Gem #33: Проверка доступности (Часть I: Ada95)

Автор: Ramón Fernández-Marina, AdaCore

Краткое содержание: Gem Ada #33 - Наличие недействительных указателей (указателей на несуществующие объекты) в программе может привести к катастрофическим последствиям. В языке Ada применяется набор «правил доступности», которые позволяют разработчику избежать подобных проблем, таким образом повышая отлаженность программного обеспечения.

Давайте начнем…

Ada - это блок-структурированный язык, что означает, что программист может встраивать блоки кода внутри других блоков. В конце блока все объекты, объявленные внутри него, выходят за рамки так называемой «области видимости» и автоматически уничтожаются, то есть они больше не существуют, поэтому язык запрещает указатели на объекты в блоках с более глубоким уровнем вложенности.

Чтобы предотвратить зависание ссылок, каждый объект связан с числом, называемым его «уровнем доступности», в соответствии с правилами доступности Ada. Когда определённые ссылки обрабатываются сущностью типа доступа `access` (язык Ada для указателя), уровень доступности объекта проверяется на уровне, разрешенном контекстом, чтобы не возникало никаких оборванных указателей.

Рассмотрим следующий пример:

```
procedure Static_Check is
   type Global is access all Integer;
   X : Global;

   procedure Init is
      Y : aliased Integer := 0;
   begin
      X := Y'Access; -- Illegal!
   end Init;

begin
   Init;
   ...
end Static_Check;
```

Присваивание является незаконным, потому что когда процедура Init заканчивается, объект Y больше не существует, таким образом делая x указатель на обрыв – так называемый «висячий указатель». Компилятор обнаружит эту ситуацию и отметит ошибку.

Красота правил доступности заключается в том, что большинство из них можно проверить и применить во время компиляции, просто используя статически известные уровни доступности.

Однако бывают случаи, когда невозможно статически определить уровень доступности, который будет иметь сущность-объект во время выполнения программы. В этих случаях компилятор вставит проверку времени выполнения, чтобы вызвать исключение, если может быть создан висячий указатель:

```
procedure Access_Params is
   type Integer_Access is access all Integer;
   Data : Integer_Access;

   procedure Init_Data (Value : access Integer) is
   begin
      Data := Integer_Access (Value);
      -- this conversion performs a dynamic accessibility check
   end;

   X : aliased Integer := 1;

begin
   Init_Data (X'Access); -- This is OK

   declare
      Y : aliased Integer := 2;
   begin
      Init_Data (Y'Access); --  Trouble!
   end;
   --  Y no longer exists!

   Process (Data);
end;
```

В приведённом выше примере мы не можем знать во время компиляции уровень доступности объекта, который будет передан в Init_Data, поэтому компилятор вставляет проверку времени выполнения, чтобы убедиться, что присвоение "Data: = ..." не вызывает оборванную ссылку («висячий указатель») – и вызвать исключение, если бы это было так.

Таким образом, когда дело доходит до «висячего указателя», компилятор языка Ada делает всё для Вас, чтобы Вам было очень трудно «стрелять себе в ногу»

Связанный исходный код

Attached Files отсутствуют

Об авторе

Рамон Фернандес - старший инженер-программист (a senior software engineer) AdaCore. С момента прихода в компанию в 2001 году он участвовал в самых разных областях, включая поддержку клиентов, GNAT Pro для разработчиков и разработчиков, системы реального времени, обеспечение качества, встроенные системы, производственную инфраструктуру и управление ИТ. Рамон имеет степень магистра компьютерных наук в Нью-Йоркском университете и степень электроники в области телекоммуникаций от ETSIT-UPM (технический университет Мадрида, Испания).

Last Updated: 10/13/2017
Posted on: 4/28/2008

Обсуждение…