Presented by Quentin Ochem

university.adacore.com

幾つかの全般的情報

- 文は複数の行にまたがる場合があります．また，「;」によって終了する必要があります

```
Put_Line
(
"Hello World"
);
```

- 文は，一連の文として（指定の場所に）記述する必要があります

```
procedure Main is
   Put_Line ("Hello World");
begin
   Put_Line ("Hello World");
end Main;
```

If 文/ If 式

- If 文

Ada

```
if A = 0 then
   Put_Line ("A is 0");
elsif B = 0 then
   Put_Line ("B is 0");
else
   Put_Line ("Else... ");
end if;
```

C++

```
if (A == 0) {
   printf ("A is 0");
} else if (B == 0) {
   printf ("B is 0");
} else {
   printf ("Else...");
}
```

- If 式

Ada

```
Put_Line (
   (if A = 0 then
     "A is 0"
    elsif B = 0 then
     "B is 0"
    else
     "Else"));
```

C++

```
printf (A == 0 ? "A is 0" :
          (B == 0 ? "B is 0" :
            "Else..."));
```

条件演算子

- 比較

=	/=	<	<=	>	>=

- 二値ブール演算子と省略制御形式

二値ブール演算子			省略 制御形式	
or	and	xor	or else	and then

二値ブール演算子についての注記

- 「and」，「or」は省略制御形式ではありません．両被演算子は，常に評価されます

```
if X /= 0 and Y / X > 1 then -- MAY RAISE AN EXCEPTION
```

- 省略制御形式による実行は，「and then」と「or else」です

```
if X /= 0 and then Y / X > 1 then -- OK
```

Case文とCase式

Ada

```
case A is
   when 0 =>
      Put_Line ("zero");
   when -9 .. -1 | 1 .. 9 =>
      Put_Line ("digit");
   when others =>
      Put_Line ("other")
end case;
```

C++

```
switch (A) {
   case 0:
      printf ("0");
      break;
   case -9:case -8:case -7:case -6:
   case -5:case -4:case -3:case -2:
   case -1:case 1:case 2:case 3:
   case 4:case 5:case 6:case 7:
   case 8:case 9:
      printf ("digit");
      break;
   default:
      printf ("other");
}
```

```
Put_Line (
   (case A is
    when 0 =>
       "zero"
    when -9 .. -1 | 1 .. 9 =>
       "digit"
    when others =>
       "other"));
```

同等の表現はC++にはありません

Case文の規則

- 式の型が必要とする全ての値を，記述する必要があります

```
   V : Integer := …;
begin
   case V is
      when 0 =>
         Put_Line (0);
   end case; -- NOK!
```

- 値は，ユニークでなければなりません

```
   V : Integer := …;
begin
   case V is
      when 0 =>
         Put_Line ("0");
      when Integer'First .. 0 => -- NOK!
         Put_Line ("Negative");
      when others =>
         null;
   end case;
```

ループ文

- ループは，「loop」と「end loop」ブロックによって囲まれます

```
loop
   Put_Line ("Hello");

   delay 1.0;
end loop;
```

- ループは，while文に脱出条件を記述することによって制御することができます

```
while A < 10 loop
   Put_Line ("Hello");
   A := A + 1;
end loop;
```

exit 文

- いつでも exit 文によってループを抜け出すことができます

```
loop
   A := A + 1;

   if A > 10 then
      exit;
   end if;

   B := B + 1;
end loop;
```

```
loop
   A := A + 1;
   exit when A > 10;
   B := B + 1;
end loop;
```

- ネスト化されたループでは，名前を用いて脱出位置を定めることができます

```
Loop_1 : loop
   A := A + 1;

   Loop_2 : loop
      B := B + 1;
      exit Loop_1 when B > 10;
   end loop;
end loop;
```

For .. Loop文

- 範囲内で繰り返しを行うことができます

```
for X in 1 .. 10 loop
   Put_Line ("Hello");
end loop;
```

- 範囲は昇順で与えます

```
for X in 10 .. 0 loop
   Put_Line ("Hello"); -- not called
end loop;
```

- 繰り返しは必ず次の値になります．逆順が指定された場合は，前の値になります

```
for X in reverse 1 .. 10 loop
   Put_Line ("Hello");
end loop;
```

「for loop」 制御変数

- 「for loop」中で宣言される変数は，明示的に型を持たなければなりません

```
for X in Integer range 1 .. 10 loop
   Put_Line ("Hello");
end loop;
```

- 全ての離散型で可能です．範囲が指定されていない場合は，型が持つ全ての範囲となります

```
for X in Character loop
   Put_Line (X);
end loop;
```

- 制御変数は，ループ中では定数です

```
for X in 1 .. 10 loop
   X := X + 1;
end loop;
```

「for loop」 範囲の評価

- ループ範囲の評価は，ループに入ったときに一度だけ行われます

```
for X in A .. B loop
   B := B + 1; -- no effect on the loop
end loop;
```

宣言ブロック

- 全ての文列において，（新しい変数の宣言や例外ハンドラなどに対して）再度宣言ブロックを用いること，副式列を用いることは有益です

- これは，新しい宣言ブロックによって行うことができます

```
declare
   A : Integer;
begin
   A := 0;
end;
```

ヌル（null）文

- 空の文列は，規則に反しています．しかし，．．．

- ．．．何もしないことを示す文を用いることは可能です
  それがヌル（null）文です．

```
declare
   A : Integer;
begin
   null;
end;
```

Quiz

正しいですか　(1/10)

はい
（チェックアイコンをクリックする）

いいえ
（エラーの場所をクリックする）

```
if A == 0 then
   Put_Line ("A is 0");
end if;
```

正しいですか　(1/10)

いいえ

```
if A == 0 then
   Put_Line ("A is 0");
end if;
```

コンパイルエラー
Adaでは，「＝」によって，同値であることを示します

正しいですか (2/10)

はい
（チェックアイコンをクリックする）

いいえ
（エラーの場所をクリックする）

```
if A := 0 then
   Put_Line ("A has been assigned to 0");
end if;
```

正しいですか (2/10)

いいえ

```
if A := 0 then
   Put_Line ("A has been assigned to 0");
end if;
```

コンパイルエラー
「:=」は，（比較）演算子ではありません．条件中で用いることはできません

正しいですか (3/10)

はい
（チェックアイコンをクリックする）

いいえ
（エラーの場所をクリックする）

```
   A : Integer := Integer'Value (Get_Line);
begin
   case A is
      when 1 .. 9 =>
         Put_Line ("Simple digit");
      when 10 .. Integer'Last =>
         Put_Line ("Long positive");
      when Integer'First .. -1 =>
         Put_Line ("Negative");
   end case;
```

正しいですか (3/10)

いいえ

```
   A : Integer := Integer'Value (Get_Line);
begin
   case A is
      when 1 .. 9 =>
         Put_Line ("Simple digit");
      when 10 .. Integer'Last =>
         Put_Line ("Long positive");
      when Integer'First .. -1 =>
         Put_Line ("Negative");
   end case;
```

コンパイルエラー
0が抜けています

正しいですか (4/10)

はい
（チェックアイコンをクリックする）

いいえ
（エラーの場所をクリックする）

```
 A : Integer := Integer'Value (Get_Line);
begin
   case A is
      when 1 .. Integer'Last=>
         Put_Line ("Positive");
      when 0 .. Integer'Last =>
         Put_Line ("Natural");
      when others =>
         Put_Line ("Other");
   end case;
```

正しいですか (4/10)

いいえ

```
   A : Integer := Integer'Value (Get_Line);
begin
   case A is
      when 1 .. Integer'Last=>
         Put_Line ("Positive");
      when 0 .. Integer'Last =>
         Put_Line ("Natural");
      when others =>
         Put_Line ("Other");
   end case;
```

コンパイルエラー
1..Integer'Last と 0..Integer'Last が重複しています

正しいですか (5/10)

はい
（チェックアイコンをクリックする）

いいえ
（エラーの場所をクリックする）

```
   A : Float := 10.0;
begin
   case A is
      when 1.0 .. Float'Last =>
         Put_Line ("Positive");
      when Float'First .. -1.0 =>
         Put_Line ("Negative");
      when others =>
         Put_Line ("Other");
   end case;
```

正しいですか (5/10)

いいえ

コンパイルエラー
評価される値は，離散的でなければなりません

```
   A : Float := 10.0;
begin
   case A is
      when 1.0 .. Float'Last =>
         Put_Line ("Positive");
      when Float'First .. -1.0 =>
         Put_Line ("Negative");
      when others =>
         Put_Line ("Other");
   end case;
```

正しいですか (6/10)

はい
（チェックアイコンをクリックする）

いいえ
（エラーの場所をクリックする）

```
for I in 0 .. 10 loop
   I := 10;
end loop;
```

正しいですか (6/10)

いいえ

```
for I in 0 .. 10 loop
   I := 10;
end loop;
```

コンパイルエラー
I はループ中では定数です．代入は禁止されています

出力は？ (7/10)

```
for I in 10 .. 0 loop
   Put_Line (Integer'Image (I));
end loop;
```

出力は？ (7/10)

```
for I in 10 .. 0 loop
   Put_Line (Integer'Image (I));
end loop;
```

何もおきません
ループの範囲は空になります.
逆順にするためには reverse 0..10 が必要です

正しいですか (8/10)

はい
（チェックアイコンをクリックする）

いいえ
（エラーの場所をクリックする）

```
if A != 0 then
   Put_Line ("A is not 0");
end if;
```

正しいですか　(8/10)

いいえ

コンパイルエラー
Adaでは，不等号は，「/=」と記述します

```
if A != 0 then
   Put_Line ("A is not 0");
end if;
```

正しいですか (9/10)

はい
（チェックアイコンをクリックする）

いいえ
（エラーの場所をクリックする）

```
   I : Natural;
begin
   for I in 0 .. 10 loop
      null;
   end loop;
```

正しいですか (9/10)

はい

```
   I : Natural;
begin
   for I in 0 .. 10 loop
      null;
   end loop;
```

正しい．しかし，I の宣言は冗長です

出力は？ (10/10)

```
   X : Integer := 1;
begin
 for I in 1 .. X loop
     X := 10;
     Put_Line ("A");
 end loop;
```

出力は？ (10/10)

```
   X : Integer := 1;
begin
 for I in 1 .. X loop
     X := 10;
     Put_Line ("A");
 end loop;
```

一度だけ ″A″ が出力されます．ループに入る前に範囲が評価されます

university.adacore.com