# VII. Ambientes em Tempo de Execução

## Procedimentos

- Uma definição de procedimento associa um identificador a um enunciado(corpo).

Ex: procedure Incrementa (var x: inteiro);
begin x:= x+1; end;

- Uma chamada de procedimento invoca a execução de seu corpo de instruções e é referenciada por uma ativação, em tempo de execução.

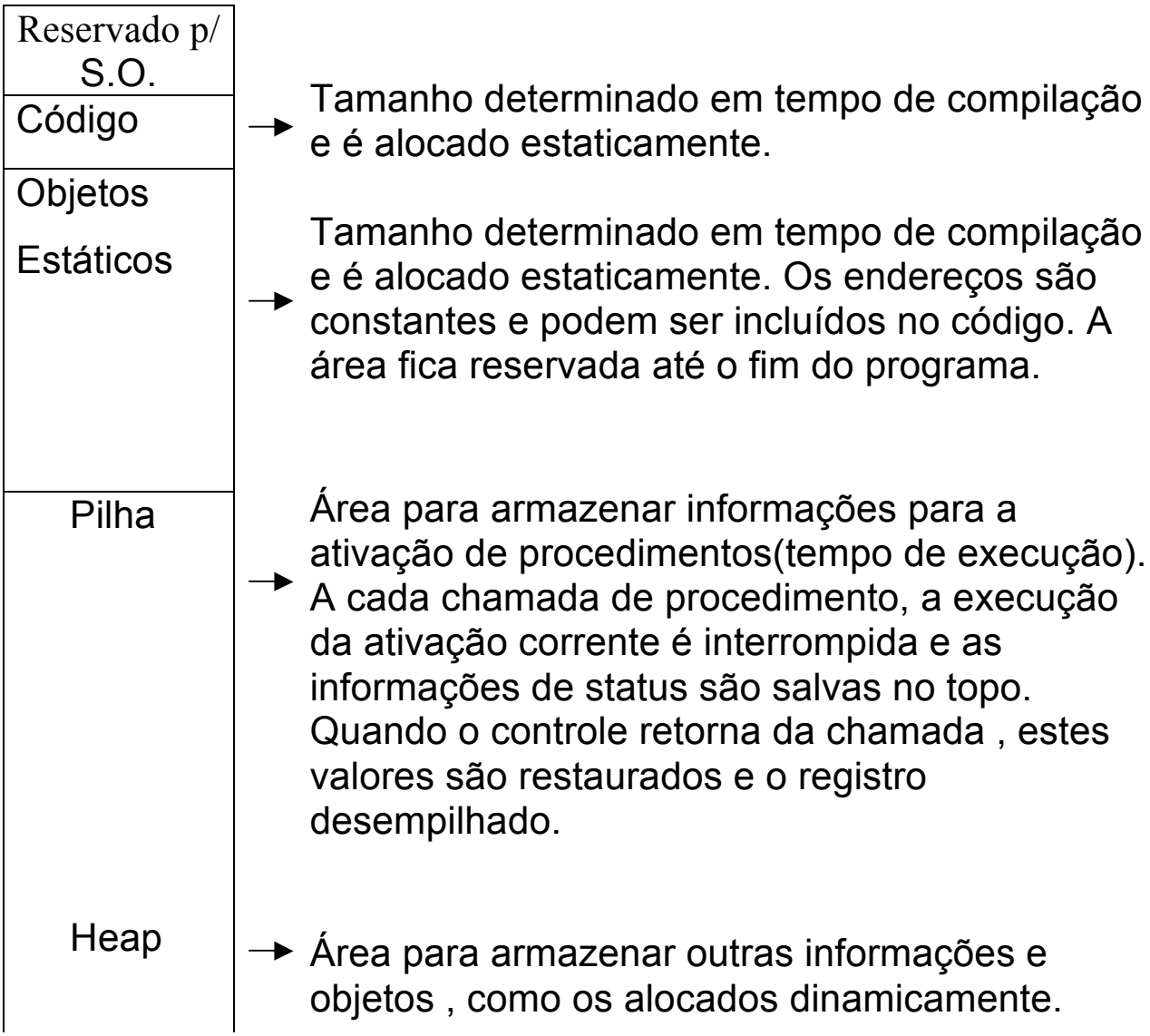
## Escopo de uma declaração

É a porção do programa à qual se aplica uma declaração. Uma ocorrência de um nome é dita local se estiver no escopo de uma declaração do procedimento, caso contrário é dita não-local.

| **Noção estática (P.F.)** | **Contraparte dinâmica (exec.)** |
|---|---|
| Definição de um procedimento | Ativação de um procedimento |
| Declaração de um nome | Amarração de um nome a uma área de memória |
| Escopo de uma declaração | Tempo de vida de uma amarração |

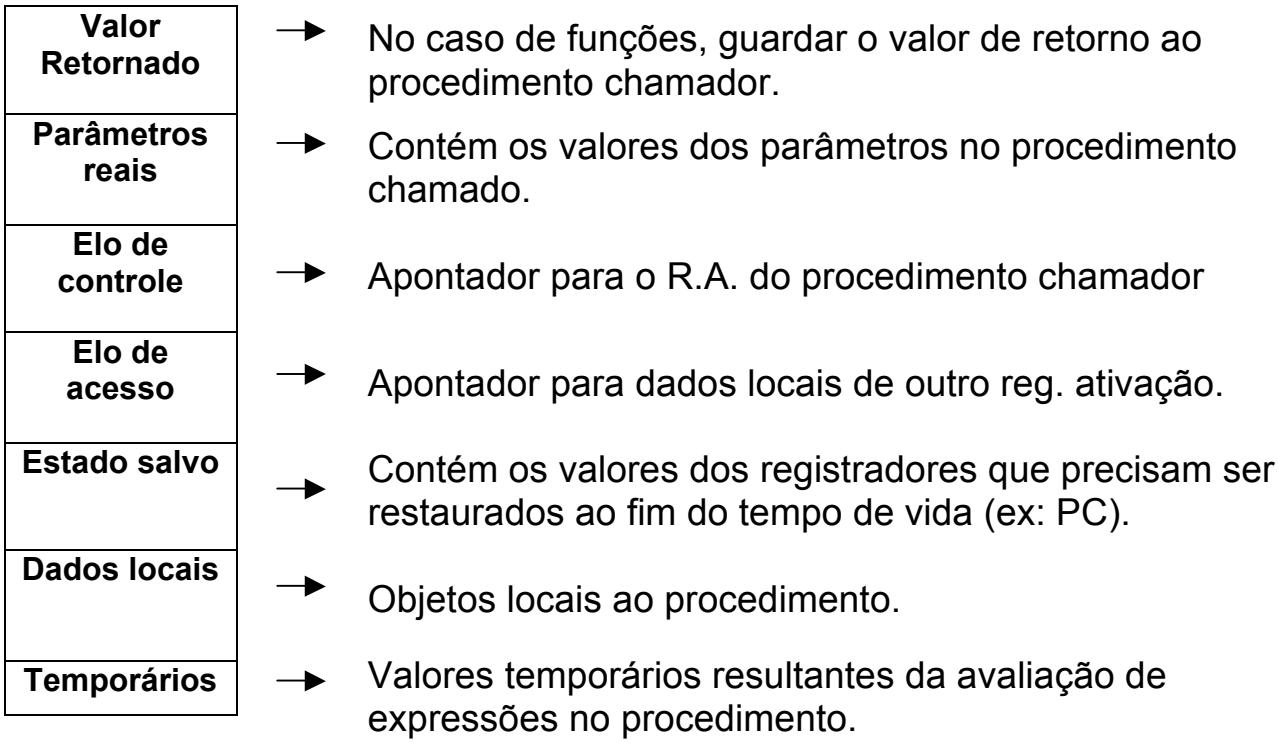
## Organização de Memória

Exemplo de Organização ( Alocação de R.A. em pilha)



## Registro de Ativação

Quando um procedimento é chamado, um conjunto de informações deve ser salvo em memória, na área estática , pilha , ou heap . O registro tem os seguintes campos (normalmente):



## Estratégias para alocação de registros de ativação

### Alocação em memória estática

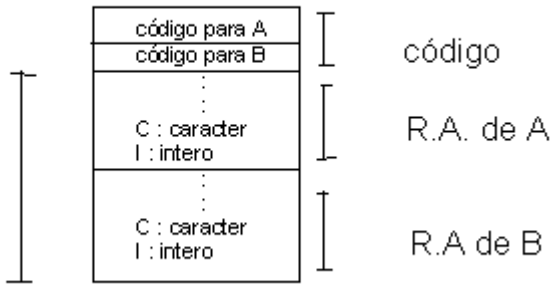
Objetos alocados estaticamente têm seus tamanhos e endereços determinados em tempo de compilação. Os endereços são obtidos através de um valor de deslocamento, a partir de uma extremidade do registro de ativação. Neste caso, os registros de ativação ficam alocados na área de dados estáticos.

O Fortran 77 é um exemplo de linguagem que usa esta estratégia. Todas as ativações de um procedimento utilizam um único registro de ativação.

Ex:

```
Program A
| character C
| integer I
End

Character Function B|
| character C
| integer I
End
```




### Alocação em Memória de Pilha

Os registros de ativação são empilhados e desempilhados à medida em que as ativações são iniciadas e terminadas, respectivamente. Cada ativação de um procedimento X reserva uma área na pilha, para seu registro de ativação.

Essa área é liberada ao fim da ativação, com a perda dos valores armazenados.

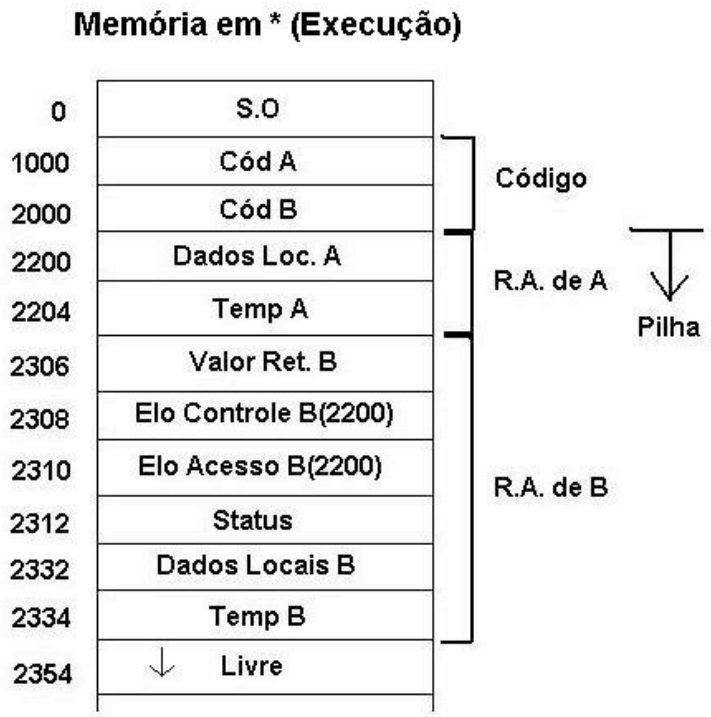
Ex:

Prog A
Var i, j: integer
.
.
.
i := 1;
j := B;
End;

Tam Código A = 1000,
temp = 102 bytes

Func B : Integer
Var i: integer
.
*
return(i);
End;

Tam Código B = 200,
temp = 20 bytes
Estado salvo = 20 bytes




## Alocação em Memória de Heap

Aloca blocos de memória contígua, à medida do necessitado, para registros de ativação ou outros objetos (Alocação Dinâmica). Estes blocos podem ser liberados em qualquer ordem. Após o fim da ativação, os dados da área reservada não são perdidos. A liberação desta área deve ser requisitada por instruções do programa.

## Acesso aos nomes não locais

Um bloco é uma parte do código que contém suas próprias declarações de dados locais.

Regra do Aninhamento mais interno

1) O escopo de uma declaração inclui o bloco em que foi declarada.
2) Se um nome **X** não foi declarado no bloco **B** onde aparece, esta ocorrência pertence ao escopo do bloco **A** mais internamente aninhado que envolve **B** e que possui declaração de **X**.
3) O escopo de um nome **X** não abrange um bloco aninhado que contenha outra declaração para **X**.

Regras de Escopo

Indicam a forma de tratamento das referencias a nomes não-locais. Se dividem em regra de escopo léxico (ou estático) e regra de escopo dinâmico.

Na R.E Léxico, determina-se o escopo das declarações a partir do exame do código estático. Na R.E dinâmico é preciso considerar de onde foi chamada a ativação do procedimento (tempo de execução) para se saber a qual escopo os nomes não-locais fazem referencia.

# Transmissão de parâmetros

**Definições:**

- Parâmetro real: argumento transmitido de um procedimento chamador a um procedimento chamado.
- Parâmetro formal: identificador que aparece na definição do procedimento chamado e é tratado como nome local.

## Transmissão de Parâmetro por Valor

O parâmetro real da chamada é avaliado e seu conteúdo passado para o procedimento chamado. O parâmetro formal é considerado um nome local e não afeta diretamente os valores no registro de ativação do procedimento chamador.

Ex:
```
Program A;

    var a,b,c : integer;
    procedure B (a,b : int);
    |    a := 2;
    |    c := a + b;

    a := 0; b := 1; c:= 2;
    B (a,b);
    Writeln (a,b,c);
```

saída: 0, 1, 3

Ex:

```
void A (*int a; int b)
{ *a := b }

main ()
{   int a=1, b=2, c=3;
    A(&c, a);
    printf( %d, %d,%d\n", a, b, c);
}
```

saída: 1, 2, 1

## Transmissão de Parâmetro por Referência

O endereço do parâmetro real é avaliado e passado para o procedimento chamado, ficando inalterado até o fim da ativação. O parâmetro formal serve como referência indireta ao parâmetro real, logo, o conteúdo do parâmetro real pode ser alterado.

Ex:

```
Prog A;
   var      b: integer;
            c: ^integer;
   Proc X ( var y: integer; var z: ^integer);
   |  y := 1;
   |  new (z); z^:=2;
begin
      b := 0;
      c := &b;                  saída
      writeln (b,c^);            0, 0
      X (b,c);
      Writeln (b,c^);            1, 2
end.
```