# **Module Code:** CSE401

## Session 10a: Applying Programming Concepts

**Session Speaker:**

**Prema Monish**

premamonish.tlll@msruas.ac.in

## Objectives

- At the end of this lecture, student will be able to
  - follow the best practices for naming of variables, functions, files and projects
  - use the best practices for documentation
  - use the best practices followed by C programmers in creation of a C project

## Contents

## A Question

Can you explain what this structure is used for?

```
struct myStr{
	char *n;
	char *r;
	int a;
}
```

## Naming Conventions: Variables

- Give full name
- Start with small letter
- Use upper case for every new word
- If it is a pointer, mention it
- Examples
  - age
  - temporaryStudent
  - myStudentPointer

## Naming Conventions: Function

- Give full name
- Start with small letter
- Use upper case for every new word
- name should explain purpose of function
- Examples
  - intAddition
  - linearSearch
  - substringFromString

## Naming Conventions: Symbolic Constants

- Give full name
- All letters in capital case
- Use underscore (_) for every new word
- name should explain purpose of function
- Examples
  - PI
  - DEGREE_TO_RADIANS_RATIO

# Naming Conventions: File Names

- Have only related functions in one file
- use same convention as variables
- Examples
  - intUtilities.c
  - floatUtilities.h

## Types of Errors

- Syntax errors
  - Violation of grammar/rules of programming language
  - E.g., int a
- Logical errors
  - Logical thinking problem
  - Difficult to debug as the system does not display them
- Run-time errors
  - Attempt to run an ambiguous instructions
  - E.g., divide by zero

## Documentation

- In each file at the start
  - Header comment
    - Define purpose of code in the file
    - State author name and contact details
    - Give version number
    - Give copyright notice if applicable

- For each function
  - Document its purpose, details of each parameter and the return value expected

## Comments

- //
- /* and */

- Comments are ignored by the compiler and do not cause any machine language object code to be generated

## Indentation

- Indent the entire body of each function one level of indentation within the braces that define the body of the function
- Emphasizes the functional structure of programs and helps make programs easier to read

```
for(i=0;i<3;i++){
        for(j=0;j<3;j++){
                printf("Enter the elements"):
                scanf("%d",&array[i][j]);
        }
}
```

## Documentation - Example

```
/*Program to compute the addition of two integer numbers */
 #include <stdio.h>
/* function main begins program execution */
int main( void ) {
        int integer1; /* first number to be input by user */
        int integer2; /* second number to be input by user */
        int sum; /* variable in which sum will be stored */
        printf( "Enter first integer\n" ); /* prompt */
        scanf( "%d", &integer1 ); /* read an integer */
        printf( "Enter second integer\n" ); /* prompt */
        scanf( "%d", &integer2 ); /* read an integer */
        sum = integer1 + integer2; /* assign total to sum */
        printf( "Sum is %d\n", sum ); /* print sum */
        return 0; /* indicate that program ended successfully */
} /* end function main */
```

## Programming Practices

- Keep your programs simple and straightforward
  - KIS (Keep It Simple)

- Although it is allowed, there should be no more than one statement per line in a program

- Type the beginning and ending braces of compound statements before typing the individual statements within the braces
  - helps avoid omitting one or both of the braces, preventing syntax errors and logic errors

# Programming Practices contd.

- Unary operators should be placed directly next to their operands with no intervening spaces

- Put a blank line before and after each control statement to make it stand out in the program

- Try to avoid more than three levels of nesting

- For a loop used to print the values 1 to 10, use the condition as counter<=10

## Programming Practices contd.

- In expressions using operator &&, make the condition that is most likely to be false the leftmost condition
  - In expressions using operator ||, make the condition that is most likely to be true the leftmost condition

- Don’t confuse with = and == operators
  - Logic error

- Function names should effectively express the task
  - Choosing meaningful function names and meaningful parameter names makes programs more readable and helps avoid excessive use of comments

## Programming Practices contd.

- Although it’s not incorrect to do so, do not use the same names for a function’s arguments and the corresponding parameters in the function definition
  - helps avoid ambiguity

- Providing more initializers in an array initializer list than there are elements in the array is a syntax error

- Ending a #define or #include preprocessor directive with a semicolon
  - preprocessor directives are not C statements

## Programming Practices contd.

- Include the letters *ptr* in pointer variable names to make it clear that these variables are pointers

- Choosing a meaningful structure tag name helps make a program self-documenting

- Assigning a structure of one type to a structure of a different type is a compilation error

## Programming Practices contd.

- When memory that was dynamically allocated is no longer needed, use free to return the memory to the system immediately

## Summary

- Code must be written keeping in mind maintainability
- Maintainable code requires that anyone should be able to read the code
- Naming conventions for variables, functions, files, constants and directories are followed to improve readability of code

## Further Reading

Kernighan, B. W. and Richie, D. (1992) *The C Programming Language.* 2nd ed., New Delhi:PHI.