INSTITUTO FEDERAL DE
EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
CEARÁ
CAMPUS DE CANINDÉ

INSTITUTO FEDERAL DE
EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
CEARÁ
CAMPUS DE CANINDÉ

Ernani Andrade Leite

ernani@ifce.edu.br

Aula 01 – Programação Estruturada

- Assunto: Revisão.
- Objetivos:

  - Apresentar os conceitos de subprogramação e sua aplicação no cotidiano;

  - Aplicação de algoritmos em situações do dia-a-dia;

  - Elaborar programas usando arquivos.

- Roteiro:

1. Introdução.
2. Definições e Conceitos.
3. Módularização.
4. Variáveis: Global X Local.
5. Parâmetros.
6. Exercícios.

INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
CEARÁ
CAMPUS DE CANINDÉ

Problemas na Produção de Software

- no “começo” do uso de computadores:
  - o custo da computação era o equipamento eletrônico
- no fim dos anos 60
  - a *crise do software* (programação estruturada)
- atualmente a situação se inverteu
  - computadores mais rápidos e baratos
  - tecnologia de software não apresentou um desenvolvimento comparável
- foi com o aumento da tecnologia e da capacidade dos computadores que problemas mais complexos puderam ser resolvidos pela máquina.

Histórico da Evolução das LP (1/2)

- Década de 40, programação física
  - linguagem de máquina

- Década de 50, programação lógica
  - linguagens montadoras, assembler (ainda exigiam conhecimentos do hardware)
  - linguagens de 1ª geração, abstração do hardware, ênfase em cálculos
    - Fortran, Algol 58, ...

Histórico da Evolução das LP (2/2)

- Década de 60, linguagens de 2ª geração, ênfase nos dados
  - Fortran, Algol 60, Cobol, LISP
- Década de 70, linguagens de 3ª geração, ênfase na estruturação do código
  - PL/1, Fortran, Algol 68, Cobol
  - Linguagem C
  - Pascal (ferramenta de aprendizagem)
  - Simula

A Crise do Software

- Problemas detectados na década 60:
  - baixa produtividade, falta de uma "metodologia formal" para o desenvolvimento de software, códigos literalmente sem a possibilidade de serem mantidos

- 1968, Dikjstra, lança os conceitos da Programação Estruturada (PE):
  - não usar goto, estruturas básicas de controle (seqüência, condição e repetição), subprogramação (ou modularização), tipo abstrato de dados = modelo matématico + operações.

Estruturas Básicas de Controle

- Seqüência, condição (ou seleção) e repetição
  - formas de raciocínio intuitivamente óbvias

- A legibilidade e compreensão de cada bloco de código é enormemente incrementada, proibindo o uso irrestrito de comandos de desvio (GOTO)

Subprogramação

- Em geral:
  - problemas complexos = programas complexos
- Mas sempre é possível dividir:
  - problemas grandes e complicados em problemas menores e de solução mais simples (na forma de um subprograma)
- Programa complexo = Subprograma 1 + Subprograma 2 + ... + Subprograma N

Definição

- Um **subprograma**, é um nome dado a um trecho de um programa mais complexo e que, em geral, encerra em si próprio um pedaço da solução de um problema maior (o programa a que ele está subordinado).

- São sinônimos: Procedimento, Função, Módulo (estrutura modular), Métodos (orientação a objetos) e Subrotina; e são conceitos da ciência conhecida como *engenharia de software*.

Método dos Refinamentos Sucessivos

- É uma "sistemática" de abordagem útil no projeto detalhado e na implementação de *softwares*.
- Partindo-se de um dado problema, para qual se deseja encontrar um programa de solução, deve-se procurar subdividi-lo em problemas menores (subproblemas) e de solução mais simples. Alguns destes problemas terão solução imediata e outros não.
- Os subproblemas para os quais não for possível encontrar uma solução direta devem ser novamente subdivididos.
- Assim, o processo é repetido até que se consiga encontrar um programa para solucionar cada um dos subproblemas definidos.
- Então, o programa de solução do problema original será composto pela justaposição dos programas usados para solucionar cada um dos subproblemas em que o problema original foi decomposto.

Em Síntese: Métodos dos Refinamentos Sucessivos

- dividir uma subrotina em outras tantas quantas forem necessárias, buscando uma solução mais simples de uma parte do problema maior

- ***técnica de programação estruturada***
  - ***dividir-para-conquistar***

Programação Estruturada

- a linguagem C é uma linguagem de programação estruturada

- à medida que os programas vão se tornando maiores e mais complexos, é possível simplificar e melhorar a clareza do programa dividindo o programa em partes menores, chamadas de *procedimentos* e *funções*.

Estrutura Modular

- É um programa maior e/ou complexo que é escrito em linguagem C e que é dividido, ou estruturado, em procedimentos e/ou funções (também chamados de módulos ou subprogramas)

- O corpo do programa principal é o “ponto inicial” da execução da estrutura modular

- Pode conter quantos procedimentos e/ou funções forem necessários e/ou convenientes

Mecanismo de Funcionamento (1/2)

- Um programa completo é dividido em **programa principal** e diversos **subprogramas** (tantos quantos forem necessários e/ou convenientes).

- O **programa principal** é aquele por onde a execução da aplicação sempre se inicia e ele pode eventualmente invocar ou chamar os demais subprogramas.

- O corpo do programa principal é sempre o último trecho do código de um programa em C. As definições dos subprogramas estão sempre colocadas na região de declaração do programa.

Mecanismo de Funcionamento (2/2)

Quando o C encontra a chamada de um procedimento ou de uma função, imediatamente *transfere o fluxo de execução para o subprograma chamado*, iniciando a execução a partir do primeiro comando do subprograma (primeira sentença após a cláusula **{**).

Depois que o último comando do subprograma (sentença imediatamente anterior a cláusula **}**) for executado, o Pascal *transfere a execução para o comando que segue imediatamente a chamada do subprograma*.

Definição de Procedimento e Função

- O **procedimento** é uma parte separada do programa, e somente é executado quando o seu nome é chamado dentro do programa principal ou por outro subprograma que esteja em execução.
- A **função** funciona de forma similar, com a diferença de que esta retorna com um valor através do seu nome.
- O nome de um subprograma é um identificador pelo qual ele será referenciado numa sentença de chamada.

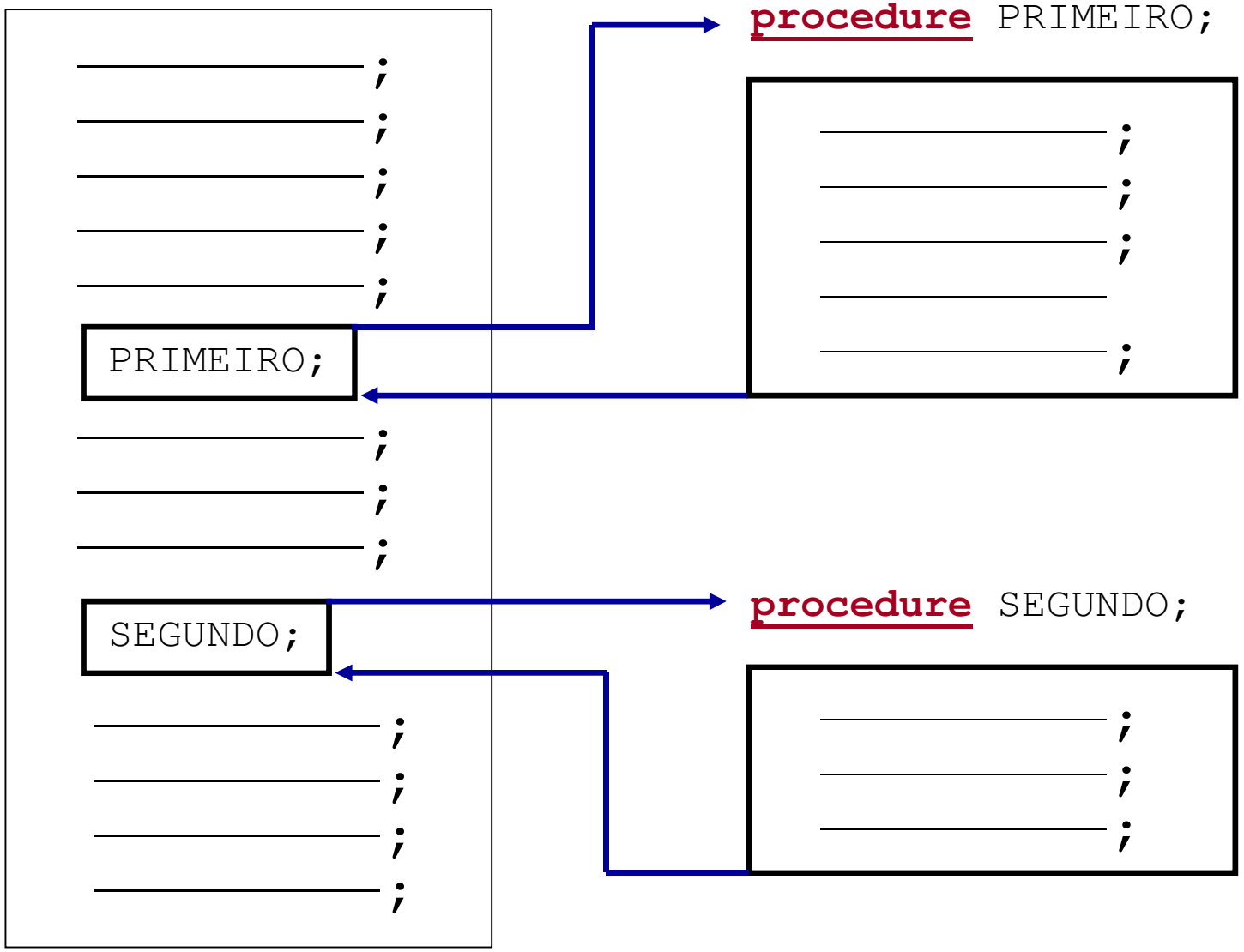
Estrutura de subprogramas

- Um **cabeçalho**, onde estão definidos o **nome** e o **tipo** do subprograma (*procedure* ou *function*), bem como os seus **parâmetros** e **declarações locais** (*const*, *type*, *var* e subprogramas subordinados);

- Um **corpo**, onde se encontram as instruções (comandos) do subprograma.

Manipulação de procedimentos

- O procedimento é uma parte separada do programa, e somente é executado quando o seu nome é chamado dentro do programa principal ou por outro subprograma que esteja em execução.
- O nome de um procedimento é um identificador pelo qual ele será referenciado numa sentença de chamada (ou ativação).
- Ao término do procedimento o controle de execução é devolvido ao módulo chamador.


program Principal;
PRIMEIRO;
SEGUNDO;
procedure PRIMEIRO;
procedure SEGUNDO;

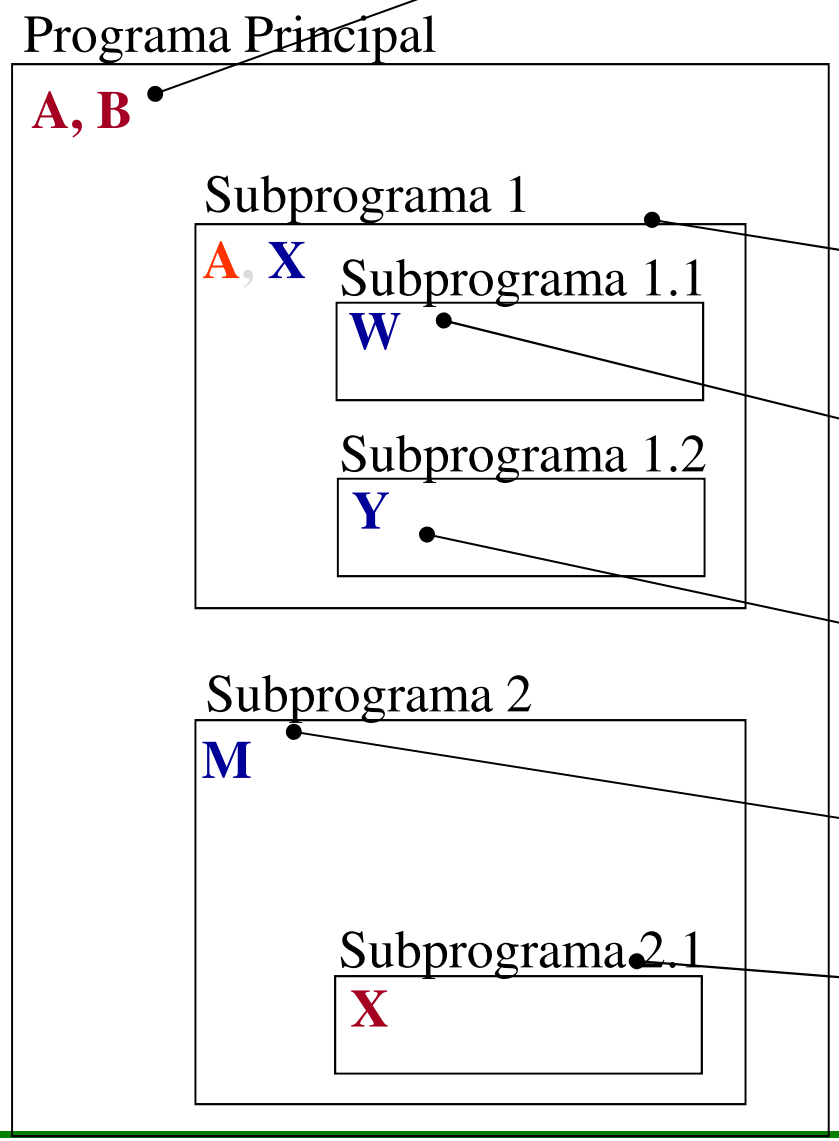
Manipulação de funções

- A função funciona de forma similar ao funcionamento de um procedimento, com a diferença de que uma função sempre retorna com um valor através do seu nome.
- A função é executada e, ao seu término, o trecho do comando que a invocou é substituído pelo valor da expressão atribuído ao nome da função.

Conceitos

- O **escopo** de uma variável ou sua **abrangência** está vinculada a sua visibilidade (Global ou Local) em relação aos subprogramas de um programa, sendo que a sua visibilidade está relacionada a sua hierarquia.
- Variável Global: pode ser usada pelo programa e por todos os subprogramas.
- Variável Local: pode ser usada somente pelo subprograma em que ela foi definida e pelos seus descendentes.

Escopo de Variáveis



as variáveis A e B, são globais aos subprogramas 1 e 2, porém dentro da subprograma 1, a variável A é definida novamente assumindo assim um contexto local para este subprograma

- variável global
A e X- variáveis locais

B, A e X- variáveis globais
W- variável local

B, A e X- variáveis globais
Y- variável local

A, B- variáveis globais
M- variável local

A, B e M- variáveis globais
X- variável local

Comunicação entre Subprogramas

- **Parâmetros** são canais pelos quais se estabelece uma comunicação bidirecional entre um subprograma e o módulo chamador (o programa principal ou outro subprograma).

- Dados são passados pelo módulo chamador ao subprograma, ou retornados por este ao primeiro por meio de parâmetros.

Parâmetros

- são variáveis opcionalmente passadas a um subprograma
- uma *procedure* ou *function* podem ter zero ou mais parâmetros
- são definidos no cabeçalho do subprograma (*procedure* ou *function*)

Tipos de parâmetros

- **Parâmetros Formais:**

  são os nomes simbólicos introduzidos no cabeçalho dos subprogramas. Dentro de um subprograma trabalha-se com estes nomes da mesma forma como se trabalha com variáveis locais ou globais.

- **Parâmetros Reais, ou efetivos (ou argumentos):**

  são aqueles que se associam aos *parâmetros formais* quando da chamada de um subprograma.

Finalidade dos parâmetros

- Através da passagem de parâmetros é feita a **transferência de informações** entre os módulos sejam: *constantes*, *variáveis*, ou *expressões*, ao invés de somente o valor de variáveis globais.

- Esta utilização formaliza a comunicação entre módulos.

- Existem dois tipos de passagem de parâmetros:
  - Tipo Valor ou Constante (cópia)
  - Tipo Variável ou Referência

Parâmetros por valor

- os parâmetros reais (módulo chamador) são calculados e os parâmetros formais (subprograma) correspondentes recebem uma cópia dos valores resultantes

- a variável passada se comporta como uma variável local, ou seja, alterações nos parâmetros formais **não afetam** os parâmetros reais

Parâmetros por referência (var)

- Os parâmetros reais (módulo chamador) compartilham seu espaço de memória com os parâmetros formais (subprograma).
- Portanto, alterações nos parâmetros formais **afetam** os parâmetros reais.
- O subprograma recebe uma *referência* à variável passada ou, em outras palavras, *a própria variável*.

Parâmetros por referência (var)

- Com esse tipo de passagem de parâmetros, o subprograma pode alterar *diretamente* a variável passada.

- Para especificar a passagem por referência, deve-se usar a palavra-chave **var** antes do nome do parâmetro.

Funcionamento de Subprogramas:

Os módulos utilizam objetos (variáveis, pôr exemplo) declarados em seu corpo, mas podem também utilizar os objetos declarados nos níveis mais externos, chamados globais.

O uso de variáveis globais dentro do subprograma serve para implementar um mecanismo de transmissão de informações de um nível mais externo para um mais interno.

A utilização de variáveis globais **não constitui**, no entanto, uma boa prática de programação (escopo muito grande). Assim, todo módulo ao ser implementado deve utilizar variáveis locais, e a transmissão de informações para dentro e fora dos procedimentos deve ser feita através dos parâmetros de transmissão.

As variáveis locais são criadas e alocadas quando da ativação e automaticamente liberadas quando do seu término.

Funcionamento de Subprogramas:

O subprograma (filho) é uma parte separada do módulo chamador (pai), e somente é executado quando o seu nome (identificador) for referenciado numa sentença de chamada. A execução do módulo se comporta como se o trecho do subprograma fosse copiado para o ponto onde foi invocado. Existem duas formas de chamada:

a) **Nome_do_Subprograma;**

b) **Nome_do_Subprograma(Lista_de_Parâmetros_Reais);**

Os parâmetros reais na lista são separados por vírgula.
O 1º parâmetro real é associado ao 1º parâmetro formal;
O 2º parâmetro real é associado ao 2º parâmetro formal
e assim por diante.

O número de parâmetros reais deve ser igual ao número e ao tipo dos parâmetros formais.

INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
CEARÁ
CAMPUS DE CANINDÉ

Funcionamento de Subprogramas:

Ativação de Subprogramas:
quando um subprograma é chamado para execução, três tarefas são executadas:
a) criação de um espaço para os objetos (variáveis) locais;
b) associação e passagem de parâmetros;
c) transferência de controle para o subprograma (módulo chamado).

Retorno de Subprogramas:
quando a execução do subprograma é concluída, três tarefas são executadas:
a) endereço de retorno é restabelecido;
b) área do procedimento chamado é liberada;
c) transferência de controle para o endereço de retorno (módulo chamador).

Funcionamento de Subprogramas:

A área de memória usada na execução de um programa varia dinamicamente durante a execução. A memória é representada graficamente como sendo uma ***pilha*** de ***caixa de variáveis***.

Cada uma destas caixas contém, pôr sua vez, um escaninho para cada uma das variáveis locais (ou parâmetros).

Sempre que um subprograma é chamado, uma caixa contendo espaço para as variáveis é colocada no topo da pilha.

Quando o algoritmo principal é iniciado a pilha contém uma caixa para as variáveis globais.

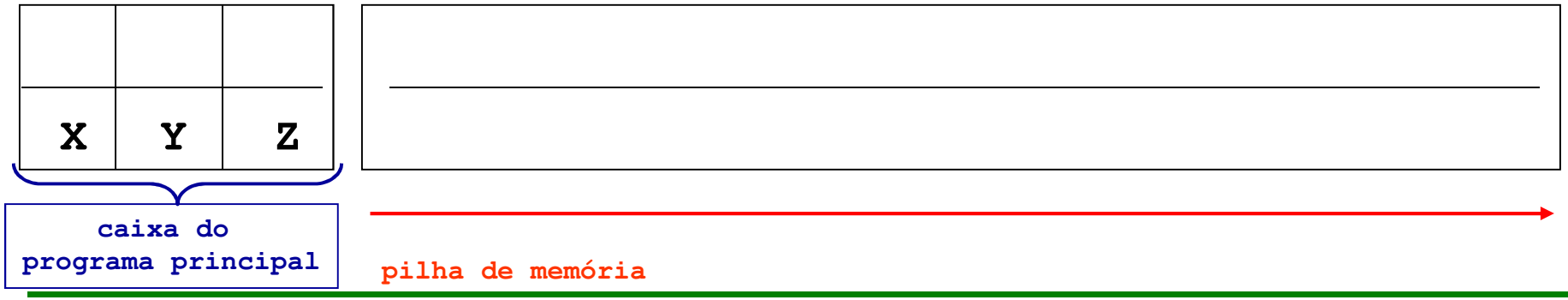
Ao terminar um subprograma, sua caixa de variáveis é automaticamente retirada da pilha.

```
Program P;
var X, Y, Z : integer;

Procedure Soma(A, B: integer; var C: integer);
begin
  C := A + B;
end;

begin
  X := 2;    Y := 3;
  Soma(X, Y, Z);
  writeln('Soma = ', Z);
end.
```

I) situação inicial



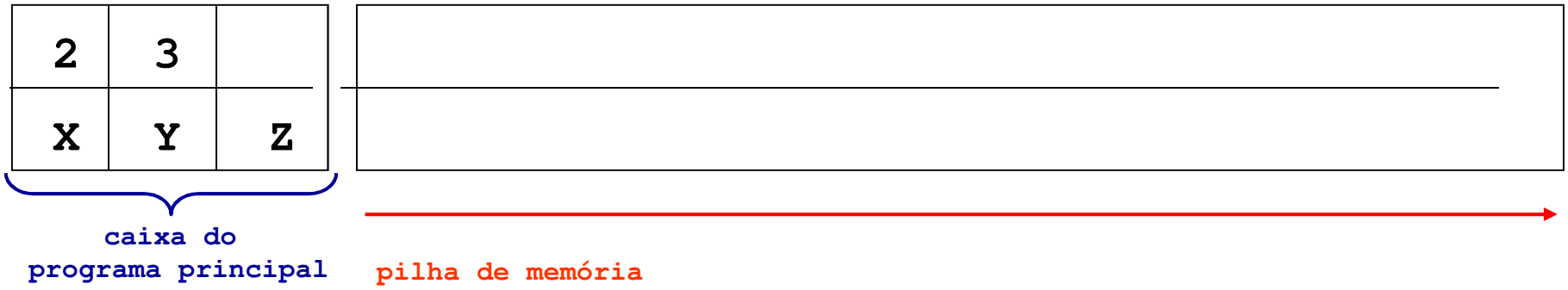
```
Program P;
var X, Y, Z : integer;

Procedure Soma(A, B: integer; var C: integer);
begin
  C := A + B;
end;

begin
  X := 2; Y := 3;
  Soma(X, Y, Z);
writeln('Soma = ', Z);
end.
```

II

II) ao fazer as atribuições



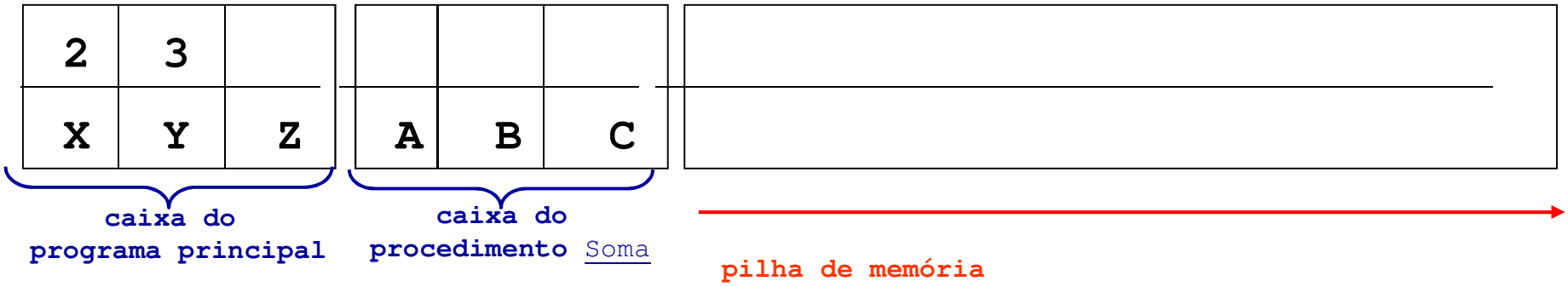
```
                Program P;
            var X, Y, Z : integer;

Procedure Soma(A, B: integer; var C: integer);
                  begin
                   C := A + B;
                  end;

                  begin
            X := 2;     Y := 3;
              Soma(X, Y, Z);
          writeln('Soma = ', Z);
                  end.
```

III

III) ao chamar o procedimento Soma



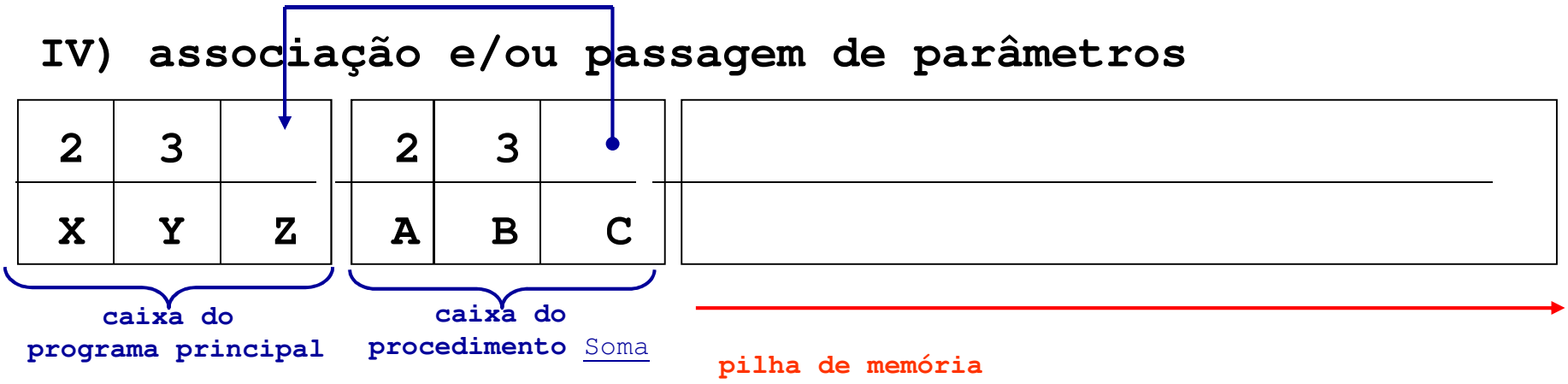
```
                Program P;
              var X, Y, Z : integer;

Procedure Soma(A, B: integer; var C: integer);
                  begin
                    C := A + B;
                  end;

                  begin
                X := 2;     Y := 3;
                  Soma(X, Y, Z);
              writeln('Soma = ', Z);
                  end.
```

IV

IV) associação e/ou passagem de parâmetros

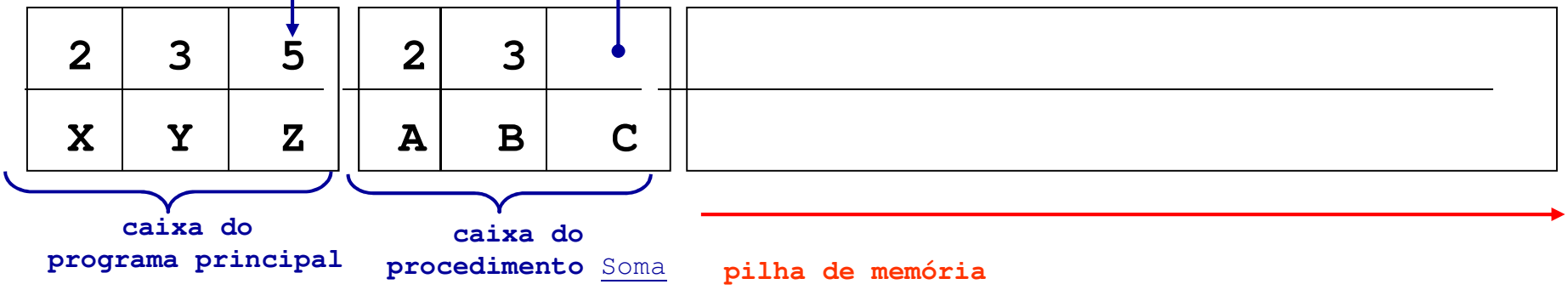


```
                Program P;
            var X, Y, Z : integer;

Procedure Soma(A, B: integer; var C: integer);
                  begin
               V  C := A + B;
                  end;

                  begin
          X := 2;     Y := 3;
             Soma(X, Y, Z);
      writeln('Soma = ', Z);
                  end.
```

V) logo antes do final do procedimento Soma

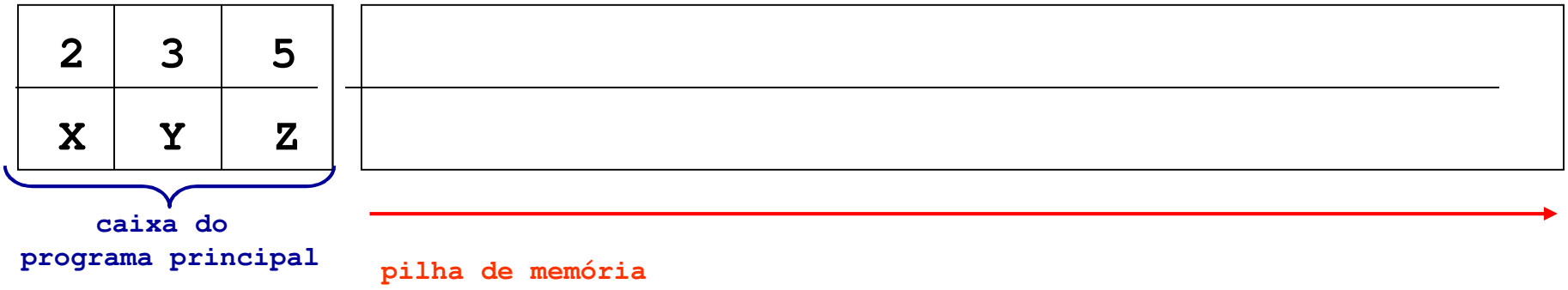


```
                Program P;
            var X, Y, Z : integer;

Procedure Soma(A, B: integer; var C: integer);
                  begin
                   C := A + B;
                  end;

                  begin
              X := 2;     Y := 3;
                Soma(X, Y, Z);
          writeln('Soma VI ', Z);
                  end.
```

VI) logo antes do final do programa P



Em Síntese:

parâmetros: estabelecem um canal de comunicação entre os subprogramas (ou módulos).

Módulo Chamador (Pai)

. variáveis globais
. argumentos ou paramêtros efetivos ou reais

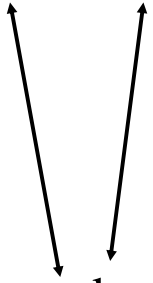
parada(1, 24);

passagem de parâmetros por valor

procedure parada(x, y: *integer*);

swap(a[i], a[min]);

passagem de parâmetros por referência



procedure swap(**var** a, b: *integer*);

Módulo Chamado (Filho)

. variáveis locais
. parâmetros formais

Vantagens da Subprogramação

- subdivisão de programas complexos
  - cada parte menor tem um código mais simples
  - facilita o entendimento (partes independentes)
- estruturação de programas
  - detecção de erros e documentação de sistemas
- modularização de sistemas
  - desenvolvimento por equipes de programadores
  - manutenção de software
  - reutilização de subprogramas (bibliotecas de subprogramas)

A questão é: "Reutilização de Software"

Objetivo:

Economia de tempo e trabalho.

Princípio:

Um conjunto de subprogramas destinado a solucionar uma série de tarefas bastante corriqueiras é desenvolvido e vai sendo aumentado com o passar do tempo, com o acréscimo de novos subprogramas. A este conjunto dá-se o nome de **biblioteca**. No desenvolvimento de novos sistemas, procura-se ao máximo basear sua concepção em subprogramas já existentes na biblioteca, de modo que a quantidade de software realmente novo que deve ser desenvolvido é minimizada.