



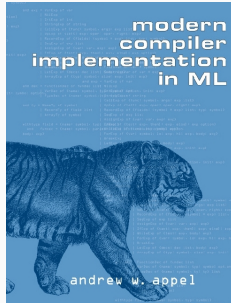
## Sequence 1.2 – Introduction to the Tiger Language

---

P. de Oliveira Castro    S. Tardieu

# The Tiger programming language

- introduced by A. Appel in 1998;
- imperative;
- typed, with two primitive types (integers and strings);
- has nested functions.



**Figure 1:** A. Appel, Modern Compiler Implementation

# Hello World

```
print("Hello, World!\n")
```

- Two primitive types:
  - `int`: signed 32 bits integers from  $-2^{31}$  to  $2^{31} - 1$ ;
  - `string`: string of ASCII 8 bit characters.
- In the compiler, a function or an expression returning no values will be denoted by the pseudo-type `void`.

## let/in/end blocks and variables

```
let
  /* Declarations */
  var thermostat : int := -17
in
  /* Expressions */
  thermostat := thermostat + 1;
  print_int(thermostat);
  print("\n")
end
```

## Explicit vs. implicit typing

```
let
  var a : int := 0    /* type is given explicitly */
  var b := 1         /* int is inferred from context */
  var c := "hello"   /* string is inferred from context */
in
  ...
end
```

## Blocks and values

A block evaluates to the value of its last expression:

```
print_int(let
    var a := 8
  in
    a := a + 2;
    a * a
  end) /* This prints 100 */
```

A test evaluates to the taken branch:

```
print_int(if 17 > 3 then 100 else 200) /* This prints 100 */
```

# Control Flow

```
let
  var j := 10
in
  /* Test */
  if 17 > 3
    then print("17 > 3, all is good\n")
    else print("Houston, we have a problem\n");

  /* for loop */
  for i := 0 to 10 do
    (print_int(i); print("\n"));

  /* while loop */
  while j > 0 do
    (print_int(j); print("\n"); j := j - 1)
end
```



# Functions declarations

```
let
  var thermostat : int := 17
  /* return type is int */
  function get_temperature() : int =
    thermostat
  /* no values returned */
  function increment(delta : int) =
    thermostat := thermostat + delta
in
  ...
end
```

# Recursive functions

```
let
  function fact(n : int) : int =
    if n > 1 then n * fact(n - 1) else 1
in
  print_int(fact(7));
  print("\n")
end
```

# Mutually recursive functions

```
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 is odd\n")
end
```

## Nested functions

```
let
  function fact(n : int): int =
    let
      function f(n : int, acc : int) : int =
        if n > 1 then f(n - 1, acc * n) else acc
      in
        f(n, 1)
    end
in
  print_int(fact(7));
  print("\n")
end
```

## Primitive functions (Tiger standard library)

The following functions are part of the Tiger language library:

```
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)
```