

Re: ADA Popularity Discussion Request

Source: <http://coding.derkeiler.com/Archive/Ada/comp.lang.ada/2004-09/0281.html>

From: jayessay (nospam_at_foo.com)

Date: 09/08/04

Date: 08 Sep 2004 15:46:53 -0400

Georg Bauhaus <sb463ba@l1-hrz.uni-duisburg.de> writes:

> jayessay <nospam@foo.com> wrote:
> : Georg Bauhaus <sb463ba@l1-hrz.uni-duisburg.de> writes:
>
> :> Sure it is insufficient. But it adds value by removing the need to
> :> test what the compiler has already found out. How about typos?
> :
> : True. But since this is determined anyway as part of incremental
> : development it doesn't actually save anything in practice. Typos are
> : a good example.
> :
> :
> :> Here is a fake:
> :>
> :> (defun next (n) (1 + n))
> :> (defun hext (n) (1 + (mod n 16)))
> :
> : Let's see this in practice:
>
> [lengthy testing snipped]
>
> Do you see what I mean? ;-) ;-) ;-) Wouldn't have happend with
> proper typing and static type checks, see below.

I never said types here would not have caught the "advance" error. I simply state that the dynamic version was as efficacious at catching that error and also caught the logic error in hext at the same time.

> : What you are saying is, range types (as in Ada) could have caught
> : this. True, but they would not have caught it quicker (or even as
> : quick) as this did.
>
> No, I'm saying that this error can occur in an executing piece
> of program if and only if the compiler allows me to mix
> operations of an integer type and operations a modular type
> without notice.

OK, that is a better way than simple ranges. But it doesn't change the point, see below.

> *Things like this don't happen if you get the types right at compiletime.*

Yes, and they don't happen at runtime if pre and post are checked during incremental development. If you don't do this, then you will likely have problems – just like if you don't get the types right in static languages.

> *Similarly, physical units can be checked at compile time using C++ templates. Is there a set of Lisp macros that guarantees unit safe computations before a program runs?*

Yes this has been done. From what I've seen it is a very nice system. Where it can't make the determination at compile time it generates code for the checks and conversions at runtime.

> *: Also, they wouldn't have caught the error in hext until you ran it anyway.*
>
> *Not true as explained above.*

Your explanation above does not cover the logic error. I don't see right off how modular types catch that.

```
with Text_Io; use Text_Io;
```

```
procedure Foo is
```

```
    type Hex_Num is mod 16;
```

```
    function Hext (N : Integer) return Hex_Num is
    begin
        -- == (1+ (n mod 16)), which is the logic error.
        return Hex_Num(1 + (N mod Hex_Num'Modulus));
    end Hext;
```

```
begin
    for I in 0..20 loop
        Put_Line("Answer is: " & Hex_Num'Image(Hext(i)));
    end loop;
end;
```

```
$ gnatmake hext.adb
```

```
==>
```

```
gcc -c hext.adb
```

```
hext.adb:3:11: warning: file name does not match unit name, should be "foo.adb"
```

```
gnatbind -x hext.ali
```

```
gnatlink hext.ali
```

comp.lang.ada: Re: ADA Popularity Discussion Request

[Yeah, I know the file name is bogus, but there are no errors or warnings about the error, not surprising since you can't know there is a bomb here at compile time]

```
$ ./hext
Answer is: 1
Answer is: 2
Answer is: 3
Answer is: 4
Answer is: 5
Answer is: 6
Answer is: 7
Answer is: 8
Answer is: 9
Answer is: 10
Answer is: 11
Answer is: 12
Answer is: 13
Answer is: 14
Answer is: 15
```

```
raised CONSTRAINT_ERROR : hext.adb:9
$
```

Same as the Lisp:

```
(deftype hex-num () `(mod 16))

(defun hext (n)
  (check-type n integer)
  (let ((result (1+ (mod n 16))))
    (check-type result hex-num)
    result))
==> hext

(dotimes (n 20)
  (format t "~%(hext ~a) --> ~a" n (hext n)))

(hext 0) --> 1
(hext 1) --> 2
(hext 2) --> 3
(hext 3) --> 4
(hext 4) --> 5
(hext 5) --> 6
(hext 6) --> 7
(hext 7) --> 8
(hext 8) --> 9
(hext 9) --> 10
(hext 10) --> 11
(hext 11) --> 12
(hext 12) --> 13
```

comp.lang.ada: Re: ADA Popularity Discussion Request

(hex 13) --> 14

(hex 14) --> 15

Error: the value of RESULT is 16, which is not of type HEX-NUM.

[condition type: TYPE-ERROR]

Restart actions (select using :continue):

0: supply a new value for RESULT.

1: Return to Top Level (an "abort" restart).

2: Abort entirely from this process.

/Jon

--

'j' - a n t h o n y at romeo/charley/november com