Gem #3: Лимитированные типы в Ada 2005 – Функции конструктора

Автор: Bob Duff, AdaCore

Краткое содержание: Gem Ada #3 – Можно использовать функции конструктора для создания объектов лимитированных типов. Результаты вызовы подобных функций реализуется «на месте» в самих создаваемых объектах.

Давайте начнем…

Учитывая, что Ada 2005 позволяет встраивание агрегатов, то есть сборку на месте для лимитированных типов, очевидный следующий шаг должен позволить таким агрегатам быть обернутыми в абстракцию – а именно, возвратить их из функций. В конце концов, интересуемые типы обычно приватные, и нам нужен некоторый способ для клиентов, чтобы создать и инициализировать объекты.

```
package P is
   type T (<>) is limited private;
   function Make_T (Name : String) return T; – constructor function

private
   type T is limited
      record
         Name : Unbounded_String;
         My_Task : Some_Task_Type;
         My_Prot : Some_Protected_Type;
      end record;
end P;

package body P is
   function Make_T (Name : String) return T is
   begin
      return (Name => To_Unbounded_String (Name), others => <>);
   end Make_T;
end P;
```

В Ada 95 функции-конструкторы (т.е. функции, которые создают новые объекты и возвращают их) не допускаются для ограниченных типов. Ada 2005 позволяет полностью использовать общие функции конструктора. Учитывая вышеизложенное, клиенты могут сказать:

```
My_T : T := Make_T (Name => "Bartholomew Cubbins");
```

Что касается агрегатов, в результате Make_T построен на месте (то есть в `My_T`), а не создается, а затем копируются в `My_T`. Добавляя еще один уровень вызова функции, мы можем сделать:

```
function Make_Rumplestiltskin return T is
begin
    return Make_T (Name => “Rumplestiltskin”);
end Make_Rumplestiltskin;

Rumplestiltskin_Is_My_Name : constant T := Make_Rumplestiltskin;
```

Это может помочь понять модель реализации: в этом случае `Rumplestiltskin_Is_My_Name` распределяется обычным способом (в стеке,

предполагая, что он объявлен локальным для некоторой подпрограммы). Его адрес передается как дополнительный неявный параметр для Make_Rumplestiltskin, который затем передает тот же адрес в Make_T, который затем создает агрегат на месте по этому адресу. Ограниченные объекты никогда не должны копироваться! В этом случае Make_T инициализирует компонент Name и создает компоненты My_Task и My_Prot, все непосредственно в Rumplestiltskin_Is_My_Name.

Обратите внимание, что Rumplestiltskin_Is_My_Name является константой. В Ada 95 невозможно создать ограниченный объект как константу, потому что нет способа инициализировать его.

Как и в Ada 95, «(<>)» типа T означает, что он имеет «неизвестные дискриминаторы» с точки зрения клиента. Это трюк, который запрещает клиентам создавать объекты, инициализированные по умолчанию (то есть «X: T;» является незаконным). Таким образом, клиенты должны вызывать Make_T всякий раз, когда создается объект типа T, предоставляя посредством пакета P полный контроль над инициализацией объектов.

В идеале ограниченные и неограниченные типы должны быть одинаковыми, за исключением существенного различия: вы не можете копировать ограниченные объекты. Предоставление функций и агрегатов для ограниченных типов в Ada 2005 приближает нас к этой цели. В некоторых языках есть специальная функция, называемая «конструктор». В Ada «конструктор» - это просто функция, которая создает новый объект. Кроме того, что в Ada 95, это работает только для неограниченных типов. Для ограниченных типов единственным способом «построить» при объявлении является использование значений по умолчанию, что ограничивает вас одним конструктором. И единственный способ передать параметры этой конструкции через дискриминаторы. В Ada 2005 мы можем сказать:

```
This_Set : Set := Empty_Set;
That_Set : Set := Singleton_Set (Element => 42);
```

Независимо от того, является или не является Set лимитируемым типом. «This_Set: Set: = Empty_Set;» мне кажется более понятным, чем:

```
This_Set : Set;
```

что может означать “по умолчанию-инициализировать пустое множество” или может означать “оставить его неинициализированным, и мы будем инициализировать его в дальнейшем”.

Обсуждение…

1. 30 мая 2007 года, 16:03

Пол Ф. Пирсон сказал:

Вау. Похоже, что много новых изменений в Ada 2005 касается Ограниченных типов (Limited Types), что делает их более удобными. Я впечатлен.

В Ada 95 я никогда особо не применял Limited Types в первую очередь из-за привычки культуры в разработке, по-видимому, предвидя проблемы, которые исправляет Ada 2005.

Еще раз спасибо.

2. 1 июня 2007 года в 16 час. 06 мин.

Франко Гасперони сказал:

Да Пол, ограниченные типы чрезвычайно полезны сейчас. Используя Ограниченные типы и фразеологизм `(<>)`, теперь вы можете принудительно использовать конструкторы, как объяснил Боб в этом GEM #3. Кроме того, вы делаете тип ограниченным и абстрактным, вы можете вызвать диспетчеризацию, если конструкторы возвращают `'Class`.