Gem #62: Конструкторы C++ и Ada 2005.

Автор: Javier Miranda, Arnaud Charlet, AdaCore

Краткое содержание: Gem Ada #62 - В предыдущем Gem было продемонстрировано, как можно просто осуществлять взаимодействие Ada и конструкторов C ++.

В данном Gem будет показано, как создавать связи Ada и конструкторов C ++ на более современном уровне.

Давайте начнем…

На основе примера, рассмотренного в Gem #61, давайте выведем импортированный класс C ++ на стороне Ada. Например:

```
type DT is limited new Root with record
    C_Value : Natural := 2009;
 end record;
```

В этом случае компоненты `DT`, унаследованные со стороны C ++, должны быть инициализированы конструктором C ++, а дополнительные компоненты Ada типа `DT` инициализируются GNAT. Инициализация такого объекта выполняется либо по умолчанию, либо посредством функции, возвращающей агрегат типа `DT`, или с помощью агрегата расширения:

```
 Obj5 : DT;
 Obj6 : DT := Function_Returning_DT (50);
 Obj7 : DT := (New_Root (30, 40) with C_Value => 50);
```

Объявление `Obj5` вызывает конструкторы по умолчанию: конструктор по умолчанию C ++ родительского типа заботится об инициализации компонентов, унаследованных от `Root`, и GNAT позаботится об инициализации по умолчанию дополнительных компонентов Ada типа `DT` (то есть `C_Value` инициализируется до значения 2009). Порядок вызова конструкторов согласуется с порядком разработки, требуемым Ada и C ++, что означает, что конструктор родительского типа всегда вызывается перед конструктором производного типа.

Теперь рассмотрим запись, в которой есть компоненты, тип которых импортируется из C ++. Например:

```
type Rec1 is limited record
    Data1 : Root := New_Root (10);
    Value : Natural := 1000;
 end record;

 type Rec2 (D : Integer := 20) is limited record
    Rec   : Rec1;
    Data2 : Root := New_Root (D, 30);
 end record;
```

Инициализация объекта типа `Rec2` вызовет конструкторы C ++, не заданные по умолчанию для импортированных компонентов. Например:

```
Obj8 : Rec2 (40);
```

Используя Ada 2005, мы можем использовать ограниченные агрегаты для инициализации объекта, вызывающего конструкторы C++, которые отличаются от указанных в объявлениях типов. Например:

```
Obj9 : Rec2 := (Rec => (Data1 => New_Root (15, 16),
                        others => <>),
                others => <>);
```

Вышеупомянутое объявление использует Ada 2005 ограниченный агрегат, чтобы инициализировать `Obj9`, и конструктор C ++, у которого есть два целочисленных параметра, вызван, чтобы инициализировать компонент `Data1` вместо конструктора, определённого в объявлении типа `Rec1`. В Ada 2005 поле в агрегате указывает, что неуказанные компоненты инициализированы, используя выражение (если таковые имеются) доступное в составляющем объявлении. То есть, в этом случае дискриминант `D` инициализируется значением `20`, `Value` инициализируется значением `1000`, а нестандартный конструктор C ++, обрабатывающий два целых числа, заботится об инициализации компонента `Data2` со значениями `20` и `30`.

В Ada 2005 мы можем использовать расширенный оператор возврата, чтобы создать Ada, эквивалентную из C ++ конструктор не по умолчанию. Например:

```
function New_Rec2 (V : Integer) return Rec2 is
 begin
    return Obj : Rec2 := (Rec => (Data1  => New_Root (V, 20),
                                  others => <>),
                          others => <>) do
       --  Further actions required for construction of
       --  objects of type Rec2
       ...
    end record;
 end New_Rec2;
```

В этом примере конструкция расширенной инструкции `return` используется для создания возвращаемого объекта, компоненты которого инициализируются с помощью ограниченного статистического выражения. Любые дальнейшие действия, связанные с конструктором, могут быть указаны в части утверждения расширенной инструкции `return`.

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

Данные об авторе отсутствуют.

Last Updated: 10/13/2017
Posted on: 4/6/2009

Обсуждение…