LAB NR. 9
C PROGRAMMING

December 13, 2023

Problem 1:

Write a program that will allocate memory to an array of the size specified by the user at runtime. Use `malloc` and `free`. Fill the array, print the elements and calculate the average value.

A. In the main function:

a) Ask the user for the size of the array.

b) Using the malloc function allocate the double array of the size specified by the user.

c) Check if the allocation was successful.

- If the address returned by `malloc` is not NULL, use the rand function in a for loop and assign pseudorandom values to the array elements. Then call the function averagevalue. Print the result. Free up memory with the `free` function.

- If the allocation failed and the address returned by `malloc` is NULL, print the message and exit the program.

B. Define the function averagevalue and then call it in main.

The function calculates the average value of the elements of the array passed as an argument and prints the array elements to the screen. The function returns the average value.

C. Use `valgrind` to check memory usage.

Problem 2:

Concatenate two strings using pointers and dynamically allocating the memory for the results. Read two strings from the keyboard using the `fgets` function. Dynamically allocate the memory for the result using `malloc` function. Using pointers, copy the first string to the result , except the null terminator ('\0') from the end from the first string. Copy the second string to the result, including the null terminator. Print your results. Free up memory with the `free` function. Use `valgrind` to check memory usage.

Problem 3:

Print a Mayo pyramid with # characters depending on user input, i.e., the pyramid height h. To this end, update the size of the array for each level of the pyramid (for each for loop passing) using the `realloc` function. First of all, allocate an array for the first row. You need the allocate the memory for $h + 2$ characters for the first row. Fill it with spaces, using the `memset(row, ' ', h+2)` command. For each row i, expand the array size to $h + 4 + i$, replace the last two characters of the line with # sign, followed by the string's null terminator. That's why you need increase the size by 2, with the respect to the original allocation, and then by 1 at each step. You also need to replace the space at the position $h - 1 - i$ with the # character (the left slope of the pyramid). Print and examine your results, free the memory and use the `valgrind` function to check memory usage.