APONTAMENTOS DE LINGUAGEM C

© Prof Vargas

www.vargasp.com

A Linguagem C foi desenvolvida na década de 70 e oferece recursos para programação em uma forma mais amigável para o programador.

Fez tanto sucesso que, em conjunto com Assembly foi usada largamente em software base tal como sistemas operacionais UNIX e MSDOS.

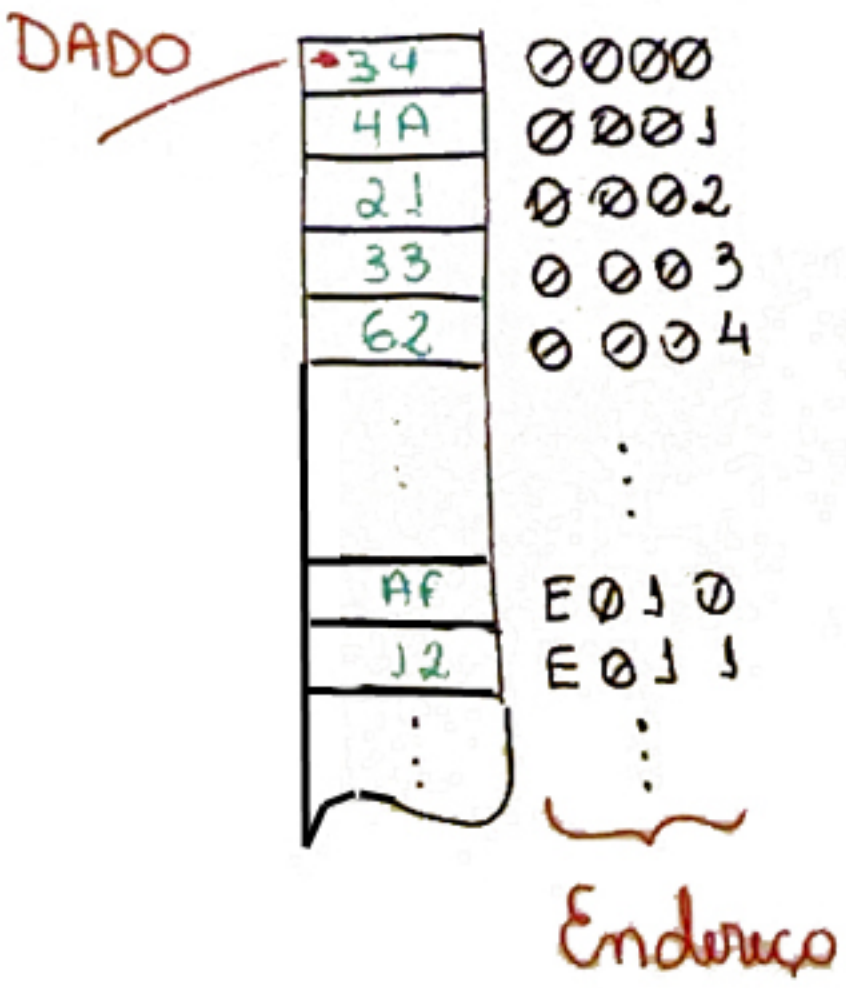
Possui um sistema de compilação, o qual gera Códigos muito compactos o que viabiliza o uso em sistemas com pouca memória que é o caso de microcontroladores

Variáveis e Constantes

Como parte integrante de linguagem estruturadas, C é rigoroso no uso de seus elementos de memória, assim como na definição dos blocos de programa, os quais serão estudados posteriormente.

Considere:

Memória



Ao invés de Colocar um dado (byte) em um endereço de memória, em C guardamos em uma constante ou variável

Exemplo:

```
Mov 10h, #23h
```

Ou em C:

```
i = 0x23;
```

e com o agravante de que i não necessariamente corresponde ao endereço 10h.

Quando definimos uma constante, seu valor NÃO deve ser alterado, isto acontece apenas com variáveis.

C é uma linguagem também denominada de sensitive case, isto é, difere maiúsculas de minúsculas.

Assim:

$$A \neq a$$

```
a = 3;
A = 3;
```
} são dois locais diferentes

Regra básica no nome

Quantidade de caracteres: até 255

a) 1ª Letra: sempre letras sem acentuação

exemplos: A1
Teste
Joao_Oliva

Evite @ # ' " - + / & ( ) [ ] ! ?
> < = *
e símbolos gráficos

b) Procure sempre dar sentido para o nome

c) Identificar o tipo correto para a variáveis/constante

O tipo de uma variável

Em linguagem estruturada em geral, Caso de C, é necessário especificar o tipo de cada variável pois a partir disto o Compilador reserva espaço adequado para elas e checa o uso delas no programa

Os tipos disponíveis são:

char	1 byte	-128 a +127
unsign char	1 byte	0 a 255
int	2 byte	-32768 a 32767
float	4 byte	$3,4\times10^{-38}$ a $3,4\times10^{38}$

OPERAÇÕES aritméticas

+ Soma
- Subtração
* Multiplicação
/ Divisão

Toda Operação deve ser atribuída

Exemplo:

```
S = A + B ;
    |   |
    |   └→ fim
    |
    └→ Soma
└→ Atribuiçao
```

Obs.:

S deve ser devidamente escolhido de acordo com a operação

ENTRANDO Com Dados

Comando scanf( )

&nomedavariável

scanf ("parâmetros do comando", variáveis);

%d → Valor decimal

%f → Valor com precisão 0,3 2×10^8

f = floating Point

%c → caracter ou byte

%s → String

Caracteristicas de um programa C

- CABEÇALHO
- CORPO

Bibliotecas (Headers)

INCLUDES

Declarações

} Cabeçalho

PARÂMETRO

main ( ) ← NOME DO BLOCO

{ ← início

i = 30 ; → fim de comando

} ← Fim

Exemplo:

```
#include <stdlib.h>
# include <stdio.h>

int ze ;

main()
{
    scanf ("%d", &ze);
}
```

- O indicador de Comentários.

  Sempre, após // temos dados que o compilador ignora mas serve de referência

- O final de um comando é sempre com ;

- Se o comando requer mais de uma ação é comum iniciar com { e terminar Com }

Decisão

A técnica de Árvore binária é, até hoje, a melhor opção. Consiste no fato que se tomamos uma decisão, é óbvio que ignoramos outra, e assim por diante.

Especialistas provam que o maior "gargalo" no processamento são as DECISÕES e isto as acelera.

Laços de Repetição

Também chamados de "loop", termo mais Comum na área. São estruturas Computacionais usadas para repetir procedimentos. Isto envolve a Contagem do número de vezes, o que pode ser manual (Controlada pelo programador) ou Automática (controlada pela instrução).

1- AUTOMÁTICA

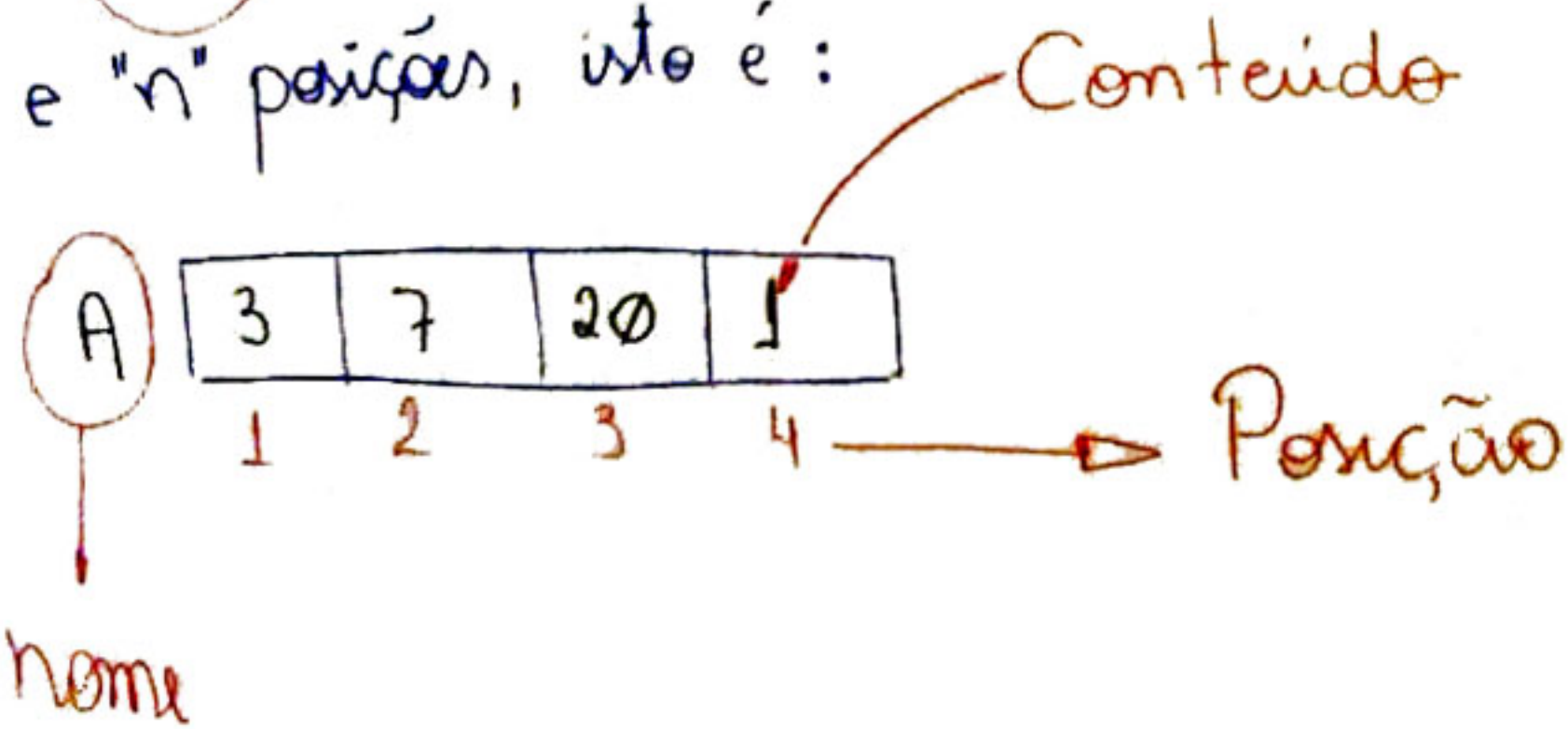
Característica estruturada: Crescente ou decrescente
Finita
step unitário
(de um em um)

Matrizes e Vetores

Quando se trata de operacionalidade, o uso de matrizes e vetores é mais aplicável do que variáveis adimensionais.

Vejamos:

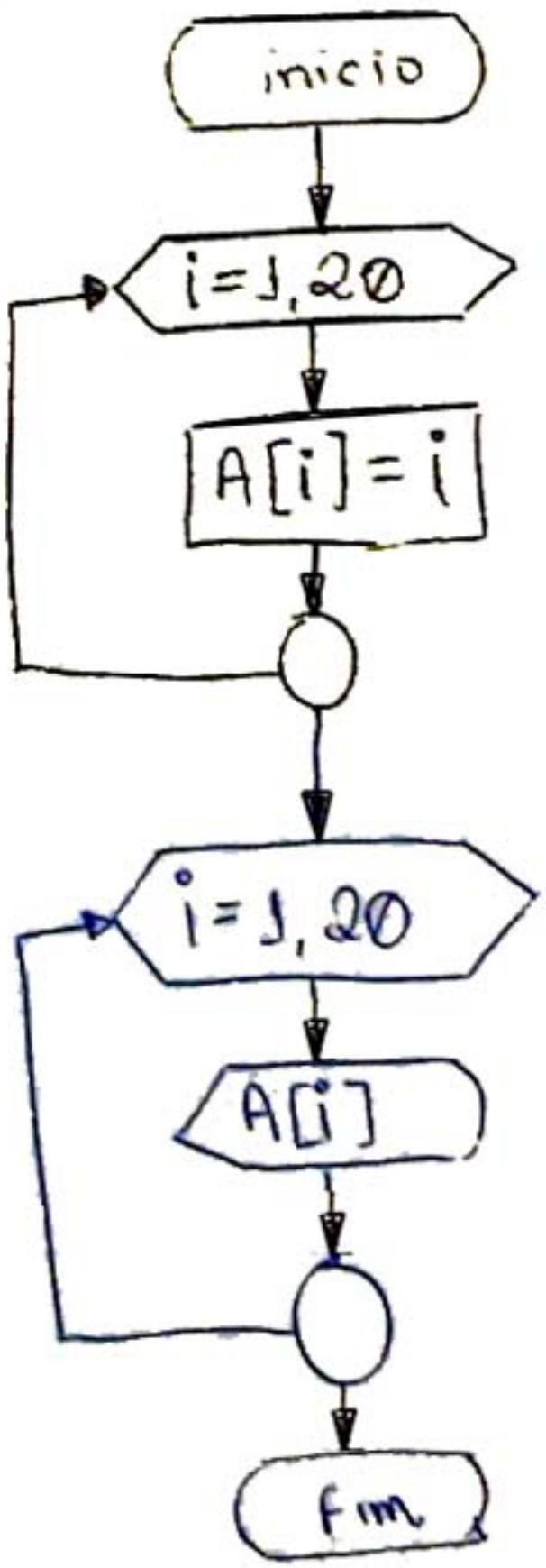
Um (Vetor) é um elemento computacional com um nome e "n" posições, isto é:



Referimo-nos a cada elemento pela forma

$$A[1] = 3;$$

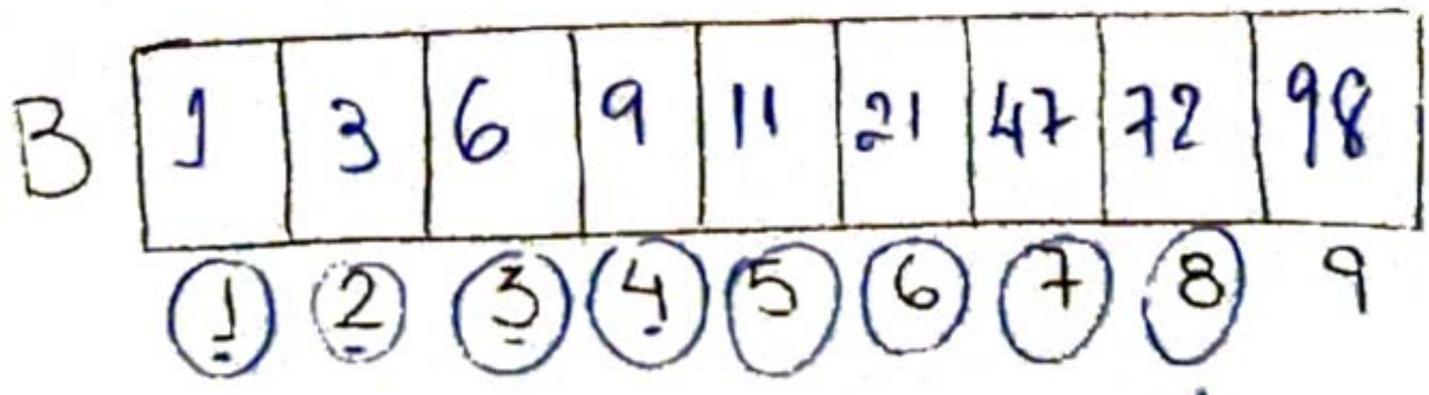
Exemplo:



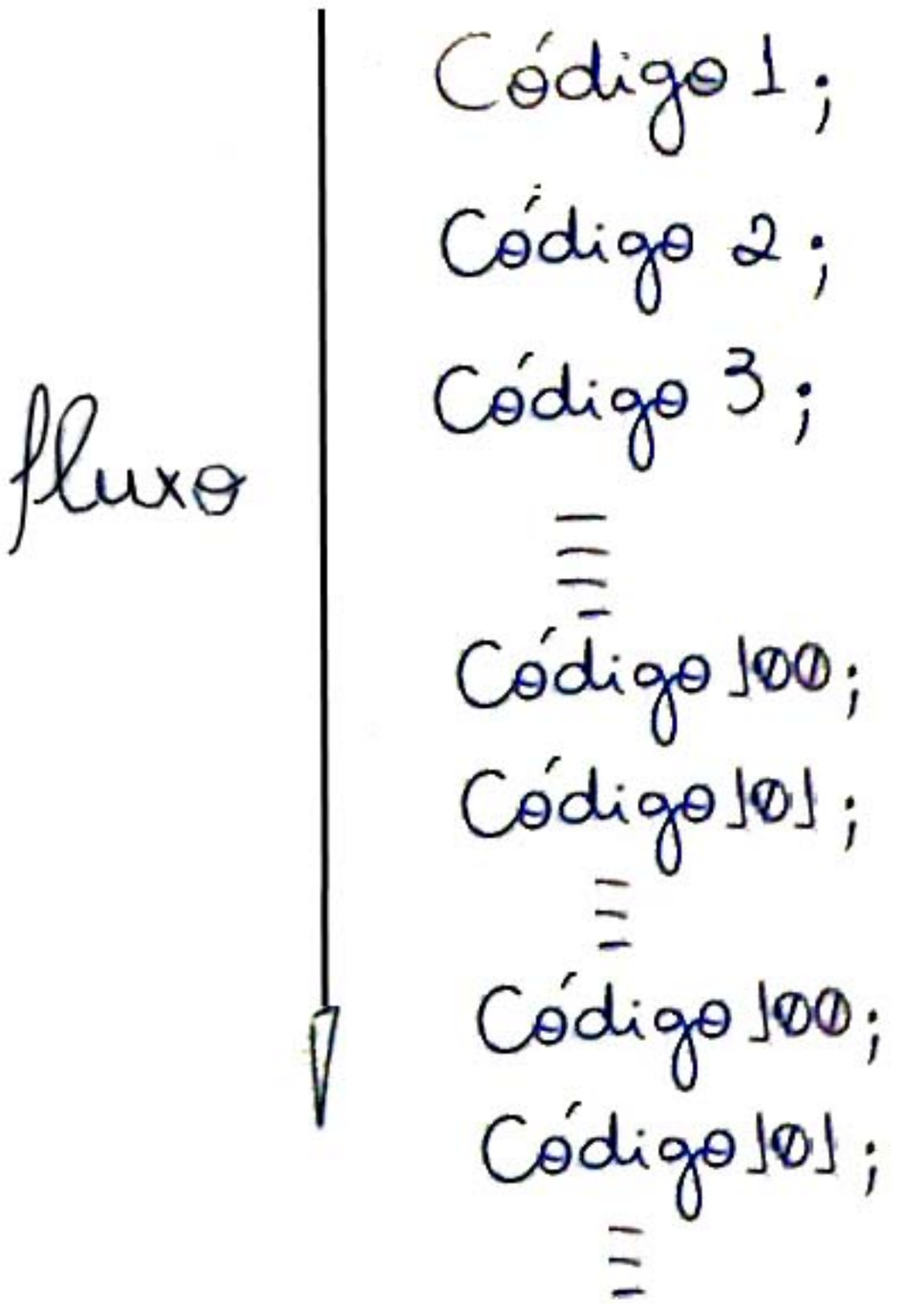
Processo de Sort

Ordenação é um processo muito útil em muitas aplicações.

Consideremos o seguinte caso:



Consideremos o fluxo das informações em um programa procedural* como segue:



* opção:
programação orientada a objetos e eventos

A repetição dos códigos 100 e 101 em várias partes do programa.

gera:

- programas desnecessariamente longos
- Má utilização de código

e traz como consequência:

- Lentidão (processamento excessivo)
- Dificuldade de correção
- Muito desperdício de espaço

Sub-rotina

O que é.

São compostas por partes do programa cujo critério serve para acomodar códigos repetidos ou partes importantes do Conjunto. Isto também permite usar as técnicas de modularidade.

Porque usar

* Porque reduzem os códigos
* Porque organizam melhor o espaço
* Porque Modularizam

Quais os tipos?

1- Passagem de valor por parâmetro

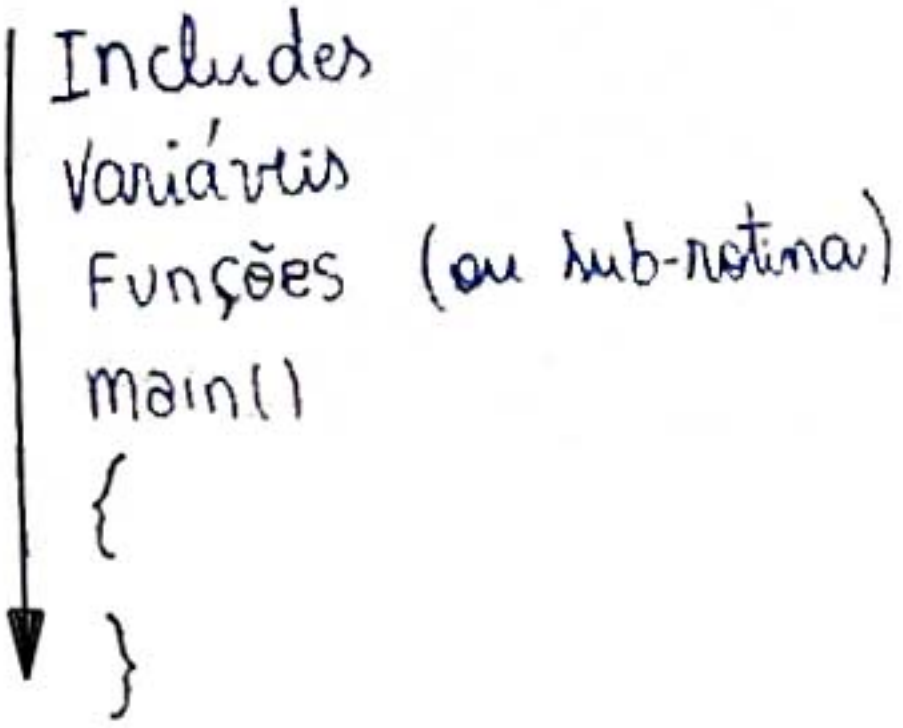
2- Passagem por referência

1- Valor por parâmetro

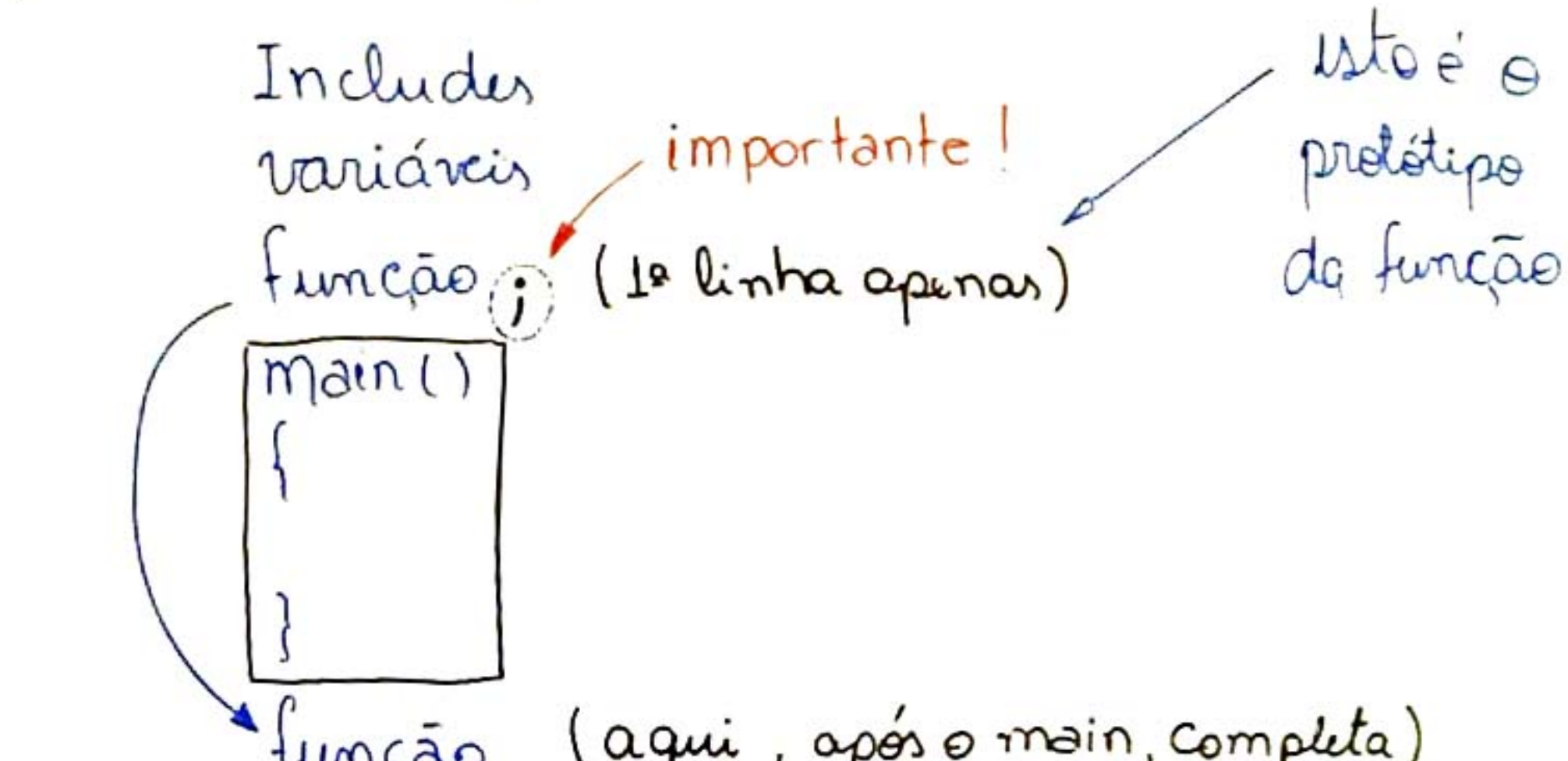
parâmetros são "Complemento" de um Comando por exemplo Raiz Quadrada -> de quem?

* Protótipo de função

C usa o conceito só para organizar-se. Por exemplo, a estrutura convencional é



Não sendo o convencional, teremos protótipo, ou seja:



Exemplo:

```
#include <stdio.h>
float sqr (float num);
main()
  float num, seq;
  printf ("entre com o numero = ");
```

```
    scanf("%f", &num);

    sq = sqr(num);
    printf(" Raiz de %f = %f \n", num, sq);
    system("pause");
}
float sqr(float num)
{
    num= num*num;
    return num;
}
```