Introduction to C Programming

Luís Nogueira, Paulo Baltarejo Sousa

{lmn,pbs}@isep.ipp.pt

2024/2025

Notes:

- From exercise 9 on, each exercise should be solved in a modular fashion. It should be organised in two or more modules and compiled using the rules described in a Makefile
- If you got some logical errors, you should debug the program (using GDB) before asking for help

1. Implement a C program that displays the size of the following data types:
   - `char, int, unsigned int, long, short, long long, float, double`
2. Implement the function `int sum(int a, int b)` that returns the sum of two integers.
3. Implement the function `int mul(int a, int b)` that returns the multiplication of two integers. The result must be calculated by successive sums (for that invoke the function developed in the exercise 2).
4. Implement the function `int sum_digits(int n)` that returns the sum of the given number digits. Invoke the function developed in the exercise 2 to sum two digits.
5. Implement a function `int cmp(int a, int b)` that returns:
   - `-1 if a < b`
   - `0 if a == b`
   - `1 if a > b`
6. Implement the function `int get_greater_digit(int n)` that returns the greater digit in a given integer number. Invoke the function developed in the exercise 5 to compare two digits.

7. Implement a function `int get_ascii_code(char c)` that returns integer ASCII code of the given character. For example, if the input is `'a'`, the return will be `97`.

8. Implement the function `char get_ascii_char(int c)` that returns the corresponding char given its integer ASCII code.

9. Review the document "Modules and Makefiles"available in Moodle. Adapt the sample (prog_avg) program as follows:
   - Add the following function in "`average.c`":`int average_array (int v [], int n)` which calculates the average of the n integer numbers of array v;
   - The "`main.c`" file should be changed to also invoke this new feature;
   - Create a Makefile to specify the construction of `prog_avg`.

10. Create a Makefile to compile the following program. Analyze and justify the output that the program produces.

```
/***** File main.c *****/
#include <stdio.h>
#include "size_string.h"
int main() {
  char x[] = "I will master ARQCP";
  printf("Size =%u\n", sizeof(x));
  printf("Size =%u\n", size_string_wrong (x));
  printf("Size =%u\n", size_string_correct(x));

  char y[25] = "I will master ARQCP";
  printf("\n Size =%u\n", sizeof(y));
  printf("Size =%u\n", size_string_wrong (y));
  printf("Size =%u\n", size_string_correct(y)); return 0;
}

/***** file size_string.h *****/
unsigned int size_string_wrong (char s[]);
unsigned int size_string_correct (char s[]);

/***** file size_string.c *****/
unsigned int size_string_wrong (char s[]) {
  return sizeof(s);
}

unsigned int size_string_correct (char s[]) {
  unsigned int cont=0;
  while(s[cont]!=0)
     cont++;
  return cont;
}
```

11. Implement the function `int string_to_int(char str[])` that transforms a string into an equivalent integer value.
    - For example, the string `"12345"` must be transformed into the integer `12345`.
    - The solution should to include the function developed in exercise 7.
    - In this exercise do not use the `atoi()` function.

12. Implement two functions, `int integer_part(char x[])` and `int fractional_part(char x[])`, both receiving a string representing a real number, and one that returns an integer referring to the integer part of the received number and another that returns an integer representing the fractional part.
    - Example:

      ```
      char x[] = ‘‘123.456’’;
      int x_int = integer_part(x);   /* assigns 123 to x_int */
      int x_frac = fractional_part(x);  /* assigns 456 to x_frac */
      ```

    -
    - The solution should to include the function developed in the exercise 11.

13. Implement the function `int count_char(char str[], int c)` that receives a string and an integer ASCII character code and returns the number of times that such character appears into the given string.
    - The solution should include the function developed exercise 8.

14. Implement the function `int count_value(int vec[], int n, int value)` that counts the number of times that a `value` appears in an array `vec` with `n` elements.
    - The solution should include the function developed exercise 5.

15. Implement the function `int count_words(char str[])` that receives a string and returns the number of words in the string. Consider that words are separated by a single space.

16. Implement a function `int fake_hash(char str[])` that receives a string and returns an integer number that is the sum of all string characters.
    - The solution should include functions developed in exercises 7 and 2.

17. Implement the function `int check_string(char str[], int h)` that receives a string and an hash and returns `1` if the given hash correspond with the string hash or `0` otherwise.
    - The solution should include functions developed in exercises 16 and 5.

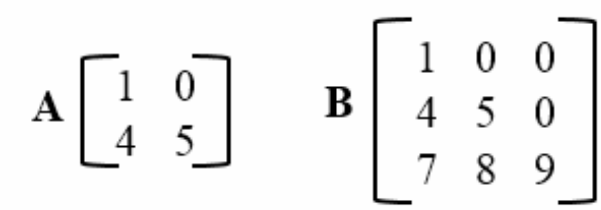
18. Implement the function `int find_pattern(char str[], char patt[])` that receives two strings (`str` and `patt`) and returns the number of times the second string, `patt`, appears into the first one, `str`.
    - Example:

```
int  x = find_pattern("ola Pedro, ola Laura", "la"); // x = 2
```

- The solution should include function developed in exercise 5.

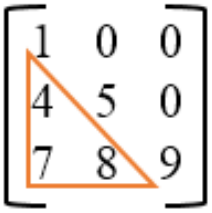
19. Implement the function `int sum_matrix_values(int mat[5][3])` that receives a matrix of positive integers, `mat`, and returns the sum of its elements.

20. A lower triangular matrix is a special square matrix whose all elements above the main diagonal are zero.



Lower triangular matrix

- Implement the function `int check_lower_triangular_matrix(int mat[][5], int lin, int col)`, which receives a matrix of positive integers, `mat`, and its number of lines, `lin`, and returns 1 if it is a lower triangular matrix or 0 otherwise.
- Implement the function `int sum_lower_triangular_matrix(int mat[][5], int lin)`, which receives a matrix of positive integers, `mat`, and its number of lines, `lin`, and returns the sum of its elements [1] or -1 if it is not a lower triangular matrix.



[1] The sum of elements marked in the red triangular area