



Kristianstad  
University  
Sweden

# **DT555B Programmering i C**

## **Lab 3 – CAI**

Zandra Hedlund

231128

---

## Table of Contents

1. Introduction .....	1
Grade 3 Task Option 2-- A Simple CAI Program .....	1
2. Design .....	2
Flowcharts and pseudocode .....	3
3. Implementation and Test.....	8
4. Results and discussion .....	10
5. References.....	11

## 1. Introduction

Algorithm development is a fundamental aspect of programming and software engineering, and it plays a crucial role in solving complex problems. It's an acquired skill that's mastered through a lot of practice and requires a combination of creativity, analytical and critical thinking, as well as an understanding of data structures.

The primary objective of this lab is to get a deeper understanding in the application of the top-down design methodology, by applying it to a more complex problem. This approach involves systematically dissecting complex issues into manageable sub-problems and coming up with a step by step approach. The crux of effective problem-solving lies in the ability to deconstruct formidable challenges into smaller, more digestible tasks. By applying this approach, this lab aims to make us practice algorithmic thinking and problem solving skills.

### Grade 3 Task Option 2-- A Simple CAI Program

The lab task involves creating a Computer-Assisted Instruction (CAI) program tailored for primary school students, focusing on numbers within the range of 0 to 100. The program is expected to offer a main menu where users can select between practicing, taking a test, or exiting the program.

For practice sessions, users can opt for additions only, subtractions only, or a mix of both. Each practice session comprises 10 questions where users provide answers. If an incorrect answer is given, the program persists in asking until a correct response is provided. After completing the practice, users are returned to the main menu.

In the test section, users can select additions only, subtractions only, or a mix of both for a test consisting of 15 questions. When an answer is given, the program continues to the next question. Upon completion, the program displays the test result as a percentage. Similar to the practice mode, users are directed back to the main menu after finishing the test, giving them the option to practice, take another test, or exit the program.

## 2. Design

I started by looking at the problem and breaking it down into smaller tasks, to figure out the best way to solve it, within my skill level. I always use pen and paper at this stage. I then translated it into pseudocode and flowcharts before I implemented the code. I made separate flowcharts for each function, since it would have been harder to get an overview if all the steps of the functions were included in the main flowchart. The function I use to randomize numbers is just one line, so I did not feel the need to make a separate flowchart for it. In the pseudocode, I wrote it the way it would be written in the program, with the helper functions first and the main last.

The program starts by asking the user to enter their name, greets them welcome and then displays a menu with three available choices. If the user chooses alternative 1 or 2, they will be presented a new menu, where they can choose between addition, subtraction or a mix of both. It will then call one of two functions: `do_practice()` or `do_test()`. Alternative 3 will exit the program.

The functions for practice and test are very similar in design. Two numbers are randomly generated and depending on which calculation method the user chose, it will generate math questions. In the practice function, the user gets 10 questions and if the answer is incorrect, they will get the same question again until they get it right. In the test function, there's a number of 15 questions and the program will continue to the next question no matter if the answer is correct or not. At the end, a calculation of the result is being performed and the program outputs the percentage of correct answers.

## Flowcharts and pseudocode

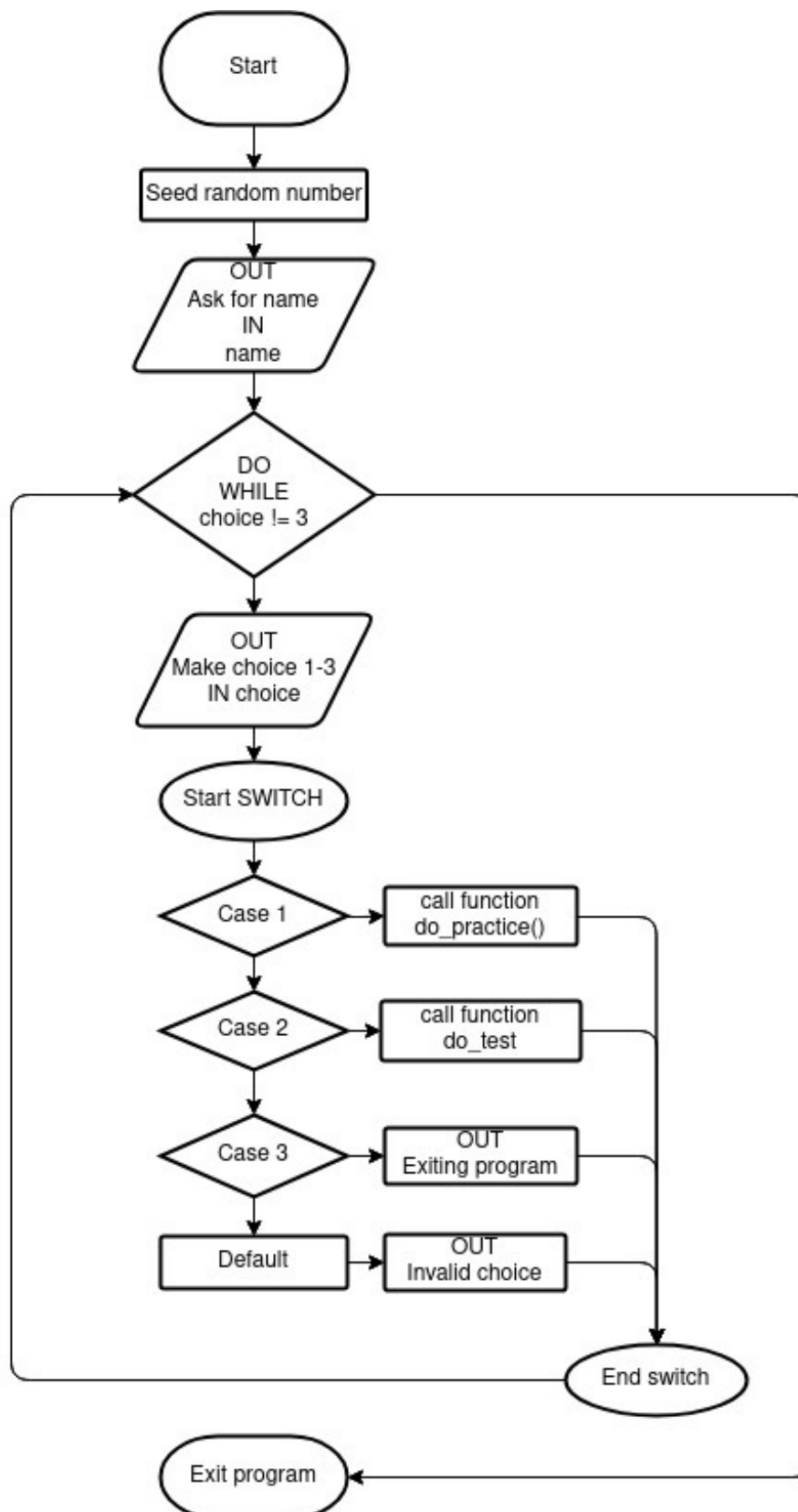


FIGURE 1. main()

**FUNCTION main():**

Seed random number generator srand(time(NULL));

Declare variables: name, choice, operation

OUT "Enter your name: "

IN user name

OUT "Welcome, <name>"

Repeat until user chooses to quit:

DO

OUT "You can choose:

1. Do practices

2. Complete a test

3. Quit the program

Enter your choice: "

IN user choice (1-3)

SWITCH

CASE 1:

OUT "Now, you can choose to do practices on:

1. addition

2. subtraction

3. addition and subtraction

Enter your choice: "

IN user choice (1-3)

IF choice is invalid

OUT "Invalid choice."

BREAK

ENDIF

**do\_practice(operation)**

BREAK

CASE 2:

OUT "Now, you can choose to do a test on:

1. Additions

2. Subtractions

3. additions and subtractions

Enter your choice: "

IN user choice (1-3)

IF choice is invalid

OUT "Invalid choice."

BREAK

ENDIF

**do\_test(operation)**

BREAK

CASE 3:

OUT "Exiting the program. Goodbye, <name>"

BREAK

DEFAULT

OUT "Invalid choice. Please enter a valid option."

WHILE choice != 3

END DO WHILE

END PROGRAM

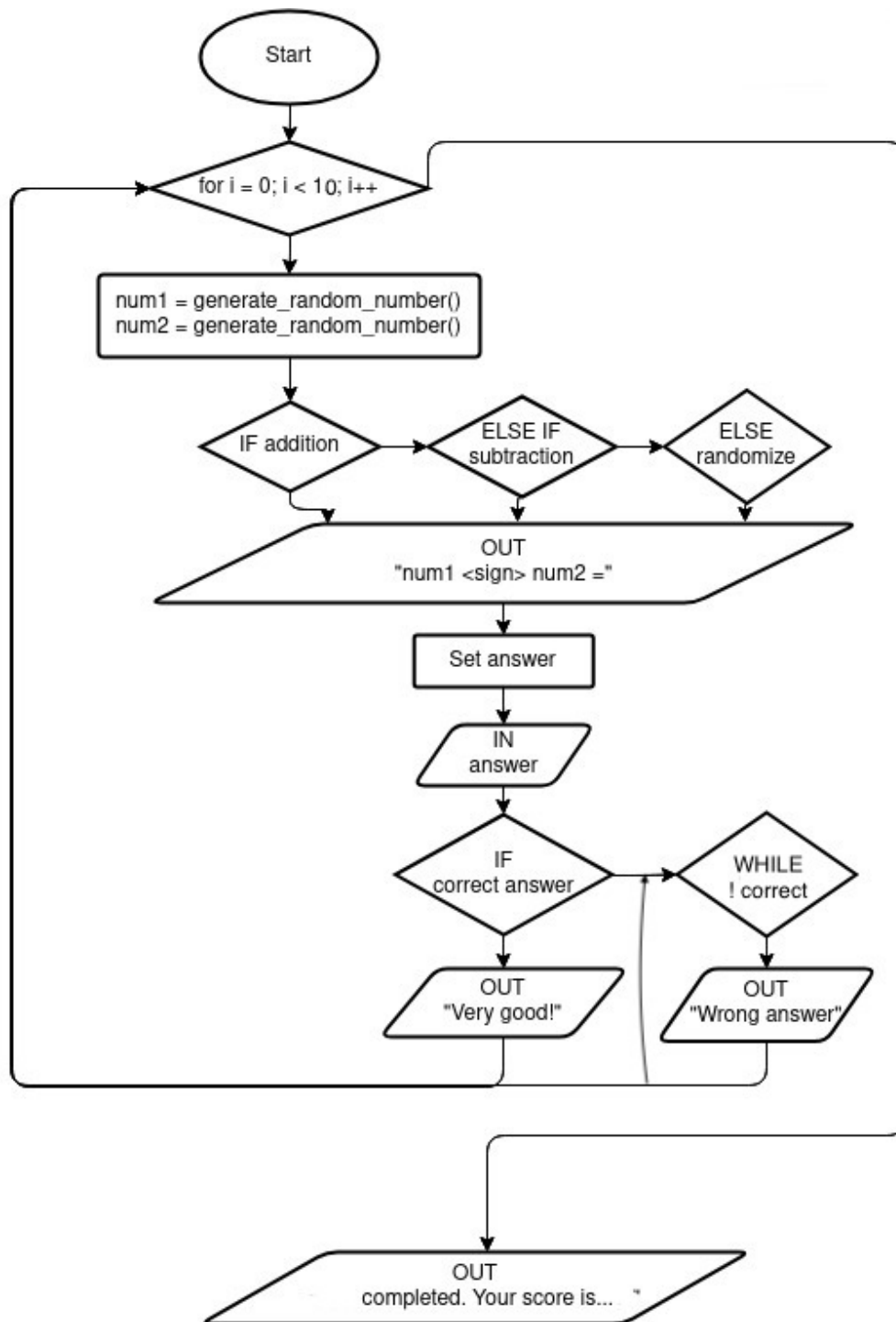


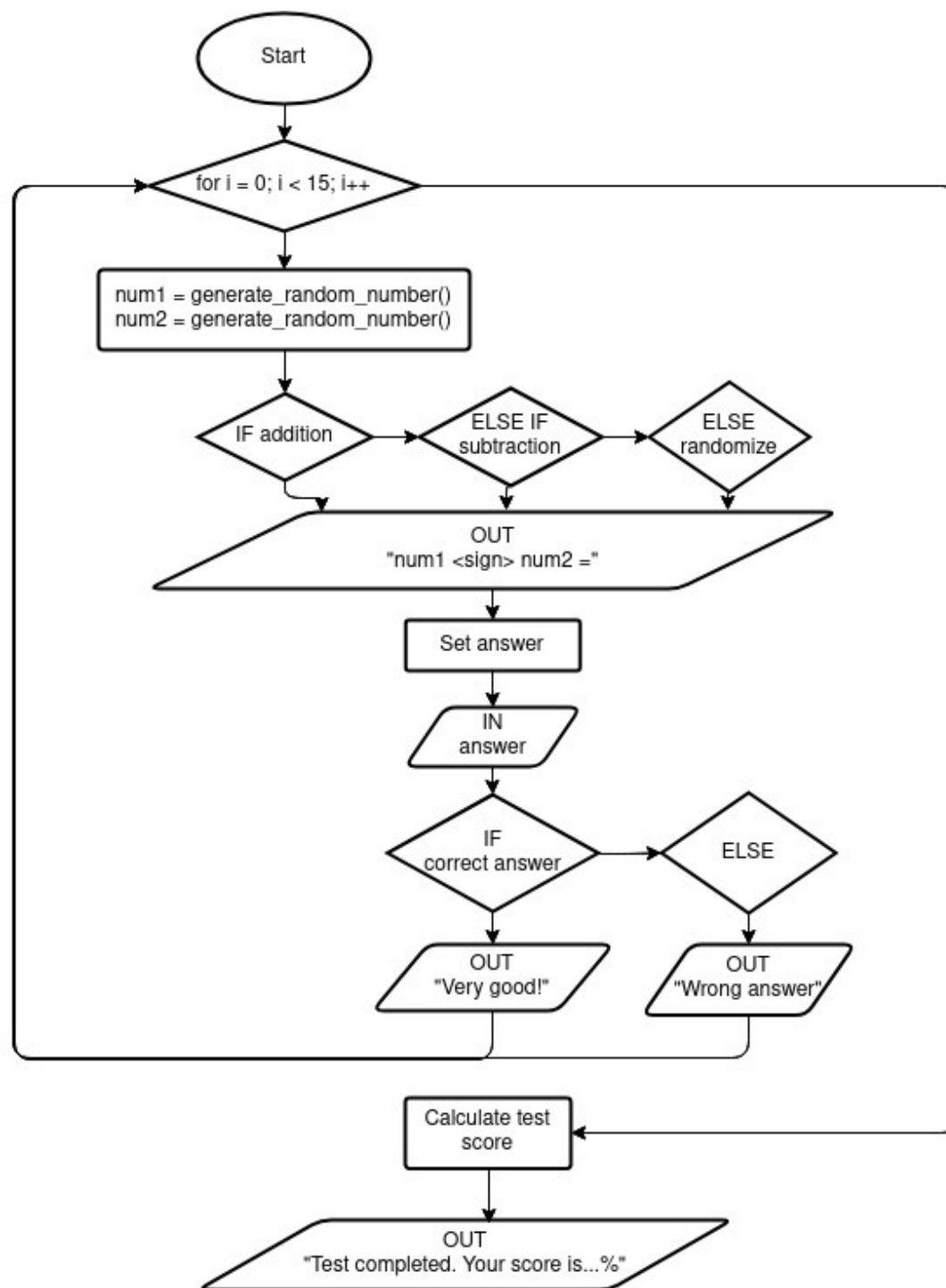
FIGURE 2. do\_practice()

```

FUNCTION void do_practice(int operation):
    correct = 0
    FOR i = 0; i < 10; i++
        num1 = generate_random_number(0, 100)
        num2 = generate_random_number(0, 100)
        IF operation is addition:
            OUT "num1 + num2 ="
            Set answer = num1 + num2
        ELSE IF operation is subtraction:
            OUT: num1 - num2 =
            Set answer = num1 - num2
        ELSE (mixed addition and subtraction):
            Randomly choose addition or subtraction
            OUT question
            Set answer based on chosen operation
        ENDIF
        IN user answer
        WHILE user answer != correct answer:
            OUT "No. Please try again. Enter your answer:"
        END WHILE
        OUT "Very good!"
        correct++;
    END FOR
    OUT "Practice completed. You got 10 out of 10 correct."

```





**FIGURE 3.** do\_test()

```

FUNCTION void do_test(int operation):
    correct = 0
    FOR i = 0; i < 15; i++;
        num1 = generate_random_number(0, 100)
        num2 = generate_random_number(0, 100)
        IF operation is addition:
            OUT "num1 + num2 ="
            Set answer = num1 + num2
        ELSE IF operation is subtraction:
            OUT: num1 - num2 =
            Set answer = num1 - num2
        ELSE (mixed addition and subtraction):
            Randomly choose addition or subtraction
            OUT question
            Set answer based on chosen operation
        END IF
        IN user answer
        IF user answer == correct answer:
            OUT "Very good!"
            correct++;
        ELSE
            OUT "Wrong answer."
        END IF
    END FOR
    Calculate test score (correct * 100 / 15)
    OUT "Test completed. Your score is: percentage%"

```

### 3. Implementation and Test

The program starts with calling `srand(time(NULL))`; to ensure that the random number generator is seeded with different values each time the program is executed. I use `scanf()` for all user input and `printf()` for all output.

Since the program is repetitive and involves giving the user the same menu multiple times, I decided to use switch case in the main function, with 3 cases that correlates to the alternatives the user is given. I chose to place the switch case within a do-while loop since it executes at

least once and for the menu to be shown, the loop will have to execute. The do-while keeps going until the user chooses alternative 3, which exits the program.

I made three separate functions with the following prototypes:

```
int    generate_random_number(int min, int max)
```

```
void   do_practice(int operation)
```

```
void   do_test(int operation)
```

### **generate\_random\_number**

This function is used to generate randomized numbers for the math exercises. It's just one line and could easily have been done directly in the main, but this way the code gets more readable. It takes two parameters, int max and int min. In this program it's called with arguments min 0, max 100. Each time it's called, it returns a random number within that range.

### **do\_practice**

This function will have the user complete a practice session. It takes one argument, int operation. The function starts with declaring variables int i, num1, num2, answer, user\_answer, correct. correct is initialized to 0 and will be used to count correct answers. If the entire practice session is completed, correct answers will always be 10 out of 10.

It then enters a for loop, which will execute 10 times. The code block inside the loop starts with calling generate\_random\_number() and assigning them to variables num1 and num2. Then it checks the value of the operation argument. If operation == 1, it will output "num1 + num2 = " and assign the correct answer to the answer variable. If operation == 2, the same thing happens, but with subtraction instead. If the user chose alternative 3, rand() is called to return a random number and if that number % 2 == 0, it will output a question with addition, else subtraction. Variable user\_answer is assigned the value of the user input and is then checked against the variable answer. If user\_answer != answer, it enters a while loop where the program asks the user to try again. The loop runs until user\_answer == answer. It then outputs "Very good!", increment correct with 1 and restarts the loop. When the loop has run

10 times, the program outputs “Practice completed, you got 10 out of 10 correct answers”. The program exits the function and returns to main().

### **do\_test**

This function works almost the same way as do\_practice. The difference is that the for loop runs 15 times and there’s no nested while loop, since it’s not giving the user more than one try for each question. It will output “Very good!” if the answer is correct, else “Wrong answer.” When the for loop has run 15 times, the program outputs “Test completed. Your score is <number>%”. The calculation is being done directly in the function call: `printf("Test completed. Your score is: %.2f%%\n", (float)(correct * 100) / 15)`. The program exits the function and returns to main().

After exiting the function, the user will again be presented with the menu and the do-while will run until the user chooses alternative 3, the program will then output “Exiting the program. Goodbye, <name>”, exit the loop and end the program by returning 0.

## **4. Results and discussion**

The program compiles and seems to behave as expected. The hardest part was to figure out how to make the program mix addition and subtraction, without hardcoding it to addition/subtraction every second question. That solution would have met the requirements, but I wanted to make a randomized solution and came up with using `rand()` to get a random number and use it for the selection, by checking if it’s even (`number % 2 == 0`) or odd and it turned out well. It was my first time using switch case, so I thought it was fun to see that it worked the way I was expecting.

Making the flowcharts was quite tricky and time consuming. I can’t get AlgoBuild to work, so I used VisualParadigm and it’s very buggy.

Overall I think it was a fun project. Algorithms are of course essential in programming and learning to plan ahead before diving into the actual coding, really makes things easier later on.

## 5. References

The system manual for functions rand, srand, time, printf, scanf, using command line arguments:

“man rand”

“man srand”

“man time”

“man printf”

“man scanf”

I didn't use any other external references, besides the lectures provided on Canvas. I used VS Code to write and run the code and VisualParadigm for the flowcharts.