Gem #5: Поиск по ключу в сетах (множествах – контейнеры SET)

Автор: Matthew Heaney (On2 Technologies)

Краткое содержание: Gem Ада #5 – Сеты (множества, они же – контейнеры SET) – это контейнеры элементов, где каждый элемент контейнера хранится только в одном экземпляре. Во всех контейнерах, в том числе – сетах, можно осуществлять поиск элементов по их значению с учетом того, что значение элемента известно. В некоторых приложениях эквивалентность элементов определяется только по некоторой их части («ключевой» части); часто приходится осуществлять поиск элемента, когда известно только значение такого ключа. Здесь представлен способ осуществления ключевого поиска элемента в контейнере.

Давайте начнем…

В этом примере показано, как использовать контейнеры хеширования (hashed set containers).

Для примера возьмем реализацию игры, в которой пользователь перемещается между компонентами в лабиринте. Карты лабиринта – сайта-карты и включают стены, комнаты и двери. Одной из особенностей игры является способность находить объекты комнаты с учетом номера комнаты, и мы реализуем эту функцию с помощью контейнера set.

Объекты `Map_Site` в нашей симуляции являются членами общего класса. Мы используем тегированный тип, объявленный следующим образом:

```
type Map_Site is abstract tagged limited null record;
```

(Абстрактный) тип имеет одну (абстрактную) операцию для обеспечения навигации к объекту сайта:

```
procedure Enter (Site : in out Map_Site) is abstract;
```

Затем мы реализуем конкретный тип Room, который происходит из Map_Site. Объявление типа Room выглядит следующим образом:

```
type Room (<>) is limited new Map_Site with private;
```

Тип `Room` ограничен и не определён (потому что он имеет «неизвестные дискриминанты»), что означает, что объекты типа должны быть явно инициализированы вызовом функции стиля конструктора, объявленной следующим образом:

```
not overriding
function New_Room (Number : Positive) return not null access Room;
```

Невозможно создать объект комнаты, кроме вызова New_Room, где мы вставляем новые объекты комнаты в набор, используемый для поиска экземпляров. Теперь мы переходим к телу пакета и созданию экземпляра пакета. Общий фактический тип – это просто простой именованный тип доступа (тот же, который мы используем для размещения экземпляров):

```
type Room_Access is not null access Room;
```

Чтобы создать экземпляр (хэшированного) пакета множеста, нам нужны как хэш-функция, так и функция эквивалентности. Хеш-функция для множеств - это отображение значения элемента в хэш-значение. Итак, как мы можем сделать хэш-значение из объекта комнаты (Room)? Естественным выбором является использование номера комнаты в качестве значения хеша для объекта комнаты, поэтому наша функция хэша для комнат выглядит так:

```
function Hash (R : Room_Access) return Hash_Type is
begin
   return Hash_Type (R.Number);
end;
```

Для функции эквивалентности мы можем просто использовать предопределенный оператор равенства для типа Room_Access, поскольку элементы набора – это значения доступа, а значения доступа уникальны для отдельных объектов Room.

Теперь у нас есть все необходимое для создания универсального хэш-набора:

```
package Room_Sets is new Ada.Containers.Hashed_Sets
  (Element_Type        => Room_Access,
   Hash                => Hash,
   Equivalent_Elements => "=");

Room_Set : Room_Sets.Set;
```

Наконец, мы в состоянии реализовать функцию конструктора Room, которая выглядит так:

```
function New_Room (Number : Natural)
  return not null access Room
is
  R : constant Room_Access := new Room (Number);
begin
  Room_Set.Insert (R);
  return R;
end New_Room;
```

Теперь, когда мы внедрили необходимую инфраструктуру для создания объектов помещения, мы должны реализовать функцию поиска объектов комнаты с использованием номера комнаты в качестве ключа. Наша функция объявлена следующим образом:

```
function Find_Room (Number : Natural) return access Room;
```

Теперь идет интересная часть. В наборе записываются все объекты Комнаты, которые были созданы, поэтому, если комната с таким номером существует, она будет в наборе. Наша проблема заключается в том, что у нас нет немедленного поиска объектов комнаты с учетом номера комнаты. Помните, что это набор комнат (а не карта с номером номера в качестве ключа), а функция поиска для создания выглядит следующим образом:

```
function Find (Container : Set; Item : Room_Access) return Cursor;
```

Проблема в том, что у нас есть номер комнаты, а не объект комнаты, так как мы можем искать объект комнаты в соответствии с его номером?

Решение состоит в том, чтобы воспользоваться преимуществами вложенного обобщенного пакета Generic_Keys, предоставляемого наборами, который позволяет просматривать элемент набора в терминах его ключа. Он имеет общие форматы, очень

похожие на сам пакет, за исключением того, что операции применяются к родовому формальному Key_Type, а не к типу элемента.

Одним из требований к использованию этого пакета является то, что общие фактические данные для ключей должны предоставлять те же значения, что и общие для элементов. Например, функция возврата хэш-значения номера комнаты (ключа) должна возвращать то же значение, что и функция, которая возвращает хеш-значение объекта комнаты (элемента) с этим номером комнаты. Таким образом, мы создаем экземпляр вложенного пакета следующим образом:

```
function Get_Room_Number (R : Room_Access) return Natural is
begin
   return R.Number;
end;

function Room_Number_Hash (N : Natural) return Hash_Type is
begin
   return Hash_Type (N);
end;

package Room_Number_Keys is new Room_Sets.Generic_Keys
 (Key_Type        => Natural,
  Key             => Get_Room_Number,
  Hash            => Room_Number_Hash,
  Equivalent_Keys => "=");
```

Теперь мы готовы реализовать функцию Find_Room. Пакет Room_Number_Keys предоставляет функцию поиска по ключевым словам, поэтому мы вызываем это для поиска набора для объекта комнаты с данным номером комнаты:

```
function Find_Room (Number : Natural) return access Room is
   C : constant Room_Sets.Cursor :=
         Room_Number_Keys.Find (Room_Set, Number);
begin
   if Has_Element (C) then
      return Element (C);
   else
      return null;
   end if;
end Find_Room;
```

Функция поиска возвращает курсор, а значение курсора указывает, был ли поиск успешным. Если объект комнаты, имеющий номер комнаты, находится в наборе, Has_Element возвращает значение True, и мы возвращаем объект Room, обозначенный курсором. Если это число не было найдено (поскольку объект с номером, имеющим это число, не был установлен), мы просто возвращаем значение null.

В пакете Generic_Keys есть еще несколько интересных функций, в том числе возможность изменять элементы набора, но мы отложим это обсуждение до будущего. А пока получайте удовольствие от контейнеров!

Обсуждение…