lista de argumentos variáveis em C usando va_list

- Talvez você queira ter uma função que aceite qualquer número de valores e então retorne a média.

- Você não sabe quantos argumentos serão passados para a função.

- Uma maneira de fazer a função seria aceitar um ponteiro para uma matriz.

- Outra maneira seria escrever 1 função que pode receber QUALQUER número de argumentos.

- Então você poderia escrever:

avg (4, 12, 2, 23, 3, 33, 3, 12, 1)

ou até mesmo...

avg (2, 2.3, 34.4);

int

float

- A vantagem dessa abordagem é que é muito mais fácil alterar o código se você quiser alterar o número de argumentos.

- Alguma funções de biblioteca podem aceitar uma lista variável de argumentos (ex: printf)

• Sempre que uma função é declarada com um nº indeterminado de argumentos, no lugar do último argumento você deve digitar ... p/ dizer ao compilador p/ aceitar quantos argumentos o programador usar, desde que seja igual a pelo menos 1, sendo o 1º, x.

Exemplo:

```
int a_function(int x, ...);
```

1º argumento é x.

... diz p/ ele aceitar quantos argumentos eu quiser usar.

• Para isso, precisamos usar MACROS (que funcionam como funções) da biblioteca stdarg.h p/ extrair os valores armazenados no argumento variável

① va_start
(que inicializa a lista)

② va_arg
(que retorna o próximo argumento na lista)

③ va_end
(limpa a lista de argumentos variáveis)

• P/ usar essas funções, precisamos de 1 variável capaz de armazenar 1 lista de argumentos de tamanho variável — essa variável será do tipo va_list.

↳ é como qualquer outro tipo!

Exemplo:

```
va_list a_list;
```

① va_start

É uma macro que aceita 2 argumentos:

→ 1 va_list ⊕ nome da variável que precede diretamente as reticências ("...").

Dessa forma, na função a_function, p/ inicializar a_list com va_start, você escreveria:

```
va_start(a_list, x);
```

→

```
int a_duvidas (int x, ...)
{
    va_list a_list;
    va_start (a_list, x);
}
```

② va_arg recebe um va_list e um tipo de variável, e retorna o próximo argumento na lista no formo de qualquer tipo de variável informado.

↳ Em seguida, ele se move pl baixo na lista pl o próximo argumento.

Exemplo:

```
va_arg (a_list, double)
```

↳ retornará o próximo argumento, supondo que ele exista, no formo de um double.

Na próxima vez que for chamado, ele retornará o argumento APÓS o último número retornado, se houver.

Observe que você precisa saber o tipo de cada argumento — isso é o motivo pelo qual printf requer uma string de formato.

③ Quando terminar, usar va_end para limpar a lista!

```
va_end(a_list);
```

```
#include <stdarg.h> *
#include <stdio.h>

// Essa função deve levar o número de dados à média seguido de todos os números até a média

double average (int num, ...)
{
    va_list arguments;
    double sum = 0;

    // Inicializando argumentos pl armazenar todos os valores após num
    va_start (arguments, num);

    // Somar todos os valores, ainda contando com a chamada da função pl nos dizer
```

quantos há:

```
for (int x = 0; x < num; x++)
{
    sum += va_arg(arguments, double);
}
va_end(arguments);
// Encerra a lista

return sum/num;
}
```

```
int main()
{
    // Isto computa a média entre 13.2, 22.3, 4.5 (3 indica o número de valores p/ a média)

    printf("%f\n", average(3, 12.2, 22.3, 4.5));

//Aqui é computada a média de 5 valores: 3.3, 2.2, 1.1, 5.5 e 3.3:

    printf("%f\n", average(5, 3.3, 2.2, 1.1, 5.5, 3.3));
```