UNIVERSIDAD AUTÓNOMA DE BAJA CALIFORNIA

Ingeniería en Software y Tecnologías Emergentes

Paradigmas de la programación

Práctica 4

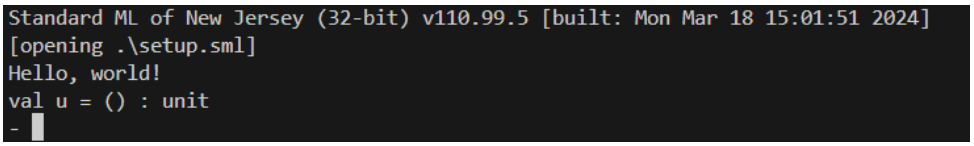
ALUMNO: Cesar Alejandro Velazquez Mercado
MATRÍCULA: 372329

GRUPO: 941

PROFESOR: Carlos Gallegos

30 de mayo del 2024

- 00-01-setup.md
  - Para instalarlo use este video: https://www.youtube.com/watch?v=BC6y1g5gr0A&t=39s



- 01-00-values.md
  - Al copilar “values.sml” obtenemos estos resultados
    - Hello!
    - val i = <hidden> : int
    - val j = 10 : real
    - val k = 10 : int
    - val i' = 11 : int
    - val i = 10 : int
    - val iEqK = true : bool

- 01-01-let-expr.md
  - Standard ML define una expresión 'let' que permite realizar declaraciones con alcance limitado. Fuera de estas expresiones, sus definiciones quedan sin vincular. El resultado de evaluar una expresión 'let', por ejemplo letexpr, es la última expresión contenida entre in ... end.

- 01-02-basic-data-types.md
  - Standard ML tiene seis tipos de datos integrados: unit, bool, int, real, string, y char.
- 01-03-data-structures.md
  - Standard ML tiene tres estructuras de datos integradas. Los campos de los registros se acceden mediante #field record. Los registros no son mapas asociativos; sus etiquetas solo pueden ser nombres alfanuméricos o números mayores que 0.
- 01-04-functions.md
  - Las funciones en Standard ML se declaran usando fn y se pueden nombrar con val. Todas las funciones toman un argumento y son currificadas.
- 01-05-fun.md
  - En Standard ML, se puede usar `fun` para declarar funciones de manera más sencilla en lugar de `fn`. Los argumentos de las funciones pueden tener tipos declarados, pero no es necesario ya que SML los infiere.

Las funciones pueden aceptar y devolver tuplas, permitiendo múltiples argumentos y resultados. Aunque todas las funciones deben aceptar y producir al menos un valor, pueden usar `unit` (`()`) si no se necesita un valor específico.

Para referirse a operadores como `+`, se utiliza la palabra clave `op`.

- 01-07-mod-sigs.md
  - En este paso recopilaremos el archivo “mod-sigs.sml”;

```
PS C:\Users\costco\paradigmas\practica3\a-tour-of-standard-ml-master\examples
sml
Standard ML of New Jersey (32-bit) v110.99.5 [built: Mon Mar 18 15:01:51 2024
[opening .\mod-sigs.sml]
structure Math : sig
  val e : real
end
```

    como resultado obtenemosstructure
    Math : sig
      val e : real
    end
- 01-10-new-data-types.md
  - Los alias de tipos se definen con la palabra clave type, mientras que los nuevos tipos de datos se declaran con la palabra clave datatype.

    Por ejemplo, major_arcana_card define un tipo que es un par de un nombre y un número.

    Los tipos de datos pueden tener múltiples casos exclusivos. Por ejemplo, card_suit define un tipo con cuatro valores, y card_value define un tipo con catorce valores.
- 01-11-recursive-datatypes.md
  - Standard ML of New Jersey (32-bit) v110.99.5 [built: Mon Mar 18 15:01:51 2024]
    [opening .\recursive-datatypes.sml]
    infixr 4 +:
    datatype 'a list = +: of 'a * 'a list | eol
    datatype 'a tree = leaf | node of {left:'a tree, right:'a tree, value:'a}
    val ints = 1 +: 2 +: 3 +: eol : int list
    val inttree =
      node
        {left=node {left=leaf,right=leaf,value=2},
         right=node {left=leaf,right=leaf,value=3},value=1} : int tree
- 02-00-pattern-matching.md
  - PS C:\Users\costco\paradigmas\practica3\a-tour-of-standard-ml-master\examples> sml .\pattern-matching.sml

Standard ML of New Jersey (32-bit) v110.99.5 [built: Mon Mar 18 15:01:51 2024]
[opening .\pattern-matching.sml]
val map = fn : ('a -> 'b) -> 'a list -> 'b list
val map' = fn : ('a -> 'b) -> 'a list -> 'b list

- 02-01-exhaustive.md
  - Standard ML of New Jersey (32-bit) v110.99.5 [built: Mon Mar 18 15:01:51 2024]
    [opening .\exhaustive.sml]
    .\exhaustive.sml:1.6-1.28 Warning: match nonexhaustive
    nil => ...

    val inexhaustive = fn : 'a list -> 'b list
- 02-02-deconstr.md
  - .\deconstr.sml:10.5-10.31 Warning: binding not exhaustive
    _ :: second :: _ :: nil = ...
    datatype dog = dog of {name:string}
    val n = (1,2,3) : int * int * int
    val two = 2 : int
    val x = 1 : int
    val y = 2 : int
    val z = 3 : int
    val charlie = dog {name="Charlie"} : dog
    val lucky = dog {name="Lucky"} : dog
    val pup1 = "Lucky" : string
    val pup2 = "Charlie" : string
    val second = 2 : int
- 02-03-pattern-fun.md
  - [opening .\pattern-fun.sml]
    [autoloading]
    [library $SMLNJ-BASIS/basis.cm is stable]
    [library $SMLNJ-BASIS/(basis.cm):basis-common.cm is stable]
    [autoloading done]
    Hullo, Grom, wielder of Gorehowl!
    opening String
    type char = ?.char
    type string = ?.string
    val maxSize : int
    val size : string -> int
    val sub : string * int -> char
    val str : char -> string
    val extract : string * int * int option -> string
    val substring : string * int * int -> string
    val ^ : string * string -> string

```
  val concat : string list -> string
  val concatWith : string -> string list -> string
  val implode : char list -> string
  val explode : string -> char list
  val map : (char -> char) -> string -> string
  val translate : (char -> string) -> string -> string
  val tokens : (char -> bool) -> string -> string list
  val fields : (char -> bool) -> string -> string list
  val isPrefix : string -> string -> bool
  val isSubstring : string -> string -> bool
  val isSuffix : string -> string -> bool
  val compare : string * string -> order
  val collate : (char * char -> order) -> string * string -> order
  val < : string * string -> bool
  val <= : string * string -> bool
  val > : string * string -> bool
  val >= : string * string -> bool
  val toString : string -> String.string
  val scan : (char,'a) StringCvt.reader -> (string,'a) StringCvt.reader
  val fromString : String.string -> string option
  val toCString : string -> String.string
  val fromCString : String.string -> string option
  val rev : string -> string
  val implodeRev : char list -> string
  val concatWithMap : string -> ('a -> string) -> 'a list -> string
datatype player
  = mage of {magic_type:string, name:string}
  | warrior of {name:string, weapon:string}
val greet_player = fn : player -> unit
val u = () : unit
```

- 02-04-cond-expr.md
  - val trueCond = 1 : int
  - val elseCond = ~1 : int
- 02-05-recursion.md
  - val sum = fn : int list -> int
  - val sum_iter = fn : int list -> int
  - val s = 6 : int
  - val s' = 6 : int
- 02-06-hofs.md
  - [opening .\hofs.sml]
    [autoloading]
    [library $SMLNJ-BASIS/basis.cm is stable]
    [library $SMLNJ-BASIS/(basis.cm):basis-common.cm is stable]
    [autoloading done]

val twos = [2,2,2] : int list
val two = [2] : int list
val foldr = fn : ('a * 'b -> 'b) -> 'b -> 'a list -> 'b
val sum = fn : int list -> int
val s = 6 : int
val length = fn : 'a list -> int
val l = 3 : int

- 02-07-inf.md
  - PS C:\Users\costco\paradigmas\practica3\a-tour-of-standard-ml-master\examples> sml .\inf.sml
    Standard ML of New Jersey (32-bit) v110.99.5 [built: Mon Mar 18 15:01:51 2024]
    y ya no paso nada
- 02-08-chaining.md
  - Standard ML of New Jersey (32-bit) v110.99.5 [built: Mon Mar 18 15:01:51 2024]
  - [opening .\chaining.sml]
  - Hello!
  - Another line!
- 02-09-mutable-refs.md
  - Standard ML of New Jersey (32-bit) v110.99.5 [built: Mon Mar 18 15:01:51 2024]
    [opening .\mutable-refs.sml]
    val x = <hidden> : int ref
    val y = ref 20 : int ref
    val ++ = fn : int ref -> int
    val x = ref 1 : int ref
    val xNewState = 1 : int
    val xValue = 1 : int
- 02-10-while.md
  - [autoloading done]
    1
    2
    3
    4
    5
    6
    7
    8
    9
    10
    val x = ref 10 : int ref
    val u = () : unit

- 03-00-cml-intro.md
    -
- 03-01-spaw[opening .\spawn.sml]
    - [autoloading]
      [library $SMLNJ-BASIS/basis.cm is stable]
      [library $SMLNJ-BASIS/(basis.cm):basis-common.cm is stable]
      Hello!
      World!
      World!
      World!
      World!
      World!

- 03-02-chan.md
    - 17 -5 12
- 03-03-select.md
    - 0
      1
      1
      2
      3
      5
      8
      13
      21
      34
      quit
- 03-04-mailboxes.md
    - envía el valor 10 a través del buzón y luego lo recibe,

- 03-05-ivars.md
    - [opening .\ivars.sml]
      [autoloading]
      [library $SMLNJ-BASIS/basis.cm is stable]
      [library $SMLNJ-BASIS/(basis.cm):basis-common.cm is stable]
      [autoloading done]
      .\ivars.sml:1.16-1.23 Error: unbound structure: SyncVar

- 03-06-mvars.md
    - No se imprime nada directamente, pero se espera que se produzca una excepción Put al intentar establecer dos veces el mismo valor en la variable de sincronización.