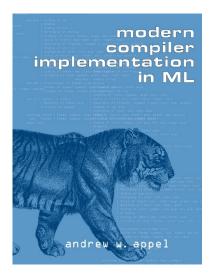
Introduction au Langage

Pablo de Oliveira <pablo.oliveira@uvsq.fr>

March 27, 2015

Langage

Introduction



Introduction

- ► Proposé par Andrew Appel en 98 dans ses livres "Modern Compiler implementation in . . . "
- ► C'est un langage
 - Impératif
 - Typé
 - ▶ Types de base: entiers et chaînes
 - ► Types dérivés: tableaux et structures
 - Fonctions imbriquées

Hello World

print("Hello World!")

Factorielle

```
let
  function fact(n : int) : int =
    if 1 < n then n * fact(n - 1) else 1
in
    print_int(fact(7))
end</pre>
```

Blocs let-in-end

```
let
    /* Declarations */
    type celsius = int
    var thermostat : celsius := 17;
    function incrementer( n : int) : int = n + 1
in
    /* Expressions */
    thermostat := incrementer(thermostat)
end
```

Fonctions

```
let
    function fonction (a1 : int, a2 : string) : int = ...
    function fonctionVoid (a1 : int, a2 : string) = ...
in
    ...
end
```

Variables

Types

- Deux types primitifs:
 - ▶ int entiers signés 32 bits de -2^{31} à $2^{31} 1$.
 - ▶ attention notre parseur (pour simplifier) ne gère pas -2^{31}
 - string chaîne de caractères
- Des types dérivés:
 - tableaux (array)
 - structures (record)
 - alias

Tableaux

```
let
    type int_array = array of int
    var tableau := int_array[100] of 0
in
    tableau[31] := 15;
    tableau[32] := tableau[33] + tableau[34]
end
```

Structures

Tableaux de Structures

Alias

```
let
    type celsius = int
    type farenheit = int
    var c : celsius = 27
    var f : farenheit = 0
    function ctof(c: celsius) : farenheit = ...
in
    /* Attention: legal */
    f := c
end
```

Fonctions imbriquées

```
let
    type int_list = {value: int, next: int_list}
    function sum(a : int_list): int =
        let
            function reduce(a : int list,
                             sum : int) : int =
                if a.next == nil then
                    sum + a.value
                else
                    reduce(a.next, sum + a.value)
        in
            reduce(a, 0)
        end
in
end
```

Fonctions récursives

```
let
   function odd(n : int) : int =
      if n == 0 then 0 else even(n-1)
   function even(n : int) : int =
      if n == 0 then 1 else odd(n-1)
in
   if odd(5) then print("5 est impair")
```

Primitives

```
print(s : string)    print_int(i : int)
getchar() : string
ord(s : string) : int
chr(i : int) : string
size(s : string) : int
concat(s1 : string, s2 : string) : string
substring(s : string, f : int, n : int) : string
not(i : int) : int
exit(code : int)
                                 4 D > 4 A > 4 B > 4 B > B 9 9 9
```

Exercices

- Ecrire une fonction qui donne le nieme terme de la suite de Fibonacci
- ► Ecrire une fonction qui trouve le plus grand élément d'un tableau
- ► Ecrire une fonction qui calcule la profondeur d'un arbre binaire
- Ecrire un tri-bulle sur un tableau