**Aula Prática Nº 5**

## Objectivos

* Passagem de argumentos a programas em C.
* Bibliotecas de sistema[[1]](#footnote-1)
* Tratamento de erros
* Acesso a variáveis de ambiente

## Guião

1. Leia atentamente o código-fonte args1.c.
   1. Gere o ficheiro executável args1 (comando gcc -Wall -o args1 args1.c). Execute-o passando-lhe diferentes conjuntos de argumentos e interprete o texto que ele imprime em cada caso.
   2. Altere o programa anterior de modo a que funcione apenas quando recebe dois argumentos; em caso de número incorrecto de argumentos, deve imprimir uma mensagem de alerta ao utilizador e terminar devolvendo um valor de erro.
   3. Com base no código anterior e na rotina de conversão numérica atof, crie o programa calculadora.c para simular uma calculadora simples, capaz de cinco operações ('+', '-', 'x', '/', 'p') com números reais. Contemple validação de argumentos – devem ser exactamente três, como no exemplo seguinte:

./calculadora 5.0 p 2.0.

(neste cálculo – potência 5.02.0 – o programa deve imprimir o valor 25.0).

* 1. Constate as lacunas do programa anterior quanto a validação de argumentos. Melhore-o recorrendo à função strtod; explore as suas vantagens para esse efeito (leia com atenção o manual da função).
  2. Justifique o comportamento do programa quando utilizado da seguinte forma (note que '\*' não designa uma operação válida: o símbolo de multiplicação da calculadora é 'x'):

./calculadora 5.0 \* 2.0

1. Leia atentamente o código fonte args2.c.
   1. Crie o ficheiro executável args2 (gcc -Wall -o args2 args2.c); execute o programa passando-lhe várias palavras como argumentos e interprete o resultado.
   2. Crie uma nova variável de ambiente na *bash* designada NEWUSER e altere o programa anterior de modo que a informação sobre o utilizador provenha desta nova variável. Comprove, testando o programa com valores distintos da variável NEWUSER.
   3. Crie um programa (joinWords.c) que junte todos os argumentos numa única frase e a imprima no terminal. Este problema pode ser resolvido de múltiplas formas. Aqui, pretende-se que crie um *array* de caracteres com todo o texto, explorando as rotinas de manipulação de *strings* das bibliotecas de sistema (string.h).
   4. Implemente uma nova versão do programa anterior, designada joinWordsText.c, em que todos os argumentos que não comecem por uma letra sejam ignorados. Explore as rotinas de manipulação de caracteres das bibliotecas de sistema (ctype.h).
2. Programe um jogo que consiste em adivinhar um número inteiro. O programa (altobaixo.c) deve gerar aleatoriamente, numa gama com limites passados como argumentos pelo utilizador, um valor secreto – sugere-se, para esse efeito, a função rand(). Em seguida, deve responder a cada valor inserido, informando se é mais alto ou mais baixo que o valor secreto, repetindo o processo até o utilizador acertar. Deve contabilizar o número de tentativas e indicá-lo no final.
3. Elabore um programa, sortWords.c, que ordene alfabeticamente todos os argumentos recebidos que comecem por uma letra e os imprima ordenados no final. A ordenação será crescente ou decrescente de acordo com uma variável de ambiente (SORTORDER) que terá que definir na *bash* e consultar dentro do programa.
   1. Altere o programa de modo que a ordenação ignore a diferença entre letras maiúsculas e minúsculas.
   2. Compare o funcionamento do seu programa com o programa sort (man sort) e implemente uma nova versão (sortWords2.c) em que as palavras são pedidas ao utilizador e não argumentos do programa.

1. Consulte o Cap. 2 no endereço <http://www.acm.uiuc.edu/webmonkeys/book/c_guide/> para estudar as bibliotecas de sistema mais importantes para este guião. [↑](#footnote-ref-1)