Programação / Programação I
LEI/1, LTSI/1, LMAT/1

Hugo Pedro Proença

Universidade da Beira Interior
Departamento de Informática

Resumo

- A Linguagem C
  - Instruções de Input / Output
    - printf
    - scanf
  - Instruções de Controlo de Fluxo
    - if
  - Exercícios

Noção de Bloco: Revisão

- Bloco: agrupamento “lógico” de instruções
  - Blocos condicionais
  - Blocos iterativos
  - Blocos iterativos dentro de ...

```
{ instrução 1 ;
  instrução 2 ;
  Instrução 3 ; }
```

Noção de Bloco: Revisão

- Sempre que não se explícita o início/fim de um bloco {...}, assume-se que o bloco é formado por uma única instrução.

```
instrução 1  ;
instrução 2  ;
instrução 3  ;
```

- Neste exemplo ao retirar os símbolos { e }, passámos a ter 3 blocos, cada 1 com uma instrução.
- Anteriormente, tínhamos um único bloco com 3 instruções.

Input / Output: printf()

- “fprintf()”: função genérica para escrita em streams.
  - “printf()”: caso particular da função anterior. A escrita é efectuada no “standard output” (écran)

- SINTAXE:
  - printf(“ qualquer coisa a escrever “);

printf(): exemplo

```
#include <stdio.h>
int main(int argc, char* argv[]){
        printf("Olá mundo!");
}
```

- A execução repetida deste programa irá produzir exactamente sempre o mesmo output (output constante).
- Como fazer para mostrar resultados / mensagens variáveis?

Input / Output: printf()

- Caractér %
  - Serve para explicitar a escrita de uma variável.
- Exemplo:
  - printf("A mensagem é %d",x);
- Neste exemplo estamos a escrever o contéudo de uma variável inteira.
- Exemplo 2:
  - printf("y=%d, z=%f, a=%c",y,z,f);

Input / Output: printf()

Escrita de variáveis:

%d → valores inteiros
%f → valores reais
%.2f → valores reais com 2 casas decimais de precisão
%c → caractér
%s → cadeias de caracteres (strings)

printf(): caracteres de escape

- Através do símbolo \, podem-se escrever alguns caracteres "especiais" no terminal:
  - \n → Mudança de linha
  - \t → Tabulação horizontal
  - \v → Tabulação vertical
  - \a → Alerta sonoro
- Exemplo:
  - printf("linha 1\nlinha2\nlinha3\n");
- A instrução acima irá escrever em 3 linhas distintas.

printf(): exemplos

- printf("O resultado é x/y\n");
  - Aparece no écran exactamente a frase "O resultado é x/y", seguido de uma mudança de linha.
- printf("O resultado é %f\n",x/y);
  - Aparece no écran a frase "O resultado é", seguido do resultado de x / y e finalizado por uma mudança de linha.
- printf("Tabela: %d\t%.3f\n",i,z);
  - Aparece no écran a palavra "Tabela", depois uma variável inteira (i), uma tabulação e um número real (z) apresentado com 3 casas decimais. Finalmente efectua-se uma mudança de linha.

Input / Output: scanf()

- Função para receber valores a partir do standard input (teclado).
  - Pressupõe o registo do valor recebido numa variável.
- Caso particular de outra mais genérica "fscanf()"
- SINTAXE:
  - scanf("%TIPO_DADOS",&VARIÁVEL);
- Exemplo:
  - scanf("%d",&x);
  - Lê um valor inteiro a partir do teclado para a variável (inteira) x.

Input / Output: scanf()

- Apesar de possível, por motivos de compatibilidade não é aconselhável receber os valores de mais que uma variável numa só instrução de scanf().
- Exemplo:
  - scanf("%d",&x);
  - scanf("%c",&y);

  - scanf("%d%c",&x,&y);

Controle de execução: if

- **Na linguagem C, a implementação de blocos condicionais e condicionais exclusivos efectua-se através de instrução "if" e "if...else".**
- **SINTAXE BLOCO CONDICIONAL:**

Avaliação TRUE || FALSE

```
if (TESTE_LÓGICO){
        instrução 1;
        instrução 2;
        ...
        instrução n;
}
```

O bloco composto por estas 3 instruções apenas será executado se o teste lógico for avaliado com valor lógico TRUE.

Controle de execução: if

- **SINTAXE BLOCO CONDICIONAL EXCLUSIVO:**

```
if (TESTE_LÓGICO){
        instrução x1;
        ...
        instrução xn;
}
else{
        instrução y1;
        ...
        instrução ym;
}
```

As instruções x's e y's serão executadas exclusivamente. Se o teste lógico for avaliado com valor TRUE será executada a parte relativa ao "if". Caso contrário, será executada a parte relativa ao "else".

Controle de execução: if

- **Exemplos:**

```
if (x>4){
        printf("X é maior que 4\n");
}
else{
        printf("X NÃO é maior que 4");
}
```

Regra de Identação

- **Analise o seguinte código:**

```
if(x>4)
printf(“frase1\n”);
if((y>7) && (z<6))
printf(“frase2\n”);
else{
if (z>100)
printf(“que confusão!\n”;
else
printf(“não se percebe nada!”);
if(x+y<5)
printf(“devia-se identar o código...”);
else
printf(“final...”);
}
```

Legibilidade Mínima:

- É díficil ver se o código tem erros de erros de sintaxe.
- É díficil verificar qual a funcionalidade pretendida para o código.

Regra de Identação

- **Regra de Identação:**
  - **Blocos integrados dentro de outros devem sempre ser escritos com uma tabulação à direita.**
  - **Aumenta a legibilidade para o próprio programador.**
  - **Torna possível a leitura / percepção por parte de outros elementos.**
    - **Projectos de maior dimensão, trabalhos em grupo / equipa..**

Regra de Identação

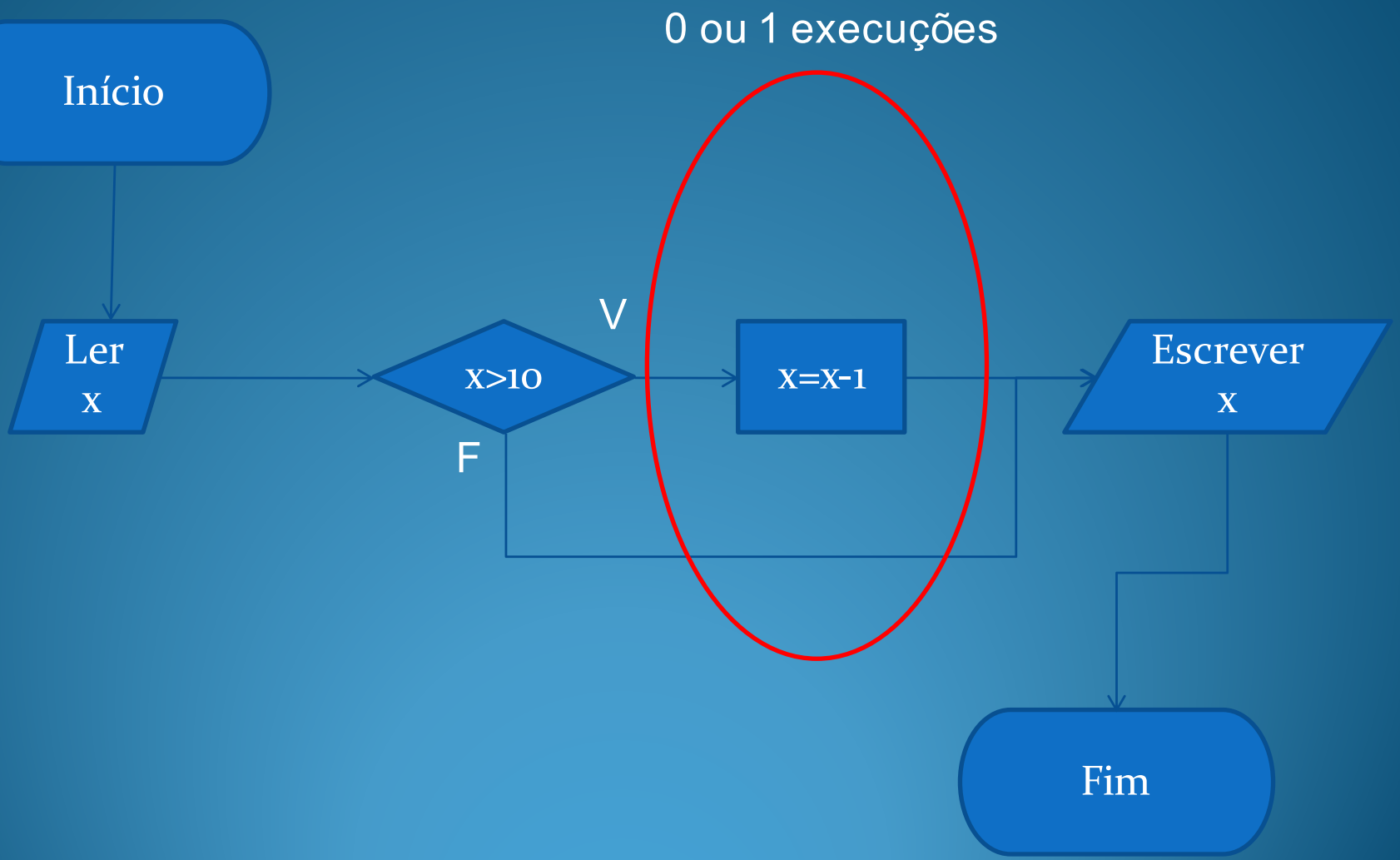
- Exemplo com código identado:

```
if(x>4)
        printf(“frase1\n”);
if((y>7) && (z<6))
        printf(“frase2\n”);
else{
        if (z>100)
                printf(“que confusão!\n”;
        else
                printf(“não se percebe nada!”);
        if(x+y<5)
                printf(“devia-se identar o código...”);
        else
                printf(“final...”);
}
```

Exercícios

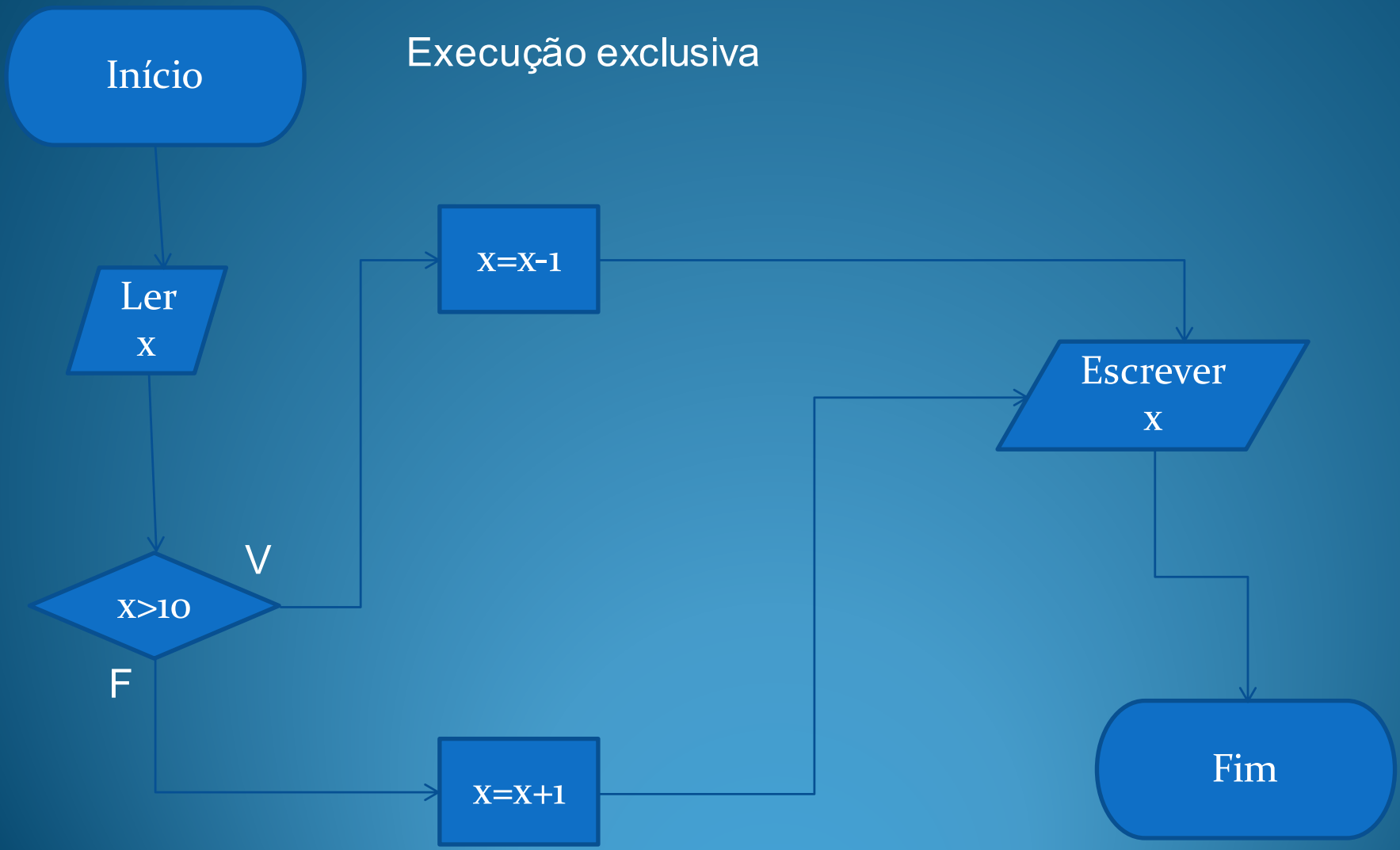
- **Efectue a passagem de cada um dos fluxogramas anteriormente apresentados nas aulas teóricas para código em linguagem C.**

Exercícios

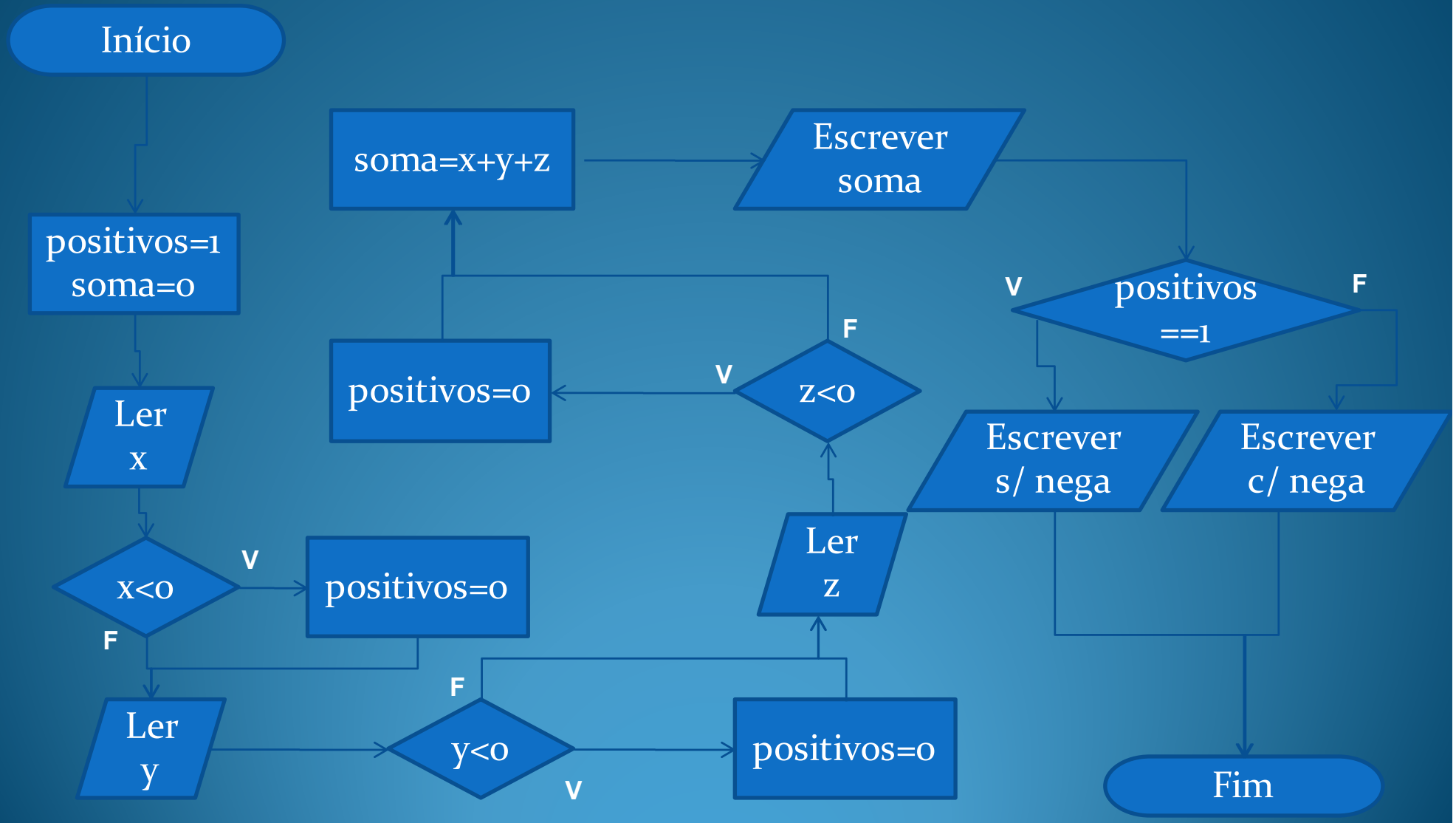


Exercícios

Execução exclusiva



Exercícios


Início
positivos=1
soma=0
Ler
x
x<0
V
positivos=0
F
Ler
y
y<0
F
V
positivos=0
Ler
z
z<0
V
F
positivos=0
soma=x+y+z
Escrever
soma
positivos
==1
V
F
Escrever
s/ nega
Escrever
c/ nega
Fim