Procedural Programming

All the instructions must be written in order and the user has to follow. All the instructions are executed one by one. It has a complete procedure that defines what to do and how to do it. All the instructions are written in a specific order to solve a particular problem. Language is very easily understood by a user because all the instructions are written in order.

Examples of Procedural Language

1. **FORTRON(formula translation)**
2. **COBOL(common business-oriented language)**
3. **C language**
4. **basic(Beginners All-Purpose Symbolic Language)**

Advantage of Procedural language

1. It is easy to understand
2. it is easier to test and debug
3. it is a well-structured language
4. Single Programs can be written by more than one program by dividing the program up into modules.
5. It consists of a step-by-step procedure that is why it is easily understood by any user.
6. These languages are very flexible.
7.

Disadvantages of Procedural Languages

1. A big disadvantage of procedural language is the inability to reuse the code.
2. The same type of code many times throughout a program can add to the development cost and time of a project.
3. Testing the program is very difficult.
4. debugging is very difficult.
5. It is not very fastly executed as compared to the low-level language code.
6. Another disadvantage of Procedural language is that it struggled to handle the situation in which the number of possible actions may lead to the desired result

Non-Procedural Languages

In these languages, all the instructions are not written in a specific order. It is also known as a declarative and functional language. The non-procedural language that does not require writing traditional program logic. Users concentrate on defining the input and output rather than the steps of the program. Examples of non-procedural languages are as follows.

1. **Java**
2. **SQL(Structured Query Language)**
3. **RGP(report program generator**)

Advantages of Non-Procedural Languages

1. When we write large program it is very difficult to write this program in a procedural language because in a procedural language. We write it's step-by-step and it becomes very messy.
2. In non- procedural language The large program is easy to handle.
3. The nonprocedural language program is written as a different function and modulus that interact with each other.
4. These languages are easy to understand.
5. these languages are used to write large Programs easily.
6. The execution time is very fast.

Disadvantages of Non-Procedural Languages

1. It is difficult to understand.
2. it is not very flexible.
3. For the large program, the code will become very complex
4. The syntax of this language is not very easy.

Procedural Programmin

file:///C:/Users/piet/Desktop/Procedural%20Programming%20_%20Procedural%20Programming%20Paradigm%20Explained.html

Procedural Programming

Introduction To Procedural Paradigm

The procedural programming aims at dividing the large program into smaller programs called procedures. The procedures are also alternately referred to as subprograms , subroutines, methods or functions.

In procedural programming , the program code is organized as set of procedures called functions. These functions operate on the program data called the variables.

In procedural programming , the program data is in the form of variables. The functions operate on the program data. Each function consist of computational statements and solves a part of the problem.

Search the web and Windows

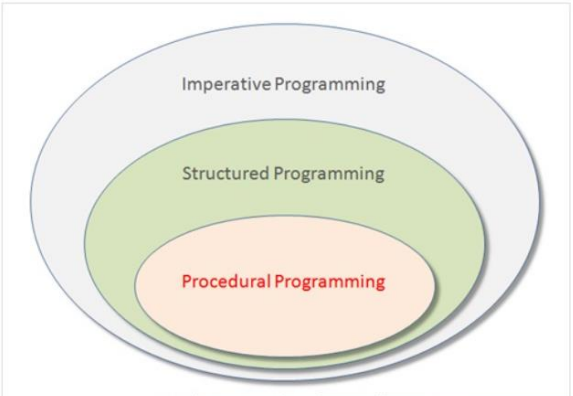
2:46 PM 1/10/2022

Procedural Programmin

file:///C:/Users/piet/Desktop/Procedural%20Programming%20_%20Procedural%20Programming%20Paradigm%20Explained.html

very popular and a part of the syllabus in most of the educational institutes and universities.

The procedural programming is a programming paradigm that is derived from structured programming. And the structured programming is derived from imperative paradigm.



Course Contents

SECTION – 1

Introduction To Computer Science

SECTION – 2

Introduction To Computer System

How Computer Works ?

SECTION – 3

CS And Binary Number System

SECTION – 4

Computer System Memory

What Is Virtual Memory ?

Random Access Memory (RAM).

SECTION – 5

Central Processing Unit (CPU)

Search the web and Windows

2:48 PM 1/10/2022

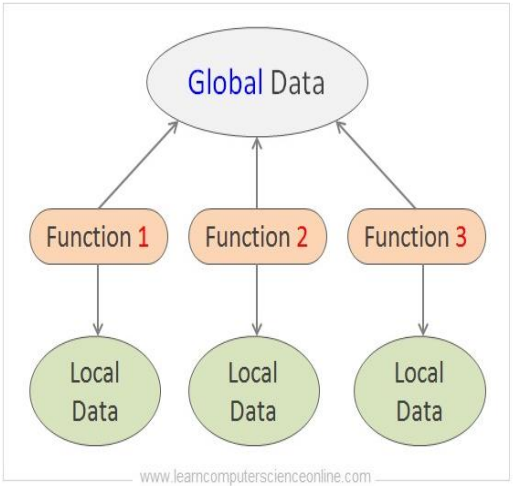
What Is Procedural Programming ?

The procedural programming paradigm is also alternately referred to as procedure oriented programming . The procedure is a key element of this paradigm.

In procedural programming , the program code is divided into group of smaller programs called functions. Each function performs a specific task depending upon the program logic.

As the name suggest, the procedural programming the program code is organized in the form of procedures. These procedures are also called as subroutines or functions.

Programming Paradigms
Program Compilation
SECTION – 11
Software Engineering
Application Software
System Software
SECTION – 12
Web Design And Development
SECTION – 14
Database Management Systems (DBMS)
Entity Relationship Diagram (ERD)
SECTION – 15
Relational Database Management System (RDBMS)



The procedural programming continues to be the preferred choice for the beginners as a first programming language. The C procedural language is extensively used in the industry for the system programming.

Despite the advent of other paradigms ,the procedural programming paradigm is still very popular and a part of the syllabus in most of the educational institutes and

Learn Computer Science
Course Contents
SECTION – 1

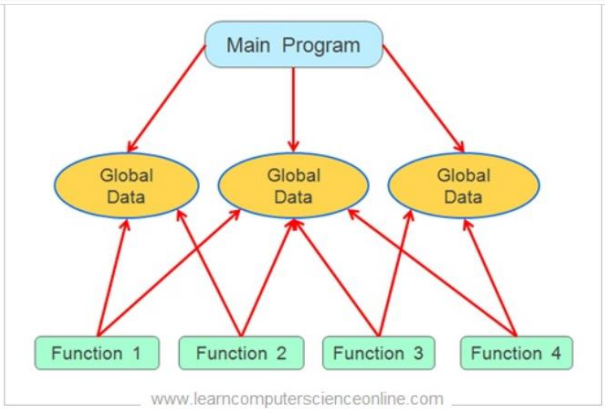
What Is Procedural Programming ?

The procedural programming paradigm is also alternately referred to as procedure oriented programming . The procedure is a key element of this paradigm.

In procedural programming , the program code is divided into group of smaller programs called functions. Each function performs a specific task depending upon the program logic.

As the name suggest, the procedural programming the program code is organized in the form of procedures. These procedures are also called as subroutines or functions.

Programming Paradigms
Program Compilation
SECTION – 11
Software Engineering
Application Software
System Software
SECTION – 12
Web Design And Development
SECTION – 14
Database Management Systems (DBMS)
Entity Relationship Diagram (ERD)
SECTION – 15
Relational Database Management System (RDBMS)



The function can access and operate upon its own local data as well as global data shared by all the functions.

In procedural programming , the function becomes the most important component of the program and the functions have unrestricted access to the global data.

SECTION – 15
Relational Database Management System (RDBMS)
How To Design Database ?
Database Normalization
Database Keys
SECTION – 16
Computer Engineering
SECTION – 17
MySQL RDBMS
SECTION – 18
Android Development
SECTION – 19
Java Programming Basics

Procedural Languages	Non-Procedural Languages
The procedural languages are command-driven or statement-oriented.	The non-procedural languages are fact-oriented.
Procedural languages are used for application and system programming.	Non-Procedural languages are used in RDBMS, expert systems, natural language processing, and education.
What to do and how to do	What to do
These are imperative programming languages.	These are declarative programming languages.
The textual context or execution sequence is considered.	There is no need to consider textual context or execution sequence.
Difficult to debug	Easy to debug
Machine efficiency is good.	The logic programs that use the only resolution face serious problems of machine efficiency.
The procedural paradigm leads to a large number of the probable network between functions and data if there are many functions and many global data items.	There are no such connections present in the non-procedural paradigm.
1. FORTRON(formula translation) 2. COBOL(common business-oriented language) 3. C language 4. basic(Beginners All-Purpose Symbolic Language)	• SQL(Structured Query Language) • RGP(report program generator) .

Procedural Languages	Non-Procedural Languages
Advantage of Procedural language •	

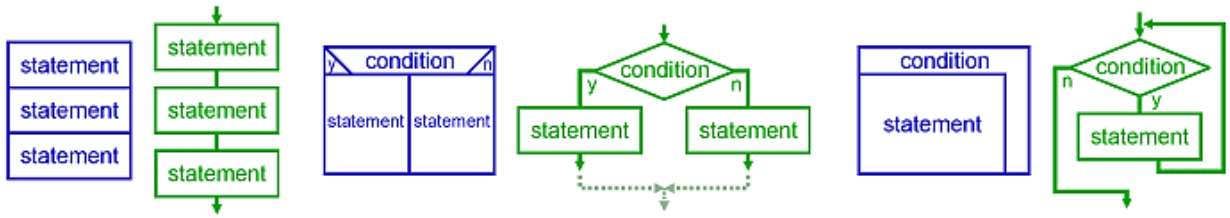
Structured Or Functional programming

Structured programming is a programming paradigm aimed at improving the clarity, quality, and development time of a computer program by making extensive use of the structured control flow constructs of selection (if/then/else) and repetition (while and for), block structures, and subroutines.

It emerged in the late 1950s with the appearance of the ALGOL 58 and ALGOL 60 programming languages with the latter including support for block structures.

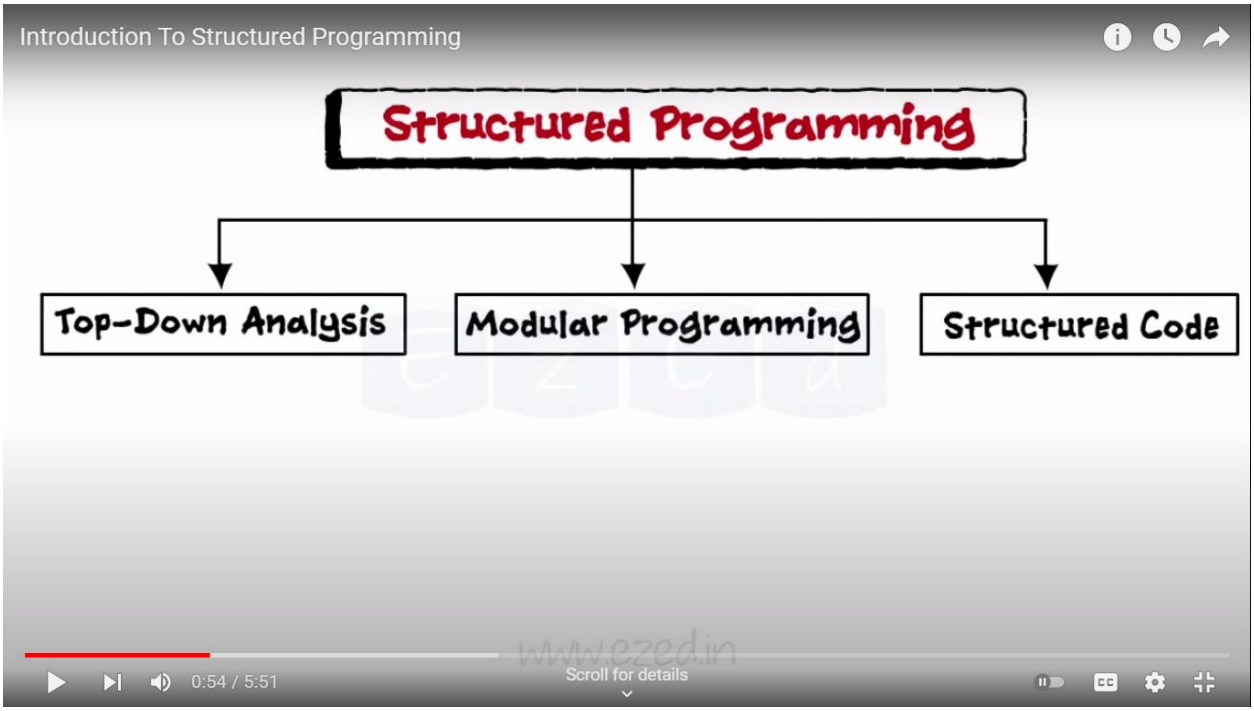
This is **a type of programming that generally converts large or complex programs into more manageable and small pieces of code**. These small pieces of codes are usually known as functions or modules or sub-programs of large complex programs.

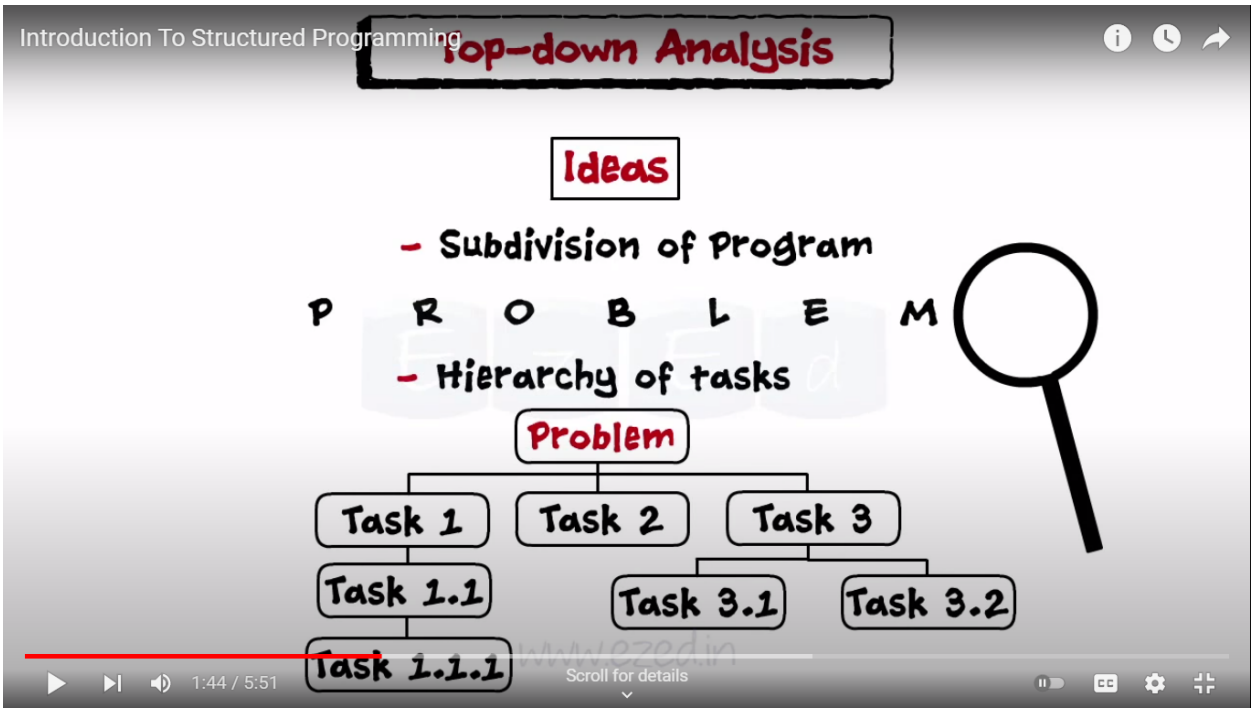
structured programming (sometimes known as *modular programming*) is a subset of procedural programming that enforces a logical structure on the program being written to make it more efficient and easier to understand and modify. Certain languages such

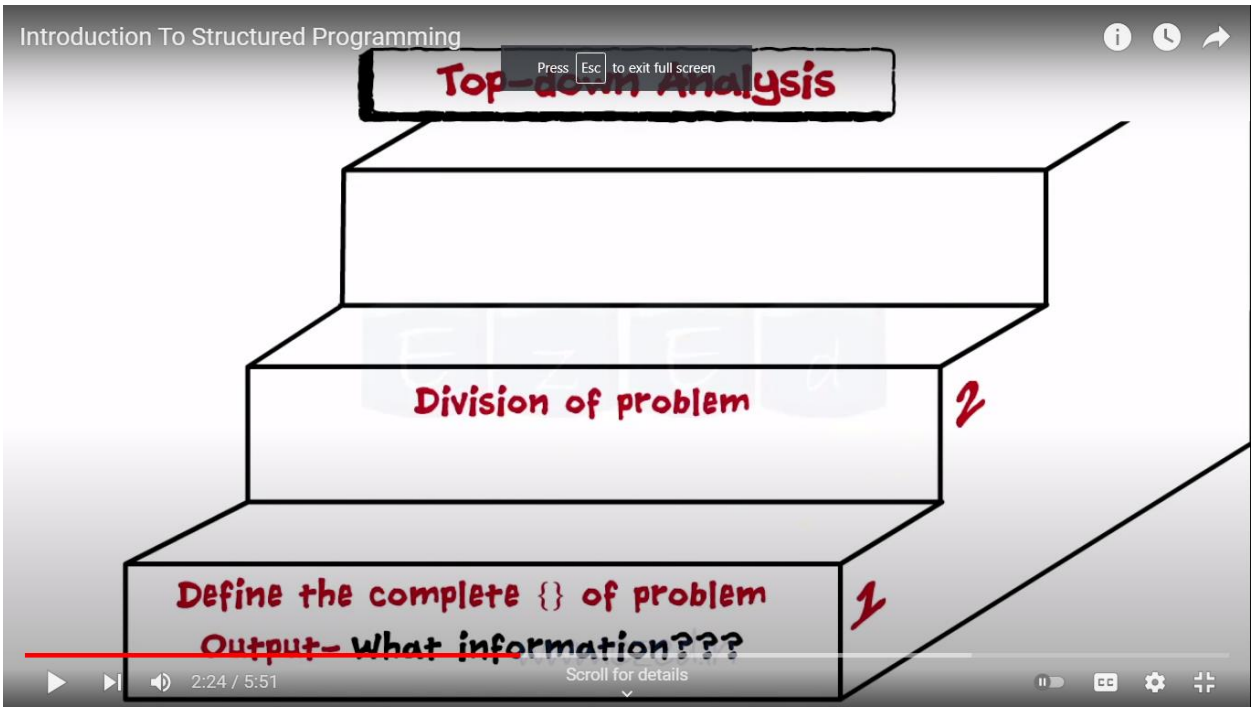


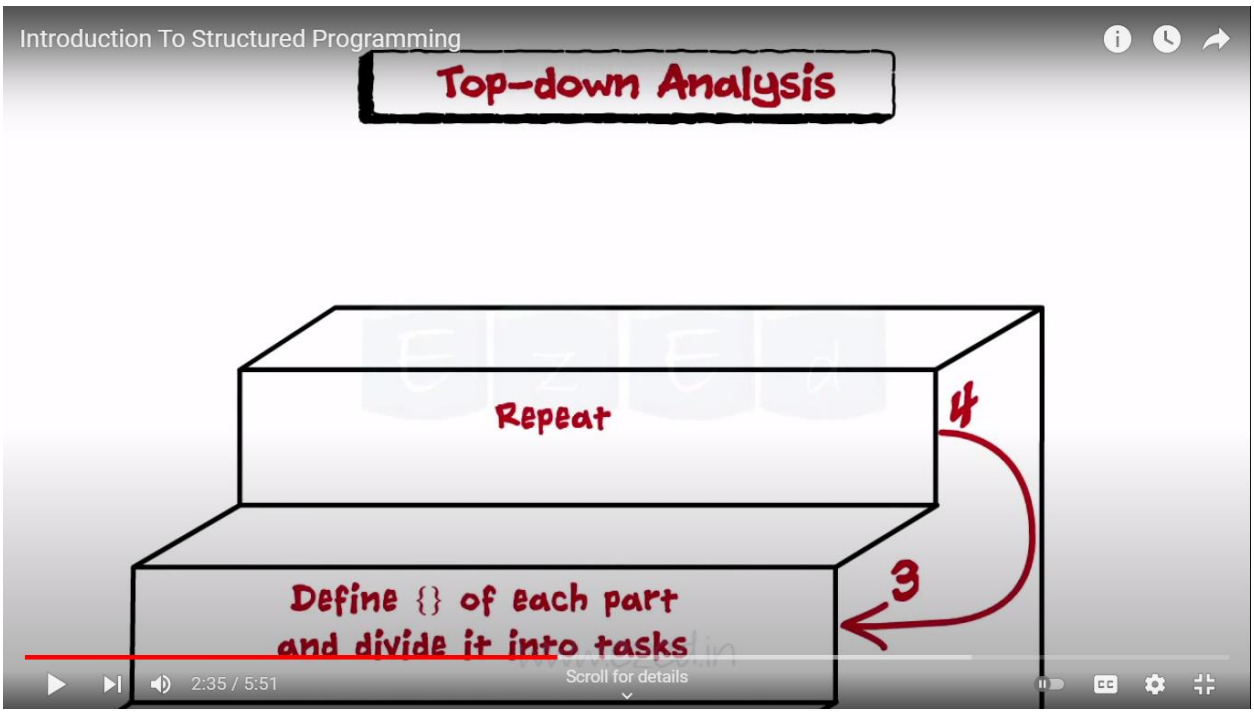
as Ada, Pascal, and dBASE are designed with features that encourage or enforce a logical program structure.

Structured programming frequently employs a top-down design model, in which developers map out the overall program structure into separate subsections.

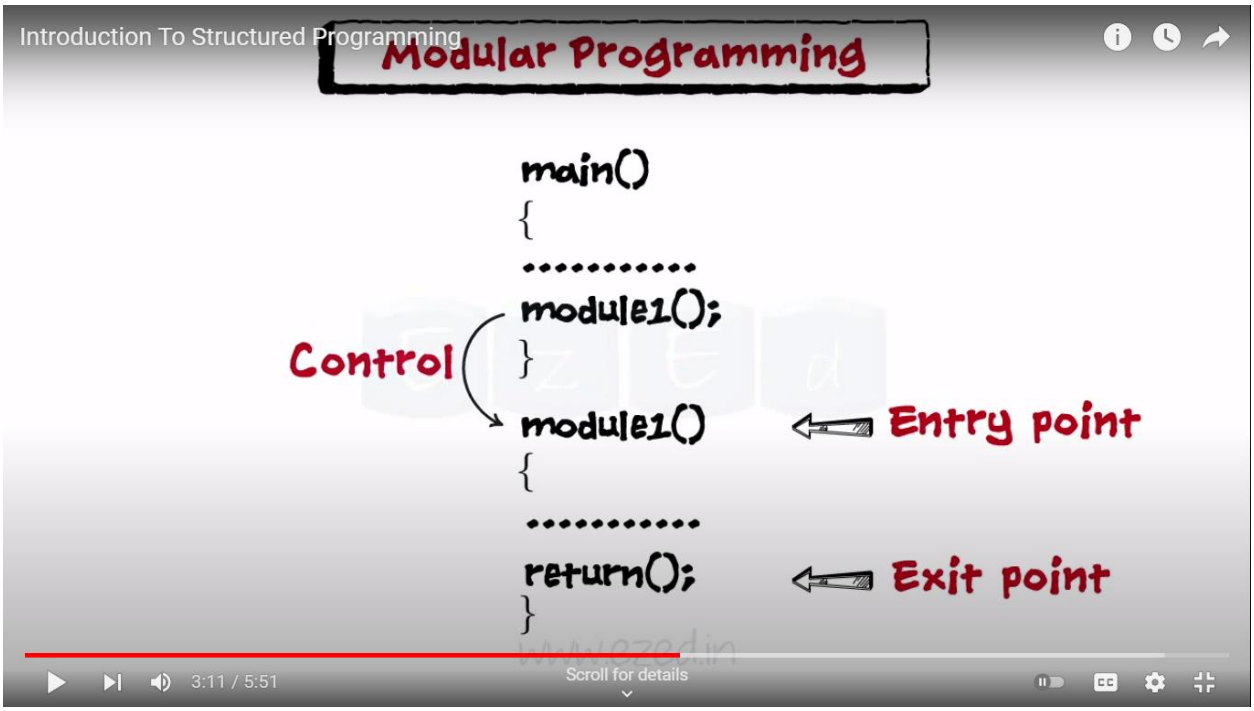

Introduction To Structured Programming
Structured Programming
Top-Down Analysis
Modular Programming
Structured Code
0:54 / 5:51
Scroll for details

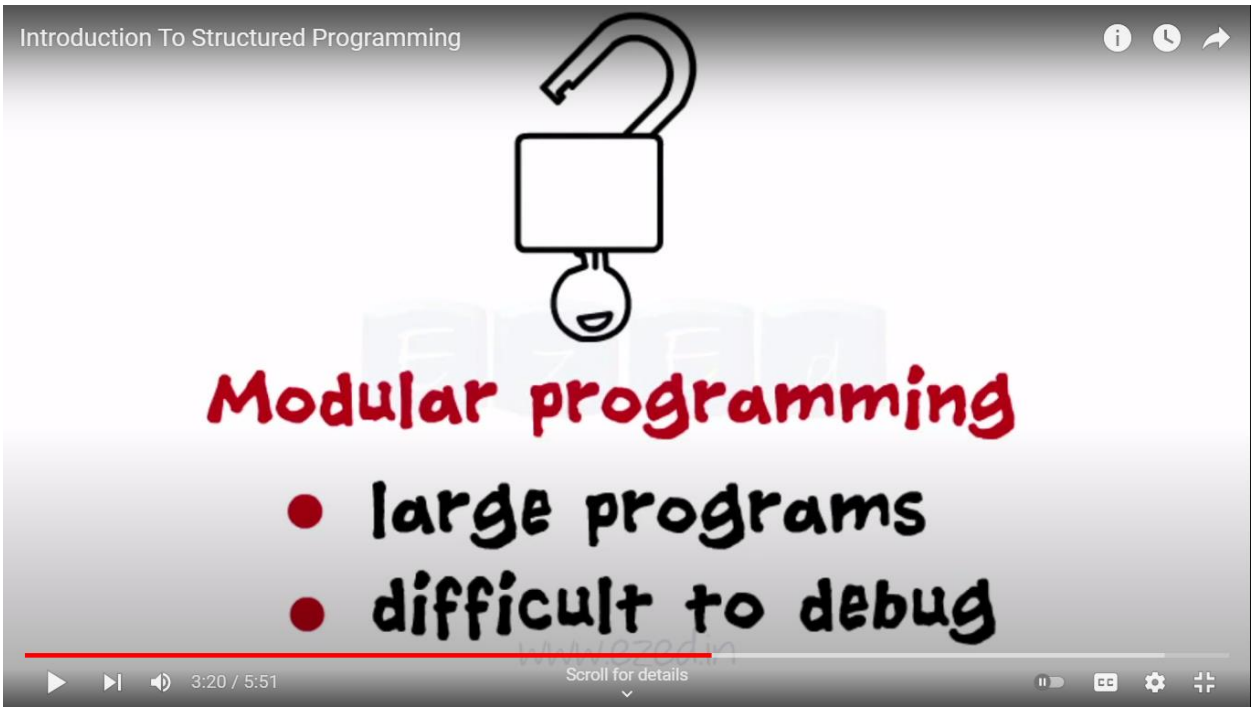

Introduction To Structured Programming
Top-down Analysis
Ideas
- Subdivision of Program
P R O B L E M
- Hierarchy of tasks
Problem
Task 1
Task 2
Task 3
Task 1.1
Task 3.1
Task 3.2
Task 1.1.1
1:44 / 5:51
Scroll for details

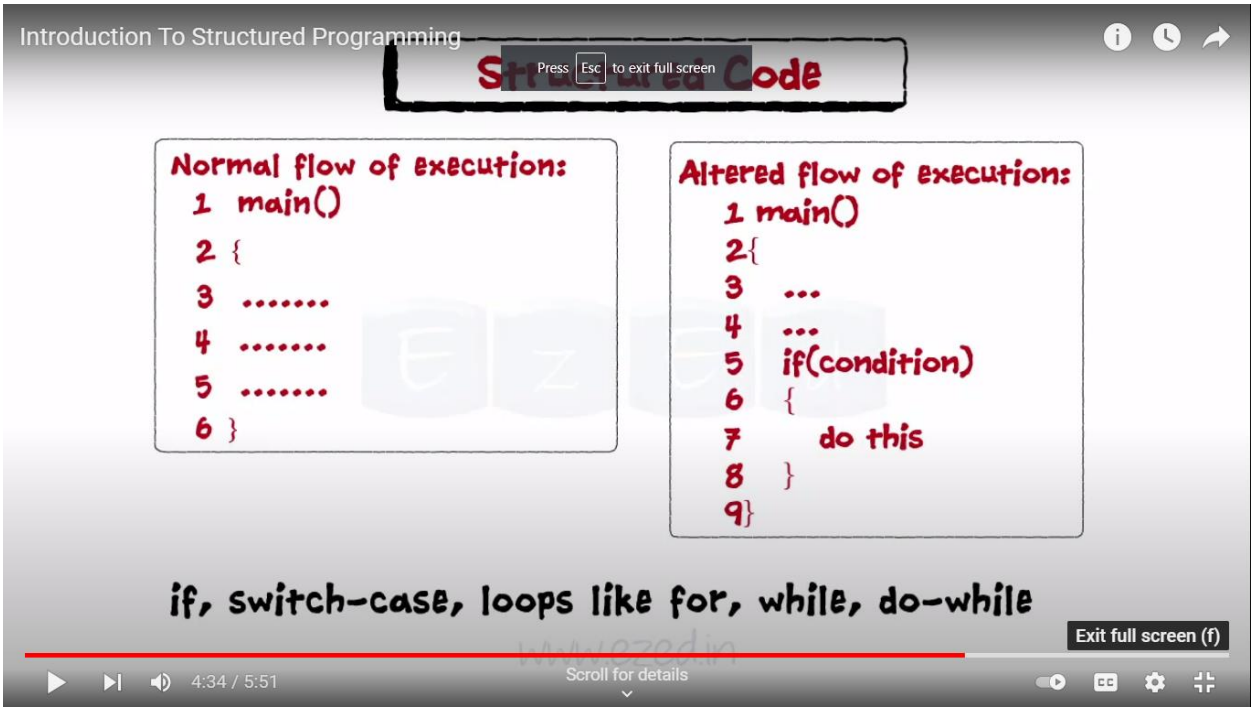




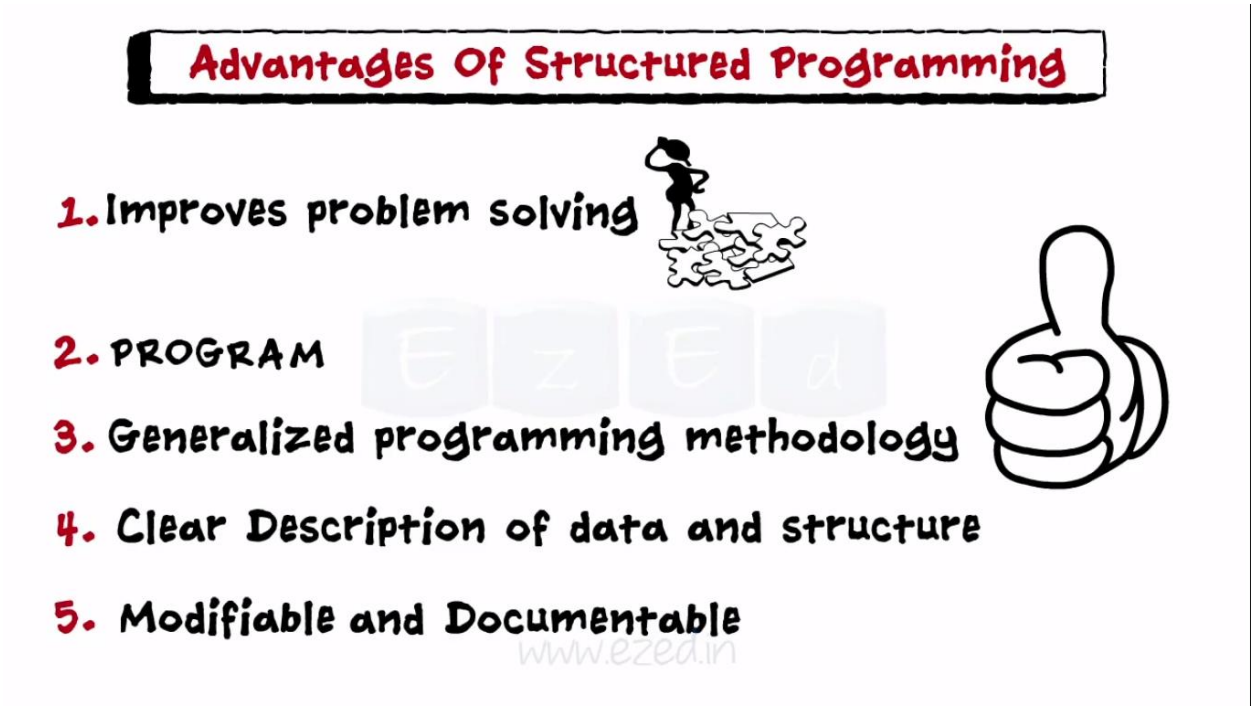
Structured programming is a programming paradigm aimed at improving the clarity, quality, and development time of a computer program by making extensive use of the structured control flow constructs of selection and repetition, block structures, and subroutines.


Introduction To Structured Programming
Modular Programming
main()
{
.....
module1();
}
Control
module1()
Entry point
{
.....
return();
Exit point
}
3:11 / 5:51
Scroll for details


Introduction To Structured Programming
Modular programming
large programs
difficult to debug
3:20 / 5:51
Scroll for details



Structured P

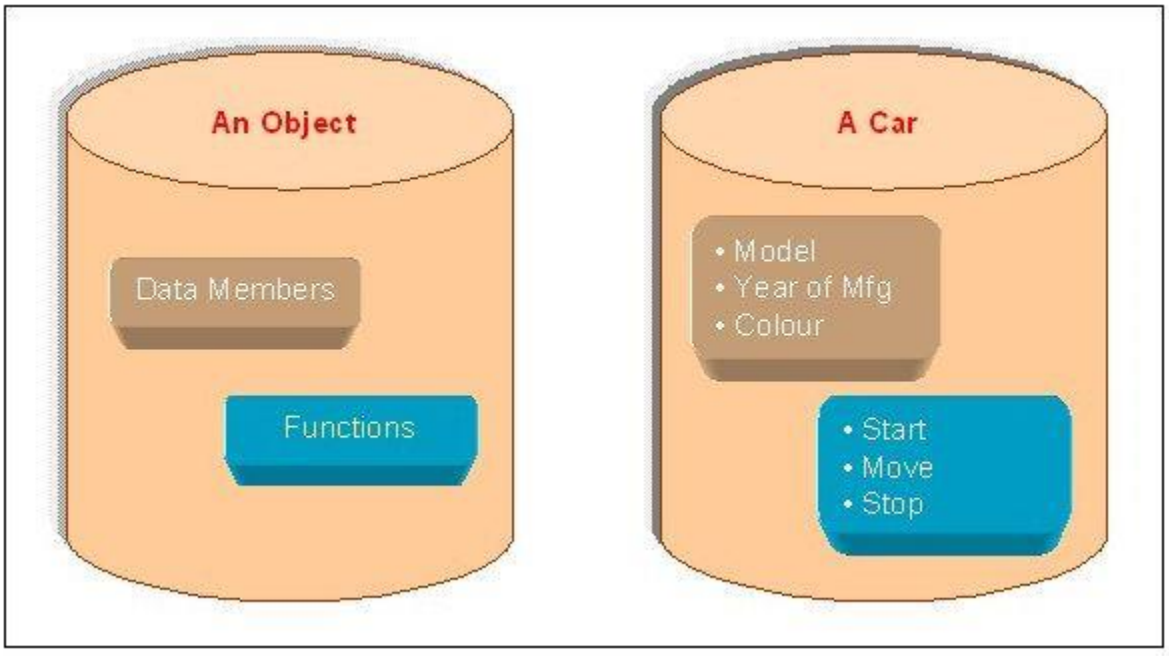


Object Oriented Programming

Object Oriented programming (OOP) is a programming paradigm that relies on the concept of classes and objects. It is used to structure a software program into classes which are used to create individual instances of objects. There are many object-oriented programming languages including JavaScript, C++, Java, and Python.

Building blocks of OOP

- Classes
- Objects
- Methods
- Attributes



Attributes

Attributes are the information that is stored. Attributes are defined in the `Class` template. When objects are instantiated individual objects contain data stored in the Attributes field.

Methods

Methods represent behaviors. Methods perform actions; methods might return information about an object, or update an object's data. The method's

OOP Features

Classes & Objects

An object is a basic unit in object-oriented programing. An object contains data and methods or functions that operate on that data. Objects take up space in memory.

A class, on the other hand, is a blueprint of the object. Conversely, an object can be defined as an instance of a class. A class contains a skeleton of the object and does not take any space in the memory.

Let us take an **Example** of a car object. A car object named "Maruti" can have data such as color; make, model, speed limit, etc. and functions like accelerate. We define another object "ford". This can have similar data and functions like that of the previous object plus some more additions.
Similarly, we can have numerous objects of different names having similar data and functions and some minor variations.

Thus instead of defining these similar data and functions in these different objects, we define a blueprint of these objects which is a class called Car. Each of the objects above will be instances of this class car.

Abstraction

Abstraction is the process of hiding irrelevant information from the user. **For Example**, when we are driving the car, first we start the engine by inserting a key. We are not aware of the process that goes on in the background for starting the engine.
Using abstraction in programming, we can hide unnecessary details from the user. By using abstraction in our application, the end user is not affected even if we change the internal implementation.

Encapsulation

Encapsulation is the process using which data and the methods or functions operating on them are bundled together. By doing this, data is not easily accessible to the outside world. In OOP we achieve encapsulation by making data members as private and having public functions to access these data members.

```
Class sample

{

Int a,b;

Public:

Void show()

{

}

}

Main()

{

Sample s;

}

s.show();
```

Inheritance

Using inheritance object of one class can inherit or acquire the properties of the object of another class. Inheritance provides reusability of code.

As such we can design a new class by acquiring the properties and functionality of another class and in this process, we need not modify the functionality of the parent class. We only add new functionality to the class.

Polymorphism

Polymorphism means many forms.

5+3=8

A+b=Ab operator overloading

Add(int,int);

Add(float,float);

Add(int,float) function overloading

Polymorphism is an important feature of OOP and is usually implemented as operator overloading or function overloading. Operator overloading is a process in which an operator behaves differently in different situations. Similarly, in function overloading, the same function behaves differently in different situations.

Dynamic Binding

OOP supports dynamic binding in which function call is resolved at runtime. This means that the code to be executed as a result of a function call is decided at runtime. Virtual functions are an example of dynamic binding.

Add(5,6.7);

Message Passing

In OOP, objects communicate with each other using messages. When objects communicate, information is passed back and forth between the objects. A message generally consists of the object name, method name and actual data that is to be sent to another object.

S1.show(3);

Advantages Of OOP

Let us discuss some of the advantages of OOP.

#1) Reusability

OOP allows the existing code to be reused through **inheritance**. We can easily acquire the existing functionality and improve on it without having to rewrite the code again. This results in less bloated code.

#2) Modularity
As we modularize the program in OOP, **it's easy to modify or troubleshoot** the program if a problem occurs or new feature or enhancement is to be added. Modularization also helps in code clarity and makes it more readable.

#3) Flexibility
OOP helps us with flexible programming using **the polymorphism feature**. As polymorphism takes many forms, we can have operators or functions that will work with many objects and thus save us from writing different functions for each object.

#4) Maintainability Maintaining code is easier as it is easy to add new classes, objects, etc without much restructuring or changes.

#5) Data and Information Hiding
OOP aids us in data hiding thereby keeping information safe from leaking. Only the data that is required for the smooth functioning of the program are exposed to the user by hiding intrinsic details.

Difference between Structured Programming and Object Oriented Programming

1. Structured Programming :
Structured Programming, as name suggests, is a technique that is considered as precursor to OOP and usually consists of well-structured and separated modules. In this programming, user can create its own user-defined functions as well as this methodology tries to resolve issues that are associated with unconditional transfers to allow programmers follow logic of programs. It also requires more discipline at the design and logical structing stage.
Example : Pascal, ALGOL, C, Modula-2, etc.

2. Object-Oriented Programming :
Object-Oriented Programming, as name suggests, is a different approach to programming that brings together data and functions that execute on them. It

basically supports encapsulation, abstraction, inheritance, polymorphism, etc. It also includes data hiding feature therefore it is more secure. This model is based on real life entities that focuses on by whom task is to be done rather than focusing on what to do.
Example : JAVA, C#, C++, etc.
Difference between Structured Programming and Object-Oriented Programming :

Structured(FUNCTION ORIENTED) Programming	Object-Oriented Programming
Structured Programming is designed which focuses on **process**	Focus on data
It is a subset of procedural programming. Also known as modular programming.	It relies on concept abstraction, polymorphism, Dynamic binding.
Programs are divided into small programs or functions.	Programs are divided into objects or entities.
Its main aim is to improve and increase quality, clarity, and development time of computer program.	Its main aim is to improve and increase both quality and productivity of system analysis and design.

It simply focuses on functions and processes that usually work on data.	It simply focuses on representing both structure and behavior of information system into tiny or small modules that generally combines data and process both.
It generally follows “Top-Down Approach”.	It generally follows “Bottom-Up Approach”.
It provides less flexibility and abstraction as compared to object-oriented programming.	It provides more flexibility and abstraction as compared to structured programming.
It gives more importance of code.	It gives more importance to data.
In Structured Programming, Programs are divided into small self contained **functions**.	In Object Oriented Programming, Programs are divided into small entities called **objects**.
Structured Programming provides **less reusability**, more function dependency.	Object Oriented Programming provides more reusability, less function **dependency**

Example : Pascal, ALGOL, C, Modula-2 **Example :** JAVA, C#, C++, etc.

Differences between C and C++ are:

C++ can be said a superset of C. Major added features in C++ are Object-Oriented Programming, Exception Handling and rich C++ Library.

Below is the table of differences between C and C++:

C	C++
C was developed by Dennis Ritchie between the year 1969 and 1973 at AT&T Bell Labs.	C++ was developed by Bjarne Stroustrup in 1979.
C does no support polymorphism, encapsulation, and inheritance which means that C does not support object oriented programming.	C++ supports polymorphism, encapsulation, and inheritance because it is an object oriented programming language.
C is a subset of C++.	C++ is a superset of C.
C contains 32 keywords.	C++ contains 63 keywords.
For the development of code, C supports procedural programming.	C++ is known as hybrid language because object oriented programming paradigms.
Data and functions are separated in C because it is a procedural programming language.	Data and functions are encapsulated together in form of an object in C++.
C does not support information hiding.	Data is hidden by the Encapsulation to ensure that data structures and operators are used as intended.

C	C++
C is a function driven language because C is a procedural programming language.	C++ is an object driven language because it is an object oriented programming.
C is a function-driven language.	C++ is an object-driven language
Functions in C are not defined inside structures.	Functions can be used inside a structure in C++.
Namespace features are not present inside the C.	Namespace is used by C++, which avoid name collisions.
Header file used by C is stdio.h.	Header file used by C++ is iostream.h.
Reference variables are not supported by C.	Reference variables are supported by C++.
Virtual and friend functions are not supported by C.	Virtual and friend functions are supported by C++.
C does not support inheritance.	C++ supports inheritance.
Instead of focusing on data, C focuses on method or process.	C++ focuses on data instead of focusing on method or procedure.
C provides malloc() and calloc() functions for dynamic memory allocation, and free() for memory de-allocation.	C++ provides new operator for memory allocation and delete operator for memory de-allocation.
Direct support for exception handling is not supported by C.	Exception handling is supported by C++.
scanf() and printf() functions are used for input/output in C.	cin and cout are used for input/output in C++.

C	C++
C structures don’t have access modifiers.	C ++ structures have access modifiers.