Tipos primitivos: int, real, bool, char e string

Operadores aritméticos: +, -, *, /, div e mod

Operadores relacionais: >, >=, <, <=, = e <>

Operadores lógicos: orelse, andalso

Listas: elementos de mesmo tipo.
Exemplos: [1, 2, 3];
1::2::3::nil;

Tuplas: elementos de tipos diferentes, onde a ordem importa.
Exemplos: (2, “Maria”);
(("gato", 3), false);

Registros: elementos nomeados, de tipos diferentes, acessados em qualquer ordem.
Exemplo:
val e = {nome = “Lucas”, idade = 23};

Acesso: val n = #nome e;
val i = #idade e;

Bibliotecas:

- load “Math”; // Carrega biblioteca
- val PI=Math.pi; // usa constante da biblioteca
- help “Math”’; // ajuda (mostra comandos e constantes)

Trabalhando com arquivos:

- use “arquivo.txt”; // carrega arquivo para uso

Como separar listas:

val h::t = [1,2,3]; (e::lst)

val lista3 = [50, 10, 20, 30];
val h1::h2::t=lista3; (e::e::lst)

val h = hd(lista3); (int)
val t = tl(lista3); (lst)

Como concatenar listas/elementos:

val l1 = 1::2::nil; (e::e::e)

val l2 = 3::l1; (e::lst)

val l3 = l2@l1; (lst@lst)

val l = tl(lista1)@[hd(lista1)];

Exemplos de funções:

```
fun fat 0 = 1                    (* critério parada *)
|   fat n = n * fat (n-1);       (*recursão *)

fun fat2 n = if n = 0 then 1     (* seleção *)
             else n * fat (n-1);
```

```
fun crialst i f = if i > f then nil
    else i::crialst (i+1) f;
```

```
fun somalista nil = 0
|   somalista (h :: t) = h + somalista (t);

fun somalista2 nil = 0
|   somalista2 lst = hd(lst) + (somalista2 (tl(lst)));
```

```
fun inc n = n + 1;
inc n;
n;                (n não é modificado!)
```

```
fun soma (x1, y1) (x2, y2) = (x1+x2, y1+y2);
```

Funções interessantes:

ord(#"A"); (* devolve o código ASCII *)

chr(65); (* devolve o caractere de código 65 *)

size("string"); (* devolve tamanho da string *)

"conc"**^**"atena"; (*Concatena strings *)