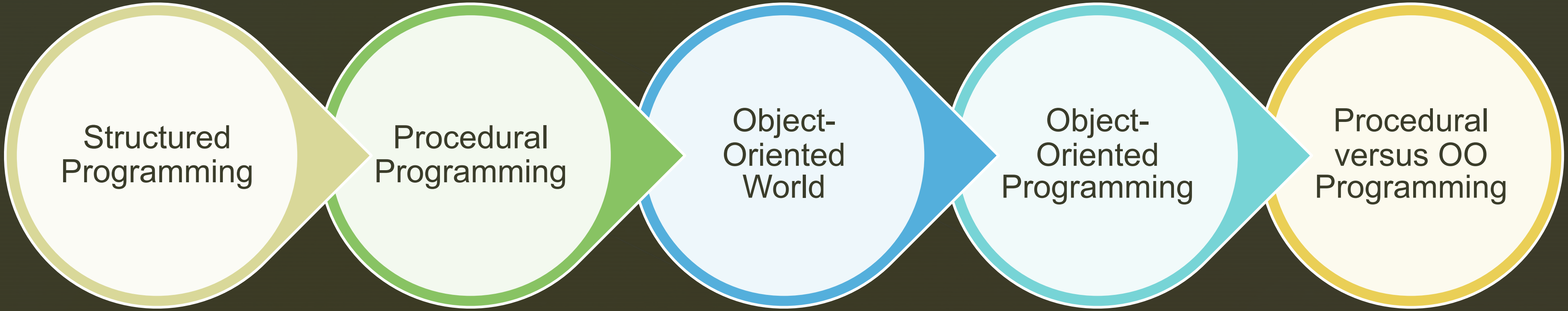
DR. (MRS) T-S.M.A. ADJAIDOO

04 Object-Oriented Thinking

Today's Lesson



Lesson Outcomes

1
- To understand two programming paradigms

2
- To draw out the differences between these different programming paradigms

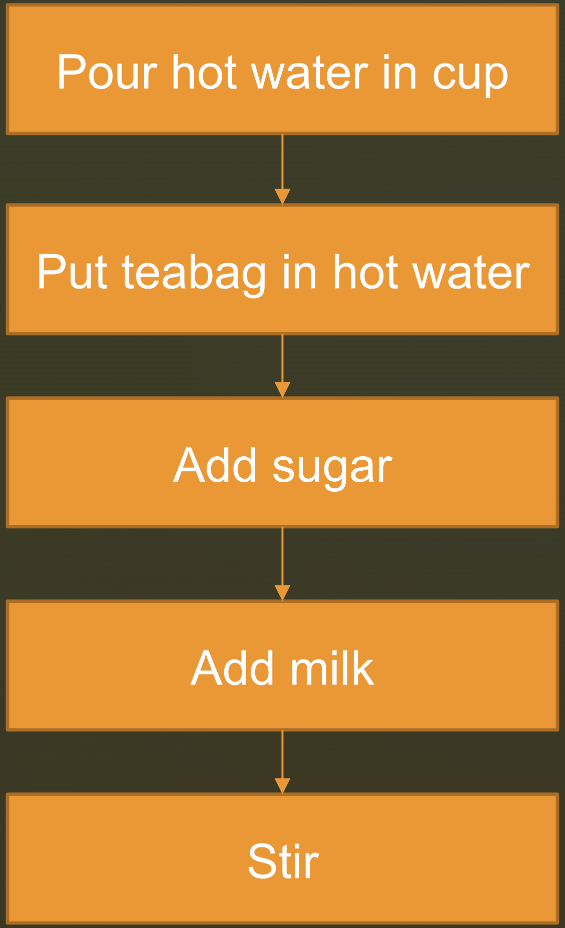
3
- To have a fundamental idea of how to think in terms of objects

Structured Programming

- It is a programming approach which involves writing the program as a single structure in which instructions are executed in series.
- It does not support jumps by the use of GOTO statements which introduce complexity in understanding the code
- Based on these three elements:
  - Sequence
  - Repetition/Iteration
  - Selection

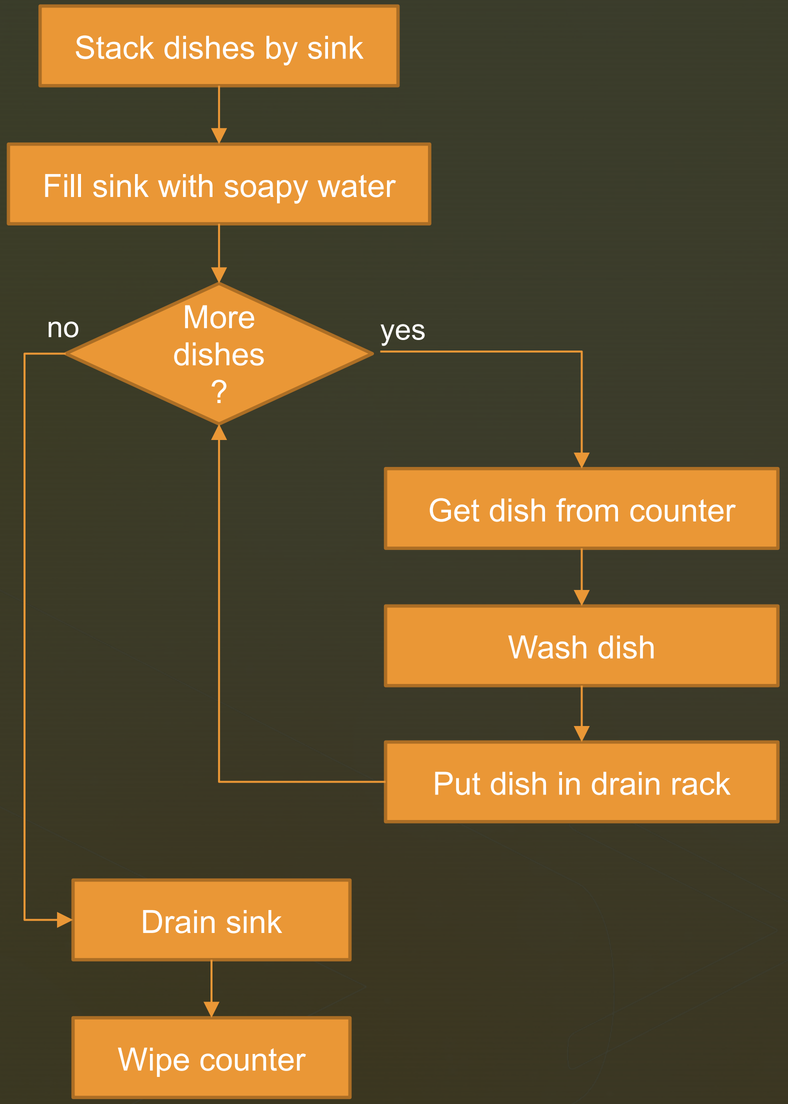
Structured Programming in the real world

- Sequence
  - Statements are executed in a sequential order
  - Example: Making Tea



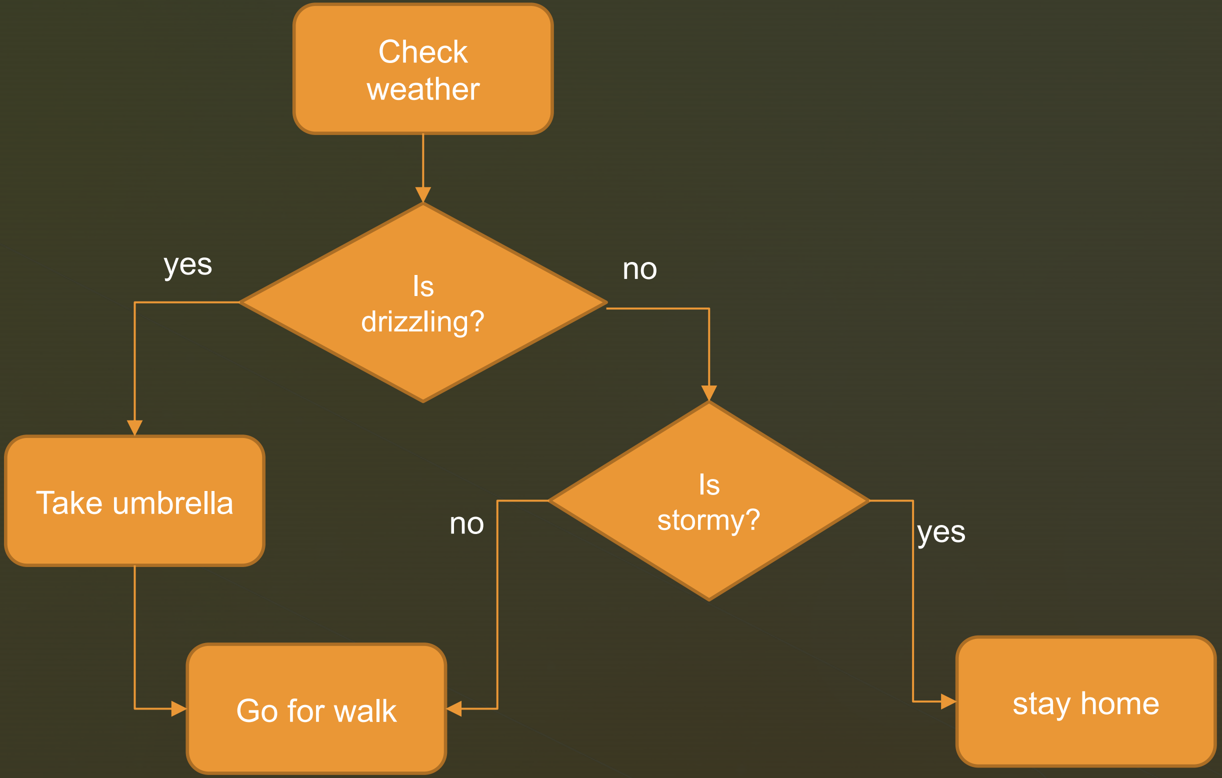
Structured Programming in the real world

- Repetition / Iteration
  - Repeat a set of instructions while some condition remains true
  - Example: Washing Dishes



Structured Programming in the real world

- Selection
- Choose to perform some instructions based on certain conditions
- Example: Going for a walk

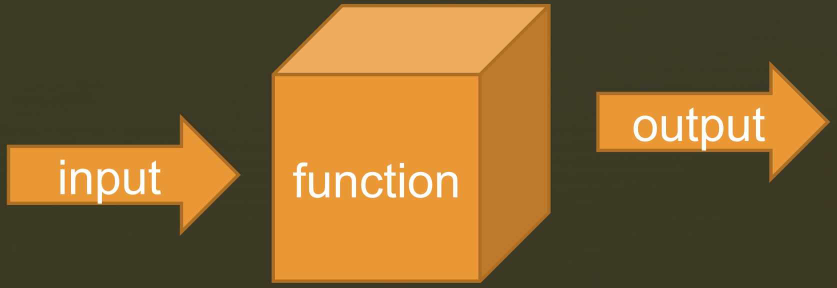


Procedural Programming

- It is a derivative of structural programming
- Based on the concept of dividing a program into sub-structures called procedures
- Procedures are also referred to as subprograms, subroutines, functions or methods
- C programming, Pascal, FORTRAN are examples of procedural languages

Procedural Programming

- Data is placed into separate structures and are manipulated by functions/procedure
- Functions essentially are like black boxes which recieve input and produce outputs



Procedural Programming

- When programs become larger, a single list of instructions becomes unwieldy.
- Few programmers can comprehend a program of more than a few hundred statements unless it is broken down into smaller units.
- For this reason the function was adopted as a way to make programs more comprehensible to their human creators.
- A procedural program is divided into functions, and each function has a clearly defined purpose and a clearly defined interface to the other functions in the program.

Procedural Programming

- The idea of breaking a program into functions can be further extended by grouping a number of functions together into a larger entity called a *module (which is often a file)*
- Dividing a program into functions and modules is one of the cornerstones of *structured programming.*

Procedural Programming

- Some features of procedural programming are:
  - Sequential execution approach
  - Header file declaration
  - Predefined and user-defined functions
  - Function Parameters/Arguments
  - Passing values to functions
  - Global and local variables

Advantages of Procedural Programming

- Relatively easier to learn for beginners in coding
- The availability of standard library functions reduces development time and cost
- The use of pointers in procedural C programming allows for low-level manipulation of memory

Disadvantages of Procedural Programming

- Difficulty in code maintenance makes it unsuitable for large and complex projects
- Focusses more on functions and not the data
- It is difficult to represent real-world objects realistically
- Access to data is uncontrolled and unpredictable because most it is global and multiple functions may have access to global data
- Note: what is spaghetti code?

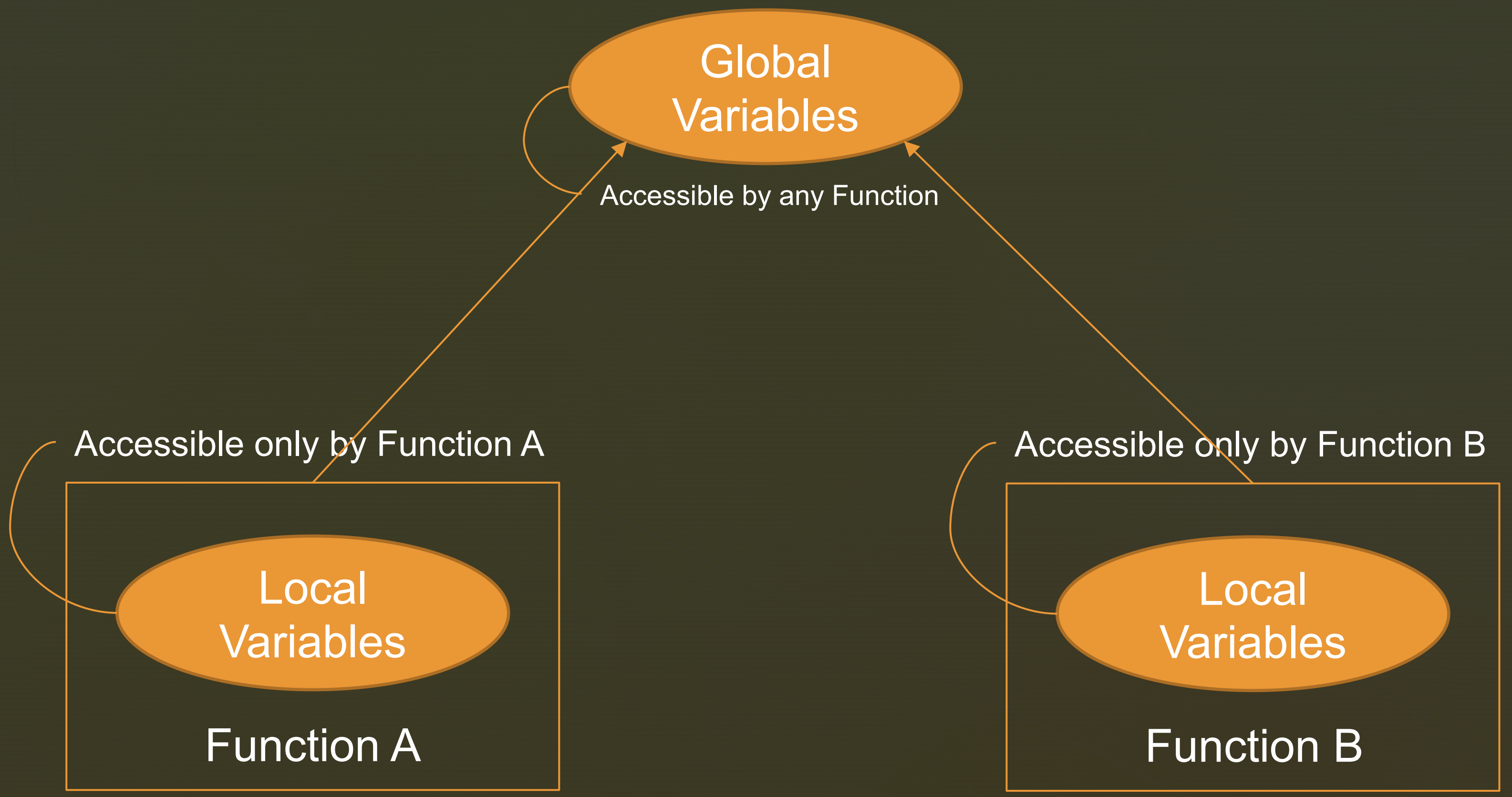
Problems with Procedural Programming

- Let’s focus on two main issues:
  - Unrestricted access to global data by functions
  - Poor representation of real-world objects

Unrestricted access to global data

- In a procedural program, one written in C for example, there are two kinds of data.
- *Local data is hidden inside a function, and is used exclusively by the function.* Local data is closely related to its function and is safe from modification by other functions.
- When two or more functions must access the same data—and this is true of the most important data in a program—then the data must be made *global.*
- Global data can be accessed by *any function in the program.*

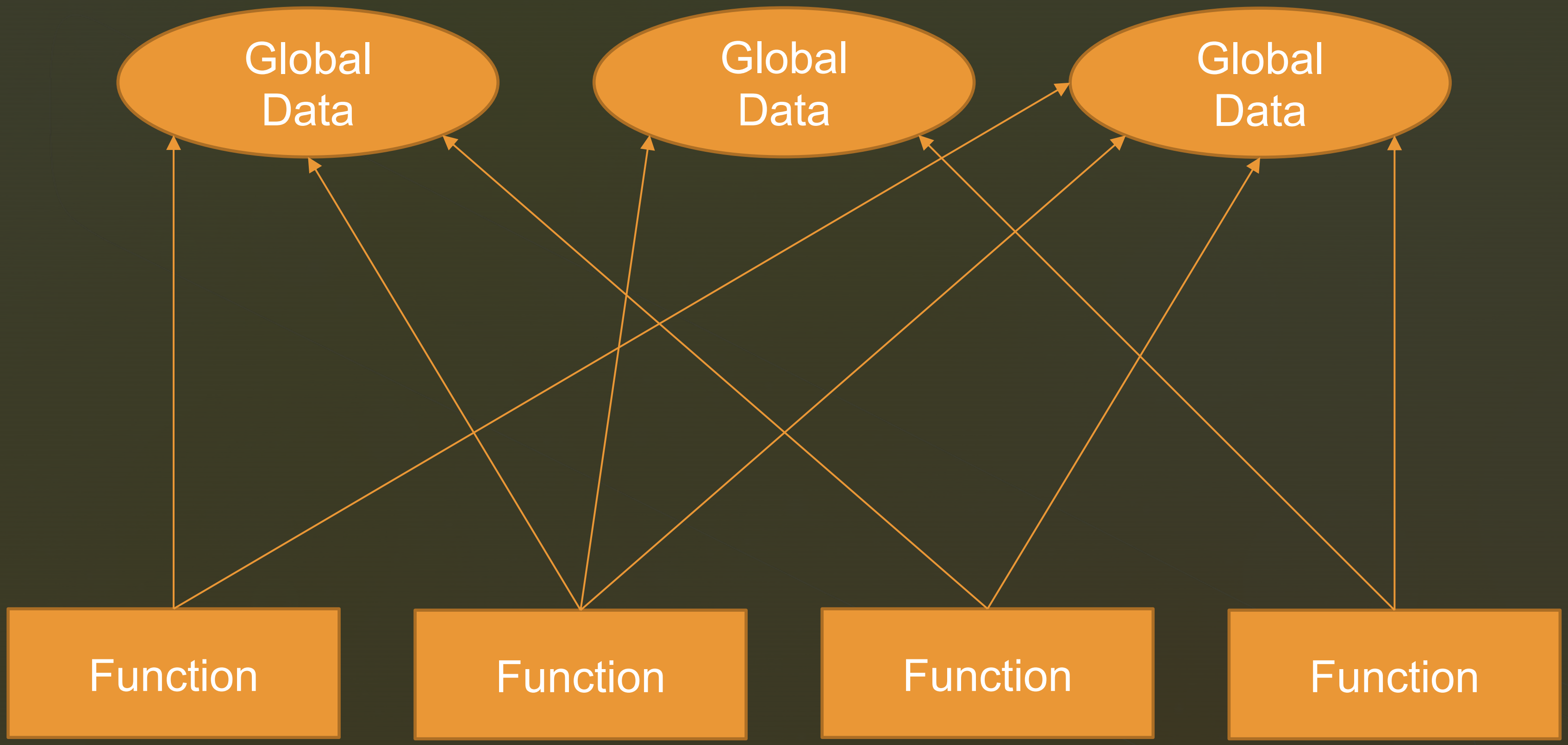
Unrestricted access to global data


Global Variables
Accessible by any Function
Accessible only by Function A
Local Variables
Function A
Accessible only by Function B
Local Variables
Function B

Unrestricted access to global data

- In a large program, there are many functions and many global data items.
- The problem with the procedural paradigm is that this leads to an even larger number of potential connections between functions and data.

Unrestricted access to global data



Unrestricted access to global data

- This large number of connections causes problems in several ways:
  - First, it makes a program's structure difficult to conceptualize.
  - Second, it makes the program difficult to modify.
    - A change made in a global data item may necessitate rewriting all the functions that access that item.

Unrestricted access to global data

- When data items are modified in a large program it may not be easy to tell which functions access the data, and even when you figure this out, modifications to the functions may cause them to work incorrectly with other global data items.
- Everything is related to everything else, so a modification anywhere has far-reaching, and often unintended, consequences.

Poor representation of real-world objects

- The second—and more important—problem with the procedural paradigm is that its arrangement of separate data and functions does a poor job of modeling things in the real world.
- In the physical world we deal with objects such as people and cars. Such objects aren’t like data and they aren’t like functions.
- Complex real-world objects have both *attributes and behavior.*

Poor representation of real-world objects

- Examples of attributes (sometimes called characteristics) for:
  - People
    - eye color and hair texture
  - Attributes in the real world are equivalent to data in a program: they have a certain specific values, such as brown(for eye color) or curly (for hair type).

Poor representation of real-world objects

- Behavior is something a real-world object does in response to some stimulus.
- Behavior is like a function: you call a function to do something and it does it.

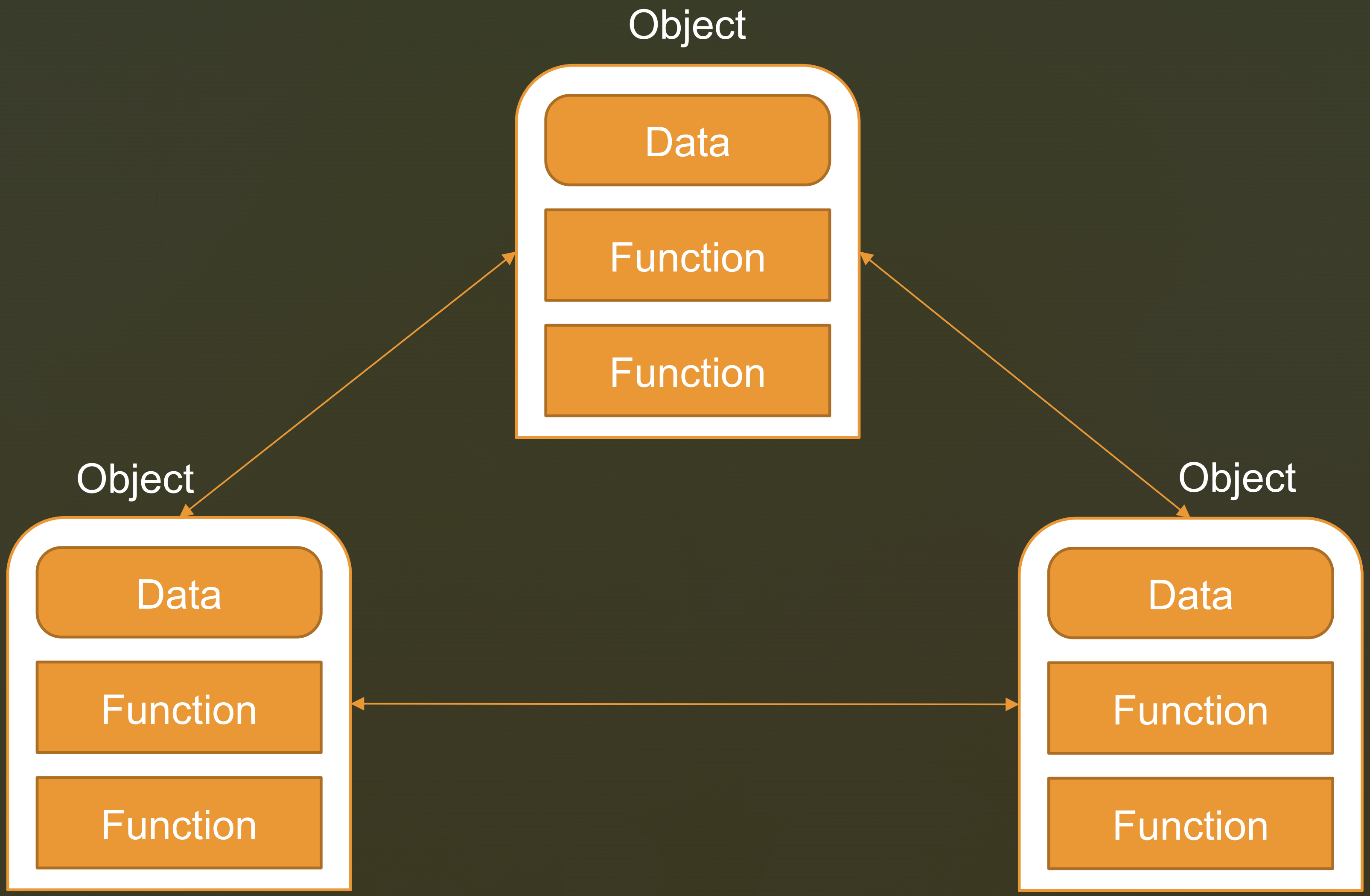
Object-Oriented World

- The fundamental idea behind object-oriented languages is to combine into a single unit both *data and the functions that operate on that data. Such a unit is called an object.*
- An object’s functions, called *member functions in C++, typically provide the only way to* access its data.
- If you want to read a data item in an object, you call a member function in the object. It will access the data and return the value to you. You can’t access the data directly.
- The data is *hidden, so it is safe from accidental alteration. Data and its functions are said to be encapsulated into a single entity.*
- *Data encapsulation and data hiding are key terms in the* description of object-oriented languages.

Object-Oriented World

- If you want to modify the data in an object, you know exactly what functions interact with it: the member functions in the object.
- No other functions can access the data. This simplifies writing, debugging, and maintaining the program.
- A C++ program typically consists of a number of objects, which communicate with each other by calling one another's member functions.
- The organization of a C++ program is shown:

Object-Oriented World



Object-Oriented World

- What are called member functions in C++ are called *methods in some* other object-oriented (OO) languages.
- Also, data items are referred to as *attributes or instance variables.*
- *Calling an object's member function* is referred to as *sending a message to the object.*

Object-Oriented World

- Keep in mind that object-oriented programming is not primarily concerned with the details of program operation.
- Instead, it deals with the overall organization of the program.
- Most individual program statements in C++ are similar to statements in procedural languages, and many are identical to statements in C.
- An entire member function in a C++ program may be very similar to a procedural function in C.
- It is only when you look at the larger context that you can determine whether a statement or a function is part of a procedural C program or an object-oriented C++ program.

Object-Oriented Programming: Objects

- When you approach a programming problem in an object-oriented language, you no longer ask how the problem will be divided into functions, but how it will be divided into objects.
- Thinking in terms of objects, rather than functions, has a surprisingly helpful effect on how easily programs can be designed.
- This results from the close match between objects in the programming sense and objects in the real world.

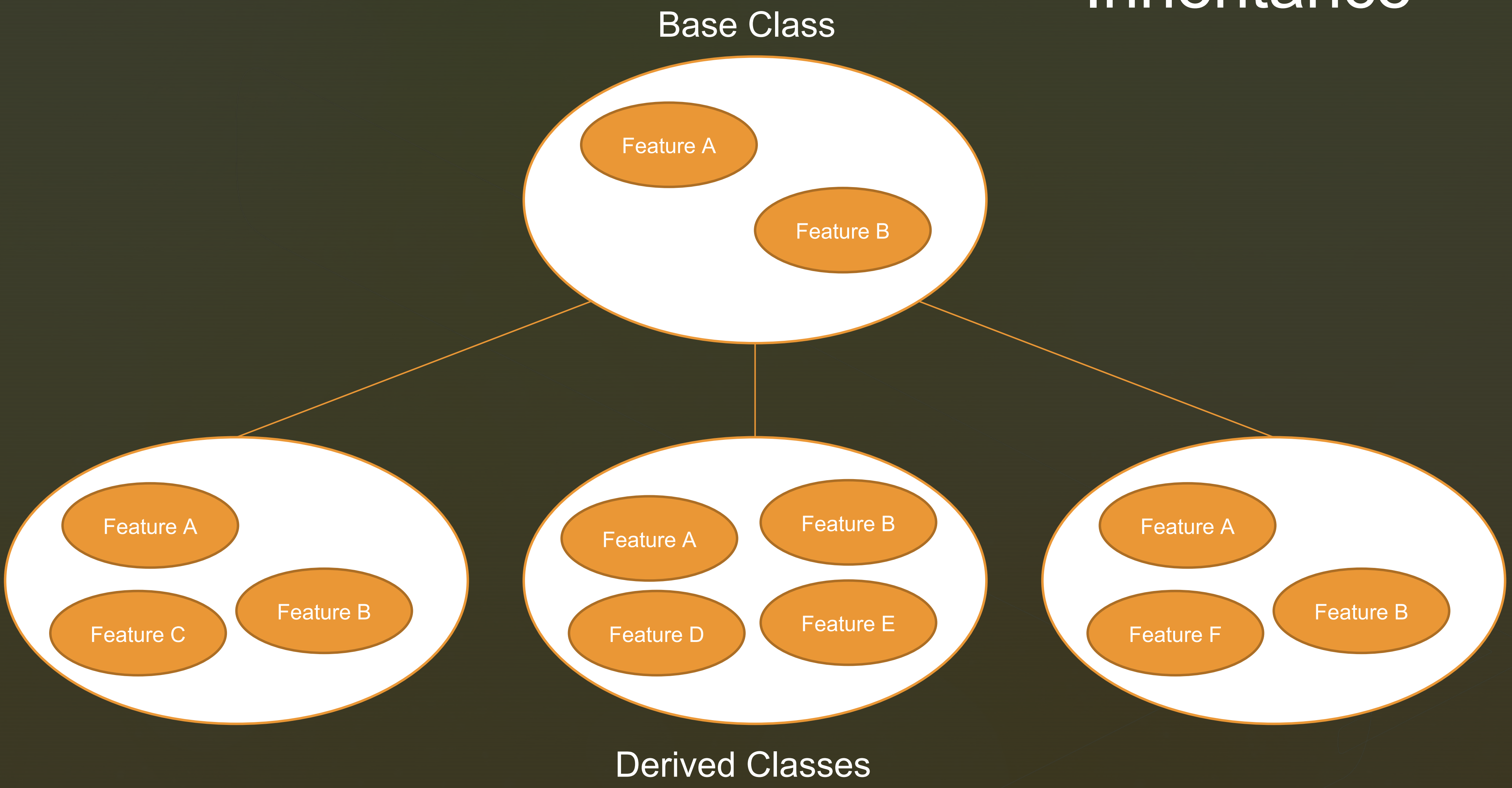
Object-Oriented Programming: Classes

- In OOP we say that objects are members of *classes.*
- A class serves as a plan, or blueprint. It specifies what data and what functions will be included in objects of that class.
- A class is thus a description of a number of similar objects.
- An object is often called an “instance” of a class.

Object-Oriented Programming: Inheritance

- The idea of classes leads to the idea of *inheritance.*
- *In our daily lives, we use the concept* of classes divided into subclasses.
- The vehicle class is divided into cars, trucks, buses, motorcycles, and so on.
- In C++ the original class is called the *base class; other classes can be defined that share its characteristics, but add* their own as well. These are called *derived classes.*

Object-Oriented Programming: Inheritance



Object-Oriented Programming: Reusability

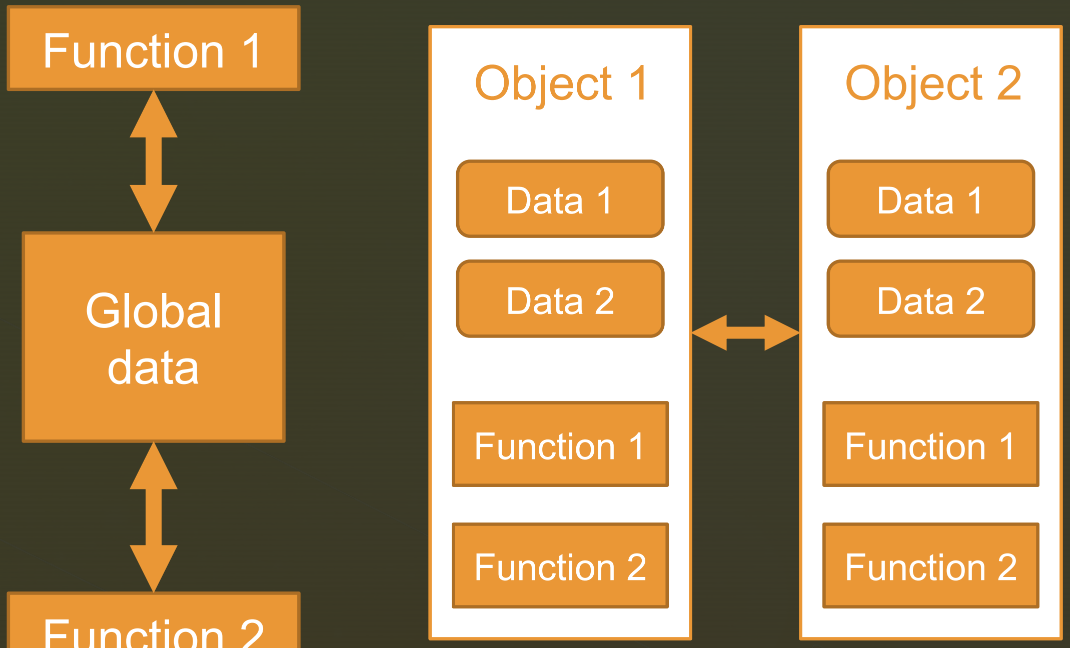
- Once a class has been written, created, and debugged, it can be distributed to other programmers for use in their own programs.
- This is called *reusability.*

Object-Oriented Programming: Polymorphism and Overloading

- Using operators or functions in different ways, depending on what they are operating on, is called polymorphism (one thing with several distinct forms).
- When an existing operator, such as + or =, is given the capability to operate on a new data type, it is said to be overloaded.
- Overloading is a kind of polymorphism; it is also an important feature of OOP.

Object-Oriented Programming: Summary

- Fundamentally, an object can be defined as an entity which contains both data and behaviour
- Objects combine data and behaviour in one entity providing a more realistic approach
- This addresses the procedural problem of uncontrolled access to global data
- Access to methods and data in an object can be controlled



Procedural Paradigm

Object-Oriented Paradigm

Procedural versus OO Programming

Procedural Paradigm	Object-Oriented Paradiagm
Program is fundamentally based on functions	Program is based on objects
Unrestricted and uncontrolled access to global data	Access to data is restricted and controlled
Focusses more on functions	Focus is on data
Data and functions that manipulate them are separated	Data and functions are combined in objects
It is difficult to represent real-world objects	Easy to represent real-world objects
Maintenance of large and complex software can be a nightmare	Maintenance of large and complex software is relatively easier

C++ and C

- C++ is derived from the C language. Strictly speaking, it is a superset of C.
- Almost every correct statement in C is also a correct statement in C++, although the reverse is not true.
- The most important elements added to C to create C++ concern classes, objects, and object-oriented programming.

Any Questions?

The End

Contact: tsadjaidoo@knust.edu.gh

Office: Caesar Building, Room 413