基本型

Presented by Quentin Ochem

university.adacore.com

幾つかの構文上の注意

識別子

- Ada の識別子は大文字と小文字の区別をしません
  - HELLO = hello = HellO
- 文字で開始します
- 文字ないしは数字で終了します
- 連続しないアンダースコアを含む場合があります

- 以下で妥当なのはどれでしょうか
  - Something
  - My__Id
  - _Hello
  - A_67_9
  - _CONSTANT
  - 09_A_2
  - YOP_

Identifiers

- Ada の識別子は大文字と小文字の区別をしません
  - HELLO = hello = HellO
- 文字で開始します
- 文字ないしは数字で終了します
- 連続しないアンダースコアを含む場合があります

- 以下で妥当なのはどれでしょうか
  - Something
  - ✗ My__Id
  - ✗ _Hello
  - A_67_9
  - ✗ _CONSTANT
  - ✗ 09_A_2
  - ✗ YOP_

コメント

- Adaでは，-- を用いて行コメントを付けることができます

```
--  This is an Ada comment
```

```
// This is a C++ comment
```

- ブロックコメントはありません (/* */)

数字

- 数字では，アンダースコアを使用することができます
  - 1_000_000 = 1000000
- 基数を用いて数字を表現することができます(2から16まで)
  - 10#255# = 2#1111_1111# = 8#377# = 16#FF#
- 浮動小数点リテラルは，必ず小数点（″.″）を持つ必要があります
  - 小数点の前後には数字が必要です
  - 1.0 /= 1

変数宣言

- 一つないしは複数の名前が定義されます．　″:″ 記号のあとに，型の名
  場合によっては更に初期値が続きます

```
A : Integer;
B : Integer := 5;
C : constant Integer := 78;
D, E : Integer := F (5);
```

```
int A;
int B = 5;
const int C = 78;
int d = F (5), e = F(5);
```

- 実行時準備処理（elaboration）は，順次行われます

```
A : Integer := 5;
B : Integer := A;
C : Integer := D; -- COMPILATION ERROR
D : Integer := 0;
```

- 変数毎に個別に初期化されます

```
A, B : Float := Compute_New_Random;
--  This is equivalent to:
A : Float := Compute_New_Random;
B : Float := Compute_New_Random;
```

- 宣言における“:= “ は，初期化であって，代入ではありません　（特
  別なプロパティについて後述します）

単純なスカラー型

Adaの強い型付け

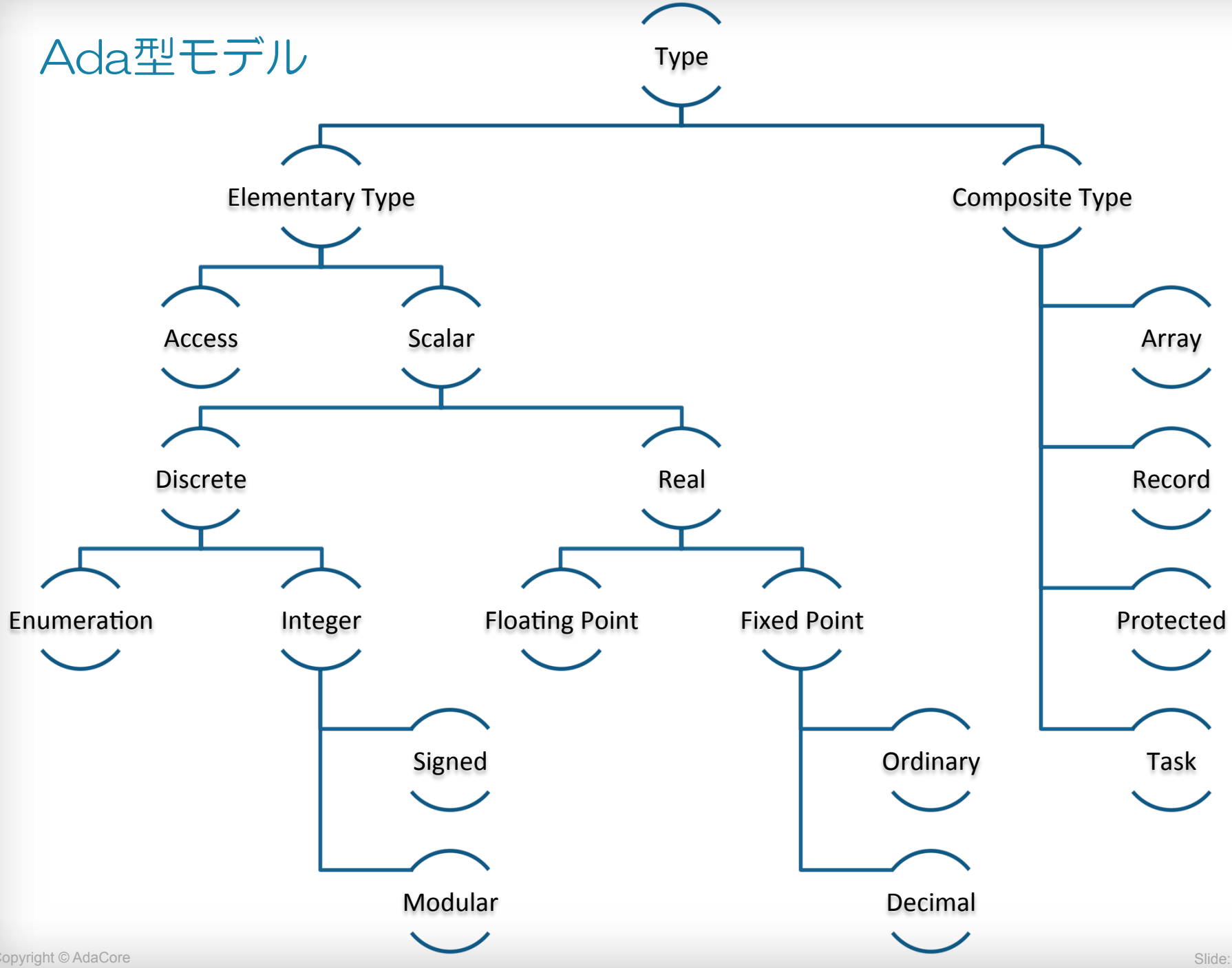
- 型は，Adaモデルの基本です
- 意味 ≠ 表現
- Adaの型は名前を持ちます
- プロパティ（範囲，属性…）や操作と関連づけられています

```
A : Integer := 10 * Integer (0.9);
A : Integer := Integer
   (Float (10) * 0.9);
```

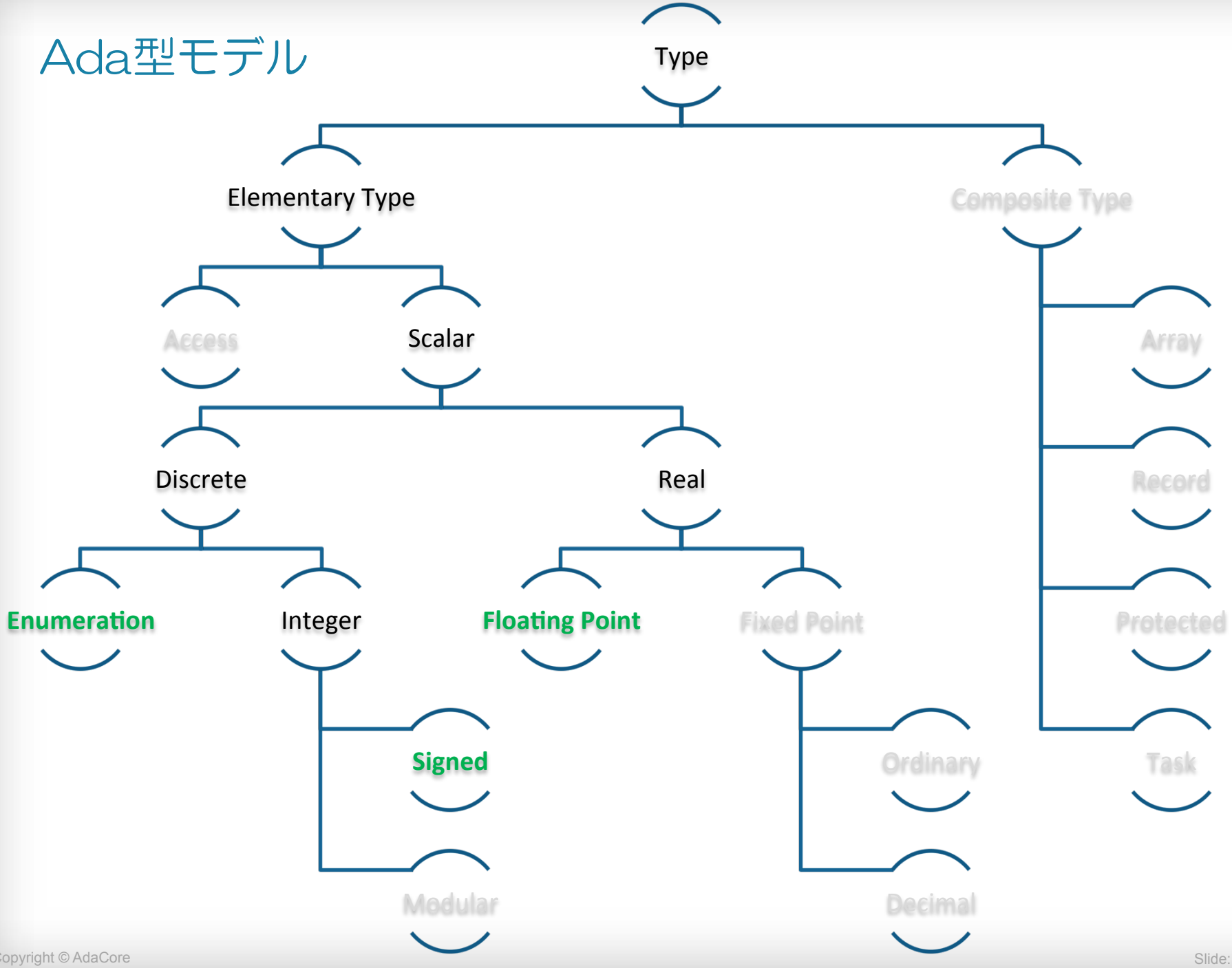
```
int A = 10 * 0.9
```

- 不整合がある場合，コンパイラは警告を出力します

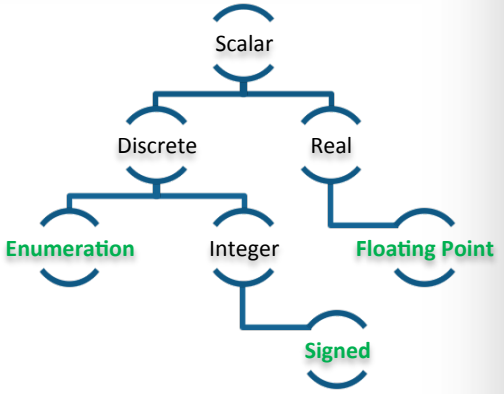
Ada型モデル



Ada型モデル



スカラー型・離散型・実数型



- スカラー型は単一の値を持ちます
  - 整数・浮動小数点・列挙は全てスカラーです
- スカラー型は，「離散」型（有限の値をとる）と「連続」型に分かれます
- 幾つかのスカラー型は，数値演算に関係しています

標準型

- 符号付き整数型
  - Short_Integer, Integer, Long_Integer, Long_Long_Integer
- 列挙型
  - 文字, ブール
- 浮動小数点型
  - Short_Float, Float, Long_Float, Long_Long_Float

型のカスタム宣言

- 整数型を値の範囲と共に定義します

```
type Score is range 0 .. 20;
type Percentage is range -100 .. 100;
```

- 列挙型を値のリストとともに定義します

```
type Color is (Red, Green, Blue, Yellow, Black);
type Ternary is (True, False, Unknown);
```

- 浮動小数点型は，最小の有効桁数とともに定義します

```
type Distance is digits 10;
type Temperature is digits 5 range -273.15 .. 1_000_000.0;
```

⛔ 範囲指定で，浮動小数点型の（演算実行の）性能が低下します

既存の型から（新しい）型を作る

- 既存の型から，新しい型を作ることが可能です

```
type Math_Score is new Score;
```

- 新しい型は，値の範囲を制限することができます

```
type Math_Score is new Score range 0 .. 10;
type Primary_Color is new Color range Red .. Blue;
```

型変換

- 型を他の型に変換できる場合があります
  - 両者が同一の構造を持っている場合
  - 一方の型が他から派生している場合

- 変換は明示的に行う必要があります

```
V1 : Float := 0.0;
V2 : Integer := Integer (V1);
```

- 変換によって検証が必要になる場合があります

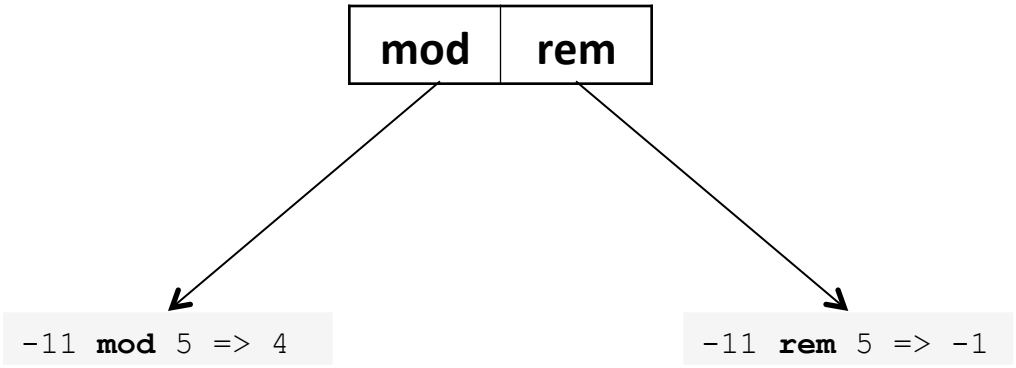
```
type T1 is range 0 .. 10;
type T2 is range 1 .. 10;

V1 : T1 := 0;
V2 : T2 := T2 (V1); -- Run-time error!
```

Adaにおける演算子

- 符号付きと浮動小数点型に共通の演算子

=	/=	<	<=	>	>=	+	-	*	/	abs	**

- 符号付き型で注意が必要な演算子



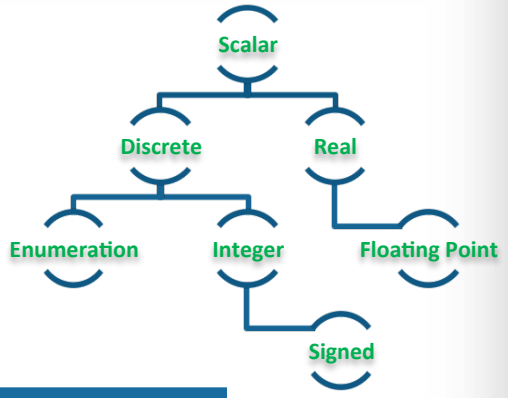
属性

- 属性は，Adaエンティティの標準的なプロパティです
- 「 ’」でアクセスすることができます

```
S : String := Integer'Image (42);
```

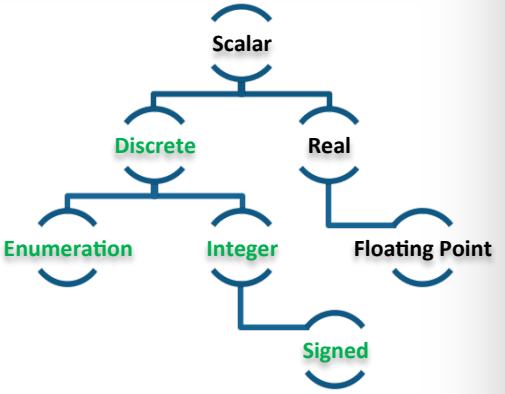
- 型が属するカテゴリによって，異なる組の属性を利用できます

スカラー型に特有の属性例



属性名	説明
First	型の最初の値を返します
Last	型の最後の値を返します
Image (X)	値を相当する文字列（String）に変換します
Value (X)	文字列（String）を相当する値に変換します
Min (X, Y)	二つの値の大きくない方を返します
Max (X, Y)	二つの値の小さくない方を返します
Pred (X)	一つ前の値を返します
Succ (X)	次の値を返します
Range	T’First ..T’Lastと同じです

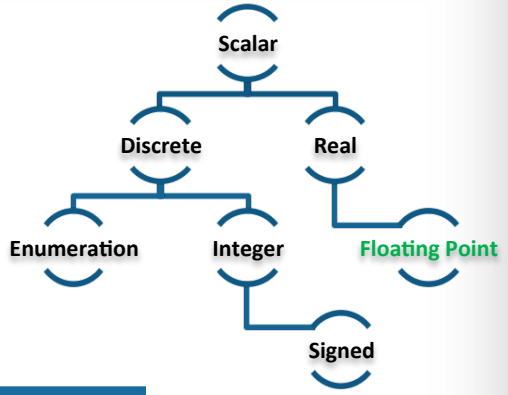
離散型に特有の属性例



属性名	説明
Pos (X)	型における位置を返します
Val (X)	位置の値を返します

```
V : Character := Character'Val (0);
W : Next_Character := Character'Val (Character'Pos (V) + 1);
```

浮動小数点型に特有の属性例



属性名	説明
Ceiling (X)	X以降の最も小さい整数値を返す
Floor (X)	X以前の最も大きな整数値を返す
Truncation (X)	0に近くなるよう小数部を切り捨てます
Rounding (X)	最も近い整数にまとめる
Remainder (X, Y)	ユークリッド除法における余りを返す

浮動小数から整数への変換は，Roundingによる丸めです．
Truncationによる小数部を切り捨てではありません！

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）

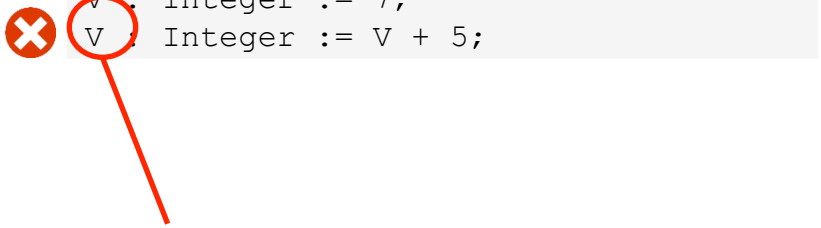
いいえ
（エラーの場所をクリックする）

```
V : Integer := 7;
V : Integer := V + 5;
```

正しいですか　(1/10)

いいえ

```
V : Integer := 7;
V : Integer := V + 5;
```



Vは，既に宣言されているのでコンパイルエラーになります

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）

```
type N is range -2 ** 256 .. 2 ** 256;
```

正しいですか (2/10)

いいえ

```
type N is range -2 ** 256 .. 2 ** 256;
```

この演算は，現在の多くのシステムでは大きすぎる値となります．従って，コンパイラは表現することができず，エラーを発行します

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）

```
V : Float := 5.0;
```

正しいですか　(3/10)

はい

```
V : Float := 5.0;
```

エラーはありません

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）

```
ClassRoom : constant Natural := 5;
Next_ClassRoom : Natural := classroom + 1;
```

正しいですか　(4/10)

はい

```
ClassRoom : constant Natural := 5;
Next_ClassRoom : Natural := classroom + 1;
```

誤りはありません
Adaは大文字／小文字を区別しません

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）

```
type T1 is new Integer range -10 .. 10;

type T2 is new T1 range -100 .. 100;
```

正しいですか (5/10)

いいえ

```
type T1 is new Integer range -10 .. 10;

type T2 is new T1 range -100 .. 100;
```

範囲を拡張することはできません
狭くすることができるだけです

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）

```
X : Float := Float'Succ (0.9);
```

正しいですか　(6/10)

はい

```
X : Float := Float'Succ (0.9);
```

合っています
浮動小数点でも，Succ を利用可能です．0.9 に最も近い次の浮動小数点の値を返します

このコードの出力は何でしょう？ (7/10)

```
   F   : Float   := 7.6;
   Div : Integer := 10;
begin
   F := Float (Integer (F) / Div);
   Put_Line (Float'Image (F));
```

このコードの出力は何でしょう？ (7/10)

```
   F   : Float   := 7.6;
   Div : Integer := 10;
begin
   F := Float (Integer (F) / Div);
   Put_Line (Float'Image (F));
```

0.0

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）

```
type T is (A, B, C);

V1 : T := T'Val ("A");
V2 : T := T'Value (2);
```

正しいですか (8/10)

いいえ

```
type T is (A, B, C);

V1 : T := T'Val ("A");
V2 : T := T'Value (2);
```

コンパイルエラーになります．
T'Valは，位置から値を返します
T'Valueは，文字列から値を返します

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）

```
type T is (A, B, C);

V1 : T := T'Value ("A");
V2 : T := T'Value ("a");
V3 : T := T'Value (" a ");
```

正しいですか (9/10)

はい

```
type T is (A, B, C);

V1 : T := T'Value ("A");
V2 : T := T'Value ("a");
V3 : T := T'Value (" a ");
```

エラーはありません
変換では，大文字／小文字を区別しません
また，自動的にスペースをトリムします

正しいですか (10/10)

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）

```
type T is range 1 .. 0;
V : T;
```

正しいですか　(10/10)

はい

```
type T is range 1 .. 0;
V : T;
```

エラーはありません
Tは空の範囲を持ちます．役に立つ場面があります

university.adacore.com