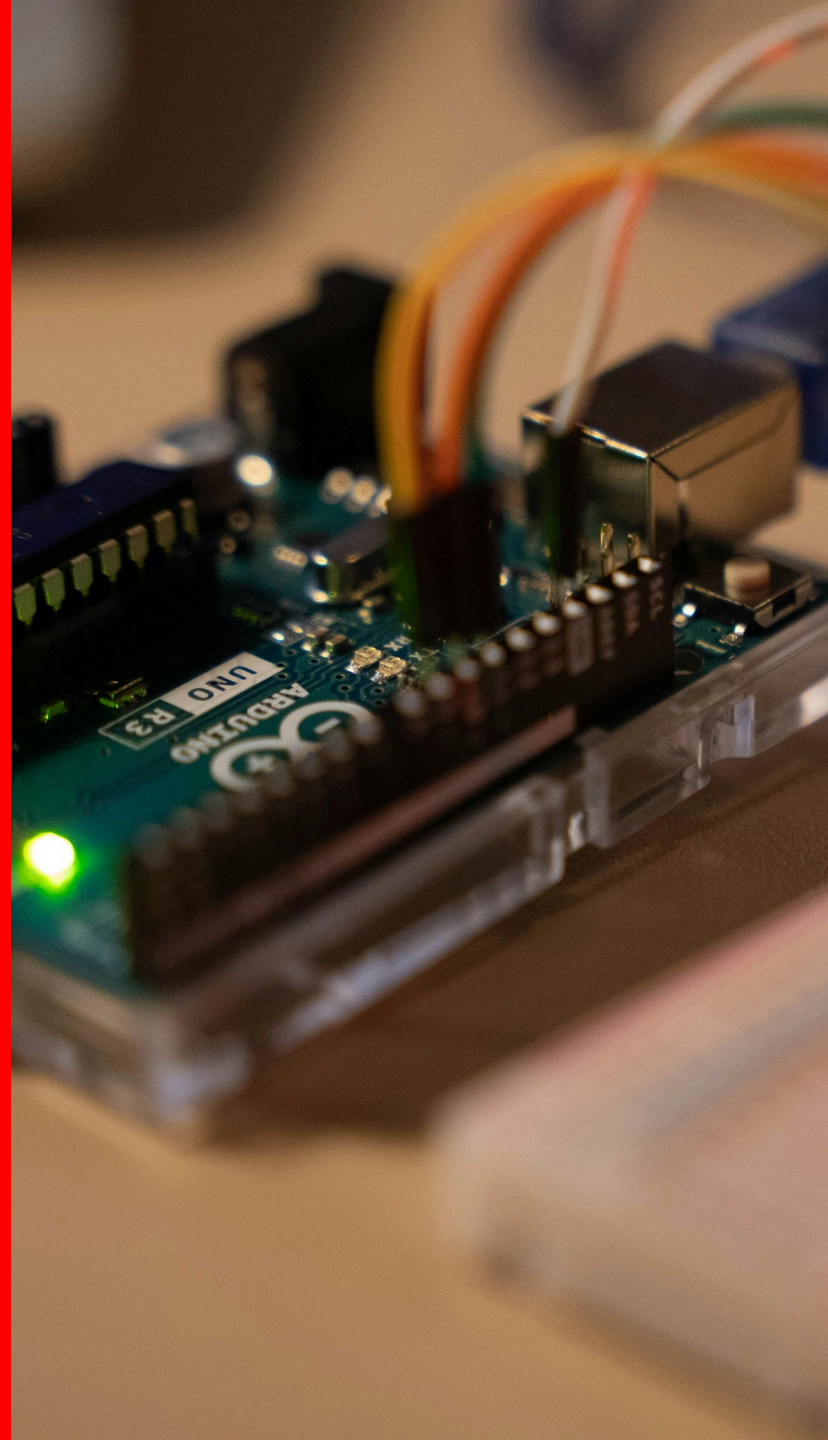
EEC 2202: STRUCTURED PROGRAMMING

Assignment 3

Variables, Pointers & Functions



Please follow these guidelines for all programming assignments!!

1. Work Locally:

a) Write and test your C programs on your local computer.
b) Make sure your code compiles and runs correctly before submission.

2. Use GitHub for Submission:

c) Create one main GitHub repository for all your structured programming assignments (e.g., FullName_StructuredProgramming) and inside it, make a separate folder for each assignment (e.g., Assignment1)
d) Push only your C source files including header files (.c, .h), and any supporting files like a README.md or Makefile.
e) Do not upload compiled files (e.g., .exe, .o, or .out).

3. Submit the GitHub Link:

f) Once your code is on GitHub, copy the repository link (URL).
g) Submit the link through Google Classroom.

Programming Tasks

Task 1:

Before starting the programming tasks, students must read and understand the following pointer-related concepts.
Write short notes explaining each of the concepts and **include them in the README file** of your repository for this assignment.

1. Explain the difference between a *normal variable and a pointer*. Your answer should clearly mention: what each one stores, how memory is accessed, how values are read and modified.
2. Using suitable examples, explain how *variable declaration and definition* differs from *pointer declaration and definition*. Clearly highlight the role of the operators * and &
3. Explain the meaning of *deferencing* a pointer. Using a simple show how a pointer accesses the value stored at a memory address, and demonstrate how a value can be modified using deferencing.

Programming Tasks

Task 1 continued:

Before starting the programming tasks, students must read and understand the following pointer-related concepts.
Write short notes explaining each of the concepts and **include them in the README file** of your repository for this assignment.

4. Describe scenarios or use cases where *pointers are preferred over normal variables*. Support your answer with at least two practical examples.
5. Explain the *limitations and risks* associated with using *pointers* compared to variables.
6. Using suitable examples compare *call by value and call by reference*. Explain how data is passed to functions in each case.
7. Discuss practical scenarios where:
    a. Call by value is preferred
    b. Call by reference is preferred

Programming Tasks

Task 2: Display Address of a Variable

Objective

- ✓ To understand the difference between a *variable* and a *pointer* and how each is defined and declared.
- ✓ To understand how pointers store and access memory addresses.

Tasks

- Declare an integer variable named `num` and assign it a value (for example, 10).
- Declare an integer pointer named `ptr`.
- Store the address of num in the pointer `ptr`.
- Print the following:

  Value of `num` Value stored in `ptr` (address of num)

  Address of `num` Value accessed using `*ptr` (dereferenced value)

Programming Tasks

Task 3: Access Variable Value Using Pointer

Objective

- ✓ To learn *pointer dereferencing* and *value modification* using *pointers*.

Instructions

- Declare an integer variable named `count` and assign it an initial value (for example, 10).
- Declare an integer pointer named `pCount`.
- Assign the address of count to `pCount`.
- Modify the value of count using the pointer `pCount`.
- Print the updated value of `count`.

Programming Tasks

Task 4: Add Two Numbers Using Pointers

Objective

- ✓ To perform arithmetic *operations using pointer dereferencing*.

Instructions

- Declare two integer variables named `num1` and `num2`.
- Declare two integer pointers named `ptr1` and `ptr2`.
- Assign the address of `num1` to `ptr1`.
- Assign the address of `num2` to `ptr2`.
- Add the values of `num1` and `num2` using *pointer dereferencing*.
- Store the result in an integer variable named `sum`.
- Print the value of `sum`.

Programming Tasks

Task 5: Swap Two Numbers Using Pointers

Objective

✓ To understand pass by reference using pointers and functions.

Instructions

- Declare two integer variables named `a` and `b` in `main()`.
- Write a function named `swapNumbers()` that:
  - Accepts two integer pointers as parameters
  - Swaps the values of a and b using dereferencing
- Print the values of `a` and `b` before calling the function.
- Call `swapNumbers()` by passing the addresses of a and b.
- Print the values of `a` and `b` after swapping.

Programming Tasks

Task 6: Pass by Value vs Pass by Reference

Objective

- ✓ Understand the difference between *pass by value* and *pass by reference* when calling functions.

Tasks

- Create two functions:
  - One that increments a number using *pass by value*.
  - One that increments a number using *pass by reference*.
- Call both functions from main() with the same variable.
- Print the value after each function call.

Expected Result

Pass by value → value remains unchanged

Pass by reference → value gets incremented