Gem #86: Тест 1 по языку Ada - Базовые типы

Автор: Quentin Ochem, AdaCore

Краткое содержание: Данный Gem - один из периодически публикуемых тестов по языку Ада, состоящих из вопросов по свойствам и функциям языка, взятых из курсов AdaCore.

Давайте начнём…

Вот десять коротких вопросов, связанных со скалярными типами и выражениями Ada. Попробуйте ответить на них без использования компилятора. Проверьте, насколько вы хороший эксперт по языку программирования Ada!

1. Есть ли ошибка компиляции?

```
V : Float := 10;
```

2. Каков результат этого кода?

```
   type Float_1 is digits 5;
   type Float_2 is digits 7;
   V1 : Float_1 := 10.0E10;
   W1 : Float_1 := V1 + 1.0;
   V2 : Float_2 := 10.0E10;
   W2 : Float_2 := V2 + 1.0;
begin
   Put_Line (Boolean'Image (V1 = W1));
   Put_Line (Boolean'Image (V2 = W2));
end;
```

3. Существует ли ошибка компиляции или выполнения?

```
   type Digit is mod 10;
   V1 : Digit := 10;
   V2 : Digit := 9 + 1;
```

4. Каков результат этого кода?

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

5. Существует ли ошибка времени выполнения?

```
   type Small_Int is range 1 .. 10;
   V : Small_Int := 9;
   W : Small_Int := 2;
begin
   V := V + W - 1;
end;
```

6. Существует ли ошибка времени выполнения?

```
   type Small_Int is range 1 .. 10;
   V : Small_Int := 9;
```

```
   W : Small_Int := 2;
begin
   V := Small_Int (V + W) - 1;
end;
```

7. Есть ли компиляция или ошибка во время выполнения?

```
   C1 : constant := 2 ** 1024;
   C2 : constant := 2 ** 1024 + 10;
   C3 : constant := C1 - C2;
   V  : Integer  := C1 - C2;
```

8. Есть ли ошибка компиляции?

```
   type T is (A, B, C);
   V1 : T := T'Val ("A");
   V2 : T := T'Value (2);
```

9. Есть ли ошибка во время выполнения?

```
   type T is (A, B, C);
   V1 : T := T'Value ("A");
   V2 : T := T'Value ("a");
   V3 : T := T'Value (" a ");
```

10. Есть ли ошибка компиляции?

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

Ответы:

1. Ошибка компиляции

V объявляется с типом Float, и он должен быть инициализирован с реальным значением. Литерал 10 имеет целочисленное значение, поэтому компилятор Ada отклоняет его. Замена его на 10.0 исправляет ошибку.

2. Выводит TRUE и FALSE (зависит от реализации)

Типы Float_1 и Float_2 были указаны с минимальным количеством десятичных цифр, но компилятор может выбирать представления с большей точностью. В этом примере GNAT выбирает 32-разрядный Тип с 6 цифрами точности для Float_1 и 64-разрядный Тип с 15 цифрами точности для Float_2. Для Float_1 операция V1: = V1 + 1.0 не изменяет значение V1, так как, учитывая точность, приращение слишком мало по сравнению с исходным значением. Для Float_2 представление имеет достаточную дополнительную точность, что приращение приводит к различному значению представления.

3. Ошибка компиляции

Тип Digit имеет модуль 10, поэтому его значения варьируются от 0 до 9. Значение 10 находится вне этого диапазона, поэтому компилятор отклоняет "X: = 10". Интересно, что "X: = 9 + 1" является законным. "+"Используемое здесь Тип модульный оператор добавления, который обернёт вокруг, давая результат 0.

4. 0.0

Аргумент преобразования в Float, Integer (F) / Div, преобразует значение F (7.6) в Тип Integer, который округляется до 8. Потом 8 делим на 10, используя Целочисленное деление, в результате 0. Преобразование этого обратно в Float дает 0.0. Это типичная ошибка, при которой преобразование не выполняется для соответствующих переменных. Предполагая, что требуется разделение с плавающей запятой, выражение должно быть записано как F / Float (Div), что дает желаемый результат 0.76.

5. Ошибка времени выполнения отсутствует

Хотя V + W равен 11, что превышает максимальное значение Small_Int, предопределенные арифметические операторы работают со значениями базового типа. В этом случае компилятор, скорее всего, выберет Тип с гораздо большим диапазоном, чем у первого подтипа Small_Int. Таким образом, выражение будет правильно вычислено как 10, что удовлетворяет ограничению целевой переменной.

6. Ошибка выполнения

В этом случае V + W вычисляется и преобразуется в Small_Int. Хотя выражение уже имеет правильный Тип (Small_int ' Base), явное преобразование проверяет, что значение находится в диапазоне целевого подтипа (Small_int'Range). Здесь сумма выходит за пределы диапазона Small_Int, поэтому во время выполнения возникает исключение. Обратите внимание, что более подходящим способом записи такого выражения (когда это имеет смысл) является использование квалифицированного выражения (Small_Int'(V + W)) вместо преобразования, так как операнд уже имеет правильный Тип.

7. Нет ошибок

Значение 2 * * 1024 намного больше, чем целочисленные значения, которые могут быть представлены во время выполнения. В частности, он превышает верхнюю границу наибольшего целочисленного типа, поддерживаемого GNAT (64-разрядное целое число). Но C1-это именованное число, а не типизированная константа. Компилятор использует внутреннее представление с неограниченной точностью для вычисления значений именованных чисел и статических выражений во время компиляции без переполнения. Потенциально большое промежуточное представление не сохраняется в объектном коде программы. В данном конкретном примере для присвоения V вычитание вычисляется компилятором, а V присваивается значение -10.

8. Ошибка компиляции

T'Val возвращает значение перечисления, соответствующее аргументу целочисленного типа. С другой стороны, T ' Value принимает строковый аргумент и возвращает соответствующее значение типа (если таковое имеется). В этом примере, аргументы имеют неверный Тип.

9. Ошибка времени выполнения отсутствует

Преобразования из строкового представления в скалярное не зависят от регистра, а окружающие пробелы игнорируются.

10. Ошибка компиляции отсутствует

Тип T имеет пустой диапазон, поэтому допустимых представимых значений нет. Хотя это странный Тип, чтобы определить, это совершенно законно. Конечно, любая попытка присвоить переменной определённое целочисленное значение вызовет исключение.

Связанный со статьёй текст программы

Attached Files отсутствуют

Файлы примеров Ada Gems распространяются AdaCore и могут быть использованы или изменены для любых целей без ограничений.

Об авторе

Quentin Ochem имеет опыт разработки программного обеспечения, специализирующийся на разработке программного обеспечения для критически важных приложений. Он имеет более чем 10-летний опыт разработки Ada. Сегодня он работает техническим менеджером AdaCore по проектам, связанным с авионикой, железнодорожной, космической и оборонной отраслями. Он также обучает стандарту DO-178B авионики в университете EPITA в Париже.

Last Updated: 10/13/2017
Posted on: 5/17/2010

Обсуждение…