Gem #24: Null считается вредным. Предотвращение значение null, когда подобное нежелательно (Часть 2 – Эффективность)

Автор: Bob Duff, AdaCore

Краткое содержание: Gem Ada #24 - С помощью синтаксиса «not null» можно повысить эффективность программ за счет отсутствия необходимости в неявно определенных динамических проверках.

Давайте начнем…

На прошлой неделе в Gem #23 мы говорили о преимуществах использования “not null”. Вот еще один пример, сначала с null:

```
procedure Iterate
   (T : Table;
    Action : access procedure (X : not null Ref_Element)
                                       := null);
--  If Action is null, do nothing.
```

...и без null:

```
procedure Do_Nothing (X : not null Ref_Element) is null;
procedure Iterate
   (T : Table;
    Action : not null access procedure (X : not null Ref_Element)
                                      := Do_Nothing'Access);
```

Я предпочитаю стиль второй процедуры `Iterate`.

Процедура «not null access procedure» довольно проста для такого усложнения описателя, но это стоит того, и в любом случае требование совместимости для Ada 2005 требует, чтобы «not null» был явным, а не наоборот.

Еще одно преимущество «not null» комментариев - это эффективность создаваемого кода. Например:

```
procedure P (X : not null Ref_Element) is
begin
   X.all.Component := X.all.Component + 1;
end P;

procedure Q (X : not null Ref_Element) is
begin
   while ... loop
      P (X);
   end loop;
end Q;

procedure R is
begin
   Q (An_Element'Access);
end R;
```

Без “not null " сгенерированный код для процедуры P будет проверять, что X /= null, который может быть дорогостоящим для некоторых систем. Процедура P вызывается в цикле, поэтому эта проверка, скорее всего, произойдет много раз. С “not null”, проверка возможна в момент вызова. Проведение проверок в момент вызова обычно полезно,

потому что (1) проверка может быть извлечена из цикла оптимизатором, или (2) проверка может быть устранена вообще, как в примере выше, где компилятор знает, что An_Element'Acccess не может быть null.

Это аналогично ситуации в Ada 95 с другими проверками времени выполнения, такими как проверки границ массива:

```
type My_Index is range 1..10;
type My_Array is array (My_Index) of Integer;

procedure Process_Array (X : in out My_Array; Index : My_Index);
```

Если “X (Index)” используется внутри процедуры Process_Array, нет необходимости проверять, что Index находится в диапазоне допустимых значений, поскольку проверка этого выполнена ранее до вызова процедуры.

Связанный исходный код

Attached Files

```
Title: Null Considered Harmful (Part 2 -- Efficiency)

Author: Bob Duff, AdaCore

Abstract:
The "not null" syntax can make programs more efficient by removing the need
for implicit run-time checks.

In last week's gem, we talked about the documentation advantages of using
"not null".  Here's another example, first with null:

   procedure Iterate
     (T : Table;
      Action : access procedure (X : not null Ref_Element)
                                       := null);
   --  If Action is null, do nothing.

...and without null:

   procedure Do_Nothing (X : not null Ref_Element) is null;
   procedure Iterate
     (T : Table;
      Action : not null access procedure (X : not null Ref_Element)
                                       := Do_Nothing'Access);

I much prefer the style of the second Iterate.

The "not null access procedure" is quite a mouthful, but it's worthwhile, and
anyway, the compatibility requirement for Ada 2005 requires that the
"not null" be explicit, rather than the other way around.

Another advantage of "not null" over comments is for efficiency. For example:

   procedure P (X : not null Ref_Element) is
   begin
      X.all.Component := X.all.Component + 1;
   end P;

   procedure Q (X : not null Ref_Element) is
   begin
```

```
      while ... loop
         P (X);
      end loop;
   end Q;

   procedure R is
   begin
      Q (An_Element'Access);
   end R;
```

Without "not null", the generated code for P will do a check that X /= null, which may be costly on some systems. P is called in a loop, so this check will likely occur many times. With "not null", the check is pushed to the call site. Pushing checks to the call site is usually beneficial because (1) the check might be hoisted out of a loop by the optimizer, or (2) the check might be eliminated altogether, as in the example above, where the compiler knows that An_Element'Access cannot be null.

This is analogous to the situation in Ada 95 with other run-time checks, such as array bounds checks:

```
   type My_Index is range 1..10;
   type My_Array is array (My_Index) of Integer;

   procedure Process_Array (X : in out My_Array; Index : My_Index);
```

If "X (Index)" occurs inside Process_Array, there is no need to check that Index is in range, because the check is pushed to the caller.package Example is

```
   type Element is limited private;
   type Ref_Element is access all Element;

   type Table is limited private;

   Not_Found : constant Ref_Element := null;
   function Lookup (T : Table) return Ref_Element;
   --  Returns Not_Found if not found.

   Not_Found_2 : exception;
   function Lookup_2 (T : Table) return not null Ref_Element;
   --  Raises Not_Found_2 if not found.

   procedure P (X : not null Ref_Element);

   procedure Q (X : not null Ref_Element);

   procedure Iterate
     (T : Table;
      Action : access procedure (X : not null Ref_Element)
                                    := null);
   --  If Action is null, do nothing.

   procedure Do_Nothing (X : not null Ref_Element) is null;
   procedure Iterate_2
     (T : Table;
      Action : not null access procedure (X : not null Ref_Element)
                                    := Do_Nothing'Access);

   type My_Index is range 1..10;
   type My_Array is array (My_Index) of Integer;

   procedure Process_Array (X : in out My_Array; Index : My_Index);
```

```
private
   type Element is limited
      record
         Component : Integer;
      end record;
   type Table is limited null record;
end Example;

package body Example is

   An_Element : aliased Element;

   function Lookup (T : Table) return Ref_Element is
   begin
      --  ...
      return Not_Found;
   end Lookup;

   function Lookup_2 (T : Table) return not null Ref_Element is
   begin
      --  ...
      raise Not_Found_2;

      return An_Element'Access;
      -- suppress error: 'missing "return" statement in function body'
   end Lookup_2;

   procedure P (X : not null Ref_Element) is
   begin
      X.all.Component := X.all.Component + 1;
   end P;

   procedure Q (X : not null Ref_Element) is
   begin
      for I in 1..1000 loop
         P (X);
      end loop;
   end Q;

   procedure R is
   begin
      Q (An_Element'Access);
   end R;

   procedure Iterate
     (T : Table;
      Action : access procedure (X : not null Ref_Element)
                                       := null) is
   begin
      if Action /= null then
         Action (An_Element'Access);
         --  In a real program, this would do something more sensible.
      end if;
   end Iterate;

   procedure Iterate_2
     (T : Table;
      Action : not null access procedure (X : not null Ref_Element)
                                       := Do_Nothing'Access) is
   begin
      Action (An_Element'Access);
      --  In a real program, this would do something more sensible.
   end Iterate_2;
```

```
   procedure Process_Array (X : in out My_Array; Index : My_Index) is
   begin
      X (Index) := X (Index) + 1;
   end Process_Array;

end Example;
```

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

Обсуждение…

1 Комментарии

Жан Франсуа Мартинес

6 февраля 2008 г.

Я не уверен, что указатели, отмеченные как не равные нулю, менее эффективны. По крайней мере, в системах, использующих виртуальную память. В них операционная система пропускает нулевую страницу и посылает сигнал любому процессу, который пытается получить к нему доступ (например, когда указатель имеет значение null). Поэтому компилятор может просто полагаться на аппаратное обеспечение и ОС для обнаружения разыменования нулевых указателей. (Я не знаю, это то, что делает Ада)

С другой стороны, если вы отметите указатель как не равный нулю, это означает, что вы хотите, чтобы ошибка была поймана во время атак, а не во время разыменования, поэтому компилятор вынужден вставлять проверки каждый раз, когда указатель влияет на новое значение.

PS: заменить указатели на типы доступа.