جامـــعة المـــســتـقـبـل
AL MUSTAQBAL UNIVERSITY

قســـــــــم الذكاء الاصطناعي
DEPARTMENT OF ARTIFICIAL INTELLIGENCE

SUBJECT:

STRUCTURED PROGRAMMING

CLASS:

1 ST STAGE

LECTURER:

DR. ABDULKADHEM A. ABDULKADHEM

LECTURE: (1)

INTRODUCTION TO STRUCTURED PROGRAMMING

1. Introduction to Programming

Programming is the process of writing instructions that a computer can execute to perform specific tasks. These instructions are written using programming languages, which serve as a bridge between human logic and machine execution.

2. What is Structured Programming?

Structured Programming is a paradigm aimed at improving code readability, maintainability, and efficiency by organizing code into well-defined structures. It emphasizes the use of:

- **Sequential execution** (step-by-step execution of statements)
- **Selection control structures** (if-else, switch-case)
- **Iteration control structures** (loops: for, while, do-while)
- **Modular programming** (functions and procedures)

3. Why Structured Programming?

Structured Programming provides multiple advantages over unstructured approaches, including:

- **Better code organization**
- **Easier debugging and testing**
- **Enhanced readability and reusability**
- **Reduced complexity**
- **Encourages logical thinking**

4. Comparison: Structured vs. Unstructured Programming

Feature	Structured Programming	Unstructured Programming
Code Organization	Uses functions and modules	Uses global jumps (goto statements)
Readability	High	Low
Debugging	Easier	Difficult
Scalability	Highly scalable	Not scalable
Efficiency	Optimized for logical execution	May lead to inefficient execution

5. Key Elements of Structured Programming

1. **Sequence** – Executing instructions in order.
2. **Selection (Decision Making)** – Using conditions like `if-else`, `switch-case`.
3. **Iteration (Loops)** – Using loops like `for`, `while`, `do-while`.
4. **Modularity** – Breaking down a program into functions for reusability and clarity.

6. First Example: A Simple C++ Program

Let's look at a basic C++ program that follows structured programming principles:

```
#include <iostream>
using namespace std;

// Function to print a welcome message
void welcomeMessage() {
    cout << "Welcome to Structured Programming!" << endl;
}

int main() {
    welcomeMessage();         // Calling a function (modular approach)
    return 0;
}
```

7. Evolution of Programming Paradigms

Structured programming was introduced as an improvement over unstructured programming (which relied heavily on `goto` statements). It later evolved into Object-Oriented Programming (OOP) and Functional Programming, but it remains fundamental in software development today.

8. Importance of Structured Programming in AI

- **Clear algorithm implementation:** Structured programming helps in organizing AI algorithms such as search, optimization, and machine learning models in a clear and logical manner.
- **Improved readability and maintenance:** AI programs often become complex; structured code makes it easier for developers and researchers to understand, modify, and extend the system.

- **Efficient debugging and testing:** Dividing AI systems into functions and modules allows easier identification of logical errors and performance issues.
- **Better handling of complex computations:** AI applications involve loops, conditions, and data processing steps; structured programming provides systematic control over these processes.
- **Scalability of AI systems:** Well-structured code supports the development of large AI applications such as intelligent agents, data analysis systems, and machine learning pipelines.

9. Summary

- Structured Programming improves readability, maintainability, and efficiency.
- It uses sequences, decisions, loops, and modularity.
- It is widely used in modern software development and AI applications.

10. Next Lectures Preview

In the upcoming lectures, we will explore fundamental concepts in C++ programming, including:

- **Introduction to Functions**: Understanding function declarations, definitions, and different types of functions (user-defined, built-in, and recursive functions).
- **Working with Arrays**: Basics of arrays, multi-dimensional arrays, and array manipulation techniques.
- **Introduction to Structures**: Defining and using structures, structure arrays, and applications in real-world programming.
- **String Handling in C++:** String operations, built-in string functions, and practical examples of string manipulation.

These topics will build a strong foundation for writing efficient and modular C++ programs.