

Qualificador `static`

- Indica que uma variável local deve ser armazenada na área de dados e seu conteúdo preservado durante todas as execuções da função.
- A inicialização é feita apenas uma vez.
- Exemplo: `exm-static-1.c`

Qualificador extern

- Declara uma variável que foi definida em outro arquivo.

F. com número var. de parâmetros

- C permite declarar funções com número variável de parâmetros, representados pelas reticências na definição da função.
- `nome (par1, par2, ...) ;`
- Para determinar o número de parâmetros na lista ... podemos usar três métodos:
 - Usar uma cadeia de formato:
`printf ("%d %f\n", i, f) ;`
 - Especificar o número de parâmetros:
`soma (3, -4, 9, 6) ;`
 - Especificar um terminador:
`soma (3, -4, 9, 6, 5, 0) ;`

stdarg.h

- Define tipos e macros para acesso aos parâmetros.
- `va_list`: tipo pré-definido para declarar um apontador para os parâmetros.
- `va_start(va_list p, ultimo_par_def)`: inicia o apontador `p` para o primeiro parâmetro da lista, que é o primeiro parâmetro depois de `ultimo_par_def`. `ultimo_par_def` é o nome do último argumento especificado na declaração da função

stdarg.h

- `va_arg(va_list p, tipo)`: retorna o valor do parâmetro apontado por `p`, de um certo tipo, e faz `p` apontar para o próximo parâmetro da lista.
- `void va_end (va_list p)`: encerra o acesso à lista de parâmetros. Deve sempre ser chamada no final da função.
- Exemplo: `exm-parvar-1.c`

Parâmetros para main

- É possível passar parâmetros diretamente para a função main.
- Os parâmetros são passados na forma de um vetor de strings.
- `main(int argc, char *argv[])`
 - `argc` é o número de parâmetros
 - `argv` é o array de parâmetros
- O primeiro parâmetro é o nome do programa e o último é 0.
- Exemplo: `exm-parmain-1.c`

Funções como parâmetros

- É possível passar uma função como parâmetro para outra.
- O protótipo deve coincidir com a definição do parâmetro.
- Exemplo: `exm-parfunc-1.c`