Gem #27: Изменение представления данных (Часть 1)

Автор: Robert Dewar, AdaCore

Краткое содержание: Gem Ada #27 - Часть 1, автоматическое изменение представления.

Давайте начнем…

Мощная функция Ada - возможность определить точный формат данных. Это особенно важно, когда у Вас есть внешнее устройство или программа, которая требует очень определенного формата. Некоторые примеры:

```
type Com_Packet is record
   Key : Boolean;
   Id  : Character;
   Val : Integer range 100 .. 227;
end record;

for Com_Packet use record
   Key at 0 range 0 .. 0;
   Id  at 0 range 1 .. 8;
   Val at 0 range 9 .. 15;
end record;
```

который выдает поля записи, а в случае Val - принудительное представление, в котором все нулевые биты представляют значение 100. Другой пример:

```
type Val is (A,B,C,D,E,F,G,H);
type Arr is array (1 .. 16) of Val;
for Arr'Component_Size use 3;
```

который вынуждает компоненты взять только 3 бита, пересекая границы байта по мере необходимости. Заключительный пример:

```
type Status is (Off, On, Unknown);
for Status use (Off => 2#001#, On => 2#010#, Unknown => 2#100#);
```

который позволяет задать специфические значения для типа перечисления, вместо эффективных значений по умолчанию 0,1,2.

Во всех этих случаях мы могли бы использовать эти предложения представления для соответствия внешним спецификациям, что может быть очень полезным. Недостатком таких макетов является то, что они неэффективны, и доступ к отдельным компонентам, или в случае типа перечисления, циклического через значения, может увеличить пространство и время требуемое для выполнения программного кода.

Один из подходов, который часто эффективен, заключается в том, чтобы читать или записывать данные, о которых идет речь, в этой заданной форме, но внутри программы представляют данные в стандартном макете по умолчанию, что обеспечивает эффективный доступ и все внутренние вычисления делает с этой более эффективной формой.

Чтобы следовать этому подходу, вам нужно будет конвертировать между эффективным форматом и указанным форматом. Ада обеспечивает очень удобный способ для этого, как описано в RM 13.6 «Изменение представления».

Идея заключается в использовании деривации типа, где один тип имеет указанный формат, а другой имеет нормальный формат по умолчанию. Например, для случая массива выше, мы бы написали:

```
type Val is (A,B,C,D,E,F,G,H);
type Arr is array (1 .. 16) of Val;

type External_Arr is new Arr;
for External_Arr'Component_Size use 3;
```

Теперь мы читаем и записываем данные с помощью типа External_Arr. Когда мы хотим преобразовать в эффективную форму arr, мы просто используем преобразование типов.

```
Input_Data  : External_Arr;
Work_Data   : Arr;
Output_Data : External_Arr;

(read data into Input_Data)

--  Now convert to internal form
Work_Data := Arr (Input_Data);

(computations using efficient Work_Data form)

—  Convert back to external form
Output_Data := External_Arr (Work_Data);
```

Используя такой подход, достаточно сложная задача копирования всех данных массива из одной формы в другую, со всеми необходимыми маскировочными и сдвиговыми операциями, полностью автоматизирована.

Аналогичный код может использоваться в случаях типа записи и перечисления. Можно даже указать два различных представления двух видах, и преобразовывать из одной формы в другую, как в:

```
type Status_In is (Off, On, Unknown);
type Status_Out is new Status_In;

for Status_In use (Off => 2#001#, On => 2#010#, Unknown => 2#100#);
for Status_Out use (Off => 103, On => 1045, Unknown => 7700);
```

Есть два ограничения, которые должны иметься в виду при использовании этой функции. Во-первых, Вы должны использовать производный тип. Вы не можете поместить пункты представления на подтипы, что означает, что преобразование должно всегда быть явным. Во-вторых, есть RM 13.1 правила (10), который ограничивает размещение интересных пунктов представления:

10 Для нетегового производного типа, никакие связанные с типом элементы представления не позволены, если родительский тип - ссылочным

типом, или имеет какие-либо определяемые пользователем примитивные подпрограммы.

Все предложения представления, интересные с точки зрения изменения представления, являются «родственными типам», поэтому, например, следующая последовательность была бы незаконной:

```
type Val is (A,B,C,D,E,F,G,H);
type Arr is array (1 .. 16) of Val;

procedure Rearrange (Arg : in out Arr);

type External_Arr is new Arr;
for External_Arr'Component_Size use 3;
```

Почему эти ограничения? Ну, ответ немного сложный и связан с соображениями эффективности, которые мы рассмотрим в GEM на следующей неделе.

Связанный исходный код

Attached Files

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

Обсуждение…

5 комментариев

Wes

3 марта 2008 г.

Интересно! Я никогда не знал, что вы можете сопоставить перечисление с другим набором значений. Я всегда создавал отдельную справочную таблицу с перечислением источника в качестве индекса в нее. Спасибо за GEM!

Питер Херманн

4 марта 2008 г.

Воодушевление! Краткое описание. Настоящий жемчуг.

Андерс Эрикссон

14 марта 2008 г.

Мне нравится, когда компилятор делает мои работы!

Когда требуется еще больше скорости, вы можете отключить проверку времени выполнения, активируя ее явно с атрибутом «Действителен» для каждого отдельного подтипированного компонента, который вы хотите проверить. Этот подход особенно полезен при проверке входящих / исходящих сообщений во встроенном приложении.

Питер Херманн

7 мая 2008 г.

надежность подземного сортировки

надежда на освобождение (Пол Маккартни) :-)

Оливер Келлог

20 апреля 2012 г.

Мне удалось закрыть запрос на расширение проекта, над которым я работаю

без изменения одной строки кода приложения, просто используя этот GEM:

Статус | INPROGRESS | RESOLVED

Разрешение | | FIXED

- Комментарий № 3 от Kellogg, Oliver <oliver.kellogg @ .....> 2012-04-20 16:33:03 CEST--

Интересно, что никаких изменений приложений не было,

tdl_dlp16_bite_fom29_from_dlp уже использует l16_fom_words_spec :: fom_29_t

и преобразование из бит-упакованного формата достигается путем преобразования типа Ada

(Http://libre.adacore.com/adaanswers/gems_single/gem-27):

---

r318095 | акеллол | 2012-04-18 12:15:00 +0200 (Wed, 18 Apr 2012) | 29 строк Измененные пути:

M /tdl/mids_if/trunk/rose/l16_fom_types_spec.idl

M /tdl/mids_if/trunk/rose/l16_fom_words_spec.idl

M /tdl/mids_if/trunk/rose/l16_init_sw_mp_types_spec.idl

Синхронизация с MKS MIDS_LVT_external_communication.cat 1.117,
MIDS_LVT_external_communication.sub 1.59:

> Внедрить модель данных
> https: //dcsulm-bugzilla2/bugzilla_f125/show_bug.cgi? id = 1466:
>
> [...]
