Construção de Compiladores
Período Especial
Aula 10: Chamadas de Procedimento Sem Parâmetros, Com Variáveis Locais

Bruno Müller Junior

Departamento de Informática
UFPR

2020

Sintaxe

- A regra de declaração de um procedimento corresponde à regra 12.

```
2.  <bloco> ::=  ...                                        |
                 <parte de declaração de subrotinas>        |
                 ...
11.  <parte de declaração de subrotinas> ::= <declaração de procedimento> |
                                             <declaração de função>
12. <declaração de procedimento> ::= PROCEDURE  <ident> [<parâmetros formais>] ; <bloco>
```

Escopo e visibilidade

- Em tempo de compilação define-se a visibilidade de cada procedimento.
  - Na linguagem Pascal existe a questão dos vários níveis léxicos, onde um procedimento de nível léxico k "vê" todos os procedimentos que:
    1. tem nível léxico menor ou igual a k nos quais ele está encaixado;
    2. tem nível léxico $k + 1$ encaixados nele.
- Em tempo de execução, uma chamada de procedimento consistem em:
  1. desviar o fluxo de execução para um escopo nomeado (o nome do procedimento);
  2. ao final do procedimento, retornar o fluxo para o comando seguinte daquele onde foi feita a chamada.

Tempo de compilação

```
program escopoProc (input, output);
|   procedure p;
|   |  procedure q;
|   |  | procedure r;
|   |  | | begin (*  r *) end
|   |  | procedure s;
|   |  | | begin (*  s *) end
|   |  | begin (*  q *) end
|   | begin (*  p *) end
|   procedure t;
|   |  procedure u;
|   |  | begin (*  u *) end
|   |  procedure v;
|   |  | begin (*  v *) end
|   | begin (*  t *) end
| begin (*  principal *) end
```

Tempo de compilação

```
program escopoProc (input, output);
|   procedure p;
|   |  procedure q;
|   |  | procedure r;
|   |  | | begin (*  r *) end  (vê p, q, r)
|   |  | procedure s;
|   |  | | begin (*  s *) end
|   |  | begin (*  q *) end
|   | begin (*  p *) end
|   procedure t;
|   |  procedure u;
|   |  | begin (*  u *) end
|   |  procedure v;
|   |  | begin (*  v *) end
|   | begin (*  t *) end
| begin (*  principal *) end
```

Tempo de compilação

```
program escopoProc (input, output);
|   procedure p;
|   |  procedure q;
|   |  | procedure r;
|   |  | | begin (*  r *) end  (vê p, q, r)
|   |  | procedure s;
|   |  | | begin (*  s *) end  (vê p, q, r, s)
|   |  | begin (*  q *) end
|   | begin (*  p *) end
|   procedure t;
|   |  procedure u;
|   |  | begin (*  u *) end
|   |  procedure v;
|   |  | begin (*  v *) end
|   | begin (*  t *) end
| begin (*  principal *) end
```

Tempo de compilação

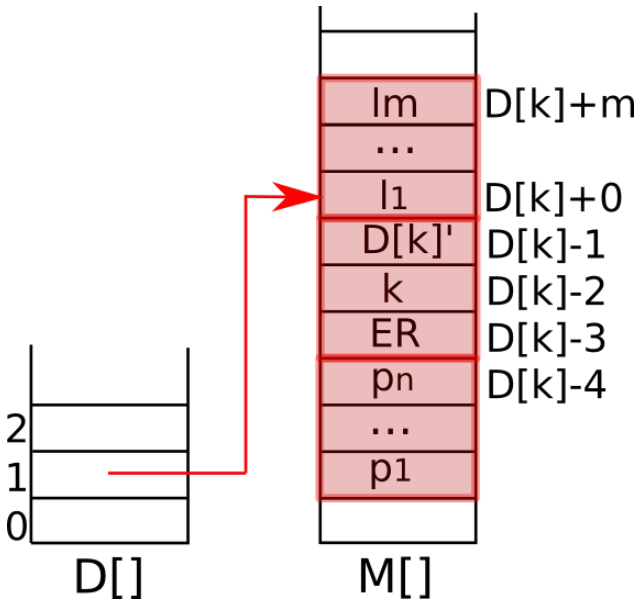
```
program escopoProc (input, output);
|   procedure p;
|   |  procedure q;
|   |  | procedure r;
|   |  | | begin (*  r *) end  (vê p, q, r)
|   |  | procedure s;
|   |  | | begin (*  s *) end  (vê p, q, r, s)
|   |  | begin (*  q *) end   (vê p, q, r, s)
|   | begin (*  p *) end      (vê p, q)
|   procedure t;
|   |  procedure u;
|   |  | begin (*  u *) end   (vê p, t, u)
|   |  procedure v;
|   |  | begin (*  v *) end   (vê p, t, u, v)
|   | begin (*  t *) end      (vê p, t, u, v)
| begin (*  principal *) end  (vê p, t)
```

Tempo de Execução

- Em tempo de execução, cada chamada de procedimento é associado a uma estrutura chamada "Registro de Ativação", que contém informações como:
  - parâmetros e variáveis locais;
  - ponto do código para onde voltar (endereço de retorno);
  - endereço de base;
- Existem várias formas de implementar um R.A.
- Em Pascal, um procedimento de nível léxico k precisa ter acesso a todas as variáveis de nível léxico menor que k (preservados os "encaixamentos" dos procedimentos).

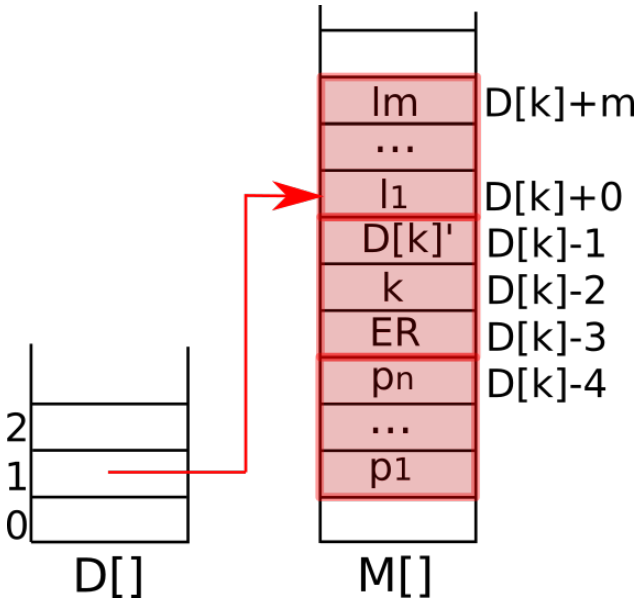
R.A da MEPA

- Um R.A. é dividido em três partes: variáveis locais, informações gerenciais e parâmetros.
- O esquema ao lado corresponde a um R.A. genérico para um procedimento de nível léxico 1, onde:
  - assinatura: $proc(p_1, p_2, \ldots, p_n)$
  - variáveis locais: $l_1, l_2, \ldots, l_n$



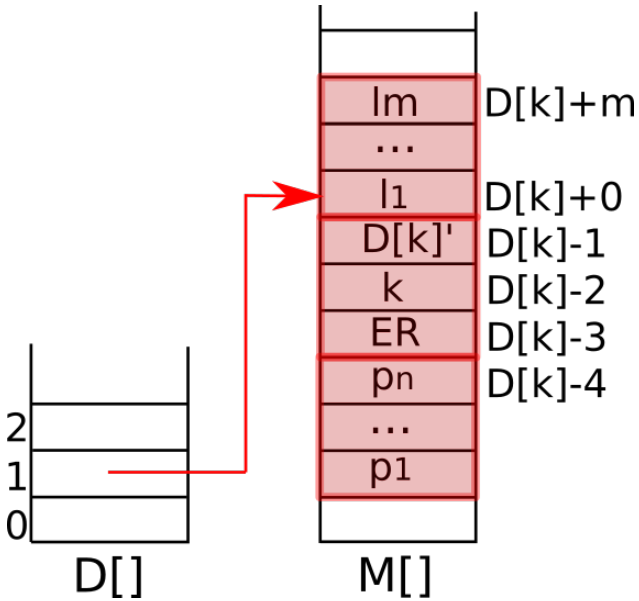
R.A da MEPA

- variáveis locais:
  - Cada D[k] aponta para a primeira variável local do R.A. de nível k.
  - Assim o endereço léxico da primeira variável local é `CRLV k,0`



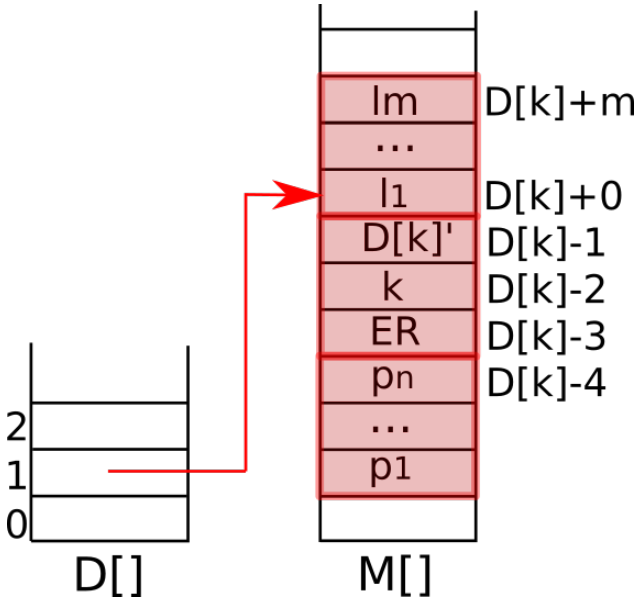
R.A da MEPA

- informações gerenciais:
  - Endereço de Retorno (ER) em $D[k] - 3$
  - Nível léxico do chamador (k) em $D[k] - 2$
  - conteúdo anterior de D[k] em $D[k] - 1$



R.A da MEPA

- parâmetros:
  - Os parâmetros estão presentes a partir de $D[k] - 4$ (`CRLV k,-4`).
  - Em $D[k] - 4$ está o último parâmetro.



Tempo de Execução

- Este esquema é implementado no uso de três instruções novas.

Instrução	Ação	Significado

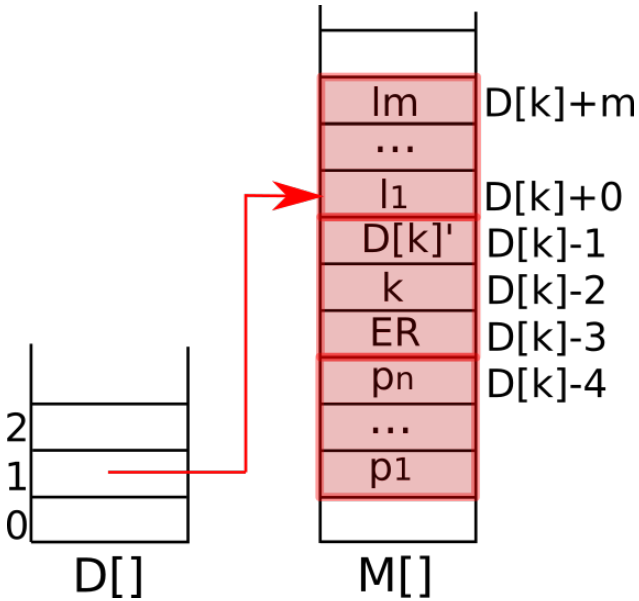
CHPR p,k	`M[s+1]:=i+1`
	`M[s+2]:=nivel lex. atual`
	`s:=s+2`
	`i:=p`
		Chama Procedimento

RTPR k,n	`D[k]:=M[s]`
	`i:=M[s-2]`
	`s:=s-(n+3)`
		Retorna de Procedimento

ENPR k	`s:=s+1`
	`M[s]:=D[k]`
	`D[k]:=s+1`
		Entra em Procedimento

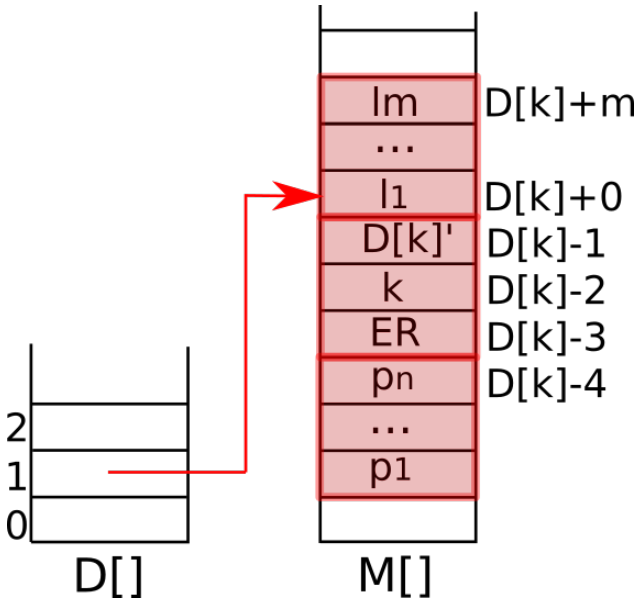
Funcionamento

- A cada chamada de procedimento, cria R.A. daquele procedimento:
  1. empilha parâmetros;
  2. insere parte das I.G. e desvia fluxo (CHPR);
  3. completa I.G. (ENPR);
  4. Aloca variáveis locais (AMEM);



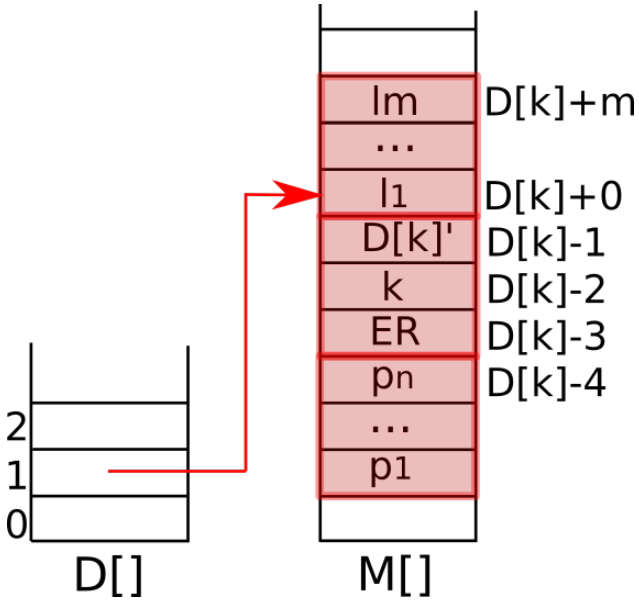
Funcionamento

- A cada saída de procedimento, destrói R.A. daquele procedimento:
  1. libera variáveis locais (DMEM);
  2. libera I.G. Ajustando D[k] (RTPR k,n);
  3. desempilha parâmetros (RTPR k,n);



Tempo de Execução

- para explicar a tradução e funcionamento das chamadas de procedimento utilizaremos várias aulas:
  10. Sem parâmetros e com variáveis locais;
  11. Com parâmetros passados por valor (cópia);
  12. Com parâmetros passados por referência;
  13. Funções;



Esquema de Tradução

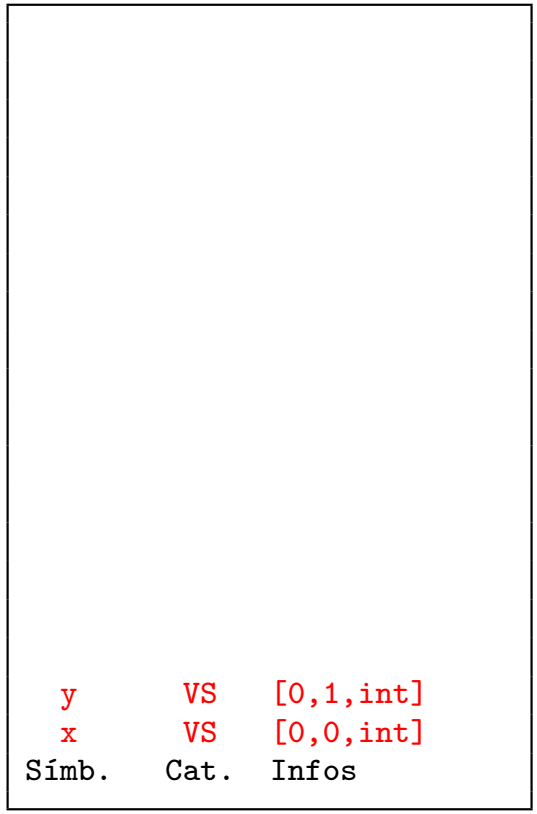
PROCEDURE p	`DSVS R00` `R01:ENPR k`
VAR ... ... BEGIN	*Traduz bloco*
END	*RTPR k, n*
... p; ...	*Empilha Parâmetros* `CHPR P01, k`

```
program proc1 (input, output);
  var x, y: integer;
  procedure p;
     var z:integer;
     begin
       z:=x;
       x:=x-1;
       if (z>1)
         then p
         else y:=1;
       y:=y*z
    end
begin
   read(x);
   p
   write (x,y)
end.
```

```
program proc1 (input, output);          INPP
  var x, y: integer;
  procedure p;
     var z:integer;
     begin
       z:=x;
       x:=x-1;
       if (z>1)
         then p
         else y:=1;
       y:=y*z
    end
begin
   read(x);
   p
   write (x,y)
end.
```

```
program proc1 (input, output);
  var x, y: integer;
  procedure p;
     var z:integer;
     begin
       z:=x;
       x:=x-1;
       if (z>1)
         then p
         else y:=1;
       y:=y*z
    end
begin
   read(x);
   p
   write (x,y)
end.
```

```
INPP
AMEM 2
```



Procedimento

```
program proc1 (input, output);
  var x, y: integer;
  procedure p;
     var z:integer;
     begin
       z:=x;
       x:=x-1;
       if (z>1)
         then p
         else y:=1;
       y:=y*z
    end
begin
   read(x);
   p
   write (x,y)
end.
```

```
        INPP
        AMEM 2
        DSVS R00
R01:ENPR 1
```

```
  p      PROC [1,R01,0{}]
  y      VS   [0,1,?]
  x      VS   [0,0,?]
Símb.   Cat.  Infos
```

Procedimento

```
program proc1 (input, output);
  var x, y: integer;
  procedure p;
     var z:integer;
     begin
       z:=x;
       x:=x-1;
       if (z>1)
         then p
         else y:=1;
       y:=y*z
    end
begin
   read(x);
   p
   write (x,y)
end.
```

```
      INPP
      AMEM 2
      DSVS R00
R01:ENPR 1
      AMEM 1
```

```
  z       VS    [1,0,int]
  p       PROC  [1,R01,0{}]
  y       VS    [0,1,int]
  x       VS    [0,0,int]
Símb.   Cat.  Infos
```

Procedimento

```
program proc1 (input, output);
  var x, y: integer;
  procedure p;
     var z:integer;
     begin
       z:=x;
       x:=x-1;
       if (z>1)
         then p
         else y:=1;
       y:=y*z
    end
begin
   read(x);
   p
   write (x,y)
end.
```

```
      INPP
      AMEM 2
      DSVS R00
R01:ENPR 1
      AMEM 1
```

```
  z       VS    [1,0,int]
  p       PROC  [1,R01,0{}]
  y       VS    [0,1,int]
  x       VS    [0,0,int]
Símb.   Cat.  Infos
```

Procedimento

```
program proc1 (input, output);
  var x, y: integer;
  procedure p;
     var z:integer;
     begin
       z:=x;
       x:=x-1;
       if (z>1)
         then p
         else y:=1;
       y:=y*z
    end
begin
   read(x);
   p
   write (x,y)
end.
```

```
      INPP
      AMEM 2
      DSVS R00
R01:ENPR 1
      AMEM 1
      CRVL 0,0
      ARMZ 1,0
```

Símb.	Cat.	Infos
x	VS	[0,0,int]
y	VS	[0,1,int]
p	PROC	[1,R01,0{}]
z	VS	[1,0,int]

Procedimento

```
program proc1 (input, output);
  var x, y: integer;
  procedure p;
     var z:integer;
     begin
       z:=x;
       x:=x-1;
       if (z>1)
         then p
         else y:=1;
       y:=y*z
    end
begin
   read(x);
   p
   write (x,y)
end.
```

```
      INPP
      AMEM 2
      DSVS R00
R01:ENPR 1
      AMEM 1
      CRVL 0,0
      ARMZ 1,0
      CRVL 0,0
      CRCT 1
      SUBT
      ARMZ 0,0
```

```
  z       VS    [1,0,int]
  p       PROC  [1,R01,0{}]
  y       VS    [0,1,int]
  x       VS    [0,0,int]
Símb.   Cat.   Infos
```

Procedimento

```
program proc1 (input, output);
  var x, y: integer;
  procedure p;
     var z:integer;
     begin
       z:=x;
       x:=x-1;
       if (z>1)
         then p
         else y:=1;
       y:=y*z
    end
begin
   read(x);
   p
   write (x,y)
end.
```

```
      INPP
      AMEM 2
      DSVS R00
R01:ENPR 1
      AMEM 1
      CRVL 0,0
      ARMZ 1,0
      CRVL 0,0
      CRCT 1
      SUBT
      ARMZ 0,0
      CRVL 1,0
      CRCT 1
      CMMA
      DSVF R02
```

```
  z       VS    [1,0,int]
  p       PROC  [1,R01,0{}]
  y       VS    [0,1,int]
  x       VS    [0,0,int]
Símb.   Cat.   Infos
```

Procedimento

```
program proc1 (input, output);
  var x, y: integer;
  procedure p;
     var z:integer;
     begin
       z:=x;
       x:=x-1;
       if (z>1)
         then p
         else y:=1;
       y:=y*z
    end
begin
   read(x);
   p
   write (x,y)
end.
```

```
     INPP
     AMEM 2
     DSVS R00
R01:ENPR 1
     AMEM 1
     CRVL 0,0
     ARMZ 1,0
     CRVL 0,0
     CRCT 1
     SUBT
     ARMZ 0,0
     CRVL 1,0
     CRCT 1
     CMMA
     DSVF R02
     CHPR R01,1
```

```
  z      VS    [1,0,int]
  p      PROC  [1,R01,0{}]
  y      VS    [0,1,int]
  x      VS    [0,0,int]
Símb.   Cat.   Infos
```

Procedimento

```
program proc1 (input, output);
  var x, y: integer;
  procedure p;
     var z:integer;
     begin
       z:=x;
       x:=x-1;
       if (z>1)
         then p
         else y:=1;
       y:=y*z
    end
begin
   read(x);
   p
   write (x,y)
end.
```

```
     INPP
     AMEM 2
     DSVS R00
R01:ENPR 1
     AMEM 1
     CRVL 0,0
     ARMZ 1,0
     CRVL 0,0
     CRCT 1
     SUBT
     ARMZ 0,0
     CRVL 1,0
     CRCT 1
     CMMA
     DSVF R02
     CHPR R01,1
     DSVS R03
R02:NADA
```

```
  z       VS    [1,0,int]
  p       PROC  [1,R01,0{}]
  y       VS    [0,1,int]
  x       VS    [0,0,int]
Símb.   Cat.   Infos
```

Procedimento

```
program proc1 (input, output);
  var x, y: integer;
  procedure p;
     var z:integer;
     begin
       z:=x;
       x:=x-1;
       if (z>1)
         then p
         else y:=1;
       y:=y*z
    end
begin
   read(x);
   p
   write (x,y)
end.
```

```
     INPP       R03:NADA
     AMEM 2
     DSVS R00
R01:ENPR 1
     AMEM 1
     CRVL 0,0
     ARMZ 1,0
     CRVL 0,0
     CRCT 1
     SUBT
     ARMZ 0,0
     CRVL 1,0
     CRCT 1
     CMMA
     DSVF R02
     CHPR R01,1
     DSVS R03
R02:NADA
     CRCT 1
     ARMZ 0,1
```

```
  z       VS    [1,0,int]
  p       PROC  [1,R01,0{}]
  y       VS    [0,1,int]
  x       VS    [0,0,int]
Símb.   Cat.   Infos
```

Procedimento

```
program proc1 (input, output);
  var x, y: integer;
  procedure p;
     var z:integer;
     begin
       z:=x;
       x:=x-1;
       if (z>1)
         then p
         else y:=1;
       y:=y*z
    end
begin
   read(x);
   p
   write (x,y)
end.
```

```
     INPP       R03:NADA
     AMEM 2          CRVL 0,1
     DSVS R00        CRVL 1,0
R01:ENPR 1           MULT
     AMEM 1          ARMZ 0,1
     CRVL 0,0
     ARMZ 1,0
     CRVL 0,0
     CRCT 1
     SUBT
     ARMZ 0,0
     CRVL 1,0
     CRCT 1
     CMMA
     DSVF R02
     CHPR R01,1
     DSVS R03
R02:NADA
     CRCT 1
     ARMZ 0,1
```

```
  z       VS    [1,0,int]
  p       PROC  [1,R01,0{}]
  y       VS    [0,1,int]
  x       VS    [0,0,int]
Símb.   Cat.   Infos
```

Procedimento

```
program proc1 (input, output);
  var x, y: integer;
  procedure p;
     var z:integer;
     begin
       z:=x;
       x:=x-1;
       if (z>1)
         then p
         else y:=1;
       y:=y*z
    end
begin
   read(x);
   p
   write (x,y)
end.
```

```
     INPP       R03:NADA
     AMEM 2          CRVL 0,1
     DSVS R00        CRVL 1,0
R01:ENPR 1           MULT
     AMEM 1          ARMZ 0,1
     CRVL 0,0        DMEM 1
     ARMZ 1,0        RTPR 1,0
     CRVL 0,0
     CRCT 1
     SUBT
     ARMZ 0,0
     CRVL 1,0
     CRCT 1
     CMMA
     DSVF R02
     CHPR R01,1
     DSVS R03
R02:NADA
     CRCT 1
     ARMZ 0,1
```

```
  p       PROC [1,R01,0{}]
  y       VS   [0,1,int]
  x       VS   [0,0,int]
Símb.   Cat.   Infos
```

Programa Principal

```
program proc1 (input, output);
  var x, y: integer;
(* *)
  procedure p;
     var z:integer;
     begin
       z:=x;
       x:=x-1;
       if (z>1)
         then p
         else y:=1;
       y:=y*z
    end
begin
   read(x);
   p
   write (x,y)
end.
```

```
      INPP       R03:NADA
      AMEM 2          CRVL 0,1
      DSVS R00        CRVL 1,0
R01:ENPR 1            MULT
      AMEM 1          ARMZ 0,1
      CRVL 0,0        DMEM 1
      ARMZ 1,0        RTPR 1,0
      CRVL 0,0  R00:NADA
      CRCT 1
      SUBT
      ARMZ 0,0
      CRVL 1,0
      CRCT 1
      CMMA
      DSVF R02
      CHPR R01,1
      DSVS R03
R02:NADA
      CRCT 1
      ARMZ 0,1
```

```
  p       PROC [1,R01,0{}]
  y       VS   [0,1,int]
  x       VS   [0,0,int]
Símb.   Cat.   Infos
```

Programa Principal

```
program proc1 (input, output);
  var x, y: integer;
  procedure p;
     var z:integer;
     begin
       z:=x;
       x:=x-1;
       if (z>1)
         then p
         else y:=1;
       y:=y*z
    end
begin
   read(x);
   p
   write (x,y)
end.
```

```
     INPP       R03:NADA
     AMEM 2          CRVL 0,1
     DSVS R00        CRVL 1,0
R01:ENPR 1           MULT
     AMEM 1          ARMZ 0,1
     CRVL 0,0        DMEM 1
     ARMZ 1,0        RTPR 1,0
     CRVL 0,0   R00:NADA
     CRCT 1          LEIT
     SUBT            ARMZ 0,0
     ARMZ 0,0
     CRVL 1,0
     CRCT 1
     CMMA
     DSVF R02
     CHPR R01,1
     DSVS R03
R02:NADA
     CRCT 1
     ARMZ 0,1
```

```
  p       PROC [1,R01,0{}]
  y       VS   [0,1,int]
  x       VS   [0,0,int]
Símb.   Cat.   Infos
```

Programa Principal

```
program proc1 (input, output);
  var x, y: integer;
  procedure p;
     var z:integer;
     begin
       z:=x;
       x:=x-1;
       if (z>1)
         then p
         else y:=1;
       y:=y*z
    end
begin
   read(x);
   p
   write (x,y)
end.
```

```
     INPP       R03:NADA
     AMEM 2          CRVL 0,1
     DSVS R00        CRVL 1,0
R01:ENPR 1           MULT
     AMEM 1          ARMZ 0,1
     CRVL 0,0        DMEM 1
     ARMZ 1,0        RTPR 1,0
     CRVL 0,0   R00:NADA
     CRCT 1          LEIT
     SUBT            ARMZ 0,0
     ARMZ 0,0        CHPR R01,0
     CRVL 1,0
     CRCT 1
     CMMA
     DSVF R02
     CHPR R01,1
     DSVS R03
R02:NADA
     CRCT 1
     ARMZ 0,1
```

```
  p       PROC [1,R01,0{}]
  y       VS   [0,1,int]
  x       VS   [0,0,int]
Símb.   Cat.   Infos
```

Programa Principal

```
program proc1 (input, output);
  var x, y: integer;
  procedure p;
     var z:integer;
     begin
       z:=x;
       x:=x-1;
       if (z>1)
         then p
         else y:=1;
       y:=y*z
    end
begin
   read(x);
   p
   write (x,y)
end.
```

```
     INPP       R03:NADA
     AMEM 2          CRVL 0,1
     DSVS R00        CRVL 1,0
R01:ENPR 1           MULT
     AMEM 1          ARMZ 0,1
     CRVL 0,0        DMEM 1
     ARMZ 1,0        RTPR 1,0
     CRVL 0,0   R00:NADA
     CRCT 1          LEIT
     SUBT            ARMZ 0,0
     ARMZ 0,0        CHPR R01,0
     CRVL 1,0        CRVL 0,0
     CRCT 1          IMPR
     CMMA            CRVL 0,1
     DSVF R02        IMPR
     CHPR R01,1
     DSVS R03
R02:NADA
     CRCT 1
     ARMZ 0,1
```

```
  p       PROC [1,R01,0{}]
  y       VS   [0,1,int]
  x       VS   [0,0,int]
Símb.   Cat.   Infos
```

Programa Principal

```
program proc1 (input, output);
  var x, y: integer;
  procedure p;
     var z:integer;
     begin
       z:=x;
       x:=x-1;
       if (z>1)
         then p
         else y:=1;
       y:=y*z
    end
begin
   read(x);
   p
   write (x,y)
end.
```

```
     INPP       R03:NADA
     AMEM 2         CRVL 0,1
     DSVS R00       CRVL 1,0
R01:ENPR 1          MULT
     AMEM 1         ARMZ 0,1
     CRVL 0,0       DMEM 1
     ARMZ 1,0       RTPR 1,0
     CRVL 0,0   R00:NADA
     CRCT 1         LEIT
     SUBT           ARMZ 0,0
     ARMZ 0,0       CHPR R01,0
     CRVL 1,0       CRVL 0,0
     CRCT 1         IMPR
     CMMA           CRVL 0,1
     DSVF R02       IMPR
     CHPR R01,1     DMEM 2
     DSVS R03
R02:NADA
     CRCT 1
     ARMZ 0,1
```

~~p PROC [1,R01,0{}]~~
~~y VS [0,1,int]~~
~~x VS [0,0,int]~~
Símb. Cat. Infos

Programa Principal

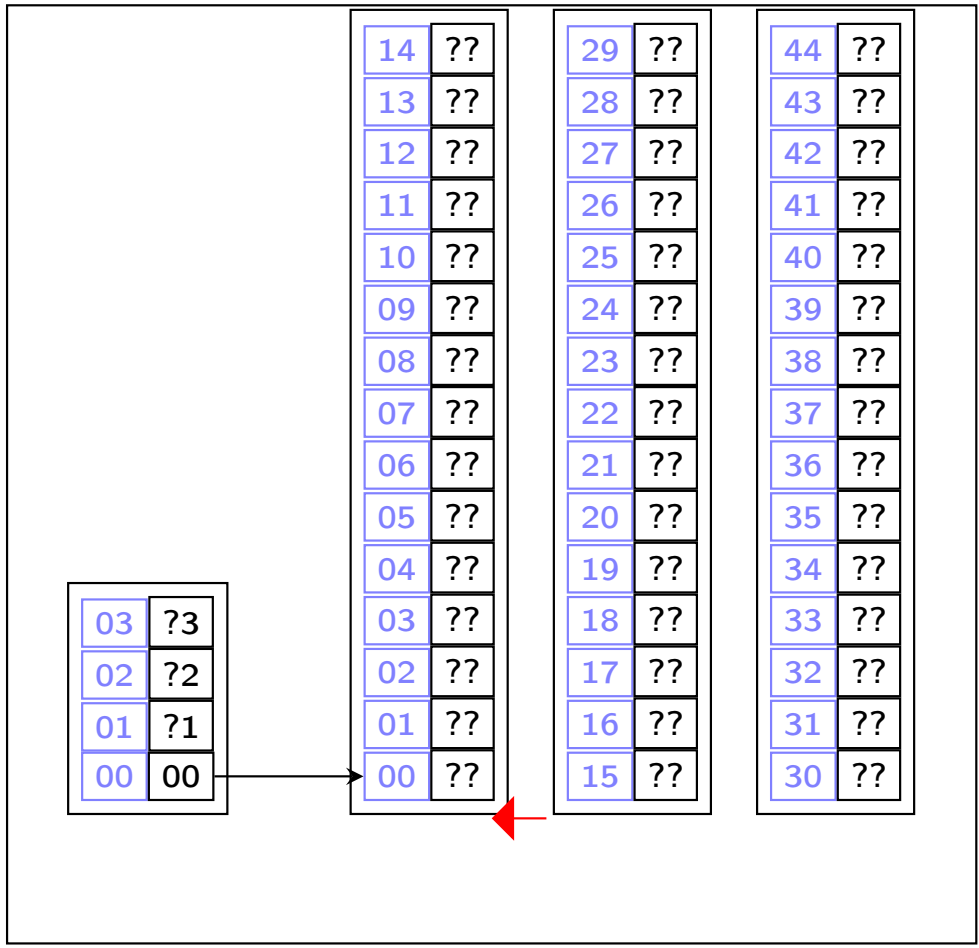
```
program proc1 (input, output);
  var x, y: integer;
  procedure p;
     var z:integer;
     begin
       z:=x;
       x:=x-1;
       if (z>1)
         then p
         else y:=1;
       y:=y*z
    end
begin
   read(x);
   p
   write (x,y)
end.
```

```
     INPP       R03:NADA
     AMEM 2          CRVL 0,1
     DSVS R00        CRVL 1,0
R01:ENPR 1           MULT
     AMEM 1          ARMZ 0,1
     CRVL 0,0        DMEM 1
     ARMZ 1,0        RTPR 1,0
     CRVL 0,0   R00:NADA
     CRCT 1          LEIT
     SUBT            ARMZ 0,0
     ARMZ 0,0        CHPR R01,0
     CRVL 1,0        CRVL 0,0
     CRCT 1          IMPR
     CMMA            CRVL 0,1
     DSVF R02        IMPR
     CHPR R01,1      DMEM 2
     DSVS R03        PARA
R02:NADA
     CRCT 1
     ARMZ 0,1
```

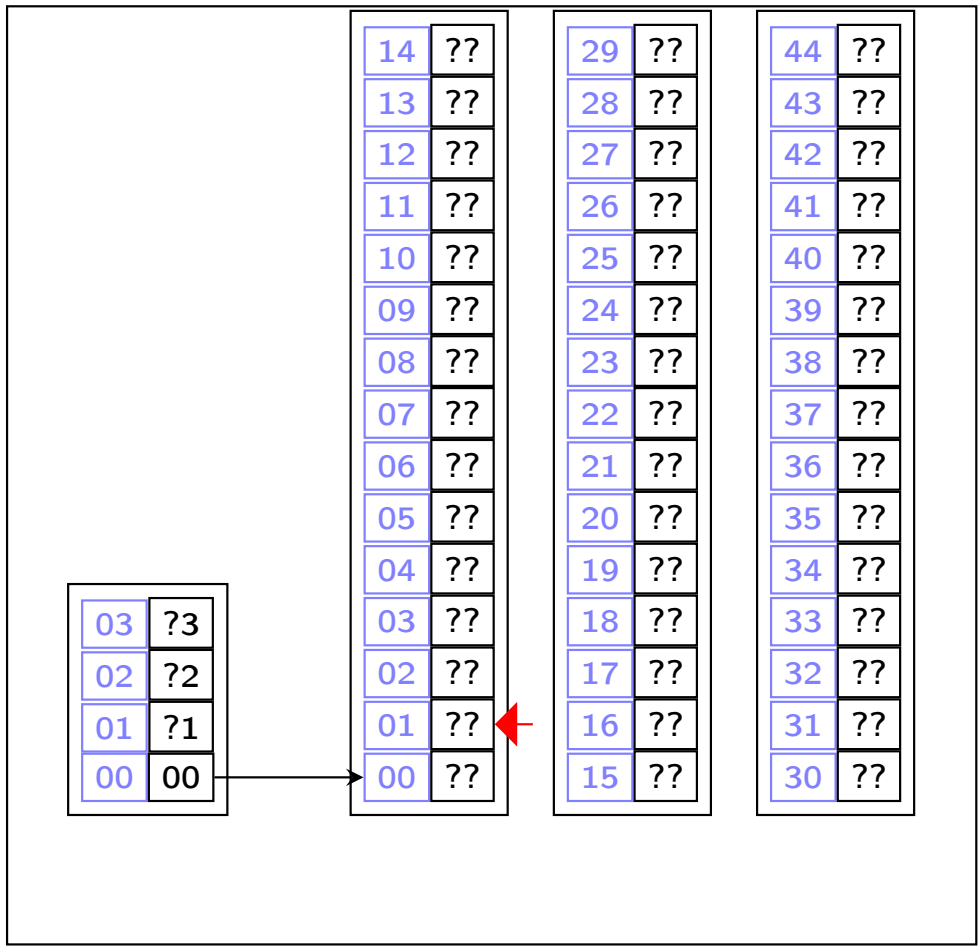
```
Símb.   Cat.   Infos
```

- Cada uma das aulas relacionadas com chamadas de procedimentos terá uma simulação de execução do programa traduzido;
- em cada aula o foco da tradução estará no registro de ativação:
  - Como as instruções "colaboram"para a construção e destruição de cada registro de ativação, com ênfase em qual instrução é a responsável por qual trecho do registro de ativação;
  - Como os vários registros de ativação de um dado nível léxico se encaixam como uma lista encadeada;
  - Escopo e visibilidade de cada símbolo em tempo de execução;
- **ATENÇÃO:** Daqui para frente é muito comum confundir o que ocorre em tempo de compilação com o que ocorre em tempo de execução. Fique atento!

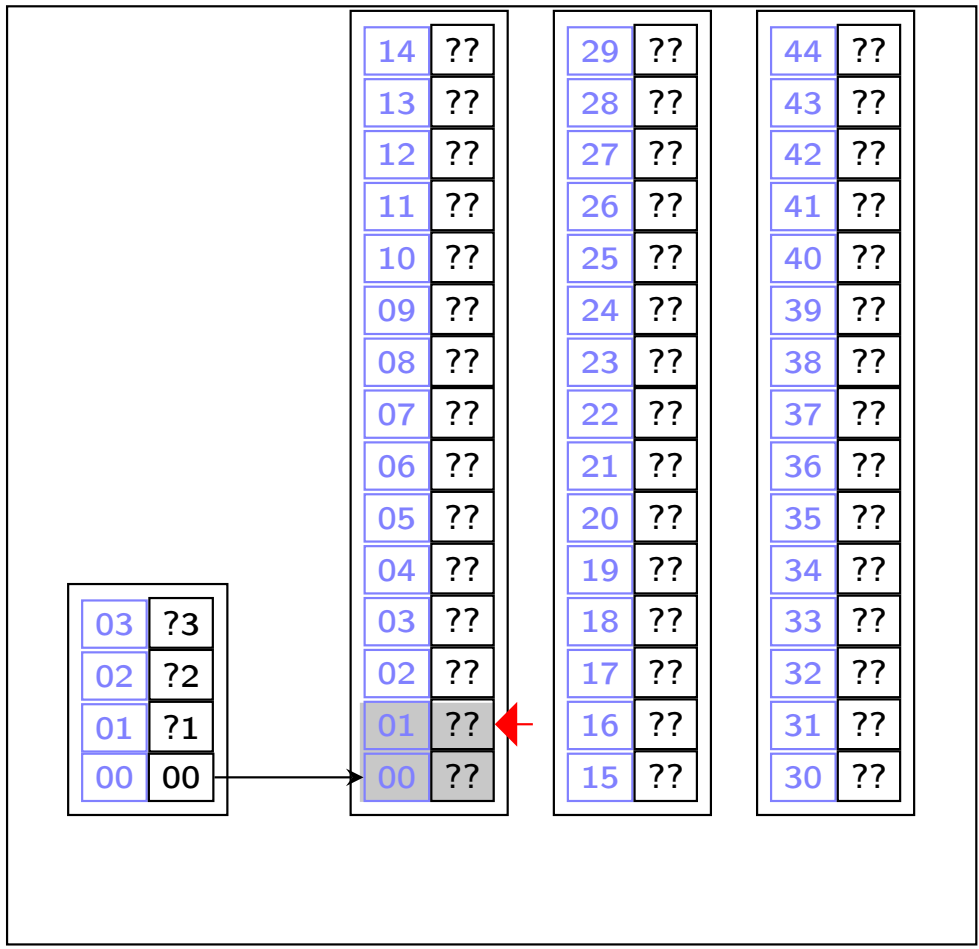
```
program proc1 (input, output);
  var x, y: integer;
  procedure p;
     var z:integer;
     begin
       z:=x;
       x:=x-1;
       if (z>1)
         then p
         else y:=1;
       y:=y*z
    end
begin
   read(x);
   p
   write (x,y)
end.
```



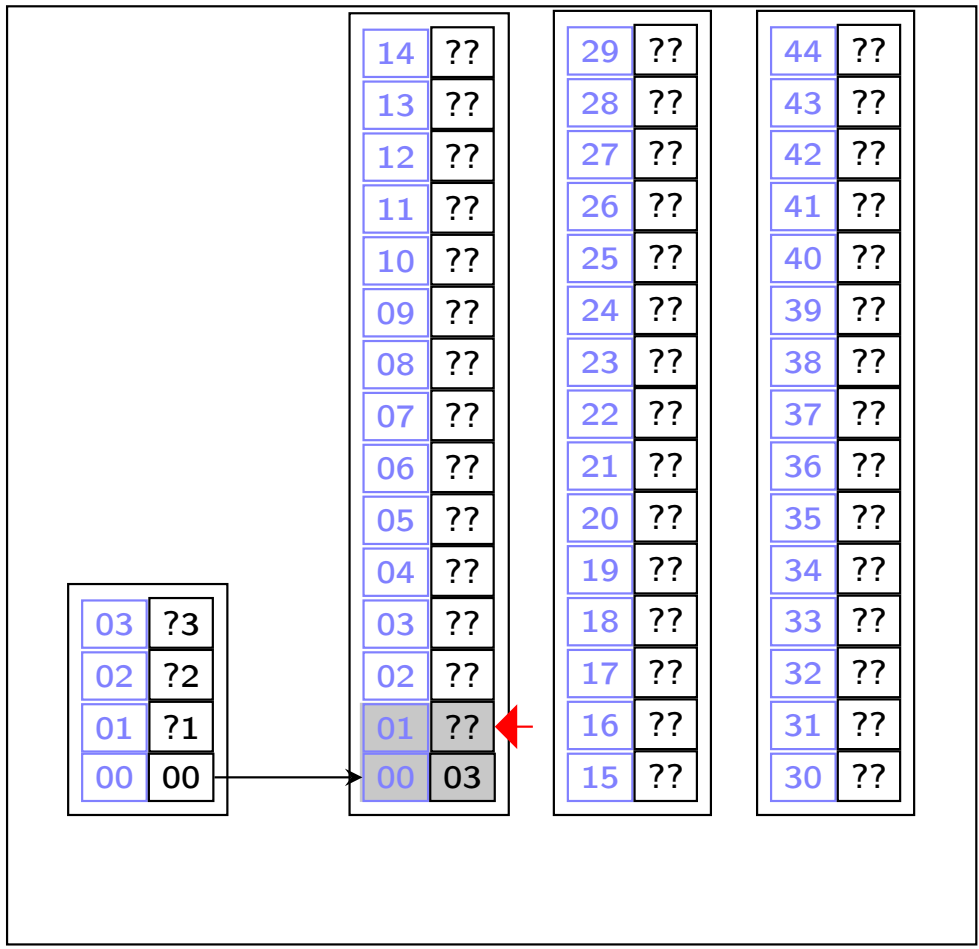
```
program proc1 (input, output);
  var x, y: integer;
  procedure p;
     var z:integer;
     begin
       z:=x;
       x:=x-1;
       if (z>1)
         then p
         else y:=1;
       y:=y*z
     end
begin
   read(x);
   p
   write (x,y)
end.
```



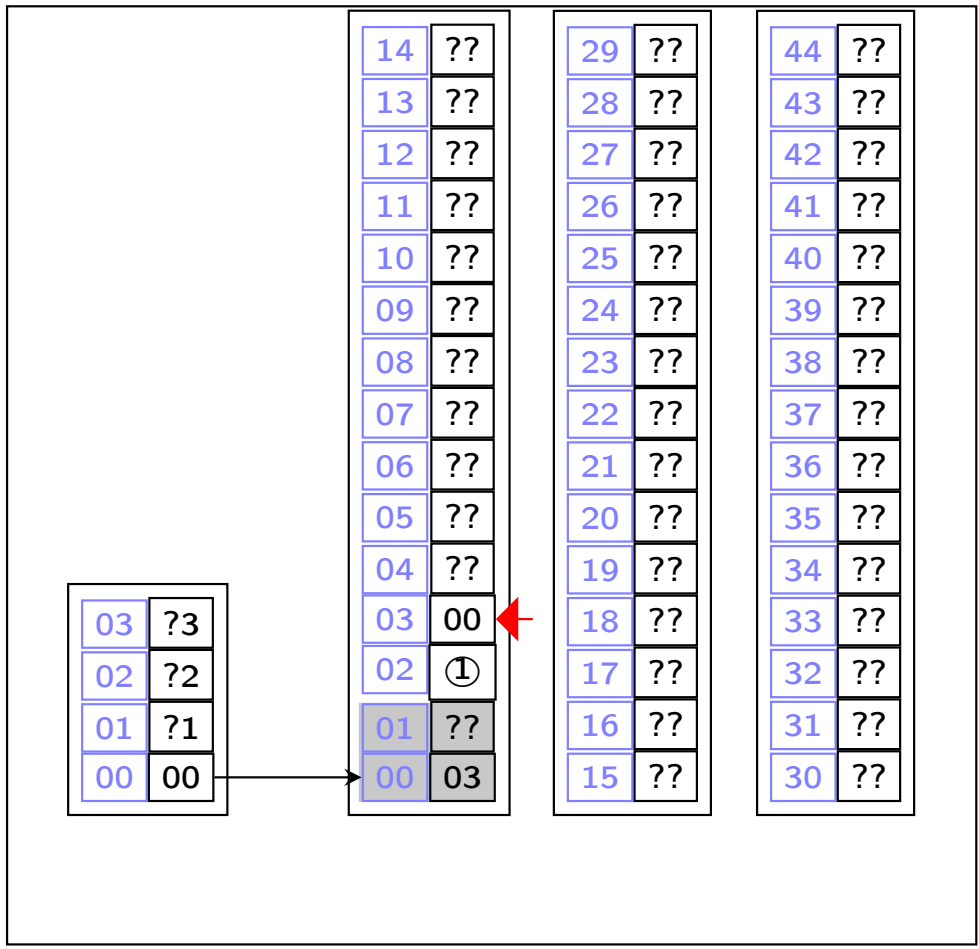
```
program proc1 (input, output);
  var x, y: integer;
  procedure p;
     var z:integer;
     begin
       z:=x;
       x:=x-1;
       if (z>1)
         then p
         else y:=1;
       y:=y*z
     end
begin
    read(x);
    p
    write (x,y)
end.
```



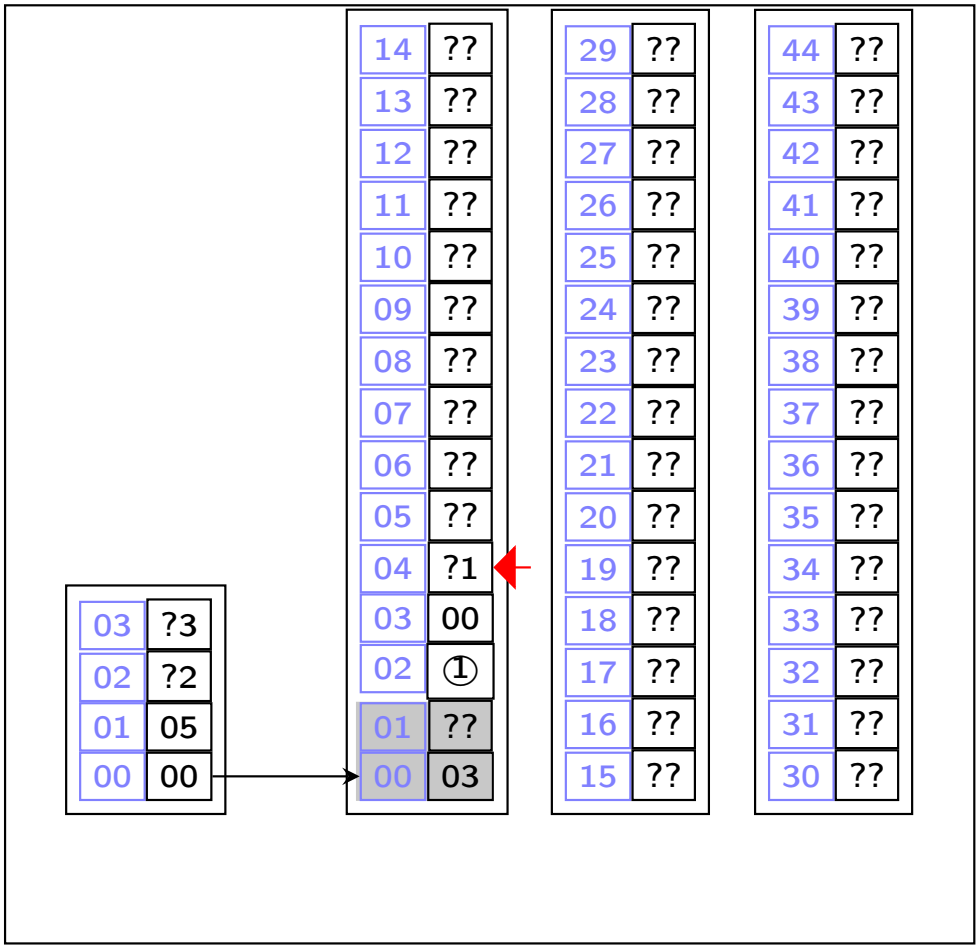
```
program proc1 (input, output);
  var x, y: integer;
  procedure p;
     var z:integer;
     begin
       z:=x;
       x:=x-1;
       if (z>1)
         then p
         else y:=1;
       y:=y*z
    end
begin
   read(x);
   p
   write (x,y)
end.
```



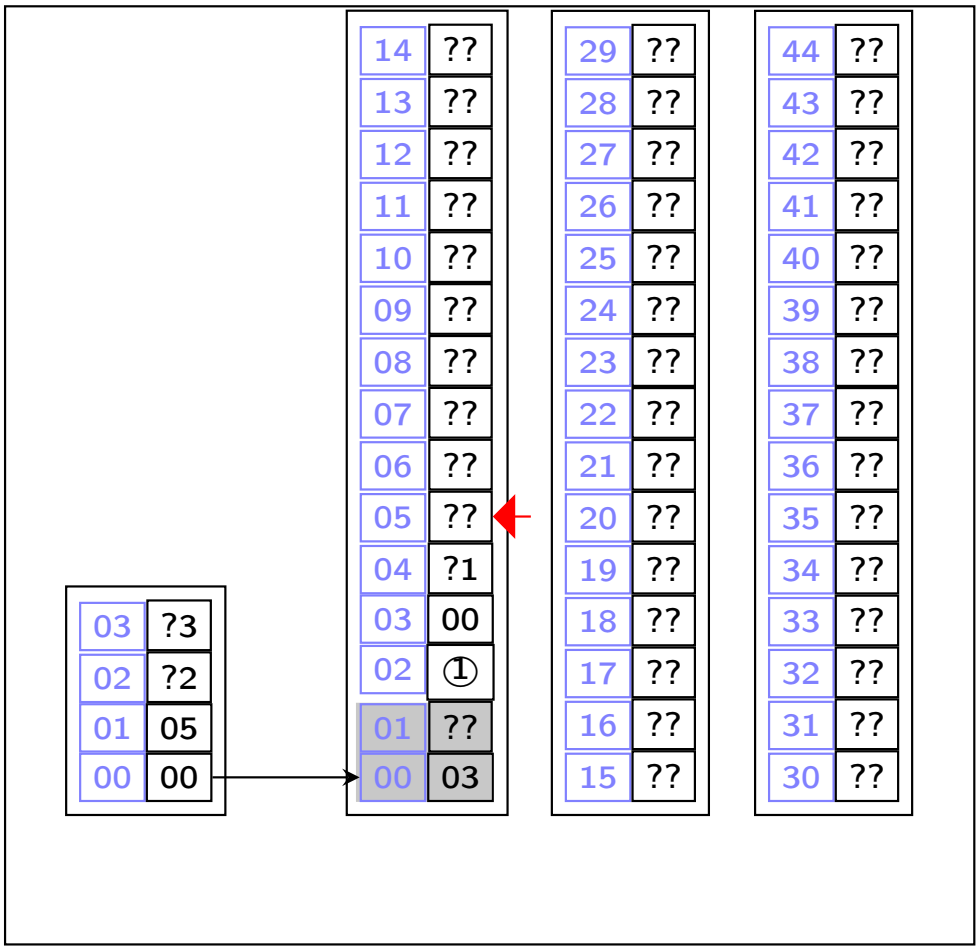
```
program proc1 (input, output);
  var x, y: integer;
  procedure p;
     var z:integer;
     begin
       z:=x;
       x:=x-1;
       if (z>1)
         then p
         else y:=1;
       y:=y*z
    end
begin
   read(x);
   p①
   write (x,y)
end.

CHPR rot,k { M[s+1]:=i;
             M[s+2]:=k;
             s:=s+2
             i:=rot}
```



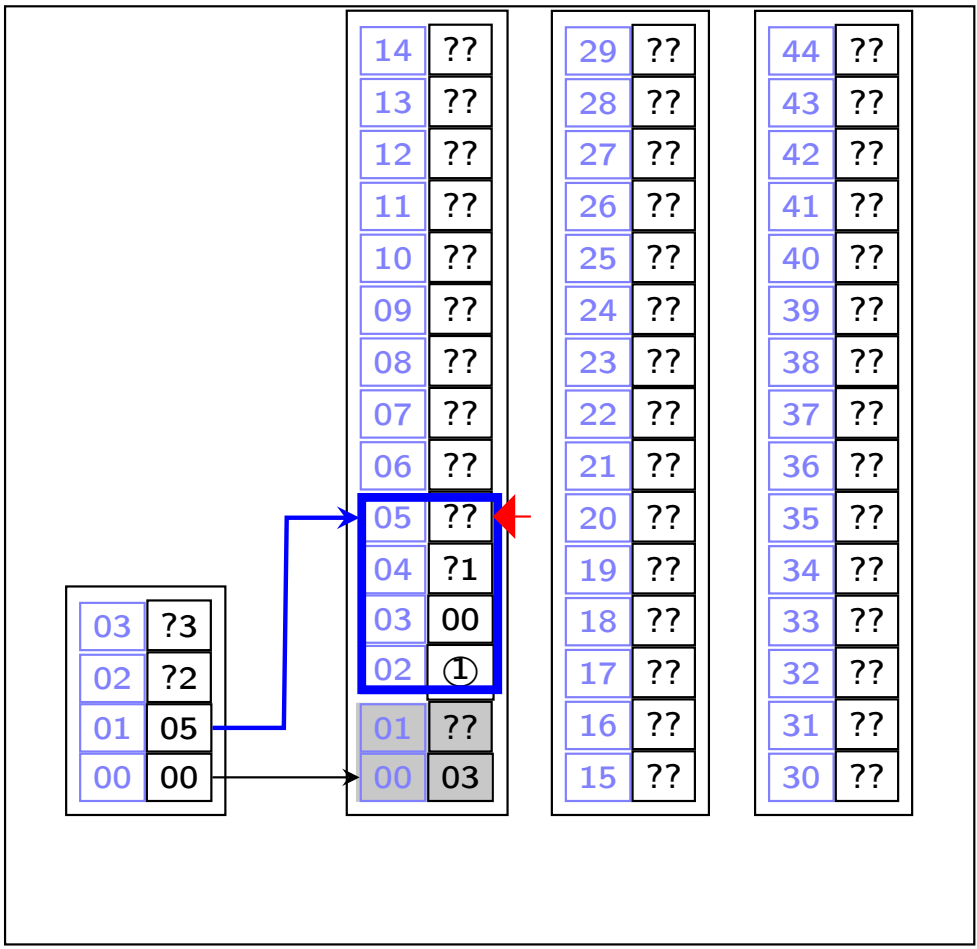
```
program proc1 (input, output);
  var x, y: integer;
  procedure p;
     var z:integer;
     begin
       z:=x;
       x:=x-1;
       if (z>1)
         then p
         else y:=1;
       y:=y*z
    end
begin
   read(x);
   p①
   write (x,y)
end.

ENPR k      { s:=s+1;
              M[s]:=D[k]
              D[k]:=s+1 }
```



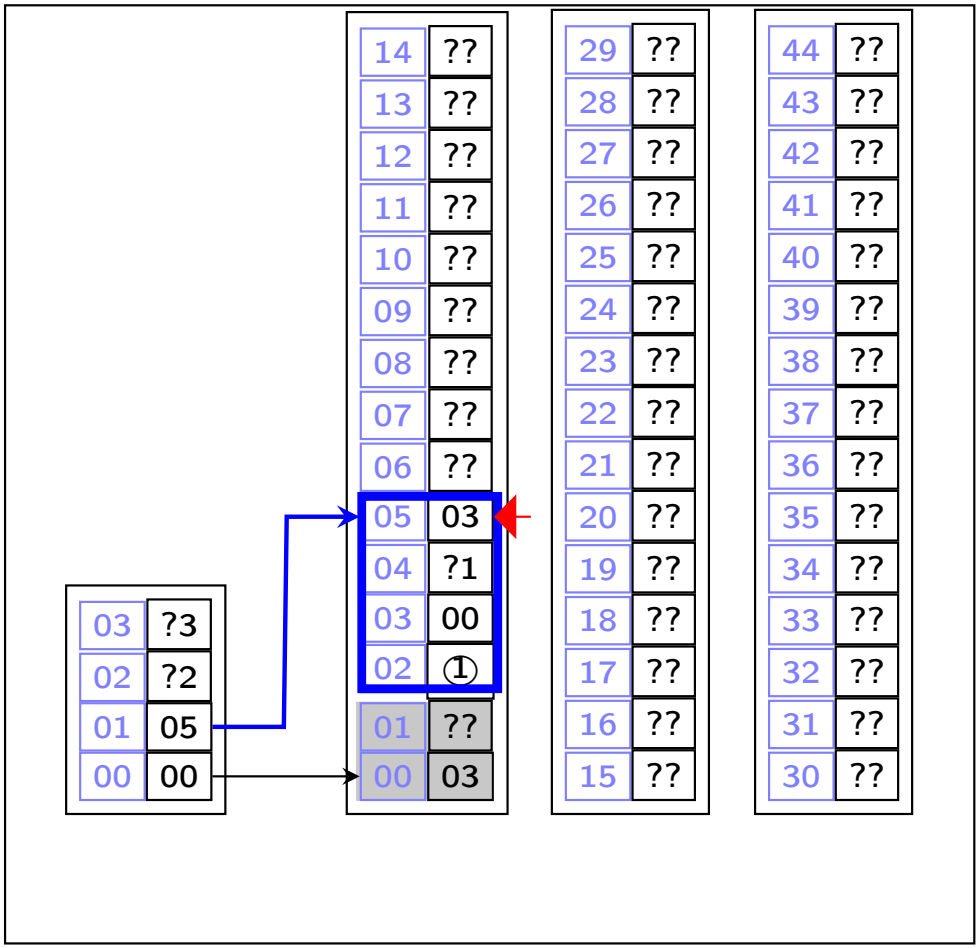
```
program proc1 (input, output);
  var x, y: integer;
  procedure p;
     var z:integer;
     begin
       z:=x;
       x:=x-1;
       if (z>1)
         then p
         else y:=1;
       y:=y*z
    end
begin
   read(x);
   p①
   write (x,y)
end.
```



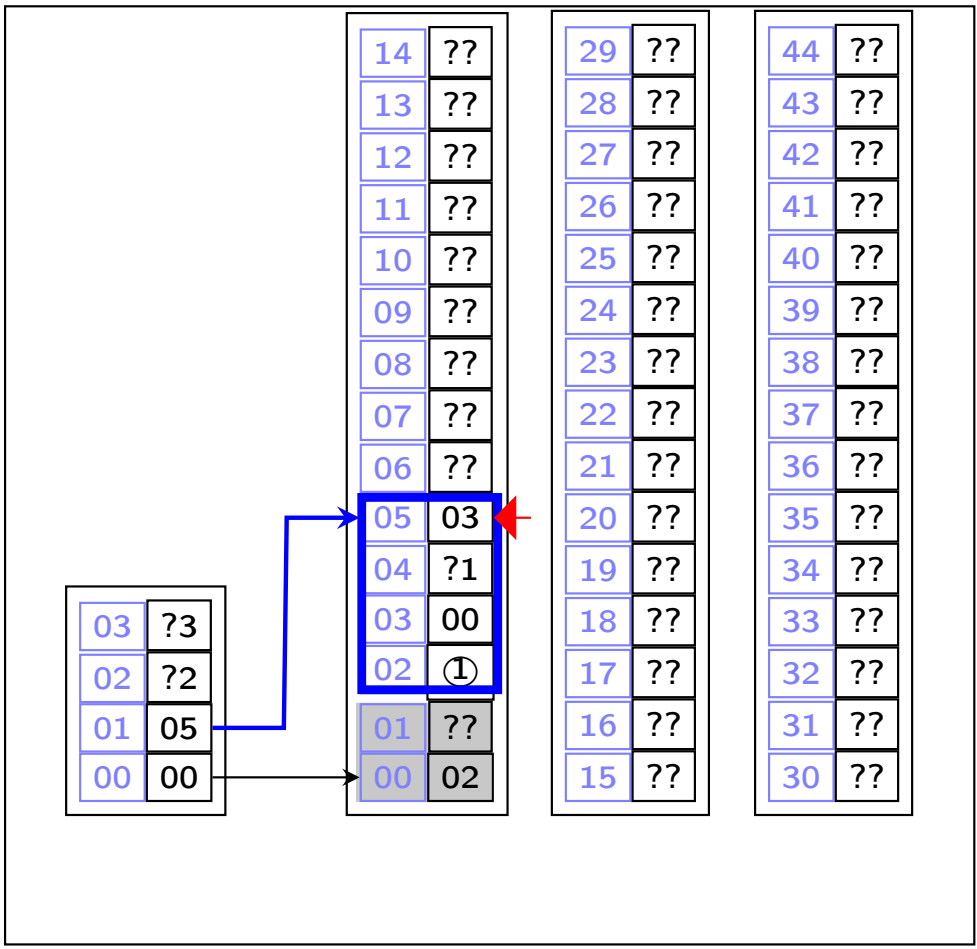
```
program proc1 (input, output);
  var x, y: integer;
  procedure p;
     var z:integer;
     begin
       z:=x;
       x:=x-1;
       if (z>1)
         then p
         else y:=1;
       y:=y*z
    end
begin
   read(x);
   p①
   write (x,y)
end.
```



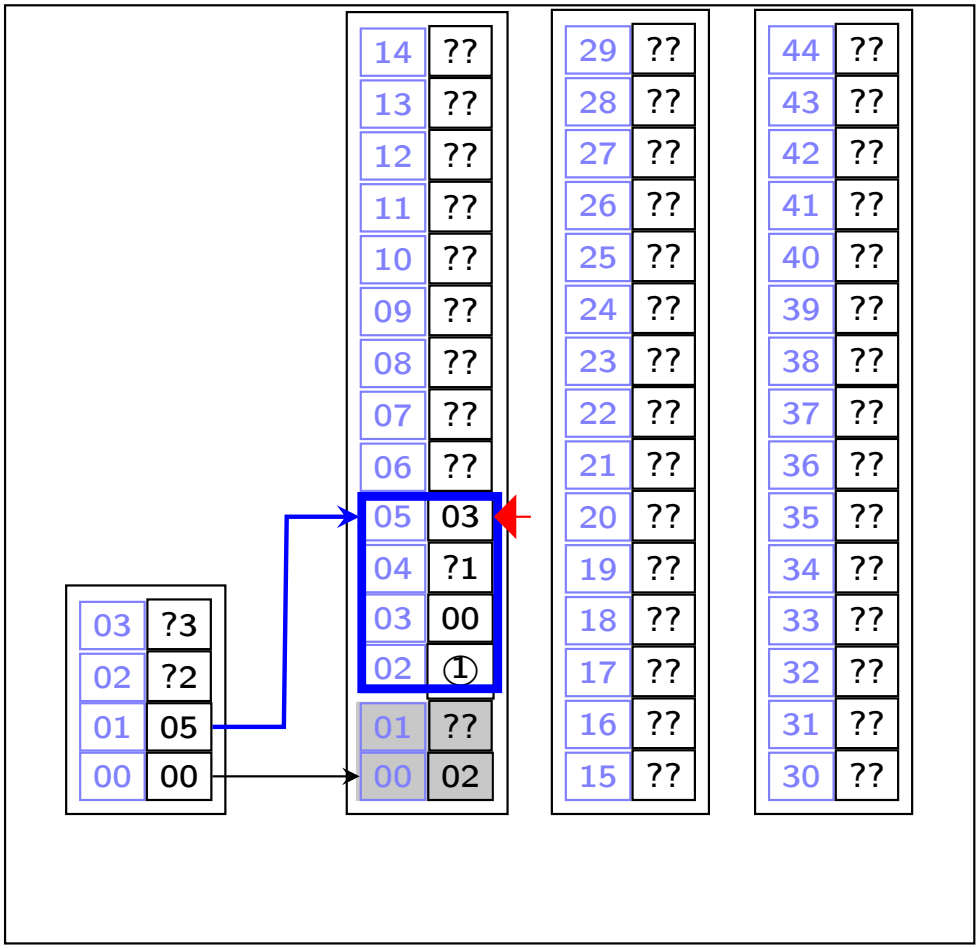
```
program proc1 (input, output);
  var x, y: integer;
  procedure p;
     var z:integer;
     begin
       z:=x;
       x:=x-1;
       if (z>1)
         then p
         else y:=1;
       y:=y*z
     end
begin
   read(x);
   p①
   write (x,y)
end.
```



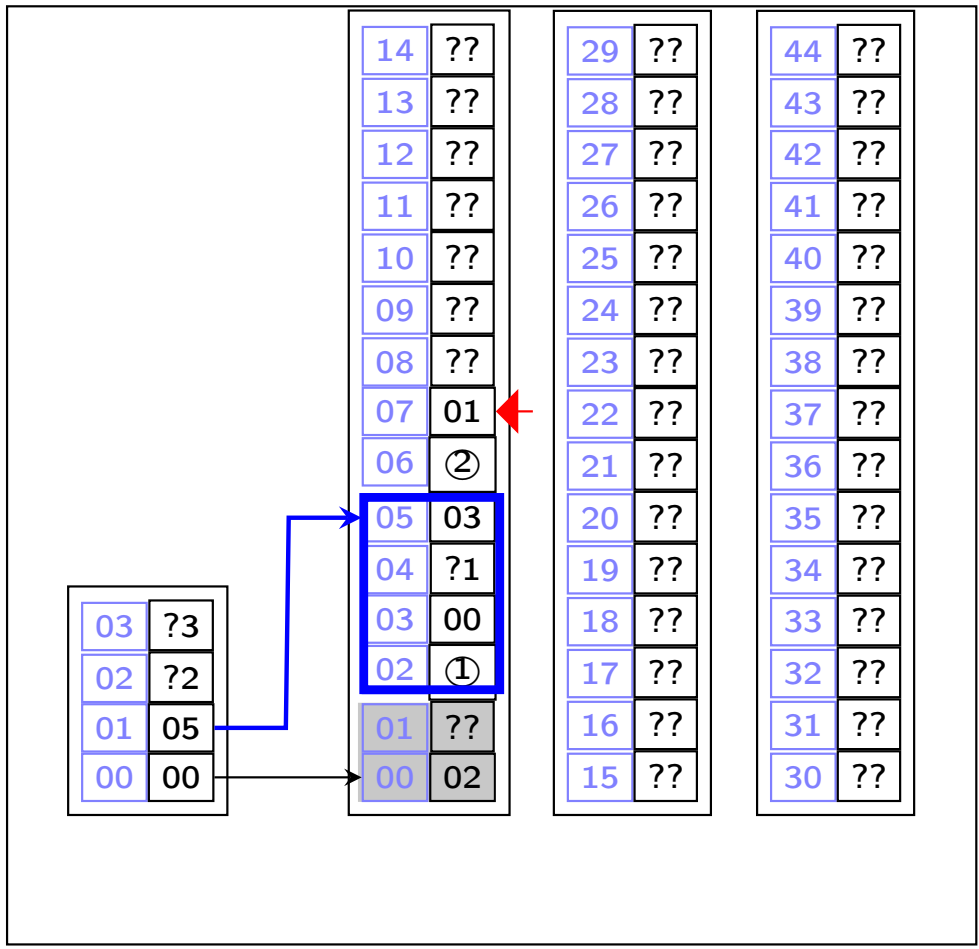
```
program proc1 (input, output);
  var x, y: integer;
  procedure p;
     var z:integer;
     begin
       z:=x;
       x:=x-1;
       if (z>1)
         then p
         else y:=1;
       y:=y*z
     end
begin
   read(x);
   p①
   write (x,y)
end.
```



```
program proc1 (input, output);
  var x, y: integer;
  procedure p;
     var z:integer;
     begin
       z:=x;
       x:=x-1;
       if (z>1)
         then p
         else y:=1;
       y:=y*z
     end
begin
   read(x);
   p①
   write (x,y)
end.
```

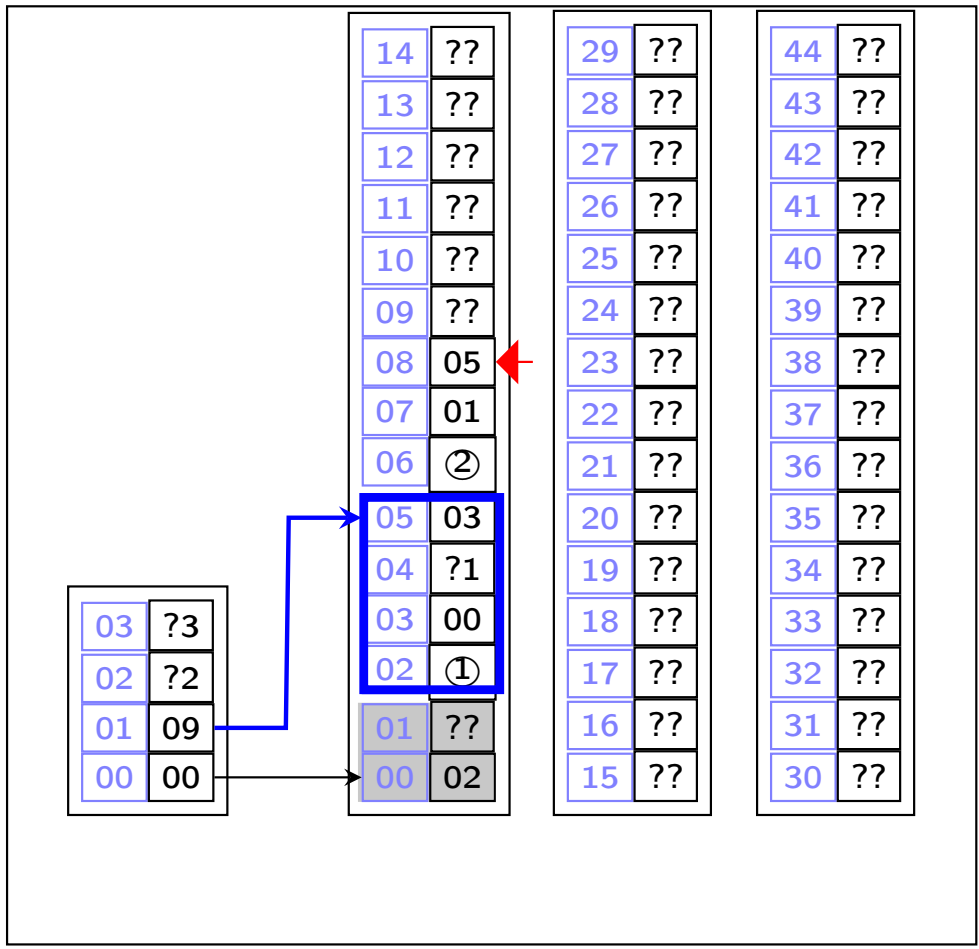


```
program proc1 (input, output);
  var x, y: integer;
  procedure p;
     var z:integer;
     begin
       z:=x;
       x:=x-1;
       if (z>1)
         then p②
         else y:=1;
       y:=y*z
    end
begin
   read(x);
   p①
   write (x,y)
end.

CHPR rot,k { M[s+1]:=i;
             M[s+2]:=k;
             s:=s+2
             i:=rot}
```



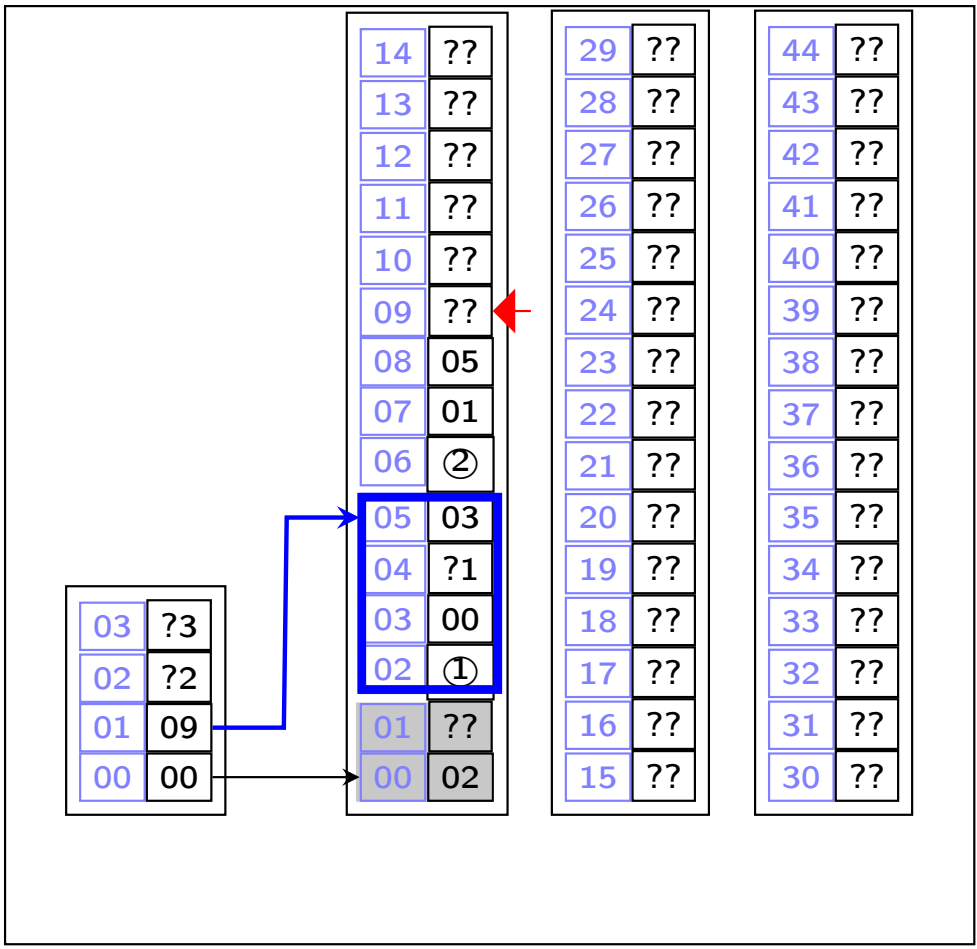
Execução

```
program proc1 (input, output);
  var x, y: integer;
  procedure p;
     var z:integer;
     begin
       z:=x;
       x:=x-1;
       if (z>1)
         then p②
         else y:=1;
       y:=y*z
    end
begin
   read(x);
   p①
   write (x,y)
end.

ENPR k      { s:=s+1;
              M[s]:=D[k]
              D[k]:=s+1 }
```



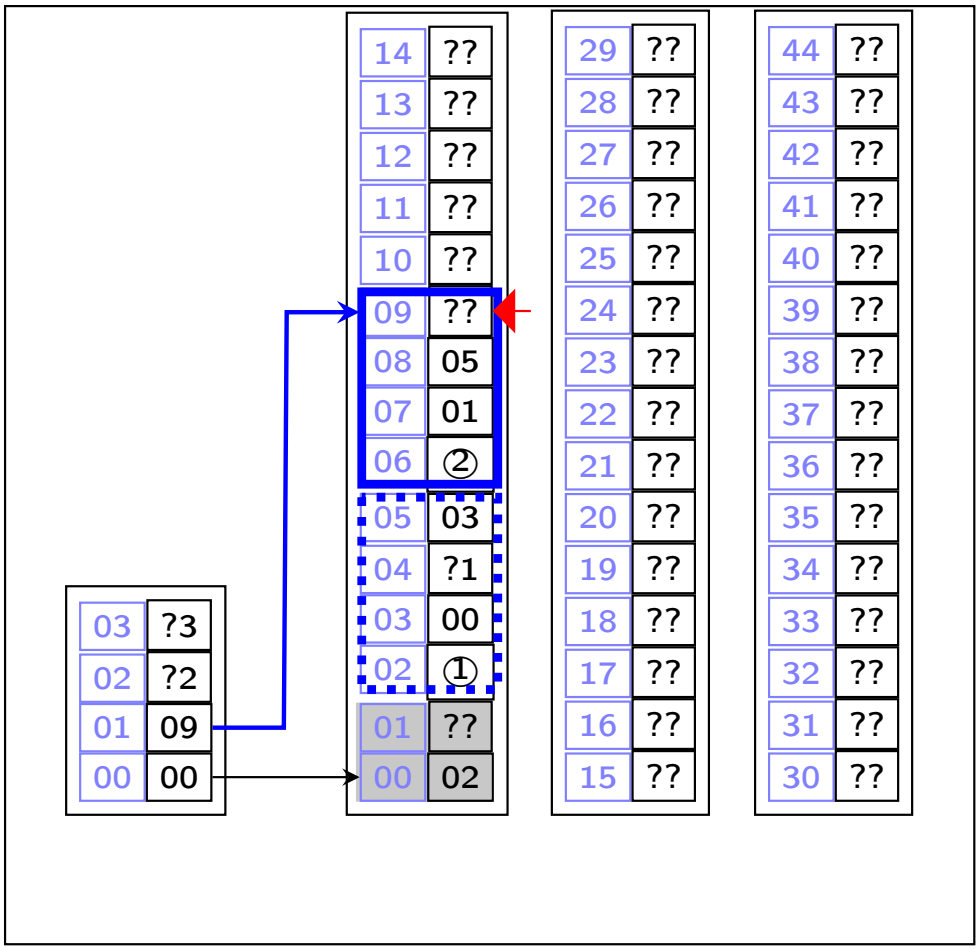
Execução

```
program proc1 (input, output);
  var x, y: integer;
  procedure p;
     var z:integer;
     begin
       z:=x;
       x:=x-1;
       if (z>1)
         then p②
         else y:=1;
       y:=y*z
    end
begin
   read(x);
   p①
   write (x,y)
end.
```



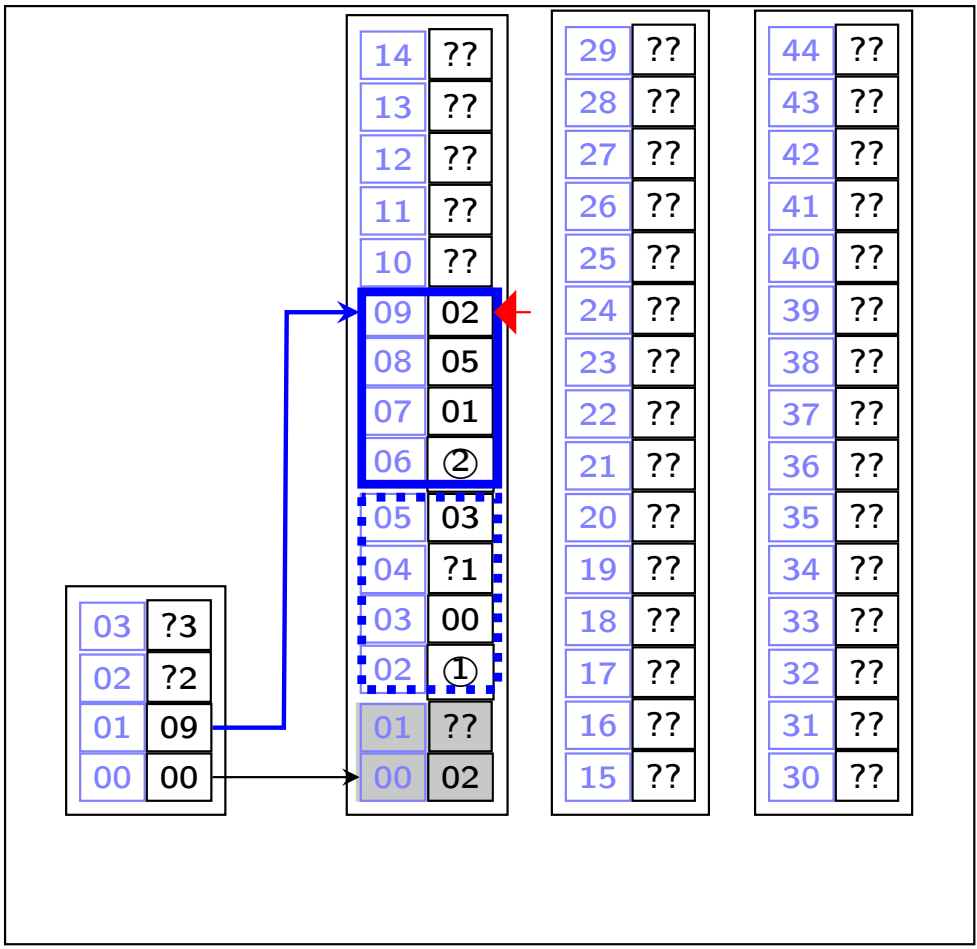
Execução

```
program proc1 (input, output);
  var x, y: integer;
  procedure p;
     var z:integer;
     begin
       z:=x;
       x:=x-1;
       if (z>1)
         then p②
         else y:=1;
       y:=y*z
    end
begin
   read(x);
   p①
   write (x,y)
end.
```



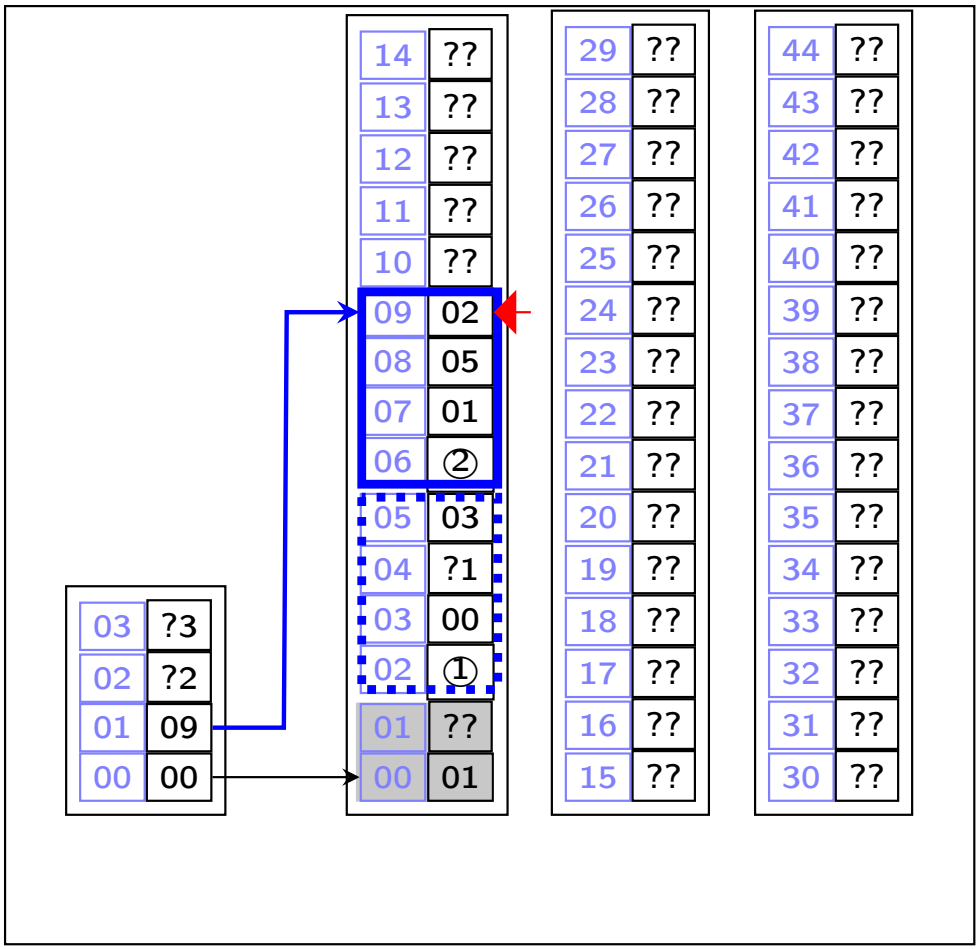
Execução

```
program proc1 (input, output);
  var x, y: integer;
  procedure p;
     var z:integer;
     begin
       z:=x;
       x:=x-1;
       if (z>1)
         then p②
         else y:=1;
       y:=y*z
     end
begin
   read(x);
   p①
   write (x,y)
end.
```



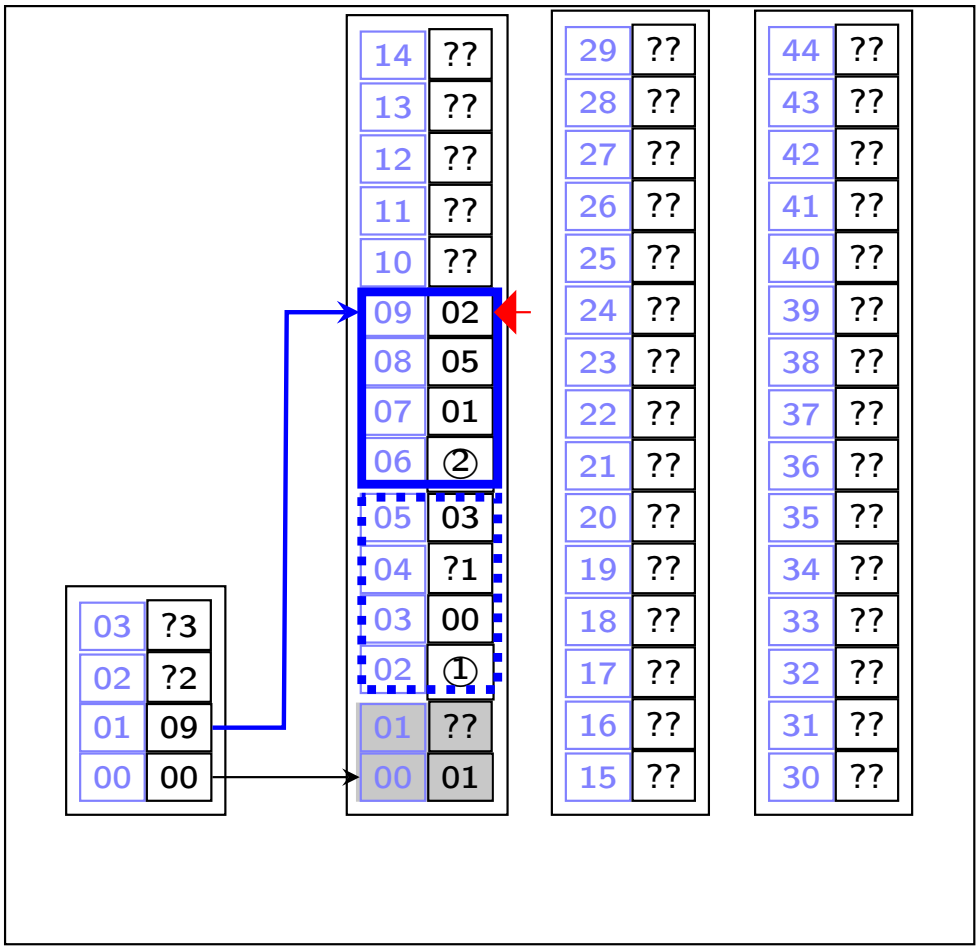
Execução

```
program proc1 (input, output);
  var x, y: integer;
  procedure p;
     var z:integer;
     begin
       z:=x;
       x:=x-1;
       if (z>1)
         then p②
         else y:=1;
       y:=y*z
     end
begin
   read(x);
   p①
   write (x,y)
end.
```



Execução

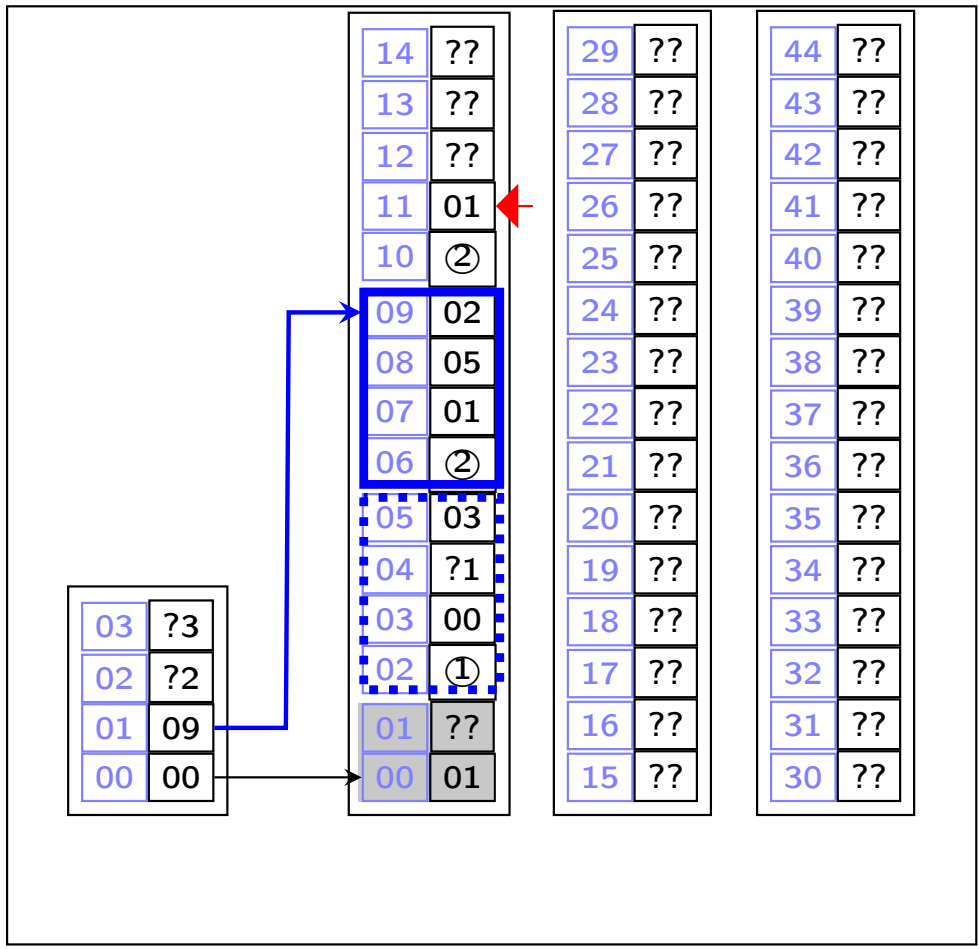
```
program proc1 (input, output);
  var x, y: integer;
  procedure p;
     var z:integer;
     begin
       z:=x;
       x:=x-1;
       if (z>1)
         then p②
         else y:=1;
       y:=y*z
    end
begin
   read(x);
   p①
   write (x,y)
end.
```



Execução

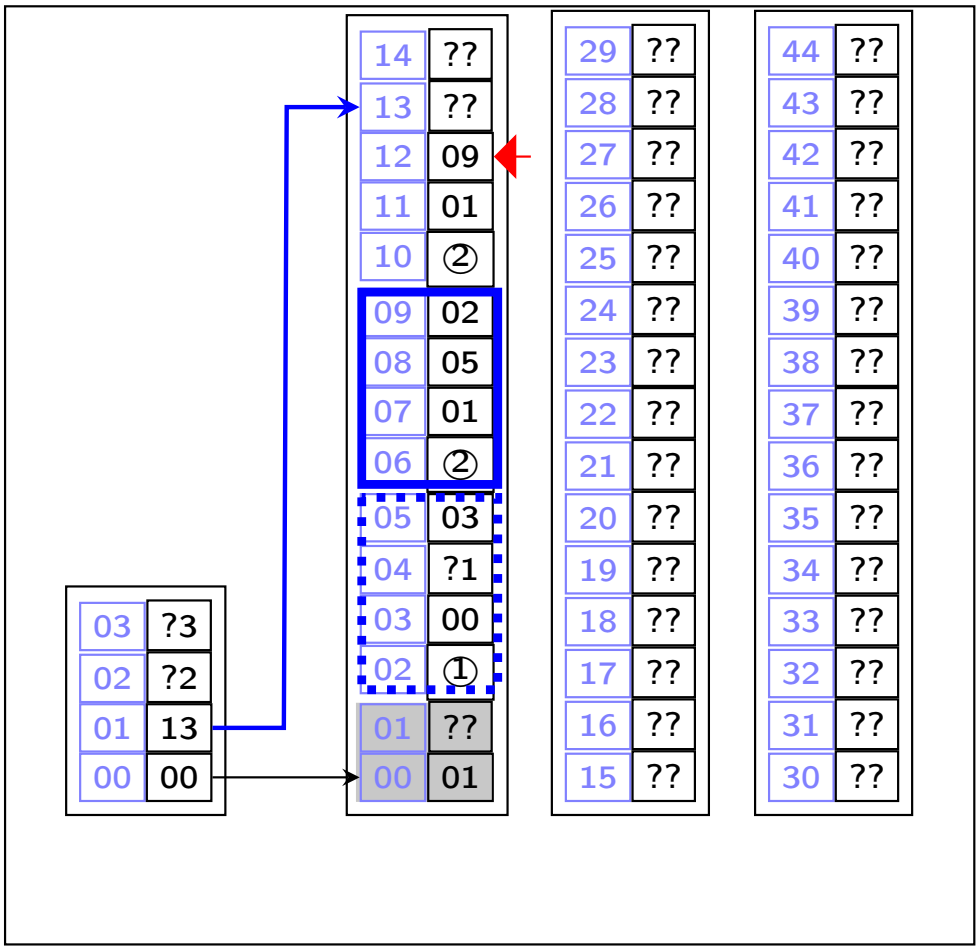
```
program proc1 (input, output);
  var x, y: integer;
  procedure p;
     var z:integer;
     begin
       z:=x;
       x:=x-1;
       if (z>1)
          then p②
          else y:=1;
       y:=y*z
     end
begin
   read(x);
   p①
   write (x,y)
end.
```

```
CHPR rot,k { M[s+1]:=i;
             M[s+2]:=k;
             s:=s+2
             i:=rot}
```



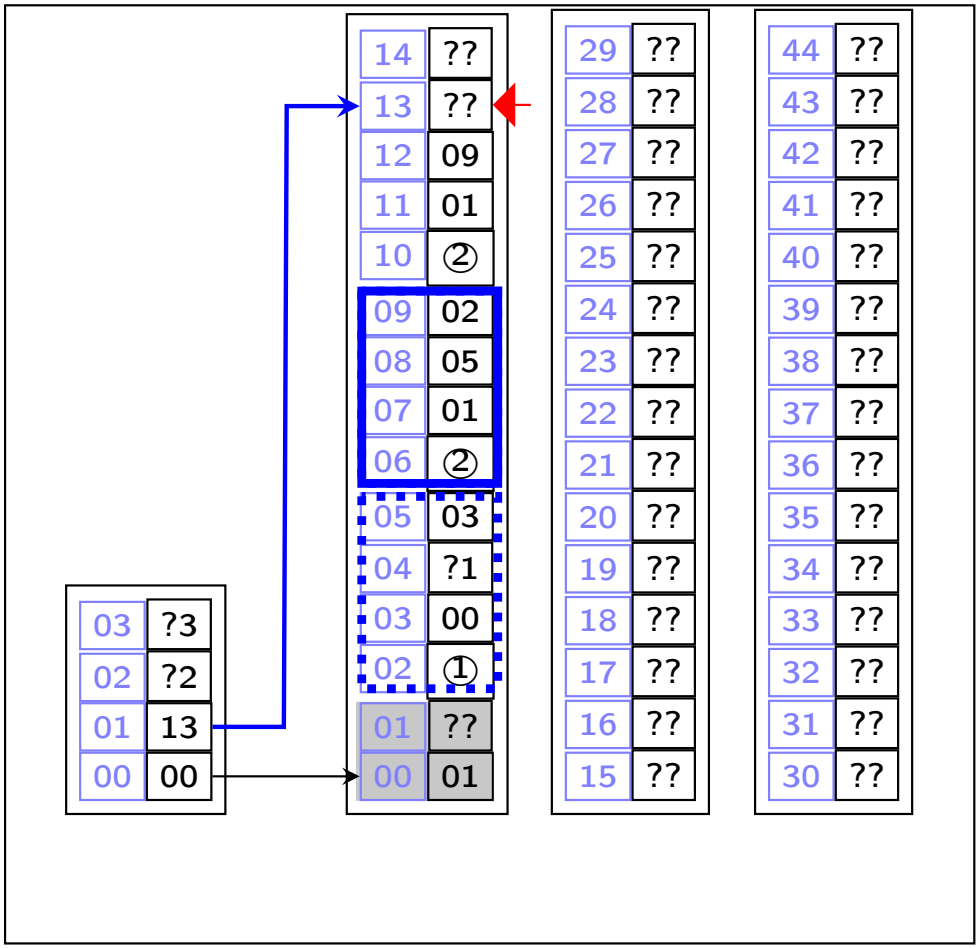
Execução

```
program proc1 (input, output);
  var x, y: integer;
  procedure p;
     var z:integer;
     begin
       z:=x;
       x:=x-1;
       if (z>1)
          then p②
          else y:=1;
       y:=y*z
     end
begin
   read(x);
   p①
   write (x,y)
end.

ENPR k      { s:=s+1;
              M[s]:=D[k]
              D[k]:=s+1 }
```



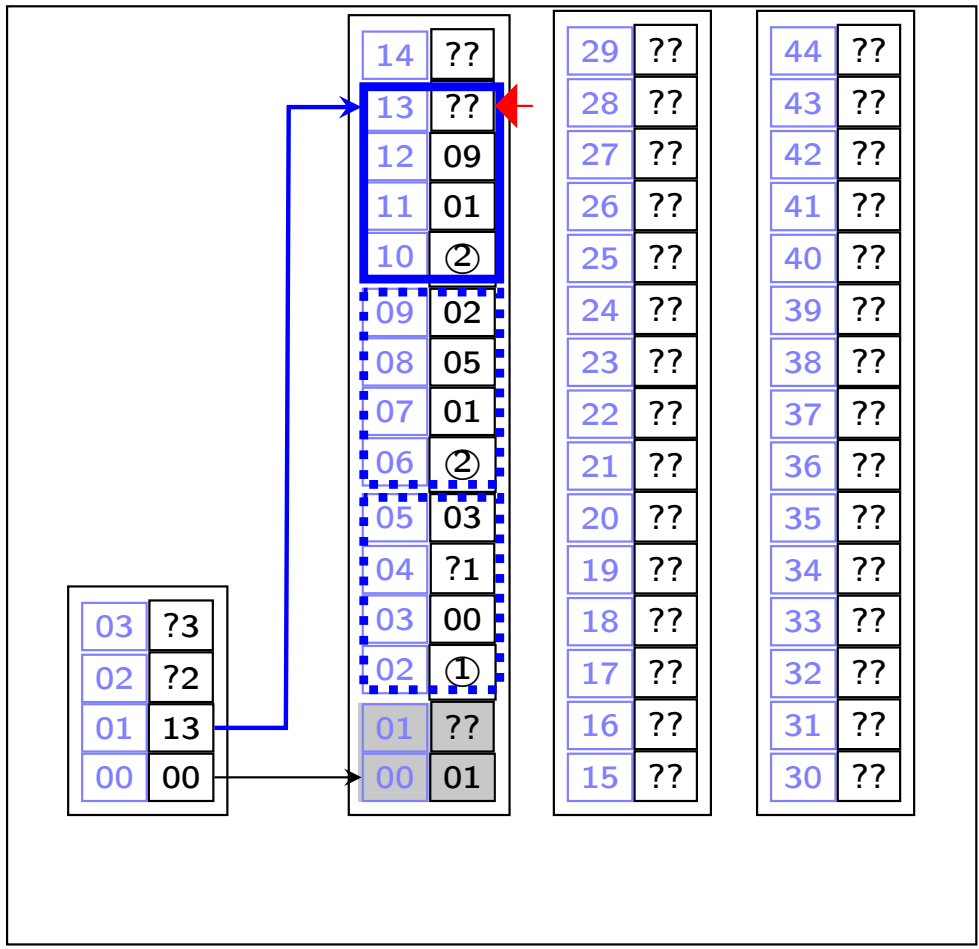
Execução

```
program proc1 (input, output);
  var x, y: integer;
  procedure p;
     var z:integer;
     begin
       z:=x;
       x:=x-1;
       if (z>1)
          then p②
          else y:=1;
       y:=y*z
     end
begin
   read(x);
   p①
   write (x,y)
end.
```



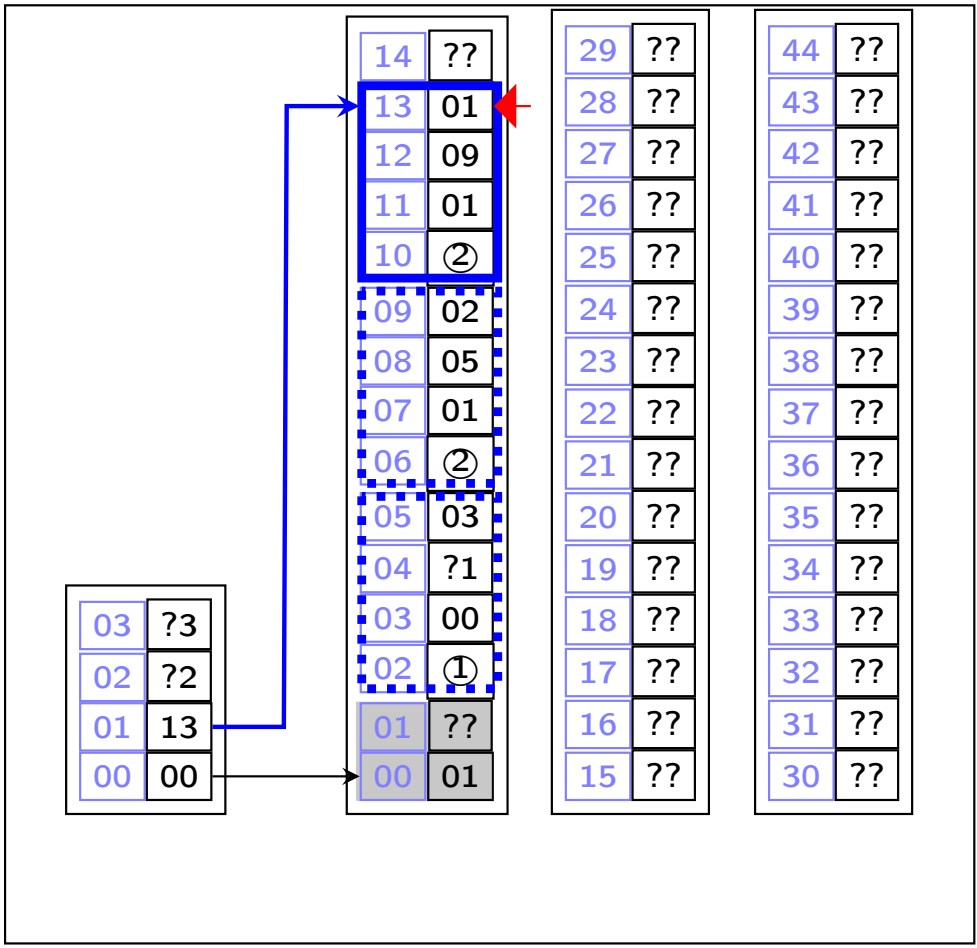
Execução

```
program proc1 (input, output);
  var x, y: integer;
  procedure p;
     var z:integer;
     begin
       z:=x;
       x:=x-1;
       if (z>1)
          then p②
          else y:=1;
       y:=y*z
     end
begin
   read(x);
   p①
   write (x,y)
end.
```



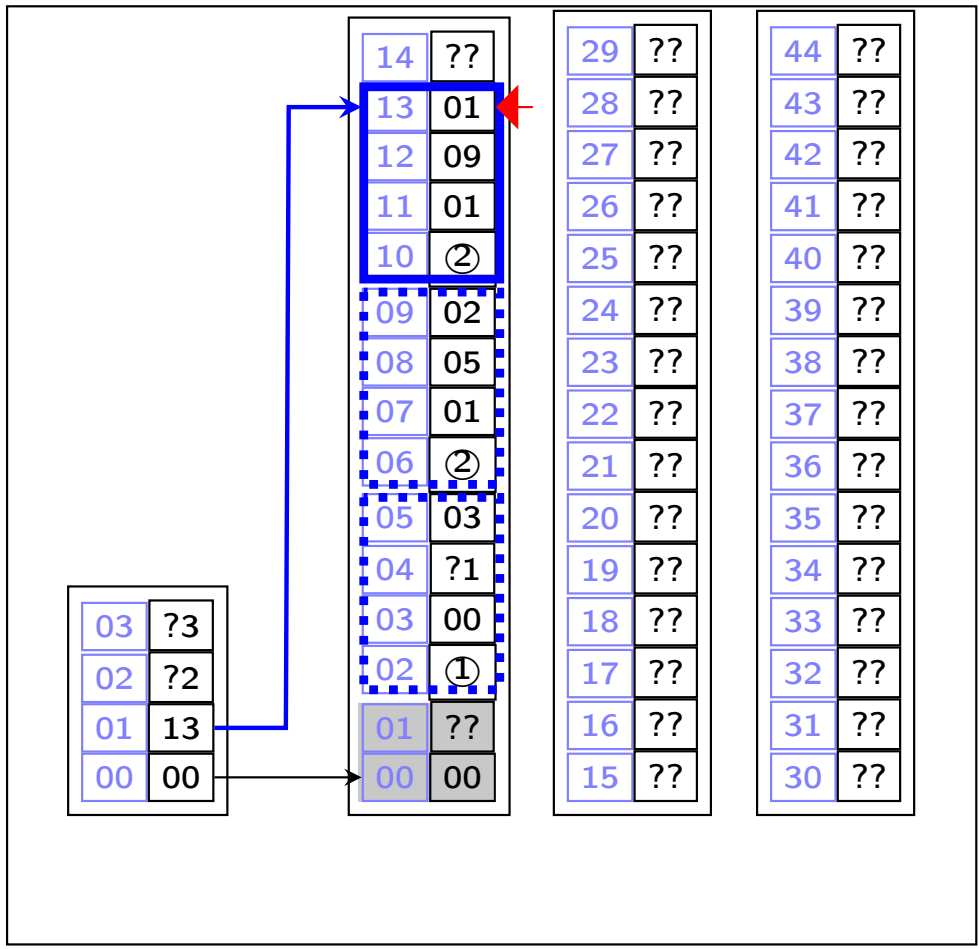
Execução

```
program proc1 (input, output);
  var x, y: integer;
  procedure p;
     var z:integer;
     begin
       z:=x;
       x:=x-1;
       if (z>1)
         then p②
         else y:=1;
       y:=y*z
     end
begin
   read(x);
   p①
   write (x,y)
end.
```



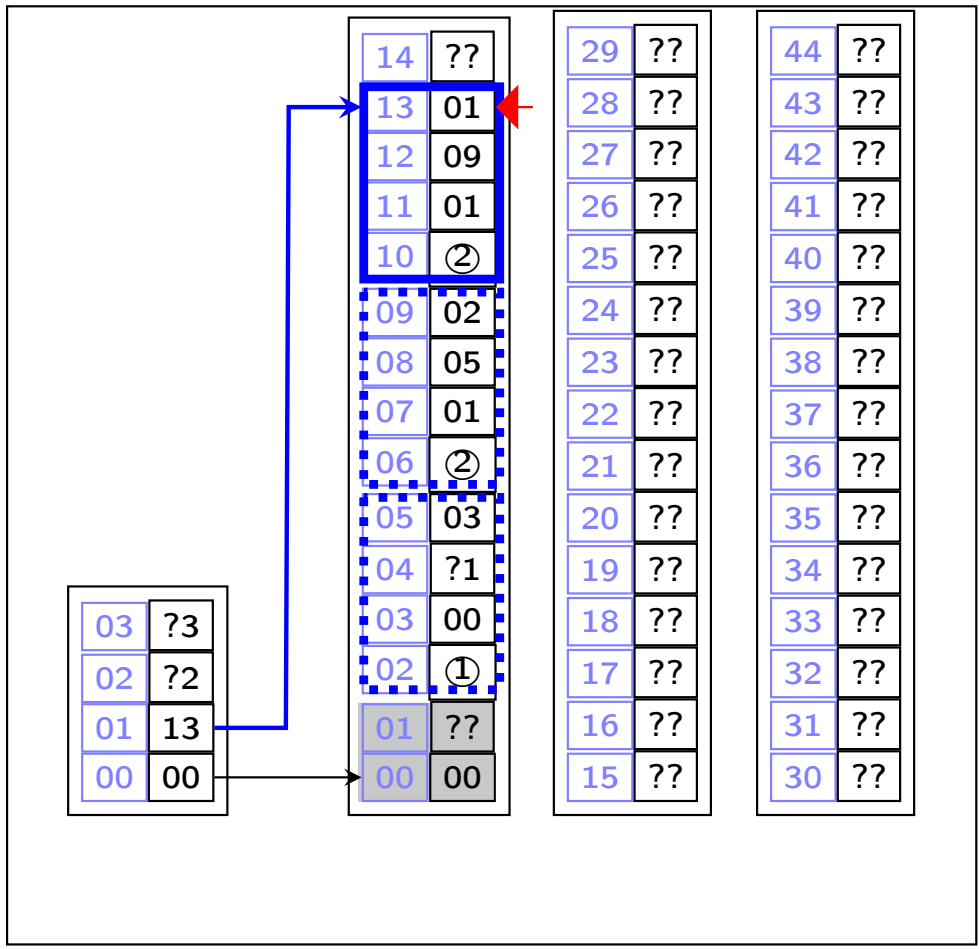
Execução

```
program proc1 (input, output);
  var x, y: integer;
  procedure p;
     var z:integer;
     begin
       z:=x;
       x:=x-1;
       if (z>1)
         then p②
         else y:=1;
       y:=y*z
     end
begin
   read(x);
   p①
   write (x,y)
end.
```



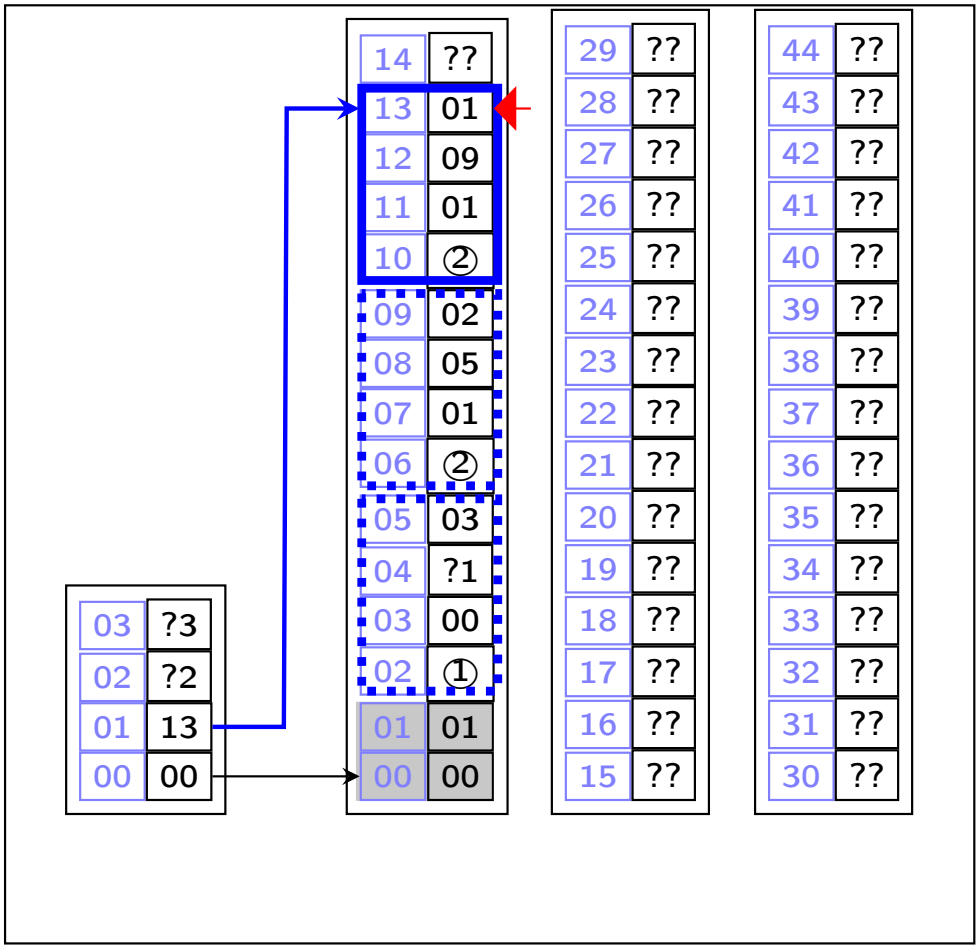
Execução

```
program proc1 (input, output);
  var x, y: integer;
  procedure p;
     var z:integer;
     begin
       z:=x;
       x:=x-1;
       if (z>1)
         then p②
         else y:=1;
       y:=y*z
     end
begin
   read(x);
   p①
   write (x,y)
end.
```



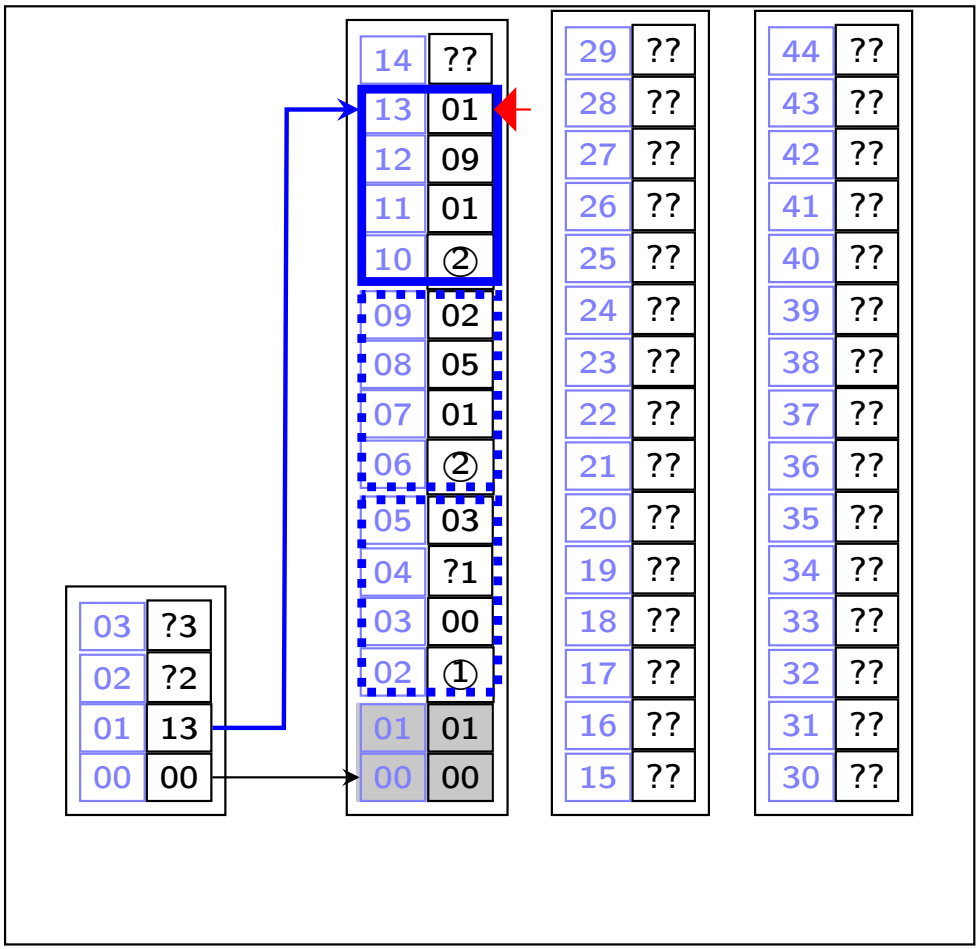
Execução

```
program proc1 (input, output);
  var x, y: integer;
  procedure p;
     var z:integer;
     begin
       z:=x;
       x:=x-1;
       if (z>1)
         then p②
         else y:=1;
       y:=y*z
    end
begin
   read(x);
   p①
   write (x,y)
end.
```



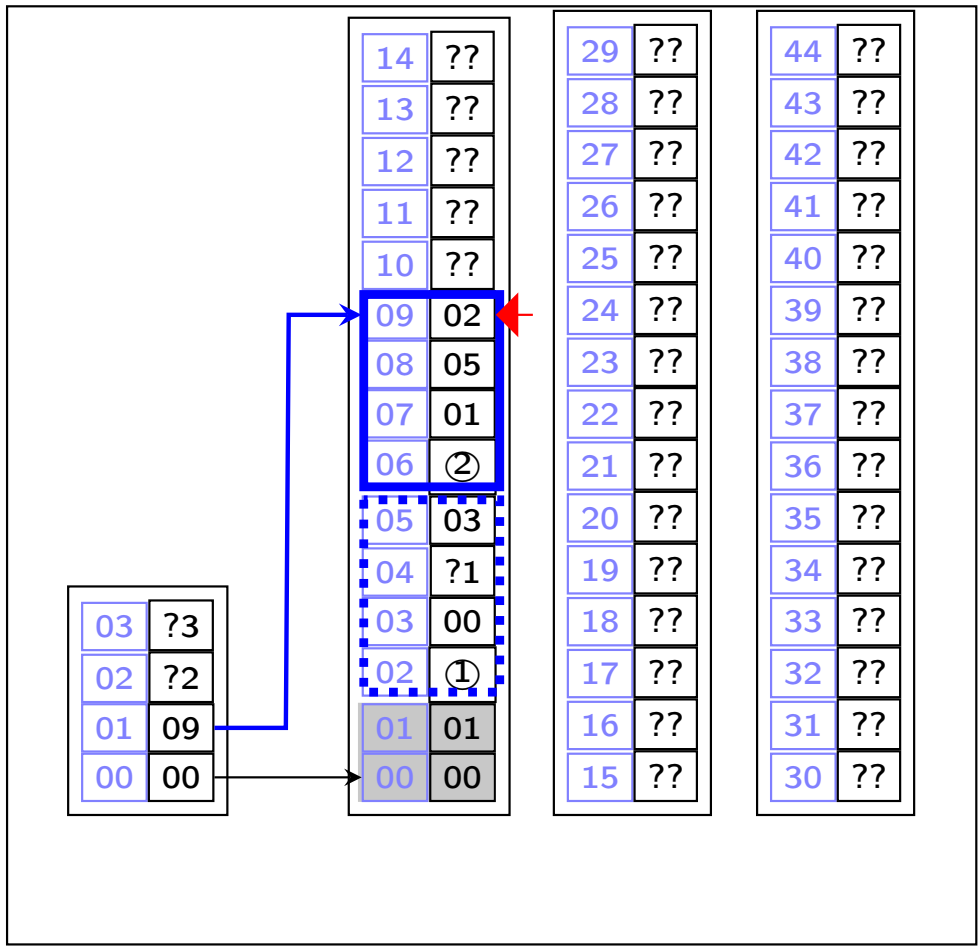
Execução

```
program proc1 (input, output);
  var x, y: integer;
  procedure p;
     var z:integer;
    begin
      z:=x;
      x:=x-1;
      if (z>1)
        then p②
        else y:=1;
      y:=y*z
    end
begin
   read(x);
   p①
   write (x,y)
end.
```



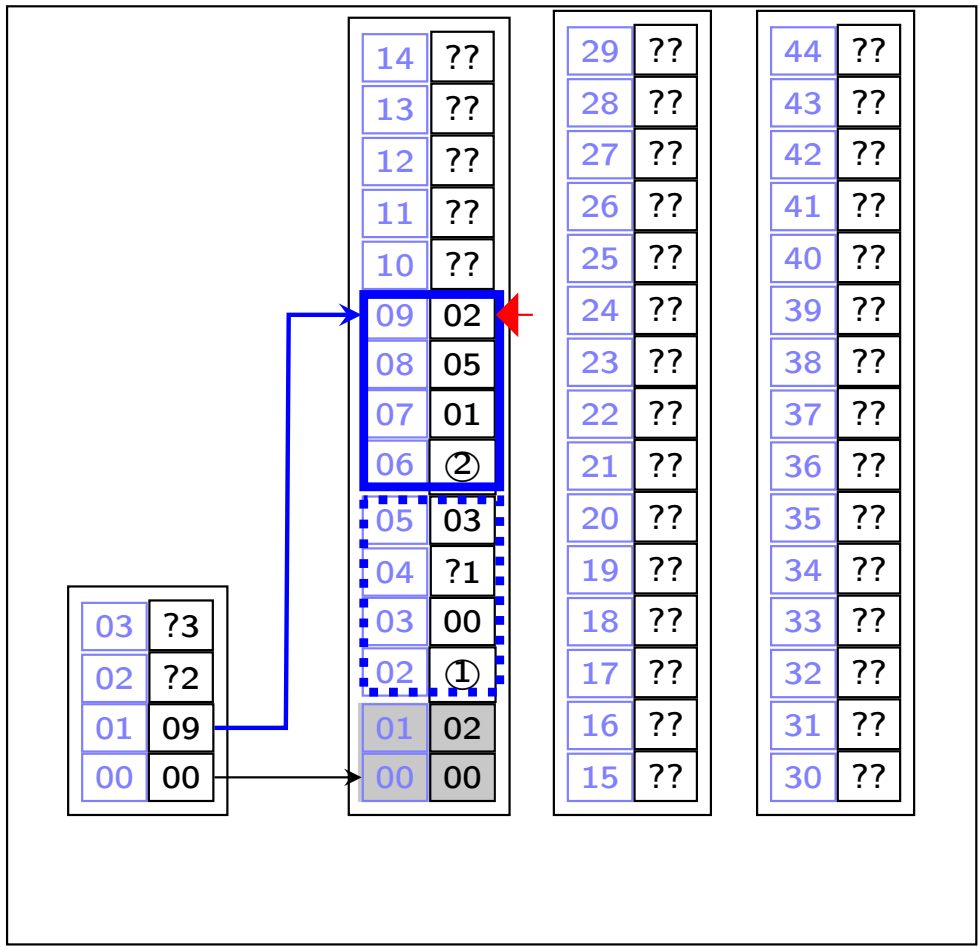
Execução

```
program proc1 (input, output);
  var x, y: integer;
  procedure p;
     var z:integer;
    begin
      z:=x;
      x:=x-1;
      if (z>1)
        then p②
        else y:=1;
      y:=y*z
    end
begin
   read(x);
   p①
   write (x,y)
end.

DMEM 1
RTPR k,n { D[K]:=M[s];
           i:=M[s-2];
           s:=s-(n+3)}
```



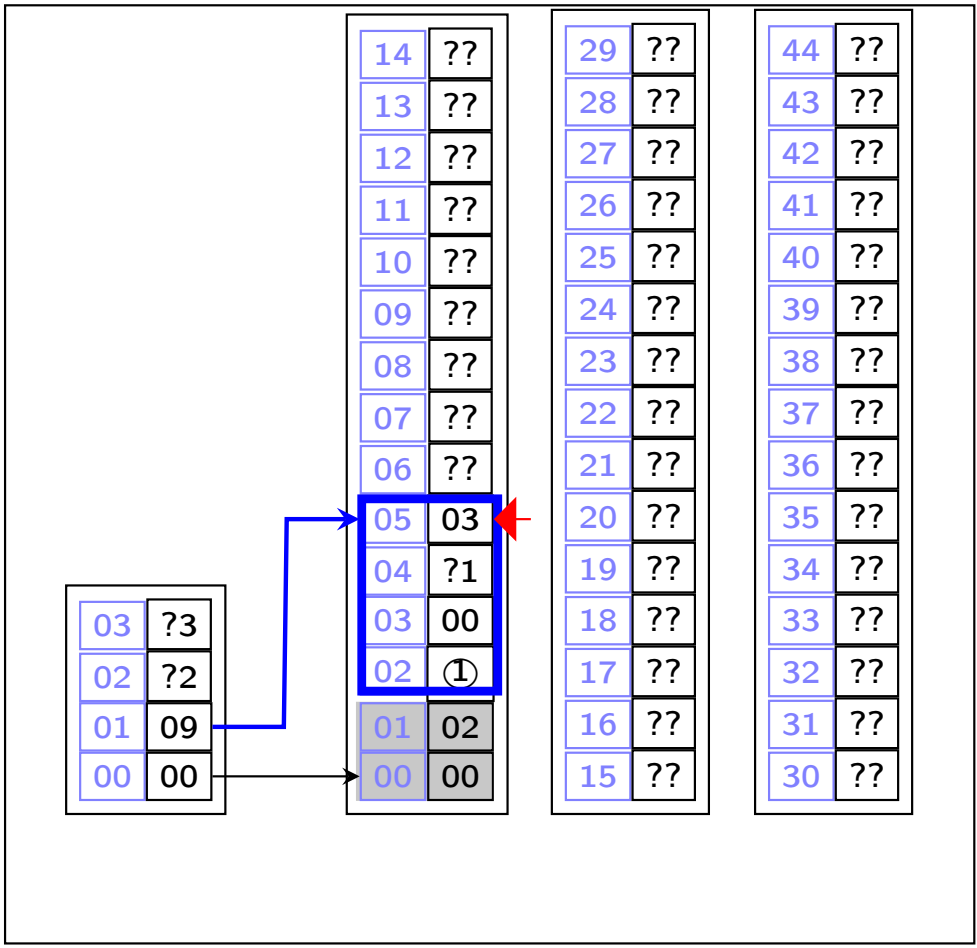
Execução

```
program proc1 (input, output);
  var x, y: integer;
  procedure p;
     var z:integer;
    begin
      z:=x;
      x:=x-1;
      if (z>1)
        then p②
        else y:=1;
      y:=y*z
    end
begin
   read(x);
   p①
   write (x,y)
end.
```



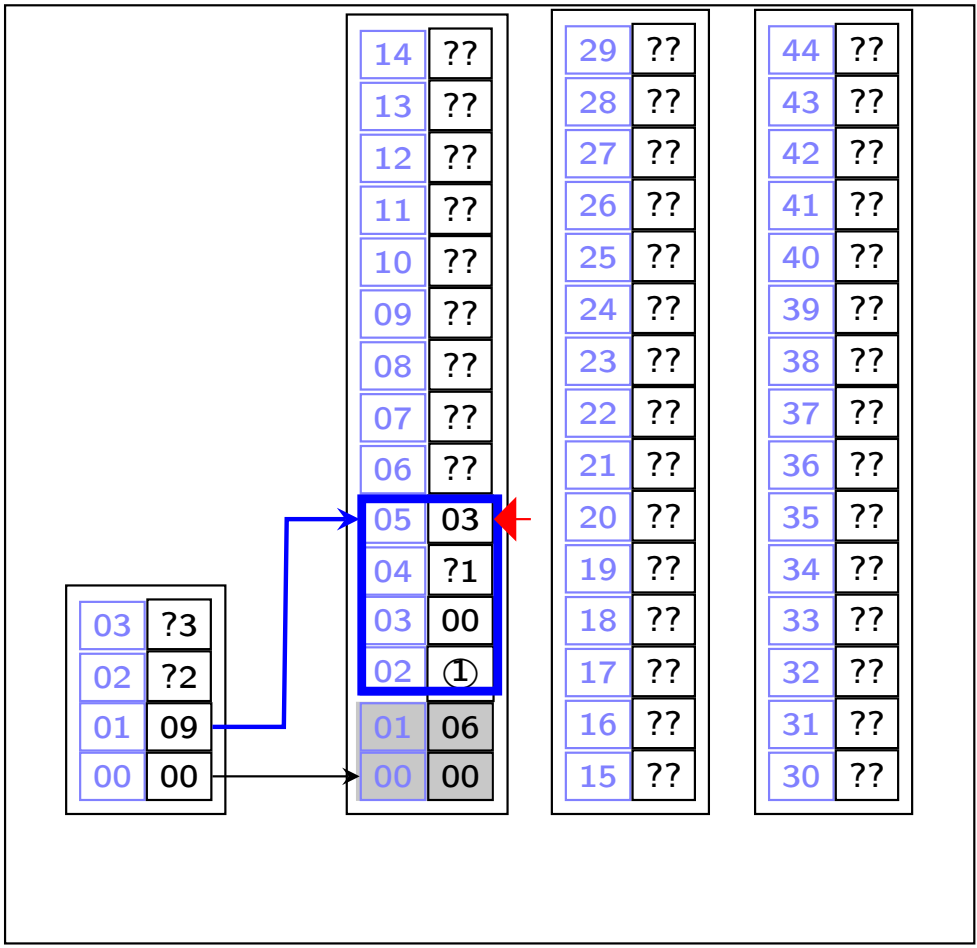
Execução

```
program proc1 (input, output);
  var x, y: integer;
  procedure p;
     var z:integer;
    begin
      z:=x;
      x:=x-1;
      if (z>1)
        then p②
        else y:=1;
      y:=y*z
    end
begin
   read(x);
   p①
   write (x,y)
end.

DMEM 1
RTPR k,n { D[K]:=M[s];
           i:=M[s-2];
           s:=s-(n+3)}
```



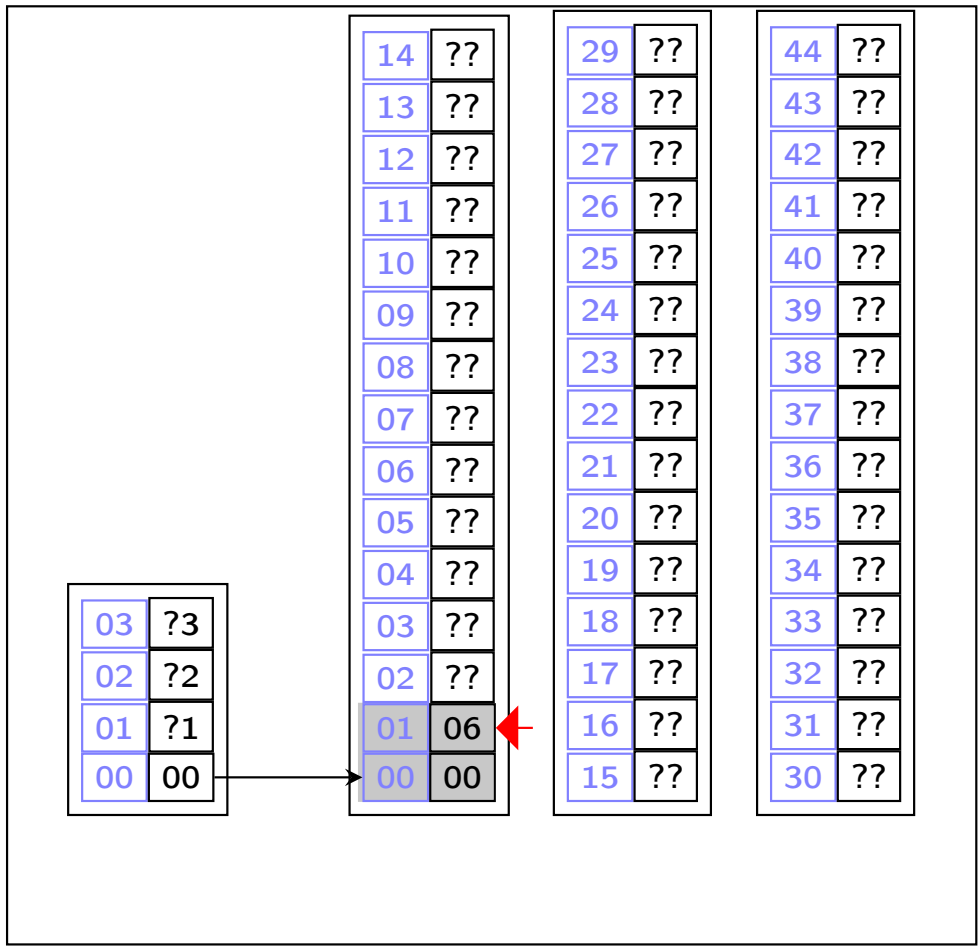
Execução

```
program proc1 (input, output);
  var x, y: integer;
  procedure p;
     var z:integer;
    begin
      z:=x;
      x:=x-1;
      if (z>1)
        then p②
        else y:=1;
      y:=y*z
    end
begin
   read(x);
   p①
   write (x,y)
end.
```



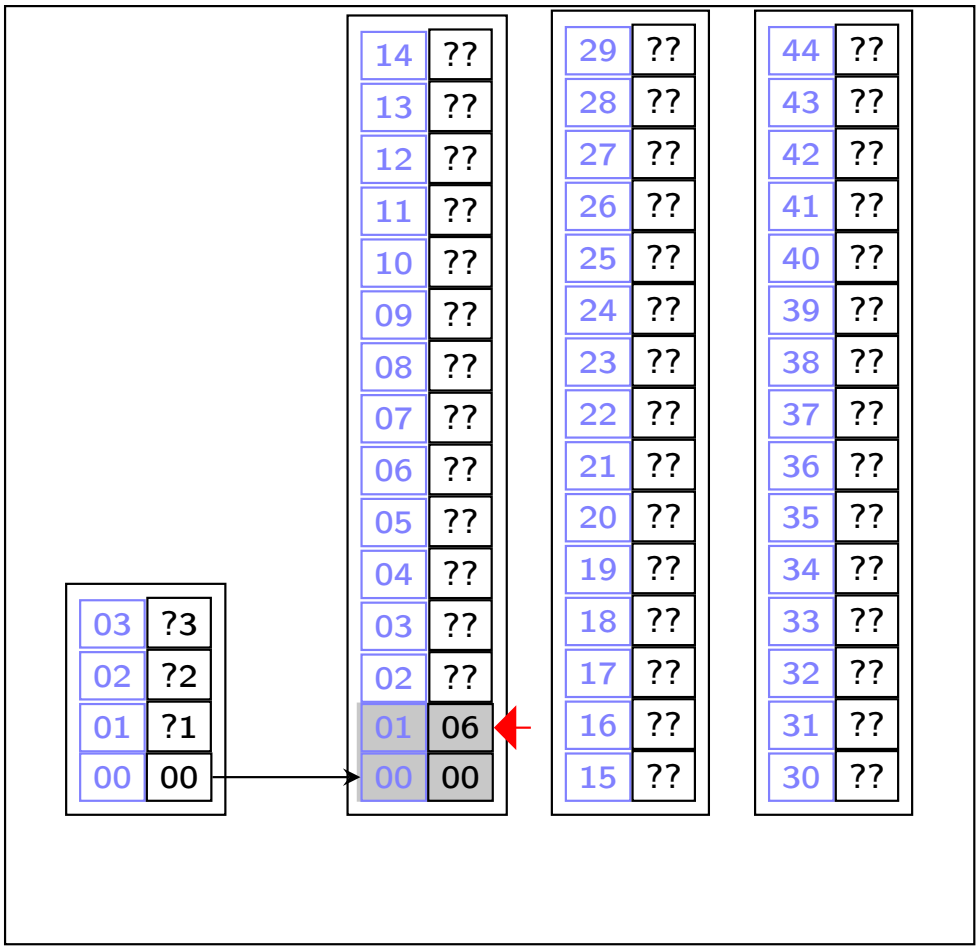
Execução

```
program proc1 (input, output);
  var x, y: integer;
  procedure p;
     var z:integer;
    begin
      z:=x;
      x:=x-1;
      if (z>1)
        then p②
        else y:=1;
      y:=y*z
    end
begin
   read(x);
   p①
   write (x,y)
end.

DMEM 1
RTPR k,n { D[K]:=M[s];
           i:=M[s-2];
           s:=s-(n+3)}
```



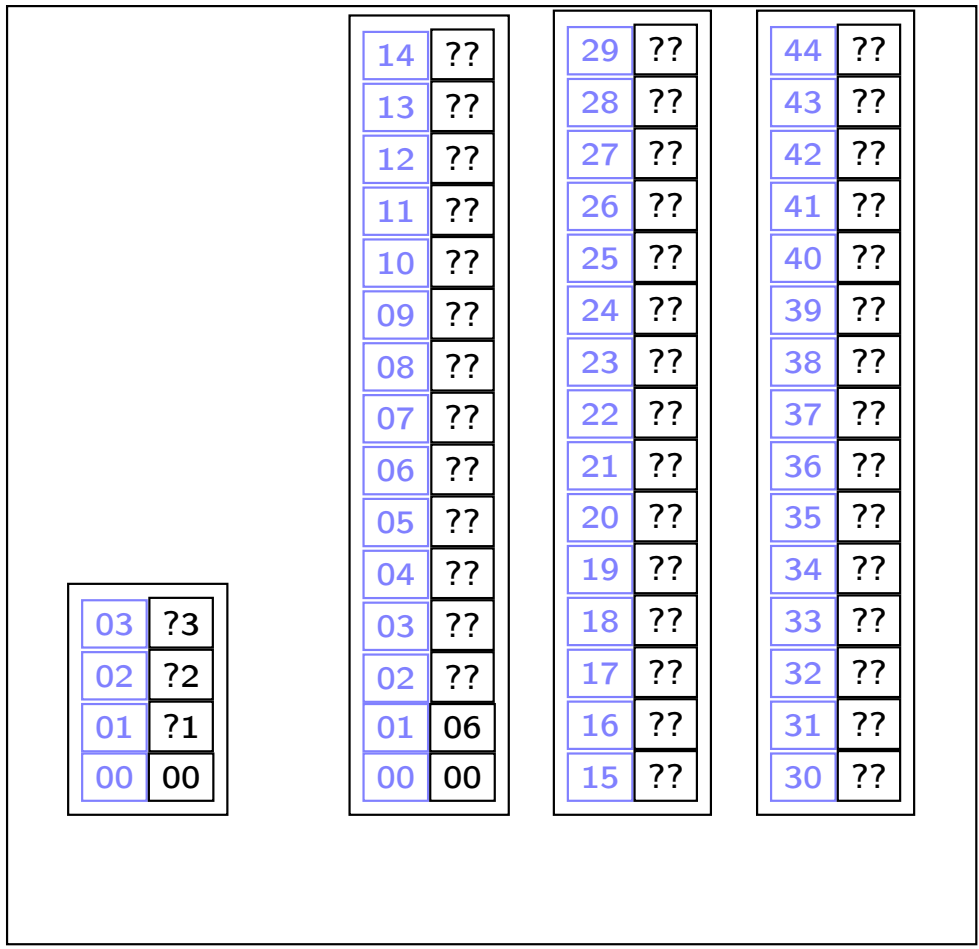
Execução

```
program proc1 (input, output);
  var x, y: integer;
  procedure p;
     var z:integer;
    begin
      z:=x;
      x:=x-1;
      if (z>1)
        then p②
        else y:=1;
      y:=y*z
    end
begin
   read(x);
   p①
   write (x,y)
end.
```



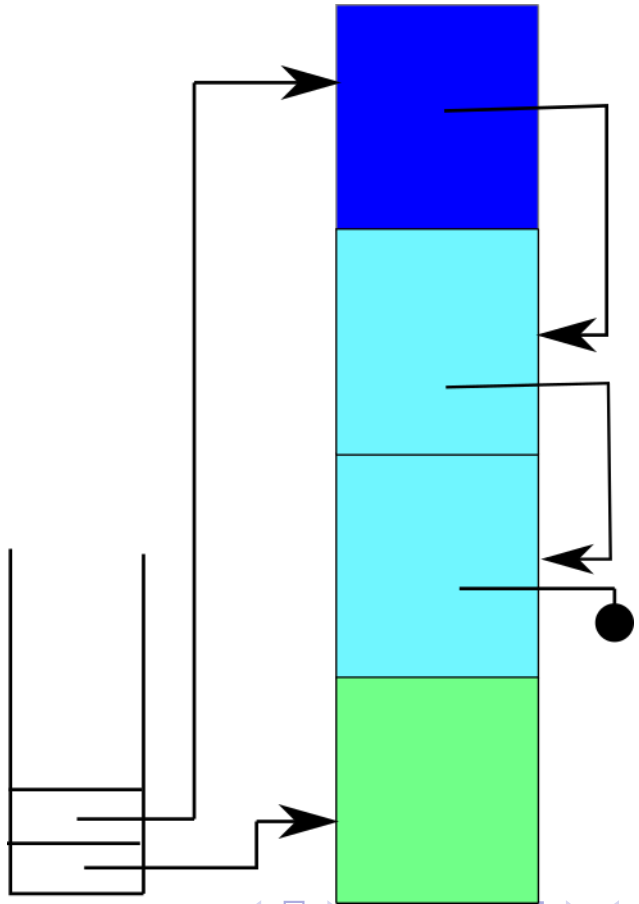
Execução

```
program proc1 (input, output);
  var x, y: integer;
  procedure p;
     var z:integer;
    begin
      z:=x;
      x:=x-1;
      if (z>1)
        then p②
        else y:=1;
      y:=y*z
    end
begin
   read(x);
   p①
   write (x,y)
end.
```



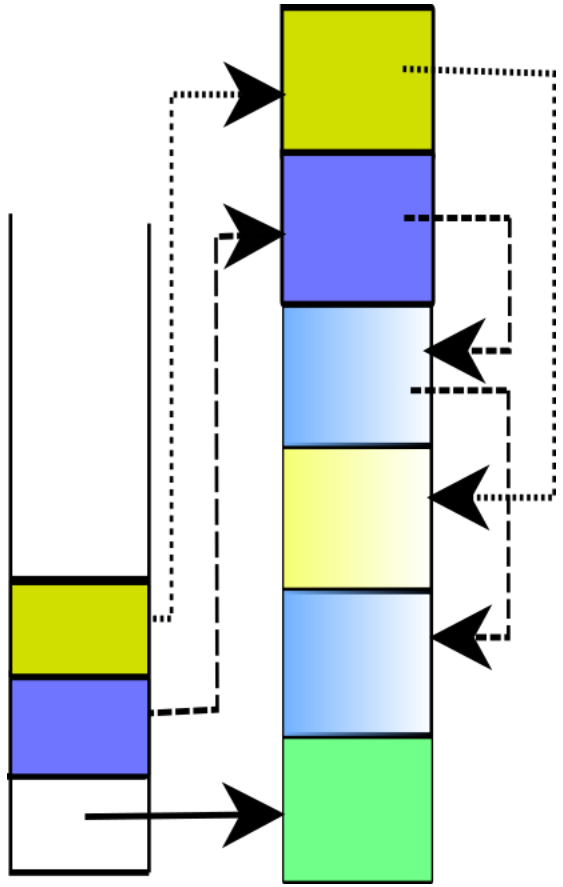
Encadeamento

- Observe que o exemplo indica um encadeamento entre os registros de ativação de nível léxico 1.
- O livro do Tomasz sugere uma analogia com folhas de papel que se sobrepõe.



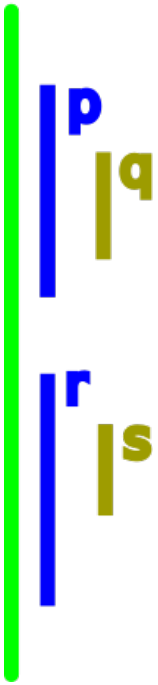
Encadeamento

- O encadeamento ocorre de maneira independente em cada nível léxico.



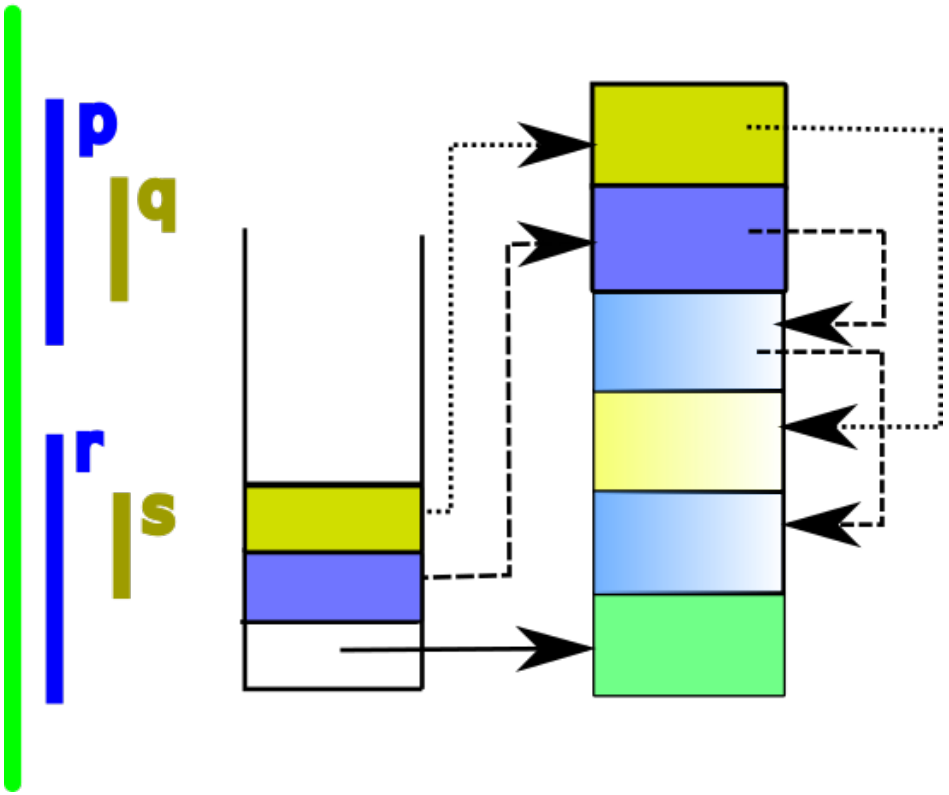
Encadeamento

princ ➔ r ➔ s ➔ r ➔ p ➔ q



Encadeamento

princ ➔ r ➔ s ➔ r ➔ p ➔ q



Projeto

- Observe a gramática abaixo:

```
A ::= <ident> ":="  ... | # Atribuição
      <ident> "("   ... | # Chama Proc (com param.)
      <ident> ";"   ...   # Chama Proc (sem param.)
```

- Ao encontrar um identificador, qual das três regras utilizar?
- O analisador sintático tem que tomar a decisão baseado no símbolo corrente (o identificador), o que é impossível.
- Solução: fatorar a gramática:

```
A ::= <ident> <A-Continua>
A-Continua ::=  ":="  ... | # Atribuição
                "("   ... | # Chama Proc (com param.)
                ";"   ...   # Chama Proc (sem param.)
```

- Acrescente a chamada de procedimentos sem parâmetros ao compilador.
- Utilize a fatoração para diferenciar atribuição e chamadas de procedimento.
- Lembre que ainda falta tratar parâmetros e funções, ou seja: é possível que algumas coisas têm que ser alteradas depois (mas, se existirem estarão mais relacionadas com a tabela de símbolos do que com o próprio analisador sintático).

- Página para anotações

Licença

- Slides desenvolvidos somente com software livre:
  - LaTeX usando beamer;
  - Inkscape.
- Licença:
  - Creative Commons Atribuição-Uso Não-Comercial-Vedada a Criação de Obras Derivadas 2.5 Brasil License. `http://creativecommons.org/licenses/by-nc-nd/2.5/br/`