3 Bench Programming Language

Wednesday, January 24, 2024 12:00 PM

Basic Components {
- literals
- expressions
- conditionals
- Iterations
- "functions"

- THE PUCK 24.1 PROGRAMMING LANGUAGE (based on Oberon-07)

  https://people.inf.ethz.ch/wirth/Oberon/index.html

```
% This is a comment
PROCEDURE main ( )
  WRITE ( "Hello" , "World" ) ;
  x := 2 + 2 ;
  y := 3 * 12 / 7.5 ;
  p := ( x > 0 ) AND ~ ( y <= 30 ) ;
  WRITE ( x * 100 )
END.
```

- Keywords are UPPERCASE
- := as assignment.
- notice space separation.
- Literals: Numbers, Strings.
- Relational and Logic
- ; is a separator, Not a terminator.
- ~ Logical negation

- bye bye { }

```
% FizzBuzz
PROCEDURE FizzBuzz ( n )
  IF n MOD 3 = 0 THEN
    IF n MOD 5 # 0 THEN WRITE ( "Fizz" )
    ELSE WRITE ( "FizzBuzz" )
    END
  ELSE
    IF n MOD 5 = 0 THEN
      WRITE ( "Buzz" )
    END
  END
END.
```

- = equality
- # not equals
- MOD modulus
- DIV Integer division //
- Indentation is optional.

```
% Functions and Loops
FUNCTION fibo ( n )
  x := 1 ; y := 2 ; c := 3 ;
  WHILE c < n DO
    x := x + y ;
```

PROCEDURE - don't return
vs
FUNCTION - do return.

RETURN is not an independent

```
  WHILE c < n DO
     x := x + y ;
     y := x - y ;
     c := c + 1
  END
RETURN x END.
```

FUNCTION - do return.

RETURN is not an independent statement.

semicolon separates

RETURN is part of FUNCTION.

```
% Greatest Common Denominator
FUNCTION gcd ( a , b )
 WHILE a > b DO
    a := a MOD b
 ELSIF b > a DO
    b := b MOD a
 END
  % post : a = b
RETURN a END.
```

• WHILE (Dijkstra)

- conditions are evaluated, the first one that evaluates to true, the corresponding block is executed
- The loop exits when all conditions are false.

```
WHILE cond1 DO
   block1
ELSIF cond2 DO
   block2
ELSIF cond3 DO
   block3
......
END
```

```
% NewLines are just WhiteSpace
PROCEDURE foo ( s )
  res := true ;
  IF ( s = "A" ) OR ( s = "B" ) AND
     ( s = "C" ) OR
     ( s = "D" ) THEN
     WRITE (
          "Hello" & "World"
          )
  END
END.
```

string concatenation.

+ is commutative

$A + B \equiv B + A$

```
% IF and WHILE may need semicolons
FUNCTION zap ( s )
 1) res := 3 ;
 2) IF s > 0 THEN
```

```
FUNCTION zap ( s )
  1)res := 3 ;
  2)IF s > 0 THEN
       WRITE ( "zero" ) ○
    END ;
  3)WHILE s > 0 DO
      s := s - 1 ○
    END ;
  4)WRITELN ( "Done!" ) ○
RETURN res END.
```

EOF