Материалы по языкам программирования

Динамическое окружение

Понятие динамического окружения. Порядок формирования и использования динамического окружения.

Динамическое окружение — окружение, в котором присутствует структура данных, связывающая имена переменных с их значением во время выполнения очередной инструкции.

Рассмотрим пример формирования динамического окружения на языке SML. В примере упростим представление динамического окружения до массива, содержащего пару “имя переменной” ↔ “значение”. Функции будем связывать с их замыканиями:

```
val x = 2 (* [(x, 2)] *)
val y = x + 3 (* [(y, 5), (x, 2)] *)
val z = x + y (* [(z, 7), (y, 5), (x, 2)] *)
(* #[(x, 49), (z, 7), (y, 5), (x, 2)] произошло затенение *)
val x = z * z
(* [(x, 49), (z, 7), (y, 5), (x, 2)] в let создается временное динамическое окружение *)
val z =
  let
    val x = 7 (* [(x, 7), (x, 49), (z, 7), (y, 5), (x, 2)] *)
    val d = 1 (* [(d, 1), (x, 7), (x, 49), (z, 7), (y, 5), (x, 2)] *)
  in
    z + x + d (* [(z, 15), (d, 1), (x, 7), (x, 49), (z, 7), (y, 5), (x, 2)] *)
  end (* [(z, 15), (x, 49), (z, 7), (y, 5), (x, 2)] *)
(* [(e, -10), (z, 15), (x, 49), (z, 7), (y, 5), (x, 2)] *)
val e = z - 25
(* [(x, 3), (e, -10), (z, 15), (x, 49), (z, 7), (y, 5), (x, 2)] *)
val x = 3
(* [(f, [x], x + 25, [(x, 3), ...]), (x, 3), (e, -10), (z, 15), (x, 49), (z, 7), (y, 5), (x, 2)]
*)
fun f x = x + 25
```

Функция в динамическое окружение записывается в виде замыкания:

```
[форм. параметры], тело, [копия текущего окружения]
```

Здесь отражён основной принцип формирования динамического окружения. Он не идеален, и существует множество других более эффективных реализаций. cred.

Абстракции

Что понимается под терминами “Абстракция данных”, “Абстракция процессов”? Что значит “построение абстракций”? Что такое “Барьеры абстракций”? За счет чего обеспечивается абстрагирование?

Абстрагирование — процесс удаления характеристик чего-либо с целью сведения его к набору существенных элементов.

Абстракция данных — сведение набора данных к упрощенному представлению целого.

Абстракция процессов фокусируется на сокрытии деталей реализации процесса или алгоритма и предоставлении упрощенного интерфейса для его запуска (**методы**). Это позволяет инкапсулировать набор операций в единую сущность. Причём пользователь объекта не обязан знать реализацию метода.

Построение абстракций — процесс компоновки техник абстракции данных и абстракции процессов для постоения одной абстрактной сущности — объекта или класса.

(?) Барьеры абстракций — те высокоуровневые операции, которые предоставлены пользователю создателями класса. Нарушение барьера абстракций — использование низкоуровневых методов и полей, когда имеются более высокоуровневые, выполняющие ту же задачу.

В объектно ориентированных языках программирования абстракция реализована через понятия классов и объектов (абстрагирование данных). Она используется для того, чтобы скрывать низкоуровневые детали компонентов программы, содержащих логику, что позволяет упростить процесс разработки.

В ООП приложения строятся вокруг объектов и **сообщений**, с помощью которых объектам можно подавать сигнал на изменение внутреннего состояния или запрашивать информацию о нём (абстрагирование процессов).

Пример на языке Ruby:

```ruby
class Point
  attr_accessor :x, :y

  def initialize(x, y)
    @x = x
    @y = y
  end

  def move(dx, dy)
    @x += dx
    @y += dy
  end
end

# абстракция данных: передаем данные о точке одним объектом
def diag_move(p, v)
  p.move(v, v)
end

p1 = Point.new(x, y)

# абстракция процесса
p1.move(4, 7)

# нарушение барьера абстракций
p1.x += 10
p1.y += 11
```