

Apr 28, 04 12:26

search.sml

Page 1/2

```

signature ARRAY =
  sig
    type 'a array

    exception Out_of_bounds

    val create : int * 'a → 'a array
    val get : 'a array * int → 'a
    val set : 'a array * int * 'a → unit
    val size : 'a array → int
  end

structure Array :> ARRAY =
  struct
    type 'a array = 'a list ref

    exception Out_of_bounds

    fun create (s: int, v: 'a) : 'a array =
      let
        fun make (0: int): 'a list = []
          | make (n: int): 'a list = v :: (make (n-1))
        in
          ref (make s)
        end

    fun get (ref l: 'a array, n: int): 'a = List.nth (l, n)

    fun set (a as ref l: 'a array, n: int, v: 'a): unit =
      let
        fun replace ([], _) = raise Out_of_bounds
          | replace (h::t, 0) = v::t
          | replace (h::t, n) = h::(replace (t, n-1))
        in
          a := replace (l, n)
        end

    fun size (ref l: 'a array): int = List.length l
  end

signature SEARCH =
  sig
    type elem (* parametric *)
    type storage (* parametric *)

    val find : (storage * elem) → elem option
  end

functor ArraySearch(structure A : ARRAY
                    type e
                    val equal : (e * e) → bool)
  :> SEARCH where type elem = e and type storage = e A.array =
  struct
    type elem = e
    type storage = e A.array

    local
      fun find' (array: storage, left: int, right: int, e: elem)
        (sc: elem → elem option)
  
```

Apr 28, 04 12:26

search.sml

Page 2/2

```

    (fc: unit → elem option): elem option =
  if left = right then
    let
      val e' = A.get (array, left)
    in
      if equal (e, e') then
        sc(e')
      else
        fc()
      end
    else
      let
        val mid = (left + right) div 2
      in
        find' (array, left, mid, e) sc
          (fn () ⇒ find' (array, mid+1, right, e) sc fc)
        end
      in
        fun find (array: storage, e: elem) =
          find' (array, 0, (A.size array) - 1, e)
            (fn e ⇒ SOME e)
            (fn () ⇒ NONE)
        end
      end
end

structure IntArraySearch = ArraySearch (
  structure A = Array
  type e = int
  val equal = op=
)

val a = Array.create (5, ~1);
val _ = Array.set (a, 0, 2);
val _ = Array.set (a, 1, 4);
val _ = Array.set (a, 2, 6);
val _ = Array.set (a, 3, 8);
val _ = Array.set (a, 4, 10);
val _ = IntArraySearch.find (a, 2);
val _ = IntArraySearch.find (a, 3);

```