」で割った商を計算すれば、桁数を1桁減らせる。

- 1357 / 10 → 135 1桁
- 135 / 10 → 13 2桁
- 13 / 10 → 1 3桁
- 1 / 10 → 0 4桁

- 整数値が「0」より大きい間、「10」で割り続け、その回数を数える。

```
int n = scan.nextInt();  
  
int digit = 0;  
while (n > 0) {  
    n /= 10;  
    digit++;  
}
```

「例題5-5」のプログラム例

Example_05_05.java

```
1 import java.util.Scanner;
2
3 public class Example_05_05 {
4     public static void main(String[] args) {
5         Scanner scan = new Scanner(System.in);
6
7         int n = scan.nextInt();
8
9         int digit = 0;
10        while (n > 0) {
11            n /= 10;
12            digit++;
13        }
14
15        System.out.println(digit);
16    }
17 }
```

◆ 実行例

標準入力

13579

標準出力

5↵

標準入力

50

標準出力

2↵

第3講のまとめ

- ◆ 繰り返しを用いた簡単なプログラムを作成した:
 - 例題を通してのプログラム作成
 - キャスト演算子

第5回 繰り返し

 第3講 繰り返しによるプログラムの作成

終わり

第5回 繰り返し

 第4講 多重ループ

第4講の学習目標

- ◆ 繰り返しについて、以下の事項を理解する:
 - 多重ループ
 - break 文
 - continue 文

多重ループ

- ◆ ループ(繰り返し)の中でループを使用することで二重のループを行うことができ、さらにその中でループを行うと三重のループを行うことができる。
- ◆ 一般に、ループの処理を入れ子構造で記述したものを、**多重ループ**と呼ぶ。

- 二重ループの例

```
for (int i=1; i<=m; i++) {  
    for (int j=1; j<=n; j++) {  
        // 繰り返したいこと  
    }  
}
```

- ◆ 多重ループを用いることで、より複雑な処理を行うプログラムを作成することができる。
- ◆ 本講では、多重ループのうち、基本的な二重ループについて具体的な例を用いて解説し、多重ループの概念を理解する。

二重ループの例 (1)

[Sample_05_09.java]

◆ 次のプログラムの出力結果はどうなる？

```
for (int i=1; i<=5; i++) {  
    for (int j=1; j<=5; j++) {  
        System.out.print(j + " ");  
    }  
    System.out.println();  
}
```

} ■ 1 2 3 4 5 ←

```
1 2 3 4 5 ←  
1 2 3 4 5 ←  
1 2 3 4 5 ←  
1 2 3 4 5 ←  
1 2 3 4 5 ←
```

- 外側の for 文 : { } 内を5回繰り返す.
- 内側の for 文 : j の値を「1」から「5」まで「1」ずつ変化させながら出力.

二重ループの例 (2)

[Sample_05_09.java]

◆ 次のプログラムの出力結果はどうなる？

```
for (int i=1; i<=5; i++) {  
    for (int j=1; j<=5; j++) {  
        System.out.print(i*j + " ");  
    }  
    System.out.println();  
}
```

```
1 2 3 4 5 ←  
2 4 6 8 10 ←  
3 6 9 12 15 ←  
4 8 12 16 20 ←  
5 10 15 20 25 ←
```

- 外側の for 文 : i の値を「1」から「5」まで「1」ずつ変化させながら `{ }` 内を繰り返す.
- 内側の for 文 : j の値を「1」から「5」まで「1」ずつ変化させながら「 $i*j$ 」を出力.

[i が「1」のとき]

```
for (int j=1; j<=5; j++) {  
    System.out.print(1*j + " ");  
}  
System.out.println();
```

1 2 3 4 5 ←

[i が「2」のとき]

```
for (int j=1; j<=5; j++) {  
    System.out.print(2*j + " ");  
}  
System.out.println();
```

2 4 6 8 10 ←

⋮

例題5-6

例題5-6

標準入力に正の整数値 n を入力すると, 文字「*」を使用して, 下図に示すような n 段の三角形を標準出力に出力するプログラムを作成してください.

標準入力

5

標準出力

```
*  
**  
***  
****  
*****
```

- ◆ 二重ループを利用して, プログラムを作成する.

三角形の出力

◆ 1段目の出力 (i が「1」のとき)

```
for (int j=1; j<=1; j++) {  
    System.out.print("*");  
}  
System.out.println();
```

*←

◆ 2段目の出力 (i が「2」のとき)

```
for (int j=1; j<=2; j++) {  
    System.out.print("*");  
}  
System.out.println();
```

**←

◆ 3段目の出力 (i が「3」のとき)

```
for (int j=1; j<=3; j++) {  
    System.out.print("*");  
}  
System.out.println();
```

***←

```
1段目: 1個の「*」 → *←  
2段目: 2個の「*」 → **←  
3段目: 3個の「*」 → ***←  
****←  
*****←
```

↑ n が「5」のとき

```
for (int i=1; i<=n; i++) {  
    for (int j=1; j<=i; j++) {  
        System.out.print("*");  
    }  
    System.out.println();  
}
```



「例題5-6」のプログラム例

```
1 import java.util.Scanner;
2
3 public class Example_05_06 {
4     public static void main(String[] args) {
5         Scanner scan = new Scanner(System.in);
6
7         int n = scan.nextInt();
8
9         for (int i=1; i<=n; i++) {
10            for (int j=1; j<=i; j++) {
11                System.out.print("*");
12            }
13            System.out.println();
14        }
15    }
16 }
```

Example_05_06.java

◆ 実行例

標準入力

5

標準出力

```
*  
**  
***  
****  
*****
```


繰り返しの流れを制御する文

◆ 繰り返しの流れを制御する文として、**break 文**と **continue 文**がある。

◆ **break 文**


- break 文は、繰り返し(for 文, while 文, do-while 文)または switch 文の中で使用することができる。
- break 文に出会うと、その break 文が記述されている繰り返しや switch 文を強制的に抜け出す。

◆ **continue 文**

- continue 文は、繰り返し(for 文, while 文, do-while 文)の中で使用することができる。
- continue 文に出会うと、その位置からその繰り返し回の終わりまでをスキップし、次の回に処理を移す。

- ◆ break 文は、繰り返し(for 文, while 文, do-while 文)または switch 文の中で使用することができる。
- ◆ break 文に出会うと, その break 文が記述されている繰り返しや switch 文を強制的に抜け出す。
- ◆ break 文の使用例

```
for (int i=1; i<=10; i++) {  
    if (i == 6) break;  
    System.out.print(i + " ");  
}  
System.out.println();
```




1 2 3 4 5 ←

- 上の場合, 変数 `i` の値が「6」になった段階で for 文から脱出する。

- ◆ continue 文は、繰り返し(for 文, while 文, do-while 文)の中で使用することができる。
- ◆ continue 文に出会うと、その位置からその繰り返し回の終わりまでをスキップし、次の回に処理を移す。
- ◆ continue 文の使用例

```
for (int i=1; i<=10; i++) {  
    if (i%3 == 0) continue;  
    System.out.print(i + " ");  
}  
System.out.println();
```



1 2 4 5 7 8 10 ←

- 上の場合、変数 i の値が「3」の倍数のときは、その回の処理をスキップする。

break 文と continue 文の使用例

◆ 次のプログラムの出力結果はどうなる？

```
public class Sample_05_11 {  
    public static void main(String[] args) {  
        for (int i=1; i<=10; i++) {  
            if (i%3 == 0) continue;  
            if (i == 6) break;  
            System.out.print(i + " ");  
        }  
        System.out.println();  
    }  
}
```

- どうなるか考えた後、実際にプログラムを作成し、結果を確認してみよう。

第4講のまとめ

- ◆ 繰り返しについて、以下の事項を理解した：
 - 多重ループ
 - break 文
 - continue 文

第5回 繰り返し

 第4講 多重ループ

終わり