

MODULE V

Unit Category	Unit Code	Unit Name	Unit Duration (Hours)
CORE	0613 551 02 A	OBJECT ORIENTED PROGRAMMING	170
CORE	0613 551 03 A	DESKTOP APPLICATION DEVELOPMENT	240
		Total	410

OBJECT ORIENTED PROGRAMMING

ISCED UNIT CODE: 0613 551 02 A

TVET CDACC UNIT CODE: ICT/CU/SD/CR/01/6/MA

Relationship to Occupational Standards

This unit addresses the Unit of Competency: Apply Object Oriented Programing

Duration of Unit: 170 Hours

Unit Description

This unit covers the competencies required to apply computer programming principles. It involves applying computer programming skills, demonstrating structured programming skills and demonstrating object oriented programming skills.

Summary of Learning Outcomes

Learning Outcomes	Duration (Hours)
1. To apply computer programming skills	30
2. To demonstrate structured programming skills	40
3. To demonstrate object-oriented programming skills	100
TOTAL	170

Learning Outcomes, Content and Suggested Assessment Methods

Learning Outcome	Content	Suggested Assessment Methods
1. Apply Computer Programming Skills	1.1 Programming Language Types 1.1.1 Overview of programming language categories 1.1.1.1. procedural	● Practical Activities ● Project work ● Group work ● Observation

	1.1.1.2. object-oriented 1.1.1.3. functional 1.1.2 Criteria for selecting languages based on user requirements 1.2 Programming Paradigms 1.2.1 Explanation of common programming paradigms (e.g., imperative, declarative) 1.2.2 Choosing the appropriate paradigm based on project needs 1.3 Program Development Life Cycle 1.3.1 Stages of the program development life cycle (e.g., planning, design, implementation) 1.3.2 Best practices for adapting the life cycle to work requirements 1.4 Program Design Tools 1.4.1 Overview of design tools (e.g. algorithms, flowcharts, wireframes, pseudocodes, decision table/trees) 1.4.2 Selecting design tools based on user requirements and project complexity 1.5 Program Writing Tools 1.5.1 Common program writing tools and IDEs (e.g. Visual	● Portfolio of Evidence ● Written tests
--	---	--

	Studio, codeblocks, DEV C++, Eclipse) 1.5.2 Evaluating tools based on system requirements and developer preferences	
2. Demonstrate structured programming skills	2.1 Declaring Identifiers 2.1.1 Guidelines for naming conventions and best practices 2.1.2 Ensuring identifiers align with program design specifications 2.1.3 Creating a naming convention guide for different types of identifiers. 2.1.4 Evaluating identifier 2.2 Initializing Variables and Constants 2.2.1 Importance of proper initialization in programming 2.2.2 Techniques for initialization based on design specifications 2.2.3 Writing code snippets demonstrating correct and incorrect initialization. 2.2.4 Conducting a workshop on variable and constant initialization techniques. 2.3 Applying Data Control	● Practical Activities ● Project work ● Group work ● Observation ● Portfolio of Evidence ● Written tests

	Structures 2.3.1 Types of control structures (e.g., loops, conditionals) 2.3.2 Best practices for implementing control structures as per design requirements 2.3.3 Solving coding challenges that require the use of different control structures. 2.3.4 Creating flowcharts to visually represent control structures in a program. 2.4 Applying Data Structures 2.4.1 Overview of common data structures (e.g., arrays, linked lists) 2.4.2 Selecting appropriate data structures based on design specifications 2.4.3 Implementing various data structures in a programming language of choice. 2.4.4 Comparing performance metrics of different data structures in a small project. 2.5 Creating Subroutines 2.5.1 Benefits of using subroutines (e.g., modularity, reusability)	
--	---	--

	2.5.2 Designing subroutines to meet user needs 2.5.3 Functions and subprograms 2.5.4 Design and implement a subroutine library for common tasks. 2.5.5 Program to create subroutines based on given specifications. 2.6 Applying User-Defined Data Types 2.6.1 Overview of user-defined data types (e.g., structs, classes) 2.6.2 Criteria for selecting data types based on system requirements 2.6.3 Designing a class or struct for a real-world object 2.6.4 Collaborating on a group project that requires the use of user-defined data types. 2.7 Performing Debugging 2.7.1 Common debugging techniques and tools 2.7.2 Participating in a debugging workshop using a sample project. 2.7.3 Creating a debugging checklist based on work procedures. 2.8 Compiling Program	
--	---	--

	2.8.1 Steps involved in the compilation process 2.8.2 Ensuring compliance with system requirements during compilation 2.8.3 Compiling a series of programs with intentional errors to learn about error messages. 2.8.4 Researching and presenting on different compilers and their features.	
3. Demonstrate object-oriented programming skills	1.1 Implementing Objects and Classes 1.1.1 Overview of objects and classes in OOP 1.1.2 Implementation of classes 1.1.3 Creating a simple class and instantiating objects to demonstrate understanding. 1.1.4 Participating in a group project where each member implements a different class based on a shared design. 1.2 Declaring Object Methods 3.2.1 Defining class methods that fulfill application requirements 3.2.2 Best practices for method naming and functionality 3.2.3 Developing a class with various methods and demonstrating their usage	● Practical Activities ● Project work ● Group work ● Observation ● Portfolio of Evidence ● Written tests

	in a small application. 3.2.4 Conducting a code review session focusing on method declarations and their alignment with application requirements. 3.3 Applying Namespaces 3.3.1 Understanding the role of namespaces in OOP 3.3.2 Implementing namespaces as per work procedures 3.3.3 Create a project that uses multiple namespaces to organize code effectively. 3.3.4 Explore and present on the benefits of using namespaces in a collaborative coding environment. 3.4 Applying Data Abstraction Concepts 3.4.1 Definition and importance of data abstraction 3.4.2 Implementing abstraction in line with work procedures 3.4.3 Designing an abstract class and demonstrate its use in a program. 3.4.4 Trainees work in pairs to identify and implement abstraction in existing codebases. 3.5 Applying Object Encapsulation 3.5.1 Understanding encapsulation and its	
--	---	--

	significance 3.5.2 Implementing encapsulation in programs 3.5.3 Type of access modifiers 3.5.3.1 Private 3.5.3.2 Public 3.5.3.3 Protected 3.5.4 A class to demonstrate encapsulation by using private and public access modifiers. 3.5.5 Class presentation on advantages of encapsulation in a program. 3.6 Implementing Class Templates 3.6.1 Overview of class templates and their applications 3.6.2 Creating class templates 3.6.3 Write a generic class template and demonstrate its usage with different data types. 3.6.4 Collaborating on a project that requires the use of class templates for various functionalities. 3.7 Implementing Class Inheritance 3.7.1 Inheritance concepts and types of inheritance (single, multiple) 3.7.2 Applying inheritance in programs 3.7.3 Creating a class hierarchy to	
--	--	--

	demonstrate inheritance concepts. 3.7.4 Participating in a coding challenge that requires implementing inheritance in a given scenario. 3.8 Implementing Polymorphism 3.8.1 Definition and types of polymorphism (compile-time vs. runtime) 3.8.2 Implementing polymorphism in a program 3.8.3 Develop a program to demonstrate both types of polymorphism. 3.8.4 Engage in a discussion or workshop on the practical applications and benefits of polymorphism in software development.	
--	---	--

Suggested Delivery Methods

- Instructor led facilitation using active learning strategies
- Demonstration by trainer
- Practical work by trainees
- Viewing of related videos
- Field Visits
- Group Work
- Role plays
- Group projects

Recommended Resources for 25 Trainees

S/No.	Category/Item	Description/	Quantity	Recommend
-------	---------------	--------------	----------	-----------

		Specifications		ed Ratio (Item: Trainee)
A	Learning Materials			
14.	Internet connection	✓ For each computer	1	1:1
15.	Flip charts	A1	1	1:25
16.	Textbooks	For reference	3	3:25
B	Learning Facilities & infrastructure			
17.	Computer Laboratory	To accommodate 25 Learners	1	1:25
18.	Theory Room	furnished with 25 Arm-chairs and a suitable trainer's table	1	1:25
C	Consumable materials			
19.	Printing papers	A4	2 Reams	1:12
20.	Toner / Ink bottles	For printers	2 pcs	1:12
21.	White board markers	Assorted colors	20	4:5
D	Tools and Equipment			
22.	Computers	✓ Genuine Windows/Linux ✓ Genuine Microsoft office Software ✓ Google Workspace Account ✓ Antivirus Software ✓ Suitable IDE	25	1:1
23.	External storage media	HDD / SSD / Flash	1	1:25
24.	Printer	Working printer	2	1:12
25.	1 Smart-board / Smart TV / Projector (with screen)	Where available	1	1:25
26.	Whiteboard/Chalkboard	4 X 8 Feet	1	1:25