DT555B Programmering i C

- Lab 1 + 2

Adis Veletanlic

22/9 - 2022

Table of Contents

1. Introduction

The objectives of Lab 1 were to create flowcharts and pseudocode for certain problems, in turn requiring the students to develop algorithms for the problems they selected. Since Lab 2 is dependent on Lab 1, I decided to merge the required reports into one, with the approval of my professor, Eric Chen.

The tasks I selected were both Grade 5 tasks; Computer Assisted Instruction and Pyramid.

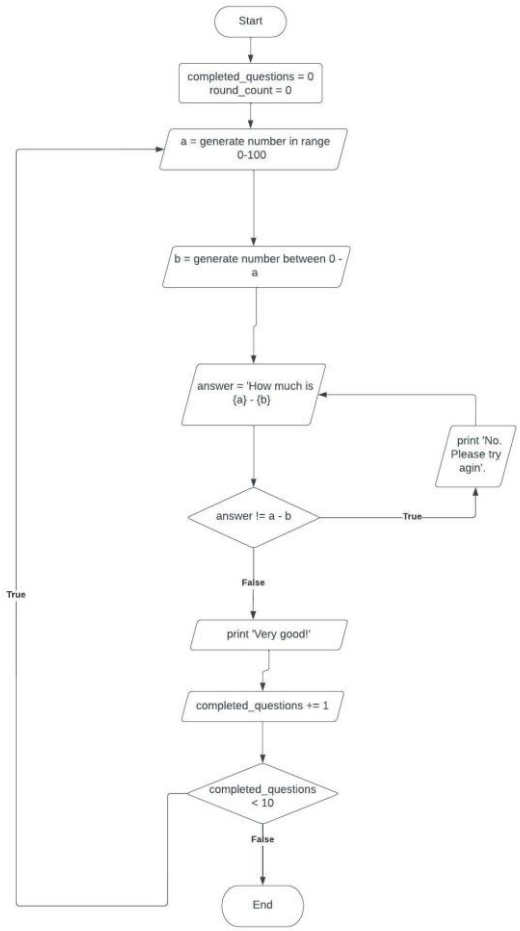
2. Design

The initial groundwork was laid out by writing pseudocode, which I found to be interchangeable with regular text, as it provides an abstract way of expressing a solution to a certain problem. The first step would therefore be to understand the problems at hand, write pseudocode, refine it and eventually design a flowchart.

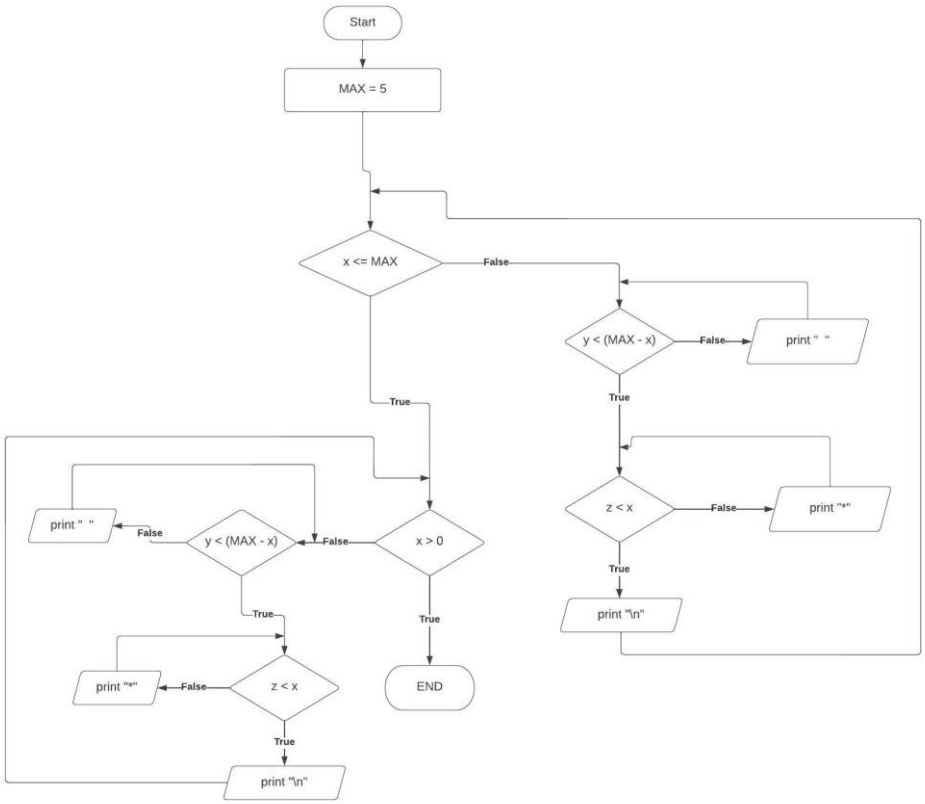
The designs I have implemented use top-down design, since this is how simple problems like this can be solved the quickest.

FLOWCHARTS:

CAI



PYRAMID:



PSEUDOCODE:

CAI:

completed_questions = 0
round_count = 0
a
b
answer

DO
a = generate number between 0 - 100
b = number between 0 - a

DO
OUT "How much is a - b?"
IN answer
WHILE answer != a-b

OUT "Very good!"

```
completed_questions = completed_questions + 1
round_count = round_count + 1


WHILE completed_questions < 10


OUT "You passed"
```

PYRAMID:

```
MAX = 5


FOR x = 1 To MAX
FOR y = 0 To MAX - x
OUT " "
EndFor
FOR z = 0 To x
OUT "*"
EndFor
OUT '\n'
EndFor


FOR x = (MAX - 1) To 0
For y = 0 To (MAX - x)
OUT " "
EndFor
FOR z = 0 To x
OUT "*"
EndFor
OUT '\n'
EndFor
```

3. Implementation and Test

Task "Grade 5-- Computer Assisted Instruction (CAI)"

This task was straightforward to implement into actual code. First, I declare the necessary variables at the top level of the main function block, all the code lives inside of a do-while loop. I took the liberty of printing round numbers at the top of the do-while loop so the user knows how far they are. After this, I generate two random numbers 'a' and 'b'. The variable 'a' can be anything from 0 to 100, the variable 'b', however, is limited to a max number of whatever 'a' is declared as in the step prior.

Then, there is another do-while loop inside of the root-level do-while loop, which contains a print statement and a scan statement. The loop asks the user what the answer of a – b is and reads the users response via scanf(). The do-while loops for as long as the input is incorrect and does not correspond to a – b. When the nested loop finishes, another print statement is executed and the variables 'completed_questions' and 'round_count' are both incremented by one. The top-level do-while loop will keep executing until the completed_questions variable is less than 10. After this, the program simply prints that the user has finished the program.

To test this program, I ran it several times manually and concluded from around 10 executions that the program runs successfully.

Task "Grade 5-- Task 2 Pyramid"

This task was trickier to implement, since there are a lot of inline variables declared inside of the nested for-loops.

The program starts with a declaration of the constant MAX value as 5. After that, we start with our first top-level for-loop for the upper half of the pyramid shape, which contains two for-loops of its own. This upper for-loop executes the containing statements until the inline upper_row variable has become more than the MAX value. Inside of this loop, we have the first nested for-loop that prints a space (" ") until the condition upper_space < (MAX – upper_row) is true. This makes sure that the number of space is limited to the the current iteration of the outer for loop, so the first time it prints (MAX – 1) spaces, the second time it prints MAX – 2 spaces, etc. After this loop has been executed, the loop responsible for the

stars starts. This loop only prints for as long as the condition upper_star < upper_row is true, meaning it only prints a star in the place where the previous loop did not print a space.

When this first top-level for-loop has executed, the second one comes into play, responsible for the lower-half of the pyramid shape. It essentially does the same thing but decrements the value lower_row each iteration instead of incrementing.
So, for each iteration of this second top-level for-loop, there are two nested for loops that print the spaces and stars. The first nested loop prints only spaces for as long as the condition lower_space < (MAX – lower_row) is true. This means that it only prints one space the first iteration of the parent-loop, two spaces the second iteration of the parent-loop, etc.
The second loop inside of the parent loop prints stars as long as the condition lower_star < lower_row is true. This means that it will only print as many stars as there are remaining spaces, which the previous loop did not fill.

To test this program, it is possible to change the constant MAX value at the top of the main function. This will decide how big the resulting pyramid becomes. I personally tested all values from 1 to 20, with successful results.

4. Results and discussion

For these two labs, I am happy with the results and how my code turned out. I had no real issues with the first task, however, in the second task I struggled for a while with the putchar() function. First, it did not display any space characters that I gave to it, so I ended up using a simple printf() instead.
For this task I also found it difficult to name the variables in a readable way, but ended up structuring the code so that it is made up of two parts, where each variable follows some sort of naming convention based on which part of the function it exists in.

From these labs, the above statements are what I can take with me to the following labs. Careful naming conventions, compromise regarding certain functions based on their functionality.

5. References

- -