A LINGUAGEM C

Parte 3

© Prof. Engº esp Luiz Antonio Vargas Pinto
www.vargasp.com

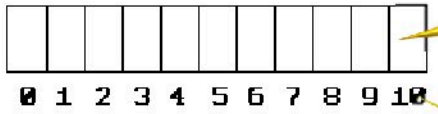
Vetores e Matrizes

- Diferentes de outras linguagens, não existe o tipo string
- Vetor é uma única linha com **n** variáveis
- Matriz é um tabela **m** x **n** de variáveis de um mesmo tipo
- Declaração da matriz:

```
tipo nome[tamanho];
```

Número de elementos

Conteúdo de um único tipo



índice

```
int idade[11];
```

© Prof. Vargas

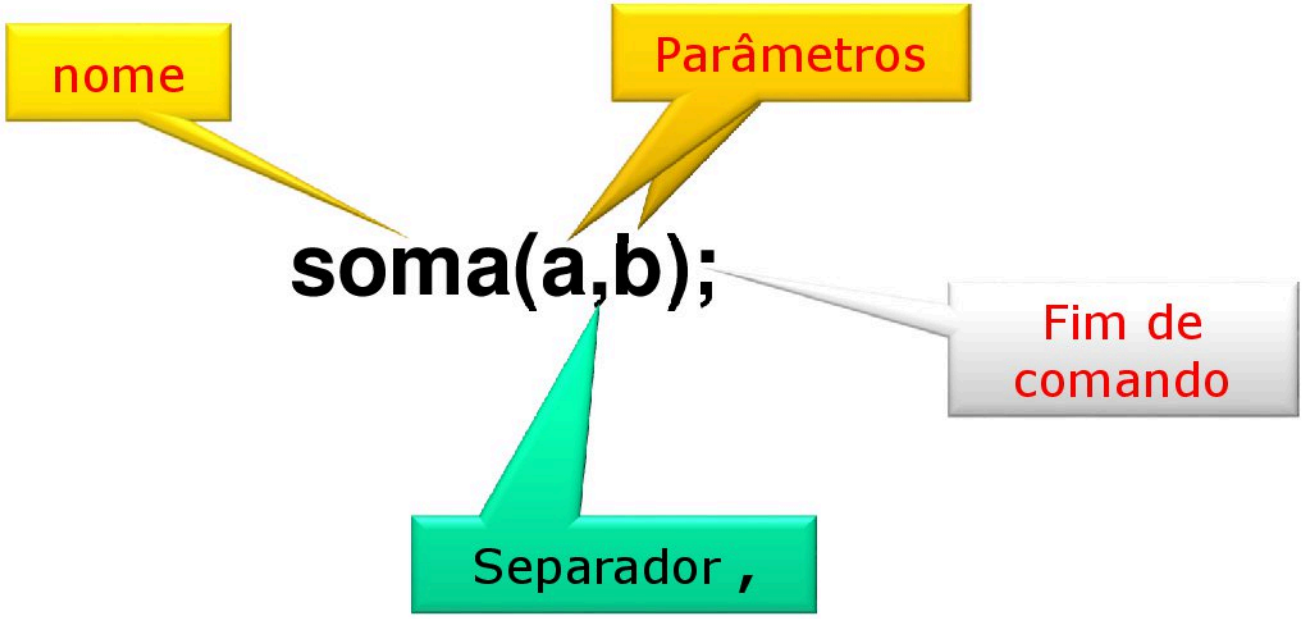
Vetores e Matrizes

- o primeiro índice é **0**
- o último elemento útil é n – 1
  - Por exemplo, **11** elementos implica elementos de **0** a **10**
- Atribuição:
  - Letra[9] = 'A'
- Decisão:
  - if (Letra[**0**] == 'C')
- A linguagem C não confere o limite do vetor

© Prof. Vargas

Funções

- C é formado por funções



© Prof. Vargas

Exemplo

```
int mensagem () // Função simples
{
   PORTB = 0b'00111000';
   return(0);   // sem retorno
}

void main(void)
{
    mensagem();
}
```

Informa o que retorna

- Parâmetros para função são passados pelo stack pointer e a origem normalmente não é alterada.

© Prof. Vargas

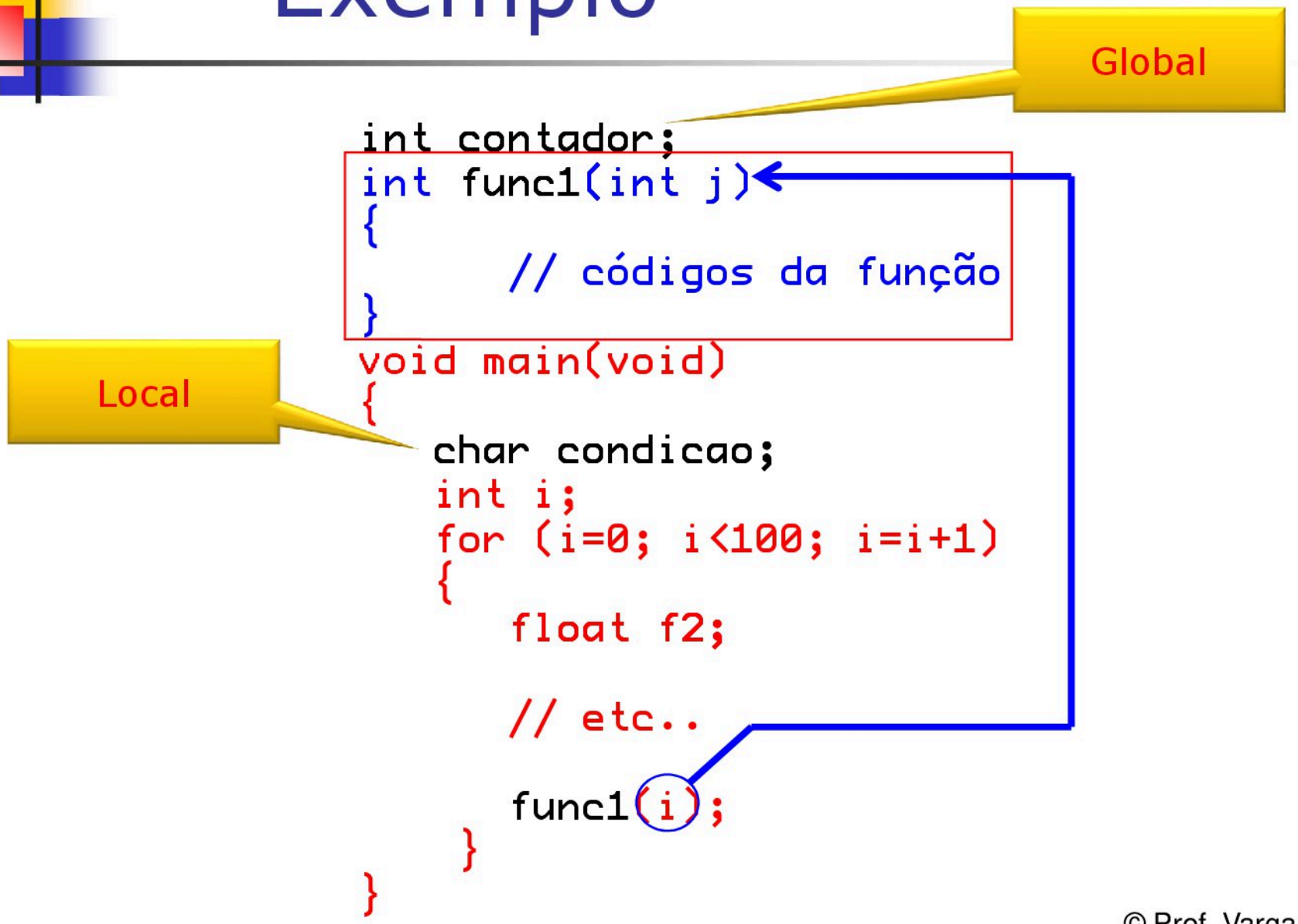
Sintaxe

```
void nome_da_função (parâmetros);

tipo nome_da_função (void);

void nome_da_função (void);
```

- O **void** é o tipo vazio utilizado para evitar códigos adicionais de retorno do Stack

© Prof. Vargas

Exemplo



© Prof. Vargas

Exemplo

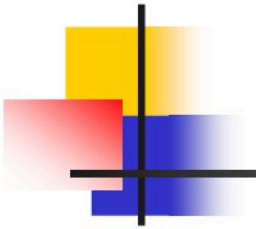
```
int mult (float a, float b, float c)  // Multiplica 3 números
{
   int Edge;
   Edge = a * b * c;
   PORTC = Edge;
   return(0);
}
void main(void)
{
    float x,y;
    x=23.5;
    y=12.9;
    mult (x,y,3.87);
}
```

Varia conforme o compilador

Cabe comentários

© Prof. Vargas

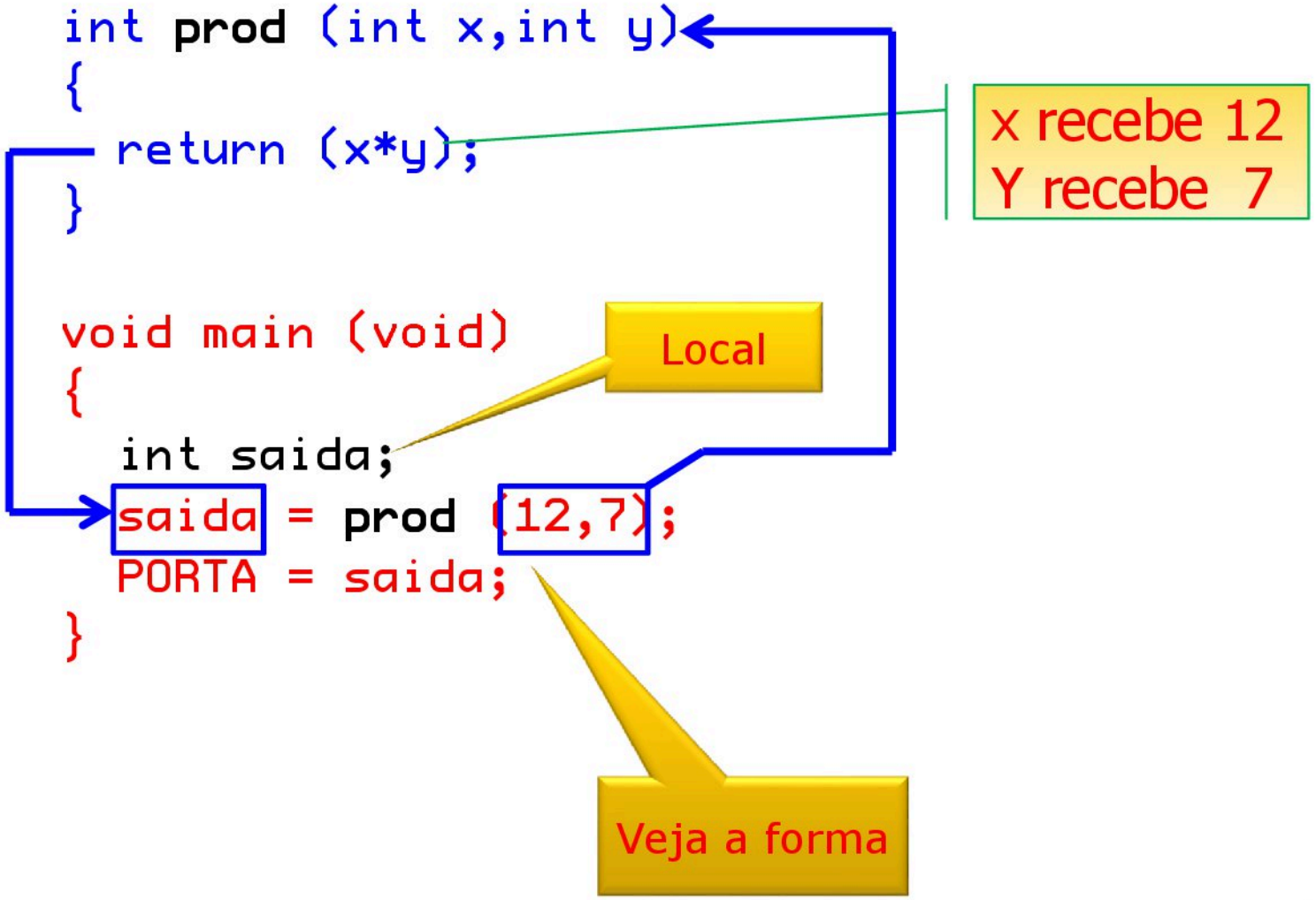
Análise



- x → a
- y → b
- 3.87 → c
- Não existe retorno para as variáveis originantes pois temos `return(0)` o que as mantém com seus valores originais

© Prof. Vargas

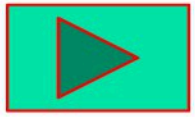
Com retorno de valores



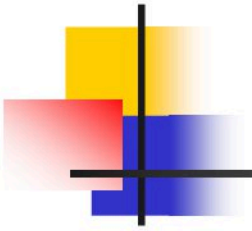
© Prof. Vargas

Análise

1. Se não especificamos o tipo de retorno o compilador considera este tipo inteiro – o que não é boa prática.
2. **main** sempre retorna **int**.
3. Idealmente usaremos `return(0);`
4. O tipo de cada receptor deve ser especificado na função.
5. Para microcontroladores não é necessário o uso do `return();`



© Prof. Vargas



Protótipo de função

- Protótipos são opcionais, mas são exigidos pela sucessora C++.
- Protótipos permitem que C forneça uma verificação mais forte de tipos.

© Prof. Vargas

exemplo

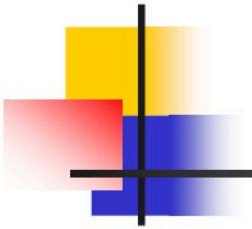
```
int prod (int x,int y);

void main(void)
{
   int saida;
   saida = prod (12,7);
   PORTA = saida;
}

int prod (int x,int y)
{
   return (x*y);
}
```

Protótipo

Código real

Observe o ';'

© Prof. Vargas



Bibliografia

Prof. Airton Kuada
airton@fesppr.br

© Prof. Vargas

APÊNDICE

© Prof. Vargas

Tipos de Dados

- **char, int, float, void, double.**

Tipo	Bytes	Formato para scanf	Intervalo	
			Início	Fim
char	1	%c	-128	127
unsigned char	1	%c	0	255
signed char	1	%c	-128	127
int	2	%i	-32.768	32.767
unsigned int	2	%u	0	65.535
signed int	2	%i	-32.768	32.767
short int	2	%hi	-32.768	32.767
unsigned short int	2	%hu	0	65.535
signed short int	2	%hi	-32.768	32.767
long int	4	%li	-2.147.483.648	2.147.483.647
signed long int	4	%li	-2.147.483.648	2.147.483.647
unsigned long int	4	%lu	0	4.294.967.295
float	4	%f	3,4E-38	3.4E+38
double	8	%lf	1,7E-308	1,7E+308
long double	10	%Lf	3,4E-4932	3,4E+4932

© Prof. Vargas

Operadores

Operador	Ação
&&	And
||	Or
!	Not

Operador	Ação
+	Soma inteira e fracionada
-	Subtração e troca de sinais
*	Multiplicação inteira ou fracionada
/	Divisão inteira e fracionada
%	Resto da divisão
++	Incremento
--	decremento

© Prof. Vargas