1.1.6. Controle de Fluxo em ML

* **Repetição**: recursão

* **Seleção**: if e case

➥ if <exp> then <then_exp> else <else_exp>
➥ <exp> quando avaliada deve retornar um booleano
➥ o else é obrigatório
➥ Não é um "comando condicional", é uma "expressão condicional"

```
- val x = 10;
> val x = 10 : int
- val y = 20;
> val y = 20 : int
- if x>y then x else y;
> val it = 20 : int
- fun negative x = if x<0 then true else false;
> val negative = fn : int -> bool
- negative ~5;
> val it = true : bool
- negative 5;
> val it = false : bool
- negative ~20;
> val it = true : bool
- negative 20;
> val it = false : bool
```

➥ case <exp> of <match>
➥ semântica simples uma vez que o pattern matching foi definido
➥ Exemplo:

```
datatype MES = Jan | Fev | Mar | Abr | Mai | Jun | Jul | Ago | Set | Out | Nov | Dez;
fun dia_do_ano(ano, mes, dia) =
    let
        val ano_bi = if eh_ano_bi(ano) then 1 else 0
    in
        case mes of Jan => dia
            Fev => 31 + dia
            Mar => 59 + dia + ano_bi
            Abr => 90 + dia + ano_bi
            Mai => 120 + dia + ano_bi
            Jun => 151 + dia + ano_bi
            Jul => 181 + dia + ano_bi
            Ago => 212 + dia + ano_bi
            Set => 243 + dia + ano_bi
            Out => 273 + dia + ano_bi
            Nov => 304 + dia + ano_bi
            Dez => 334 + dia + ano_bi
    end;
```

➥ Exercício: Fazer uma função onde seja definido um tipo "dir" que pode ser N,S,L,O e uma função que retorne 0 caso a direção seja N, 90 caso a direção seja L, 180 caso a direção seja S e 270 caso a direção seja O (usar case).

```
- datatype DIR = Norte | Sul | Leste | Oeste;
> datatype DIR
  con Norte = Norte : DIR
  con Sul = Sul : DIR
  con Leste = Leste : DIR
  con Oeste = Oeste : DIR
- fun direcao(d) =
        case d of Norte => 0
                 |Leste => 90
                 |Sul => 180
                 |Oeste => 270;
> val direcao = fn : DIR -> int
```

➥ Dá para usar o case sem definir o pattern matching, mas neste caso aparece uma mensagem de warning.

```
- fun teste (x) = case x of "um" => 1
                  |"dois" => 2
                  |"tres" => 3;
! Toplevel input:
! ........................."um" => 1
!                  |"dois" => 2
!                  |"tres" => 3.
! Warning: pattern matching is not exhaustive
> val teste = fn : string -> int
- teste("dois");
> val it = 2 : int
- teste("um");
> val it = 1 : int
- teste("cinco");
! Uncaught exception:
! Match
```

Obs1: ML tem tratamento de exceção

Obs2: Para suspender a execução de um arquivo .sml para verificar as mensagens exibidas, há duas opções:
1) Fazer com que o usuário digite alguma coisa (mesmo que ele não faça nada com o valor digitado). Por exemplo, que nem no help que tem que digitar "n" para ver o resto. O código seria:

```
- print "Digite n para continuar";
- val n = BasicIO.input(BasicIO.std_in, 2);
```

2) No WindowsNT é possível configurar a janela do DOS para ter scroll. Tem que posicionar o mouse na barra de título, clicar com o botão direito e selecionar propriedades. Depois tem que ver... Eu não lembro de todas as opções, mas sei que tem que selecionar a segunda ou terceira guia e alterar um valor "height" para +- 120.

1.1.7. Bibliotecas

*** Array**

```
(* Testes com Array *)
load "Array";
open Array;

val L = [2.2, 4.1, 29.1, 33.9, 18.7, 5.6, 10.3, 24.2, 8.5];

(* cria um array com os elementos da lista acima *)
```

```
val A1 = fromList(L);

(* cria array de 10 posições, sendo que cada posição é inicializada com "z" *)
val A2 = array(10,"z");

(* verifica o tamanho do array A1 *)
val tam = length A1;

(* retorna o terceiro elemento de A1 *)
sub(A1, 2);

(* retorna o primeiro elemento de A2 *)
sub(A2, 0);

(* substitui o terceiro elemento de A3 por 8*)
update (A3,2,8);

(* retorna um vetor de elementos a[i..length a-1] de a *)
val V = extract(A2,5,NONE);
```

*** List**

```
(* Testes com Lista *)
load "list";
open list;

val L1 = [6, 18, 5, 2, 17, 4];     (* lista de numeros inteiros *)

val L2 = [#"a", #"2", #"E", #"g", #"7", #"P"];    (* lista de caracteres *)

val R = rev(L1);     (* inverte a lista *)

val T = take(L2, 3);     (* retorna os três primeiros elementos da lista *)
```

*** Math**

```
(* Testes com a biblioteca Math *)
load "Math";
open Math;
val S = sin(90.0);
val C = cos(90.0);
val S = sqrt(9.0);
val x = S + pi;
val y = pow(2.0,8.0);
```

*** Real (incompleta)**

```
(* Testes com a biblioteca Real *)
load "Real";
open Real;
- ~2.0;
> val it = ~2.0 : real
- abs(it);
> val it = 2.0 : real
- floor(3.56);
```

```
> val it = 3 : int
- ceil(3.56);
> val it = 4 : int
- toString(56.78);
> val it = "56.78" : string
- round(45.76);
> val it = 46 : int
- round(45.12);
> val it = 45 : int
- trunc(45.76);
> val it = 45 : int
- trunc(45.12);
> val it = 45 : int
- fromInt(3);
> val it = 3.0 : real
- fromInt(56);
> val it = 56.0 : real
```

*** TextIO (incompleta)**

```
(* Testes com a biblioteca TextIO *)
load "TextIO";
open TextIO;

(* Escrita no arquivo *)
val os = openOut "arq.txt";
val lista = [92.0,38.2,47.6,10.5];
fun Imprime(L) = if (tl(L) = nil)
                 then output(os, toString(hd(L)) ^ "\n")
                 else ( output(os, toString(hd(L)) ^ "\n"); Imprime(tl(L)) );
Imprime(lista);
closeOut os;

(* Leitura do arquivo *)
val is = openIn "arq.txt";
val n = input(is);
closeIn is;

(* Escrita na tela *)
print "Numeros lidos: \n";
print n;
print "\n";
```

*** Vector**

```
(* Testes com Vetor *)
load "Vector";
open Vector;

(* vetor de numeros reais *)
val V1 = #[2.2, 84.1, 92.1, 39.3, 18.7, 5.6, 10.3, 42.2, 8.5, 7.2];

(* cria vetor a partir de uma lista *)
val L = [9, 6, 3];
val V2 = fromList(L);
```

```
(* tamanho do vetor V1 *)
val tam = length V1;
```

*** String (incompleta)**

```
(* Testes com a biblioteca String *)
load "String";
open String;
- val S1 = "teste_do_mosml";
> val S1 = "teste_do_mosml" : string
- val S2 = "PUC";
> val S2 = "PUC" : string
- val LS = ["ParadigmasI ", "turma", " 128"];
> val LS = ["ParadigmasI ", "turma", " 128"] : string General.list

-size(S1);
> val it = 14 : int

-sub(S1,0);
> val it = #"t" : char

- sub(S1,5);
> val it = #"_" : char

- sub(S1,size(S1)-5);
> val it = #"m" : char

-substring(S1,6,2);
> val it = "do" : string
```

Obs: dá problema carregar duas bibliotecas (com *load* e *open*) ao mesmo tempo, por exemplo Array e Vector, pois possuem várias funções com o mesmo nome. A impressão é que a última é que fica valendo, mas na verdade continuando ocorrendo alguns problemas. Neste caso, deve-se usar

<nome da biblioteca>.<nome da função>

Por exemplo:

```
- val L = ["Testando", "Cidade", "MosML", "PUC", "ParadigmasI"];
> val L = ["Testando", "Cidade", "MosML", "PUC", "ParadigmasI"] : string list

(* Retorna uma lista de inteiros com o tamanho de cada string da lista de strings *)
- list.map String.size L;
> val it = [8, 6, 5, 3, 11] : int list
```